

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего
образования

«Алтайский государственный университет»

Международный институт экономики, менеджмента и информационных систем

Кафедра цифровых технологий и бизнес-аналитики

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ УЧЕТА ОПЛАТЫ ЗА ПОТРЕБЛЕНИЕ
ЭЛЕКТРОЭНЕРГИИ
(курсовая работа)

Выполнил студент
2 курса, 2009а группы
Д.С. Беляев

(подпись)

Научный руководитель
канд. физ.-мат. наук, доцент
А.Ю. Юдинцев

(подпись)

Курсовая работа защищена
«__»_____2022 г.
Оценка_____

(подпись)

Барнаул 2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	3
1 АНАЛИЗ ТЕХНОЛОГИЙ РАЗРАБОТКИ ПРИЛОЖЕНИЯ	5
1.1 Рассмотрение технологий по разработке баз данных	5
1.2 Рассмотрение технологий по разработке клиентских частей приложения...	8
1.3 Анализ готовых программных продуктов.....	12
2 ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ	16
2.1 Проектирование базы данных	16
2.2 Проектирование клиентских частей	20
2.3 Выбор технологий разработки приложения	27
3 РАЗРАБОТКА ПРИЛОЖЕНИЯ	29
3.1 Разработка базы данных.....	29
3.2 Разработка клиентских частей.....	35
ЗАКЛЮЧЕНИЕ	42
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ.....	43
ПРИЛОЖЕНИЕ.....	45

ВВЕДЕНИЕ

Одним из важных и весомых секторов розничных рынков электрической энергии, на которых функционируют энергосбытовые организации, является сектор продаж электрической энергии населению. Сейчас энергоснабжение потребителей во многом определяется способностью энергосбытовых предприятий адаптироваться к постоянно меняющимся условиям внутри страны и на внешних рынках. Развитие методов ведения энергосбытовой деятельности при использовании информационных технологий может существенно повысить эффективность этой деятельности. Таким образом, торговля энергоресурсами основана на использовании автоматизированного приборного энергоучета, сводящего к минимуму участие человека на этапе измерения, сбора и обработки данных и обеспечивающего достоверный, точный, оперативный и гибкий, адаптируемый к различным тарифным системам учет со стороны поставщика энергоресурсов и потребителя.

Эффективность деятельности энергосбытовой организации зависит как от темпов внедрения автоматизированной системы коммерческого учета электроэнергии (АСКУЭ), так и от возможностей, внедренных разработчиком в АСКУЭ. Необходимость разработки и совершенствования такой системы учета электроэнергии на конкретном предприятии обуславливает тему и цель курсовой работы.

Объект работы – организация, занимающаяся сбытом электроэнергии.

Предметом являются технологии разработки программных приложений для учета оплаты за потребление электроэнергии.

Цель работы – разработка приложения для учета оплаты за потребление электроэнергии для энергосбытовой организации.

Задачи:

- провести анализ технологий разработки приложений;
- провести анализ готовых программных продуктов;
- спроектировать и разработать приложение для энергосбытовой организации.

Работа состоит из введения, трех частей, заключения, списка использованных источников и литературы, двух приложений.

Во введении обоснована актуальность выбранной темы, определены объект, предмет, цель и задачи работы, описана структура работы.

Первая часть посвящена рассмотрению технологий разработки приложений.

Во второй части описано проектирование базы данных и обоснован выбор технологий разработки программного продукта.

В третьей части представлены разработка приложения для учета оплаты за потребление электроэнергии и инструкция по работе с приложением.

В заключении приведены основные выводы по работе.

В приложении приведены программный код разработки базы данных и код клиентских частей приложения.

Список использованных источников и литературы включает 17 источников.

1 АНАЛИЗ ТЕХНОЛОГИЙ РАЗРАБОТКИ ПРИЛОЖЕНИЯ

1.1 Рассмотрение технологий по разработке баз данных

Работа организаций основана на использовании большого массива информации. Для того, чтобы этот массив как-то хранить, придумали базу данных. А для обработки и управления данными в базах появилась система управления базами данных или сокращенно СУБД.

База данных (database) – это организованная совокупность совместно используемых логически связанных данных и описаний этих данных, относящаяся к определенной предметной области, предназначенная для удовлетворения информационных потребностей организации.

Система управления базами данных (Database Management System, DBMS) – это комплекс программных средств, с помощью которого можно создавать и поддерживать базу данных, а также осуществлять к ней контролируемый доступ пользователей. [2]

Основной отличительный признак классификации СУБД – модель реализации данных. Модель данных – это совокупность структур данных и операций их обработки. [3] Существуют сетевая, иерархическая, реляционная, объектно-ориентированная и другие модели реализации базы данных.

Иерархическая модель базы данных подразумевает, что элементы организованы в структуры, связанные между собой иерархическими или древовидными связями. Родительский элемент может иметь несколько дочерних элементов. Но у дочернего элемента может быть только один предок. Такая модель подразумевает возможность существования одинаковых (преимущественно дочерних) элементов.

Сетевая модель базы данных подразумевает, что у родительского элемента может быть несколько потомков, а у дочернего элемента — несколько предков. Записи в такой модели связаны списками с указателями.

В реляционной модели, в отличие от иерархической или сетевой, не существует физических отношений. Вся информация хранится в виде таблиц (отношений), состоящих из рядов и столбцов. А данные двух таблиц связаны общими столбцами, а не физическими ссылками или указателями. Для манипуляций с рядами данных существуют специальные операторы.

В отличие от двух других типов СУБД, в реляционных моделях данных нет необходимости просматривать все указатели, что облегчает выполнение запросов на выборку информации по сравнению с сетевыми и иерархическими СУБД. Это одна из основных причин, почему реляционная модель оказалась более удобна. [3]

В перечень получивших широкое признание многопользовательских реляционных СУБД входят: Oracle, SQL Server компании Microsoft, InterBase, Firebird, MySQL, PostgreSQL, Informix и т. д. [2]

При этом среди реляционных СУБД по архитектуре доступа в большинстве своём используют клиент-серверную технологию, а язык запроса – SQL.

Рассмотрим следующие популярные реляционные СУБД:

1. Microsoft SQL Server
2. MySQL
3. PostgreSQL

Microsoft SQL Server является одной из наиболее популярных систем управления базами данных (СУБД) в мире. Данная СУБД подходит для самых различных проектов: от небольших приложений до больших высоконагруженных проектов.

SQL Server долгое время был исключительно системой управления базами данных для Windows, однако начиная с версии 16 эта система доступна и на Linux.

SQL Server характеризуется такими особенностями как:

- Производительность.
- Надежность и безопасность. SQL Server предоставляет шифрование данных.
- Простота. С данной СУБД относительно легко работать и вести администрирование. [4]

Существенным недостатком Microsoft SQL Server является большая стоимость приобретения лицензии.

MySQL является наиболее приспособленной для применения в среде web СУБД (системой управления базами данных). Для исполнения приложений клиента на большинстве хостинг-площадок провайдеры предоставляют небольшое количество ресурсов (как вычислительных, так и дисковых). Поэтому для данного применения необходима высокоэффективная СУБД, обладающая при этом высокой надежностью, так как большинство web-приложений и сайтов должны работать в режиме 24/7.

По всем этим причинам MySQL стала незыблемым стандартом в области СУБД для web, а теперь в ней развиваются возможности для использования ее в любых критичных бизнес-приложениях, то есть конкурирует на равных с СУБД таких производителей, как Oracle, IBM, Microsoft и Sybase.

Основные преимущества MySQL:

1. многопоточность, поддержка нескольких одновременных запросов;
2. оптимизация связей с присоединением многих данных за один проход;
3. записи фиксированной и переменной длины;
4. ODBC драйвер;
5. гибкая система привилегий и паролей;
6. гибкая поддержка форматов чисел, строк переменной длины и меток времени;
7. интерфейс с языками C и Perl, PHP. [5]

Самое главное преимущество – есть возможность получить СУБД бесплатно.

Однако имеются и следующие недостатки:

1. Нет встроенной поддержки OLAP или XML;
2. Усилия и время, необходимые MySQL для выполнения некоторых действий, таких как создание инкрементных резервных копий, намного больше по сравнению с другими системами.

СУБД PostgreSQL позиционируется в качестве профессионального решения. В отличие от многих других аналогичных современных систем, эта максимально полно поддерживает синтаксис SQL. Она имеет множество функций, которые необходимы

приложениям, предъявляющим очень высокие требования к надежности и безопасности.

СУБД отличается надежностью, из-за этого ее используют, например, банки, которым нужно максимально сократить риски потери данных или их несанкционированного изменения. [6]

Документации по данной СУБД намного меньше, чем по остальным СУБД, приведённым в списке, что является недостатком.

Учитывая специфику курсовой работы, необходима разработка приложения с использованием многопользовательской реляционной СУБД. Таким образом, для дальнейшей разработки приложения можно использовать Microsoft SQL Server, MySQL или PostgreSQL.

1.2 Рассмотрение технологий по разработке клиентских частей приложения

После того, как выбрали базу данных и систему управления базой данных, необходимо найти подходящий язык программирования и сопутствующие технологии, основанные на нем, чтобы разработать приложение.

На данный момент в разработке приложений для организаций наиболее используемыми языками являются объектно-ориентированные.

Объектно-ориентированное программирование (ООП) — это парадигма программирования, которая использует концепцию объектов для создания четко определенных фрагментов кода, которыми можно управлять. Парадигма программирования описывает способ организации программы. До появления ООП процедурное и структурное программирование были основными парадигмами программирования того времени. [7]

Концепция ООП такова, что разработчикам важно представить программу как систему взаимодействующих объектов. Каждый объект обладает свойствами или атрибутами, которые определены в классах объектов. После создания класса можно объявлять конкретные объекты данного класса. Например, разработчики могут определить класс «Системный блок» с такими атрибутами, как объём памяти ОЗУ,

количество портов USB, тактовая частота процессора, а затем создать несколько объектов или экземпляров. Помимо этого, в такой парадигме можно наследовать классы, то есть разработчики определяют основной или родительский класс с некоторыми атрибутами, а затем добавляют производный от него класс с дополнительными атрибутами. Инкапсуляция позволяет разработчикам скрыть некоторые части кода для пользователей, делая такие части недоступными для изменения пользователями. Другой особенностью является полиморфизм. Данная особенность разрешает разработчику создавать для одних и тех же методов разную реализацию. Так метод, описывающий сложение чисел, может быть реализован для сложения строк. Для разработки крупных проектов ООП является лучшей парадигмой программирования, так как она дает разработчику возможность быстро вносить коррективы в код благодаря описанным выше возможностям.

Рейтинг языков [8] гласит о том, что самыми популярными языками программирования в 2022 году являются следующие:

1. Python;
2. Java;
3. C;
4. C#;
5. C++;
6. JavaScript.

Python – это язык программирования общего назначения, распространяемый с открытыми исходными текстами (open source). Он оптимизирован для создания качественного программного обеспечения, высокой производительности труда разработчиков, переносимости программ и интеграции компонентов. Язык Python используется сотнями тысяч разработчиков по всему миру в таких областях, как создание веб-сценариев, системное программирование, создание пользовательских интерфейсов, настройка программных продуктов под пользователя, численное программирование и в других. [9, т. 1]

На основе Python при создании сайтов используют следующие фреймворки:

- Django;

- Flask;
- Pyramid;
- CherryPy;

Java — это достаточно универсальный язык программирования, который часто используется для веб-разработки и для разработки под Android.

Программы на Java транслируются в байт-код, который затем выполняется виртуальной машиной Java (JVM). JVM — это программа, которая обрабатывает байтовый код и передает инструкции оборудованию как интерпретатор. Достоинством подобной реализации является независимость байт-кода от операционной системы и оборудования, что позволяет выполнять Java-приложения на любом устройстве, для которого существует JVM. [10]

На основе Java используются следующие платформы и библиотеки:

1. Spring;
2. Hibernate;
3. Play;
4. Vaadin.

Язык программирования Си — универсальный язык программирования, который завоевал особую популярность у программистов, благодаря сочетанию возможностей языков программирования высокого и низкого уровней. Большинство программистов предпочитают использовать язык Си для серьезных разработок потому, что их привлекают такие особенности языка, как свобода выражения мыслей, мобильность и чрезвычайная доступность.

Язык Си даёт возможность программисту осуществлять непосредственный доступ к ячейкам памяти и регистрам компьютера, требуя при этом знания особенностей функционирования ЭВМ. В этом Си схож с языком низкого уровня — ассемблером, хотя на самом деле он представляет собой гораздо более мощное средство решения трудных задач и создания сложных программных систем. [11]

Язык C# создан для взаимодействия с фреймворком .NET. Он перенял многое от своих предшественников — языков C++, Delphi, Smalltalk и, в особенности, Java. При разработке C# были взяты лучшие моменты из всех этих языков. Например, C#

в отличие от C++ не поддерживает множественное наследование классов. Так было решено по причине их неудобства использования. C# подходит в решении распространённых задач в сфере разработки крупных приложений, сохраняющих гибкость, расширяемость и масштабируемость. [12]

Фреймворк .NET включает в себя такие технологии, как:

- WPF;
- ASP.NET;
- Xamarin;
- Windows Forms.

C++ является мощным языком, унаследовав от Си богатые возможности по работе с памятью. Поэтому нередко C++ находит свое применение в системном программировании, в частности, при создании операционных систем, драйверов, различных утилит, антивирусов и т.д. К примеру, ОС Windows большей частью написана на C++. Но только системным программированием применение данного языка не ограничивается. C++ можно использовать в программах любого уровня, где важны скорость работы и производительность. Нередко он применяется для создания графических приложений, различных прикладных программ. Также особенно часто его используют для создания игр с богатой насыщенной визуализацией. Кроме того, в последнее время набирает ход мобильное направление, где C++ тоже нашел свое применение. И даже в веб-разработке также можно использовать C++ для создания веб-приложений или каких-то вспомогательных сервисов, которые обслуживают веб-приложения. В общем C++ - язык широкого пользования, на котором можно создавать практически любые виды программ. [13]

Использовать такие же технологии .NET для C++ получится, но необходима предварительная, долгая настройка проекта по сравнению с C# или VB, при этом есть фреймворки или библиотеки для него:

- Qt;
- POCO;
- Cairo.

JavaScript – это язык программирования для Веб. Подавляющее большинство веб-сайтов используют JavaScript, и все современные веб-браузеры – для настольных компьютеров, игровых приставок, электронных планшетов и смартфонов – включают интерпретатор JavaScript. Он является высокоуровневым, динамическим, нетипизированным и интерпретируемым языком программирования, который хорошо подходит для программирования в объектно-ориентированном и функциональном стилях. Свой синтаксис JavaScript унаследовал из языка Java, свои первоклассные функции – из языка Scheme, а механизм наследования на основе прототипов – из языка Self. [14]

По итогу, если учитывать все нюансы рассмотренных выше языков программирования и сопутствующих технологий, то для дальнейшей разработки приложения подойдут Python, Java, C#. При этом главный критерий выбора – быстрота разработки клиентских частей.

1.3 Анализ готовых программных продуктов

При разработке собственного программного продукта необходимо знать информацию о том, есть ли конкуренты на рынке, какое программное обеспечение они используют, какие достоинства и недостатки имеет программное обеспечение.

Рассмотрим следующую систему – «яЭнергетик».

Система имеет следующие возможности:

- Сбор текущих и архивных показаний;
- Настройка параметров качества электроэнергии;
- Синхронизация времени;
- Оповещения о критических параметрах электроэнергии;
- Удаленное отключение и управление нагрузкой электроэнергии;
- Чтение внутренних журналов электросчетчика;
- Оповещения о вмешательстве в приборы учета (нарушение пломб);
- Формирование пофидерного баланса;
- Контроль достоверности учета при приемке счетчиков;

- Формирование отчетов и ведомостей. [15]

Стоимость получения лицензии на программное обеспечение варьируется от количества счётчиков, которое стоит подключить к системе (рис. 1.1). Если необходимо подключить средний город до 100000 точек учета, то придется отдать за это один раз 5 миллионов 500 тысяч рублей. При этом годовая лицензия идет по договорённости. Если учитывается количество приборов учета до 5000, тогда годовая лицензия обойдется в 450 тысяч рублей, что также дорого.

Варианты поставки

■ КОЛИЧЕСТВО ПРИБОРОВ УЧЕТА — до 100 000



 <p>Годовая лицензия</p> <p>Использование системы, развернутой в датацентре в г. Санкт-Петербург</p> <hr/> <p>По договорённости</p> <p><small>*до 100 000 счётчиков</small></p>	 <p>Бессрочная лицензия</p> <p>Система устанавливается на серверах вашей организации. Целесообразно организациям от 5000 точек учета</p> <hr/> <p>5 500 000 руб единовременный платеж</p> <p><small>*до 100 000 счётчиков</small></p>
--	--

Рисунок 1.1- Проверка стоимости лицензий «яЭнергетик»

Второй программный продукт: 1С:ERP Энергетика 2. Продукт предназначен для автоматизации основных бизнес-процессов межрегиональных распределительных сетевых компаний и территориальных сетевых организаций (рис. 1.2).

Его некоторые возможности:

1. Ведение информации об объектах паспортизации;
2. Ведение сведений об организационных объектах и структуре сети;
3. Формирование базы данных потребителей (физических и юридических лиц);
4. Учет договоров энергоснабжения;
5. Учет передачи электроэнергии;

6. Формирование базы данных точек учета потребления электроэнергии и мощности;
7. Регистрация информации об отключениях точек учета;
8. Регистрация показаний приборов учета;
9. Учет потребляемых энергетических ресурсов;
10. Мониторинг и анализ показателей деятельности предприятия.[16]

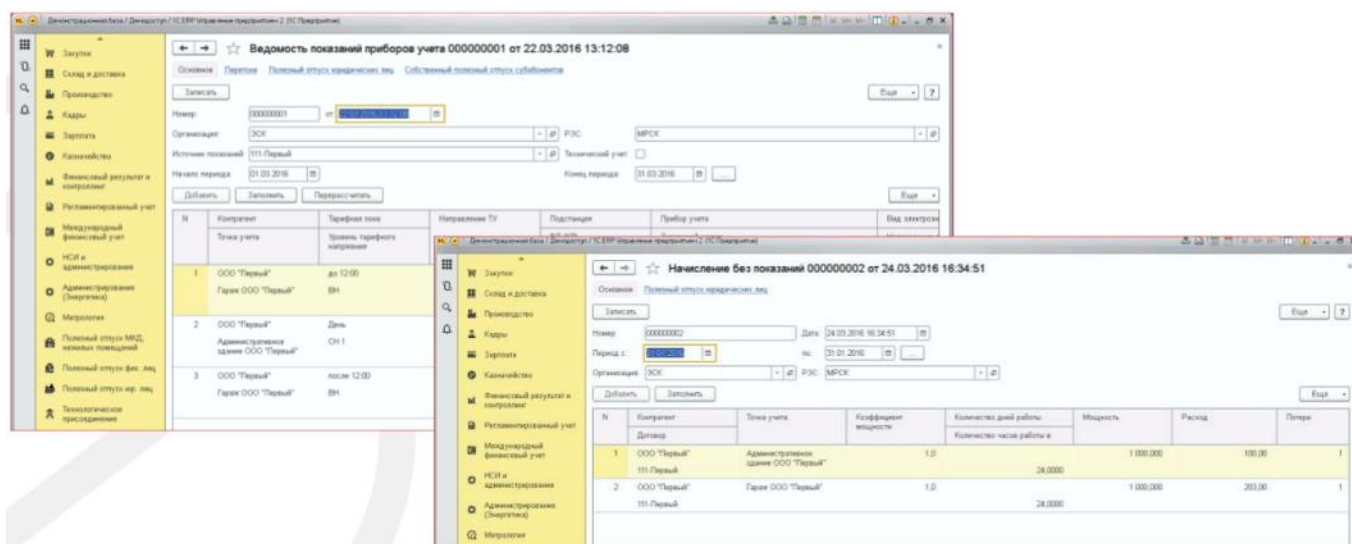


Рисунок 1.2 - Пример 1C:ERP Энергетика 2

Стоимость этого продукта также варьируется, однако не от количества подключаемых точек учета, а от количества клиентских лицензий на рабочие места организации. При этом также нужна основная версия для разработки. Стоит она 819 тысяч рублей. Предположим, что в организации задействованы в бизнес-процессе 100 человек, тогда лицензия на них обойдется в 1 миллион 80 тысяч, что в общем на организацию выходит 1 миллион 899 тысяч рублей. По сравнению с рассмотренным выше продуктом от «яЭнергетик», продукт от 1С обойдется намного дешевле. Лицензия 1С продукта бессрочная, в то время как продукт от «яЭнергетик» имеет годовой и бессрочный планы лицензирования. Также имеется основное преимущество разработки 1С перед конкурентом: интеграция с остальными продуктами 1С.

Главным недостатком среди этих продуктов является, конечно же, стоимость лицензии, несмотря на сравнительный анализ по стоимости между ними. Создаётся

ощущение, что собственная разработка для определённого количества приборов учёта по итогу выйдет дешевле, чем покупка лицензии у другой организации.

После того, как были проанализированы программные продукты для организаций по сбыту электроэнергии, можно приступать к проектированию приложения.

2 ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ

2.1 Проектирование базы данных

Чтобы спроектировать базу данных для массива информации, появляющейся в ходе использования приложения, необходимо разобраться с тем, что из себя представляет организация по сбыту электроэнергии и её деятельность на рынке, а также с тем, что необходимо для её проектирования.

Организация заключает договора как о покупке электроэнергии на оптовых рынках, так и о поставках электроэнергии населению. Поставки ведутся как физическим, так и юридическим лицам по выбранным тарифам: одноставочный, разделенный по двум зонам суток и по трём зонам суток. Каждому тарифу соответствует разная стоимость 1 кВт/ч электроэнергии. Также тарифы зависят от категории места проживания и определенных параметров, например, наличия возможности оборудования электроплиты в квартире. Среди абонентов организации есть категории людей, которым положены льготы: например, ветераны ВОВ или люди, получившие радиационное излучение вследствие радиоактивных испытаний или ликвидации последствий. Каждому потребителю, заключившему договор, соответствует определённый прибор учёта или, иным словом, счётчик, который автоматически на 30-е или 31-е число месяца передаёт показания в организацию для дальнейшей обработки. Приборы учёта бывают разных моделей, что необходимо учитывать в организации. Также в организации заинтересованы в неплательщиках за услуги поставок электроэнергии. У должников имеется статус долга, например, «не погашен». К обработке данных в системе допускаются не все, а наделённые правами люди, однако просматривать данные может большинство из сотрудников организации. В начале работы с системой работники регистрируются в ней. Роли наделаются администраторами.

Главными документами в данной предметной области являются: квитанция о необходимости оплаты услуг, договор об оказании услуг

Для проектирования базы данных сузим входные данные с точки зрения того, что организация проводит учёт показаний и оплаты за электроэнергию именно у физических лиц.

Ключевую позицию при проектировании реляционных баз данных занимает нормализация.

Нормализация – это метод проектирования базы данных, который позволяет привести базу данных к минимальной избыточности. [17]

Избыточность данных проявляется тогда, когда в разные таблицы добавляют одинаковые данные. Избыточность данных может привести к аномалиям, понижению производительности, неудобному управлению данными.

Выделяют несколько форм нормализации, но в основном для проектов достаточно третьей нормальной формы. Третья нормальная форма (3NF) подразумевает отсутствие зависимости одного неключевого столбца от значений другого. Таким образом, необходимо все время декомпозировать какие-то сущности на две и более других сущностей, чтобы не появились ошибки.

После того, как определились с предметной областью и необходимостью приведения базы данных к третьей нормальной форме, можно приступать непосредственно к проектированию.

Определим в базе данных следующие сущности:

1. Потребители;
2. Должники;
3. Статус долга;
4. Тарифные ставки;
5. Льготы;
6. Счётчики;
7. Тип счётчика;
8. Документы;
9. Шапки документов;
10. Показания счётчиков.

Первая сущность «Потребители» содержит основную информацию о потребителях организации: номер, имя, фамилия, отчество, дата рождения, улица проживания, номер дома, номер квартиры и наличие льгот. Имеет связь с сущностями «Тарифные ставки», «Льготы», «Счётчики».

Сущность «Должники» строится на основе сущности «Потребители» и содержит следующие атрибуты: номер должника, сумма долга, дата образования долга и фактическая дата погашения.

«Статус долга» содержит данные, которые следует использовать при обозначении статуса долга должника: код и наименование.

Атрибуты сущности «Тарифные ставки» таковы: номер ставки, тип ставки, ставка 1, ставка 2, ставка 3. Разные ставки нужны в соответствии с зонами суток.

Сущности «Льготы» соответствуют следующие атрибуты: номер, краткое наименование и описание.

Следующая сущность «Счётчики» содержит в себе такие атрибуты, как: код и серийный номер.

«Тип счётчика» определяет модель счётчика, точнее его наименование и содержит такие атрибуты: номер типа счётчика и наименование.

«Шапки документов» определяют, как документ выглядит на печати. Задаются атрибутами код, наименование и образец, основное содержание.

«Документы» содержат следующие атрибуты: номер договора, дата подписания.

Последняя сущность – «Показания счётчиков» - определена следующими атрибутами: дата снятия показания и показание счётчика.

На основе определённых сущностей и их атрибутов можно построить логическую модель базы данных. Строиться она будет в программе IBM Rational Data Architect (рис. 2.1). При проектировании удобно пользоваться естественным, русским языком.

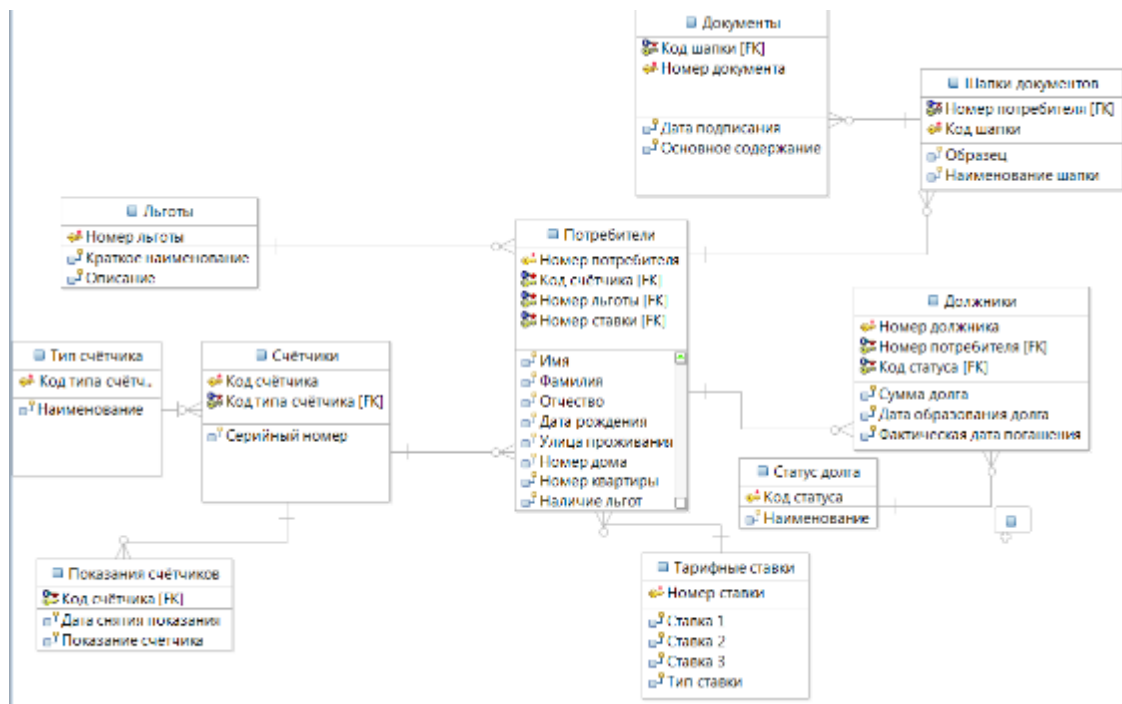


Рисунок 2.1 - Логическая модель данных

После этого на основе логической модели строится физическая модель данных (рис. 2.2). Подписываются связи между сущностями.

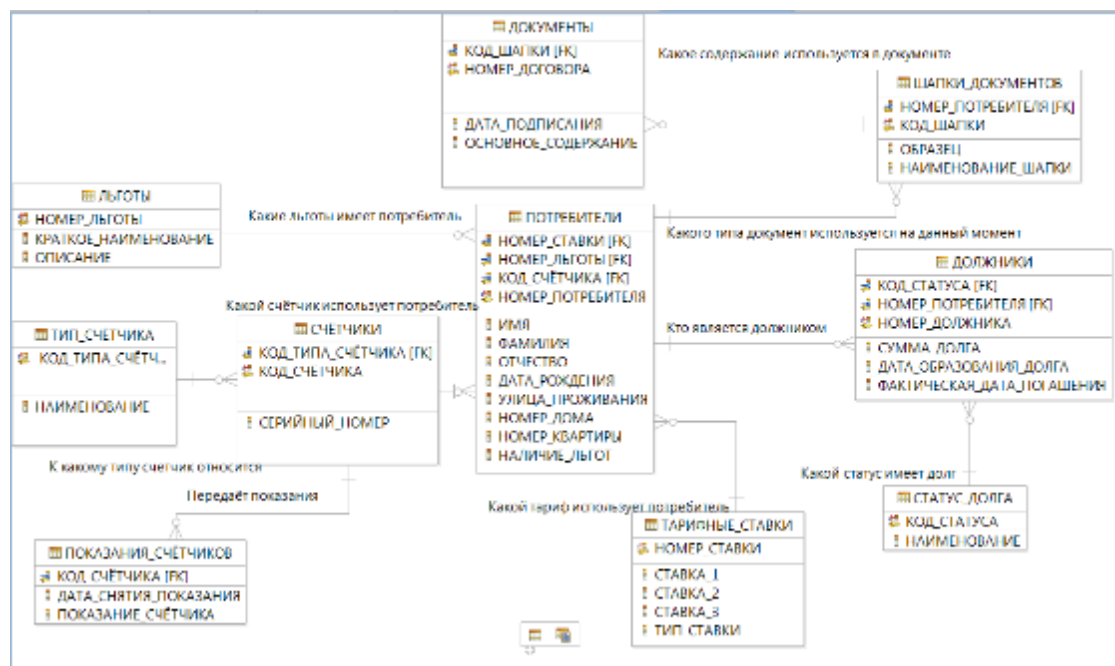


Рисунок 2.2 - Физическая модель данных

Также отдельно следует выделить две сущности: «Профили» и «Роли», входящие в предметную область лишь косвенно. Они нужны для хранения регистрационных данных сотрудников и соответствующих сотрудникам ролей. Атрибуты сущности «Профили»: номер, имя, фамилия, отчество, электронная почта,

логин, хэш пароля, соль, статус профиля. При регистрации пользователю автоматически присваивается активный статус. А атрибуты сущности «Роли» такие: код, наименование. Для них также были построены логическая и физическая модель (рис. 2.3).

Зашифрованные пароли профиля необходимы, так как при получении данных несанкционированным путём подобрать исходный пароль будет невероятно трудно. Забота о персональных данных потребителей выполняется на основании Федерального закона «О персональных данных». [1]

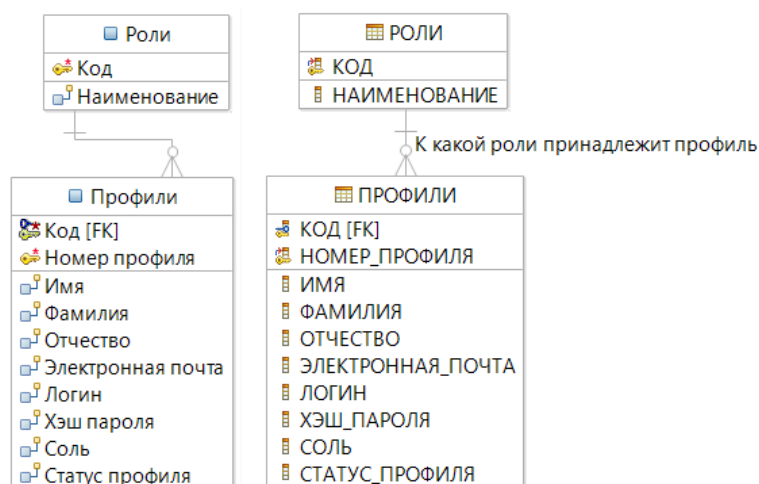


Рисунок 2.3 - Дополнительные логическая и физическая модели

Документы будут сформированы с помощью создания представлений, либо с помощью использования процедур. Например, для вывода документа можно использовать представление.

Некоторые процедуры будут созданы для просмотра статистических данных, то есть с использованием агрегатных функций. Например, можно создать процедуру для просмотра актуального списка должников.

После того, как спроектирована база данных, необходимо приступить к проектированию клиентских частей приложения.

2.2 Проектирование клиентских частей

Для того, чтобы работать с базой данных, необходимо спроектировать интуитивно понятные клиентские части.

Несмотря на удобство использования интерфейса MDI, будет использован собственный интерфейс приложения.

Отправной точкой для использования системы является регистрация и авторизация в системе. Изначально открывается форма с приветствием, двумя кнопками. Одна кнопка отвечает за регистрацию, вторая за авторизацию. Выглядеть форма будет примерно так, как на рисунке 2.4.

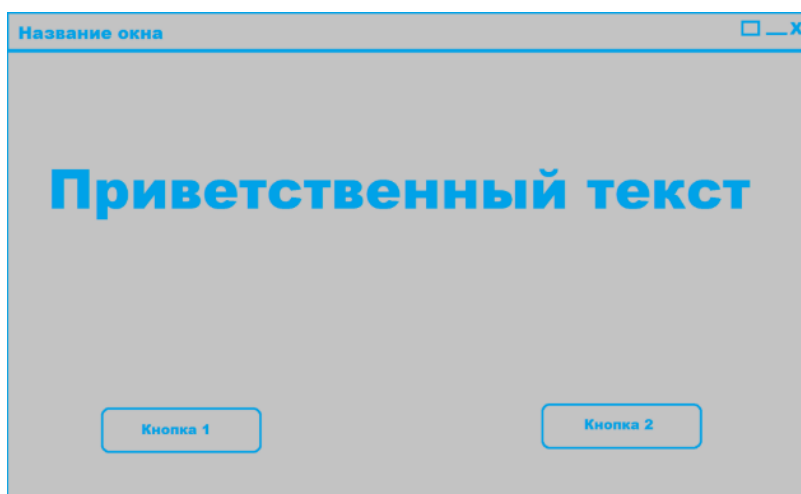


Рисунок 2.4 – Макет начального окна

При нажатии на первую кнопку выводится диалоговое окно регистрации (рис. 2.5). Метки 1,2,3,4,5,6 отвечают за названия полей, поля по порядку используются для имени, фамилии, отчества, электронной почты, логина и пароля. Кнопка 1 отвечает за регистрацию в системе, а кнопка 2 – за возвращение назад к начальному окну. Метка 7 отвечает за системные сообщения.

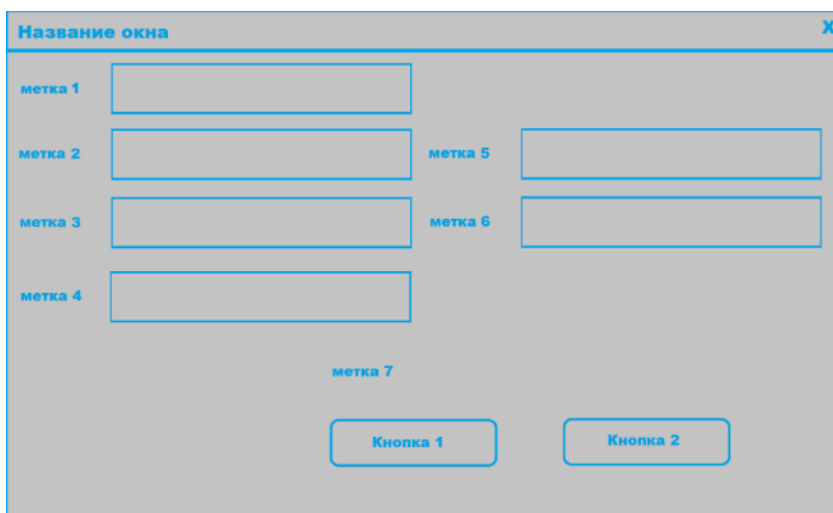


Рисунок 2.5 – Макет окна регистрации

При нажатии кнопки 2 на начальном окне открывается окно авторизации, состоящее из двух полей, трёх меток и двух кнопок (рис. 2.6). Метки 1 и 2 отвечают за описание полей 1 и 2. Поле 1 отвечает за ввод логина, поле 2 отвечает за ввод пароля. Метка 3 отвечает за системные сообщения. При нажатии Кнопки 1 происходит вход в систему, при нажатии Кнопки 2 – возврат на начальное окно.

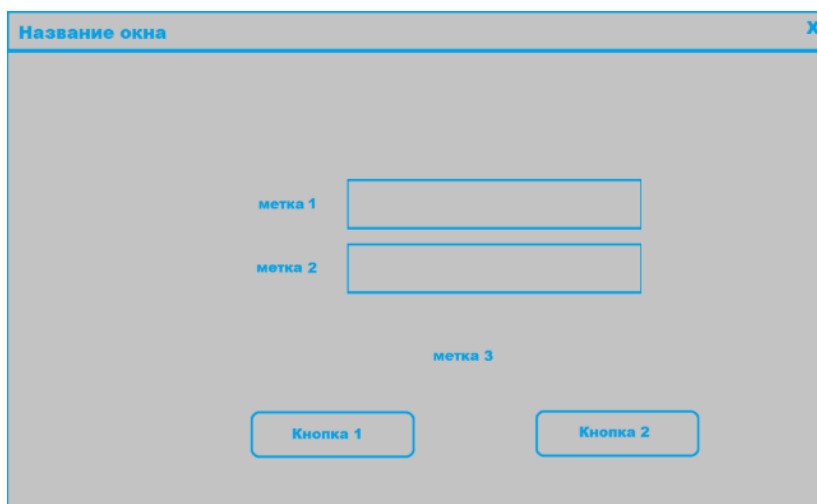


Рисунок 2.6 – Макет окна авторизации

Далее, в зависимости от роли авторизованного работника, выдаётся общее окно-маршрутизатор, на котором можно нажимать кнопки и открывать конкретные окна.

Во всех последующих макетах метка 1 отвечает за отображение логина профиля.

Администратору доступны почти все функции и окна приложения. Окно после авторизации выглядит примерно так, как на рисунке 2.7.

Примерный набор окон, которые выводятся при нажатии на кнопки:

- Кнопка 1. Просмотр списка и редактирования показаний счётчиков;
- Кнопка 2. Просмотр и редактирование списка тарифных ставок;
- Кнопка 3. Просмотр и редактирование списка должников;
- Кнопка 4. Просмотр и редактирование списка потребителей;
- Кнопка 5. Просмотр и редактирование списков счётчиков и их моделей;
- Кнопка 6. Просмотр списка документов;
- Кнопка 7. Просмотр и редактирование профилей сотрудников, ролей;

- Кнопка 8. Выход из профиля и системы. Возврат на начальное окно.

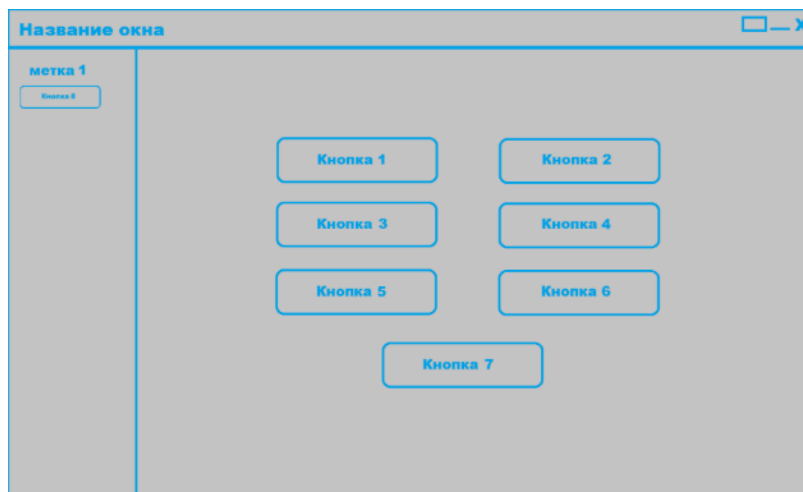


Рисунок 2.7 – Примерный макет окна, доступный всем после авторизации

В зависимости от доступных привилегий макеты открываемых окон будут разными, конкретизированные готовые окна будут созданы во время разработки приложения. Рассмотрим общие случаи макетов.

Окно просмотра и редактирования показаний счётчиков подразумевает собой таблицу с данными, а также поля с метками, данные которых используются при обновлении и добавлении показания (рис. 2.8). Для большинства сотрудников возможно лишь посмотреть таблицу и посмотреть отчёт, соответственно некоторые элементы становятся скрытыми и неактивными. Кнопка 1 отвечает за возврат на окно с основным функционалом, Кнопка 2 – за добавление, Кнопка 3 – за обновление, Кнопка 4 – за удаление, а Кнопка 5 – за просмотр отчета, созданного процедурой. Заголовки 1 и 2 – наименования атрибутов, метка 2,3,4 отвечают за описание элементов 1,2,3. Элемент 1 – поле выбора даты, элемент 2 – поле выбора чисел с плавающей запятой, элемент 3 – текстовое поле, отвечающее за ввод кода счётчика. Все элементы, отвечающие за обработку данных для большинства сотрудников скрытаны и неактивны.

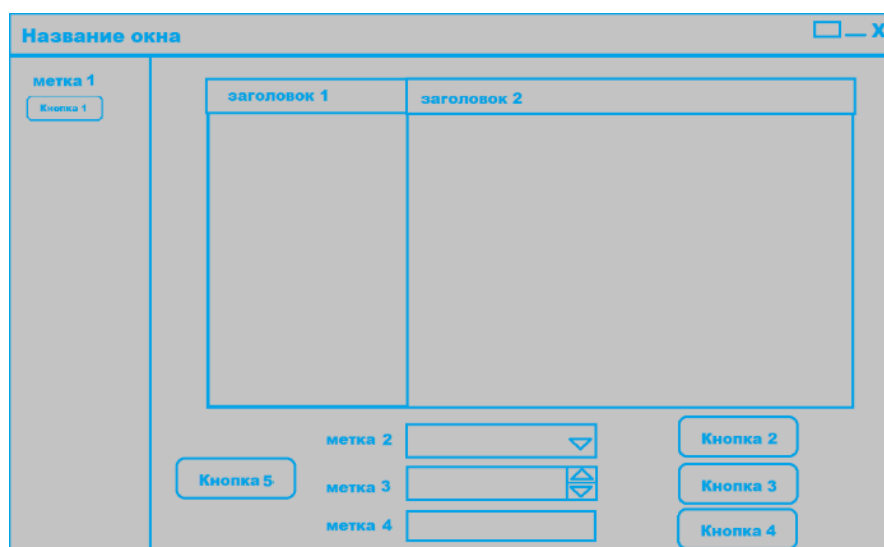


Рисунок 2.8 – Макет окна для просмотра списка показаний счетчиков

Следующий макет - для окна просмотра и редактирования тарифных ставок (рис. 2.9). Заголовки 1,2,3,4 отвечают за названия атрибутов таблицы, Кнопки 2,3,4 отвечают за добавление, обновление и удаление соответственно, метки 2,3,4,5,6 описывают поля 1,2,3,4,5. Поля 1,2,3 нужны для ввода чисел с плавающей запятой, то есть значений ставок, поле 4 нужно для выбора кода ставки, поле 5 - для наименования ставки.

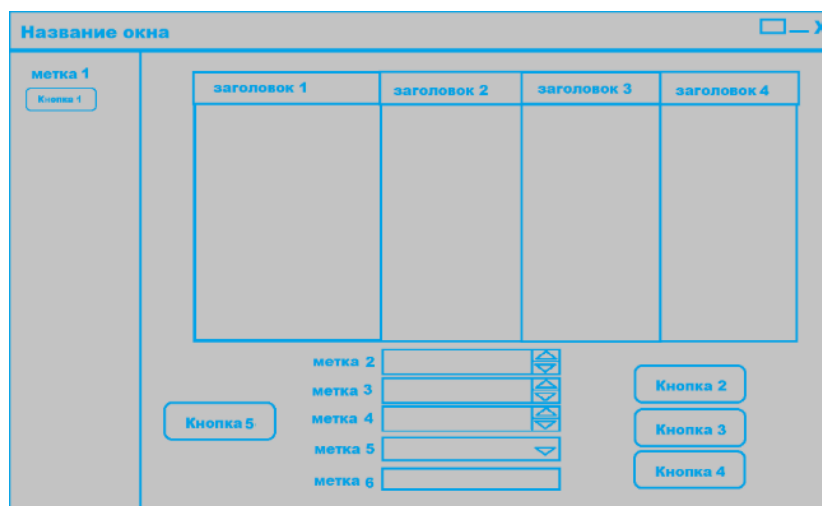


Рисунок 2.9 Макет окна для просмотра списка тарифных ставок

Следом идёт макет окна для просмотра и редактирования списка должников (рис. 2.10). Здесь список будет формироваться процедурой или представлением, а значит поля, их количество, и кнопки будут зависеть от того, какие параметры использованы в процедуре или представлении. Но скорее всего кнопки останутся

прежними, как для предыдущего макета. Метки x, y, z отвечают за разные количества полей для набора чисел с плавающей запятой, полей выбора даты и текстовых полей.

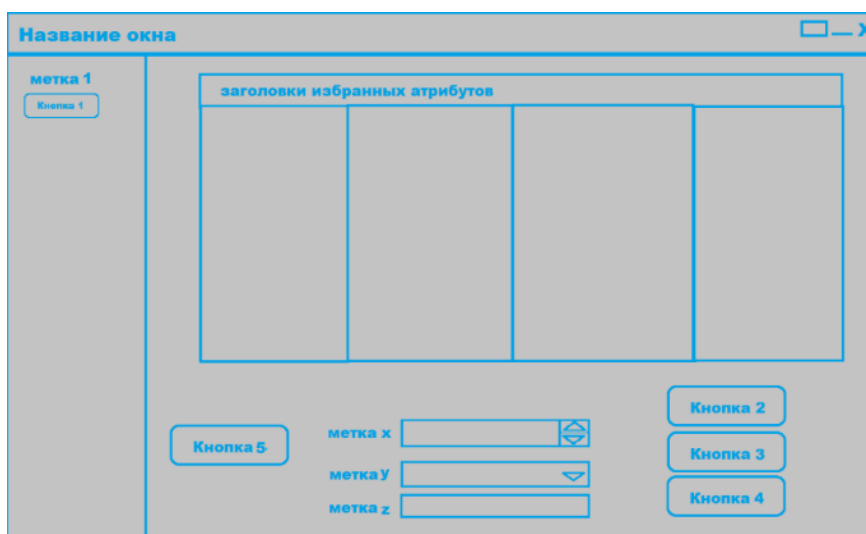


Рисунок 2.10 – Макет окна для просмотра и редактирования списка должников

Далее идёт макет для просмотра и редактирования списка потребителей (рис. 2.11). Однако по сравнению с предыдущим макетом, этот отличается определёнными атрибутами в таблице, когда в прошлом макете в таблице был микс атрибутов сущностей «Потребители» и «Должники». Наличие элементов такое же, так как довольно трудно разместить их все на макете.

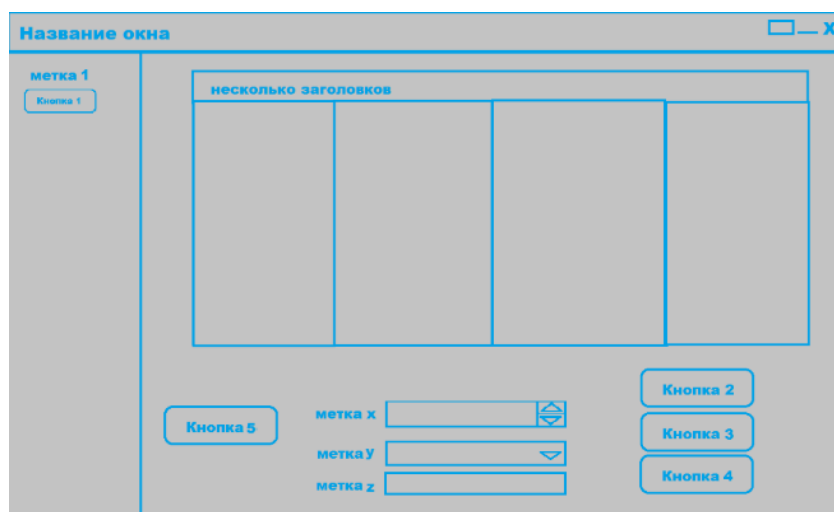


Рисунок 2.11 – Макет окна для просмотра и редактирования списка потребителей

Затем идёт макет окна для просмотра и редактирования списка счётчиков и их моделей (рис. 2.12). Здесь даны две связанные отношением таблицы, позволяющие при выборе одной записи в первой таблице, посмотреть записи, относящиеся к выбранной во второй таблице.

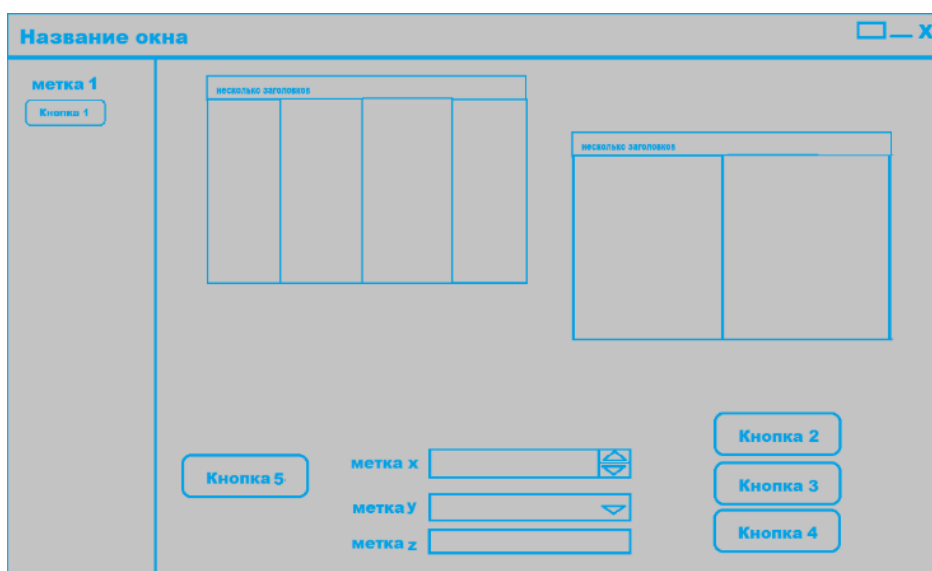


Рисунок 2.12 – Макет окна для просмотра и редактирования связанных таблиц

Потом идёт макет для просмотра и редактирования документов (рис. 2.13).

Здесь дана таблица, некоторые поля для заполнения, кнопки для удаления, добавления и обновления данных. А также кнопка 5, отвечающая за экспорт выбранного документа в Word.

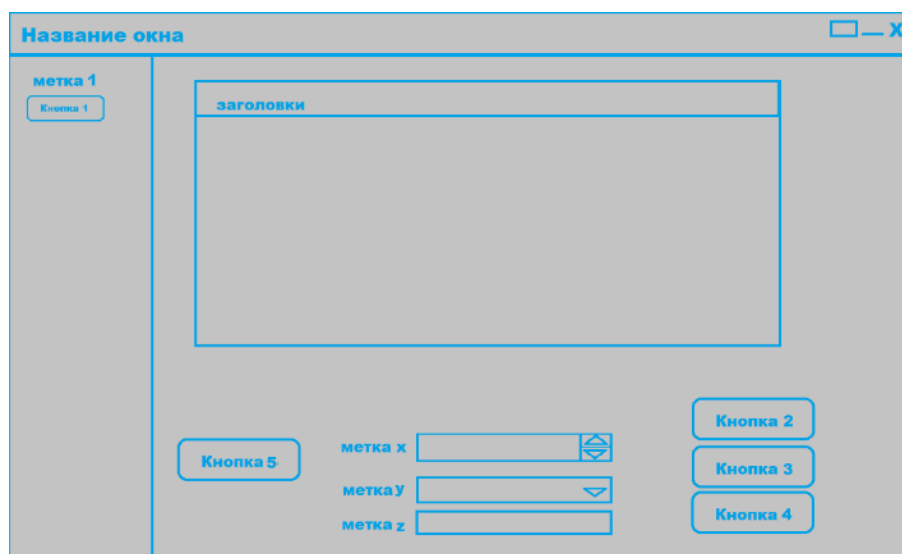


Рисунок 2.13 – Макет окна для просмотра и редактирования документов

Последний макет окна связан с просмотром и редактированием профилей сотрудников (рис. 2.14). Макет содержит такие же элементы, что и макет окна для просмотра и редактирования списка счётчиков и их моделей. Только таблицы «Профили» и «Роли» связаны между собой, то есть по выбранной роли можно просмотреть список профилей.

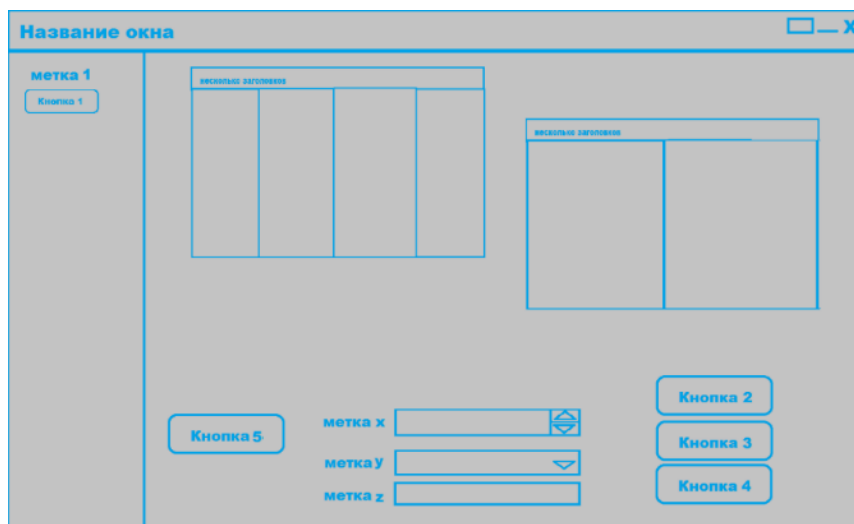


Рисунок 2.14 – Макет окна для просмотра и редактирования профилей сотрудников

После того, как были созданы макеты, необходимо выбрать СУБД и язык программирования для реализации проекта.

2.3 Выбор технологий разработки приложения

Учитывая все достоинства и недостатки, рассмотренные при анализе технологий разработки базы данных и клиентский частей, можно приступить к подбору конкретных для разработки приложения технологий.

Для разработки реляционной базы данных, как разъяснено ранее, подойдут СУБД Microsoft SQL Server, MySQL и PostgreSQL. Основные критерии подбора – быстрота обработки данных, надёжность, интуитивно понятное администрирование базы данных и приоритетное направление использования СУБД. Зная, что приложение разрабатывается именно для десктопа, сразу исключаем MySQL из этого списка, так как MySQL используется в основном для веб-приложений. PostgreSQL надёжная, часто используемая в разработке десктопных приложений. Однако она медленнее обрабатывает запросы в базе данных, а документации намного меньше по её использованию, по сравнению с Microsoft SQL Server и MySQL. Таким образом, Microsoft SQL Server будет использована в качестве инструмента разработки базы данных для приложения.

Насчет выбора языка программирования и используемых технологий. Здесь также остановились на выборе из трёх языков: Python, C# и Java.

Несмотря на быстроту разработки на Python, я бы не сказал, что работа по созданию GUI быстрее, по сравнению с C# и технологиями .NET Framework. И в Java, и в Python приходится импортировать библиотеки, а затем большую часть форм создавать руками. Технологии .NET Framework позволяют решить и эту проблему, так как при создании проекта необходимые пространства имён уже вшиты в шаблон. А элементы форм располагаются на панели инструментов, их легко перетаскивать и настраивать. Среди рассмотренных при анализе технологий не подходят для разработки Xamarin, так как он используется для разработки мобильных приложений, и ASP.NET, так как технология нацелена именно на веб-разработку.

Среди оставшихся двух технологий WPF занимает лидирующее положение, так как она производительнее, позволяет создавать GUI как для десктопных, так и для веб-приложений и отвечает современным стандартам разработки. Несмотря на все преимущества перед Windows Forms, я буду использовать вторую технологию, так как она проверена временем, документация по ней намного понятнее, при этом в WPF необходимо понимать и решать все проблемы, возникающие при разработке XAML-части.

Таким образом, для разработки приложения были выбраны СУБД Microsoft SQL Server, язык программирования C# и технология Windows Forms.

3 РАЗРАБОТКА ПРИЛОЖЕНИЯ

3.1 Разработка базы данных

Разработка базы данных происходит в программе SQL Server Management Studio. В результате проектирования появились сущности и атрибуты, однако их тип и параметры не были указаны на стадии проектирования, но в зависимости от используемой СУБД системные типы данных могут быть разными. Именование сущностей и атрибутов будет приведено к английскому языку.

Для сущности «Потребители» (Consumers) атрибуты: номер (int, PK, NOT NULL), имя, фамилия, отчество (varchar(50) NOT NULL), дата рождения (date NOT NULL), улица проживания (varchar(100) NOT NULL), номер дома (varchar(10) NOT NULL), номер квартиры (int), наличие льгот (bit), код ставки (int, FK NOT NULL), код льготы (int, FK NOT NULL), код счётчика (int, FK NOT NULL).

Сущность «Должники» (Debtors) содержит следующие атрибуты: номер должника (int, PK, NOT NULL), сумма долга (decimal(18,2) NOT NULL), дата образования долга и фактическая дата погашения (date NOT NULL), код статуса долга (int, FK NOT NULL).

Атрибуты «Статус долга» (States): код (int, PK NOT NULL) и наименование (varchar(20) NOT NULL).

Атрибуты сущности «Тарифные ставки» (Rates) таковы: номер ставки (int, PK NOT NULL), тип ставки (varchar(50) NOT NULL), ставка 1, ставка 2, ставка 3 (decimal(6,4) NOT NULL).

Сущности «Льготы» (Benefits) соответствуют следующие атрибуты: номер (int, PK NOT NULL), наименование (varchar(150) NOT NULL) и описание (varchar(500)).

Следующая сущность «Счётчики» (Counters) содержит в себе такие атрибуты, как: код (int, PK NOT NULL), серийный номер (varchar(12) NOT NULL), номер типа счётчика (int, FK NOT NULL), номер потребителя (int, FK NOT NULL).

«Тип счётчика» (Models) содержит такие атрибуты: номер типа счётчика (int, PK NOT NULL) и наименование (varchar(35) NOT NULL).

«Шапки документов» (Doc_headers) задаются атрибутами код (int, PK NOT NULL), наименование (varchar(75) NOT NULL), образец (varchar(500) NOT NULL) и основное содержание (varchar(MAX) NOT NULL).

«Документы» (Documents) содержат следующие атрибуты: номер документа (int, PK NOT NULL), дата подписания (date NOT NULL), номер шапки (int, FK NOT NULL) и номер потребителя (int, FK NOT NULL).

Последняя сущность – «Показания счётчиков» (Indications) - определена следующими атрибутами: код счётчика (int, PK FK NOT NULL), дата снятия показания (date NOT NULL) и показание счётчика (decimal(18,4) NOT NULL).

При разработке базы данных было решено изменить связь между «Счетчики» и «Показания счётчиков», сделав её как один-к-одному.

Создание таблиц и их атрибутов происходит визуально (рис. 3.1).

	Имя столбца	Тип данных	Разрешить з...
►P	id_h_doc	int	<input type="checkbox"/>
	doc_name	varchar(75)	<input type="checkbox"/>
	sample	varchar(500)	<input type="checkbox"/>
	body	varchar(MAX)	<input type="checkbox"/>
			<input type="checkbox"/>

Рисунок 3.1 – Пример создания таблицы «Шапки документов»

По итогу созданы все таблицы и размещены они на ER-диаграмме (рис. 3.2). Кроме этого в базе данных, созданы ранее определённые таблицы «Роли» и «Профили» (рис. 3.3). Атрибуты сущности «Профили»: номер (int, PK NOT NULL), имя, фамилия, отчество, электронная почта (varchar(50) NOT NULL), логин (varchar(18) NOT NULL), хэш пароля (varchar(200) NOT NULL), соль (varchar(15) NOT NULL), код роли (int, FK), статус профиля (bit NOT NULL). При регистрации пользователю автоматически присваивается активный статус. А атрибуты сущности «Роли» такие: код (int, PK NOT NULL), наименование (varchar(25)).

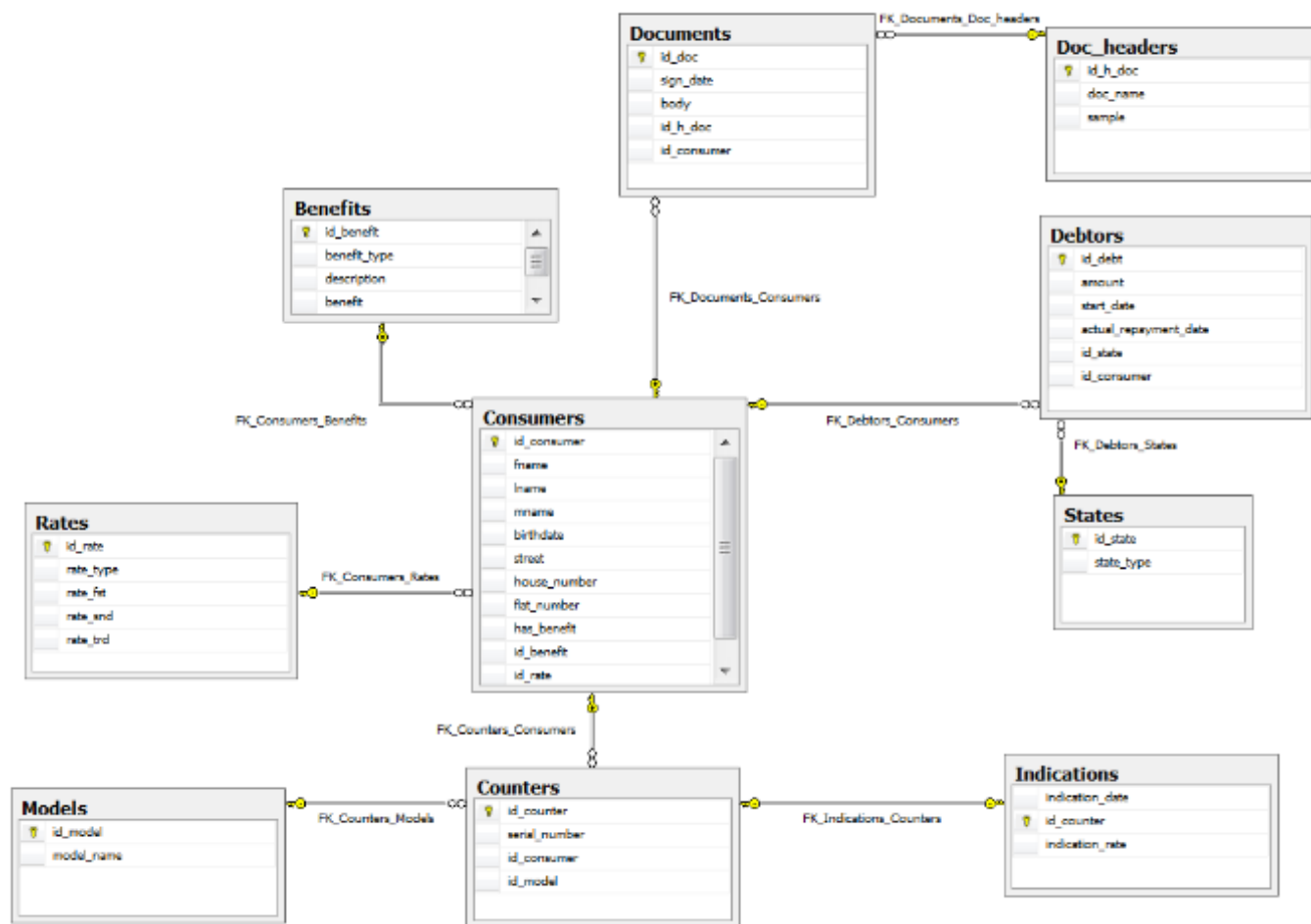


Рисунок 3.2 – ER-диаграмма основных таблиц базы данных

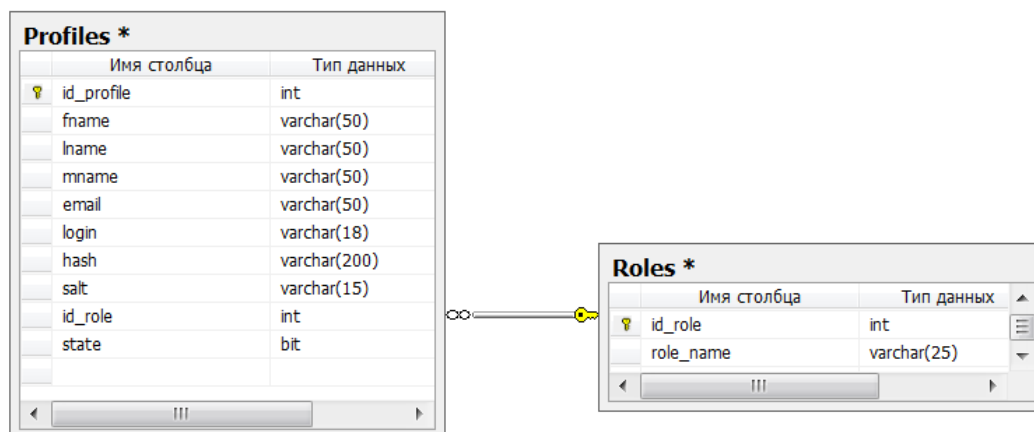


Рисунок 3.3 – Дополнительная ER-диаграмма для профилей и ролей

Если учитывать связи между таблицами, то заполнение данных ведётся сначала в таблицах Rates (Тарифные ставки), States (Статус долга), Models (Тип счётчика), Benefits (Льготы). Таблицы заполняются визуально.

Для вставки данных в следующие таблицы преимущественно используются сгенерированные строки запросов.

Таблица Doc_headers заполнена следующим образом (рис. 3.4). Имеется минимум два типа договоров: о купле-продажи электроэнергии и об оказании электромонтажных работ.

	id_h_doc	doc_name	sample	body
1	9	Договор о купле-продажи электроэнергии	АО "Электросбыт" Юридическ...	1.1. В соответствии с настоящим Договором Гарант...
2	17	Договор об оказании электромонтажных услуг	АО "Электросбыт" Юридическ...	1.1. В соответствии с настоящим Договором Гарант...

Рисунок 3.4 – Заполненная таблица Doc_headers

Определим скалярные функции в базе данных. Первая функция будет находить адрес дома, в котором установлен определённый счётчик, то есть поиск ведётся по серийному номеру счётчика (рис. 3.5).

```
DECLARE @address varchar(100)
SELECT @address=(cast(Consumers.street as varchar)+' '+cast(Consumers.house_number as varchar)+' '
+cast(Consumers.flat_number as varchar))
FROM Consumers INNER JOIN Counters ON Counters.id_consumer = Consumers.id_consumer
WHERE Counters.serial_number=@serial
RETURN @address
```

Рисунок 3.5 – Тело скалярной функции GetAddress

Вторая функция позволит определить общую сумму задолженности перед организацией (рис. 3.6).

```
CREATE FUNCTION [dbo].[TotalDebtAmount] ()
RETURNS decimal(18,2)
AS
BEGIN
    DECLARE @total decimal(18,2)
    SELECT @total=SUM(amount)
    FROM Debtors
    RETURN @total
END
```

Рисунок 3.6– Создание функции TotalDebtAmount

Определим функцию, возвращающую готовые документы, параметром которого станет номер документа (рис. 3.7).

```
CREATE FUNCTION GetDocuments(@docNum int)
RETURNS TABLE
AS RETURN(
    SELECT id_doc as 'Номер', doc_name as 'Имя документа', sample as 'Шанка', fname as 'Имя потребителя',
    lname as 'Фамилия потребителя', mname as 'Отчество потребителя', body as 'Тело документа',
    sign_date as 'Дата подписания'
    FROM Documents INNER JOIN Doc_headers ON Documents.id_h_doc=Doc_headers.id_h_doc
    INNER JOIN Consumers ON Doc_headers.id_consumer=Consumers.id_consumer
    WHERE Documents.id_doc = @docNum
)
```

Рисунок 3.7 – Создание табличной функции GetDocuments

Результат выполнения для номера 3 на рисунке 3.8.

```
select * from GetDocuments(3)
```

Результаты								
Номер	Имя документа	Шапка	Имя потребите...	Фамилия потребит...	Отчество потребит...	Тело докум...	Дата подписан...	
1	3	Договор о купле...	АО "Элек...	Илья	Смирнов	Сергеевич	1.1. В соотв...	2022-03-04

Рисунок 3.8 – Выполнение табличной функции

Определим некоторые процедуры, выводящие табличное значение, а также процедуры, используемые для облегчения удаления, обновления и вставки данных.

Первая процедура покажет актуальный список должников (рис. 3.9). Результат выполнения на рисунке 3.10.

```
CREATE PROCEDURE [dbo].[Actual Debtors]
AS
BEGIN
    SELECT lname, fname, mname, amount, start_date, state_type from Debtors
    INNER JOIN States ON Debtors.id_state=States.id_state
    INNER JOIN Consumers ON Debtors.id_consumer=Consumers.id_consumer
    WHERE Debtors.id_state in (1,2)
END
```

Рисунок 3.9 – Создание процедуры вывода актуального списка должников

	lname	fname	mname	amount	start_date	state_type
1	Ильин	Фёдор	Михайлович	2501.34	2022-05-10	Не погашен
2	Сергеев	Фёдор	Михайлович	2438.15	2022-05-10	Не погашен
3	Сергеев	Никита	Сергеевич	2756.34	2022-05-10	Частично погашен

Рисунок 3.10 – Вывод актуального списка должников

Следующая процедура покажет список 15 максимальных показаний счётчиков (рис. 3.11).

```
CREATE PROCEDURE [dbo].[15_Max_Indications]
AS
SET ROWCOUNT 15
BEGIN
    SELECT indication_date, indication_rate, serial_number, lname, fname, mname
    from Indications INNER JOIN Counters ON Indications.id_counter=Counters.id_counter
    INNER JOIN Consumers ON Counters.id_consumer=Consumers.id_consumer
    Order by indication_rate DESC
END
```

Рисунок 3.11 – Создание процедуры вывода списка 15 показаний счётчиков

Результат выполнения процедуры на рисунке 3.12.

	indication_d...	indication_r...	serial_number	lname	fname	mname
1	2022-05-31	273.997700	CO2F6DTXPLU	Зеленин	Илья	Дмитриевич
2	2022-05-31	272.768900	COG9YZHOPWC	Комаров	Виктор	Михайлович
3	2022-05-31	272.351300	COC9N4AEB9I	Лавров	Виктор	Дмитриевич
4	2022-05-31	268.255100	COV9QA5KTL1	Комаров	Никита	Дмитриевич
5	2022-05-31	267.446900	COHB03D6JOW	Сидоров	Сергей	Иванович
6	2022-05-31	264.882400	COFSI9QOBAV	Иванов	Никита	Михайлович
7	2022-05-31	264.348800	COZ4UK0YKPI	Никитин	Иван	Петрович
8	2022-05-31	262.481300	CO039SF9TMM	Сергеев	Илья	Павлович
9	2022-05-31	257.680800	COWBC81AY2I	Никитин	Василий	Анатольевич
10	2022-05-31	257.431200	CONDDXRVP LI	Петров	Андрей	Сергеевич
11	2022-05-31	254.958200	CO2XOJ08M3E	Смирнов	Владимир	Дмитриевич
12	2022-05-31	249.378600	COIWXCV8PXY	Зеленин	Никита	Иванович
13	2022-05-31	248.582900	CO9T2M9YF6I	Никитин	Константин	Михайлович
14	2022-05-31	245.290600	CO3TDNJPSN8	Иванов	Дмитрий	Павлович

Рисунок 3.12 – Вывод списка 15 максимальных показаний счётчиков

Последняя процедура по выводу списка должна выводить суммы стоимостных выражений по показаниям счётчиков по каждой ставке тарифа (рис. 3.13)

```
SELECT Consumers.id_rate, rate_type,
CASE WHEN Consumers.id_rate=7 THEN SUM(indication_rate*Rates.rate_fst)
WHEN Consumers.id_rate=8 THEN SUM(indication_rate*(16*Rates.rate_fst + 8*Rates.rate_snd)/24)
ELSE SUM(indication_rate*(7*rate_fst + 9*Rates.rate_snd + 8*Rates.rate_trd)/24) end as total
from Indications
INNER JOIN Counters ON Counters.id_counter=Indications.id_counter
INNER JOIN Consumers ON Consumers.id_consumer=Counters.id_consumer
INNER JOIN Rates ON Rates.id_rate=Consumers.id_rate
GROUP BY Consumers.id_rate, rate_type
```

Рисунок 3.13 – Тело процедуры по выводу сумм по тарифной ставке

Результат выполнения представлен на рисунке 3.14.

	id_rate	rate_type	total
1	7	Городское население	23189.7813840000
2	8	Городское население с электроплитами	10779.8002706667
3	9	Приравненные к населению	13403.6291214583

Рисунок 3.14 – Выполненная процедура по выводу сумм по тарифной ставке

Последней процедурой станет та, что удаляет из списка должников те позиции, которые заданы статусом «Погашен» (рис. 3.15)

```
CREATE PROCEDURE DeleteRepayment
AS
BEGIN
Delete from Debtors
where id_state = 3
END
GO
```

Рисунок 3.15 – Создание процедуры по удалению должников, погасивших долг

На рисунке 3.16 представлен результат выполнения процедуры. До выполнения процедуры в таблице были должники с кодами 1,3 и 5.

	id_debt	amount	start_date	actual_repayment_d...	id_st...	id_consumer
1	2	2501.34	2022-05-10	NULL	1	4
2	4	2438.15	2022-05-10	NULL	1	13
3	6	2756.34	2022-05-10	NULL	2	17

Рисунок 3.16 – Результат использования процедуры по удалению должников

Полный код создания базы данных и вставки данных в таблицы доступен для просмотра в Приложении 1.

Данного объёма функций, процедур достаточно для описания. Таким образом, можно приступать к разработке клиентских частей приложения.

3.2 Разработка клиентских частей

Начальная форма – приветственная. Здесь можно либо авторизоваться, либо зарегистрироваться (рис. 3.17). Соответственно этим кнопкам прописан код открытия форм.

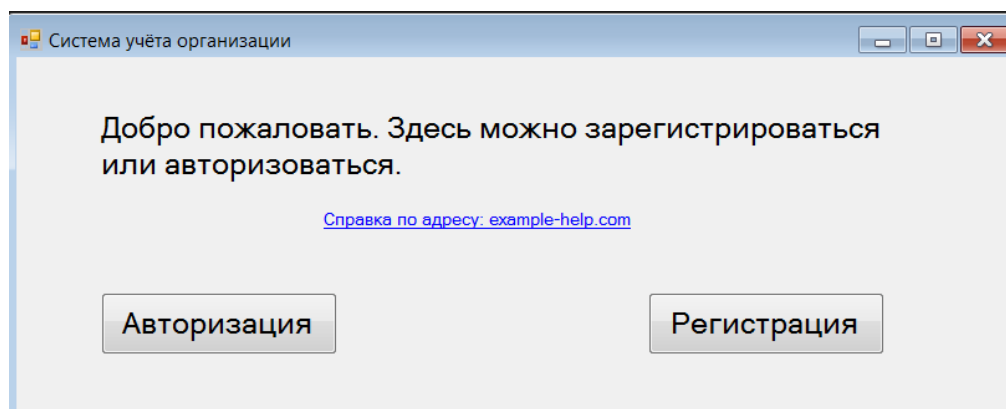


Рисунок 3.17 – Приветственная форма приложения

Изначально в базе предусмотрена единственная запись администратора, поэтому следующей отправной точкой является форма регистрации (рис. 3.18).

Рисунок 3.18– Регистрационная форма

Здесь реализован собственный класс SHACrypt, в котором расположен конструктор без параметров и два метода: один метод SHA512M отвечает за создание контрольной суммы из строки, полученной конкатенацией входных пароля и «соли», второй метод используется как раз для создания «соли» длиной в 15 символов.

Непосредственно на форме регистрации реализована проверка валидности полей, за исключением поля e-mail. Изначально по параметрическому запросу проверяется наличие профилей с одинаковой почтой или одинаковым логином, если записей нет, проводится проверка на правильность ввода данных. Есть проверка на отсутствие данных и проверка на длину пароля или логина. После выполнения условий выполняется запрос на вставку данных. Если введены совпадающие данные, то кнопка становится неактивной до тех пор, пока значения в полях не изменены.

После успешной регистрации можно вернуться назад в начальное окно и нажать на кнопку «Авторизация». Выдаётся обычное окно с вводом логина и пароля (рис. 3.19)

В случае неправильно введённого пароля или логина выведется сообщение в label1.

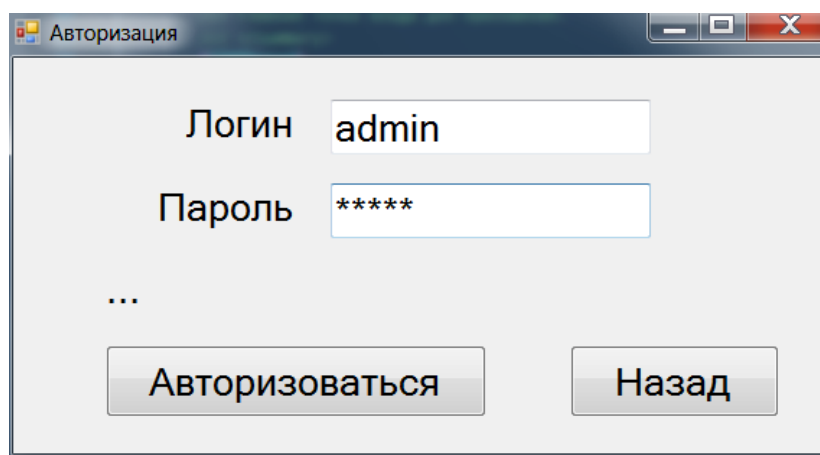


Рисунок 3.19 – Окно авторизации

После успешной авторизации данное окно закрывается, а начальное окно скрывается. Открывается главное окно, где наличие элементов управления зависит от роли профиля (рис. 3.20).

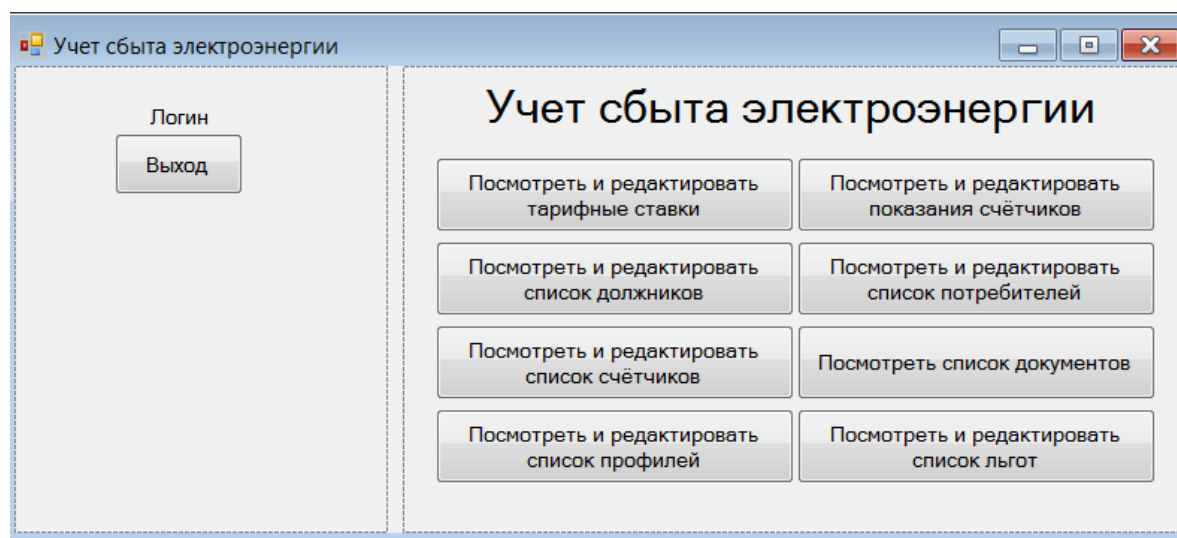


Рисунок 3.20 – Главное окно управления для администратора

Слева располагаются логин и кнопка «Выход». Справа расположены элементы управления. Ведётся построенный запрос по проверке названия роли, соотнесенной к логину профиля. В зависимости от роли профиля некоторые кнопки имеют другое название, некоторые просто не активны.

Далее будут рассмотрены некоторые формы без привязки к роли. Первая форма по просмотру и редактированию тарифных ставок (рис 3.21). В зависимости от роли можно просматривать, добавлять, удалять и обновлять значения ставок. Также имеется просмотр списка, сформированного по хранимой процедуре.

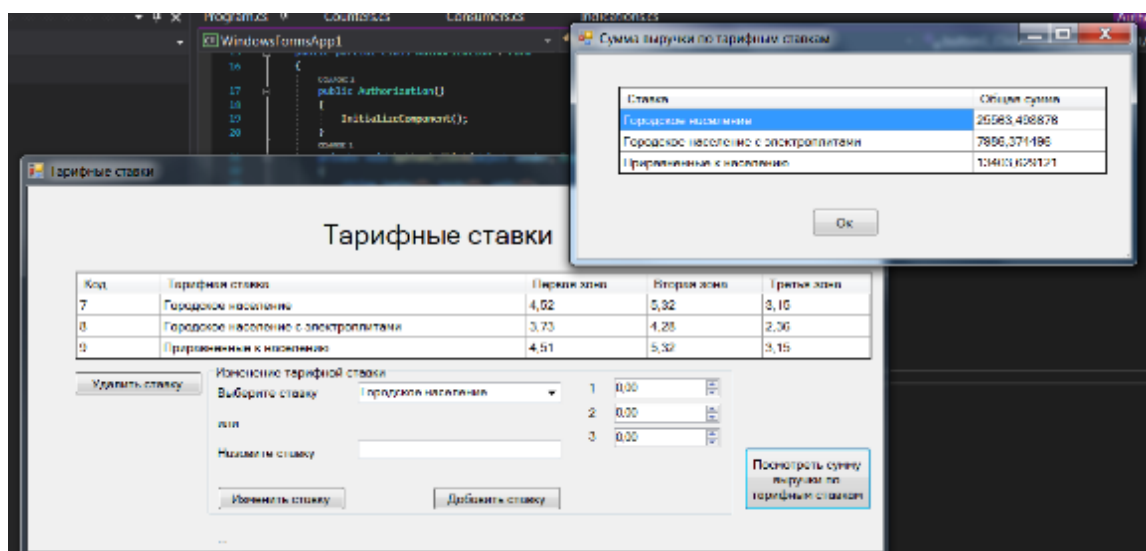


Рисунок 3.21 – Просмотр и редактирование тарифных ставок

Следующая форма – окно просмотра и редактирования списка потребителей (рис. 3.22). Здесь можно как добавлять, так и обновлять данные. Также есть возможность искать потребителя по определённым параметрам.

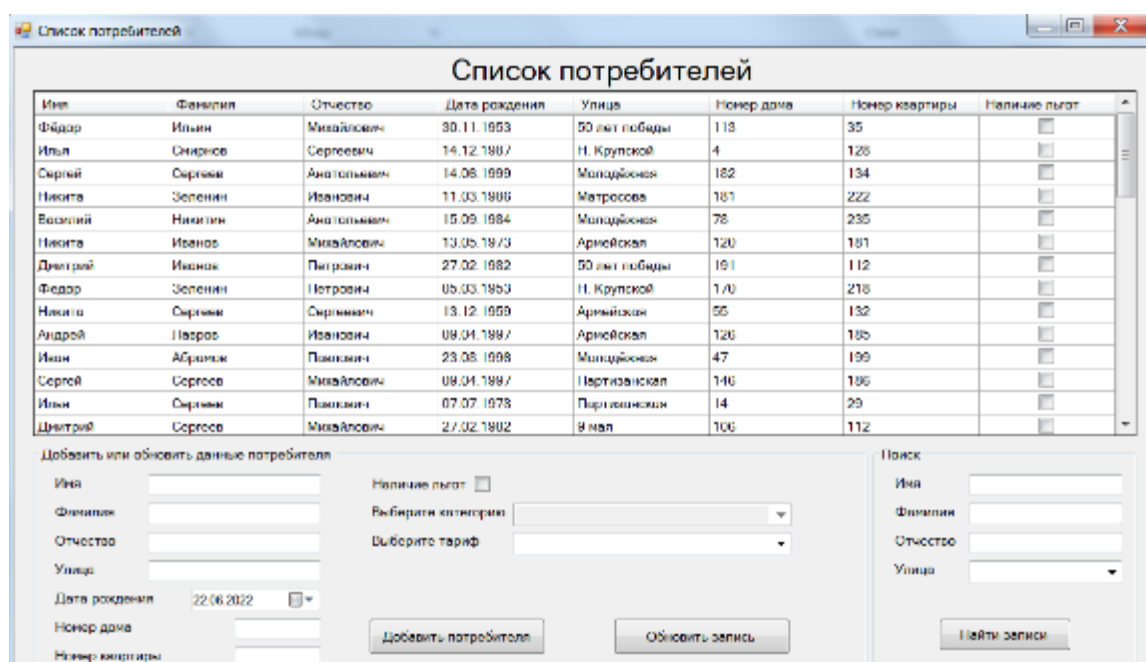


Рисунок 3.22 – Просмотр и редактирование списка потребителей

На следующей форме можно по серийному номеру вывести информацию о потребителе и показании счётчика, который ему принадлежит (рис 3.23). Предусмотрен поиск по серийному номеру счётчика. Также возможен вывод результата хранимой процедуры в отдельном окне.

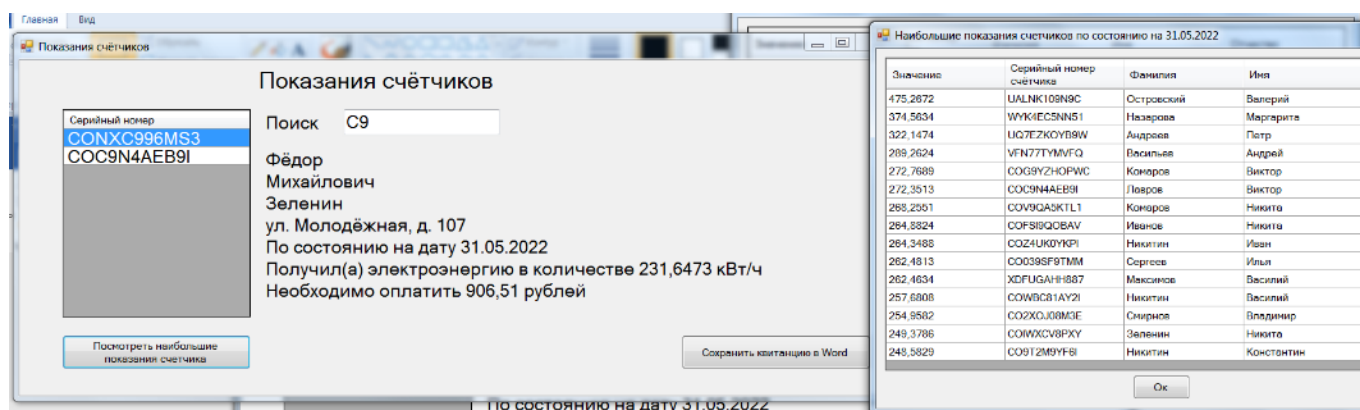


Рисунок 3.23 – Просмотр показаний счётчиков и отчёта по показаниям

Есть возможность сохранить квитанцию в документ Word, пример документа представлен на рисунке 3.24. Для работы с документами была использована ссылка на пространство имён Microsoft.Office.Interop.Word. В зависимости от того, указан ли номер дома или есть ли льгота, выводятся разные значения в таблице.

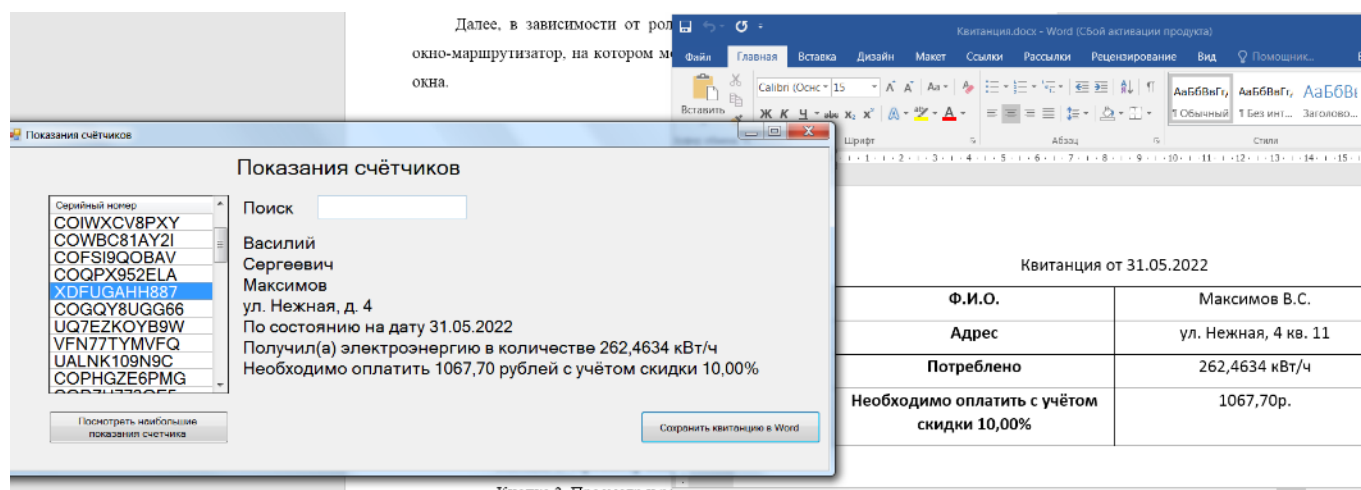


Рисунок 3.24 – Сохранение квитанции в документ Word

Также в приложении реализованы форма редактирования списка льгот (рис. 3.25), форма редактирования профилей (рис. 3.26), форма просмотра и редактирования списка должников (рис. 3.27).

Наименование	Описание	Процент скидки
Ветеран гос. службы		0,10
Ветеран труда		0,25
Ветеран ВОВ		0,25
Имущей неприкосновенности		0,20
Гражданам, подвергшимся ра...	Семипалатинск, Чернобыль	0,20
Гражданам, подвергшимся ра...	В зоне скопления	0,20
Пенсионер	Пенсионеры по потере корм...	0,05

Изменить льготу

Выбор льгот: Ветеран гос. службы

или

Назначить льготу: Пенсионер

Скидка: 0,05

Описание: Пенсионеры по потере кормильца, пенсионеры по возрасту

Изменить льготу Добавить льготу Удалить льготу

Рисунок 3.25 – Форма редактирования льгот

Имя	Фамилия	Отчество	Эл. почта	Логин	Хеш пароля	Соль	Статус профиля	Роль
Дмитрий	Бычков	Сергеевич	bychkov@...ru	by	008c88d632...	byruTn62IO...	<input checked="" type="checkbox"/>	Администратор

Изменить профиль

Пароль:

Активировать профиль: ☐

Назначить роль: Администратор

Изменить профиль Удалить профиль

Изменить пароль

Имя: рий

Фамилия:

Отчество:

Логин: *

Найти

Рисунок 3.26 – Форма редактирования профилей

Имя	Фамилия	Отчество	Улица	Номер дома	Номер квартиры
Илья	Ильин	Михайлович	50 лет победы	113	35
Илья	Орлов	Сергеевич	Н. Крупской	4	128
Сергей	Сергеев	Анатолиевич	Молдажонки	182	134
Никита	Зеленин	Иванович	Молдажонки	181	222
Василий	Никитин	Анатолиевич	Молдажонки	78	235
Никита	Иванов	Михайлович	Армейская	120	181
Дмитрий	Иванов	Петрович	50 лет победы	181	112
Федор	Зеленин	Петрович	Н. Крупской	170	218
Никита	Сергеев	Сергеевич	Армейская	55	132

Сумма долга: 2501,34

Дата образования долга: 10.05.2022

Дата погашения:

Статус долга: Не погашен

Показать актуальный список должников Удалить погашенные долги Общая сумма задолженности перед организацией: 6297,68

Нет больше записей для удаления

Актуальный список должников

Фамилия	Имя	Отчество	Сумма долга	Дата образования долга	Состояние
Ильин	Федор	Михайлович	2501,34	10.05.2022	Не погашен
Сергеев	Никита	Сергеевич	2756,34	10.05.2022	Частично погашен

Ок

Рисунок 3.27 – Форма редактирования списка должников

Последняя форма – просмотра списка документов и вывода документа в Word – представлена на рисунке 3.28.

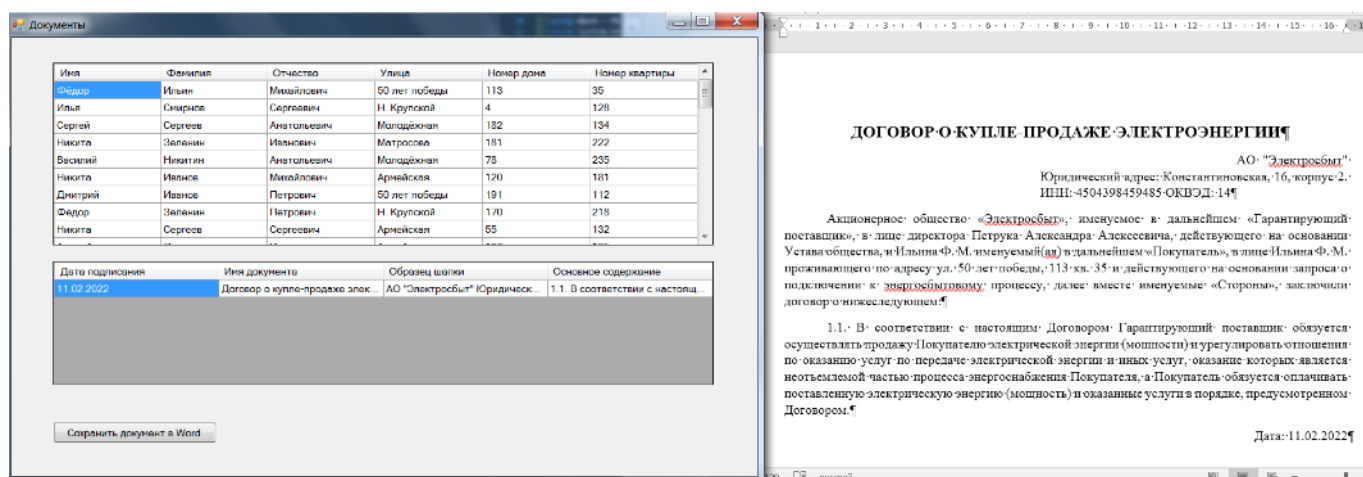


Рисунок 3.28 – Форма редактирования списка должников

Весь код клиентских частей и собственного класса SHACrypt представлен в Приложении 2.

Суммируя все вышеизложенное, а именно анализ технологий разработки приложения, сравнительный анализ готовых программные продукты, и проектирование собственного приложения на основе выбранных технологий, можно сказать, что приложение готово к реализации и эксплуатации в организации.

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы были проведены:

- анализ технологий разработки приложений;
- анализ готовых программных продуктов;
- проектирование и разработка приложения для энергосбытовой организации.

Для разработки приложения были выбраны: СУБД Microsoft SQL Server, язык программирования C# и технология Windows Forms. Приложение разработано в соответствии с результатами, полученными в ходе проектирования и сравнительного анализа готовых программных решений. Оно предназначено как для просмотра и редактирования отдельных таблиц, так и для просмотра статистических данных, сводных отчетов, экспорта документов. Прилагается инструкция по использованию программного продукта.

Достоинствами разработанного продукта являются: интуитивно понятный интерфейс, разграничение доступа к отдельным клиентским частям и низкий уровень затрат по разработке и распространению программного продукта внутри организации.

Для того, чтобы использовать разработанное приложение не только по отношению к физическим лицам, но и юридическим, необходимы доработка существующей базы данных или создание новой и усовершенствование клиентских частей. А для удобства обращения к организации возможно создание мобильного приложения для самостоятельной фиксации показаний счётчиков и оплаты услуг напрямую.

Но учитывая то, что при проектировании базы данных было определено, что она охватывает лишь физические лица, можно сказать, что разработанные база данных и клиентские части соответствуют тематике курсовой работы.

Данное приложение было успешно протестировано и готово к использованию. Значит, цель курсовой работы была достигнута.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ

1. Федеральный закон от 27.07.2006 N 152-ФЗ. (ред. от 02.07.2021) «О персональных данных»
2. Осипов, Д.Л. Технологии проектирования баз данных [Электронный ресурс] / Д.Л. Осипов - М.: ДМК Пресс, 2019. – 498 с.: ил. - Режим доступа: <https://vk.cc/cebAp7> (Дата обращения: 31.04.2022)
3. Кузнецов, М. Модели баз данных - шпаргалки для начинающих [Электронный ресурс]: курсы по программированию. / М. Кузнецов - Режим доступа: <https://www.internet-technologies.ru/articles/modeli-baz-dannyh-sistemy-upravleniya-bazami-dannyh.html> (Дата обращения: 01.05.2022)
4. Введение в MS SQL Server и T-SQL [Электронный ресурс]: сайт о программировании Metanit. - Режим доступа: <https://metanit.com/sql/sqlserver/1.1.php> (Дата обращения: 01.05.2022)
5. СУБД MySQL [Электронный ресурс]: ресурс компании «Метод Лаб». - Режим доступа: <https://www.methodlab.ru/technology/mysql.shtml> (Дата обращения: 01.05.2022)
6. Сидоренко, Д.Е. Современные и популярные СУБД [Электронный ресурс]: ресурс студенческого научного форума. / Д.Е. Сидоренко - Режим доступа: <https://scienceforum.ru/2019/article/2018012172> (Дата обращения: 01.05.2022)
7. Что такое объектно-ориентированное программирование (ООП)? [Электронный ресурс]: ресурс кадрового агентства «IT and Digital». - Режим доступа: <https://itanddigital.ru/oop> (Дата обращения: 03.05.2022)
8. Популярные языки программирования в 2022 году: ТОП-6 [Электронный ресурс]: ресурс онлайн-школы программирования «itProger». - Режим доступа: <https://itproger.com/news/populyarnie-yaziki-programmirovaniya-v-2022-godu-top-6> (Дата обращения: 03.05.2022)
9. Лутц, М. Программирование на Python [Электронный ресурс] / М. Лутц - Пер. с англ. - СПб.: Символ-плюс, 2011. - 992 с., ил. - Режим доступа: <https://vk.cc/cebAhx> (Дата обращения: 03.05.2022)

10. Язык программирования Java [Электронный ресурс]: «компания Web Creator». - Режим доступа: <https://web-creator.ru/articles/java> (Дата обращения: 03.05.2022)
11. Язык Си [Электронный ресурс]: ресурс о программировании. - Режим доступа: <https://prog-cpp.ru/c/> (Дата обращения: 04.05.2022)
12. Уроки C# [Электронный ресурс]: ресурс онлайн-школы программирования «itProger». - Режим доступа: <https://itproger.com/course/csharp> (Дата обращения: 05.05.2022)
13. Введение в C++ [Электронный ресурс]: сайт о программировании Metanit. - Режим доступа: <https://metanit.com/cpp/tutorial/1.1.php> (Дата обращения: 05.05.2022)
14. Флэнаган, Д. JavaScript. Подробное руководство, 6-е издание. [Электронный ресурс]. / Д. Флэнаган – Пер. с англ. – СПб.: Символ-плюс, 2012. – 1080 с., ил. – Режим доступа: <https://vk.cc/cebAdX> (Дата обращения: 05.05.2022)
15. API функции для взаимодействия с внешними системами [Электронный ресурс]: ресурс компании ООО «Технологии энергоучета». – Режим доступа: <https://yaenergetik.ru/electricgrids/> (Дата обращения: 07.05.2022)
16. 1С:ERP Энергетика 2 [Электронный ресурс]: отраслевые и специализированные решения 1С:Предприятие. – Режим доступа: https://solutions.1c.ru/catalog/erp_power_grid/features (Дата обращения: 08.05.2022)
17. Нормализация баз данных простыми словами [Электронный ресурс]: блог о компьютерах и программировании для начинающих. – Режим доступа: <https://info-comp.ru/database-normalization> (Дата обращения: 16.05.2022)

ПРИЛОЖЕНИЕ

Код создания базы данных

```

USE [master]
GO
/***** Object: Database [Energy]    Script Date: 06/23/2022 08:12:24 *****/
CREATE DATABASE [Energy] ON PRIMARY
( NAME = N'Energy', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL10_50.SQLEXPRESS\MSSQL\DATA\Energy.mdf'
, SIZE = 3072KB , MAXSIZE = UNLIMITED, FILEGROWTH = 1024KB )
LOG ON
( NAME = N'Energy_log', FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL10_50.SQLEXPRESS\MSSQL\DATA\Energy_log.ldf' , SIZE = 1024KB , MAXSIZE = 2048GB , FILEGROWTH = 10%)
GO
ALTER DATABASE [Energy] SET COMPATIBILITY_LEVEL = 100
GO
IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [Energy].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO

USE [Energy]
GO
/***** Object: Table [dbo].[Benefits]    Script Date: 06/23/2022 08:12:25 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Benefits](
    [id_benefit] [int] IDENTITY(1,1) NOT NULL,
    [benefit_type] [varchar](200) NOT NULL,
    [description] [varchar](500) NULL,
    [benefit] [decimal](3, 2) NOT NULL,
    CONSTRAINT [PK_Benefits] PRIMARY KEY CLUSTERED
(
    [id_benefit] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/***** Object: Table [dbo].[Doc_headers]    Script Date: 06/23/2022 08:12:25 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Doc_headers](
    [id_h_doc] [int] IDENTITY(1,1) NOT NULL,
    [doc_name] [varchar](75) NOT NULL,
    [sample] [varchar](500) NOT NULL,
    [body] [varchar](max) NOT NULL,
    CONSTRAINT [PK_Doc_headers] PRIMARY KEY CLUSTERED
(
    [id_h_doc] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/***** Object: Table [dbo].[Models]    Script Date: 06/23/2022 08:12:25 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO

```

```

CREATE TABLE [dbo].[Models](
    [id_model] [int] IDENTITY(1,1) NOT NULL,
    [model_name] [varchar](35) NOT NULL,
    CONSTRAINT [PK_Models] PRIMARY KEY CLUSTERED
(
    [id_model] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/***** Object: Table [dbo].[States]    Script Date: 06/23/2022 08:12:25 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[States](
    [id_state] [int] IDENTITY(1,1) NOT NULL,
    [state_type] [varchar](20) NOT NULL,
    CONSTRAINT [PK_States] PRIMARY KEY CLUSTERED
(
    [id_state] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/***** Object: Table [dbo].[Roles]    Script Date: 06/23/2022 08:12:25 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Roles](
    [id_role] [int] IDENTITY(1,1) NOT NULL,
    [role_name] [varchar](25) NOT NULL,
    CONSTRAINT [PK_Roles] PRIMARY KEY CLUSTERED
(
    [id_role] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/***** Object: Table [dbo].[Rates]    Script Date: 06/23/2022 08:12:25 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Rates](
    [id_rate] [int] IDENTITY(1,1) NOT NULL,
    [rate_type] [varchar](50) NOT NULL,
    [rate_fst] [decimal](6, 2) NOT NULL,
    [rate_snd] [decimal](6, 2) NOT NULL,
    [rate_trd] [decimal](6, 2) NOT NULL,
    CONSTRAINT [PK_Rates] PRIMARY KEY CLUSTERED
(
    [id_rate] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/***** Object: Table [dbo].[Profiles]    Script Date: 06/23/2022 08:12:25 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON

```

```

GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Profiles](
    [id_profile] [int] IDENTITY(1,1) NOT NULL,
    [fname] [varchar](50) NOT NULL,
    [lname] [varchar](50) NOT NULL,
    [mname] [varchar](50) NOT NULL,
    [email] [varchar](50) NOT NULL,
    [login] [varchar](18) NOT NULL,
    [hash] [varchar](200) NOT NULL,
    [salt] [varchar](15) NOT NULL,
    [id_role] [int] NULL,
    [state] [bit] NULL,
    CONSTRAINT [PK_Profiles] PRIMARY KEY CLUSTERED
(
    [id_profile] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/***** Object: Table [dbo].[Consumers]    Script Date: 06/23/2022 08:12:25 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Consumers](
    [id_consumer] [int] IDENTITY(1,1) NOT NULL,
    [fname] [varchar](50) NOT NULL,
    [lname] [varchar](50) NOT NULL,
    [mname] [varchar](50) NULL,
    [birthdate] [date] NOT NULL,
    [street] [varchar](100) NOT NULL,
    [house_number] [varchar](10) NOT NULL,
    [flat_number] [int] NULL,
    [has_benefit] [bit] NOT NULL,
    [id_benefit] [int] NULL,
    [id_rate] [int] NOT NULL,
    CONSTRAINT [PK_Consumers] PRIMARY KEY CLUSTERED
(
    [id_consumer] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/***** Object: Table [dbo].[Debtors]    Script Date: 06/23/2022 08:12:25 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Debtors](
    [id_debt] [int] IDENTITY(1,1) NOT NULL,
    [amount] [decimal](18, 2) NOT NULL,
    [start_date] [date] NOT NULL,
    [actual_repayment_date] [date] NULL,
    [id_state] [int] NOT NULL,
    [id_consumer] [int] NOT NULL,
    CONSTRAINT [PK_Debtors] PRIMARY KEY CLUSTERED
(
    [id_debt] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/***** Object: Table [dbo].[Counters]    Script Date: 06/23/2022 08:12:25 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON

```



```

GO
CREATE TABLE [dbo].[Counters](
    [id_counter] [int] IDENTITY(1,1) NOT NULL,
    [serial_number] [varchar](12) NOT NULL,
    [id_consumer] [int] NOT NULL,
    [id_model] [int] NOT NULL,
    CONSTRAINT [PK_Counters_1] PRIMARY KEY CLUSTERED
(
    [id_counter] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/***** Object: Table [dbo].[Documents]    Script Date: 06/23/2022 08:12:25 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Documents](
    [id_doc] [int] IDENTITY(1,1) NOT NULL,
    [sign_date] [date] NOT NULL,
    [id_h_doc] [int] NOT NULL,
    [id_consumer] [int] NULL,
    CONSTRAINT [PK_Documents] PRIMARY KEY CLUSTERED
(
    [id_doc] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/***** Object: UserDefinedFunction [dbo].[GetCostWithBenefit]    Script Date: 06/23/2022 08:12:26 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE FUNCTION [dbo].[GetCostWithBenefit]
(
    @cost decimal(18,2),
    @cons_id int
)
RETURNS decimal(18,2)
AS
BEGIN

    RETURN (SELECT CASE WHEN Consumers.id_benefit IS NOT NULL THEN @cost*(1.00-Benefits.benefit)
    ELSE @cost*1.00 end
    FROM Consumers
    LEFT JOIN Benefits ON Consumers.id_benefit=Benefits.id_benefit
    WHERE Consumers.id_consumer=@cons_id
    )

END
GO
/***** Object: StoredProcedure [dbo].[DeleteRepayment]    Script Date: 06/23/2022 08:12:30 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[DeleteRepayment]
AS
BEGIN
    Delete from Debtors
    where id_state = 3
END
GO
/***** Object: StoredProcedure [dbo].[Actual_Debtors]    Script Date: 06/23/2022 08:12:30 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[Actual_Debtors]
AS
BEGIN

```

```

        SELECT lname,fname,mname, amount, start_date, state_type from Debtors
        INNER JOIN States ON Debtors.id_state=States.id_state
        INNER JOIN Consumers ON Debtors.id_consumer=Consumers.id_consumer
        WHERE Debtors.id_state in (1,2)
END
GO
/***** Object: UserDefinedFunction [dbo].[GetAddress]    Script Date: 06/23/2022 08:12:30 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE FUNCTION [dbo].[GetAddress] (@serial varchar(12))
RETURNS varchar(100)
AS
BEGIN
    DECLARE @address varchar(100)
    SELECT @address=(cast(Consumers.street as varchar)+' '+cast(Consumers.house_number as varchar)+'
'+cast(Consumers.flat_number as varchar))
    FROM Consumers INNER JOIN Counters ON Counters.id_consumer = Consumers.id_consumer
    WHERE Counters.serial_number=@serial
    RETURN @address
END
GO
/***** Object: Table [dbo].[Indications]    Script Date: 06/23/2022 08:12:30 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Indications](
    [indication_date] [date] NOT NULL,
    [id_counter] [int] NOT NULL,
    [indication_rate] [decimal](18, 4) NOT NULL,
    CONSTRAINT [PK_Indications] PRIMARY KEY CLUSTERED
(
    [id_counter] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/***** Object: UserDefinedFunction [dbo].[GetDocuments]    Script Date: 06/23/2022 08:12:30 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE FUNCTION [dbo].[GetDocuments](@docNum int)
RETURNS TABLE
AS RETURN(
    SELECT id_doc as 'íííâð', doc_name as 'Èìÿ äîëóíâíðà', sample as 'Øâíêà', fname as 'Èìÿ ïîððááëèððäëÿ',
    lname as 'Òàìèèèÿ ïîððááëèððäëÿ', mname as 'Ìð÷âñòâî ïîððááëèððäëÿ', body as 'Òäëí äîëóíâíðà', sign_date as
'Ààðà ïîäèèñâèèÿ'
    FROM Documents INNER JOIN Doc_headers ON Documents.id_h_doc=Doc_headers.id_h_doc
    INNER JOIN Consumers ON Doc_headers.id_consumer=Consumers.id_consumer
    WHERE Documents.id_doc = @docNum
)
GO
/***** Object: UserDefinedFunction [dbo].[TotalDebtAmount]    Script Date: 06/23/2022 08:12:30 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE FUNCTION [dbo].[TotalDebtAmount] ()
RETURNS decimal(18,2)
AS
BEGIN
    DECLARE @total decimal(18,2)
    SELECT @total=SUM(amount)
    FROM Debtors
    RETURN @total
END
GO
/***** Object: StoredProcedure [dbo].[TotalAmountByRate]    Script Date: 06/23/2022 08:12:30 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[TotalAmountByRate]

```

```

AS
BEGIN
    SELECT Consumers.id_rate, rate_type,
    CASE WHEN Consumers.id_rate=7 THEN SUM(indication_rate*Rates.rate_fst)
    WHEN Consumers.id_rate=8 THEN SUM(indication_rate*(16*Rates.rate_fst + 8*Rates.rate_snd)/24)
    ELSE SUM(indication_rate*(7*rate_fst + 9*Rates.rate_snd + 8*Rates.rate_trd)/24) end as total
    from Indications
    INNER JOIN Counters ON Counters.id_counter=Indications.id_counter
    INNER JOIN Consumers ON Consumers.id_consumer=Counters.id_consumer
    INNER JOIN Rates ON Rates.id_rate=Consumers.id_rate
    GROUP BY Consumers.id_rate, rate_type
END
GO
/***** Object: View [dbo].[IndicationsByConsumer]    Script Date: 06/23/2022 08:12:31 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE VIEW [dbo].[IndicationsByConsumer]
AS
SELECT dbo.Indications.indication_date, dbo.Indications.indication_rate, dbo.Counters.serial_number,
dbo.Models.model_name, dbo.Consumers.fname,
        dbo.Consumers.lname, dbo.Consumers.mname
FROM   dbo.Counters INNER JOIN
        dbo.Indications ON dbo.Counters.id_counter = dbo.Indications.id_counter INNER JOIN
        dbo.Consumers ON dbo.Counters.id_consumer = dbo.Consumers.id_consumer INNER JOIN
        dbo.Models ON dbo.Counters.id_model = dbo.Models.id_model
GO

CREATE FUNCTION [dbo].[GetIndicationCost]
(
    @cons_id int
)
RETURNS decimal(18,2)
AS
BEGIN

RETURN (SELECT CASE WHEN Consumers.id_rate=7 THEN indication_rate*Rates.rate_fst
WHEN Consumers.id_rate=8 THEN indication_rate*(16*Rates.rate_fst + 8*Rates.rate_snd)/24
ELSE indication_rate*(7*rate_fst + 9*Rates.rate_snd + 8*Rates.rate_trd)/24 end as total
from Indications
INNER JOIN Counters ON Counters.id_counter=Indications.id_counter
INNER JOIN Consumers ON Consumers.id_consumer=Counters.id_consumer
INNER JOIN Rates ON Rates.id_rate=Consumers.id_rate
WHERE Consumers.id_consumer=@cons_id)
END
GO
/***** Object: StoredProcedure [dbo].[15_Max_Indications]    Script Date: 06/23/2022 08:12:31 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[15_Max_Indications]
AS
SET ROWCOUNT 15
BEGIN
    SELECT indication_date, indication_rate, serial_number, lname, fname, mname from Indications INNER JOIN
Counters ON Indications.id_counter=Counters.id_counter
    INNER JOIN Consumers ON Counters.id_consumer=Consumers.id_consumer
    Order by indication_rate DESC
END
GO
/***** Object: Default [DF_Profiles_state]    Script Date: 06/23/2022 08:12:25 *****/
ALTER TABLE [dbo].[Profiles] ADD CONSTRAINT [DF_Profiles_state] DEFAULT ((1)) FOR [state]
GO
/***** Object: Default [DF_Consumers_has_benefit]    Script Date: 06/23/2022 08:12:25 *****/
ALTER TABLE [dbo].[Consumers] ADD CONSTRAINT [DF_Consumers_has_benefit] DEFAULT ((0)) FOR [has_benefit]
GO
/***** Object: ForeignKey [FK_Profiles_Roles]    Script Date: 06/23/2022 08:12:25 *****/
ALTER TABLE [dbo].[Profiles] WITH CHECK ADD CONSTRAINT [FK_Profiles_Roles] FOREIGN KEY([id_role])
REFERENCES [dbo].[Roles] ([id_role])
GO
ALTER TABLE [dbo].[Profiles] CHECK CONSTRAINT [FK_Profiles_Roles]
GO
/***** Object: ForeignKey [FK_Consumers_Benefits]    Script Date: 06/23/2022 08:12:25 *****/
ALTER TABLE [dbo].[Consumers] WITH CHECK ADD CONSTRAINT [FK_Consumers_Benefits] FOREIGN KEY([id_benefit])

```

```

REFERENCES [dbo].[Benefits] ([id_benefit])
GO
ALTER TABLE [dbo].[Consumers] CHECK CONSTRAINT [FK_Consumers_Benefits]
GO
/***** Object: ForeignKey [FK_Consumers_Rates]    Script Date: 06/23/2022 08:12:25 *****/
ALTER TABLE [dbo].[Consumers] WITH CHECK ADD CONSTRAINT [FK_Consumers_Rates] FOREIGN KEY([id_rate])
REFERENCES [dbo].[Rates] ([id_rate])
GO
ALTER TABLE [dbo].[Consumers] CHECK CONSTRAINT [FK_Consumers_Rates]
GO
/***** Object: ForeignKey [FK_Debtors_Consumers]    Script Date: 06/23/2022 08:12:25 *****/
ALTER TABLE [dbo].[Debtors] WITH CHECK ADD CONSTRAINT [FK_Debtors_Consumers] FOREIGN KEY([id_consumer])
REFERENCES [dbo].[Consumers] ([id_consumer])
GO
ALTER TABLE [dbo].[Debtors] CHECK CONSTRAINT [FK_Debtors_Consumers]
GO
/***** Object: ForeignKey [FK_Debtors_States]    Script Date: 06/23/2022 08:12:25 *****/
ALTER TABLE [dbo].[Debtors] WITH CHECK ADD CONSTRAINT [FK_Debtors_States] FOREIGN KEY([id_state])
REFERENCES [dbo].[States] ([id_state])
GO
ALTER TABLE [dbo].[Debtors] CHECK CONSTRAINT [FK_Debtors_States]
GO
/***** Object: ForeignKey [FK_Counters_Consumers]    Script Date: 06/23/2022 08:12:25 *****/
ALTER TABLE [dbo].[Counters] WITH CHECK ADD CONSTRAINT [FK_Counters_Consumers] FOREIGN KEY([id_consumer])
REFERENCES [dbo].[Consumers] ([id_consumer])
GO
ALTER TABLE [dbo].[Counters] CHECK CONSTRAINT [FK_Counters_Consumers]
GO
/***** Object: ForeignKey [FK_Counters_Models]    Script Date: 06/23/2022 08:12:25 *****/
ALTER TABLE [dbo].[Counters] WITH CHECK ADD CONSTRAINT [FK_Counters_Models] FOREIGN KEY([id_model])
REFERENCES [dbo].[Models] ([id_model])
GO
ALTER TABLE [dbo].[Counters] CHECK CONSTRAINT [FK_Counters_Models]
GO
/***** Object: ForeignKey [FK_Documents_Consumers]    Script Date: 06/23/2022 08:12:25 *****/
ALTER TABLE [dbo].[Documents] WITH CHECK ADD CONSTRAINT [FK_Documents_Consumers] FOREIGN KEY([id_consumer])
REFERENCES [dbo].[Consumers] ([id_consumer])
GO
ALTER TABLE [dbo].[Documents] CHECK CONSTRAINT [FK_Documents_Consumers]
GO
/***** Object: ForeignKey [FK_Documents_Doc_headers]    Script Date: 06/23/2022 08:12:25 *****/
ALTER TABLE [dbo].[Documents] WITH CHECK ADD CONSTRAINT [FK_Documents_Doc_headers] FOREIGN KEY([id_h_doc])
REFERENCES [dbo].[Doc_headers] ([id_h_doc])
GO
ALTER TABLE [dbo].[Documents] CHECK CONSTRAINT [FK_Documents_Doc_headers]
GO
/***** Object: ForeignKey [FK_Indications_Counters]    Script Date: 06/23/2022 08:12:30 *****/
ALTER TABLE [dbo].[Indications] WITH CHECK ADD CONSTRAINT [FK_Indications_Counters] FOREIGN KEY([id_counter])
REFERENCES [dbo].[Counters] ([id_counter])
GO
ALTER TABLE [dbo].[Indications] CHECK CONSTRAINT [FK_Indications_Counters]
GO

```

Код вставки данных в таблицы

Роли

```

INSERT INTO Roles (role_name) VALUES ('Администратор')
INSERT INTO Roles (role_name) VALUES ('Бухгалтер')
INSERT INTO Roles (role_name) VALUES ('Юрист')
INSERT INTO Roles (role_name) VALUES ('Сотрудник')

```

Профили

```

INSERT INTO Profiles (fname,lname,mname,email,login,hash,salt,id_role,state) VALUES
('Валерий','Лекомцев','Валерьевич','lecom19@mail.ru','admin','e2388690083baf8ed44ea44b462cdb8997e5a010818828256f863d
52573723434040d7702efe7c4a6787b95fb7ee836fe54a86de83d88e83b2f76bda6ceddfd6','wX6PHgwQ12ng7yv',1,True)
INSERT INTO Profiles (fname,lname,mname,email,login,hash,salt,id_role,state) VALUES
('Дмитрий','Беляев','Сергеевич','spydetector@mail.ru','spy','025a14779ec330b2db8ce809292afce66c35b7e99a29b11134b14c9
142e427260c40b5caf50d061ec225be472b748f038315c86c7617ded3f660e3d3b202951c','xvpyTh62i0QMgvd',4,True)
INSERT INTO Profiles (fname,lname,mname,email,login,hash,salt,id_role,state) VALUES
('Владимир','Пугачёв','Филаретович','sandbox@mail.ru','pheelings','97ce0f84d5e4991497a061c560aebac8c8b6c48db36e89148
f911d0debd57fd7c59df51e35eab426994e79b8ef60e61499f11407402232099b5252fad31c9294','HsxBgJFFTP9W5iQ',2,True)
INSERT INTO Profiles (fname,lname,mname,email,login,hash,salt,id_role,state) VALUES
('Максим','Гундяев','Александрович','example@mail.ru','maximilian','ad9c6b34ad016e6d5728ac12989437adee7553d6217d4b4b
204bcf2f9bceec9a855b3dc38e139a6dcd86ea3fbdbd790511f8fbc9c241eba712fff32a0b0c75b8','sUZdsZ3XACj6Avk',3,True)

```

Статус долга

```
INSERT INTO States (state_type) VALUES ('Не погашен')
INSERT INTO States (state_type) VALUES ('Частично погашен')
INSERT INTO States (state_type) VALUES ('Погашен')
```

Льготы

```
INSERT INTO Benefits (benefit_type, description, benefit) VALUES ('Ветеран гос. службы',NULL,0.10)
INSERT INTO Benefits (benefit_type, description, benefit) VALUES ('Ветеран труда',NULL,0.25)
INSERT INTO Benefits (benefit_type, description, benefit) VALUES ('Ветеран ВОВ',NULL,0.25)
INSERT INTO Benefits (benefit_type, description, benefit) VALUES ('Имеющий инвалидность',NULL,0.20)
INSERT INTO Benefits (benefit_type, description, benefit) VALUES ('Гражданин, подвергшийся радиационному воздействию вследствие ядерных испытаний или ликвидации последствий аварии', 'Семипалатинск, Чернобыль',0.20)
INSERT INTO Benefits (benefit_type, description, benefit) VALUES ('Реабилитированное лицо или пострадавший от политических репрессий', 'В связи с законом',0.20)
```

Модели счётчиков

```
INSERT INTO Models (model_name) VALUES ('Меркурий-12')
INSERT INTO Models (model_name) VALUES ('Меркурий-13')
INSERT INTO Models (model_name) VALUES ('HEBA M3')
INSERT INTO Models (model_name) VALUES ('HEBA M4')
```

Тарифные ставки

```
INSERT INTO Rates (rate_type,rate_fst,rate_snd,rate_trd) VALUES ('Городское население',4.51,5.32,3.15)
INSERT INTO Rates (rate_type,rate_fst,rate_snd,rate_trd) VALUES ('Городское население с электроплитами',3.73,4.28,2.36)
INSERT INTO Rates (rate_type,rate_fst,rate_snd,rate_trd) VALUES ('Приравненные к населению',4.52,5.32,3.15)
```

Потребители

```
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Фёдор','Ильин','Михайлович','1953-11-30','50 лет победы','113',35,0,NULL,7)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Илья','Смирнов','Сергеевич','1987-12-14','Н. Крупской','4',128,0,NULL,8)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Сергей','Сергеев','Анатольевич','1999-06-14','Молодёжная','182',134,0,NULL,9)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Никита','Зеленин','Иванович','1986-03-11','Матросова','181',222,0,NULL,7)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Василий','Никитин','Анатольевич','1984-09-15','Молодёжная','78',235,0,NULL,8)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Никита','Иванов','Михайлович','1973-05-13','Армейская','120',181,0,NULL,9)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Дмитрий','Иванов','Петрович','1982-02-27','50 лет победы','191',112,0,NULL,7)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Фёдор','Зеленин','Петрович','1953-03-05','Н. Крупской','170',218,0,NULL,9)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Никита','Сергеев','Сергеевич','1959-12-13','Армейская','55',132,0,NULL,8)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Андрей','Лавров','Иванович','1997-04-09','Армейская','126',185,0,NULL,7)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Иван','Абрамов','Павлович','1996-08-23','Молодёжная','47',199,0,NULL,7)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Сергей','Сергеев','Михайлович','1997-04-09','Партизанская','146',186,0,NULL,9)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Илья','Сергеев','Павлович','1978-07-07','Партизанская','14',29,0,NULL,7)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Дмитрий','Сергеев','Михайлович','1982-02-27','9 мая','106',112,0,NULL,8)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Константин','Комаров','Михайлович','1999-06-14','Н. Крупской','180',237,0,NULL,9)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Андрей','Комаров','Павлович','1986-03-11','Фрунзе','162',46,0,NULL,7)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Никита','Абрамов','Сергеевич','1953-03-05','Партизанская','103',194,0,NULL,7)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Фёдор','Абрамов','Сергеевич','1999-06-14','Фрунзе','98',229,0,NULL,7)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Никита','Ильин','Иванович','1982-02-27','50 лет победы','159',62,0,NULL,8)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Иван','Лавров','Дмитриевич','1953-03-05','Матросова','55',39,0,NULL,8)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Василий','Зеленин','Сергеевич','1978-07-07','Н. Крупской','184',200,0,NULL,9)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Константин','Никитин','Михайлович','1999-06-14','Молодёжная','153',166,0,NULL,9)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Андрей','Лавров','Сергеевич','1953-11-30','Армейская','126',183,0,NULL,9)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Иван','Смирнов','Петрович','1964-05-12','Партизанская','13',133,0,NULL,7)
```

```

INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Илья','Абрамов','Иванович','1989-12-31','50 лет победы','141',125,0,NULL,9)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Виктор','Смирнов','Сергеевич','1978-07-07','9 мая','172',215,0,NULL,7)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Никита','Комаров','Дмитриевич','1996-08-23','Комсомольская','141',151,0,NULL,9)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Василий','Петров','Сергеевич','1948-04-06','Фрунзе','86',72,0,NULL,8)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Фёдор','Ильин','Иванович','1953-03-05','Партизанская','106',63,0,NULL,7)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Илья','Петров','Сергеевич','1999-06-14','Армейская','164',132,0,NULL,7)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Дмитрий','Иванов','Павлович','1982-02-27','Комсомольская','144',85,0,NULL,9)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Владимир','Смирнов','Дмитриевич','1964-05-12','Н. Крупской','83',108,0,NULL,9)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Фёдор','Зеленин','Михайлович','1989-12-02','Молодёжная','107',224,0,NULL,8)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Владимир','Иванов','Анатольевич','1973-05-13','9 мая','62',120,0,NULL,7)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Виктор','Комаров','Михайлович','1953-03-05','Н. Крупской','163',245,0,NULL,7)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Иван','Никитин','Петрович','1973-04-13','Армейская','37',200,0,NULL,7)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Василий','Зеленин','Михайлович','1999-06-14','Молодёжная','18',247,0,NULL,7)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Иван','Сергеев','Ильич','1986-03-11','9 мая','82',238,0,NULL,7)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Василий','Лавров','Анатольевич','1996-08-23','Молодёжная','88',123,0,NULL,8)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Виктор','Лавров','Дмитриевич','1984-09-15','Матросова','26',187,0,NULL,9)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Андрей','Сергеев','Петрович','1953-03-05','Армейская','121',123,0,NULL,7)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Виталина','Андреева','Сергеевна','1978-09-07','Энтузиастов','14',NULL,1,3,9)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Маргарита','Назарова','Владленовна','1978-07-08','Первомайская','19',NULL,0,NULL,7)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Василий','Максимов','Сергеевич','1978-02-14','Нежная','4',11,1,1,7)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Петр','Андреев','Сергеевич','1953-09-10','Комсомольская','4',13,1,2,7)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Андрей','Васильев','Сергеевич','1982-01-30','Василькова','4',1,1,1,7)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Валерий','Островский','Александрович','1986-09-12','Тракторная','11',NULL,0,NULL,7)
INSERT INTO Consumers(fname,lname,mname,birthdate,street,house_number,flat_number,has_benefit,id_benefit,id_rate)
VALUES ('Владимир','Путилин','Родионович','1974-12-14','Норвежская','14',NULL,1,1,7)

```

Документы

```

insert into Documents(sign_date,id_h_doc,id_consumer) VALUES ('2021-04-11',9,5)
insert into Documents(sign_date,id_h_doc,id_consumer) VALUES ('2022-04-13',17,5)
insert into Documents(sign_date,id_h_doc,id_consumer) VALUES ('2022-02-11',9,6)
insert into Documents(sign_date,id_h_doc,id_consumer) VALUES ('2022-05-16',9,8)
insert into Documents(sign_date,id_h_doc,id_consumer) VALUES ('2022-04-11',9,9)
insert into Documents(sign_date,id_h_doc,id_consumer) VALUES ('2022-03-11',9,10)
insert into Documents(sign_date,id_h_doc,id_consumer) VALUES ('2022-01-14',9,12)
insert into Documents(sign_date,id_h_doc,id_consumer) VALUES ('2022-04-11',9,17)
insert into Documents(sign_date,id_h_doc,id_consumer) VALUES ('2022-04-11',9,18)
insert into Documents(sign_date,id_h_doc,id_consumer) VALUES ('2022-04-15',9,19)
insert into Documents(sign_date,id_h_doc,id_consumer) VALUES ('2022-04-11',9,20)
insert into Documents(sign_date,id_h_doc,id_consumer) VALUES ('2022-04-12',9,21)
insert into Documents(sign_date,id_h_doc,id_consumer) VALUES ('2022-03-11',9,22)
insert into Documents(sign_date,id_h_doc,id_consumer) VALUES ('2022-04-11',9,23)
insert into Documents(sign_date,id_h_doc,id_consumer) VALUES ('2022-04-12',17,21)
insert into Documents(sign_date,id_h_doc,id_consumer) VALUES ('2022-03-11',17,22)
insert into Documents(sign_date,id_h_doc,id_consumer) VALUES ('2022-04-11',17,23)

```

Шапки документов

```

INSERT INTO Doc_headers (doc_name,sample,body) VALUES ('Договор о купле-продаже электроэнергии','АО "Электросбыт"
Юридический адрес: Константиновская, 16, корпус 2. ИНН: 4504398459485 ОКВЭД: 14',1.1. В соответствии с настоящим
Договором Гарантирующий поставщик обязуется осуществлять продажу Покупателю электрической энергии (мощности) и
урегулировать отношения по оказанию услуг по передаче электрической энергии и иных услуг, оказание которых является
неотъемлемой частью процесса энергоснабжения Покупателя, а Покупатель обязуется оплачивать поставленную
электрическую энергию (мощность) и оказанные услуги в порядке, предусмотренном Договором.')

```

INSERT INTO Doc_headers (doc_name,sample,body) VALUES ('Договор об оказании электромонтажных услуг','АО "Электросбыт" Юридический адрес: Константиновская, 16, корпус 2. ИНН: 4504398459485 ОКВЭД: 14','1.1. В соответствии с настоящим Договором Гарантирующий поставщик обязуется осуществлять продажу Покупателю электрической энергии (мощности) и урегулировать отношения по оказанию услуг по передаче электрической энергии и иных услуг, оказание которых является неотъемлемой частью процесса энергоснабжения Покупателя, а Покупатель обязуется оплачивать поставленную электрическую энергию (мощность) и оказанные услуги в порядке, предусмотренном Договором.')

Счётчики

```
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('COC1ZR9SMSS',4,1)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('C00LAFYVGZ50',5,1)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('C096ISYCP10',6,2)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('C0IWXCV8PXY',7,3)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('COWBC81AY2I',8,4)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('COFSI9Q0BAV',9,4)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('COQPX952ELA',10,2)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('COGQY8UGG66',12,1)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('COPHGZE6PMG',17,3)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('CODZU773QE5',18,2)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('CONGCLPNE4Y',19,4)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('COIGZ594Q19',20,1)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('C0039SF9TMM',21,3)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('CORXUAYLTUV',22,2)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('COV2YMJ01WA',23,4)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('C06KCD7Y4QA',24,4)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('C06DWVGX00P',25,4)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('C0980W13IF0',26,3)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('COT2BQJTI4',28,2)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('COXQDX1K92R',29,1)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('COE423TLY02',30,1)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('C09T2M9YF6I',31,2)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('COC8KR4J9BF',32,2)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('COBPZ4240CH',33,2)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('COHSVY5JFPT',34,3)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('COE5WI6PFS',35,3)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('COV9QA5KTL1',36,4)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('COOYD8KD18Z',37,1)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('COCHEWMXS73',38,4)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('CODHEBZG0CB',39,3)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('C03TDNJPSN8',41,3)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('C02X0J08M3E',42,4)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('CONXC996MS3',43,4)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('COEJH1SXHZ2',44,3)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('COG9YZHOPWC',46,1)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('COZ4UK0YKPI',47,3)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('COLQPNQHYAY',48,4)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('COR7IMKM5SF',49,1)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('COVZWIJA971',50,2)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('COC9N4AEB9I',51,3)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('C02181FYEUT',52,4)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('COF4SJV351',53,1)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('WYK4EC5NN51',56,1)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('XDFUGAHH887',81,2)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('UQ7EZKOYB9W',84,3)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('VFN77TYMVFQ',104,4)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('UALNK109N9C',108,2)
INSERT INTO Counters(serial_number,id_consumer,id_model) VALUES ('XVK98DNGIK2',109,1)
```

Показания счётчиков

```
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',2,207.5105)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',3,234.8607)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',4,159.0589)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',5,249.3786)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',6,257.6808)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',7,264.8824)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',8,178.3800)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',10,190.3226)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',15,232.6082)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',16,154.5047)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',17,241.6648)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',18,181.9910)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',19,262.4813)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',20,232.4140)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',21,169.3799)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',22,217.8684)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',23,155.2666)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',24,208.9800)
```

```

INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',26,233.5895)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',27,215.9649)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',28,243.8998)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',29,248.5829)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',30,185.5586)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',31,179.3429)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',32,200.1880)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',33,192.3516)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',34,268.2551)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',35,229.2344)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',36,209.5037)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',37,191.9075)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',39,245.2906)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',40,254.9582)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',41,231.6473)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',42,180.5665)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',44,272.7689)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',45,264.3488)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',46,175.3191)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',47,174.1045)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',48,172.8114)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',49,272.3513)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',50,215.6891)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',51,189.2138)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',52,374.5634)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',53,262.4634)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',54,322.1474)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',55,289.2624)
INSERT INTO Indications(indication_date,id_counter,indication_rate) VALUES ('2022-05-31',56,475.2672)

```

Должники

```

INSERT INTO Debtors(amount,start_date,actual_repayment_date,id_state,id_consumer) VALUES (2501.34,'2022-05-10',NULL,1,4)
INSERT INTO Debtors(amount,start_date,actual_repayment_date,id_state,id_consumer) VALUES (2756.34,'2022-05-10',NULL,2,17)
INSERT INTO Debtors(amount,start_date,actual_repayment_date,id_state,id_consumer) VALUES (2476.21,'2022-05-10','2022-05-14',3,5)
INSERT INTO Debtors(amount,start_date,actual_repayment_date,id_state,id_consumer) VALUES (2432.15,'2022-05-10',NULL,1,32)
INSERT INTO Debtors(amount,start_date,actual_repayment_date,id_state,id_consumer) VALUES (2221.78,'2022-05-10',NULL,1,51)
INSERT INTO Debtors(amount,start_date,actual_repayment_date,id_state,id_consumer) VALUES (2890.12,'2022-05-10','2022-05-16',3,48)
INSERT INTO Debtors(amount,start_date,actual_repayment_date,id_state,id_consumer) VALUES (1906.12,'2022-05-10',NULL,2,84)
INSERT INTO Debtors(amount,start_date,actual_repayment_date,id_state,id_consumer) VALUES (1785.15,'2022-05-10',NULL,2,56)
INSERT INTO Debtors(amount,start_date,actual_repayment_date,id_state,id_consumer) VALUES (2456.13,'2022-05-10',NULL,1,22)
INSERT INTO Debtors(amount,start_date,actual_repayment_date,id_state,id_consumer) VALUES (2500.12,'2022-05-10','2022-05-15',3,44)
INSERT INTO Debtors(amount,start_date,actual_repayment_date,id_state,id_consumer) VALUES (2411.12,'2022-05-10','2022-05-11',3,29)

```


Код приложения

Листинг кода класса SHACrypt

```

public class SHACrypt
{
    public SHACrypt() { }

    public string SHA512M(string pwd, string salt)
    {
        //получаем байткод пароля и соли
        var data = Encoding.UTF8.GetBytes(pwd + salt);
        //создаём реализацию класса SHA512
        var hash = SHA512.Create();

        //вычисляем хеш
        var result = hash.ComputeHash(data);
        //создаём построитель строки ёмкостью 128 символов, так как шифрование 512 бит, в байте 8 бит, 2 символа
        для байта var hashedInputStringBuilder = new System.Text.StringBuilder(128);

        foreach (var b in result)
            //форматируем части байткода в два 16-ричных символа в нижнем регистре
            hashedInputStringBuilder.Append(b.ToString("x2"));
        return hashedInputStringBuilder.ToString();
    }
    public string Saltate()
    {
        Random rand = new Random();
        string set="qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM1234567890";
        string result = "";

        for(int i = 1; i < 16; i++)
        {
            result = result + set[rand.Next(0, set.Length - 1)];
        }
        return result;
    }
}

```

Листинг кода клиентских частей

Начальное окно

```

public partial class StartWindow : Form
{
    public StartWindow()
    {
        InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e)
    {
        Authorization auth = new Authorization();
        auth.Show();
    }

    private void button2_Click(object sender, EventArgs e)
    {
        Registration reg = new Registration();
        reg.Show();
    }
}

```

Форма регистрации

```

public partial class Registration : Form
{
    public Registration()
    {
        InitializeComponent();
    }
}

```

```

private void button1_Click(object sender, EventArgs e)
{
    //Процесс регистрации начинается с
    this.sqlDataAdapter1.SelectCommand.CommandText = "SELECT [email], [login] FROM [Profiles] WHERE
email=@email or login=@login";
    this.sqlDataAdapter1.SelectCommand.CommandType = CommandType.Text;
    SqlParameter p1 = new SqlParameter("@email",this.textBox4.Text);
    this.sqlDataAdapter1.SelectCommand.Parameters.Add(p1);
    SqlParameter p2 = new SqlParameter("@login", this.textBox5.Text);
    this.sqlDataAdapter1.SelectCommand.Parameters.Add(p2);
    try
    {
        DataTable table = new DataTable();
        this.sqlDataAdapter1.Fill(table);
        if (table.Rows.Count > 0)
        {
            this.label7.Text = "Извините, но в системе уже есть профиль с таким логином или эл. почтой";
            button1.Enabled = false;
        }
        else
        {
            if (textBox1.Text == "" || textBox2.Text == "" || textBox3.Text == "" || textBox4.Text == "" ||
textBox5.Text == "" || textBox6.Text == "")
            {
                this.label7.Text = "Остались незаполненные данные";
            }
            else
            {
                if(this.textBox5.Text.Length<9 || this.textBox6.Text.Length < 9)
                {
                    this.label7.Text = "Логин или пароль должен состоять из более чем 9 символов.";
                }
                else
                {
                    //добавляем в бд данные аккаунта
                    this.sqlDataAdapter1.InsertCommand.Parameters["@fname"].Value = this.textBox1.Text;
                    this.sqlDataAdapter1.InsertCommand.Parameters["@lname"].Value = this.textBox2.Text;
                    this.sqlDataAdapter1.InsertCommand.Parameters["@mname"].Value = this.textBox3.Text;
                    this.sqlDataAdapter1.InsertCommand.Parameters["@email"].Value = this.textBox4.Text;
                    this.sqlDataAdapter1.InsertCommand.Parameters["@login"].Value = this.textBox5.Text;
                    //подгружаем собственный класс по хешированию
                    SHACrypt sh = new SHACrypt();
                    string s = sh.Saltate();
                    string h = sh.SHA512M(this.textBox6.Text, s);
                    this.sqlDataAdapter1.InsertCommand.Parameters["@hash"].Value = h;
                    this.sqlDataAdapter1.InsertCommand.Parameters["@salt"].Value = s;
                    try
                    {
                        int count;
                        count = this.sqlDataAdapter1.InsertCommand.ExecuteNonQuery();
                        if (count == 1)
                        {
                            this.label7.Text = "Профиль успешно зарегистрирован ";
                        }
                        else
                        {
                            this.label7.Text = "Профиль не зарегистрирован ";
                        }
                    }
                    catch
                    {
                        this.label7.Text = "Соединение с сервером прервано";
                    }
                }
            }
        }
        //очищаем параметры, чтобы снова их задать
        this.sqlDataAdapter1.SelectCommand.Parameters.Clear();
    }
    catch
    {
        this.label7.Text = "Что-то пошло не так";
    }
}

private void textBox5_TextChanged(object sender, EventArgs e)
{

```

```

        button1.Enabled = true;
    }

    private void textBox4_TextChanged(object sender, EventArgs e)
    {
        button1.Enabled = true;
    }

    private void button2_Click(object sender, EventArgs e)
    {
        this.Close();
    }

    private void Registration_Load(object sender, EventArgs e)
    {
        try
        {
            sqlConnection1.Open();
        }
        catch
        {
            this.label7.Text = "Невозможно соединиться с сервером";
        }
    }

    private void Registration_FormClosed(object sender, FormClosedEventArgs e)
    {
        sqlConnection1.Close();
    }
}

```

Форма авторизации

```

public partial class Authorization : Form
{
    public Authorization()
    {
        InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e)
    {
        string login="", hash="", salt="";
        bool state;
        this.sqlDataAdapter1.SelectCommand.CommandText = "SELECT [login], [hash], [salt], [state] FROM
[Profiles]";
        this.sqlDataAdapter1.SelectCommand.CommandType = CommandType.Text;
        SHACrypt sh = new SHACrypt();
        DataTable table = new DataTable();
        this.sqlDataAdapter1.Fill(table);
        foreach(DataRow row in table.Rows)
        {
            login = row.ItemArray.GetValue(0).ToString();
            hash = row.ItemArray.GetValue(1).ToString();
            salt = row.ItemArray.GetValue(2).ToString();
            state = Convert.ToBoolean(row.ItemArray.GetValue(3));
            if (login == this.textBox1.Text && hash == sh.SHA512M(this.textBox2.Text, salt))
            {
                if (state == true)
                {
                    this.Close();
                    StartWindow.ActiveForm.Hide();
                    //передаём логин на главную панель
                    SystemPanel sp = new SystemPanel(login);
                    sp.Show();
                }
                else
                {
                    this.label11.Text = "Аккаунт не активен. Обратитесь к администратору";
                }
            }
            else
            {
                this.label11.Text = "Введён неправильный логин или пароль";
            }
        }
    }

    private void button2_Click(object sender, EventArgs e)
    {

```

```

        this.Close();
    }
    private void Authorization_Load(object sender, EventArgs e)
    {
        try
        {
            sqlConnection1.Open();
        }
        catch
        {
            this.label1.Text = "Невозможно соединиться с сервером";
        }
    }
    private void Authorization_FormClosed(object sender, FormClosedEventArgs e)
    {
        sqlConnection1.Close();
    }
}

```

Главная навигационная форма

```

public partial class SystemPanel : Form
{
    public SystemPanel(string login)
    {
        InitializeComponent();
        string role = "";
        DataTable data = new DataTable();
        this.sqlDataAdapter1.SelectCommand.Parameters["@login"].Value = login;
        this.sqlDataAdapter1.Fill(data);
        //правила в том случае, если у аккаунта нет заданной роли в бд
        if (data.Rows.Count > 0)
        {
            role = data.Rows[0].ItemArray.GetValue(0).ToString();
        }
        else
        {
            button1.Visible = false;
            button2.Visible = false;
            button3.Visible = false;
            button4.Visible = false;
            button10.Visible = false;
            button7.Visible = false;
            button8.Visible = false;
            button9.Visible = false;
            label1.Text = "У вас недостаточно привилегий. Обратитесь к администратору.";
        }
        //проверка роли и раздача привилегий в зависимости от роли
        switch (role)
        {
            case "Администратор":
                break;
            case "Бухгалтер":
                button1.Enabled = false;
                button2.Enabled = false;
                button8.Enabled = false;
                button9.Enabled = false;
                break;
            case "Юрист":
                button1.Enabled = false;
                button2.Enabled = false;
                button4.Enabled = false;
                button8.Enabled = false;
                button9.Enabled = false;
                button10.Enabled = false;
                button3.Text = "Посмотреть список должников";
                button7.Text = "Посмотреть и редактировать список документов";
                break;
            case "Сотрудник":
                button8.Enabled = false;
                button10.Enabled = false;
                button9.Text = "Посмотреть и редактировать список счётчиков";
                button1.Text = "Посмотреть список потребителей";
                button2.Text = "Посмотреть показания счётчиков";
                button3.Text = "Посмотреть список должников";
                button4.Text = "Посмотреть тарифные ставки";

```

```

        break;
    default:
        break;
    }
    this.label2.Text = login;
}
private void button1_Click(object sender, EventArgs e)
{
    Consumers frm2 = new Consumers();
    frm2.Show();
}
private void button2_Click(object sender, EventArgs e)
{
    Indications frm3 = new Indications();
    frm3.Show();
}
private void button3_Click(object sender, EventArgs e)
{
    Debtors frm4 = new Debtors();
    frm4.Show();
}
private void button4_Click(object sender, EventArgs e)
{
    Rates frm5 = new Rates();
    frm5.Show();
}
private void button5_Click(object sender, EventArgs e)
{
    Authorization auth = new Authorization();
    auth.Show();
}
private void button6_Click(object sender, EventArgs e)
{
    this.Close();
}
private void SystemPanel_FormClosed(object sender, FormClosedEventArgs e)
{
    Form active = Application.OpenForms[0];
    active.Show();
}

private void button9_Click(object sender, EventArgs e)
{
    Counters frm6 = new Counters();
    frm6.Show();
}

private void button7_Click(object sender, EventArgs e)
{
    Documents frm7 = new Documents();
    frm7.Show();
}

private void button8_Click(object sender, EventArgs e)
{
    ProfilesEditing frm8 = new ProfilesEditing();
    frm8.Show();
}

private void button10_Click(object sender, EventArgs e)
{
    Benefits frm9 = new Benefits();
    frm9.Show();
}
}

```

Форма редактирования списка потребителей

```

public partial class Consumers : Form
{
    public Consumers()
    {
        InitializeComponent();
    }
    private void Form2_Load(object sender, EventArgs e)
    {

```

```

try
{
    this.sqlConnection1.Open();
    this.sqlDataAdapter1.Fill(consumersSet1);
    DataTable street_dt = new DataTable();
    street_dt.Rows.Add();
    street_dt.AcceptChanges();
    SqlCommand street_cmd = new SqlCommand("select DISTINCT street from Consumers",sqlConnection1);
    SqlDataAdapter street_adapt = new SqlDataAdapter(street_cmd);
    street_adapt.Fill(street_dt);
    if(street_dt.Rows.Count > 0)
    {
        this.comboBox3.DisplayMember = street_dt.Columns[0].ToString();
        this.comboBox3.ValueMember = street_dt.Columns[0].ToString();
        this.comboBox3.DataSource = street_dt;
    }
    else
    {
        this.label10.Text = "Произошел сбой";
        this.label10.ForeColor = Color.Red;
    }
    DataTable benefits_dt = new DataTable();
    benefits_dt.Rows.Add();
    benefits_dt.AcceptChanges();
    SqlCommand benefits_cmd = new SqlCommand("select id_benefit, benefit_type from Benefits",
sqlConnection1);
    SqlDataAdapter benefits_adapt = new SqlDataAdapter(benefits_cmd);
    benefits_adapt.Fill(benefits_dt);
    if (benefits_dt.Rows.Count > 0)
    {
        this.comboBox1.DisplayMember = benefits_dt.Columns[1].ToString();
        this.comboBox1.ValueMember = benefits_dt.Columns[0].ToString();
        this.comboBox1.DataSource = benefits_dt;
    }
    else
    {
        this.label10.Text = "Произошел сбой";
        this.label10.ForeColor = Color.Red;
    }
    DataTable rates_dt = new DataTable();
    rates_dt.Rows.Add();
    rates_dt.AcceptChanges();
    SqlCommand rates_cmd = new SqlCommand("select id_rate, rate_type from Rates", sqlConnection1);
    SqlDataAdapter rates_adapt = new SqlDataAdapter(rates_cmd);
    rates_adapt.Fill(rates_dt);
    if (rates_dt.Rows.Count > 0)
    {
        this.comboBox2.DisplayMember = rates_dt.Columns[1].ToString();
        this.comboBox2.ValueMember = rates_dt.Columns[0].ToString();
        this.comboBox2.DataSource = rates_dt;
    }
    else
    {
        this.label10.Text = "Произошел сбой";
        this.label10.ForeColor = Color.Red;
    }
}
catch
{
    this.label10.Text = "Произошел сбой";
    this.label10.ForeColor = Color.Red;
}
this.dataGridView1.ClearSelection();
}

private void button1_Click(object sender, EventArgs e)
{
    //вставка
    SqlCommand command = new SqlCommand();
    command.CommandType = CommandType.Text;
    command.CommandText = "insert into Consumers
(fname,lname,mname,street,birthdate,house_number,flat_number,has_benefit,id_benefit,id_rate) VALUES
(@fname,@lname,@mname,@street,@birthdate,@house_number,@flat_number,@has_benefit,@id_benefit,@id_rate)";
    SqlParameter p1 = new SqlParameter("@fname", textBox1.Text.ToString());
    command.Parameters.Add(p1);
    SqlParameter p2 = new SqlParameter("@lname", textBox2.Text.ToString());

```

```

command.Parameters.Add(p2);
SqlParameter p3 = new SqlParameter("@mname", textBox3.Text.ToString());
command.Parameters.Add(p3);
SqlParameter p4 = new SqlParameter("@street", textBox4.Text.ToString());
command.Parameters.Add(p4);
SqlParameter p5 = new SqlParameter("@house_number", textBox5.Text.ToString());
command.Parameters.Add(p5);
SqlParameter p6;
if (String.IsNullOrEmpty(textBox6.Text))
{
    p6 = new SqlParameter("@flat_number", DBNull.Value);
    command.Parameters.Add(p6);
}
else
{
    p6 = new SqlParameter("@flat_number", Convert.ToInt32(textBox6.Text));
    command.Parameters.Add(p6);
}
SqlParameter p7, p8;
if (comboBox1.SelectedIndex == 0 && !checkBox1.Checked)
{
    p7 = new SqlParameter("@has_benefit", false);
    command.Parameters.Add(p7);
    p8 = new SqlParameter("@id_benefit", DBNull.Value);
    command.Parameters.Add(p8);
}
else if (comboBox1.SelectedIndex != 0 && checkBox1.Checked)
{
    p7 = new SqlParameter("@has_benefit", true);
    command.Parameters.Add(p7);
    p8 = new SqlParameter("@id_benefit", Convert.ToInt32(comboBox1.SelectedIndex));
    command.Parameters.Add(p8);
}
SqlParameter p9 = new SqlParameter("@id_rate", Convert.ToInt32(comboBox2.SelectedIndex)+6);
command.Parameters.Add(p9);
SqlParameter p10 = new SqlParameter("@birthdate", dateTimePicker1.Value.Date);
command.Parameters.Add(p10);
command.Connection = this.sqlConnection1;
try
{
    command.ExecuteNonQuery();
}
catch
{
    this.label10.Text = "Произошел сбой";
    this.label10.ForeColor = Color.Red;
}
}
private void button4_Click(object sender, EventArgs e)
{
    //фильтрация
    var filters = new List<String>();
    if (!String.IsNullOrEmpty(textBox7.Text))
    {
        filters.Add(String.Format("fname LIKE '%{0}%'", textBox7.Text));
    }
    if (!String.IsNullOrEmpty(textBox8.Text))
    {
        filters.Add(String.Format("lname LIKE '%{0}%'", textBox8.Text));
    }
    if (!String.IsNullOrEmpty(textBox9.Text))
    {
        filters.Add(String.Format("mname LIKE '%{0}%'", textBox9.Text));
    }
    if (comboBox3.SelectedIndex != 0 || comboBox3.SelectedIndex != -1)
    {
        filters.Add(String.Format("street LIKE '%{0}%'", comboBox3.SelectedValue.ToString()));
    }
    try
    {
        if(filters.Count == 1)
        {
            this.consumersBindingSource.Filter = filters.First();
        }
        else if(filters.Count > 1)
        {

```

```

        this.consumersBindingSource.Filter = String.Join(" AND ", filters);
    }
    else
    {
        this.consumersBindingSource.RemoveFilter();
    }
}
catch
{
    this.label10.Text = "Произошёл сбой при поиске информации";
    this.label10.ForeColor = Color.Red;
}
}

private void Consumers_FormClosed(object sender, FormClosedEventArgs e)
{
    this.sqlConnection1.Close();
}

private void button3_Click(object sender, EventArgs e)
{
    //обновление данных
    SqlCommand command = new SqlCommand();
    command.CommandType = CommandType.Text;
    string cmdtext_p1 = "UPDATE Consumers SET ";
    string cmdtext_p2 = " WHERE id_consumer = @id_consumer";
    var parameters = new List<String>();
    string cmdtext;
    if (!String.IsNullOrEmpty(textBox1.Text))
    {
        parameters.Add("fname=@fname");
        SqlParameter p1 = new SqlParameter("@fname", textBox1.Text.ToString());
        command.Parameters.Add(p1);
    }
    if (!String.IsNullOrEmpty(textBox2.Text))
    {
        parameters.Add("lname=@lname");
        SqlParameter p2 = new SqlParameter("@lname", textBox2.Text.ToString());
        command.Parameters.Add(p2);
    }
    if (!String.IsNullOrEmpty(textBox3.Text))
    {
        parameters.Add("mname=@mname");
        SqlParameter p3 = new SqlParameter("@mname", textBox3.Text.ToString());
        command.Parameters.Add(p3);
    }
    if (!String.IsNullOrEmpty(textBox4.Text))
    {
        parameters.Add("street=@street");
        SqlParameter p4 = new SqlParameter("@street", textBox4.Text.ToString());
        command.Parameters.Add(p4);
    }
    if (!String.IsNullOrEmpty(textBox5.Text))
    {
        parameters.Add("house_number=@house_number");
        SqlParameter p5 = new SqlParameter("@house_number", textBox5.Text.ToString());
        command.Parameters.Add(p5);
    }
    if (!String.IsNullOrEmpty(textBox6.Text))
    {
        parameters.Add("flat_number=@flat_number");
        SqlParameter p6 = new SqlParameter("@flat_number", Convert.ToInt32(textBox6.Text));
        command.Parameters.Add(p6);
    }
    SqlParameter p7, p8;
    if (comboBox1.SelectedIndex == 0 && !checkBox1.Checked)
    {
        parameters.Add("has_benefit=@has_benefit");
        parameters.Add("id_benefit=@id_benefit");
        p7 = new SqlParameter("@has_benefit", false);
        command.Parameters.Add(p7);
        p8 = new SqlParameter("@id_benefit", DBNull.Value);
        command.Parameters.Add(p8);
    }
    else if (comboBox1.SelectedIndex != 0 && checkBox1.Checked)
    {
        parameters.Add("has_benefit=@has_benefit");
    }
}

```



```

        parameters.Add("id_benefit=@id_benefit");
        p7 = new SqlParameter("@has_benefit", true);
        command.Parameters.Add(p7);
        p8 = new SqlParameter("@id_benefit", Convert.ToInt32(comboBox1.SelectedIndex));
        command.Parameters.Add(p8);
    }
    if (comboBox2.SelectedIndex != 0)
    {
        parameters.Add("id_rate=@id_rate");
        SqlParameter p9 = new SqlParameter("@id_rate", Convert.ToInt32(comboBox2.SelectedIndex));
        command.Parameters.Add(p9);
    }
    SqlParameter p10 = new SqlParameter("@id_consumer",
Convert.ToInt32(((DataRowView)this.consumersBindingSource.Current).Row["id_consumer"]));
    command.Parameters.Add(p10);

    if (parameters.Count == 1)
    {
        cmdtext = cmdtext_p1 + parameters.First() + cmdtext_p2;
    }
    else if (parameters.Count > 1)
    {
        cmdtext = cmdtext_p1 + String.Join(",", parameters) + cmdtext_p2;
    }
    else
    {
        cmdtext = "";
        this.label10.Text = "Произошёл сбой при поиске информации";
    }
    command.CommandText = cmdtext;
    command.Connection = this.sqlConnection1;
    try
    {
        command.ExecuteNonQuery();
    }
    catch
    {
        this.label10.Text = "Произошел сбой";
        this.label10.ForeColor = Color.Red;
    }
}
}
private void checkBox1_CheckedChanged(object sender, EventArgs e)
{
    if (!this.checkBox1.Checked)
    {
        this.comboBox1.SelectedIndex = 0;
        this.comboBox1.Enabled = false;
    }
    else
    {
        this.comboBox1.Enabled = true;
    }
}
}
}

```

Форма просмотра максимальных показателей счётчиков

```

public partial class MaxInd : Form
{
    public MaxInd()
    {
        InitializeComponent();
    }
    private void Form6_Load(object sender, EventArgs e)
    {
        try
        {
            sqlConnection1.Open();
            this.sqlDataAdapter1.Fill(maxIndSet1);
            this.Text = this.Text + String.Format(" по состоянию на {0:dd/MM/yyyy}",
DateTime.Parse(((DataRowView)this.maxIndicationsBindingSource.Current).Row["indication_date"].ToString()));
        }
        catch
        {
            MessageBox.Show("Что-то пошло не так...");
        }
    }
}

```

```

        this.dataGridView1.ClearSelection();
        this.sqlDataAdapter1.SelectCommand.Connection.Close();
    }
    private void button1_Click(object sender, EventArgs e)
    {
        this.Close();
    }
    private void MaxInd_FormClosed(object sender, FormClosedEventArgs e)
    {
        sqlConnection1.Close();
    }
}

```

Форма редактирования тарифных ставок

public partial class Rates : Form

```

{
    public Rates()
    {
        InitializeComponent();
    }
    private void Form5_Load(object sender, EventArgs e)
    {
        try
        {
            sqlConnection1.Open();
            this.sqlDataAdapter1.Fill(ratesSet1);
        }
        catch
        {
            this.label6.Text = "Произошел сбой";
            this.label6.ForeColor = Color.Red;
        }
        this.dataGridView1.ClearSelection();
    }
    private void button1_Click(object sender, EventArgs e)
    {
        RateAmount frm8 = new RateAmount();
        frm8.ShowDialog();
    }
    private void button2_Click(object sender, EventArgs e)
    {
        this.sqlDataAdapter1.UpdateCommand.CommandText = "UPDATE Rates SET rate_fst = @rate_fst, rate_snd = @rate_snd, rate_trd = @rate_trd WHERE id_rate = @id_rate";
        this.sqlDataAdapter1.UpdateCommand.CommandType = CommandType.Text;
        this.sqlDataAdapter1.UpdateCommand.Parameters.Clear();
        SqlParameter p1 = new SqlParameter("@rate_fst", this.numericUpDown1.Value);
        this.sqlDataAdapter1.UpdateCommand.Parameters.Add(p1);
        SqlParameter p2 = new SqlParameter("@rate_snd", this.numericUpDown2.Value);
        this.sqlDataAdapter1.UpdateCommand.Parameters.Add(p2);
        SqlParameter p3 = new SqlParameter("@rate_trd", this.numericUpDown3.Value);
        this.sqlDataAdapter1.UpdateCommand.Parameters.Add(p3);
        SqlParameter p4 = new SqlParameter("@id_rate", Convert.ToInt32(this.comboBox1.SelectedValue));
        this.sqlDataAdapter1.UpdateCommand.Parameters.Add(p4);
        try
        {
            this.sqlDataAdapter1.UpdateCommand.ExecuteNonQuery();
        }
        catch
        {
            this.label6.Text = "Произошел сбой";
            this.label6.ForeColor = Color.Red;
        }
    }
    private void Rates_FormClosed(object sender, FormClosedEventArgs e)
    {
        sqlConnection1.Close();
    }
    private void button3_Click(object sender, EventArgs e)
    {
        this.sqlDataAdapter1.InsertCommand.CommandText = "INSERT INTO Rates
(rate_type,rate_fst,rate_snd,rate_trd) VALUES (@rate_type_new, @rate_fst_new, @rate_snd_new, @rate_trd_new)";
        this.sqlDataAdapter1.InsertCommand.CommandType = CommandType.Text;
        this.sqlDataAdapter1.InsertCommand.Parameters.Clear();
        SqlParameter type = new SqlParameter("@rate_type_new", this.textBox1.Text);
        this.sqlDataAdapter1.InsertCommand.Parameters.Add(type);
        SqlParameter fst = new SqlParameter("@rate_fst_new", this.numericUpDown1.Value);
    }
}

```

```

        this.sqlDataAdapter1.InsertCommand.Parameters.Add(fst);
        SqlParameter snd = new SqlParameter("@rate_snd_new", this.numericUpDown2.Value);
        this.sqlDataAdapter1.InsertCommand.Parameters.Add(snd);
        SqlParameter trd = new SqlParameter("@rate_trd_new", this.numericUpDown3.Value);
        this.sqlDataAdapter1.InsertCommand.Parameters.Add(trd);
        try
        {
            this.sqlDataAdapter1.InsertCommand.ExecuteNonQuery();
        }
        catch
        {
            this.label6.Text = "Произошел сбой";
            this.label6.ForeColor = Color.Red;
        }
    }
    private void button4_Click(object sender, EventArgs e)
    {
        this.sqlDataAdapter1.DeleteCommand.CommandText = "DELETE FROM Rates WHERE id_rate=@del_id_rate";
        this.sqlDataAdapter1.DeleteCommand.CommandType = CommandType.Text;
        this.sqlDataAdapter1.DeleteCommand.Parameters.Clear();
        SqlParameter id = new SqlParameter("@del_id_rate",
Convert.ToInt32(((DataRowView)this.ratesBindingSource.Current).Row["id_rate"]));
        this.sqlDataAdapter1.DeleteCommand.Parameters.Add(id);
        try
        {
            this.sqlDataAdapter1.DeleteCommand.ExecuteNonQuery();
        }
        catch
        {
            this.label6.Text = "Произошел сбой";
            this.label6.ForeColor = Color.Red;
        }
    }
}

```

Форма просмотра выручек по тарифным ставкам

```

public partial class RateAmount : Form
{
    public RateAmount()
    {
        InitializeComponent();
    }
    private void Form8_Load(object sender, EventArgs e)
    {
        this.dataGridView1.ClearSelection();
        try
        {
            sqlConnection1.Open();
            this.sqlDataAdapter1.Fill(amountRateSet1);
        }
        catch
        {
            MessageBox.Show("Что-то пошло не так...");
        }
        this.sqlDataAdapter1.SelectCommand.Connection.Close();
    }
    private void button1_Click(object sender, EventArgs e)
    {
        this.Close();
    }
    private void RateAmount_FormClosed(object sender, FormClosedEventArgs e)
    {
        sqlConnection1.Close();
    }
}

```

Форма редактирования списка счётчиков и моделей

```

public partial class Counters : Form
{
    public Counters()
    {
        InitializeComponent();
    }
    private void Counters_Load(object sender, EventArgs e)
    {
        try

```

```

{
    this.sqlConnection1.Open();
    DataTable street_dt = new DataTable();
    street_dt.Rows.Add();
    street_dt.AcceptChanges();
    SqlCommand street_cmd = new SqlCommand("select DISTINCT street from Consumers", sqlConnection1);
    SqlDataAdapter street_adapt = new SqlDataAdapter(street_cmd);
    street_adapt.Fill(street_dt);
    if (street_dt.Rows.Count > 0)
    {
        this.comboBox2.DisplayMember = street_dt.Columns[0].ToString();
        this.comboBox2.ValueMember = street_dt.Columns[0].ToString();
        this.comboBox2.DataSource = street_dt;
    }
    else
    {
        this.label2.Text = "Произошел сбой";
        this.label2.ForeColor = Color.Red;
    }
}
catch
{
    this.label2.Text = "Произошел сбой";
    this.label2.ForeColor = Color.Red;
}
this.modelsTableAdapter.Fill(this.countersSet1.Models);
this.consumersTableAdapter.Fill(this.countersSet1.Consumers);
this.sqlDataAdapter1.Fill(this.countersSet1.Counters);
this.dataGridView1.ClearSelection();
this.dataGridView2.ClearSelection();
this.dataGridView3.ClearSelection();
}
private void button1_Click(object sender, EventArgs e)
{
    //фильтрация
    var filters = new List<String>();
    if (!String.IsNullOrEmpty(textBox1.Text))
    {
        filters.Add(String.Format("fname LIKE '%{0}%',", textBox1.Text));
    }
    if (!String.IsNullOrEmpty(textBox2.Text))
    {
        filters.Add(String.Format("lname LIKE '%{0}%',", textBox2.Text));
    }
    if (!String.IsNullOrEmpty(textBox3.Text))
    {
        filters.Add(String.Format("mname LIKE '%{0}%',", textBox3.Text));
    }
    if (comboBox2.SelectedIndex != 0 || comboBox2.SelectedIndex != -1)
    {
        filters.Add(String.Format("street LIKE '%{0}%',", comboBox2.SelectedValue.ToString()));
    }
    try
    {
        if (filters.Count == 1)
        {
            this.consumersBindingSource.Filter = filters.First();
        }
        else if (filters.Count > 1)
        {
            this.consumersBindingSource.Filter = String.Join(" AND ", filters);
        }
        else
        {
            this.consumersBindingSource.RemoveFilter();
        }
    }
    catch
    {
        this.label2.Text = "Произошёл сбой при поиске информации";
        this.label2.ForeColor = Color.Red;
    }
}
private void Counters_FormClosed(object sender, FormClosedEventArgs e)
{
    this.sqlConnection1.Close();
}

```

```

    }
    private void button2_Click(object sender, EventArgs e)
    {
        SqlCommand modelcmd = new SqlCommand("insert into Models (model_name) VALUES (@model_name)",
sqlConnection1);
        SqlParameter modelname = new SqlParameter("@model_name", textBox4.Text);
        modelcmd.Parameters.Add(modelname);
        SqlDataAdapter model_adapt = new SqlDataAdapter();
        model_adapt.InsertCommand = modelcmd;
        try
        {
            model_adapt.InsertCommand.ExecuteNonQuery();
        }
        catch
        {
            this.label2.Text = "Произошёл сбой";
            this.label2.ForeColor = Color.Red;
        }
    }

    private void button3_Click(object sender, EventArgs e)
    {
        SqlCommand modelcmd = new SqlCommand("Update Models SET model_name = @model_name WHERE
id_model=@id_model", sqlConnection1);
        SqlParameter modelname = new SqlParameter("@model_name", textBox4.Text);
        modelcmd.Parameters.Add(modelname);
        SqlParameter id = new
SqlParameter("@id_model", Convert.ToInt32(((DataRowView)this.modelsBindingSource.Current).Row["id_model"]));
        modelcmd.Parameters.Add(id);
        SqlDataAdapter model_adapt = new SqlDataAdapter();
        model_adapt.UpdateCommand = modelcmd;
        try
        {
            model_adapt.UpdateCommand.ExecuteNonQuery();
        }
        catch
        {
            this.label2.Text = "Произошёл сбой";
            this.label2.ForeColor = Color.Red;
        }
    }

    private void button4_Click(object sender, EventArgs e)
    {
        SqlCommand modelcmd = new SqlCommand("Delete from Models WHERE id_model=@id_model", sqlConnection1);
        SqlParameter id = new SqlParameter("@id_model",
Convert.ToInt32(((DataRowView)this.modelsBindingSource.Current).Row["id_model"]));
        modelcmd.Parameters.Add(id);
        SqlDataAdapter model_adapt = new SqlDataAdapter();
        model_adapt.DeleteCommand = modelcmd;
        try
        {
            model_adapt.DeleteCommand.ExecuteNonQuery();
        }
        catch
        {
            this.label2.Text = "Произошёл сбой";
            this.label2.ForeColor = Color.Red;
        }
    }

    private void button7_Click(object sender, EventArgs e)
    {
        SqlCommand countercmd = new SqlCommand("insert into Counters (serial_number,id_consumer,id_model) VALUES
(@serial,@consumer,@model)", sqlConnection1);
        SqlParameter modelid = new SqlParameter("@model",
Convert.ToInt32(((DataRowView)this.modelsBindingSource.Current).Row["id_model"]));
        countercmd.Parameters.Add(modelid);
        SqlParameter consumerid = new SqlParameter("@consumer",
Convert.ToInt32(((DataRowView)this.consumersBindingSource.Current).Row["id_consumer"]));
        countercmd.Parameters.Add(consumerid);
        SqlParameter serial = new SqlParameter("@serial", textBox5.Text);
        countercmd.Parameters.Add(serial);
        SqlDataAdapter counter_adapt = new SqlDataAdapter();
        counter_adapt.InsertCommand = countercmd;
        try
        {

```

```

        counter_adapt.InsertCommand.ExecuteNonQuery();
    }
    catch
    {
        this.label2.Text = "Произошёл сбой";
        this.label2.ForeColor = Color.Red;
    }
}
private void dataGridview3_CellClick(object sender, DataGridViewCellEventArgs e)
{
    this.label12.Text = ((DataRowView)this.modelsBindingSource.Current).Row["model_name"].ToString();
}
private void dataGridview2_CellClick(object sender, DataGridViewCellEventArgs e)
{
    this.label14.Text = ((DataRowView)this.consumersBindingSource.Current).Row["lname"].ToString() + " " +
        ((DataRowView)this.consumersBindingSource.Current).Row["fname"].ToString().Substring(0,1) + ". " +
        + ((DataRowView)this.consumersBindingSource.Current).Row["mname"].ToString().Substring(0, 1) + ".";
}
}

```

Форма просмотра показаний счётчиков

```

public partial class Indications : Form
{
    string fname="",mname = "", lname = "", street = "", house_number = "", indication = "", money = "", benefit
= "";

    private void textBox1_TextChanged(object sender, EventArgs e)
    {
        if (!String.IsNullOrEmpty(textBox1.Text))
        {
            this.countersBindingSource.Filter = String.Format("serial_number LIKE '%{0}%'", textBox1.Text);
        }
        else
        {
            this.countersBindingSource.RemoveFilter();
        }
    }
    DateTime date;
    private void button2_Click(object sender, EventArgs e)
    {
        //код формирования документа Word
        string flat_number=
        ((DataRowView)this.consumersCountersBindingSource.Current).Row["flat_number"].ToString();
        Word.Application app = new Word.Application();
        app.Visible = false;
        Word.Document doc = app.Documents.Add();
        Word.Range r = doc.Range();
        doc.Paragraphs.Add();
        r.Paragraphs[1].Range.Text = "Квитанция от " + String.Format("{0:dd/MM/yyyy}", this.date).ToUpper();
        r.Paragraphs[1].Range.FormattedText.Font.Size = 15;
        doc.Paragraphs[1].Range.ParagraphFormat.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;
        doc.Paragraphs.Add();
        doc.Paragraphs.Add();
        r = doc.Range(doc.Paragraphs[2].Range.Start-1, doc.Paragraphs[2].Range.Start-1);
        Word.Table t = doc.Paragraphs[2].Range.Tables.Add(r, 4, 2);
        t.Borders.Enable = 1;
        try
        {
            t.Cell(1, 1).Range.Text = "Ф.И.О.";
            t.Cell(2, 1).Range.Text = "Адрес";
            t.Cell(3, 1).Range.Text = "Потреблено";
            if (((DataRowView)this.countersIndicationsBindingSource.Current).Row["Benefit"] != DBNull.Value)
            {
                t.Cell(4, 1).Range.Text = "Необходимо оплатить с учётом скидки " + this.benefit+"%";
            }
            else
            {
                t.Cell(4, 1).Range.Text = "Необходимо оплатить";
            }
            t.Cell(1, 2).Range.Text = this.lname + " " + this.fname.Substring(0, 1) + "." +
this.mname.Substring(0, 1) + ".";
            if (!String.IsNullOrEmpty(flat_number)){
                t.Cell(2, 2).Range.Text = "ул. " + this.street + ", " + this.house_number + " кв. " +
flat_number;
            }
            else

```

```

        {
            t.Cell(2, 2).Range.Text = "ул. " + this.street + ", " + this.house_number;
        }
        t.Cell(3, 2).Range.Text = this.indication + " кВт/ч";
        t.Cell(4, 2).Range.Text = this.money + "р.";
        foreach (Word.Column col in t.Columns)
        {
            foreach (Word.Cell cell in col.Cells)
            {
                if (cell.ColumnIndex == 1)
                {
                    cell.Range.Bold = 1;
                }
            }
        }
        try
        {
            doc.Save();
            doc.Close();
            app.Quit();
        }
        catch
        {
            MessageBox.Show("Что-то пошло не так...");
        }
    }
    catch
    {
        MessageBox.Show("Вы не указали счётчик в таблице.");
    }
}
public Indications()
{
    InitializeComponent();
}
private void Form3_Load(object sender, EventArgs e)
{
    this.indicationsTableAdapter.Fill(this.iCCDataSet.Indications);
    this.consumersTableAdapter.Fill(this.iCCDataSet.Consumers);
    this.countersTableAdapter.Fill(this.iCCDataSet.Counters);
    //снимаем выделение строки при загрузке
    this.dataGridView1.ClearSelection();
}
private void button1_Click(object sender, EventArgs e)
{
    MaxInd frm6 = new MaxInd();
    frm6.ShowDialog();
}
private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs e)
{
    try
    {
        //вывод данных текущих строк связанных данных в метку
        this.fname = ((DataRowView)this.consumersCountersBindingSource.Current).Row["fname"].ToString();
        this.mname = ((DataRowView)this.consumersCountersBindingSource.Current).Row["mname"].ToString();
        this.lname = ((DataRowView)this.consumersCountersBindingSource.Current).Row["lname"].ToString();
        this.street = ((DataRowView)this.consumersCountersBindingSource.Current).Row["street"].ToString();
        this.house_number =
        ((DataRowView)this.consumersCountersBindingSource.Current).Row["house_number"].ToString();
        this.indication =
        ((DataRowView)this.countersIndicationsBindingSource.Current).Row["indication_rate"].ToString();
        //парсим объект в datetime
        this.date =
        DateTime.Parse(((DataRowView)this.countersIndicationsBindingSource.Current).Row["indication_date"].ToString());
        this.money = ((DataRowView)this.countersIndicationsBindingSource.Current).Row["Costs"].ToString();

        if(((DataRowView)this.countersIndicationsBindingSource.Current).Row["Benefit"] == DBNull.Value) {
            //создаём форматную строку (так же форматируем объект datetime)
            this.label2.Text = String.Format("{0} \n{1}\n{2}\нул. {3}, д. {4}\nПо состоянию на дату
{6:dd/MM/yyyy}\nПолучил(а) электроэнергию в количестве {5} кВт/ч\nНеобходимо оплатить {7} рублей", fname, mname,
lname, street, house_number, indication, date, money);
        }
    }
}

```

```

        else
        {
            this.benefit =
((DataRowView)this.countersIndicationsBindingSource.Current).Row["Benefit"].ToString();
            this.label2.Text = String.Format("{0} \n{1}\n{2}\нул. {3}, д. {4}\nПо состоянию на дату
{6:dd/MM/yyyy}\nПолучил(а) электроэнергию в количестве {5} кВт/ч\nНеобходимо оплатить {7} рублей с учётом скидки
{8}%", fname, mname, lname, street, house_number, indication, date, money, benefit);
        }
    }
    catch
    {
        this.label2.Text = "Произошёл сбой. Попробуйте позже.";
    }
}
}

```

Форма просмотра списка должников

```

public partial class Debtors : Form
{
    public Debtors()
    {
        InitializeComponent();
    }

    private void Form4_Load(object sender, EventArgs e)
    {
        this.sqlConnection1.Open();
        this.consumersTableAdapter.Fill(this.debtsSet.Consumers);
        this.sqlDataAdapter1.Fill(this.debtsSet.Debtors);
        //запрос на выполнение скалярной функции
        SqlCommand cmd = new SqlCommand();
        cmd.CommandType = CommandType.Text;
        cmd.CommandText = "select dbo.TotalDebtAmount()";
        cmd.Connection = this.sqlConnection1;
        string total = cmd.ExecuteScalar().ToString();
        this.label3.Text+=total+"п.";
    }

    private void button1_Click(object sender, EventArgs e)
    {
        try
        {
            //запрос на выполнение хранимой процедуры
            SqlCommand command = new SqlCommand();
            command.CommandType = CommandType.StoredProcedure;
            command.CommandText = "exec dbo.DeleteRepayment";
            command.Connection = this.sqlConnection1;
            command.ExecuteNonQuery();
            this.dataGridView1.EndEdit();
            this.dataGridView1.Refresh();
        }
        catch
        {
            this.label2.Text = "Нет больше записей для удаления";
            this.label2.ForeColor = Color.Red;
        }
    }

    private void button2_Click(object sender, EventArgs e)
    {
        ActualDebtors frm7 = new ActualDebtors();
        frm7.ShowDialog();
    }

    private void Debtors_FormClosed(object sender, FormClosedEventArgs e)
    {
        this.sqlConnection1.Close();
    }
}

```

Форма просмотра актуального списка должников

```

public partial class ActualDebtors : Form
{
    public ActualDebtors()
    {
        InitializeComponent();
    }

    private void Form7_Load(object sender, EventArgs e)
    {

```



```

        this.sqlConnection1.Open();
        this.sqlDataAdapter1.Fill(adSet1);
    }

    private void button1_Click(object sender, EventArgs e)
    {
        this.Close();
    }

    private void ActualDebtors_FormClosed(object sender, FormClosedEventArgs e)
    {
        this.sqlConnection1.Close();
    }
}

```

Форма просмотра документов

```

public Documents()
{
    InitializeComponent();
}

private void Documents_Load(object sender, EventArgs e)
{
    this.sqlConnection1.Open();
    this.consumersTableAdapter.Fill(this.docsSet1.Consumers);
    this.sqlDataAdapter1.Fill(this.docsSet1.Doc_headers);
    this.dataGridView1.ClearSelection();
    this.dataGridView2.ClearSelection();
}

private void Documents_FormClosed(object sender, FormClosedEventArgs e)
{
    this.sqlConnection1.Close();
}

private void button1_Click(object sender, EventArgs e)
{
    Word.Application app = new Word.Application();
    app.Visible = false;
    Word.Document doc = app.Documents.Add();
    Word.Range range = doc.Range();
    //Почерёдно добавляю абзацы и наполняю их информацией из БД
    doc.Paragraphs.Add();
    Word.Range r1 = doc.Range(doc.Paragraphs[1].Range.Start, doc.Paragraphs[1].Range.End);
    r1.Text = ((DataRowView)this.consumersDoheadersBindingSource.Current).Row["doc_name"].ToString();
    r1.Text = r1.Text.ToUpper();
    r1.FormattedText.Font.Bold = 1;
    r1.FormattedText.Font.Size = 14;
    range.FormattedText.Font.Name = "Times New Roman";
    r1.ParagraphFormat.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;

    doc.Paragraphs.Add();
    Word.Range r2 = doc.Range(doc.Paragraphs[2].Range.Start, doc.Paragraphs[2].Range.End-1);
    r2.Text = ((DataRowView)this.consumersDoheadersBindingSource.Current).Row["sample"].ToString();
    r2.FormattedText.Font.Bold = 0;
    //задаю отступ
    r2.ParagraphFormat.Alignment = Word.WdParagraphAlignment.wdAlignParagraphJustify;
    r2.ParagraphFormat.FirstLineIndent = app.CentimetersToPoints(5.75f);
    r2.ParagraphFormat.LeftIndent = app.CentimetersToPoints(7.5f);
    doc.Paragraphs.Add();
    doc.Paragraphs.Add();
    Word.Range r3 = doc.Range(doc.Paragraphs[3].Range.Start, doc.Paragraphs[3].Range.End-1);

    r3.ParagraphFormat.FirstLineIndent = app.CentimetersToPoints(1.25f);
    r3.ParagraphFormat.LeftIndent = app.CentimetersToPoints(0);

    string fio = ((DataRowView)this.consumersBindingSource.Current).Row["lname"].ToString() + "a " +
        ((DataRowView)this.consumersBindingSource.Current).Row["fname"].ToString().Substring(0, 1) + ". " +
        ((DataRowView)this.consumersBindingSource.Current).Row["mname"].ToString().Substring(0, 1) + ".";
    string street;
    if (((DataRowView)this.consumersBindingSource.Current).Row["flat_number"] != DBNull.Value)
    {
        street = String.Format("ул. {0}, {1} кв. {2}",
            ((DataRowView)this.consumersBindingSource.Current).Row["street"].ToString(),
            ((DataRowView)this.consumersBindingSource.Current).Row["house_number"].ToString(),

```

```

        ((DataRowView)this.consumersBindingSource.Current).Row["flat_number"].ToString());
    }
    else
    {
        street = String.Format("ул. {0}, {1}",
        ((DataRowView)this.consumersBindingSource.Current).Row["street"].ToString(),
        ((DataRowView)this.consumersBindingSource.Current).Row["house_number"].ToString());
    }

    r3.Text = String.Format("Акционерное общество «Электросбыт», именуемое в дальнейшем «Гарантирующий поставщик», " +
        "в лице директора Петрука Александра Алексеевича, действующего на основании Устава общества, и {0} именуемый(ая) в дальнейшем «Покупатель», " +
        "в лице {0} проживающего по адресу {1} и действующего на основании запроса о подключении к энергосбытовому процессу, далее вместе именуемые «Стороны», заключили договор о нижеследующем:"
        , fio, street);

    Word.Range r4 = doc.Range(doc.Paragraphs[4].Range.Start, doc.Paragraphs[4].Range.End);
    r4.Text = ((DataRowView)this.consumersDoheadersBindingSource.Current).Row["body"].ToString();
    r4.ParagraphFormat.Alignment = Word.WdParagraphAlignment.wdAlignParagraphJustify;
    r4.ParagraphFormat.FirstLineIndent = app.CentimetersToPoints(1.25f);
    r4.ParagraphFormat.LeftIndent = app.CentimetersToPoints(0);
    doc.Paragraphs.Add();
    Word.Range r5 = doc.Range(doc.Paragraphs[5].Range.Start, doc.Paragraphs[5].Range.End);
    r5.Text = "Дата: " + String.Format("{0:dd/MM/yyyy}",
    DateTime.Parse(((DataRowView)this.consumersDoheadersBindingSource.Current).Row["sign_date"].ToString()));
    r5.ParagraphFormat.Alignment = Word.WdParagraphAlignment.wdAlignParagraphRight;

    try
    {
        doc.Save();
        doc.Close();
        app.Quit();
    }
    catch
    {
        MessageBox.Show("Что-то пошло не так...");
        doc.Close();
        app.Quit();
    }

    try
    {
    }
    catch
    {
        MessageBox.Show("Вы не указали документ в таблице.");
    }
}
}
}

```

Форма просмотра и редактирования списка ЛЬГОТ

```

public partial class Benefits : Form
{
    public Benefits()
    {
        InitializeComponent();
    }

    private void Benefits_Load(object sender, EventArgs e)
    {
        this.sqlConnection1.Open();
        this.sqlDataAdapter1.Fill(benefitsSet1);
    }

    private void button3_Click(object sender, EventArgs e)
    {
        this.sqlDataAdapter1.InsertCommand.CommandText = "INSERT INTO Benefits
        (benefit_type,description,benefit) VALUES (@bt, @desc, @benefit)";
        this.sqlDataAdapter1.InsertCommand.CommandType = CommandType.Text;
        this.sqlDataAdapter1.InsertCommand.Parameters.Clear();
        SqlParameter type = new SqlParameter("@bt", this.textBox1.Text);
        this.sqlDataAdapter1.InsertCommand.Parameters.Add(type);
        SqlParameter desc = new SqlParameter("desc", this.textBox2.Text);
        this.sqlDataAdapter1.InsertCommand.Parameters.Add(desc);
        SqlParameter benefit = new SqlParameter("@benefit", this.numericUpDown1.Value);
    }
}

```

```

        this.sqlDataAdapter1.InsertCommand.Parameters.Add(benefit);
        try
        {
            this.sqlDataAdapter1.InsertCommand.ExecuteNonQuery();
        }
        catch
        {
            this.label5.Text = "Произошел сбой";
            this.label5.ForeColor = Color.Red;
        }
    }

    private void button2_Click(object sender, EventArgs e)
    {
        this.sqlDataAdapter1.UpdateCommand.CommandText = "UPDATE Benefits SET benefit_type = @bt,
description=@desc, benefit = @benefit WHERE id_benefit = @id_ben";
        this.sqlDataAdapter1.UpdateCommand.CommandType = CommandType.Text;
        this.sqlDataAdapter1.UpdateCommand.Parameters.Clear();
        SqlParameter type = new SqlParameter("@bt", this.textBox1.Text);
        this.sqlDataAdapter1.UpdateCommand.Parameters.Add(type);
        SqlParameter desc = new SqlParameter("@desc", this.textBox2.Text);
        this.sqlDataAdapter1.UpdateCommand.Parameters.Add(desc);
        SqlParameter benefit = new SqlParameter("@benefit", this.numericUpDown1.Value);
        this.sqlDataAdapter1.UpdateCommand.Parameters.Add(benefit);
        SqlParameter id = new SqlParameter("@id_ben", Convert.ToInt32(this.comboBox1.SelectedValue));
        this.sqlDataAdapter1.UpdateCommand.Parameters.Add(id);
        try
        {
            this.sqlDataAdapter1.UpdateCommand.ExecuteNonQuery();
        }
        catch
        {
            this.label5.Text = "Произошел сбой";
            this.label5.ForeColor = Color.Red;
        }
    }

    private void button1_Click(object sender, EventArgs e)
    {
        this.sqlDataAdapter1.DeleteCommand.CommandText = "DELETE FROM Benefits WHERE id_benefit=@id_ben";
        this.sqlDataAdapter1.DeleteCommand.CommandType = CommandType.Text;
        this.sqlDataAdapter1.DeleteCommand.Parameters.Clear();
        SqlParameter id = new SqlParameter("@id_ben",
Convert.ToInt32(((DataRowView)this.benefitsBindingSource.Current).Row["id_benefit"]));
        this.sqlDataAdapter1.DeleteCommand.Parameters.Add(id);
        try
        {
            this.sqlDataAdapter1.DeleteCommand.ExecuteNonQuery();
        }
        catch
        {
            this.label5.Text = "Произошел сбой";
            this.label5.ForeColor = Color.Red;
        }
    }

    private void Benefits_FormClosed(object sender, FormClosedEventArgs e)
    {
        this.sqlConnection1.Close();
    }
}

```

Форма просмотра и редактирования списка профилей системы

```

public partial class ProfilesEditing : Form
{
    public ProfilesEditing()
    {
        InitializeComponent();
    }

    private void ProfilesEditing_Load(object sender, EventArgs e)
    {
        this.rolesTableAdapter.Fill(this.profilesSet1.Roles);
        this.sqlConnection1.Open();
        this.sqlDataAdapter1.Fill(profilesSet1);
    }
}

```

```

        this.dataGridView1.ClearSelection();
    }

    private void ProfilesEditing_FormClosed(object sender, FormClosedEventArgs e)
    {
        this.sqlConnection1.Close();
    }

    private void button1_Click(object sender, EventArgs e)
    {
        this.sqlDataAdapter1.DeleteCommand.CommandText = "DELETE FROM Profiles WHERE id_profile=@id";
        this.sqlDataAdapter1.DeleteCommand.CommandType = CommandType.Text;
        this.sqlDataAdapter1.DeleteCommand.Parameters.Clear();
        SqlParameter id = new SqlParameter("@id",
        Convert.ToInt32(((DataRowView)this.profilesBindingSource.Current).Row["id_profile"]));
        this.sqlDataAdapter1.DeleteCommand.Parameters.Add(id);
        try
        {
            this.sqlDataAdapter1.DeleteCommand.ExecuteNonQuery();
        }
        catch
        {
            this.label3.Text = "Произошел сбой";
            this.label3.ForeColor = Color.Red;
        }
    }

    private void button2_Click(object sender, EventArgs e)
    {
        SHACrypt sh = new SHACrypt();
        SqlCommand command = new SqlCommand();
        command.CommandType = CommandType.Text;
        string cmdtext_p1 = "UPDATE Profiles SET ";
        string cmdtext_p2 = " WHERE id_profile = @id_profile";
        var parameters = new List<String>();
        string cmdtext;
        if (!String.IsNullOrEmpty(textBox1.Text))
        {
            parameters.Add("hash=@hash");
            SqlParameter p1 = new SqlParameter("@hash", sh.SHA512M(this.textBox1.Text,
            ((DataRowView)this.profilesBindingSource.Current).Row["salt"].ToString()));
            command.Parameters.Add(p1);
        }
        SqlParameter p2;
        if (checkBox1.Checked)
        {
            parameters.Add("state=@state");
            p2 = new SqlParameter("@state", true);
            command.Parameters.Add(p2);
        }
        if (((DataRowView)this.profilesBindingSource.Current).Row["id_role"] == DBNull.Value ||
        ((DataRowView)this.profilesBindingSource.Current).Row["id_role"] != DBNull.Value)
        {
            parameters.Add("id_role=@id_role");
            SqlParameter p1 = new SqlParameter("@id_role", Convert.ToInt32(this.comboBox1.SelectedValue));
            command.Parameters.Add(p1);
        }
        if (parameters.Count == 1)
        {
            cmdtext = cmdtext_p1 + parameters.First() + cmdtext_p2;
        }
        else if (parameters.Count > 1)
        {
            cmdtext = cmdtext_p1 + String.Join(", ", parameters) + cmdtext_p2;
        }
        else
        {
            cmdtext = "";
            this.label3.Text = "Произошёл сбой при поиске информации";
        }
        SqlParameter prof = new SqlParameter("id_profile",
        Convert.ToInt32(((DataRowView)this.profilesBindingSource.Current).Row["id_profile"]));
        command.Parameters.Add(prof);
        command.CommandText = cmdtext;
        command.Connection = this.sqlConnection1;
        try
    }

```

```

        {
            command.ExecuteNonQuery();
        }
        catch
        {
            this.label3.Text = "Произошел сбой";
            this.label3.ForeColor = Color.Red;
        }
    }
    private void button3_Click(object sender, EventArgs e)
    {
        //фильтрация
        var filters = new List<String>();
        if (!String.IsNullOrEmpty(textBox2.Text))
        {
            filters.Add(String.Format("fname LIKE '{0}%',", textBox2.Text));
        }
        if (!String.IsNullOrEmpty(textBox3.Text))
        {
            filters.Add(String.Format("lname LIKE '{0}%',", textBox3.Text));
        }
        if (!String.IsNullOrEmpty(textBox4.Text))
        {
            filters.Add(String.Format("mname LIKE '{0}%',", textBox4.Text));
        }
        if (!String.IsNullOrEmpty(textBox5.Text))
        {
            filters.Add(String.Format("login LIKE '{0}%',", textBox5.Text));
        }
        try
        {
            if (filters.Count == 1)
            {
                this.profilesBindingSource.Filter = filters.First();
            }
            else if (filters.Count > 1)
            {
                this.profilesBindingSource.Filter = String.Join(" AND ", filters);
            }
            else
            {
                this.profilesBindingSource.RemoveFilter();
            }
        }
        catch
        {
            this.label3.Text = "Произошёл сбой при поиске информации";
            this.label3.ForeColor = Color.Red;
        }
    }
}

```