СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc183573730)

[1. ОБЩАЯ ЧАСТЬ 4](#_Toc183573731)

[1.1. Цель разработки 4](#_Toc183573732)

[2. СПЕЦИАЛЬНАЯ ЧАСТЬ 6](#_Toc183573733)

[2.1. Постановка задачи 6](#_Toc183573734)

[2.1.1. Входные данные предметной области 6](#_Toc183573735)

[2.1.2. Требования к проекту 7](#_Toc183573736)

[2.2. Внешняя спецификация 8](#_Toc183573737)

[2.2.1. Описание задачи 8](#_Toc183573738)

[2.2.2. Входные и выходные данные 14](#_Toc183573739)

[2.2.3. Методы 17](#_Toc183573740)

[2.2.4. Тесты 21](#_Toc183573741)

[2.2.5. Контроль целостности данных 22](#_Toc183573742)

[2.3. Проектирование 23](#_Toc183573743)

[2.3.1. Схема архитектуры приложения 23](#_Toc183573744)

[2.3.2. Логическая схема данных 24](#_Toc183573745)

[2.3.3. Физическая схема данных 25](#_Toc183573746)

[2.3.4. Структурная схема 26](#_Toc183573747)

[2.3.5. Функциональная схема 26](#_Toc183573748)

[2.3.6. Диаграмма классов 27](#_Toc183573749)

[2.3.7. Схема тестирования 27](#_Toc183573750)

[2.3.8. Схема пользовательского интерфейса 28](#_Toc183573751)

[2.4. Результат работы программы 30](#_Toc183573752)

[3. ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ 35](#_Toc183573753)

[3.1. Инструментальные средства 35](#_Toc183573754)

[3.2. Отладка программы 36](#_Toc183573755)

[3.3. Защитное программирование 36](#_Toc183573756)

[3.4. Характеристики программы 36](#_Toc183573757)

[ЗАКЛЮЧЕНИЕ 37](#_Toc183573758)

[СПИСОК ИСПОЛЬЗУЕМЫХ МАТЕРИАЛОВ 39](#_Toc183573759)

# ВВЕДЕНИЕ

С развитием технологий и увеличением числа электронных устройств в промышленных и производственных процессах, возникла необходимость в эффективном отслеживании их показателей. Эти устройства часто используются для автоматизации и управления различными системами, и их надежная работа является критически важной для бесперебойного функционирования предприятий. Однако, несмотря на наличие различных инструментов и программ для мониторинга, многие организации сталкиваются с трудностями в централизованном управлении и анализе данных с таких устройств.

В связи с этим в данной дипломной работе разработан программный комплекс контроля аварийных ситуаций устройств (на примере АО «Корпорация Комета»). Данная система предоставляет функционал для сбора, анализа и визуализации данных с устройств, используемых на предприятиях. Она будет включать в себя удобный интерфейс для отслеживания устройств, возможность отображения устройств при выходе показателей за допустимые пределы, что позволит пользователям легко отслеживать состояние своих устройств и принимать своевременные меры для их оптимальной работы.

# ОБЩАЯ ЧАСТЬ

## Цель разработки

Целью разработки является обеспечение централизованного мониторинга и управления электронными устройствами на предприятии.

* 1. Средства разработки

Программные средства, используемые для проектирования, разработки и тестирования приложения представлены в таблице 1.

Таблица – Программные средства

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тип средства | Название средства | Назначение |
| 1 | 2 | 3 | 4 |
| 1 | Операционные системы | Astra Linux 1.6 | Организация взаимодействия  программ и пользователя |
| 2 | Операционные системы | Windows 10 | Организация взаимодействия  программ и пользователя |
| 3 | Операционные системы | Android 13 | Организация взаимодействия  программ и пользователя |
| 4 | Среда разработки | Qt Creator 4.15 | Среда запуска |
| 5 | Среда разработки | Android Studio 17.0.9 | Среда запуска |
| 6 | Сервер базы данных | PostgreSQL 9.4 | Хранение данных |
| 7 | Графическая оболочка | pgAdmin 3 | Разработка базы данных и функционала работы с ней |

В качестве средств вычислительной техники при разработке ПО использовался персональный компьютер, ноутбук и телефон. Характеристики представлены в Таблице 2.

Таблица - Технические характеристики ПК при разработке ПО

| № | Тип оборудования | Наименование оборудования | |
| --- | --- | --- | --- |
| 1 | 2 | 3 | |
| Для разработки | | | |
| 1 | Процессор | Intel Core Duo e7500 | |
| 2 | Оперативная память | 2 GB | |
| 4 | Тип видеокарты | Встроенная | |
| 5 | Видеокарта | AMD Radeon Graphics | |
| 6 | Конфигурация накопителей | SSD | |
| 7 | Общий объем накопителей | 256 ГБ | |
| 8 | Компьютерная мышь | Logitech M100R | |
| 9 | Клавиатура | Любая рабочая клавиатура с английской и русской раскладкой | |
| 10 | Операционная система | Astra Linux | |
| Для разработки | | | |
| 1 | Процессор | AMD Ryzen 3 3250U | |
| 2 | Оперативная память | 16 GB | |
| 4 | Тип видеокарты | Встроенная | |
| 5 | Видеокарта | AMD Radeon Graphics | |
| 6 | Конфигурация накопителей | SSD | |
| 7 | Общий объем накопителей | 256 ГБ | |
| 8 | Компьютерная мышь | Qumo Techno | |
| 9 | Клавиатура | Любая рабочая клавиатура с английской и русской раскладкой | |
| 10 | Операционная система | Windows 10 | |
| Для тестирования | | | |
| 1 | Процессор | | AMD Ryzen 3 3250U |
| 2 | Оперативная память | | 16 GB |
| 4 | Тип видеокарты | | Встроенная |
| 5 | Видеокарта | | AMD Radeon Graphics |
| 6 | Конфигурация накопителей | | SSD |
| 7 | Общий объем накопителей | | 256 ГБ |
| 8 | Компьютерная мышь | | Qumo Techno |
| 9 | Клавиатура | | Любая рабочая клавиатура с английской и русской раскладкой |
| 10 | Операционная система | | Windows 10 |

# СПЕЦИАЛЬНАЯ ЧАСТЬ

## Постановка задачи

Разработать программный комплекс «DeviceMonitor» который представляет из себя комплекс из двух Desktop-приложений и одного мобильного контроля аварийных ситуаций устройств.

## Входные данные предметной области

Входными данными программного комплекса «DeviceMonitor» являются следующие данные:

* Добавление пользователя (Логин, пароль, роль);
* Добавление устройства (Название, напряжение, минимальное напряжение, максимальное напряжение, ток, максимальный ток, температура, максимальная температура, емкость, название, расположение, включено ли);
* Добавление расположения (Название);
* Фильтрация устройств (Минимальное значение напряжения, минимальное значение тока, минимальное значение);
* Сетевые данные для подключения к серверу (IP-адрес, порт)

Выходные данные предметной области

* Пользователи (Логин, роль, дата регистрации);
* Устройства (Название, напряжение, минимальное напряжение, максимальное напряжение, ток, максимальный ток, температура, максимальная температура, емкость, название, расположение, включено ли, дата получения данных);
* Расположения (Название);
* История аварийных случаев (Дата аварии, копия данных устройства в момент аварии: Название, напряжение, минимальное напряжение, максимальное напряжение, ток, максимальный ток, температура, максимальная температура, емкость, название, расположение, включено ли, дата получения данных);

## Требования к проекту

Требования к программному комплексу «DeviceMonitor» предоставлены ниже:

* Многопользовательская поддержка – данные о пользователях и их ролях должны хранится в БД;
* В программном комплексе должно быть две роли: Администратор и Пользователь;
* Администратор может пользоваться админ-панелью и не может зайти в клиентскую версию программы;
* Администратор может запустить сервер, авторизоваться в админ-панели, подключить счетчик, отслеживать устройства, просматривать аварийные ситуации, просматривать историю аварийных ситуаций, добавлять и изменять устройства, добавлять и изменять пользователей, добавлять и изменять расположения;
* Пользователь может пользоваться клиентской версией программы и не может зайти в админ-панель;
* Пользователь в клиентской версии программы может подключиться к серверу, авторизоваться, отслеживать устройства, просматривать историю аварийных ситуаций, фильтровать устройства по изменяемым физическим показателям (ток, напряжение, температура), просматривать динамический график показателей устройств, сохранять данные об устройствах в формате json и csv;
* Приложение должно иметь простой и интуитивный интерфейс;
* Интерфейс и цветовая палитра приложения не должны ярко контрастировать и напрягать зрение;
* Приложение должно предоставлять возможность добавлять и отслеживать устройства;
* Вход в админ-панель должен начинаться с ввода логина и пароля от аккаунта;
* Приложение должно позволять формировать историю аварийных случаев с показателями и датами аварий;
* Серверное Desktop-приложение должно запускать сервер, принимающий подключения на получение данных об устройствах и других данных от клиентских версий приложения и отправляющий ответы в формате json-файла по протоколу TCP;
* Все входящие запросы и информация о подключениях к серверу должны быть отображены на основном экране в виде объемной текстовой строки;
* Серверное приложение должно предоставлять возможность использовать дополнительный режима работы приложения, в котором сервер по TCP подключается к серверу-отслеживателю как клиент, к которому подключены счетчики, для извлечения из него данных и записи их в БД;
* Клиентское приложение должно предоставлять возможность подключения к серверу по IP-адресу и порту приложения, после чего предоставлять возможность авторизации и дальнейшей работы программы, получая данные с подключенного сервера.
* Пароль должен храниться в виде хэша формата sha256;
* База данных должна быть приведена в 3 нормальную форму.

## Внешняя спецификация

## Описание задачи

Основной задачей является разработка программного решения, которое будет упрощать процесс взаимодействия между арендодателями, арендаторами и администраторами в сфере аренды недвижимости.

На этапе анализа были определены ключевые роли пользователей, такие как администратор, модератор и пользователь (арендодатель и арендатор), а также процессы, связанные с созданием, редактированием и управлением объявлениями, поиском и фильтрацией объектов недвижимости, и взаимодействием между сторонами.

Диаграммы прецедентов и модели AS-IS и TO-BE описаны в приложении Д «Бизнес процессы».

## Входные и выходные данные

На таблице 3 изображены входные данные программного комплекса контроля аварийных ситуаций устройств.

Таблица – Входные данные

| Имя | Ограничения | Обязательное | Структура, формат | Форма ввода |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| Входные данные | | | | |
| Логин | Максимальная длина строки 50 символов | Да | Строка | Текстовое поле |
| Пароль | Максимальная длина строки 50 символов | Да | Строка | Текстовое поле |
| Название расположения | Максимальная длина строки 50 символов | Да | Строка | Текстовое поле |
| Название устройства | Максимальная длина строки 70 символов | Да | Строка | Текстовое поле |
| Напряжение |  | Да | Число | Текстовое поле |
| Минимальное напряжение |  | Да | Число | Текстовое поле |
| Максимальное напряжение |  | Да | Число | Текстовое поле |
| Ток |  | Да | Число | Текстовое поле |
| Максимальный ток |  | Да | Число | Текстовое поле |
| Емкость |  | Да | Число | Текстовое поле |
| Температура |  | Да | Число | Текстовое поле |
| Максимальная температура |  | Да | Число | Текстовое поле |
| IP-адрес |  | Да | Строка | Текстовое поле |
| Порт |  | Да | Число | Текстовое поле |

На таблице 4 изображены выходные данные программного комплекса.

Таблица – Выходные данные

| Имя | Ограничения | Обязательное | Структура, формат | Форма ввода |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| Выходные данные | | | | |
| Логин | Максимальная длина строки 50 символов | Да | Строка | Текстовое поле |
| Пароль | Максимальная длина строки 50 символов | Да | Строка | Текстовое поле |
| Название расположения | Максимальная длина строки 50 символов | Да | Строка | Текстовое поле |
| Дата регистрации |  | Да | Строка | Текстовое поле |
| Дата аварийной ситуации |  | Да | Строка | Текстовое поле |
| Дата добавления данных устройства |  | Да | Строка | Текстовое поле |
| Название устройства | Максимальная длина строки 70 символов | Да | Строка | Текстовое поле |
| Напряжение |  | Да | Число | Текстовое поле |
| Минимальное напряжение |  | Да | Число | Текстовое поле |
| Максимальное напряжение |  | Да | Число | Текстовое поле |
| Ток |  | Да | Число | Текстовое поле |
| Максимальный ток |  | Да | Число | Текстовое поле |
| Емкость |  | Да | Число | Текстовое поле |
| Температура |  | Да | Число | Текстовое поле |
| Максимальная температура |  | Да | Число | Текстовое поле |

## Методы

Возможность объектов различных классов реагировать на одни и те же запросы с использованием общего интерфейса. Это позволяет использовать один и тот же метод для разных объектов, что упрощает кодирование и повышает его гибкость.

Использована инкапсуляция, которая позволяет защитить данные от нежелательного доступа и изменений, а также облегчить поддержку кода и его повторное использование с помощью скрытии определённых данных программы от других данных. Также, использовался рефакторинг и оптимизация.

На рисунке 12 представлен пример без оптимизации. На рисунке представлен пример кода без оптимизации. Данный код содержит множество проблем, которые делают его небезопасным и неоптимизированным.

В коде отсутствует проверка типов данных при обработке значений JSON-объекта. Это может привести к ошибкам, если значения не соответствуют ожидаемому типу, например, если ожидается строка, но передано число. Такие ошибки могут вызвать аварийное завершение программы или некорректное выполнение логики.

Также, код не проверяет наличие ключа "checksum" перед его использованием. Если этот ключ отсутствует в JSON-объекте, программа может столкнуться с исключением, что сделает её уязвимой к сбоям.

Кроме того, используется неэффективный способ чтения данных из потока. Вместо использования стандартных методов чтения, код выполняет лишние операции, что замедляет его выполнение и увеличивает нагрузку на систему. Это особенно критично при работе с большими объемами данных, где производительность имеет первостепенное значение.

Также отсутствует проверка корректности контрольной суммы, что делает передачу данных ненадежной. Без этой проверки невозможно гарантировать целостность данных, что может привести к использованию поврежденных или поддельных данных.

Код содержит избыточные преобразования данных, такие как преобразование JSON-объекта обратно в строку, что является лишней операцией и замедляет выполнение программы. Эти преобразования не только увеличивают время выполнения, но и потребляют дополнительные ресурсы памяти, что недопустимо в высоконагруженных системах.

Наконец, код использует неэффективный способ обработки ошибок, что также замедляет его выполнение. Вместо оптимальных алгоритмов обработки исключений используются методы, которые увеличивают сложность операций и снижают общую производительность программы.

Все эти проблемы делают код непригодным для использования в реальных приложениях, где требуется надежность, безопасность и высокая производительность.

Оптимизированный код демонстрирует хорошие практики обработки данных и обеспечения безопасности. Он корректно проверяет типы значений в JSON-объекте, что предотвращает ошибки при несоответствии типов. Код также включает проверку наличия ключа "checksum" и выполняет проверку контрольной суммы, что гарантирует целостность данных. Использование эффективных методов чтения данных и обработки исключений обеспечивает высокую производительность. Этот код надежен, безопасен и оптимизирован для работы с JSON-данными, что делает его подходящим для использования в реальных приложениях.



Рисунок – Неоптимизированный код

На рисунке 13 представлен оптимизированный код функционала.



Рисунок – Оптимизированный код

На рисунке 14 представлено время исполнения неоптимизированного кода.

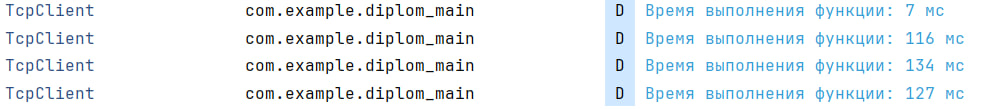


Рисунок – Время исполнения неоптимизированного кода

На рисунке 15 представлено время исполнения оптимизированногор кода:

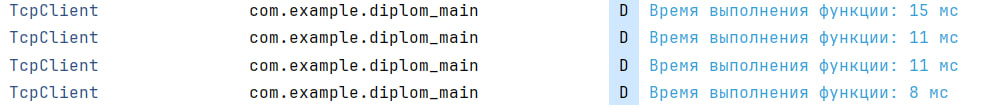


Рисунок – Время исполнения оптимизированного кода

## Тесты

Тестирование приложения описано в ПРИЛОЖЕНИИ Б «Сценарий и результаты тестовых испытаний».

## Контроль целостности данных

В таблице 5 изображен алгоритм контроля целостности данных приложения контроля аварийных ситуаций устройств.

Таблица 5 – Контроль целостности данных

| № | Ситуация | Аномалия | Реакция | Примечание |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Авторизация | Введён несуществующий логин и/или пароль и/или введены некорректные, согласно ограничениям, данные в соответствующие поля | Приложение обработает ответ базы данных и выведет пользователю диалоговое окно о несовпадении логина и пароля | Авторизация возможна только по существующим в БД данным |
| 2 | Добавление устройства | Не введено название устройства | Приложение сообщает о пустом поле, данные не добавляются | Данные при добавлении устройства не могут быть пустыми |
| Введены некорректные, согласно ограничениям, данные в соответствующие поля | Приложение обработает ошибку с базы данных и выведет пользователю диалоговое окно с содержанием  ошибки | Добавление устройства возможно только с валидными полями, соответствующим ограничениям полей |
| 3 | Добавление расположения | Не введено название расположения | Приложение сообщает о пустом поле, данные не добавляются | Данные при добавлении расположения не могут быть пустыми |
| Введены некорректные, согласно ограничениям, данные в соответствующие поля | Приложение обработает ошибку с базы данных и выведет пользователю диалоговое окно с содержанием  ошибки | Добавление расположения возможно только с валидными полями, соответствующим ограничениям полей |
| 4 | Подключение счетчика | Введены пустые или некорректные, данные в соответствующие поля | Приложение обработает ошибку подключения и выведет пользователю диалоговое окно с сообщением о неверных данных подключения | Данные при подключении счетчика должны быть корректными, и не пустыми, а на самом адресе должен быть запущен сервер |
| 5 | Подключение к серверу | Введены пустые или некорректные, данные в соответствующие поля | Приложение обработает ошибку подключения и выведет пользователю диалоговое окно с сообщением о неверных данных подключения | Данные при подключении счетчика должны быть корректными, и не пустыми, а на самом адресе должен быть запущен сервер |
| 6 | Фильтрация устройств | Введены пустые или некорректные, данные в соответствующие поля | Приложение обработает ошибку данных и выведет пользователю диалоговое окно с сообщением о неверных данных подключения | Данные при подключении счетчика должны быть корректными, и не пустыми, а на самом адресе должен быть запущен сервер |

## Проектирование

## Схема архитектуры приложения

Компонент пользовательский интерфейс виден пользователю и взаимодействует с ним. В зависимости от действий пользователя осуществляются запросы на сервер на котором хранятся данные, который возвращает ответы на запрос в виде данных.

На Рисунке 16 представлена архитектурная схема приложения.

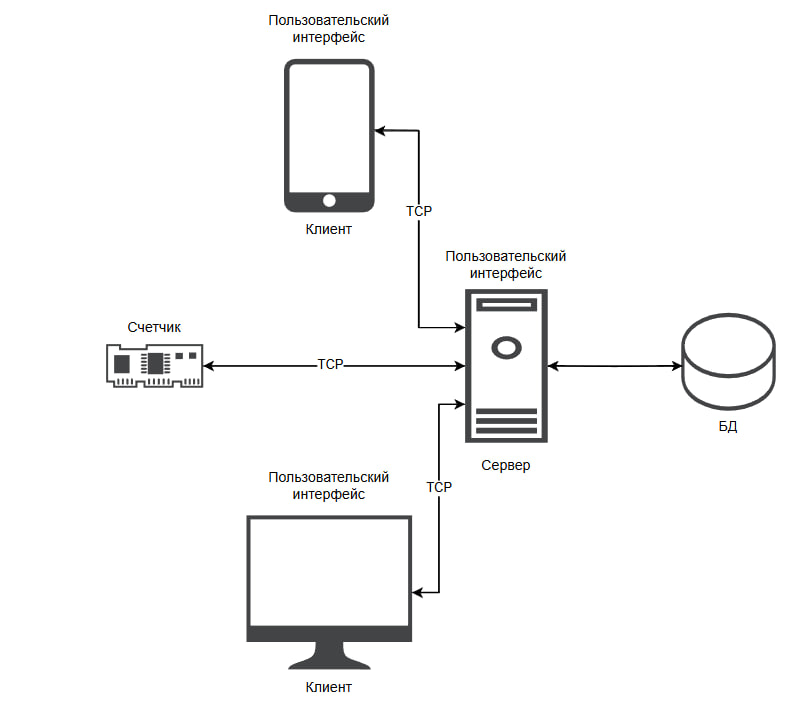


Рисунок – Архитектурная схема приложения

## Логическая схема данных

Разработанная логическая модель базы данных, представленная на Рисунке 17, послужила основой для реализации логики манипуляции данными в проектируемой базе данных.

В базе данных содержатся 5 сущностей: Расположения устройств, устройства, история аварийных случаев устройств, роль пользователя, пользователь.

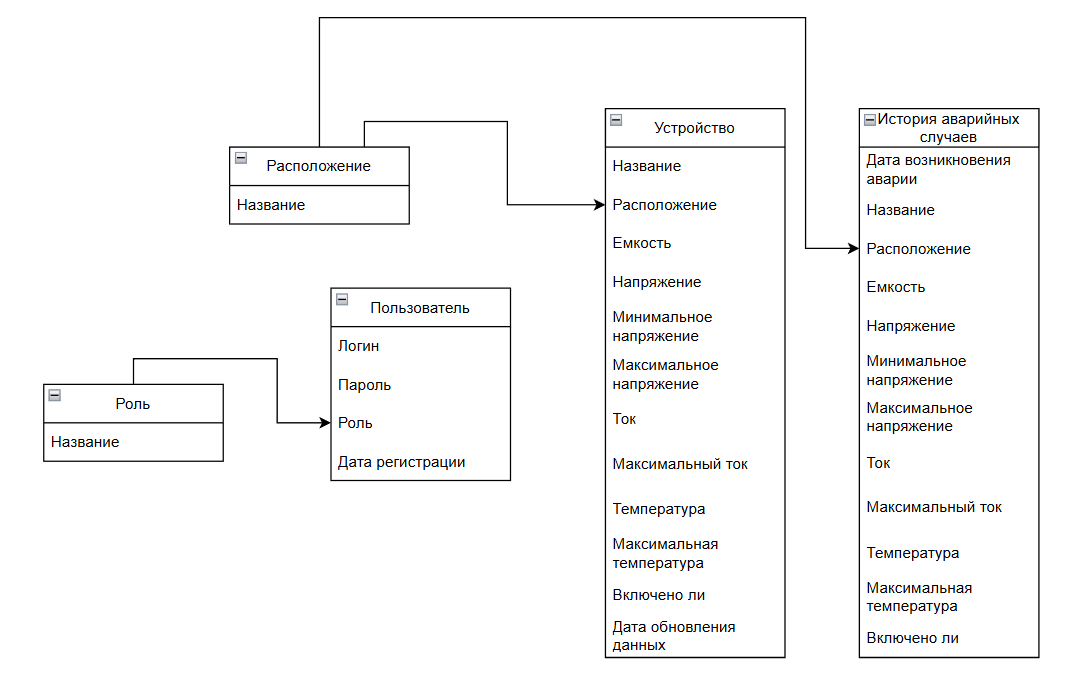


Рисунок – Логическая схема данных

## Физическая схема данных

На Рисунке 18 представлена модель базы данных с фактическими названиями и типами данных в схематическом виде.

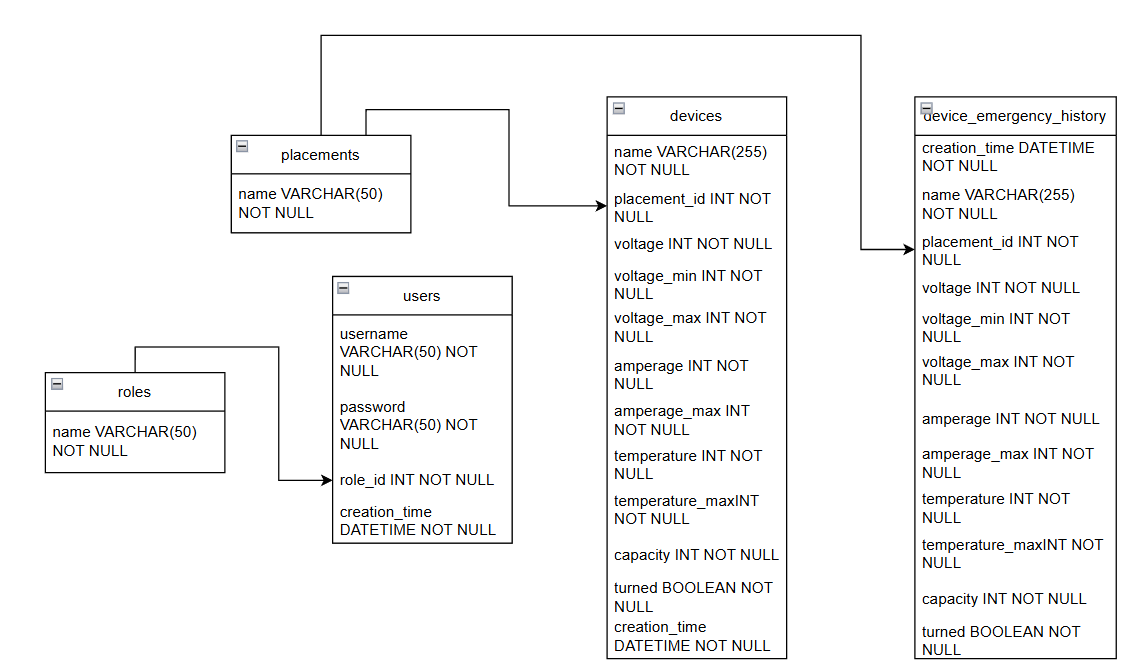


Рисунок – Физическая схема данных

Словарь данных представлен в приложении Г «Скрипт базы данных».

## Структурная схема

На Рисунке 19 представлен план структурной схемы Desktop-приложения серверной части, в которой графически показано взаимодействие всех компонентов и их переходы между друг другом.

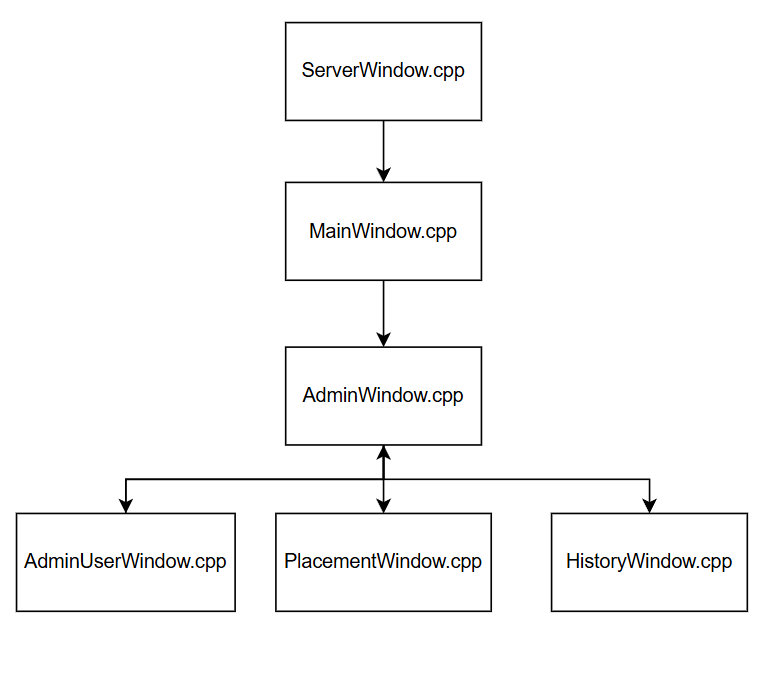


Рисунок – Структурная схема серверной части

На Рисунке 20 представлен план структурной схемы Desktop-приложения клиентской части.

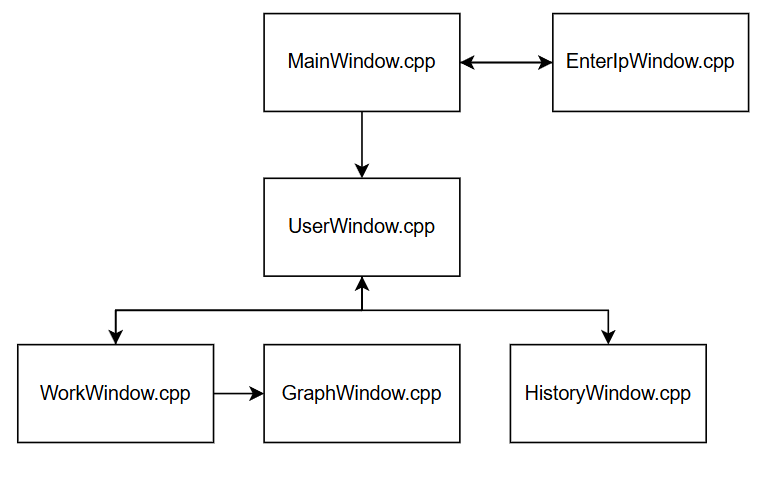


Рисунок – Структурная схема Desktop-клиента

На Рисунке 21 представлен план структурной схемы Android-приложения клиентской части.

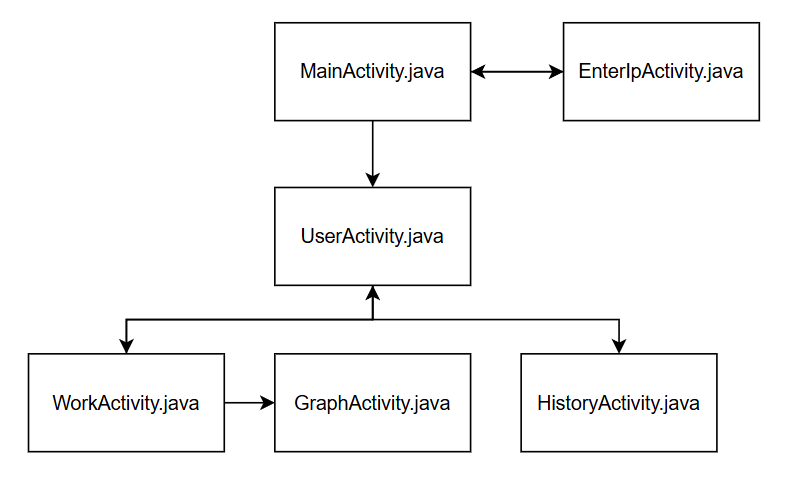


Рисунок – Структурная схема Android-клиента

## Функциональная схема

На Рисунке 22 представлена функциональная схема Desktop-приложения серверной части, на которой графически изображены возможности пользователя в процессе эксплуатации.

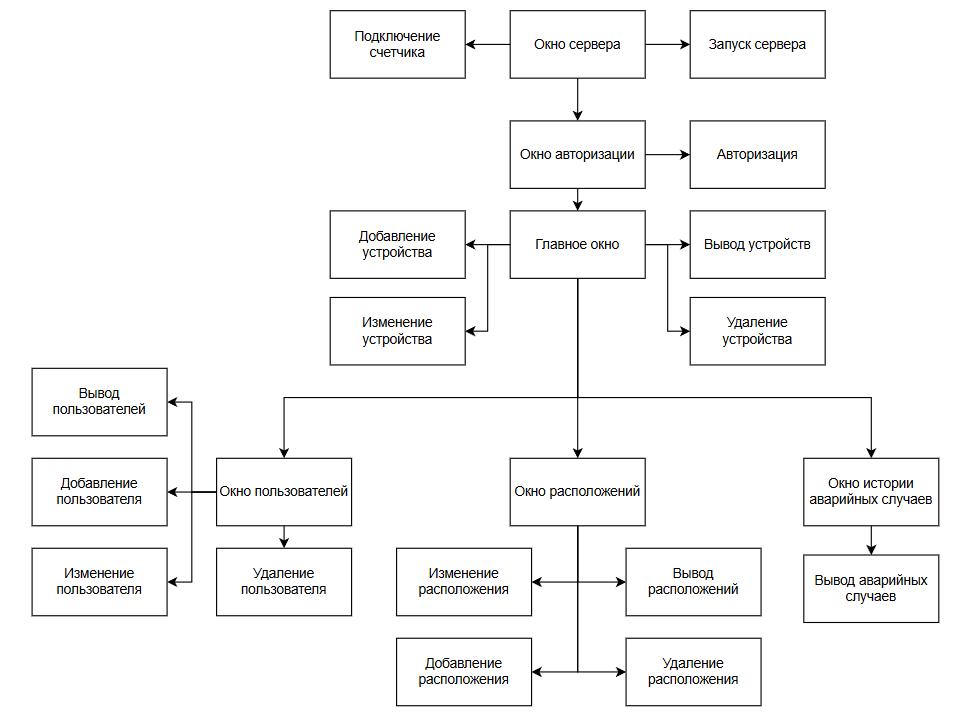


Рисунок – Функциональная схема серверной части

На Рисунке 23 представлена функциональная схема приложений клиентской части, на которой графически изображены возможности пользователя в процессе эксплуатации.

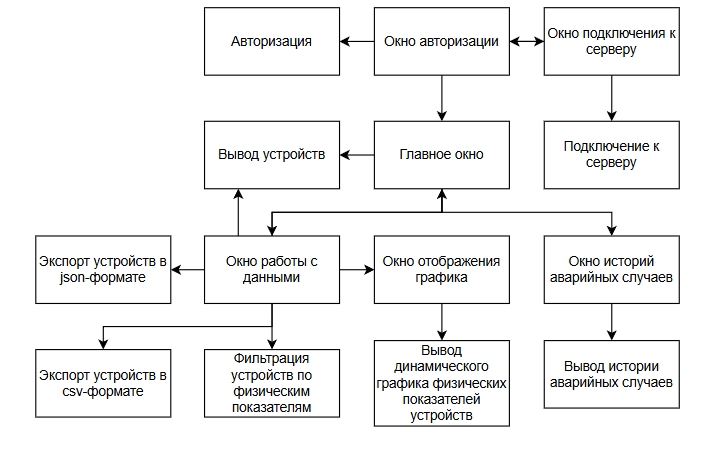


Рисунок – Функциональная схема клиентской части

## Диаграмма классов

На Рисунке 24 представлена диаграмма классов серверной части программного комплекса, на котором графически изображены модули, поля и атрибуты классов.

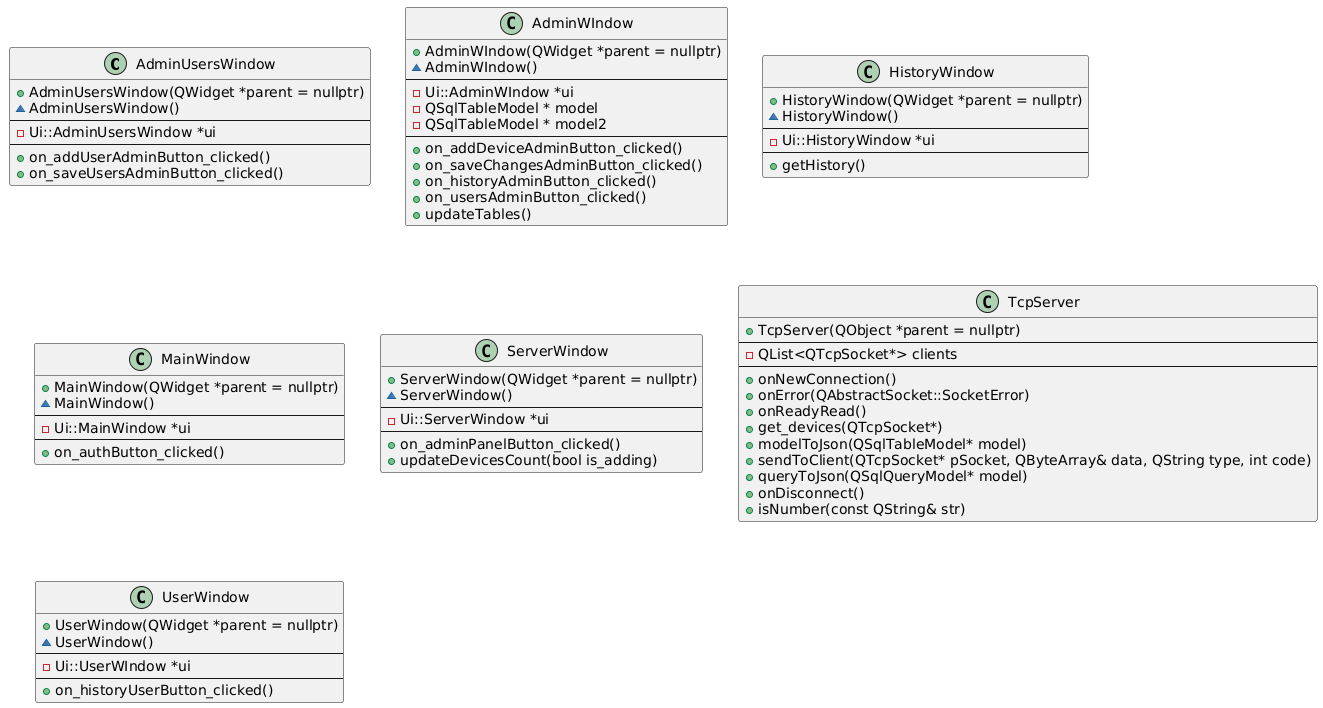


Рисунок – Диаграмма классов серверной части

На Рисунке 25 представлена диаграмма классов клиентской Desktop части программного комплекса.

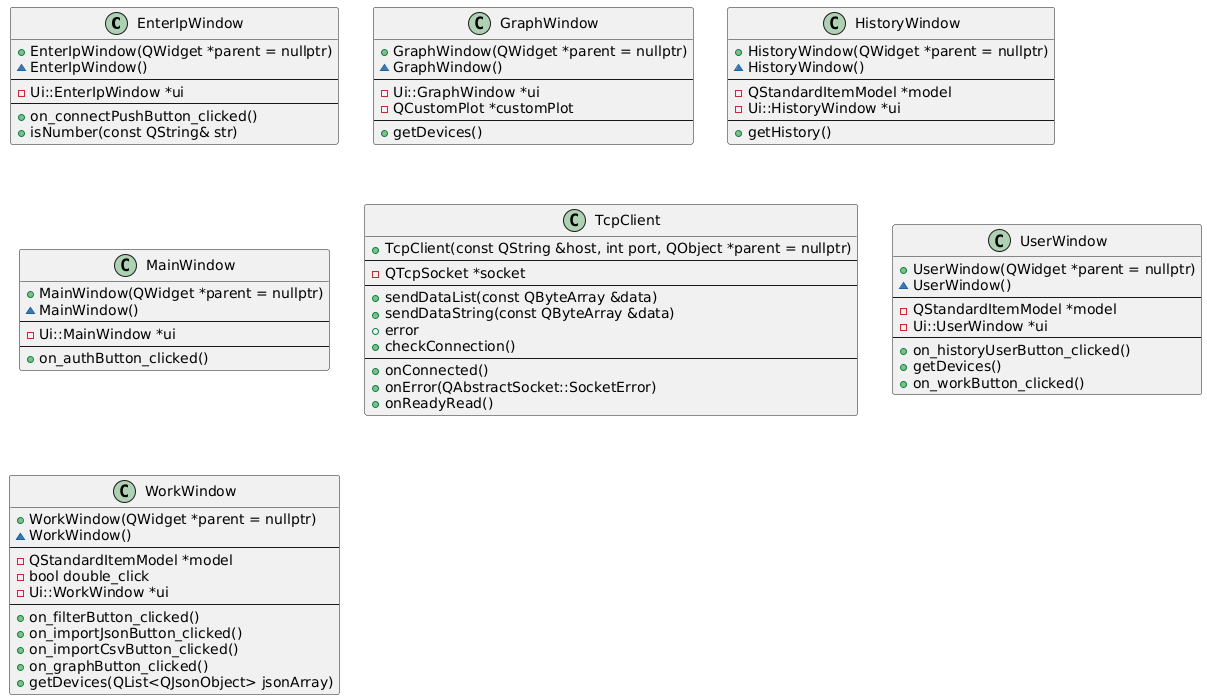


Рисунок - Диаграмма классов клиентской Desktop части

На рисунке 26 изображена диаграмма классов клиентской Android части программного комплекса, которая, для удобства, разделена границами, по которым диаграмма будет разделена на фрагменты на рисунках 27-29. Фрагменты разделены границами в виде соответствующих цветов (Красный, синий, зеленый).

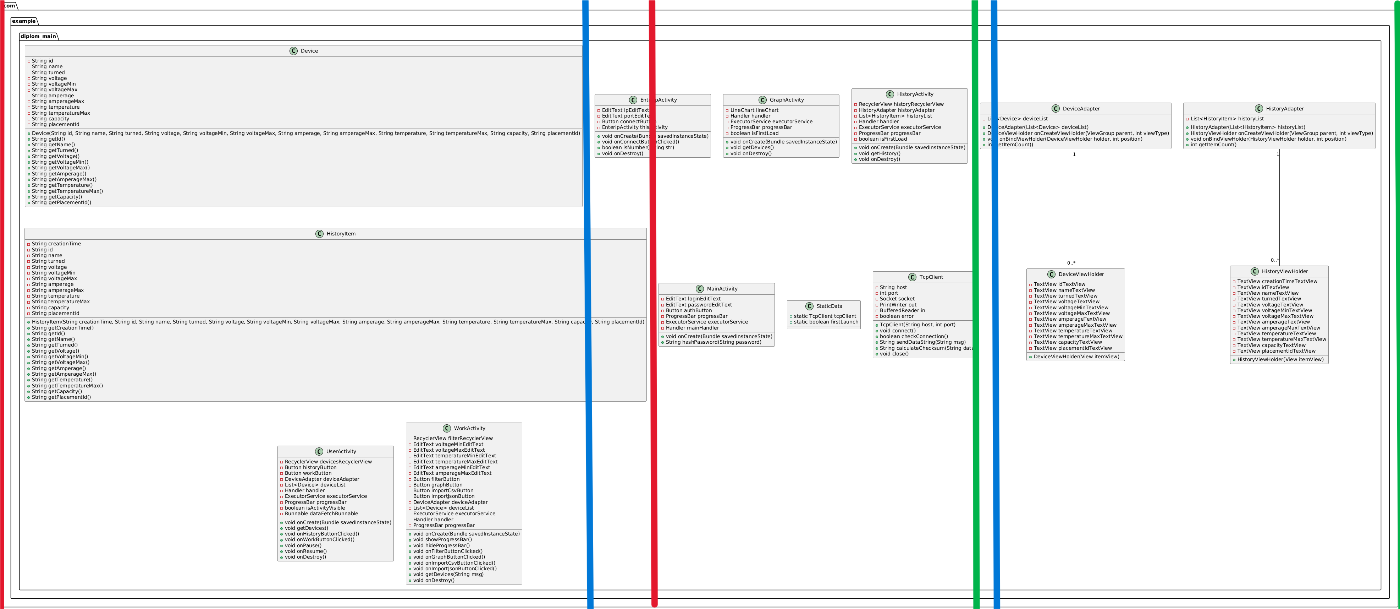


Рисунок – Диаграмма классов клиентской Android части

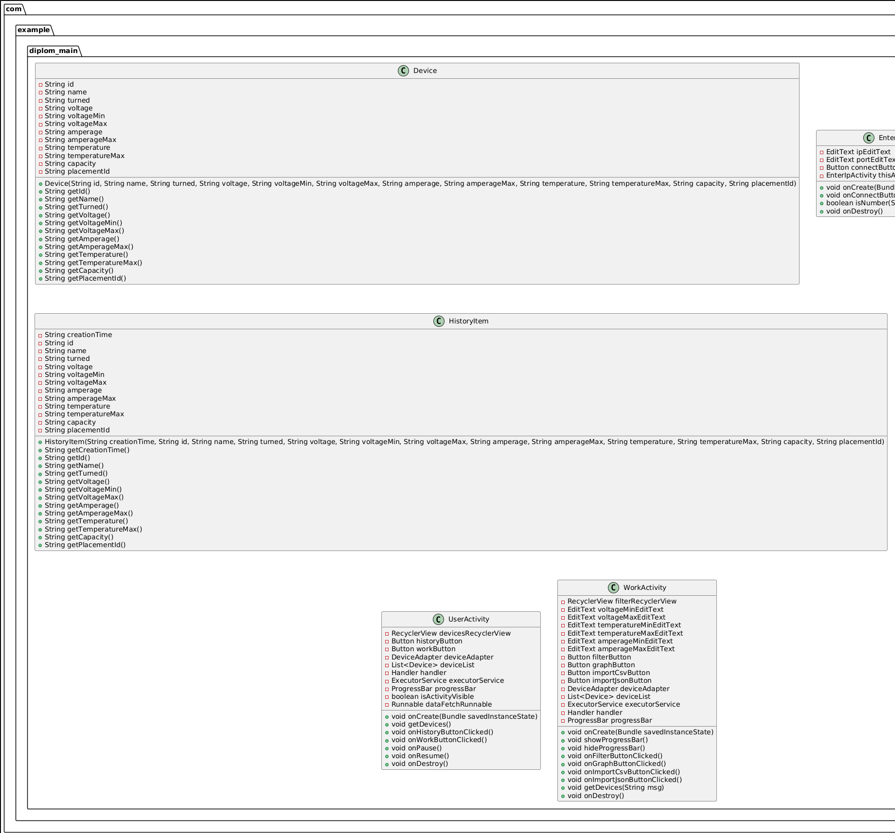


Рисунок – Фрагмент диаграммы классов клиентской Android части №1

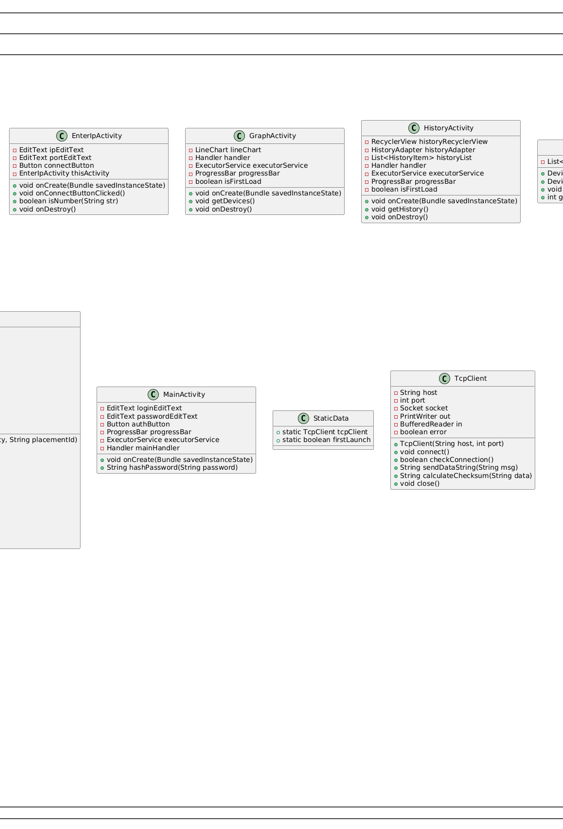


Рисунок - Фрагмент диаграммы классов клиентской Android части №2

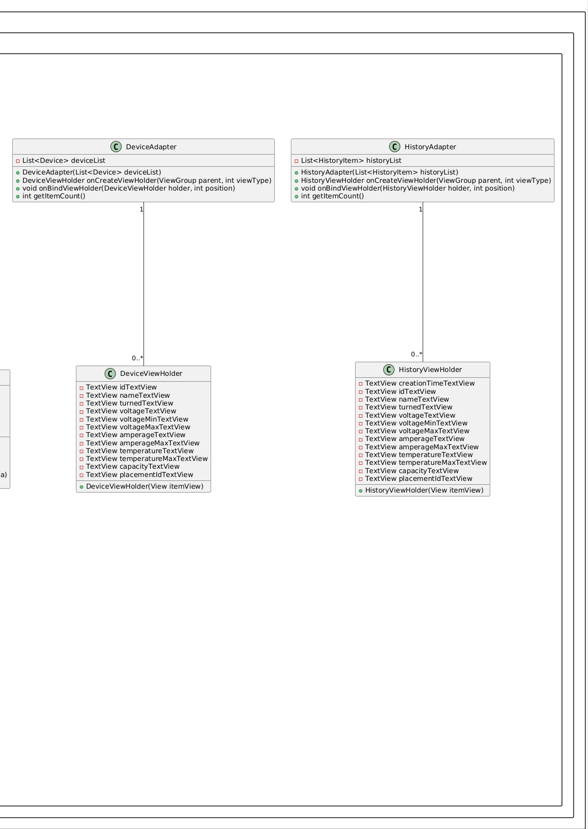


Рисунок - Фрагмент диаграммы классов клиентской Android части №3

## Схема тестирования

На Рисунке 30 представлена схема тестирования серверной части приложения, на котором графически изображены возможности приложения в процессе эксплуатации.

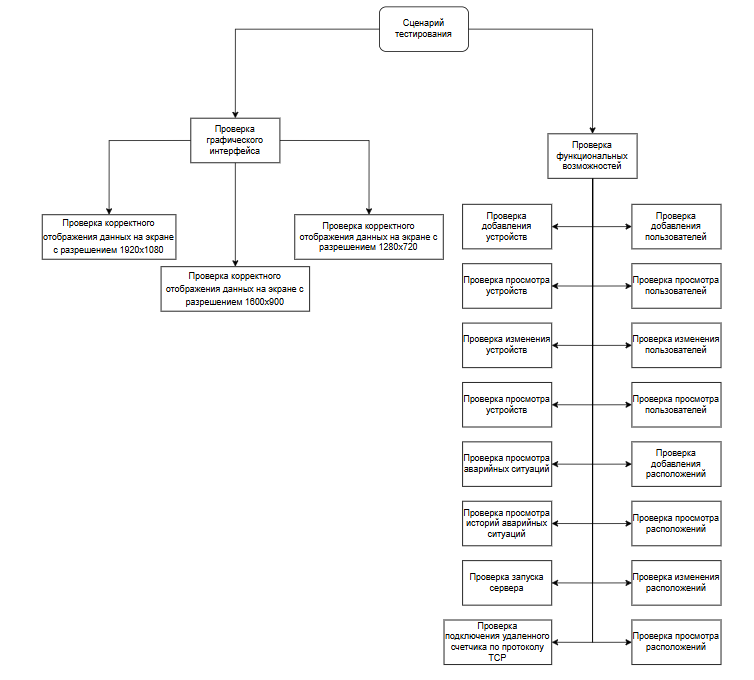


Рисунок – Схема тестирования серверной части

На Рисунке 31 представлена схема тестирования клиентской части приложения.

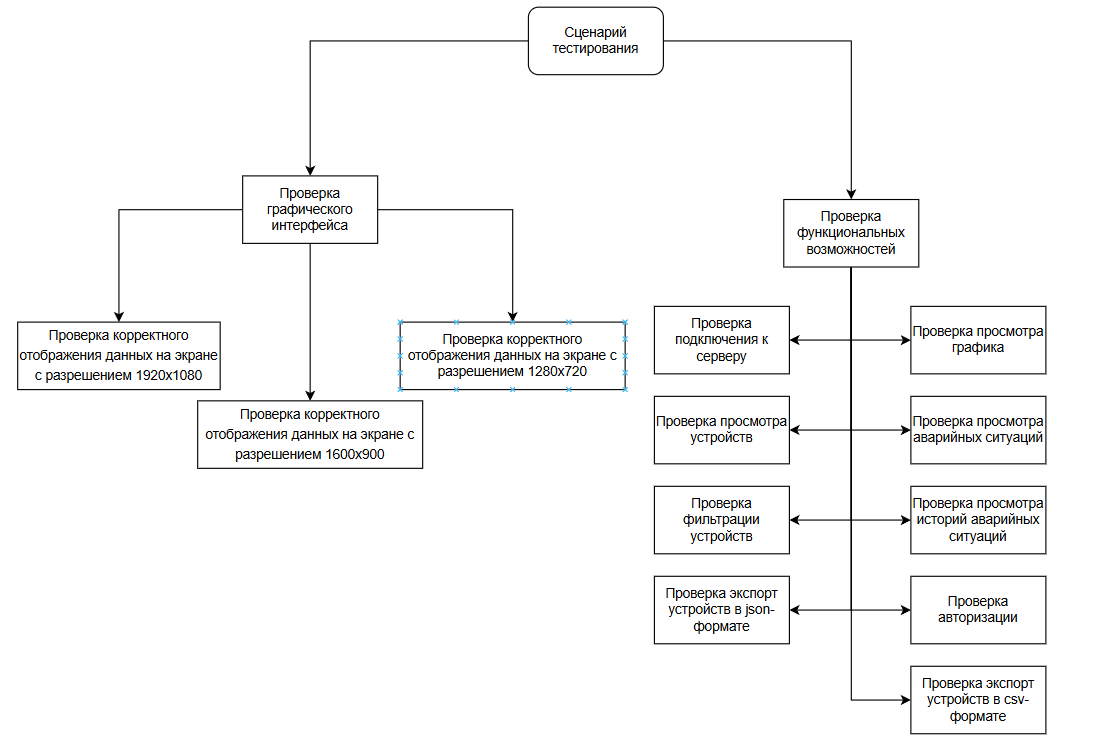


Рисунок – Схема тестирования клиентской части

## Схема пользовательского интерфейса

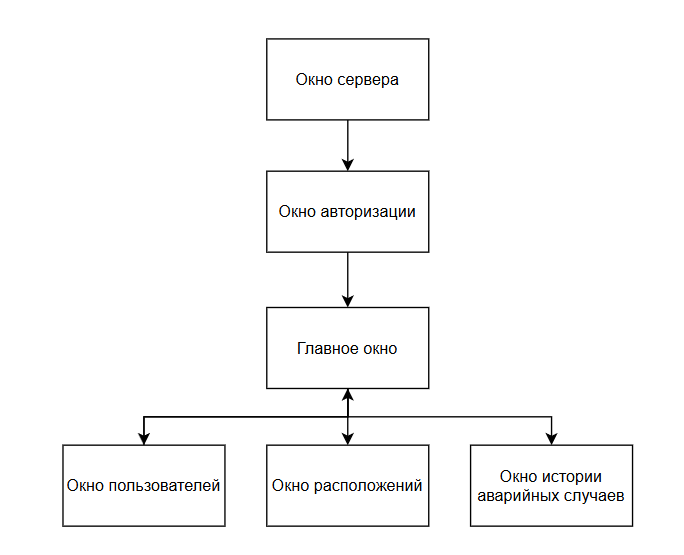


Рисунок – Схема пользовательского интерфейса серверной части

Согласно схеме, на Рисунке 32, у пользователя есть возможность открыть страницу сервера, где есть возможность перейти на страницу авторизации, где необходимо ввести данные для входа в свой аккаунт. Авторизовавшись, откроется админ-панель в случае, если пользователь с введенными авторизационными данными обладает ролью «Администратор», где отображаются устройства, аварийные устройства, есть возможность перейти на страницу пользователей, страницу расположений, страницу историй аварийных случаев, а также есть возможность добавить устройство в БД или изменить его. Страница расположений содержит в себе список расположений устройств в БД и дает возможность его изменять, пополнять или удалять расположения из БД. Страница пользователей хранит тот же функционал, однако выводит не всю информацию, а за исключением пароля, для безопасности данных. Перейдя на страницу истории аварийных случаев, выведутся все аварийные ситуации, когда-либо происходившие в процессе отслеживания.

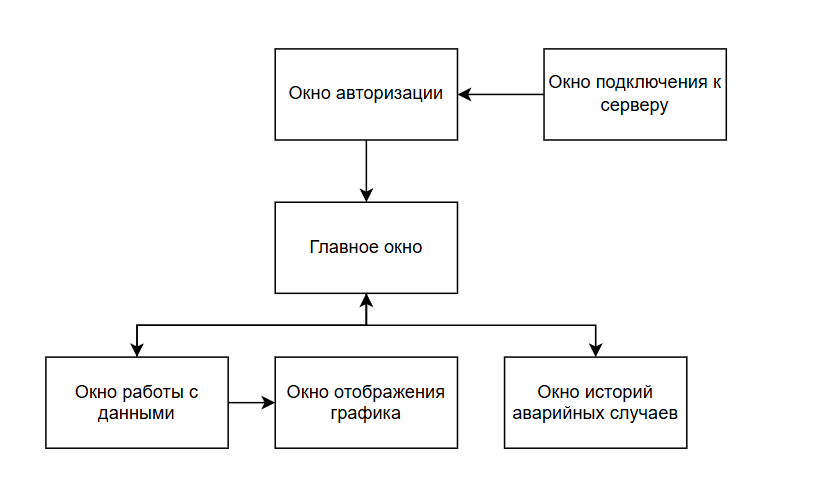


Рисунок – Схема пользовательского интерфейса клиентской части

Согласно схеме, на Рисунке 33, у пользователя есть возможность открыть страницу подключения к серверу, где возможно ввести IP-адрес сервера и его порт, который отображается на начальном окне сервера (по умолчанию 1234). После успешного подключения откроется страница авторизации, где необходимо ввести авторизационные данные для допуска к дальнейшей программе. После авторизации, откроется главное окно, где отображаются устройства (в том числе и аварийные), а также кнопки перехода на окно истории аварийных случаев и перехода на окно работы с данными. Перейдя на окно истории аварийных случаев отобразится список, в котором будут указаны все когда либо случившиеся аварийные случаи с фиксацией значений и датой аварии. Перейдя на окно работы с данными, отобразятся данные устройств, а также поля для их фильтрации по физическим показателям. Кроме того, на окне расположены кнопки для сохранения данных устройств в json и csv формате, а также кнопка перехода на окно графика. На этом окне отображается динамический график по физическим показателям устройств.

## Результат работы программы

В результате поставленной задачи было разработано требуемое программное решение в виде программного комплекса из трех приложений.

На рисунке 34 представлен основной экран Desktop сервера. Окно авторизации, главное окно, окно пользователей, окно расположений, окно истории аварийных случаев, окно подключения к серверу, окно работы с данными, окно отображения графика описаны в ПРИЛОЖЕНИИ Г «Руководство пользователя».

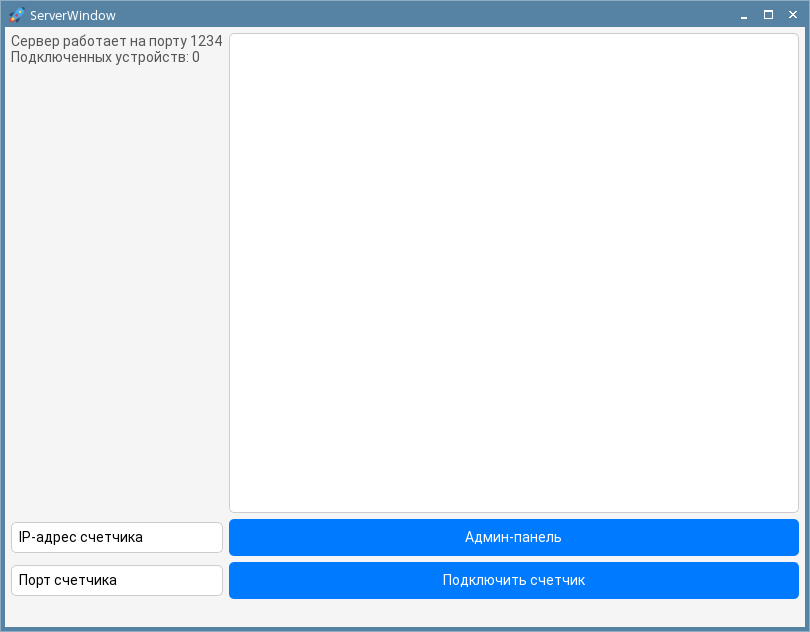


Рисунок – Страница авторизации

# ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

## Инструментальные средства

Для разработки программного комплекса контроля аварийных ситуаций устройств были использованы интегрированные средства разработки Qt Creator и Android Studio. Они предлагает мощные инструменты для написания, отладки и тестирования приложений на многих языках программирования и для разных платформ. Qt Creator и Android Studio поддерживает множество библиотек, что делает разработку более гибкой и продуктивной. Удобный интерфейс и встроенные средства отладки помогают разработчикам эффективно работать с кодом.

В качестве системы управления базами данных была выбрана PostgreSQL — мощная объектно-реляционная база данных, которая обеспечивает высокую производительность и надежное хранение данных. PostgreSQL идеально подходит для средних и крупных приложений, благодаря своей масштабируемости, поддержке сложных запросов и расширенным возможностям управления данными. Это требует настройки отдельного сервера, но позволяет интегрировать базу данных в приложение с учетом высоких требований к производительности и безопасности.

C++ был выбран в качестве основного языка разработки благодаря своей мощной поддержке создания нативных приложений и гибкости в работе с графическими интерфейсами. Он поддерживает объектно-ориентированный подход и множество современных библиотек, таких как Qt, что делает его идеальным для разработки контроля аварийных ситуаций устройств. Java, как второй язык разработки был выбран для разработки мобильной части программного комплекса, поскольку Java является основным языком программирования для Android с момента его запуска, и многие библиотеки и фреймворки, используемые в Android-разработке, написаны на Java, что делает его естественным выбором для создания приложений для мобильных платформ.

## Отладка программы

Отладка — это важный этап разработки программного обеспечения, на котором выявляются и исправляются ошибки в коде. В рамках разработки Desktop-приложения контроля аварийных ситуаций устройств помогает обеспечить корректную работу всех функций приложения, таких как управление рецептами, личный профиль и взаимодействие с базой данных.

Qt Creator и Android Studio: интегрированные среды разработки, которые предоставляют мощные инструменты для отладки, включая точки останова, просмотр переменных и трассировку стека.

Логирование: использование встроенного логирования Qt, Logcat, Toast, а также библиотеки qDebug для записи информации о работе программы, что позволяет отслеживать выполнение кода и выявлять проблемные места.

## Защитное программирование

На Рисунке 35 представлена обработка ошибок при авторизации пользователя в случае неверных данных или ошибки подключения.



Рисунок – Обработка ошибок при авторизации

## Характеристики программы

Характеристики программного комплекса представлены в приложении А «Текст программы» в таблице 1 «Модули».

# ЗАКЛЮЧЕНИЕ

В рамках дипломной работы был разработан программный комплекс контроля аварийных ситуаций устройств, написанный на Qt C++ и Android Studio Java. В процессе разработки была реализована вся функциональность, предусмотренная техническим заданием. Выполнены ключевые этапы проектирования, включая разработку архитектуры приложения, создание моделей данных, диаграмм взаимодействия компонентов и описание входных и выходных данных.

На подготовительном этапе проведена настройка среды разработки Qt Creator и Android Studio, установка сервера базы данных PostgreSQL. Эти действия обеспечили надежное взаимодействие интерфейса приложения с базой данных, а также организовали хранение данных об аварийных ситуациях и профилях устройств.

В ходе разработки системы были реализованы функции мониторинга состояния устройств, отображения аварийных ситуаций, создания сервера и подключения к нему, управления учетными записями пользователей. Приложение создано с использованием языков программирования C++ и Java, а также фреймворков Qt и Android, что позволило разработать производительное, функциональное и удобное для пользователя приложение.

На этапе тестирования системы были проверены все реализованные функции, выявлены и устранены недочеты. Благодаря использованию методов защитного программирования система продемонстрировала стабильную работу и отсутствие критических ошибок.

Итоговый результат полностью соответствует поставленным задачам и создает возможности для дальнейшего развития системы.

Несмотря на достигнутые результаты, система обладает потенциалом для совершенствования. Среди направлений для дальнейшего улучшения можно выделить:

* Добавление системы автоматического уведомления о критических аварийных ситуациях.
* Реализацию функции удаленного управления устройствами.
* Адаптацию интерфейса под различные операционные системы.
* Внедрение автоматического формирования отчетов

Разработанный программный комплекс уже является эффективным инструментом для автоматизации процессов контроля аварийных ситуаций устройств и закладывает основу для дальнейшего расширения функциональности.

# СПИСОК ИСПОЛЬЗУЕМЫХ МАТЕРИАЛОВ

1. Документация QT: сайт - <https://www.doc.qt.io/>. Дата обращения:11.02.2025.

2. Документация QtSql: сайт - <https://doc.qt.io/qt-5/qtsql-index.html>. Дата обращения:15.02.25.

3. Начало работы с PostgreSQL: сайт - <https://www.postgresql.org/docs/current/tutorial-start.html>. Дата обращения: 16.02.2025.

4. Техническая документация устройства ЩП120: сайт - <https://www.elpribor.ru/upload/iblock/ba0/RE_342_SHCHP_.pdf>. Дата обращения: 20.02.2025.

5. Документация Android Studio: сайт - <https://developer.android.com/develop?hl=ru>. Дата обращения: 21.03.2025.

6. Документация Java: сайт - <https://docs.oracle.com/en/java/>. Дата обращения: 21.03.2025.

7. Документация TCP: сайт - <https://datatracker.ietf.org/doc/html/rfc793>. Дата обращения: 21.03.2025.