МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное

образовательное учреждение высшего образования

«Самарский национальный исследовательский университет

имени академика С.П. Королева»

(Самарский университет)

Институт информатики, математики и электроники

Факультет информатики

Кафедра информационных систем и технологий

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

«ИССЛЕДОВАНИЕ ХАРАКТЕРИСТИК И РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ СИСТЕМЫ ДИСТАНЦИОННОГО СНЯТИЯ ПОКАЗАНИЙ ПРИБОРОВ УЧЁТА ЭЛЕКТРОЭНЕРГИИ»

по направлению подготовки 09.03.01 Информатика

и вычислительная техника

(уровень бакалавриата)

профиль «Информационные системы»

Студент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_М.А. Гуреев

(подпись, дата)

Руководитель ВКР,

к.т.н.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Н.Г. Крупец

(подпись, дата)

Нормоконтролер\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.А. Столбова

(подпись, дата)

Самара 2019

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное

образовательное учреждение высшего образования

«Самарский национальный исследовательский университет

имени академика С.П. Королева»

(Самарский университет)

Институт информатики, математики и электроники

Кафедра информационных систем и технологий

УТВЕРЖДАЮ

Заведующий кафедрой ИСТ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_С.А. Прохоров

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г.

ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ БАКАЛАВРА

студенту Гурееву Михаилу Андреевичу

группа № 6401-090301D

Тема работы: «Исследование характеристик и разработка программного обеспечения системы дистанционного снятия показаний приборов учёта электроэнергии»

утверждена приказом по университету №451-ст от «01» апреля 2019г.

Исходные данные к работе:

* справочные данные по принципу работы автоматизированных систем коммерческого учёта электроэнергии;
* тип операционной системы: Windows;
* язык программирования: C#;
* редактор кода: Visual Studio 2017;
* система управления реляционными базами данных MS SQL Server;
* тип ЭВМ – IBM PC совместимый;

Перечень вопросов, подлежащих разработке в работе:

1. описание предметной области и постановка задачи;
2. анализ существующих методов решения задачи и обоснование выбора реализуемого метода;
3. обзор имеющихся систем-аналогов;
4. описание информационно-логической модели системы и логической модели базы данных;
5. обоснование и выбор комплекса программных средств, языка программирования, среды разработки, системы управления базой данных, операционной системы;
6. описание физической и логической моделей базы данных;
7. разработка алгоритмов обработки данных;
8. разработка контрольного примера обработки реальных данных или разработка плана исследований эффективности разработанной программной системы на модельных или реальных данных;
9. проверка эффективности разработанной программной системы для решения поставленной задачи;
10. исследование характеристик и анализ производительности системы.

Руководитель работы

доцент кафедры ИСТ Н.Г. Крупец

(подпись)

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_\_\_г.

Задание принял к исполнению М.А. Гуреев

(подпись)

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_\_\_г.

РЕФЕРАТ

Пояснительная записка 48 с, 18 рисунков, 10 таблиц, 13 источников, 2 приложения.

Графическая часть: 20 слайдов Microsoft PowerPoint.

АВТОМАТИЗИРОВАННАЯ СИСТЕМА КОММЕРЧЕСКОГО УЧЁТА ЭЛЕКТРОЭНЕРГИИ, БАЗА ДАННЫХ, МОДЕЛИРОВАНИЕ ОПРОСА ПРИБОРОВ УЧЁТА.

Целью данной выпускной работы бакалавра является разработка, программного обеспечения для моделирования дистанционного опроса и снятия показаний электроэнергии, а также исследование характеристик и производительности полученной системы при работе с большим объёмом данных.

В процессе работы был проведён анализ предметной области и разработана логическая модель системы на основе методологии UML. Диаграммы вариантов использования, деятельности, состояний и классов разрабатывались в StarUML. Также разработаны логическая и физическая структуры базы данных в ERwin Data Modeler.

Пользовательская клиентская часть приложения была разработана на языке на языке C# в редакторе Visual Studio 2017. Серверная часть приложения спроектирована и реализована для сервера MS SQL.

СОДЕРЖАНИЕ

[Введение 7](#_Toc10484135)

[1 Постановка задачи и обоснование метода решения задачи 8](#_Toc10484136)

[1.1 Описание предметной области и постановка задачи 8](#_Toc10484137)

[1.2 Обзор имеющихся систем-аналогов 8](#_Toc10484138)

[1.2.1 АИИС КУЭ «Телескоп+» 8](#_Toc10484139)

[1.2.2 АИИС КУЭ «Матрица» 9](#_Toc10484140)

[1.2.3 АИИС КУЭ «Энергомера» 12](#_Toc10484141)

[1.2.4 АИИС КУЭ «Меркурий‑Энергоучёт» 13](#_Toc10484142)

[2 Проектирование и программная реализация метода решения задачи 15](#_Toc10484143)

[2.1 Анализ существующих методов решения задачи и обоснование выбора реализуемого метода 15](#_Toc10484144)

[2.2 Описание информационно-логической модели системы и логической модели базы данных 18](#_Toc10484145)

[2.2.1 Информационно-логическая модель базы данных 18](#_Toc10484146)

[2.2.2 Разработка логической модели данных 20](#_Toc10484147)

[2.2.3 Физическая модель базы данных 22](#_Toc10484148)

[2.2.4 Диаграмма вариантов использования 28](#_Toc10484149)

[2.2.5 Сценарии разрабатываемой системы 29](#_Toc10484150)

[2.2.6 Диаграмма состояний 31](#_Toc10484151)

[2.2.7 Диаграмма деятельности 33](#_Toc10484152)

[2.2.8 Диаграмма классов 33](#_Toc10484153)

[2.3 Обоснование и выбор комплекса программных средств, языка программирования, среды разработки, системы управления базой данных, операционной системы. 33](#_Toc10484154)

[2.3.1 Выбор средства реализации базы данных 33](#_Toc10484155)

[2.3.2 Выбор средства разработки 36](#_Toc10484156)

[2.3.4 Выбор технологии доступа к базе данных 37](#_Toc10484157)

[2.3.5 Разработка алгоритмов работы системы 37](#_Toc10484158)

[3 Экспериментальная проверка эффективности программной реализации метода решения задачи 39](#_Toc10484159)

[3.1. Разработка контрольного примера обработки данных 39](#_Toc10484160)

[3.2. Проверка эффективности разработанной программной системы для решения поставленной задачи 40](#_Toc10484161)

[Заключение 43](#_Toc10484163)

[Список используемых источников 44](#_Toc10484164)

[Приложение А Руководство пользователя 46](#_Toc10484165)

[Приложение Б Листинг программы 49](#_Toc10484166)

ВВЕДЕНИЕ

В настоящее время актуальным является вопрос об учёте и анализе потребления энергоресурсов. Наиболее перспективным считается организация учета на базе автоматизированных систем коммерческого учета электроэнергии (АСКУЭ) [1].

Основной целью внедряемых АСКУЭ многоквартирных жилых домов (МКЖД) является автоматизация процесса снятия показаний общедомовых и квартирных приборов учета для выставления счетов за потребленную электроэнергию каждой квартирой и расчета потребления электроэнергии, затраченной на общедомовые нужды (ОДН) – освещение подъездов, лифты, насосы водоснабжения и проч.

Наличие такой системы для энергосбытовой организации является первостепенной и необходимой задачей. Также не менее важным представляется вопрос настройки и оптимизации её работы. Это связано с тем, что на сервер поступает огромное количество данных, ежедневно ведётся и заполняется огромное количество таблиц, которые, в конечном счёте, нужно обработать и проанализировать. Подобная проблема решается оптимизацией поисковых запросов к серверу или внедрением эффективной индексации для ряда используемых таблиц. Производительность индексов может сильно между собой различаться, что и является предметом исследования в данной работе.

Таким образом, можно сказать, что разработка моделирующей системы дистанционного снятия показаний электроэнергии, а также анализ характеристик её производительности являются актуальным вопросом, не только в данной сфере, но и в других смежных, занимающихся поиском нужной информации из большого объёма данных.

# 1 Постановка задачи и обоснование метода решения задачи

## 1.1 Описание предметной области и постановка задачи

Задача разрабатываемой системы заключается в создании базы данных для АСКУЭ, а точнее для учёта электроэнергии.

База данных представляет собой сервер, доступ к которому осуществляется через разрабатываемое клиентское приложение, через локальное подключение, сеть интернет и иные способы подключения.

База данных спроектирована по реляционной модели, так как она является наиболее распространённой и поддерживаемой большинством разработчиков на сегодняшний день. Реляционные системы управления базой данных (СУБД) отличаются наиболее эффективным способом представления и хранения данных (представлены в виде таблиц), необходимыми типами связи между таблицами (один к одному, один ко многим, многие ко многим), доступным и распространённым программным обеспечением, таким как MS SQL Server, PostgeSQL, MySQL и другими, с доступной документацией, своевременной поддержкой и обновлением.

1.2 Обзор имеющихся систем-аналогов

Системами аналогами является ряд крупных автоматизированных систем коммерческого учёта электроэнергии (АСКУЭ). Как правило, они предоставляют к своему программному обеспечению аппаратную систему в виде счётчиков, устройств сбора и передачи данных (УСПД), автоматизированных рабочих мест и прочего необходимого оборудования.

**1.2.1 АИИС КУЭ «Телескоп+»**

**Аппартно–программные комплексы для учёта энергоресурсов предназначенные для измерений и удалённого опроса электроэнергии, природного газа, воды, пара и тепловой энергии в режиме реального времени, хранения показаний счётчиков, формирование многообразным форм отчётов о потреблении электроэнергии, газа, воды пара, тепловой электроэнергии, а также для создания многоуровневых автоматизированных систем контроля и управления энергопотреблением в энергосистемах предприятий и ЖКХ [2].**

**Иерархическую многоуровневую структуру системы Телескоп+ можно настраивать в соответствии с требованиями заказчика. Объектами нижнего уровня в иерархической структуре являются терминальные контроллеры и периферийное оборудование, обслуживающее датчики, интеллектуальные устройства и счетчики электроэнергии. На верхнем уровне иерархической структуры размещена система серверов, база данных и автоматизированными рабочими местами (АРМы) [2].**

**На рисунке 1 показан интерфейс АИИС КУЭ «Телескоп+»**

**1.2.2 АИИС КУЭ «Матрица»**

**АИИС КУЭ «Матрица»** [3] **— это автоматизированная информационно-измерительная система на базе оборудования производства ООО "Матрица", ориентированная на решение широкого спектра задач:**

* **дистанционный учет потребления различных видов ресурсов (электроэнергии, газа, воды, тепла);**
* **программное и/или дистанционное управление потреблением электроэнергии;**
* **управление уличным освещением.**

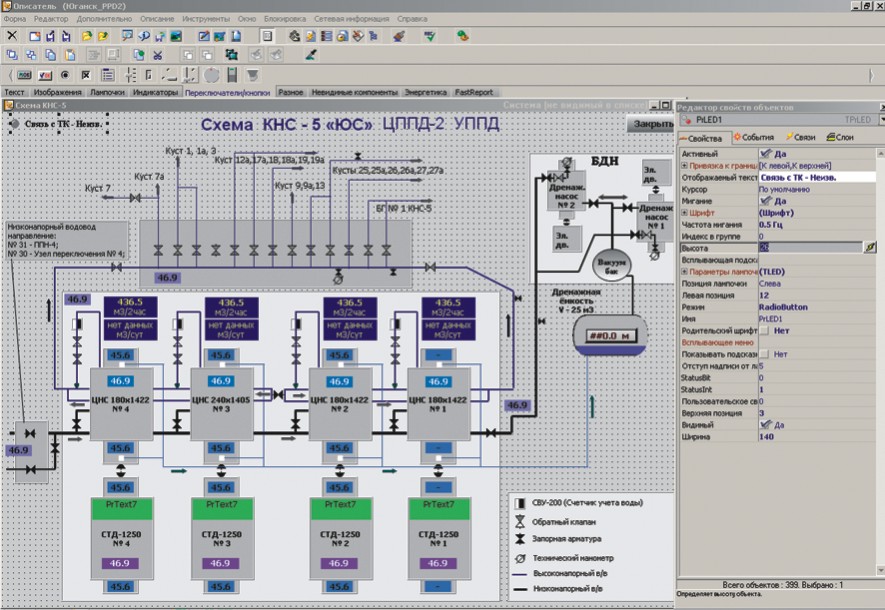
АИИС "Матрица" имеет простую трехуровневую структуру.

Нижний уровень составляют приборы учета электроэнергии и другие абонентские устройства;

Средний уровень состоит из маршрутизатора (УСПД) и распределенной сети передачи данных.

Верхний уровень представляет собой серверный центр, осуществляющий сбор, хранение и обработку данных.

На рисунке 2 приведена иллюстрация схемы построения АИИС КУЭ «Матрица».



**Рисунок 1 – Интерфейс АИИС КУЭ «Телескоп+»**



**Рисунок 2 – Схема построения АИИС КУЭ «Матрица»**

**1.2.3 АИИС КУЭ «Энергомера»**

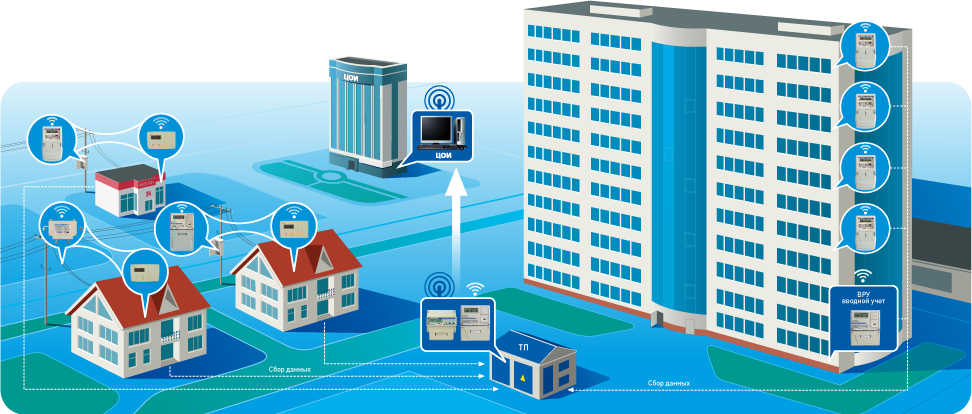
**Информационно-измерительные системы (ИИС) «Энергомера» [4] предназначена для контроля и коммерческого учета электроэнергии и мощности, автоматизированного сбора, хранения и обработки и отображения данных об энергопотреблении.**

**ИИС «Энергомера» применяется на энергетических объектах розничного рынка электроэнергии, промышленных предприятиях, коммунально-бытовом хозяйстве и мелкомоторном секторе.**

**Функции системы:**

1. **Измерение и многотарифный учет активной и реактивной электрической энергии и мощности**
2. **Измерение параметров сети и диагностической информации, с информированием о внештатных ситуациях**
3. **Управление нагрузкой**
4. **Удаленное конфигурирование приборов учета**
5. **Установку и синхронизацию времени на всех уровнях системы**
6. **Хранение измеренных данных в базе**
7. **Аналитическую обработку собранных данных и расчет небалансов**
8. **Визуальное предоставление данных и генерацию отчетных форм**
9. **Защиту результатов измерений о несанкционированного доступа**
10. **Обмен данными со сторонними системами коммерческого учета, биллинговыми системами и системами телемеханики**

**На рисунке 3 показана структурная схема работы в сфере жилищно‑коммунального хозяйства (ЖКХ).**



**Рисунок 3 – Структурная схема работы АИИС КУЭ «Энергомера»**

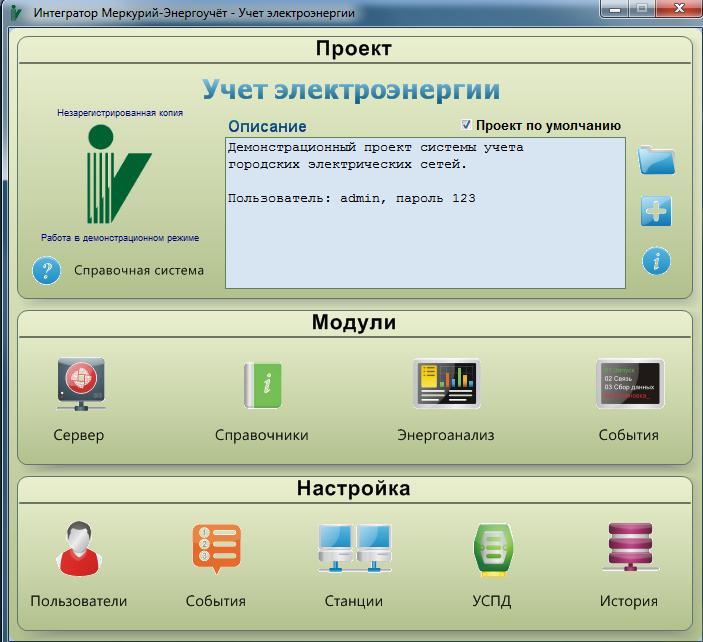
**1.2.4 АИИС КУЭ «Меркурий‑Энергоучёт»**

**«Меркурий‑Энергоучёт» [5] — это система для коммерческого и технического учета энергоресурсов, диспетчеризации, телемеханики, задач учета и диспетчеризации объектов энергетики, промышленности, ЖКХ и зданий. Современный, инновационный, мощный и удобный инструмент для быстрого и качественного внедрения систем автоматизации.**

**При создании МЭ большое внимание уделялось возможности оптимального и быстрого построения проектов автоматизации с большим числом объектов, в том числе и распределенных, и большим числом параметров на каждый объект. Типичная система автоматизации - это учет энергоресурсов, диспетчеризация, телемеханика сотен и тысяч объектов городского хозяйства - электрические и тепловые сети, водоканал, многоквартирные дома. Интерфейс АСКУЭ показан на рисунке 4.**

**Функции:**

* **сбор и регистрация первичной информации о ходе технологического процесса;**
* **обработка информации по алгоритмам пользователя;**
* **предоставление информации в виде мнемосхем технологического процесса;**
* **оперативное, диспетчерское управление;**
* **ведение истории технологического процесса;**
* **просмотр и анализ хода технологического процесса;**
* **формирование отчетной документации;**
* **экспорт оперативной и исторической информации в WEB;**
* **сигнализация и регистрация событий и нарушений в ходе технологического процесса;**
* **регистрация всех действий операторов;**
* **механизм настройки прав пользователей.**



**Рисунок 4 – Интерфейс программы «Меркурий‑Энергоучёт»**

2 Проектирование и программная реализация метода решения задачи

2.1 Анализ существующих методов решения задачи и обоснование выбора реализуемого метода

Построение автоматизированной информационной системы обработки информации является сложной инженерной задачей. Для её решения необходимо провести системный анализ объекта автоматизации, выбрать наиболее эффективное архитектурное решение и формализовать его в комплекте проектной документации, организовать проект по разработке необходимого алгоритмического, технического, программного и информационного обеспечения и реализовать управление этим проектом в течение всего его жизненного цикла.

Для обеспечения адекватности модели и её высокой степени формализации, которая требуется при использовании этой модели для описания технического решения, необходимо использовать современные технологии автоматизированного проектирования.

В данной работе будет использоваться UML (UnifiedModelingLanguage) методология объектно-ориентированного проектирования, описанная стандартом UML 2.0.

Создание UML началось в октябре 1994 г., когда Джим Рамбо и Гради Буч из Rational Software Corporation стали работать над объединением своих методов ОМТ и Booch. Осенью 1995 г. увидела свет первая черновая версия объединенной методологии, которую они назвали Unified Method 0.8. После присоединения в конце 1995 г. к Rational Software Corporation Айвара Якобсона и его фирмы Objectory, усилия трех создателей наиболее распространенных объектно-ориентированных методологий были объединены и направлены на создание UML.

В настоящее время консорциум пользователей UML Partners включает в себя представителей таких грандов информационных технологий, как Microsoft, IBM, Hewlett-Packard, Oracle, DEC, Unisys, IntelliCorp, Platinum Technology.

UML представляет собой объектно-ориентированный язык моделирования, обладающий следующими основными характеристиками:

* является языком визуального моделирования, который обеспечивает разработку репрезентативных моделей для организации взаимодействия заказчика и разработчика информационных систем (ИС), различных групп разработчиков ИС;
* содержит механизмы расширения и специализации базовых концепций языка.

UML – это стандартная нотация визуального моделирования программных систем, принятая консорциумом Object Managing Group (OMG) осенью 1997 г., и на сегодняшний день она поддерживается многими объектно-ориентированными CASE-продуктами. UML включает внутренний набор средств моделирования, которые сейчас приняты во многих методах и средствах моделирования. Эти концепции необходимы в большинстве прикладных задач, хотя не каждая концепция необходима в каждой части каждого приложения.

Пользователям языка предоставлены возможности:

* строить модели на основе средств ядра, без использования механизмов расширения для большинства типовых приложений;
* добавлять при необходимости новые элементы и условные обозначения, если они не входят в ядро, или специализировать компоненты, систему условных обозначений (нотацию) и ограничения для конкретных предметных областей.

Формальная спецификация версии UML 2.0 опубликована в августе 2005 года. Семантика языка была значительно уточнена и расширена для поддержки методологии Model Driven Development — [MDD](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D1%8F%D0%B5%D0%BC%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8F%D0%BC%D0%B8).

Несмотря на свою молодость, UML уже прекрасно зарекомендовал себя на множестве успешных программных проектов. Средства автоматической кодогенерации позволяют переводить модели на языке UML в исходный код объектно-ориентированных языков программирования, что ещё более ускоряет процесс разработки.

UML широко распространен, благодаря ряду преимуществ:

* UML объектно-ориентирован, в результате чего методы описания результатов анализа и проектирования семантически близки к методам [программирования](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) на современных [объектно-ориентированных языках](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F);
* UML позволяет описать систему практически со всех возможных точек зрения и разные аспекты поведения системы;
* диаграммы UML сравнительно просты для чтения после достаточно быстрого ознакомления с его синтаксисом;
* UML расширяет и позволяет вводить собственные текстовые и графические [стереотипы](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D1%80%D0%B5%D0%BE%D1%82%D0%B8%D0%BF_(UML)), что способствует его применению не только в сфере программной инженерии;
* UML динамично развивается.

Практически все мировые производители CASE-средств заявили о реализации поддержки UML в ближайших версиях своих продуктов. Но уже сегодня существуют множество CASE-средств, автоматизирующих процесс анализа и проектирования в UML (Rational Rose, Paradigm Plus, Select Enterprise, Microsoft Visual Modeler for Visual Basic и др.), поддерживающих множество языков программирования, таких, как C++, Java, Delphi, Power Builder, Visual Basic, Centura, Forte, Ada, Smalltalk, а также позволяющих осуществлять генерацию базы данных для большинства из существующих SQL-серверов [6].

При проектировании базы данных возникает необходимость разработки схемы или модели разрабатываемой системы, которая отражала бы связи и особенности используемых данных.

Методы структурного проектирования помогают упростить процесс разработки сложных систем за счет использования алгоритмов как готовых строительных блоков. Аналогично, методы объектно-ориентированного проектирования созданы, чтобы помочь разработчикам применять мощные выразительные средства объектного и объектно-ориентированного программирования, использующего в качестве блоков классы и объекты.

Объектный подход зарекомендовал себя как унифицирующая идея всей компьютерной науки, применимая не только в программировании, но также в проектировании интерфейса пользователя, баз данных и даже архитектуры компьютеров. Причина такой широты в том, что ориентация на объекты позволяет нам справляться со сложностью систем самой разной природы.

Объектно-ориентированное проектирование - это методология проектирования, соединяющая в себе процесс объектной декомпозиции и приемы представления логической и физической, а также статической и динамической моделей проектируемой системы. В настоящее время объектно-ориентированное проектирование - единственная методология, позволяющая справиться со сложностью, присущей очень большим системам. Однако следует заметить, что иногда применение OOD может оказаться нецелесообразным, например, из-за неподготовленности персонала или отсутствия подходящих средств разработки [7].

2.2 Описание информационно-логической модели системы и логической модели базы данных

2.2.1 Информационно-логическая модель базы данных

Информационно-логическая модель отображает данные предметной области в виде совокупности информационных объектов и связей между ними. Эта модель представляет данные, подлежащие хранению в базе данных. Каждый информационный объект в модели данных должен иметь уникальное имя.

Информационный объект — это информационное описание некоторой сущности предметной области; реального объекта, процесса, явления или события. Информационный объект является совокупностью логически взаимосвязанных реквизитов, представляющих качественные и количественные характеристики сущности.

Информационный объект имеет множество реализаций - экземпляров объекта. Экземпляр объекта должен однозначно определяться среди всего множества экземпляров, т.е. идентифицироваться значением уникального (первичного) ключа информационного объекта. Уникальность ключа означает, что любое значение ключа не может повториться в каком-либо другом; экземпляре объекта. Простои ключ состоит из одного реквизита. Составной ключ — из нескольких реквизитов. Таким образом, реквизиты информационного объекта подразделяются на ключевые и описательные, которые являются функционально зависимыми от ключа. Необходимость установления функциональных зависимостей связана с требованием баз данных по однозначной определяемости любых данных для их размещения и доступа к ним.

Модели баз данных:

* иерархическая (древовидная) модель данных представляет собой иерархию элементов, называемых узлами. Узел — это совокупность атрибутов данных, описывающих информационный объект. На самом верхнем уровне имеется только один узел — корень. Каждый узел кроме корня связан только с одним узлом на более высоком уровне, называемом исходным узлом для данного узла. Каждый узел может быть связан с одним или несколькими узлами более низкого уровня, называемыми порожденными (подчиненными). Узлы, не имеющие порожденных, называются листьями. Количество деревьев в базе данных определяется числом корневых записей. К каждой записи базы данных существует только один путь;
* сетевая модель также основывается на понятиях узел, уровень, связь. Сетевая модель данных — это модель, в которой порожденный узел может иметь более одного исходного узла. В сетевой структуре любой элемент любого уровня может быть связан с любым другим элементом;
* реляционная модель базы данных состоит из одного или нескольких файлов, каждый из которых соответствует одной таблице[8].

Главной особенностью реляционных баз данных является, то, что объекты внутри таких баз данных хранятся в виде набора двумерных таблиц. То есть, таблица состоит из набора столбцов, в котором может указываться: название, тип данных (дата, число, строка, текст и т.д.). Еще одной важной особенностью реляционных БД является, то, что число столбцов фиксировано, то есть, структура базы данных известна заранее, а вот число строк или рядов в реляционных базах данных ничем не ограничено, если говорить грубо, то строки в реляционных базах данных и есть объекты, которые хранятся в базе данных.

2.2.2 Разработка логической модели данных

Целью построения логической модели является получение графического представления логической структуры исследуемой предметной области.

Объекты модели, представляемые на логическом уровне, называются сущностями и атрибутами. Логическая модель данных может быть построена на основе другой логической модели, например, на основе модели процессов. Логическая модель данных является универсальной и никак не связана с конкретной реализацией СУБД.

Логическая модель предметной области иллюстрирует сущности, а также их взаимоотношения между собой.

Сущности описывают объекты, являющиеся предметом деятельности предметной области, и субъекты, осуществляющие деятельность в рамках предметной области. Свойства объектов и субъектов реального мира описываются с помощью атрибутов.

Взаимоотношения между сущностями иллюстрируются с помощью связей. Правила и ограничения взаимоотношений описываются с помощью свойств связей. Обычно связи определяют либо зависимости между сущностями, либо влияние одной сущности на другую.

Основные требования к содержанию модели:

* логическая модель должна отображать все сущности и связи, значимые для той цели, ради которой мы ее рисуем;
* все объекты модели (и сущности, и связи) должны быть именованы. Именование сущностей и связей должно выполняться в терминах предметной области;
* для связей должна быть указана кратность (один — многие);
* для каждой связи должно быть указано направление чтения.
* Различают три уровня логической модели, отличающихся по глубине представления информации о данных:
* диаграмм сущность-связь (Entity Relationship Diagram, ERD);
* модель данных, основанная на ключах (Key Based model, KB);
* полная атрибутная модель (Fully Attributed model, FA).

Диаграмма сущность-связь представляет собой модель данных верхнего уровня. Она включает сущности и взаимосвязи, отражающие основные бизнес-правила предметной области. Такая диаграмма не слишком детализирована, в нее включаются основные сущности и связи между ними, которые удовлетворяют основным требованиям, предъявляемым к ИС.

Диаграмма сущность-связь может включать связи многие-ко-многим и не включать описание ключей. Как правило, ERD используется для презентаций и обсуждения структуры данных с экспертами предметной области.

Модель данных, основанная на ключах, отражает более подробное представление данных. Она включает описание всех сущностей и первичных ключей и предназначена для представления структуры данных и ключей, которые соответствуют предметной области.

Полная атрибутивная модель – наиболее детальное представление структуры данных: представляет данные в третьей нормальной форме и включает все сущности, атрибуты и связи.

Атрибут выражает свойство объекта, характеризующее его экземпляр (определенное свойство объекта). С точки зрения БД (физическая модель) сущности соответствует таблица, экземпляру сущности – строка в таблице, а атрибуту – колонка таблицы.

Первичный ключ – это атрибут или набор атрибутов, уникально идентифицирующий экземпляр сущности. Если несколько наборов атрибутов могут уникально идентифицировать сущность, то выбор одного из них осуществляется разработчиком на основании анализа предметной области.

Если между некоторыми сущностями существует связь, то факты из одной сущности ссылаются или некоторым образом связаны с фактами из другой сущности.

Связь – это функциональная зависимость между сущностями. Поддержание непротиворечивости функциональных зависимостей между сущностями называется ссылочной целостностью. Поскольку связи содержатся "внутри" реляционной модели, реализация ссылочной целостности может выполняться как приложением, так и самой СУБД (с помощью механизмов декларативной ссылочной целостности, триггеров). Связь является понятием логического уровня, которому соответствует внешний ключ на физическом уровне. Связь называется идентифицирующей, если экземпляр дочерней сущности идентифицируется через ее связь с родительской сущностью. Атрибуты, составляющие первичный ключ родительской сущности, при этом входят в первичный ключ дочерней сущности. Дочерняя сущность при идентифицирующей связи всегда является зависимой. Связь называется не идентифицирующей, если экземпляр дочерней сущности идентифицируется иначе, чем через связь с родительской сущностью. Атрибуты, составляющие первичный ключ родительской сущности, при этом входят в состав не ключевых атрибутов дочерней сущности [9].

Опираясь на теорию реляционных баз дынных, разработаем логическую модель автоматизированной информационной системы «Генератор базы данных АСКУЭ». Данная модель представлена на рисунке 5.

2.2.3 Физическая модель базы данных

При описании логической модели базы данных используются такие термины, как "сущность", "экземпляр", "связь", "атрибут". При переходе от логической к физической модели реляционной БД нужно будет работать с таблицами, записями, полями и прочими объектами, и ограничениями, поддерживаемыми выбранной СУБД. При этом для построения диаграмм для физической модели можно использовать нотации IDEF1X. Данная модель представлена на рисунке 6.

В таблицах 1 – 9 приведена физическая структура базы данных, в которых в соотнесены таблицы физической модели с отношениями логической модели.

Таблица 1 – Таблица sHouse

|  |  |  |
| --- | --- | --- |
| Имя стобца | Описание | Тип данных |
| IdHouse(PK) | id дома | bigint |
| NumFlat | число квартир в доме | int |
| IdStreet | id города | int |
| IdTown | id города | int |
| NumUSPD | номер УСПД | int |

Таблица 2 – Таблица sFlat

|  |  |  |
| --- | --- | --- |
| Имя столбца | Описание | Тип данных |
| IdFlat(PK) | id квартиры | bigint |
| IdPoint(FK) | id точки учёта | bigint |
| IdHouse(FK) | id дома | bigint |
| Numer | номер квартиры | int |
| Comment | комментарий | varchar |

Таблица 3 – rTypeMeter

|  |  |  |
| --- | --- | --- |
| Имя столбца | Описание | Тип данных |
| IdTypeMeter(PK) | id типа счётчика | int |
| ManufacturerMeter | производитель счётчика | varchar |
| NameTypeMeter | название типа счётчика | varchar |

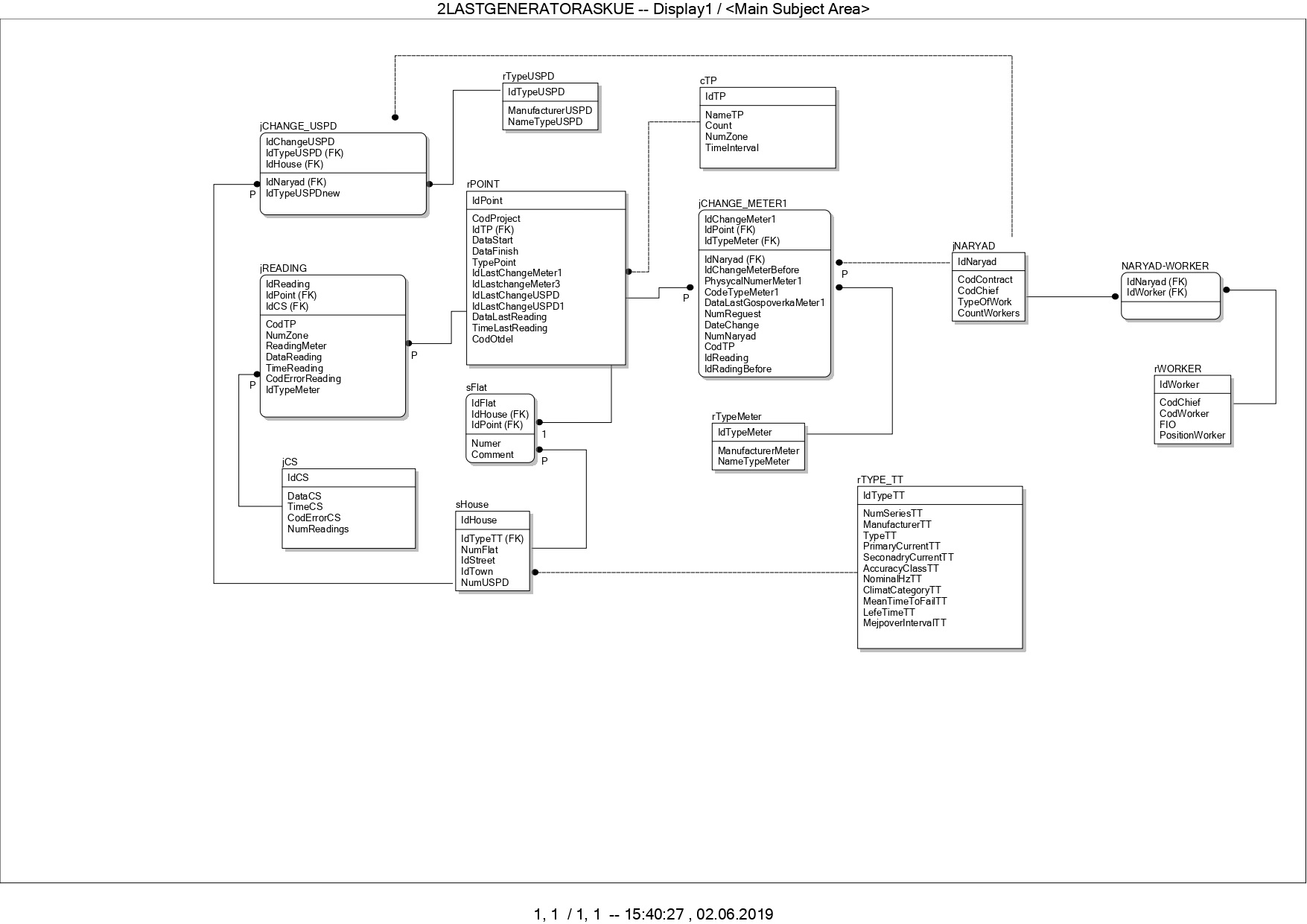


Рисунок 5 – Логическая модель системы

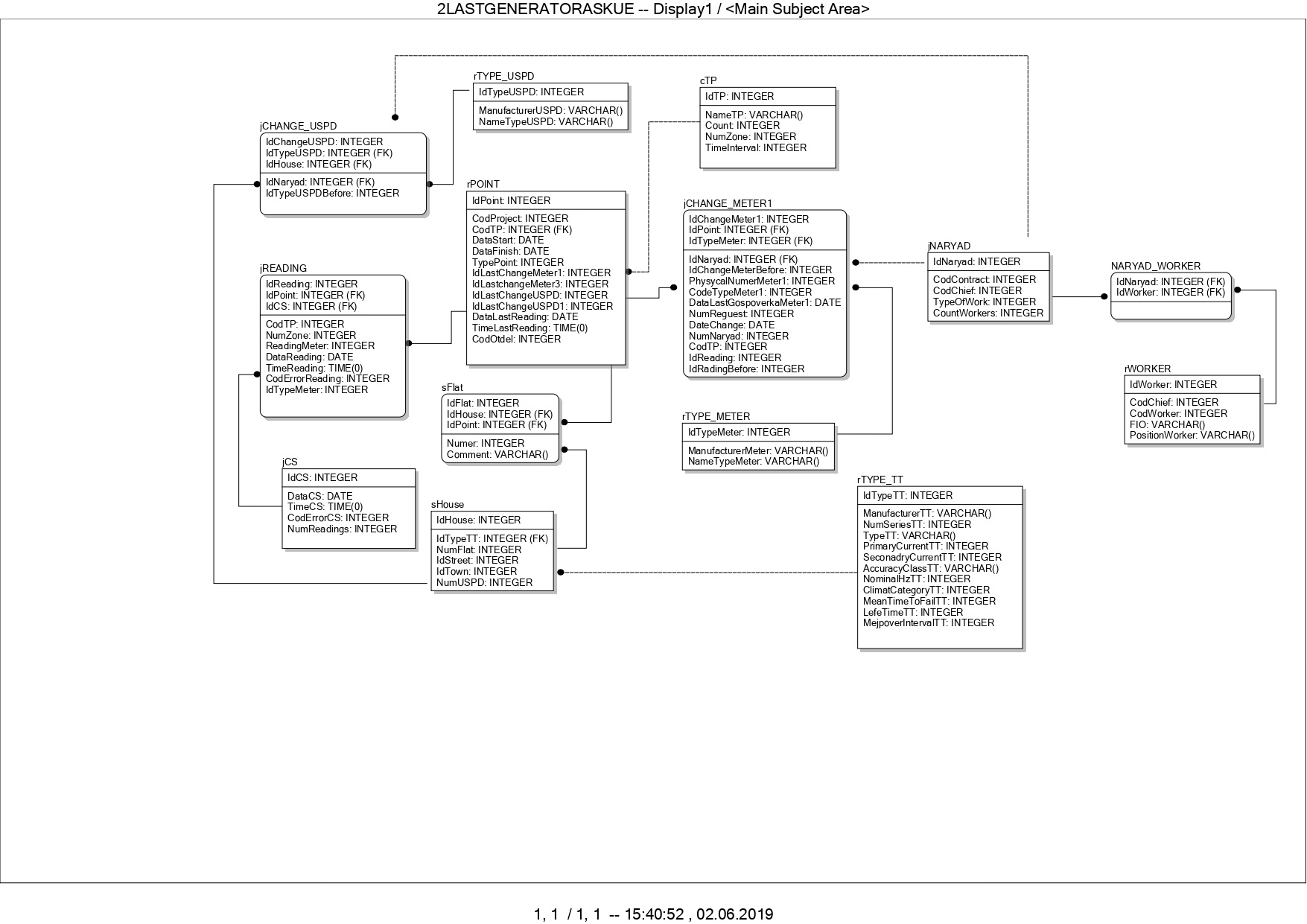


Рисунок 6 – Физическая модель базы данных

Таблица 4 – Таблица rPoint

|  |  |  |
| --- | --- | --- |
| Имя стобца | Описание | Тип данных |
| IdPoint(PK) | id точки | bigint |
| DataStart | дата установки | date |
| DataFinish | дата отключения | date |
| TypePoint | тип точки учёта | int |
| IdLastChangeMeter1 | id последнего установленного однофазного счётчика | int |
| IdLastChangeMeter3 | id последнего установленного трёхфазного счётчика | int |
| IdLastChangeUSPD | id последнего установленного УСПД | int |
| IdLastChangeUSPD1 | id второго последнего установленного УСПД | int |
| DataLastReading | дата последнего ручного снятия показаний | date |
| TimeLastReading | время последнего ручного снятия показаний | time(0) |
| CodOtdel | код отделения обслуживания | int |
| CodProject | код проекта установки | int |

Таблица 5 – Таблица jCHANGE\_USPD

|  |  |  |
| --- | --- | --- |
| Имя столбца | Описание | Тип данных |
| IdChangeUSPD(PK) | id замены УСПД | bigint |
| IdHouse(FK) | id дома | bigint |
| CodNaryada | код наряда установки/замены | int |
| IdNaryada | id наряда установки/замены | int |
| TypeOfWork | тип работ | int |
| CodChief | код бригадира | int |
| IdTypeUSPD | id типа УСПД | int |
| IdTypeUPSDnew | новый id типа УСПД при замене | int |

Таблица 6 – Таблица rTypeUSPD

|  |  |  |
| --- | --- | --- |
| Имя столбца | Описание | Тип данных |
| IdTypeUSPD(PK) | id типа УСПД | int |
| ManufacturerUSPD | производитель УСПД | varchar |
| NameTypeUSPD | название типа УСПД | varchar |

Таблица 7 – Таблица jCHANGE\_METER1

|  |  |  |
| --- | --- | --- |
| Имя столбца | Описание | Тип данных |
| IdCHANGE\_METER1(PK) | id последней замены/установки счётчика | bigint |
| IdPoint(FK) | id точки учёта | bigint |
| IdTypeMeter(FK) | id типа счётчика | int |
| IdChangeMeterBefore | id последнего установленного счётчика | int |
| PhusikalnumberMeter1 | заводской номер счётчика | int |
| CodetypeMeter1 | код типа счётчика | int |
| DataLastGospoverkaMeter1 | дата последней гос. поверки | date |
| NumRequest | номер заявки на замену | int |
| TypeChange | тип замены/установки | int |
| CodContract | код договора | int |
| CodWorker | код работника | int |
| CodChief | код бригадира | int |
| CodTP | код тарифного плана | int |
| IdReading | id начальные показания снятого счётчика | int |
| IdReadingbefore | id конечные показания снятого счётчика | int |

Таблица 8 – Таблица jCS

|  |  |  |
| --- | --- | --- |
| Имя столбца | Описание | Тип данных |
| 1 | 2 | 3 |
| IdCS(PK) | id сеанса связи | bigint |
| DataCS | дата сеанса связи | date |
| TimeCS | время сеанса связи | time(0) |
| CodeErrorCS | код ошибки | int |
| NumReadings | число считанных показаний | int |

Таблица 9 – Таблица jReading

|  |  |  |
| --- | --- | --- |
| Имя столбца | Описание | Тип данных |
| IdReading(PK) | id чтения показаний | bigint |
| IdPoint(FK) | id точки учёта | bigint |
| IdCS(FK) | id сеанса связи | bigint |
| CodTP | код тарифного плана | int |
| NumZone | номер времени суток | int |
| ReadingMeter | показания | int |
| DateReading | дата снятия показаний | date |
| TimeReading | время снятия показаний | time(0) |
| CodErrorReading | код ошибки при чтении | int |
| CodProject | код договора обслуживания | int |
| IdTypeMeter | id типа счётчика | int |

2.2.4 Диаграмма вариантов использования

На этапе проектирования автоматизированной системы электронного полиса ОМС были созданы диаграммы на языке моделирования UML, призванные упростить процесс разработки.

Язык UML представляет собой унифицированный язык визуального моделирования, который разработан для специфицирования, конструирования и визуализации программных систем – компонентов программного обеспечения, бизнес-процессов и т.д.

Диаграмма использования (use case diagram) ‒ это наиболее общее представление функционального назначения системы.

Диаграмма использования призвана ответить на главный вопрос моделирования: что делает система во внешнем мире.

На диаграмме использования применяются два типа основных сущностей: варианты использования и действующие лица, между которыми устанавливаются следующие основные типы отношений:

* ассоциация между действующим лицом и вариантом использования;
* обобщение между действующими лицами;
* обобщение между вариантами использования;
* зависимости (различных типов) между вариантами использования.

В приложении нет разграничений прав и все функции для работы с базой данных доступны одному пользователю, что показано диаграмме (рисунок 7).

2.2.5 Сценарии разрабатываемой системы

Вариант использования: Генерация отчёта.

Краткое описание. Пользователю доступна работа с базой данных и конфигурирование параметров для генерации.

Предусловия. Открыта главная форма на вкладке «Отчёты»

Основной поток событий.

1) Пользователь выбирает необходимую форму отчёта, заполняет параметры.

2) Пользователь нажимает клавишу «Создать отчёт».

3) Система создаёт txt файл с соответствующими записями из базы данных в папке с программой.

Вариант использования: Генерация справочных данных.

Предусловия. Открыта главная форма на вкладке «Справочные данные».

Основной поток событий.

1. Пользователь заполняет параметры для генерации.
2. В правой части формы будут доступны предварительные данные по генерации.

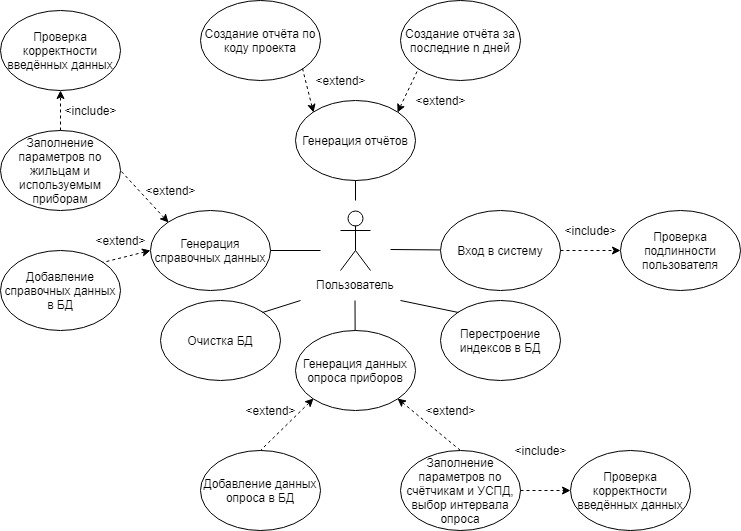


Рисунок 7 – Диаграмма вариантов использования

1. Пользователь нажимает клавишу «Сгенерировать данные».
2. По окончанию выполнения система выдаст уведомления о завершении.

Вариант использования: Генерация (моделирование) данных опроса приборов.

Предусловия. Открыта главная форма на вкладке «Опрос приборов».

Основной поток событий.

1. Пользователь заполняет параметры для генерации.
2. В правой части формы будут доступны предварительные данные по генерации.
3. Пользователь нажимает клавишу «Сгенерировать опрос».
4. По окончанию выполнения система выдаст уведомления о завершении.

2.2.6 Диаграмма состояний

На диаграмме состояний представлены состояния, в которых может находиться пользователь. Начально состояние характеризуется авторизацией пользователя с помощь логина и пароля к базе данных. Выход из программы доступен по нажатию соответствующей клавиши «Выход».

На рисунке 8 проиллюстрирована диаграмма состояний для разработанного приложения.

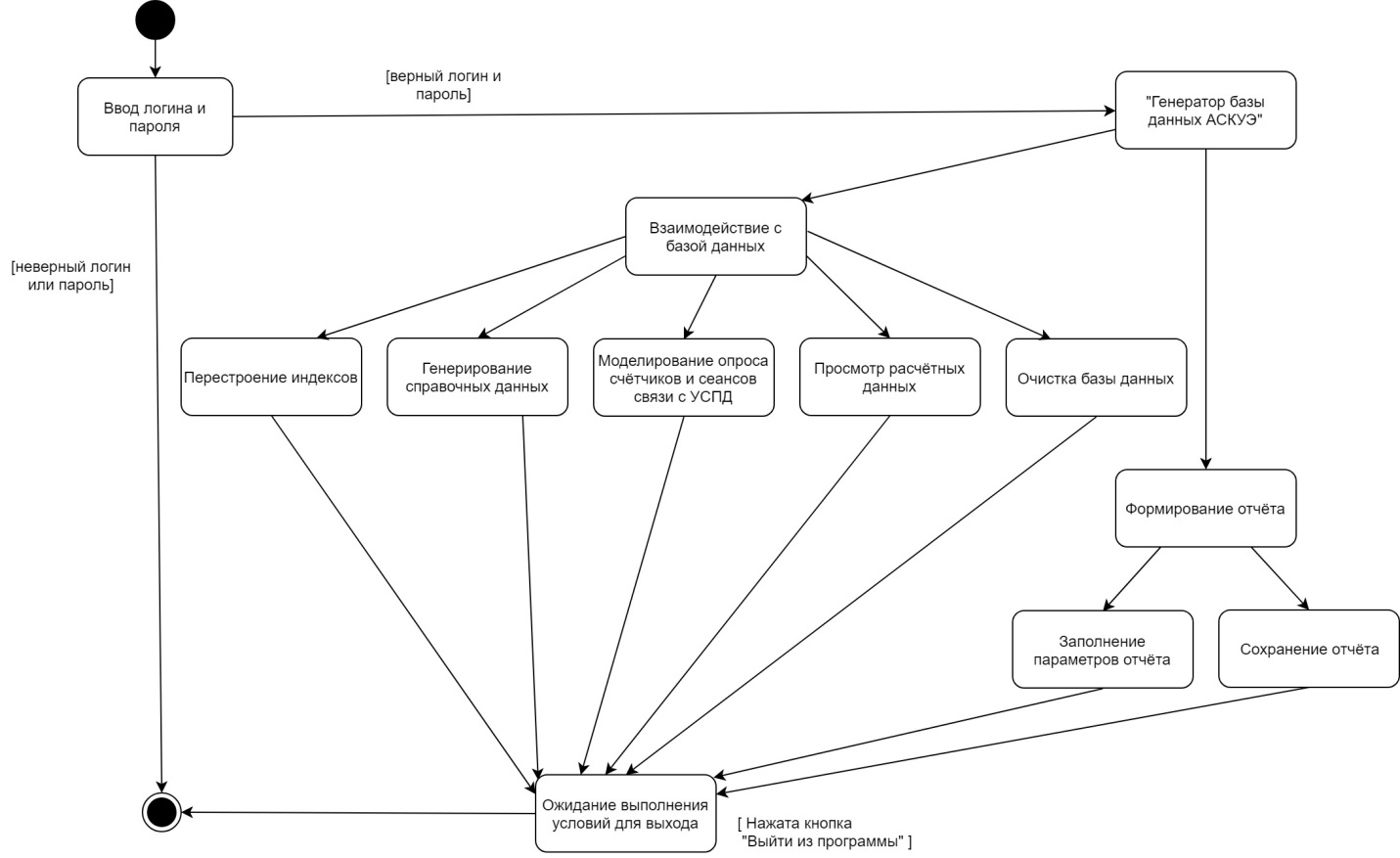


Рисунок 8 – Диаграмма состояний

2.2.7 Диаграмма деятельности

На диаграмме деятельности, помимо начального, конечного и основных состояний, показаны альтернативные переходы (обозначены ромбами). Над стрелками переходов надписаны условия выбора той или иной альтернативы.

Диаграмма создана для варианта использования «Генерация справочных данных» и показана на рисунке 9.

2.2.8 Диаграмма классов

Диаграмма классов служит для представления статической структурной модели системы в терминологии классов объектно-ориентированного проектирования. Диаграмма классов может отражать, в частности, различные взаимосвязи между отдельными сущностями предметной области, а также описывает их внутреннюю структуру и типы отношений. При этом на данной диаграмме не указывается информация о временных аспектах функционирования системы.

Класс в языке UML служит для обозначения множества объектов, которые обладают одинаковой структурой, поведением и отношениями с объектами из других классов. Описание класса состоит в определении атрибутов (свойств) и методов (операций или сервисов). Иллюстрация диаграммы классов представлена на рисунке 10.

2.3 Обоснование и выбор комплекса программных средств, языка программирования, среды разработки, системы управления базой данных, операционной системы.

2.3.1 Выбор средства реализации базы данных

Для разработки программного продукта, рассматриваемого в выпускной квалификационной работе, была выбрана среда СУБД MS SQL Server.

Среди преимуществ использования SQL Server стоит отметить [10]:

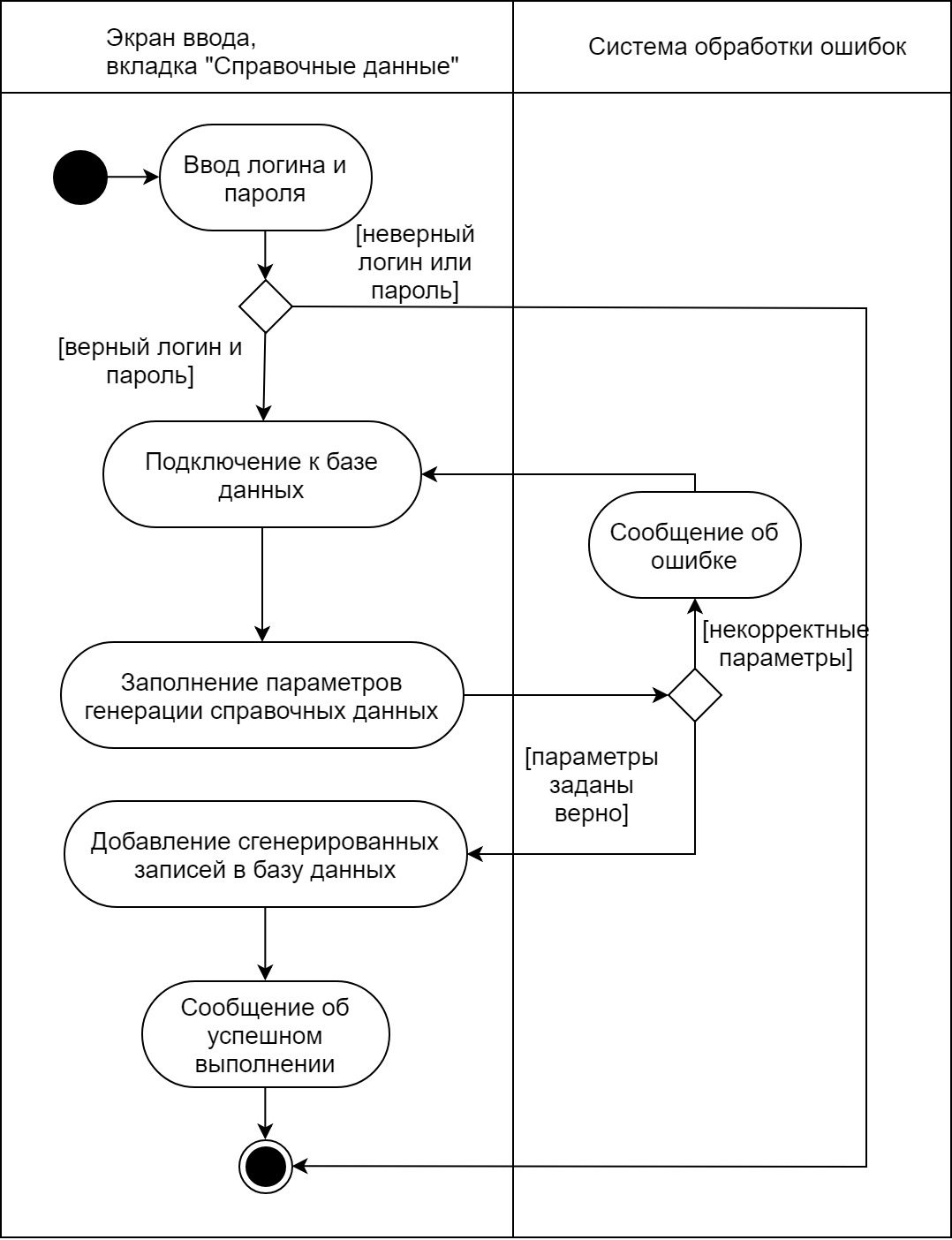


Рисунок 9 – Диаграмма деятельности

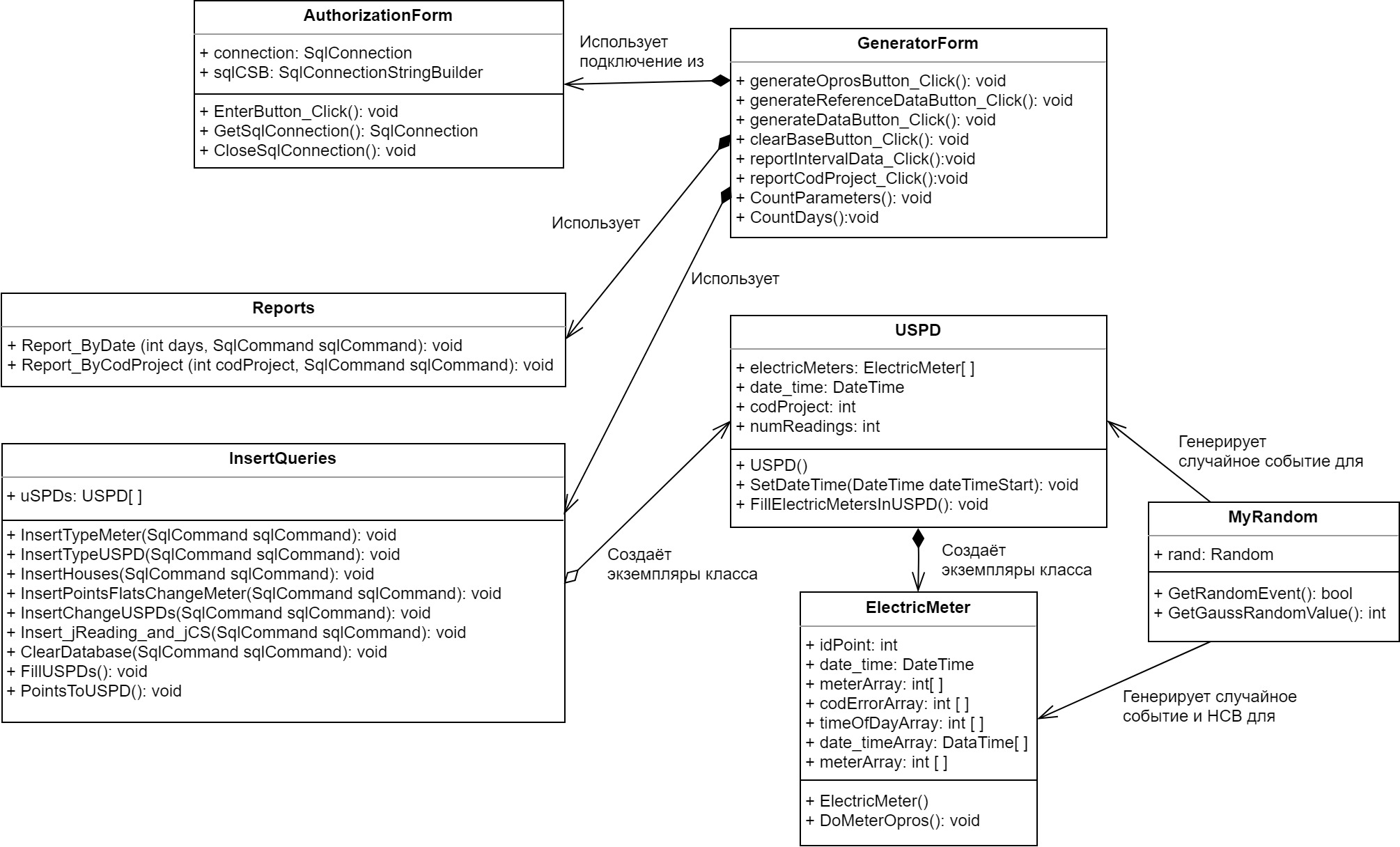


Рисунок 10 – Диаграмма классов

− эффективный инструмент для аналитики больших объемов информации;

− улучшенные функциональные возможности и тесная интеграция с разнообразными средствами разработки (Microsoft Office System, Visual Studio и другими);

− отличные показатели безопасности, масштабируемости и надежности;

− тесная интеграция с облачными системами;

− стабильность работы системы;

− высокая производительность.

Сервер баз данных Microsoft SQL Server позволяет надежно и безопасно хранить информацию практически неограниченного объема, поддерживает вертикальное масштабирование до 256 процессоров. Поддерживаются хранимые процедуры, триггеры и функции, технологиями доступа к данным ADO.NET и LINQ.

Для создания, доступа и управления базами данных и всеми элементами SQL Server используется специализированная среда Microsoft SQL Server Management. Редактор кода позволяет разрабатывать скриптовые сценарии на Transact-SQL, программировать многомерные запросы доступа к данным и проводить их анализ с поддержкой сохранения результатов в XML. Создание запросов и сценариев возможно без подключения к сети или серверу, с последующим выполнением и синхронизацией. Имеется широкий выбор предустановленных шаблонов и система управления версиями [11].

2.3.2 Выбор средства разработки

В качестве языка программирования был выбран C# – один из самых мощных, быстро развивающихся и востребованных языков в ИТ-отрасли. C# является объектно-ориентированным, поддерживает полиморфизм, наследование, перегрузку операторов, статическую типизацию. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений.

В качестве среды разработки была выбрана бесплатная и полнофункциональная среда разработки Visual Studio 2017. Данный продукт поставляется с конструкторами графических пользовательских интерфейсов, с возможностями работы с базами данных, с поддержкой фрагментов кода, с возможностями для просмотра всего проекта в целом, с просмотром свойств объектов, включает технологии WCF, WPF, ASP.NET, ADO.NET [12].

2.3.4 Выбор технологии доступа к базе данных

Для осуществления связи между базой данных и приложением на C# была выбрана технология ADO.NET.

ADO.NET предоставляет собой технологию работы с данными, которая основана на платформе .NET Framework. Эта технология представляет набор классов, реализующих программные интерфейсы для облегчения подключения к базам данных из приложения независимо от особенностей реализации конкретной системы управления базами данных и от структуры самой базы данных, а также независимо от места расположения этой самой базы – в частности, в распределенной среде (клиент-серверное приложение) на стороне сервера. То есть функционал ADO.NET построен таким образом, чтобы предоставить разработчикам унифицированный интерфейс для работы с самыми различными СУБД, включая MS SQL Server [13].

2.3.5 Разработка алгоритмов работы системы

Разработанная система выполняет следующие основные функции:

1. генерация справочных данных;
2. моделирование и генерация данных опроса счётчиков;
3. формирование отчётов по сгенерированных данным.

На рисунке 11 представлен общий алгоритм работы системы генерации.

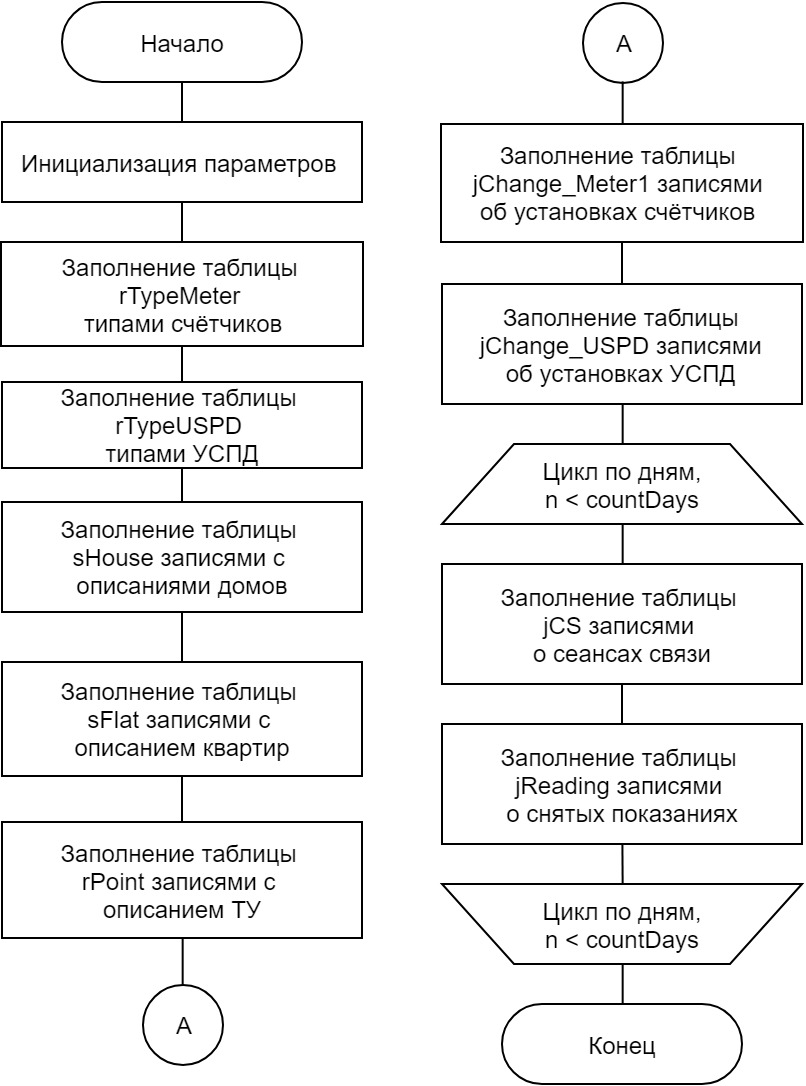


Рисунок 11 – Общий алгоритм генерации данных

3 Экспериментальная проверка эффективности программной реализации метода решения задачи

3.1. Разработка контрольного примера обработки данных

Целью испытаний является проверка работоспособности системы, то есть выполнение всех требований функционального назначения в соответствии с требованиями технического задания. Контрольный пример описывает авторизацию пользователя и формирование отчёта по сгенерированным данным.

Проверка возможности запуска системы.

Действия: пройти авторизацию в системе.

Результат: на экране отображается главная форма системы (рисунок 12).

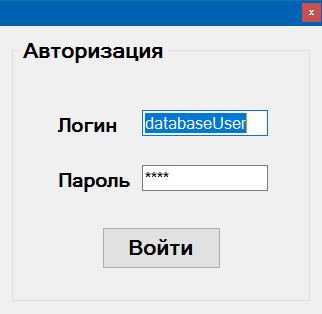


Рисунок 12 – Окно авторизации

Проверка возможности создания отчёта.

Действия:

1. на главном экране нажать вкладку «Отчёт»;
2. на экранной форме выбрать нужные параметры;
3. нажать клавишу «Создать отчёт».

Результат:

В папке с программой появится файл с отчётом (рисунок 13, 14).

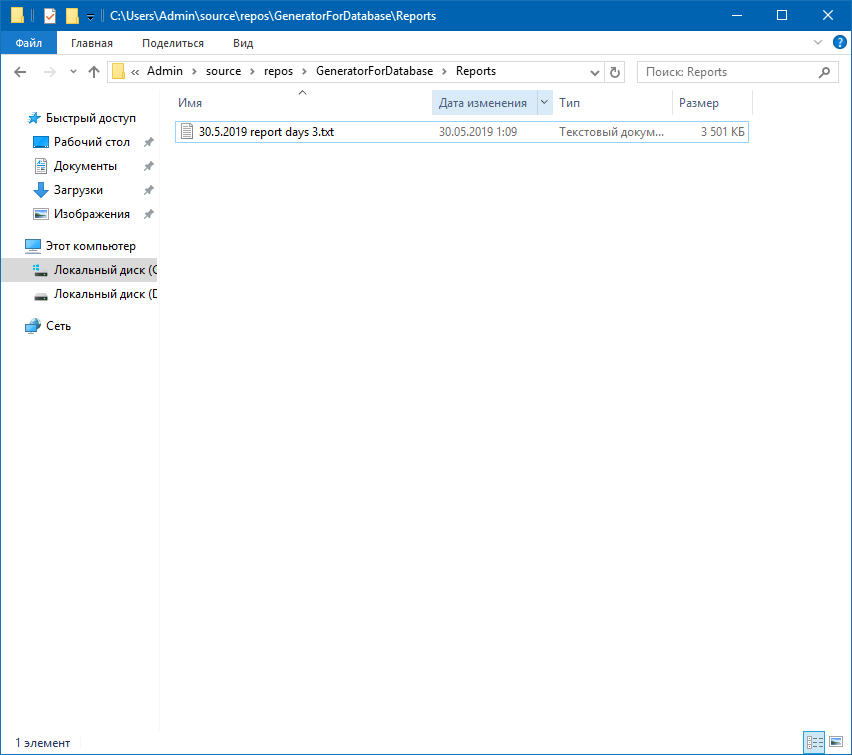


Рисунок 13 – Созданный файл с отчётом

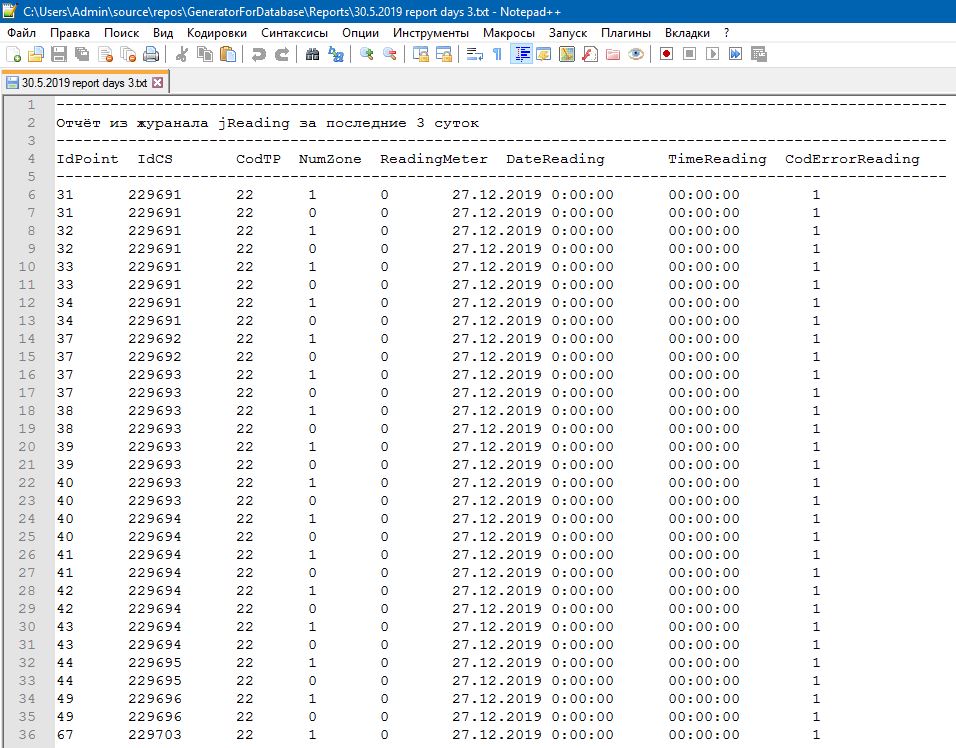


Рисунок 14 – Содержимое файла отчёта

3.2. Проверка эффективности разработанной программной системы для решения поставленной задачи

Эффективность работы модуля генерации отчётов будем поведена на нескольких индексах в базе данных.

СУБД MS SQL имеет несколько индексов для таблиц: кластеризованный, некластеризованный, фильтруемый, уникальный, колоночный и другие. Предлагается использовать колоночный индекс и некластеризованный по тем полям, которые будут участвовать при извлечении данных из БД для создания отчётов. Для сравнения возьмём также кластеризованный индекс, который работает по первичному ключу.

Исходные данные: количество жильцов 150 000 человек; ср. число жильцов в квартире 3,5; процент населения в N квартирных домах 4 – 1%, 8 – 1%, 16 – 2%, 50 – 20%, 100 – 40%, 200 – 36%; вероятность отказа УСПД – 10%; вероятность отказа счётчика – 10%; средний расход электроэнергии днём 8 кВт (отклонение 3), ночью 2 кВт (отклонение 1); интервал опроса 01:01:2019 – 31:12:2019.

Исходя из полученных данных, можно произвести оценку производительности системы. В сравнении с кластеризованным индексом, скорость работы запросов при использовании колоночного индекса или некластеризованного эффективнее в 70-100 раз при получении данных из БД по коду проекта внедрения АСКУЭ (номер договора на монтаж и пуско‑наладку АСКУЭ в МКЖД), а при получении данных за последние 3, 7 или 30 дней в среднем в 20-40 раз. Разница между использованием некластеризованного и колоночного кластеризованного оказалась несущественной, но предпочтение стоит отдать первому, т.к. в нём гораздо больший потенциал в детальной настройке, в отличие от второго, который просто автоматически работает на всей таблице.

Таблица 3.2 – Оценка производительности

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Запрос | Кластеризованный (созданный по умолчанию), мс | Некластеризованный, мс | Колоночный кластеризованный, мс | Затронуто строк, шт. |
| по коду проекта внедрения | 21405 | 207 | 299 | 21 584 |
| выписка за последние 3 дня | 46051 | 1204 | 1247 | 18 216 |
| выписка за последние 7 дней | 46924 | 1394 | 1335 | 37 084 |
| выписка за последние 30 дней | 46839 | 2584 | 2439 | 154 782 |
| Всего строк в таблице поиска: | | | | 37 720 498 |

ЗАКЛЮЧЕНИЕ

В соответствии с заданием была изучена предметная область и разработана информационная система «Генератор базы данных АСКУЭ»

Проанализированы имеющиеся системы аналоги и способы решения поставленной задачи.

Разработан информационно-логический проект системы в нотации UML с помощью инструментального средства StarUML.

Программное обеспечение было разработано в виде двух частей: клиентская часть, реализованная на языке С#, с использованием редактора Visual Studio 2017; серверная часть, реализована и спроектирована для сервера MS SQL Server.

Реализованная система предоставляет пользователям возможность модерирования базы данных АСКУЭ с последующей возможностью оценить производительность системы.

В ходе работы были сформулированы требования к программному обеспечению и обоснован выбор средств автоматизации. Проведён практический эксперимент по проверке корректности работы разработанного программного обеспечения. Также был выполнен анализ производительности системы на разных индексах в базе данных. Таким образом, цель работы выполнена, созданная система полностью соответствует заданию.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Красник, В.В. 102 способа хищения электроэнергии [Текст] /В.В. Красник – М.: ЭНАС, 2010. – 160 с.
2. АИИС КУЭ (АСКУЭ) [Электронный ресурс]. – URL: https://http://matritca.ru/produktsiya/aiis-kue-askue (дата обращения: 01.06.2019)
3. Программный комплекс ТЕЛЕСКОП+ ЗАО «НПФ Прорыв» [Электронный ресурс]. – URL: http://www.proryv.com/soft/telescope\_plus\_four/ (дата обращения: 01.06.2019).
4. Назначение АСКУЭ - АО «Энергомера» [Электронный ресурс]. – URL: http://www.energomera.ru/ru/products/askue/appointment (дата обращения: 01.06.2019).
5. Описание Меркурий-Энергоучет [Электронный ресурс]. – URL: https://www.incotexcom.ru/files/em/soft/aiis-opisanie.pdf (дата обращения: 01.06.2019).
6. Буч, Г. Язык UML. Руководство пользователя [Текст] / Г.Буч, Д. Рамбо, А. Джекобсон. – М.: [ДМК Пресс](https://ru.wikipedia.org/w/index.php?title=%D0%94%D0%9C%D0%9A_%D0%9F%D1%80%D0%B5%D1%81%D1%81_(%D0%B8%D0%B7%D0%B4%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%BE)&action=edit&redlink=1), 2004. – 432 с.
7. Объектная модель. Объектно-ориентированное проектирование. [Электронный ресурс]. – URL: [http://www.helloworld.ru/texts/comp/other/ oop/ch02.htm](http://www.helloworld.ru/texts/comp/other/%20oop/ch02.htm) (дата обращения: 01.06.2019).
8. Модели баз данных [Электронный ресурс]. – URL: <https://www.internet-technologies.ru/articles/modeli-baz-dannyh-sistemy-upravleniya-bazami-dannyh.html> (дата обращения: 01.06.2019).
9. Этапы разработки баз данных [Электронный ресурс]. – URL: http://inftis.narod.ru/db/db-10.htm (дата обращения: 01.06.2019).
10. Microsoft SQL Server 2012 [Электронный ресурс]. – URL: http://windowsfan.ru/microsoft-sql-server-2012-fes (дата обращения: 01.06.2019)
11. Система управления базами данных Microsoft Servers SQL [Электронный ресурс]. – URL: http://fb.ru/article/145800/sistema-upravleniya-bazami-dannyih-microsoft-servers-sql (дата обращения: 01.06.2019).
12. Visual Studio описание [Электронный ресурс]. – URL: https://alekseygulynin.ru/visual-studio (дата обращения: 01.06.2019).
13. Основы ADO .NET [Электронный ресурс]. – URL: http://citforum.ru/programming/csharp/ado\_dot\_net (дата обращения: 01.06.2019).

ПРИЛОЖЕНИЕ А

Руководство пользователя

При запуске программы открывается окно авторизации (Рисунок А.1). Пользователю необходимо заполнить поля логина и пароля, чтобы успешно авторизоваться на сервере. Далее откроется главное меню программы на вкладке «Справочные данные» (рисунок А.2).

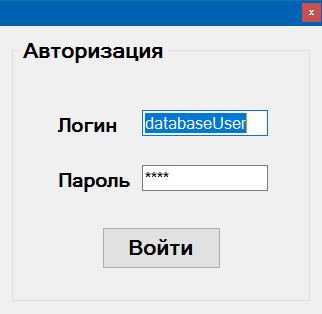


Рисунок А.1 – Окно авторизации

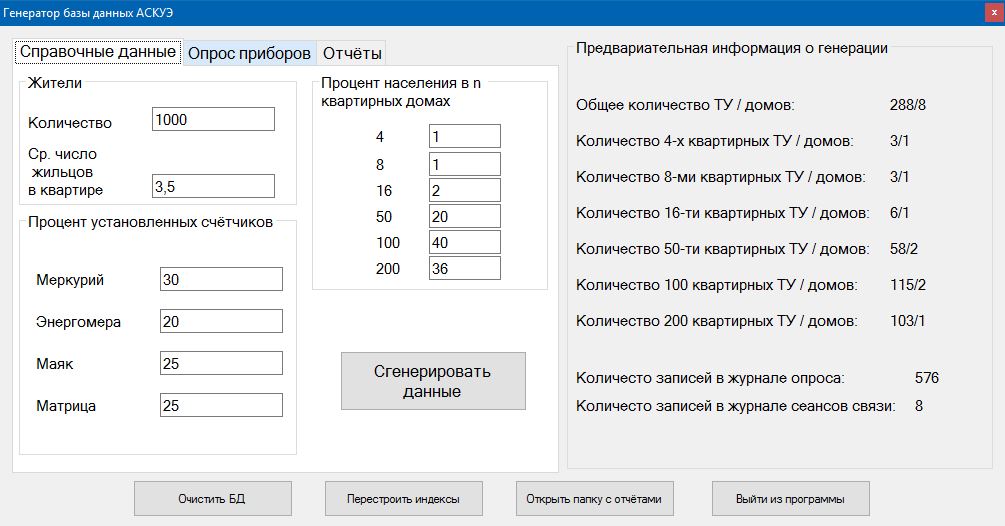


Рисунок А.2 – Вкладка «Справочные данные»

На ней пользователь может указать желаемые настройки для генерации: общее число жителей, среднее число жильцов в квартирах, проценты по установленным счётчикам и заселённость жителей по разным типам домов. В правой части окна доступна предварительная расчётная информация. Она обновляется динамически, по изменению в полях ввода. Для того чтобы заполнить базу данных справочными записями, следует нажать кнопку «Сгенерировать данные».

Следующая вкладка «Опрос приборов» аналогична по своим функциям предыдущей (Рисунок А.3). На ней пользователь может задать: вероятности отказа приборов, средний расход электроэнергии днём и ночью, а также указать временной интервал работы. При изменении параметров, справа динамически будет обновлено предварительное количество записей в журналах опроса и сеансов связи. Рядом с полями ввода интервала даты, динамически подсчитывается число дней опроса.

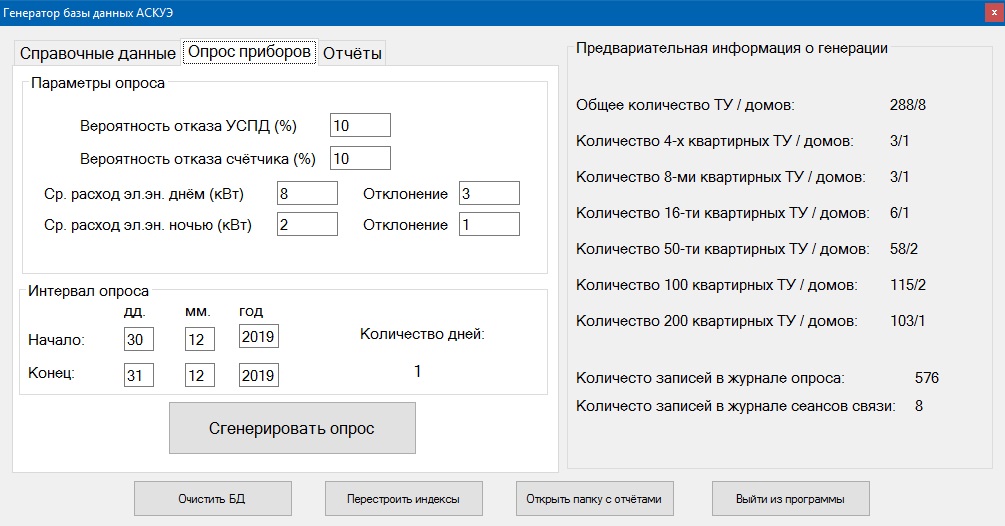


Рисунок А.3 – Вкладка «Опрос приборов»

На вкладке «Отчёты» пользователь имеет возможность получить выписку из базы данных по сгенерированным записям (Рисунок А.4).

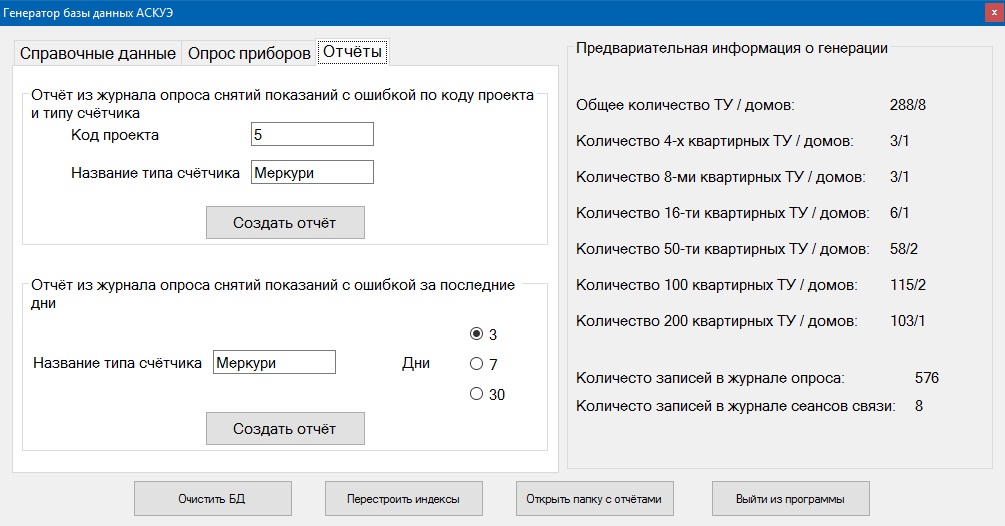


Рисунок А.4 – Вкладка «Отчёты»

Пользователю доступны два вида отчёта по неудачно опрошенным точкам учёта: в первом случае доступна выборка по названию типа счётчика и коду проекта, по которому установлен прибор, второй отчёт предоставляет данные по названию типа счётчика за указанный период времени. Для того чтобы сгенерировать отчёт следует нажать соответствующую кнопку на данной вкладке. Отчёт сохраняться в формате txt в папке с проектом.

В любой момент пользователю доступны клавиши с возможностью полной очистки базы данных, перестроения индекса в таблицах, просмотра папки с отчётами или выхода из программы, они расположены в нижней части окна программы.

ПРИЛОЖЕНИЕ Б

Листинг программы

Реализация компонентов базы данных представлена SQL-кодом:

Листинг 1. Создание базы данных

create database ModellerBase

go

use ModellerBase

CREATE TABLE sHouse

(

IdHouse bigint not null identity(1,1) primary key,

NumFlat int,

IdStreet int,

IdTown int,

NumUSPD int

)

CREATE TABLE rPoint

(

IdPoint bigint not null identity(1,1) primary key,

DataStart date,

DataFinish date,

TypePoint int,

IdLastChangeMeter1 int,

IdLastChangeMeter3 int,

IdLastChangeUSPD int,

IdLastChangeUSPD1 int,

DataLastReading date,

TimeLastReading time(0),

CodOtdel int,

CodProject int

)

CREATE TABLE sFlat

(

IdFlat bigint not null identity(1,1) primary key,

IdPoint bigint references rPoint(IdPoint),

IdHouse bigint references sHouse(IdHouse),

Numer int,

Comment varchar(20)

)

CREATE TABLE rTypeMeter

(

IdTypeMeter int not null identity(1,1) primary key,

ManufacturerMeter varchar(20),

NameTypeMeter varchar (20)

)

CREATE TABLE rTypeUSPD

(

IdTypeUSPD int not null identity(1,1) primary key,

ManufacturerUSPD varchar(20),

NameTypeUSPD varchar (20)

)