

Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерного проектирования
Кафедра экономической информатики
Дисциплина «Средства и технологии анализа и разработки информационных
систем»

«К ЗАЩИТЕ ДОПУСТИТЬ»

Руководитель курсового проекта
кандидат технических наук, доцент

_____. В.А. Федосенко

_____.2022

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему:

**«РАЗРАБОТКА АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ ПОДДЕРЖКИ
ВЗАИМОДЕЙСТВИЯ ПОТРЕБИТЕЛЕЙ С ПОСТАВЩИКАМИ ЭНЕР-
ГОРЕСУРСОВ И УЧЁТА СДЕЛОК»**

БГУИР КП 1-40 05 01-10 015 ПЗ

Выполнил студент группы 914301

ЯРМОЛИК Максим Андреевич

(подпись студента)

Курсовой проект представлен на
проверку 04.05.2022

(подпись студента)

Минск 2022

СОДЕРЖАНИЕ

Введение.....	6
1 Анализ и моделирование предметной области программного средства...	7
1.1 Описание предметной области.....	7
1.2 Разработка функциональной модели предметной области.....	11
1.3 Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований.....	16
1.4 Разработка информационной модели предметной области.....	17
1.5 UML-модели представления программного средства и их описание..	19
2 Проектирование и конструирование программного средства.....	24
2.1 Постановка задачи.....	24
2.2 Архитектурные решения.....	24
2.3 Описание алгоритмов, реализующих ключевую бизнес-логику разрабатываемого программного средства.....	25
2.4 Проектирование пользовательского интерфейса.....	27
3 Тестирование и проверка работоспособности программного средства.....	30
4 Инструкция по развертыванию приложения и сквозной тестовый пример.....	31
Заключение.....	32
Список использованных источников.....	33
Приложение А (обязательное) Антиплагиат.....	34
Приложение Б (обязательное) Листинг алгоритмов, реализующих бизнес- логику.....	35
Приложение В (обязательное) Листинг скрипта генерации базы данных....	43

ВВЕДЕНИЕ

Во многих странах предприятия энергетического сектора экономики переживают период реформирования. Происходящие процессы слияния, поглощения и изменения структуры управления, границ сферы деятельности и территориального присутствия заставляют многие бывшие монополии искать для себя новые модели создания стоимости. Неизбежно меняются задачи компаний и их бизнес-процессы. Формируются рынки предоставления коммунальных услуг. Внедряются рыночные механизмы. Требуются технологические изменения, отвечающие современным потребностям развития отрасли. Хотя все эти изменения отличаются в зависимости от местоположения и вида деятельности энергокомпаний, инновации неизбежно ведут к преобразованию всей сферы коммунальных услуг.

Конкурентоспособность важна для компании. За каждой разработкой стоит увеличение рабочей нагрузки, ответственности и рисков, а это значит, что она должна постоянно двигаться вперед, ища новые методы оптимизации работы и автоматизации управления.

В рамках курсового проекта будет построено примитивное веб-приложение с базовой функциональностью, которое позволит клиентам взаимодействовать с поставщиками энергоресурсов.

Актуальность: актуальность темы курсового проекта заключается в том что в современных условиях, для ведения бизнеса нужно использовать такие методы общения, которые будут позволять клиентам и поставщикам быстро находить друг друга, а также быстро, легко и эффективно обмениваться встречными предложениями, изменениями условий поставок и прочим.

Цель: спроектировать и запрограммировать клиент-серверное веб-приложение для обеспечения и поддержки взаимодействия потребителей с поставщиками энергоресурсов.

Задачи:

- исследование предметной области;
- реализация клиент-серверного приложения;
- создание понятного интерфейса для взаимодействия с заявками и предложениями;
- реализация базового функционала для работы с заявками и предложениями.

1 АНАЛИЗ И МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ ПРОГРАММНОГО СРЕДСТВА

1.1 Описание предметной области

Для энергокомпаний ключевыми преследуемыми целями развития технологий являются:

- снижение потерь энергоресурсов;
- повышение своевременности и полноты оплаты за потребляемые энергоресурсы;
- управление неравномерностью графика электрической нагрузки;
- повышение эффективности управления активами энергокомпаний;
- повышение качества интеграции объектов возобновляемой генерации и распределенной генерации в энергосистему;
- повышение надежности функционирования энергосистемы в случае возникновения аварийных ситуаций;
- повышение визуализации работы объектов энергетической инфраструктуры.

Электроэнергетические компании проявляют значительный интерес к данному сегменту, потому что именно здесь происходит непосредственное взаимодействие с клиентами. Здесь же компании могут стать участниками либо регулируемого, либо свободного рынка. Хотя традиционно в центре внимания данного сегмента находились продажи, сейчас здесь развивается направление более широкого предложения продуктов и услуг, учитывающих специфические потребности клиента.

Чтобы выжить в этом переменчивом мире, нефтегазовым компаниям придется как следует постараться, чтобы удовлетворить запрос потребителей на более удобное и качественное, «подключенное» и персонализированное взаимодействие. Трансформация энергетического рынка в сочетании с активной цифровизацией всех областей жизни способствует изменению поведения потребителей, их системы ценностей и убеждений. Сегодня взаимоотношения между поставщиками энергии и потребителями полностью меняются—от осознанного подхода к защите окружающей среды до перехода на принципиально новые методы работы. Однако, в то время как энергетические компании полностью меняют свой подход к работе, ориентируясь главным образом на взаимодействие с потребителями, потребители в свою очередь мало заинтересованы в каком бы то ни было взаимодействии с поставщиками энергии или топлива. Это представляет собой определенный риск для нефтегазовых компаний,

поскольку многие из них пытаются полностью трансформировать свою работу, чтобы стать поставщиками e-Mobility или даже поставщиками электроэнергии.

Цифровые технологии меняют все —подход к принятию решений в бизнесе, методы повышения производительности, потребительский опыт, транспортировку и способы доставки энергии, а также мониторинг потребления и социальную активность. Повсеместная «подключенность» формирует новую цифровую реальность, в которой компании могут использовать оперативно-аналитические данные из множества источников, применяя такие технологии, как искусственный интеллект, для преобразования ключевых функций, оптимизации работы персонала, сокращения эксплуатационных затрат, повышения скорости работы и внедрения интеллектуальных систем. Лишь треть нефтяных компаний заключает партнерские соглашения с другими игроками этого рынка, чтобы обеспечить поддержку новых моделей бизнеса —это значительно меньший процент, чем в других отраслях. Цифровые технологии меняют конкурентный ландшафт в энергетической отрасли, который становится все более сложным и неоднозначным. На каждом этапе цепочки создания стоимости появляются новые игроки, которые предлагают потребителям энергии более совершенные и эффективные альтернативы. К таким игрокам относятся компании прямого инвестирования, специалисты по рынкам и ресурсам, независимые нишевые игроки, которые предлагают более выгодное управление традиционными ресурсами и сервисами. Кроме них есть еще коммунальные службы, автоконцерны и технологические компании, которые разрабатывают инновационные электромобили или беспилотные автомобили или предлагают комплексные решения в области солнечной энергетики, транспорта и коммуникаций, призванные обеспечить постоянно растущую потребность «подключенного» транспорта и жилых домов в электроэнергии. Учитывая такие изменения в конкурентной среде, участникам рынка придется объединить усилия и сформировать единую экосистему. Тем не менее, в сравнении с другими сегментами рынка энергетический сектор в целом по-прежнему слабо использует возможности инновационных цифровых технологий и экосистем перспективных стартапов. Фактически исследование Accenture Trust свидетельствует о том, что лишь треть нефтегазовых компаний заключает партнерские соглашения с другими игроками этого рынка, чтобы обеспечить поддержку новых моделей бизнеса —это значительно меньший процент, чем в других отраслях.

Учитывая, что конкурентный ландшафт становится все более сложным и неоднозначным, а отраслевая конвергенция возрастает, появляется потребность в трансформации нефтяных и газовых компаний. Что касается поставок, таким компаниям приходится иметь дело с более многочисленными ресурсами

и разнообразными источниками энергии, которые предлагают конкуренты, с требованиями к разработке тех или иных ресурсов, которые предъявляют инвесторы, а также с более низкими в целом, но при этом гораздо более неустойчивыми ценами.

Что касается спроса, то здесь добывающим предприятиям придется отказаться от модели бизнеса, ориентированной на продукцию, и переключиться в первую очередь на потребителей, формируя такие предложения, которые удовлетворяли бы запросам на экологичность и энергоэффективность, а также решали бы проблему энергетической нищеты, при этом обеспечивая существующий спрос на нефть и природный газ.

До сегодняшнего дня трансформация ключевых функций нефтегазовой компании - тех процессов преобразования, которые протекают при поддержке цифровых технологий - носила эпизодический характер, но не применялась комплексно. Некоторые технологии, которые раньше использовались лишь для решения узкоспециализированных задач, сегодня уже широко применяются в основных направлениях бизнеса, и это уже хорошее начало. По данным отчета Accenture Energy Technology Vision за 2019 год, 80% руководителей добывающих и 73% руководителей перерабатывающих компаний признает, что цифровые технологии прочно вошли в жизнь их компаний и оказали существенное влияние на бизнес.

Нефтегазовая отрасль по-прежнему активно интегрирует новые платформы и цифровые возможности, но уже начинает осознавать, насколько важно сформировать экосистемы и заключить стратегические партнерские соглашения, чтобы максимально эффективно использовать весь потенциал цифровых технологий. По данным последнего отчета Accenture Digital Fuels Retail, основной причиной, по которой почти 50% компаний заключили стратегические партнерские соглашения, стало желание более эффективно использовать технологии и цифровые инновации. В 2019 году, например, Accenture и SAP стали партнерами более чем для 25 нефтяных компаний, начав совместную разработку стандартов для специализированного индустриального решения, используемого при планировании ресурсов предприятия (ERP) в общедоступном облаке.

По мере того, как нефтегазовые компании все чаще принимают решения на основе данных, те инновационные цифровые инструменты и новые технологии, к которым они получают доступ, дают им реальную возможность пройти трансформацию и начать менять ситуацию во всей отрасли. Ключевыми с этой точки зрения являются следующие технологии: распределенный реестр блокчейн, искусственный интеллект (ИИ), расширенная реальность (XR) и квантовые вычисления. Accenture называет это квартетом важнейших

технологий РИРК. Все они эффективны и по отдельности, однако вместе они способны изменить работу целых отраслей, включая и энергетику.

Хотя рост отрасли все больше зависит от развивающихся рынков, отдельные зрелые рынки все равно требуют инвестиций - например, рынок в США, где текущая ситуация дает ряд ценных возможностей для новых операций с капиталом.

Чтобы максимально эффективно использовать существующие физические активы, рыночные позиции и взаимоотношения с потребителями, нефтегазовым компаниям необходимо применять подключенные платформы, ориентируясь в первую очередь на потребителей и бизнес-результат, чтобы повысить производительность, скорость и оперативность реагирования при внедрении инноваций. Первостепенной задачей является совершенствование навыков персонала за счет внедрения ИИ и новых технологий (как человеческих, так и машинных) для трансформации основных направлений бизнеса и формирования новых потоков прибыли. Такая трансформация лишь выиграет при наличии общей цели, ориентированной на достижение устойчивого развития, преобразования всей отрасли и изменения существующих подходов к работе.

В будущем, вероятнее всего, для успешного роста потребуются гораздо более широкий спектр взаимоотношений, чем сегодняшние договоренности между поставщиками и их ближайшими партнерами. В условиях специализированного рынка с жесткой конкуренцией в выигрыше скорее всего окажутся те, кто сумеет наиболее эффективно использовать собственные возможности и возможности партнеров по экосистеме на всех этапах цепочки создания стоимости во всех направлениях бизнеса.

Использование цифровых технологий (для создания всестороннего представления потребителей и повышения качества взаимодействия с ними) поможет привлечь новых и удержать существующих клиентов. Расширенные аналитические данные по клиентам и интеллектуальные платежные системы позволят принимать более взвешенные решения и продавать больше дополнительных продуктов и услуг (связанных или не связанных с энергетикой), чтобы повысить уровень удержания клиентов и увеличить прибыль.

Кроме того, иммерсивные технологии, такие как расширенная реальность, позволяют нефтегазовым компаниям изменить подход к взаимодействию с потребителями в процессе перехода в розничный сегмент (например, газо- и электроснабжение розничных потребителей, а также обслуживание инфраструктуры электротранспорта).

Переходный период, который сегодня переживает энергетическая отрасль, является одним из самых сложных среди всех остальных отраслей - не

только с точки зрения внедрения цифровых технологий, но и в плане обеспечения устойчивого развития и постепенного отказа от углеводородов.

По мере того как нефтегазовые компании постепенно осознают преимущества цифровых технологий и предоставляют своим сотрудникам инструменты и возможности для реализации инновационных программ, им приходится менять операционную модель, характерную для традиционных нефтяных компаний. Энергетическая компания нового типа будет выглядеть совершенно иначе. Она станет руководствоваться более глобальными целями, уделять гораздо больше внимания инновациям и работать в других условиях - более строгих к сокращению затрат и повышению производительности, совершенствования культуры взаимодействия - стремясь успешно достичь поставленных целей в области устойчивого развития. В результате нефтегазовые компании смогут увеличить свою прибыль, повысить конкурентоспособность и лучше подготовиться к переходу на альтернативные источники энергии.

1.2 Разработка функциональной модели предметной области

Методология функционального моделирования IDEF0 – это технология описания системы в целом как множества взаимозависимых действий, или функций. IDEF0 используется для создания функциональной модели, отображающей структуру и функции системы, а также потоки информации и материальных объектов, связывающие эти функции.

IDEF0 — методология функционального моделирования и графическая нотация, предназначенная для формализации и описания бизнес-процессов.

В 1981 году, департамент BBC США разрабатывал программу автоматизации промышленных предприятий, которая сокращённо называлась ICAM (Integrated Computer Aided Manufacturing). Для успешного хода проекта необходимо было уметь моделировать автоматизированное предприятие всем участникам параллельно. Специально для этих целей был разработан набор стандартов IDEF (ICAM DEFinition).

Одним из стандартов набора являлась нотация функционального моделирования под кодовым названием IDEF0, которая слегка видоизменялась с ходом времени, и спецификация для последней на данный момент версии была выпущена в декабре 1993 года.

Двумя наиболее важными компонентами, из которых строятся диаграммы IDEF0, являются бизнес-функции или работы (представленные на диаграммах в виде прямоугольников) и данные и объекты (изображаемые в виде стрелок), связывающие между собой работы. При этом стрелки, в зависимости от того в какую грань прямоугольника работы они входят или из какой грани выходят, делятся на пять видов:

- Стрелки входа (входят в левую грань работы) - изображают данные или объекты, изменяемые в ходе выполнения работы.
- Стрелки управления (входят в верхнюю грань работы) - изображают правила и ограничения, согласно которым выполняется работа.
- Стрелки выхода (выходят из правой грани работы) - изображают данные или объекты, появляющиеся в результате выполнения работы.
- Стрелки механизма (входят в нижнюю грань работы) - изображают ресурсы, необходимые для выполнения работы, но не изменяющиеся в процессе работы (например, оборудование, людские ресурсы)
- Стрелки вызова (выходят из нижней грани работы) - изображают связи между разными диаграммами или моделями, указывая на некоторую диаграмму, где данная работа рассмотрена более подробно.

Первая диаграмма в иерархии диаграмм IDEF0 всегда изображает функционирование системы в целом. Такие диаграммы называются контекстными. В контекст входит описание цели моделирования, области (описания того, что будет рассматриваться как компонент системы, а что как внешнее воздействие) и точки зрения (позиции, с которой будет строиться модель). Обычно в качестве точки зрения выбирается точка зрения лица или объекта, ответственного за работу моделируемой системы в целом.

После того как контекст описан, проводится построение следующих диаграмм в иерархии. Каждая последующая диаграмма является более подробным описанием (декомпозицией) одной из работ на вышестоящей диаграмме. Пример декомпозиции контекстной работы показан далее.

При построении IDEF0 имена функций должны быть глаголами или глагольными оборотами.

Наиболее важные свойства объекта выявляются на верхнем уровне иерархии, по мере декомпозиции функции верхнего уровня и разбиения ее на подфункции эти свойства уточняются. Каждая подфункция декомпозируется на элементы следующего уровня. Декомпозиция происходит до тех пор, пока не будет получена релевантная структура, позволяющая ответить на вопросы, сформулированные в цели моделирования.

Рассмотрим бизнес-процесс «Добавить сотрудника».

Контекстная диаграмма бизнес-процесса верхнего уровня представлена на рисунке 1.1. Входная информация: информация о сотруднике, данные аккаунта. Информация обрабатывается и конвертируется в конечные цели: сотрудник в системе. Управляющий механизм: сотрудник отдела кадров, администратор. Механизм ограничения: трудовой кодекс, инструкция.

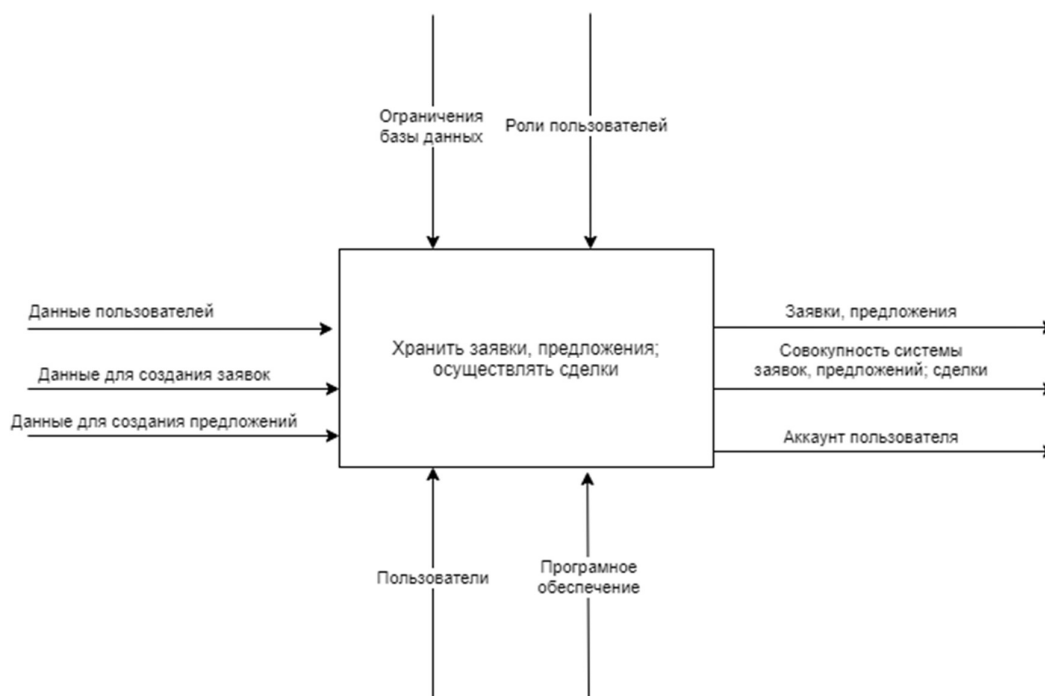


Рисунок 1.1 – Контекстная диаграмма «Хранение заявки, предложения. Осуществление сделки»

Для более подробного описания работы функционального блока выполняется его декомпозиция и моделируется диаграмма первого уровня, которая представлена на рисунке 1.2. На ней представлено четыре функциональных блока декомпозиции.

Данный уровень включает в себя:

- авторизацию;
- создание заявки;
- создание предложения;
- заключение сделки.

Конечные цели системы, получаемые в результате выполнения процесса: заключение сделки.

Далее следует декомпозировать процесс «Авторизировать», результат показан на рисунке 1.3.

На данном уровне выделены следующие подпроцессы: регистрация и вход. В зависимости от того, зарегистрирован ли уже пользователь в приложении будет выбран тот или иной вариант.

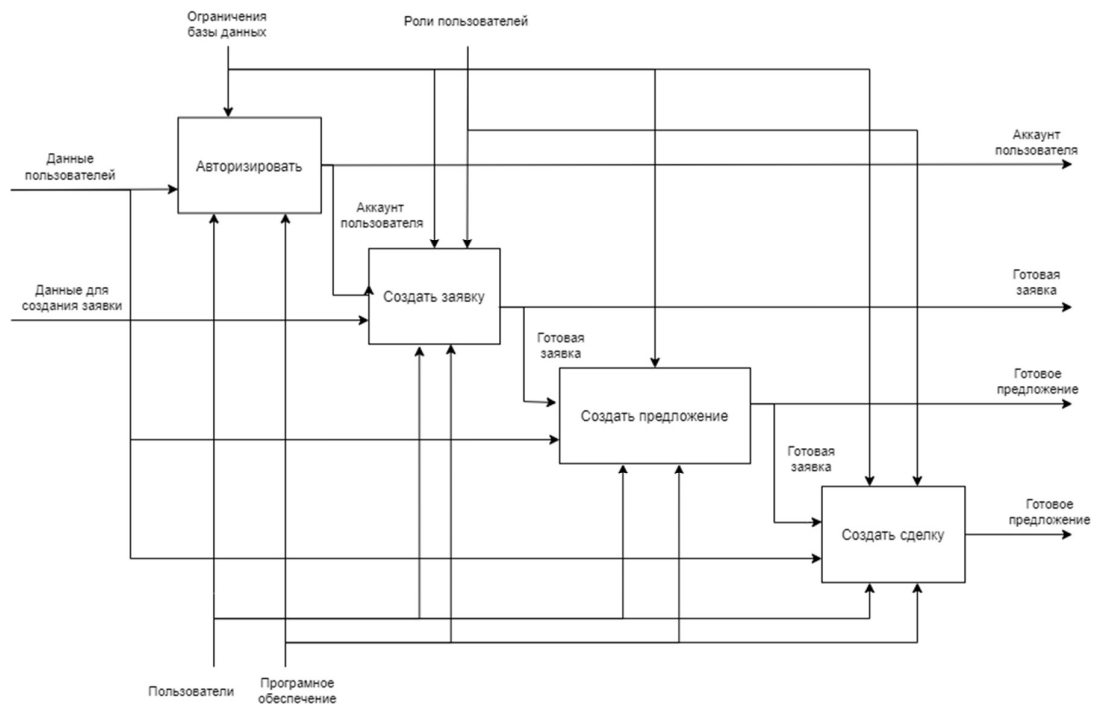


Рисунок 1.2 – Декомпозиция блока контекстной диаграммы

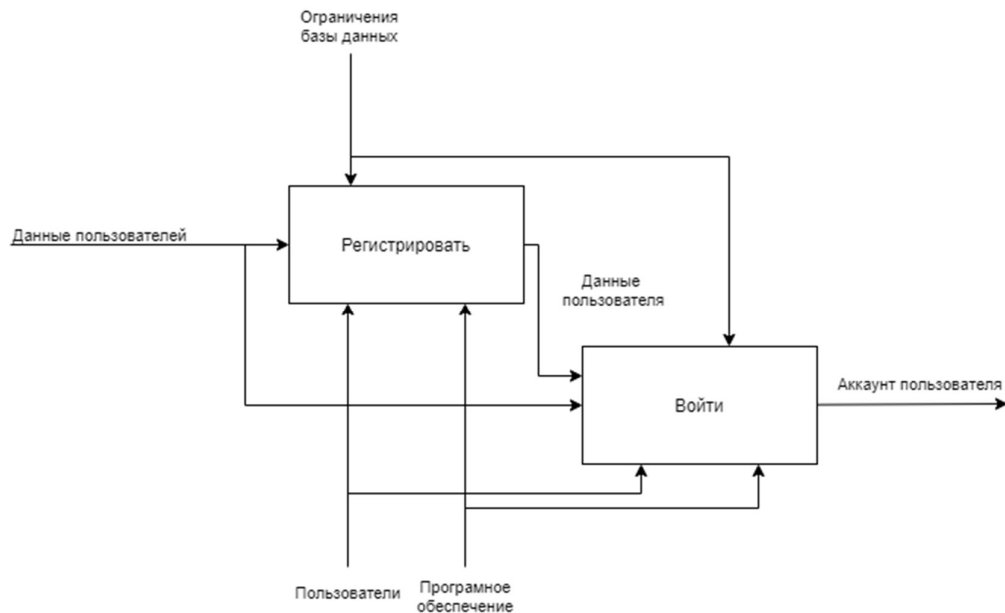


Рисунок 1.3 – Декомпозиция блока «Авторизовать»

После последовательного выполнения всех подпроцессов будет достигнута цель процесса этого уровня: роль. В системе может быть две роли: покупатель и поставщик. В зависимости от роли пользователю доступны разные функции приложения.

На следующем этапе разработки модели выполняется декомпозиция подпроцесса «Создание заявки», показанная на рисунке 1.4.



Рисунок 1.4 – Декомпозиция блока «Создание заявки»

Данный уровень состоит из одного процесса – заполнение данных заявки. На выходе данного уровня в системе появляется заявка.

На следующем этапе разработки модели выполняется декомпозиция подпроцесса «Создание предложения», показанная на рисунке 1.5.



Рисунок 1.5 – Декомпозиция блока «Создание предложения»

Данный уровень также состоит из одного процесса – «Заполнение информации предложения».

Результатом выполнения данного блока будет являться созданное предложение на заявку потребителя.

На следующем этапе разработки модели выполняется декомпозиция подпроцесса «Создание сделки», показанная на рисунке 1.6.



Рисунок 1.6 – Декомпозиция блока «Создание сделки»

Данный уровень всё также состоит из одного процесса – «Принятие предложения». На выходе данного блока получаем принятое предложение.

По завершения всех основных этапов будет, как уже упоминалось ранее, завершён полный цикл – создание заявки, создание предложения, принятие предложения – в результате которого потребитель и поставщик заключат сделку.

1.3 Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований

Анализ требований являются частью процесса разработки программного обеспечения, которая включает в себя сбор требований к программному обеспечению, систематизацию этих требования, выявление взаимосвязей, а также их документирование.

При разработке и проектировании программного средства будут использованы следующие технологии:

1. Для реализации серверной (back-end) части программного средства – язык программирования Java. Используется фреймворк Spring для создания каркаса приложения, связывания его компонентов.
2. Для реализации клиентской (front-end) части программного средства – язык программирования TypeScript и фреймворк Angular.

3. Для организации хранения данных используется СУБД MySQL.
4. Для разметки используется язык гипертекстовой разметки HTML так как это стандартное, встроенное в Angular средство для разметки страниц.
5. Для стилистического оформления используется SASS (syntactically awesome style sheets) – надстройка над языком CSS, которая добавляет множество удобств разработки, а в конечном итоге компилируется в CSS.
6. Для проектирования программного средства и его графического описания используется язык UML, так как позволяет наиболее гибко описывать систему.

Диаграмма вариантов использования для данного курсового проекта представлена на рисунке 1.7.

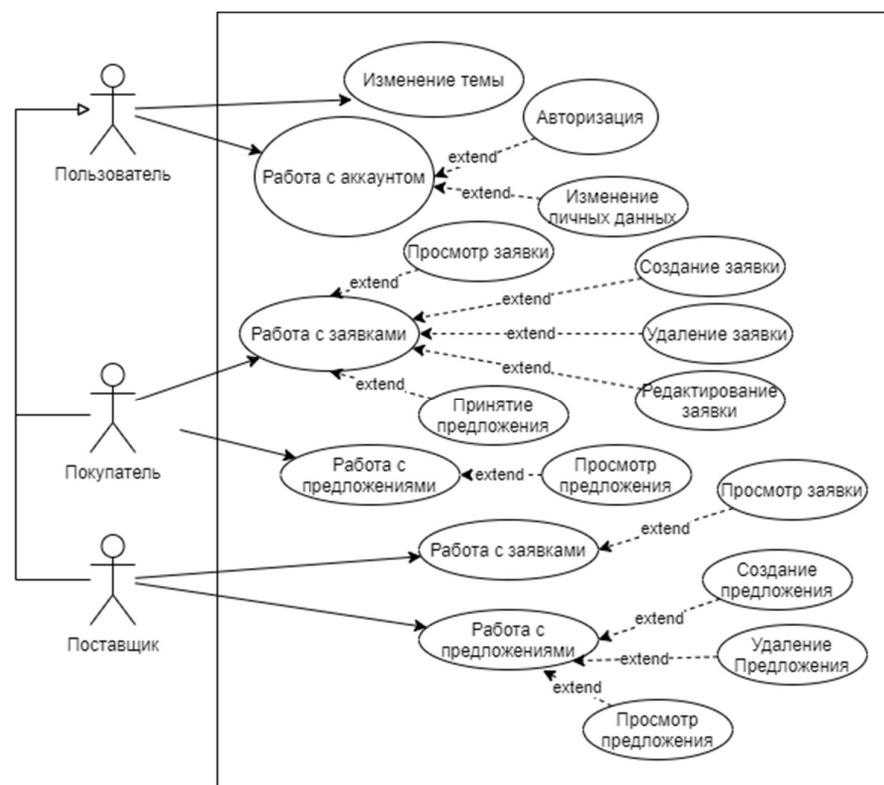


Рисунок 1.7 – Диаграмма вариантов использования

В данной диаграмме наглядно показаны все возможности взаимодействия с сайтом двух ролей (актёров): поставщик и потребитель.

1.4 Разработка информационной модели предметной области

Информационная модель — это модель объекта, представленная в виде информации, описывающей параметры и переменные объекта, которые являются существенными для данного рассмотрения, связи между ними, входы и

выходы объекта и позволяющая, путем представления информации об изменениях входных значений в модель, моделировать возможные состояния объекта. Кроме того, в более широком смысле информационная модель - это набор информации, которая характеризует существенные свойства и состояния объекта, процесса, явления, а также отношения с внешним миром.

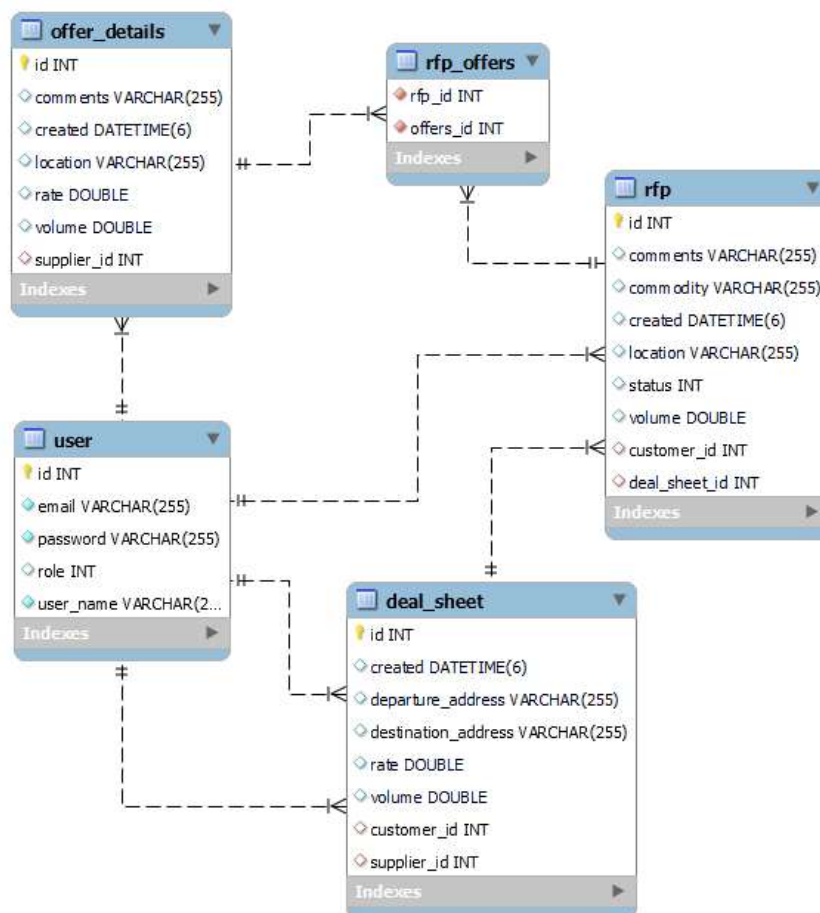


Рисунок 1.8 – Информационная модель системы в нотации IDEF1.X

Физическая модель базы данных содержит все детали, необходимые для создания базы данных: имя таблицы и столбца, типы данных, определения первичного и внешнего ключей.

IDEF1X — это метод разработки реляционных баз данных, использующий условный синтаксис, специально разработанный для удобного построения концептуальных схем.

Основной целью IDEF1X является поддержка интеграции. Интеграционный подход фокусируется на захвате, управлении и использовании единого семантического определения ресурса данных, называемого "концептуальной схемой". "Концептуальная схема" обеспечивает единое интегрированное определение данных внутри предприятия, которое не привязано к какому-либо

отдельному приложению обработки данных и не зависит от того, как физически эти данные хранятся или к ним осуществляется доступ.

1.5 UML-модели представления программного средства и их описание

Диаграмма вариантов использования — это диаграмма, отражающая взаимоотношения между вариантами использования системы и действующими лицами, которые участвуют в процессе.

Диаграммы последовательности используются для более подробного описания диаграмм вариантов использования. Это отличный инструмент для документирования проекта с точки зрения вариантов использования.

Диаграммы последовательности обычно содержат объекты, которые взаимодействуют в сценарии, сообщения, которыми обмениваются, и возвращаемые результаты, связанные с сообщениями. Однако часто возвращаемые результаты указываются только в том случае, если это не очевидно из контекста.

Объекты обозначаются прямоугольниками с подчеркнутыми именами (чтобы отличать их от классов).

Сообщения (вызовы методов) представляют собой строки со стрелками.

Возвращаемые результаты представляют собой пунктирные линии со стрелками.

Прямоугольники на вертикальных линиях под каждым из объектов показывают “время жизни” (фокус) объектов. Однако довольно часто они не изображены на схеме, все зависит от индивидуального стиля дизайна.

Диаграмма компонентов описывает особенности физического представления системы. Диаграмма компонентов позволяет вам определить архитектуру разрабатываемой системы, устанавливая зависимости между программными компонентами, которые могут выступать в качестве исходного, двоичного и исполняемого кода. Во многих средах разработки модуль или компонент эквивалентны файлу. Пунктирные стрелки, соединяющие модули, показывают соотношение зависимостей, аналогичное тому, которое наблюдается при компиляции исходного кода программ. Наиболее важными графическими элементами диаграммы компонентов являются компоненты, интерфейсы и зависимости между ними.

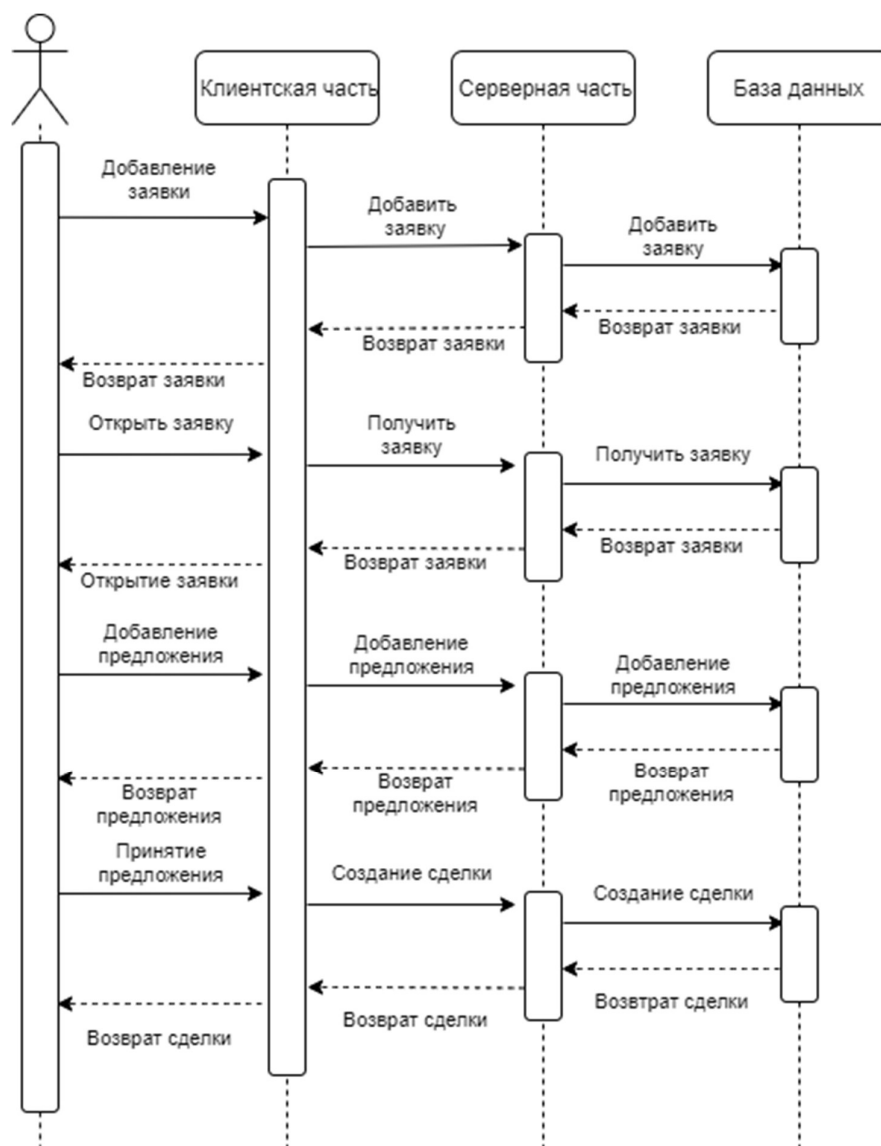


Рисунок 1.9 – Диаграмма последовательности оформления заказа пользователем

Диаграмма компонентов была разработана для следующих целей:

- визуализация общей структуры исходного кода программной системы;
- технические характеристики исполняемой версии программной системы;
- обеспечение многократного использования отдельных фрагментов программного кода;
- представления концептуальных и физических схем баз данных.

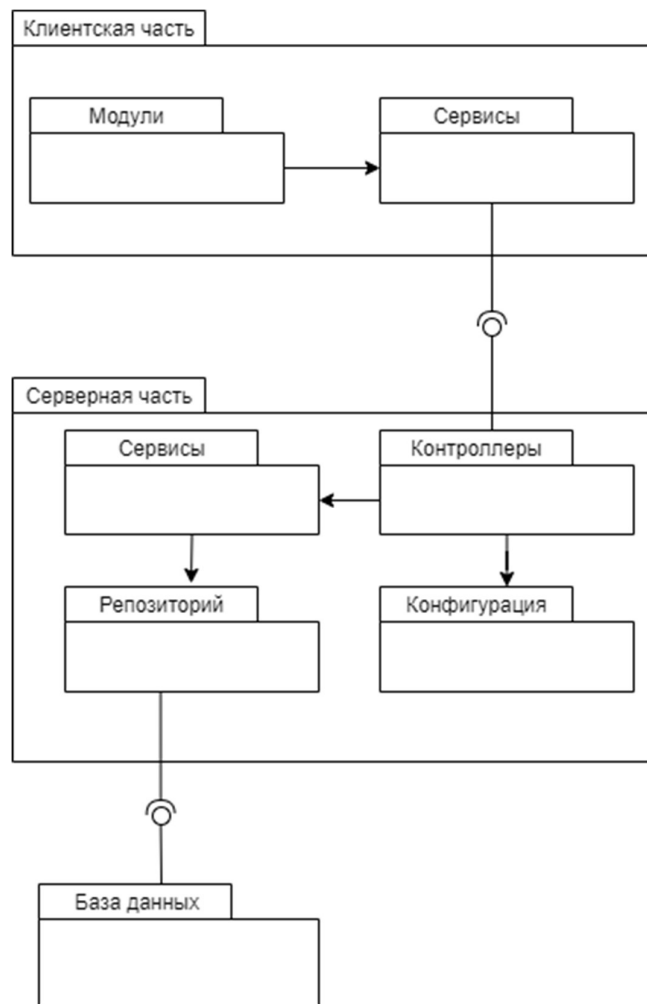


Рисунок 1.10 – Диаграмма компонентов системы

Схема развертывания предназначена для визуализации элементов и компонентов системы, которые существуют только во время выполнения, включая исполняемые файлы, динамические библиотеки, таблицы БД и т.д.

Основные цели, которые преследуются при разработке схемы развертывания:

- распределение компонентов системы по ее физическим узлам;
- отображение физических связей между узлами системы на этапе исполнения;
- выявление узких мест системы и реконфигурация ее топологии для достижения требуемой производительности.

Диаграмма классов определяет типы классов системы и различного рода статические связи, которые существуют между ними. На диаграммах классов изображаются также атрибуты классов, операции классов и ограничения, которые накладываются на связи между классами. Вид и интерпретация диаграммы классов существенно зависит от точки зрения (уровня абстракции):

классы могут представлять сущности предметной области (в процессе анализа) или элементы программной системы (в процессах проектирования и реализации).

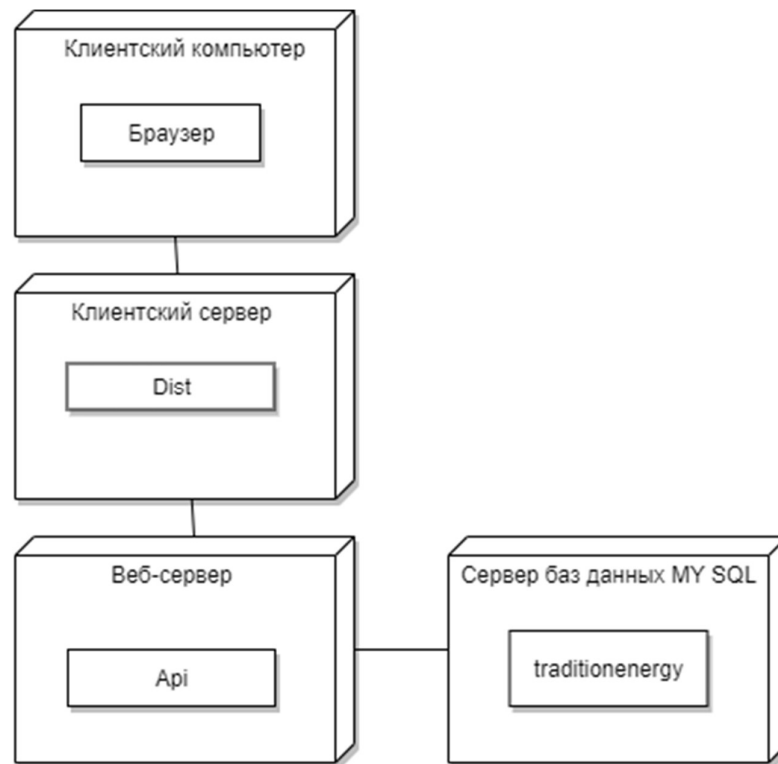


Рисунок 1.11 – Диаграмма развертывания системы

Диаграмма состояний — это хорошо известная технология, используемая для описания поведения системы. В той или иной форме диаграммы состояний существуют с 1960 года, и на заре объектно-ориентированного программирования они использовались для представления поведения системы. В объектно-ориентированном подходе диаграмма состояний рисуется для одного класса, чтобы показать поведение одного объекта в течение его жизненного цикла.

Диаграмма состояний показывает, как объект переходит из одного состояния в другое. Диаграммы состояний используются для объяснения того, как работают сложные объекты.



Рисунок 1.12 – Диаграмма состояния

Диаграмма состояния процесса заказа книги показывает нам в каких состояниях может находиться программа во время исполнения процесса.

2 ПРОЕКТИРОВАНИЕ И КОНСТРУИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

2.1 Постановка задачи

В курсовом проекте необходимо автоматизировать процесс нахождения потребителем поставщиков и наоборот.

Программное средство должно обладать следующими основными функциями, предоставляемые пользователю:

- регистрация;
- авторизация;
- просмотр и редактирование профиля;
- публикация заявок;
- публикация предложений;
- принятие предложений.

Проект должен быть написан на языке программирования Java с использованием ORM-технологии (Hibernate/JPA). Серверная часть должна быть реализована на основе фреймворка Spring или EJB (по согласованию с руководителем). Клиентская часть может быть реализована на веб-технологиях: JSP/Servlets/JSF/React/Angular (по согласованию с руководителем).

Бизнес-логику необходимо реализовать только на серверной части. Доступ к данным в СУБД должен осуществляться через драйвер СУБД. Схему базы данных (не менее пяти связанных таблиц) необходимо привести к 3-ей нормальной форме. Функциональные возможности серверной части должны насчитывать не менее 12 высокоуровневых вариантов использования, исключая тривиальные операции работы с БД.

Разработанное программное обеспечение должно выполняться в системе Windows 7/8/10 с возможной предустановкой библиотек или пакетов выбранной среды программирования.

Курсовой проект должен храниться в публичном репозитории на GitHub.

2.2 Архитектурные решения

Для создания серверного приложения использовался паттерн MVC. Шаблон проектирования контроллера представления модели (MVC) определяет, что приложение состоит из модели данных, информации представления и управляющей информации. Шаблон требует, чтобы каждый из них был разделен на разные объекты.

MVC — это скорее архитектурный шаблон, но не для полного приложения. MVC в основном относится к уровню пользовательского интерфейса /

взаимодействия приложения. Вам все равно понадобится уровень бизнес-логики, возможно, какой-то уровень обслуживания и уровень доступа к данным.

Преимущества:

- несколько разработчиков могут одновременно работать над моделью, контроллером и представлениями;
- MVC позволяет логически группировать связанные действия на контроллере вместе. Представления для конкретной модели также сгруппированы вместе;
- модели могут иметь несколько видов.

Недостатки:

- навигация по фреймворку может быть сложной, поскольку она вводит новые уровни абстракции и требует от пользователей адаптации к критериям декомпозиции MVC;
- знание множества технологий становится нормой. Разработчики, использующие MVC, должны владеть несколькими технологиями.

Модель содержит только чистые данные приложения, она не содержит логики, описывающей, как представить данные пользователю.

Представление (сама страница) представляет данные модели пользователю. Представление знает, как получить доступ к данным модели, но оно не знает, что означают эти данные или что пользователь может сделать, чтобы манипулировать ими.

Контроллер существует между представлением и моделью. Он прослушивает события, вызванные представлением (или другим внешним источником), и выполняет соответствующую реакцию на эти события. В большинстве случаев реакция заключается в вызове метода на модели. Поскольку представление и модель связаны через механизм уведомлений, результат этого действия затем автоматически отражается в представлении.

2.3 Описание алгоритмов, реализующих ключевую бизнес-логику разрабатываемого программного средства

Рассмотрим алгоритм принятия и создания предложения.

На первом этапе пользователь авторизуется, после чего он попадает на страницу со всеми заявками. Если пользователь оказался потребителем, то он может:

- создать новую заявку;
- отредактировать свою выбранную заявку;
- просмотреть другие заявки.

Если пользователь оказался поставщиком, то тогда он может:

- просмотреть заявки;
- создать предложение на заявку;
- отредактировать или удалить созданное этим пользователем предложение.

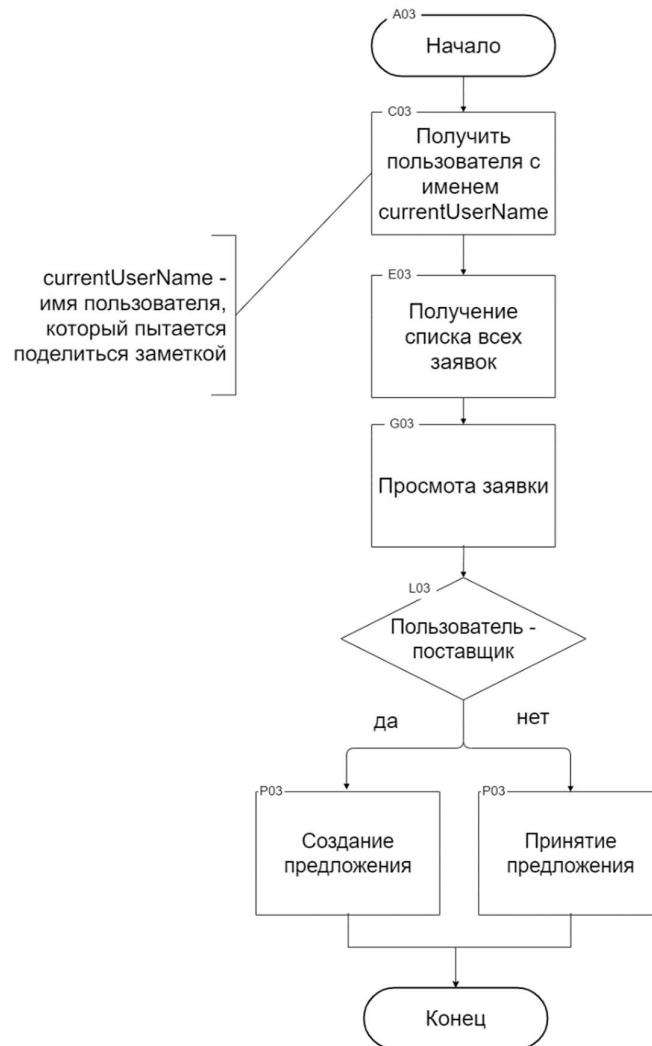
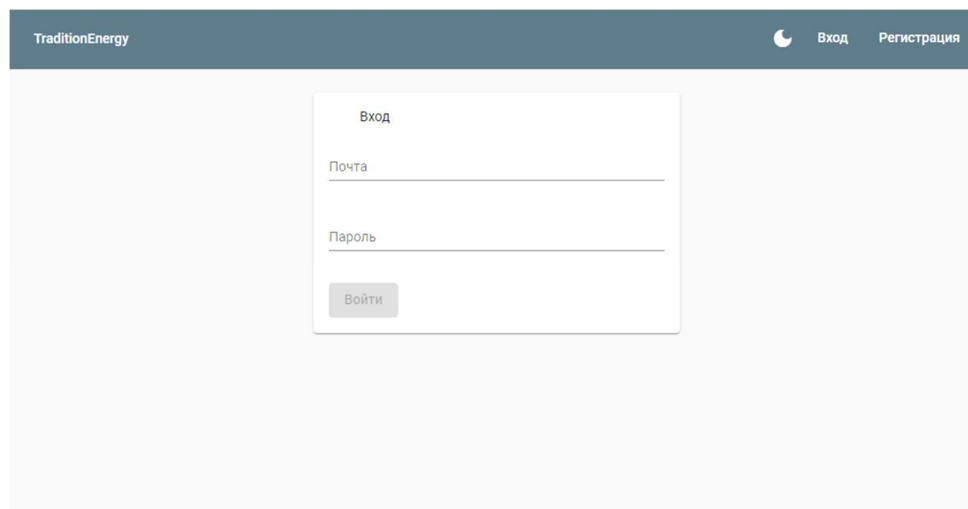


Рисунок 2.1 – Алгоритм принятия/создания предложения

После создания заявки и появления и принятия предложения для этой заявки, заявка становится более неактивной, то есть создавать новые предложения более невозможно, а также редактировать саму заявку, удалять её, а также редактировать. Заявка переходит в так называемое «заархивированное» состояние, в котором её можно только посматривать. Также появляется возможность скачать файл со всей информацией о заключённой сделке.

2.4 Проектирование пользовательского интерфейса

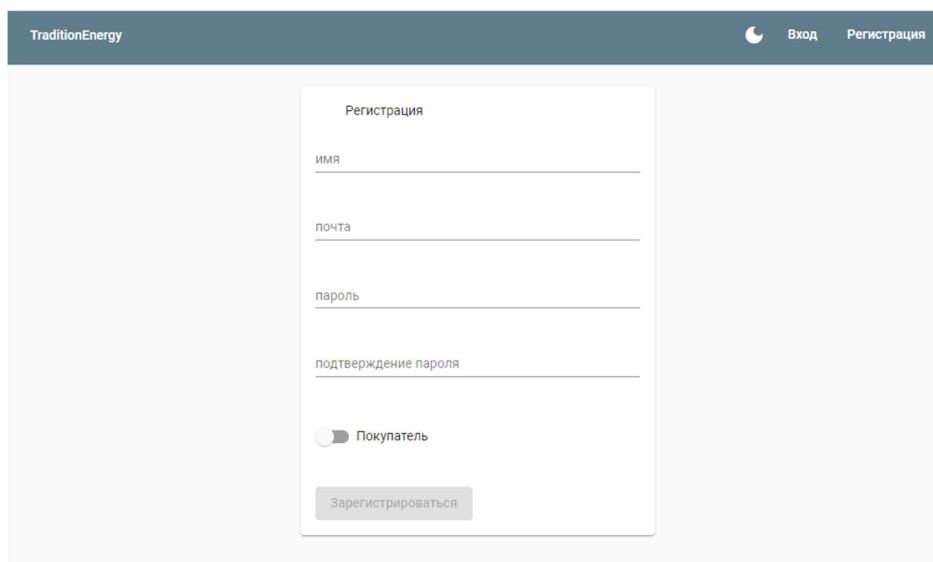
При запуске приложения автоматически происходит перенаправление на страницу с авторизацией. По стандарту используется светлая тема (рисунок 2.2).



The screenshot shows the login page of the TraditionEnergy application. At the top, there is a dark blue header with the logo 'TraditionEnergy' on the left and a moon icon, 'Вход', and 'Регистрация' on the right. The main content area is light gray and contains a white login form. The form has a title 'Вход', two input fields labeled 'Почта' and 'Пароль', and a 'Войти' button at the bottom.

Рисунок 2.2 – Вход

Если же пользователь не зарегистрирован, то можно нажать на кнопку регистрации, где будет форма регистрации (рисунок 2.3). После регистрации пользователю нужно продублировать данные в форме входа.

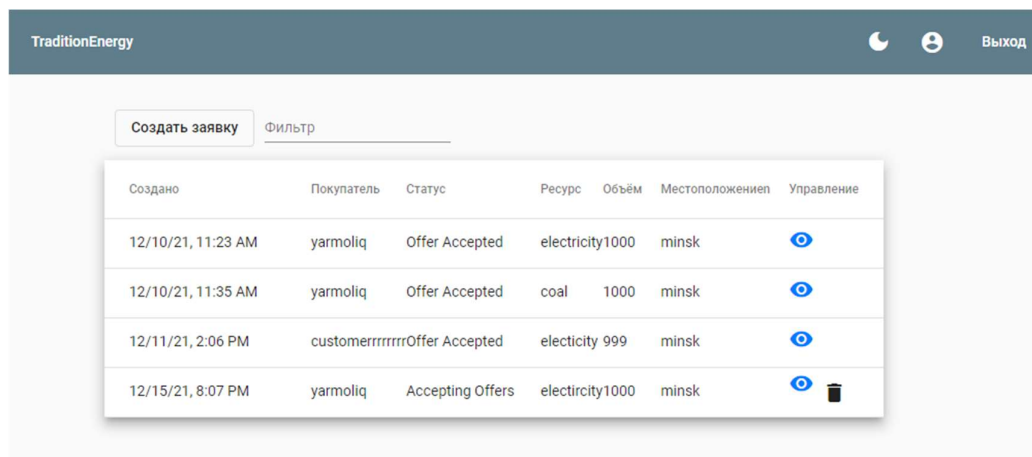


The screenshot shows the registration page of the TraditionEnergy application. It has the same header as the login page. The main content area is light gray and contains a white registration form. The form has a title 'Регистрация', four input fields labeled 'Имя', 'Почта', 'Пароль', and 'Подтверждение пароля', a toggle switch labeled 'Покупатель', and a 'Зарегистрироваться' button at the bottom.

Рисунок 2.3 – Регистрация

После успешного входа в приложение пользователь попадает на главную страницу, на которой есть таблица заявок (рисунок 2.4). На данной странице доступен просмотр заявок, фильтрация, удаление (для покупателей). В

столбике управление таблицы на главной странице имеются кнопки: просмотр для всех пользователей и удаление активной заявки для их авторов.



Создано	Покупатель	Статус	Ресурс	Объём	Местоположение	Управление
12/10/21, 11:23 AM	yarmoliq	Offer Accepted	electricity1000		minsk	
12/10/21, 11:35 AM	yarmoliq	Offer Accepted	coal	1000	minsk	
12/11/21, 2:06 PM	customerrrrrrr	Offer Accepted	electricity 999		minsk	
12/15/21, 8:07 PM	yarmoliq	Accepting Offers	electircity1000		minsk	

Рисунок 2.4 – Главная страница

Пользователям с ролью покупателя над таблицей доступна кнопка создания новой заявки. При нажатии на неё пользователь попадает на страницу с формой создания заявки (рисунок 2.5).



Commodity Type

Commodity Volume

Location

Comments

Создать

Рисунок 2.5 – Форма создания заявки

При нажатии на кнопку просмотра пользователь переносится на страницу просмотра заявки (рисунок 6.6). Здесь автор заявки может изменять комментарий заявки, принимать предложения. Все пользователи могут скачивать информацию о сделке (рисунок 6.7). Поставщики энергоресурсов в свою очередь могут оставлять предложения на заявку.

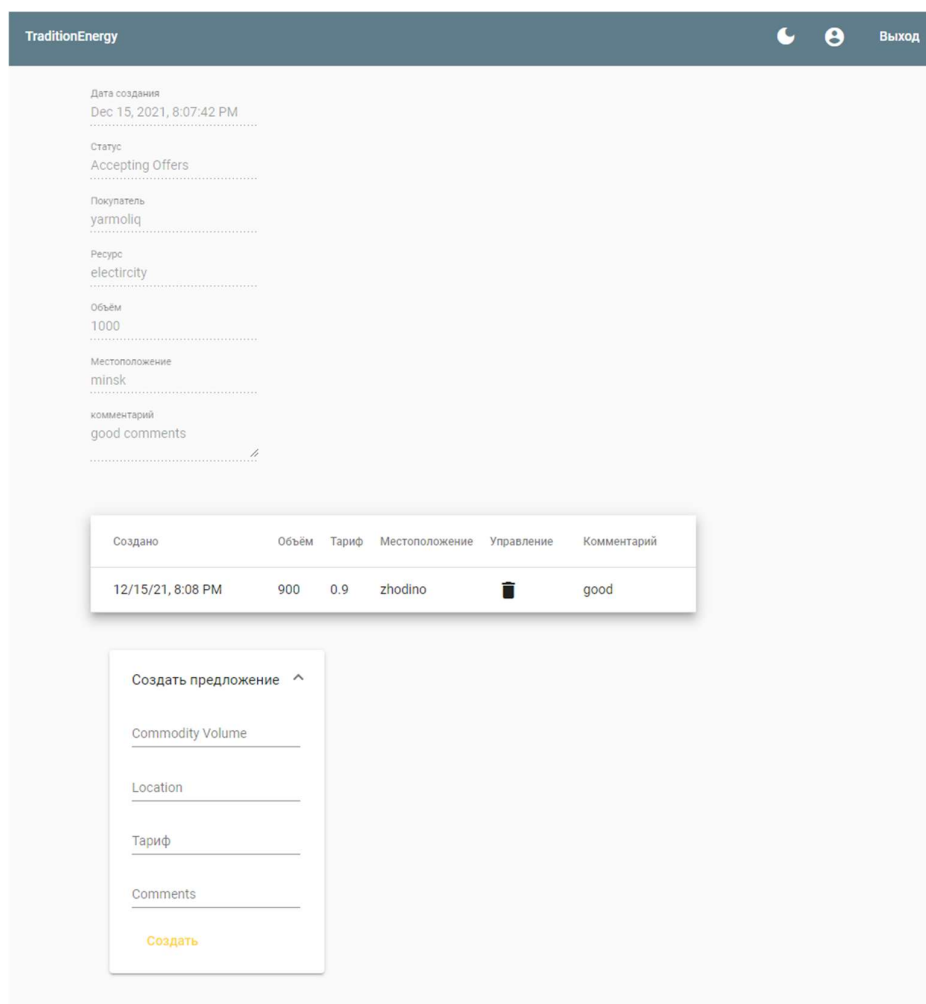


Рисунок 2.6 – Страница заявки

При нажатии на кнопку просмотра пользователь переносится на страницу просмотра заявки (рисунок 2.6). Здесь автор заявки может изменять комментарий заявки, принимать предложения. Все пользователи могут скачивать информацию о сделке. Поставщики энергоресурсов в свою очередь могут оставлять предложения на заявку.

3 ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРОГРАММНОГО СРЕДСТВА

В системе предусмотрена обработка неправильного ввода с помощью реактивных форм. Валидация предусмотрена на формах входа, регистрации и смены данных. Предусмотрена валидация как на клиентской части приложения, так и на серверной. Кнопки в каждой форме при каких-либо ошибках блокируются. Также предусмотрен глобальный обработчик ошибок.

The screenshot shows a registration form with the following fields and errors:

- Имя:** yfvt
- Почта:** Выфлд. Error: Incorrect email
- Пароль:** *****. Error: Your password contain the following: at least 1 uppercase letter; at least 1 uppercase letter; at least 1 number;
- Подтверждение пароля:** *****. Error: Password is not confirm

Below the fields is a toggle switch for 'Покупатель' (disabled) and a 'Зарегистрироваться' button (disabled). At the bottom, a red error message states: 'Something went wrong'.

Рисунок 3.1 – Попытка регистрации

Валидация пароля также сделала таким образом, чтобы говорить клиенту о произошедших ошибках во время ввода данных.

Валидация уникальности почты и имени пользователя также есть. Во время печати происходит отправление запроса на сервер с проверкой на уникальность. После этого происходит обработка ответа сервера на стороне клиента.

4 ИНСТРУКЦИЯ ПО РАЗВЁРТЫВАНИЮ ПРИЛОЖЕНИЯ И СКВОЗНОЙ ТЕСТОВЫЙ ПРИМЕР

В рамках данного раздела представлен алгоритм по развёртыванию системы.

Для запуска программного приложения требования такие:

- MySQL;
- Java Development Kit 18;
- Google Chrome.

Для настройки СУБД необходимо в файле `application.properties` указать логин и пароль от базы данных, а также строку для подключения к БД (`ConnectionString` – `spring.datasource.url`) (рисунок 4.1).

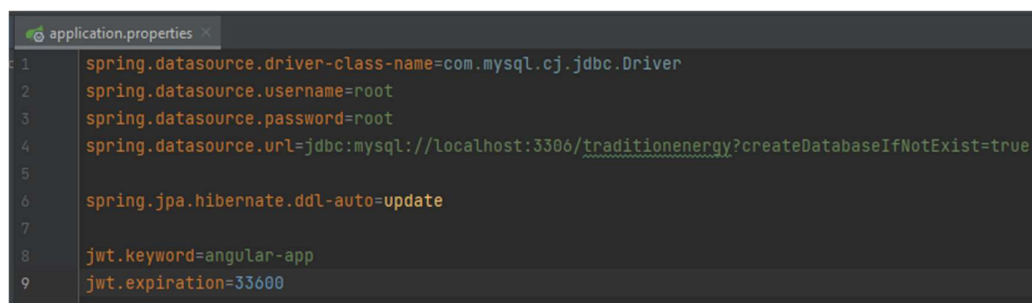


Рисунок 4.1 – Файл конфигурации приложения

После настройки СУБД вам потребуется запустить интегрированную среду разработки (желательно IntelliJ IDEA). После запуска среды разработки потребуется обновить зависимости в файле `pom.xml`, который отвечает за подтягивание сторонних библиотек и модулей.

Обновление зависимостей Maven показано на рисунке 4.2:

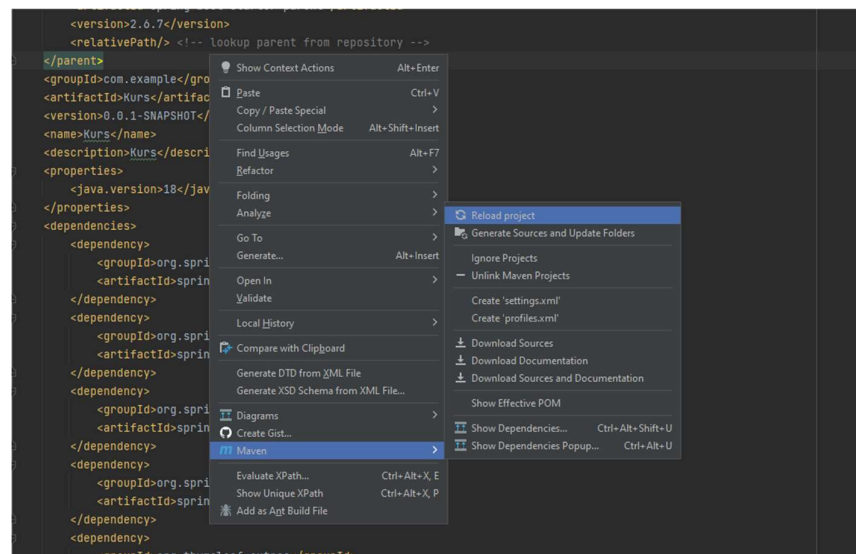


Рисунок 4.2 – Обновление зависимостей Maven

ЗАКЛЮЧЕНИЕ

В ходе данной работы было создано веб-приложение для обеспечения и поддержки взаимодействия потребителей с поставщиками энергоресурсов. Оно не является аналогом некоего коммерческого проекта, так как не реализует некий уникальный функционал.

Были изучены основные принципы построения приложений, архитектурные подходы для формирования веб-приложений, учитывая особенности используемых фреймворков. Изучен функционал базы данных MySQL, подход к построению обращения к базе данных по средствам паттернов.

В данной курсовой были проведены исследования в предметной области построение систем, хранящих заметки в виде веб-приложений.

Проведен синтез и анализ структурных, функциональных и логических схем разрабатываемой системы и отдельных их элементов.

Для удобства использования нужной информации база данных разбита на несколько взаимосвязанных таблиц, что обеспечивает быстрое и эффективное их использование.

Результатом работы является рабочая система для обеспечения и поддержки взаимодействия потребителей с поставщиками энергоресурсов.



СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- [1] Кадровое делопроизводство – учет кадров [Электронный ресурс]. – Режим доступа: <https://opyt.by/kadrovyj-uchyot/>.
- [2] Управление предприятием [Электронный ресурс]. – Режим доступа: <https://center-yf.ru/data/Menedzheru/upravlenie-predpriyatiem.php>.
- [3] Спецификация требований программного обеспечения [Электронный ресурс]. – Режим доступа: <http://unetway.com/tutorial/specifikacia-trebovanij-programmnogo-obespecenia>.
- [4] Язык программирования Java: особенности, популярность, ситуация на рынке [Электронный ресурс]. – Режим доступа: <https://ru.hexlet.io/blog/posts/yazyk-programmirovaniya-java-osobennosti-populyarnost-situatsiya-na-rynke-truda>.
- [5] База данных. Реляционная база данных [Электронный ресурс]. – Режим доступа: <https://htmlacademy.ru/tutorial/php/databases>.
- [6] Почему СУБД MySQL стала главной во всемирной паутине [Электронный ресурс]. – Режим доступа: <https://webformyself.com/pochemu-subd-mysql-stala-glavnoj-vo-vsemirnoj-pautine/>.
- [7] Что такое Spring Framework? От внедрения зависимостей до Web MVC [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/post/490586/>.
- [8] Развивая энергетику будущего [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.accenture.com/_acnmedia/PDF-123/Accenture-Reinventing-Oil-Gas-New-Energy-Era.pdf
- [9] Hibernate (framework) [Электронный ресурс]. – Электронные данные. – Режим доступа: [https://en.wikipedia.org/wiki/Hibernate_\(framework\)](https://en.wikipedia.org/wiki/Hibernate_(framework))
- [10] Design the infrastructure persistence layer [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://docs.microsoft.com/en-us/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/infrastructure-persistence-layer-design>
- [11] Введение в Angular [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://metanit.com/web/angular2/1.1.php>

ПРИЛОЖЕНИЕ А

(обязательное)


Проверка на антиплагиат



ТАРИФ
Free
[ИЗМЕНИТЬ](#)

БАЛЛЫ
0

ПОЛЬЗОВАТЕЛЬ
maximj75@gmail.com
[ПРОВЕРИТЬ ДОКУМЕНТ](#)

 МЕНЮ ru ▼

ГЛАВНАЯ / КАБИНЕТ / РЕЗУЛЬТАТЫ ПРОВЕРКИ


Оригинальность 95.32%


Заемствования 4.68%


Цитирования 0%


Самоцитирования 0%

[ПОЛНЫЙ ОТЧЕТ](#) [КРАТКИЙ ОТЧЕТ](#) [ИСТОРИЯ ОТЧЕТОВ](#) [РАСПЕЧАТАТЬ](#) [ВЫГРУЗИТЬ](#) [СОЗДАТЬ ССЫЛКУ](#)

 Свойства документа

 Текстовые метрики **NEW**

 Параметры проверки

 Статистика по документу

Отчет №1

Проверено: 03.09.2022 19:06:10
Начало проверки: 03.09.2022 19:06:08
Длительность проверки: 00:00:01
Модуль поиска Заемствования Самоцитирования Цитирования Оригинальность
Интернет Free 4.68% 0% 0% 95.32%

ПРИЛОЖЕНИЕ Б

(обязательное)

Листинг алгоритмов, реализующих бизнес-логику

```
package com.traditionenergy.services;

import com.traditionenergy.domain.entities.users.User;
import com.traditionenergy.domain.entities.users.UserRepository;
import com.traditionenergy.domain.entities.users.UserRole;
import com.traditionenergy.mappers.UserMapper;
import com.traditionenergy.models.requests.SignInUserRequest;
import com.traditionenergy.models.requests.SignUpUserRequest;
import com.traditionenergy.models.responses.ServiceResponse;
import com.traditionenergy.models.responses.ServiceResponseT;
import com.traditionenergy.models.responses.SignInResponse;
import com.traditionenergy.security.jwt.TokenProvider;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.authentication.BadCredentialsException;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.AuthenticationException;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.stereotype.Service;

@Service
@Slf4j
@RequiredArgsConstructor
public class AuthService {
    private final UserRepository userRepository;
    private final AuthenticationManager authenticationManager;
    private final TokenProvider tokenProvider;
    private final UserMapper userMapper;
    private final BCryptPasswordEncoder passwordEncoder;

    public ServiceResponseT<SignInResponse> signIn(SignInUserRequest authenticationDTO) {
        if (authenticationDTO.getEmail() == null || authenticationDTO.getPassword() == null) {
            return new ServiceResponseT<>("Bad credential");
        }
        try {
            User user = findUser(authenticationDTO);
            return new ServiceResponseT<>(createSignInResponse(user));
        } catch (AuthenticationException e) {
            return new ServiceResponseT<>("Invalid email or password");
        }
    }

    public ServiceResponseT<SignUpUserRequest> signUp(SignUpUserRequest signUpUserRequest) {
        if (!signUpUserRequest.getPassword().equals(signUpUserRequest.getConfirmPassword())) {
            return new ServiceResponseT<>("password mustMatch");
        }
    }
}
```



```

        }
        if (signUpUserRequest.getEmail() == null || signUpUserRe-
quest.getPassword() == null ||
            signUpUserRequest.getConfirmPassword() == null ||
signUpUserRequest.getUserName() == null) {
            throw new BadCredentialsException("Fill in required
fields");
        }

        var emailUnique = checkUniqueEmail(signUpUserRe-
quest.getEmail());
        if (!emailUnique.isSuccess())
        {
            return new ServiceResponseT<>(emailUnique);
        }
        var userNameUnique = checkUniqueUserName(signUpUserRe-
quest.getUserName());
        if (!userNameUnique.isSuccess())
        {
            return new ServiceResponseT<>(userNameUnique);
        }

        User user = userMapper.signUpDtoToUser(signUpUserRequest);
        user.setRole(signUpUserRequest.getIsCustomer() ?
UserRole.customer : UserRole.supplier);
        user.setPassword(passwordEncoder.encode(user.getPassword()));
        userRepository.save(user);
        return new ServiceResponseT<>(signUpUserRequest);
    }

    public ServiceResponse checkUniqueEmail(String email)
    {
        if (userRepository.existsByEmail(email))
        {
            return new ServiceResponse("email notUnique");
        }
        return new ServiceResponse();
    }

    public ServiceResponse checkUniqueUserName(String userName)
    {
        if (userRepository.existsByUsername(userName))
        {
            return new ServiceResponse("userName notUnique");
        }
        return new ServiceResponse();
    }

    private User findUser(SignInUserRequest authenticationDto) {
        String email = authenticationDto.getEmail();
        String password = authenticationDto.getPassword();
        var user = userRepository.findByEmail(email).get();
        authenticationManager.authenticate(new UsernamePasswordAu-
thenticationToken(user.getUserName(), password));
        return user;
    }

    private SignInResponse createSignInResponse(User user) {
        String token = tokenProvider.createToken(user.getUserName());
        SignInResponse signInResponse = new SignInResponse();
        signInResponse.setToken(token);
        try{
            signInResponse.setRole(user.getRole());

```

```

        }
        catch (Exception ex) {
            signInResponse.setRole(UserRole.supplier);
        }
        signInResponse.setEmail(user.getEmail());
        return signInResponse;
    }
}

package com.traditionenergy.services;

import com.traditionenergy.domain.entities.users.User;
import com.traditionenergy.domain.entities.users.UserRepository;
import com.traditionenergy.mappers.UserMapper;
import com.traditionenergy.models.requests.ChangeUserInfoRequest;
import com.traditionenergy.models.responses.ServiceResponseT;
import com.traditionenergy.models.responses.UserInfoResponse;
import com.traditionenergy.models.responses.UserInfoTokenResponse;
import com.traditionenergy.security.jwt.TokenProvider;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.http.HttpStatus;
import org.springframework.security.authentication.BadCredentialsException;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.util.Optional;

@Service
@Slf4j
@RequiredArgsConstructor
public class UsersService {
    private final UserRepository userRepository;
    private final UserMapper userMapper;
    private final BCryptPasswordEncoder passwordEncoder;
    private final TokenProvider tokenProvider;

    @Transactional
    public User findUserByEmail(String email) {
        if (email == null || email.isBlank()) {
            throw new BadCredentialsException(HttpStatus.BAD_REQUEST
+
                String.format(" email cannot be null or empty"));
        }
        User result;
        Optional<User> optionalUser = userRepository.findByEmail(email);
        if (optionalUser.isPresent()) {
            result = optionalUser.get();
        } else {
            throw new BadCredentialsException(HttpStatus.BAD_REQUEST
+
                String.format(" User with email %s do not exist",
email));
        }
        return result;
    }
}

```

```

        public ServiceResponseT<UserInfoResponse> getUserInfo(String
userName)
        {
            var user = userRepository.findByUserName(userName);
            if (user == null)
            {
                return new ServiceResponseT<UserInfoResponse>("user not-
Found");
            }
            UserInfoResponse userInfo = userMapper.userToUserInfoRe-
sponse(user.get());
            userInfo.setRole(user.get().getRole());
            return new ServiceResponseT<UserInfoResponse>(userInfo);
        }

        public ServiceResponseT<UserInfoTokenResponse>
setUserInfo(ChangeUserInfoRequest userInfo, String userName) {
            var userOptional = userRepository.findByUserName(userName);
            if (userOptional == null)
            {
                return new ServiceResponseT<>("user notFound");
            }
            if (userInfo.getNewPassword() != null &&
                userInfo.getConfirmNewPassword() != null &&
                !userInfo.getNewPassword().equals(userInfo.getCon-
firmNewPassword()))
            {
                return new ServiceResponseT<>("newPassword mustMatch");
            }
            if(userInfo.getUserName() == null) {
                return new ServiceResponseT<>("userName empty");
            }
            if(userInfo.getEmail() == null) {
                return new ServiceResponseT<>("email empty");
            }
            var user = userOptional.get();
            if(passwordEncoder.encode(userInfo.getOldPass-
word()).equals(user.getPassword())) {
                return new ServiceResponseT<>("oldPassword mustMatch");
            }
            user.setUserName(userInfo.getUserName());
            if(userInfo.getNewPassword() != null) {
                user.setPassword(passwordEncoder.encode(userInfo.getNew-
Password()));
            }
            user.setEmail(userInfo.getEmail());
            userRepository.save(user);
            String token = tokenProvider.createToken(user.getUserName());
            var resultUserInfoToken = userMapper.userToUserInfoToken-
Response(user);
            resultUserInfoToken.setToken(token);
            return new ServiceResponseT<>(resultUserInfoToken);
        }
    }

```

```

package com.traditionenergy.controllers;

```

```

import com.traditionenergy.domain.entities.rfp.Rfp;
import com.traditionenergy.domain.entities.rfp.RfpRepository;
import com.traditionenergy.domain.entities.rfp.RfpStatus;

```

```

import com.traditionenergy.domain.entities.users.UserRepository;
import lombok.RequiredArgsConstructor;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

import java.util.Date;
import java.util.List;

@RestController
@RequiredArgsConstructor
@RequestMapping("api/rfp")
public class RfpController extends BaseController{
    private final RfpRepository rfpRepository;
    private final UserRepository userRepository;

    @RequestMapping(method = RequestMethod.GET, value = "get")
    public Rfp get(int rfpId)
    {
        var rfp = rfpRepository.findById(rfpId).get();
        return rfp;
    }

    @RequestMapping(method = RequestMethod.GET, value = "get-all")
    public List<Rfp> getAll()
    {
        var rfps = rfpRepository.findAll();
        return rfps;
    }

    @RequestMapping(method = RequestMethod.GET, value = "delete")
    public Rfp delete(int rfpId)
    {
        var rfp = rfpRepository.findById(rfpId).get();
        rfpRepository.delete(rfp);
        return rfp;
    }

    @RequestMapping(method = RequestMethod.POST, value="create")
    public Rfp create(@RequestBody Rfp rfp)
    {
        var currentUser = userRepository.findByUserName(getCurrentUserName()).get();

        rfp.setCreated(new Date());
        rfp.setStatus(RfpStatus.AcceptingOffer);
        rfp.setCustomer(currentUser);

        rfpRepository.save(rfp);
        return rfp;
    }

    @RequestMapping(method = RequestMethod.POST, value="update")
    public Rfp update(@RequestBody Rfp rfp)
    {
        var currentUser = userRepository.findByUserName(getCurrentUserName()).get();
        var originalRfp = rfpRepository.findById(rfp.getId()).get();

        if(currentUser.getId() != originalRfp.getCustomer().getId())
        {
            return null;
        }
    }
}

```

```

    }

    originalRfp.setComments(rfp.getComments());

    rfpRepository.save(originalRfp);
    return originalRfp;
}
}

package com.traditionenergy.controllers;

import com.traditionenergy.domain.entities.offer.OfferRepository;
import com.traditionenergy.domain.entities.rfp.Rfp;
import com.traditionenergy.domain.entities.rfp.RfpRepository;
import com.traditionenergy.domain.entities.users.UserRepository;
import com.traditionenergy.models.requests.CreateOfferRequest;
import lombok.RequiredArgsConstructor;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

import java.util.Date;

@RestController
@RequiredArgsConstructor
@RequestMapping("api/offer")
public class OfferController extends BaseController {
    private final OfferRepository offerRepository;
    private final RfpRepository rfpRepository;
    private final UserRepository userRepository;

    @RequestMapping(method = RequestMethod.POST, value="create")
    public void create(@RequestBody CreateOfferRequest request)
    {
        var currentUser = userRepository.findByUserName(getCurrent-
tUserName()).get();
        var rfp = rfpRepository.findById(request.getRfpId()).get();

        var offer = request.getOffer();
        offer.setCreated(new Date());
        offer.setSupplier(currentUser);
        offerRepository.save(offer);

        rfp.getOffers().add(offer);
        rfpRepository.save(rfp);
    }

    @RequestMapping(method = RequestMethod.GET, value = "delete")
    public Rfp delete(int offerId)
    {
        var rfp = findRfpByOfferId(offerId);
        var offer = offerRepository.findById(offerId).get();
        rfp.getOffers().remove(offer);
        rfpRepository.save(rfp);
        offerRepository.delete(offer);
        return rfp;
    }

    Rfp findRfpByOfferId(int offerId) {

```

```

        for (var rfp : rfpRepository.findAll()) {
            for (var o : rfp.getOffers()) {
                if (o.getId() == offerId) {
                    return rfp;
                }
            }
        }
        return null;
    }
}

```

```
package com.traditionenergy.controllers;
```

```

import com.traditionenergy.domain.entities.dealSheet.DealSheet;
import com.traditionenergy.domain.entities.dealSheet.DealSheetRepository;
import com.traditionenergy.domain.entities.offer.OfferRepository;
import com.traditionenergy.domain.entities.rfp.Rfp;
import com.traditionenergy.domain.entities.rfp.RfpRepository;
import com.traditionenergy.domain.entities.rfp.RfpStatus;
import com.traditionenergy.domain.entities.users.UserRepository;
import com.traditionenergy.models.requests.CreateOfferRequest;
import lombok.RequiredArgsConstructor;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

```

```
import java.util.Date;
```

```

@RestController
@RequiredArgsConstructor
@RequestMapping("api/deal-sheet")
public class DealSheetController extends BaseController {
    private final OfferRepository offerRepository;
    private final RfpRepository rfpRepository;
    private final UserRepository userRepository;
    private final DealSheetRepository dealSheetRepository;

    @RequestMapping(method = RequestMethod.GET, value="create")
    public void create(int offerId)
    {
        var currentUser = userRepository.findByUserName(getCurrentUserName()).get();
        var offer = offerRepository.findById(offerId).get();
        var rfp = findRfpByOfferId(offerId);

        var ds = new DealSheet();
        ds.setCreated(new Date());
        ds.setCustomer(rfp.getCustomer());
        ds.setSupplier(offer.getSupplier());
        ds.setRate(offer.getRate());
        ds.setVolume(offer.getVolume());
        ds.setDepartureAddress(offer.getLocation());
        ds.setDestinationAddress(rfp.getLocation());
        dealSheetRepository.save(ds);
        rfp.setDealSheet(ds);
    }
}

```

```

        rfp.setStatus(RfpStatus.OfferAccepted);
        rfpRepository.save(rfp);
    }

    Rfp findRfpByOfferId(int offerId) {
        for (var rfp : rfpRepository.findAll()) {
            for (var o : rfp.getOffers()) {
                if (o.getId() == offerId) {
                    return rfp;
                }
            }
        }
        return null;
    }
}

```

ПРИЛОЖЕНИЕ В

(обязательное)

Листинг скрипта генерации базы данных

```
-- MySQL Script generated by MySQL Workbench
-- Wed Dec 15 16:32:54 2021
-- Model: New Model      Version: 1.0
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

--
-- Schema mydb
--
--
-- Schema traditionenergy
--

--
-- Schema traditionenergy
--

CREATE SCHEMA IF NOT EXISTS `traditionenergy` DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci ;
USE `traditionenergy` ;

--
-- Table `traditionenergy`.`user`
--
CREATE TABLE IF NOT EXISTS `traditionenergy`.`user` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `email` VARCHAR(255) NOT NULL,
  `password` VARCHAR(255) NOT NULL,
  `role` INT NULL DEFAULT NULL,
  `user_name` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB
AUTO_INCREMENT = 8
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

--
-- Table `traditionenergy`.`deal_sheet`
--
CREATE TABLE IF NOT EXISTS `traditionenergy`.`deal_sheet` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `created` DATETIME(6) NULL DEFAULT NULL,
  `departure_address` VARCHAR(255) NULL DEFAULT NULL,
  `destination_address` VARCHAR(255) NULL DEFAULT NULL,
  `rate` DOUBLE NULL DEFAULT NULL,
  `volume` DOUBLE NULL DEFAULT NULL,
  `customer_id` INT NULL DEFAULT NULL,
  `supplier_id` INT NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `FK5gnurwjglgu9lsgyasmfnllg4` (`customer_id` ASC) VISIBLE,
```



```

INDEX `FKnnyb374ikrqortl0a99fjb5i8` (`supplier_id` ASC) VISIBLE,
CONSTRAINT `FK5gnurwjg1gu9lsgyasmfnllg4`
  FOREIGN KEY (`customer_id`)
  REFERENCES `traditionenergy`.`user` (`id`),
CONSTRAINT `FKnnyb374ikrqortl0a99fjb5i8`
  FOREIGN KEY (`supplier_id`)
  REFERENCES `traditionenergy`.`user` (`id`))
ENGINE = InnoDB
AUTO_INCREMENT = 4
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

```

```

-----
-- Table `traditionenergy`.`offer_details`
-----

```

```

CREATE TABLE IF NOT EXISTS `traditionenergy`.`offer_details` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `comments` VARCHAR(255) NULL DEFAULT NULL,
  `created` DATETIME(6) NULL DEFAULT NULL,
  `location` VARCHAR(255) NULL DEFAULT NULL,
  `rate` DOUBLE NULL DEFAULT NULL,
  `volume` DOUBLE NULL DEFAULT NULL,
  `supplier_id` INT NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `FKoybe5adu0xsyolnhuls1bts70` (`supplier_id` ASC) VISIBLE,
  CONSTRAINT `FKoybe5adu0xsyolnhuls1bts70`
    FOREIGN KEY (`supplier_id`)
    REFERENCES `traditionenergy`.`user` (`id`))
ENGINE = InnoDB
AUTO_INCREMENT = 6
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

```

```

-----
-- Table `traditionenergy`.`rfp`
-----

```

```

CREATE TABLE IF NOT EXISTS `traditionenergy`.`rfp` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `comments` VARCHAR(255) NULL DEFAULT NULL,
  `commodity` VARCHAR(255) NULL DEFAULT NULL,
  `created` DATETIME(6) NULL DEFAULT NULL,
  `location` VARCHAR(255) NULL DEFAULT NULL,
  `status` INT NULL DEFAULT NULL,
  `volume` DOUBLE NULL DEFAULT NULL,
  `customer_id` INT NULL DEFAULT NULL,
  `deal_sheet_id` INT NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `FKqubuwfov5926cnq4djyvu65f` (`customer_id` ASC) VISIBLE,
  INDEX `FK7vgot5ydaviwqqf5rtvmxh6b9` (`deal_sheet_id` ASC) VISIBLE,
  CONSTRAINT `FK7vgot5ydaviwqqf5rtvmxh6b9`
    FOREIGN KEY (`deal_sheet_id`)
    REFERENCES `traditionenergy`.`deal_sheet` (`id`),
  CONSTRAINT `FKqubuwfov5926cnq4djyvu65f`
    FOREIGN KEY (`customer_id`)
    REFERENCES `traditionenergy`.`user` (`id`))
ENGINE = InnoDB
AUTO_INCREMENT = 7
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

```

```

-----
-- Table `traditionenergy`.`rfp_offers`
-----
CREATE TABLE IF NOT EXISTS `traditionenergy`.`rfp_offers` (
  `rfp_id` INT NOT NULL,
  `offers_id` INT NOT NULL,
  UNIQUE INDEX `UK_3uratdhhgd7bwayamx2ygagla` (`offers_id` ASC) VISIBLE,
  INDEX `FK4wfjbx5lbrv3tj63rshgbnlxl` (`rfp_id` ASC) VISIBLE,
  CONSTRAINT `FK4wfjbx5lbrv3tj63rshgbnlxl`
    FOREIGN KEY (`rfp_id`)
      REFERENCES `traditionenergy`.`rfp` (`id`),
  CONSTRAINT `FK14yuu7k4yullvkb15sml8vj8c`
    FOREIGN KEY (`offers_id`)
      REFERENCES `traditionenergy`.`offer_details` (`id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;

```