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

КУРСОВАЯ РАБОТА

по дисциплине «Цифровизация производства и продаж»

Разработка программного обеспечения для анализа данных о рынке строительных материалов
Тема

Преподаватель — подпись, дата 26.12.24 — М. А. Масюк (инициалы, фамилия)

Обучающийся БПЦ21-01, 211519018 (номер группы, зачетной книжки) — подпись, дата 26.12.24 — Е. А. Семенов (инициалы, фамилия)

Красноярск 2024

Содержание

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: от структурированных таблиц и баз данных до неструктурированных текстов на сайтах производителей и маркетплейсах.

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, изучении характеристик и отзывов. 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 структурированных и неструктурированных 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.

Целью данной курсовой работы является разработка программного обеспечения для автоматизированного сбора данных о строительных материалах. Программа должна обеспечивать сбор как структурированных данных (характеристики, цены), так и неструктурированных (описания) 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.

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, имеющей как структурированную, так и неструктурированную природу.

Данная курсовая работа направлена на создание практического инструмента, который будет полезен как в строительной отрасли, так и в других сферах, где важен быстрый и точный доступ к информации.

ОСНОВНАЯ ЧАСТЬ

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:
   1.1.Бетон, цемент, песок, гравий.
   1.2.Кирпич (керамический, силикатный).
   1.3.Металлоконструкции.
2. Отделочные материалы:
   2.1.Штукатурки, шпаклёвки, краски.
   2.2.Плитка (керамическая, мраморная), ламинат, паркет.
   2.3.Обои, декоративные панели.
3. Изоляционные материалы:
   3.1.Теплоизоляция (минеральная вата, пенопласт).
   3.2.Звукоизоляция.
   3.3.Гидроизоляционные плёнки и мембраны.
4. Инженерные материалы:
   4.1. Трубы и фитинги.
   4.2. Электрические кабели, системы водоснабжения и вентиляции.
5. Инновационные и экологичные материалы:
   5.1. Эко-материалы (солома, переработанный пластик, бамбук).

5.2. 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.

1.1.Использование композитных материалов с высокой прочностью и низким весом.

1.2.Внедрение 3*D*-печати для создания строительных конструкций.

1.3.Применение «зелёных» материалов, способствующих снижению углеродного следа.

2. Цифровизация отрасли.

2.1.*BIM*-технологии (*Building Information Modeling*) для проектирования и управления строительными объектами.

2.2.Использование программного обеспечения для оценки смет и управления поставками строительных материалов.

3. Автоматизация процессов.

3.1.Использование роботов и дронов для выполнения строительных задач и мониторинга.

3.2.Внедрение умных систем на объектах строительства (контроль за качеством материалов, автоматическое управление климатом).

4. Устойчивое строительство.

4.1. Переход на возобновляемые источники энергии в строительных процессах.

4.2. 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.

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.

Разработка программного обеспечения, которое может собирать и обрабатывать как структурированные (*JSON, XML*), так и неструктурированные (*HTML*) данные о стройматериалах, отвечает этим вызовам. Такое решение позволит объединить данные из разных источников, упростить их анализ и повысить эффективность работы строительных компаний.

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 курсовой работы актуальной и практически значимой.

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. 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.

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.

2. "*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).

2. *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.

2. *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.

Проблемы существующих решений.

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) и *API* поставщиков (структурированные 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.

1.2 Обоснование необходимости разработки и юридической правомерности

Современный строительный бизнес сталкивается с рядом вызовов, связанных с эффективным поиском, сравнением и выбором строительных материалов. Разработка программы для сбора структурированных и неструктурированных данных о строительных материалах позволит решить следующие ключевые задачи:

1. Автоматизация процесса поиска. Многие строительные компании тратят значительное время на поиск материалов вручную через каталоги и сайты поставщиков. Автоматизированная система снизит трудозатраты и ускорит принятие решений.
2. Повышение качества данных. Сравнение материалов, цен и условий поставки из разных источников на одном интерфейсе снизит вероятность ошибок и предоставит наиболее актуальную информацию.
3. Улучшение конкурентоспособности. Быстрый доступ к широкому спектру поставщиков и материалов даст строительным компаниям преимущество на рынке за счёт оптимизации затрат и выбора лучших предложений.
4. Снижение операционных затрат. Использование автоматизированного сбора данных исключает необходимость ручного мониторинга сайтов и каталогов, что уменьшает затраты на трудовые ресурсы.

Актуальность разработки обусловлена:

1. Ростом объемов данных. Увеличение количества поставщиков и ассортимента строительных материалов создаёт потребность в эффективной обработке больших массивов информации.
2. Переход бизнеса в онлайн. Строительные компании всё чаще используют цифровые каналы для закупок и управления процессами, что требует современных инструментов автоматизации.
3. Повышение требований к скорости и качеству работы. Конкуренция на рынке требует от компаний быстрого реагирования на изменения цен и наличие материалов у поставщиков.

Таким образом, разработка системы сбора данных является не только актуальной, но и необходимой для повышения эффективности работы участников строительного рынка.

Юридическая правомерность разработки. Для создания и эксплуатации системы, собирающей данные из различных источников, важно соблюдать правовые нормы, регулирующие использование информации, размещённой в интернете. Рассмотрим ключевые аспекты юридической правомерности:

1. Использование открытых данных. Согласно ст. 1260 Гражданского кодекса Российской Федерации (ГК РФ), информация, размещённая в открытом доступе в интернете, может быть использована без согласия автора, если это не противоречит условиям использования, указанным владельцем ресурса.

2. Соблюдение условий использования данных. Важно учитывать положения пользовательских соглашений и политики конфиденциальности сайтов, с которых планируется сбор данных. Некоторые ресурсы запрещают автоматизированный сбор информации (*scraping*). Для таких случаев потребуется запрос разрешения у владельцев.

3. Защита персональных данных. если на сайтах, с которых собираются данные, имеются персональные данные (например, контактная информация), необходимо соблюдать нормы закона "О персональных данных" (Федеральный закон № 152-ФЗ). В рамках системы следует исключить сбор таких данных или использовать их только с согласия субъекта.

4. Интеллектуальная собственность. Данные, такие как текстовые описания или фотографии, могут быть защищены авторским правом. Для использования таких данных в коммерческих целях необходимо получить разрешение правообладателей.

5. Ответственность за вмешательство в работу сайтов. Согласно ст. 272 и 273 УК РФ, вмешательство в работу сайтов (например, обход ограничений доступа) является незаконным. Поэтому разработка системы должна учитывать правила добросовестного использования, включая соблюдение ограничения частоты запросов.

Рекомендации для правомерного использования:

1. Анализ условий использования данных. Перед началом сбора данных изучать пользовательские соглашения и политики конфиденциальности сайтов.

2. Сбор только общедоступной информации. Исключить сбор данных, защищённых паролем или иных способов аутентификации.

3. Информирование о веб-скрейпинге. При необходимости уведомлять владельцев сайтов о сборе данных для обеспечения прозрачности.

4. Анонимизация данных. Если данные содержат идентифицирующую информацию, она должна быть анонимизирована или исключена.

Разработка системы сбора данных о строительных материалах отвечает потребностям современного строительного бизнеса, решая задачи автоматизации, повышения эффективности и конкурентоспособности. При этом, соблюдение правовых норм, таких как использование только открытых данных, уважение авторских прав и защита персональной информации, делает проект юридически правомерным и безопасным.

2 ПРОЕКТИРОВАНИЕ СИСТЕМЫ

2.1 Требования к системе

Функциональные требования.

В своей сути система с точки зрения функционала состоит из 2 частей это подсистема пользователя (человека, желающего получить данные у системы) и системного администратора (человека, желающего добавить данные в систему).

Функциональные требования пользователя:

1. Система должна предоставлять пользователю возможность настроить параметры поиска данных в виде формы ввода:
    1.1. Регион поиска.
    1.2. Город поиска.
    1.3. Название стройматериала.
    1.4. Артикул.
    1.5. Бренд.
    1.6. Диапазон цен.
    1.7. Характеристики материала.
2. Система должна предоставлять пользователю информацию о найденных стройматериалах в виде таблицы:
    2.1. Ссылка на источник.
    2.2. Дата получения данных.
    2.3. Название стройматериала.
    2.4. Артикул.
    2.5. Характеристики.
    2.6. Город продажи.
    2.7. Регион продажи.
3. Система должна предоставлять пользователю возможность отсортировать найденные данные по цене, дате запроса, городу наличия.
4. Система должна позволить пользователю экспортировать данных в формат электронных таблиц.
5. Система должна предоставлять пользователю возможность построения графиков для анализа уровня цены стройматериала в зависимости от времени запроса, то есть должны быть параметры построения графика:
    5.1. Период запроса данных.
    5.2. Наименование (артикул) стройматериала.
    5.3. Ценовой диапазон.
6. Система должна информировать пользователя об ошибках поиска данных о стройматериалах.

Функциональные требования системного администратора:

1. Система должна позволять системному администратору возможность производить поиск данных и указать параметры поиска:
   1.1. Ссылка на источник.
   1.2. Прокси, с которого будет происходить запрос.
   1.3. Вариант веб-скрейпинга (*API/html*)
   1.4. Ключевые слова.
   1.5. Периодичность запросов к ресурсу.
2. Система должна позволять системному аналитику просматривать полученные данные в виде таблицы с указанием следующих параметров:
   2.1. Ссылка на источник.
   2.2. Дата получения данных.
   2.3. Название стройматериала.
   2.4. Артикул.
   2.5. Характеристики.
   2.6. Город продажи.
   2.7. Регион продажи.
3. Система должна уведомлять системного администратора об ошибках в сборе данных с указанием логов запроса и предлагать изменить параметры запроса.
4. Система должна позволять системному администратору возможность сохранять полученные из веб-скрейпинга данных в базу данных системы.
5. Система должна позволять системному администратору просматривать ресурсы, из которых уже были изъяты данные и настраивать периодичность дальнейшего сбора, а также уведомлять о сбоях в автоматическом сборе данных (изменение структуры страницы html/блокировка прокси/изменение структуры json), настройка содержит следующие параметры:
   5.1. Период сбора.
   5.2. Период ротации прокси.
   5.3. Список прокси участвующих в ротации.

Общие функциональные требования:

1. Система должна позволять провести регистрацию в систему пользователя и системного администратора, указав логин, пароль, повторно пароль, код на электронную почту.
2. Система должна позволять провести вход в систему с указанием логина и пароля для обоих видов пользователя, а также отправлять проверочный код для восстановления аккаунта при 4 не удачных вводах логина и пароля.

Нефункциональные требования:

1. Требования к производительности:
   1.1. Система должна обрабатывать не менее 100 запросов пользователей в минуту с задержкой ответа не более 2 секунд.
   1.2. Парсер должен извлекать данные с сайтов с задержкой на запрос к одному и тому же ресурсу исходя из расчета секунда, помноженная на 10 что бы исключить или оттянуть блокировку прокси.

1.3. Парсер должен периодически извлекать данные с ресурсов (период указывает системный администратор)

1.4. Система должна поддерживать одновременно до 500 активных пользователей без снижения производительности.

2. Требования к безопасности:

2.1.Данные пользователей и результаты анализа должны шифроваться при хранении с использованием алгоритма *AES*-256.

2.2.Передача данных между клиентом и сервером должна выполняться через протокол *HTTPS* и доступ к серверу должно иметь только приложение клиента (запрет сторонних *IP*/доменов).

3. Требования к масштабируемости:

3.1.Система должна быть масштабируемой для добавления новых функций (например, интеграции с другими *API*) без необходимости значительной переработки архитектуры.

4. Требования к интерфейсу:

4.1.Интерфейс должен быть адаптивным и корректно отображаться на устройствах с экраном от 4 до 15 дюймов.

4.2.Среднее время загрузки страниц пользовательского интерфейса не должно превышать 1 секунды при скорости интернета от 10 Мбит/с.

5. Требования к совместимости:

5.1.Система должна поддерживать работу в современных браузерах (*Google Chrome, Mozilla Firefox, Microsoft Edge, Safari*) версии не старше 3 лет.

5.2.Система должна быть совместима с операционными системами *Windows, macOS, iOS* и *Android*.

5.3.Система должна позволять развернуть себя на операционных системах Unix-подобных операционных системах.

6. Логирование и аудит:

6.1. Система должна сохранять логи всех операций пользователей (включая действия парсера) за последние 6 месяцев.

6.2. Логи должны быть доступны для анализа только администраторам системы.

2.2 Функциональная модель системы

В рамках проектирования нашей системы было принято решение разработать диаграмму вариантов использования, чтобы визуально отобразить взаимодействие пользователей с системой и определить основные функции, которые она должна выполнять. Данная диаграмма позволяет понять, как именно пользователи будут взаимодействовать с системой для достижения своих целей, а также помогает выявить ключевые сценарии использования и возможные зависимости между ними.

Использование диаграммы вариантов использования важно для обеспечения четкого понимания требований к функциональности системы как со стороны разработчиков, так и заинтересованных сторон. Такой подход способствует формализации всех возможных операций, которые могут быть

выполнены в системе, и их взаимосвязей. Кроме того, диаграмма позволяет более эффективно организовать работу над системой, выявляя области, требующие более детальной проработки.

Мы стремились отразить на диаграмме все важные варианты использования системы, чтобы охватить широкий спектр действий, которые могут выполнять пользователи. Это также дает возможность на ранних этапах проектирования выявить потенциальные улучшения, обеспечить модульность разработки и проверить, соответствуют ли функции системы ожиданиям пользователя.

Таким образом, создание диаграммы вариантов использования не только помогло структурировать процесс проектирования, но и станет основой для дальнейшей реализации системы, делая её более удобной и понятной как для разработчиков, так и для конечных пользователей. Сама диаграмма вариантов использования представлена на рисунке 1.

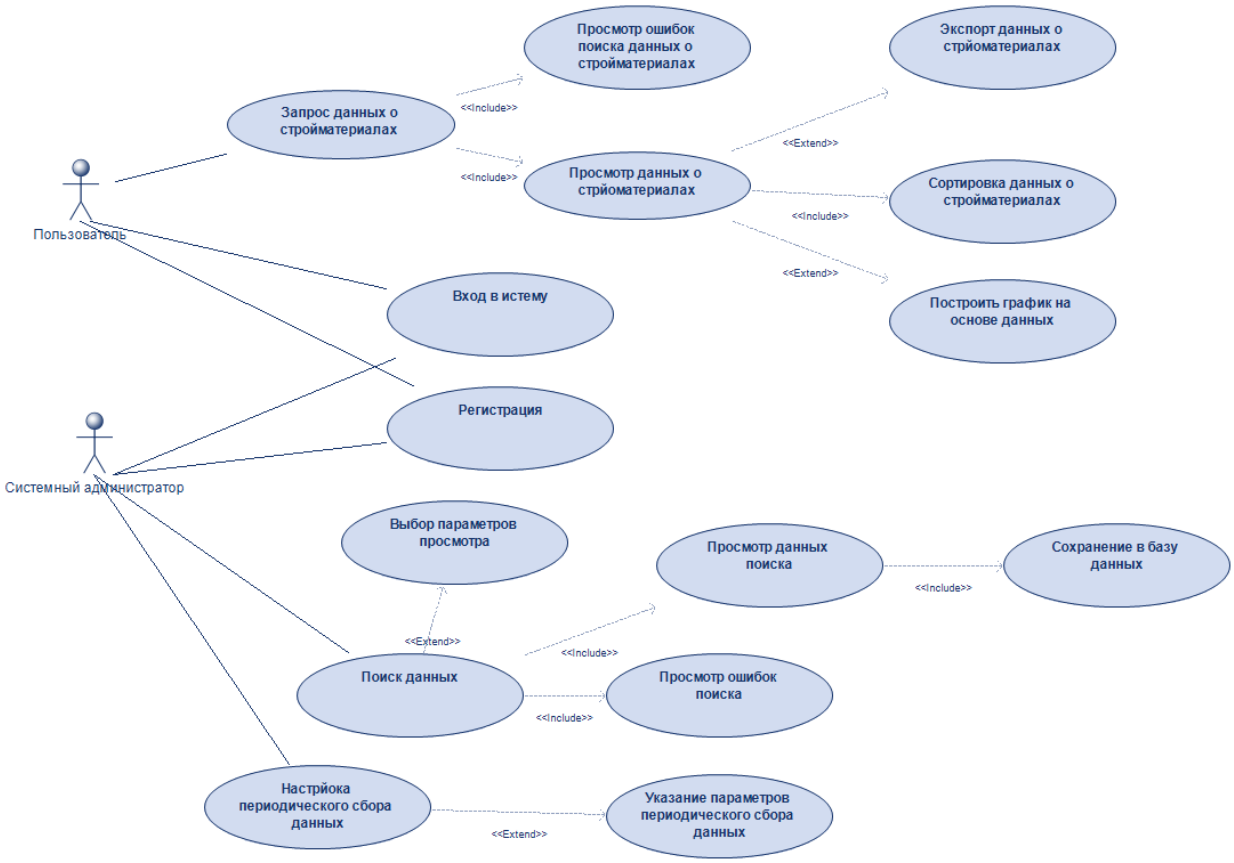

Рисунок 1 – Диаграмма вариантов использования проектируемой системы.

На данной диаграмме представлено как два вида пользователей взаимодействуют с системой.

Оба пользователя используют прецедент вход и регистрацию, которые подразумевают ввод регистрационных данных или данных для входа, что позволит взаимодействовать с системой.

Пользователь использует функцию «Запрос данных о стройматериалах», что позволяет ему запросить эти данные и перейти в функцию «Просмотр

данных о стройматериалах», что уже само по себе подразумевает сортировку данных при отображении, но пользователь так же может изменить фильтры данной сортировки. После просмотра данных о стройматериалах пользователь может построить график по указанным параметрам или экспортировать данные в формате таблицы. Также после поиска данных пользователь может получить уведомление об ошибках поиска.

Системный администратор может использовать функцию «Поиск данных», что так же требует параметров для поиска и он должен их указать в функции «Выбор параметров поиска». После завершения поиска есть возможность открыть функцию «Просмотр данных поиска» и далее «Сохранения в базу данных». Так же после поиска данных возможен вариант наличия ошибок, о чем системный администратор будет уведомлен и получит функцию «Просмотр ошибок поиска». Также системный администратор может настраивать автоматический периодический сбор данных из тех ресурсов, которые уже были обработаны, это происходит в функции «Настройка периодического сбора данных», что подразумевает указание параметров для периодического сбора в функции «Указание параметров периодического сбора данных».

Также для проектирования нашей системы было принято решение разработать модель в нотации

2.3 Клиент-серверная архитектура системы

Клиент-серверная архитектура системы предполагает взаимодействие между несколькими компонентами, включая клиента, сервер для клиента и основной сервер. В рамках данной архитектуры функции системы разделяются между слоями, что обеспечивает гибкость, производительность и масштабируемость. Клиент-серверная архитектура представлена на рисунке 2.

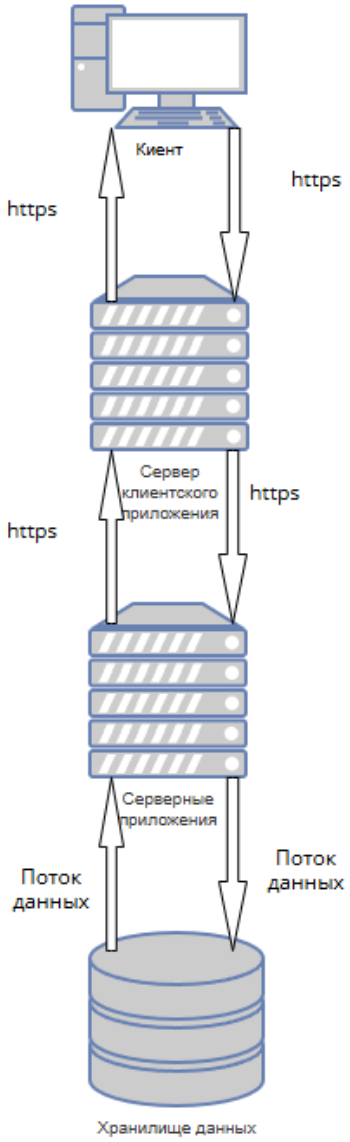

Рисунок 2 - Клиент-серверная архитектура системы

Основные элементы архитектуры:

1. Клиент. Представляет собой пользовательский интерфейс, с которым взаимодействует конечный пользователь. Это может быть веб-приложение, мобильное приложение или десктопная программа. Клиент отправляет запросы серверу через стандартные протоколы (*HTTP/HTTPS*) и отображает полученные результаты.

2. Сервер для клиента (*Frontend*-сервер). Это промежуточный слой между клиентом и основным сервером. Выполняет предварительную обработку запросов, включая: авторизацию и аутентификацию, Кэширование данных для ускорения ответов, Маршрутизацию запросов к соответствующим сервисам основного сервера, скрывает внутреннюю структуру системы от клиента, повышая безопасность.

3. Основной сервер (*Backend*-сервер). Обеспечивает выполнение основной бизнес-логики приложения, выполняет сложную обработку данных и взаимодействие с базой данных, принимает запросы от *Frontend*-сервера и возвращает готовые данные или результаты выполнения операций.

4. Хранилище данных. Используется для долгосрочного хранения и управления данными. Основной сервер взаимодействует с хранилищем через специализированные драйверы или протоколы, такие как *SQL, MongoDB Wire Protocol* или *REST API*.

Клиент-серверная архитектура позволяет построить надежную, эффективную и масштабируемую систему, соответствующую современным требованиям.

2.4 Моделирование данных

В качестве описания данных используем диаграмму сущностных классов.

Диаграмма сущностных классов (или диаграмма классов) в *UML* (*Unified Modeling Language*) представляет собой статическую модель, которая описывает структуру системы в терминах классов, их атрибутов, методов и связей между ними. Эта диаграмма позволяет визуализировать, как компоненты системы взаимодействуют между собой, а также какие данные и функциональные возможности они представляют.

Классы на диаграмме отображают основные сущности системы. Каждый класс включает атрибуты, которые представляют данные, а также методы, которые описывают поведение. Классы могут быть связаны различными отношениями, такими как наследование, агрегация и композиция, что позволяет моделировать иерархию и структуру системы.

Диаграмма классов используется на этапе проектирования программного обеспечения, чтобы показать, как система будет устроена, какие компоненты в ней будут, и как они будут взаимодействовать. Она помогает разработчикам и архитекторам лучше понять архитектуру системы, определить взаимосвязи между объектами и выявить ключевые объекты и их поведение. Это также облегчает документирование системы и передачу информации между участниками проекта, включая новых разработчиков или тестировщиков.

Таким образом, диаграмма сущностных классов является важным инструментом для моделирования структуры системы и проектирования её архитектуры, обеспечивая чёткое представление о том, как классы и их взаимодействия будут реализованы в коде.

Диаграмма сущностных классов представлена на рисунке 3.

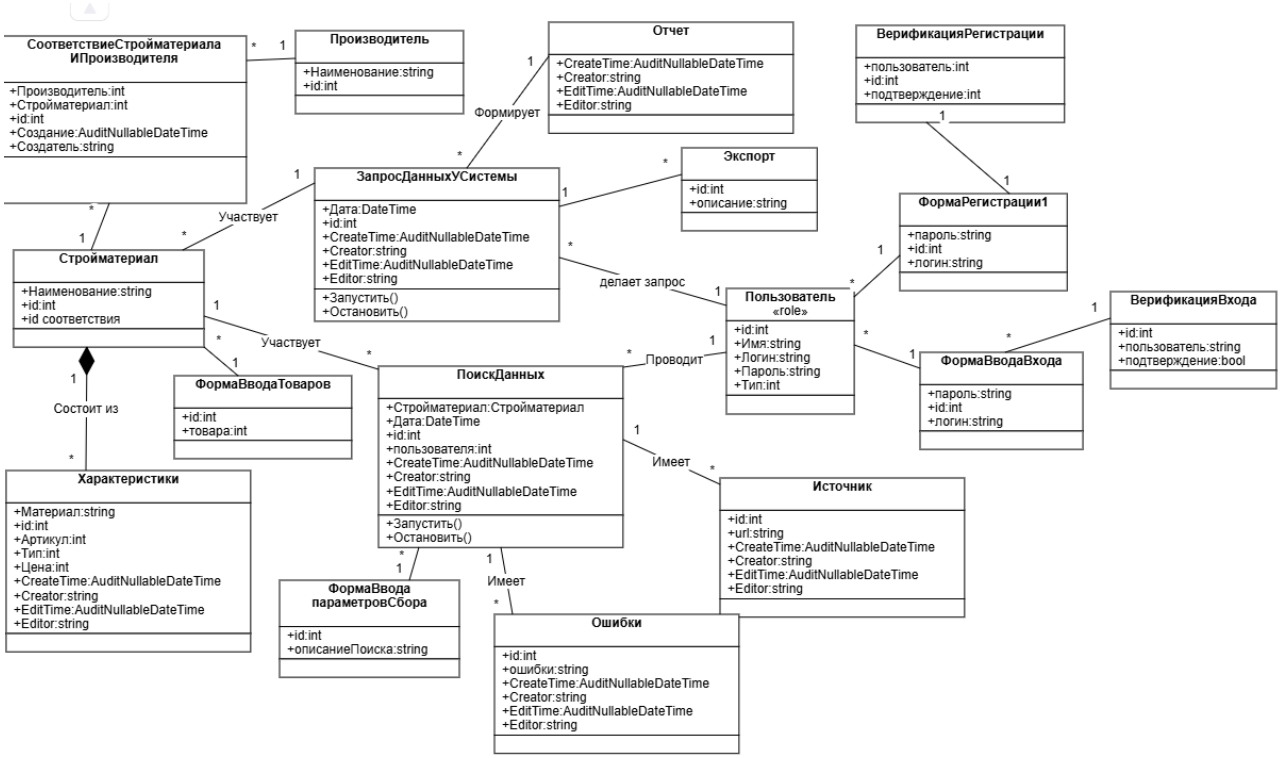

Рисунок 3 – Диаграмма сущностных классов.

Диаграмма сущностных классов, представленная в системе конкурентного анализа стройматериалов, включает несколько ключевых объектов и их взаимосвязи. Основные классы:

1. Строительный материал: Этот класс представляет данные о стройматериале, такие как наименование, характеристики и состав товара. Он связан с другими классами, такими как Характеристика, которая определяет свойства материала, и Пользователь, который может быть участником системы. Строительный материал также может быть частью Формы выбора товара, которая используется для создания запросов на получение данных о материале.

2. Пользователь: Класс представляет участников системы, которые могут быть как пользователями, так и администраторами. Каждый пользователь имеет уникальный идентификатор и связан с действиями, которые они выполняют в системе, например, с запросами на получение данных. Пользователь может взаимодействовать с Запросом данных/системы, формировать отчеты и иметь доступ к различным функциям платформы.

3. Запрос данных/системы: Этот класс отвечает за хранение данных о запросах, которые отправляются в систему. Запросы включают информацию о типе запроса, его времени и результатах. Запросы могут быть инициированы пользователями для получения данных о строительных материалах.

4. Источник: Источник представляет собой веб-ресурс или *API*, откуда система получает информацию о товарах. Источники могут быть различными — от маркетплейсов до сайтов производителей.

5. Форма выбора товара: Этот класс используется для отображения и выбора строительных материалов на платформе. Он взаимодействует с классом

Запрос данных/системы и помогает пользователю получить нужную информацию для дальнейшего анализа.

6. Отчет: Этот класс предназначен для создания отчетов на основе собранных данных. Отчеты генерируются для пользователей и могут включать такие атрибуты, как дата создания и создающий пользователь.

7. Характеристика: Этот класс описывает характеристики строительных материалов, такие как размер, вес, материал, и другие параметры. Характеристики могут быть связаны с конкретными товарами и отображаться в результатах запроса.

Каждый класс в системе содержит атрибуты, которые описывают сущности, и методы для работы с ними, например, для создания записей, запроса информации и генерации отчетов. Взаимосвязи между классами (например, между Пользователем и Запросом данных/системы) обеспечивают эффективное управление данными и позволяют пользователям получать нужную информацию для анализа конкурентоспособности строительных материалов.

2.5 Моделирование логики работы

Диаграмма состояний – это визуальный инструмент, который описывает все возможные состояния объекта в системе и переходы между ними в ответ на определённые события. Она помогает понять, как объект изменяет своё поведение на разных этапах жизненного цикла.

Диаграммы состояний представлены на рисунках 4 – 7.

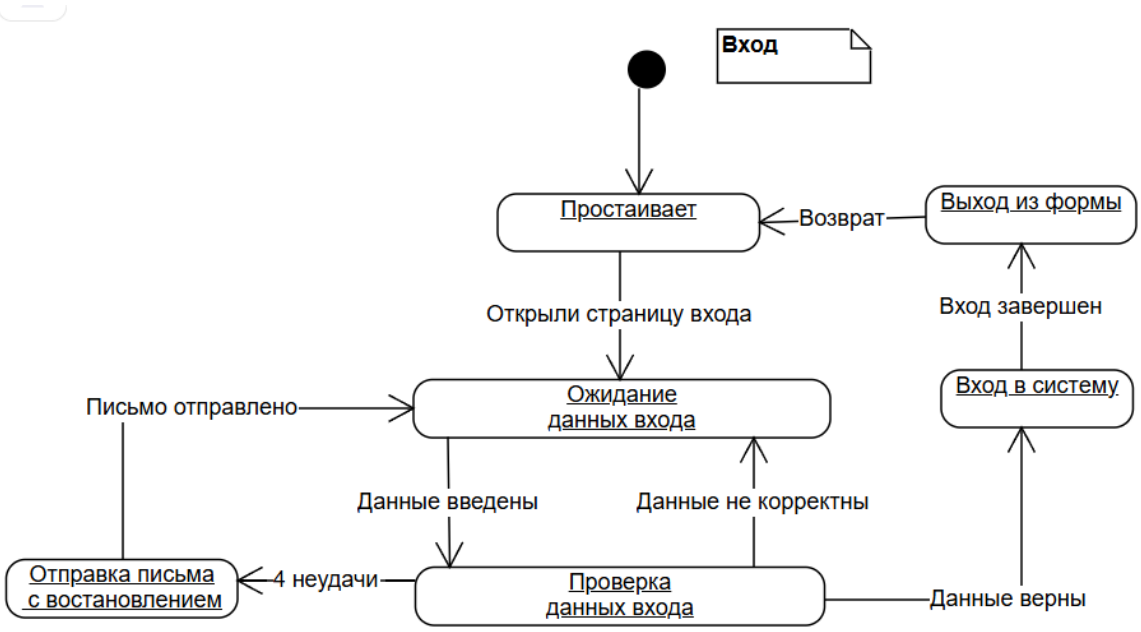

Рисунок 4 – Диаграмма состояний процесса «Вход».

На представленной диаграмме состояний показан процесс входа пользователя в систему, включая возможные сценарии и состояния, через которые проходит система:

1. Начальное состояние – система находится в состоянии "Простаивает". Пользователь может открыть страницу входа, что переводит систему в состояние "Ожидание данных входа". Пользователь также может вернуться назад, выбрав "Выход из формы", завершив процесс.

2. Ожидание данных входа – система ожидает ввода учетных данных от пользователя. Если данные введены, система переходит в состояние "Проверка данных входа". Если пользователь запросил восстановление доступа, отправляется письмо, и система переходит в состояние "Отправка письма с восстановлением".

3. Проверка данных входа – введенные данные проверяются. Если данные корректны, пользователь входит в систему (переход в состояние "Вход в систему"). Если данные некорректны, система возвращается в состояние "Ожидание данных входа".

4. Отправка письма с восстановлением – если пользователь запросил восстановление, отправляется письмо. При успешной отправке процесс завершен. В случае неудачи система возвращается в состояние "Ожидание данных входа".

5. Завершение процесса. После успешного входа или выхода из формы процесс завершен.

Диаграмма демонстрирует основные сценарии взаимодействия пользователя с формой входа, включая корректный ввод, ошибки и запрос восстановления.

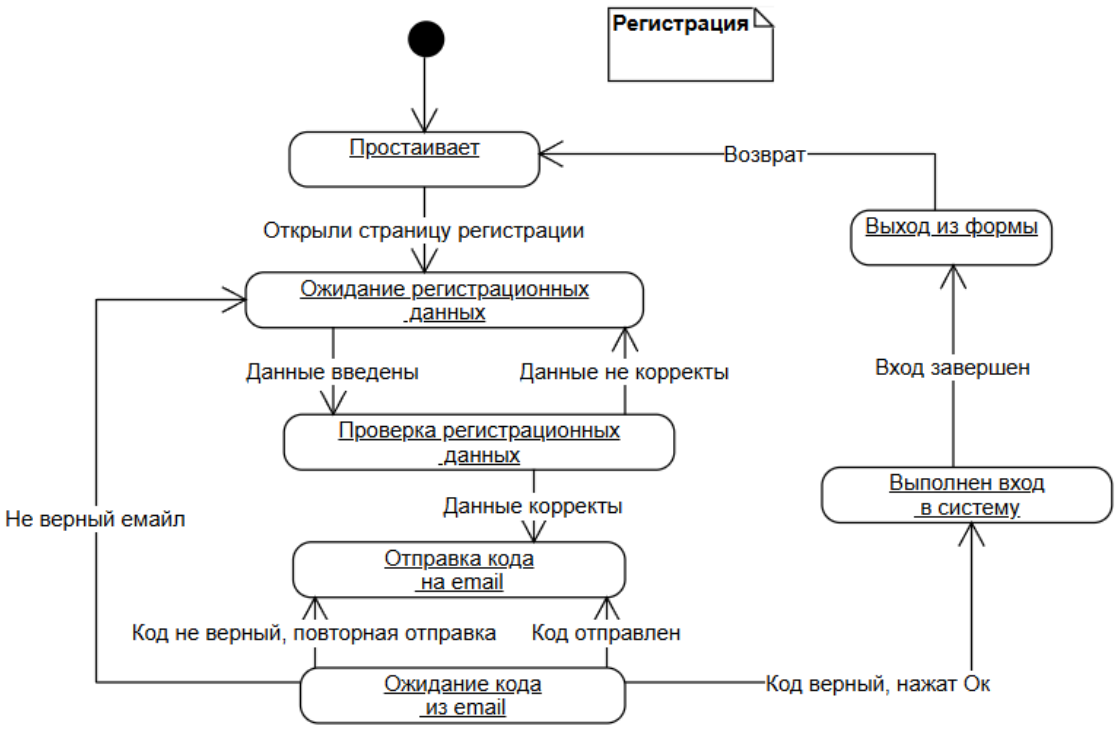

Рисунок 5 – Диаграмма состояний процесса «Регистрация».

На диаграмме состояний изображён процесс регистрации пользователя в системе. Рассмотрим основные состояния и переходы между ними:

1. Начальное состояние. Система находится в состоянии "Простаивает". Пользователь может открыть страницу регистрации, переводя систему в состояние "Ожидание регистрационных данных". Пользователь может вернуться, выбрав "Выход из формы", завершив процесс.

2. Ожидание регистрационных данных. Система ожидает ввода данных от пользователя (например, логина, пароля, *email*). После ввода данных. Если данные корректны, система переходит в состояние "Проверка регистрационных данных". Если данные некорректны (например, неверный *email*), система возвращается в "Ожидание регистрационных данных"**.**

3. Проверка регистрационных данных. Введённые данные проверяются. При корректных данных система переходит в состояние "Отправка кода на *email*". Если *email* неверный, система возвращается в "Ожидание регистрационных данных".

4. Отправка кода на *email*. Система отправляет код подтверждения на указанный *email*. После отправки система переходит в состояние "Ожидание кода из *email*".

5. Ожидание кода из *email*. Система ожидает ввода кода подтверждения. Если код верный и пользователь нажал "ОК", система переходит в состояние "Выполнен вход в систему". Если код неверный, пользователь может запросить повторную отправку, оставаясь в этом состоянии.

6. Завершение процесса. После успешного ввода кода регистрация завершена, и пользователь входит в систему **(**"Выполнен вход в систему"). Если пользователь выбирает "Выход из формы", процесс регистрации завершается.

Диаграмма охватывает весь процесс регистрации, включая ввод данных, их проверку, отправку и ввод кода подтверждения. Также учтены сценарии возврата пользователя и повторной отправки кода.

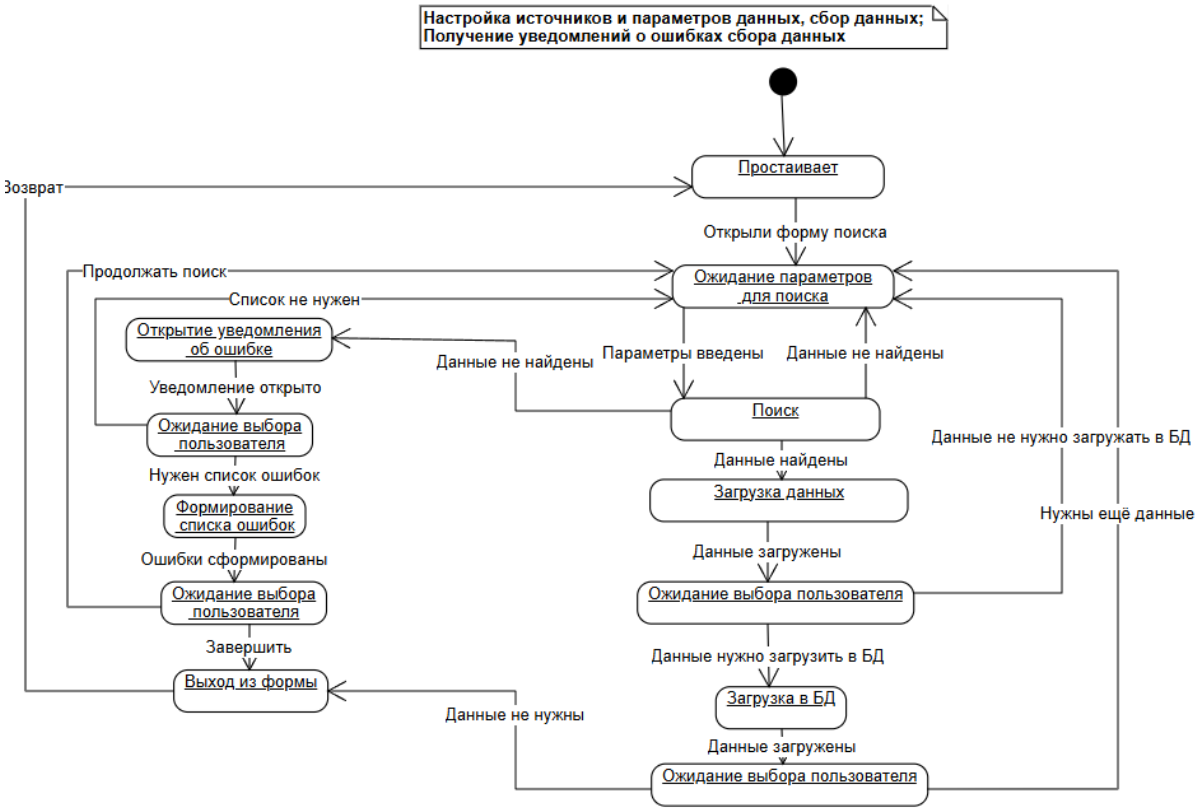

Рисунок 6 – Диаграмма состояний процессов «Настройка параметров данных. уведомлений о ошибках сбора данных».

На диаграмме состояний представлен процесс настройки источников данных, параметров поиска, сбора данных, а также получения уведомлений об ошибках сбора данных. Рассмотрим ключевые состояния и переходы:

1. Простаивает. Начальное состояние системы. Пользователь может открыть форму поиска, переводя систему в состояние "Ожидание параметров для поиска". Или завершить процесс, выбрав "Выход из формы".

2. Ожидание параметров для поиска. Система ожидает ввода параметров от пользователя. Если параметры введены, система переходит к "Поиску". Если данные не найдены или список не нужен, можно вернуться в "Простаивает".

3. Поиск. Система выполняет поиск по заданным параметрам. Если данные найдены, система переходит в состояние "Загрузка данных". Если данные не найдены, пользователь возвращается в состояние "Ожидание параметров для поиска".

4. Загрузка данных. Данные, найденные на предыдущем шаге, загружаются в систему. Если данные не нужно загружать в базу данных (БД), система переходит в "Ожидание выбора пользователя". Если данные загружены, система остаётся в состоянии ожидания действий пользователя.

5. Ожидание выбора пользователя. Пользователь решает, что делать с данными. Если данные нужны, происходит дальнейшая работа с ними. Если данные не нужны, процесс завершён, и система возвращается в начальное состояние.

6. Открытие уведомления об ошибке. В случае ошибки система переходит в состояние "Ожидание выбора пользователя" или позволяет формировать список ошибок. После формирования списка ошибок возможен возврат к другим состояниям, включая "Простаивает" или завершение процесса.

7. Формирование списка ошибок. Пользователь может сформировать список ошибок, чтобы понять, какие проблемы возникли. После этого система либо завершает процесс, либо возвращается в состояние ожидания действий.

Пользователь может завершить процесс в любой момент, выбрав "Выход из формы".

Если данные обработаны успешно, процесс завершается, и система возвращается в состояние "Простаивает".

Диаграмма охватывает полный процесс работы с данными, включая их поиск, загрузку, проверку ошибок и взаимодействие с пользователем. Учтены сценарии ошибок, повторных запросов и возвратов.

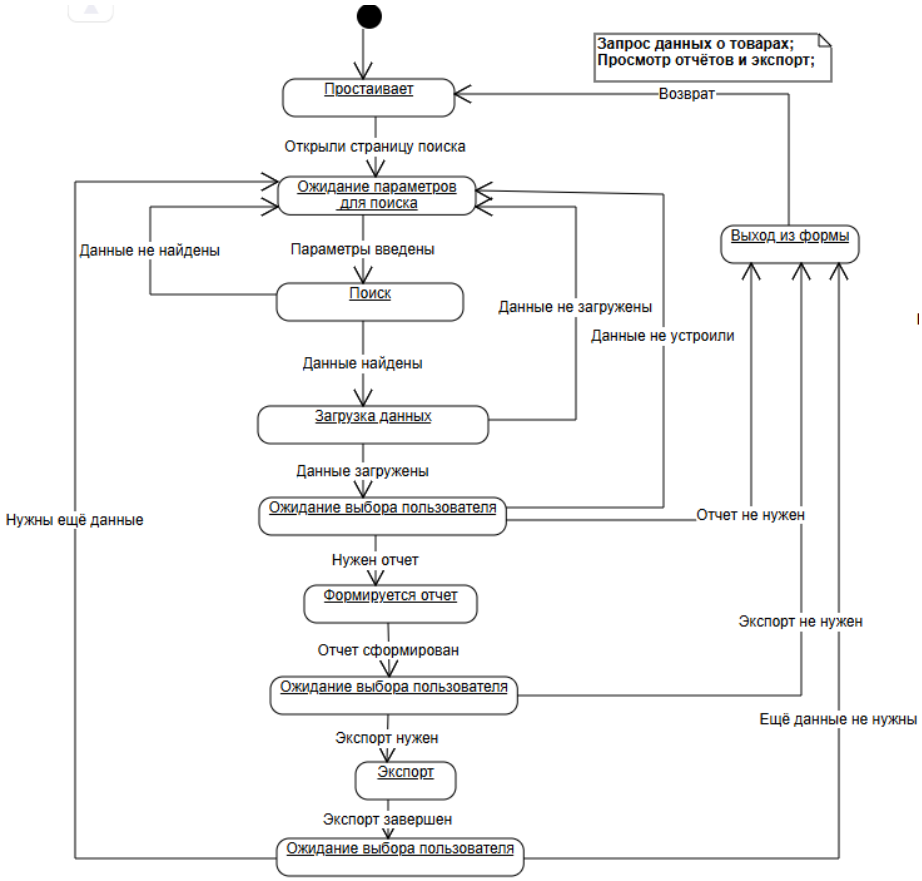

Рисунок 7 – Диаграмма состояний «Запрос данных о товарах. Просмотр отчётов и экспорт»

На диаграмме представлена блок-схема процесса запроса данных о товарах, включая этапы поиска, загрузки данных и формирования отчетов.

1. Начало процесса – открывается страница поиска.

2. Ожидание параметров – пользователь вводит параметры для поиска.

3. Поиск – если данные не найдены, процесс завершен, если данные найдены, происходит их загрузка.

4. Загрузка данных – после загрузки данных происходит ожидание выбора пользователя.

5. Формирование отчета – если отчет нужен, он формируется, если нет, процесс может завершиться.

6. Экспорт данных – если экспорт нужен, он производится, после чего процесс завершается.

Последовательность выполнения процессов описаны диаграммами последовательности.

Диаграмма последовательности – это один из типов диаграмм в *UML* (*Unified Modeling Language*), предназначенный для отображения взаимодействий между объектами системы по времени. Она представляет собой схематическое изображение того, как объекты взаимодействуют друг с другом, когда выполняются определенные действия или процессы.

Основные элементы диаграммы последовательности:

1. Объекты (или актеры) – это элементы, которые участвуют во взаимодействии. Они представлены вертикальными линиями.

2. Сообщения – это взаимодействия между объектами, которые отображаются в виде стрелок, указывающих направление и порядок обмена информацией.

3. Линия жизни – вертикальная линия, которая представляет собой существование объекта в процессе взаимодействия.

4. Активности – отображают, когда объект выполняет какую-либо задачу (обычно как прямоугольники вдоль линии жизни).

Диаграммы последовательности представлена на рисунках 8 – 13.

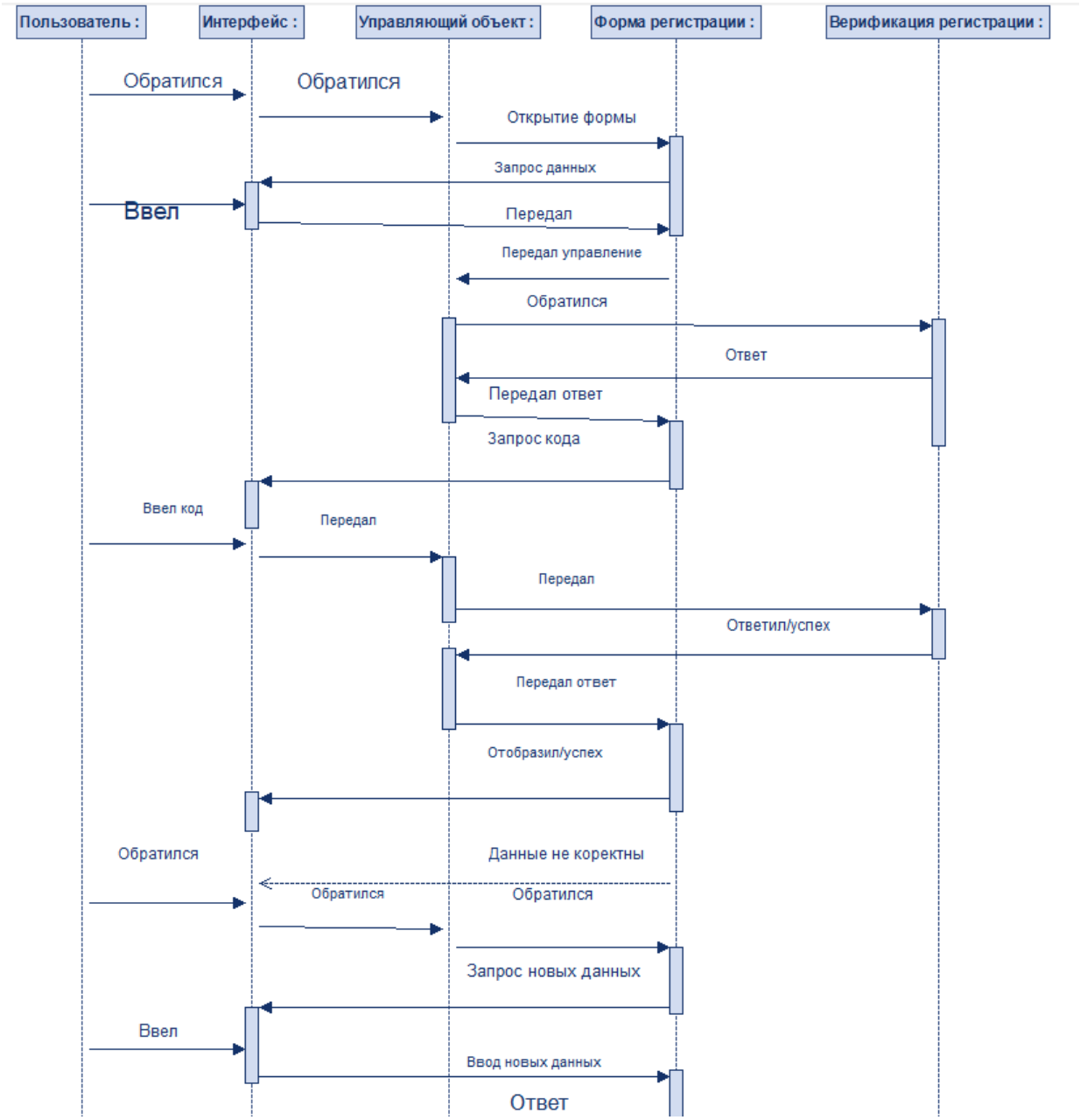

Рисунок 8 – Диаграмма последовательности прецедента «Регистрация».

Схема показывает, как пользователь вводит данные, обращается к системе, получает ответы и в случае ошибки может повторить ввод или запросить новые данные. Основные шаги:

1. Пользователь обращается к интерфейсу.
2. Открывается форма для ввода данных.
3. Пользователь вводит данные и передает их системе.
4. Система отвечает и запрашивает дополнительные данные (если это необходимо).
5. В случае ошибки пользователь может повторить ввод или обратиться за помощью.

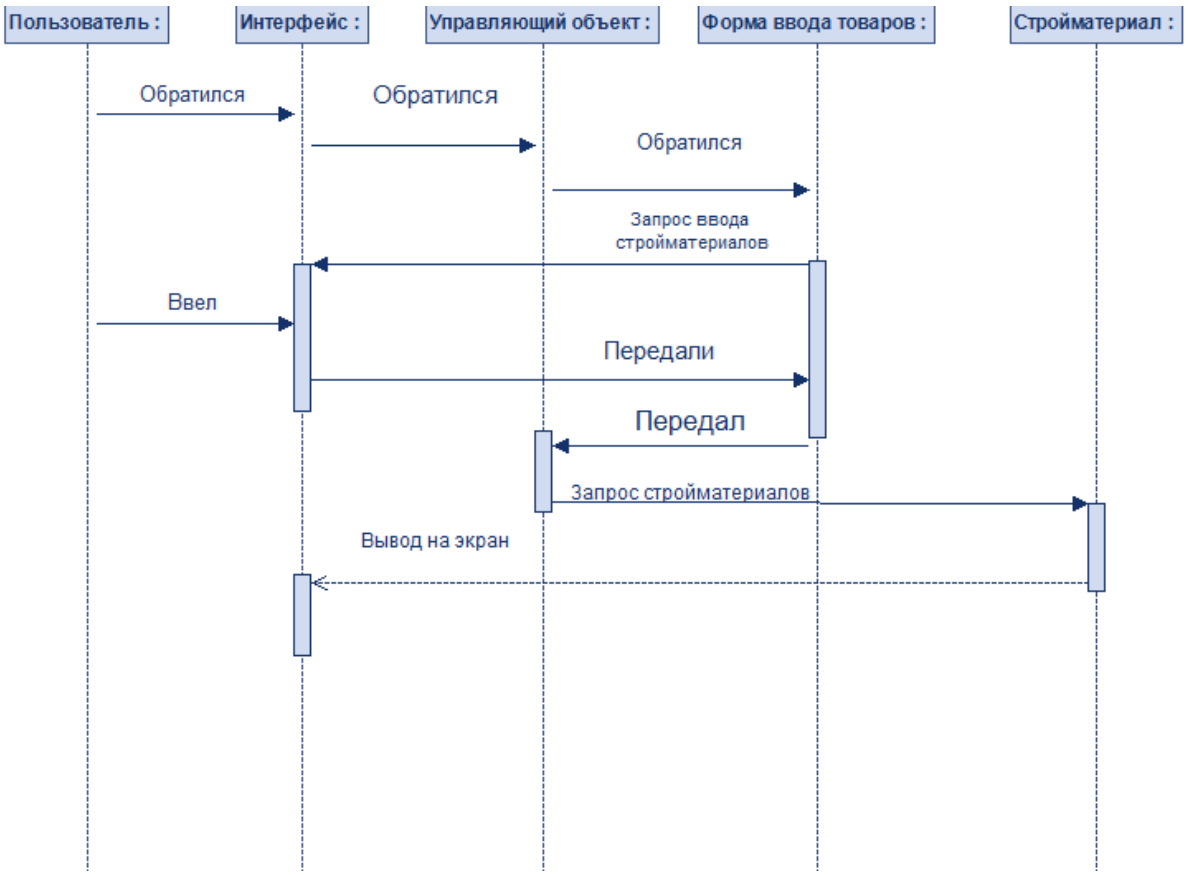

Рисунок 9 – Диаграмма последовательности прецедента «Запрос данных о товарах».

Диаграмма последовательности, которая иллюстрирует взаимодействие между пользователем, интерфейсом и управляющим объектом при вводе строительных материалов.

Вот основные элементы:

1. Пользователь: инициирует обращение.
2. Интерфейс: обрабатывает ввод.
3. Управляющий объект: принимает запросы и передает данные.
4. Строительные материалы: конечный объект, с которым происходит взаимодействие.

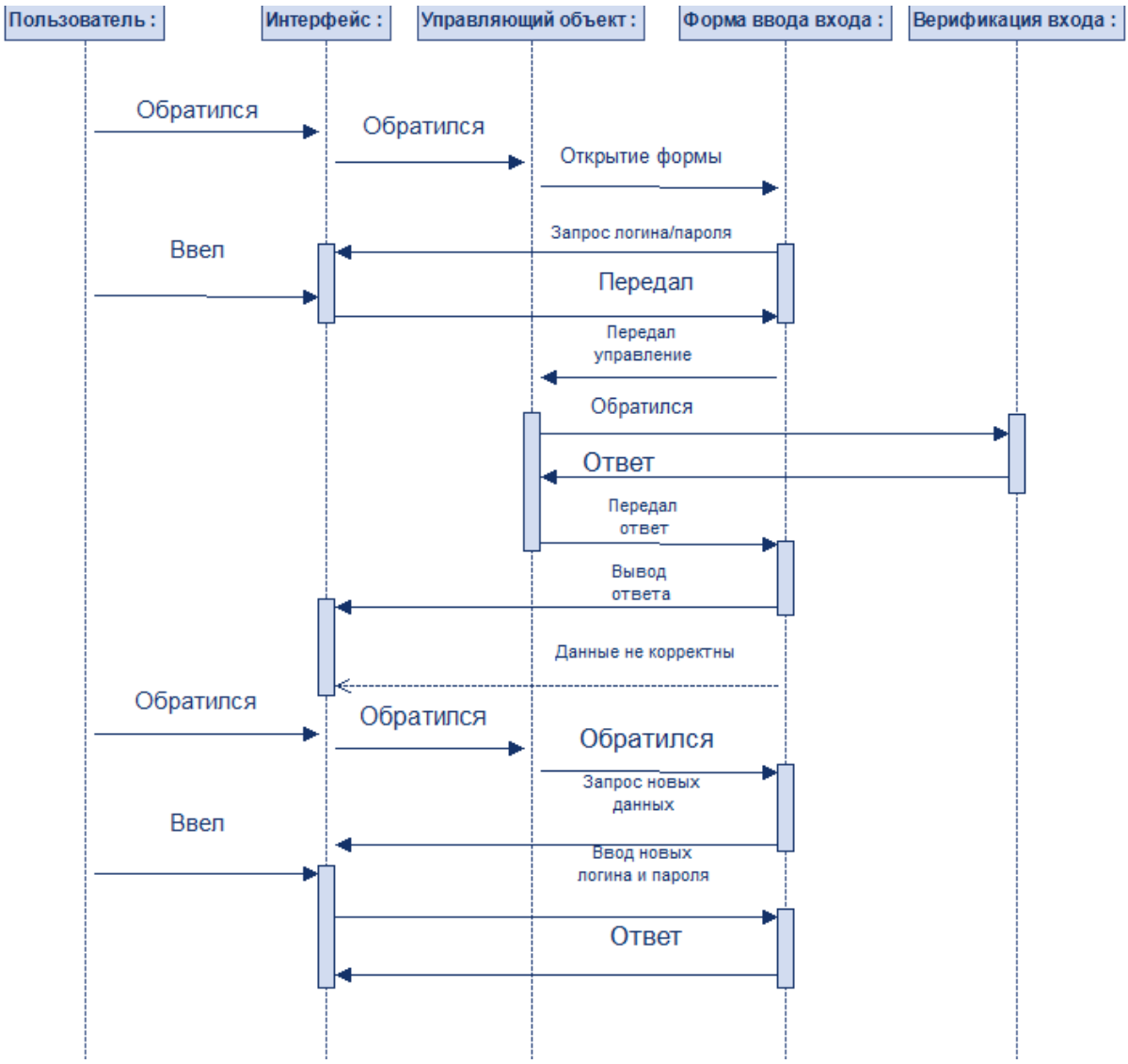

Рисунок 10 – Диаграмма последовательности прецедента «Вход».

Диаграмма последовательности для управления входом пользователя в систему.

Основные этапы:

1. Пользователь обращается к интерфейсу.
2. Интерфейс открывает форму для ввода логина и пароля.
3. Пользователь вводит данные и передает их.
4. Система проверяет данные и возвращает ответ. Если данные корректны, выводится успешный ответ. Если данные некорректны, запрашиваются новые данные.

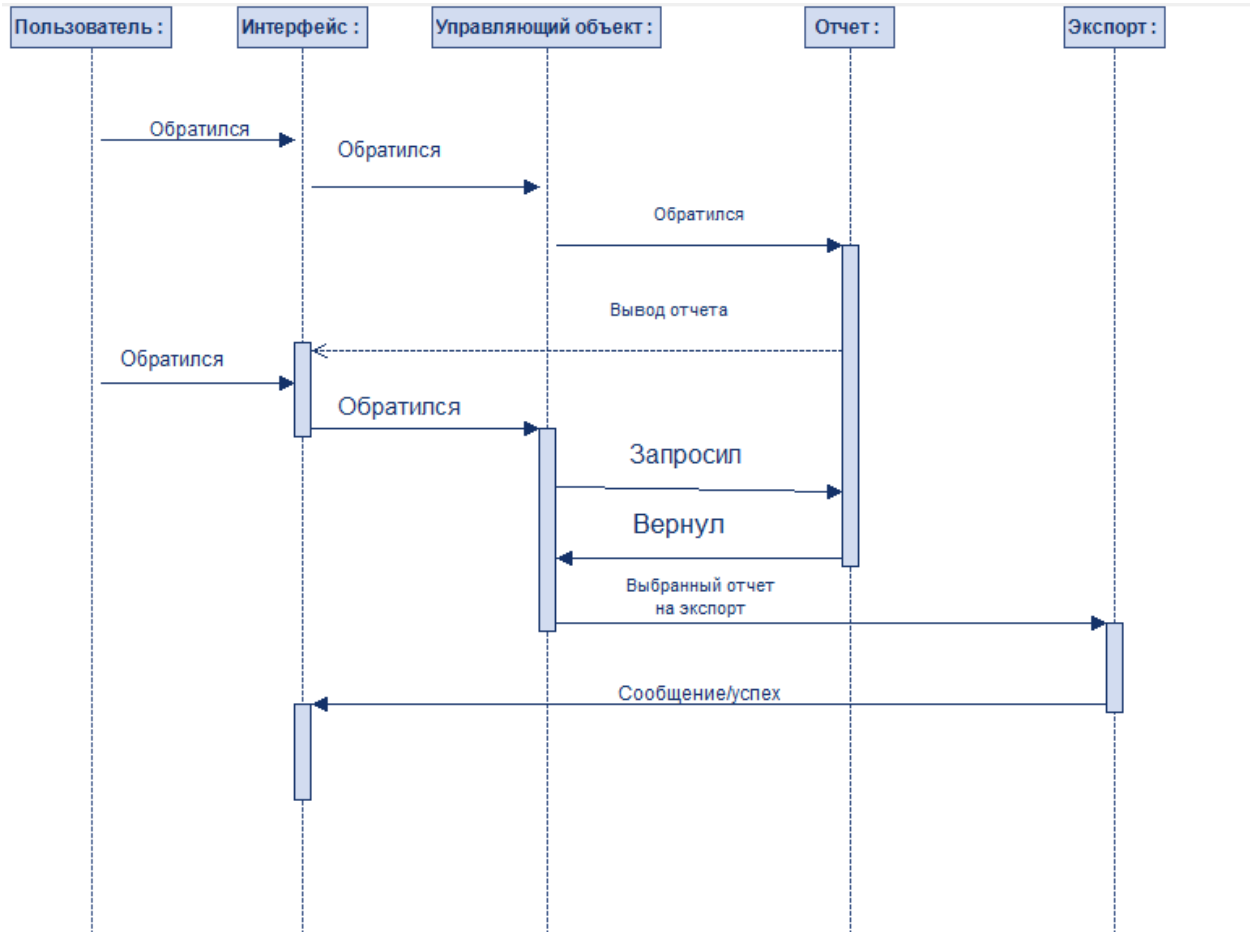

Рисунок 11 – Диаграмма последовательности прецедента «Просмотр отчётов и экспорт».

Диаграмма последовательности, показывающая взаимодействие между пользователем, интерфейсом и управляющим объектом.

Основные этапы, которые можно выделить на диаграмме:

1. Обращение пользователя к интерфейсу.
2. Запрос на вывод отчета от интерфейса к управляющему объекту.
3. Возврат отчета от управляющего объекта к интерфейсу.
4. Выбор отчета на экспорт.
5. Сообщение об успехе.

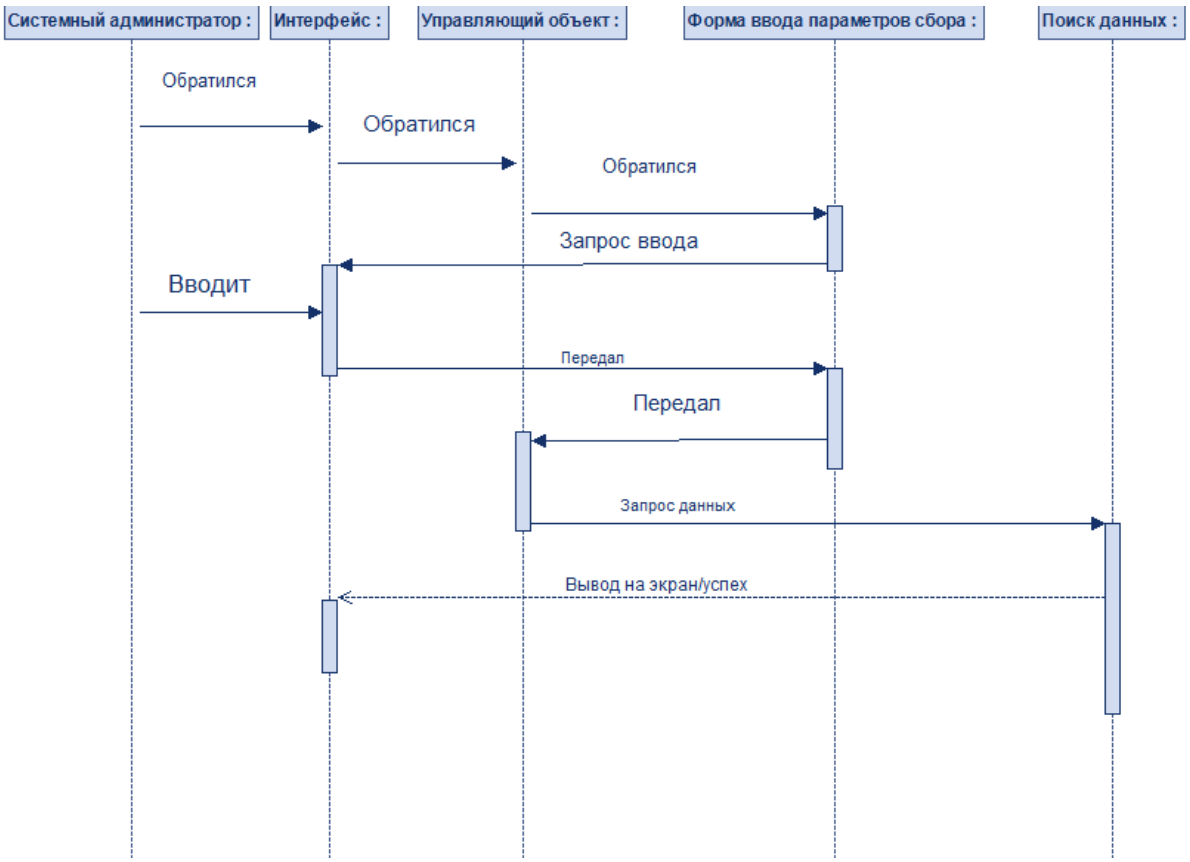

Рисунок 12 – Диаграмма последовательности прецедента «Настройка источников и параметров данных, сбор данных».

Основные этапы взаимодействия на диаграмме последовательности:

1. Запрос от системного администратора к интерфейсу.
2. Отправка запроса на сбор данных от интерфейса к управляющему объекту.
3. Запрос параметров сбора от управляющего объекта к форме ввода.
4. Ввод параметров пользователем через форму.
5. Отправка параметров обратно к управляющему объекту.
6. Поиск данных по заданным параметрам.
7. Возврат найденных данных от управляющего объекта к интерфейсу.
8. Отображение данных пользователю.

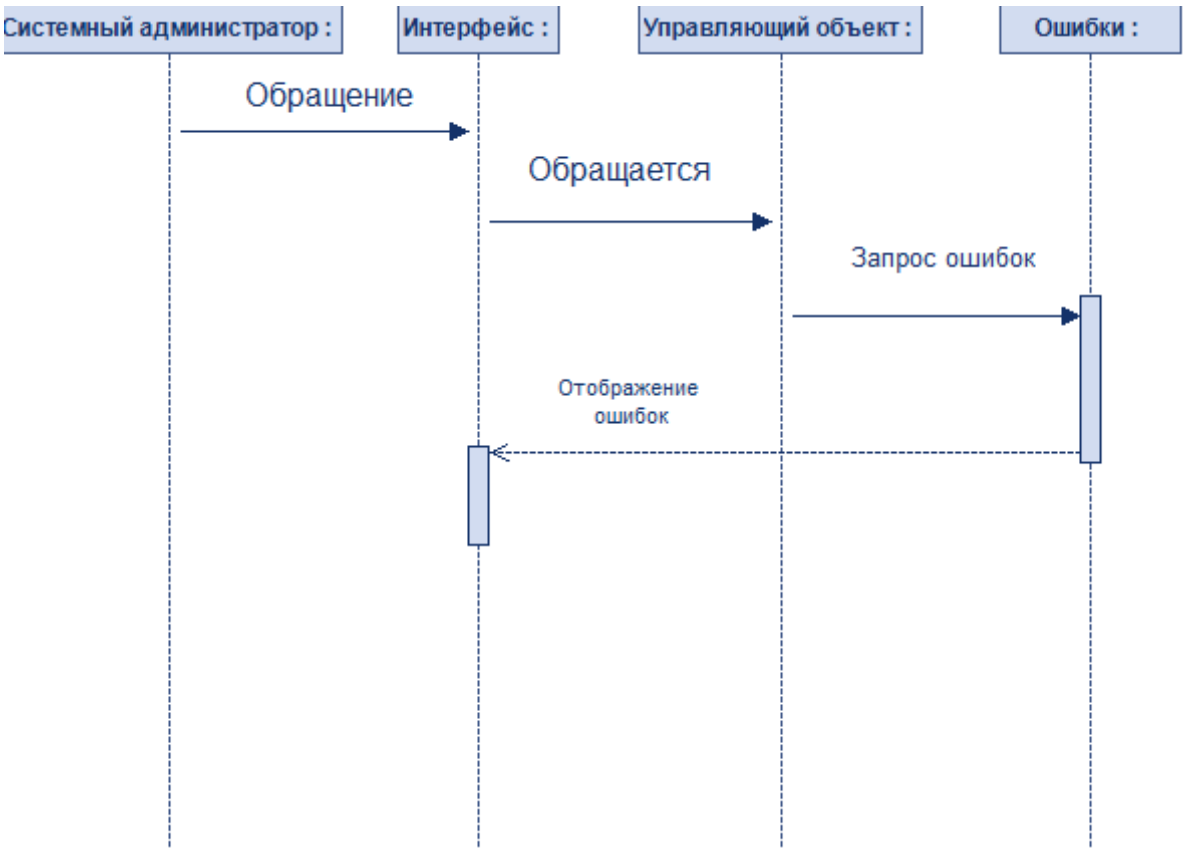

Рисунок 13 – Диаграмма последовательности прецедента «Получение уведомлений о ошибках сбора данных».

На диаграмме показано:

1. Запрос от администратора – администратор инициирует действие через интерфейс.
2. Передача запроса – интерфейс отправляет запрос управляющему объекту.
3. Запрос параметров – управляющий объект запрашивает параметры у формы ввода.
4. Ввод параметров – пользователь вводит данные в форму.
5. Отправка параметров – параметры возвращаются к управляющему объекту.
6. Поиск данных – управляющий объект ищет данные по заданным параметрам.
7. Возврат данных – найденные данные возвращаются интерфейсу.
8. Отображение данных – интерфейс показывает данные пользователю.

3 РАЗРАБОТКА СИСТЕМЫ

3.1 Этапы разработки

Весь процесс разработки системы проведен в соответствии со спиральной моделью с небольшими отклонениями от канонов в угоду эффективности. Каждый этап направлен на решение конкретной задачи, что позволяет минимизировать риски и упростить процесс тестирования и внедрения.

1. Создание макетов интерфейса в *Figma*. На первом этапе разрабатываются детализированные макеты пользовательского интерфейса. Это позволяет определить внешний вид системы, структуру страниц, логические взаимосвязи элементов и обеспечить удобство взаимодействия пользователей с системой. Макеты согласуются с требованиями к функционалу и дизайну.
2. Разработка шаблонного кода для всей системы. На этом этапе создается базовая основа приложения:
   2.1. Реализуются базовые функции, такие как регистрация, вход, распределение ролей (пользователь и системный администратор).
   2.2. Настраивается серверная часть с подключением клиентской стороны через *API*.
   2.3. Выполняется настройка портов и подключения базы данных.
   2.4. Реализуются механизмы безопасности, включая шифрование данных в базе с использованием алгоритма *AES*-256 и настройка защищенного соединения по протоколу *HTTPS*.
3. Разработка и тестирование первой страницы клиентского приложения:
   3.1. Верстается одна из страниц клиентского приложения на основе ранее созданных макетов.
   3.2. Разрабатывается серверная логика для обработки данных этой страницы.
   3.3. Клиентская и серверная части соединяются для обеспечения функциональности страницы.
   3.4. Выполняется тестирование: модульные тесты для проверки отдельных функций, интеграционные тесты для проверки взаимодействия клиентской и серверной частей.
4. Итеративная разработка остальных страниц. Каждая последующая страница разрабатывается по аналогии с прошлым пунктом: сначала верстается интерфейс страницы, программируется серверная логика для нее, выполняется интеграция клиентской и серверной частей, проводится тестирование всех функций страницы.
5. Разработка блока обработки исходных данных. Когда все основные страницы клиентской и серверной частей завершены, переходят к созданию модели обработки исходных данных. На этом этапе:
   5.1. Разрабатывается функционал для обработки полученных данных в формате *HTML, JSON* или *XML*.

5.2.Модульная логика обработки покрывается тестами для проверки корректности работы.

6. Интеграция модели обработки данных с серверной частью. После создания модели обработки данных она подключается к серверной части. Выполняется интеграционное тестирование для проверки взаимодействия всех модулей.

7. Тестирование всей системы. Финальный этап включает комплексное тестирование системы:

7.1. Проверяется функциональность всех компонентов в совокупности вручную.

7.2. Тестируются пользовательские сценарии взаимодействия с системой.

7.3. Проводится нагрузочное тестирование для проверки производительности системы.

7.4. Исправляются выявленные недочеты.

Таким образом, разработка системы ведется поэтапно, с последовательным добавлением функциональности и тщательной проверкой на каждом этапе. Такой подход обеспечивает стабильность, минимизацию ошибок и высокое качество финального продукта.

3.2 Средства разработки

Для успешной реализации системы будет использован определенный стек технологий и аппаратное обеспечение, подобранное для обеспечения удобства разработки, тестирования и последующего развертывания системы. Эти средства выбраны исходя из требований к функциональности и производительности системы.

Аппаратное обеспечение:

1. Для разработки:

1.1. Компьютер с характеристиками, достаточными для комфортной работы с несколькими средами разработки, браузерами и базами данных. Рекомендуемые параметры:

1.1.1. Процессор: не менее 6 ядер (*Intel Core i*5 или *AMD Ryzen* 5 и выше).

1.1.2. Оперативная память: минимум 8 ГБ (рекомендуется 16 ГБ для оптимальной работы).

1.1.3. Накопитель: *SSD* объемом от 256 ГБ для быстрого доступа к файлам проекта.

1.1.4. Графический адаптер: интегрированный или дискретный для работы с графическими интерфейсами.

1.2. Доступ к высокоскоростному интернету для взаимодействия с удаленными репозиториями, библиотеками и *API*.

2. Для рабочей версии:

2.1. Серверное оборудование или облачная платформа, поддерживающая размещение веб-приложений и баз данных.

2.2. Минимальные требования для сервера:

2.2.1. Процессор: от 2 ядер.

2.2.2. Оперативная память: 4 ГБ (рекомендуется 8 ГБ для стабильной работы под нагрузкой).

2.2.3. Накопитель: *SSD* от 100 ГБ.

2.2.4. Обязательное наличие подключения к интернету с низкой задержкой и стабильной пропускной способностью для обработки запросов пользователей.

Стек технологий:

1. Клиентская часть:

1.1. *HTML5* и *CSS*3 – для создания структуры и оформления пользовательского интерфейса.

1.2. *JavaScript* – основной язык для работы с клиентской логикой.

1.3. *React.js* – библиотека для построения динамических пользовательских интерфейсов, обеспечения удобства разработки и поддержки компонентной архитектуры.

1.4. *TypeScript* – надстройка над JavaScript для добавления строгой типизации, что уменьшает вероятность ошибок при разработке и упрощает поддержку кода.

2. Серверная часть:

2.1. *PHP* – основной язык программирования для серверной логики.

2.2. *Laravel* – фреймворк на *PHP*, который обеспечивает удобную структуру, маршрутизацию, безопасность и встроенные механизмы работы с базами данных.

2.3. *Node.js* – для реализации вспомогательных серверных задач, например, настройки прокси или обработки некоторых нестандартных запросов.

3. Модуль обработки данных для веб-скрейпинга – *Python* 3 – язык программирования для создания модуля обработки данных, благодаря его гибкости и наличию мощных библиотек (*Beautiful Soup, Requests, lxml*) для работы с *HTML, JSON* и *XML*.

4. База данных – *MySQL* 9. Это реляционная база данных для хранения структурированных данных системы. Она выбрана за ее производительность, надежность и широкую поддержку сообществом.

5. Серверное окружение:

5.1. *Apache*2 – тестовый сервер для отладки приложения.

5.2. *Nginx* – сервер для прокси и балансировки нагрузки на этапе релиза.

6. Редакторы кода и администрирования баз данных:

6.1. *Visual Studio Code* – основной редактор кода, благодаря удобному интерфейсу, поддержке расширений для всех используемых языков программирования и возможностей для работы с системой контроля версий (Git).

6.2. *MySQL Workbench* – инструмент для проектирования, настройки и управления базой данных.

7. Контроль версий – *Git*.

Среда разработки. Проект разрабатывается в среде *Linux* (дистрибутив *Ubuntu* 24.04 *LTS*), так как этот выбор соответствует будущему окружению для развертывания системы. Преимущества Ubuntu включают:

1. Высокую стабильность и производительность в серверной среде.
2. Широкую поддержку инструментов разработки и работы с базами данных.
3. Наличие встроенных средств для настройки сетевых сервисов и безопасности.

Разработка будет вестись на тестовом сервере с использованием описанного стека технологий и аппаратного обеспечения. Такой подход обеспечит стабильность работы системы, упрощение процесса отладки и возможность дальнейшего масштабирования проекта.

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной работы был разработан проект системы для поиска, анализа и управления данными о строительных материалах. В основу проекта положена архитектура *MVC*, которая обеспечила разделение логики, представления данных и пользовательского интерфейса. Такой подход позволил достичь высокой модульности системы и упростить процесс ее сопровождения и масштабирования.

Система включает две основные подсистемы:

1. Пользовательскую подсистему, предназначенную для поиска строительных материалов, анализа данных и их экспорта. Пользователь может гибко настроить параметры поиска, сортировать результаты, а также строить графики изменения цен на материалы.

2. Административную подсистему, позволяющую управлять настройками парсера, контролировать источники данных и логировать все действия в системе.

Основные функциональные возможности системы:

Настройка гибких параметров поиска для пользователей.

Вывод результатов в удобной форме с возможностью экспорта данных.

Автоматизированный сбор информации с использованием прокси и различных методов веб-скрейпинга (*HTML*, *API*).

Поддержка безопасности за счет использования шифрования данных (*AES*-256) и передачи по защищенному протоколу *HTTPS*.

Масштабируемость системы для добавления новых функций без значительных изменений архитектуры.

Разработанная система соответствует заявленным функциональным и нефункциональным требованиям, включая производительность, безопасность и совместимость. Она успешно выполняет задачу сбора, обработки и анализа данных, предоставляя пользователям простой и удобный интерфейс, а администраторам — гибкие инструменты для управления.

В рамках данной работы было показано, что архитектура системы позволяет эффективно решать поставленные задачи, обеспечивая масштабируемость и гибкость. Разработанный проект может стать основой для дальнейшего развития, например интеграции с внешними сервисами или добавления аналитических модулей.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Дакетт, Дж. *HTML* и *CSS*. Разработка и дизайн веб-сайтов. — Москва: Вильямс, 2014. — 512 с.

2. Дакетт, Дж. *JavaScript* и *jQuery*. Интерактивная веб-разработка. — Москва: Вильямс, 2015. — 640 с.

3. Фримен, Э., Робсон, Э. Изучаем программирование на *JavaScript*. — Санкт-Петербург: Питер, 2017. — 640 с.

4. Лутц, М. Изучаем *Python*. — Москва: Вильямс, 2021. — 880 с.

5. Дейт, К. Дж. Введение в системы баз данных. — Москва: Вильямс, 2020. — 1024 с.

6. Бэнкс, К., Парсонс, Э. *Node.js*. В разработке веб-приложений. — Санкт-Петербург: Питер, 2019. — 352 с.

7. Макфарланд, Д. *CSS*. Подробное руководство. — Москва: Вильямс, 2020. — 720 с.

8. Шарп, Б. *PHP* и *MySQL*. Разработка современных веб-приложений. — Москва: Эксмо, 2018. — 592 с.