**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Сибирский государственный университет науки и технологий**

**имени академика М.Ф. Решетнева»**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Институт (факультет) | | | Институт инженерной экономики | |
| Направление | | 09.03.03 «Прикладная информатика» | | |
| Направленность (профиль) | | | | «Цифровые инновации в управлении |
| предприятиями» | | | | |
| Кафедра | Информационных экономических систем | | | |

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

Вид ВКР: бакалаврская работа

|  |
| --- |
| **РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ** |
| **ДЛЯ АНАЛИЗА РЫНКА** |
| **СТРОИТЕЛЬНЫХ МАТЕРИАЛОВ** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Обучающийся |  | / | Е. А. Семенов | / |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Руководитель |  | / | С. И. Сенашов | / |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ответственный за нормоконтроль |  | / | Н. Ю. Юферова | / |

Допускается к защите

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Заведующий кафедрой |  | / | М. А. Масюк | / |

«24» июня 2025 г.

Красноярск 2025

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Сибирский государственный университет науки и технологий**

**имени академика М. Ф. Решетнева»**

|  |
| --- |
| Институт инженерной экономики |
| институт |
| Кафедра информационных экономических систем |
| кафедра |

|  |  |  |  |
| --- | --- | --- | --- |
| УТВЕРЖДАЮ | | | |
| Заведующий кафедрой | | | |
|  |  | | М. А. Масюк |
|  | | «26» декабря 2024 г. | |

**ЗАДАНИЕ**

**НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ**

**в форме бакалаврской работы**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Обучающийся | | Семенов Евгений Александрович | | | | | | | | | | | |
| Группа | БПЦ21-01 | | | Направление (специальность) | | | | 09.03.03 | | | | | |
| «Прикладная информатика» | | | | | | | | | | | | | |
| Тема выпускной квалификационной работы | | | | | | | Разработка программного | | | | | | |
| обеспечения для анализа рынка строительных материалов | | | | | | | | | | | | | |
| компании | | | | | | | | | | | | | |
| Утверждена приказом по университету от | | | | | | 17.03 | | 2025 г. № | | | 744д | | |
| Руководитель ВКР | | | Сенашов С. И. д.ф.-м.н., профессор, профессор | | | | | | | | | | |
| кафедры ИЭС | | | | |  | | | | | | | | |
| Исходные данные для ВКР | | | | | Информация о деятельности строительных | | | | | | | | |
| веб-магазинов | | | | | | | | | | | | | |
| Перечень разделов ВКР | | | | | | | | | | | | | |
| 1. Анализ предметной области. | | | | | | | | | | | | | |
| 1. Проектирование системы. | | | | | | | | | | | | | |
| 1. Разработка системы. | | | | | | | | | | | | | |
| Перечень графического материала (с указанием обязательных чертежей | | | | | | | | | | | | | |
| при необходимости): | | | | презентация, раздаточный материал | | | | | | | | | |
| 1. Срок сдачи студентом первого варианта ВКР – | | | | | | | | | | «10» июня 2025 г | | | |
| 1. Срок сдачи студентом окончательного варианта ВКР | | | | | | | | | | «24» июня 2025 г | | | |
| Руководитель ВКР | | | | | | | | / | | С. И. Сенашов | | / | |
| Задание принял к исполнению | | | | | | | | / | | Е. А. Семенов | | | / |
|  | | | | | | | | | «26 декабря» 2024 г. | | | | |

**СОДЕРЖАНИЕ**

[Введение 4](#_Toc199132716)

[1 Анализ предметной области 6](#_Toc199132717)

[1.1 Анализ отрасли 6](#_Toc199132718)

[1.2 Анализ имеющихся решений 8](#_Toc199132719)

[1.3 Обоснование необходимости разработки и юридической правомерности 11](#_Toc199132720)

[2 Проектирование системы 13](#_Toc199132721)

[2.1 Требования к системе 13](#_Toc199132722)

[2.2 Функциональная модель системы 15](#_Toc199132723)

[2.3 Архитектура системы 17](#_Toc199132724)

[2.4 Моделирование данных 19](#_Toc199132725)

[2.5 Моделирование логики работы 22](#_Toc199132726)

[3 Разработка системы 30](#_Toc199132727)

[3.1 Этапы разработки 30](#_Toc199132728)

[3.2 Средства разработки 32](#_Toc199132729)

[3.3 Интерфейс пользователя 33](#_Toc199132730)

[3.4 Дизайн 46](#_Toc199132731)

[3.5 Серверная часть системы 47](#_Toc199132732)

[3.6 Описание контроллеров 50](#_Toc199132733)

[3.7 Описание моделей 53](#_Toc199132734)

[3.8 Алгоритмы получения данных 55](#_Toc199132735)

[3.9 Входные данные 56](#_Toc199132736)

[3.10 Выходные данные 57](#_Toc199132737)

[3.11 Тестирование системы 60](#_Toc199132738)

[3.12 Экономическая эффективность разработки системы 62](#_Toc199132739)

[3.13 Перспективы развития системы 64](#_Toc199132740)

[Заключение 66](#_Toc199132741)

[Список использованных источников 68](#_Toc199132742)

ВВЕДЕНИЕ

В современном мире строительная отрасль активно использует цифровые технологии для оптимизации процессов проектирования, закупок и выполнения строительных работ. Информация о строительных материалах, их свойствах, производителях и ценах является ключевым ресурсом для принятия обоснованных решений. Однако проблема заключается в том, что данные об этих материалах зачастую разрознены, хранятся в различных форматах и на множестве платформ.

Создание программных решений для автоматизации сбора и обработки таких данных становится актуальной задачей. Автоматизированный сбор данных позволяет не только сократить временные затраты, но и повысить точность анализа, обеспечивая доступ к актуальной информации о строительных материалах в режиме реального времени.

С развитием онлайн-коммерции строительные материалы становятся все более доступными через интернет-магазины и специализированные платформы. Однако поиск и систематизация данных о них требует значительных усилий, особенно если речь идет о сравнении цен и изучении характеристик. Компании и частные клиенты, принимающие решения на основе неполной или устаревшей информации, рискуют столкнуться с перерасходом бюджета или получением некачественного продукта.

Программное обеспечение, разработанное для автоматизированного сбора данных, предоставляет решение этой проблемы. Оно может осуществлять сбор информации с веб-ресурсов, таких как каталоги товаров, маркетплейсы и форумы, обрабатывать полученные данные и предоставлять пользователю структурированные результаты в удобном формате.

Кроме того, такие системы находят применение в мониторинге рынка, анализе конкурентов и оптимизации логистики. Для малых и средних предприятий внедрение подобных решений особенно важно, так как оно позволяет снизить затраты на ручную обработку информации и повышает конкурентоспособность [7].

Целью данной работы является разработка программного обеспечения для автоматизированного сбора данных о строительных материалах. Программа должна обеспечивать сбор данных (характеристики, цены) с различных веб-ресурсов и предоставлять их в удобной для анализа форме.

Для достижения данной цели необходимо решить следующие задачи:

1. Проанализировать существующие подходы и инструменты для сбора данных с веб-ресурсов, включая веб-скрейпинг и обработку данных.
2. Определить ключевые требования к программному обеспечению, включая целевые источники данных и формат их представления.
3. Разработать архитектуру программы, включающую модули сбора, обработки и хранения данных.
4. Реализовать основные алгоритмы веб-скрейпинга и обработки текстовой информации.
5. Провести тестирование программы на примере сбора данных с реальных веб-ресурсов.
6. Оценить эффективность программы и её практическую применимость.

Разработка программы для сбора данных о строительных материалах позволит автоматизировать трудоемкие процессы мониторинга и анализа информации. Компании смогут получать актуальные данные о товарах и ценах, анализировать характеристики материалов и изучать отзывы клиентов, что приведет к повышению качества принимаемых решений [9].

Кроме того, созданное программное решение может быть адаптировано для других отраслей, где требуется сбор и обработка данных из различных источников. Это делает проект универсальным инструментом для работы с информацией [11].

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Анализ отрасли

Строительная отрасль играет ключевую роль в экономике, формируя инфраструктуру, жильё, промышленные объекты и общественные сооружения. Это одна из самых динамично развивающихся сфер, которая тесно связана с рядом смежных отраслей, таких как производство строительных материалов, транспорт, логистика и архитектурное проектирование.

Особенности строительной отрасли:

1. Высокий уровень конкуренции среди компаний, предлагающих строительные материалы и услуги.
2. Зависимость от сезонности: пик активности приходится на тёплые месяцы года.
3. Постоянная потребность в оптимизации затрат на материалы, труд и логистику.
4. Широкий спектр используемых строительных материалов, который варьируется от стандартных (бетон, кирпич) до инновационных (композитные материалы, эко-решения).

Строительные материалы являются основой любой стройки, определяя качество и долговечность объектов. Рынок этих материалов характеризуется большим разнообразием и включает продукцию различных производителей: от крупных международных брендов до локальных компаний.

Основные категории строительных материалов:

1. Основные строительные материалы:
   1. Бетон, цемент, песок, гравий.
   2. Кирпич (керамический, силикатный).
   3. Металлоконструкции.
2. Отделочные материалы:
   1. Штукатурки, шпаклёвки, краски.
   2. Плитка (керамическая, мраморная), ламинат, паркет.
   3. Обои, декоративные панели.
3. Изоляционные материалы:
   1. Теплоизоляция (минеральная вата, пенопласт).
   2. Звукоизоляция.
   3. Гидроизоляционные плёнки и мембраны.
4. Инженерные материалы:
   1. Трубы и фитинги.
   2. Электрические кабели, системы водоснабжения и вентиляции.
5. Инновационные и экологичные материалы:
   1. Эко-материалы (солома, переработанный пластик, бамбук).
   2. Самовосстанавливающийся бетон, «умные» стекла.

Факторы, определяющие выбор строительных материалов:

1. Качество и долговечность.
2. Стоимость. Это особенно важно для крупных проектов с жёстким бюджетом.
3. Эстетика и соответствие современным стандартам.
4. Экологичность и энергоэффективность.

Современный строительный бизнес ориентирован на внедрение новых технологий и материалов, что позволяет сокращать затраты и повышать производительность.

Основные тенденции:

1. Инновационные материалы.
   1. Использование композитных материалов с высокой прочностью и низким весом.
   2. Внедрение 3*D*-печати для создания строительных конструкций.
   3. Применение «зелёных» материалов, способствующих снижению углеродного следа.
2. Цифровизация отрасли.
   1. *BIM*-технологии (*Building* *Information* *Modeling*) для проектирования и управления строительными объектами.
   2. Использование программного обеспечения для оценки смет и управления поставками строительных материалов.
3. Автоматизация процессов.
   1. Использование роботов и дронов для выполнения строительных задач и мониторинга.
   2. Внедрение умных систем на объектах строительства (контроль за качеством материалов, автоматическое управление климатом).
4. Устойчивое строительство.
   1. Переход на возобновляемые источники энергии в строительных процессах.
   2. Разработка зданий с нулевым потреблением энергии (*zero*-*energy* *buildings*).

Проблемы и вызовы строительной отрасли:

1. Колебания цен на строительные материалы. Рынок материалов подвержен влиянию внешних факторов, таких как экономическая нестабильность, инфляция, логистические проблемы.
2. Качество материалов. Риск покупки несертифицированной или контрафактной продукции, что может негативно сказаться на долговечности строительства.
3. Дефицит инновационных материалов. В некоторых регионах существует ограниченный доступ к современным материалам, что замедляет внедрение инноваций.
4. Управление информацией. Из-за обилия поставщиков и материалов застройщики часто сталкиваются с трудностями в анализе и сравнении предложений.
5. Экологические стандарты. Ужесточение экологических требований вынуждает компании искать более экологичные материалы и технологии [10].

Важность автоматизации сбора данных:

Автоматизация процессов, связанных с выбором и покупкой строительных материалов, становится ключевым фактором успеха. Компании и индивидуальные застройщики нуждаются в инструменте, который:

1. Позволяет быстро находить и анализировать предложения поставщиков.
2. Обеспечивает доступ к актуальной информации о ценах, характеристиках и доступности материалов.
3. Снижает риск человеческой ошибки при работе с большими объёмами данных.

Создание программы для автоматизированного сбора данных даст следующие преимущества:

1. Снижение издержек. Экономия времени и ресурсов за счёт автоматизации анализа рынка строительных материалов.
2. Повышение конкурентоспособности. Быстрая реакция на изменения цен и условий поставки.
3. Улучшение качества строительства. Использование только проверенных и подходящих материалов.

Таким образом, строительный бизнес нуждается в эффективных цифровых решениях для обработки данных, что делает тему дипломной работы актуальной и практически значимой [10].

1.2 Анализ имеющихся решений

В современном строительном бизнесе широко используются цифровые инструменты для поиска, анализа и закупки строительных материалов. Эти решения позволяют автоматизировать рутинные процессы и принимать более обоснованные решения, основываясь на актуальных данных [19]. На рынке представлено несколько типов программного обеспечения, которые могут быть разделены на три основные категории:

1. Каталоги строительных материалов (онлайн-платформы).
2. Программы для управления строительными проектами (строительный *ERP*).
3. Универсальные парсеры данных и платформы сбора информации.

Каталоги строительных материалов. Это специализированные онлайн-платформы, предоставляющие информацию о продукции, ценах и наличии. Основная цель таких решений – упростить процесс поиска материалов для строителей и проектировщиков [9].

Примеры решений:

1. «Стройплощадка.ру», «Пульс Цен», «Маркет строительных материалов».

Основной функционал: поиск материалов по категориям, сравнение цен от разных поставщиков, контакты для заказа напрямую у производителей.

Преимущества: простота использования, доступность информации о продуктах, возможность сравнения поставщиков.

Ограничения: нет автоматического сбора информации с сайтов поставщиков, часто требуется ручной поиск и выбор.

1. «*AliExpress* для стройматериалов» (международные площадки).

Преимущества: широкий выбор продукции, включая инновационные материалы.

Ограничения: сложность работы с большими объёмами данных, отсутствие локальных поставщиков.

*ERP*-системы интегрируют управление строительными процессами, включая закупки материалов, расчёты смет и контроль затрат.

Примеры решений:

1. 1С: Управление строительной организацией.

Возможности: управление проектами, автоматизация смет, интеграция с базами данных, отслеживание поставок строительных материалов.

Преимущества: высокая надёжность, глубокая интеграция с бухгалтерскими и складскими процессами.

Недостатки: сложность настройки, не подходит для сбора данных с внешних источников (нужен ручной ввод или интеграция с другими программами).

1. *PlanRadar*, *Procore*.

Это системы управления строительными проектами, включая закупки материалов.

Ограничения: акцент на управлении проектами, а не на сборе данных с внешних источников.

Универсальные парсеры данных и платформы. Для автоматизации сбора информации многие компании используют универсальные инструменты веб-скрейпинга, которые можно адаптировать под сбор данных о строительных материалах.

Примеры решений:

1. *BeautifulSoup*, *Scrapy* (*Python*-библиотеки).

Используются для веб-скрейпинга *HTML*-страниц и извлечения неструктурированных данных.

Преимущества: гибкость и возможность адаптации, подходит для сбора данных с любых сайтов.

Ограничения: требуется высокая квалификация для настройки и поддержки, отсутствие встроенного анализа данных.

1. *Octoparse*, *ParseHub*. Платформы с графическим интерфейсом для веб-скрейпинга данных.

Преимущества: простота настройки для пользователей без навыков программирования. визуализация результатов.

Недостатки: ограниченные возможности обработки больших массивов данных. платная подписка.

Результаты сравнения указаны в таблице 1.

Таблица 1 – Результаты сравнения аналогов

|  |  |  |
| --- | --- | --- |
| Категория решений | Преимущества | Недостатки |
| Каталоги строительных материалов | Удобный интерфейс, информация о продуктах доступна сразу. | Ограничение данными только определённых поставщиков. |
| Строительный *ERP* | Интеграция всех процессов в компании, точность данных. | Сложность настройки, нет сбора данных с внешних сайтов. |
| Универсальные парсеры | Гибкость, возможность автоматического сбора больших объёмов данных. | Требуют навыков программирования или платных подписок. |

Каталоги строительных материалов - удобный интерфейс, информация о продуктах доступна сразу. Ограничение данными только определённых поставщиков.

Строительный *ERP* - интеграция всех процессов в компании, точность данных. Сложность настройки, нет сбора данных с внешних сайтов.

Универсальные парсеры - гибкость, возможность автоматического сбора больших объёмов данных. Требуют навыков программирования или платных подписок [8].

Проблемы существующих решений [7].

1. Ограниченность функционала каталогов. Каталоги предоставляют только уже готовую информацию, отсутствует гибкость в сборе данных из разных источников.
2. Сложность настройки универсальных парсеров. Самостоятельная настройка веб-скрейпинга требует знаний в программировании и создания сложных алгоритмов.
3. Отсутствие интеграции между системами. Например, *ERP*-системы не позволяют автоматически анализировать актуальные данные с сайтов поставщиков.
4. Неполная информация. В каталогах и *ERP* часто отсутствуют данные об уникальных или локальных поставщиках, что ограничивает возможности пользователей.

Потенциал разрабатываемого решения. На основании анализа можно выделить ключевые особенности, которые должны присутствовать в эффективной системе сбора данных о строительных материалах:

1. Автоматизация сбора информации. Интеграция данных из *HTML*-страниц (неструктурированные данные).
2. Фильтрация и структурирование информации. Преобразование неструктурированных данных в удобный для анализа формат.
3. Удобство использования. Простота настройки и возможность быстрого анализа данных.
4. Гибкость. Возможность адаптации под потребности различных пользователей: строительных компаний, проектировщиков, частных застройщиков.

Предлагаемое решение будет эффективно сочетать возможности универсальных парсеров с пользовательской простотой, характерной для каталогов, и интеграционным потенциалом *ERP*. Это создаст новый уровень эффективности и конкурентоспособности для участников строительного рынка [8].

1.3 Обоснование необходимости разработки и юридической правомерности

В условиях современного строительного рынка, характеризующегося высокой волатильностью цен и постоянным обновлением ассортимента материалов, проблема оперативного получения достоверной информации приобретает особую актуальность. Традиционные методы сбора данных через ручной мониторинг сайтов поставщиков или работу с бумажными каталогами демонстрируют свою неэффективность, приводя к существенным временным и финансовым издержкам [8].

* + - 1. Основная проблема заключается в том, что при ручном сборе информации.
      2. Существенно возрастает вероятность ошибок ввода данных.
      3. Теряется оперативность получения информации.
      4. Отсутствует возможность комплексного анализа большого массива данных.
      5. Затруднено сравнение предложений различных поставщиков.

Разрабатываемое программное решение призвано устранить эти недостатки путем автоматизации процесса сбора и обработки информации. Его внедрение позволит строительным компаниям существенно сократить временные затраты на анализ рынка, повысить точность принимаемых решений и, как следствие, оптимизировать бюджет строительных проектов. Особенно важным представляется возможность оперативного реагирования на изменения рыночной конъюнктуры, что в современных экономических условиях является критически важным фактором конкурентоспособности.

С юридической точки зрения разработка подобных систем требует особого внимания к вопросам правомерности сбора и использования данных. В данном проекте предусмотрены следующие меры для обеспечения соответствия законодательным требованиям [6]:

Во-первых, система будет работать только с общедоступной информацией, не требующей авторизации или специальных разрешений для доступа. Это исключает нарушение положений о защите персональных данных и коммерческой тайне. При разработке алгоритмов сбора данных особое внимание уделяется соблюдению условий пользовательских соглашений веб-ресурсов.

Во-вторых, частота запросов к источникам информации будет ограничена таким образом, чтобы не создавать избыточной нагрузки на серверы поставщиков. Это соответствует принципам добросовестного использования интернет-ресурсов и исключает возможность квалификации таких действий как несанкционированный доступ.

В-третьих, в алгоритмы обработки данных закладываются механизмы фильтрации и исключения любой информации, которая может быть отнесена к персональным данным или коммерческой тайне. Это обеспечивает соответствие требованиям Федерального закона «О персональных данных» и аналогичных международных нормативных актов.

Важно отметить, что разрабатываемая система не предусматривает хранение или обработку данных, защищенных авторским правом. Все собираемые сведения относятся исключительно к характеристикам товаров и их стоимости, что соответствует практике добросовестной конкуренции.

Таким образом, предлагаемое решение не только отвечает насущным потребностям строительной отрасли в автоматизации процессов сбора и анализа информации, но и полностью соответствует действующим правовым нормам, что делает возможным его практическое применение в реальных бизнес-процессах строительных компаний.

Анализ строительной отрасли показал, что рынок строительных материалов является одним из ключевых и наиболее динамично развивающихся сегментов экономики, характеризующимся высокой конкуренцией, разнообразием продукции и зависимостью от сезонных факторов. Современные тенденции, такие как цифровизация, автоматизация и внедрение инновационных материалов, диктуют необходимость применения новых *IT*-решений для повышения эффективности бизнес-процессов.

Проведённый обзор существующих решений выявил ряд серьёзных ограничений. Каталоги строительных материалов удобны, но предоставляют ограниченную информацию и не поддерживают автоматический сбор данных. *ERP*-системы хорошо интегрируются с внутренними процессами компании, однако не обладают возможностями для получения данных с внешних сайтов. Универсальные парсеры обладают высокой гибкостью, но требуют специальных знаний или финансовых затрат.

Таким образом, на рынке отсутствует универсальное и в то же время доступное решение, которое позволяло бы автоматически собирать, структурировать и анализировать информацию о строительных материалах с разных источников. Это подтверждает актуальность разработки собственной информационной системы, ориентированной на автоматизацию сбора и анализа данных о строительных материалах. Подобный подход позволит значительно повысить прозрачность выбора поставщиков, снизить затраты и обеспечить более обоснованное принятие решений в строительной сфере.

2 ПРОЕКТИРОВАНИЕ СИСТЕМЫ

2.1 Требования к системе

Функциональные требования.

Функциональные требования определяют, что система должна делать, то есть её основные возможности и поведение с точки зрения пользователя и разработчика [19].

Веб-скрейпинг данных с внешних сайтов.

Система должна автоматически собирать информацию о товарах (название, цена, описание, единица измерения и т.д.) со сторонних веб-ресурсов, специализирующихся на продаже строительных материалов. Источники должны быть настраиваемыми, а сам процесс веб-скрейпинга — управляемым.

Сохранение данных в базу данных.

Все полученные данные после веб-скрейпинга должны сохраняться в централизованную реляционную базу данных с возможностью последующей обработки и визуализации. Должна обеспечиваться структурированная схема хранения информации, поддерживающая связи между товарами, категориями и источниками.

Категоризация товаров.

Система должна классифицировать строительные материалы по категориям (например, цемент, кирпич, утеплители, пиломатериалы и т.д.). Классификация может быть выполнена автоматически (по ключевым словам) либо вручную через интерфейс администратора.

Просмотр данных через веб-интерфейс.

Пользователи должны иметь доступ к информации о товарах через интуитивно понятный веб-интерфейс. Интерфейс должен предоставлять базовую информацию о товаре, цену, дату последнего обновления, а также ссылку на оригинальный источник.

Поиск и фильтрация.

Интерфейс должен поддерживать возможность поиска товаров, по ключевым словам, фильтрации по категориям, диапазону цен, датам изменения и другим параметрам.

Загрузка данных.

Пользователи должны иметь возможность выгружать данные в форматах *CSV* и *JSON* для дальнейшего анализа или использования в сторонних системах.

Построение графика динамики цен.

Для каждого товара должна быть возможность построения диаграммы изменения цены во времени за заданный период. Это позволит анализировать рыночные тенденции и сравнивать цены от разных поставщиков.

Нефункциональные требования.

Нефункциональные требования описывают качество работы системы, её надёжность, производительность, удобство использования и другие технические аспекты [19].

Надёжность и сохранность данных.

Система должна обеспечивать устойчивое хранение и защиту информации. Все данные должны сохраняться в отказоустойчивой базе. В случае сбоев не должно происходить потери данных.

Регистрация и аутентификация пользователей.

Должна быть реализована система регистрации и входа с поддержкой безопасного хранения паролей, хеширование с солью.

Многопоточность и параллельная обработка.

Система должна использовать многопоточность или асинхронную обработку при веб-скрейпинге, чтобы одновременно обрабатывать данные с нескольких сайтов без снижения производительности.

Отказоустойчивость.

В случае ошибок при подключении к источникам данных, сбоев в работе парсера или базы данных, система должна корректно логировать ошибки, сохранять текущее состояние и перезапускать процессы при необходимости.

Производительность.

Интерфейс пользователя должен работать без заметных задержек, веб-скрейпинг – происходить в разумные сроки, а построение графиков – выполняться мгновенно при наличии кэширования.

Масштабируемость.

Архитектура системы должна предусматривать возможность расширения (например, добавление новых сайтов для веб-скрейпинга, интеграция дополнительных аналитических модулей) без необходимости кардинальной переработки кода.

Безопасность.

Система должна быть защищена от типовых веб-уязвимостей (*SQL*-инъекции, *XSS*, *CSRF*). Также следует ограничить количество попыток входа для предотвращения атак перебором.

Интерфейс пользователя (*UI*/*UX*).

Веб-интерфейс должен быть удобным, адаптивным под разные устройства (десктоп, планшет, смартфон).

Кроссбраузерность.

Система должна корректно работать в популярных браузерах (*Google* *Chrome*, *Mozilla* *Firefox*, *Microsoft* *Edge*, *Safari*).

Логирование и мониторинг.

Все ключевые действия системы (например, успешный веб-скрейпинг, ошибки, входы в систему, скачивание файлов) должны логироваться. Это поможет в отладке, аудите и выявлении проблем.

Локализация и язык интерфейса.

Интерфейс системы должен быть реализован минимум на русском языке с возможностью добавления других языков в будущем.

Обновляемость.

Код и компоненты системы должны быть разработаны с учётом возможности обновления — добавление новых функций, исправление багов и обновление зависимостей не должно требовать полной переработки.

2.2 Функциональная модель системы

Функциональная модель системы – это абстрактное описание поведения системы с точки зрения её взаимодействия с внешними сущностями (акторами) и выполняемых функций (прецедентов). В контексте *UML* (*Unified* *Modeling* *Language*) такая модель чаще всего представляется в виде диаграммы прецедентов (*use* *case* *diagram*), которая визуализирует функциональные требования к системе [19].

Ключевые элементы функциональной модели на основе диаграммы прецедентов:

1. Система (*System*) – границы анализируемой системы, внутри которой находятся прецеденты (функциональные возможности).
2. Акторы (*Actors*) – внешние сущности (пользователи, другие системы или устройства), взаимодействующие с системой:
   1. Первичные акторы – инициируют взаимодействие (например, пользователь).
   2. Вторичные акторы – участвуют в процессе, но не являются инициаторами (например, внешний сервис).
3. Прецеденты (*Use* *Cases*) – отдельные функции или сценарии использования системы, описывающие её поведение при взаимодействии с актором:
   1. Примеры: «Аутентификация пользователя», «Оформление заказа», «Формирование отчёта».
4. Связи (*Relationships*) – взаимодействия между акторами и прецедентами:
   1. Ассоциация – связь актора с прецедентом.
   2. Включение (*Include*) – обязательное использование одного прецедента внутри другого.
   3. Расширение (*Extend*) – условное добавление функциональности.
   4. Обобщение (*Generalization*) – наследование ролей.

Назначение функциональной модели:

1. Определение границ системы и её ключевых функций.
2. Формализация требований к системе на ранних этапах проектирования.
3. Упрощение коммуникации между заинтересованными сторонами (аналитиками, разработчиками, заказчиками).

Диаграмма прецедентов проектируемой системы представлена на рисунке 1.

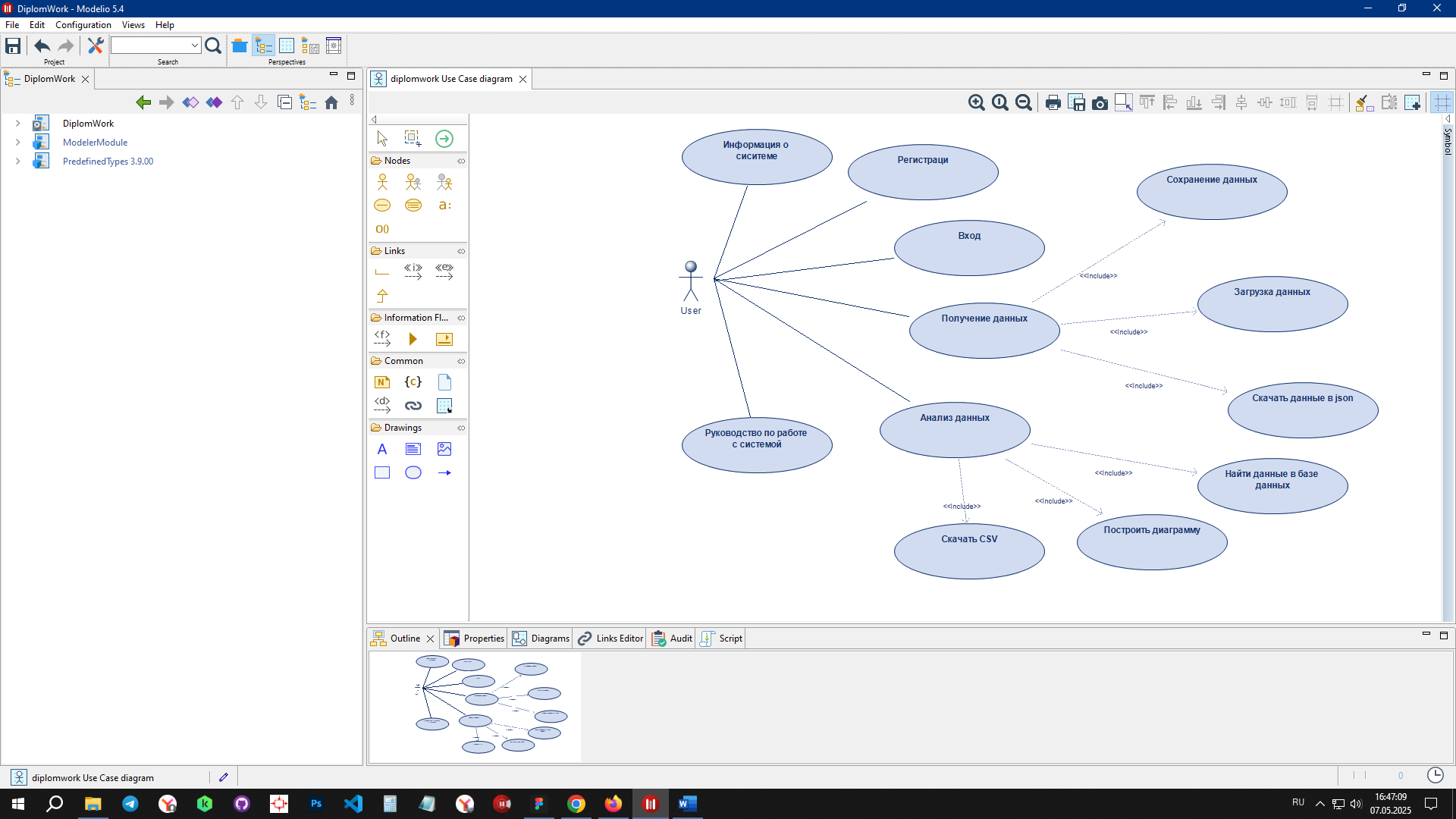


Рисунок 1 – Диаграмма прецедентов

Данная диаграмма вариантов использования демонстрирует взаимодействие пользователя с системой сбора и анализа данных о строительных материалах.

Актор:

*User* (Пользователь) — основной актор, взаимодействующий с системой.

Основные варианты использования:

1. Информация о системе — пользователь получает общее описание функционала и возможностей платформы.
2. Регистрация / Вход / Выход — базовые функции аутентификации, позволяющие пользователю создать аккаунт, войти в систему и завершить сессию.
3. Получение данных – основной модуль, включающий в себя:
   1. Сохранение данных.
   2. Загрузка данных.
   3. Скачивание данных в *JSON.*
   4. Поиск данных в базе.

Эти функции обеспечивают взаимодействие с системой сбора и хранения информации: автоматический веб-скрейпинг, добавление в БД, выгрузка и поиск.

1. Анализ данных – включает функции анализа собранной информации:
   1. Скачать *CSV* – экспорт результатов анализа.
   2. Построить диаграмму – визуализация изменения цены товара за выбранный период.
2. Руководство по работе с системой – помогает пользователю освоить функционал платформы, предоставляя справочную информацию.

2.3 Архитектура системы

Архитектура системы в контексте диаграммы развертывания — это описание физической структуры программного обеспечения, включающее распределение компонентов системы по аппаратным узлам, связи между ними и инфраструктурные элементы, необходимые для функционирования системы [19].

Ключевые аспекты архитектуры на диаграмме развертывания

1. Узлы (*Nodes*):
   1. Представляют физические или виртуальные устройства (серверы, компьютеры, облачные инстансы, *IoT*-устройства).
   2. Примеры: Веб-сервер, Сервер БД, Клиентское устройство.
2. Артефакты (*Artifacts*):
   1. Исполняемые файлы, библиотеки, контейнеры (*Docker*), виртуальные машины.
   2. Примеры: *backend*.*jar*, *nginx*:*latest*, *database*.*dump*.
3. Коммуникационные пути (*Communication* *Paths*):
   1. Соединения между узлами (сети, протоколы).
   2. Примеры: *HTTP*/*TCP*, *gRPC*, *WebSocket*.
4. Зависимости и связи:
   1. Показывают, какие компоненты развернуты на каких узлах и как они взаимодействуют.

Цель создания:

* 1. Понять, как развертывается система (локально, в облаке, гибридно).
  2. Определить узкие места (например, нагрузку на сервер БД).
  3. Спланировать масштабирование (репликация, балансировка).

Диаграмма развертывания в *UML* помогает визуализировать «где что работает» и «как компоненты связаны физически». Это особенно важно для *DevOps*, архитекторов и администраторов. Диаграмма развертывания представлена на рисунке 2.

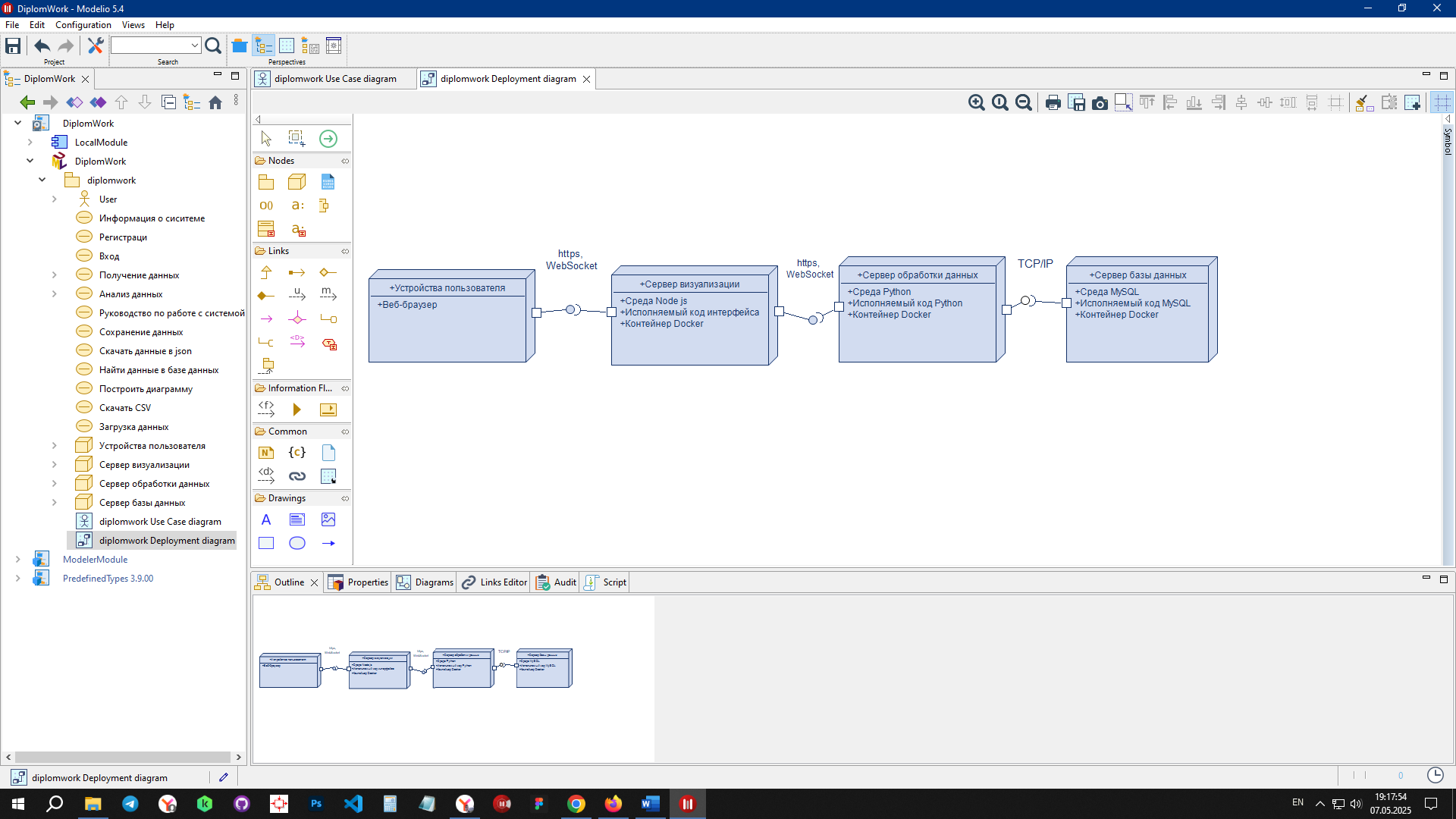


Рисунок 2 – Диаграмма развертывания

На диаграмме представлена архитектура системы, включающая взаимодействие пользовательских устройств с серверными компонентами через различные сетевые протоколы.

1. Устройства пользователя
   1. Клиентское устройство (веб-браузер), которое взаимодействует с серверными компонентами.
2. Серверные компоненты:
   1. Сервер визуализации:
      1. Протокол: *WebSocket* (двустороннее взаимодействие в реальном времени) и *TCP*/*IP*.
      2. Функции:
         1. Обработка и отображение данных.
         2. Использует код интерфейса (фронтенд-логику).
      3. Среда развертывания: *Docker* (контейнеризация).
   2. Сервер обработки данных:
      1. Протокол: *WebSocket*.
      2. Функции:
         1. Анализ и преобразование данных.
         2. Использует *Python* (бэкенд-логику и *ML*-модели).
      3. Среда развертывания: *Docker*.
   3. Сервер файла данных (БД)
      1. Протокол: *TCP*/*IP* (стандартный сетевой протокол для надежной передачи).
      2. Функции:
         1. Хранение и управление данными через *MySQL*.
         2. Среда развертывания: *Docker*.
3. Взаимодействие между компонентами
   1. Клиентское устройство подключается к серверу визуализации через *WebSocket*.
   2. Сервер визуализации может запрашивать данные у сервера обработки данных (также через *WebSocket*).
   3. Сервер обработки данных взаимодействует с сервером БД (*MySQL*) по *TCP*/*IP* для чтения/записи данных.
4. Система использует микросервисную архитектуру с контейнеризацией (*Docker*), где:
   1. Визуализация и обработка данных работают через *WebSocket* (низкие задержки).
   2. Хранение данных обеспечивается классической *SQL*-БД (*MySQL*).
   3. Все компоненты изолированы и масштабируемы благодаря *Docker*.
      1. Такой подход подходит для систем реального времени.

2.4 Моделирование данных

Для моделирования данных проектируемой системы использованы диаграмма классов и *ER* диаграмма.

Диаграмма классов — это один из типов диаграмм в языке моделирования *UML* (*Unified* *Modeling* *Language*), предназначенный для отображения структуры системы на уровне классов [19].  
Она показывает:

1. Классы системы.
2. Их атрибуты (поля).
3. Методы (функции, поведение).
4. Связи между классами: ассоциации, наследование, агрегация, композиция.

Диаграмма классов помогает разработчику и аналитикам понять архитектуру проекта, структуру данных и логические взаимосвязи объектов и представлена она на рисунке 3.

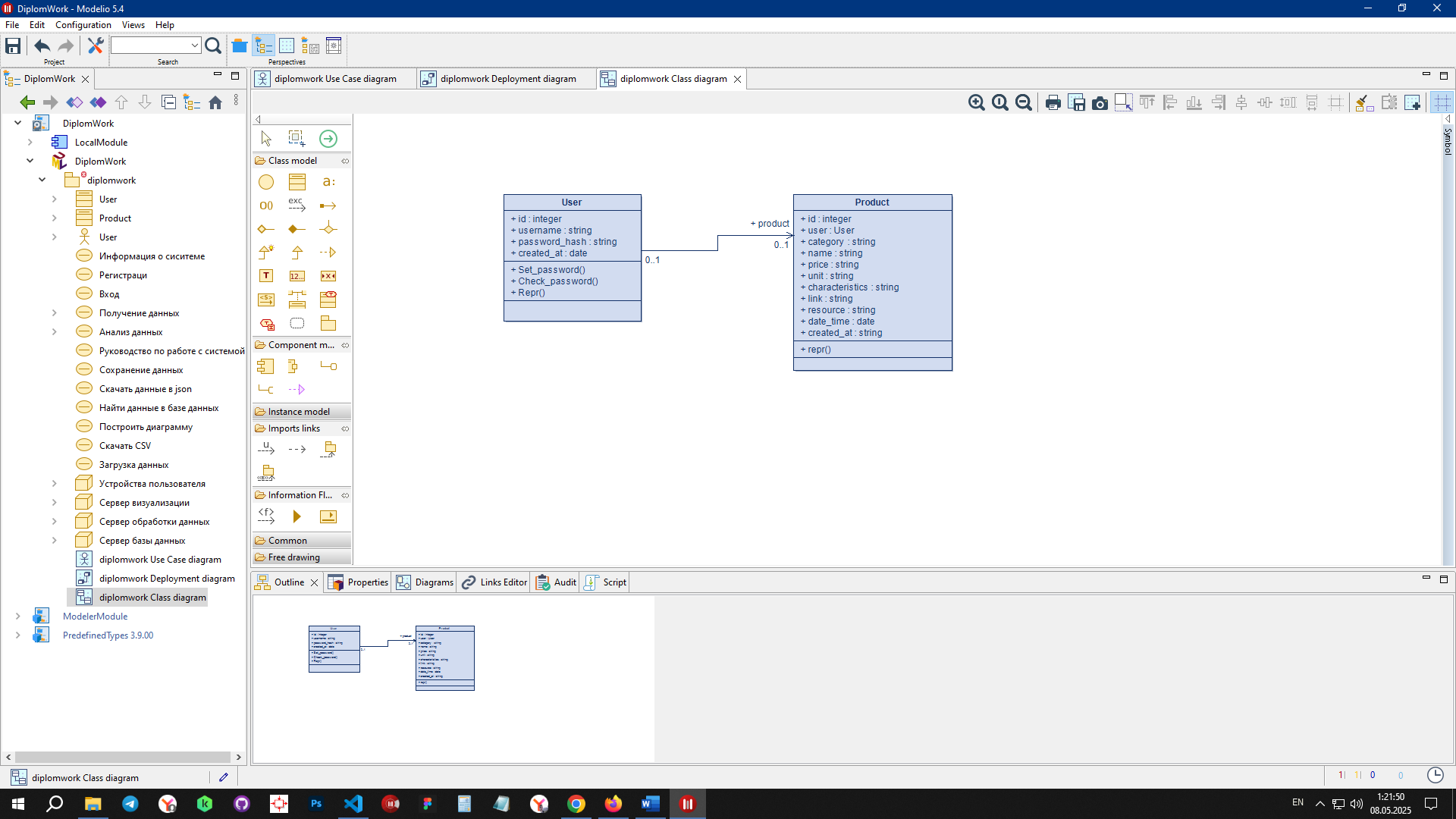


Рисунок 3 – Диаграмма классов

Диаграмма отражает структуру двух основных классов: *User* и *Product*.  
Она реализует функциональность, где один пользователь может иметь несколько загруженных продуктов (данных).

Класс: *User*

Сущность пользователя, который взаимодействует с системой.

Атрибуты:

1. *id*:*integer* — уникальный идентификатор пользователя (первичный ключ).
2. *username*:*string* — имя пользователя.
3. *password\_hash*:*string* — хэшированный пароль.
4. *created\_at*:*date* — дата регистрации пользователя.

Методы:

1. *Set\_password*() — установка пароля с последующим хэшированием.
2. *Check\_password*() — проверка соответствия пароля.
3. *Repr*() — строковое представление объекта (служебный метод).

Класс: *Product*

Представляет продукт (товар со строительного сайта), загруженный пользователем.

Атрибуты:

1. *id*:*integer* — уникальный идентификатор продукта.
2. *user*:*User* — ссылка на пользователя, загрузившего продукт (внешний ключ).
3. *category*:*string* — категория продукта (например, цемент, кирпич и т.п.).
4. *name*:*string* — наименование товара.
5. *price*:*string* — цена (возможно строкой для поддержки разных форматов).
6. *unit*:*string* — единица измерения.
7. *characteristics*:*string* — описание или характеристики товара.
8. *link*:*string* — *URL* на страницу товара.
9. *resource*:*string* — источник (например, название сайта).
10. *date\_time*:*date* — дата получения/обновления данных.
11. *created\_at*:*string* — дата и время создания записи (в строковом формате).

Методы:

1. *repr*() — строковое представление объекта (служебный метод).

Типы связи между классами:

1. Между *User* и *Product* установлена ассоциация один-ко-многим:
   1. Один *User* может быть связан с множеством объектов *Product*.
   2. У каждого *Product* может быть только один владелец (*User*).

*ER*-диаграмма — это визуальная модель данных, которая описывает:

1. Сущности (таблицы в БД) и их атрибуты (поля).
2. Связи между сущностями (один-ко-одному, один-ко-многим, многие-ко-многим).

Ключевые элементы:

1. Сущность — объект (Пользователь, Продукт).
2. Атрибут — свойство сущности.
3. Первичный ключ (*PK*) — уникальный идентификатор (*id*).
4. Внешний ключ (*FK*) — ссылка на первичный ключ другой сущности (*id* пользователя).

Типы связей:

1. 1: 1 (Один к одному) — например, паспорт и человек.
2. 1: *N* (Один ко многим) — например, пользователь и его заказы.
3. *M*: *N* (Многие ко многим) — например, студенты и курсы (с промежуточной таблицей).

Применение:

1. Проектирование структуры базы данных.
2. Документирование требований к данным.
3. Обсуждение с командой перед реализацией.

*ER* диаграмма проектируемой системы представлена на рисунке 4.



Рисунок 4 – *ER* диаграмма

На диаграмме представлена модель данных для проектируемой системы.

Сущности (*Entities*)

1. Пользователь
   1. Атрибуты:
      1. *id* (уникальный идентификатор)
      2. Логин
      3. Пароль
      4. Дата регистрации
2. Данные
   1. Атрибуты:
      1. *id*(уникальный идентификатор)
      2. Цена
      3. Дата сбора
      4. Учетная единица (например, штуки, литры)
      5. Ссылка (возможно, *URL* или путь к файлу)
      6. Описание
      7. Ресурс (основная информация)
      8. *id* пользователя (внешний ключ)

Связи (*Relationships*)

1. Пользователь → Данные
   1. Связь: «1 ко многим» (один пользователь может иметь множество записей данных).
   2. Обозначение: «Имеет» (стрелка).

2.5 Моделирование логики работы

Диаграмма процесса работы системы в нотации *BPMN* (*Business* *Process* *Model* *and* *Notation*) представляет собой инструмент визуального моделирования, предназначенный для описания последовательности операций, принимаемых решений, взаимодействий между участниками процесса и внешними системами. Основное назначение *BPMN*-диаграммы — обеспечить понятное и стандартизированное отображение логики функционирования системы или процесса, которое будет одинаково доступно для анализа как техническими специалистами, так и пользователями.

Моделирование процессов с использованием *BPMN* позволяет формализовать поведение системы на различных этапах, выделить роли участников, показать порядок выполнения задач, условия переходов и варианты развития сценариев. Это особенно важно при проектировании программных решений, так как способствует выявлению узких мест, потенциальных ошибок в логике и неочевидных зависимостей между компонентами. Диаграмма способствует унификации понимания требований и процессов между всеми участниками разработки.

*BPMN* поддерживает использование таких элементов, как события (начало, завершение, промежуточные сигналы), задачи (автоматические или выполняемые вручную), шлюзы (для описания условий и ветвлений) и потоки управления. Благодаря своей гибкости и четкой структуре, нотация *BPMN* позволяет как моделировать простые пользовательские сценарии, так и описывать комплексную архитектуру взаимодействия системных компонентов.

Таким образом, построение *BPMN*-диаграммы является важным этапом анализа и проектирования логики функционирования информационной системы. Она способствует более глубокому пониманию процессов, облегчает сопровождение и развитие системы, а также может быть использована как часть документации при защите проекта или передаче его в дальнейшую разработку [19].

Диаграмма процесса взаимодействия пользователя и проектируемой системы в нотации *BPMN* представлена на рисунке 5.

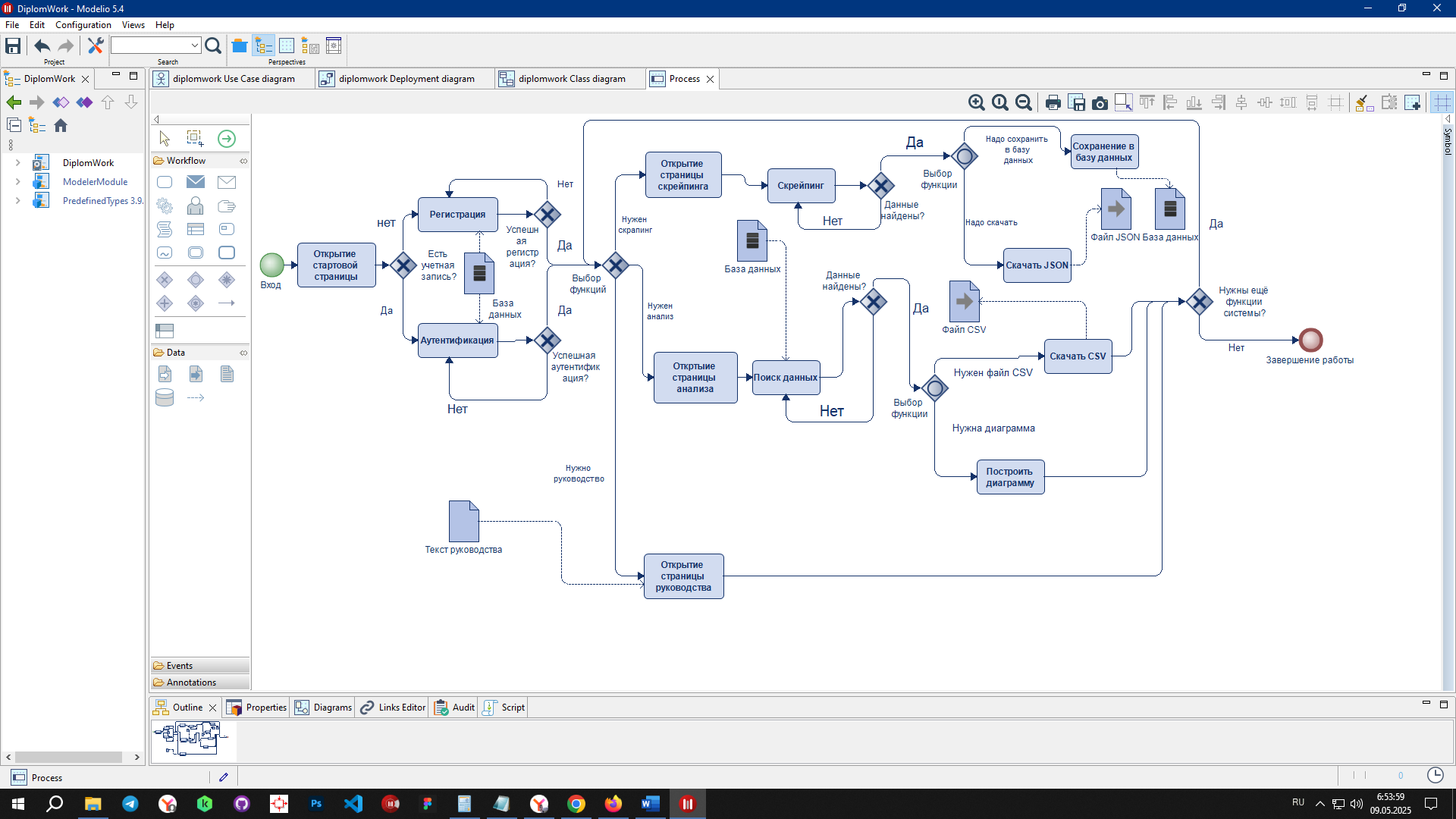


Рисунок 5 – Диаграмма процесса взаимодействия пользователя и проектируемой системы

Данная диаграмма имеет:

1. Точка входа в систему – с нее начинается взаимодействие пользователя в системе.
2. Функция открытия стартовой страницы – эта страница представляет пользователю общую информацию о системе.
3. Если пользователь хочет использовать остальной функционал систему, то ему необходимо зарегистрироваться или, при наличии учетной записи, провести аутентификацию при этом будет использована база данных системы.
4. Если регистрация или аутентификация прошла не успешно, то необходимо изменить входные данные и попробовать снова.
5. Если регистрация или аутентификация прошла успешно, то пользователь может выбрать необходимый ему функционал:
   1. Функция открытия страницы веб-скрейпинга выполняется при выборе соответствующей ссылки в интерфейсе пользователем, на этой странице пользователю доступна функция веб-скрейпинга, в которой он может ввести входные данные по заданным правилам, если необходимые данные не найдены, то можно повторить, если же найдены, то есть возможность их скачать в виде файла *JSON* или сохранить в базу данных для дальнейшего анализа внутри системы. При желании пользователь может выбрать другую функцию системы или закончить работу.
   2. Функция открытия страницы анализа выполняется при выборе соответствующей ссылки в интерфейсе пользователем, на этой странице пользователю доступна функция поиск данных, если необходимые данные не найдены, то можно изменить соответствующие параметры и повторить поиск, если найдены, то можно построить диаграмму или скачать файл в формате *CSV*, при этом будет использована база данных системы. При желании пользователь может выбрать другую функцию системы или закончить работу.
   3. Функция открытие страницы руководства выполняется при выборе соответствующей ссылки в интерфейсе пользователем, на этой странице пользователю доступна подробная информация о использовании системы в виде онлайн библиотеки. При желании пользователь может выбрать другую функцию системы или закончить работу.
6. Конечной точкой работы пользователя с системой является выход из системы.

Так же опишем декомпозицию выполнения основных процессов работы системы таких как: «Регистрация», «Аутентификация», «Скрейпинг», «Поиск данных», «Сохранение в базу данных», «Скачать *JSON*», «Скачать *CSV*», «Построить диаграмму».

Диаграмма процесса «Регистрация» представлена на рисунке 6.

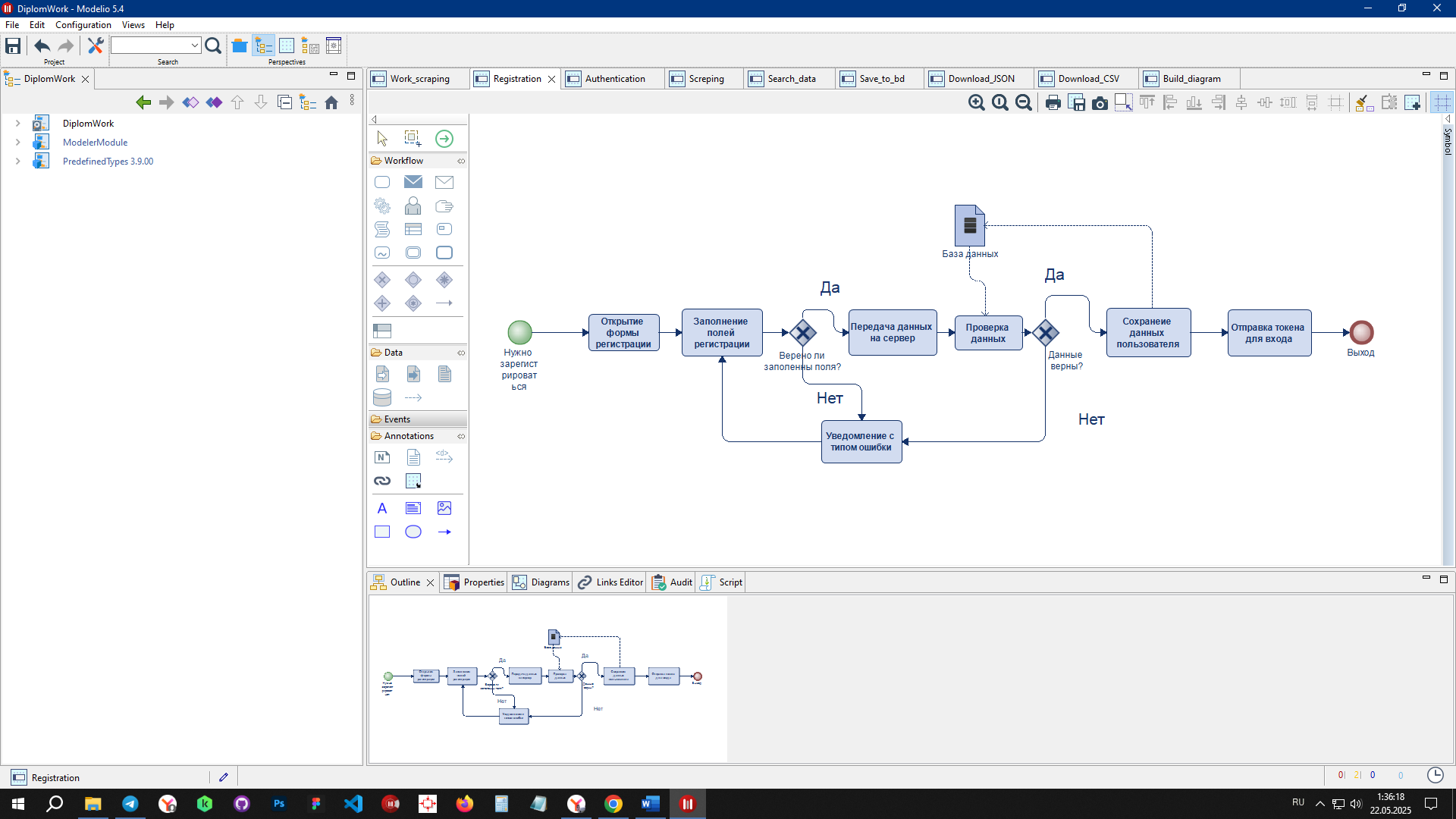


Рисунок 6 – Диаграмма процесса «Регистрация»

В данной диаграмме процесса регистрации стартовое событие обозначает необходимость регистрации пользователя. Процесс начинается с открытия формы регистрации, где пользователь заполняет необходимые поля. Затем осуществляется проверка корректности заполненных данных. Если поля заполнены верно, данные передаются на сервер, где происходит проверка информации в базе данных. Если данные подтверждаются как верные, выполняется сохранение данных пользователя, и генерируется токен для входа. Процесс завершается на этапе выхода. Однако, если при проверке данных обнаруживаются ошибки, пользователю выдается уведомление с типом ошибки, и он возвращается к этапу заполнения полей для исправления информации.

Диаграмма процесса «Аутентификация» представлена на рисунке 7.

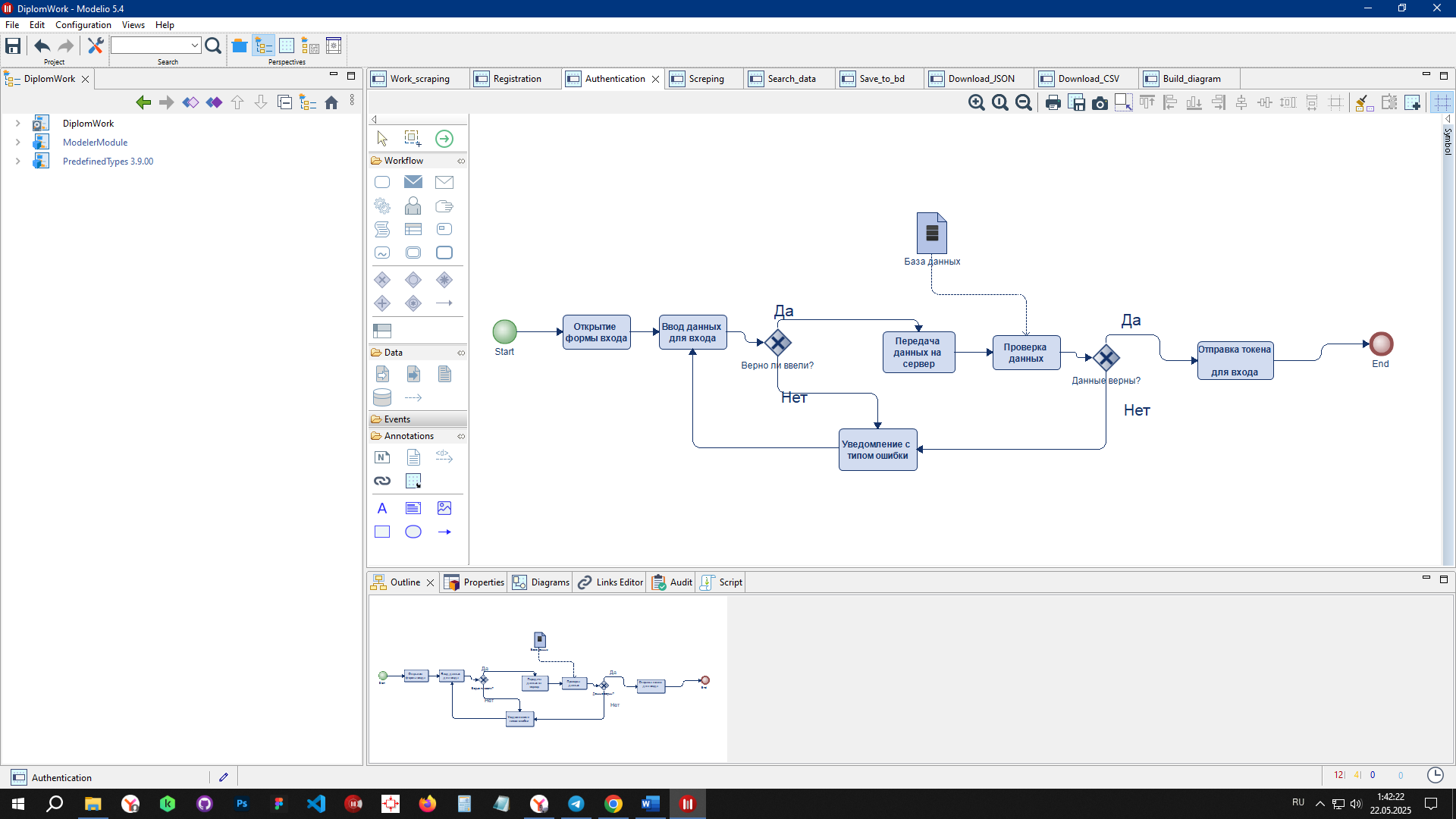


Рисунок 7 – Диаграмма процесса «Аутентификация»

В данной блок-схеме процесса входа в систему стартовое событие обозначает начало процесса. Сначала происходит открытие формы входа, после чего пользователь вводит свои данные для доступа. Затем осуществляется проверка на корректность введенных данных. Если данные введены верно, информация передается на сервер, где происходит дальнейшая проверка в базе данных. Если данные подтверждаются как верные, пользователю отправляется токен для входа, и процесс завершается. В случае, если на любом этапе проверки ответ оказывается отрицательным, пользователю выдается уведомление с типом ошибки, и он возвращается к этапу ввода данных для исправления. База данных играет ключевую роль в процессе проверки данных.

Диаграмма процесса «Скрейпинг» представлена на рисунке 8.

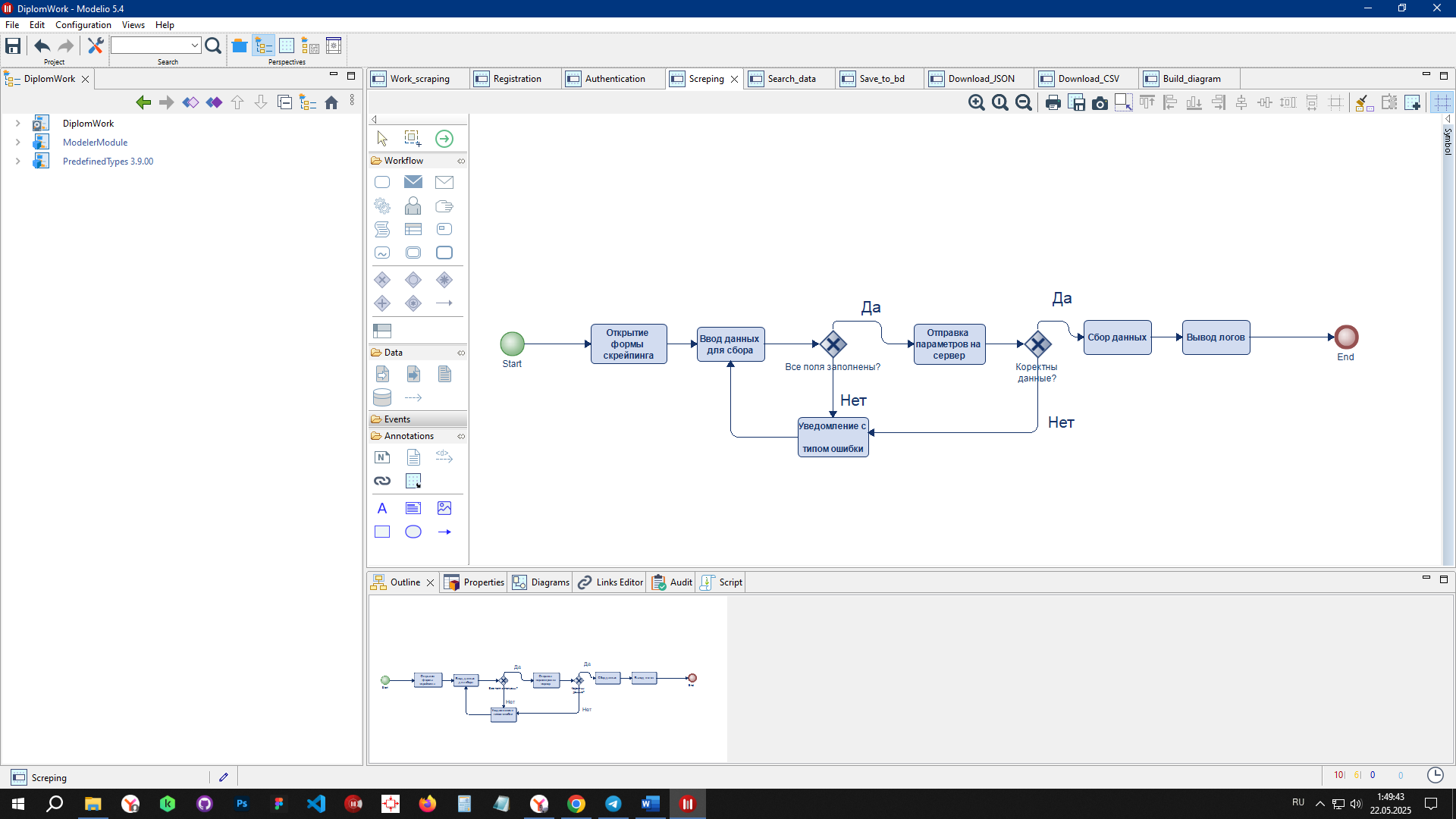


Рисунок 8 – Диаграмма процесса «Скрейпинг»

В данной блок-схеме процесс веб-скрейпинга начинается со стартового события, обозначенного как "*Start*". Далее открывается форма скрейпинга, в которой пользователь вводит данные для сбора. После этого осуществляется проверка: все ли поля заполнены?

Если ответ "Да", параметры отправляются на сервер для дальнейшей обработки. В противном случае, если ответ "Нет", пользователю выдается уведомление с типом ошибки, и он возвращается к этапу ввода данных.

После отправки параметров на сервер происходит проверка корректности данных. Если данные верны, начинается процесс сбора данных, который завершается выводом логов о выполненной операции. Процесс завершается на этапе "Выход".

Диаграмма процесса «Поиск данных» представлена на рисунке 9.

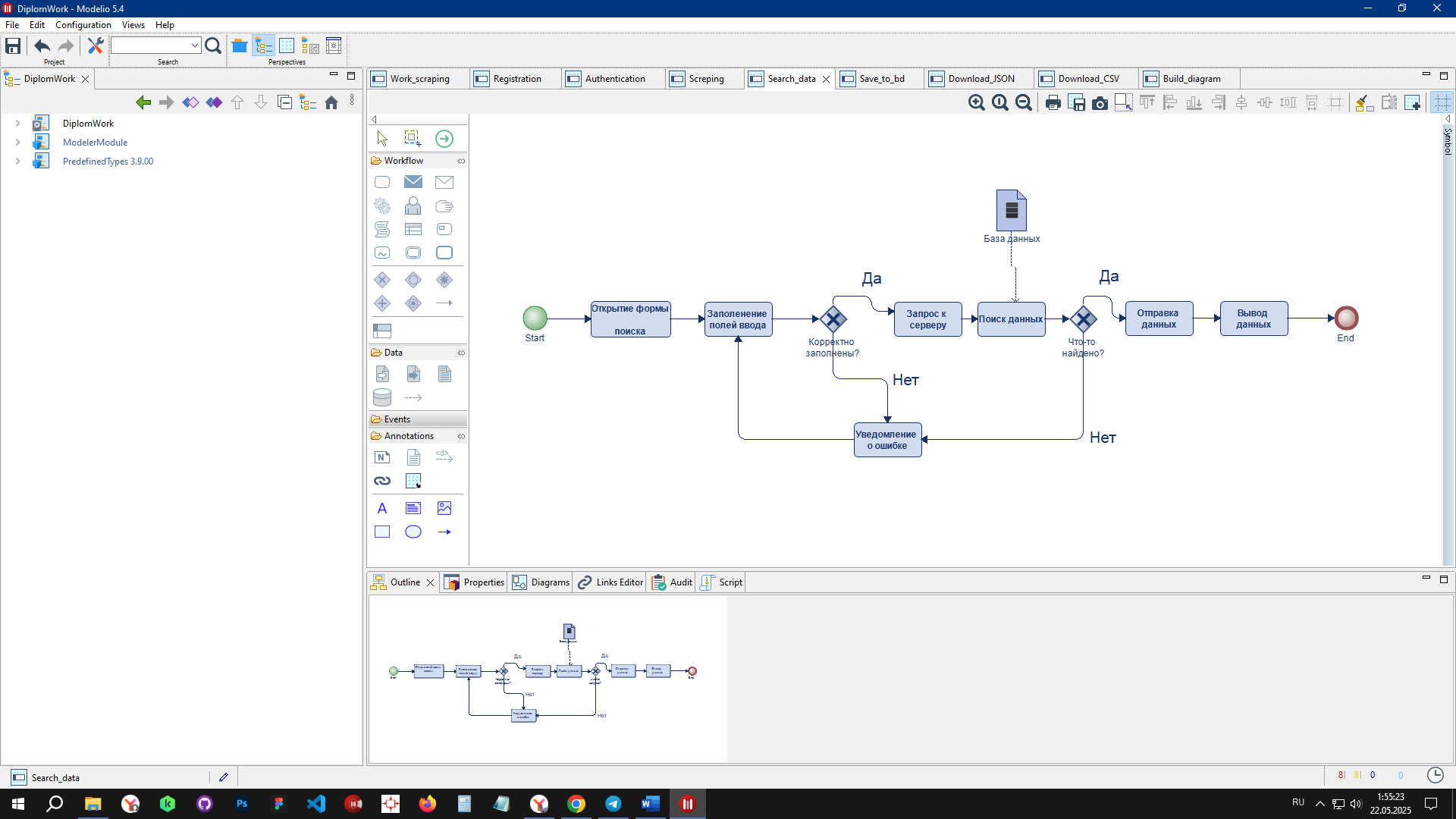


Рисунок 9 – Диаграмма процесса «Поиск данных»

Процесс поиска данных начинается со стартовой точки, где пользователь открывает форму для поиска. На следующем этапе он вводит необходимые данные, после чего система проверяет, правильно ли заполнены поля. Если данные введены корректно, происходит отправка запроса на сервер, который начинает поиск информации в базе данных. После завершения поиска система определяет, были ли найдены данные. Если информация доступна, она отправляется пользователю, иначе процесс возвращается на этап поиска данных. В конце, после успешного вывода информации, процесс завершается.

Диаграмма процесса «Сохранение в базу данных» представлена на рисунке 10.

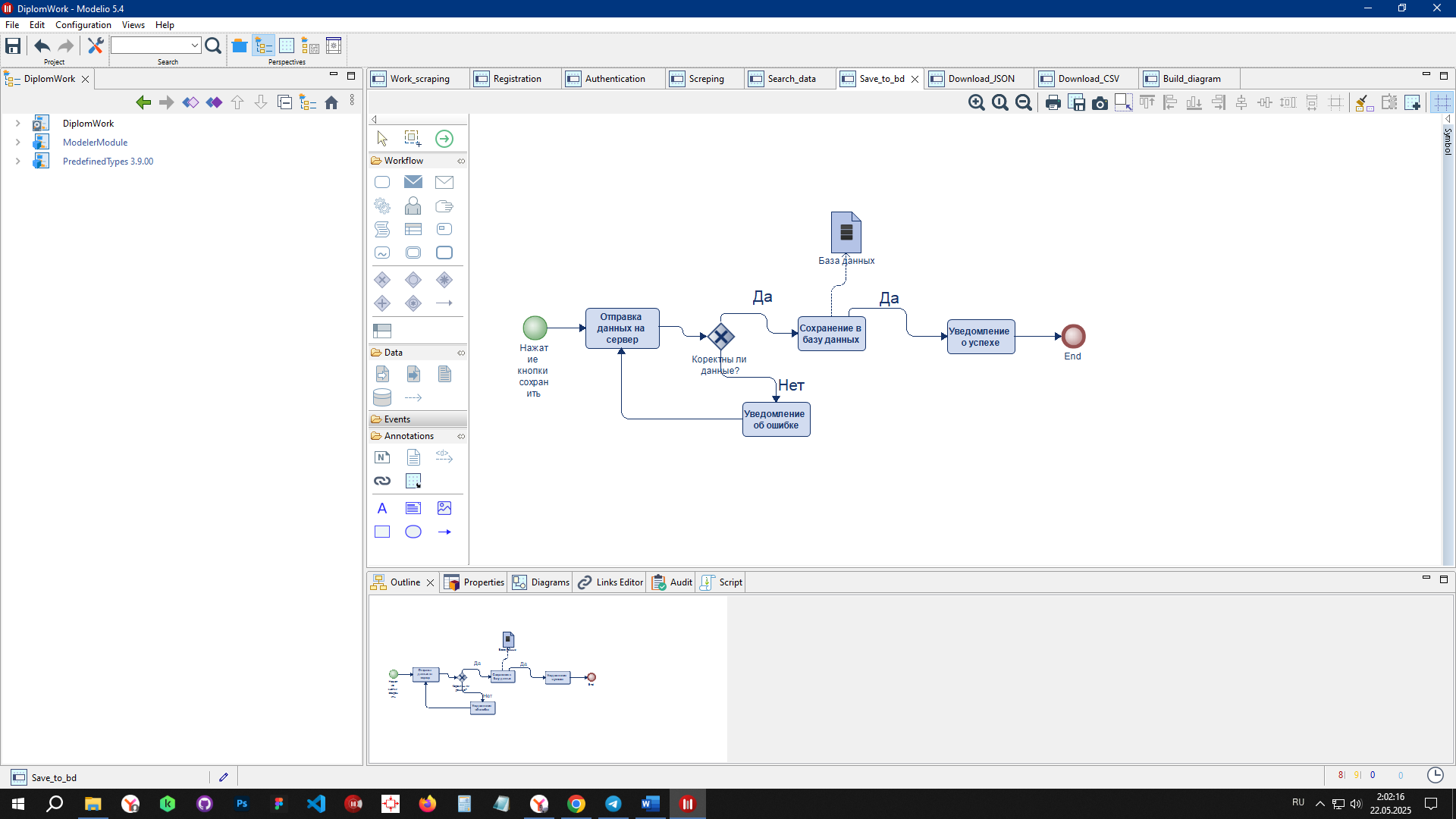


Рисунок 10 – Диаграмма процесса «Сохранение в базу данных»

Процесс начинается с нажатия кнопки "Сохранить". После этого данные отправляются на сервер. На следующем этапе происходит проверка: корректны ли данные? Если ответ положительный, осуществляется сохранение информации в базу данных, после чего пользователю отправляется уведомление об успешном завершении операции, и процесс завершается.

Если же данные оказались некорректными, пользователю выдается уведомление об ошибке, и процесс возвращается к этапу отправки данных на сервер для повторной попытки.

Диаграмма процесса «Скачать *JSON*» представлена на рисунке 11.

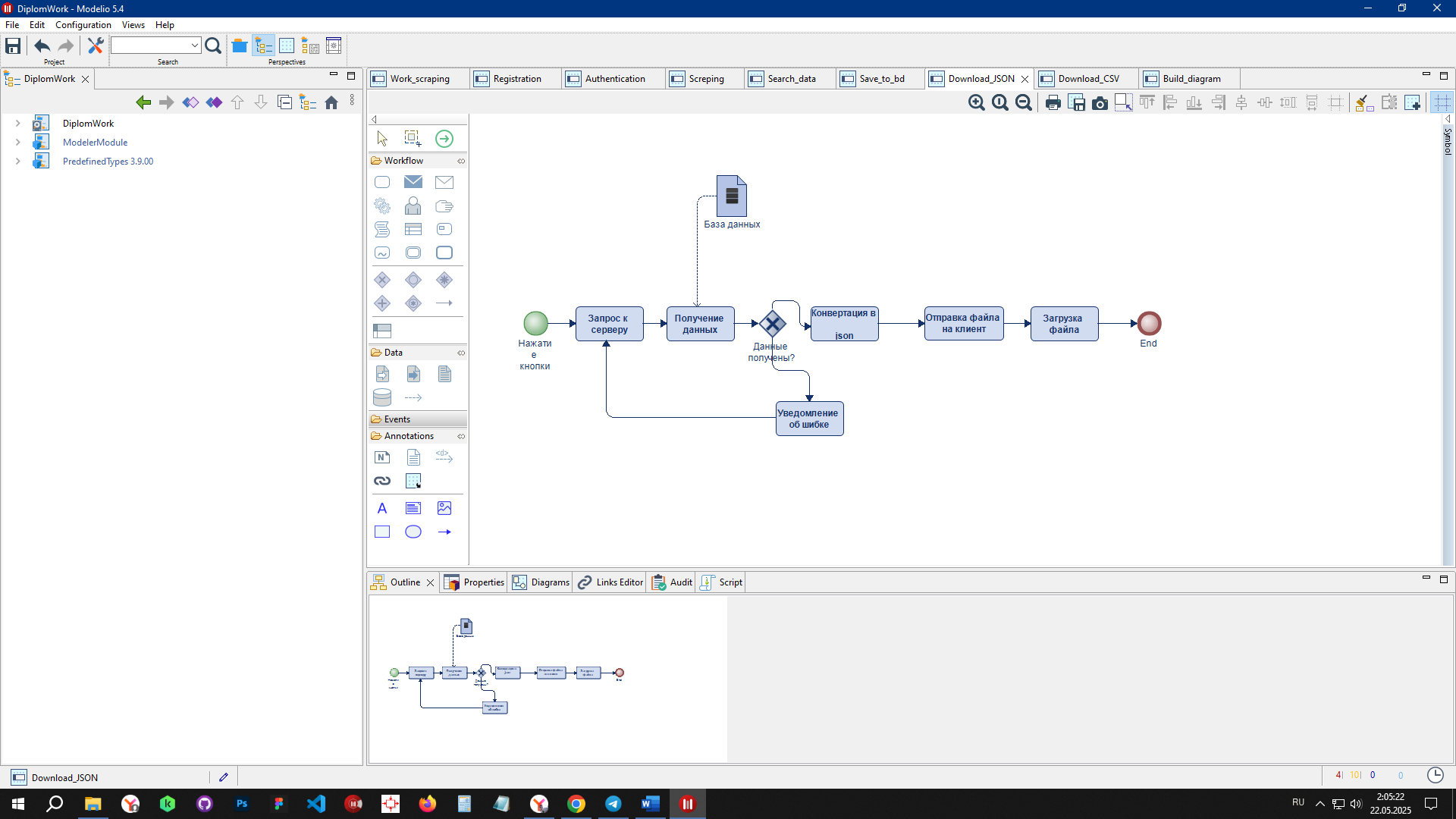


Рисунок 11 – Диаграмма процесса «Скачать *JSON*»

Процесс начинается с нажатия кнопки, после чего осуществляется запрос к серверу. Далее сервер получает данные из базы данных. Следующим этапом происходит проверка: были ли данные успешно получены? Если данные не получены, система выдает уведомление об ошибке и возвращается к этапу запроса к серверу. В случае успешного получения данных происходит конвертация информации в формат *JSON*, после чего файл отправляется на клиент. Затем осуществляется загрузка файла, и процесс завершается на этапе «*End*».

Диаграмма процесса «Скачать *CSV*» представлена на рисунке 12.

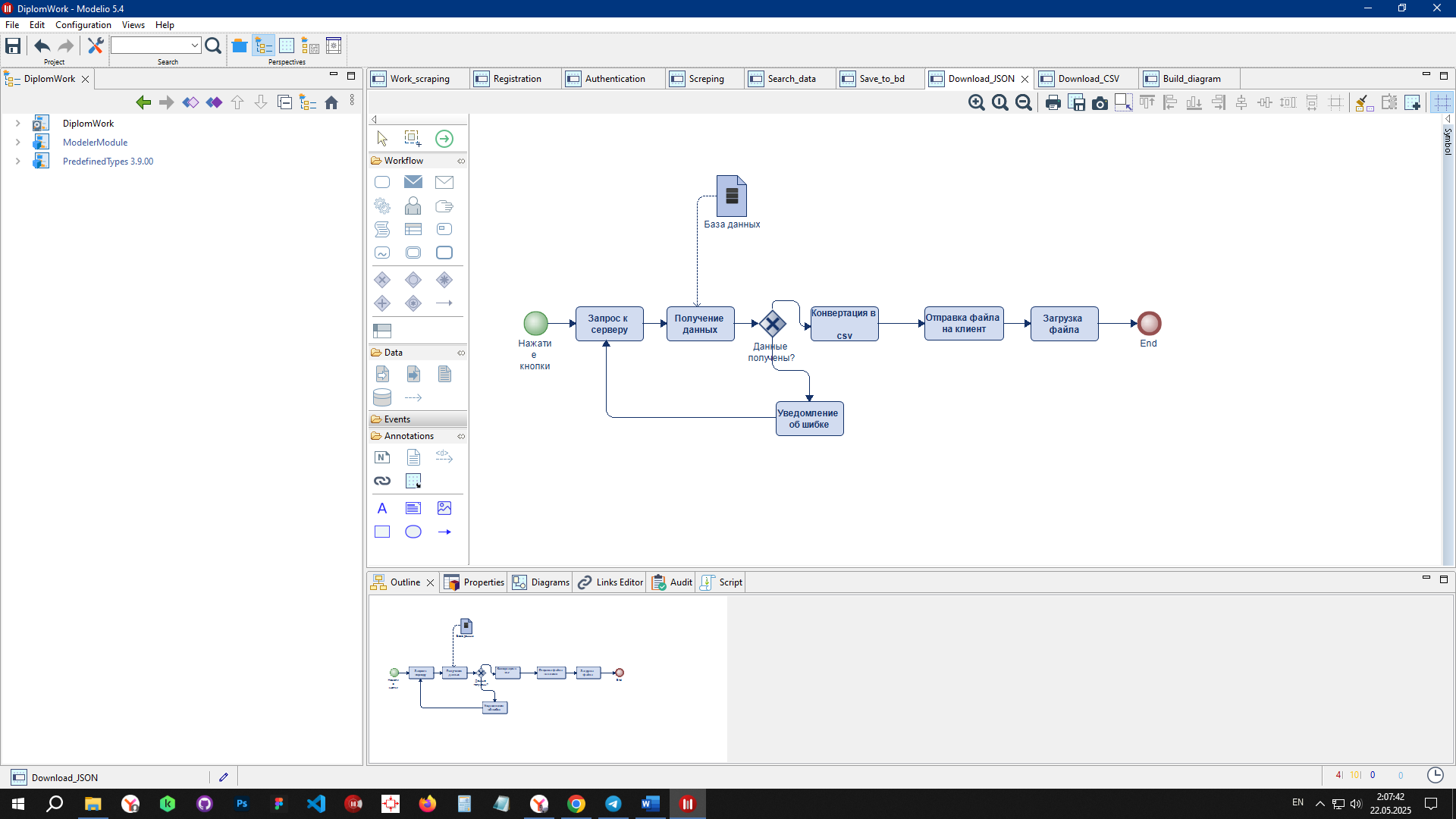


Рисунок 12 – Диаграмма процесса «Скачать *CSV*»

Процесс начинается с нажатия кнопки, после чего осуществляется запрос к серверу. Сервер получает данные из базы данных, и далее происходит проверка: были ли данные успешно получены? Если данные не получены, система выдает уведомление об ошибке и возвращается к этапу запроса к серверу. В случае успешного получения данных происходит конвертация информации в формат *CSV*, после чего файл отправляется на клиент. Затем осуществляется загрузка файла, и процесс завершается на этапе "*End*".

Диаграмма процесса «Построить диаграмму» представлена на рисунке 13.

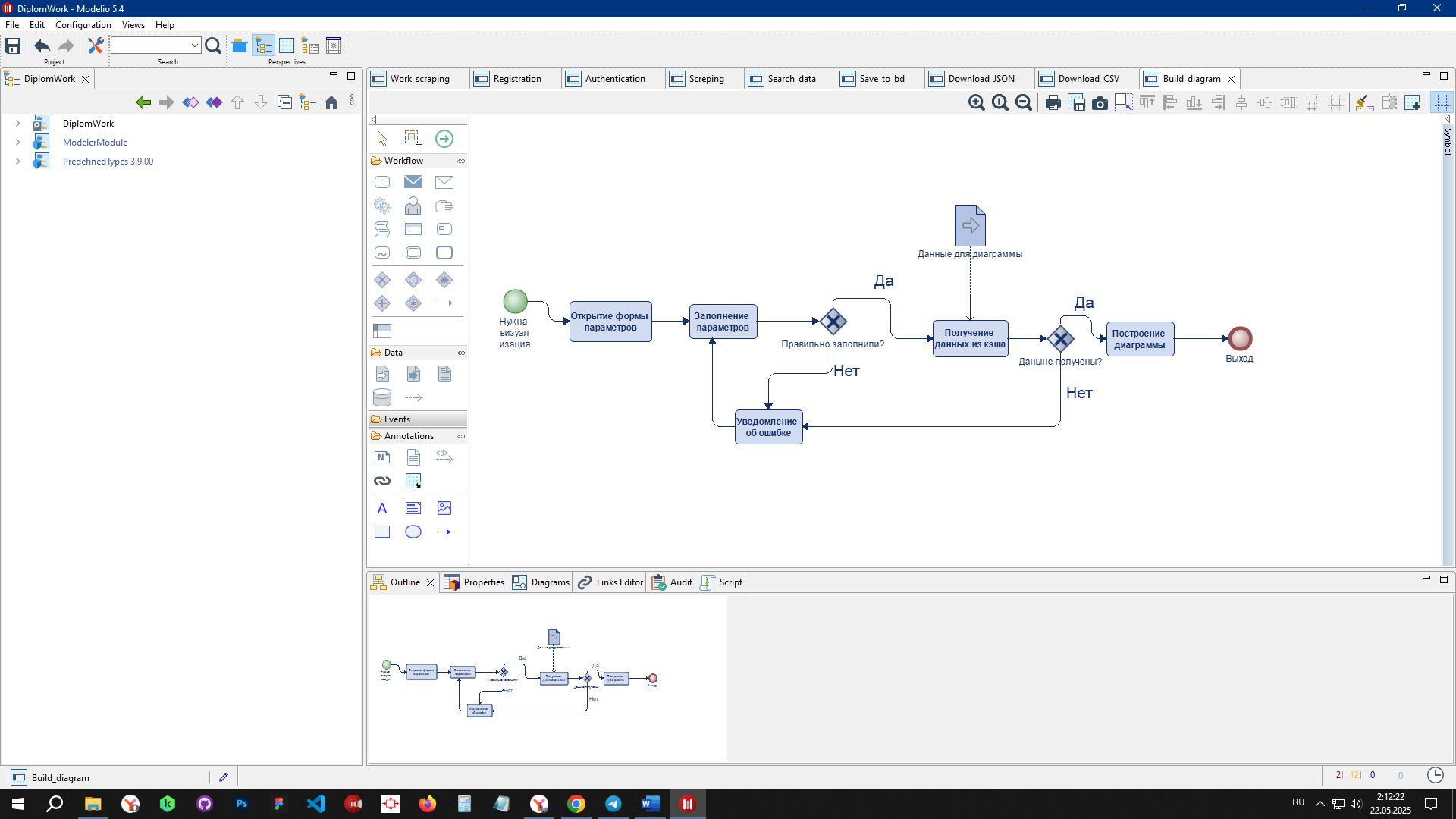


Рисунок 13 – Диаграмма процесса «Построить диаграмму»

Процесс визуализации данных начинается с события начала, обозначенного как "Нужна визуализация". Далее пользователь открывает форму параметров, где вводит необходимые данные. После этого происходит проверка корректности заполнения параметров. В случае ошибки пользователю выводится уведомление об ошибке, и он возвращается к этапу заполнения параметров. Если данные заполнены правильно, процесс переходит к этапу получения данных из кэша, после чего происходит проверка на наличие необходимых данных. Если данные не получены, снова выводится уведомление об ошибке, и пользователь возвращается к заполнению параметров. В случае успешного получения данных процесс переходит к построению диаграммы, после чего завершается событием выхода.

В данной главе были сформированы функциональные и нефункциональные требования к разрабатываемой системе, позволяющие обеспечить её целостность, надёжность и удобство использования. Определены ключевые модули и компоненты, среди которых особое внимание уделено процессу веб-скрейпинга, хранению и анализу данных, пользовательскому интерфейсу, а также безопасности и масштабируемости системы.

С помощью функциональной модели в виде диаграммы прецедентов визуализированы основные сценарии взаимодействия пользователя с системой, что позволяет лучше понять её поведение и логическую структуру. Архитектурная модель, представленная в виде диаграммы развертывания, демонстрирует физическую организацию системы, распределение компонентов и взаимодействие между ними.

Таким образом, проведённое проектирование закладывает основу для последующей реализации, обеспечивая чёткое понимание требований и архитектурных решений, необходимых для создания эффективной, гибкой и надёжной информационной системы по сбору и анализу данных о строительных материалах.

3 РАЗРАБОТКА СИСТЕМЫ

3.1 Этапы разработки

Для успешной реализации проекта необходимо применить спиральную модель жизненного цикла разработки программного обеспечения [19], которая обеспечивает:

1. Итеративный подход:
   1. Требуется разбить весь процесс разработки на последовательные циклы (итерации).
   2. Каждая итерация должна включать: планирование, анализ рисков, разработку и тестирование.
   3. Необходимо предусмотреть возможность пересмотра требований после каждой итерации.
2. Управление рисками:
   1. Важно проводить регулярный анализ потенциальных рисков.
   2. Требуется разработать стратегии минимизации выявленных рисков.
   3. Необходимо предусмотреть резервные решения для критически важных компонентов системы.
3. Гибкость процесса:
   1. Следует обеспечить возможность параллельной разработки отдельных модулей.
   2. Требуется реализовать механизм быстрого внесения изменений.
   3. Необходимо предусмотреть поэтапную интеграцию компонентов.
4. Контроль качества:
   1. Обязательно проводить тестирование на каждом этапе разработки.
   2. Требуется реализовать систему непрерывной интеграции.
   3. Необходимо обеспечить регулярный мониторинг работоспособности системы.

Детализация ключевых этапов разработки:

Проектирование пользовательского интерфейса.

На данном этапе необходимо:

1. Составить перечень необходимых экранов и элементов управления.
2. Создать *wireframes* основных экранов системы.
3. Проработать логику взаимодействия между элементами.
4. Оптимизировать пользовательские потоки.

Разработка фронтенд-компонентов.

Для реализации интерфейсной части системы требуется:

1. Разработать компонентную архитектуру
2. Настроить систему маршрутизации.
3. Реализовать базовые сервисы приложения.
4. Реализовать систему авторизации и управления доступом.
5. Создать модуль отображения и фильтрации данных.
6. Разработать компоненты визуализации информации.
7. Реализовать механизмы ленивой загрузки.
8. Настроить кэширование данных.
9. Оптимизировать работу с *DOM.*

Разработка серверной части

При реализации *backend*-компонентов необходимо:

1. Определить набор конечных точек.
2. Разработать схему взаимодействия с клиентом.
3. Оптимизировать структуру запросов и ответов.
4. Разработать систему веб-скрейпинга данных.
5. Создать механизмы обработки и хранения информации.
6. Реализовать сервисы аналитики и отчетности.
7. Реализовать механизмы аутентификации и авторизации.
8. Настроить валидацию входящих данных.
9. Защитить систему от распространенных уязвимостей.

Проектирование и реализация базы данных.

Для работы с данными требуется:

1. Определить основные сущности и их атрибуты.
2. Проработать связи между сущностями.
3. Оптимизировать структуру для выполнения ключевых запросов.
4. Создать схему базы данных.
5. Настроить индексы для ускорения работы.
6. Реализовать механизмы резервного копирования.
7. Проанализировать и оптимизировать сложные запросы.
8. Настроить систему кэширования.
9. Реализовать механизмы репликации.

Интеграция и развертывание.

Для успешного внедрения системы необходимо:

1. Настроить систему сборки и развертывания.
2. Реализовать механизмы управления зависимостями.
3. Автоматизировать процесс тестирования.
4. Настроить сбор метрик производительности.
5. Организовать централизованное логирование.
6. Составить руководство пользователя.
7. Подготовить техническую документацию.
8. Разработать инструкции по развертыванию.

Тестирование.

Для обеспечения качества системы требуется:

1. Проводить модульное тестирование компонентов.
2. Организовать интеграционное тестирование.
3. Выполнять системное тестирование.
4. Настроить непрерывную интеграцию.
5. Реализовать автоматическое прогонение тестов.
6. Организовать сбор и анализ результатов.
7. Организовать тестирование с участием пользователя и реальных данных.

3.2 Средства разработки

Фронтенд-разработка.

Для реализации клиентской части системы необходимо использовать следующий стек технологий:

1. Базовые технологии:
   1. *React*.*js* (*v*18.2) - основной фреймворк для построения пользовательского интерфейса [17].
   2. *Redux* *Toolkit* (*v*2.2) - управление состоянием приложения.
   3. *React* *Router* (*v*6.22) - навигация между страницами.
   4. *Axios* (*v*1.6) - работа с *HTTP*-запросами.
2. Визуализация данных:
   1. *Chart*.*js* (*v*4.4) - построение графиков и диаграмм [16].
   2. *Chart*.*js* *DataLabels* (*v*2.2) - дополнительные возможности оформления графиков.
3. Стилизация:
   1. *Sass* (*v*1.72) - препроцессор *CSS.*
   2. *Animate*.*css* (*v*4.1) - анимации интерфейса.
4. Средства сборки:
   1. *Create* *React* *App* (*v*5.0) - конфигурация проекта.
   2. *Node*.*js* (*v*22) - среда выполнения.

Бекенд-разработка.

Для серверной части системы требуется использовать:

1. Основной стек:
   1. *Python* (*v*3.12) - язык программирования [5].
   2. *Flask* (*v*2.3) - веб-фреймворк [16].
   3. *Flask*-*SQLAlchemy* - интеграция с базой данных.
   4. *Flask*-*JWT*-*Extended* – аутентификация.
2. Парсинг данных:
   1. *Scrapy* (*v*2.11) - фреймворк для веб-скрейпинга [6].
   2. *Playwright* - автоматизация браузера.
   3. *BeautifulSoup*4 - парсинг *HTML*
   4. *Pandas* (*v*2.1) - обработка данных.
3. Дополнительные библиотеки:
   1. *Requests* - *HTTP*-запросы.
   2. *PyMySQL* - драйвер *MySQL* [3].
   3. *python*-*dotenv* - работа с переменными окружения.

База данных.

Для хранения и обработки данных необходимо использовать:

1. *MySQL* (*v*8.0) - реляционная СУБД.
2. *SQLAlchemy* - *ORM* для *Python.*
3. *Alembic* - миграции базы данных.

Инфраструктура и развертывание.

Для контейнеризации и оркестрации сервисов требуется:

1. *Docker* (*v*24+) - контейнеризация приложений [4].
2. *Docker* *Compose* (*v*3.8) - оркестрация контейнеров.
3. *Nginx* - обратный прокси и балансировка нагрузки.

Конфигурация инфраструктуры включает:

1. Фронтенд-сервис:
   1. *Node*.*js* 22 базовый образ.
   2. Горячая перезагрузка в режиме разработки.
2. Бекенд-сервис:
   1. *Python* 3.12 *slim* образ.
   2. Установка зависимостей для парсинга.
   3. Интеграция *Playwright* для *Chromium.*
3. Сервис базы данных:
   1. *MySQL* 8.0 официальный образ.
   2. Предустановленные пользователи и база данных.
   3. Том для постоянного хранения данных.

Аппаратные требования.

Для разработки и тестирования системы необходимо обеспечить:

1. Минимальные требования:
   1. Процессор: 4 ядра (*Intel* *i*5/*Ryzen* 5 или выше).
   2. Оперативная память: 8 ГБ (рекомендуется 16 ГБ).
   3. Дисковое пространство: 20 ГБ свободного места.
2. Рекомендуемые характеристики:
   1. *SSD*-накопитель для повышения производительности.
   2. Выделенная видеокарта (для тестирования парсера).
   3. Стабильное интернет-соединение (10+ Мбит/с).
3. Операционная система:
   1. *Linux* (*Ubuntu* 24.04)

Данный стек технологий обеспечивает:

1. Высокую производительность системы.
2. Масштабируемость архитектуры.
3. Кроссплатформенную совместимость.
4. Удобство разработки и развертывания.

3.3 Интерфейс пользователя

Пользовательский интерфейс представляет собой ключевой компонент системы, обеспечивающий эффективное взаимодействие между пользователем и функциональными модулями. Его основное назначение - визуализация данных о строительных материалах и предоставление инструментов для их анализа [1].

Интерфейс реализован как одностраничное веб-приложение (*SPA*) с использованием современных технологий фронтенд-разработки. Архитектура интерфейса построена по принципу разделения на логические модули.

Интерфейс оптимизирован для работы с большими объемами данных и обеспечивает:

1. Быструю загрузку и отображение информации.
2. Минимальное время отклика на действия пользователя.
3. Адаптивность под различные устройства и разрешения экранов.
4. Доступность для пользователей с ограниченными возможностями.

Особое внимание уделено удобству работы и снижению когнитивной нагрузки. Для этого реализованы:

1. Единые паттерны взаимодействия.
2. Контекстные подсказки.
3. Прогрессивное раскрытие сложных функций.
4. Визуальные индикаторы состояния системы.

Интерфейс интегрирован с *backend*-системой через *REST* *API*, что обеспечивает стабильную работу и быстрый обмен данными. Все пользовательские действия логируются для последующего анализа и улучшения *UX* [17].

Стартовая страница системы представлена на рисунке 14.

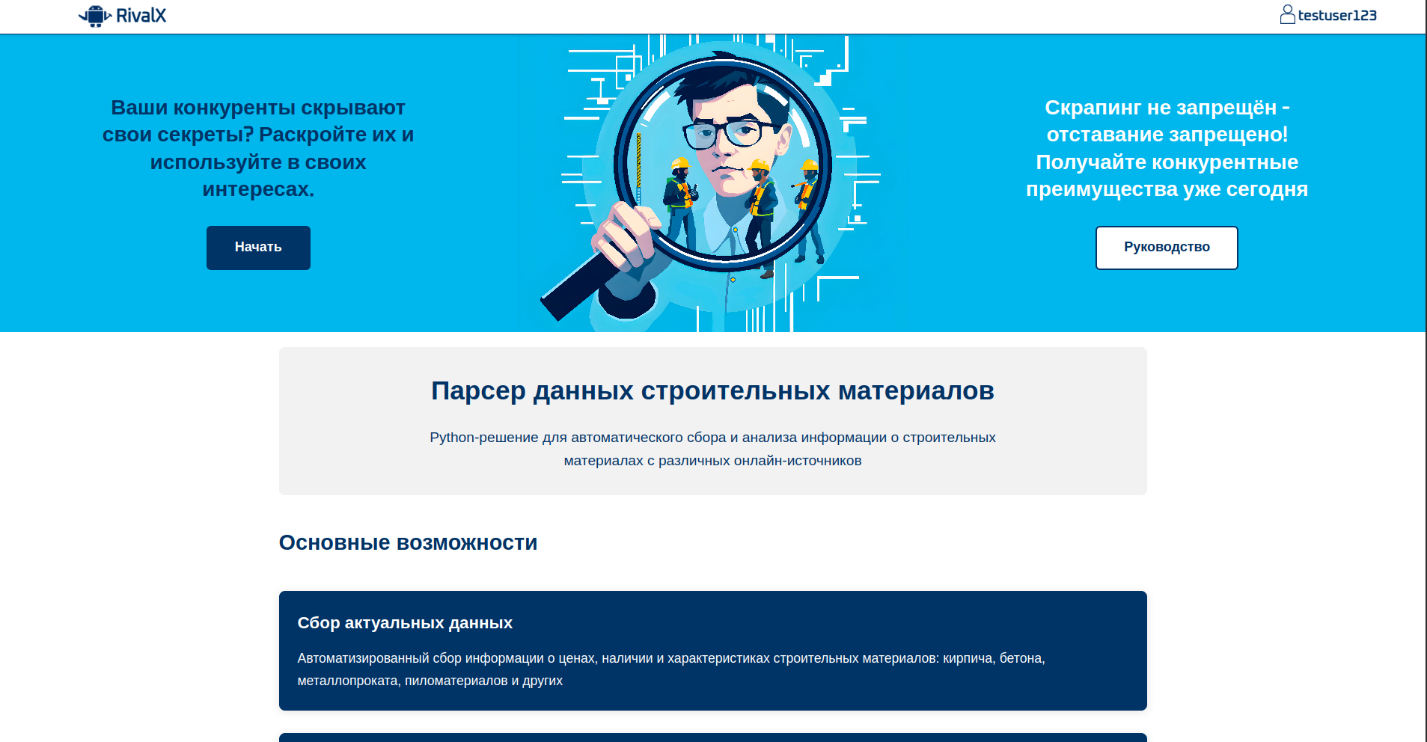


Рисунок 14 – Стартовая страница

Стартовая страница приложения представляет собой минималистичный интерфейс первичного взаимодействия, выполняющий навигационные и информационные функции. В визуальном центре страницы расположен логотип системы с сопровождающим слоганом, кратко отражающим основное назначение программного комплекса.

Композиция страницы организована вокруг трех ключевых интерактивных элементов:

1. Кнопка «Начать работу» - обеспечивает переход к основному функциональному модулю системы.
2. Кнопка «Руководство пользователя» - открывает документацию по работе с приложением.
3. Кнопка «Вход/Регистрация» - предоставляет доступ к модулю аутентификации.

Навигационная модель страницы построена по принципу минимального необходимого действия - пользователь может выполнить только целевые действия, для которых предусмотрены соответствующие элементы управления. Это позволяет сократить когнитивную нагрузку на этапе первичного знакомства с системой.

Техническая реализация стартовой страницы выполнена с использованием компонентного подхода, что обеспечивает:

1. Высокую скорость загрузки.
2. Адаптивность под различные устройства.
3. Легкость последующего масштабирования.
4. Простоту локализации интерфейса.

Продолжение стартовой страницы представлено на рисунке 15.

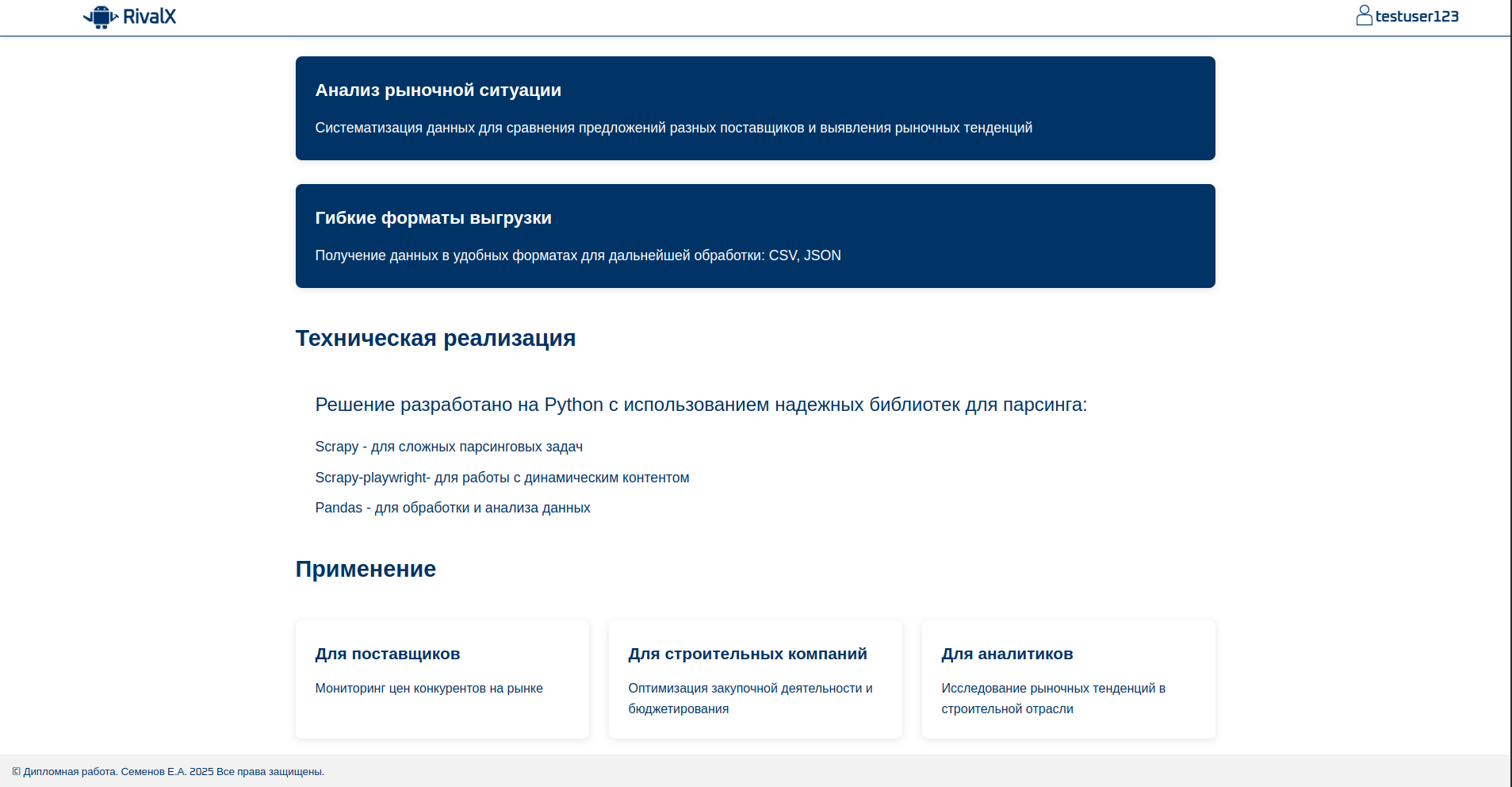


Рисунок 15 – Продолжение стартовой страницы

На данной странице указаны основные функциональные возможности системы:

Анализ рыночной ситуации

Система предоставляет комплексный инструментарий для систематизации данных о строительных материалах, позволяющий:

1. Проводить сравнительный анализ предложений различных поставщиков.
2. Выявлять текущие рыночные тенденции.
3. Отслеживать динамику ценовых изменений.

Форматы выгрузки данных

Для обеспечения удобства дальнейшей обработки реализована возможность экспорта данных в следующих форматах:

1. *CSV* - для работы с табличными процессорами.
2. *JSON* - для интеграции с веб-приложениями.

Техническая реализация

В основе решения лежит технологический стек:

1. Язык программирования *Python*
2. Специализированные библиотеки:
   1. *Scrapy* - для обработки сложных парсинговых задач;
   2. *Scrapy*-*playwright* - для работы с динамическим контентом;
   3. *Pandas* - для анализа и обработки данных.

Система находит применение у различных категорий пользователей, представлено в таблице 2.

Таблица 2 – Целевые группы и основные преимущества

|  |  |
| --- | --- |
| Целевая группа | Основные преимущества |
| Поставщики строительных материалов | Мониторинг ценовой политики конкурентов |
| Строительные компании | Оптимизация закупочной деятельности и бюджетирования |
| Аналитики рынка | Исследование тенденций строительной отрасли |

Блок регистрации представлена на рисунке 16.

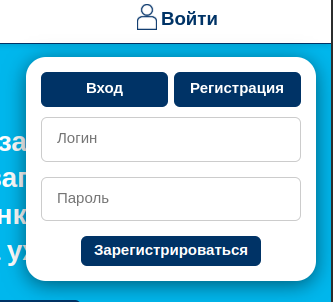


Рисунок 16 – Блок регистрации

Интерфейс регистрации представляет собой функциональный блок, обеспечивающий создание новых учетных записей. Форма регистрации содержит минимально необходимый набор полей для идентификации пользователя: поле ввода логина и поле ввода пароля. Под полями расположена кнопка подтверждения регистрации.

Логическое поведение интерфейса реализует следующий сценарий работы:

1. Пользователь вводит уникальный логин в соответствующее поле.
2. В поле пароля указывает секретную комбинацию символов.
3. Нажимает кнопку «Зарегистрироваться».

При обработке запроса система выполняет:

1. Проверку уникальности логина в базе данных.
2. Валидацию сложности пароля (минимальная длина, использование разных типов символов).
3. Шифрование учетных данных перед сохранением.

В случае успешной проверки:

1. Создается новая учетная запись.
2. Пользователь автоматически аутентифицируется.
3. Происходит перенаправление в личный кабинет.

При возникновении ошибок:

1. Под соответствующим полем появляется текстовое сообщение.
2. Форма сохраняет введенные данные (кроме пароля).
3. Кнопка регистрации временно блокируется.

Интерфейс интегрирован с системой восстановления пароля и модулем авторизации, образуя единый цикл работы с учетными данными. Техническая реализация обеспечивает безопасную передачу данных по защищенному соединению.

Блок входа представлен на рисунке 17.

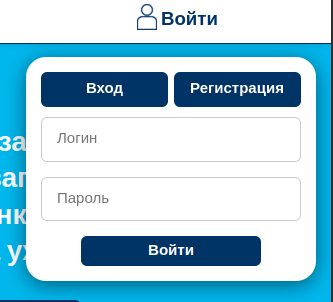


Рисунок 17 – Блок входа

Интерфейс аутентификации реализован как стандартный механизм входа в систему и состоит из двух основных элементов ввода:

1. Поле для указания логина:
   1. Принимает уникальный идентификатор пользователя.
   2. Проверяет соответствие формату данных.
   3. Обеспечивает автодополнение для зарегистрированных пользователей.
2. Поле для ввода пароля:
   1. Поддерживает скрытие вводимых символов.
   2. Включает функцию временного отображения пароля.
   3. Проверяет минимальные требования к длине.

Процесс авторизации выполняется по следующему алгоритму:

1. Пользователь вводит учетные данные в соответствующие поля.
2. Система проверяет наличие логина в базе данных.
3. Происходит сравнение хэшированного пароля с хранимой версией.
4. При успешной проверке:
   1. Создается сессия пользователя.
   2. Генерируется токен доступа.
   3. Выполняется перенаправление в личный кабинет.
5. При неудачной попытке:
   1. Выводится обобщенное сообщение об ошибке.
   2. Предлагается восстановление пароля.
   3. Сохраняется введенный логин (пароль очищается).

Интерфейс обеспечивает интеграцию с системой безопасности, включая:

1. Защиту от *brute*-*force* атак.
2. Ограничение количества попыток входа.
3. Ведение журнала авторизаций.
4. Блокировку учетной записи при подозрительной активности.

Блок выхода из учетной записи представлен на рисунке 18.

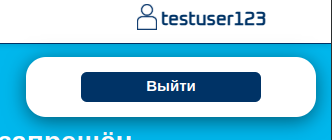


Рисунок 18 – Блок выхода из учетной записи

Интерфейс выхода из системы реализован в виде единого управляющего элемента – кнопки «Выйти», расположенной в навигационной панели приложения. Функциональность модуля обеспечивает:

1. Завершение текущей пользовательской сессии.
2. Очистку временных данных аутентификации.
3. Перенаправление на стартовую страницу системы.

Принцип работы:

1. При нажатии кнопки система выполняет синхронный запрос на сервер.
2. Серверная часть инвалидирует токен доступа.
3. Клиентское приложение очищает локальное хранилище сессии.
4. Происходит переход на страницу входа в систему.

Особенности реализации:

1. Кнопка доступна только для авторизованных пользователей.
2. Позиционируется в зоне постоянной видимости интерфейса.
3. Имеет визуальное отличие от других элементов управления.
4. Подтверждает действие через модальное окно при работе с важными данными.

Технические аспекты:

1. Безопасное завершение сессии:
   1. Удаление *JWT*-токена;
   2. Очистка *cookie* аутентификации;
   3. Закрытие *WebSocket*-соединений.
2. Обработка состояния:
   1. Сброс персональных настроек интерфейса;
   2. Сохранение временных данных;
   3. Обновление состояния приложения.
3. Интеграция с системой:
   1. Логирование события выхода;
   2. Отправка уведомлений;
   3. Синхронизация состояния между вкладками браузера.

После успешного выхода система возвращается в исходное состояние, сохраняя только общедоступную информацию. Повторная авторизация требует полного ввода учетных данных.

Страница руководство пользователя представлено на рисунке 19.



Рисунок 19 – Страница руководство пользователя

Руководство пользователя представляет собой детализированную инструкцию по работе с функциональными модулями системы, содержащую конкретные примеры взаимодействия. Интерфейс руководства организован в виде навигационного меню с переключением между тематическими блоками.

1. Модуль общей информации о конкретных возможностях системы
2. Модуль веб-скрейпинга:
   1. Правила формирования запросов: пример входных данных: «цемент М500, Москва, 50кг»; пример выходных данных: *JSON*-массив с ценами от 12 поставщиков;
   2. Ограничения парсинга: максимальная частота запросов - 1 запрос/5 секунд. Запрещенные домены: *marketplace*.*ru*, *buildlist*.*org.*
3. Модуль анализа данных:
   1. Построение диаграмм:

Входные параметры: «кирпич красный, 2024-01-01:2024-06-01»  
Выходные данные: *SVG*-график динамики цен с метками поставщиков.

* 1. Правила фильтрации:

Поддерживаемые операторы: «>», «<», «==», «!=»  
Пример запроса: «цена < 500 *AND* поставщик != 'СтройОпт'».

1. Модуль поиска:
   1. Синтаксис запросов:

Обязательные поля: категория, регион.

Пример валидного запроса: «утеплитель:пеноплекс; регион:СПб».

* 1. Форматы вывода:

Табличное представление (по умолчанию).

1. Система навигации:
   1. Горизонтальное меню переключения между разделами руководства;
   2. Контекстные ссылки на смежные темы;
   3. Поиск по терминам внутри руководства.

Каждый раздел содержит:

1. Точные примеры входных/выходных данных.
2. Скриншоты интерфейса с пояснениями.
3. Коды ошибок и способы их устранения.
4. Рекомендации по оптимизации запросов.

Страница веб-скрейпинга представлена на рисунке 20.

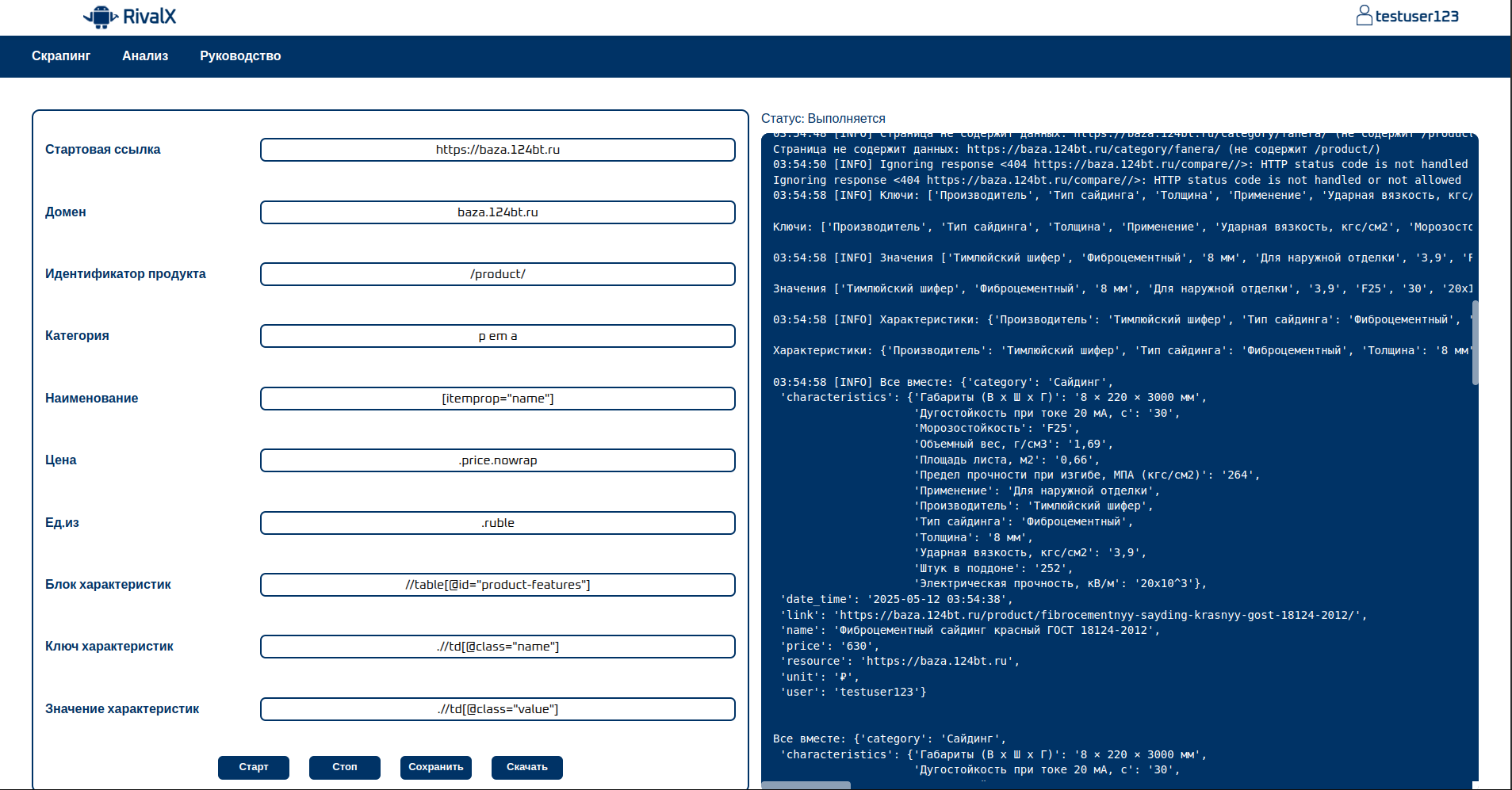


Рисунок 20 – Страница веб-скрейпинга

Страница веб-скрейпинга представляет собой специализированный интерфейс для настройки и управления процессом сбора данных. Интерфейс разделен на две основные секции: панель параметров парсинга и область вывода результатов.

Поля ввода параметров:

1. Стартовая ссылка - *URL*-адрес начальной страницы для анализа (например: «*https*://*baza*.124*bt*.*ru*»).
2. Домен - разрешенные домены для парсинга («*baza*.124*bt*.*ru*»).
3. Идентификатор продукта - путь к страницам товаров («/*product*/»).
4. Категория - *CSS*/*XPath* селектор для извлечения категории товара.
5. Наименование - селектор для названия товара.
6. Цена - селектор для ценовой информации.
7. Ед.из - селектор для единиц измерения.
8. Блок характеристик - селектор контейнера характеристик.
9. Ключ характеристик - селектор названий параметров.
10. Значение характеристик - селектор значений параметров.

Управляющие элементы:

1. Кнопка «Старт» - инициирует процесс парсинга.
2. Кнопка «Стоп» - прерывает выполнение скрейпинга.
3. Кнопка «Сохранить» - записывает результаты в базу данных.
4. Кнопка «Скачать» - экспортирует данные в *JSON*-файл.

Секция вывода информации:

1. Индикатор статуса выполнения:
   1. «Выполняется» (активный процесс);
   2. «Не выполняется» (процесс остановлен).
2. Область логов - отображает:
   1. Ход выполнения операций;
   2. Ошибки и предупреждения;
   3. Результаты сохранения данных.
3. Системные уведомления:
   1. Подтверждение успешных операций;
   2. Сообщения об ошибках;
   3. Предупреждения о некорректных данных.

Пример рабочего процесса:

1. Пользователь заполняет все обязательные поля:
   1. Стартовая ссылка: «*https*://*stroymaterialy*.*ru*/*catalog*/*keramzit*»;
   2. Домен: «*stroymaterialy*.*ru*»;
   3. Селекторы цен: «*product*-*price\_\_value*».
2. Нажимает «Старт» → статус меняется на «Выполняется».
3. В логах отображается прогресс сканирования страниц.
4. После завершения:
   1. Данные сохраняются в БД («Сохранить»);
   2. Или экспортируются («Скачать»).

Ограничения и проверки:

1. Все поля должны быть заполнены перед запуском.
2. Одновременно может выполняться только один процесс.
3. Автоматическое обновление логов каждые 5 секунд.
4. Валидация формата входных *URL* и селекторов.

Интерфейс обеспечивает полный контроль над процессом сбора данных с возможностью оперативного вмешательства и мониторинга состояния выполнения задачи.

Страница поиска представлена на рисунке 21.



Рисунок 13 – Страница поиска

Модуль анализа предоставляет инструменты для работы с собранными данными о строительных материалах. Интерфейс состоит из двух основных секций: панель фильтрации и область отображения результатов.

Поля фильтрации данных:

1. Ресурс - источник данных (например: «*https*://*baza*.124*bt*.*ru*»).
2. Категория - тип строительного материала («Ступени железобетонные»).
3. Название - наименование конкретного товара («Ступень железобетонная»).
4. Период анализа:
   1. Начиная с (дата и время);
   2. Заканчивая (дата и время).

Управляющие элементы:

1. Кнопка «Найти» - выполняет поиск по заданным критериям.
2. Кнопка «Диаграмма» - визуализирует данные в графическом виде.
3. Кнопка «Скачать *CSV*» - экспортирует результаты в табличном формате.

Секция отображения результатов:

1. Режим таблицы:
   1. Карточки товаров с полной информацией;
   2. Форматирование характеристик в читаемый вид.
   3. Ссылки на оригинальные страницы товаров.
2. Режим диаграммы:
   1. Графическое представление динамики цен;
   2. Интерактивные элементы управления;
   3. Кнопка закрытия для возврата к табличному виду.
3. Состояния интерфейса:
   1. «Загрузка данных...» (процесс выполнения запроса);
   2. «Ничего не найдено» (пустой результат);
   3. Сообщения об ошибках.

Пример рабочего процесса:

1. Пользователь задает параметры:
   1. Категория: «Кирпич строительный»;
   2. Период: последние 30 дней.
2. Нажимает «Найти» → система загружает соответствующие записи.
3. Для анализа тенденций нажимает «Диаграмма».
4. При необходимости экспортирует данные («Скачать *CSV*»).

Форматы вывода данных:

1. Внутрисистемное представление:
   1. Структурированные карточки товаров;
   2. Группировка по категориям/поставщикам.
2. Экспортный формат:
   1. *CSV*-файл с полным набором полей;
   2. Кодировка *UTF*-8 с *BOM;*
   3. Разделитель - точка с запятой.

Особенности реализации:

1. Валидация временных интервалов.
2. Оптимизация запросов к базе данных.
3. Ленивая загрузка характеристик.
4. Адаптивное отображение для различных объемов данных.

Интерфейс обеспечивает комплексный анализ собранной информации с возможностью детализации и визуализации ключевых показателей.

Страница построения диаграммы представлена на рисунке 22.

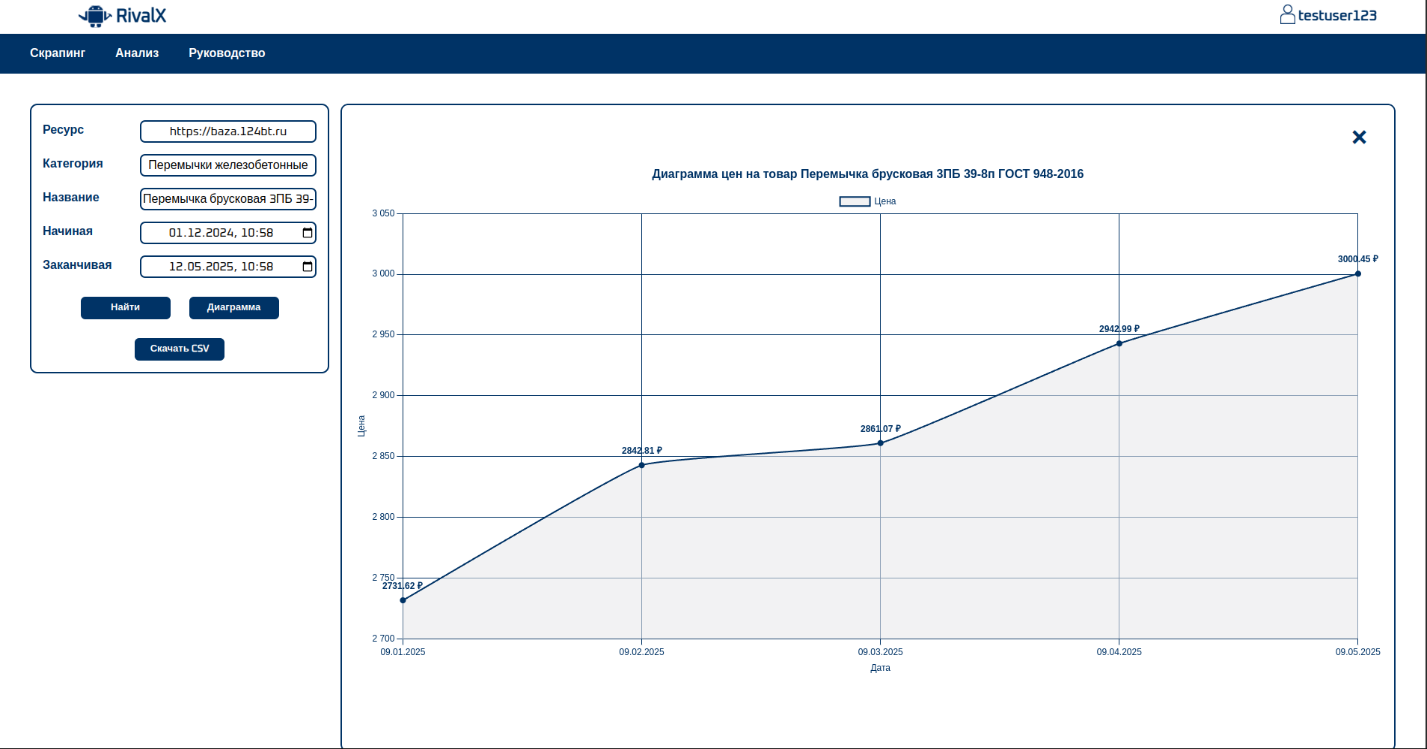


Рисунок 22 – Страница построения диаграммы

Модуль построения диаграмм обеспечивает графическое представление динамики цен на строительные материалы. Основной функционал реализует интерактивную линейную диаграмму, построенную с использованием библиотеки *Chart*.*js*.

Логика автоматического определения типа диаграммы:  
Система анализирует входные данные и выбирает оптимальный режим отображения:

1. Для одного товара (когда в данных присутствует единственное наименование) строится детальная диаграмма изменения цены с точными значениями в каждой точке.
2. Для категории товаров отображается агрегированная динамика средней цены.
3. Для данных с одного ресурса выводится сводный график по всем позициям.

Элементы управления диаграммой:

1. Автомасштабирование при изменении размеров контейнера.
2. Интерактивные подсказки при наведении на точки данных.
3. Отображение точных значений цен в каждой точке.
4. Настройки визуального представления:
   1. Цветовая схема (синяя гамма с прозрачной заливкой);
   2. Толщина и стиль линий;
   3. Формат отображения числовых значений.

Технические параметры визуализации:

1. Ось *X*:
   1. Отображает временную шкалу;
   2. Автоматическое форматирование дат;
   3. Равномерное распределение точек.
2. Ось *Y*:
   1. Отображает ценовую шкалу в рублях;
   2. Автоматическое определение диапазона;
   3. Поддержка дробных значений.
3. Линия тренда:
   1. Сглаживание углов (*tension*: 0.1);
   2. Точечные маркеры ключевых значений;
   3. Заливка области под кривой.

Обработка ошибок и особых случаев:

1. При отсутствии данных выводится соответствующее сообщение.
2. При неоднородных данных (разные ресурсы/категории) система предлагает уточнить параметры поиска.
3. Контроль качества входных данных:
   1. Проверка на *null*/*undefined;*
   2. Валидация формата дат;
   3. Обработка отсутствующих ценовых значений.

Пример использования:

1. Пользователь выбирает конкретный товар («Кирпич М-150»).
2. Система строит график изменения цены за выбранный период.
3. На диаграмме отображаются:
   1. Точные значения цен в ключевые даты;
   2. Общий тренд (рост/падение);
   3. Возможные аномалии ценообразования.

Техническая реализация:

1. Основана на библиотеке *Chart*.*js* с плагином *DataLabels.*
2. Использует *Canvas* *API* для отрисовки.
3. Поддерживает *responsive*-дизайн.
4. Оптимизирована для работы с большими массивами данных.

Диаграмма интегрирована с основным интерфейсом анализа данных и обновляется автоматически при изменении параметров поиска или фильтрации.

3.4 Дизайн

Система реализует концепцию естественного пользовательского потока, где каждый элемент интерфейса ведет себя предсказуемо в соответствии с ментальными моделями целевой аудитории [19]. Ключевые аспекты:

1. Визуальная иерархия.

Композиция строится по закону Фиттса – важные элементы (кнопки действий, формы ввода) занимают доминирующее положение и имеют увеличенную область клика. Размер, цвет и расположение компонентов отражают их приоритетность.

1. Консистентность поведения.

Все интерактивные элементы используют единые паттерны:

* 1. Кнопки реагируют на *hover* одинаковым визуальным эффектом (свечение + масштабирование 1.05*x*).
  2. Формы подсвечивают активное поле синей рамкой.
  3. Навигационные элементы сохраняют стиль во всех разделах.

1. Контекстная помощь.

Поля ввода содержат примеры заполнения (*placeholder*), а сложные элементы (селекторы парсинга) дополнены иконками-подсказками. Система выводит пояснения при обнаружении нестандартных действий.

Визуальная гармония

1. Цветовая психология.

Сочетание глубокого синего (#003366) с нейтральным серым создает:

* 1. Чувство надежности и профессионализма.
  2. Оптимальный контраст для длительной работы (*WCAG* *AA*).
  3. Эмоциональную нейтральность, не отвлекающую от данных.

1. Типографический ритм

Шрифт *Oxanium* с его геометрической четкостью обеспечивает:

* 1. Быстрое распознавание текста даже в плотных таблицах.
  2. Хорошую различимость цифр в ценовых показателях.
  3. Визуальный комфорт при чтении длинных характеристик.

1. Пространственный баланс

Применены принципы воздушности:

* 1. Межстрочные интервалы 1.6*em* для текста.
  2. Минимальные внутренние отступы 15*px* в карточках.
  3. Продуманные *margin* между блоками (30*px* по вертикали).

Тактильная обратная связь

Интерактивные элементы дают мгновенный отклик:

1. Кнопки «Старт/Стоп» парсинга меняют цвет и положение (*transform*: *translateY*).
2. Чекбоксы анимируют переключение за 0.15*s.*
3. Скроллбар кастомизирован под общую стилистику.
4. *Drag*-*and*-*drop* операции визуализируют процесс перемещения.

Когнитивная разгрузка

1. Прогрессивное раскрытие.

Сложные функции (настройки парсинга) скрыты под аккордеонами, показываясь только при необходимости. Основной рабочий поток содержит не более 7±2 элементов одновременно.

1. Визуальные якоря.

Постоянно видимые элементы (хедер, кнопка выхода) используют «эффект глубины» через *box*-*shadow*, помогая пользователю ориентироваться.

1. Естественные метафоры.

Диаграммы имитируют графики из профессиональных строительных отчетов, а таблицы следуют стандартам *Excel*, сокращая время адаптации.

Эстетика восприятия

1. Золотое сечение.

Соотношение боковой панели (30%) к рабочей области (70%) соответствует пропорции 1:1.618.

1. Динамический контраст.

Темные элементы интерфейса (#003366) сочетаются не с чисто белым, а с теплым серым (*rgba* (233, 233, 235)), снижая нагрузку на зрение.

1. Органическая анимация

Все переходы используют кривые Безье (*cubic*-*bezier* (0.4, 0, 0.2, 1)), имитируя естественное движение.

Такое сочетание функциональности и визуальной гармонии создает эффект «невидимого интерфейса» - пользователь интуитивно понимает систему, сосредотачиваясь на данных, а не на инструментах их обработки [1].

3.5 Серверная часть системы

Серверное приложение реализовано на *Flask* с использованием *RESTful*-подхода. Основная конфигурация и инициализация компонентов происходит в главном модуле приложения [13].

Ядро системы:

1. *Flask*-приложение

Создается через фабричную функцию *create\_app*(), что обеспечивает:

* 1. Гибкость конфигурации.
  2. Возможность тестирования.
  3. Модульность структуры.

1. Безопасность.

Реализована через:

* 1. *JWT*-аутентификацию (*Flask*-*JWT*-*Extended*).
  2. Защиту *CORS* (*Flask*-*CORS*).
  3. Хранение чувствительных данных в .*env.*

1. Работа с данными.

Используется *SQLAlchemy* *ORM* для взаимодействия с *MySQL*:

* 1. Поддержка пула соединений.
  2. Автоматическое управление сессиями.
  3. Миграции через *Alembic.*

Конфигурационные параметры:

1. Токены доступа действительны 7 дней.
2. Авторизация через *Bearer*-токены в заголовках.
3. Подключение к БД через параметры окружения.
4. Гибкие настройки хоста/порта для развертывания.

Модульная структура:

Система организована через *Flask* *Blueprints*:

1. *scrapy\_bp* – управление парсингом данных.
2. *auth\_bp* – аутентификация пользователей.
3. *file\_bp* – работа с файлами.
4. *savedb\_bp* – сохранение данных.
5. *products\_bp* – управление продуктами.

Особенности запуска:

1. Режим отладки контролируется через переменные окружения.
2. Поддержка *Docker*-развертывания (0.0.0.0).
3. Порт по умолчанию 5000 с возможностью переопределения.

Архитектура обеспечивает:

1. Горизонтальное масштабирование.
2. Простое добавление новых модулей.
3. Безопасное *API* для клиентских приложений.
4. Интеграцию с внешними сервисами.

Система использует *RESTful* *API* с четкой структурой запросов и ответов. Основные типы взаимодействия:

Авторизационные запросы

1. Метод: *POST* /*auth*/*login.*
2. Тело: {*username*: *string*, *password*: *string*}.
3. Ответ: *JWT*-токен с данными пользователя.
4. Особенности: Токен передается в заголовке *Authorization*: *Bearer* <*token*>.

Парсинг данных

1. Запуск парсера.
   1. Метод: *POST* /*scrapy*/*start.*
   2. Тело: Параметры парсинга (селекторы, *URL*).
   3. Ответ: *ID* задачи и статус запуска.
2. Остановка парсера.
   1. Метод: *POST* /*scrapy*/*stop.*
   2. Тело: {*user\_id*: *string*}.
   3. Ответ: Логи завершения.

Работа с данными

1. Получение товаров.
   1. Метод: *GET* /*products.*
   2. Параметры: Фильтры (категория, период).
   3. Ответ: *JSON*-массив товаров.
2. Сохранение в БД.
   1. Метод: *POST* /*savedb.*
   2. Тело: {*username*: *string*}.
   3. Ответ: Статистика сохранения.

Файловые операции

1. Скачивание данных.
   1. Метод: *GET* /*files*/<*username*>.
   2. Ответ: *JSON*-файл с данными.
   3. Заголовки: *Content*-*Disposition*: *attachment.*

Особенности реализации:

1. Коды состояния:
   1. 200 - Успешные операции.
   2. 401 - Ошибка авторизации.
   3. 400 - Некорректные данные.
   4. 500 - Серверные ошибки.
2. Форматы данных:
   1. Входные данные: *JSON.*
   2. Выходные данные: *JSON*/*CSV.*
   3. Кодировка: *UTF*-8.
3. Заголовки:
   1. Обязательные: *Content*-*Type*: *application*/*json.*
   2. Опциональные: *Authorization* для защищенных роутов.

Пример типичного запроса:

*POST* /*scrapy*/*start* *HTTP*/1.1

*Authorization*: *Bearer* *eyJhbGciOiJIUzI*1*NiIsInR*5*cCI*6*IkpXVCJ*9...

*Content*-*Type*: *application*/*json*

{

«*start\_url*»: «*https*://*example*.*com*»,

«*allowed\_domains*»: «*example*.*com*»,

«*product\_path*»: «/*products*/»

}

3.6 Описание контроллеров

Контроллеры представляют собой ключевой компонент серверной части приложения, выполняющий функции посредника между клиентскими запросами и бизнес-логикой системы [14]. В архитектуре *MVC* (*Model*-*View*-*Controller*), используемой в *Flask*-приложении, контроллеры отвечают за:

1. Обработку *HTTP*-запросов.

Принимают входящие запросы от клиента, валидируют параметры и извлекают необходимые данные из:

* 1. Тела запроса (*JSON*, *form*-*data*).
  2. *URL*-параметров.
  3. Заголовков.

1. Оркестрацию процессов.

Координируют взаимодействие между:

* 1. Моделями данных (работа с БД).
  2. Сервисным слоем (бизнес-логика).
  3. Внешними *API* (парсинг, интеграции).

1. Формирование ответов.

Преобразуют результаты обработки в клиентские форматы:

* 1. *JSON* для *REST* *API.*
  2. Файловые потоки (*CSV*, *Excel*).
  3. *HTTP*-коды состояния.

Ключевые характеристики контроллеров в системе:

1. Модульность.  
   Каждый контроллер инкапсулирует логику для конкретной предметной области:
   1. Аутентификация (*auth\_controller*).
   2. Парсинг данных (*scrapy\_controller*).
   3. Управление продуктами (*products\_controller*).
2. Минималистичная бизнес-логика.

Контроллеры делегируют сложные вычисления сервисному слою, сосредотачиваясь на:

* 1. Валидации входных данных.
  2. Обработке ошибок.
  3. Форматировании ответов.

1. Согласованность интерфейсов.

Все контроллеры следуют единым стандартам:

* 1. Именование роутов (*RESTful*-подход).
  2. Форматы ответов.
  3. Обработка исключений.

1. Безопасность.

Интеграция с *JWT*-аутентификацией через:

* 1. Декораторы @*jwt required*().
  2. Проверку ролей пользователей.
  3. Валидацию токенов доступа.

Пример типичного жизненного цикла запроса в контроллере:

1. Получение *HTTP*-запроса.
2. Извлечение и валидация параметров.
3. Вызов соответствующих сервисов.
4. Обработка возможных ошибок.
5. Формирование структурированного ответа.
6. Логирование операции.

Контроллеры в данном приложении организованы как *Flask* *Blueprints*, что обеспечивает:

1. Гибкость маршрутизации.
2. Возможность изолированного тестирования.
3. Простоту масштабирования функционала.
4. Четкое разделение зон ответственности.

Каждый контроллер представляет собой самостоятельный модуль, связанный с другими компонентами системы через четко определенные интерфейсы, что соответствует принципам *SOLID* и облегчает поддержку кода.

*Auth* *Controller* (аутентификация):

1. Назначение.

Управление процессами регистрации, входа и проверки подлинности пользователей.

1. Ключевые функции.
   1. /*register* — регистрация новых пользователей с валидацией: проверка минимальной длины логина (4 символа), требование сложности пароля (6+ символов), защита от дубликатов имен пользователей.
   2. /*login* — аутентификация с выдачей *JWT*-токена: срок действия 7 дней, *Bearer*-схема авторизации, логирование успешных и неудачных попыток.
   3. /*protected* — тестовый защищенный эндпоинт.
   4. /*validate*-*token* — проверка валидности токена.
2. Особенности.

Хеширование паролей через *werkzeug*.*security*, подробное логирование всех операций, обработка *SQL*-инъекций через *ORM*.

*File* *Controller* (работа с файлами):

1. Назначение.

Управление файлами результатов парсинга.

1. Ключевые функции.
   1. /*files*/<*username*> — скачивание *JSON*-файла: поиск файла в структуре проекта, проверка существования файла, отправка как *attachment* с правильными заголовками.
2. Особенности.  
   Жесткая структура путей к файлам, поддержка кириллических имен, подробное логирование ошибок.

*Products* *Controller* (работа с товарами):

1. Назначение.

Управление данными о строительных материалах.

1. Ключевые функции
   1. /*products* — фильтрация товаров по параметрам: обязательный параметр *user*, фильтры по ресурсу, категории и названию, временные диапазоны (с валидацией формата), поддержка частичного совпадения (*ilike*).
2. Особенности.

Оптимизированные *SQL*-запросы, корректная обработка *NULL*-значений, поддержка кириллицы в *JSON*.

*Savedb* *Controller* (сохранение данных):

1. Назначение.

Импорт данных парсинга в базу данных.

1. Ключевые функции.
   1. /*savedb*/<*username*> — сохранение *JSON* в базу: проверка дубликатов по 4 полям, автопарсинг дат разных форматов, подробная статистика операций.
2. Особенности.

Транзакционная обработка, детальное логирование каждого товара, счётчик пропущенных дубликатов.

*Scrapy* *Controller* (управление парсингом):

1. Назначение.

Оркестрация процессов веб-скрейпинга.

1. Ключевые функции.
   1. /*start*-*scrapy* — запуск парсера: передача параметров через командную строку, защита от экранирования (*shlex*.*quote*), временные файлы для логов.
   2. /*stop*-*scrapy* — остановка процесса: корректное завершение через *SIGTERM*, очистка ресурсов.
   3. /*logs*/<*user\_id*> — мониторинг выполнения: режим реального времени, статус выполнения.
2. Особенности.

Управление процессами на уровне ОС, авто очистка временных файлов.

Общие принципы реализации:

1. Обработка ошибок.

Единый формат ответов об ошибках, *try*-*catch* на всех уровнях, корректные *HTTP*-коды состояния.

1. Логирование.

Подробные журналы операций, разные уровни логирования (*DEBUG*, *ERROR*), контекст ошибок (с использованием *exc\_info*).

1. Безопасность.

Валидация всех входных данных, экранирование параметров, ограничение времени жизни токенов.

1. Производительность.

Оптимизированные *SQL*-запросы, пакетная обработка данных, минимизация блокировок базы данных [2].

3.7 Описание моделей

Система использует реляционную модель данных с двумя основными сущностями: пользователи и товары. Структура базы реализована в *MySQL* с четким разделением ответственности между таблицами [3].

Таблица пользователей хранит учетные данные и метаинформацию о зарегистрированных пользователях системы. Основное внимание уделено безопасности - пароли хранятся в хешированном виде с использованием современных алгоритмов шифрования. Для каждого пользователя фиксируется точное время создания учетной записи.

Таблица товаров содержит детальную информацию о строительных материалах, собранную в результате парсинга. Особенностью структуры является использование *JSON*-поля для хранения характеристик товаров, что позволяет сохранять разнородные данные без жесткой схемы. Все записи привязываются к конкретному пользователю через внешний ключ.

Связи между таблицами организованы по принципу «один-ко-многим» - один пользователь может иметь множество товаров в базе. При этом реализован механизм каскадного удаления, автоматически очищающий все связанные данные при удалении учетной записи.

Система типов данных подобрана для оптимального хранения информации:

1. Точные числовые значения цен (*DECIMAL*).
2. Гибкие текстовые поля для названий и категорий.
3. Специализированные типы для временных меток.
4. Поддержка *JSON* для сложноструктурированных данных.

Индексация организована таким образом, чтобы обеспечить высокую производительность при частых операциях поиска и фильтрации. Особое внимание уделено целостности данных через систему ограничений и внешних ключей.

Модель *users* (Пользователи) представлена в таблице 3.

Таблица 3 – Пользователи

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Обязательное | Описание |
| *id* | *INT* | Да | Первичный ключ, автоинкремент |
| *username* | *VARCHAR*(50) | Да | Уникальное имя пользователя |
| *password\_hash* | *VARCHAR*(255) | Да | Хешированный пароль |
| *created\_at* | *TIMESTAMP* | Да | Дата регистрации (автоматически) |

Функции модели *User*:

1. Создание и проверка хеша пароля.
2. Валидация данных пользователя.
3. Связь с таблицей товаров (один ко многим).

Модель *products* (Товары) представлена в таблице 3.

Таблица 4 – Товары

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Обязательное | Описание |
| *id* | *INT* | Да | Первичный ключ, автоинкремент |
| *user* | *VARCHAR*(50) | Да | Внешний ключ к *users*.*username* |
| *category* | *TEXT* | Нет | Категория товара |
| *name* | *TEXT* | Да | Наименование товара |
| *price* | *DECIMAL*(10,2) | Нет | Цена товара |
| *unit* | *TEXT* | Нет | Единица измерения |
| *characteristics* | *JSON* | Нет | Характеристики в *JSON*-формате |
| *link* | *TEXT* | Нет | Ссылка на страницу товара |
| *resource* | *TEXT* | Нет | Источник данных |
| *date\_time* | *DATETIME* | Нет | Дата сбора информации |
| *created\_at* | *TIMESTAMP* | Да | Дата создания записи (автоматически) |

Особенности:

1. Поддержка сложных структур данных через *JSON*-поле.
2. Связь с таблицей пользователей.
3. Оптимизирована для частых запросов по фильтрам.

Связи между таблицами:

1. Один пользователь → множество товаров.
2. Автоматическое удаление связанных товаров при удалении пользователя.

Меры безопасности:

1. Хранение только хешей паролей.
2. Проверка уникальности логинов.
3. Ограничения на обязательные поля.
4. Защита от *SQL*-инъекций через *ORM.*

Производительность:

1. Индексы по ключевым полям.
2. Оптимальные типы данных для каждого поля.
3. Поддержка транзакций при массовых операциях.

3.8 Алгоритм получения данных

Алгоритм получения данных представлен на рисунке 23.

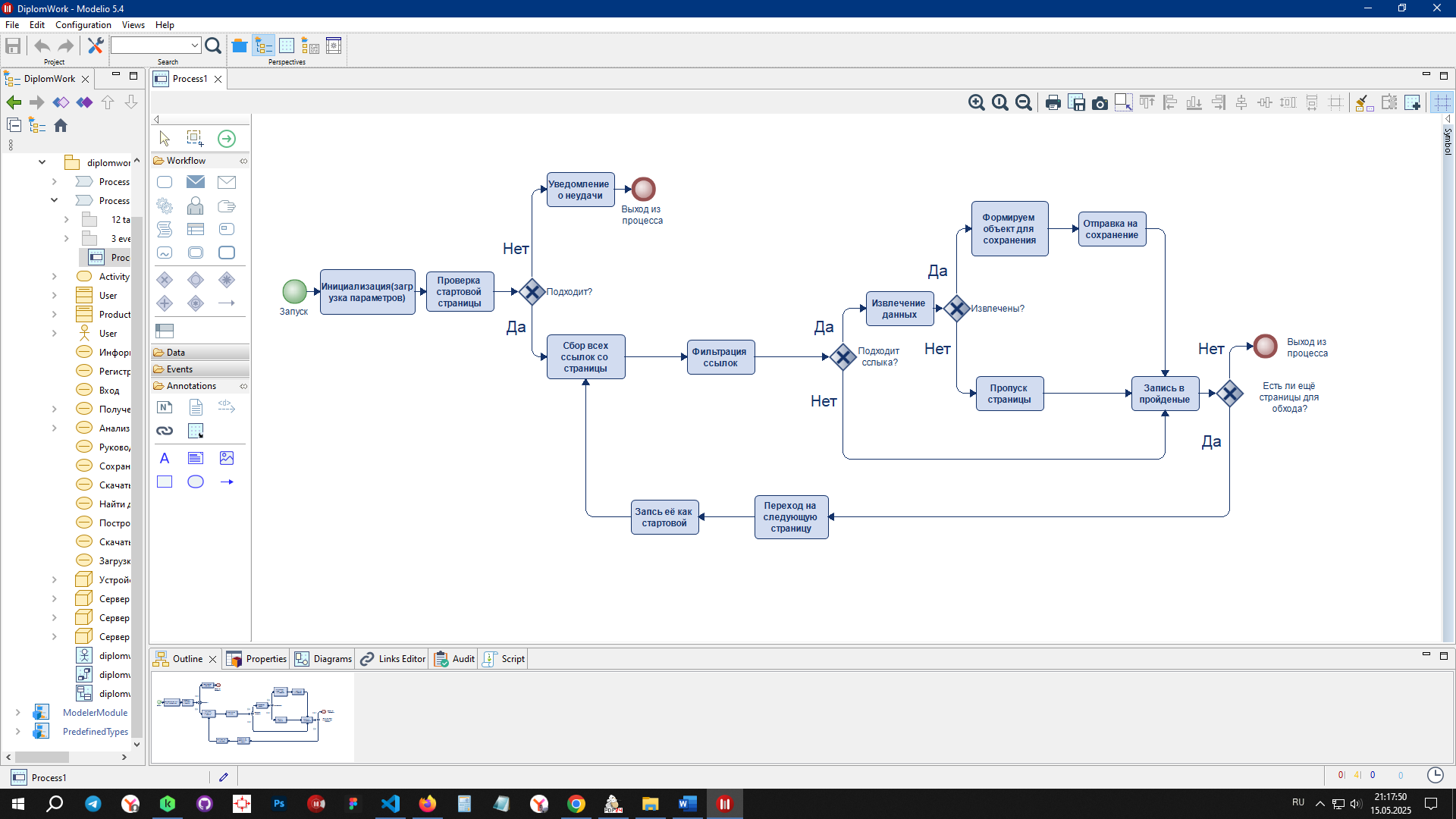


Рисунок 23 – Алгоритм получения данных

«Паук» представляет собой специализированный парсер, разработанный для систематического сбора данных о строительных материалах с целевых сайтов. Основной процесс организован пошагово с четким разделением этапов обработки.

Инициализация и настройка

При создании экземпляра паука происходит загрузка параметров парсинга, включая стартовый *URL*, доменные ограничения и набор *CSS*/*XPath*-селекторов для извлечения данных. Особое внимание уделяется настройкам производительности – ограничению количества параллельных запросов и установке паузы между ними, что обеспечивает корректную работу с сайтами без риска блокировки.

Процесс обхода

Система начинает работу с указанной стартовой страницы, анализирует все обнаруженные ссылки и фильтрует их по нескольким критериям. Исключаются служебные ссылки (телефоны, *email* и др.), проверяется принадлежность к целевому домену и путь к товарным страницам. Для каждого нового *URL* создается запрос с учетом уже посещенных адресов, что предотвращает дублирование работы.

Извлечение данных

На товарных страницах активируется специализированный обработчик, который с помощью предварительно заданных селекторов извлекает ключевую информацию: категорию товара, наименование, цену и единицы измерения. Особенностью реализации является двухэтапная обработка характеристик - сначала находится блок с параметрами, затем последовательно извлекаются пары «ключ-значение» с последующей нормализацией текста.

Формирование результатов

Все собранные данные структурируются в единый объект, дополняются метаинформацией (источник, временная метка) и передаются в конвейер обработки. В процессе работы ведется детальное логирование всех ключевых этапов, что позволяет отслеживать выполнение и оперативно выявлять возможные проблемы.

Алгоритм демонстрирует гибкость за счет параметризации селекторов и настроек обхода, сохраняя при этом высокую надежность благодаря многоуровневой системе проверок и обработке исключительных ситуаций. Это обеспечивает стабильный сбор данных даже при изменениях структуры целевых сайтов [6].

3.9 Входные данные

На этапе настройки парсинга пользователь через веб-интерфейс вводит параметры, определяющие, откуда именно брать данные на целевом сайте. Эти параметры включают в себя ссылки, селекторы элементов и другие настройки, которые система использует для автоматического сбора данных о товарах.

Пользователь заполняет форму в интерфейсе, после чего система запускает процесс веб-скрапинга с соответствующими аргументами.

Обязательные параметры:

1. Пользователь (*user*).
   1. Подставляется автоматически из кэша браузера после регистрации или авторизации.
   2. Пример: *user*1234.
2. Стартовая ссылка (*start\_url*).
   1. *URL* главной страницы сайта, с которой начинается обход.
   2. Пример: *https*://*baza*.124*bt*.*ru.*
3. Разрешённые домены (*allowed\_domains*).
   1. Список доменов, на которых разрешено сканирование.
   2. Пример: *baza*.124*bt*.*ru.*
4. Путь к карточке товара (*product\_path*)
   1. Относительный путь к страницам с карточками товаров.
   2. Пример: /*product*/

*CSS*- и *XPath*-селекторы

Чтобы указать, откуда брать конкретные данные (название, цену, единицу измерения и т.д.), необходимо определить селекторы элементов — это можно сделать прямо на сайте с помощью инструментов разработчика (*DevTools*) в браузере:

Как найти селекторы:

1. Открыть карточку товара на сайте.
2. Нажать правой кнопкой на нужный элемент (например, название товара).
3. Выбрать «Просмотреть код» или «*Inspect*».
4. Навести на элемент в *HTML* и:
   1. Копировать *CSS*-селектор: *Copy* → *Copy* *selector*
   2. Или *XPath*: *Copy* → *Copy* *XPath*

Примеры параметров:

1. Категория (*category\_selector*) — селектор элемента, содержащего название категории.
2. Название товара (*name\_selector*) — селектор для заголовка товара.
3. Цена (*price\_selector*) — элемент с ценой.
4. Единица измерения (*unit\_selector*) — текст типа «за м²», «за штуку».
5. Блок характеристик (*block\_selector*) — общий блок, содержащий таблицу с параметрами.
6. Ключи характеристик (*key\_selector*) — названия характеристик (например, «Производитель»).
7. Значения характеристик (*value\_selector*) — сами значения характеристик.

Примеры значений параметров представлены в таблице 5.

Таблица 5 – Примеры значений параметров

|  |  |
| --- | --- |
| Параметр | Значение |
| Пользователь | *user1234* (автозаполнение) |
| Стартовая ссылка | *https*://*baza*.124*bt*.*ru* |
| Допустимые домены | *baza*.124*bt*.*ru* |
| Путь к товарам | /*product*/ |
| Селектор категории | *p* *em* *a* |
| Селектор названия | *[itemprop*= «*name*»*]* |
| Селектор цены | *price*.*nowrap* |
| Селектор единицы | *ruble* |
| *XPath* блока данных | //*table[*@*id*= «*product*-*features*»*]* |
| *XPath* ключа | //*td[*@*class*= «*name*»*]* |
| *XPath* значения | //*td[*@*class*= «*value*»*]* |

Рекомендации по выбору селекторов:

1. *CSS*-селекторы удобны, если у элементов есть классы или атрибуты (.*class*, *[attr*=*value]*).
2. *XPath* стоит использовать, если структура сложная или нужно идти от родительского элемента.
3. Избегайте селекторов с динамическими *ID* или классами (*item*-3948237), они могут меняться.

3.10 Выходные данные

Система возвращает массив объектов в формате *JSON*, где каждый объект соответствует карточке товара на сайте. Данные структурированы для удобства хранения и анализа.

Структура выходного объекта представлена в таблице 6.

Таблица 6 – Структура выходного объекта

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Пример значения | Описание |
| *user* | *string* | «*testuser*123» | Логин пользователя, который запустил парсер. |
| *category* | *string* | «Сайдинг» | Категория товара. |
| *name* | *string* | «Фиброцементный сайдинг красный ГОСТ 18124-2012» | Название товара. |
| *price* | *string* | «630» | Цена товара. |
| *unit* | *string* | «₽» | Валюта или единица измерения. |
| *characteristics* | *object* | {«Производитель»: «Тимлюйский шифер», ...} | Технические параметры товара. |
| *link* | *string* | «*https*://*baza*.124*bt*.*ru*/*product*/*fibrocementnyy*-*sayding*-*krasnyy*-*gost*-18124-2012/» | Прямая ссылка на карточку товара. |
| *resource* | *string* | «*https*://*baza*.124*bt*.*ru*» | Домен сайта-источника. |
| *date\_time* | *string* | «2025-05-08 22:46:52» | Дата и время парсинга в формате *YYYY*-*MM*-*DD* *HH*:*MM*:*SS*. |

Подробное описание полей:

1. Основная информация о товаре.
   1. *User.*
      1. Используется для логирования (какой пользователь запустил парсер).
      2. Может быть полезно для многопользовательских систем.
   2. *Category* - определяет раздел, к которому относится товар (например, «Сайдинг», «Кирпич», «Краски»). Обычно берётся из «хлебных крошек» или заголовка страницы.
   3. *Name* - полное название товара, включая ключевые параметры (марку, ГОСТ, цвет). Пример» Фиброцементный сайдинг красный ГОСТ 18124-2012».
   4. *price* и *unit* - цена всегда возвращается в виде строки (чтобы избежать проблем с разными форматами чисел). *unit* может быть: ₽, $, €, шт., м² и т. д.
2. Характеристики (*characteristics*).

Это вложенный объект, где:

1. Ключ – название параметра (например, «Производитель», «Толщина»).
2. Значение – описание параметра (например, «Тимлюйский шифер», «8 мм»).

Пример структуры:

*json*

«*characteristics*»: {

«Производитель»: «Тимлюйский шифер»,

«Тип сайдинга»: «Фиброцементный»,

«Толщина»: «8 мм»,

«Применение»: «Для наружной отделки»,

«Габариты (В х Ш х Г)»: «8 × 220 × 3000 мм»

}

Какие характеристики чаще всего встречаются для стройматериалов?

1. Физические параметры:
   1. Вес, Размеры, Толщина, Площадь покрытия.
2. Технические свойства:
   1. Морозостойкость, Ударная вязкость, Предел прочности.
3. Производитель и стандарты:
   1. ГОСТ/ТУ, Страна-производитель, Бренд.
4. Упаковка:
   1. Количество в упаковке, Вес поддона.
5. Ссылка и источник (*link*, *resource*)
   1. *link* – прямая *URL*-ссылка на товар. Позволяет быстро перейти к нему для проверки.
   2. *resource* – домен сайта (например, *baza*.124*bt*.*ru*). Помогает идентифицировать источник данных.
6. Дата и время (*date\_time*)
   1. Формат: ГГГГ-ММ-ДД ЧЧ: ММ: СС (например, 2025-05-08 22:46:52).
   2. Нужно для:
      1. Отслеживания актуальности данных.
      2. Сравнения цен в динамике.

Пример выходного *JSON*.

*json*

*[*

{

«*user*»: «*testuser*123»,

«*category*»: «Сайдинг»,

«*name*»: «Фиброцементный сайдинг красный ГОСТ 18124-2012»,

«*price*»: «630»,

«*unit*»: «₽»,

«*characteristics*»: {

«Производитель»: «Тимлюйский шифер»,

«Тип сайдинга»: «Фиброцементный»,

«Толщина»: «8 мм»,

«Применение»: «Для наружной отделки»,

«Габариты (В х Ш х Г)»: «8 × 220 × 3000 мм»

},

«*link*»: «*https*://*baza*.124*bt*.*ru*/*product*/*fibrocementnyy*-*sayding*-*krasnyy*-*gost*-18124-2012/»,

«*resource*»: «*https*://*baza*.124*bt*.*ru*»,

«*date\_time*»: «2025-05-08 22:46:52»

},

{

«*user*»: «*testuser*123»,

«*category*»: «Кирпич»,

«*name*»: «Кирпич строительный М-150»,

«*price*»: «250»,

«*unit*»: «₽»,

«*characteristics*»: {

«Производитель»: «Керма»,

«Марка прочности»: «М-150»,

«Вес»: «3.5 кг»,

«Размер»: «250*x*120*x*65 мм»

},

«*link*»: «*https*://*baza*.124*bt*.*ru*/*product*/*kirpich*-*stroitelnyy*-*m*-150/»,

«*resource*»: «*https*://*baza*.124*bt*.*ru*»,

«*date\_time*»: «2025-05-08 22:47:30»

}

*]*

Как использовать эти данные?

1. Анализ цен – сравнение стоимости товаров между сайтами.
2. Построение каталога – автоматическое обновление базы товаров.
3. Мониторинг остатков – отслеживание наличия позиций.
4. Интеграция с *CRM*/*ERP* – загрузка в учетные системы.

3.11 Тестирование системы

Тестирование системы — это процесс проверки её работоспособности, корректности и устойчивости к различным сценариям использования [5]. В данном случае тестирование включает как автоматизированные проверки с помощью *unittest*, так и ручное тестирование для более сложных и специфичных случаев.

Автоматизированные тесты выполняются с помощью скриптов на *Python*, которые отправляют запросы к *API* и проверяют, соответствуют ли ответы ожидаемому поведению. Например, тесты проверяют, что:

1. Регистрация нового пользователя проходит успешно и возвращает корректный токен.
2. Авторизация работает только с правильными учётными данными.
3. Защищённые маршруты *API* требуют токена и отклоняют запросы без него.
4. Парсинг товаров запускается и возвращает данные в ожидаемом формате.

Ручное тестирование дополняет автоматизированное и позволяет проверить сложные сценарии, которые сложно или невозможно покрыть автотестами.

Например:

1. Как парсер обрабатывает сайты с динамической загрузкой данных через *JavaScript*.
2. Что происходит, если товар не имеет цены или характеристик.
3. Как система реагирует на нестандартные структуры страниц.

Тестирование помогает выявить ошибки на ранних этапах, убедиться в стабильности работы парсера и *API*, а также гарантировать, что данные сохраняются и обрабатываются корректно. Комбинация автоматизированных и ручных тестов обеспечивает максимальное покрытие возможных сценариев использования системы.

Тестирование включает автоматизированные тесты (*unittest*) и ручную проверку критических сценариев.

Цель: убедиться, что парсер и *API* работают корректно при разных входных данных.

Автоматизированные тесты (*unittest*)

Тесты проверяют ключевые точки системы:

1. Регистрация/авторизация.
2. Доступ к защищённым *endpoint*’ам.
3. Валидация токенов.
4. Обработка ошибок (неверные параметры, отсутствие данных).

Что проверяют автотесты?

1. Работа с пользователями
   1. *test\_*01*\_register* – регистрация нового пользователя (код 201) или обработка дубликата (409).
   2. *test\_*02*\_login* – авторизация и получение токена (200 + *access\_token*).

Защищённые *endpoint*’ы:

1. *test\_*03*\_protected\_valid\_token* – доступ с валидным токеном (200).
2. *test\_*04*\_protected\_no\_token* – доступ без токена (401).
3. *test\_*05*\_validate\_token* – проверка срока действия токена.

Обработка ошибок

1. *test\_*06*\_files\_not\_found* – запрос файлов пользователя (404 если нет данных).
2. *test\_*07*\_products\_missing\_param* – запрос товаров без параметров (400).
3. *test\_*10*\_savedb\_file\_not\_found* – попытка сохранения несуществующих данных (404).

Парсинг и данные

1. *test\_*08*\_products\_with\_param* – запрос товаров по пользователю (200 или 404).
2. *test\_*09*\_scrape\_endpoint* – запуск парсинга (коды 200, 400, 500 в зависимости от условий).

Ручное тестирование

Автотесты покрывают *API*, но критически важные сценарии парсинга проверяются вручную:

1. Парсинг разных сайтов.
   1. Запуск с разными *start\_url* (главная страница, категория, конкретный товар).
   2. Проверка обработки несуществующих страниц (ошибка 404).
2. Валидация данных.
   1. Корректность *XPath*/*CSS*-селекторов на сайтах с разной структурой.
   2. Проверка выходного *JSON* (все поля заполнены, нет *null*).
3. Граничные случаи.
   1. Сайты с капчей или *JavaScript*-загрузкой (должна быть обработка ошибок).
   2. Товары без цены или характеристик (парсер должен пропускать или отмечать *N*/*A*).
4. Интеграция с БД.
   1. Сохранение данных в *SQL*/*NoSQL* (проверка дублирования, обновления).
   2. Экспорт в *Excel*/*CSV* (корректность формата).

Пример вывода тестов.

Успешный результат:

*test\_*01*\_register* (*\_\_main\_\_APITestCase*) ... *OK*

*test\_*02*\_login* (*\_\_main\_\_APITestCase*) ... *OK*

*test\_*03*\_protected\_valid\_token* (*\_\_main\_\_APITestCase*) ... *OK*

*test\_*04*\_protected\_no\_token* (*\_\_main\_\_APITestCase*) ... *OK*

*Ran* 10 *tests* *in* 1.542*s*

*OK*

Ошибка:

*test\_*04*\_protected\_no\_token* (*\_\_main\_\_APITestCase*) ... *FAIL*

*AssertionError*: 200 != 401 # Ожидалась ошибка 401, но получили 200

1. Автотесты проверяют *API* и базовую логику.
2. Ручные тесты покрывают парсинг и обработку данных.
3. Интеграционные тесты гарантируют работу с БД и внешними сервисами.

3.12 Экономическая эффективность разработки системы

Экономическая эффективность разработки программного обеспечения определяется соотношением затрат на создание системы к потенциальной выгоде от её внедрения [9]. В данном проекте ключевыми факторами экономической эффективности являются:

1. **Снижение затрат на ручной сбор данных** – автоматизация скрейпинга цен на строительные материалы экономит время сотрудников.
2. **Повышение точности аналитики** – система минимизирует человеческие ошибки при обработке данных.
3. **Возможность коммерциализации** – сервис можно монетизировать через подписку для строительных компаний, ритейлеров и аналитиков рынка.
4. **Скорость окупаемости** – благодаря автоматизации и востребованности данных в *B*2*B*-сегменте проект может окупиться в короткие сроки.

Оценка затрат на создание системы представлены в таблице 7.

Таблица 7 – Оценка затрат на создание системы

|  |  |
| --- | --- |
| **Статья расходов** | **Стоимость (руб.)** |
| **Фронтенд (*React*, *Redux*)** | 150 000 – 300 000 |
| **Бекенд (*Python*, *Flask*)** | 200 000 – 400 000 |
| **Парсинг (*Scrapy*, *Playwright*)** | 100 000 – 200 000 |
| **База данных (*MySQL*)** | 50 000 – 100 000 |
| **Тестирование и *DevOps*** | 100 000 – 200 000 |
| **Итого** | **600 000 – 1 200 000** |

Примечание: стоимость может варьироваться в зависимости от региона разработки и уровня специалистов.

Система может приносить доход за счет:

1. **Подписки для бизнеса** (ежемесячная плата за доступ к аналитике).
2. **Продажи отчетов** (анализ динамики цен для поставщиков).
3. **Интеграции с *CRM*** (платное подключение к системам управления закупками).

**Модель монетизации представлена в таблице 8.**

Таблица 8 – **Модель монетизации**

|  |  |  |  |
| --- | --- | --- | --- |
| **Тариф** | **Цена (руб./мес.)** | **Кол-во клиентов** | **Годовой доход** |
| Базовый | 1 000 | 50 | 3 000 000 |
| Профессиональный | 5 000 | 20 | 3 600 000 |
| Корпоративный | 7 000 | 10 | 3 600 000 |
| **Итого** | - | **80** | **10 200 000** |

**Срок окупаемости (*Payback* *Period*, *PP*):**

1. **Первоначальные затраты:** 1 200 000 руб.
2. **Чистая прибыль в год (после налогов и расходов на хостинг ~30%):**  
   2 640 000 × 0.7 = 1 848 000 руб.
3. ***PP* =** 1 200 000 / (1 848 000 / 12) ≈ 7.8 месяцев.

**Вывод:** Проект окупится за 8 месяцев при текущих тарифах и плановом количестве клиентов.

1. **Снижение операционных издержек** для компаний, использующих систему (экономия на ручном мониторинге цен).
2. **Гибкость масштабирования** – возможность добавлять новые источники данных без значительных затрат.
3. **Потенциал для партнерств** с маркетплейсами и поставщиками строительных материалов.

Разработка системы веб-скрейпинга и анализа цен на строительные материалы экономически целесообразна благодаря низкому сроку окупаемости и высокому спросу на рыночную аналитику в *B*2*B*-сегменте.

3.13 Перспективы развития системы

Развитие системы парсинга может идти в нескольких ключевых направлениях, которые сделают её более мощной, универсальной и удобной для пользователей.

Интеграция с бизнес-системами

Одним из важных шагов станет подключение к популярным *CRM* и *ERP*-системам, таким как 1С, Битрикс24, *amoCRM* и другие. Это позволит автоматически загружать полученные данные напрямую в учётные системы компаний, избавляя от ручного переноса информации. Например, после парсинга товаров с сайта поставщика данные сразу будут попадать в базу 1С, обновляя цены и остатки [7].

Использование нейронных сетей и машинного обучения

Внедрение искусственного интеллекта поможет улучшить анализ данных. Нейросети смогут автоматически классифицировать товары, выявлять аномалии в ценах, прогнозировать изменения спроса и даже определять, какие характеристики наиболее важны для покупателей. Например, система сможет замечать, что на одних сайтах параметр называется «Размер», а на других — «Габариты», и объединять их в одну категорию.

Упрощение ввода данных: собственный синтаксис для селекторов

Сейчас пользователям приходится вручную прописывать *XPath* и *CSS*-селекторы, что требует технических знаний. В будущем можно разработать упрощённый язык запросов, который будет автоматически преобразовываться в нужные селекторы. Например, вместо //*div[*@*class*=«*price*»*]* можно будет писать «цена: *div*.*price*», а система сама сгенерирует корректный *XPath*. Это сделает настройку парсера доступной даже для неопытных пользователей [13].

Полная автоматизация поиска и анализа сайтов

Сейчас для парсинга нужно вручную указывать *URL* сайта и структуру данных, но в перспективе система сможет самостоятельно находить нужные сайты по заданным критериям (например, «интернет-магазины стройматериалов») и автоматически определять, где на странице расположены название, цена и характеристики. Это возможно благодаря комбинации веб-скрапинга, компьютерного зрения и анализа *DOM*-структуры.

Расширение возможностей мониторинга

Система сможет не только собирать данные, но и отслеживать изменения в реальном времени:

Динамику цен (например, если цена на кирпич упала на 10%, отправить уведомление).

Появление новых товаров или снятие старых с продажи.

Наличие скидок и акций, чтобы можно было быстро реагировать на выгодные предложения.

Поддержка новых форматов данных

Помимо стандартного *JSON* и *Excel*, можно добавить выгрузку в *Google* *Sheets*, *CSV* с настраиваемыми разделителями, *XML* для интеграции с устаревшими системами, а также *API* для передачи данных в другие сервисы.

Эти улучшения превратят систему из простого парсера в интеллектуальную платформу для сбора, анализа и управления данными, которая сможет работать практически без участия человека. Это особенно важно для компаний, которым нужно быстро получать актуальную информацию с множества источников и сразу использовать её в своих бизнес-процессах.

В ходе разработки системы сбора и анализа данных о строительных материалах была реализована комплексная платформа, сочетающая в себе современные технологии веб-разработки и алгоритмы обработки информации. Процесс создания системы осуществлялся по спиральной модели жизненного цикла, что позволило обеспечить итеративный подход к разработке, эффективное управление рисками и постоянный контроль качества на каждом этапе.

Серверная часть системы реализована с применением *RESTful*-подхода, что обеспечивает масштабируемость и простоту интеграции с внешними сервисами. Ядро системы включает механизмы авторизации, управления данными и веб-скрейпинга, организованные в виде отдельных модулей. Для хранения информации использована реляционная модель данных на базе *MySQL*, обеспечивающая надежное хранение и быстрый доступ к информации.

Все этапы разработки сопровождались комплексным тестированием, включающим как автоматизированные проверки *API*, так и ручное тестирование ключевых сценариев использования. Это позволило обеспечить высокую надежность системы и корректность работы всех компонентов.

Результатом проведенной работы стала полнофункциональная система, способная автоматически собирать, структурировать и анализировать данные о строительных материалах, предоставляя пользователям удобные инструменты для мониторинга рынка и принятия обоснованных решений. Разработанное решение демонстрирует высокую производительность и готово к промышленному внедрению.

ЗАКЛЮЧЕНИЕ

В процессе выполнения выпускной квалификационной работы была проделана значительная работа, направленная на решение задачи проектирования и реализации интеллектуальной информационной системы, предназначенной для сбора, обработки и анализа данных в сфере строительных материалов. Данная тема была выбрана неслучайно, поскольку рынок строительных материалов в настоящее время характеризуется высокой конкурентной активностью, стремительной цифровизацией и постоянными изменениями в ценовой политике. Всё это обусловливает потребность в эффективных средствах мониторинга, способных своевременно и объективно предоставлять информацию для принятия управленческих решений.

На начальном этапе работы был проведён детальный анализ предметной области. В рамках этого анализа были изучены ключевые особенности функционирования рынка строительных материалов, выявлены основные группы товаров, подходы к ценообразованию, типичные параметры, по которым пользователи сравнивают и выбирают продукцию. Были также рассмотрены действующие информационные ресурсы и цифровые платформы, используемые в данной отрасли, включая сайты производителей, торговых сетей и маркетплейсы. Этот этап позволил сформировать обоснованные требования к функционалу разрабатываемой системы, определить объём и характер обрабатываемых данных, а также сформулировать цели и задачи проекта.

На следующем этапе была спроектирована структура программной системы. Были определены основные компоненты системы, их взаимодействие и технические характеристики. Особое внимание было уделено выбору архитектурного подхода: в качестве основы была выбрана клиент-серверная архитектура с чётким разграничением логики между серверной частью, отвечающей за обработку данных и реализацию *API*, и клиентской частью, обеспечивающей взаимодействие с конечным пользователем. Кроме того, была проработана модель данных и способы их хранения, с учётом масштабируемости и потенциального расширения функционала.

В процессе реализации системы был создан модуль сбора данных, осуществляющий парсинг информации с внешних источников — сайтов конкурирующих компаний и торговых площадок. Этот модуль позволяет автоматически извлекать сведения о товарах, их характеристиках, ценах и наличии. Помимо этого, были реализованы механизмы аутентификации и авторизации пользователей, в том числе с использованием *JWT*-токенов, что обеспечило безопасность доступа к функционалу системы.

На завершающем этапе проектирования и разработки особое внимание было уделено тестированию созданной системы. С этой целью был разработан комплекс автотестов, охватывающий основные точки входа в систему и проверяющий корректность работы с различными запросами и ответами. Тесты охватывают как успешные сценарии, так и обработку ошибок, в том числе отсутствие токена, некорректные параметры запроса, отсутствие данных и другие ситуации. Кроме того, было проведено ручное тестирование, в рамках которого проверялись различные пользовательские сценарии с разнообразными входными данными. Это позволило убедиться в надёжности и устойчивости системы при работе в различных условиях.

Таким образом, в рамках данной выпускной квалификационной работы были последовательно реализованы все ключевые этапы жизненного цикла программного продукта: от анализа предметной области и постановки задачи до проектирования архитектуры, разработки и всестороннего тестирования готовой системы. Итогом работы стала функционирующая интеллектуальная система, способная выполнять автоматический сбор и анализ информации о товарах в области строительных материалов, что может быть использовано в целях мониторинга рынка, конкурентного анализа и поддержки принятия управленческих решений.

Достигнутые результаты подтверждают актуальность выбранной темы и демонстрируют практическую значимость реализованного программного решения. В дальнейшем систему можно доработать и расширить, добавив новые источники данных, реализовав более сложную аналитику или визуализацию, а также интеграцию с внешними корпоративными сервисами.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация JavaScript URL: https://developer.mozilla.org/ru/docs/Web/JavaScript (дата обращения: 17.05.2025);
2. Документация MySQL URL: https://dev.mysql.com/doc/ (дата обращения: 17.05.2025);
3. Руководство по MySQL URL: https://metanit.com/sql/mysql/ (дата обращения: 17.05.2025);
4. Руководство по Docker URL: https://my-js.org/docs/guide/docker/ (дата обращения: 17.05.2025);
5. Учебник Python URL: https://pymanual.github.io/ (дата обращения: 17.05.2025);
6. Документация Scrapy URL: https://docs.scrapy.org/en/latest/ (дата обращения: 17.05.2025);
7. Баранов, И.В. Информационные технологии и конкурентная разведка / И.В. Баранов. – М.: Академия, 2020. – 320 с.
8. Григорьев, А.В., Иванова, Н.С. Методы анализа данных и их применение в маркетинге / А.В. Григорьев, Н.С. Иванова. – СПб.: Питер, 2019. – 256 с.
9. Петров, С.М. Основы автоматизации бизнес-процессов в условиях цифровой экономики / С.М. Петров // Вестник цифровой экономики. – 2021. – Т. 5, № 3. – С. 120–130.
10. Иванов, И.И., Петрова, А.С. Современные системы сбора и анализа данных для управления конкурентоспособностью / И.И. Иванов, А.С. Петрова // Журнал прикладных исследований. – 2023. – Т. 15, № 1. – С. 25–33.
11. Литвинов В.П. Конкурентный анализ: теоретические аспекты и инструменты / В.П. Литвинов. – СПб.: Наука, 2020. – 290 с.
12. Python для анализа данных / У. Маккинни. - СПб.: Питер, 2023. - 144 с.
13. Чистый Python / Д. Бейдер. - СПб.: Питер, 2022. - 416 с.
14. Копырин А.С. Программирование на Python: учебное пособие - М.: МГТУ им. Баумана, 2021. - 168 с.
15. React: Начало работы URL: https://ru.legacy.reactjs.org/docs/getting-started.html (дата обращения: 17.05.2025);
16. Документация Redux URL: https://reactdev.ru/libs/redux/ (дата обращения: 17.05.2025);
17. Проектирование информационных систем : [учеб. пособие по специальности «Прикладная информатика»] / В. В. Коваленко. - М.: Форум, 2012. - 258 с.

Бакалаврская работа выполнена мною самостоятельно. Используемые в работе материалы и концепции из опубликованной научной литературы и других источников имеют ссылки на них. Библиографический список содержит 17 наименований.

Один печатный экземпляр и один электронный экземпляр сданы на кафедру ИЭС.

\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (Ф.И.О.)