МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»
Кафедра IT-инжиниринг

ДОПУЩЕН К ЗАЩИТЕ
Заведующий кафедрой
PhD, доцент
_____ Т.С. Картбаев
«____» _____2019 г.

ДИПЛОМНЫЙ ПРОЕКТ

На тему: Разработка информационной системы для ТОО «BEST – Group A»

Специальность: 5В070400 – «Вычислительная техника и программное обеспечение»

Выполнил: Сыгаев А.А. Группа: ВТу-16-4
Научный руководитель: ст.преп., Мусатаева Г.Т.

Консультанты:
по экономической части: к.э.н., профессор _____Ж.Г. Аренбаева
«22» мая 2019 г.

по безопасности
жизнедеятельности: д.т.н., ст.преп._____Ш.Ш. Бекбасаров
«15» мая 2019 г.

по применению
вычислительной техники: ст. преп._____М.Н. Майкотов
«09» мая 2019 г.

Нормоконтролер: ассис._____ Ш.П. Жумагулова
«23» 05 2019 г.

Рецензент: к.т.н., зав. каф._____В.В. Сербин
«____» _____ 2019 г.

Алматы 2019

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»

Институт систем управления и информационных технологий

Кафедра IT-инжиниринг

Специальность 5В070400 – «Вычислительная техника и программное обеспечение»

ЗАДАНИЕ
на выполнение дипломногопроекта

Студенту Сыгаев Арман Алтаевич

Тема проекта: Разработка информационной системы для ТОО «BEST – Group A»

Утверждена приказом по университету№124 от «26» октября 2018 г.

Срок сдачи законченного проекта «24» мая 2019 г.

Исходные данные к проекту (требуемые параметры результатов исследования (проектирования) и исходные данные объекта):Руководство системы менеджмента качества на предприятии; международные стандарты ИСО-9001, данные преддипломной практики.

Перечень вопросов, подлежащих разработке в дипломном проекте, или краткое содержание дипломного проекта:

- аналитическая часть;
- проектная часть;
- экспериментальная часть;
- экономическая часть;
- безопасность жизнедеятельности;
- приложение А. Техническое задание;
- приложение Б. Листинг программы;
- приложение В. Акт внедрения.

Перечень графического материала (с точным указанием обязательных чертежей): представлены 15 таблиц, 20 иллюстрации.

Основная рекомендуемая литература:

1 Разработка управляемого интерфейса / Ажеронок В.А., Радченко М.Г., Хрусталева Е.Ю. – 523 с.

2 Разработка сложных отчетов в 1С: Предприятие 8.3 / Хрусталева Е.Ю. -357 с.

3 Профессиональная разработка в системе 1С:Предприятие 8.3 / Ажеронок В.А., Габец А.П., Гончаров Д.И., 2015 – 1055 с.

4 1С: Предприятие 8.3. Практическое пособие разработчика / Радченко М.Г, Хрусталева Е.Ю. ,2010. – 896 с.

Консультации по проекту с указанием относящихся к ним разделов проекта

Раздел	Консультант	Сроки	Подпись
Экономическая часть	Аренбаева Ж.Г.	04.03. 2019 – 22.05. 2019	
Безопасность жизнедеятельности	Бекбасаров Ш.Ш.	04.03. 2019 – 22.05. 2019	
Программное обеспечение	Майкотов М.Н.	01.04. 2019 – 15.05. 2019	
Нормоконтролер	Жумагулова Ш.П.	03.04.2019 – 23.05.2019	

ГРАФИК

подготовки дипломной проекта

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечание
Аналитическая часть	05.11.2018 – 22.12.208	
Проектная часть	07.01.2019 – 30.01.2019	
Экспериментальная часть	04.02.2019 – 13.04.2019	

Дата выдачи задания «14» января 2019 г.

Заведующий кафедрой _____ Т.С. Картбаев

Научный руководитель проекта _____ Г.Т. Мусатаева

Задание принял к исполнению студент _____ А.А. Сыгаев

Аңдатпа

Дипломдық жобаның тақырыбы: «BEST-Group А» ЖШС ақпараттық жүйесін дамыту.

Дипломдық жобаның мақсаты кеңсе қызметкерлерінің жұмыс процесін талдау және компанияларда жұмыс істеу үшін қарапайым және қолайлы конфигурация жасау болып табылады.

Алға қойған мақсатқа жету үшін келесідей технологиялар қолданылды:

– клиенттік интерфейсті құру үшін: 1С: Предприятие 8.3;

– дерекқорды бір-біріне қосу: СУБД MS SQL;

Дипломдық жоба кіріспеден, 5 бөлімнен және қорытындыдан тұрады.

Кіріспеде таңдалған тақырыптың өзектілігі ашылады, өңдеу мақсаты қойылады және орындауға қажетті міндеттер, өңдеуді қолдану аймағы анықталады.

Бірінші бөлімде есепті шешуге арналған технологияларға талдау жасау мен таңдау келтірілген.

Екінші бөлімде қосымшаны жобалау сипатталады.

Үшінші бөлім қосымшаны құруды сипаттаудан, қойылған мақсатты практикалық шешудің қадамдық сипатталуынан тұрады.

Төртінші бөлімде өңделген жобаның экономикалық мақсаттылығын негіздеу қарастырылған.

Бесінші бөлімде жүзеге асырылатын жоба аясында еңбек шарттарын жақсарту бойынша шаралар қарастырылған.

Қорытындыда клиент-серверлік веб-қосымшаларды құрудың қазіргі заманғы құралдары меңгерілген және қойылған мақсатқа сай өңдеу орындалған, ары қарай қосымшаны толыққанды өнімге дейін жақсарту стратегиялары құрылған.

Аннотация

Тема дипломного проекта: «Разработка информационной системы для ТОО «BEST – Group A»»

Цель дипломного проекта – проанализировать рабочий процесс сотрудников офисов и создать простую и удобную конфигурацию для работы в компаний.

Для достижения поставленной цели использовались технологии:

– для разработки клиентского интерфейса: 1С: Предприятие 8.3;

– для соединения базы данных между собою использовался: СУБД MS SQL;

В дипломный проект входит введение, 5 глав и итоговое заключение.

Во введении раскрывается актуальность выбранной темы, ставится цель разработки и задачи, которые необходимо выполнить, определяется область применения разработки.

В первой главе производится анализ и выбор технологий для решения задачи.

Во второй главе описывается проектирование приложения.

Третья глава представляет собой описание разработки приложения, содержится пошаговое описание практического решения поставленной задачи.

В четвертой главе производится обоснование экономической целесообразности разрабатываемого проекта.

В пятой главе рассматриваются мероприятия по улучшению условий труда в рамках реализуемого проекта.

Заключение освещает результаты исследования и разработки:

– изучены современные средства разработки клиент-серверных веб-приложений и выполнена разработка согласно поставленной цели;

– разработана дальнейшая стратегия улучшения приложения до полноценного продукта.

Annotation

Theme of the graduation project: «Development of an information system for «BEST - Group A» LLP»

The aim of the graduation project is to analyze the workflow of the office staff and create a simple and convenient configuration for working in companies.

Technologies used to achieve the goal:

– development of the client user interface: 1C: Предприятие 8.3;

– creation of the database: DBMS SQL;

The graduation project includes an introduction, 5 chapters and a conclusion.

The introduction reveals the relevance of the chosen topic, sets the development goal and the tasks that need to be performed, and determines the scope of the development.

The first chapter analyzes and selects technologies for solving the problem.

The second chapter describes the design of the application.

The third chapter is a description of the development stage of the application and contains a step-by-step description of the practical solution of the task.

The fourth chapter examines measures to improve working conditions in the framework of the ongoing project.

The conclusion highlights research and development results such as:

– studied modern development tools for client-server web applications and developed according to the set goal;

– further strategies to improve the application in to a full product.

Содержание

Введение

Всю историю вычислительной техники принято делить на три основных этапа:

- домеханический;
- механический;
- электронно-вычислительный.

Необходимость проводить не сложные арифметически операции появились с самого начала существования человека. Задолго до появления первых счетных машин люди изыскивали различные средства для проведения вычислений.

Первым инструментом для счета были руки. Все арифметические операции выполнялись при помощи десяти пальцев рук. В Западной Европе существовала целая система позволяющая представлять на пальцах числа до 9999.

Счет на пальцах, конечно, удобен, только с ним достаточно тяжело хранить информацию.

С возникновением у древних людей способности счета появилась необходимость в использовании приспособлений, которые смогли бы облегчить эту работу. Одно из таких орудий труда наших предков было обнаружено при раскопках поселения Дольни Вестоници на юго-востоке Чехии в Моравии. Обыкновенная кость с зарубками, получившая название “вестоницкая кость”, использовалась ими для ведения счета предположительно за 30 тыс. лет до н. э.

Первые идеи механизации вычислительного процесса появились в конце 15 века. Эскиз суммирующего устройства был разработан не безызвестным Леонардо да Винчи.

1642 год, французский физик Блез Паскаль создал первую механическую счетную машину. Она представляла собой шкатулку, на крышке которой, как на часах, были расположены циферблаты. На них устанавливали числа. Для цифр разных разрядов были отведены различные зубчатые колеса. Каждое предыдущее колесо соединялось с последующим с помощью одного зубца. Этот зубец вступал в сцепление с очередным колесом только после того, как были пройдены все девять цифр данного разряда.

1677 год, немецкий математик и философ Готфрид Вильгельм Лейбниц сконструировал свою счетную машину, позволяющую не только складывать и вычитать но также умножать многозначные числа. Вместо колец использовались цилиндры, на которые были нанесены цифры. Каждый цилиндр имел девять рядов выступов: один выступ на первом ряду, два на втором и так далее. Эти цилиндры были подвижны и устанавливались в определенном положении. Хоть машина Лейбница и была похожа на “Паскалину”, она имела движущуюся часть и ручку, с помощью которой можно было крутить специальное колесо или цилиндры, расположенные внутри аппарата. Такой механизм позволил ускорить

повторяющиеся операции сложения, необходимые для умножения. Само повторение тоже осуществлялось автоматически.

1830 год, английский математик Чарльз Бэббидж попытался построить универсальное вычислительное устройство, т.е. компьютер. Бэббидж называл его Аналитической машиной. Именно Бэббидж додумался до того, что компьютер должен содержать память и управляться с помощью программы. Бэббидж хотел построить свой компьютер как механическое устройство, а программой собирался управлять посредством перфокарт – карт из плотной бумаги с информацией наносимой с помощью отверстий (в то время они активно использовались на ткацких станках).

1941 год, немецкий инженер Конрад Цузе построил небольшой компьютер на основе электромеханического реле. Но из-за войны его работы не были опубликованы.

1943 го, в США на одном из предприятий фирмы IBM Говард Эйкен создал более мощный компьютер под названием «Марк-1», который реально использовался для военных расчетов. В нем использовалось сочетание электрических сигналов и механических приводов. Программа обработки данных вводилась с перфоленты. Размеры: 15 Х 2,5 м., 750000 деталей. “Марк-1” мог перемножить два 23-х разрядных числа за 4 с.

Первая ЭВМ «ЭНИАК» (цифровой интегратор и вычислитель) была создана в США после второй мировой войны в 1946 году. В группу создателей этой ЭВМ входил один из самых выдающихся ученых XX в. Джон фон Нейман.

В 1945 году к этой работе был привлечен знаменитый математик Джон фон Нейман, который подготовил доклад об этом компьютеры. Тезисы выдвинутые фон Нейманом сформировали понятие архитектуры компьютера, которая лежит в основе построения компьютеров до настоящего времени.

«Фон Неймановская» архитектура компьютера:

– арифметическое - логическое устройство;
– устройство управления;
– запоминающее устройство;
– внешние устройства для ввода-вывода.

Такой принцип построения ЭВМ используется без особых изменений и по сей день. При этом не важно настольный стационарный это компьютер и карманный портативный.

Основной целью дипломной работы является автоматизирования регистрационных процессов авиа перевозок и мониторинга.

Назначение системы

Основным назначением Системы является регистрация пользователей и грузов, мониторинг маршрутов. Система «Регистрация и Мониторинг» предназначена для автоматизирования регистрации билетов грузов и

мониторингамаршрутов. Используемый язык программирования С++компилируемый, статистический типизированный программирования. Поддерживает парадигмы программирования как процедурное программирование, обьектно-орентированное программирование, обеспечивает модульность, раздельную компиляцию, обработку исключений, абстракцию данных,объявление типов (классов)объектов,виртуальные функции.Стандартная библиотека включает, в том числе, общеупотребительные контейнеры и алгоритмы. С++ сочетает свойства как высокоуровневых, так и низкоуровневых языков.Всравнении с его предшественником языкомС, наибольшее внимание уделено поддержке обьектно-ориентированного программирования.

1 Описание программного продукта

1С: Предприятие – программный продукт компании 1С, предназначенный для автоматизации деятельности на предприятии.

Первоначально программы 1С:Предприятие были предназначены для автоматизации бухгалтерского учёта и управленческого учёта (включая начисление зарплаты и управление кадрами). Но сегодня программы 1С: Предприятие находят своё применение даже в областях, далёких от бухгалтерских задач.

1С:Предприятие – это (одновременно) и технологическая платформа, и пользовательский режим работы. Технологическая платформа предоставляет объекты (данных и метаданных) и механизмы управления объектами. Объекты (данные и метаданные) описываются в виде конфигураций. При автоматизации какой-либо деятельности составляется своя конфигурация объектов, которая и представляет собой законченное прикладное решение. Конфигурация создаётся в специальном режиме работы программного продукта под названием «Конфигуратор», затем запускается режим работы под названием «1С:Предприятие», в котором пользователь получает доступ к основным функциям, реализованным в данном прикладном решении (конфигурации).

Технологическая платформа «1С:Предприятие» представляет собой программную оболочку над базой данных (используются базы на основе DBF-файлов в 7.7, собственный формат 1CD с версии 8.0 или СУБД Microsoft SQL Server на любой из этих версий). Кроме того, с версии 8.1 хранение данных возможно в СУБД PostgreSQL и IBM DB2, а с версии 8.2 добавилась и Oracle.

Программы 1С: Предприятие имеют свой внутренний язык программирования, обеспечивающий, помимо доступа к данным, возможность взаимодействия с другими программами посредством OLE и DDE, в версиях 7.7, 8.0 и 8.1 – с помощью COM-соединения.

Клиентская часть платформы функционирует только в среде ОС Microsoft Windows. Начиная с версии 8.1, серверная часть платформы в клиент-серверном варианте работы «1С:Предприятия» может функционировать на ОС Linux.

Существуют специальные версии среды исполнения 1С для ноутбуков и PDA, ПО создания веб-приложений, взаимодействующих с базой данных «1С:Предприятие».

1.1 Назначение программного продукта

Функции программного комплекса «1С:Предприятие» классифицируются по направлениям автоматизации и группам пользователей. Эти функции системы имеют своей целью обеспечение руководителей информацией, необходимой для

оценки ситуации и принятия актуальных решений. Это, например, такие механизмы, как бюджетирование, анализ рентабельности деятельности предприятия, сбыта продукции и многое другое.

Эта функциональность решает задачи работников, которые занимаются торговой, производственной, а также деятельностью в области оказания услуг. С помощью системы можно эффективно организовать ежедневную работу организации: подготовка документов, управление выпуском продукции и запасами, оформление заказов, контроль исполнения задач и т.п.

Еще одной важной возможностью программного комплекса является учет и отчетность. Данная функция решает задачи бухгалтерии: обеспечение ведения учета в соответствии с актуальными требованиями законодательства. Это такие задачи, как, например: расчет заработной платы, ведение бухгалтерского и налогового учета, составление отчетной документации и т.д.

Отличительные особенности решений программного продукта «1С»: проработка состава функциональности для типовых решений. При создании системы был проанализирован опыт пользователей, которые применяют программы системы «1С:Предприятие», отследив изменение их потребностей.

Одним из основных и уникальных качеств программы является совокупность стандартизации решений и учета индивидуальных потребностей определенного предприятия. Это происходит следующим образом: сразу происходит выпуск набора типовых решений, которые направлены на массовые типы предприятий. При их разработке обязательно учитывается опыт использования программы на различных предприятиях. Это позволяет детально проработать функциональность программного продукта в тандеме с методологическими решениями и концентрацией именно на специфических отраслевых потребностях.

Более того, возможности программного комплекса «1С:Предприятие» позволяют создавать и индивидуальные решения с учетом потребностей конкретной организации. Такие решения, как правило, являются развитием типового решения фирмы «1С» или специализированного решения, но, если потребуется, могут быть разработаны «с нуля».

Руководитель может выбрать оптимальный вариант автоматизации – исходя из приоритетов своего предприятия, допустимых сроков и экономической целесообразности. А устройство программного продукта «1С:Предприятие» и принцип его построения позволяют оперативно реагировать на изменения потребностей конечных пользователей.

Основа системы программ «1С:Предприятие» – единая технологическая платформа, фундамент для построения всех прикладных решений. Еще одним важным плюсом программы»1С:Предприятие» является открытость системы – возможность понять ее работу.

В комплект системы входят средства, которые могут стать необходимыми для доработки прикладных решений и внесения в них изменений любой сложности; полный комплект документов.

1.2 Потенциальные пользователи программного продукта

Потенциальными пользователями данного продукта станут сотрудники компаний, так как для таких компаний очень важным является этап планирования их рабочего времени. В первую очередь, это заключается в том, что людей не так много, как задач, которые на них возлагаются. А главное, что все эти задачи должны быть выполнены в поставленный срок, и ни какую из задач нельзя забыть или пропустить.

Пользоваться данным программным продуктом будут все сотрудники компании, вне зависимости от их положения в компании. Это обусловлено тем, что любой сотрудник, несмотря на занимаемую должность, должен приносить пользу своим товарищам по команде выполненными заданиями. Любая большая компания, являющаяся коммерчески успешной и узнаваемой, начинала с небольшой команды, где каждый отвечал за поставленные ему задачи.

Еще одна важная идея проекта состоит в том, что помимо упрощения тайм-менеджмента сотрудников компании, это приложение также будет являться более безопасным, чем аналогичные сервисы в Интернете. Ни у кого в Интернете не будет возможности заполучить пароль сотрудника и прочитать его деловую переписку с коллегами. А это отразится на уникальности наработок компании, так как идея будет надежно охраняться.

Единственный момент, когда может произойти утечка данных – напрямую от сотрудников компании. К сожалению, решить данный фактор программным способом не получится. Здесь все зависит от качества отбора сотрудников при приеме в компанию и их мотивации. Так, например, в банковской системе сотрудникам выплачивается большая заработная плата, чтобы у них не возникало желания списать себе денег с чьего-либо банковского счета

Большинством потенциальных пользователей будут сотрудники офисов, которые имеют доступ к персональному компьютеру в течение дня. Но в перспективе развития, когда будет выпущена мобильная версия приложения, то данный продукт подойдет как для работы в офисах, так и в крупных магазинах, на фабриках и в других подобных предприятиях.

Здесь можно привести такой пример: часто в больших продуктовых магазинах менеджеров каких-либо отделов просят пройти для приемки товара, или кассира просят пройти к его рабочему месту, так как поток покупателей увеличился. В большинстве магазинов просто стоит система, которая на весь торговый зал делает достаточно громкое заявление. Но и это не гарантирует

того, что в этот момент сотрудник будет в зоне слышимости этого сообщения. Гораздо лучше, если ему на телефон будет приходить оповещение от старшего менеджера, которое он точно услышит или почувствует благодаря звуковому сигналу или вибрации телефона. К тому же рассылка сообщений будет зависеть только от мощности Wi-Fiмодемов, расположенных на территории магазина. К тому же человеку, который делает объявление можно найти другую должность, либо если системой громкоговорителя может воспользоваться любой сотрудник, которому необходимо сделать объявление, то это избавит его от нужды отходить от своей работы до системы громкоговорителя, чтобы найти рабочего в торговом зале. Это будет здорово экономить время и повысит процент выполненных задач.

Возвращаясь к теме офисных сотрудников, можно сказать, что также подобный планировщик избавляет пользователей от постоянного ведения записей в ежедневниках, на каких-то бумагах для заметок, которые теряются с завидной частотой. Это позволит сократить время на поиски необходимой задачи в своем ежедневнике, чтобы, например, вычеркнуть ее за ненадобностью.

Другими словами, продукт предназначен для пользователей, которые желают систематизировать свои задачи, и не тратить большое количество времени на ведение понятных и читаемых записей.

Интерфейс данного приложения спроектирован таким образом, чтобы любому пользователю, даже тому, который видит это приложение впервые, можно было легко сориентироваться в структуре системы и ее возможностях.

Разумеется, для этого ему будет необходимо пройтись хотя бы раз по всем вкладкам приложения, попробовать весь его функционал. Но с большой вероятностью для пользователя не останется «белых» пятен в приложении, которые будут смущать его в ходе использования.

Навигационная панель располагается сверху, в центре, и выделена контрастным цветом, кнопки регистрации/входа привычны для пользователя в верхнем углу. Также пользователи привыкли, что окно для ввода сообщения в чат располагается в нижней части экрана. Все эти мелочи позволяют пользователя распознать знакомые «силуэты» и с легкостью определить значение того или иного элемента в веб-приложении [1].

Все вышеперечисленное говорит о том, что для того, чтобы пользователь захотел использовать приложение, необходимо завлечь его взор хороший интерфейсом, актуальным функционалом, который будет способен работать четко и слаженно, как часы.

Стремлением любого проекта является привлечение большого числа пользователей, а для этого всего лишь достаточно обратить внимание на тех людей, которые будут пользоваться приложением. Определить, что для них является приоритетным, а что не так уж и важно.

Важно использовать семантическую верстку, поскольку пользователь с нарушением слуха не должен чувствовать себя ограниченным при использовании приложения. В последнее время замечается положительная тенденция у многих компаний: использовать семантическую верстку в любом проекте, который когда-либо увидит свет. Это означает, что люди, запускающие приложение с чтением экранного текста, смогут правильно услышать и понять смысл написанного текста, т. к. голосовые помощники выбирают разный темп чтения и тональность для каждого из элементов верстки страницы.

1.3 Обзор существующих аналогичных программных продуктов

В современных реалиях каждый день разрабатывается большое количество программных продуктов, каждый из которых по-своему уникален. Все программные продукты, занимающие определенную нишу в сфере предоставляемого функционала, имеют как сходные компоненты, так и что-то своё, что-то уникальное.

Очень часто пользователи по тем или иным причинам (чаще субъективным) хотят найти альтернативу программам 1С-предприятие. Как правило, отрицательные эмоции вызваны медленной работой, плохой настройкой 1С 8.3 или простым незнанием возможностей конфигурации.

Так ли просто найти зарубежный или отечественный аналог 1С? Есть ли конкурентоспособная замена 1С? Так ли легко подобрать альтернативу всеми любимого 1С-предприятия? Я попробую разобрать основных конкурентов 1С-Предприятия в области ERP на нашем рынке.

Среди отечественных систем альтернативу 1С могут составить, пожалуй, только системы Галактика и Парус. Среди зарубежных можно выделить как аналоги SAP и Microsoft Dynamics Axapta (Navision)**.**

1.3.1 Галактика

Данная система как неплохой аналог программы 1С 8.3 существует на рынке уже 25 лет. Программа завоевала свою нишу в области ERP, имеет достойное количество клиентов, доверивших им автоматизацию предприятия. Это полноценная альтернатива 1С.

Систему нельзя корректировать, кроме мелких исправлений интерфейса. Все доработки необходимо заказывать у разработчика системы, по сравнению с 1С это выйдет дорого и неприемлемо по срокам. Система доступна для внешней интеграции по средствам XML, COM, ActiveX, ODBC.

1.3.2 Парус

Данная программа также существует очень давно и полноценно может быть названа аналогом 1С, основана компания в 1990 году. Большинство доходов компания получает от государственных компаний. Парус так же состоит из модулей: финансы, бухгалтерия, MRP, CRM и т.д.

Систему имеют право дорабатывать только сами разработчики, софт приходится использовать «как есть». Как правило, устанавливается в БД Oracle, что выходит достаточно дорого при использовании лицензионного ПО. Один из главных плюсов системы – масштабируемость решения.

1.3.3 SAP

SAP – это программа класса ERP, родом из Германии. SAP является самой перспективной заменой 1С. Сап, как и остальные решения, состоит из модулей, различающихся по видам назначения. САП очень популярная система в мире. В России в основном внедряется крупными предприятиями. Это обусловлено высокой ценой лицензий и услуг (в 3-10 раз выше, чем 1С). САП имеет хорошую репутацию среди ERP, решение хорошо масштабируемое.

Доработка SAP возможна, но достаточно трудоёмкая. Практически то, что в 1С 8.2 можно доработать за 1-2 часа, в SAP занимает 1-2 дня. Как правило, в SAP ничего не дописывают, лишь настраивают.

1.3.4 Todoist (todoist.com)

Первая версия Axapta была выпущена в 1998 году. Достойная альтернатива SAP, программа как аналог 1С 8, несмотря на распространенность в мире, не может похвастаться большой отечественной аудиторией. Axapta содержит все современные модули: MRP, HR, CRM и т.д. Среди особенностей данного решения высокая степень интегрируемости с продуктами Microsoft – Outlook, Excel и тд. Лицензии не такие дорогие относительно SAP.

Доработки софта, подобно САП, достаточно трудоемкие**.** Специалистов на отечественном рынке не так уж много.

1.4 Выбор средств и технологий

Данный подраздел повествует о том, какие технологии были использованы при разработке дипломного проекта, их основные достоинства и недостатки. Также поясняется, почему были выбраны именно эти технологии, и за какую часть в приложении они несут ответственность.

1.4.1 1С:Предприятие

Программный продукт компании «1С», предназначенный для автоматизации деятельности на предприятии.Технологическая платформа «1С:Предприятие» представляет собой программную оболочку над базой данных. Используются базы на основе DBF-файлов в 7.7, собственный формат 1CD с версии 8.0 или СУБД MicrosoftSQLServer на любой из этих версий. Кроме того, с версии 8.1 хранение данных возможно в PostgreSQL и IBMDB2, а с версии 8.2 добавилась и Oracle. Платформа имеет свой внутренний язык программирования, обеспечивающий, помимо доступа к данным, возможность взаимодействия с другими программами посредством OLE и DDE, в версиях 7.7, 8.0 и 8.1 – с помощью COM-соединения.

1.4.2Microsoft SQLServer

Система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов – Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка.

1.5 Постановка цели и задач

Достижение цели не является одномоментным мероприятием. Достижение цели представляет собой решение совокупности задач, связанных между собой по смыслу и направленных для достижения единой цели.

Каждая из задач направлена на изучение или разработку каких-либо вещей в проекте. Таким образом, получается, что задачи – своеобразные «кирпичики» при выполнении которых достигается определенная цель.

Ниже приведены цель дипломного проекта и задачи, которые необходимо реализовать для достижения поставленной цели.

Целью данного проекта является – обеспечение новой информации для компании, дополняющей печатные издания, служащее для индивидуального и индивидуализированного обучения и позволяющее в ограниченной мере тестировать полученные знания и умения обучаемого. Данной конфигурации облегчает работу пользователю в компании, создание новых отчетов и таблиц, ранее не представленных в конфигураций 1С.

Основные задачи дипломного проекта:

– изучение предметной области на примере виртуальной IT-компании;

– определение основных требований к программному продукту;

– выбор и обоснование технологий для разработки программного продукта;

– разработка пользовательского веб-интерфейса;

– проектирование базы данных для хранения данных пользователей;

– обоснование экономической целесообразности разрабатываемого продукта;

– предложение мероприятий по улучшению условий труда в рамках реализуемого проекта.

2. Проектирование программного продукта

Данная глава представляет собой описание проектирования базы данных. Также раскроются понятия структура ПО и функциональной структуры приложения. Важно отметить, почему мы выбираем именно этот тип базы данных и каким образом этот выбор сможет повлиять на дальнейшее развитие приложения. В связи с этим необходимо предусмотреть все преимущества и недостатки выбранного типа базы данных, чтобы в дальнейшем учесть их при проектировании

Также описание изучения предметной области: какие таблицы и сущности должны присутствовать в базе данных для корректной работы приложения.

База данных должна содержать в себе четкие, упорядоченные таблицы, данные в которые разделены по группам и связаны между собой.

2.1 Структура программного обеспечения

Структура программного обеспечения включает в себя любое программное обеспечение, которое было использовано при создании определенного приложения или пакета приложений.

Все программное обеспечение, используемое в создании ПО, можно разделить на 3 группы:

– системное ПО: программное обеспечение общего пользования. Сюда относятся операционные системы и их оболочки, различные специфические драйверы и утилиты, системы, которые предназначены для технического обслуживания машины разработчика;

– прикладное ПО: сюда относятся различные прикладные программы и пакеты этих программ. Например, это программы для работы с текстовой информацией, её оформлением, программы для просмотра веб-страниц или веб-браузеры и т. д.;

– инструментальное ПО: также называют системами программирования. Это инструменты, которые разработчик приложения использует для просмотра исходного кода программы, его редактирования и дальнейшего запуска. В настоящее время такие системы чаще всего называют интегрированными средами разработки, поскольку они включают в себя большой перечень инструментов для работы с различными языками программирования и СУБД.

Также существует 4 группа ПО, которую редко упоминают, — это базовое ПО. Базовым является то ПО, которое втроенно в аппаратное обеспечение компьютера. Чаще всего к данной категории относят базовую систему ввода-вывода (BIOS).

Исходя из схемы на рисунке 2.1.1, видно, что разработка приложения происходила под управлением операционной системы Windows 10 последней версии. Данная версия операционной системы по умолчанию имеет графическую оболочку Gnome.

Для оформления отчета по проделанной работе был использовал пакет программ MicrosoftOffice и веб-приложение draw.io, которое использовалось для построения структурных и других схем.

Разработка программного продукта велась в программе1С: Предприятие.

Далее приведена схема структуры разрабатываемого приложения:

Рисунок 2.1.1 – СтруктураПО

Такие продукты, как интегрированные среды разработки, позволяют сделать разработку программного обеспечения более простой и наглядной. Также они влияют на скорость разработки программного обеспечения и более простую реализацию каких-либо сложных моментов.

2.2 Функциональная структура

Функциональная структура приложения предназначена для того, чтобы наглядно показать какие функции может предоставить приложение. Глядя на такую структуру, сразу становится ясным стоит ли использовать это приложение и приведет ли оно к желаемому результату.

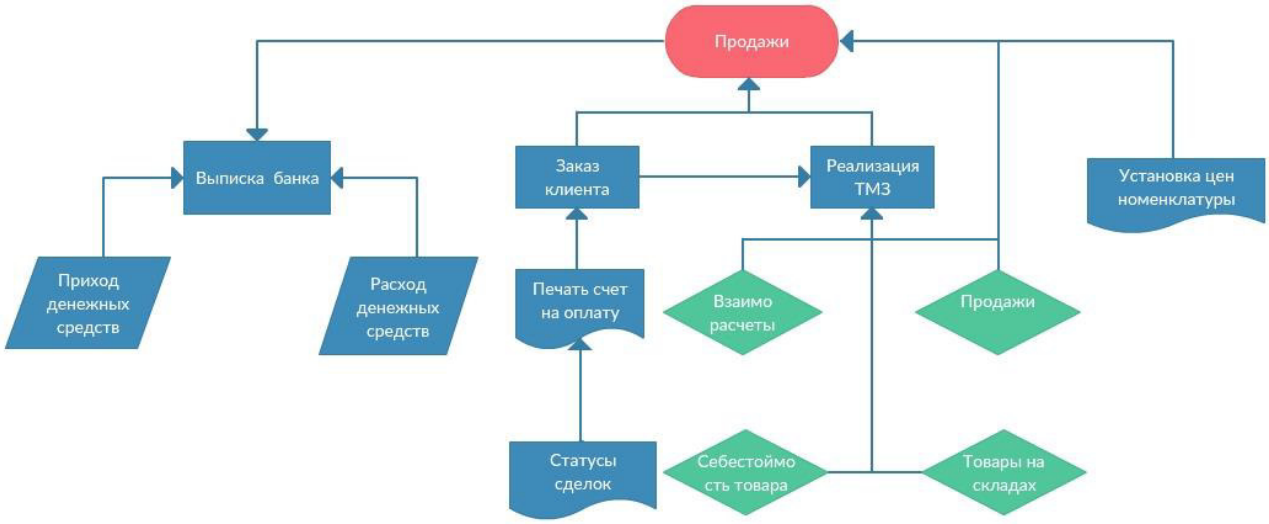

Рисунок 2.2.1 – Функциональная структура

2.3 Проектирование базы данных

В большинстве разрабатываемых приложений должна существовать база данных, в которой будет храниться информация, необходимая для корректной работы приложения и для его взаимодействия с пользователем.

Предполагается, что для базы данных всегда должен быть отведен достаточно большой объем памяти компьютера (или сервера). Поскольку одновременно база данных должна иметь доступ к огромнейшему количеству информации.

Проектирование базы данных – это всегда достаточно сложный процесс, который требует не только внимательности разработчика, но и проявление его умственных способностей. От того насколько точно будет продумана логика приложения, а затем и перенесена в таблицы базы данных, зависит насколько просто потом будет разработчику получать данные из этой таблицы.

При разработке предлагаемого приложения возникла необходимость создать свою базу данных, поскольку пользователю необходимо хранить данные, загружаемые в приложение и после его выхода из аккаунта, и отключения браузера. База данных предназначается для того, чтобы хранить в ней все данные пользователя, его переписку с другими пользователями, задачи пользователя вместе с их статусами, которые определяют степень выполнения задачи в данный момент.

Сначала может показаться, что нужно разрабатывать сложную базу данных, с большим количеством таблиц и связей в ней. Но на самом деле при

правильном подходе к построению таблиц и продумывании для чего нам нужны те или данные, выясняется, что данные можно хранить в более структурированном виде.

Именно этот структурированный вид и логическая продуманность базы позволяет ускорить процесс выполнения запросов в базу данных. А также иметь доступ к конкретному сегменту СУБД, а не искать нужную запись в хаотичном потоке информации, как это было бы, если бы данные хранились в обычном файле.

Также необходимо хранить в базе данных лишь те значения, которые действительно нужны для работы с приложением. Лишние значения, которые в дальнейшем в базе данных не используются, лишь нагружают ее и не дают в полной мере ощутить преимущества скорости работы современных СУБД. К тому же все СУБД обладают функцией, позволяющей нам в любой момент времени добавить нужные поля в любую из таблиц базы данных. Так что данное свойство позволяет разработчику не создавать поля в структуру таблицы впрок.

Также свойство транзакционности при пользовании базами данных позволяет избежать утечки данных из базы данных. Все запросы, отправляемые в базу данных, либо будут выполнены, либо нет, но данные при этом не исчезнут никуда, как это могло бы произойти, если бы для хранения данных приложения использовались обычные текстовые файлы.

В современном проектировании баз данных существует две основные технологии, согласно которым проектируется большинство актуальных на сегодняшний день баз данных: SQL и NoSQL, реляционные и нереляционные базы данных [17].

Различия этих технологий заключаются в том, какой принцип используется в каждой технологии для проектирования базы данных, какие типы информации поддерживают, как хранят информацию и разница в функционале, предоставляемом базами данных.

Реляционные базы отличаются тем, что хранят в себе описание каких-либо конкретных объектов. Это могут сведения о человеке, сотруднике, описание содержимого товарной корзины в интернет-магазине и т. д. Все эти данные хранятся в таблицах, сгруппированных по типу объектов. Формат таблиц для хранения объектов обычно задается на этапе проектирования базы данных.

Нереляционные базы данных работают по иному принципу. То, что в реляционных базах данных разбито на взаимосвязи нескольких таблиц, в нереляционной базе данных те же самые значения могут храниться в виде целостной самостоятельной сущности.

Каждая из множества систем управления базами данных имеет собственное внутренне устройство, которое влияет на методы работы с ней. Например, нереляционные базы данных лучше поддаются процессу

масштабирования, в то время как реляционные базы данных больше подходят для хранения уже структурированной информации [24].

Конечный выбор той или другой системы управления зависит от того, какие особенности имеет проект, для которого, собственно, и проектируется хранилище данных. На данных момент в большинстве крупных проектов одновременно используют оба типа систем управления базами данных.

Для приложения, разрабатываемого в рамках дипломного проекта, будет достаточно использования только реляционной базы данных, поскольку все данные, которыми оперируют компоненты приложения могут быть преобразованы в несколько взаимосвязанных таблиц. В данном случае структура базы данных составляется на этапе проектирования базы данных и мало подвержена дальнейшим изменениям. Поскольку приложения предоставляет конкретный неизменяющийся со временем функционал, то нет необходимости вольно обращаться с потоком предоставляемой информации и все время подстраивать его под определенные моменты.

Система управления базами данных – это совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

СУБД – это комплекс программ, позволяющих создать базу данных и манипулировать данными (вставлять, обновлять, удалять и выбирать). Система обеспечивает безопасность, надёжность хранения и целостность данных, а также предоставляет средства для администрирования БД.

Для проектирования базы данных в дипломном проекте была выбрана СУБД PostgreSQL. PostgreSQL – это одна из самых популярных объектно-реляционных систем управления базами данных на сегодняшний день и заслужила симпатии программистов по всему миру. Система была разработана в Калифорнийском университете в 1996 году. Эта СУБД является одной из самых новаторских.

Решения, которые были разработаны для работы с данными в этой СУБД, гораздо позднее появились в аналогичных системах управления. К тому же данная система является системой с открытым исходным кодом, а это в первую очередь влияет на скорость ее развития, так как чем больше человек задействовано в разработке ПО, тем более неожиданные решения могут появиться в перечне функциональных возможностей системы.

В процессе выбора реляционной СУБД для работы с проектом уделялось внимание лицензии, которая предоставлялась разработчиками. PostgreSQL имеет свободную лицензию, что позволяет использовать данную СУБД для любых необходимых целей, а также позволяет видоизменять функционал приложения опираясь на свои нужды. К тому же из всех известных СУБД PostgreSQL является самой развитой системой управления. Также свободная лицензия – это

большое преимущество перед другими, платными, СУБД, потому что стоимость лицензии других популярных СУБД может доходить до нескольких сотен тысяч долларов, что просто неприемлемо для небольших компаний.

Данная СУБД поддерживает большую часть официального стандарта SQL.

По мимо этого здесь реализован ряд важных современных функций для работы с базами данных: внешние ключи, сложные запросы, изменяемые представления, триггеры и т. п.

По истине уникальными функциями в работе с PostgreSQL является возможность создавать собственные элементы для работы с данными. Например, можно создать собственную функцию, оператор, агрегатные функции, методы индексирования, и даже собственные типы данных.

PostgreSQL имеет клиент-серверную архитектуру. Таким образом, есть главный серверный процесс, именуемый postgres. Он принимает запросы от клиента и оперирует с файлами базы данных. Этот серверный процесс может работать как локально на клиентской машине, так и находясь на сервере. Одновременно серверный процесс может принимать подключения сразу от нескольких клиентов и работать с ними. Он все время находится в ожидании подключения клиентов.

В качестве клиента используется специальное клиентское приложение, которое имеет возможность выполнять запросы в базе данных. Клиентские приложения очень разнообразны: различные текстовые утилиты, графические приложения, либо другой специализированный инструмент для работы с базой данных.

Использование подобной СУБД является мощным заделом к будущему развитию программного продукта. Гораздо проще сразу создать хранилище данных с использованием современного инструмента, который в будущем сможет удовлетворить все необходимые потребности, нежели переводить проект с одной СУБД на другую.

Перевод проекта на новую СУБД процесс сложный и дорогой. Может возникнуть проблема утечки информации или конфликта типов данных, что усложнит этот процесс. К тому же СУБД PostgreSQL активно поддерживается, дорабатывается, совершенствуется и обновляется по сегодняшний день.

После выбора СУБД можно переходить к анализу предметной области и проектированию схемы базы данных.

Анализ предметной области представляет собой один из наиболее важных этапов в разработке приложения. Этот этап позволяет выявить все данные, которые потребуются для корректного взаимодействия пользователя с приложением, а также учтут какие данные пользователь захочет хранить в базе данных.

Для того, чтобы не забыть добавить данные из какой-либо области приложения, необходимо поэтапно переходить к каждому функционалу приложения от начала взаимодействия с пользователем и анализировать, какие данные окажутся востребованными для корректной реализации данного функционала.

Первое взаимодействие с системой: вход в приложение.

Для того, чтобы пользователь мог осуществить вход в систему ему необходимо иметь собственный идентификатор и пароль. В качестве идентификатора может выступать какой-либо уникальный номер, имя пользователя, или, например, номер телефона. В данном случае мы будем использовать номер телефона, потому что он уникален для каждого пользователя, а также потому, что при помощи вывода этого поля рядом с именем пользователя можно предоставить номер пользователя, который будет использован для срочной связи служащих посредством мобильного телефона.

Пароль – какой-либо набор символов, придуманный пользователем. Единственное особое условие этого поля состоит в том, что пароль пользователя хранится в базе данных в зашифрованном виде, и недоступен для просмотра системному администратору команды [20].

В том случае, если пользователь не зарегистрирован, то от формы входа он направится к форме регистрации. Здесь уже будут использоваться два ранее рассмотренных свойства: номер телефона, который служит идентификатором пользователя и пароль от аккаунта пользователя. Также здесь необходимо будет добавить поля имени и фамилии пользователя, для удобства поиска необходимого сотрудника, его должность.

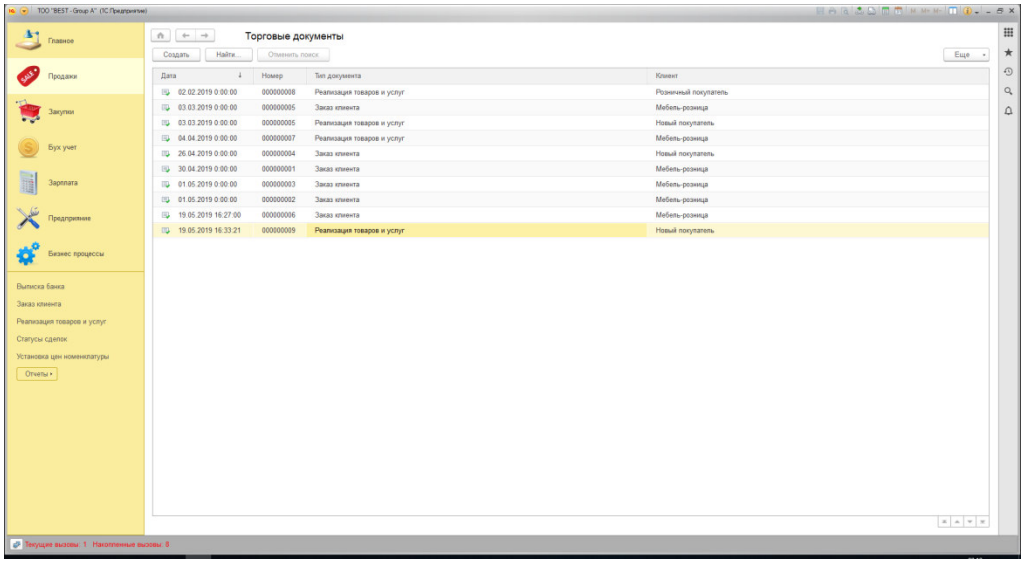

Рисунок 2.3.1 – Запрос для создания таблицы

В итоге подошел момент к созданию первой таблицы базы данных с названием «Пользователи». Эта таблица будет хранить в себе все данные относительно каждого пользователя в полях, названия которых были перечислены выше: номер телефона, имя, фамилия, должность, пароль.

Результат запроса на все данные из таблицы, которая уже заполнена тестовыми значениями выглядит следующим образом:

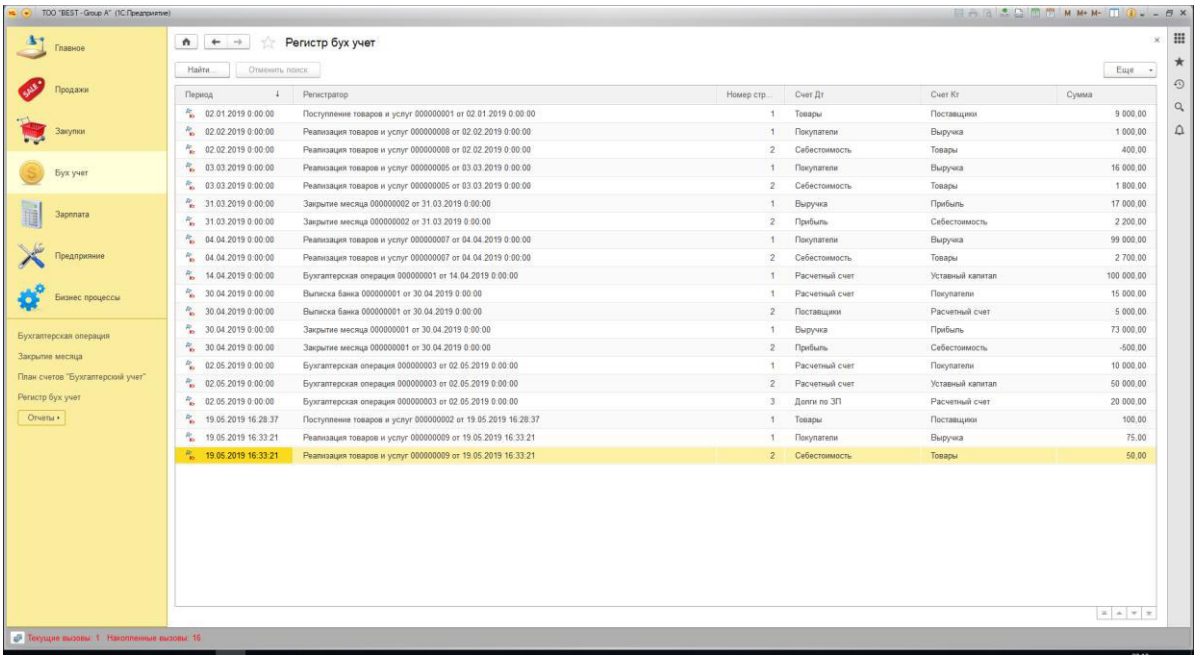

Рисунок 2.3.2 – Данные таблицы

После входа в систему пользователь может выбрать каким инструментом ему воспользоваться: чатом или планировщиком.

Для того, чтобы в базе данных могли храниться все сообщения из всех переписок пользователей, необходимо, чтобы каждое сообщение хранило о себе следующие данные: отправитель сообщения, получатель сообщения, сам текст сообщения, идентификатор сообщения, а также время отправки сообщения. Далее рассматривается каждое из поле и его пригодность.

Поле, которое хранит в себе идентификатор отправителя сообщения хранится для того, чтобы у получателя корректно отображалось от кого пришло данное сообщение, а также при распределении сообщений между чатами будет сразу понятно, чату с каким пользователем принадлежит данное сообщение.

Поле, хранящее в себе идентификатор получателя сообщения, предназначено для того, чтобы сообщение пришло именно к конкретному

получателю в конкретный чат с пользователем, а не какому-либо случайному пользователю, для которого это сообщение вообще не предназначалось.

Поле с текстом сообщения хранит в себе само сообщение, которое пересылается от пользователя к пользователю.

Идентификатор сообщения необходим для того, чтобы каждое сообщение было уникальным, и разработчик при необходимости мог с легкостью добраться для каждого из сообщений. Это поле пригодится для дальнейшего совершенствования программы, когда появится возможность пересылать сообщения между пользователями или вовсе удалять их.

Время отправки сообщения фиксируется для пользователя. Например, если пользователю пришло сообщение, которое он не прочитал сразу, то возможно к тому моменту, когда он его прочитает оно уже потеряет актуальность и пользователь будет знать об этом. Время отправки сообщения – это поле, которое на самом деле несет в себе много пользы для использования чата.

В итоге будет создана вторая таблица базы данных с названием «Сообщения». Эта таблица будет хранить в себе все данные относительно каждого сообщения в полях, названия которых были перечислены выше: отправитель сообщения, получатель сообщения, текст сообщения, идентификатор сообщения, время отправки сообщения.

Запрос на создание таблицы в базе данных выглядит следующим образом:

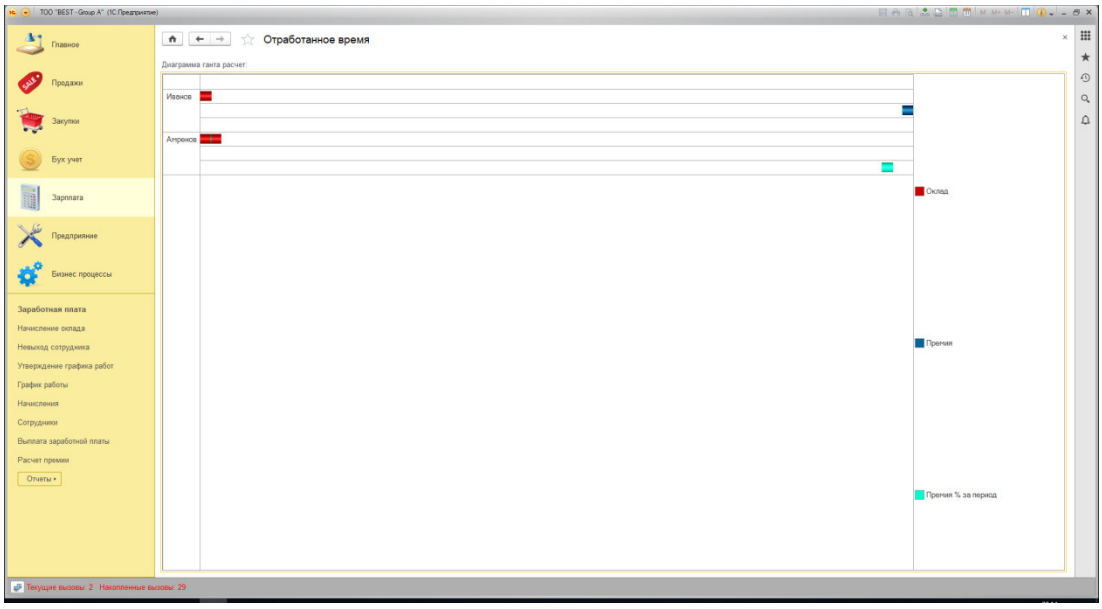

Рисунок 2.3.3 – Запрос для создания таблицы

Результат запроса на все данные из таблицы, которая уже заполнена тестовыми значениями выглядит следующим образом:

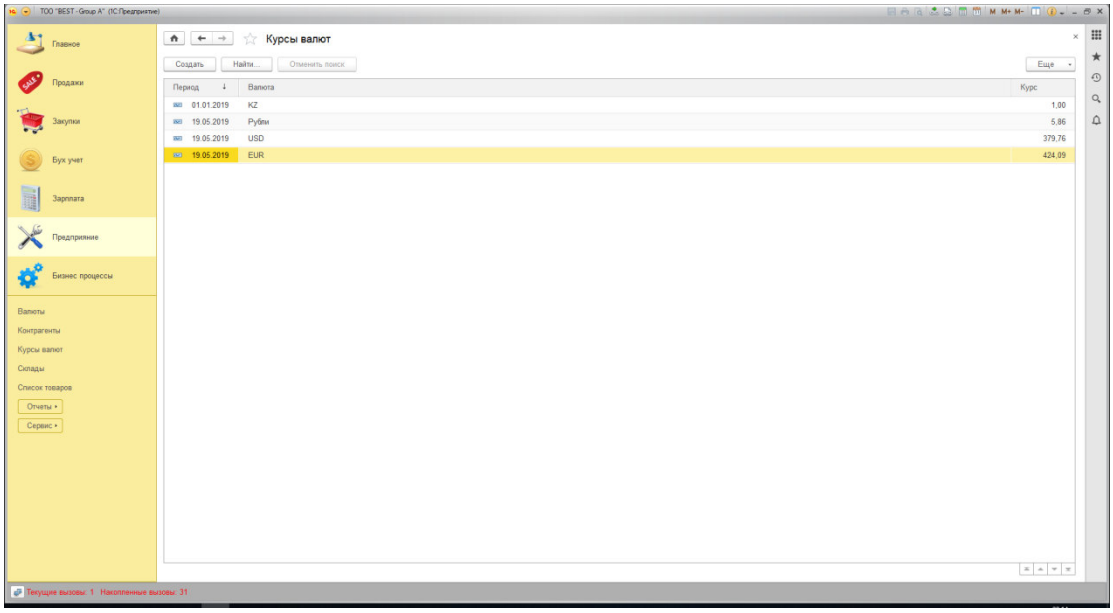

Рисунок 2.3.4 – Данные таблицы

Далее осталось создание еще одной таблицы, которая будет хранить в себе результаты работы с планировщиком.

Каждая задача, добавляемая в хранилище базы данных, имеет своего пользователя, который создал эту задачу. Таким образом, можно выбирать только те задачи из базы, которые принадлежат конкретному пользователю и отображать ему только его задачи. Это поможет избежать появления в списке задач пользователя тех задач, которые ему не принадлежат.

Далее, каждая задача представляет собой какое-либо слово, набор слов или предложение. Это является самым основным полем в таблице, которое несет в себе главную информацию во всем сервисе.

Далее идет дата выполнения задачи. В это поле базы данных записывается та дата, на которую записывается задача. Это поле позволяет пользователю из всего списка задач, выбрать те задачи, которые относятся к определенной дате. Таким образом, заходя в планировщик, пользователь первым делом видит все те задачи, которые предназначены для выполнения в текущую дату.

Каждая задача имеет статус прогресса. Статус прогресса задачи показывает на каком этапе сейчас находится выполнение задачи. Пользователь мог не приступать к выполнению задачи, что будет соответствовать статусу «не выполнялась», либо мог уже выполнить задачу, значит, статус в данном случае будет иметь значение «выполнена». Также статус имеет два промежуточных состояния: задача может находиться «в процессе» выполнения, либо может быть «отложена», если пользователю надобилось переключится на другую задачу,

либо он ждет какого-либо действия другого сотрудника, чтобы продолжить работу.

В итоге будет создана третья таблица базы данных с названием «Задачи». Эта таблица будет хранить в себе все данные относительно каждой задачи в полях, названия которых были перечислены выше: пользователь, текст задачи, статус прогресса задачи, дата выполнения задачи.

Запрос на создание таблицы в базе данных выглядит следующим образом:

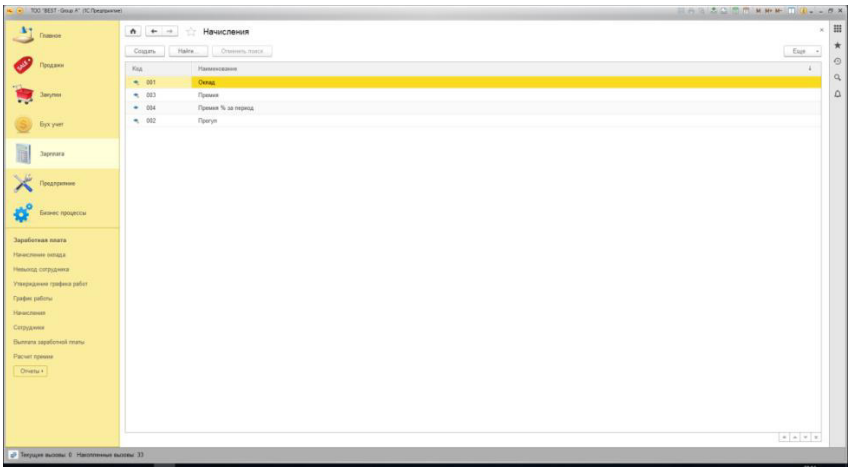

Рисунок 2.3.5 – Запрос для создания таблицы

Результат запроса на все данные из таблицы, которая уже заполнена тестовыми значениями выглядит следующим образом:

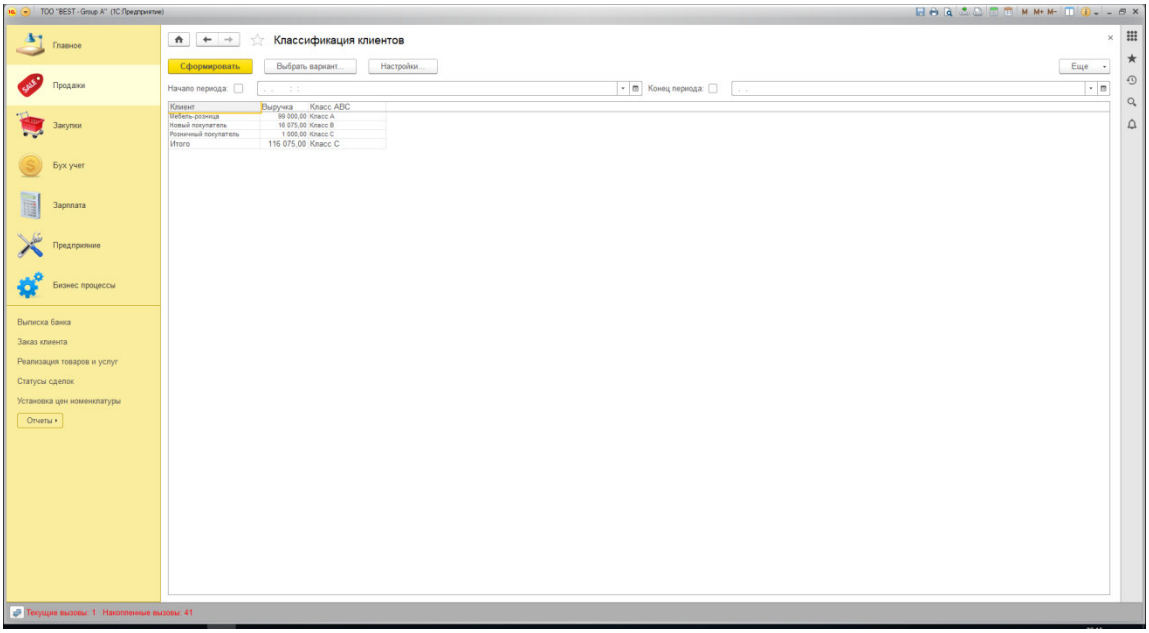

Рисунок 2.3.6 – Данные таблицы

Далее представлена структура базы данных, которая показывает связи между таблицами в базе данных, и обобщенный вид всех таблиц, включенных в базу данных.

Создание таких таблиц позволяет наглядно продемонстрировать не только саму структуру базы данных, но и описывает все данные, которые в дальнейшем должны использоваться в приложении. Каждая таблица на схеме выделена в отдельный прямоугольник, в котором указывается название таблицы и все ее поля. Ключевые поля в таблицах выделяются специальным символом слева от названия поля.

Также схема демонстрирует связь таблиц по конкретным полям этих таблиц. Эта связь демонстрирует взаимоотношения данных таблиц баз данных.

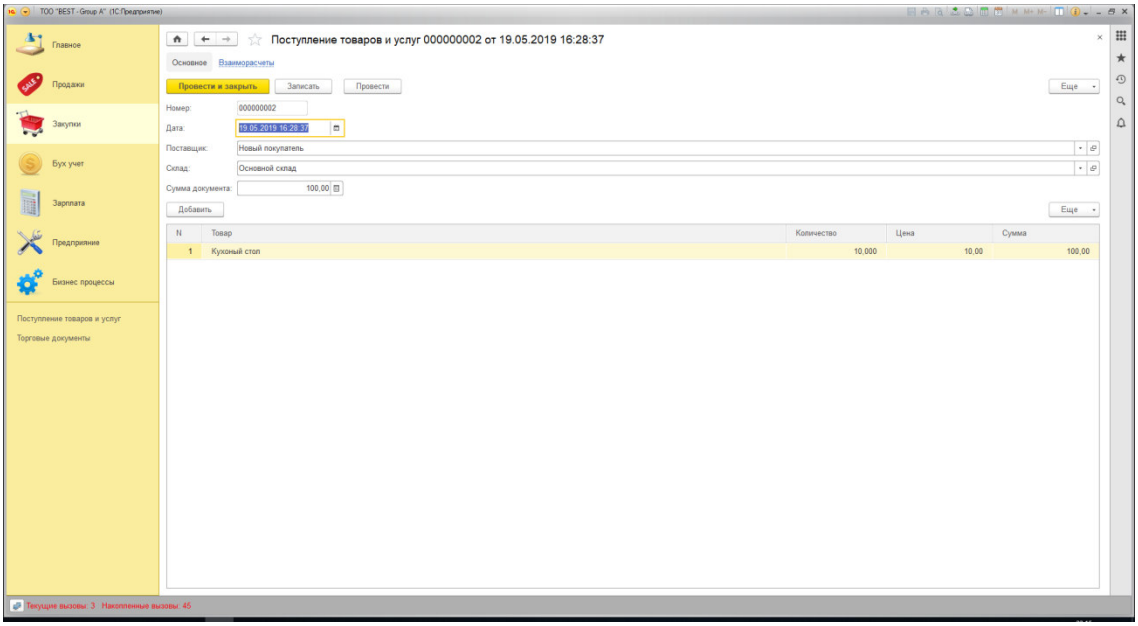

Рисунок 2.3.7 – Структура базы данных

2.4 Инструменты для работы с базой данных

В условиях разрабатываемого проекта непосредственная работа с базой данных происходила под руководством плагина DatabaseSupport для работы с базами данных в Intellij IDEA от компании-разработчика JetBrains [23]. Плагин позволяет работать с базами данных напрямую из интегрированной среды разработки. Это одно из самых полезных свойств среды разработки, которое позволяет проводить разработку и отладку продукта с использованием только одного приложения.

Также стоит отметить, что данный плагин является универсальным средством для работы с базами данных. Интерфейс и внешний вид данного приложения абсолютно не зависит от типа базы данных, которую подключает разработчик. В случае первичного использования СУБД на данном компьютере плагин сам предложит уставить все необходимые для работы плагины, что очень упрощает задачу разработчику и сокращает время на поиск необходимой версии драйвера.

Так как основным языком программирования для приложеня является высокоуровневый язык программирования Java, то для соединения приложения с базой данных необходимо было выбрать фреймворк. Несмотря на то, что большинство разработчиков считают фреймворк Hibernate стандартным выбором для таких действий, выбор был сделан в пользу альтернативного фреймворка MyBatis, который имеет ряд преимуществ, которые были прописаны в первом разделе. Поэтому здесь будет конкретное описание как работать с этим фреймворком.

MyBatis связывает методы интерфейса маппера с SQL-запросом. Данный фреймворк не умеет создавать схемы и вообще никак не соприкасается с ней в ходе свой работы. Он лишь превращает вызов метода в реальный запрос в базу данных, а затем, как результат вызова метода, возвращает то, что вернулось бы нам, если бы мы просто выполняли этот запрос с помощью специального приложения.

3 Разработка программного продукта

Клиент-серверное приложение представляет собой приложение, которое имеет две части: клиент и сервер. Такие приложения в основном отображаются и взаимодействуют с пользователем.

Клиент, это та часть приложения, которая отображается пользователю, выполняется в веб-браузере и взаимодействует визуально с пользователем. На этой стороне работают такие языки разметки, стилей и программирования как 1С.

Серверная часть приложения не имеет собственного визуального представления и взаимодействует с пользователем через. Название этой части вытекает из того, что все действия выполняются на сервере – специальном компьютере, который может быть расположен как за тысячи километров от браузера, так и в непосредственной близости, вплоть до одной машины. На сервере обычно располагается база данных и оперируют такие языки как Java, PHP, C# и т. д. Данное приложение разрабатывается на языке программирования Java.

В начале разработки программного продукта необходимо создать проект, который будет включать в себя все необходимые средства для разработки данного проекта.

Для этого нужно зайти на сайт start.spring.io и выбрать все необходимые параметры для создания проекта. Сборщиком проектов выбран Gradle, язык программирования Java, версия SpringBoot 2.1.4. Далее идут метаданные, которые указываются для проекта. Артефакт проекта по сути является названием проекта.

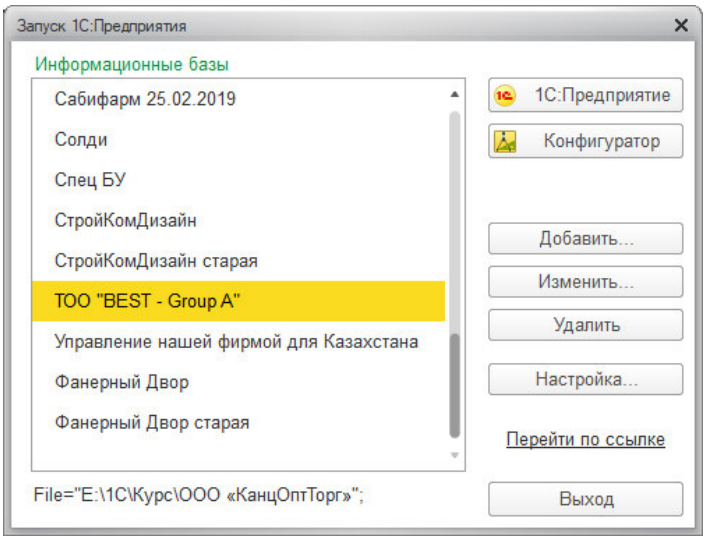

Рисунок 3.1 – Настройка сборки проекта

После того, как мы заполнили первую часть формы переходим ниже и выбираем все технологии, которые необходимы для разработки:

Рисунок 3.2 – Выбор необходимых технологий

Подключив в проект все необходимые зависимости генерируется проект командой «alt+Enter» и скачивается на компьютер.

Вся дальнейшая разработка проводится в интегрированной среде разработки Intellij IDEA.

В первую очередь в новом проекте необходимо создать клиентскую часть проекта. Для этого прописывается команда:

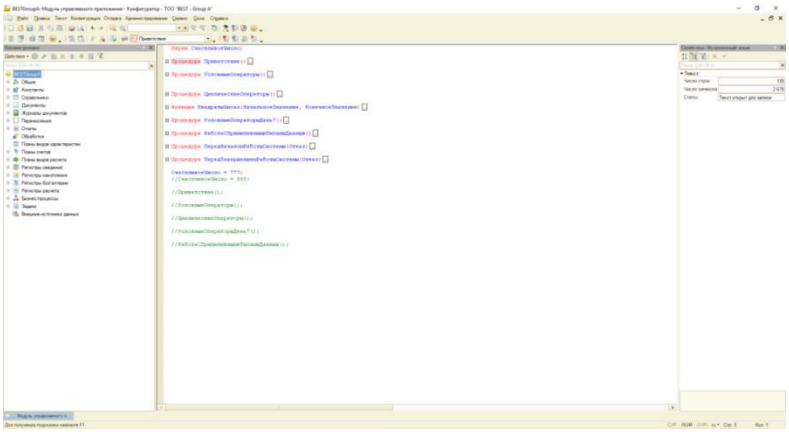

Рисунок 3.3 – Команда создания клиентской части проекта

Этой командой в папке с данным проектом создается папка client, в которой будет храниться вся клиентская часть. Как раз в этот момент к работе подключается фреймворк Angular, который внутри папки client создает главный компонент приложения app.component. Далее в папке client подтягиваются все необходимые зависимости следующей командой:

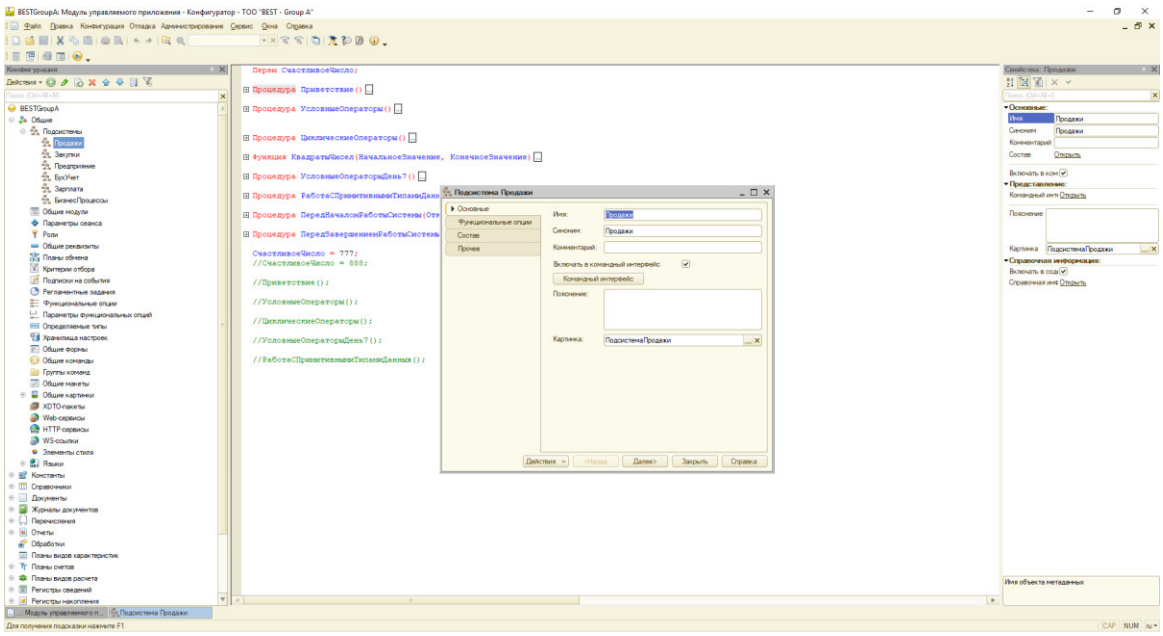

Рисунок 3.4 – Команда сборки необходимых зависимостей

Теперь можно переходить к непосредственной разработке клиентской части проекта. Так как используется фреймворкAngular, то все части проекта создавались в виде отдельных компонентов. Таким образом в ходе разработке получилось 5 компонентов, каждый из которых ответственен за свою роль в функционировании приложения:

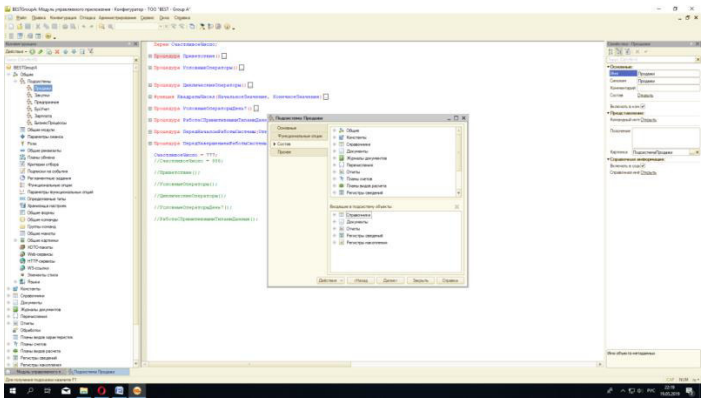

Рисунок 3.5 – Структура клиентской части проекта

– компонент header: компонент, который отвечает за формирование шапки сайта на каждой из страниц приложения;

– компонент login-page: компонент, отвечающий за форму входа и ее функционирование;

– компонент registration: компонент, отвечающий за регистрацию пользователя в приложении;

– компонент chat: компонент, отвечающий за функционирование пользовательского чата;

– компонент planner: компонент, отвечающий за функционирование планировщика задач.

В Angular главная страница или app.component создается лишь однажды на весь проект, другие компоненты лишь подставляются в него в зависимости от действий пользователя.

За корректное переключение между компонентами в браузере отвечает маршрутизация, специальная возможность фреймворка Angular которая позволяет сопоставлять запросы к приложению с определенными запросами внутри приложения.

Ключевым модулем для работы этой возможности является RouterModule, который располагается в @angular/router. Для этого необходимо записать RouterModule в список модулей файла app.module.ts.

Для указания путей перехода по разным страницам создается специальный файл app-routing.module.ts. В файле указывается массив объектов, каждый из которых содержит в себе путь к компоненту в браузере и название компонента, который будет загружаться по данному пути:

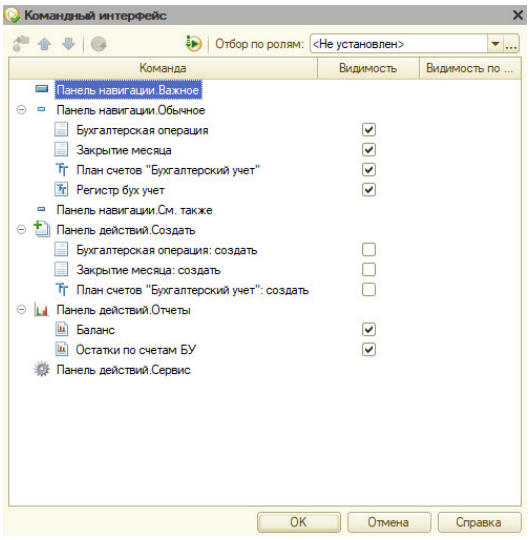

Рисунок 3.6 – Роутинг между адресами и компонентами

Далее приводится описание каждого из компонентов приложения.

Каждый компонент сохраняется в отдельной папке и имеет в себе три файла: файл html-разметки, файл css-стилей и файл функционала компонента, написанный на языке Typescript.

3.1 Компонент header

Этот компонент представляет из себя верхнюю часть сайта, которая отображается в том или другом виде при открытии любого другого компонента. Таким образом, он является неотъемлемым и основным компонентом приложения.Структура этого компонента выглядит следующим образом:

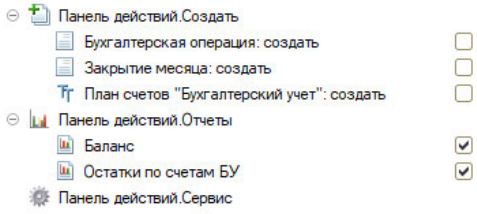

Рисунок 3.1.1 – Структура компонента

Компонент имеет два основных состояния: вход выполнен и вход не выполнен.

При состоянии, когда вход не выполнен в левом верхнем углу отображаются кнопки входа и регистрации, и больше не отображается ничего:

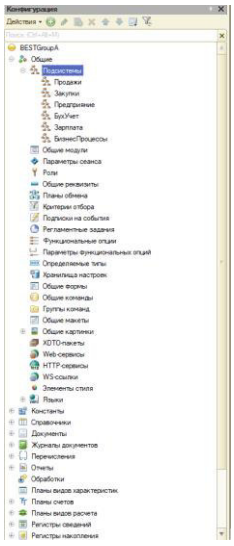

Рисунок 3.1.2 – Верхняя часть приложения до входа в систему

Когда вход выполнен эти кнопки исчезают, на их место ставится иконка пользователя, с возможностью выхода, а также появляется навигация по приложению в виде ссылок на чат и планировщик.

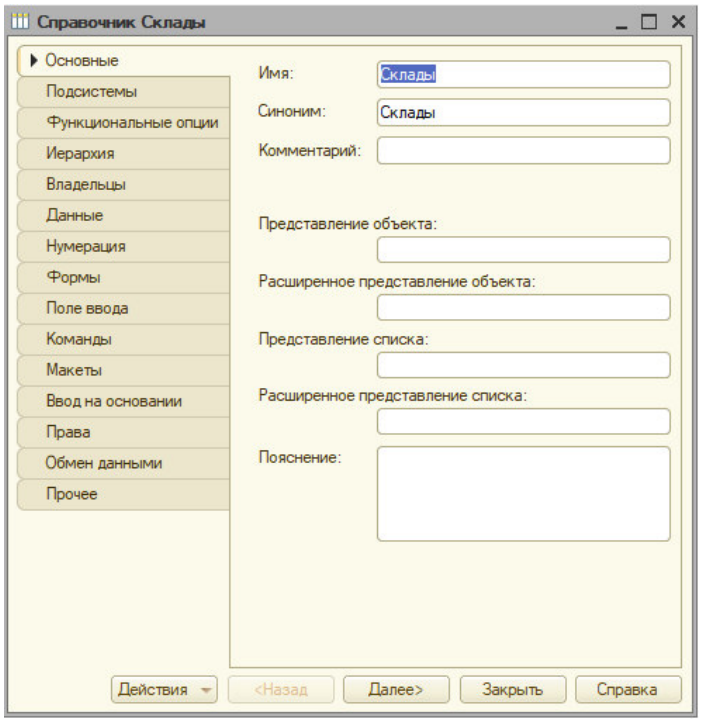

Рисунок 3.1.3 – Вход выполнен

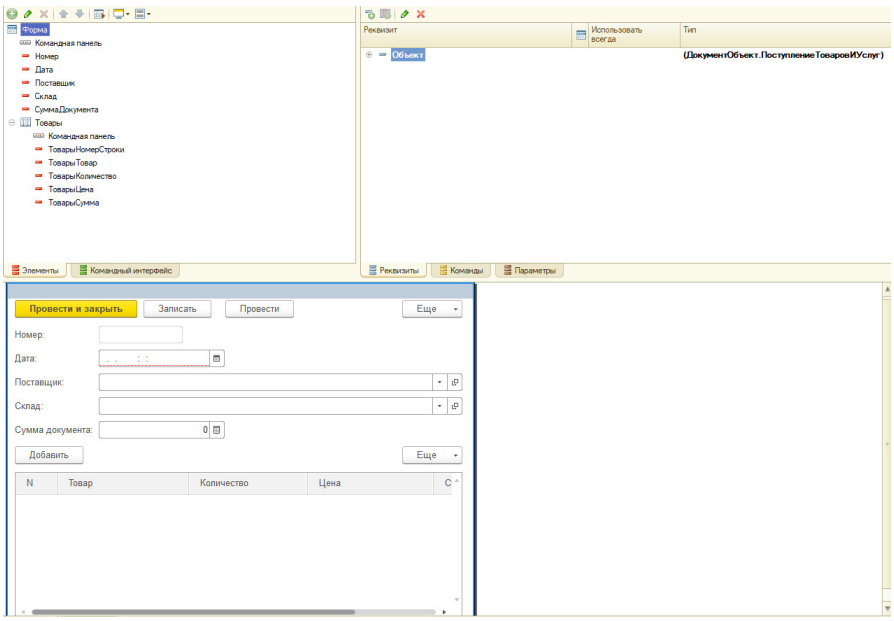

Рисунок 3.1.4 – Вход выполнен. Меню для выхода из аккаунта пользователя.

Эти состояния переключаются при помощи функции isLogin(), которая с помощью компонентов чата и сервиса loginService возвращает логическое значение, сообщающее о том, выполнен ли вход в приложение:

```
Процедура ПередЗавершениемРаботыСистемы(Отказ)

    ТекущаяДата1800 = КонецДня(ТекущаяДата())-6*3600+1;
    Если ТекущаяДата()<ТекущаяДата1800 Тогда

        //Сообщить("Солнце еще высоко!");
        //Отказ = Истина;

    КонецЕсли;

КонецПроцедуры
```

Рисунок 3.1.5 – Функция

В разметке страницы эта функция применяется следующим образом:

```
Процедура ЦиклическиеОператоры()

    ИскомаяСумма = КвадратыЧисел(10, 15);
    Сообщить(ИскомаяСумма);

    ИскомоеЧисло = 1;
    Пока ИскомоеЧисло*ИскомоеЧисло<30000 Цикл
        //Продолжить;
        ИскомоеЧисло = ИскомоеЧисло + 1;
    КонецЦикла;

    Сообщить(ИскомоеЧисло-1);

КонецПроцедуры

Функция КвадратыЧисел(НачальноеЗначение, КонечноеЗначение)

    СуммаКвадратов = 0;

    Для Счетчик = НачальноеЗначение По КонечноеЗначение Цикл

        Сообщить(Счетчик);
        Сообщить(Счетчик*Счетчик);

        СуммаКвадратов = СуммаКвадратов+Счетчик*Счетчик;

    КонецЦикла;

    Возврат СуммаКвадратов;

КонецФункции
```

Рисунок 3.1.6 – Разметка верхней части сайта

3.2 Компонент login-page

Компонент login-page представляет собой форму входа пользователя в приложение. Данная форма предлагает пользователю ввести его номер телефона (идентификационное значение каждого пользователя) и пароль для того, чтобы осуществить вход в аккаунт.

Структура этого компонента выглядит следующим образом:

```
Процедура УсловныеОператорыДень7()

    Если СчастливоеЧисло>=800 И СчастливоеЧисло<=1000 Тогда
        Сообщить("Счастливое число входит в диапазон от 800 до 1000");
    КонецЕсли;

    Если СчастливоеЧисло<800 ИЛИ СчастливоеЧисло>1000 Тогда
        Сообщить("Счастливое число не входит в диапазон от 800 до 1000");
    КонецЕсли;

    Если НЕ (СчастливоеЧисло>1000) Тогда
        Сообщить("Счастливое число меньше или равно 1000");
    КонецЕсли;

КонецПроцедуры
```

Рисунок 3.2.1 – Структура компонента

В том случае, если пользователь не зарегистрирован в системе и является новым для нее сотрудником, то форма предлагает пользователю перейти на страницу регистрации.

Сейчас будет рассматриваться тот случай для этой формы, если у пользователя есть логин и пароль.

В данном случае пользователь вводит данные в установленные для этого поля и нажимает на кнопку «Вход». В этот момент все данные, введенные пользователем, отправляются на проверку, и система принимает решение аутентифицировать этого пользователя или его данные не подходят ни к одному из зарегистрированных пользователей. Отправляются данные посредством функции login() при срабатывании обработчика формы submit().

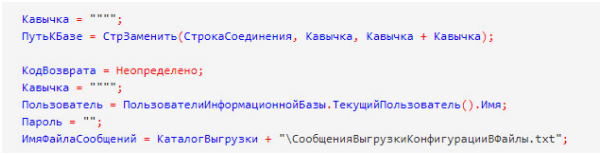

Рисунок 3.2.2 – Функция

Данная функция вызывает сервисную функцию login(), чтобы отправить данные на сервер и сравнить с данными, которые хранятся в базе данных. После получения данных от сервисной функции приложение пытается перейти в компонент chat и в том случае, если данные верны, то пользователь попадает в приложение, иначе высвечивается сообщение об ошибочном имени пользователя или пароле.

```
ЗапуститьПриложение(Кавычка + КаталогПрограммы + "1cv8.exe" + Кавычка + " DESIGN
    + " /IBConnectionString " + Кавычка + ПутьКБазе + Кавычка
    + " /N "                  + Кавычка + Пользователь + Кавычка
    + " /P "                  + Кавычка + Пароль + Кавычка
    + " /DumpConfigToFiles "  + Кавычка + КаталогВыгрузки + Кавычка
    + " /Out "                + Кавычка + ИмяФайлаСообщений + Кавычка
    + " /DisableStartupMessages /DisableStartupDialogs "
    ,
    ,
    Истина,
    КодВозврата);
```

Рисунок 3.2.3 – Функция

Сервисная функция login() отправляет по URL-адресу содержащему «/api/login» те данные, которые пользователь ввел в форму перед ее отправкой. В случае получения данных с сервера в ответ на этот запрос, функция устанавливает токен для данного соединения, сохраняет имя пользователя, который совершил вход в свой аккаунт, а также инициализирует соединение для общения в чате.

Далее данные из сервисной функции оказываются в функции createAuthenticationToken(). Эта функция предназначена для попытки создания токена для аутентификации пользователя.

Токен – специальное зашифрованное значение, хранящееся в кэше браузера и позволяющее пользователю, находится в приложении. До тех пор, пока токен хранится в кэше пользователь является аутентифицированным и может пользоваться приложением, но как только токен удалится – пользователь больше не сможет использовать приложение.

```
Функция ПолучитьТаблицуАвтоматическихОбменовДанными(ТолькоДляПервогоВходаВПрограмму = Ложь,
            ТолькоДляВходаВПрограмму = Ложь, ТолькоДляВыходаИзПрограммы = Ложь, ОграничениеПоДнюНедели = Истина,
            ССылкаНаНастройку = Неопределено, ОграничиватьПоПользователю = Истина)

    // есть ли доступ к настройкам обмена данными
    Если НЕ ПравоДоступа("Чтение", Метаданные.Справочники.НастройкиВыполненияОбмена)
        ИЛИ НЕ ПравоДоступа("Чтение", Метаданные.РегистрыСведений.ПараметрыОбменаДанными) Тогда

        Возврат Неопределено;

    КонецЕсли;

    Запрос = Новый Запрос;

    ДополнительныеОграничения = "   НЕ  НастройкиОбменаДанными.ПометкаУдаления
                                |";

    // строим ограничения в зависимости от параметров
    Если ТолькоДляПервогоВходаВПрограмму Тогда
        ДополнительныеОграничения = "   ПараметрыОбменаДанными.ВыполнятьПриПервомВходеВСистему";

    ИначеЕсли ТолькоДляВходаВПрограмму Тогда
        ДополнительныеОграничения = ДополнительныеОграничения + "
                                    |   И   НастройкиОбменаДанными.Ответственный = &ТекущийПользователь
                                    |   И   НастройкиОбменаДанными.КаждыйЗапускПрограммы";

    ИначеЕсли ТолькоДляВыходаИзПрограммы Тогда
        ДополнительныеОграничения = ДополнительныеОграничения + "
                                    |   И   НастройкиОбменаДанными.Ответственный = &ТекущийПользователь
                                    |   И   НастройкиОбменаДанными.КаждоеЗавершениеРаботыСПрограммой";
```

Рисунок 3.2.4 – Функция

Перед тем, как данная функция переходит к созданию нового токена, она вызывает функцию authenticate(). Функция authenticate() принимает в качестве параметров значения логина и пароля, а затем проверяет их на валидность. Сначала значения проверяются на то, являются ли они хранящими значение null. В том случае, если значения присутствуют, проверяемые значения сравниваются с теми значениями, которые хранятся в базе данных приложения.

```
// параметры блокировки
Блокировка = Новый БлокировкаУстановкиСоединений;

Блокировка.Начало           = НачалоБлокировки;
Блокировка.Конец            = ОкончаниеБлокировки;
Блокировка.Сообщение        = Сообщение;
Блокировка.Установлена      = УстановитьБлокировкуСоединений;
Блокировка.КодРазрешения    = КодРазрешения;
```

Рисунок3.2.5 – Функция authenticate () изLoginController.java

Если данные совпадают, то создается новый токен и отправляется назад к клиенту, а если нет, то на клиент отправляется ошибка и пользователю будет сообщено о том, что он ввел неправильные значения для входа в систему.

Также здесь следует упомянуть о том, что в приложении реализован функционал, который ограничивает пользователя в просмотре страниц до тех пор, пока он не прошел аутентификацию. В том случае, если пользователь попытается перейти к чату или планировщику, он будет перенаправлен на

страницу с компонентом входа и будет перенаправляться на нее до тех пор, пока не осуществит вход в приложение.

3.3 Компонент registration

Компонент registration предназначен для регистрации пользователя, в том случае если он является новым сотрудником для компании. При регистрации ему необходимо ввести определенный минимум валидных для базы данных значений и тогда он сможет выполнить вход в приложение и начать его использование.

Структура этого компонента выглядит следующим образом:

```
    // установка блокировки соединений
    УстановитьБлокировкуУстановкиСоединений(Блокировка)
КонецПроцедуры // УстановитьБлокировку()
```

Рисунок 3.3.1 – Функция

Когда пользователь ввел необходимые данные он нажимает на кнопку «Зарегистрироваться». В этот момент срабатывает обработчик событий для формы submit() и вызывается к исполнению функция toRegister().

В первую очередь данная функция проверяет введенные пароли. Если пароли совпадают, то функция запускает процесс отправки данных на сервер, в ином случае пользователю сообщается, что пароли не идентичны и нужно заполнить эти поля заново.

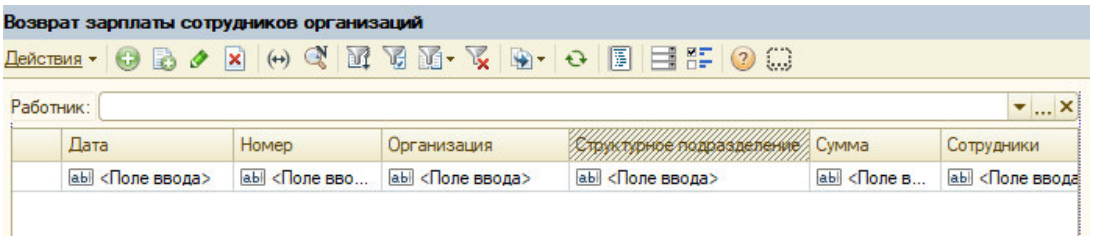

Рисунок 3.3.2 – Компонент регистрации

```
// Процедура - обработчик события "ПередОткрытием" формы. //...
Процедура ПередОткрытием(Отказ, СтандартнаяОбработка)

    // Установка подменю "Советы".
    РаботаСДиалогами.УстановитьПодменюСоветы(ЭлементыФормы.ДействияФормы);
КонецПроцедуры // ПередОткрытием()
```

Рисунок 3.3.3 – ФункциятоRegister() изregistration.component.ts

При правильно введенных паролях вызывается функция registration() из LoginService и все данные пользователя передаются по URL-адресу содержащему «/api/registration». В случае корректного ответа от сервера пользователю высвечивается окно с информацией о том, что регистрация прошла успешно. В обратном же случае и сообщение сообщает пользователю, что регистрация не удалась.

```
// Процедура открывает форму категорий документа //...
Процедура ДействияФормыДействиеОткрытьКатегории(Кнопка)
    Если ЭтаФорма.ЭлементыФормы.ДокументСписок.ТекущаяСтрока = Неопределено тогда
        Возврат
    КонецЕсли;
    РаботаСДиалогами.ОткрытьКатегорииДокумента(ЭлементыФормы.ДокументСписок.ТекущиеДанные.Ссылка, ЭтаФорма);
КонецПроцедуры
```

Рисунок 3.3.4 – Функция

В контроллере выполняется метод registration(), который прежде, чем отправить на клиент результат о положительной регистрации проверяет два условия.

Во-первых, проверяется существует ли в базе данных другой пользователь с таким же номером телефона. В базе данных не должно существовать две записи с идентичными номерами, поскольку номера телефонов являются уникальными для любого человека. Поэтому если будет найден пользователь с таким же номером телефона, то процесс регистрации будет прерван и пользователю вернется сообщение о неудачной попытке регистрации.

```
// Процедура вызывается при выборе пункта подменю "Структура подчиненности" меню "Перейти". //...
Процедура ДействияФормыСтруктураПодчиненностиДокумента(Кнопка)

    Если ЭлементыФормы.ДокументСписок.ТекущиеДанные = Неопределено тогда
        Возврат
    КонецЕсли;

    РаботаСДиалогами.ПоказатьСтруктуруПодчиненностиДокумента(ЭлементыФормы.ДокументСписок.ТекущиеДанные.Ссылка);

КонецПроцедуры // ДействияФормыСтруктураПодчиненностиДокумента()
```

Рисунок 3.3.5– МетодcheckPhoneNumber() из UserService.java

Затем будет проверяться условие, обозначающее тот случай, если пользователь не может быть зарегистрирован. То есть, по сути, происходит процесс регистрации и в ходе его выполнения отслеживаются случайные ошибки.

Для осуществления процесса регистрации вызывается метод registry() из UserService.java.

```
// Процедура - обработчик события "ПередОткрытием" формы. //...
Процедура ПередОткрытием(Отказ, СтандартнаяОбработка)

    // Установка подменю "Советы".
    РаботаСДиалогами.УстановитьПодменюСоветы(ЭлементыФормы.ДействияФормы);
КонецПроцедуры // ПередОткрытием()
```

Рисунок 3.3.6 – Методregistry() изUserService.java

Первоначально этот метод шифрует значение пароля, чтобы в базе данных пароли хранились в закрытом доступе. Далее при помощи интерфейса UserDao и функции saveUser() происходит сохранение данных пользователя в базу данных.

Сохранение данных осуществляется при помощи фреймворка MyBatis, который при помощи аннотаций позволяет посылать запросы на выполнение в базу данных. В данном случае используется аннотация @Insert и соответствующий ей insert-запрос, который и записывает данные о пользователе в необходимую таблицу.

```
Перем   КартинкаВыполнено;
Перем   КартинкаИсправлено;
Перем   КартинкаОшибок;
Перем   КартинкаУдаления;
```

Рисунок 3.3.7 – Метод saveUser() из UserDao.java

3.4 Компонент chat

Компонент chat представляет собой сервис, который предоставляет услуги для общения пользователей системы между собой.

Структура этого компонента выглядит следующим образом:

```
// Процедура вызова структуры подчиненности документа //...
Процедура ДействияФормыСтруктураПодчиненностиДокумента(Кнопка)

    Если ЭлементыФормы.ДокументСписок.ТекущиеДанные = Неопределено тогда
        Возврат
    КонецЕсли;

    РаботаСДиалогами.ПоказатьСтруктуруПодчиненностиДокумента(ЭлементыФормы.ДокументСписок.ТекущиеДанные.Ссылка);

КонецПроцедуры
```

Рисунок 3.4.1 – Метод saveUser() из UserDao.java

При переходе в навигации к этому компоненту перед пользователем открывается страница с чатом.

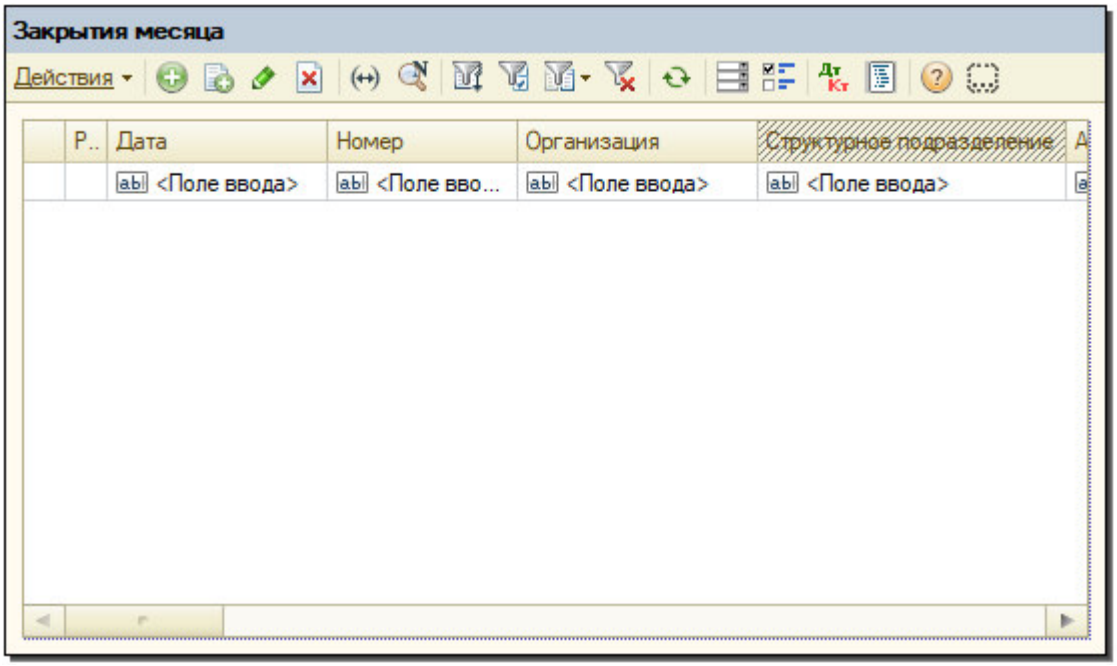

Рисунок 3.4.2

Слева на странице располагается список всех доступных ему контактов. Этот список загружается при инициализации соединения и осуществляется функцией loadChatList().

```
КонецПроцедуры

⊞ //Процедура открывает форму движений документа //...
⊞ Процедура ОсновныеДействияФормыНастройка(Кнопка) ...

⊞ // Процедура вызывается принажатии кнопки в подменю "Советы" командной панели //...
⊞ Процедура ДействияФормыОткрытьСоветы(Кнопка) ...

⊞ // Процедура открывает форму свойств документа //...
⊞ Процедура ДействияФормыДействиеОткрытьСвойства(Кнопка) ...

⊞ // Процедура открывает форму категорий документа //...
⊞ Процедура ДействияФормыДействиеОткрытьКатегории(Кнопка) ...
```

Рисунок 3.4.3 – Функция loadChatList() из chat.service.ts

Данная функция делает запрос на сервер по URL-адресу, который в своем пути содержит «/api_chat/chats». Запрос призывает к выполнению функцию контроллера с названием chats().

```
Перем   КартинкаВыполнено;
Перем   КартинкаИсправлено;
Перем   КартинкаОшибок;
Перем   КартинкаУдаления;
```

Рисунок 3.4.4 – Функция chats() из ChatController.java

Функция chats() реализована при помощи сервисного класса с названием ChatService внутри которого содержится функция getAllChats().

```
// параметры блокировки
Блокировка = Новый БлокировкаУстановкиСоединений;

Блокировка.Начало            = НачалоБлокировки;
Блокировка.Конец             = ОкончаниеБлокировки;
Блокировка.Сообщение         = Сообщение;
Блокировка.Установлена       = УстановитьБлокировкуСоединений;
Блокировка.КодРазрешения     = КодРазрешения;
```

Рисунок 3.4.5 – Функция getAllChats() из ChatService.java

Эта функция возвращает список всех чатов для пользователя, который авторизован в системе. Для этого функция при помощи ChatDao обращается в базу данных с select-запросом, который в ответ показывает всех пользователей, данные о которых хранятся в базе данных, кроме самого владельца аккаунтом.

```
    // установка блокировки соединений
    УстановитьБлокировкуУстановкиСоединений(Блокировка)
КонецПроцедуры // УстановитьБлокировку()
```

Рисунок 3.4.6 – Функция getAllChatsFor() из ChatDao.java

Если пользователь нажимает на название любого контакта из списка, то перед ним открывается диалоговое окно с этим пользователем. Это срабатывает благодаря тому, что на весь список пользователей навешено событие click, обозначающее нажатие пользователя на этот элемент. При клике срабатывает функция openChatWith(), в параметры которой передается идентификатор того пользователя, с которым должно быть открыто диалоговое окно.

```
Процедура ПолеВводаОрганизацияПриИзменении(Элемент) ...

// Процедура вызова структуры подчиненности документа //...
Процедура ДействияФормыСтруктураПодчиненностиДокумента(Кнопка) ...

//Процедура открывает форму движений документа //...
Процедура ОсновныеДействияФормыНастройка(Кнопка) ...

// Процедура вызывается при нажатии кнопки "Советы" командной панели формы. //...
Процедура ДействияФормыОткрытьСоветы(Кнопка) ...

Процедура ДокументСписокПриАктивизацииСтроки(Элемент) ...

Процедура ДействияФормыПереключитьАктивность(Кнопка) ...

// Процедура открывает форму свойств документа //...
Процедура ДействияФормыДействиеОткрытьСвойства(Кнопка) ...

// Процедура открывает форму категорий документа //...
Процедура ДействияФормыДействиеОткрытьКатегории(Кнопка) ...
```

Рисунок 3.4.7 - Функция openChatWith() изchat.component.ts

Функция openChatWith() вызывает сервисную функцию чата с названием loadMessagesFor(), которая загружает все предыдущие сообщения пользователей из базы данных.

```
КонецПроцедуры

⊞ //Процедура открывает форму движений документа //...
⊞ Процедура ОсновныеДействияФормыНастройка(Кнопка) ...

⊞ // Процедура вызывается принажатии кнопки в подменю "Советы" командной панели //...
⊞ Процедура ДействияФормыОткрытьСоветы(Кнопка) ...

⊞ // Процедура открывает форму свойств документа //...
⊞ Процедура ДействияФормыДействиеОткрытьСвойства(Кнопка) ...

⊞ // Процедура открывает форму категорий документа //...
⊞ Процедура ДействияФормыДействиеОткрытьКатегории(Кнопка) ...
```

Рисунок 3.4.8 - Функция loadMessagesFor() изchat.service.ts

Данная функция отправляет на сервер запрос, который побуждает к выполнению метод контроллера ChatController.javac названием messages().

```
    // установка блокировки соединений
    УстановитьБлокировкуУстановкиСоединений(Блокировка)
КонецПроцедуры // УстановитьБлокировку()
```

Рисунок 3.4.9 – Методmessages() изChatController.java

В свою очередь функция контроллера обращается к сервисным функциям чата со стороны сервера(ChatService) и при помощи интерфейса ChatDao, предназначенного для работы с базами данных, возвращает коллекцию всех сообщений, которые пользователи отправляли друг другу.

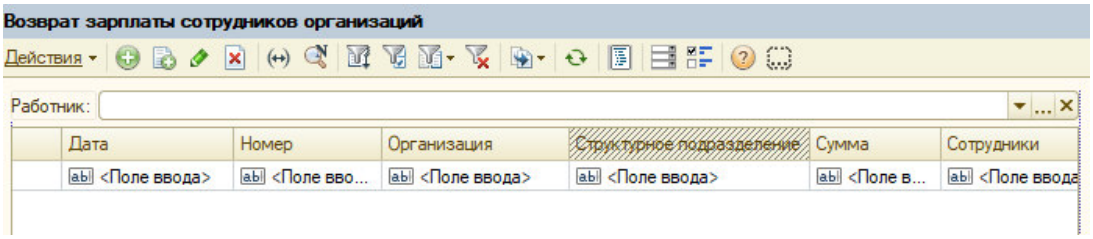

Рисунок 3.4.10 – МетодgetMessages() изChatService.java

В интерфейсе ChatDao срабатывает аннотация @Select, которая позволяет осуществить select-запрос в базу данных.

```
Кавычка = """";
ПутьКБазе = СтрЗаменить(СтрокаСоединения, Кавычка, Кавычка + Кавычка);

КодВозврата = Неопределено;
Кавычка = """";
Пользователь = ПользователиИнформационнойБазы.ТекущийПользователь().Имя;
Пароль = "";
ИмяФайлаСообщений = КаталогВыгрузки + "\СообщенияВыгрузкиКонфигурацииВФайлы.txt";
```

Рисунок 3.4.11 – Метод getMessages() из ChatDao.java

Далее расположено диалоговое окно, в котором выводятся все сообщения между пользователями. Ниже диалогового окна располагается строка ввода сообщения, в которой пользователь набирает сообщение, чтобы затем отправить тому человеку, с которым у него открыт чат.

Когда пользователь напечатал сообщение для отправки и нажимает на кнопку отправки, либо клавишу Enter на клавиатуре своего компьютера, срабатывает функция sendMessage(), которое отвечает за отправку сообщения в базу данных и дальнейшее сохранение сообщения.

```
КонецПроцедуры

//Процедура открывает форму движений документа //...
Процедура ОсновныеДействияФормыНастройка(Кнопка) ...

// Процедура вызывается принажатии кнопки в подменю "Советы" командной панели //...
Процедура ДействияФормыОткрытьСоветы(Кнопка) ...

// Процедура открывает форму свойств документа //...
Процедура ДействияФормыДействиеОткрытьСвойства(Кнопка) ...

// Процедура открывает форму категорий документа //...
Процедура ДействияФормыДействиеОткрытьКатегории(Кнопка) ...
```

Рисунок 3.4.12 – Метод sendMessage() изchat.component.ts

Этот метод вызывает сервисную функцию sendMessage() и создает для передачи в параметры новый объект типа Message, который в свою очередь содержит в себе всю информацию о новом сообщении пользователя.

Сама сервисная функция sendMessage() при помощи объекта stompClient отправляет на сервер сообщение в формате json.

```
Процедура ПриОткрытии()

    // Проверка однофирменности
    РаботаСДиалогами.УстановитьОтборПоОрганизации(ЭтаФорма, УчетПоВсемОрганизациям, ОсновнаяОрганизация, "ДокументСписок");

    // Видимость колонки "СтруктурноеПодразделение"
    РаботаСДиалогами.ВидимостьКолонкиСтруктурногоПодразделения(ЭлементыФормы.ДокументСписок);

    Если НЕ УчетПоВсемОрганизациям Тогда

        ЭлементыФормы.ФлагОрганизация.Доступность      = Ложь;
        ЭлементыФормы.ПолеВводаОрганизация.Доступность = Ложь;

    КонецЕсли;
```

Рисунок 3.4.13 – Метод sendMessage() изchat.service.ts

Согласно пути, определенному в параметрах функции, предметом на сервере, который будет продолжать работу по отправке сообщения, является WebSocketController, а именно метод send_message().

```
        // установка блокировки соединений
        УстановитьБлокировкуУстановкиСоединений(Блокировка)
КонецПроцедуры // УстановитьБлокировку()
```

Рисунок 3.4.14 – Методsend_message()изWebSocketController.java

Изначально этот метод сохраняет сообщение при помощи метода saveMessage(). Метод saveMessage() реализован внутри класса ChatService и является реализацией метода из интерфейса ChatDao, который предназначен для работы с базой данных приложения. Аннотация @Insert перед объявлением

метода saveMessage() внутри интерфейса ChatDao позволяет отправить в базу данных insert-запрос, который сохранит сообщение в базе данных.

```
//Процедура открывает форму движений документа //...
Процедура ОсновныеДействияФормыНастройка(Кнопка) ...

// Процедура вызывается принажатии кнопки в подменю "Советы" командной панели //...
Процедура ДействияФормыОткрытьСоветы(Кнопка) ...

// Процедура открывает форму свойств документа //...
Процедура ДействияФормыДействиеОткрытьСвойства(Кнопка) ...

// Процедура открывает форму категорий документа //...
Процедура ДействияФормыДействиеОткрытьКатегории(Кнопка) ...
```

Рисунок 3.4.15 – МетодsaveMessage () из ChatService.java

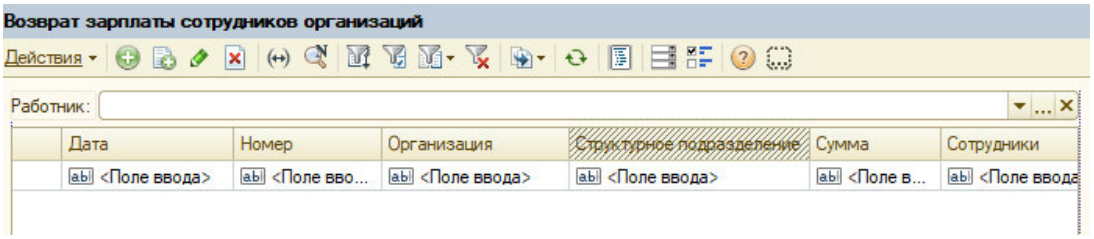

Рисунок 3.4.16 – МетодsaveMessage () из ChatDao.java

Так как чат, созданный в приложении, реализован при помощи технологии WebSocket, то далее происходит отправка сообщений только тем пользователям, которые прослушивают данные каналы связи. Таким образом, это будут два пользователя: отправитель сообщения и получатель сообщения. В дальнейшем на клиентской стороне приложения эти данные используются для корректного отображения сообщений в ленте чата пользователей.

3.5 Компонент planner

Компонент planner представляет собой сервис приложения, который предназначен для планирования списка дел. Такой сервис позволяет полноценно подходить к планированию рабочего дня и определять опорные моменты для работы. Составление планов повышает эффективность и продуктивность тех служащих, которые пользуются данным функционалом.

Структура этого компонента выглядит следующим образом:

```
Перем мКэшСоответствиеСчетов;

Процедура ПолеВводаОрганизацияПриИзменении(Элемент)

    Отбор.Организация.Использование = ЗначениеЗаполнено(Элемент.Значение);

КонецПроцедуры
```

Рисунок 3.5.1 – Метод saveMessage () из ChatDao.java

Для того, чтобы добраться до планировщика необходимо перейти в пункт «Планировщик» навигационной панели в верхней части экрана. После перехода к этому пункту перед пользователем откроется страница, на которой присутствует форма для заполнения задач и календарь, предназначенный для перехода по разным датам.

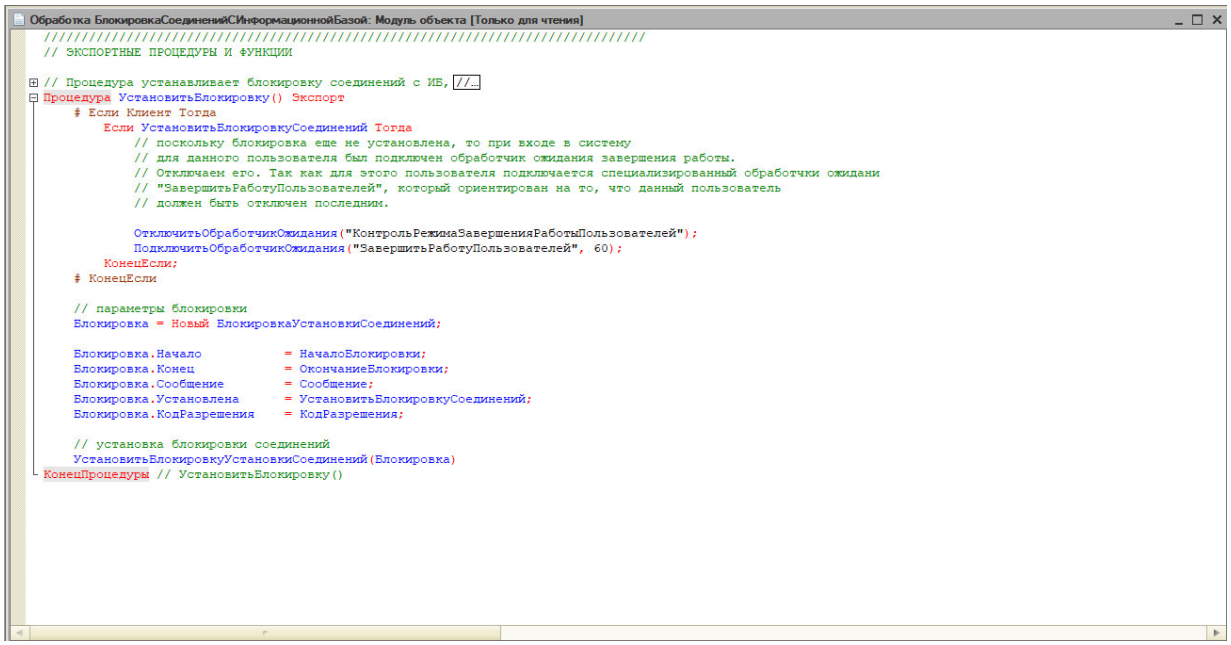

Рисунок 3.5.2 – Планировщик задач

Весь функционал страницы взаимодействует с базой данных и серверной частью благодаря сервису PlannerService, написанном на языке TypeScript.

Изначально, сразу после загрузки страницы слева, в списке задач, отображаются те задачи, которые запланированы на текущую дату, либо не отображается ничего, если на текущую дату отсутствуют запланированные задачи.

Таким образом, для того чтобы загрузка задач текущей даты осуществлялась сразу во время загрузки страницы, в файле planner.component.ts

существует функция ngOnInit(), которая выполняет свои действия вместе с загрузкой страницы.

```
Процедура ПриОткрытии()

    // Проверка однофирменности
    РаботаСДиалогами.УстановитьОтборПоОрганизации(ЭтаФорма, УчетПоВсемОрганизациям, ОсновнаяОрганизация, "ДокументСписок");

    // Видимость колонки "СтруктурноеПодразделение"
    РаботаСДиалогами.ВидимостьКолонкиСтруктурногоПодразделения(ЭлементыФормы.ДокументСписок);

    Если НЕ УчетПоВсемОрганизациям Тогда

        ЭлементыФормы.ФлагОрганизация.Доступность      = Ложь;
        ЭлементыФормы.ПолеВводаОрганизация.Доступность = Ложь;

    КонецЕсли;
```

Рисунок 3.5.3 – ФункцияngOnInit() дляPlannerComponent

Эта функция загружает все статусы задач, которые показывают степень выполненности задачи, а также с помощью функции getTasks() получает все задачи за текущую дату.

Название статусов задач загружаются при помощи функции loadStatus(), которая вызывается из сервисных функций планировщика. Эта выполняет запрос на выполнение функции контроллера, которая находится по URL-адресу «/api_planner/allStatus». Функция контроллера обращается к перечислению с названием TaskStatus и, при помощи метода getAllStatus(), в виде коллекции возвращает все значения, соответствующие каждому из элементов перечисления.

```
КонецПроцедуры

⊞ //Процедура открывает форму движений документа //...
⊞ Процедура ОсновныеДействияФормыНастройка(Кнопка) ...

⊞ // Процедура вызывается принажатии кнопки в подменю "Советы" командной панели //...
⊞ Процедура ДействияФормыОткрытьСоветы(Кнопка) ...

⊞ // Процедура открывает форму свойств документа //...
⊞ Процедура ДействияФормыДействиеОткрытьСвойства(Кнопка) ...

⊞ // Процедура открывает форму категорий документа //...
⊞ Процедура ДействияФормыДействиеОткрытьКатегории(Кнопка) ...
```

Рисунок 3.5.4 – ФункцияloadStatus()изplanner.service.ts

```
Блокировка.Начало          = НачалоБлокировки;
Блокировка.Конец           = ОкончаниеБлокировки;
Блокировка.Сообщение       = Сообщение;
```

Рисунок 3.5.5 – МетодallStatus() изPlannerController.java

```
    // установка блокировки соединений
    УстановитьБлокировкуУстановкиСоединений(Блокировка)
КонецПроцедуры // УстановитьБлокировку()
```

Рисунок 3.5.6 – МетодgetAllStatus () изTaskStatus.java

Сама функция getTasks() предназначена для отображения списка задач за переданную ей дату. Таким образом, когда загружается приложение, оно записывает текущую дату в переменную selectedDate при помощи преобразования данных полученных от функции Date() в формате NgbDate(). А затем эта дата передается в функцию getTasks(), которая выводит все задачи за текущий день.

```
Перем мКэшСоответствиеСчетов;

Процедура ПолеВводаОрганизацияПриИзменении(Элемент)

    Отбор.Организация.Использование = ЗначениеЗаполнено(Элемент.Значение);

КонецПроцедуры
```

Рисунок 3.5.7 – Функция getTasks()

Также эта функция применяется, если пользователю нужно получить нужно получить задачи за какую-либо дату отличную от сегодняшней. Календарь, созданный при помощи библиотеки ng-bootstrap отслеживает все изменения в пределах своего элемента. Следовательно, когда пользователь нажимает на другую дату в календаре, элемент фиксирует изменения и вызывает функцию getTasks() для отработки. В свою очередь эта функция выводит нам задачи за новую дату с их статусами, а также при ее отработке меняется значение переменной selectedDate, название которой дословно переводится как «выбранная дата». Отсюда следует, что для приложения по умолчанию выбранной датой является текущая дата, и именно с ней приложение и начинает свою работу.

```
#Область ОбработчикиСобытийФормы

&НаСервере
Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)

    Если ЭСФКлиентСерверПереопределяемый.ИспользуютсяСтруктурныеПодразделения() Тогда
        Элементы.СтруктурноеПодразделение.Видимость = Истина;
    КонецЕсли;

    Если Параметры.Свойство("ТолькоПросмотр") И Параметры.ТолькоПросмотр = Истина Тогда
        РежимТолькоПросмотр = Истина;
        Элементы.Причина.Видимость = Истина;
    КонецЕсли;
```

Рисунок 3.5.8 – Элемент, отслеживающий изменение даты

Функция getTasks() внутри себя вызывает функцию сервиса планировщика, которая называется loadTasks(). Эта функция возвращает нам с сервера данные, которые найдет по адресу URL, который оканчивается на «/api_planner/getTasks».

```
ВедетсяУчетПоТоварамОрганизацийБУ = НомераГТДСервер.ВедетсяУчетПоТоварамОрганизаций(Объект.Дата);
ВыпискаБумажногоЭСФ = (ЗначениеЗаполнено(Объект.ДатаВыпискиНаБумажномНосителе) Или ЗначениеЗаполнено(Объект.ПричинаВыпискиНаБумажномНосителе));
БезДоговора = НЕ (ЗначениеЗаполнено(Объект.ДоговорПоставкиНомер) ИЛИ ЗначениеЗаполнено(Объект.ДоговорПоставкиДата));

УправлениеФормой();

ПоставщикиПолучателиУстановитьТекущуюСтрокуИРежимРедактирования();

Если Объект.Ссылка.Пустая() Тогда
    УстановитьОграниченияТиповДляПолейВыбораПоставщиковИПолучателей();
    ЗаполнитьДопРеквизитыТаблицТоваровПоУчастникамСД();
    ЗаполнитьТаблицуОшибок(Объект.Ошибки);
КонецЕсли;

ЭСФСерверПереопределяемый.ЭСФПриСозданииНаСервере(ЭтаФорма);

ОбновитьПредставлениеКодаКлассификатора("УсловияПоставки");
ОбновитьПредставлениеКодаКлассификатора("СпособыОтправления");
```

Рисунок 3.5.9 – Функция loadTasks()

Далее начинается работа на сервере. Со стороны сервера для работы был выбран фреймворк SpringMVC. Запрос, исходящий от клиента, находит контроллер планировщика и выполняет метод контроллера, аннотации которого соответствуют отправленному в запросе URL.

```
&НаСервере
Процедура ПриЧтенииНаСервере(ТекущийОбъект)

    УстановитьОграниченияТиповДляПолейВыбораПоставщиковИПолучателей();
    ЗаполнитьДопРеквизитыТаблицТоваровПоУчастникамСД();
    ЗаполнитьТаблицуОшибок(ТекущийОбъект.Ошибки);

    УправлениеФормой();

    ЭСФСерверПереопределяемый.ЭСФПриЧтенииНаСервере(ТекущийОбъект, ЭтаФорма);

КонецПроцедуры
```

Рисунок 3.5.10–МетодgetTasks() изPlannerController.java

В свою очередь этот метод, возвращает нам данные, полученные при отработке метода в интерфейсе plannerDao. Создание и использование подобного интерфейса в данном случае предназначено для использования технологии фреймворка MyBatis, который осуществляет доступ к базе данных при помощи SQL-запросов. Необходимые SQL-запросы записываются в аннотацию по типу необходимо запроса. Для функции getTasks() это будет аннотация @Select и, соответственно, select-запрос. С помощью этого запроса мы получаем все задачи за определенную переданную дату для конкретного пользователя, который направил запрос к базе данных.

```
&НаКлиенте
Процедура ИдентификаторПриИзменении(Элемент)

    Если ЗначениеЗаполнено(Объект.Идентификатор) Тогда
        ТекстСообщения = "";
        Элемент.ОтметкаНезаполненного = НЕ ЭСФКлиентСервер.ИдентификаторКорректен(Объект.Идентификатор, ТекстСообщения);
        Элемент.Подсказка             = ТекстСообщения;
    КонецЕсли;

КонецПроцедуры
```

Рисунок 3.5.11 – Описание запроса getTasks() из PlannerDao.java

Помимо этого, в планировщике можно добавлять новые задачи на указанную дату, удалять задачи и изменять статус, описывающий степень их готовности.

Добавление новой задачи в список задач происходит при нажатии кнопки «Добавить» рядом с полем для ввода текста задачи. Таким образом, пользователь вводит название задачи в окно ввода, нажимает на кнопку «Добавить» и задача отображается в списке задач, расположенной под формой ввода задачи.

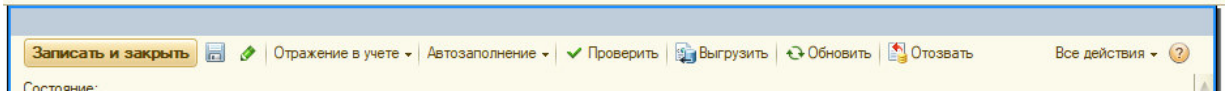

Рисунок 3.5.12 – Форма для добавления новой задачи

Для добавления новой задачи используется функция addTask(). Для отправки задачи на сервер создается специальный объект Task, в который записывается вся информация о задаче: текст задачи, на какую дату и кем записывается задача. Статус задачи по умолчанию определен как «Новая» при создании задачи. Функция addTask() записывает в объект все необходимые значения и затем вызывает сервис планировщика, который осуществляет дальнейшую отправку на сервер объекта Task.

В сервисе планировщика для сохранения задания выступает функция saveTask(). Данная функция возвращает значения, найденные по адресу с URL, который оканчивается на «/api_planner/saveTask».

```
&НаКлиенте
Процедура ОшибкиПередНачаломДобавления(Элемент, Отказ, Копирование, Родитель, Группа)
    Отказ = Истина;
КонецПроцедуры
```

Рисунок 3.5.13 – Функция addTask() из planner.component.ts

```
&НаКлиенте
Процедура ТоварыПередНачаломДобавления(Элемент, Отказ, Копирование, Родитель, Группа)
    ЗапретитьИзменениеТаблицы(Отказ);
КонецПроцедуры
```

Рисунок 3.5.14 – Функция saveTask() из planner.service.ts

Как и при выполнении предыдущего запроса, на получение всех задач за текущую дату, запрос, исходящий от клиента, находит контроллер планировщика и выполняет метод контроллера, аннотации которого соответствуют отправленному в запросе URL.

```
&НаКлиенте
Процедура ПерезаполнитьПоСчетуФактуре(Команда)
    ПерезаполнитьПоСчетуФактуреНаСервере();
КонецПроцедуры
```

Рисунок 3.5.15 – МетодsaveTask() изPlannerController.java

Этот метод обращается к PlannerDao, для выполнения операции по вставке новой задачи в таблицу базы данных (insert-запрос с аннотацией @Insert), а затем возвращает обновленный список задач на запрошенную дату при помощи команд, отрабатывающих при запросе задач на определенную дату. Таким образом, у пользователя, в окне браузера, выводится уже обновленный список задач, в который входит та задача, которая была добавлена только что.

```
&НаКлиенте
Процедура СоздатьСчетФактуруНаКлиенте()

    Если Объект.Ссылка.Пустая() ИЛИ ЭтаФорма.Модифицированность Тогда
        СоздатьСчетФактуруНаКлиентеПослеЗаписи = Новый ОписаниеОповещения("СоздатьСчетФактуруНаКлиентеПослеЗаписи", ЭтаФорма);
        ЭСФКлиент.ВопросЗаписатьОбъектПередВыполнением(СоздатьСчетФактуруНаКлиентеПослеЗаписи);
    Иначе
        СоздатьСчетФактуруНаКлиентеПослеЗаписи(КодВозвратаДиалога.ОК, Неопределено);
    КонецЕсли;

КонецПроцедуры
```

Рисунок 3.5.16 – Описание запроса saveTask() из PlannerDao.java

Удаление из списка задач производится по нажатии на иконку мусорной корзины. Когда пользователь нажимает на эту кнопку для данной задачи из списка вызывается функция deleteItem(). Для того, чтобы удалить именно выбранную задачу, в параметры функции передается конкретный объект, подлежащий удалению.

```
&НаКлиенте
Процедура ВыбратьСчетФактуруНаКлиенте()

    Если Объект.Направление = ПредопределенноеЗначение("Перечисление.НаправленияЭСФ.ПустаяСсылка") Тогда
        ПоказатьПредупреждение(, НСтр("ru = 'Не указано направление документа.'"));
        Возврат;
    КонецЕсли;
```

Рисунок 3.5.17 - Метод deleteItem() изplanner.component.ts

Сам функционал метода заключается в том, что вызывается все тот же сервис планировщика, который имеет метод deleteTask() для общения с сервером. В запросе post передается идентификатор задания, которое нужно удалить, а сам контроллер ищется по адресу URL с содержанием «/api_planner/deleteTask». Далее URL содержит в себе данные для удаления задания с конкретным идентификатором. Функция encodeURIComponent() служит для экранирования особых символов в строке, чтобы передаваемое значение не меняло своей структуры.

```
&НаКлиенте
Процедура СоздатьНовогоКонтрагентаЗавершение(РезультатВопроса, ДополнительныеПараметры) Экспорт

    Если РезультатВопроса = КодВозвратаДиалога.Да Тогда
        ПоставщикСоздатьНаКлиенте();
    КонецЕсли;

    ЗаполнитьРеквизитДоговорПоставкиПоДаннымЭСФ();

КонецПроцедуры
```

Рисунок 3.5.18 - Метод deleteItem() изplanner.service.ts

Далее метод контроллера PlannerContriller.java обращается к PlannerDao, чтобы была произведена функция удаления задания из таблицы базы данных. Для этого к объявлению метода интерфейса приписывается аннотация @Delete которая в свою очередь соответствует delete-запросу в базу данных. После выполнения этого запроса строка, которая хранила в себе данные об этой задаче удаляется, а в файле planner.component.ts метод deleteItem() вызывает метод getTasks() для обновления списка задач.

```
&НаКлиенте
Процедура ПоставщикЗаполнитьРеквизиты(Команда)

    ТекущиеДанные = Элементы.Поставщики.ТекущиеДанные;

    Если ТекущиеДанные <> Неопределено Тогда

        Если ЗначениеЗаполнено(ТекущиеДанные.Поставщик) Тогда

            ТекстВопроса = НСтр(
            "ru = 'Данные поставщика будут перезаполнены по данным ЭСФ.
            |Продолжить?'");
```

Рисунок 3.5.19 – МетоддeleteTask() изPlannerController.java

```
&НаКлиенте
Процедура ПоставщикСоздать(Команда)

    ПоставщикСоздатьНаКлиенте();

КонецПроцедуры
```

Рисунок 3.5.20 – МетодdeleteTask() изPlanerDao

Помимо основных функций отображения, добавления и удаления, для каждой задачи также можно изменить значение статуса, который определяет степень выполненности задачи. Выбор конкретного статуса задачи осуществляется при помощи элемента <select>, который позволяет показать список статусов при нажатии по нему. В том случае, если статус задачи был изменен, то в planner.component.ts сработает метод changeStatus(). Этот метод для осуществления своей функции принимает параметры в виде идентификатора задачи и название нового статуса, который будет ей приписан.

```
&НаКлиенте
Процедура ДоговорЗаполнитьРеквизиты(Команда)

    ИмяТаблицыКонтрагентов = "";
    ИмяРеквизитаКонтрагента = "";
    Если ДанныеДляСозданияОбновленияДоговораЗаполнены(ИмяТаблицыКонтрагентов, ИмяРеквизитаКонтрагента) Тогда

        Если ЗначениеЗаполнено(Объект.ДоговорПоставки) Тогда
            ЭСФКлиентПереопределяемый.ОткрытьЗаполненнуюФормуСтарогоДоговора(ЭтаФорма, ИмяТаблицыКонтрагентов, ИмяРеквизитаКонтрагента);
        Иначе
            Сообщить(НСтр("ru = 'Невозможно выполнить действие, так как договор поставки не заполнен.'"));
        КонецЕсли;

    КонецЕсли;
```

Рисунок 3.5.21 – МетодchangeStatus() изplanner.component.ts

Данный метод вызывает внутри себя к реализации метод сервиса планировщика, а именно updateTaskStatus(). Этот метод отправляет на сервер запрос на изменение статуса задачи. Метод контроллера, который выполняется в данном случае располагается по URL-адресу «/api_planner/changeStatus».

```
Функция ДанныеДляСозданияОбновленияДоговораЗаполнены(ИмяТаблицыКонтрагентов, ИмяРеквизитаКонтрагента)

    ДанныеЗаполнены = Истина;

    Если Объект.Направление = ПредопределенноеЗначение("Перечисление.НаправленияЭСФ.Входящий") Тогда
        ИмяТаблицыКонтрагентов = "Поставщики";
        ИмяРеквизитаКонтрагента = "Поставщик";
    Иначе
        ИмяТаблицыКонтрагентов = "Получатели";
        ИмяРеквизитаКонтрагента = "Получатель";
    КонецЕсли;
```

Рисунок 3.5.22 – Метод updateTaskStatus() изplanner.service.ts

```
&НаКлиенте
Процедура ТоварыСоздатьТовары(Команда)

    ТекущиеДанные = Элементы.Товары.ТекущиеДанные;

    Если ТекущиеДанные <> Неопределено Тогда
        ЭСФКлиентПереопределяемый.ОткрытьЗаполненнуюФормуНовогоТовара(ЭтаФорма, ТекущиеДанные);
    КонецЕсли;

КонецПроцедуры
```

Рисунок 3.5.23 – Метод changeStatus() изPlannerController.java

При отработке этой функции метод контроллера changeStatus() обращается к объявлению функции updateStatus(), которая помечена аннотацией @Update выполняет в базу данных update-запрос. Выполнение этого запроса приводит к изменению статуса задачи не только в интерфейсе пользователя, но и в базе данных приложения. Это означает, что, когда пользователь позже снова вернется к просмотру задач за ту дату, в которой он изменил статус у задачи, он уже увидит обновленные значения.

```
&НаКлиенте
Процедура ЗапретитьИзменениеТаблицы(Отказ)

    Если НЕ Элементы.ФормаРедактироватьНедоступныеРеквизиты.Пометка Тогда
        Отказ = Истина;
    КонецЕсли;

КонецПроцедуры
```

Рисунок 3.5.24 – Метод updateStatus() изPlannerDao.java

В ходе разработки было изучено большое множество технологий, которые позволяют разрабатывать современные и функциональные приложения для большого количества пользователей. Эти технологии относятся как к frontend-разработке, так и к backend-разработке и проектированию базы данных приложения.

Главной задачей, которая и была выполнена в ходе разработки, являлось создание приложения, которое может полноценно использоваться в реальных условиях труда и выполнять все положенные ему функции

По итогам разработки было получено функциональное приложение, которое позволяет пользователям оставаться на связи и планировать свой рабочий ритм в течение многих дней и недель. Все данные пользователей доступны лишь им самим и процесс использования приложения запускается только после регистрации каждого сотрудника.

4 Экономическая часть

4.1 Описание работы и обоснование необходимости

Темой дипломного проекта является «Разработка информационной систем для ТОО «BEST – Group A»»

Целью данного проекта является – обеспечение новой информации для компании, дополняющей печатные издания, служащее для индивидуального и индивидуализированного обучения и позволяющее в ограниченной мере тестировать полученные знания и умения обучаемого. Данная конфигурация облегчает работу пользователю в компании, создание новых отчетов и таблиц, ранее не представленных в конфигураций 1С. Так же добавлена опраделенная подсистема для компании ТОО «BEST – Group A»».

В главном меню «Бизнес процессы», для отчележывалия работы сотрудников в орагинзаций.

Прием на работу, ознакомнение в обязательными требованиями работодателя, обучение сотрудника по политике компании и его развития в ней.

В разработке программного обеспечения будет участвовать группа специалистов, которая включает в себя: технический руководитель, программист-разработчик. В обязанности технического руководителя входит соблюдение и разработка рабочих графиков, их контроль и оптимизация. В обязанности программиста-разработчика входит разработка технического обоснования, разработка программного обеспечения, его тестирование и сопровождение. Следовательно, основная работа ложится на плечи программиста-разработчика, в то время как технический руководитель занимается организационными вопросами. Технико-экономическое обоснование содержит следующие пункты:

- трудоемкость разработки программного продукта;
- материальные затраты;
- затраты на оплату труда;
- социальный налог;
- амортизация основных фондов.

4.2 Расчет трудоемкости разработки программного продукта

Таблица 4.1– Распределение работ по этапам и видам и оценка их трудоемкости

Этапразработки	Видработы	Трудоемкость разработки, чел.×ч.
1	Составление задачи	10
2	Создание алгоритмов и блок схемы	10
3	Создание клиентской части программного проекта	50
4	Созданиесерверной части проекта	100
5	Тестирование	20
6	Документация	35
ИТОГО трудоемкость выполнения дипломной работы		225

Чтобы определить сколько это будет в днях, нам нужно разделить итоговое число на 8 (рабочий день): 225 / 8 = 28

4.3 Расчет затрат на разработку программного продукта

Для вычисления затрат на реализацию ПП необходимо найти через составления сметы, которая учитывает следующие статьи:

- материальные затраты;
- затраты на оплату труда;
- социальный налог;
- амортизация основных фондов.

Таблица 4.2– Затраты на материальные ресурсы

Наименование материального ресурса	Единица измерения	Количество	Цена за единицу, тг	Сумма, тг
НоутбукHPProBookG	шт.	1	390000	390000
Принтер Scx-3400	шт.	1	53000	53000
Картридж	шт.	1	1500	1500
Бумага А4	шт.	1	2000	2000
ИТОГО				446500,00

Общая сумма затрат на материальные ресурсы ($З_М$) определяется по формуле:

$$З_м = \sum P_i * Ц_i, \quad (4.1)$$

где P_i– расход i-го вида материального ресурса, натуральные единицы;

$Ц_i$– цена за единицу i-го вида материального ресурса, тг;

i – вид материального ресурса;

n – количество видов материальных ресурсов.

$$З_{ноутбук} = 1 \times 390000 = 390000 \text{ тг}$$
$$З_{ПО} = 1 \times 53000 = 53000 \text{ тг}$$
$$З_{бумага} = 1 \times 2000 = 2000 \text{ тг}$$
$$З_{картридж} = 1 \times 1500 = 70000 \text{ тг}$$
$$З_{общие} = 390000 + 53000 + 2000 + 1500 = 446500{,}00 \text{тг}$$

Если для разработки ПП используется электрооборудование, то необходимо рассчитать затраты на электроэнергию по форме, приведенной в таблице 6.3.

Общая сумма затрат на электроэнергию ($З_Э$) рассчитывается по формуле:

$$З_Э = \sum_{i=1}^{n} M_i \times K_i \times T_i \times Ц, \quad (4.2)$$

где M_i– паспортная мощность i-го электрооборудования, кВт;

K_i– коэффициент использования мощности i-го электрооборудования (принимается K_i=0,9);

T_i– время работы i-го оборудования за весь период разработки ПП ч;

Ц– цена электроэнергии, тг/кВт×ч;

i – вид электрооборудования;

n – количество электрооборудования.

Затраты на электроэнергию находятся исходя из продолжительности периода разработки ПО, количества кВт/ч, затраченных на проектирование ПО и тарифа за 1 кВт/ч. Тариф по городу Алматы для юридических лиц в 2019 году составляет 18,32 тенге за 1 кВт/ч с учетом НДС (согласно данным представленным на официальном сайте ТОО «АлматыЭнергоСбыт»).

$$З_э=0,9\cdot0,9\cdot240\cdot18,32=3561,40 \text{ тг}$$
$$З_э=0,3\cdot0,7\cdot240\cdot18,32= 923.32 \text{тг}$$

Таблица 4.3– Затраты на электроэнергию

Наименование оборудования	Паспортная мощность, кВт	Коэффициент использования мощности	Время работы оборудования для разработки ПП, ч	Цена электро энергии, тг/кВт·ч	Сумма, тг
Ноутбук HPProBookG430-g1	0,9	0,9	240	18,32	3561,40
Освещение	0,3	0,7	240	18,32	923.32
ИТОГО					4484.73

4.4 Расчет расходов на оплату труда

В разработке программного продукта заняты 2 сотрудника: руководитель проекта и разработчик. Средняя заработная плата руководитель проекта в 2019 году составляет 225 000 тг, а разработчика 195 000 тг (для города Алматы).

Рабочие часы сотрудника за месяц определяются по формуле:

$$Ч_м = N_м \cdot Ч_{рд}, \qquad (4.2.2)$$

где $Ч_м$— рабочие часы сотрудника за месяц;
$N_м$—количество рабочих дней за месяц;
$Ч_{рд}$—количество рабочих часов в день.

$$Ч_м = 22 \times 8 = 176 \text{ ч.}$$

Часовая ставка работника может быть рассчитана по формуле:

$$ЧС_i = \frac{ЗП_i}{ФРВ_i}$$

Руководитель проекта:

$$ЧС_i = \frac{225000}{176} = 1\,278{,}40 \text{ тг}$$

Разработчик:

$$ЧС_i = \frac{195000}{176} = 1\,107{,}95 \text{ тг} \quad (4.3)$$

где $ЗП_i$– месячная заработная плата i-го работника, тг;
$ФРВ_i$– месячный фонд рабочего времени i-го работника, час.

Для определения трудоемкости разработки ПП используются данные из таблицы 4.1.

Трудоемкость разработки ПП плата руководитель проектаравна 125чел.×ч(составление задачи, разработка алгоритмов и блок схемы, реализация клиентской части проекта, тестирование,документация).

$$T_2 = 10+10+50+20+35 = 125 \text{ чел.×ч.}$$

Трудоемкость разработки ПП разработчика равна 130чел.×ч.(разработка блок-схемы алгоритма, разработка администраторской части проекта, тестирование программы).

$$T_2 = 10 + 100 + 20 = 130 \text{ чел.×ч.}$$

Общая сумма затрат на оплату труда ($З_{ТР}$) определяется по формуле:

$$З_{ТР} = \sum_{i=1}^{n} ЧС_i \times T_i, \quad (4.4)$$

где $ЧС_i$– часовая ставка i-го работника, тг;
T_i– трудоемкость разработки ПП, чел.×ч;

i – категория работника;
n – количество работников, занятых разработкой ПП.

Руководитель проекта:

$$З_{тр} = 1278,40 \times 125 = 159\ 800 тг.$$

Разработчик:

$$З_{тр} = 1107,95 \times 130 = 144\ 033,5 тг.$$

Общая сумма:

$$З_{тр} = 159\ 800 + 144\ 033,5 = 303\ 833,5 тг.$$

Таблица 4.4– Затраты на оплату труда

Квалификация	Трудоемкость разработки ПП, чел.×ч	Часовая ставка, тг/ч	Сумма, тг
Руководитель проекта	125	1278,40	159800
Разработчик	130	1107,95	144033,5
ИТОГО			303833,5

Дополнительная заработная плата:

$$З_{доп} = З_{тр} \times 10\%$$

$$З_{доп} = 303833,5 \times 0,1 = 30\ 383,35 тг.$$

Фонд заработной платы:

$$Ф_{зп} = З_{тр.о} + З_{доп}$$

$$Ф_{зп} = 303833,5 + 30\ 383,35 = 334\ 216,85 тг.$$

Расчет социального налога:

$$Н_с = (Ф_{зп} – ОПВ) \times 11\%,$$

где ОПВ – обязательные пенсионные взносы–10% от $Ф_{зп}$.

$$Н_с = (334\ 216,85 – (334\ 216,85 \times 0,1)) \times 0,11 = 33\ 087,46 тг.$$

Расчет амортизационных основных фондов

Общая сумма амортизационных отчислений определяется по формуле:

$$З_{АМ} = \sum_{i=1}^{n} \frac{Ф_i \times Н_{Ai} \times Т_{НИРi}}{100 \times Т_{Эфi}}, \quad (4.5)$$

где $Ф_i$– стоимость i-го ОФ, тг;

H_{Ai}– годовая норма амортизации i-го ОФ, %;

$T_{НИРi}$– время работы i-го ОФ за весь период разработки ПП, ч;

$T_{Эфi}$– эффективный фонд времени работы i-го ОФ за год, ч/год (по информации egov.kzза 2019 год принимается $T_{Эфi}$=1968);

i – вид ОФ;

n – количество ОФ.

Расчет годовой нормы амортизации ОФ:

$$H_{Ai} = \frac{100}{4} = 25$$

$$H_{Ai} = \frac{100}{T_{Ni}}, \quad (4.6)$$

где T_{Ni}– возможный срок использования i-го ОФ, год;

Для определения времени работы ПО для разработки ПП используются данные из таблицы 4.1.

Время работы ПО для разработки ПП составляет 170 часов (реализация клиентской части проекта, реализация администраторской части проекта, тестирование программы).

$$T_i = 100 + 50 + 20 = 170 \text{ ч.}$$

Оборудование:

$$З_{АМ} = \frac{443\ 000*25*225}{100*1968} = 12\ 661{,}96$$

Таблица 4.5– Амортизация основных фондов (ОФ)

Наименование оборудования и ПО	Стоимость оборудования и ПО, тг	Годовая норма амортизации, %	Эффективный фонд времени работы оборудования и ПО, ч/год	Время работы оборудования и ПО для разработки ПП, ч	Сумма, тг
Ноутбук HPProBook G430-g1	390000	25	1968	225	11 147,10
Принтер Scx-3400	53000	25	1968	225	1 514,86
ИТОГО					12 661,96

Таблица 4.6 – Смета затрат на разработку ПП

Статьи затрат	Сумма, тг	%
1. Затратына материальные ресурсы	446500,00	56
2. Затраты на электроэнергию	4484.73	1
3. Затраты на оплату труда	303833,5	38
4. Отчисления на социальные нужды	33 087,46	4
5. Амортизация основных фондов	12 661,96	2
ИТОГО	800 567,65	100

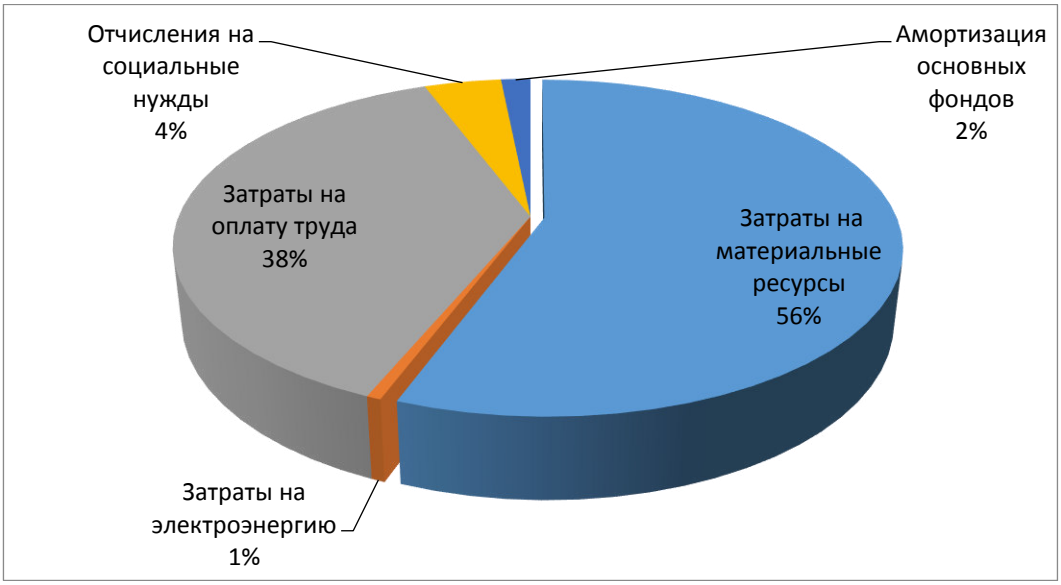

Рисунок 4.4.1 – Смета затрат на разработку программного продукта

4.5 Определение возможной (договорной) цены программного продукта

Договорная цена ($Ц_Д$) для прикладных ПП рассчитывается по формуле:

$$Ц_Д = З_{НИР} \cdot \left(1 + \frac{P}{100}\right), \qquad (4.7)$$

где $З_{НИР}$– затраты на разработку ПП (из таблицы 6.6), тг;
P– средний уровень рентабельности ПП – 25%.

$Ц_Д$ = 800567,65 * (1 + 0,25) =800567,65 + 200 141,91=1 000 709,56тг

Далее определяется цена реализации с учетом налога на добавленную стоимость (НДС), ставка НДС устанавливается законодательно Налоговым Кодексом РК. На 2019 год ставка НДС установлена в размере 12%.

Цена реализации с учетом НДС рассчитывается по формуле:

$$Ц_Р = Ц_Д \cdot (1 + НДС)$$

$$Ц_Р = 1\ 000\ 709{,}56 * (1 + 0{,}12) = 1\ 000\ 409{,}56 + 120\ 049{,}14 = 1\ 120\ 458{,}7 \text{ тг}$$

4.6 Вывод по технически-экономической части

Таким образом, цена реализации с учетом НДС равна 1 120 458,7 тг, себестоимость –800 567,65 тг и прибыль – 200 141,91 тг.

Основную часть затрат составляют затратына материальные ресурсы (56%).

5 Охрана труда и безопасность жизнедеятельности

В данном дипломном проекте разрабатывается информационная база данных для ТОО «BEST – Group A». В данной работе я рассчитываю естественное и искусственное освещение помещения.

Помещение имеет следующие размеры: длина 8 метров, ширина 5метров .Схема размещения рабочих мест приведена на рисунке 1. Имеем: 2 оконных проема, 3 лампочки и 7 рабочих мест. В данном помещении условия вентиляция и шумоизоляция рабочего места соответствует требованиям, так как там нет шумных приборов и в помещении поддерживается необходимая температура, циркуляция и влажность воздуха.

Помещение имеет естественное освещение. Оно подразделяется на боковое (проемы в стенах), верхнее (фонари в перекрытии) или комбинированное (верхнее плюс боковое). Искусственное освещение состоит состоит из люминесцентных ламп ПВЛМ- 2х40. Световой поток имеет заданное значение освещенности по техническим условиям 300 лк, разряд зрительной зоны работы определен пятой малой точности. А освещение не соответствует требованиям. Поэтому в данной части дипломного проекта принято решение о расчете искусственного освещения помещения.

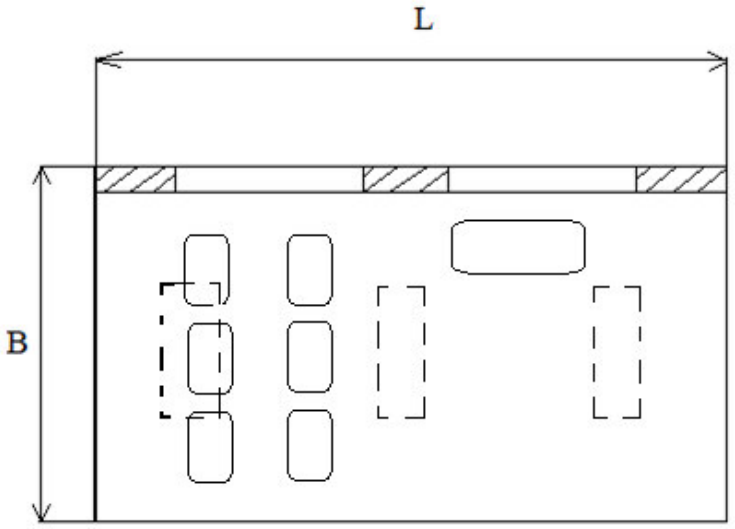

Рисунок 5.1 – Схема размещения групп

Расчет естественного освещения

Группа имеет длину 8 метров, ширину 5 метра и высоту 4 метра. А так же имеет два окна (ширина 2,5 метр и высота 2 метр) с площадью 10 м^2 и установлены 3 лампа ПВЛМ- 2х40 с световым потоком 1000 лм.

Разряд зрительных работ – III, в , высота окна 2 м, высота начала окна 1 м.

Расчет заключается в предварительном определении площади световых проемов при боковом освещении по следующей формуле:

$$S_0 = \frac{S_n * e_H * \eta_O * K_{ЗД} * K_З}{100 * \tau_O * r_1}, \text{ м}^2$$

Где S_n – площадб пола помещения, м^2:

$$S_n = B * L = 5 * 8 = 40\ м^2$$

e_H – нормированное значение КЕО:

$$e_H = e_{KEO} * m_N,$$

e_{KEO} – значение КЕО для III, б: $e_{KEO} = 1,2$

m_N – коэфициент светого климата, определяется по таблице 3.1 [1]: $m_N = 0.75$

$$e_H = 1,2 * 0,75 = 0,9$$

$К_З$ – коэффициент запаса по таблице 3.11 [1]: $К_З$=1,2;

t_0 – общий коэффициент светопропускания: τ0=τ1·τ2·τ3·τ4·τ5

τ_1 – коэффициент светопропускания материала, принимаемый по таблице 3.3 [1]. Так как в качестве светопропускающего материала используется стеклопакет, то: τ1=0,8

Таблица Е.1 -Значения коэффициента светопропускания стекла в окнах □$_1$

Вид стекла, установленного в окнах	Коэффициент□$_1$
Стеклолистовоеодинарное	0,9
Стеклолистовоедвойное	0,8
Стеклолистовоетройное	0,75
Стеклопакеты	0,8
Стеклолистовоеармированное	0,6

τ_2 – коэффициент, учитывающий потери света в переплетах светопроема. Определяется с помощью таблицы 3.4 из [1]: τ2=0,75 ,т.к. Переплеты для окон и фонарей промышленных зданий деревянные одинарные.

Таблица Е.2 - Значения коэффициента, учитывающего потери света воконныхпереплетах τ_2

Тип здания	Видоконногопереплета	Коэффициент τ_2
Промышленноездание	Стальныеодинарныеоткрывающиеся	0,75
	Стальныеодинарныеглухие	0,9
	Стальныедвойныеоткрывающиеся	0,6
	Стальныедвойныеглухие	0,8

τ_3 – коэффициент, учитывающий потери света в несущих конструкциях, при боковом освещении равен 1.

τ_4– коэффициент, учитывающий потери светы в солнцезащитных устройствах : τ4=1

Тогда $t_0 = 0{,}8 * 0{,}75 * 1 * 1 = 0{,}6$

η0 – световая характеристика окон;

Для определения световой характеристики, η0, необходимо расчитать отношение длины помещения к его глубине $\frac{L}{B/_2}$ и отношение глубины помещения к расчетной высоте $\frac{B/_2}{\square_{расч.}}$,

$$\frac{L}{B/_2} = \frac{8}{5/_2} = 3{,}2$$

Найдем $h_{расч}$:

$$h_{расч}=h_0+h_{н.о.}-h_{р.пов.}(1.3)$$

$$h_{расч}=2+1-1=2 \text{ м}$$

$$\frac{B/_2}{h_{расч}} = \frac{5/_2}{2} = 1{,}25$$

Учитывая найденные отношения примем световую характеристику ,η0, по таблице 3.2 из [1]: η_0=7,5

r_1 - коэффициент, учитывающий повышение КЕО при боковом освещении благодаря свету, отраженному от поверхности помещения и подстилающего слоя, примыкающего к зданию, определяют из [3]. Для этого следует учесть:

отношение ширины помещения к расчетной высоте $\frac{B/_2}{h_{расч}} = 1{,}25$

отношение глубины помещения к ширине $\frac{B/_2}{B} = 0.5$

отношение длины помещения к ширине $\frac{L}{B/2} = 3{,}2$

$$\frac{P_{пот} + P_{ст} + P_{пол}}{3} = \frac{50 + 50 + 10}{3} = 36.67\%$$

r1=1,1.

Кзд - коэффициент, учитывающий затемнение окон противостоящими зданиями по таблице 3.8 [1]:

$$\frac{P}{H_{зд}} = \frac{32}{15} = 2{,}13$$

Определим коэффициент, учитывающий затемнение окон противостоящими зданиям Кзд=1.1.

Зная значение всех параметров, рассчитаем площадь боковых проемов при естественном освещении по следующей формуле:

$$S_0 = \frac{40 \cdot 0{,}9 \cdot 7{,}5 \cdot 2{,}13 \cdot 1{,}1}{100 \cdot 0{,}6 \cdot 1{,}1} = 9{,}6$$

Вывод: Был приведен расчет естественного освещения. При проектировании естественного освещения помещений необходимо определить площадь световых проемов, обеспечивающих нормированное значение КЕО. В помещении для обеспечения нормированного значения КЕО, e_N=0.9, при разряде IIIв зрительных работ требуется площадь световых проемов равная 10 м^2, что соответствует нормативному показателю. Но в зимнее время и темное время суток наблюдается недостаток света, поэтому нужно также рассчитать искусственное освещение.

Проверка искусственного освещения

Точечным методом проверим соответствие данного количества и типа светильников нормируемой величине. В комнате установлено 3 светильника ПВЛМ – 2х40,

1.Определение расчетной высоты подвеса:

$$H_{расч}=H-(h_{р.п.}+h_{св.}),$$

$$h_{расч}=3-(0,8+0,2)=2 \text{ м}$$

2.Определение наивыгоднейшего расстояния между светильниками [3]

$$L_{A,B}=\lambda * h_{расч}, \text{ где } \lambda=0,6\div 2$$

В длину:

$$L_A=1,6*2,5=4 \text{ м}$$

В ширину:

$$L_B=0,8*2,5=2\text{м}$$

$$l_{A,B}= L_{A,B}(0.4\div 0.6) \Rightarrow l_A=2\text{м} l_B=1,5\text{м}$$

Намечаем контрольную точку А. Для нее определяем суммарную условную освещенность всех светильников следующим образом:

Находим проекцию расстояния на потолок от точки А до светильника d_i.

Далее определяем угол между потолком и прямой d_i. По этому углу находим условную освещенность. Проверим, выполняется ли условие:

$E_г \geq E_{норм}$.

$$E_Г = \Phi \cdot \mu \cdot \frac{\sum_{i=1}^{m} e_Г}{1000 * K_з}$$

Коэффициент запаса $K_З=1,4$

Коэффициент, учитывает действие равноудаленных светильников $\mu=1,2$

Световой поток F=1000 лм

В качестве светильника возьмем ПВЛМ-2х40 с люминесцентной лампой рассчитанный на две лампы мощностью 40 вт

$$E_{Г\text{i}} = \frac{I_{\alpha_i} \cos^3(\alpha_i)}{h^2}$$

Где $\alpha_i = \text{arctg}(\frac{d_i}{h})$

Тип св-ка	Сила света Iα, кд в направлении угла α										
	0	5	15	25	35	45	55	65	75	85	90
ПВЛМ-1х40	278	270	264	230	208	168	126	88	44	12	0

Расстояние от центральной точки до светильника d_1 найдем как:

$$d_1 = \sqrt{\mathbf{4^2 + 2^2}} = \mathbf{4.5}$$

Тогда

$$\alpha_1 = \text{arctg}\left(\frac{\mathbf{4.5}}{\mathbf{4}}\right) = \mathbf{1,125},$$

по этому значению I_α=277кд

$$e_1 = \left(\frac{\mathbf{277 * 0,97}}{\mathbf{3^2}}\right) = 16,7$$

Суммарная условная освещенность равна:

$$\sum \mathbf{e_r} = \mathbf{3 * 17 = 51}\ \text{лк}$$

Суммарная освещенность равна:

$$E_{АГ} = \frac{\mathbf{1,2 * 1000}}{\mathbf{100 * 1,4}} * \mathbf{30 = 150}\ \text{лк}$$

Освещенность на рабочем месте недостаточна, так как $E_K < E_{нор}$, где $E_{нор}$=300лк.Поэтому необходимо произвести реконструкцию системы освещения помещения.

Произведем реконструкцию, применяя метод коэффициента использования.

Индекс помещения i рассчитывается по формуле:

$$i = \frac{A \cdot B}{h \cdot (A + B)} = \frac{8*5}{2,8*(8+5)} = 1,09$$

где А- длина помещения (8 м)

B- ширина помещения (5 м)

h - расчетная высота (2,8 м)

Значение коэффициента использования (%) найдем по справочным данным, связывающий индекс помещения с коэффициентом отражения, η = 50%.

Необходимое количество светильников определяется по формуле:

$$N = \frac{E_н \cdot K_з \cdot S \cdot Z}{n \cdot Ф \cdot \eta}$$

$E_н$ = 300 лк - заданное номинальное освещение.

S = 40 м^2 – площадь помещения.

Z = 1.15- коэффициент неравномерности освещения.

n- количество ламп в светильнике.

$$N = \frac{300*1,15*40*1,4}{2*4100*0,5} = 4 \text{ шт.}$$

Нужно провести реконструкцию, поменяв количество и расположение светильники ПВЛМ.

Расстояния между соседними светильниками L определяется как $L = \lambda * h$. Рекомендуемое значение λ равное 0,6-2,0 для равномерного освещения.

La= 2,5*0,8=2 м

la= 2,5*1,6= 4 м

L_b= 2,5*0,8=2 м

l_b= 2,5*0,6 =1,5 м

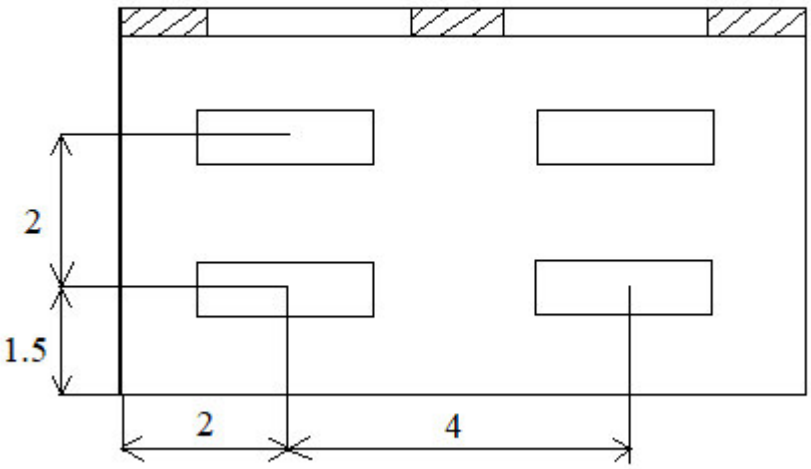

Рисунок 5.2 – Новая схема расположения светильников

Вывод: На основе расчетов была проведена реконструкция искусственного освещения, так как ранее установленные три лампы ПВЛМ – 2х40 с световым потоком не обеспечивали достаточно количества света в рабочем помещении с размерами 8х5х4. В результате реконструкции количество ламп было увеличено, что в результате стало выполнять норму освещенности.

Заключение

В ходе выполнения дипломной проекта был разработан программный продукт, который повышает продуктивность работы служащих и повышает уровень их организованности.

Для достижения поставленной цели использовались технологии:

– для разработки клиентского интерфейса: 1С: Предприятие 8.3;

– для соединения базы данных между собою использовался: СУБД MSSQL.

В дипломный проект входит введение, 5 глав и итоговое заключение.

Во введении раскрывается актуальность выбранной темы, ставится цель разработки и задачи, которые необходимо выполнить, определяется область применения разработки.

В первой главе производится анализ и выбор технологий для решения задачи.

Во второй главе описывается проектирование приложения.

Третья глава представляет собой описание разработки приложения, содержится пошаговое описание практического решения поставленной задачи.

В четвертой главе производится обоснование экономической целесообразности разрабатываемого проекта.

В пятой главе рассматриваются мероприятия по улучшению условий труда в рамках реализуемого проекта.

Заключение освещает результаты исследования и разработки:

– изучены современные средства разработки клиент-серверных веб-приложений и выполнена разработка согласно поставленной цели;

– разработана дальнейшая стратегия улучшения приложения до полноценного продукта.

Список литературы

1 1С:Предприятие 8.3. Практическое пособие разработчика /Авторы: Радченко Максим Григорьевич, Хрусталева Елена Юрьевна.

2 Решение специальных прикладных задач в «1С:Предприятии 8.2» /Авторы: Гончаров Д.И., Хрусталева Е.Ю

3 Профессиональная разработка в системе 1С:Предприятие 8.3″ / Авторы: Ажеронок В.А., Габец А.П., Гончаров Д.И., Козырев Д.В., Кухлевский Д.С., Островерх А.В., Радченко М.Г., Хрусталева Е.Ю.

4 Разработка управляемого интерфейса /Авторы: Ажеронок В.А., Островерх А. В., Радченко М. Г., Хрусталева Е. Ю.

5 Разработка сложных отчетов в «1С:Предприятии 8.3». Система компоновки данных /Авторы: Хрусталева Е.Ю.

6 Технологии интеграции 1С:Предприятия/ Авторы: Д. И. Гончаров, Е. Ю. Хрусталева

7 1С:Предприятие 8. Конвертация данных: обмен данными между прикладными решениями /Авторы книги: Бояркин В.Э. Филатов А.И.

8 Облачные технологии «1С:Предприятия»/ Хрусталева Е. Ю

9 Разработка сложных отчетов в «1С:Предприятии 8.3». Система компоновки данных /Авторы: Хрусталева Е.Ю.

10 1С:Предприятие 8. Конвертация данных: обмен данными между прикладными решениями /Авторы книги: Бояркин В.Э. Филатов А.И.

11 Решение специальных прикладных задач в «1С:Предприятии 8.2» /Авторы: Гончаров Д.И., Хрусталева Е.Ю

12 Разработка управляемого интерфейса /Авторы: Ажеронок В.А., Островерх А. В., Радченко М. Г., Хрусталева Е. Ю.

13 Решение специальных прикладных задач в «1С:Предприятии 8.2» /Авторы: Гончаров Д.И., Хрусталева Е.Ю

14 Облачные технологии «1С:Предприятия»/ Хрусталева Е. Ю

15 1С:Предприятие 8. Конвертация данных: обмен данными между прикладными решениями /Авторы книги: Бояркин В.Э. Филатов А.И.

16 1С:Предприятие 8.3. Практическое пособие разработчика /Авторы: Радченко Максим Григорьевич, Хрусталева Елена Юрьевна

17 Облачные технологии «1С:Предприятия»/ Хрусталева Е. Ю

18 Решение специальных прикладных задач в «1С:Предприятии 8.2» /Авторы: Гончаров Д.И., Хрусталева Е.Ю

19 1С:Предприятие 8. Конвертация данных: обмен данными между прикладными решениями /Авторы книги: Бояркин В.Э. Филатов А.И.

20 Решение специальных прикладных задач в «1С:Предприятии 8.2» /Авторы: Гончаров Д.И., Хрусталева Е.Ю

21 Разработка управляемого интерфейса /Авторы: Ажеронок В.А., Островерх А. В., Радченко М. Г., Хрусталева Е. Ю.

22 1С:Предприятие 8. Конвертация данных: обмен данными между прикладными решениями /Авторы книги: Бояркин В.Э. Филатов А.И.

Приложение А
(обязательное)
(Техническое задание)

Программа: 1С: Предприятие
Краткое описание: Создание конфигурации для ускорения работы в 1С
Постановка задачи:

1. Создание основных подсистем (Продажи, Закупки, Запрлата, Бизнес процессы)

2. В подсистеме «Продажи» реализовать:

– выписка банка;
– Заказ клиента;
– реализация ТМЗ;
– установка цен номенклатуры;
– печать счет на оплату;
– статусы сделок;

На основание заказа клиента создавать реализацию ТМЗ с прилагающими документами и обработками.

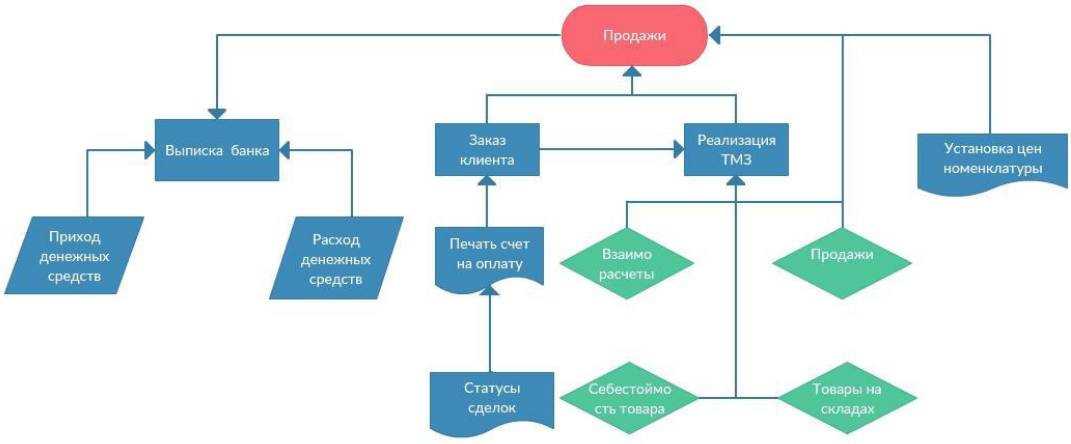

Рисунок 1 – продажи

3. Добавить «Закупки»:

– поступление ТМЗ;
– торговые документы

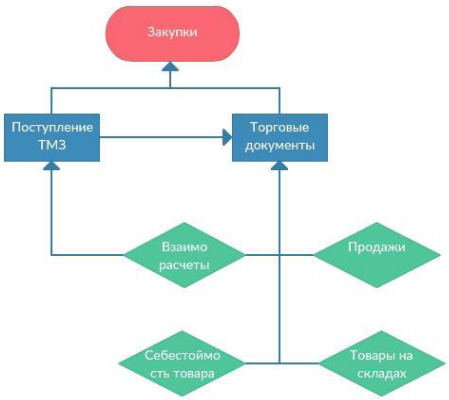

Рисунок 2 – закупки

4. В подсистеме «Зарплата» соединить справочники между собою:

– начисления оклада;
– невыход сотрудника;
– утверждение графика работ;
– график работ;
– начисление;
– выплата заработной платы;
– расчет премии;

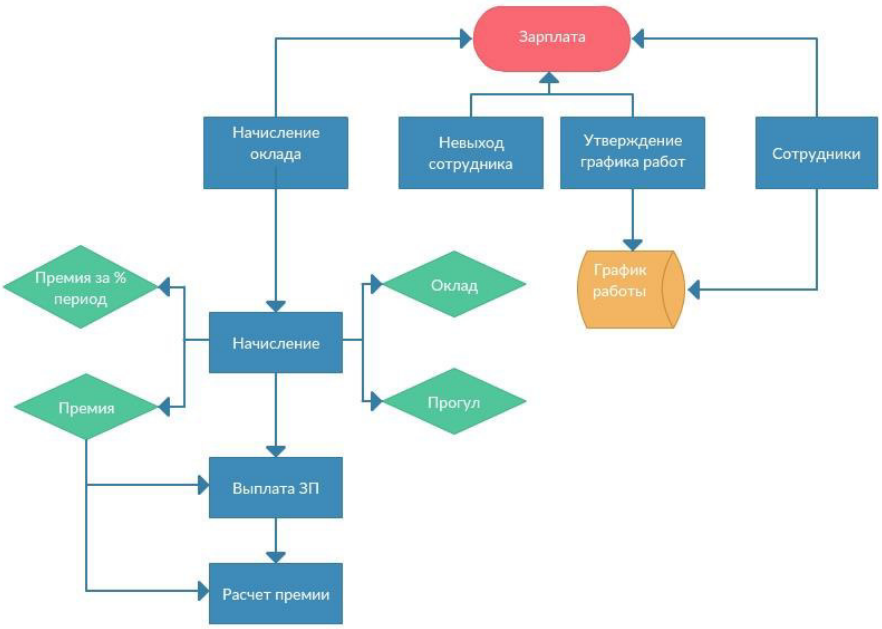

Рисунок 3 - зарплата

5. Добавление новой подсистемы «Бизнес процессы»:

– задачи настойчивых продаж;
– задачи приема сотрудника;
– настойчивая продажа;
– прием сотрудника на работу;
– точка маршрута;
– старт, сбор данных о клиенте, коммерческое предложение, условие продаж, подписание договора, отгрузка товаров, завершение;
– старт, подготовка рабочего места, знакомство с офисом, знакомство с коллективом, завершение;

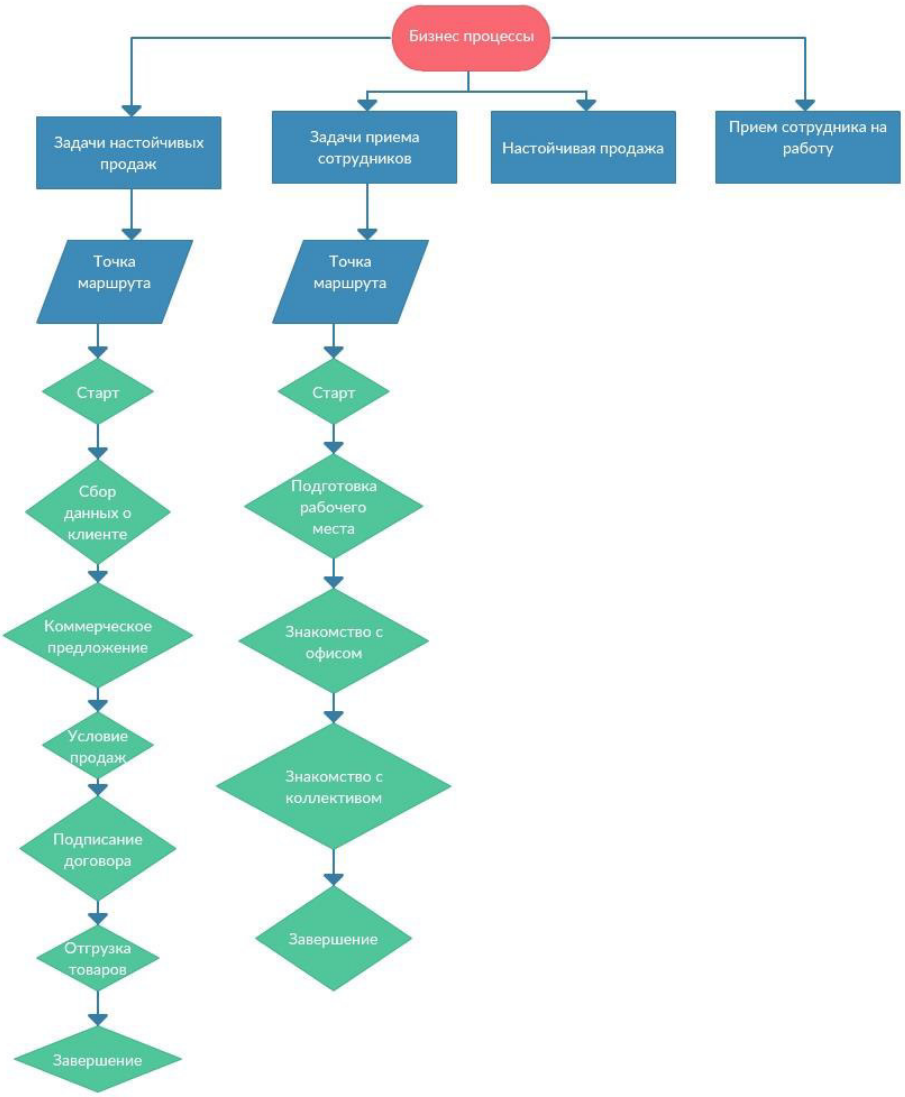

Рисунок 4 – бизнес процессы

Приложение Б
(обязательное)

Листинг программы

```
#Область ОписаниеПеременных

// СтандартныеПодсистемы

// Хранилище глобальных переменных.
//
// ПараметрыПриложения - Соответствие - хранилище переменных, где:
//          * Ключ - Строка - имя переменной в формате "ИмяБиблиотеки.ИмяПеременной";
//   * Значение - Произвольный - значение переменной.
//
// Инициализация (на примере СообщенияДляЖурналаРегистрации):
//                                                ИмяПараметра = "СтандартныеПодсистемы.СообщенияДляЖурналаРегистрации";
//   Если ПараметрыПриложения[ИмяПараметра] = Неопределено Тогда
//          ПараметрыПриложения.Вставить(ИмяПараметра, Новый СписокЗначений);
//   КонецЕсли;
//
// Использование (на примере СообщенияДляЖурналаРегистрации):
//
ПараметрыПриложения["СтандартныеПодсистемы.СообщенияДляЖурналаРегистрации"].Добавить(...);
//
ПараметрыПриложения["СтандартныеПодсистемы.СообщенияДляЖурналаРегистрации"] = ...;
Перем ПараметрыПриложения Экспорт;

// Конец СтандартныеПодсистемы

// ЭлектронныеСчетаФактуры
Перем СеансовыеДанныеЭСФ Экспорт;
// Конец ЭлектронныеСчетаФактуры

// ТехнологияСервиса
```

```
	Перем
ОповещениеПриПримененииЗапросовНаИспользованиеВнешнихРесурсовВМоделиСервиса Экспорт;
	// Конец ТехнологияСервиса

	//РаботаСВнешнимОборудованием
	Перем глПодключаемоеОборудование Экспорт; // для кэширования на клиенте

	Перем глДоступныеТипыОборудования Экспорт;
	//Конец РаботаСВнешнимОборудованием

	#КонецОбласти

	#Область ОбработчикиСобытий

	Процедура ПередНачаломРаботыСистемы()

	// СтандартныеПодсистемы
	СтандартныеПодсистемыКлиент.ПередНачаломРаботыСистемы();
	// Конец СтандартныеПодсистемы

	// ИнтернетПоддержкаПользователей
	ИнтернетПоддержкаПользователейКлиент.ПередНачаломРаботыСистемы();
	// Конец ИнтернетПоддержкаПользователей

	// ПодключаемоеОборудование
	МенеджерОборудованияКлиент.ПередНачаломРаботыСистемы();
	// Конец ПодключаемоеОборудование

	КонецПроцедуры

	Процедура ПриНачалеРаботыСистемы()

	// СтандартныеПодсистемы
	СтандартныеПодсистемыКлиент.ПриНачалеРаботыСистемы();
	// Конец СтандартныеПодсистемы
```

```
    //РаботаСВнешнимОборудованием
    МенеджерОборудованияКлиент.ПриНачалеРаботыСистемы();
    //Конец РаботаСВнешнимОборудованием

    КонецПроцедуры

    Процедура ПередЗавершениемРаботыСистемы(Отказ,
ТекстПредупреждения)

    // СтандартныеПодсистемы
    СтандартныеПодсистемыКлиент.ПередЗавершениемРаботыСистемы(Отка
з, ТекстПредупреждения);
    // Конец СтандартныеПодсистемы

    // ПодключаемоеОборудование
    МенеджерОборудованияКлиент.ПередЗавершениемРаботыСистемы();
    // Конец ПодключаемоеОборудование

    КонецПроцедуры

    Процедура ОбработкаВнешнегоСобытия(Источник, Событие, Данные)

    // ПодключаемоеОборудование
    // Подготовить данные
    ОписаниеСобытия = Новый Структура();
    ОписаниеОшибки  = "";

    ОписаниеСобытия.Вставить("Источник", Источник);
    ОписаниеСобытия.Вставить("Событие",  Событие);
    ОписаниеСобытия.Вставить("Данные",   Данные);

    // Передать на обработку данные.
    Результат =
МенеджерОборудованияКлиент.ОбработатьСобытиеОтУстройства(ОписаниеСо
бытия, ОписаниеОшибки);
    Если Не Результат Тогда
```

```
        ОбщегоНазначенияКлиентСервер.СообщитьПользователю(НСтр("ru='При
обработке внешнего события от устройства произошла ошибка.'")
                                                + Символы.ПС + ОписаниеОшибки);
        КонецЕсли;
        // Конец ПодключаемоеОборудование

        КонецПроцедуры

        #КонецОбласти
        #Если Сервер Или ТолстыйКлиентОбычноеПриложение Или
ВнешнееСоединение Тогда

        #Область СлужебныеПроцедурыИФункции

        // Установить или снять блокировку информационной базы,
        // исходя из значений реквизитов обработки.
        //
        Процедура ВыполнитьУстановку() Экспорт

        ВыполнитьУстановкуБлокировки(ЗапретитьРаботуПользователей);

        КонецПроцедуры

        // Отменить ранее установленную блокировку сеансов.
        //
        Процедура ОтменитьБлокировку() Экспорт

        ВыполнитьУстановкуБлокировки(Ложь);

        КонецПроцедуры

        // Зачитать параметры блокировки информационной базы
        // в реквизиты обработки.
        //
        Процедура ПолучитьПараметрыБлокировки() Экспорт

        Если Пользователи.ЭтоПолноправныйПользователь(, Истина) Тогда
                ТекущийРежим = ПолучитьБлокировкуСеансов();
```

```
            КодДляРазблокировки = ТекущийРежим.КодРазрешения;
        Иначе
            ТекущийРежим                                    =
СоединенияИБ.ПолучитьБлокировкуСеансовОбластиДанных();
        КонецЕсли;

        ЗапретитьРаботуПользователей = ТекущийРежим.Установлена;
        СообщениеДляПользователей                           =
СоединенияИБКлиентСервер.ИзвлечьСообщениеБлокировки(ТекущийРежим.Со
общение);

        Если ЗапретитьРаботуПользователей Тогда
            НачалоДействияБлокировки    = ТекущийРежим.Начало;
            ОкончаниеДействияБлокировки = ТекущийРежим.Конец;
        Иначе
            // Если блокировка не установлена, можно предположить, что
            // пользователь открыл форму для установки блокировки.
            // Поэтому устанавливаем дату блокировки равной текущей дате.
            НачалоДействияБлокировки                        =
НачалоМинуты(ТекущаяДатаСеанса() + 5 * 60);
        КонецЕсли;

        КонецПроцедуры

        Процедура ВыполнитьУстановкуБлокировки(Значение)

        Если Пользователи.ЭтоПолноправныйПользователь(, Истина) Тогда
            Блокировка = Новый БлокировкаСеансов;
            Блокировка.КодРазрешения    = КодДляРазблокировки;
        Иначе
            Блокировка                                      =
СоединенияИБ.НовыеПараметрыБлокировкиСоединений();
        КонецЕсли;

        Блокировка.Начало          = НачалоДействияБлокировки;
        Блокировка.Конец           = ОкончаниеДействияБлокировки;
```

```
	Блокировка.Сообщение = СоединенияИБ.СформироватьСообщениеБлокировки(СообщениеДляПользователей,
		КодДляРазблокировки);
	Блокировка.Установлена = Значение;

	Если Пользователи.ЭтоПолноправныйПользователь(, Истина) Тогда
		УстановитьБлокировкуСеансов(Блокировка);
	Иначе

	СоединенияИБ.УстановитьБлокировкуСеансовОбластиДанных(Блокировка);
	КонецЕсли;

	КонецПроцедуры

	#КонецОбласти

	#КонецЕсли
	#Если Сервер Или ТолстыйКлиентОбычноеПриложение Или ВнешнееСоединение Тогда

	////////////////////////////////////////////////////////////////////////////////
	// СЛУЖЕБНЫЕ ПРОЦЕДУРЫ И ФУНКЦИИ

	////////////////////////////////////////////////////////////////////////////////
	// Процедуры и функции выгрузки данных отчета в XML

	//Выгружает данные отчета в формат XML
	// Параметры:
	//	РевизияВыгрузки - ревизия выгрузки файла, определяет структуру файла
	//
	Функция ВыгрузитьОтчетВXMLДляСОНО(ДокументДляВыгрузки, ЗНАЧ РевизияВыгрузки) Экспорт

	ДанныеДляВыгрузки = ЗаполнениеДанныхДляВыгрузки(ДокументДляВыгрузки);
```

```
Префикс  = РевизияВыгрузки;
ИмяФайла = ПолучитьИмяВременногоФайла("xml");

// Запишем общие атрибуты (code, version) и создадим элемент form.
Запись = Новый ЗаписьXML;
Запись.ОткрытьФайл(ИмяФайла,"UTF-8");
Запись.ЗаписатьОбъявлениеXML();

// Запишем корневой элемент.
Запись.ЗаписатьНачалоЭлемента("fno");

МакетВыгрузки = ПолучитьМакет("ВыгрузкаBXML");

ТаблОбщиеРеквизиты = Новый ТаблицаЗначений;
ТаблОбщиеРеквизиты.Колонки.Добавить("ИмяАтрибута");
ТаблОбщиеРеквизиты.Колонки.Добавить("ЗначениеАтрибута");

ОбластьОбщихРеквизитов = МакетВыгрузки.Область("ОбщиеРеквизиты"
+ Префикс);
Для Ном = ОбластьОбщихРеквизитов.Верх По
ОбластьОбщихРеквизитов.Низ Цикл
    Строка = ТаблОбщиеРеквизиты.Добавить();
    Строка.ИмяАтрибута =
СокрП(МакетВыгрузки.Область(Ном, 1).Текст);
    Строка.ЗначениеАтрибута = СокрП(МакетВыгрузки.Область(Ном,
2).Текст);
КонецЦикла;

// Запишем атрибуты корневого элемента.
Для Каждого Строка Из ТаблОбщиеРеквизиты Цикл
    Запись.ЗаписатьАтрибут(Строка.ИмяАтрибута,
Строка.ЗначениеАтрибута);
КонецЦикла;

// Создадим таблицу для выгрузки.
ТаблицаВыгрузки = Новый ТаблицаЗначений;
ТаблицаВыгрузки.Колонки.Добавить("Форма");
```

```
    ТаблицаВыгрузки.Колонки.Добавить("Страница");
    ТаблицаВыгрузки.Колонки.Добавить("ИмяАтрибута");
    ТаблицаВыгрузки.Колонки.Добавить("ИмяЯчейки");
    ТаблицаВыгрузки.Колонки.Добавить("КоличествоROW");
    ТаблицаВыгрузки.Колонки.Добавить("ГруппаROW");
    ТаблицаВыгрузки.Колонки.Добавить("ИмяФормы");
    ТаблицаВыгрузки.Колонки.Добавить("ИмяЛиста");
    ТаблицаВыгрузки.Колонки.Добавить("Отображение");

    ТаблицаВыгрузки.Колонки.Добавить("КодФормы");
    ТаблицаВыгрузки.Колонки.Добавить("Выгружать");
    ТаблицаВыгрузки.Колонки.Добавить("Многострочность");
    ТаблицаВыгрузки.Колонки.Добавить("ЗаголовокФормы");
    ТаблицаВыгрузки.Колонки.Добавить("КоличествоНаЛисте");
    ТаблицаВыгрузки.Колонки.Добавить("Значение");

    ОбластьВыгрузки = МакетВыгрузки.Область("Выгрузка" + Префикс);
    ТекущееИмяФормы = Неопределено;
    Для Ном = ОбластьВыгрузки.Верх По ОбластьВыгрузки.Низ Цикл
        Строка = ТаблицаВыгрузки.Добавить();
        Строка.Форма = СокрП(МакетВыгрузки.Область(Ном,
1).Текст);
        Строка.Страница = СокрП(МакетВыгрузки.Область(Ном,
2).Текст);
        Строка.ИмяАтрибута = СокрП(МакетВыгрузки.Область(Ном,
3).Текст);
        Строка.ИмяЯчейки = СокрП(МакетВыгрузки.Область(Ном,
4).Текст);
        Строка.КоличествоROW = СокрП(МакетВыгрузки.Область(Ном,
5).Текст);
        Строка.ИмяФормы = СокрП(МакетВыгрузки.Область(Ном,
6).Текст);
        Строка.ИмяЛиста = СокрП(МакетВыгрузки.Область(Ном,
7).Текст);
        Строка.Отображение = СокрП(МакетВыгрузки.Область(Ном,
8).Текст);
        Строка.ГруппаROW = СокрП(МакетВыгрузки.Область(Ном,
9).Текст);
```

```
            НовоеИмяФормы = Строка.ИмяФормы;
            Если НовоеИмяФормы <> ТекущееИмяФормы Тогда
                Попытка
                    Макет = ПолучитьМакет("Показатели");
                    Форма = Макет.ПолучитьОбласть(Строка.ИмяФормы
+"_"+ Префикс);
                Исключение
                    Форма = Неопределено;
                КонецПопытки;
            КонецЕсли;
            Если Форма <> Неопределено Тогда
                Строка.КодФормы = СокрП(Форма.Область(1,
1).Текст);
                Строка.Выгружать = Число(Форма.Область(1,
4).Текст);
                Строка.Многострочность = Число(Форма.Область(1,
6).Текст);
                Строка.ЗаголовокФормы = СокрП(Форма.Область(1,
9).Текст);
                Строка.КоличествоНаЛисте = Число(Форма.Область(1,
10).Текст);
                Строка.Значение = "";
            КонецЕсли;
            ТекущееИмяФормы = Строка.ИмяФормы;
        КонецЦикла;

        // Выгрузка в XML.
        КоличествоСтрокВТаблице = ТаблицаВыгрузки.Количество();
        НомерСтроки = 0;

        ТекущаяФорма = Неопределено;
        ТекущаяСтраница = Неопределено;

        Пока Истина Цикл
            Если НомерСтроки >= КоличествоСтрокВТаблице Тогда
                Прервать;
            КонецЕсли;
```

```
            ВыгрузитьСтрокуВXML(Запись, ТаблицаВыгрузки,
КоличествоСтрокВТаблице, НомерСтроки, ДанныеДляВыгрузки,
ТекущаяФорма, ТекущаяСтраница);
            НомерСтроки = НомерСтроки + 1;

        КонецЦикла;

        Запись.ЗаписатьКонецЭлемента();          // </sheet>
        Запись.ЗаписатьКонецЭлемента();          // </sheetGroup>
        Запись.ЗаписатьКонецЭлемента();          // </form>

        Запись.ЗаписатьКонецЭлемента();    // fno

        Запись.Закрыть();

        // Данный текст модуля добавлен для совместимости с COHO (в COHO для
UTF-8 нет BOM поля).
        ТекстДок = Новый ТекстовыйДокумент;
           ТекстДок.Прочитать(ИмяФайла, "windows-1251");
           Строка = ТекстДок.ПолучитьСтроку(1);
           ТекстДок.ЗаменитьСтроку(1,Прав(Строка,СтрДлина(Строка)-3));
           ТекстДок.Записать(ИмяФайла, "windows-1251");

        ДвоичныеДанныеФайла = Новый ДвоичныеДанные(ИмяФайла);

        ФайлВременный = Новый Файл(ИмяФайла);
        Попытка
            ФайлВременный.УстановитьТолькоЧтение(Ложь);
            УдалитьФайлы(ИмяФайла);
        Исключение
            ЗаписьЖурналаРегистрации(
                НСтр("ru                                                    =
'ВыгрузкаВXMLЗаявленияОВвозеТоваров.Удаление файлов при выгрузке'",
                    ОбщегоНазначенияКлиентСервер.КодОсновногоЯзыка()),
                УровеньЖурналаРегистрации.Ошибка,
                ,
                ,
                ИнформацияОбОшибке());
```

```
КонецПопытки;

Возврат ПоместитьВоВременноеХранилище(ДвоичныеДанныеФайла);

КонецФункции

// Формирует построчно структуру XML файла из макета "ВыгрузкаВXML"
//
Процедура ВыгрузитьСтрокуВXML(Запись, ТаблицаВыгрузки, КоличествоСтрокВТаблице, НомерСтроки, ДанныеДляВыгрузки, ТекущаяФорма = Неопределено, ТекущаяСтраница = Неопределено)

СтрокаВыгрузки  = ТаблицаВыгрузки.Получить(НомерСтроки);
НоваяФорма      = СтрокаВыгрузки.Форма;
НоваяСтраница   = СтрокаВыгрузки.Страница;
Если ТекущаяФорма <> Неопределено И НоваяФорма <> ТекущаяФорма Тогда
  Запись.ЗаписатьКонецЭлемента();                    // </sheet>
КонецЕсли;
Если НЕ НоваяФорма = ТекущаяФорма Тогда
    Если НЕ ТекущаяФорма = Неопределено Тогда
        Запись.ЗаписатьКонецЭлемента();              // </sheetGroup>
        Запись.ЗаписатьКонецЭлемента();              // </form>
    КонецЕсли;
    Запись.ЗаписатьНачалоЭлемента("form");
    Запись.ЗаписатьАтрибут("name",СтрокаВыгрузки.Форма);
    Запись.ЗаписатьНачалоЭлемента("sheetGroup");
КонецЕсли;
Если НЕ НоваяСтраница = ТекущаяСтраница Тогда
    Если ТекущаяФорма <> Неопределено И НоваяФорма <> ТекущаяФорма Тогда
        // ничего не делаем
  ИначеЕсли НЕ ТекущаяСтраница = Неопределено Тогда
        Запись.ЗаписатьКонецЭлемента();              // </sheet>
    КонецЕсли;
    Запись.ЗаписатьНачалоЭлемента("sheet");
```

```
            Запись.ЗаписатьАтрибут("name",СтрокаВыгрузки.Страница);
        КонецЕсли;

        Если СтрокаВыгрузки.Многострочность <> Неопределено И
СтрокаВыгрузки.Многострочность = 1 Тогда
            ВыгрузитьМногострочнуюФорму(Запись, ТаблицаВыгрузки,
КоличествоСтрокВТаблице, НомерСтроки, ДанныеДляВыгрузки);
        // проверим многострочное ли значение
        ИначеЕсли ЗначениеЗаполнено(СтрокаВыгрузки.КоличествоROW) И
СтрокаВыгрузки.КоличествоROW <> "0" Тогда  // для обычных форм
            // если в обычной форме встретилась многострочность
            КоличествоПовторений =
Число(СтрокаВыгрузки.КоличествоROW);
            ГруппаRow = СтрокаВыгрузки.ГруппаROW;

            Для Н = 1 По КоличествоПовторений Цикл
                Запись.ЗаписатьНачалоЭлемента("row");
                СтрокаВыгрузки = ТаблицаВыгрузки.Получить(НомерСтроки);
                Если Н = 1 Тогда
                    ТаблицаМногострочныхТегов =
ТаблицаВыгрузки.СкопироватьКолонки();
```