МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____ Д.В. Топольский
«___»_____ 2024 г.

Методические указания к выполнению выпускных квалификационных работ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ ЮУРГУ-090301.2024.405 ПЗ ВКР

Руководитель работы,
к.т.н., доцент каф. ЭВМ
_____Д.В. Топольский
«___»_____2024 г.

Автор работы,
студент группы КЭ-405
_____А.У.Узакбеков
«___»_____2024 г.

Нормоконтролёр,
ст. преподаватель каф. ЭВМ
_____ С.В. Сяськов
«___»_____2024 г.

Челябинск-2024

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____ Д.В. Топольский
«___»_____2024 г.

ЗАДАНИЕ

на выпускную квалификационную работу бакалавра

студенту группы КЭ-405
Узакбекову Арсену Узакбековичу
обучающемуся по направлению
09.03.01 «Информатика и вычислительная техника»

1. **Тема работы:** «Разработка информационной модели энергообъекта малой генерации» утверждена приказом по университету от «__» декабря 2023 г. №____

2. **Срок сдачи студентом законченной работы:** «__»_____2024 г.

3. **Исходные данные к работе:**

- малая гидроэлектростанция мощностью от 1 до 10 МВт
- интегрированная среда разработки PyCharm
- язык программирования Python
- библиотека для работы с базой данных «SQlite»
- библиотека для работы с графическим интерфейсом «Tkinter»
- документация к малым гидроэлектростанциям

4. Перечень вопросов, подлежащих разработке:

1. Написание введения и обзора литературы, анализ предметной области и существующих информационных моделей энергообъектов.
2. Разработка структуры информационной модели
3. Программная реализация информационной модели
4. Тестирование информационной модели

5. Дата выдачи задания: «__» декабря 2023 г.

Руководитель работы _____/Д.В. Топольский/

Студент _____/А.У. Узакбеков

КАЛЕНДАРНЫЙ ПЛАН

Наименование этапов выпускной квалификационной работы	Срок выполнения этапов работы	Подпись руководителя
Написание введения и обзора литературы, анализ предметной области и существующих информационных моделей энергообъектов	04.04.2024	
Разработка структуры информационной модели	15.05.2024	
Программная реализация информационной модели	17.05.2024	
Тестирование информационной модели	19.05.2024	
Компоновка текста работы и сдача на нормоконтроль	20.05.2024	
Подготовка презентации и доклада	25.05.2024	

Заведующий кафедрой _____/Д.В. Топольский/

Студент _____/А.У. Узакбеков

Аннотация

А.У. Узакбеков. Разработка информационной модели энергообъекта малой генерации. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШ ЭКН; 2024, 58 с., 14 ил., библиогр. список – 15 наим.

В выпускной квалификационной работе рассматривается разработка информационной модели энергообъекта малой генерации на примере гидроэлектростанции. Цель работы заключается в разработке эффективной и надежной информационной системы, обеспечивающей управление, мониторинг и оптимизацию работы гидроэлектростанции.

В рамках проекта проведен всесторонний анализ требований к системе, включающий функциональные и нефункциональные аспекты. Функциональные требования охватывают мониторинг и контроль ключевых параметров, управление операциями, создание отчетов и аналитики, а также механизмы уведомлений и оповещений. Нефункциональные требования включают надежность, доступность, масштабируемость, безопасность, производительность и удобство использования.

Особое внимание уделено аспектам надежности и безопасности системы, что особенно важно для критически важных инфраструктурных объектов, таких как гидроэлектростанции. Рассматриваются методы обеспечения отказоустойчивости, защиты данных и противодействия кибератакам.

Практическая часть работы включает разработку прототипа информационной модели, а также анализ результатов.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Информационная модель представляет собой объект в виде данных, описывающих его ключевые характеристики, связи, а также входы и выходы. Она позволяет моделировать состояния объекта на основе изменений входных параметрах.

В более общем смысле информационная модель – это набор информации, описывающей основные свойства и состояния объекта, процесса или явления, а также их взаимодействие с окружающим миром [1].

Использование информационной модели для энергообъектов малой генерации становится необходимостью в условиях растущего спроса на устойчивые и экологически чистые источники энергии. Информационная модель позволяет эффективно описывать основные характеристики энергообъектов, включая параметры, взаимосвязи между компонентами, а также входные и выходные данные.

В России, согласно нормативным и законодательным актам, объекты малой энергетики включают в себя следующие типы электростанций:

1. Микроэлектростанции (мощность до 100 кВт).
2. Миниэлектростанции (мощность от 100 кВт до 1 МВт).
3. Малые электростанции (мощность свыше 1 МВт и до 30 МВт или до 25 МВт в некоторых источниках), оснащенные агрегатами мощностью до 10 МВт.
4. Теплогенерирующие установки, включая котлы и котельные, производительностью до 20 Гкал на одного человека.
5. Гидроэлектростанции и микро-ГЭС с агрегатами мощностью не более 100 кВт установленной мощности.
6. Атомные электростанции с мощностью энергоблоков: электрической - не более 150 МВт и тепловой - не более 500 МВт.

7. В перечень также включаются установки по производству энергии на основе потребления нетрадиционных видов топлива [2].

Таким образом, в дипломном проекте основной целью является создание информационной модели энергообъекта малой генерации.

Для достижения поставленной цели решались следующие задачи:

1)проведение обзора научно-технической литературы по созданию информационных моделей;

2) анализ существующих информационных моделей разных энергообъектов;

3) описание этапов разработки информационной модели;

4) анализ выбранного энергообъекта малой генерации;

5) создание информационной модели для выбранного энергообъекта;

6) проверка работоспособности модели.

Создание информационной модели для энергообъекта малой генерации предоставляет возможность анализировать и прогнозировать его работу, оптимизировать энергетические процессы, а также планировать эффективное использование ресурсов. Такой подход способствует повышению энергоэффективности, снижению затрат и обеспечению устойчивости работы энергообъекта малой генерации в различных условиях.

1. ОБЗОР НАУЧНО-ТЕХНИЧЕСКОЙ ЛИТЕРАТУРЫ

При создании баз данных учитываются две основные информационные модели: модель предприятия и модель данных. Модель предприятия определяет структурные подразделения организации, использующие информацию из базы данных, и направление потоков информации между ними. Модель данных более подробно описывает источники информации, структурные подразделения фирмы, переходы между различными типами моделей и подразделениями, которые потребляют информацию.

На рисунке 1 отображены структурные подразделения фирмы – подразделение 1, … подразделение N. В левой части рисунка эти подразделения выступают как источники концептуальных требований, то есть предоставляют сведения для создания базы данных. В верхней части рисунка эти же подразделения выступают как потребители информации и источники данных для базы данных.

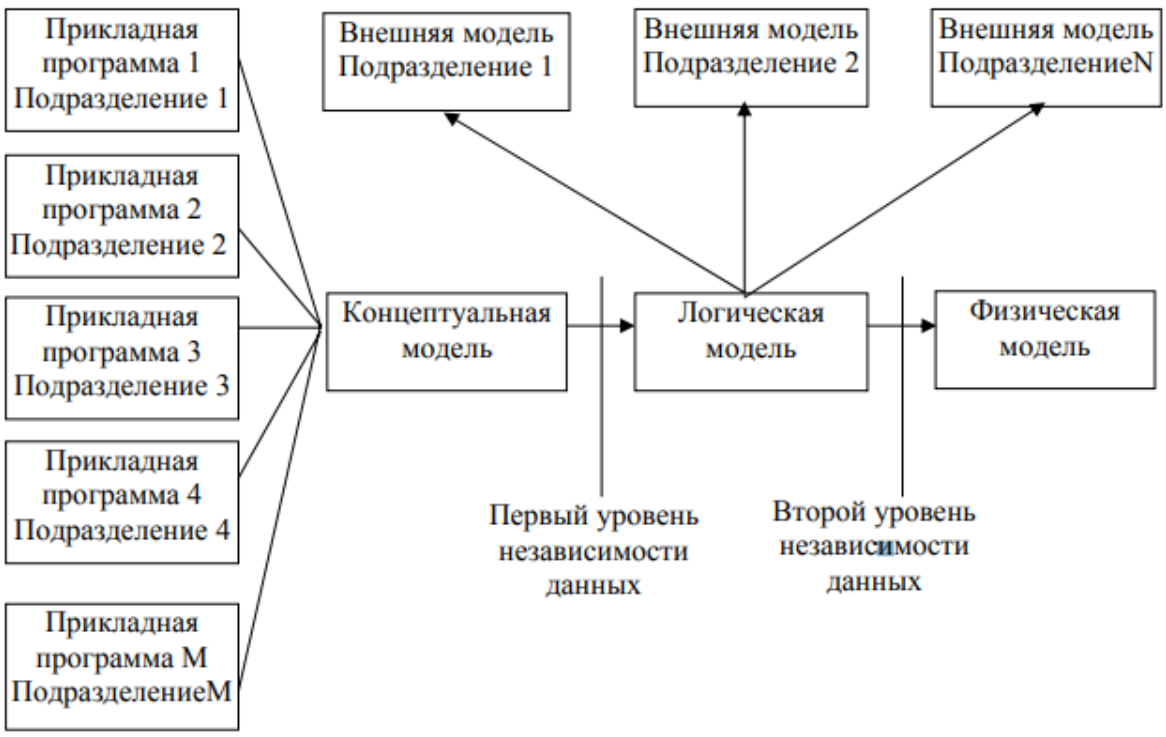

Рисунок 1 – Информационная модель данных

Концептуальная модель представляет собой совокупность требований, выдвигаемых сотрудниками различных подразделений компании.

Перевод концептуальной модели в логическую модель данных происходит при использовании системы управления базами данных (СУБД). При этом выбирается наиболее подходящая СУБД, и в случае невозможности удовлетворения всех требований проводится аргументация с последующим уточнением концептуальных требований.

Далее, после построения логической модели, создается письменный протокол, в котором перечисляются все концептуальные требования и операции по обработке информации в базе данных.

При переходе от концептуальной к логической модели обеспечивается первый уровень независимости данных, при котором внешние модели не привязаны к типу физической памяти или методам обработки данных в базе данных. Логическая модель данных разбивается на внешние модели, которые представляют собой составные части общей модели. Границы между внешними моделями нечеткие, и различные структурные подразделения фирмы могут иметь доступ к одним и тем же данным, но только одно подразделение может вносить изменения в данные. Форма отображения одних и тех же данных может различаться. В зависимости от поставленных целей логическая модель данных может быть либо иерархической, либо сетевой, либо реляционной.

Далее осуществляется переход от логической модели к физической. При построении физической модели определяются технические характеристики персонального компьютера: объем оперативной памяти, объем памяти жесткого диска и т. д. При переходе от логической к физической модели определяются

количество и виды индексов, а также способы доступа к данным. Этот процесс требует соблюдения второго уровня независимости данных.

Рассмотрим три типа логических моделей.

1) Иерархическая модель данных

Из названия следует, что эта модель имеет иерархическую структуру, где каждый элемент связан только с одним элементом выше в иерархии, но при этом может быть ссылкой для одного или нескольких элементов ниже в иерархии, что изображено на рисунке 2.

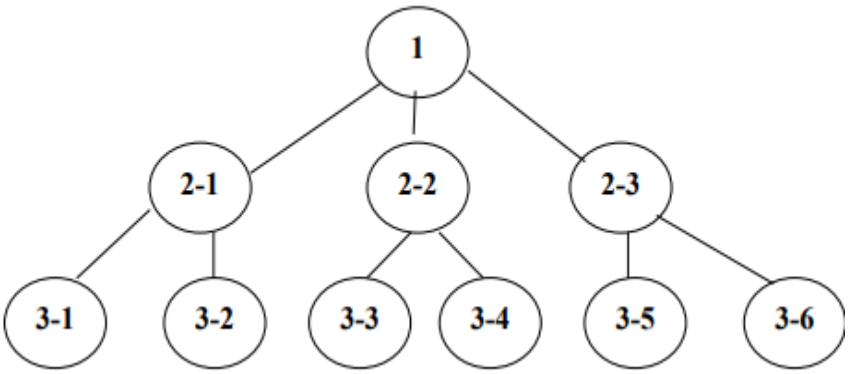

Рисунок 2 – Логическая иерархическая модель

Данная модель представляется в виде графа, представленная на рисунке 3, где элементы расположены в иерархическом порядке, от общего к частному, формируя древовидную структуру. Число вершин на первом уровне определяет количество деревьев в базе данных. На одном уровне не допускаются взаимосвязи между элементами.

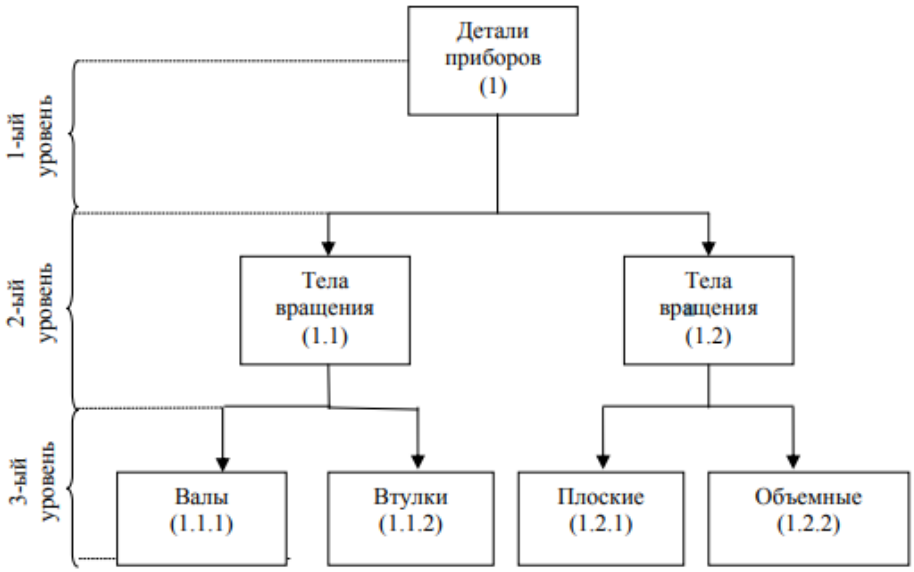

Рисунок 3 – Иерархическая модель в виде графа

2) Сетевая модель данных

В сетевой модели отсевает понятие главного и подчиненного объекта. Один объект может быть как главным, так и подчиненным, имея любое количество взаимосвязей, как показано на рисунок 4.

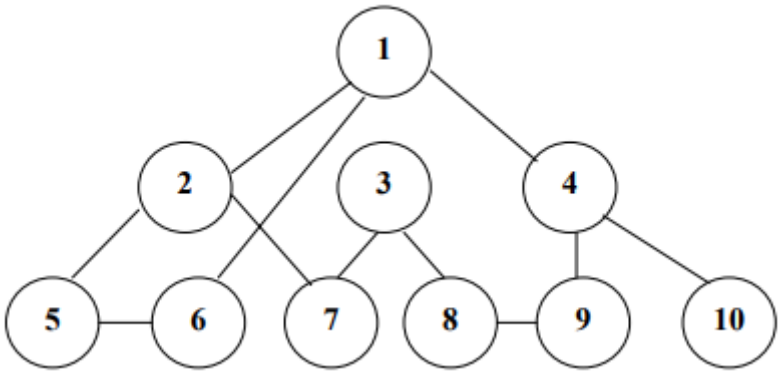

Рисунок 4 – Логическая сетевая модель

Сетевая модель может быть преобразована в граф-дерево как на рисунке 5.

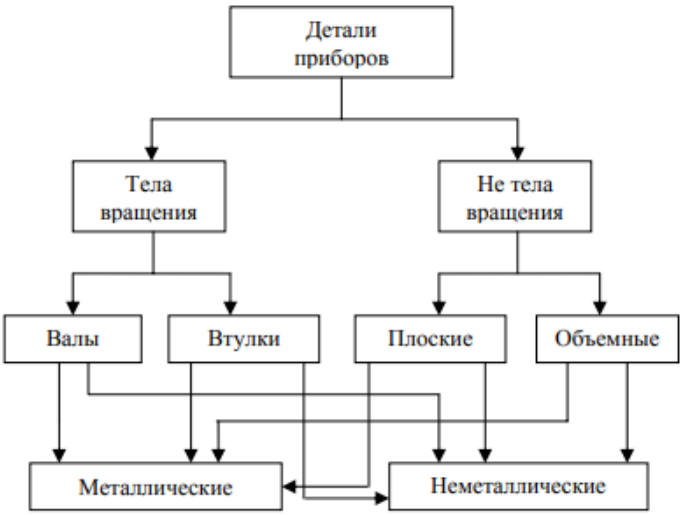

Рисунок 5 – Сетевая модель в виде графа

Основной идеей реляционной модели данных является представление набора данных в виде двумерного массива – таблицы как на рисунок 6.

Имя файла						
Поле		Признак ключа	Формат поля			
Имя (обозначение)	Полное наименование		Тип	Длина	Точность (для чисел)	N/NN
имя1						
…						
имя n						

Рисунок 6 – Структура реляционной таблицы

Для построения реляционной модели базы данных необходимо, чтобы каждая таблица содержала первичный ключ. Этот тип модели наиболее часто используется при разработке баз данных. [3].

Поскольку реляционные модели данных, или реляционные базы данных, являются в настоящее время основным способом в проектировании и

организации информационных систем, при разработке информационной модели энергообъекта малой генерации воспользуемся этой же моделью.

1.1. АНАЛИЗ СУЩЕСТВУЮЩИХ ИНФОРМАЦИОННЫХ МОДЕЛЕЙ ПРЕДМЕТНОЙ ОБЛАСТИ

Для повышения качества данных, снижения их разнородности и разновременности обновления, предложена единая информационная модель электроэнергетики на рисунке 7.

Единая цифровая модель включает совокупность существующих и планируемых к созданию объектов электроэнергетики, оборудования, устройств, их связей, свойств, а также иных сущностей в соответствии с серией ГОСТ Р 58651 «Единая энергетическая система и изолировано работающие энергосистемы».

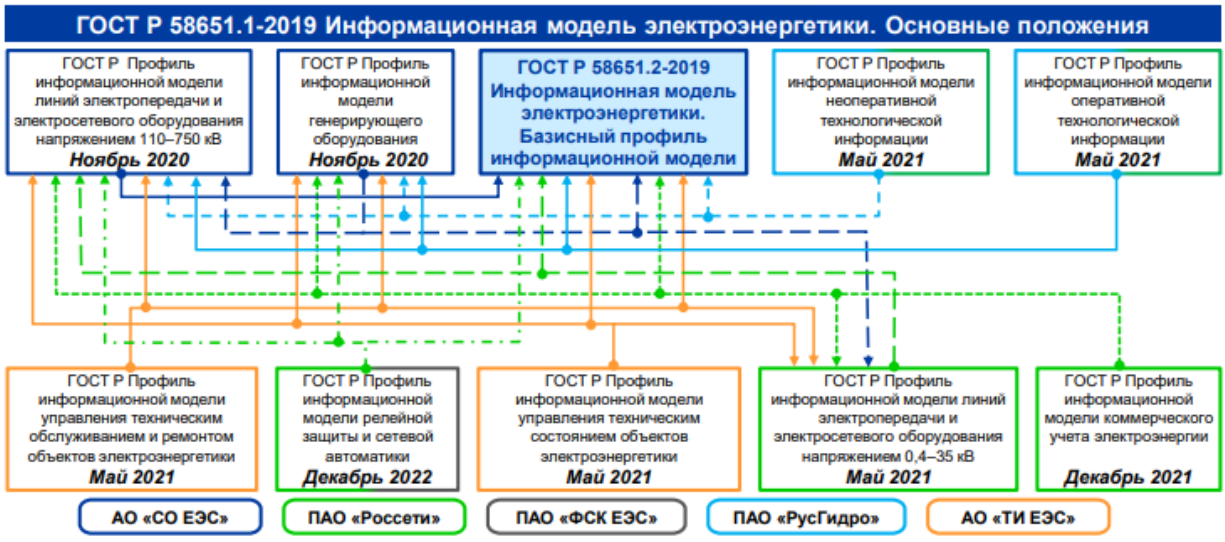

Рисунок 7 – Информационная модель электроэнергетики

Информационная модель электроэнергетики представляет собой иерархическое взаимосвязанное множество цифровых моделей, создаваемых и поддерживаемых в актуальном состоянии на основе принципа единства

идентификации объектов модели и с учетом жизненного цикла энергообъектов [4].

С целью оптимизации режимов расхода топлива ТЭС и снижения выбросов парниковых газов, предложена иерархическая модель ТЭС на рисунке 8.

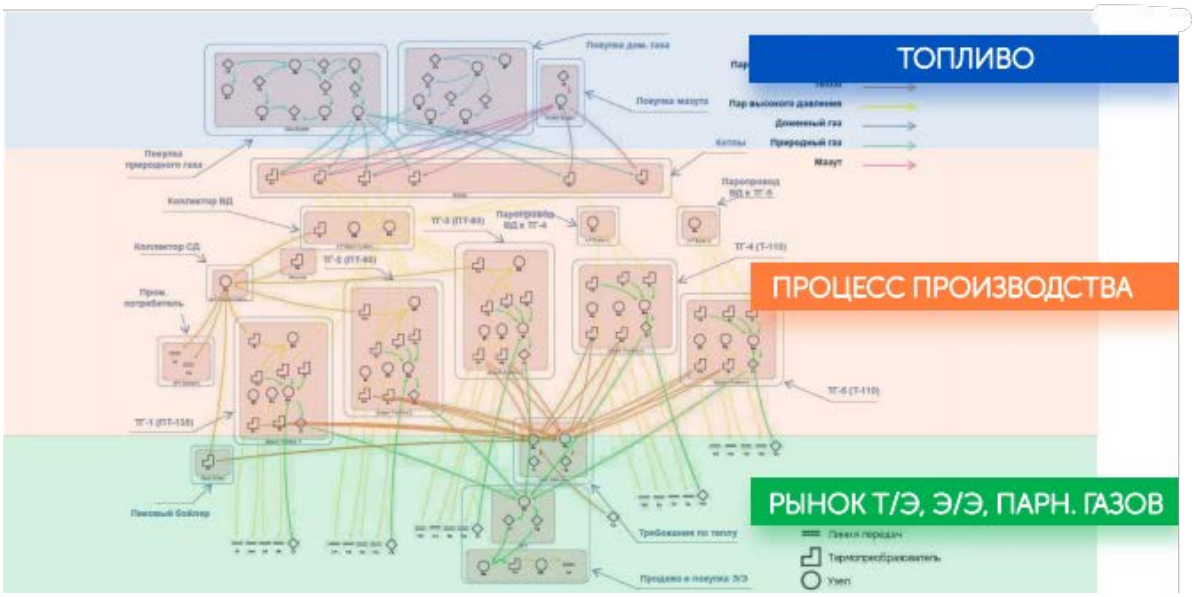

Рисунок 8 – Информационная модель ТЭС

Модель энергообъекта представляет из себя набор разнообразных компонентов, с помощью которых формируется единое решение по оптимизации работы станции [5].

Еще одним примером информационной модели является модель ЛЭП на рисунке 9, представленная двумя разделами. Первый содержит сведения, необходимые для эксплуатации, вторая – для моделирования режимов электрических сетей .

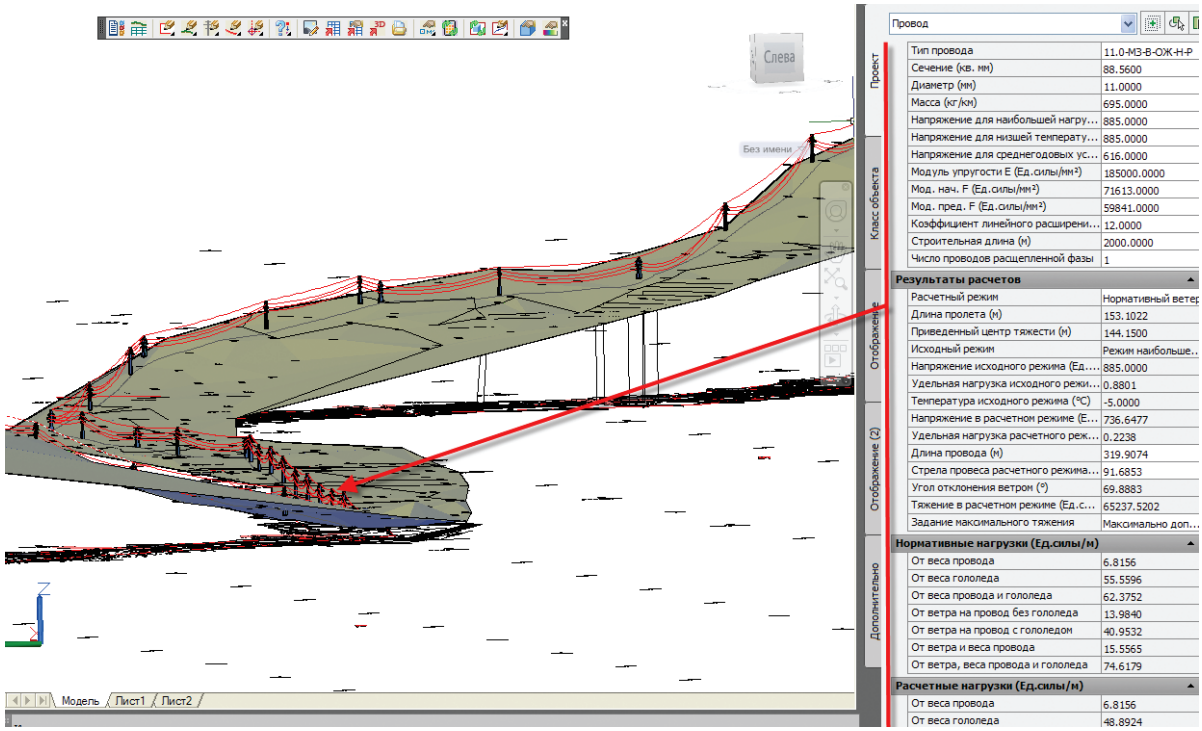

Рисунок 9 – Информационная модель ЛЭП

Эксплуатационная часть представляется графическими изображениями трасс ЛЭП на цифровой модели местности и атрибутивной, отражающей технические характеристики и все сведения о событиях в «жизни» ЛЭП. Структура модели ЛЭП является реляционной.

На рисунке 10 представлена информационная модель концепции единого информационного пространства АЭС, позволяющая аккумулировать информацию по всем процессам на станции

Рисунок 10 – Информационная модель АЭС

Основу информационной модели составляет WEB-портал, куда поступают данные о внешних системах, 3D проектировании, состоянии оборудования и даже управление закупками [6].

Этапы разработки информационной модели:

Выделяют пять основных этапов информационного моделирования:

- Этап первый. Постановка задачи.

В процессе постановки задачи, задача определяется как конкретная проблема, требующая решения. Этот этап включает в себя описание проблемы, определение целей моделирования и анализ объекта или процесса.

В процессе постановки задачи важно сформулировать ее на понятном языке. Основное внимание уделяется определению объекта моделирования и ожидаемого результата. Во время анализа объекта или процесса проводится

четкое выделение моделируемого объекта, его основных свойств, элементов и связей между ними.

- Этап второй. Разработка модели.

На данном этапе происходит выявление свойств, состояний, действий и других характеристик элементарных объектов в различных формах: в виде схем, таблиц и т.д. Создается представление об элементарных объектах, составляющих исходный объект, то есть информационную модель. Модели должны отражать основные признаки, свойства, состояния и взаимосвязи объектов реального мира, поскольку именно они обеспечивают полную информацию об объекте.

Информационная модель никогда не охватывает объект полностью. Для одного объекта можно создать различные информационные модели. Выбор наиболее важной информации при построении информационной модели и сложность самой модели зависят от цели моделирования. Создание информационной модели является отправной точкой на этапе разработки модели. При анализе выделенные входные параметры объектов упорядочиваются по убыванию значимости, а затем модель упрощается с учетом цели моделирования.

Перед тем как начать моделирование, часто создают предварительные черновики чертежей или схем на бумаге и разрабатывают расчетные формулы. Таким образом, формируется информационная модель в различных символических формах, которая может быть как компьютерной, так и не компьютерной.

- Этап третий. Построение компьютерной модели.

Компьютерная модель — это модель, которая создается и реализуется с помощью программного обеспечения. Существует множество программных

комплексов, которые позволяют проводить исследования и моделирование информационных моделей. Каждая программная среда обладает своими инструментами и предназначена для работы с определенными типами информационных объектов. Пользователь уже знает, как должна выглядеть модель, и использует компьютер для ее воплощения в знаковой форме. Например, для создания геометрических моделей или схем используются графические среды, а для создания текстовых или табличных описаний - текстовые редакторы.

Основные функции компьютера в процессе моделирования систем:

Действует в качестве вспомогательного инструмента для решения задач, которые можно решить с помощью обычных вычислительных методов, алгоритмов и технологий.

Используется для постановки и решения новых задач, которые нельзя решить традиционными средствами, алгоритмами и технологиями.

Применяется для создания компьютерных обучающих и моделирующих сред, включая различные типы взаимодействий между обучающими и компьютером.

Используется в качестве средства моделирования для получения новых знаний.

Применяется для "обучения" новых моделей, включая самообучение моделей.

- Этап четвертый. Компьютерный эксперимент.

Вычислительный эксперимент — это новый метод исследования, который проводится с использованием компьютера или компьютерной среды. Он становится важным инструментом научного познания из-за необходимости исследования сложных и нелинейных математических моделей систем. Этот метод позволяет обнаруживать новые закономерности, проверять гипотезы,

визуализировать ход событий и так далее. В прошлом эксперименты проводились либо в лабораторных условиях на специальных установках, либо на натуре, на реальных образцах. С развитием вычислительной техники появился новый метод - компьютерный эксперимент, который включает последовательность действий с компьютерной моделью.

Этап пятый. Анализ результатов моделирования.

Конечная цель моделирования заключается в принятии решения на основе всестороннего анализа полученных результатов. Этот этап является решающим: либо исследование продолжается, либо завершается. Принятие решения основано на результатах тестирования и экспериментов. Если результаты не соответствуют поставленным целям, это может указывать на ошибки на предыдущих этапах, такие как упрощенное построение информационной модели, неудачный выбор метода или среды моделирования, или нарушение технологических приемов при построении модели. Если такие ошибки выявлены, требуется корректировка модели, возврат к одному из предыдущих этапов. Процесс повторяется до тех пор, пока результаты эксперимента не будут соответствовать целям моделирования.

2. РАЗРАБОТКА СТРУКТУРЫ ИНФОРМАЦИОННОЙ МОДЕЛИ

В качестве энергообъекта малой генерации выбрана гидроэлектростанция (ГЭС).

В сравнении с традиционными источниками энергии, малые гидроэлектростанции обладают рядом особенностей:

- небольшая единичная мощность ГЭС (от 100 кВт до 20 МВт) и часто низкий коэффициент использования установленной мощности в течение суток не позволяют получать значительные доходы от продажи электроэнергии, что приводит к необходимости максимально сокращать эксплуатационные расходы;

- один субъект энергетического рынка может управлять 10 и более малыми ГЭС, расположенными в различных областях и регионах страны. Это значительно усложняет централизованное диспетчерское управление ими, особенно с учетом практического отсутствия промышленных каналов связи;

- повышение требований энергорынка к автоматизированным системам коммерческого учета электроэнергии (АСКУЭ) в отношении оперативности обмена информацией между операторами и потребителями требует усовершенствования информационных средств, установленных на малых ГЭС;

- производительность малых ГЭС в значительной степени зависит от непредсказуемого воздействия окружающей среды, что усложняет процесс планирования их режимов работы;

- несогласованность норм и правил эксплуатации водных ресурсов, наряду с человеческим фактором, приводит к искусственным и часто необоснованным ограничениям, которые препятствуют обеспечению эффективности работы гидростанций данного класса.

Исходя из описания проблемы, целью создания информационной модели ГЭС является обеспечение эффективности эксплуатации малых ГЭС. Разработка информационной модели станет основой для автоматизированных систем управления, обеспечивая эффективное взаимодействие всех элементов ГЭС и позволяя оптимизировать процессы для достижения максимальной производительности и надежности.

2.1. СБОР И АНАЛИЗ ТРЕБОВАНИЙ

Для исследования информационной модели следует понять, кто будет пользоваться данной системой и какие обязанности будут у каждой группы пользователей. Соответственно, требуется идентифицировать заинтересованные стороны [7].

Идентификация заинтересованных сторон — это критический этап в процессе проектирования информационной модели гидроэлектростанции малой генерации. Заинтересованные стороны — это все лица и организации, которые непосредственно или косвенно влияют на работу гидроэлектростанции или зависят от её функционирования. Правильная идентификация и понимание их потребностей позволяют создать эффективную и функциональную систему.

Шаги идентификации заинтересованных сторон:

- Определение категорий заинтересованных сторон

Внутренние заинтересованные стороны:

1. Операторы гидроэлектростанции.
2. Технический персонал (инженеры, механики).
3. Управляющий персонал (менеджеры, директора).

Внешние заинтересованные стороны:

1. Регуляторы и контролирующие органы.
2. Поставщики оборудования и технологий.
3. Потребители электроэнергии.
4. Финансовые и инвестиционные учреждения.

- Анализ ролей и обязанностей каждой группы

Операторы гидроэлектростанции: ответственные за повседневное управление и контроль работы станции и выполняющие мониторинг параметров, управление генераторами и водохранилищем.

Технический персонал: инженеры и механики, обеспечивающие техническое обслуживание и ремонт оборудования и реагирующие на аварийные ситуации и проводящие профилактическое обслуживание.

Управляющий персонал: менеджеры и директора, принимающие стратегические и оперативные решения и отвечающие за финансовое управление, планирование и оптимизацию процессов.

Регуляторы и контролирующие органы: организации, устанавливающие нормы и стандарты для работы гидроэлектростанций, проводящие инспекции и аудит для обеспечения соблюдения нормативных требований.

Поставщики оборудования и технологий: компании, предоставляющие оборудование, программное обеспечение и услуги для гидроэлектростанции и обеспечивающие техническую поддержку и консультации по использованию продуктов.

Потребители электроэнергии: клиенты, получающие электроэнергию от гидроэлектростанции, которые заинтересованы в стабильном и надёжном снабжении электроэнергией.

Местные сообщества и экологи: местные жители, проживающие вблизи гидроэлектростанции и экологические организации, следящие за воздействием станции на окружающую среду.

Финансовые и инвестиционные учреждения: банки и инвесторы, финансирующие строительство и эксплуатацию гидроэлектростанции, которые заинтересованы в рентабельности и устойчивости проекта.

- Выявление потребностей и ожиданий каждой группы

Операторы гидроэлектростанции: потребность в удобных и интуитивно понятных интерфейсах для управления станцией, а также, ожидание точных и своевременных данных о состоянии оборудования и параметрах работы.

Технический персонал: потребность в детализированной информации о техническом состоянии и параметрах оборудования и ожидание наличия инструментов для диагностики и прогнозирования неисправностей.

Управляющий персонал: потребность в отчетах и аналитике для принятия обоснованных решений и ожидание инструментов для планирования и оптимизации производственных процессов.

Регуляторы и контролирующие органы: потребность в доступе к данным о соблюдении нормативных требований и стандартов и ожидание прозрачности и готовности к инспекциям и аудитам.

Поставщики оборудования и технологий: потребность в данных о производительности и использовании их продуктов и ожидание своевременной информации для предоставления технической поддержки.

Потребители электроэнергии: потребность в стабильном и надёжном электроснабжении и ожидание минимизации перебоев и быстрого реагирования на аварийные ситуации.

Местные сообщества и экологи: потребность в информации о воздействии гидроэлектростанции на окружающую среду и ожидание соблюдения экологических норм и минимизации негативного воздействия.

Финансовые и инвестиционные учреждения: потребность в отчетах о финансовых показателях и устойчивости проекта, а также, ожидание высокой рентабельности и минимизации рисков.

- Методы взаимодействия с заинтересованными сторонами

Интервью и встречи: проведение индивидуальных и групповых интервью с ключевыми представителями каждой группы для выявления их потребностей и ожиданий, а также, организация регулярных встреч и совещаний для обсуждения текущих вопросов и получения обратной связи.

Анкетирование и опросы: разработка и распространение анкет и опросов для сбора мнений и предложений от всех заинтересованных сторон и анализ полученных данных и учет предложений при проектировании системы.

Документирование требований: создание единого документа, описывающего требования и ожидания всех заинтересованных сторон и регулярное обновление и согласование документа с заинтересованными сторонами.

Обратная связь и корректировка: организация механизмов для регулярного сбора обратной связи от пользователей системы и внесение необходимых изменений и корректировок на основе полученных данных.

2.2. АНАЛИЗ ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ

Анализ функциональных требований — это важнейший этап проектирования информационной модели, который позволяет определить, какие конкретные функции и возможности должны быть реализованы в системе для удовлетворения потребностей всех заинтересованных сторон. Функциональные

требования описывают, что именно система должна делать для обеспечения эффективного управления и эксплуатации гидроэлектростанции малой генерации [8].

- Мониторинг и контроль параметров

Состояние оборудования: Мониторинг состояния ключевых компонентов гидроэлектростанции для обеспечения их исправности и своевременного обслуживания.

Частота обновления данных:

При изменении состава ГЭС: Некоторые параметры, такие как состояние оборудования и его тип, должны обновляться в зависимости от состава элементов, установленных в ГЭС для оперативного управления.

Визуализация данных:

Окно вывода информации об объекте: разработка удобных и информативных панелей, отображающих ключевые параметры и индикаторы состояния в реальном времени.

- Управление операциями

Оперативное управление оборудованием: При необходимости изменять типы определенных объектов, таких как турбины, станции, месторасположение обектов.

- Отчетность и аналитика

Создание отчетов:

Генерация отчетов с основными показателями состава гидроэлектростанции.

- Уведомления и оповещения

Информационные уведомления: Уведомления о выполнении определенных операций, которые требуют внимания.

Анализ функциональных требований позволил четко определить, какие конкретные функции и возможности должны быть исследованы и реализованы в информационной модели ГЭС малой генерации. Этот этап обеспечивает понимание того, что система должна делать для удовлетворения потребностей всех заинтересованных сторон и для эффективного управления гидроэлектростанцией.

2.3. АНАЛИЗ НЕФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ

Анализ нефункциональных требований не менее важен, чем анализ функциональных требований, так как он определяет качество системы и её соответствие ожиданиям пользователей. Нефункциональные требования описывают атрибуты системы, такие как производительность, надёжность, безопасность, масштабируемость и удобство использования. Правильный учет этих требований обеспечивает стабильную и эффективную работу информационной модели гидроэлектростанции малой генерации [9].

- Надежность и доступность

Требования к надежности системы:

1. Отказоустойчивость: система должна продолжать работу при возникновении сбоев и неисправностей.
2. Среднее время безотказной работы (MTBF): установление целевого значения MTBF, обеспечивающего высокую надежность системы.

3. Мониторинг состояния системы: постоянный мониторинг ключевых показателей надежности (например, частота сбоев, время отклика) и оперативное реагирование на их изменения.
4. Требования к доступности системы: установление целевого значения доступности системы (99.9% времени).
5. Плановое обслуживание: разработка графика планового обслуживания с минимальным воздействием на доступность системы.
6. Гарантии непрерывной работы: меры для обеспечения непрерывной работы в критически важные периоды (например, во время пиковых нагрузок).

- Масштабируемость

1. Горизонтальная и вертикальная масштабируемость: возможность добавления дополнительных элементов ГЭС для более полного описания информационной модели.
2. Возможность создавать дополнительные функциональные системы для более качественной обработки данных
3. Возможность увеличения производительности путем добавления ресурсов (процессоров, памяти) к существующим серверам.
4. Эффективные методы обработки и хранения больших объемов данных.
5. Оптимизация запросов к базе данных для повышения производительности.

- Производительность

1. Требования к скорости обработки данных: определение максимального допустимого времени отклика для различных операций.
2. Оптимизация запросов и структуры базы данных для повышения скорости обработки данных.
3. Использование индексов и кэширования для ускорения выполнения запросов.
4. Эффективное управление вычислительными и сетевыми ресурсами.

5. Постоянный мониторинг производительности системы и выявление узких мест.

- Поддержка высокой нагрузки

1. Планирование и подготовка к обработке пиковых нагрузок.
2. Возможность динамического масштабирования ресурсов в зависимости от текущей нагрузки.

- Удобство использования

1. Разработка интерфейса, который будет понятен и удобен для пользователей с разным уровнем технической подготовки.
2. Обеспечение доступности функций на всех популярных типах операционных систем.

- Сохранение информации об объектах и их элементах.

1. Возможность при выходе из программы автоматически сохранять добавленные данные

Анализ нефункциональных требований позволил определить ключевые атрибуты системы, которые обеспечивают её качество, надёжность и эффективность. Включение этих требований в проектирование информационной модели гидроэлектростанции малой генерации гарантирует, что система будет соответствовать ожиданиям пользователей, работать стабильно и эффективно, а также легко адаптироваться к изменяющимся условиям и требованиям. Рассмотренные аспекты надёжности и доступности, масштабируемости, безопасности, производительности и удобства использования создали основу для создания высококачественной информационной системы.

2.4. ОПРЕДЕЛЕНИЕ ОСНОВНЫХ КОМПОНЕНТОВ СИСТЕМЫ

Определение основных компонентов системы является ключевым этапом при проектировании информационной модели гидроэлектростанции малой генерации. На этом этапе определяются и описываются все основные элементы

системы, их функции и взаимосвязи. Это позволяет создать целостное и эффективное решение, обеспечивающее надежную и безопасную работу гидроэлектростанции [10].

Основные компоненты системы:

Центральный сервер (Core Server)

Функции:

1. Обработка и хранение данных.
2. Управление центральной логикой системы.
3. Координация взаимодействия всех компонентов системы.
4. Связи:
5. Связь с базой данных для хранения данных.
6. Взаимодействие с клиентскими приложениями для предоставления данных пользователям.
7. Обмен данными с внешними системами и модулями.

База данных (Database)

Функции:

1. Хранение всех данных, связанных с работой гидроэлектростанции
2. Обеспечение быстрого доступа к данным для анализа и отчетности.

Связи:

1. Связь с центральным сервером для записи и извлечения данных.
2. Взаимодействие с аналитическими инструментами для обработки данных.

Модуль мониторинга и управления (Monitoring and Control Module)

Функции:

1. Сбор данных с сенсоров и датчиков, установленных на гидроэлектростанции.
2. Мониторинг ключевых параметров работы.

Связи:

1. Связь с сенсорами и датчиками для сбора данных.
2. Взаимодействие с центральным сервером для передачи данных и получения команд.

Интерфейс пользователя (User Interface)

Функции:

1. Обеспечение доступа пользователей к системе для управления и мониторинга.
2. Визуализация данных в виде графиков, диаграмм и отчетов.
3. Предоставление инструментов для настройки и персонализации системы.

Связи:

1. Связан с центральным сервером для получения данных и команд.
2. Взаимодействие с модулем уведомлений и оповещений для информирования пользователей о событиях.

Модуль уведомлений и оповещений (Notifications and Alerts Module)

Функции:

1. Генерация уведомлений и оповещений при возникновении критических ситуаций или отклонений параметров.
2. Отправка уведомлений пользователям через различные каналы.

Связи:

1. Связь с центральным сервером для получения данных о событиях.
2. Взаимодействует с интерфейсом пользователя для отображения уведомлений.

Взаимосвязи компонентов.

Центральный сервер и база данных:

Центральный сервер выполняет операции чтения и записи в базе данных, обеспечивая хранение и доступ к информации.

Центральный сервер и модуль мониторинга и управления:

Центральный сервер получает данные от модуля мониторинга и управления для обработки и принятия решений. В свою очередь, он отправляет команды для управления оборудованием гидроэлектростанции.

Центральный сервер и интерфейс пользователя:

Интерфейс пользователя взаимодействует с центральным сервером для отображения данных и управления системой. Сервер отправляет обновления интерфейсу пользователя и принимает команды от пользователей.

Центральный сервер и модуль уведомлений и оповещений:

Модуль уведомлений и оповещений получает данные о событиях и отклонениях от центрального сервера и отправляет соответствующие уведомления пользователям.

Центральный сервер и аналитический модуль:

Аналитический модуль использует данные, хранящиеся на сервере и в базе данных, для проведения анализа и создания отчетов. Результаты анализа передаются обратно на сервер для дальнейшего использования и отображения.

2.5. РАЗРАБОТКА КОНЦЕПТУАЛЬНОЙ МОДЕЛИ

Разработка концептуальной модели является ключевым этапом проектирования информационной системы, поскольку она предоставляет абстрактное представление данных и их взаимосвязей. Концептуальная модель должна быть четко структурирована и включать все основные сущности, атрибуты и взаимосвязи, необходимые для обеспечения функциональности системы. В данном разделе будет рассмотрен процесс разработки концептуальной модели для информационной системы управления гидроэлектростанцией малой генерации [11].

2.5.1. СУЩНОСТИ И ИХ ОПИСАНИЕ

Таблица Stations (Гидроэлектростанции)

1. StationID (int, Primary Key) - Уникальный идентификатор станции.
2. Name (varchar) - Название станции.
3. Location (varchar) - Местоположение станции.
4. Capacity (float) - Мощность станции в мегаваттах.
5. CommissionDate (date) - Дата ввода в эксплуатацию.

Таблица Turbines (Турбины)

1. TurbineID (int, Primary Key) - Уникальный идентификатор турбины.
2. StationID (int, Foreign Key) - Идентификатор станции, к которой относится турбина.
3. Type (varchar) - Тип турбины.
4. Capacity (float) - Мощность турбины в мегаваттах.

5. InstallationDate (date) - Дата установки турбины.

Таблица Sensors (Датчики)

1. SensorID (int, Primary Key) - Уникальный идентификатор датчика.
2. TurbineID (int, Foreign Key) - Идентификатор турбины, к которой относится датчик.
3. Type (varchar) - Тип датчика (например, температурный, давления, вибрационный и т.д.).
4. InstallationDate (date) - Дата установки датчика.

Таблица Readings (Показания)

1. ReadingID (int, Primary Key) - Уникальный идентификатор показания.
2. SensorID (int, Foreign Key) - Идентификатор датчика, от которого получено показание.
3. Timestamp (datetime) - Время снятия показания.
4. Value (float) - Значение показания.

Таблица Maintenance (Техобслуживание)

1. MaintenanceID (int, Primary Key) - Уникальный идентификатор записи о техобслуживании.
2. TurbineID (int, Foreign Key) - Идентификатор турбины, для которой проведено техобслуживание.
3. Date (date) - Дата проведения техобслуживания.
4. Description (varchar) - Описание проведенных работ.
5. Technician (varchar) - Имя техника, проводившего техобслуживание.

Таблица Personnel (Персонал)

1. PersonnelID (int, Primary Key) - Уникальный идентификатор сотрудника.
2. Name (varchar) - Имя сотрудника.

3. Position (varchar) - Должность.
4. ContactInfo (varchar) - Контактная информация.

2.5.2. ОПРЕДЕЛЕНИЕ СВЯЗЕЙ МЕЖДУ СУЩНОСТЯМИ

1. Одна станция (Stations) может иметь много турбин (Turbines). (1 ко многим)
2. Одна турбина (Turbines) может иметь много датчиков (Sensors). (1 ко многим)
3. Один датчик (Sensors) может иметь много показаний (Readings). (1 ко многим)
4. Одна турбина (Turbines) может иметь много записей о техобслуживании (Maintenance). (1 ко многим)

2.5.3. РАЗРАБОТКА ЛОГИЧЕСКОЙ МОДЕЛИ

Для графического изображения логической модели разработанной базы данных была ER-диаграмма, показанная на рис. 11.

ER-диаграмма (Entity-Relationship Diagram, ERD) – это графическое представление структуры базы данных, которое показывает связи (отношения) между различными сущностями (таблицами) в базе данных. ER-диаграммы используются для моделирования данных и помогают понять, как данные связаны между собой [12].

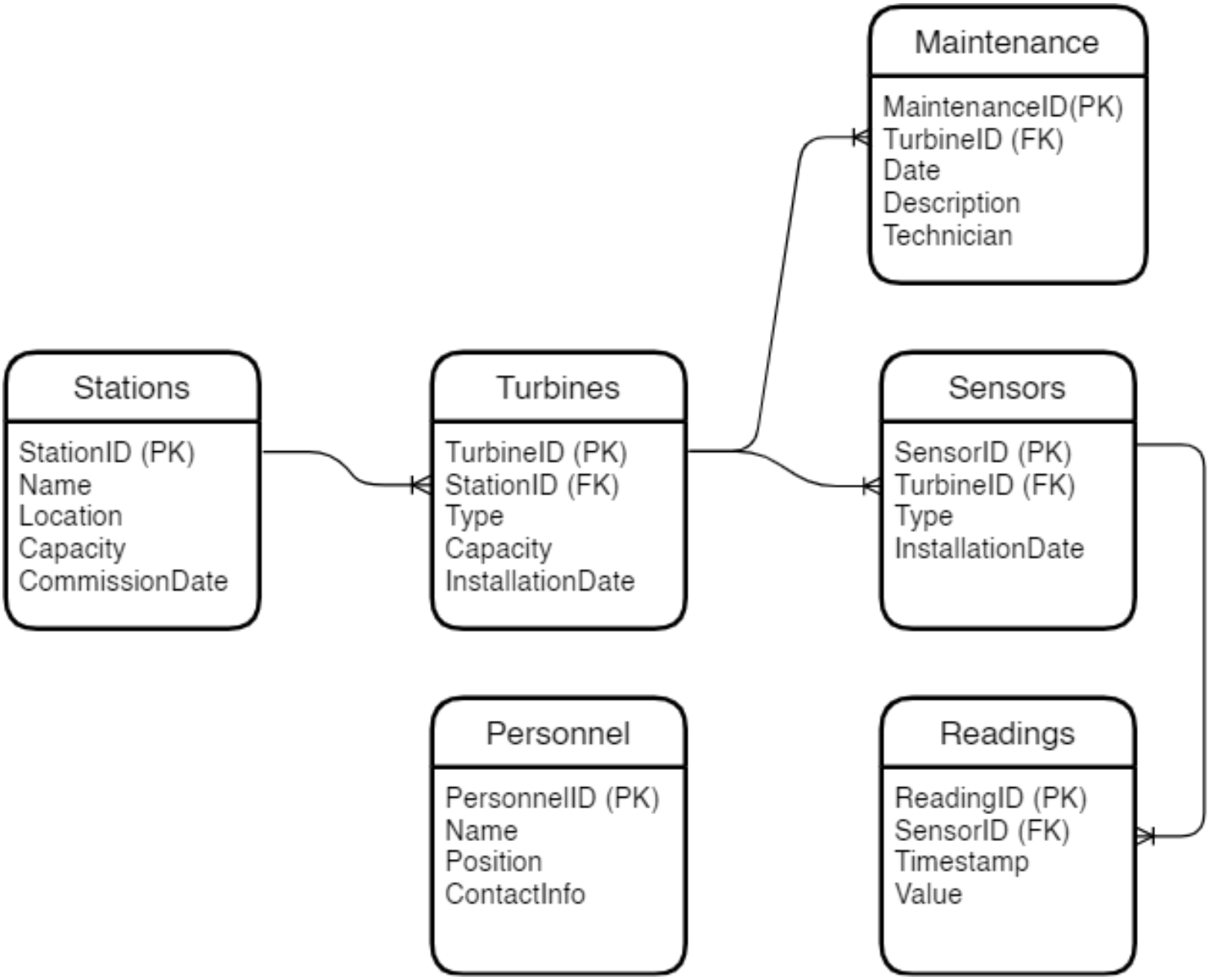

Рисунок 11 – ER-диаграмма базы данных

3. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ИНФОРМАЦИОННОЙ МОДЕЛИ

3.1. СРЕДСТВА РАЗРАБОТКИ

В качестве среды разработки был выбран Pycharm, где на языке программирования Python и подключением двух описанных ниже библиотек начнется программная разработка информационной модели [13].

Реализация информационной модели на языке Python с использованием библиотеки SQLite для создания базы данных и библиотеки Tkinter для разработки графического интерфейса представляет собой процесс, включающий несколько ключевых этапов.

Библиотека SQLite

SQLite является встраиваемой реляционной базой данных, которая позволяет хранить данные в локальном файле без необходимости установки отдельного сервера баз данных. В контексте Python для работы с SQLite используется стандартная библиотека sqlite3. Эта библиотека предоставляет API для создания и управления базами данных SQLite, выполнения SQL запросов, создания таблиц и индексов, а также управления транзакциями.

Библиотека Tkinter

Tkinter является стандартной библиотекой для создания графического пользовательского интерфейса (GUI) в Python. Она предоставляет инструменты для создания различных виджетов (кнопок, полей ввода, текстовых меток и др.), организации компоновки элементов на экране (сетки, упаковщики), а также для обработки событий, взаимодействия с пользователем и обновления интерфейса.

3.2. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

Для программной реализации информационной модели в PyCharm с использованием SQLite и Tkinter требуется настроить окружение и подключить к

базе данных библиотеку SQLite с помощью модуля sqlite3 для создания и управления базой данных. В листинге 1 приведены таблицы базы данных. Далее требуется написать функции и классы для выполнения операций с данными (добавление, обновление, удаление, выборка). Запросы к базе данных приведены в листинге 2. Для разработки графического интерфейса с помощью Tkinter требуется создать основное окно приложения, разместить виджеты (кнопки, метки, поля ввода) на форме и организовать взаимодействие между виджетами и функциональной частью приложения.

Также, были созданы комплексные SQL запросы для базы данных с объединением различных таблиц и последующим выводом этих данных в окно графического интерфейса и возможность добавлять, удалять, изменять и выводить основные данные [14].

В качестве устройства для использования программы выбран стационарный компьютер с операционной системой Windows 10 и комплектующими среднеценового сегмента.

3.3. ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС

Разработка графического интерфейса с использованием библиотеки Tkinter в Python представляет собой важный этап создания пользовательских приложений. Tkinter является стандартным инструментом для создания GUI в Python благодаря своей простоте в использовании и широкому набору предустановленных виджетов.

Основой разработки GUI с Tkinter является создание главного окна приложения, в котором размещены различные элементы интерфейса, такие как кнопки, поля для ввода и текстовые области. В листинге 3 приведен код графического интерфейса и функции работы с элементами управления. Главный экран интерфейса показан на рисунке 12.

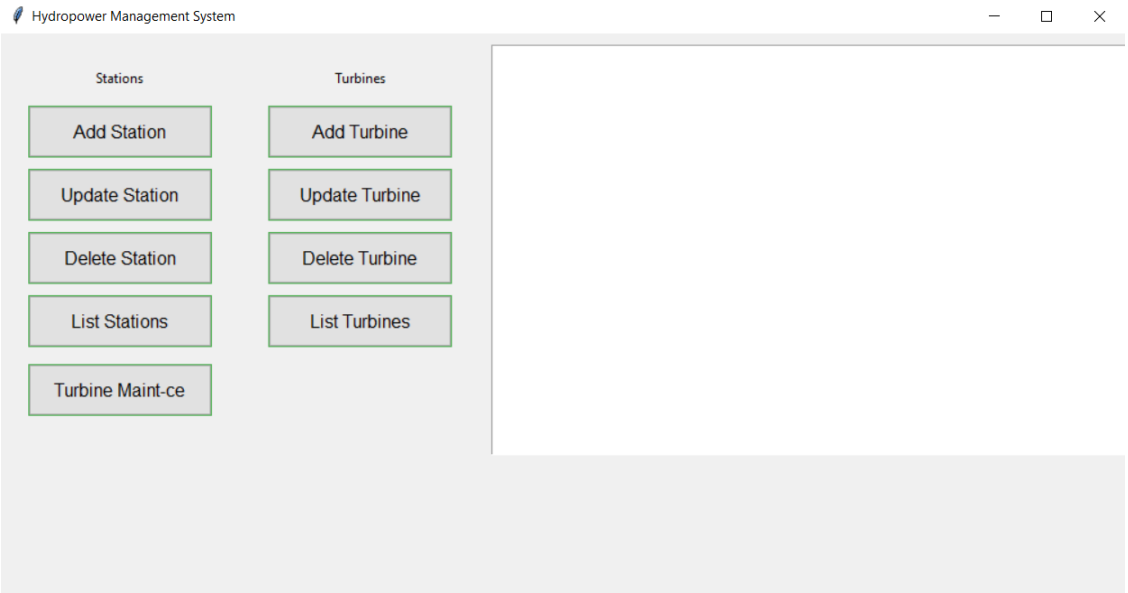

Рисунок 12 – Главное окно программы

Каждый виджет в Tkinter имеет ряд настроек и опций, которые можно задавать в зависимости от дизайна и функциональности приложения. Это включает в себя изменение цвета и шрифта текста, задание размеров и позиции элементов на экране, а также управление событиями, которые будут вызываться при взаимодействии пользователя с интерфейсом.

Помимо основных элементов, таких как кнопки и текстовые поля, с помощью библиотеки Tkinter была организована компоновка элементов на форме. Это достигнуто с использованием различных менеджеров компоновки, таких как grid, pack и place, которые позволяют располагать элементы в окне приложения согласно заданным правилам.

Один из важных аспектов разработки GUI с Tkinter - это обработка событий. В разработанной программе присутствует определение функций, которые будут вызываются при нажатии кнопок. Это позволило приложению реагировать на действия пользователя и обеспечить нужное поведение интерфейса в зависимости от ситуации. На рисунке 13 можно увидеть появившееся окно при нажатии на кнопку «Add Station» [15].

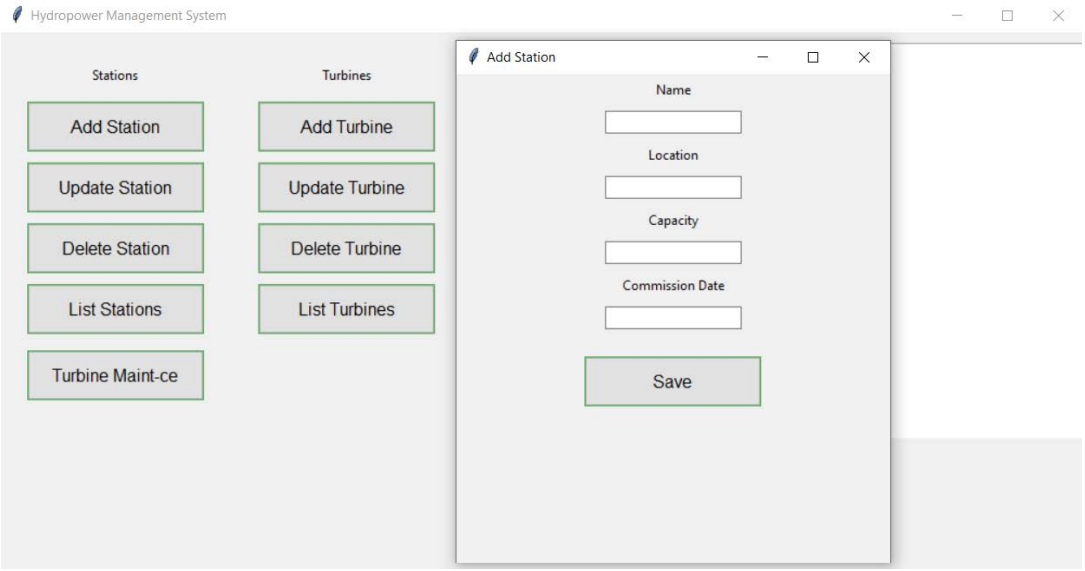

Рисунок 13 – Обработка события нажатия кнопки

Также, при нажатии кнопки «Turbine Maint-ce» (Turbine maintenance) выводится результат сложного запроса к базе данных, а именно объединение информации о станциях, типах турбин, последнем обслуживании и периоде эксплуатации. Результат данного запроса показан в текстовом поле на рисунке 14.

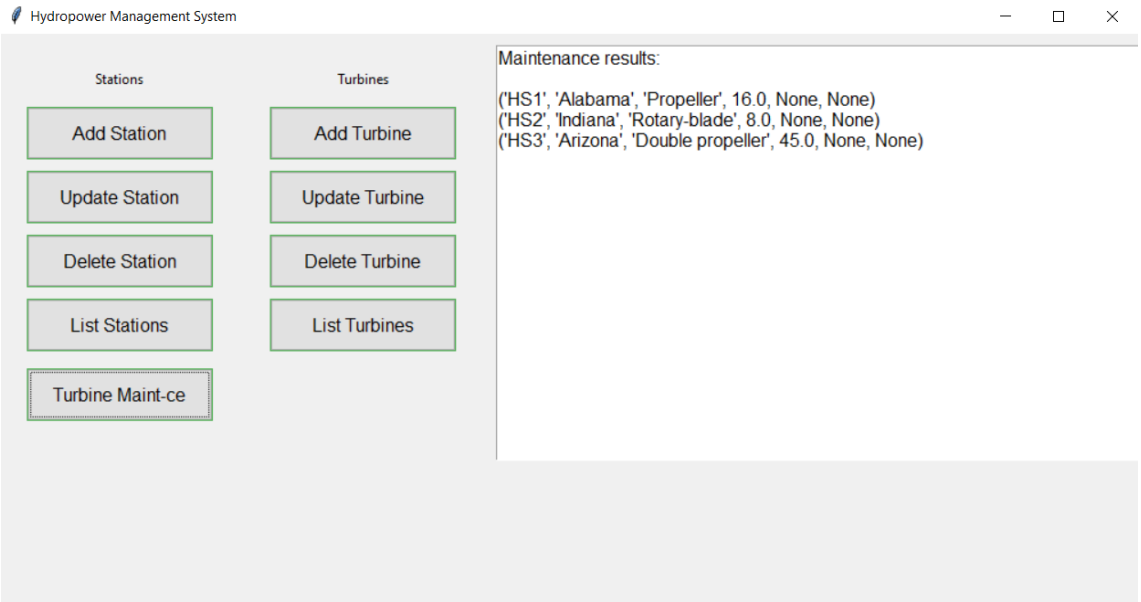

Рисунок 14 – Результат обработки комплексного запроса

4. ТЕСТИРОВАНИЕ ИНФОРМАЦИОННОЙ МОДЕЛИ

Тестирование играет ключевую роль в разработке графического интерфейса для обеспечения его качества, надежности и соответствия требованиям. Оно направлено на проверку взаимодействия пользователя с интерфейсом, а также на обеспечение корректной работы всех функциональных элементов приложения.

Тестирование графического интерфейса (GUI) включает в себя оценку пользовательского опыта и проверку работоспособности элементов интерфейса, таких как кнопки, поля ввода, метки и другие компоненты. Оно не только обеспечивает соответствие дизайна и функциональности заданным требованиям, но и помогает выявить потенциальные ошибки и улучшить удобство использования.

Самые популярные способы тестирования

1. Ручное тестирование: Один из самых распространенных методов, при котором тестировщик вручную проверяет каждый элемент интерфейса на соответствие спецификациям и пользовательским ожиданиям. Ручное тестирование позволяет выявить визуальные и логические ошибки, а также оценить удобство использования.
2. Тестирование пользовательского опыта (Usability testing): Этот вид тестирования фокусируется на оценке удобства использования интерфейса конечными пользователями. Тестировщики наблюдают за действиями пользователей, анализируют их взаимодействие с интерфейсом и собирают обратную связь для дальнейших улучшений.
3. Функциональное тестирование: Проверка функциональности элементов интерфейса с использованием автоматизированных сценариев или тестовых случаев. Функциональное тестирование оценивает работоспособность

кнопок, взаимодействие с формами ввода, правильность отображения данных и другие аспекты работы интерфейса.

В данном случае лучшем решением будет провести функциональное тестирование, так как это позволит активно взаимодействовать с интерфейсом и проверить условия выполнения функциональных требований поспособствует выявлению пользовательских проблем и недочетов, которые могут остаться незамеченными при автоматизированном тестировании.

4.1. ФУНКЦИОНАЛЬНОЕ ТЕСТИРОВАНИЕ

Тестирование всех элементов интерфейса представлено в таблице 1, где в первом столбце описано действие, которое выполняется пользователем, во втором столбце результат работы программы и в третьем – показывается, пройден ли этот тест.

Таблица 1 – Тестирование информационной модели

Действие	**Результат**	**Тест пройден?**
Добавление пользователем нового объекта	Появление нового окна с возможностью добавить объект и сохранить его	Да
Изменение пользователем технических характеристик объекта	Появление нового окна с возможностью изменить объект, обратившись к нему по уникальному идентификатору	Да
Запрос пользователем на предоставление информации по определенному объекту с его характеристиками	Вывод информации в текстовом виде о запрошенном пользователем объекте	Да
Удаление пользователем определенного объекта	Появление нового окна с возможностью указать уникальный идентификатор объекта и сохранить изменения	Да

Функциональное тестирование информационной модели гидроэлектростанций (ГЭС) было направлено на проверку соответствия системы заявленным функциональным требованиям, таким как мониторинг и контроль параметров, управление операциями, отчетность и аналитика, а также уведомления и оповещения.

- Мониторинг и контроль параметров

Тестирование функциональности мониторинга состояния оборудования подтвердило, что система эффективно отслеживает состояние ключевых компонентов гидроэлектростанции.

Особое внимание было уделено визуализации данных. Проведенные тесты продемонстрировали, что разработанная панель отображает ключевые параметры в удобном и информативном формате. Это позволяет операторам легко интерпретировать данные и принимать обоснованные решения на основе визуальной информации.

- Управление операциями

Функциональность оперативного управления оборудованием была протестирована на предмет возможности изменять типы определенных объектов, таких как турбины и станции, а также месторасположение объектов. Результаты тестирования подтвердили, что система обеспечивает гибкость и удобство в управлении элементами ГЭС, позволяя вносить необходимые изменения оперативно и без затруднений.

- Отчетность и аналитика

Тестирование функциональности создания отчетов показало, что система способна генерировать подробные отчеты с основными показателями состава гидроэлектростанции. В процессе тестирования были проверены различные сценарии создания отчетов, и система успешно продемонстрировала способность формировать точные и информативные отчеты, которые могут быть использованы для анализа и принятия решений.

- Уведомления и оповещения

Функциональность информационных уведомлений была проверена на предмет своевременного и корректного уведомления пользователей о выполнении определенных операций, требующих внимания. Тесты показали, что система надежно уведомляет пользователей о важных событиях и изменениях, что позволяет оперативно реагировать на критические ситуации и обеспечивает высокий уровень информированности операторов.

Проведенное функциональное тестирование подтвердило, что информационная модель гидроэлектростанции полностью соответствует заявленным функциональным требованиям. Система демонстрирует высокую эффективность в мониторинге и контроле параметров, обеспечивает удобное и гибкое управление операциями, предоставляет мощные инструменты для отчетности и аналитики, а также надежно уведомляет пользователей о важных событиях. Все это делает систему надежным и эффективным инструментом для управления и анализа данных в сфере гидроэнергетики.

4.2. НЕФУНКЦИОНАЛЬНОЕ ТЕСТИРОВАНИЕ

Нефункциональное тестирование информационной модели гидроэлектростанций (ГЭС) было проведено с целью проверки соблюдения

предъявленных требований по надежности, доступности, масштабируемости, производительности и удобству использования.

- Надежность и доступность

Первоначально проведены тесты на отказоустойчивость системы. В результате тестирования выяснилось, что система способна продолжать работу при возникновении сбоев и неисправностей. Это было подтверждено серией искусственно вызванных отказов, после которых система демонстрировала стабильную работу, сохраняя доступность ключевых функций.

Для оценки среднего времени безотказной работы (MTBF) проводился длительный мониторинг системы в режиме реального времени. Полученные данные подтвердили, что установленное целевое значение MTBF достигнуто, обеспечивая высокую надежность системы.

Мониторинг состояния системы осуществлялся с использованием инструментов, отслеживающих частоту сбоев и время отклика. Результаты мониторинга показывают, что система оперативно реагирует на изменения ключевых показателей надежности, что позволяет своевременно устранять потенциальные проблемы.

Тестирование требований к доступности системы подтвердило, что целевое значение доступности в 99.9% времени соблюдается. В ходе проверки проводилось моделирование различных сценариев нагрузки и отказов, что подтвердило способность системы поддерживать высокий уровень доступности.

Меры по обеспечению непрерывной работы во время пиковых нагрузок также были протестированы. Система продемонстрировала способность работать без перебоев в условиях повышенного спроса, что подтверждает её готовность к обработке критически важных периодов.

- Масштабируемость

Тесты на горизонтальную и вертикальную масштабируемость показали, что система способна к добавлению дополнительных элементов ГЭС и созданию дополнительных функциональных систем для улучшения обработки данных. При увеличении ресурсов (процессоров, памяти) к существующим серверам наблюдалось соответствующее увеличение производительности системы.

Для поддержки больших объемов данных были применены эффективные методы обработки и хранения. Использование распределённых баз данных и хранилищ данных продемонстрировало высокую эффективность в тестах, проводимых с большими объемами информации.

- Производительность

Тестирование требований к скорости обработки данных показало, что система способна обрабатывать запросы в установленные допустимые времена отклика. Оптимизация структуры базы данных и использование индексов способствовали значительному ускорению выполнения операций.

Постоянный мониторинг производительности системы выявил узкие места, которые были оперативно устранены. Использование инструментов для управления и оптимизации производительности позволило достичь высокой эффективности в управлении вычислительными и сетевыми ресурсами.

- Удобство использования

Пользовательский интерфейс (UI) был протестирован на предмет удобства и доступности для пользователей с разным уровнем технической подготовки. Результаты показали, что интерфейс интуитивно понятен и удобен, обеспечивая доступность всех функций на различных операционных системах.

- Сохранение информации

Проверка механизма сохранения информации об объектах и их элементах подтвердила, что данные автоматически сохраняются при выходе из программы, что предотвращает потерю информации и обеспечивает её сохранность.

Проведенное нефункциональное тестирование показало, что информационная модель гидроэлектростанций соответствует всем предъявленным требованиям по надежности, доступности, масштабируемости, производительности и удобству использования. Система демонстрирует высокую устойчивость к сбоям, поддерживает обработку больших объемов данных и обеспечивает непрерывную работу в условиях высокой нагрузки, что делает её надежным инструментом для управления и анализа данных в сфере гидроэнергетики.

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной дипломной работы была проведена комплексная разработка информационной модели для гидроэлектростанций (ГЭС), начиная с анализа научно-технической литературы и заканчивая программной реализацией и тестированием системы.

Первоначально был проведен обзор существующих информационных моделей предметной области. Этот этап позволил выявить основные направления и подходы, используемые в моделировании гидроэлектростанций, а также определить недостатки и возможности для улучшения.

Далее, в процессе разработки структуры информационной модели, был осуществлен сбор и анализ требований, как функциональных, так и нефункциональных. Это позволило сформировать четкое представление о необходимых компонентах системы и их характеристиках.

На этапе программной реализации были выбраны соответствующие средства разработки, обеспечивающие эффективное создание и тестирование системы. Важным аспектом этого этапа стала разработка удобного и интуитивно понятного пользовательского интерфейса, который позволяет пользователям легко взаимодействовать с системой и получать необходимую информацию.

Завершающим этапом работы стало тестирование информационной модели. Проведенное функциональное тестирование подтвердило, что система соответствует всем заявленным функциональным требованиям.

Таким образом, разработанная информационная модель гидроэлектростанций полностью соответствует современным требованиям и обеспечивает эффективное управление и анализ данных в сфере гидроэнергетики.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Информационная модель / [Электронный ресурс] // Wikipedia URL: https://ru.wikipedia.org/wiki/Информационная_модель#:~:text=Информационная%20модель%20%20модель%20объекта%2C,величин%20моделировать%20возможные%20состояния%20объекта (дата обращения: 17.05.2024).

2. Малая энергетика / [Электронный ресурс] // ENPOWER URL: https://enpowertech.ru/blog/malaya-energetika (дата обращения: 17.05.2024).

3. Иштерякова, Т.И. Базы данных / Т. И. Иштерякова // Оренбургский государственный университет – 2009. – 133с.

4. Создание CIM-модели / [Электронный ресурс] // URL: https://www.digital-energy.ru/wp-content/uploads/2021/11/CO_Создание-CIM-модели.pdf (дата обращения: 19.05.2024).

5. Цифровые модели ТЭС - инструмент снижения углеродного следа / [Электронный ресурс] // NBI URL: https://files.mpei.ru/ESGpresent/10_ШатуновБВ_НБИ_Цифровые%20модели%20ТЭС%20-%20инструмент%20снижения%20углеродного%20следа.pdf

6. Информационная модель АЭС и комплексное управление информацией по объекту / [Электронный ресурс] // Атомэнергопроект URL: http://www.atomeks.ru/old/mediafiles/u/files/Atomex_2014/Materials/Section_2/Ergopulo_S.V..pdf (дата обращения: 21.05.2024).

7. Кохановский, В.В. Информационные системы и технологии / В. В. Кохановский // Московский государственный технический университет имени Н.Э. Баумана – 2010. – 256с.

8. Титова, Е.А. Анализ и проектирование информационных систем / Е. А. Титова // Санкт-Петербургский государственный политехнический университет – 2011. – 198с.

9. Смирнов, В.А. Системный анализ и проектирование информационных систем / В. А. Смирнов // Издательство "Лань" – 2014. – 320с.

10. Плотников, А.В. Проектирование информационных систем / А. В. Плотников // Высшая школа экономики – 2011. – 276с.

11. Дейт, К. Дж. Введение в системы баз данных / К. Дж. Дейт // Питер – 2008. – 800с.

12. Мартин, Дж. Управление базами данных / Дж. Мартин // М.: Финансы и статистика – 2006. – 416с.

13. Программирование на Python: базы данных и GUI / [Электронный ресурс] // Python.org URL: https://python.org/doc/tutorial (дата обращения: 25.05.2024).

14. Кузнецов, А.В. SQL и реляционные базы данных / А.В. Кузнецов // Москва: БХВ-Петербург, 2011. – 456с.

15. Захаров, В.М. Tkinter для начинающих / В.М. Захаров // Санкт-Петербург: Питер, 2015. – 320с.

ПРИЛОЖЕНИЕ А

ЛИСТИНГИ ПРОГРАММНОГО КОДА

Листинг 1 – Таблицы разработанной базы данных

```
-- create_database.sql

CREATE TABLE IF NOT EXISTS Stations (
    StationID INTEGER PRIMARY KEY,
    Name TEXT NOT NULL,
    Location TEXT NOT NULL,
    Capacity REAL NOT NULL,
    CommissionDate DATE NOT NULL
);

CREATE TABLE IF NOT EXISTS Turbines (
    TurbineID INTEGER PRIMARY KEY,
    StationID INTEGER,
    Type TEXT NOT NULL,
    Capacity REAL NOT NULL,
    InstallationDate DATE NOT NULL,
    FOREIGN KEY (StationID) REFERENCES Stations (StationID)
);

CREATE TABLE IF NOT EXISTS Sensors (
    SensorID INTEGER PRIMARY KEY,
    TurbineID INTEGER,
    Type TEXT NOT NULL,
    InstallationDate DATE NOT NULL,
    FOREIGN KEY (TurbineID) REFERENCES Turbines (TurbineID)
);

CREATE TABLE IF NOT EXISTS Readings (
    ReadingID INTEGER PRIMARY KEY,
    SensorID INTEGER,
    Timestamp DATETIME NOT NULL,
    Value REAL NOT NULL,
    FOREIGN KEY (SensorID) REFERENCES Sensors (SensorID)
);

CREATE TABLE IF NOT EXISTS Maintenance (
    MaintenanceID INTEGER PRIMARY KEY,
    TurbineID INTEGER,
    Date DATE NOT NULL,
    Description TEXT NOT NULL,
    Technician TEXT NOT NULL,
    FOREIGN KEY (TurbineID) REFERENCES Turbines (TurbineID)
);

CREATE TABLE IF NOT EXISTS Personnel (
    PersonnelID INTEGER PRIMARY KEY,
    Name TEXT NOT NULL,
    Position TEXT NOT NULL,
    ContactInfo TEXT NOT NULL
);
```

Листинг 2 - Инициализация базы данных и функции с запросами к ней

```
import sqlite3

 def initialize_database():
     conn = sqlite3.connect('hydropower.db')
     cursor = conn.cursor()

     with open('sql/create_database.sql', 'r') as f:
         sql = f.read()

     cursor.executescript(sql)
     conn.commit()
     conn.close()


 class Database:
     def __init__(self, db_name='hydropower.db'):
         self.conn = sqlite3.connect(db_name)
         self.cursor = self.conn.cursor()

     def add_station(self, name, location, capacity, commission_date):
         self.cursor.execute("INSERT INTO Stations (Name, Location, Capacity, CommissionDate)
 VALUES (?, ?, ?, ?)",
                             (name, location, capacity, commission_date))
         self.conn.commit()

     def get_stations(self):
         self.cursor.execute("SELECT * FROM Stations")
         return self.cursor.fetchall()

     def update_station(self, station_id, name, location, capacity, commission_date):
         self.cursor.execute("UPDATE    Stations    SET    Name=?,    Location=?,    Capacity=?,
 CommissionDate=? WHERE StationID=?",
                             (name, location, capacity, commission_date, station_id))
         self.conn.commit()

     def delete_station(self, station_id):
         self.cursor.execute("DELETE FROM Stations WHERE StationID=?", (station_id,))
         self.conn.commit()



     def add_turbine(self, station_id, turbine_type, capacity, installation_date):
         self.cursor.execute("INSERT    INTO    Turbines    (StationID,    Type,    Capacity,
 InstallationDate) VALUES (?, ?, ?, ?)",
                             (station_id, turbine_type, capacity, installation_date))
         self.conn.commit()

     def get_turbines(self):
         self.cursor.execute("SELECT * FROM Turbines")
         return self.cursor.fetchall()

     def    update_turbine(self,    turbine_id,    station_id,    turbine_type,    capacity,
 installation_date):
         self.cursor.execute("UPDATE    Turbines    SET    StationID=?,    Type=?,    Capacity=?,
 InstallationDate=? WHERE TurbineID=?",
                             (station_id,    turbine_type,    capacity,    installation_date,
 turbine_id))
         self.conn.commit()
```

```
    def delete_turbine(self, turbine_id):
        self.cursor.execute("DELETE FROM Turbines WHERE TurbineID=?", (turbine_id,))
        self.conn.commit()

    def close(self):
        self.conn.close()
        results = self.cursor.fetchall()
        return results
```

Листинг 3 – Стилицация и интерфейс с функциями для работы с базой данных

```
import tkinter as tk
from tkinter import ttk, messagebox
from database import Database, initialize_database


class HydropowerApp:
    def __init__(self, root):
        self.db = Database()
        self.root = root
        self.root.title('Hydropower Management System')
        self.root.geometry('1000x500')

        style = ttk.Style()
        style.configure('TButton',
                        font=('Helvetica', 12),
                        padding=10,
                        background='#4CAF50',
                        foreground='black',
                        borderwidth=0,
                        width=15,
                        focuscolor='#4CAF50',
                        focusthickness=0,
                        )

        self.station_frame = ttk.Frame(root, padding="10")
        self.station_frame.grid(row=0, column=0, padx=10, pady=10, sticky='n')

        self.station_label = ttk.Label(self.station_frame, text='Stations')
        self.station_label.grid(row=0, column=0, pady=10)

        self.add_station_button = ttk.Button(self.station_frame, text='Add Station',
command=self.add_station)
        self.add_station_button.grid(row=1, column=0, padx=5, pady=5)

        self.update_station_button = ttk.Button(self.station_frame, text='Update Station',
command=self.update_station)
        self.update_station_button.grid(row=2, column=0, padx=5, pady=5)

        self.delete_station_button = ttk.Button(self.station_frame, text='Delete Station',
command=self.delete_station)
        self.delete_station_button.grid(row=3, column=0, padx=5, pady=5)

        self.list_stations_button = ttk.Button(self.station_frame, text='List Stations',
command=self.list_stations)
        self.list_stations_button.grid(row=4, column=0, padx=5, pady=5)

        self.turbine_frame = ttk.Frame(root, padding="10")
        self.turbine_frame.grid(row=0, column=1, padx=10, pady=10, sticky='n')
```

```
        self.turbine_label = ttk.Label(self.turbine_frame, text='Turbines')
        self.turbine_label.grid(row=0, column=0, pady=10)

        self.add_turbine_button = ttk.Button(self.turbine_frame, text='Add Turbine',
command=self.add_turbine)
        self.add_turbine_button.grid(row=1, column=0, padx=5, pady=5)

        self.update_turbine_button = ttk.Button(self.turbine_frame, text='Update Turbine',
command=self.update_turbine)
        self.update_turbine_button.grid(row=2, column=0, padx=5, pady=5)

        self.delete_turbine_button = ttk.Button(self.turbine_frame, text='Delete Turbine',
command=self.delete_turbine)
        self.delete_turbine_button.grid(row=3, column=0, padx=5, pady=5)

        self.list_turbines_button = ttk.Button(self.turbine_frame, text='List Turbines',
command=self.list_turbines)
        self.list_turbines_button.grid(row=4, column=0, padx=5, pady=5)

        self.output_text = tk.Text(root, height=20, width=90, font=('Helvetica', 12))
        self.output_text.grid(row=0, column=2, padx=10, pady=10, sticky='nsew')

        self.custom_query_button = ttk.Button(self.station_frame, text='Turbine Maint-ce',
command=self.custom_query)
        self.custom_query_button.grid(row=5, column=0, columnspan=2, pady=10)

    def add_station(self):
        def save_station():
            name = name_entry.get()
            location = location_entry.get()
            capacity = float(capacity_entry.get())
            commission_date = commission_date_entry.get()
            self.db.add_station(name, location, capacity, commission_date)
            messagebox.showinfo('Info', 'Station added successfully')
            add_window.destroy()

        add_window = tk.Toplevel(self.root)
        add_window.title('Add Station')
        add_window.geometry('400x450')

        name_label = ttk.Label(add_window, text='Name')
        name_label.pack(pady=5)
        name_entry = ttk.Entry(add_window)
        name_entry.pack(pady=5)

        location_label = ttk.Label(add_window, text='Location')
        location_label.pack(pady=5)
        location_entry = ttk.Entry(add_window)
        location_entry.pack(pady=5)

        capacity_label = ttk.Label(add_window, text='Capacity')
        capacity_label.pack(pady=5)
        capacity_entry = ttk.Entry(add_window)
        capacity_entry.pack(pady=5)

        commission_date_label = ttk.Label(add_window, text='Commission Date')
        commission_date_label.pack(pady=5)
        commission_date_entry = ttk.Entry(add_window)
        commission_date_entry.pack(pady=5)

        save_button = ttk.Button(add_window, text='Save', command=save_station)
        save_button.pack(pady=20)
```

```
def update_station(self):
    def save_update():
        station_id = int(id_entry.get())
        name = name_entry.get()
        location = location_entry.get()
        capacity = float(capacity_entry.get())
        commission_date = commission_date_entry.get()
        self.db.update_station(station_id, name, location, capacity, commission_date)
        messagebox.showinfo('Info', 'Station updated successfully')
        update_window.destroy()

    update_window = tk.Toplevel(self.root)
    update_window.title('Update Station')
    update_window.geometry('400x450')

    id_label = ttk.Label(update_window, text='Station ID')
    id_label.pack(pady=5)
    id_entry = ttk.Entry(update_window)
    id_entry.pack(pady=5)

    name_label = ttk.Label(update_window, text='Name')
    name_label.pack(pady=5)
    name_entry = ttk.Entry(update_window)
    name_entry.pack(pady=5)

    location_label = ttk.Label(update_window, text='Location')
    location_label.pack(pady=5)
    location_entry = ttk.Entry(update_window)
    location_entry.pack(pady=5)

    capacity_label = ttk.Label(update_window, text='Capacity')
    capacity_label.pack(pady=5)
    capacity_entry = ttk.Entry(update_window)
    capacity_entry.pack(pady=5)

    commission_date_label = ttk.Label(update_window, text='Commission Date')
    commission_date_label.pack(pady=5)
    commission_date_entry = ttk.Entry(update_window)
    commission_date_entry.pack(pady=5)

    save_button = ttk.Button(update_window, text='Save', command=save_update)
    save_button.pack(pady=20)

def delete_station(self):
    def delete():
        station_id = int(id_entry.get())
        self.db.delete_station(station_id)
        messagebox.showinfo('Info', 'Station deleted successfully')
        delete_window.destroy()

    delete_window = tk.Toplevel(self.root)
    delete_window.title('Delete Station')
    delete_window.geometry('300x150')

    id_label = ttk.Label(delete_window, text='Station ID')
    id_label.pack(pady=5)
    id_entry = ttk.Entry(delete_window)
    id_entry.pack(pady=5)

    delete_button = ttk.Button(delete_window, text='Delete', command=delete)
    delete_button.pack(pady=20)
```

```
    def list_stations(self):
        stations = self.db.get_stations()
        self.output_text.delete('1.0', tk.END)
        for station in stations:
            self.output_text.insert(tk.END, f"{station}\n")

    def add_turbine(self):
        def save_turbine():
            station_id = int(station_id_entry.get())
            turbine_type = type_entry.get()
            capacity = float(capacity_entry.get())
            installation_date = installation_date_entry.get()
            self.db.add_turbine(station_id, turbine_type, capacity, installation_date)
            messagebox.showinfo('Info', 'Turbine added successfully')
            add_window.destroy()

        add_window = tk.Toplevel(self.root)
        add_window.title('Add Turbine')
        add_window.geometry('400x450')

        station_id_label = ttk.Label(add_window, text='Station ID')
        station_id_label.pack(pady=5)
        station_id_entry = ttk.Entry(add_window)
        station_id_entry.pack(pady=5)

        type_label = ttk.Label(add_window, text='Type')
        type_label.pack(pady=5)
        type_entry = ttk.Entry(add_window)
        type_entry.pack(pady=5)

        capacity_label = ttk.Label(add_window, text='Capacity')
        capacity_label.pack(pady=5)
        capacity_entry = ttk.Entry(add_window)
        capacity_entry.pack(pady=5)

        installation_date_label = ttk.Label(add_window, text='Installation Date')
        installation_date_label.pack(pady=5)
        installation_date_entry = ttk.Entry(add_window)
        installation_date_entry.pack(pady=5)

        save_button = ttk.Button(add_window, text='Save', command=save_turbine)
        save_button.pack(pady=20)

    def update_turbine(self):
        def save_update():
            turbine_id = int(turbine_id_entry.get())
            station_id = int(station_id_entry.get())
            turbine_type = type_entry.get()
            capacity = float(capacity_entry.get())
            installation_date = installation_date_entry.get()
            self.db.update_turbine(turbine_id,     station_id,     turbine_type,     capacity,
installation_date)
            messagebox.showinfo('Info', 'Turbine updated successfully')
            update_window.destroy()

        update_window = tk.Toplevel(self.root)
        update_window.title('Update Turbine')
        update_window.geometry('400x450')

        turbine_id_label = ttk.Label(update_window, text='Turbine ID')
        turbine_id_label.pack(pady=5)
```

```
        turbine_id_entry = ttk.Entry(update_window)
        turbine_id_entry.pack(pady=5)

        station_id_label = ttk.Label(update_window, text='Station ID')
        station_id_label.pack(pady=5)
        station_id_entry = ttk.Entry(update_window)
        station_id_entry.pack(pady=5)

        type_label = ttk.Label(update_window, text='Type')
        type_label.pack(pady=5)
        type_entry = ttk.Entry(update_window)
        type_entry.pack(pady=5)

        capacity_label = ttk.Label(update_window, text='Capacity')
        capacity_label.pack(pady=5)
        capacity_entry = ttk.Entry(update_window)
        capacity_entry.pack(pady=5)

        installation_date_label = ttk.Label(update_window, text='Installation Date')
        installation_date_label.pack(pady=5)
        installation_date_entry = ttk.Entry(update_window)
        installation_date_entry.pack(pady=5)

        save_button = ttk.Button(update_window, text='Save', command=save_update)
        save_button.pack(pady=20)

    def delete_turbine(self):
        def delete():
            turbine_id = int(turbine_id_entry.get())
            self.db.delete_turbine(turbine_id)
            messagebox.showinfo('Info', 'Turbine deleted successfully')
            delete_window.destroy()

        delete_window = tk.Toplevel(self.root)
        delete_window.title('Delete Turbine')
        delete_window.geometry('300x150')

        turbine_id_label = ttk.Label(delete_window, text='Turbine ID')
        turbine_id_label.pack(pady=5)
        turbine_id_entry = ttk.Entry(delete_window)
        turbine_id_entry.pack(pady=5)

        delete_button = ttk.Button(delete_window, text='Delete', command=delete)
        delete_button.pack(pady=20)

    def list_turbines(self):
        turbines = self.db.get_turbines()
        self.output_text.delete('1.0', tk.END)
        for turbine in turbines:
            self.output_text.insert(tk.END, f"{turbine}\n")

    def custom_query(self):
        query = '''
        SELECT
            st.Name AS StationName,
            st.Location AS StationLocation,
            tu.Type AS TurbineType,
            tu.Capacity AS TurbineCapacity,
            MAX(mt.Date) AS LastMaintenanceDate,
            (julianday('now') - julianday(tu.InstallationDate)) AS OperatingDays
        FROM
            Stations st
```

```
        INNER JOIN
            Turbines tu ON st.StationID = tu.StationID
        LEFT JOIN
            Maintenance mt ON tu.TurbineID = mt.TurbineID
        GROUP BY
            st.StationID, tu.TurbineID
        ORDER BY
            st.Name, tu.Type;
        '''

        self.output_text.delete('1.0', tk.END)
        self.output_text.insert(tk.END, "Maintenance results:\n\n")
        try:
            self.db.cursor.execute(query)
            results = self.db.cursor.fetchall()
            for result in results:
                self.output_text.insert(tk.END, f"{result}\n")
        except Exception as e:
            messagebox.showerror("Error", f"Error executing query: {str(e)}")

if __name__ == '__main__':
    initialize_database()
    root = tk.Tk()
    app = HydropowerApp(root)
    root.mainloop()
```