Министерство науки и высшего образования РФ

Ульяновский государственный университет

**Факультет Математики, информационных и авиационных технологий**

**Кафедра Телекоммуникационные технологии и сети**

**КУРСОВАЯ РАБОТА**по дисциплине

«Методы и средства проектирования информационных систем»

**Разработка прототипа ИС «Учёт транспортных средств в ГИБДД»**

-------------------------------------------------------------------------------------

(название темы)

Направление бакалавриата Информационные системы и технологии.   
09.03.02

--------------------------------------------------------------------------------------

(наименование и номер специальности)

Работу выполнил студент ИС-З-21 \_\_\_\_\_ Гусев В. Е.

группа подпись, дата Ф.И.О.

Научный руководитель \_ Доцент Чекал Е.Г.

должность подпись, дата Ф.И.О.

Оценка

У Л Ь Я Н О В С К

2025

Оглавление

[Оглавление 2](#_Toc197805688)

[Введение 4](#_Toc197805689)

[ГЛАВА 1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ 6](#_Toc197805690)

[1.1. Общие сведения 6](#_Toc197805691)

[1.2. Назначение разработки 6](#_Toc197805692)

[1.3. Характеристика объектов автоматизации 7](#_Toc197805693)

[1.4. Требования к системе в целом 8](#_Toc197805694)

[1.4.1. Перечень подсистем 8](#_Toc197805695)

[1.4.2. Уровни иерархии 8](#_Toc197805696)

[1.4.3. Степень централизации 8](#_Toc197805697)

[1.4.4. Режим функционирования 9](#_Toc197805698)

[1.4.5. Взаимодействие со смежными системами 9](#_Toc197805699)

[1.4.6. Перспективы развития 9](#_Toc197805700)

[1.4.7. Требования к надёжности 9](#_Toc197805701)

[1.4.8. Требования к персоналу 10](#_Toc197805702)

[1.4.9. Порядок подготовки пользователя 10](#_Toc197805703)

[1.4.10. Показатели назначения 10](#_Toc197805704)

[1.5. Требования к функциям 11](#_Toc197805705)

[1.5.1. Перечень автоматизируемых задач 11](#_Toc197805706)

[1.5.2. Временной регламент реализации функций 11](#_Toc197805707)

[1.5.3. Требования к качеству 11](#_Toc197805708)

[1.5.4. Требования к видам обеспечения 12](#_Toc197805709)

[1.6. Состав и содержание работ по созданию системы 12](#_Toc197805710)

[1.7. Порядок контроля и приемки 13](#_Toc197805711)

[1.7.1. Виды испытаний 13](#_Toc197805712)

[1.7.2. Этапы приемки 13](#_Toc197805713)

[1.8. Требования к составу 14](#_Toc197805714)

[1.9. Требования к документированию 14](#_Toc197805715)

[1.10. Источники разработки 14](#_Toc197805716)

[ГЛАВА 2. РУКОВОДСТВО ПРОГРАММИСТА 16](#_Toc197805717)

[2.1. Назначение и условия применения программы 16](#_Toc197805718)

[2.1.1. Назначение программы 16](#_Toc197805719)

[2.1.2. Функции, выполняемые программой 16](#_Toc197805720)

[2.1.3. Условия, необходимые для выполнения программы 17](#_Toc197805721)

[2.2. Характеристика программы 17](#_Toc197805722)

[2.2.1. Основные характеристики 17](#_Toc197805723)

[2.2.2. Особенности программы 18](#_Toc197805724)

[2.3. Обращения к программе 19](#_Toc197805725)

[2.3.1. Загрузка и запуск 19](#_Toc197805726)

[2.3.2. Работа с модулями 20](#_Toc197805727)

[2.3.3. Завершение работы 21](#_Toc197805728)

[2.4. Входные и выходные данные 21](#_Toc197805729)

[2.4.1. Входные данные 21](#_Toc197805730)

[2.4.2. Выходные данные 22](#_Toc197805731)

[2.5. Сообщения об ошибках 22](#_Toc197805732)

[Заключение 24](#_Toc197805733)

[Список информативных источников 26](#_Toc197805734)

[Приложения 27](#_Toc197805735)

# ****Введение****

**Информационная система «Учёт транспортных средств в ГИБДД»** предназначена для автоматизации процессов регистрации, контроля и управления данными о транспортных средствах, их владельцах, технических осмотрах и сотрудниках ГИБДД. Система предоставляет функционал для ввода, редактирования, удаления и просмотра данных через графический интерфейс, а также обеспечивает разграничение прав доступа пользователей. Область применения системы охватывает подразделения ГИБДД, ответственные за учёт автотранспорта, проведение техосмотров и работу с базами данных.

Курсовая работа выполняется в соответствии с учебным планом по направлению **09.03.02 «Информационные системы и технологии»** и направлена на закрепление навыков проектирования клиент-серверных приложений, работы с базами данных и разработки программной документации.

**Целью работы** является создание прототипа информационной системы, обеспечивающей:

* Удобное взаимодействие пользователя с данными через графический интерфейс.
* Безопасное хранение и обработку информации с использованием REST API и СУБД.
* Гибкое управление правами доступа для сотрудников ГИБДД.

**Для достижения цели необходимо решить следующие задачи:**

1. Разработать **техническое задание**на программное средство в соответствии с ГОСТ 19.201-78.
2. Реализовать **клиентское приложение** на Qt (C++) с функциями:

* Авторизации (JWT-токены).
* CRUD-операций для ТС, владельцев, техосмотров и пользователей.

1. Создать **серверную часть** (PHP + MariaDB) для обработки API-запросов.
2. Оформить **руководство программиста** по ГОСТ 19.505-79.

**Структура пояснительной записки:**

* **Введение** – обоснование актуальности, цели и задач.
* **Глава 1. Техническое задание** – требования к системе, описание модулей, условия эксплуатации.
* **Глава 2. Руководство программиста** – архитектура, инструкции по сборке, примеры кода.
* **Заключение** – итоги выполнения работы, оценка достижения цели.
* **Приложения** – исходные коды, схемы БД, скриншоты интерфейса.

**Актуальность разработки** обусловлена необходимостью замены устаревших бумажных систем учёта на современные цифровые решения, сокращающие время обработки данных и снижающие риск ошибок.

# ****ГЛАВА 1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ****

## ****1.1. Общие сведения****

* **Наименование темы разработки**: Прототип информационной системы "Учёт транспортных средств в ГИБДД".
* **Условное обозначение**:Шифр не используется.
* **Организация-разработчик**: «УлГУ, ФМИАТ, кафедра ТТС, направление ИСТ 09.03.02».
* **Основание для разработки**:
  + Учебный план ИСТ 09.03.02.
  + Программа дисциплины "Методы и средства проектирования систем и технологий".
* **Плановые сроки**: 15.06.2024 – 30.06.2024.
* **Источник финансирования**: Не определен.
* **Порядок оформления**:
  + Оформляется как главы 1–2 курсовой работы.
  + Соответствует ГОСТ 19.201-78 (ТЗ) и ГОСТ 19.505-79 (руководство программиста).
  + Демонстрация прототипа проводится в аудитории 213/3.
  + Сдача на проверку преподавателю.

## ****1.2. Назначение разработки****

**Функциональное назначение**:

Автоматизация учёта:

* Транспортных средств (марка, модель, VIN, госномер, теххарактеристики, собственник).
* Владельцев ТС (персональные данные, паспортные данные).
* Технических осмотров (Траспортное средство, дата, пробег, статус).
* Пользователей системы (сотрудников ГИБДД с разграничением прав).

**Эксплуатационное назначение**:используется сотрудниками ГИБДД для:

* Регистрации и редактирования данных.
* Контроля техосмотров.
* Формирования отчётности.
* Изменения прав собственности.

**Ключевые функции**:

* CRUD-операции через графический интерфейс (Qt).
* Авторизация по JWT-токенам.
* REST API для интеграции с внешними системами.

**Технические показатели**:

* Время отклика API: ≤ 1 сек.
* Поддержка до 1000 одновременных пользователей.

## ****1.3. Характеристика объектов автоматизации****

**Объекты автоматизации**:

* База данных ТС, владельцев, техосмотров.
* Процессы:
  + Регистрация/снятие с учёта ТС.
  + Внесение данных о собственниках ТС.
  + Проведение техосмотров.
  + Управление доступом сотрудников.

**Требования к условиям эксплуатации**:

Системы должны соответствовать условиям эксплуатации группы 2 ГОСТ 21552-84 «Средства вычислительной техники. Общие технические требования, приемка, методы испытаний, маркировка, упаковка, транспортировка, хранение».

Характеристики окружающей среды в местах установки технических средств соответствуют требованиям следующих документов:

• ГОСТ Р ИСО 14001-98. Системы управления окружающей средой. Требования и руководство по применению;

• СанПиН 2.2.24.548-96. Физические факторы производственной среды. Гигиенические требования к микроклимату производственных помещений;

• СанПиН 2.2.2/2.4.1340-03. Гигиенические требования к персональным электронно-вычислительным машинам и организации работы.

Условия эксплуатации персональных компьютеров Системы соответствуют Гигиеническим требованиям к видео дисплейным терминалам, персональным электронно-вычислительным машинам и организации работы (Санитарные правила и нормы. СанПиН 2.2.2.542-96).

* **Клиент**:
  + ОС: Linux или Windows 7/8.0/8.1/10.
  + Не менее 2 ГБ ОЗУ.
* **Сервер**:
  + PHP 7.4+, MariaDB 10.1+, 2 ГБ ОЗУ.
  + Соответствие ГОСТ Р 56939-2016 (защита персональных данных).
* **Окружающая среда**:
  + Температура: 18–25°C.
  + Влажность: 40–60%.

## ****1.4. Требования к системе в целом****

### ****1.4.1. Перечень подсистем****

В состав демострационной модели ИС «Учёт транспортных средств в ГИБДД» должны входить следующие подсистемы:

1. **Подсистема аутентификации.**
2. **Подсистема учёта ТС**.
3. **Подсистема техосмотров**.
4. **Подсистема управления пользователями**.

### ****1.4.2. Уровни иерархии****

Иерархия ИС «Учёт транспортных средств в ГИБДД» представлена в приложении №1.

### ****1.4.3. Степень централизации****

* Централизованное хранение данных в СУБД MariaDB.
* Клиенты, подсистемы и модули запрашивают и записывают информацию путем обращения к единому API – https://gvev.ru/ulsu/api/engine.

### ****1.4.4. Режим функционирования****

* ИС «Учёт транспортных средств в ГИБДД» имеет круглосуточный доступ (кроме времени технического обслуживания).

### ****1.4.5. Взаимодействие со смежными системами****

* Возможна интеграция с:
  + Единой системой идентификации (ЕСИА).
  + Мобильными системами и приложениями.

### ****1.4.6. Перспективы развития****

* Добавление модуля штрафов.
* Подключение к мобильному приложению для инспекторов.

### ****1.4.7. Требования к надёжности****

Осуществление контроля данных происходит после авторизации пользователя.

* **Допустимые отказы**: Не чаще 1 раза в 3 месяца.
* **Восстановление**: Перезапуск сервиса (≤ 5 мин).
* **Резервирование**:
  + Ежедневное копирование БД.
  + Хранение копий на отдельном NAS-устройстве.

Программное средство должно храниться на жестком диске ПК и переносном устройстве в нескольких экземплярах, в целях предотвращения потерь или утраты программы.

Для уменьшения рисков выхода из строя программного обеспечения и технических средств необходимо поддержание комфортных условий климатических условий, где температура помещения должна быть в диапазоне от 18-25 °C, относительная влажность 40-60%, а также необходимо избегать магнитных воздействий на устройства хранения информации. С данным программным средством может пользоваться человек с квалификацией «Пользователь ПК», имеющий базовые знания пользования персональным компьютером.

### ****1.4.8. Требования к персоналу****

Для эксплуатации демонстрационной модели ИС «Учёт транспортных средств в ГИБДД» определены следующие роли:

| **Роль** | **Обязанности** | **Квалификация** |
| --- | --- | --- |
| **Администратор** | * Ввод/редактирование данных в ИС (Ведение учетных записей пользователей, смена паролей, контроль прав доступа пользователей). * Установка, настройка, мониторинг работоспособности системного, базового, прикладного программного обеспечение и ИС. | Навыки работы с ПК выше среднего |
| **Оператор** | * Авторизация в системе. * Ввод/редактирование данных. | Базовые навыки работы с ПК |

Системный администратор должен обладать высоким уровнем квалификации и практическим опытом выполнения работ по установке, настройке и администрированию программных и технических средств, применяемых в системе

### ****1.4.9. Порядок подготовки пользователя****

Для подготовки персонала к работе с ИС необходимо осуществить следующие мероприятия:

1. Подготовить документацию по пользованию системой.
2. Подготовить электронную справку (электронная версия документации)
3. Провести курсы обучения по пользованию системой.
4. Практический тренинг (2 академических часа).
5. Тестирование знаний.

### ****1.4.10. Показатели назначения****

Степень приспособленности не определяется. Существенных изменений в системе не требуется.

## ****1.5. Требования к функциям****

### ****1.5.1. Перечень автоматизируемых задач****

Согласно требованиям, в системе должны быть автоматизированы следующие задачи:

1. **Управление транспортными средствами**:
   * Регистрация новых ТС (ввод марки, модели, VIN, госномера, теххарактеристик).
   * Редактирование и удаление записей.
2. **Учет владельцев ТС**:
   * Ввод и обновление паспортных данных.
   * Связь "ТС-владелец".
3. **Технические осмотры**:
   * Фиксация даты, пробега, статуса осмотра.
4. **Управление пользователями**:
   * Авторизация по логину/паролю с JWT.
   * Разграничение прав (16 уровней доступа).
5. **Интеграционные задачи**:
   * Обмен данными с внешними системами через REST API.

### ****1.5.2. Временной регламент реализации функций****

* **Операции с ТС**: в реальном времени при обращении пользователя.
* **Синхронизация данных**: ежедневное резервное копирование во временном диапазоне с 00:00 по 04:00.
* **Проверка доступа**: при каждом входе в систему.
* **Проверка привелегий: при каждом запросе.**

### ****1.5.3. Требования к качеству****

Обеспечение полноты, сохранности, истинности и актуальности информации.

* **Полнота**: 100% охват обязательных полей (марка, госномер и т.д.)
* **Актуальность**: мгновенное обновление данных при изменениях
* **Безопасность**:
  + Шифрование паролей.
  + SSL сертификат.
  + Валидация входных данных.

### ****1.5.4. Требования к видам обеспечения****

В состав базы данных входят сущности, представленные в приложении №2.

Временными характеристиками являются осуществление запросов длительностью менее одной секунды.

| **Вид обеспечения** | **Состав** |
| --- | --- |
| **Математическое** | Не требуется |
| **Программное** | Сервер: Apache/Nginx, PHP 7.4+, MariaDB 10.1+ |
| **Техническое** | Сервер: ≥2 ГБ ОЗУ.  Клиент: Linux или Windows 7/8.0/8.1/10, ≥2 ГБ ОЗУ |
| **Лингвистическое** | Интерфейс на русском языке. Реализация ИС «Учёт транспортных средств в ГИБДД» на языке программирования С++. |
| **Организационное** | Руководства для администратора и оператора |
| **Метрологическое** | Не требуется |
| **Методологическое** | Техническое задание, руководство программиста. |

## ****1.6. Состав и содержание работ по созданию системы****

Разработка системы предполагает создание следующих проектов:

1. **Эскизный проект**:
   * Проектирование структуры БД.
2. **Технический проект**:
   * Реализация API.
   * Создание физической БД.
   * Создание клиентского приложения.
3. **Рабочий проект**:
   * Тестирование.
   * Написание документации.

**Исполнитель**: Гусев Владислав Евгеньевич.

**Сроки**: 15.06.2024 – 30.06.2024.

Оформление отчетов по лабораторной и курсовой работе.

Защита курсовой работы.

## ****1.7. Порядок контроля и приемки****

### ****1.7.1. Виды испытаний****

* **Функциональное тестирование**: Более 50 тест-кейсов (проверка CRUD-операций)
* **Нагрузочное тестирование**: до 1000 одновременных пользователей
* **Проверка безопасности**: тесты на SQL-инъекции

Так же, к виду испытаний относится предварительный показ работающего комплекта ИС преподавателю, защита курсовой работы.

### ****1.7.2. Этапы приемки****

Приемка работы осуществляется по этапам:

* **Эскизный проект**:

Согласование с преподавателем:

* + Логической структуры базы данных.
  + Выбора среды разработки.
  + Выбора языка программирования.
  + Выбора ОС.
* **Технический проект**:
  + Демонстрация работающего API.
* **Рабочий проект**:
  + Демонстрация готовой информационной системы.

**Вид приемки:** Защита курсовой и лабораторных работ.

**Приемочная комиссия**: преподаватели кафедры ТТС.

Контроль за выполнение программы «Разработка прототипа ИС «Учёт транспортных средств в ГИБДД»» осуществляет преподаватель Чекал Е.Г.

## ****1.8. Требования к составу****

К содержанию работ по подготовке объектов автоматизации к вводу систем:

1. Установить ОС Linux.
2. Установить и настроить ПО на сервере:
   * Веб-сервер (Apache/Nginx).
   * PHP 7.4+.
   * MariaDB 10.1+.
3. Установить ПК ИС «Учёт транспортных средств в ГИБДД».
4. Разработать:
   * Руководство администратора.
   * Руководство пользователя

## ****1.9. Требования к документированию****

Перечень разрабатываемых документов: отчет по лабораторным работам, пояснительная записка курсовой работы.

При разработке программы согласно ЕСПД ГОСТ, следующие документы в единственном экземпляре:

1. **Техническое задание** (согласно ГОСТ 19.201-78, который содержит Введение, в котором обозначены название программы c характеристикой и основная цель разработки, разбитая на задачи; основание для разработки; требования к программе и документации).
2. **Руководство программиста** (согласно ГОСТ 19.505-79, в котором отображена суть программного средства, условия выполнения программы, выполнение программы, в котором описывается инструкция для пользователя и сообщения оператору).

## ****1.10. Источники разработки****

Источниками разработки являются следующие документы:

1. ГОСТ 34.602-89 "Техническое задание на создание АС".
2. ГОСТ 19.505-79 "Руководство программиста".
3. Учебные материалы кафедры ТТС.
4. Научно-техническая литература.

# ****ГЛАВА 2. РУКОВОДСТВО ПРОГРАММИСТА****

## ****2.1. Назначение и условия применения программы****

### ****2.1.1. Назначение программы****

Программа «Прототип ИС учёта транспортных средств в ГИБДД» предназначена для автоматизации процессов учета:

* Транспортных средств (ТС) с их характеристиками (марка, модель, госномер, VIN и т. д.).
* Владельцев ТС (ФИО, паспортные данные).
* Технических осмотров (дата, пробег, статус).
* Пользователей системы (логины, пароли, права доступа).

Цель разработки — централизованное хранение данных, устранение дублирования информации и минимизация ручного ввода.

### ****2.1.2. Функции, выполняемые программой****

Программа реализует следующие функции:

1. **Управление ТС:**
   * Добавление, редактирование, удаление записей.
   * Изменение собственника ТС.
2. **Управление владельцами:**
   * Добавление, редактирование, удаление записей.
   * Внесение и редактирование паспортных и контактных данных.
3. **Техосмотры:**
   * Добавление, редактирование, удаление записей.
   * Фиксация результатов технических осмотров транспортных средств.
4. **Пользователи и права доступа:**
   * Добавление, редактирование, удаление записей.
   * Создание учетных записей.
   * Смена и шифрование паролей.
   * Настройка прав (чтение, запись, удаление для каждого модуля).
5. **Авторизация:**
   * JWT-аутентификация.

### ****2.1.3. Условия, необходимые для выполнения программы****

#### ****2.1.3.1. Требования к аппаратному обеспечению****

* **Оперативная память:** Не менее 2 ГБ.
* **Процессор:** x86-64, 1.5 ГГц.
* **Дисковое пространство:** Не менее 500 МБ (для клиента), не менее 1 ГБ (для сервера).

#### ****2.1.3.2. Требования к периферийным устройствам****

**Обязательно:** Мышь, клавиатура.

#### ****2.1.3.3. Требования к программному обеспечению****

* **Сервер:**
  + ОС: Linux/Windows.
  + Веб-сервер: Apache/Nginx.
  + PHP: 7.4+.
  + СУБД: MariaDB.
* **Клиент:**
  + ОС: Windows 7/8.0/8.1/10 или Linux.

#### ****2.1.3.4. Требования к персоналу****

* **Системный администратор:**
  + Навыки работы с персональным компьютером выше среднего.
  + Должен иметь минимум среднее техническое образование.
  + Должен поддерживать работоспособность технических и программных средств.
* **Оператор (пользователь):**
  + Базовые навыки работы с персональным компьютером.
  + Должен обладать практическими навыками работы с графическим пользовательским интерфейсом операционной системы.

## ****2.2. Характеристика программы****

### ****2.2.1. Основные характеристики****

* **Режим работы:** Интерактивный (GUI).
* **Контроль данных:**
  + Логирование запросов на API.

Работоспособность программы проверяется следующими способами:

* Модуль авторизации:

После запуска программы, указать логин и пароль и нажать кнопку «Войти». Ожидаемые ответы от программы: сообщение «Неверный логин/пароль», переход на вторую форму ИС.

* Модули «Транспортные средства», «Техосмотры», «Люди», «Сотрудники»:

Во второй форме ИС (после успешной авторизации) нажать на кнопку соответствующего раздела. Ожидаемый ответ программы: Открывается форма с таблицей данных и тремя кнопками «Добавить», «Изменить», «Удалить».

* Кнопка «Добавить»:

При нажатии на кнопку «Добавить» открывается форма для ввода данных. После заполнения данных и нажатия кнопки «Добавить», форма добавления закрывается, в таблицу добавляется новая строка.

* Кнопка «Изменить»:

При выделении необходимой строки и нажатии на кнопку «Изменить» открывается форма с данными. После внесении корректировок и нажатия на кнопку «Изменить», форма изменения закрывается, в таблице обновляются данные.

* Кнопка «Удалить»:

При выделении необходимой строки и нажатии на кнопку «Удалить» открывается окно с подтверждением действия «Вы уверены, что хотите удалить строку №#). После нажатия на кнопку «Да», окно подтсверждения закрывается, в таблице, выбранная ранее, строка удаляется.

### ****2.2.2. Особенности программы****

1. **Графический интерфейс:**
   * Упрощает взаимодействие с компьютером, делая его более интуитивно понятным и доступным. Он позволяет визуально представлять данные и функции, что способствует более быстрому усвоению информации и повышению продуктивности.
2. **Актуальность:**
   * Имеет единую базу данных на СУБД MariaDB.
   * Данные обновляются при любом взаимодействии с ними (получении, удалении, добавлении, изменении).
3. **Модульность:**
   * Отдельные окна для ТС, владельцев, техосмотров, сотрудников.
4. **Безопасность:**
   * Доступ к данным имеют только авторизированные пользователи.
   * Аутентификация через JWT.
   * Разграничение прав доступа пользователей.
   * Защищенные запросы к серверу (SSL).
5. **Работа с API:**
   * Все данные передаются в формате JSON.

JSON – это легкий формат обмена данными, который легко читается людьми и машинами. Он хорошо поддерживается во многих языках программирования, что делает его универсальным для интеграции различных систем.

API в комбинации с JSON являются ключевым аспектом современной разработки программного обеспечения и обмена данными, широкоиспользуемым методом обмена информации между различными системами, программами и приложениями.

## ****2.3. Обращения к программе****

### ****2.3.1. Загрузка и запуск****

1. Клиент: Запустите исполняющий файл GAIDB (GAIDB.exe – на Windows).
2. Введите логин/пароль (для доступа к учетной записи администратора по умолчанию используется: admin|admin).

### ****2.3.2. Работа с модулями****

#### ****2.3.2.1 Модуль «Транспортные средства»****

* **Добавление ТС:**
  + Заполните поля: марка, модель, тип кузова, гос.номер, номер двигтаеля, номер кузова, мощность (л.с.), идентификатор собственника/владельца.
  + Нажмите кнопку «Добавить».
* **Редактирование информации ТС:**
  + Выберите запись в таблице и нажмите «Изменить».
  + Измените необходимые данные и нажмите «Изменить»
* **Удаление ТС:**
  + Выберите запись в таблице и нажмите «Удалить».

#### ****2.3.2.2 Модуль «Техосмотры»****

* **Фиксация осмотра:**
  + Укажите дату, идентификатор автомобиля, пробег, статус автомобиля.
  + Нажмите кнопку «Добавить».
* **Редактирование информации осмотра:**
  + Выберите запись в таблице и нажмите «Изменить».
  + Измените необходимые данные и нажмите «Изменить»
* **Удаление осмотра:**
  + Выберите запись в таблице и нажмите «Удалить».

#### ****2.3.2.3 Модуль «Люди»****

* **Добавление человека в базу данных собственников:**
  + Укажите имя, фамилию, отчество, паспортные данные, дату рождения, прописку.
  + Нажмите кнопку «Добавить».
* **Редактирование информации о человеке:**
  + Выберите запись в таблице и нажмите «Изменить».
  + Измените необходимые данные и нажмите «Изменить»
* **Удаление человека:**
  + Выберите запись в таблице и нажмите «Удалить».

#### ****2.3.2.4 Модуль «Сотрудники»****

* **Добавление сотрудника:**
  + Укажите имя, фамилию, логин и пароль для входа.
  + Проставьте привелегии для сотрудника.
  + Нажмите кнопку «Добавить».
* **Редактирование информации о сотруднике и изменение логина/пароля:**
  + Выберите запись в таблице и нажмите «Изменить».
  + Измените необходимые данные и нажмите «Изменить»
* **Удаление сотрудника:**
  + Выберите запись в таблице и нажмите «Удалить».

### ****2.3.3. Завершение работы****

* Программа предоставляет стандартный способ завершения программы. Закройте окно программы, путем нажатия на «крестик», или нажмите комбинацию клавиш Alt+F4.

## ****2.4. Входные и выходные данные****

### ****2.4.1. Входные данные****

* **Технические средства:** 
  + Марка, модель, тип кузова, госномер (формат А123БВ45), номер кузова, номер двигателя – tinytext (текст не более 255 символов).
  + Идентификатор, идентификатор владелеца, мощность двигателя – integer (целое число)
* **Владельцы:**.
  + Имя, фамилия, отчество, кем выдан паспорт, место жительство – tinytext (текст не более 255 символов).
  + Идентификатор, серия паспорта, номер паспорта – integer (целое число).
  + Дата рождения, дата выдачи паспорта – date (ГГГГ.ММ.ДД)
* **Технические осмотры:** 
  + Статус технического средства – tinytext (текст не более 255 символов).
  + Идентификатор, идентификатор транспортного средства, пробег транспортного средства – integer (целое число).
  + Дата проведение технического осмотра – date (ГГГГ.ММ.ДД).
* **Сотрудники:** 
  + Имя, фамилия, логин для входа в ИС, пароль для входа в ИС – tinytext (текст не более 255 символов).
  + Идентификатор – integer (целое число).
  + Права доступа – Boolean (true|false, 1|0).

### ****2.4.2. Выходные данные****

* **Ответы от API:** JSON.

## ****2.5. Сообщения об ошибках****

| **Код** | **Сообщение** | **Причина** | **Действие** |
| --- | --- | --- | --- |
| 401 | * «Ошибка авторизации». * «Host requires authentication». | * Требуется авторизация. * Истек срок действия ключа. | * Необходимо авторизироваться. * В случае, если вы уже авторизированны в программе и при попытке выполнить запрос к модулям выдает ошибку, то это означает, что срок действия ключа истек (по умолчанию срок действия ключа – 1 час). Требуется: перезапустить программу и повторно пройти авторизацию. |
| 403 | * «Доступ запрещен». * «Forbidden». | Нет прав на операцию | Обратиться к администратору. |
| 404 | * «Страница не найдена». * «Not Found». | Запрос был отправлен на несуществующую страницу. | Обратиться к администратору. |
| 500 | * «Ошибка сервера». * «Internal Server Error». | Сбой на сервере | Повторить запрос позже. |
| - | * «Connection refused», «Соединение отклонено». * «Timeout», «Время ожидания истекло». * «Network unreachable», «Сеть недоступна». | * Сервер перегружен. * У сервера проводятся технические работы. | Повторить запрос позже. |
| - | «Неверный логин/пароль» | Указан несуществующий логин или неверный пароль. | * Вспомнить правильный логин или пароль. * Обратиться к администратору для восстановления доступа. |
| - | «Выберите строку для удаления» | Нажатие кнопки «Удалить» без выделения необходимой строки | Выделить необходимую строку и нажать кнопку «Удалить». |
| - | «Выберите строку для редактирования» | Нажатие кнопки «Изменить» без выделения необходимой строки | Выделить необходимую строку и нажать кнопку «Изменить». |

**Заключение**

В ходе выполнения курсовой работы была разработан прототип информационной системы «учёта транспортных средств в ГИБДД», включающая:

* **Техническое задание:**
* Определены цели, задачи и требования к системе.
* Описаны функциональные возможности (учёт ТС, владельцев, техосмотров, управление пользователями).
* Установлены технические требования к серверу и клиентской части.
* **Руководство программиста:**
* Детально описаны архитектура, алгоритмы работы и взаимодействие компонентов.
* Приведены инструкции по настройке, запуску и использованию системы.
* **Программная реализация:**
* **Серверная часть:** REST API на PHP с аутентификацией через JWT.
* **Клиентская часть:** Qt-приложение с интуитивным и понятным интерфейсом для Winows/Linux.
* **База данных:** СУБД MariaDB с таблицами для хранения данных о ТС, владельцах, техосмотрах и сотрудниках.
* **Итоги работы:**
* Система автоматизирует ключевые процессы ГИБДД: учёт транспортных средств, контроль техосмотров, управление правами доступа сотрудников.
* Снижены риски ошибок за счёт валидации данных и централизованного хранения.
* Обеспечена безопасность (JWT-токены, разграничение прав, SSL-сертификат).
* **Перспективы развития:**
* Добавление модуля штрафов и нарушений.
* Интеграция с государственными системами.
* Разработка мобильного приложения для инспекторов.

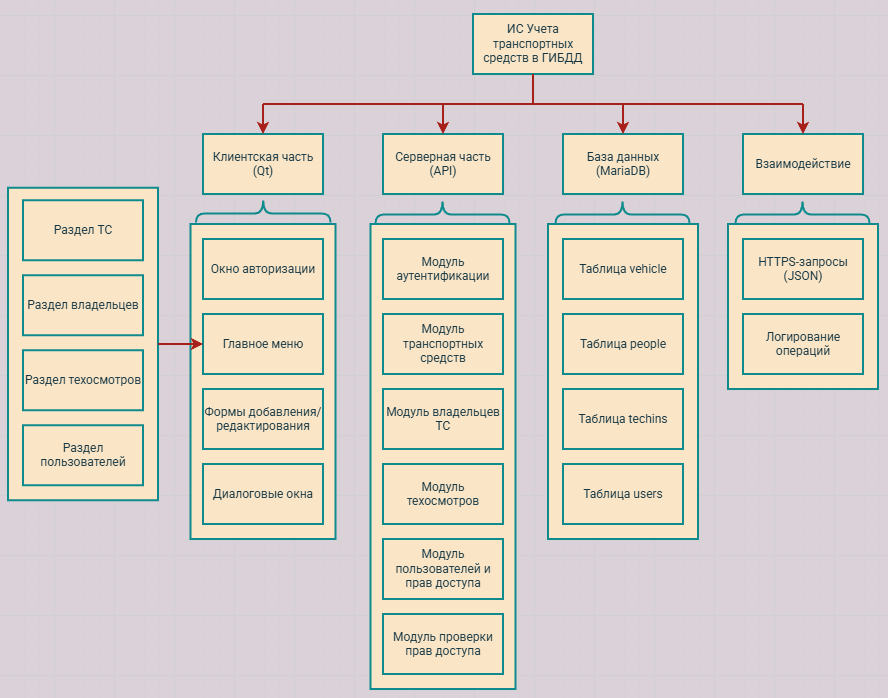
Система готова к внедрению и работе. Выполненные работы соответствуют требованиям ГОСТ и учебному заданию.

Список информативных источников

1. ГОСТ 19.201-78 Единая система программной документации. Техническое задание. Требования к содержанию и оформлению. 1978. Режим доступа: http://protect.gost.ru/document.aspx?control=7&id=155153
2. ГОСТ 19.504-79 Единая система программной документации. Руководство программиста. Требования к содержанию и оформлению. 1979. Режим доступа: https://docs.cntd.ru/document/1200007675.

Приложения

**Приложение 1. Иерархия ИС «учет транспортных средств в ГИБДД».**



**Приложение 2. Диаграма базы данных.**



**Приложение 3. Исходный код создания базы данных:**

CREATE TABLE `people` (

`id` int(11) NOT NULL,

`fname` tinytext COLLATE utf8\_unicode\_ci NOT NULL,

`lname` tinytext COLLATE utf8\_unicode\_ci NOT NULL,

`llname` tinytext COLLATE utf8\_unicode\_ci NOT NULL,

`pass\_number` int(11) NOT NULL,

`pass\_serial` int(11) NOT NULL,

`dob` date NOT NULL,

`pass\_issued` tinytext COLLATE utf8\_unicode\_ci NOT NULL,

`pass\_issued\_date` date NOT NULL,

`por` text COLLATE utf8\_unicode\_ci NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci;

CREATE TABLE `techins` (

`id` int(11) NOT NULL,

`date` date NOT NULL,

`vehicle\_id` int(11) NOT NULL,

`mileage` int(11) NOT NULL,

`status` tinytext COLLATE utf8\_unicode\_ci NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci;

CREATE TABLE `users` (

`id` int(11) NOT NULL,

`fname` tinytext COLLATE utf8\_unicode\_ci NOT NULL,

`lname` tinytext COLLATE utf8\_unicode\_ci NOT NULL,

`login` tinytext COLLATE utf8\_unicode\_ci NOT NULL,

`pass` tinytext COLLATE utf8\_unicode\_ci NOT NULL,

`can\_people\_select` tinyint(1) NOT NULL,

`can\_people\_delete` tinyint(1) NOT NULL,

`can\_people\_insert` tinyint(1) NOT NULL,

`can\_people\_update` tinyint(1) NOT NULL,

`can\_techins\_select` tinyint(1) NOT NULL,

`can\_techins\_delete` tinyint(1) NOT NULL,

`can\_techins\_insert` tinyint(1) NOT NULL,

`can\_techins\_update` tinyint(1) NOT NULL,

`can\_vehicle\_select` tinyint(1) NOT NULL,

`can\_vehicle\_delete` tinyint(1) NOT NULL,

`can\_vehicle\_insert` tinyint(1) NOT NULL,

`can\_vehicle\_update` tinyint(1) NOT NULL,

`can\_users\_select` tinyint(1) NOT NULL,

`can\_users\_delete` tinyint(1) NOT NULL,

`can\_users\_insert` tinyint(1) NOT NULL,

`can\_users\_update` tinyint(1) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci;

CREATE TABLE `vehicle` (

`id` int(11) NOT NULL,

`make` tinytext COLLATE utf8\_unicode\_ci NOT NULL,

`model` tinytext COLLATE utf8\_unicode\_ci NOT NULL,

`type` tinytext COLLATE utf8\_unicode\_ci NOT NULL,

`state\_number` tinytext COLLATE utf8\_unicode\_ci NOT NULL,

`number\_engine` tinytext COLLATE utf8\_unicode\_ci NOT NULL,

`number\_body` tinytext COLLATE utf8\_unicode\_ci NOT NULL,

`owner` int(11) NOT NULL,

`horsepower` int(11) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci;

ALTER TABLE `people`

ADD PRIMARY KEY (`id`);

ALTER TABLE `techins`

ADD PRIMARY KEY (`id`),

ADD KEY `fk\_techins\_vehicle` (`vehicle\_id`);

ALTER TABLE `users`

ADD PRIMARY KEY (`id`);

ALTER TABLE `vehicle`

ADD PRIMARY KEY (`id`),

ADD KEY `fk\_vehicle\_owner` (`owner`);

ALTER TABLE `people`

MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT;

ALTER TABLE `techins`

MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT;

ALTER TABLE `users`

MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT;

ALTER TABLE `vehicle`

MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT;

ALTER TABLE `techins`

ADD CONSTRAINT `fk\_techins\_vehicle` FOREIGN KEY (`vehicle\_id`) REFERENCES `vehicle` (`id`) ON UPDATE CASCADE;

ALTER TABLE `vehicle`

ADD CONSTRAINT `fk\_vehicle\_owner` FOREIGN KEY (`owner`) REFERENCES `people` (`id`) ON UPDATE CASCADE;

COMMIT;

**Приложение 4. Исходный код двигателя API:**

<?php

require\_once('settings.php');

require\_once("vendor/autoload.php");

require\_once('logs.php');

use Firebase\JWT\JWT;

use Firebase\JWT\Key;

$logs = new Logs();

$logs->requestReceived();

$headers = getallheaders();

if (!isset($headers['Authorization']) && $\_POST['method'] != 'auth'){

connectionClose(401);

exit();

}

function generateAuthToken($userId, $expirationTime = 3600) {

$payload = [

"sub" => $userId,

"iat" => time(), // Время создания токена

"exp" => time() + $expirationTime // Время истечения токена (через сколько секунд токен будет недействителен)

];

$jwt = JWT::encode($payload, TOKEN\_SECRETKEY, 'HS256'); // Подписываем токен с использованием секретного ключа

return $jwt;

}

function decodeAuthToken($token) {

try {

$decoded = JWT::decode($token, new Key(TOKEN\_SECRETKEY, 'HS256'));

return $decoded;

} catch (\Throwable $th) {

return $th;

}

}

if (isset($headers['Authorization'])){

$data = file\_get\_contents("php://input");

if ($headers['Content-Type']== 'application/json') {

$data = json\_decode($data, true);

switch ($data['method']) {

case 'getMe':

$token = explode(" ",$headers['Authorization']);

$data = decodeAuthToken($token[1]);

if (!$data instanceof Throwable) {

$pdo = connect();

$sql = "SELECT \* FROM " . DB\_TABLE\_USERS . " WHERE `id` = :id";

$sth = $pdo->prepare($sql);

try {

$res = $sth->execute(

array(

'id' => $data->sub,

)

);

$row = $sth->fetch();

if ($row == false) {

echo '2001'; // Пользователь не найден

}else{

$filteredResult = removeNumericKeys($row);

echo json\_encode($filteredResult);

}

}catch (\Throwable $th) {

echo $th->getMessage();

}

}else{

if ($data->getMessage() === 'Expired token') {

connectionClose(401);

}

}

break;

case 'getVehicles':

$token = explode(" ",$headers['Authorization']);

$data = decodeAuthToken($token[1]);

if (!$data instanceof Throwable) {

if (accessAllowed($data->sub, 'select', DB\_TABLE\_VEHICLES)) {

$pdo = connect();

$sql = "SELECT \* FROM " . DB\_TABLE\_VEHICLES;

$sth = $pdo->prepare($sql);

try {

$res = $sth->execute();

$row = $sth->fetchAll();

if ($row == false) {

echo '3002'; // Ошибка запроса

}else{

$filteredResult = removeNumericKeys($row);

echo json\_encode($filteredResult);

}

}catch (\Throwable $th) {

echo $th->getMessage();

}

}else{

// echo "3003"; // Недостаточно прав

connectionClose(403);

}

}else{

if ($data->getMessage() === 'Expired token') {

connectionClose(401);

}

}

break;

case 'newVehicle':

$token = explode(" ",$headers['Authorization']);

$newVehicle = $data;

$data = decodeAuthToken($token[1]);

if (!$data instanceof Throwable) {

if (accessAllowed($data->sub, 'insert', DB\_TABLE\_VEHICLES)) {

$pdo = connect();

$sql = "INSERT INTO " . DB\_TABLE\_VEHICLES ." (`make`,`model`,`type`,`state\_number`,`number\_engine`,`number\_body`,`owner`,`horsepower`) VALUES (:make, :model, :type, :state\_number, :number\_engine, :number\_body, :owner, :horsepower)";

$sth = $pdo->prepare($sql);

try {

$res = $sth->execute([

'make' => $newVehicle['make'],

'model' => $newVehicle['model'],

'type' => $newVehicle['type'],

'state\_number' => $newVehicle['state\_number'],

'number\_engine' => $newVehicle['number\_engine'],

'number\_body' => $newVehicle['number\_body'],

'owner' => $newVehicle['owner'],

'horsepower' => $newVehicle['horsepower']

]);

$row = $sth->fetch();

echo $row;

}catch (\Throwable $th) {

echo $th->getMessage();

connectionClose(500);

}

}else{

// echo "3003"; // Недостаточно прав

connectionClose(403);

}

}else{

if ($data->getMessage() === 'Expired token') {

connectionClose(401);

}

}

break;

case 'deleteVehicle':

$token = explode(" ",$headers['Authorization']);

$vehicle = $data;

$data = decodeAuthToken($token[1]);

if (!$data instanceof Throwable) {

if (accessAllowed($data->sub, 'delete', DB\_TABLE\_VEHICLES)) {

$pdo = connect();

$sql = "DELETE FROM " . DB\_TABLE\_VEHICLES ." WHERE `id`=:id";

$sth = $pdo->prepare($sql);

try {

$res = $sth->execute([

'id' => $vehicle['id']

]);

$row = $sth->fetch();

echo $row;

}catch (\Throwable $th) {

echo $th->getMessage();

connectionClose(500);

}

}else{

// echo "3003"; // Недостаточно прав

connectionClose(403);

}

}else{

if ($data->getMessage() === 'Expired token') {

connectionClose(401);

}

}

break;

case 'editVehicle':

$token = explode(" ",$headers['Authorization']);

$vehicle = $data;

$data = decodeAuthToken($token[1]);

if (!$data instanceof Throwable) {

if (accessAllowed($data->sub, 'update', DB\_TABLE\_VEHICLES)) {

$pdo = connect();

$sql = "UPDATE " . DB\_TABLE\_VEHICLES ." SET `make`=:make,`model`=:model,`type`=:type,`state\_number`=:state\_number,`number\_engine`=:number\_engine,`number\_body`=:number\_body,`owner`=:owner,`horsepower`=:horsepower WHERE `id`=:id";

$sth = $pdo->prepare($sql);

try {

$res = $sth->execute([

'id' => $vehicle['id'],

'make' => $vehicle['make'],

'model' => $vehicle['model'],

'type' => $vehicle['type'],

'state\_number' => $vehicle['state\_number'],

'number\_engine' => $vehicle['number\_engine'],

'number\_body' => $vehicle['number\_body'],

'owner' => $vehicle['owner'],

'horsepower' => $vehicle['horsepower']

]);

$row = $sth->fetch();

echo $row;

}catch (\Throwable $th) {

echo $th->getMessage();

connectionClose(500);

}

}else{

// echo "3003"; // Недостаточно прав

connectionClose(403);

}

}else{

if ($data->getMessage() === 'Expired token') {

connectionClose(401);

}

}

break;

case 'getTechins':

$token = explode(" ",$headers['Authorization']);

$data = decodeAuthToken($token[1]);

if (!$data instanceof Throwable) {

if (accessAllowed($data->sub, 'select', DB\_TABLE\_TECHINS)) {

$pdo = connect();

$sql = "SELECT

techins.id,

techins.date,

techins.vehicle\_id,

vehicle.make AS vehicle\_make,

vehicle.model AS vehicle\_model,

vehicle.number\_body AS vehicle\_number\_body,

techins.mileage,

techins.status

FROM

techins

JOIN

vehicle ON techins.vehicle\_id = vehicle.id;

";

$sth = $pdo->prepare($sql);

try {

$res = $sth->execute();

$row = $sth->fetchAll();

if ($row == false) {

echo '5002'; // Ошибка запроса

}else{

$filteredResult = removeNumericKeys($row);

echo json\_encode($filteredResult);

}

}catch (\Throwable $th) {

echo $th->getMessage();

}

}else{

// echo "5003"; // Недостаточно прав

connectionClose(403);

}

}else{

if ($data->getMessage() === 'Expired token') {

connectionClose(401);

}

}

break;

case 'newTechins':

$token = explode(" ",$headers['Authorization']);

$object = $data;

$data = decodeAuthToken($token[1]);

if (!$data instanceof Throwable) {

if (accessAllowed($data->sub, 'insert', DB\_TABLE\_TECHINS)) {

$pdo = connect();

$sql = "INSERT INTO " . DB\_TABLE\_TECHINS ." (`date`, `vehicle\_id`, `mileage`, `status`)VALUES (:date, :vehicle\_id, :mileage, :status)";

$sth = $pdo->prepare($sql);

try {

$res = $sth->execute([

'mileage' => $object['mileage'],

'date' => $object['date'],

'status' => $object['status'],

'vehicle\_id' => $object['vehicle\_id']

]);

$row = $sth->fetch();

echo $row;

}catch (\Throwable $th) {

echo $th->getMessage();

connectionClose(500);

}

}else{

// echo "3003"; // Недостаточно прав

connectionClose(403);

}

}else{

if ($data->getMessage() === 'Expired token') {

connectionClose(401);

}

}

break;

case 'editTechins':

$token = explode(" ",$headers['Authorization']);

$object = $data;

$data = decodeAuthToken($token[1]);

if (!$data instanceof Throwable) {

if (accessAllowed($data->sub, 'update', DB\_TABLE\_TECHINS)) {

$pdo = connect();

$sql = "UPDATE " . DB\_TABLE\_TECHINS ." SET `mileage`=:mileage,`date`=:date,`status`=:status,`vehicle\_id`=:vehicle\_id WHERE `id`=:id";

$sth = $pdo->prepare($sql);

try {

$res = $sth->execute([

'id' => $object['id'],

'mileage' => $object['mileage'],

'date' => $object['date'],

'status' => $object['status'],

'vehicle\_id' => $object['vehicle\_id']

]);

$row = $sth->fetch();

echo $row;

}catch (\Throwable $th) {

echo $th->getMessage();

connectionClose(500);

}

}else{

// echo "3003"; // Недостаточно прав

connectionClose(403);

}

}else{

if ($data->getMessage() === 'Expired token') {

connectionClose(401);

}

}

break;

case 'deleteTechins':

$token = explode(" ",$headers['Authorization']);

$object = $data;

$data = decodeAuthToken($token[1]);

if (!$data instanceof Throwable) {

if (accessAllowed($data->sub, 'delete', DB\_TABLE\_TECHINS)) {

$pdo = connect();

$sql = "DELETE FROM " . DB\_TABLE\_TECHINS ." WHERE `id`=:id";

$sth = $pdo->prepare($sql);

try {

$res = $sth->execute([

'id' => $object['id']

]);

$row = $sth->fetch();

echo $row;

}catch (\Throwable $th) {

echo $th->getMessage();

connectionClose(500);

}

}else{

// echo "3003"; // Недостаточно прав

connectionClose(403);

}

}else{

if ($data->getMessage() === 'Expired token') {

connectionClose(401);

}

}

break;

case 'getPeople':

$token = explode(" ",$headers['Authorization']);

$data = decodeAuthToken($token[1]);

if (!$data instanceof Throwable) {

if (accessAllowed($data->sub, 'select', DB\_TABLE\_PEOPLES)) {

$pdo = connect();

$sql = "SELECT \* FROM " . DB\_TABLE\_PEOPLES;

$sth = $pdo->prepare($sql);

try {

$res = $sth->execute();

$row = $sth->fetchAll();

if ($row == false) {

echo '6002'; // Ошибка запроса

}else{

$filteredResult = removeNumericKeys($row);

echo json\_encode($filteredResult);

}

}catch (\Throwable $th) {

echo $th->getMessage();

}

}else{

// echo "6003"; // Недостаточно прав

connectionClose(403);

}

}else{

if ($data->getMessage() === 'Expired token') {

connectionClose(401);

}

}

break;

case 'newPeople':

$token = explode(" ",$headers['Authorization']);

$object = $data;

$data = decodeAuthToken($token[1]);

if (!$data instanceof Throwable) {

if (accessAllowed($data->sub, 'insert', DB\_TABLE\_PEOPLES)) {

$pdo = connect();

$sql = "INSERT INTO " . DB\_TABLE\_PEOPLES ." (`fname`,`lname`,`llname`,`pass\_number`,`pass\_serial`,`dob`,`pass\_issued`,`pass\_issued\_date`,`por`) VALUES (:fname, :lname, :llname, :pass\_number, :pass\_serial, :dob, :pass\_issued, :pass\_issued\_date, :por)";

$sth = $pdo->prepare($sql);

try {

$res = $sth->execute([

'fname' => $object['name'],

'lname' => $object['lname'],

'llname' => $object['llname'],

'pass\_number' => $object['pn'],

'pass\_serial' => $object['ps'],

'dob' => $object['dob'],

'pass\_issued' => $object['pi'],

'pass\_issued\_date' => $object['pid'],

'por' => $object['por']

]);

$row = $sth->fetch();

echo $row;

}catch (\Throwable $th) {

echo $th->getMessage();

connectionClose(500);

}

}else{

// echo "3003"; // Недостаточно прав

connectionClose(403);

}

}else{

if ($data->getMessage() === 'Expired token') {

connectionClose(401);

}

}

break;

case 'editPeople':

$token = explode(" ",$headers['Authorization']);

$object = $data;

$data = decodeAuthToken($token[1]);

if (!$data instanceof Throwable) {

if (accessAllowed($data->sub, 'update', DB\_TABLE\_PEOPLES)) {

$pdo = connect();

$sql = "UPDATE " . DB\_TABLE\_PEOPLES ." SET `fname`=:fname,`lname`=:lname,`llname`=:llname,`pass\_number`=:pass\_number,`pass\_serial`=:pass\_serial,`dob`=:dob,`pass\_issued`=:pass\_issued,`pass\_issued\_date`=:pass\_issued\_date,`por`=:por WHERE `id`=:id";

$sth = $pdo->prepare($sql);

try {

$res = $sth->execute([

'id' => $object['id'],

'fname' => $object['name'],

'lname' => $object['lname'],

'llname' => $object['llname'],

'pass\_number' => $object['pn'],

'pass\_serial' => $object['ps'],

'dob' => $object['dob'],

'pass\_issued' => $object['pi'],

'pass\_issued\_date' => $object['pid'],

'por' => $object['por']

]);

$row = $sth->fetch();

echo $row;

}catch (\Throwable $th) {

echo $th->getMessage();

connectionClose(500);

}

}else{

// echo "3003"; // Недостаточно прав

connectionClose(403);

}

}else{

if ($data->getMessage() === 'Expired token') {

connectionClose(401);

}

}

break;

case 'deletePeople':

$token = explode(" ",$headers['Authorization']);

$object = $data;

$data = decodeAuthToken($token[1]);

if (!$data instanceof Throwable) {

if (accessAllowed($data->sub, 'delete', DB\_TABLE\_PEOPLES)) {

$pdo = connect();

$sql = "DELETE FROM " . DB\_TABLE\_PEOPLES ." WHERE `id`=:id";

$sth = $pdo->prepare($sql);

try {

$res = $sth->execute([

'id' => $object['id']

]);

$row = $sth->fetch();

echo $row;

}catch (\Throwable $th) {

echo $th->getMessage();

connectionClose(500);

}

}else{

// echo "3003"; // Недостаточно прав

connectionClose(403);

}

}else{

if ($data->getMessage() === 'Expired token') {

connectionClose(401);

}

}

break;

case 'getUsers':

$token = explode(" ",$headers['Authorization']);

$data = decodeAuthToken($token[1]);

if (!$data instanceof Throwable) {

if (accessAllowed($data->sub, 'select', DB\_TABLE\_USERS)) {

$pdo = connect();

$sql = "SELECT \* FROM " . DB\_TABLE\_USERS;

$sth = $pdo->prepare($sql);

try {

$res = $sth->execute();

$row = $sth->fetchAll();

if ($row == false) {

echo '6002'; // Ошибка запроса

}else{

$filteredResult = removeNumericKeys($row);

echo json\_encode($filteredResult);

}

}catch (\Throwable $th) {

echo $th->getMessage();

}

}else{

// echo "6003"; // Недостаточно прав

connectionClose(403);

}

}else{

if ($data->getMessage() === 'Expired token') {

connectionClose(401);

}

}

break;

case 'deleteUser':

$token = explode(" ",$headers['Authorization']);

$object = $data;

$data = decodeAuthToken($token[1]);

if (!$data instanceof Throwable) {

if (accessAllowed($data->sub, 'delete', DB\_TABLE\_USERS)) {

$pdo = connect();

$sql = "DELETE FROM " . DB\_TABLE\_USERS ." WHERE `id`=:id";

$sth = $pdo->prepare($sql);

try {

$res = $sth->execute([

'id' => $object['id']

]);

$row = $sth->fetch();

echo $row;

}catch (\Throwable $th) {

echo $th->getMessage();

connectionClose(500);

}

}else{

// echo "3003"; // Недостаточно прав

connectionClose(403);

}

}else{

if ($data->getMessage() === 'Expired token') {

connectionClose(401);

}

}

break;

case 'newUser':

$token = explode(" ",$headers['Authorization']);

$object = $data;

$data = decodeAuthToken($token[1]);

if (!$data instanceof Throwable) {

if (accessAllowed($data->sub, 'insert', DB\_TABLE\_USERS)) {

$pdo = connect();

$sql = "INSERT INTO " . DB\_TABLE\_USERS ." (`fname`, `lname`, `login`, `pass`, `can\_people\_select`, `can\_people\_delete`, `can\_people\_insert`, `can\_people\_update`, `can\_techins\_select`, `can\_techins\_delete`, `can\_techins\_insert`, `can\_techins\_update`, `can\_vehicle\_select`, `can\_vehicle\_delete`, `can\_vehicle\_insert`, `can\_vehicle\_update`, `can\_users\_select`, `can\_users\_delete`, `can\_users\_insert`, `can\_users\_update`) VALUES (:fname, :lname, :login, :pass, :can\_people\_select, :can\_people\_delete, :can\_people\_insert, :can\_people\_update, :can\_techins\_select, :can\_techins\_delete, :can\_techins\_insert, :can\_techins\_update, :can\_vehicle\_select, :can\_vehicle\_delete, :can\_vehicle\_insert, :can\_vehicle\_update, :can\_users\_select, :can\_users\_delete, :can\_users\_insert, :can\_users\_update)";

$sth = $pdo->prepare($sql);

try {

$res = $sth->execute([

'fname' => $object['name'],

'lname' => $object['lname'],

'login' => $object['login'],

'pass' => md5($object['pass']),

'can\_people\_select' => $object['can\_people\_select'],

'can\_people\_delete' => $object['can\_people\_delete'],

'can\_people\_insert' => $object['can\_people\_insert'],

'can\_people\_update' => $object['can\_people\_update'],

'can\_techins\_select' => $object['can\_techins\_select'],

'can\_techins\_delete' => $object['can\_techins\_delete'],

'can\_techins\_insert' => $object['can\_techins\_insert'],

'can\_techins\_update' => $object['can\_techins\_update'],

'can\_vehicle\_select' => $object['can\_vehicle\_select'],

'can\_vehicle\_delete' => $object['can\_vehicle\_delete'],

'can\_vehicle\_insert' => $object['can\_vehicle\_insert'],

'can\_vehicle\_update' => $object['can\_vehicle\_update'],

'can\_users\_select' => $object['can\_users\_select'],

'can\_users\_delete' => $object['can\_users\_delete'],

'can\_users\_insert' => $object['can\_users\_insert'],

'can\_users\_update' => $object['can\_users\_update']

]);

$row = $sth->fetch();

echo $row;

}catch (\Throwable $th) {

echo $th->getMessage();

connectionClose(500);

}

}else{

// echo "3003"; // Недостаточно прав

connectionClose(403);

}

}else{

if ($data->getMessage() === 'Expired token') {

connectionClose(401);

}

}

break;

case 'editUser':

$token = explode(" ",$headers['Authorization']);

$object = $data;

$data = decodeAuthToken($token[1]);

if (!$data instanceof Throwable) {

if (accessAllowed($data->sub, 'update', DB\_TABLE\_USERS)) {

$pdo = connect();

var\_dump(mb\_strlen($object['pass']));

if (mb\_strlen($object['pass']) > 0){

$sql = "UPDATE " . DB\_TABLE\_USERS ." SET `fname`=:fname,`lname`=:lname,`login`=:login,`pass`=:pass,`can\_people\_select`=:can\_people\_select,`can\_people\_delete`=:can\_people\_delete,`can\_people\_insert`=:can\_people\_insert,`can\_people\_update`=:can\_people\_update,`can\_techins\_select`=:can\_techins\_select,`can\_techins\_delete`=:can\_techins\_delete,`can\_techins\_insert`=:can\_techins\_insert,`can\_techins\_update`=:can\_techins\_update,`can\_vehicle\_select`=:can\_vehicle\_select,`can\_vehicle\_delete`=:can\_vehicle\_delete,`can\_vehicle\_insert`=:can\_vehicle\_insert,`can\_vehicle\_update`=:can\_vehicle\_update,`can\_users\_select`=:can\_users\_select,`can\_users\_delete`=:can\_users\_delete,`can\_users\_insert`=:can\_users\_insert,`can\_users\_update`=:can\_users\_update WHERE `id`=:id";

}else{

$sql = "UPDATE " . DB\_TABLE\_USERS ." SET `fname`=:fname,`lname`=:lname,`login`=:login,`can\_people\_select`=:can\_people\_select,`can\_people\_delete`=:can\_people\_delete,`can\_people\_insert`=:can\_people\_insert,`can\_people\_update`=:can\_people\_update,`can\_techins\_select`=:can\_techins\_select,`can\_techins\_delete`=:can\_techins\_delete,`can\_techins\_insert`=:can\_techins\_insert,`can\_techins\_update`=:can\_techins\_update,`can\_vehicle\_select`=:can\_vehicle\_select,`can\_vehicle\_delete`=:can\_vehicle\_delete,`can\_vehicle\_insert`=:can\_vehicle\_insert,`can\_vehicle\_update`=:can\_vehicle\_update,`can\_users\_select`=:can\_users\_select,`can\_users\_delete`=:can\_users\_delete,`can\_users\_insert`=:can\_users\_insert,`can\_users\_update`=:can\_users\_update WHERE `id`=:id";

}

$sth = $pdo->prepare($sql);

try {

$arr = [

'id' => $object['id'],

'fname' => $object['name'],

'lname' => $object['lname'],

'login' => $object['login'],

'pass' => mb\_strlen($object['pass']) > 0 ? md5($object['pass']) : null,

'can\_people\_select' => $object['can\_people\_select'],

'can\_people\_delete' => $object['can\_people\_delete'],

'can\_people\_insert' => $object['can\_people\_insert'],

'can\_people\_update' => $object['can\_people\_update'],

'can\_techins\_select' => $object['can\_techins\_select'],

'can\_techins\_delete' => $object['can\_techins\_delete'],

'can\_techins\_insert' => $object['can\_techins\_insert'],

'can\_techins\_update' => $object['can\_techins\_update'],

'can\_vehicle\_select' => $object['can\_vehicle\_select'],

'can\_vehicle\_delete' => $object['can\_vehicle\_delete'],

'can\_vehicle\_insert' => $object['can\_vehicle\_insert'],

'can\_vehicle\_update' => $object['can\_vehicle\_update'],

'can\_users\_select' => $object['can\_users\_select'],

'can\_users\_delete' => $object['can\_users\_delete'],

'can\_users\_insert' => $object['can\_users\_insert'],

'can\_users\_update' => $object['can\_users\_update']

];

if (isset($arr['pass']) && $arr['pass'] == null) {

unset($arr['pass']);

}

$res = $sth->execute($arr);

$row = $sth->fetch();

echo $row;

}catch (\Throwable $th) {

echo $th->getMessage();

connectionClose(500);

}

}else{

// echo "3003"; // Недостаточно прав

connectionClose(403);

}

}else{

if ($data->getMessage() === 'Expired token') {

connectionClose(401);

}

}

break;

default:

connectionClose(404);

break;

}

}

}

function connectionClose(int $code) {

http\_response\_code($code);

header("Connection: close");

ob\_start();

$size = ob\_get\_length();

header("Content-Length: $size");

ob\_end\_flush();

flush();

}

function removeNumericKeys($data) {

$result = array();

foreach ($data as $key => $value) {

if (is\_array($value)) {

$result[$key] = removeNumericKeys($value);

} else {

if (is\_string($key)) {

$result[$key] = $value;

}

}

}

return $result;

}

// Проверка разрешений

function accessAllowed($userId, $method, $table) : bool {

switch ($method) {

case 'select':

switch ($table) {

case DB\_TABLE\_VEHICLES:

$pdo = connect();

$sql = "SELECT `can\_vehicle\_select` FROM " . DB\_TABLE\_USERS . " WHERE `id` = :id";

$sth = $pdo->prepare($sql);

$res = $sth->execute(

array(

'id' => $userId,

)

);

$row = $sth->fetch();

if ($row) {

return $row[0];

}

break;

case DB\_TABLE\_PEOPLES:

$pdo = connect();

$sql = "SELECT `can\_people\_select` FROM " . DB\_TABLE\_USERS . " WHERE `id` = :id";

$sth = $pdo->prepare($sql);

$res = $sth->execute(

array(

'id' => $userId,

)

);

$row = $sth->fetch();

if ($row) {

return $row[0];

}

break;

case DB\_TABLE\_USERS:

$pdo = connect();

$sql = "SELECT `can\_users\_select` FROM " . DB\_TABLE\_USERS . " WHERE `id` = :id";

$sth = $pdo->prepare($sql);

$res = $sth->execute(

array(

'id' => $userId,

)

);

$row = $sth->fetch();

if ($row) {

return $row[0];

}

break;

case DB\_TABLE\_TECHINS:

$pdo = connect();

$sql = "SELECT `can\_techins\_select` FROM " . DB\_TABLE\_USERS . " WHERE `id` = :id";

$sth = $pdo->prepare($sql);

$res = $sth->execute(

array(

'id' => $userId,

)

);

$row = $sth->fetch();

if ($row) {

return $row[0];

}

break;

}

break;

case 'update':

switch ($table) {

case DB\_TABLE\_VEHICLES:

$pdo = connect();

$sql = "SELECT `can\_vehicle\_update` FROM " . DB\_TABLE\_USERS . " WHERE `id` = :id";

$sth = $pdo->prepare($sql);

$res = $sth->execute(

array(

'id' => $userId,

)

);

$row = $sth->fetch();

if ($row) {

return $row[0];

}

break;

case DB\_TABLE\_PEOPLES:

$pdo = connect();

$sql = "SELECT `can\_people\_update` FROM " . DB\_TABLE\_USERS . " WHERE `id` = :id";

$sth = $pdo->prepare($sql);

$res = $sth->execute(

array(

'id' => $userId,

)

);

$row = $sth->fetch();

if ($row) {

return $row[0];

}

break;

case DB\_TABLE\_USERS:

$pdo = connect();

$sql = "SELECT `can\_users\_update` FROM " . DB\_TABLE\_USERS . " WHERE `id` = :id";

$sth = $pdo->prepare($sql);

$res = $sth->execute(

array(

'id' => $userId,

)

);

$row = $sth->fetch();

if ($row) {

return $row[0];

}

break;

case DB\_TABLE\_TECHINS:

$pdo = connect();

$sql = "SELECT `can\_techins\_update` FROM " . DB\_TABLE\_USERS . " WHERE `id` = :id";

$sth = $pdo->prepare($sql);

$res = $sth->execute(

array(

'id' => $userId,

)

);

$row = $sth->fetch();

if ($row) {

return $row[0];

}

break;

}

break;

case 'delete':

switch ($table) {

case DB\_TABLE\_VEHICLES:

$pdo = connect();

$sql = "SELECT `can\_vehicle\_delete` FROM " . DB\_TABLE\_USERS . " WHERE `id` = :id";

$sth = $pdo->prepare($sql);

$res = $sth->execute(

array(

'id' => $userId,

)

);

$row = $sth->fetch();

if ($row) {

return $row[0];

}

break;

case DB\_TABLE\_PEOPLES:

$pdo = connect();

$sql = "SELECT `can\_people\_delete` FROM " . DB\_TABLE\_USERS . " WHERE `id` = :id";

$sth = $pdo->prepare($sql);

$res = $sth->execute(

array(

'id' => $userId,

)

);

$row = $sth->fetch();

if ($row) {

return $row[0];

}

break;

case DB\_TABLE\_USERS:

$pdo = connect();

$sql = "SELECT `can\_users\_delete` FROM " . DB\_TABLE\_USERS . " WHERE `id` = :id";

$sth = $pdo->prepare($sql);

$res = $sth->execute(

array(

'id' => $userId,

)

);

$row = $sth->fetch();

if ($row) {

return $row[0];

}

break;

case DB\_TABLE\_TECHINS:

$pdo = connect();

$sql = "SELECT `can\_techins\_delete` FROM " . DB\_TABLE\_USERS . " WHERE `id` = :id";

$sth = $pdo->prepare($sql);

$res = $sth->execute(

array(

'id' => $userId,

)

);

$row = $sth->fetch();

if ($row) {

return $row[0];

}

break;

}

break;

case 'insert':

switch ($table) {

case DB\_TABLE\_VEHICLES:

$pdo = connect();

$sql = "SELECT `can\_vehicle\_insert` FROM " . DB\_TABLE\_USERS . " WHERE `id` = :id";

$sth = $pdo->prepare($sql);

$res = $sth->execute(

array(

'id' => $userId,

)

);

$row = $sth->fetch();

if ($row) {

return $row[0];

}

break;

case DB\_TABLE\_PEOPLES:

$pdo = connect();

$sql = "SELECT `can\_people\_insert` FROM " . DB\_TABLE\_USERS . " WHERE `id` = :id";

$sth = $pdo->prepare($sql);

$res = $sth->execute(

array(

'id' => $userId,

)

);

$row = $sth->fetch();

if ($row) {

return $row[0];

}

break;

case DB\_TABLE\_USERS:

$pdo = connect();

$sql = "SELECT `can\_users\_insert` FROM " . DB\_TABLE\_USERS . " WHERE `id` = :id";

$sth = $pdo->prepare($sql);

$res = $sth->execute(

array(

'id' => $userId,

)

);

$row = $sth->fetch();

if ($row) {

return $row[0];

}

break;

case DB\_TABLE\_TECHINS:

$pdo = connect();

$sql = "SELECT `can\_techins\_insert` FROM " . DB\_TABLE\_USERS . " WHERE `id` = :id";

$sth = $pdo->prepare($sql);

$res = $sth->execute(

array(

'id' => $userId,

)

);

$row = $sth->fetch();

if ($row) {

return $row[0];

}

break;

}

break;

}

}

if (isset($\_POST['method']) && $\_POST['method'] == 'auth' && isset($\_POST['login']) && isset($\_POST['pass'])){

$pdo = connect();

$sql = "SELECT `id` FROM " . DB\_TABLE\_USERS . " WHERE `login` = :login AND `pass` = :pass";

$sth = $pdo->prepare($sql);

try {

$res = $sth->execute(

array(

'login' => $\_POST['login'],

'pass' => md5($\_POST['pass'])

)

);

$row = $sth->fetch();

if ($row == false) {

echo '1001'; // Неверный логин/пароль

}else{

echo generateAuthToken($row['id']);

}

}catch (\Throwable $th) {

echo $th->getMessage();

}

}

function connect(){

try {

$conn = new PDO('mysql:dbname=' . DB\_NAME . ';host=' . DB\_SERVER, DB\_LOGIN, DB\_PASS);

$conn->exec("set names utf8mb4");

$conn->exec("set character set utf8mb4");

return $conn;

} catch (PDOException $exception) {

$logs->everywhere('db', "Ошибка подключения к базе данных: " . $exception->getMessage());

return "Ошибка подключения к базе данных: " . $exception->getMessage();

}

}

?>

**Приложение 5. Исходный код программы клиента (QT):**

// main.cpp

#include "mainwindow.h"

#include <QApplication>

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

MainWindow w;

w.show();

return a.exec();

}

// mainwindow.ui

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>MainWindow</class>

<widget class="QMainWindow" name="MainWindow">

<property name="windowModality">

<enum>Qt::NonModal</enum>

</property>

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>243</width>

<height>204</height>

</rect>

</property>

<property name="sizePolicy">

<sizepolicy hsizetype="Preferred" vsizetype="Preferred">

<horstretch>0</horstretch>

<verstretch>0</verstretch>

</sizepolicy>

</property>

<property name="minimumSize">

<size>

<width>243</width>

<height>204</height>

</size>

</property>

<property name="maximumSize">

<size>

<width>243</width>

<height>204</height>

</size>

</property>

<property name="windowTitle">

<string>Авторизация</string>

</property>

<widget class="QWidget" name="centralwidget">

<widget class="QLabel" name="label">

<property name="geometry">

<rect>

<x>70</x>

<y>11</y>

<width>111</width>

<height>16</height>

</rect>

</property>

<property name="text">

<string>Авторизация</string>

</property>

<property name="alignment">

<set>Qt::AlignCenter</set>

</property>

</widget>

<widget class="QPushButton" name="pushButton">

<property name="geometry">

<rect>

<x>40</x>

<y>121</y>

<width>171</width>

<height>31</height>

</rect>

</property>

<property name="text">

<string>Войти</string>

</property>

</widget>

<widget class="QLineEdit" name="lineEdit">

<property name="geometry">

<rect>

<x>40</x>

<y>31</y>

<width>171</width>

<height>31</height>

</rect>

</property>

<property name="placeholderText">

<string>Логин</string>

</property>

</widget>

<widget class="QLineEdit" name="lineEdit\_2">

<property name="geometry">

<rect>

<x>40</x>

<y>70</y>

<width>171</width>

<height>31</height>

</rect>

</property>

<property name="sizePolicy">

<sizepolicy hsizetype="Expanding" vsizetype="Fixed">

<horstretch>0</horstretch>

<verstretch>0</verstretch>

</sizepolicy>

</property>

<property name="styleSheet">

<string notr="true"/>

</property>

<property name="inputMask">

<string/>

</property>

<property name="echoMode">

<enum>QLineEdit::Password</enum>

</property>

<property name="placeholderText">

<string>Пароль</string>

</property>

</widget>

</widget>

<widget class="QMenuBar" name="menubar">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>243</width>

<height>21</height>

</rect>

</property>

</widget>

<widget class="QStatusBar" name="statusbar"/>

</widget>

<resources/>

<connections/>

</ui>

// mainwindow.h

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QtNetwork/QNetworkAccessManager>

#include <QtNetwork/QNetworkRequest>

#include <QtNetwork/QNetworkReply>

#include <QUrl>

#include <QUrlQuery>

#include <QMessageBox>

#include "secondaryWindow.h"

QT\_BEGIN\_NAMESPACE

namespace Ui { class MainWindow; }

QT\_END\_NAMESPACE

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

MainWindow(QWidget \*parent = nullptr);

~MainWindow();

private slots:

void on\_pushButton\_clicked();

private:

Ui::MainWindow \*ui;

QNetworkAccessManager \*networkManager;

SecondaryWindow \*secondaryWindow;

QString authToken;

};

#endif // MAINWINDOW\_H

// mainwindow.cpp

#include "mainwindow.h"

#include "ui\_mainwindow.h"

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::MainWindow)

, networkManager(new QNetworkAccessManager(this))

, secondaryWindow(nullptr)

{

ui->setupUi(this);

}

MainWindow::~MainWindow()

{

delete ui;

}

void MainWindow::on\_pushButton\_clicked()

{

QNetworkRequest request(QUrl("https://gvev.ru/ulsu/api/engine"));

request.setHeader(QNetworkRequest::ContentTypeHeader, "application/x-www-form-urlencoded");

QUrlQuery postData;

postData.addQueryItem("method", "auth");

postData.addQueryItem("login", ui->lineEdit->text());

postData.addQueryItem("pass", ui->lineEdit\_2->text());

QNetworkReply \*reply = networkManager->post(request, postData.toString(QUrl::FullyEncoded).toUtf8());

connect(reply, &QNetworkReply::finished, [=]() {

if (reply->error() == QNetworkReply::NoError) {

QByteArray response = reply->readAll();

QString responseString(response);

if (responseString == "1001"){

QMessageBox msgBox;

msgBox.setText("Неверный логин/пароль");

msgBox.exec();

}else{

authToken = responseString;

if (!secondaryWindow){

secondaryWindow = new SecondaryWindow(this, authToken);

this->hide();

secondaryWindow->show();

}

}

} else {

QMessageBox msgBox;

msgBox.setText("Ошибка:" + reply->errorString());

msgBox.exec();

}

reply->deleteLater();

});

}

// secondaryWindow.ui

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>SecondaryWindow</class>

<widget class="QWidget" name="SecondaryWindow">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>290</width>

<height>300</height>

</rect>

</property>

<property name="minimumSize">

<size>

<width>290</width>

<height>300</height>

</size>

</property>

<property name="maximumSize">

<size>

<width>290</width>

<height>300</height>

</size>

</property>

<property name="windowTitle">

<string>Меню</string>

</property>

<widget class="QPushButton" name="pushButton\_vehicles">

<property name="geometry">

<rect>

<x>70</x>

<y>112</y>

<width>141</width>

<height>31</height>

</rect>

</property>

<property name="text">

<string>Автомобили</string>

</property>

</widget>

<widget class="QPushButton" name="pushButton\_people">

<property name="geometry">

<rect>

<x>70</x>

<y>190</y>

<width>141</width>

<height>31</height>

</rect>

</property>

<property name="text">

<string>Люди</string>

</property>

</widget>

<widget class="QPushButton" name="pushButton\_techins">

<property name="geometry">

<rect>

<x>70</x>

<y>150</y>

<width>141</width>

<height>31</height>

</rect>

</property>

<property name="text">

<string>Тех.осмотры</string>

</property>

</widget>

<widget class="QPushButton" name="pushButton\_users">

<property name="geometry">

<rect>

<x>70</x>

<y>230</y>

<width>141</width>

<height>31</height>

</rect>

</property>

<property name="text">

<string>Сотрудники</string>

</property>

</widget>

<widget class="QLabel" name="label">

<property name="geometry">

<rect>

<x>60</x>

<y>10</y>

<width>171</width>

<height>20</height>

</rect>

</property>

<property name="text">

<string>Здравствуйте</string>

</property>

<property name="alignment">

<set>Qt::AlignCenter</set>

</property>

</widget>

<widget class="QLabel" name="label\_2">

<property name="geometry">

<rect>

<x>10</x>

<y>30</y>

<width>271</width>

<height>20</height>

</rect>

</property>

<property name="text">

<string/>

</property>

<property name="alignment">

<set>Qt::AlignCenter</set>

</property>

</widget>

</widget>

<resources/>

<connections/>

</ui>

// secondaryWindow.h

#ifndef SECONDARYWINDOW\_H

#define SECONDARYWINDOW\_H

#include <QMainWindow>

#include <QJsonDocument>

#include <QJsonArray>

#include <QJsonObject>

#include <QNetworkAccessManager>

#include <QNetworkReply>

#include <QNetworkRequest>

#include <QPushButton>

#include <QMessageBox>

#include "vehiclewindow.h"

#include "techinswindow.h"

#include "peoplewindow.h"

#include "userswindow.h"

namespace Ui {

class SecondaryWindow;

}

class SecondaryWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit SecondaryWindow(QWidget \*parent = nullptr, const QString &token = "");

~SecondaryWindow();

private slots:

void onNetworkReplyFinished();

void setButtonAvailability(int value, QPushButton \*button);

void onVehicleButtonClicked();

void onTechinsButtonClicked();

void onPeopleButtonClicked();

void onUsersButtonClicked();

private:

Ui::SecondaryWindow \*ui;

QString authToken;

QNetworkAccessManager \*networkManager;

QJsonArray jsonArray;

QJsonObject userPermissions;

};

#endif // SECONDARYWINDOW\_H

// secondaryWindow.cpp

#include "secondaryWindow.h"

#include "ui\_secondaryWindow.h"

#include "techinswindow.h"

#include <QJsonDocument>

#include <QJsonObject>

#include <QDebug>

SecondaryWindow::SecondaryWindow(QWidget \*parent, const QString &token) :

QMainWindow(parent),

ui(new Ui::SecondaryWindow),

authToken(token),

networkManager(new QNetworkAccessManager(this))

{

ui->setupUi(this);

QJsonObject requestData;

requestData["method"] = "getMe";

QJsonDocument requestDoc(requestData);

QByteArray requestDataByteArray = requestDoc.toJson();

QNetworkRequest request(QUrl("https://gvev.ru/ulsu/api/engine"));

request.setRawHeader("Authorization", ("Bearer " + authToken).toUtf8());

request.setHeader(QNetworkRequest::ContentTypeHeader, "application/json");

QNetworkReply \*reply = networkManager->post(request, requestDataByteArray);

connect(reply, &QNetworkReply::finished, this, &SecondaryWindow::onNetworkReplyFinished);

connect(ui->pushButton\_vehicles, &QPushButton::clicked, this, &SecondaryWindow::onVehicleButtonClicked);

connect(ui->pushButton\_techins, &QPushButton::clicked, this, &SecondaryWindow::onTechinsButtonClicked);

connect(ui->pushButton\_people, &QPushButton::clicked, this, &SecondaryWindow::onPeopleButtonClicked);

connect(ui->pushButton\_users, &QPushButton::clicked, this, &SecondaryWindow::onUsersButtonClicked);

}

SecondaryWindow::~SecondaryWindow()

{

delete ui;

}

void SecondaryWindow::onNetworkReplyFinished()

{

QNetworkReply \*reply = qobject\_cast<QNetworkReply\*>(sender());

if (reply->error() == QNetworkReply::NoError) {

QByteArray responseData = reply->readAll();

QJsonDocument jsonDoc = QJsonDocument::fromJson(responseData);

QJsonObject jsonObject = jsonDoc.object();

userPermissions = jsonObject;

QString fullName = jsonObject.value("fname").toString() + " " + jsonObject.value("lname").toString();

ui->label\_2->setText(fullName);

setButtonAvailability(jsonObject.value("can\_vehicle\_select").toInt(), ui->pushButton\_vehicles);

setButtonAvailability(jsonObject.value("can\_people\_select").toInt(), ui->pushButton\_people);

setButtonAvailability(jsonObject.value("can\_techins\_select").toInt(), ui->pushButton\_techins);

setButtonAvailability(jsonObject.value("can\_users\_select").toInt(), ui->pushButton\_users);

} else {

QMessageBox msgBox;

msgBox.setText("Ошибка:" + reply->errorString());

msgBox.exec();

}

reply->deleteLater();

}

void SecondaryWindow::setButtonAvailability(int value, QPushButton \*button)

{

if (value == 1) {

button->setEnabled(true);

} else {

button->setEnabled(false);

}

}

void SecondaryWindow::onVehicleButtonClicked()

{

VehicleWindow \*vehicleWindow = new VehicleWindow(this, authToken, userPermissions);

vehicleWindow->setWindowTitle("Автомобили");

vehicleWindow->show();

}

void SecondaryWindow::onTechinsButtonClicked()

{

TechinsWindow \*techinsWindow = new TechinsWindow(this, authToken, userPermissions);

techinsWindow->setWindowTitle("Тех.осмотры");

techinsWindow->show();

}

void SecondaryWindow::onPeopleButtonClicked()

{

PeopleWindow \*peopleWindow = new PeopleWindow(this, authToken, userPermissions);

peopleWindow->setWindowTitle("Люди");

peopleWindow->show();

}

void SecondaryWindow::onUsersButtonClicked()

{

UsersWindow \*usersWindow = new UsersWindow(this, authToken, userPermissions);

usersWindow->setWindowTitle("Пользователи");

usersWindow->show();

}

// addvehiclewindow.ui

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>AddVehicleWindow</class>

<widget class="QWidget" name="AddVehicleWindow">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>400</width>

<height>300</height>

</rect>

</property>

<property name="windowTitle">

<string>Form</string>

</property>

</widget>

<resources/>

<connections/>

</ui>

// addvehiclewindow.h

#ifndef ADDVEHICLEWINDOW\_H

#define ADDVEHICLEWINDOW\_H

#include <QMainWindow>

#include <QNetworkAccessManager>

#include <QNetworkRequest>

#include <QNetworkReply>

#include <QJsonObject>

#include <QJsonDocument>

#include <QVBoxLayout>

#include <QFormLayout>

#include <QLineEdit>

#include <QPushButton>

#include <QMessageBox>

#include <QString>

namespace Ui {

class AddVehicleWindow;

}

class AddVehicleWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit AddVehicleWindow(QWidget \*parent, const QString &token, int id = -1, const QString &make = "", const QString &model = "", const QString &type = "", const QString &stateNumber = "", const QString &engineNumber = "", const QString &bodyNumber = "", const QString &owner = "", int horsepower = 0);

~AddVehicleWindow();

void setWindowTitle(const QString &title);

void setSubmitButtonText(const QString &text);

private slots:

void onSubmitButtonClicked();

void onNetworkReplyFinished();

signals:

void vehicleAdded();

private:

Ui::AddVehicleWindow \*ui;

QString authToken;

QNetworkAccessManager \*networkManager;

QLineEdit \*idField;

QLineEdit \*makeField;

QLineEdit \*modelField;

QLineEdit \*typeField;

QLineEdit \*stateNumberField;

QLineEdit \*numberEngineField;

QLineEdit \*numberBodyField;

QLineEdit \*ownerField;

QLineEdit \*horsepowerField;

QPushButton \*submitButton;

int vehicleId;

};

#endif // ADDVEHICLEWINDOW\_H

// addvehiclewindow.cpp

#include "addvehiclewindow.h"

#include "ui\_addvehiclewindow.h"

#include <QNetworkAccessManager>

#include <QNetworkRequest>

#include <QNetworkReply>

#include <QJsonDocument>

#include <QJsonObject>

#include <QMessageBox>

AddVehicleWindow::AddVehicleWindow(QWidget \*parent, const QString &token, int id, const QString &make, const QString &model, const QString &type, const QString &stateNumber, const QString &engineNumber, const QString &bodyNumber, const QString &owner, int horsepower) :

QMainWindow(parent),

ui(new Ui::AddVehicleWindow),

authToken(token),

networkManager(new QNetworkAccessManager(this)),

vehicleId(id)

{

ui->setupUi(this);

makeField = new QLineEdit(this);

modelField = new QLineEdit(this);

typeField = new QLineEdit(this);

stateNumberField = new QLineEdit(this);

numberEngineField = new QLineEdit(this);

numberBodyField = new QLineEdit(this);

ownerField = new QLineEdit(this);

horsepowerField = new QLineEdit(this);

submitButton = new QPushButton("Добавить", this);

if (vehicleId != -1) {

idField = new QLineEdit(this);

idField->setText(QString(id));

idField->setEnabled(false);

makeField->setText(make);

modelField->setText(model);

typeField->setText(type);

stateNumberField->setText(stateNumber);

numberEngineField->setText(engineNumber);

numberBodyField->setText(bodyNumber);

ownerField->setText(owner);

horsepowerField->setText(QString::number(horsepower));

submitButton->setText("Изменить");

}

QFormLayout \*formLayout = new QFormLayout;

formLayout->addRow("Марка", makeField);

formLayout->addRow("Модель", modelField);

formLayout->addRow("Тип кузова", typeField);

formLayout->addRow("Гос. Номер", stateNumberField);

formLayout->addRow("Номер двигателя", numberEngineField);

formLayout->addRow("Номер кузова", numberBodyField);

formLayout->addRow("Владелец", ownerField);

formLayout->addRow("Мощность двигателя (л.с.)", horsepowerField);

QVBoxLayout \*layout = new QVBoxLayout;

layout->addLayout(formLayout);

layout->addWidget(submitButton);

QWidget \*container = new QWidget;

container->setLayout(layout);

setCentralWidget(container);

connect(submitButton, &QPushButton::clicked, this, &AddVehicleWindow::onSubmitButtonClicked);

}

AddVehicleWindow::~AddVehicleWindow()

{

delete ui;

}

void AddVehicleWindow::onSubmitButtonClicked()

{

QNetworkRequest request(QUrl("https://gvev.ru/ulsu/api/engine"));

request.setHeader(QNetworkRequest::ContentTypeHeader, "application/json");

request.setRawHeader("Authorization", "Bearer " + authToken.toUtf8());

QJsonObject postData;

if (vehicleId > 0) {

postData["method"] = "editVehicle";

postData["id"] = vehicleId;

} else {

postData["method"] = "newVehicle";

}

postData["make"] = makeField->text();

postData["model"] = modelField->text();

postData["type"] = typeField->text();

postData["state\_number"] = stateNumberField->text();

postData["number\_engine"] = numberEngineField->text();

postData["number\_body"] = numberBodyField->text();

postData["owner"] = ownerField->text();

postData["horsepower"] = horsepowerField->text().toInt();

QNetworkReply \*reply = networkManager->post(request, QJsonDocument(postData).toJson());

connect(reply, &QNetworkReply::finished, this, &AddVehicleWindow::onNetworkReplyFinished);

}

void AddVehicleWindow::setWindowTitle(const QString &title){

QWidget::setWindowTitle(title);

}

void AddVehicleWindow::setSubmitButtonText(const QString &text){

submitButton->setText(text);

}

void AddVehicleWindow::onNetworkReplyFinished()

{

QNetworkReply \*reply = qobject\_cast<QNetworkReply\*>(sender());

if (reply->error() == QNetworkReply::NoError) {

close();

emit vehicleAdded();

} else {

QMessageBox::warning(this, "Ошибка", "Не удалось выполнить запрос: " + reply->errorString());

}

reply->deleteLater();

}

// addpeoplewindow.ui

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>AddPeopleWindow</class>

<widget class="QWidget" name="AddPeopleWindow">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>400</width>

<height>300</height>

</rect>

</property>

<property name="windowTitle">

<string>Form</string>

</property>

</widget>

<resources/>

<connections/>

</ui>

// addpeoplewindow.h

#ifndef ADDPEOPLEWINDOW\_H

#define ADDPEOPLEWINDOW\_H

#include <QMainWindow>

#include <QNetworkAccessManager>

#include <QNetworkRequest>

#include <QNetworkReply>

#include <QJsonObject>

#include <QJsonDocument>

#include <QVBoxLayout>

#include <QFormLayout>

#include <QLineEdit>

#include <QPushButton>

#include <QMessageBox>

#include <QString>

namespace Ui {

class AddPeopleWindow;

}

class AddPeopleWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit AddPeopleWindow(QWidget \*parent = nullptr, const QString &token = "", int id = -1,

const QString &name = "", const QString &lname = "", const QString &llname = "",

int pass\_number = 0, int pass\_serial = 0,

const QString &pass\_issued = "", const QString &pass\_issued\_date = "", const QString &dob = "", const QString &por = "");

~AddPeopleWindow();

void setWindowTitle(const QString &title);

void setSubmitButtonText(const QString &text);

private slots:

void onSubmitButtonClicked();

void onNetworkReplyFinished();

signals:

void peopleAdded();

private:

Ui::AddPeopleWindow \*ui;

QString authToken;

QNetworkAccessManager \*networkManager;

QLineEdit \*idField;

QLineEdit \*nameField;

QLineEdit \*lnameField;

QLineEdit \*llnameField;

QLineEdit \*pnField;

QLineEdit \*psField;

QLineEdit \*piField;

QLineEdit \*pidField;

QLineEdit \*dobField;

QLineEdit \*porField;

QPushButton \*submitButton;

int peopleId;

};

#endif // ADDPEOPLEWINDOW\_H

// addpeoplewindow.cpp

#include "addpeoplewindow.h"

#include "ui\_AddPeopleWindow.h"

#include <QNetworkAccessManager>

#include <QNetworkRequest>

#include <QNetworkReply>

#include <QJsonDocument>

#include <QJsonObject>

#include <QMessageBox>

AddPeopleWindow::AddPeopleWindow(QWidget \*parent, const QString &token, int id, const QString &name, const QString &lname,const QString &llname, int pass\_number, int pass\_serial,const QString &pass\_issued, const QString &pass\_issued\_date, const QString &dob, const QString &por) :

QMainWindow(parent),

ui(new Ui::AddPeopleWindow),

authToken(token),

networkManager(new QNetworkAccessManager(this)),

peopleId(id)

{

ui->setupUi(this);

nameField = new QLineEdit(this);

lnameField = new QLineEdit(this);

llnameField = new QLineEdit(this);

pnField = new QLineEdit(this);

psField = new QLineEdit(this);

piField = new QLineEdit(this);

pidField = new QLineEdit(this);

dobField = new QLineEdit(this);

porField = new QLineEdit(this);

submitButton = new QPushButton("Добавить", this);

if (peopleId != -1) {

idField = new QLineEdit(this);

idField->setText(QString(id));

idField->setEnabled(false);

nameField->setText(name);

lnameField->setText(lname);

llnameField->setText(llname);

pnField->setText(QString::number(pass\_number));

psField->setText(QString::number(pass\_serial));

piField->setText(pass\_issued);

pidField->setText(pass\_issued\_date);

dobField->setText(dob);

porField->setText(por);

submitButton->setText("Изменить");

}

QFormLayout \*formLayout = new QFormLayout;

formLayout->addRow("Имя", nameField);

formLayout->addRow("Фамилия", lnameField);

formLayout->addRow("Отчество", llnameField);

formLayout->addRow("Номер паспорта", pnField);

formLayout->addRow("Серия паспорта", psField);

formLayout->addRow("Кем выдан", piField);

formLayout->addRow("Дата выдачи", pidField);

formLayout->addRow("Дата рождения", dobField);

formLayout->addRow("Прописка", porField);

QVBoxLayout \*layout = new QVBoxLayout;

layout->addLayout(formLayout);

layout->addWidget(submitButton);

QWidget \*container = new QWidget;

container->setLayout(layout);

setCentralWidget(container);

connect(submitButton, &QPushButton::clicked, this, &AddPeopleWindow::onSubmitButtonClicked);

}

AddPeopleWindow::~AddPeopleWindow()

{

delete ui;

}

void AddPeopleWindow::onSubmitButtonClicked()

{

QNetworkRequest request(QUrl("http://gvev.ru/ulsu/api/engine"));

request.setHeader(QNetworkRequest::ContentTypeHeader, "application/json");

request.setRawHeader("Authorization", "Bearer " + authToken.toUtf8());

QJsonObject postData;

if (peopleId > 0) {

postData["method"] = "editPeople";

postData["id"] = peopleId;

} else {

postData["method"] = "newPeople";

}

postData["name"] = nameField->text();

postData["lname"] = lnameField->text();

postData["llname"] = llnameField->text();

postData["pn"] = pnField->text().toInt();

postData["ps"] = psField->text().toInt();

postData["pi"] = piField->text();

postData["pid"] = pidField->text();

postData["dob"] = dobField->text();

postData["por"] = porField->text();

QNetworkReply \*reply = networkManager->post(request, QJsonDocument(postData).toJson());

connect(reply, &QNetworkReply::finished, this, &AddPeopleWindow::onNetworkReplyFinished);

}

void AddPeopleWindow::setWindowTitle(const QString &title){

QWidget::setWindowTitle(title);

}

void AddPeopleWindow::setSubmitButtonText(const QString &text){

submitButton->setText(text);

}

void AddPeopleWindow::onNetworkReplyFinished()

{

QNetworkReply \*reply = qobject\_cast<QNetworkReply\*>(sender());

if (reply->error() == QNetworkReply::NoError) {

close();

emit peopleAdded();

} else {

QMessageBox::warning(this, "Ошибка", "Не удалось выполнить запрос: " + reply->errorString());

}

reply->deleteLater();

}

// addtechinswindow.ui

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>AddTechinsWindow</class>

<widget class="QWidget" name="AddTechinsWindow">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>400</width>

<height>300</height>

</rect>

</property>

<property name="windowTitle">

<string>Form</string>

</property>

</widget>

<resources/>

<connections/>

</ui>

// addtechinswindow.h

#ifndef ADDTECHINSWINDOW\_H

#define ADDTECHINSWINDOW\_H

#include <QMainWindow>

#include <QNetworkAccessManager>

#include <QNetworkReply>

#include <QLineEdit>

#include <QPushButton>

namespace Ui {

class AddTechinsWindow;

}

class AddTechinsWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit AddTechinsWindow(QWidget \*parent = nullptr, const QString &token = "", int id = -1, const QString &date = "", const QString &vehicleId = "", const QString &mileage = "", const QString &status = "");

~AddTechinsWindow();

signals:

void techinsAdded();

private slots:

void onSubmitButtonClicked();

void onNetworkReplyFinished();

private:

Ui::AddTechinsWindow \*ui;

QString authToken;

int techinsId;

QNetworkAccessManager \*networkManager;

QLineEdit \*dateEdit;

QLineEdit \*vehicleIdEdit;

QLineEdit \*mileageEdit;

QLineEdit \*statusEdit;

QPushButton \*submitButton;

};

#endif // ADDTECHINSWINDOW\_H

// addtechinswindow.cpp

#include "addtechinswindow.h"

#include "ui\_addtechinswindow.h"

#include <QMessageBox>

#include <QJsonDocument>

#include <QJsonObject>

#include <QNetworkAccessManager>

#include <QNetworkRequest>

#include <QNetworkReply>

#include <QLineEdit>

#include <QPushButton>

#include <QFormLayout>

#include <QVBoxLayout>

AddTechinsWindow::AddTechinsWindow(QWidget \*parent, const QString &token, int id, const QString &date, const QString &vehicleId, const QString &mileage, const QString &status) :

QMainWindow(parent),

ui(new Ui::AddTechinsWindow),

authToken(token),

techinsId(id),

networkManager(new QNetworkAccessManager(this))

{

ui->setupUi(this);

dateEdit = new QLineEdit(this);

vehicleIdEdit = new QLineEdit(this);

mileageEdit = new QLineEdit(this);

statusEdit = new QLineEdit(this);

submitButton = new QPushButton("Добавить", this);

if (id != -1) {

dateEdit->setText(date);

vehicleIdEdit->setText(vehicleId);

mileageEdit->setText(mileage);

statusEdit->setText(status);

submitButton->setText("Изменить");

}

QFormLayout \*formLayout = new QFormLayout;

formLayout->addRow("Дата", dateEdit);

formLayout->addRow("ID автомобиля", vehicleIdEdit);

formLayout->addRow("Пробег", mileageEdit);

formLayout->addRow("Статус", statusEdit);

QVBoxLayout \*layout = new QVBoxLayout;

layout->addLayout(formLayout);

layout->addWidget(submitButton);

QWidget \*container = new QWidget;

container->setLayout(layout);

setCentralWidget(container);

connect(submitButton, &QPushButton::clicked, this, &AddTechinsWindow::onSubmitButtonClicked);

}

AddTechinsWindow::~AddTechinsWindow()

{

delete ui;

}

void AddTechinsWindow::onSubmitButtonClicked()

{

QNetworkRequest request(QUrl("http://gvev.ru/ulsu/api/engine"));

request.setHeader(QNetworkRequest::ContentTypeHeader, "application/json");

request.setRawHeader("Authorization", "Bearer " + authToken.toUtf8());

QJsonObject postData;

if (techinsId == -1) {

postData["method"] = "newTechins";

} else {

postData["method"] = "editTechins";

postData["id"] = techinsId;

}

postData["date"] = dateEdit->text();

postData["vehicle\_id"] = vehicleIdEdit->text().toInt();

postData["mileage"] = mileageEdit->text().toInt();

postData["status"] = statusEdit->text();

QNetworkReply \*reply = networkManager->post(request, QJsonDocument(postData).toJson());

connect(reply, &QNetworkReply::finished, this, &AddTechinsWindow::onNetworkReplyFinished);

}

void AddTechinsWindow::onNetworkReplyFinished()

{

QNetworkReply \*reply = qobject\_cast<QNetworkReply\*>(sender());

if (reply->error() == QNetworkReply::NoError) {

QMessageBox::information(this, "Успех", techinsId == -1 ? "Запись успешно добавлена." : "Запись успешно изменена.");

emit techinsAdded();

close();

} else {

QMessageBox::warning(this, "Ошибка", "Не удалось выполнить запрос: " + reply->errorString());

}

reply->deleteLater();

}

// adduserwindow.ui

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>AddUserWindow</class>

<widget class="QWidget" name="AddUserWindow">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>400</width>

<height>300</height>

</rect>

</property>

<property name="windowTitle">

<string>Form</string>

</property>

</widget>

<resources/>

<connections/>

</ui>

// adduserwindow.h

#ifndef ADDUSERWINDOW\_H

#define ADDUSERWINDOW\_H

#include <QMainWindow>

#include <QNetworkAccessManager>

#include <QNetworkRequest>

#include <QNetworkReply>

#include <QJsonObject>

#include <QJsonDocument>

#include <QVBoxLayout>

#include <QFormLayout>

#include <QLineEdit>

#include <QPushButton>

#include <QMessageBox>

#include <QString>

#include <QCheckBox>

namespace Ui {

class AddUserWindow;

}

class AddUserWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit AddUserWindow(QWidget \*parent = nullptr, const QString &token = "", int id = -1, const QString &name = "", const QString &lname = "", const QString &login = "", const QString &pass = "", const bool &cps = false, const bool &cpd = false, const bool &cpi = false, const bool &cpu = false, const bool &cts = false, const bool &ctd = false, const bool &cti = false, const bool &ctu = false, const bool &cvs = false, const bool &cvd = false, const bool &cvi = false, const bool &cvu = false, const bool &cus = false, const bool &cud = false, const bool &cui = false, const bool &cuu = false);

~AddUserWindow();

void setWindowTitle(const QString &title);

void setSubmitButtonText(const QString &text);

signals:

void userAdded();

private slots:

void onSubmitButtonClicked();

void onNetworkReplyFinished();

private:

Ui::AddUserWindow \*ui;

QString authToken;

int userId;

QNetworkAccessManager \*networkManager;

QLineEdit \*idField;

QLineEdit \*nameField;

QLineEdit \*lnameField;

QLineEdit \*loginField;

QLineEdit \*passField;

QCheckBox \*cpsField;

QCheckBox \*cpdField;

QCheckBox \*cpiField;

QCheckBox \*cpuField;

QCheckBox \*ctsField;

QCheckBox \*ctdField;

QCheckBox \*ctiField;

QCheckBox \*ctuField;

QCheckBox \*cvsField;

QCheckBox \*cvdField;

QCheckBox \*cviField;

QCheckBox \*cvuField;

QCheckBox \*cusField;

QCheckBox \*cudField;

QCheckBox \*cuiField;

QCheckBox \*cuuField;

QPushButton \*submitButton;

};

#endif // ADDUSERWINDOW\_H

// adduserwindow.cpp

#include "adduserwindow.h"

#include "ui\_adduserwindow.h"

#include <QNetworkAccessManager>

#include <QNetworkRequest>

#include <QNetworkReply>

#include <QJsonDocument>

#include <QJsonObject>

#include <QMessageBox>

AddUserWindow::AddUserWindow(QWidget \*parent, const QString &token, int id, const QString &name, const QString &lname, const QString &login, const QString &pass, const bool &cps, const bool &cpd, const bool &cpi, const bool &cpu, const bool &cts, const bool &ctd, const bool &cti, const bool &ctu, const bool &cvs, const bool &cvd, const bool &cvi, const bool &cvu, const bool &cus, const bool &cud, const bool &cui, const bool &cuu) :

QMainWindow(parent),

ui(new Ui::AddUserWindow),

authToken(token),

networkManager(new QNetworkAccessManager(this)),

userId(id)

{

ui->setupUi(this);

nameField = new QLineEdit(this);

lnameField = new QLineEdit(this);

loginField = new QLineEdit(this);

passField = new QLineEdit(this);

cpsField = new QCheckBox(this);

cpdField = new QCheckBox(this);

cpiField = new QCheckBox(this);

cpuField = new QCheckBox(this);

ctsField = new QCheckBox(this);

ctdField = new QCheckBox(this);

ctiField = new QCheckBox(this);

ctuField = new QCheckBox(this);

cvsField = new QCheckBox(this);

cvdField = new QCheckBox(this);

cviField = new QCheckBox(this);

cvuField = new QCheckBox(this);

cusField = new QCheckBox(this);

cudField = new QCheckBox(this);

cuiField = new QCheckBox(this);

cuuField = new QCheckBox(this);

submitButton = new QPushButton("Добавить", this);

if (userId != -1) {

idField = new QLineEdit(this);

idField->setText(QString(id));

idField->setEnabled(false);

nameField->setText(name);

lnameField->setText(lname);

loginField->setText(login);

cpsField->setCheckState(Qt::CheckState(cps));

cpsField->setCheckState(Qt::CheckState(cps));

cpdField->setCheckState(Qt::CheckState(cpd));

cpiField->setCheckState(Qt::CheckState(cpi));

cpuField->setCheckState(Qt::CheckState(cpu));

ctsField->setCheckState(Qt::CheckState(cts));

ctdField->setCheckState(Qt::CheckState(ctd));

ctiField->setCheckState(Qt::CheckState(cti));

ctuField->setCheckState(Qt::CheckState(ctu));

cvsField->setCheckState(Qt::CheckState(cvs));

cvdField->setCheckState(Qt::CheckState(cvd));

cviField->setCheckState(Qt::CheckState(cvi));

cvuField->setCheckState(Qt::CheckState(cvu));

cusField->setCheckState(Qt::CheckState(cus));

cudField->setCheckState(Qt::CheckState(cud));

cuiField->setCheckState(Qt::CheckState(cui));

cuuField->setCheckState(Qt::CheckState(cuu));

submitButton->setText("Изменить");

}

QFormLayout \*formLayout = new QFormLayout;

formLayout->addRow("Имя", nameField);

formLayout->addRow("Фамилия", lnameField);

formLayout->addRow("Логин", loginField);

formLayout->addRow("Пароль", passField);

formLayout->addRow("Люди | Смотреть", cpsField);

formLayout->addRow("Люди | Удалять", cpdField);

formLayout->addRow("Люди | Добавлять", cpiField);

formLayout->addRow("Люди | Изменять", cpuField);

formLayout->addRow("Тех.осмотры | Смотреть", ctsField);

formLayout->addRow("Тех.осмотры | Удалять", ctdField);

formLayout->addRow("Тех.осмотры | Добавлять", ctiField);

formLayout->addRow("Тех.осмотры | Изменять", ctuField);

formLayout->addRow("Автомобили | Смотреть", cvsField);

formLayout->addRow("Автомобили | Удалять", cvdField);

formLayout->addRow("Автомобили | Добавлять", cviField);

formLayout->addRow("Автомобили | Изменять", cvuField);

formLayout->addRow("Сотрудники | Смотреть", cusField);

formLayout->addRow("Сотрудники | Удалять", cudField);

formLayout->addRow("Сотрудники | Добавлять", cuiField);

formLayout->addRow("Сотрудники | Изменять", cuuField);

QVBoxLayout \*layout = new QVBoxLayout;

layout->addLayout(formLayout);

layout->addWidget(submitButton);

QWidget \*container = new QWidget;

container->setLayout(layout);

setCentralWidget(container);

connect(submitButton, &QPushButton::clicked, this, &AddUserWindow::onSubmitButtonClicked);

}

AddUserWindow::~AddUserWindow()

{

delete ui;

}

void AddUserWindow::onSubmitButtonClicked()

{

QNetworkRequest request(QUrl("https://gvev.ru/ulsu/api/engine"));

request.setHeader(QNetworkRequest::ContentTypeHeader, "application/json");

request.setRawHeader("Authorization", "Bearer " + authToken.toUtf8());

QJsonObject postData;

if (userId > 0) {

postData["method"] = "editUser";

postData["id"] = userId;

} else {

postData["method"] = "newUser";

}

postData["name"] = nameField->text();

postData["lname"] = lnameField->text();

postData["login"] = loginField->text();

postData["pass"] = passField->text();

postData["can\_people\_select"] = cpsField->isChecked();

postData["can\_people\_delete"] = cpdField->isChecked();

postData["can\_people\_insert"] = cpiField->isChecked();

postData["can\_people\_update"] = cpuField->isChecked();

postData["can\_techins\_select"] = ctsField->isChecked();

postData["can\_techins\_delete"] = ctdField->isChecked();

postData["can\_techins\_insert"] = ctiField->isChecked();

postData["can\_techins\_update"] = ctuField->isChecked();

postData["can\_vehicle\_select"] = cvsField->isChecked();

postData["can\_vehicle\_delete"] = cvdField->isChecked();

postData["can\_vehicle\_insert"] = cviField->isChecked();

postData["can\_vehicle\_update"] = cvuField->isChecked();

postData["can\_users\_select"] = cusField->isChecked();

postData["can\_users\_delete"] = cudField->isChecked();

postData["can\_users\_insert"] = cuiField->isChecked();

postData["can\_users\_update"] = cuuField->isChecked();

QNetworkReply \*reply = networkManager->post(request, QJsonDocument(postData).toJson());

connect(reply, &QNetworkReply::finished, this, &AddUserWindow::onNetworkReplyFinished);

}

void AddUserWindow::setWindowTitle(const QString &title){

QWidget::setWindowTitle(title);

}

void AddUserWindow::setSubmitButtonText(const QString &text){

submitButton->setText(text);

}

void AddUserWindow::onNetworkReplyFinished()

{

QNetworkReply \*reply = qobject\_cast<QNetworkReply\*>(sender());

if (reply->error() == QNetworkReply::NoError) {

close();

emit userAdded();

} else {

QMessageBox::warning(this, "Ошибка", "Не удалось выполнить запрос: " + reply->errorString());

}

reply->deleteLater();

}

// vehiclewindow.ui

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>VehicleWindow</class>

<widget class="QWidget" name="VehicleWindow">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>400</width>

<height>300</height>

</rect>

</property>

<property name="windowTitle">

<string>Form</string>

</property>

</widget>

<resources/>

<connections/>

</ui>

// vehiclewindow.h

#ifndef VEHICLEWINDOW\_H

#define VEHICLEWINDOW\_H

#include <QMainWindow>

#include <QTableView>

#include <QPushButton>

#include <QVBoxLayout>

#include <QNetworkAccessManager>

#include <QNetworkRequest>

#include <QNetworkReply>

#include <QJsonDocument>

#include <QJsonObject>

#include <QJsonArray>

#include <QStandardItemModel>

#include <QMessageBox>

namespace Ui {

class VehicleWindow;

}

class VehicleWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit VehicleWindow(QWidget \*parent = nullptr, const QString &token = "", const QJsonObject &userPermissions = QJsonObject());

~VehicleWindow();

private slots:

void onAddButtonClicked();

void onDeleteButtonClicked();

void onEditButtonClicked();

void onNetworkReplyFinished();

void updateTable();

void onDeleteNetworkReplyFinished();

private:

Ui::VehicleWindow \*ui;

QString authToken;

QJsonObject permissions;

QNetworkAccessManager \*networkManager;

QTableView \*tableView;

QPushButton \*addButton;

QPushButton \*deleteButton;

QPushButton \*editButton;

};

#endif // VEHICLEWINDOW\_H

// vehiclewindow.cpp

#include "vehiclewindow.h"

#include "ui\_vehiclewindow.h"

#include "addvehiclewindow.h"

#include <QMessageBox>

#include <QJsonDocument>

#include <QJsonArray>

#include <QJsonObject>

#include <QHeaderView>

#include <QModelIndex>

#include <QDebug>

VehicleWindow::VehicleWindow(QWidget \*parent, const QString &token, const QJsonObject &userPermissions) :

QMainWindow(parent),

ui(new Ui::VehicleWindow),

authToken(token),

permissions(userPermissions),

networkManager(new QNetworkAccessManager(this))

{

ui->setupUi(this);

tableView = new QTableView(this);

addButton = new QPushButton("Добавить", this);

deleteButton = new QPushButton("Удалить", this);

editButton = new QPushButton("Изменить", this);

addButton->setEnabled(permissions.value("can\_vehicle\_insert").toInt() == 1);

deleteButton->setEnabled(permissions.value("can\_vehicle\_delete").toInt() == 1);

editButton->setEnabled(permissions.value("can\_vehicle\_update").toInt() == 1);

QVBoxLayout \*layout = new QVBoxLayout;

layout->addWidget(tableView);

layout->addWidget(addButton);

layout->addWidget(deleteButton);

layout->addWidget(editButton);

QWidget \*container = new QWidget;

container->setLayout(layout);

setCentralWidget(container);

connect(addButton, &QPushButton::clicked, this, &VehicleWindow::onAddButtonClicked);

connect(deleteButton, &QPushButton::clicked, this, &VehicleWindow::onDeleteButtonClicked);

connect(editButton, &QPushButton::clicked, this, &VehicleWindow::onEditButtonClicked);

QNetworkRequest request(QUrl("https://gvev.ru/ulsu/api/engine"));

request.setHeader(QNetworkRequest::ContentTypeHeader, "application/json");

request.setRawHeader("Authorization", "Bearer " + authToken.toUtf8());

QJsonObject postData;

postData["method"] = "getVehicles";

QNetworkReply \*reply = networkManager->post(request, QJsonDocument(postData).toJson());

connect(reply, &QNetworkReply::finished, this, &VehicleWindow::onNetworkReplyFinished);

}

VehicleWindow::~VehicleWindow()

{

delete ui;

}

void VehicleWindow::onAddButtonClicked()

{

AddVehicleWindow \*addWindow = new AddVehicleWindow(this, authToken);

connect(addWindow, &AddVehicleWindow::vehicleAdded, this, &VehicleWindow::updateTable);

addWindow->show();

}

void VehicleWindow::onDeleteButtonClicked()

{

QModelIndexList selection = tableView->selectionModel()->selectedRows();

if (selection.isEmpty()) {

QMessageBox::warning(this, "Ошибка", "Выберите строку для удаления.");

return;

}

int row = selection.first().row();

QStandardItemModel \*model = qobject\_cast<QStandardItemModel\*>(tableView->model());

int id = model->item(row, 0)->text().toInt();

QMessageBox::StandardButton reply;

reply = QMessageBox::question(this, "Подтверждение удаления","Вы уверены, что хотите удалить строку #" + QString::number(id) + "?",QMessageBox::Yes|QMessageBox::No);

if (reply == QMessageBox::Yes) {

QNetworkRequest request(QUrl("https://gvev.ru/ulsu/api/engine"));

request.setHeader(QNetworkRequest::ContentTypeHeader, "application/json");

request.setRawHeader("Authorization", "Bearer " + authToken.toUtf8());

QJsonObject postData;

postData["method"] = "deleteVehicle";

postData["id"] = id;

QNetworkReply \*networkReply = networkManager->post(request, QJsonDocument(postData).toJson());

connect(networkReply, &QNetworkReply::finished, this, &VehicleWindow::onDeleteNetworkReplyFinished);

}

}

void VehicleWindow::onEditButtonClicked()

{

QModelIndexList selectedRows = tableView->selectionModel()->selectedRows();

if (selectedRows.isEmpty()) {

QMessageBox::warning(this, "Ошибка", "Пожалуйста, выберите строку для редактирования.");

return;

}

int selectedRow = selectedRows.first().row();

QStandardItemModel \*model = qobject\_cast<QStandardItemModel\*>(tableView->model());

if (!model) {

QMessageBox::warning(this, "Ошибка", "Невозможно получить модель данных.");

return;

}

int id = model->item(selectedRow, 0)->text().toInt();

QString make = model->item(selectedRow, 1)->text();

QString modelStr = model->item(selectedRow, 2)->text();

QString type = model->item(selectedRow, 3)->text();

QString stateNumber = model->item(selectedRow, 4)->text();

QString engineNumber = model->item(selectedRow, 5)->text();

QString bodyNumber = model->item(selectedRow, 6)->text();

QString owner = model->item(selectedRow, 7)->text();

int horsepower = model->item(selectedRow, 8)->text().toInt();

AddVehicleWindow \*editWindow = new AddVehicleWindow(this, authToken, id, make, modelStr, type, stateNumber, engineNumber, bodyNumber, owner, horsepower);

editWindow->setWindowTitle("Редактировать запись");

editWindow->setSubmitButtonText("Изменить");

connect(editWindow, &AddVehicleWindow::vehicleAdded, this, &VehicleWindow::updateTable);

editWindow->show();

}

void VehicleWindow::updateTable()

{

QNetworkRequest request(QUrl("https://gvev.ru/ulsu/api/engine"));

request.setHeader(QNetworkRequest::ContentTypeHeader, "application/json");

request.setRawHeader("Authorization", "Bearer " + authToken.toUtf8());

QJsonObject postData;

postData["method"] = "getVehicles";

QNetworkReply \*reply = networkManager->post(request, QJsonDocument(postData).toJson());

connect(reply, &QNetworkReply::finished, this, &VehicleWindow::onNetworkReplyFinished);

}

void VehicleWindow::onNetworkReplyFinished()

{

QNetworkReply \*reply = qobject\_cast<QNetworkReply\*>(sender());

if (reply->error() == QNetworkReply::NoError) {

QByteArray response = reply->readAll();

QJsonDocument jsonResponse = QJsonDocument::fromJson(response);

QJsonArray jsonArray = jsonResponse.array();

QStandardItemModel \*model = new QStandardItemModel(jsonArray.size(), 9, this);

model->setHeaderData(0, Qt::Horizontal, "ID");

model->setHeaderData(1, Qt::Horizontal, "Марка");

model->setHeaderData(2, Qt::Horizontal, "Модель");

model->setHeaderData(3, Qt::Horizontal, "Тип кузова");

model->setHeaderData(4, Qt::Horizontal, "Гос. Номер");

model->setHeaderData(5, Qt::Horizontal, "Номер двигателя");

model->setHeaderData(6, Qt::Horizontal, "Номер кузова");

model->setHeaderData(7, Qt::Horizontal, "Владелец");

model->setHeaderData(8, Qt::Horizontal, "Мощность двигателя (л.с.)");

for (int i = 0; i < jsonArray.size(); ++i) {

QJsonObject jsonObject = jsonArray[i].toObject();

model->setItem(i, 0, new QStandardItem(QString::number(jsonObject["id"].toInt())));

model->setItem(i, 1, new QStandardItem(jsonObject["make"].toString()));

model->setItem(i, 2, new QStandardItem(jsonObject["model"].toString()));

model->setItem(i, 3, new QStandardItem(jsonObject["type"].toString()));

model->setItem(i, 4, new QStandardItem(jsonObject["state\_number"].toString()));

model->setItem(i, 5, new QStandardItem(jsonObject["number\_engine"].toString()));

model->setItem(i, 6, new QStandardItem(jsonObject["number\_body"].toString()));

model->setItem(i, 7, new QStandardItem(QString::number(jsonObject["owner"].toInt())));

model->setItem(i, 8, new QStandardItem(QString::number(jsonObject["horsepower"].toInt())));

}

tableView->setModel(model);

} else {

QMessageBox::warning(this, "Ошибка", "Не удалось получить данные: " + reply->errorString());

}

reply->deleteLater();

}

void VehicleWindow::onDeleteNetworkReplyFinished()

{

QNetworkReply \*reply = qobject\_cast<QNetworkReply\*>(sender());

if (reply->error() == QNetworkReply::NoError) {

QMessageBox::information(this, "Успех", "Строка успешно удалена.");

updateTable();

} else {

QMessageBox::warning(this, "Ошибка", "Не удалось удалить строку: " + reply->errorString());

}

reply->deleteLater();

}

// peoplewindow.ui

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>PeopleWindow</class>

<widget class="QWidget" name="PeopleWindow">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>400</width>

<height>300</height>

</rect>

</property>

<property name="windowTitle">

<string>Form</string>

</property>

</widget>

<resources/>

<connections/>

</ui>

// peoplewindow.h

#ifndef PEOPLEWINDOW\_H

#define PEOPLEWINDOW\_H

#include <QWidget>

#include <QMainWindow>

#include <QNetworkAccessManager>

#include <QNetworkRequest>

#include <QNetworkReply>

#include <QStandardItemModel>

#include <QTableView>

#include <QPushButton>

#include <QVBoxLayout>

#include <QJsonObject>

#include <QJsonArray>

#include <QJsonDocument>

#include <QMessageBox>

namespace Ui {

class PeopleWindow;

}

class PeopleWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit PeopleWindow(QWidget \*parent = nullptr, const QString &token = "", const QJsonObject &userPermissions = QJsonObject());

~PeopleWindow();

private slots:

void onAddButtonClicked();

void onDeleteButtonClicked();

void onEditButtonClicked();

void onNetworkReplyFinished();

void updateTable();

void onDeleteNetworkReplyFinished();

private:

Ui::PeopleWindow \*ui;

QString authToken;

QJsonObject permissions;

QNetworkAccessManager \*networkManager;

QTableView \*tableView;

QPushButton \*addButton;

QPushButton \*deleteButton;

QPushButton \*editButton;

};

#endif // PEOPLEWINDOW\_H

// peoplewindow.cpp

#include "peoplewindow.h"

#include "ui\_PeopleWindow.h"

#include "addpeoplewindow.h"

#include <QMessageBox>

#include <QJsonDocument>

#include <QJsonArray>

#include <QJsonObject>

#include <QHeaderView>

#include <QModelIndex>

#include <QDebug>

PeopleWindow::PeopleWindow(QWidget \*parent, const QString &token, const QJsonObject &userPermissions) :

QMainWindow(parent),

ui(new Ui::PeopleWindow),

authToken(token),

permissions(userPermissions),

networkManager(new QNetworkAccessManager(this))

{

ui->setupUi(this);

tableView = new QTableView(this);

addButton = new QPushButton("Добавить", this);

deleteButton = new QPushButton("Удалить", this);

editButton = new QPushButton("Изменить", this);

addButton->setEnabled(permissions.value("can\_people\_insert").toInt() == 1);

deleteButton->setEnabled(permissions.value("can\_people\_delete").toInt() == 1);

editButton->setEnabled(permissions.value("can\_people\_update").toInt() == 1);

QVBoxLayout \*layout = new QVBoxLayout;

layout->addWidget(tableView);

layout->addWidget(addButton);

layout->addWidget(deleteButton);

layout->addWidget(editButton);

QWidget \*container = new QWidget;

container->setLayout(layout);

setCentralWidget(container);

connect(addButton, &QPushButton::clicked, this, &PeopleWindow::onAddButtonClicked);

connect(deleteButton, &QPushButton::clicked, this, &PeopleWindow::onDeleteButtonClicked);

connect(editButton, &QPushButton::clicked, this, &PeopleWindow::onEditButtonClicked);

updateTable();

}

PeopleWindow::~PeopleWindow()

{

delete ui;

}

void PeopleWindow::updateTable()

{

QNetworkRequest request(QUrl("https://gvev.ru/ulsu/api/engine"));

request.setHeader(QNetworkRequest::ContentTypeHeader, "application/json");

request.setRawHeader("Authorization", "Bearer " + authToken.toUtf8());

QJsonObject postData;

postData["method"] = "getPeople"

QNetworkReply \*reply = networkManager->post(request, QJsonDocument(postData).toJson());

connect(reply, &QNetworkReply::finished, this, &PeopleWindow::onNetworkReplyFinished);

}

void PeopleWindow::onNetworkReplyFinished()

{

QNetworkReply \*reply = qobject\_cast<QNetworkReply\*>(sender());

if (reply->error() == QNetworkReply::NoError) {

QByteArray response = reply->readAll();

QJsonDocument jsonResponse = QJsonDocument::fromJson(response);

QJsonArray jsonArray = jsonResponse.array();

QStandardItemModel \*model = new QStandardItemModel(jsonArray.size(), 9, this);

model->setHeaderData(0, Qt::Horizontal, "ID");

model->setHeaderData(1, Qt::Horizontal, "Имя");

model->setHeaderData(2, Qt::Horizontal, "Фамилия");

model->setHeaderData(3, Qt::Horizontal, "Отчество");

model->setHeaderData(4, Qt::Horizontal, "Номер паспорта");

model->setHeaderData(5, Qt::Horizontal, "Серия паспорта");

model->setHeaderData(6, Qt::Horizontal, "Дата рождения");

model->setHeaderData(7, Qt::Horizontal, "Кем выдан");

model->setHeaderData(8, Qt::Horizontal, "Дата выдачи");

model->setHeaderData(9, Qt::Horizontal, "Прописка");

for (int i = 0; i < jsonArray.size(); ++i) {

QJsonObject jsonObject = jsonArray[i].toObject();

model->setItem(i, 0, new QStandardItem(QString::number(jsonObject["id"].toInt())));

model->setItem(i, 1, new QStandardItem(jsonObject["fname"].toString()));

model->setItem(i, 2, new QStandardItem(jsonObject["lname"].toString()));

model->setItem(i, 3, new QStandardItem(jsonObject["llname"].toString()));

model->setItem(i, 4, new QStandardItem(QString::number(jsonObject["pass\_number"].toInt())));

model->setItem(i, 5, new QStandardItem(QString::number(jsonObject["pass\_serial"].toInt())));

model->setItem(i, 6, new QStandardItem(jsonObject["dob"].toString()));

model->setItem(i, 7, new QStandardItem(jsonObject["pass\_issued"].toString()));

model->setItem(i, 8, new QStandardItem(jsonObject["pass\_issued\_date"].toString()));

model->setItem(i, 9, new QStandardItem(jsonObject["por"].toString()));

}

tableView->setModel(model);

} else {

QMessageBox::warning(this, "Ошибка", "Не удалось получить данные: " + reply->errorString());

}

reply->deleteLater();

}

void PeopleWindow::onAddButtonClicked()

{

AddPeopleWindow \*addWindow = new AddPeopleWindow(this, authToken);

connect(addWindow, &AddPeopleWindow::peopleAdded, this, &PeopleWindow::updateTable);

addWindow->show();

}

void PeopleWindow::onDeleteButtonClicked()

{

QModelIndexList selection = tableView->selectionModel()->selectedRows();

if (selection.isEmpty()) {

QMessageBox::warning(this, "Ошибка", "Выберите строку для удаления.");

return;

}

int row = selection.first().row();

QStandardItemModel \*model = qobject\_cast<QStandardItemModel\*>(tableView->model());

int id = model->item(row, 0)->text().toInt();

QMessageBox::StandardButton reply;

reply = QMessageBox::question(this, "Подтверждение удаления",

"Вы уверены, что хотите удалить строку #" + QString::number(id) + "?",

QMessageBox::Yes|QMessageBox::No);

if (reply == QMessageBox::Yes) {

QNetworkRequest request(QUrl("https://gvev.ru/ulsu/api/engine"));

request.setHeader(QNetworkRequest::ContentTypeHeader, "application/json");

request.setRawHeader("Authorization", "Bearer " + authToken.toUtf8());

QJsonObject postData;

postData["method"] = "deletePeople";

postData["id"] = id;

QNetworkReply \*networkReply = networkManager->post(request, QJsonDocument(postData).toJson());

connect(networkReply, &QNetworkReply::finished, this, &PeopleWindow::onDeleteNetworkReplyFinished);

}

}

void PeopleWindow::onEditButtonClicked()

{

QModelIndexList selectedRows = tableView->selectionModel()->selectedRows();

if (selectedRows.isEmpty()) {

QMessageBox::warning(this, "Ошибка", "Пожалуйста, выберите строку для редактирования.");

return;

}

int selectedRow = selectedRows.first().row();

QStandardItemModel \*model = qobject\_cast<QStandardItemModel\*>(tableView->model());

if (!model) {

QMessageBox::warning(this, "Ошибка", "Невозможно получить модель данных.");

return;

}

int id = model->item(selectedRow, 0)->text().toInt();

QString name = model->item(selectedRow, 1)->text();

QString lname = model->item(selectedRow, 2)->text();

QString llname = model->item(selectedRow, 3)->text();

int pass\_number = model->item(selectedRow, 4)->text().toInt();

int pass\_serial = model->item(selectedRow, 5)->text().toInt();

QString pass\_issued = model->item(selectedRow, 6)->text();

QString pass\_issued\_date = model->item(selectedRow, 7)->text();

QString dob = model->item(selectedRow, 8)->text();

QString por = model->item(selectedRow, 9)->text();

AddPeopleWindow \*editWindow = new AddPeopleWindow(this, authToken, id, name, lname, llname, pass\_number, pass\_serial, pass\_issued, pass\_issued\_date, dob, por);

editWindow->setWindowTitle("Редактировать запись");

editWindow->setSubmitButtonText("Изменить");

connect(editWindow, &AddPeopleWindow::peopleAdded, this, &PeopleWindow::updateTable);

editWindow->show();

}

void PeopleWindow::onDeleteNetworkReplyFinished()

{

QNetworkReply \*reply = qobject\_cast<QNetworkReply\*>(sender());

if (reply->error() == QNetworkReply::NoError) {

QMessageBox::information(this, "Успех", "Строка успешно удалена.");

updateTable();

} else {

QMessageBox::warning(this, "Ошибка", "Не удалось удалить строку: " + reply->errorString());

}

reply->deleteLater();

}

// techinswindow.ui

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>TechinsWindow</class>

<widget class="QWidget" name="TechinsWindow">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>400</width>

<height>300</height>

</rect>

</property>

<property name="windowTitle">

<string>Form</string>

</property>

</widget>

<resources/>

<connections/>

</ui>

// techinswindow.h

#ifndef TECHINSWINDOW\_H

#define TECHINSWINDOW\_H

#include <QMainWindow>

#include <QNetworkAccessManager>

#include <QNetworkReply>

#include <QTableView>

#include <QPushButton>

#include <QVBoxLayout>

#include <QJsonObject>

namespace Ui {

class TechinsWindow;

}

class TechinsWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit TechinsWindow(QWidget \*parent = nullptr, const QString &token = "", const QJsonObject &userPermissions = QJsonObject());

~TechinsWindow();

private slots:

void onAddButtonClicked();

void onDeleteButtonClicked();

void onEditButtonClicked();

void updateTable();

void onNetworkReplyFinished();

void onDeleteNetworkReplyFinished();

private:

Ui::TechinsWindow \*ui;

QString authToken;

QJsonObject permissions;

QNetworkAccessManager \*networkManager;

QTableView \*tableView;

QPushButton \*addButton;

QPushButton \*deleteButton;

QPushButton \*editButton;

};

#endif // TECHINSWINDOW\_H

// techinswindow.cpp

#include "techinswindow.h"

#include "ui\_techinswindow.h"

#include "addtechinswindow.h" // Добавляем форму для добавления записи

#include <QMessageBox>

#include <QJsonDocument>

#include <QJsonArray>

#include <QJsonObject>

#include <QHeaderView>

#include <QNetworkAccessManager>

#include <QNetworkRequest>

#include <QNetworkReply>

#include <QStandardItemModel>

#include <QFormLayout>

#include <QVBoxLayout>

TechinsWindow::TechinsWindow(QWidget \*parent, const QString &token, const QJsonObject &userPermissions) :

QMainWindow(parent),

ui(new Ui::TechinsWindow),

authToken(token),

permissions(userPermissions),

networkManager(new QNetworkAccessManager(this))

{

ui->setupUi(this);

tableView = new QTableView(this);

addButton = new QPushButton("Добавить", this);

deleteButton = new QPushButton("Удалить", this);

editButton = new QPushButton("Изменить", this);

addButton->setEnabled(permissions.value("can\_vehicle\_insert").toInt() == 1);

deleteButton->setEnabled(permissions.value("can\_vehicle\_delete").toInt() == 1);

editButton->setEnabled(permissions.value("can\_vehicle\_update").toInt() == 1);

QVBoxLayout \*layout = new QVBoxLayout;

layout->addWidget(tableView);

layout->addWidget(addButton);

layout->addWidget(deleteButton);

layout->addWidget(editButton);

QWidget \*container = new QWidget;

container->setLayout(layout);

setCentralWidget(container);

connect(addButton, &QPushButton::clicked, this, &TechinsWindow::onAddButtonClicked);

connect(deleteButton, &QPushButton::clicked, this, &TechinsWindow::onDeleteButtonClicked);

connect(editButton, &QPushButton::clicked, this, &TechinsWindow::onEditButtonClicked);

QNetworkRequest request(QUrl("https://gvev.ru/ulsu/api/engine"));

request.setHeader(QNetworkRequest::ContentTypeHeader, "application/json");

request.setRawHeader("Authorization", "Bearer " + authToken.toUtf8());

QJsonObject postData;

postData["method"] = "getTechins";

QNetworkReply \*reply = networkManager->post(request, QJsonDocument(postData).toJson());

connect(reply, &QNetworkReply::finished, this, &TechinsWindow::onNetworkReplyFinished);

}

TechinsWindow::~TechinsWindow()

{

delete ui;

}

void TechinsWindow::onAddButtonClicked()

{

AddTechinsWindow \*addWindow = new AddTechinsWindow(this, authToken);

connect(addWindow, &AddTechinsWindow::techinsAdded, this, &TechinsWindow::updateTable);

addWindow->show();

}

void TechinsWindow::onDeleteButtonClicked()

{

QModelIndexList selectedRows = tableView->selectionModel()->selectedRows();

if (selectedRows.isEmpty()) {

QMessageBox::warning(this, "Ошибка", "Пожалуйста, выберите строку для удаления.");

return;

}

int selectedRow = selectedRows.first().row();

QStandardItemModel \*model = qobject\_cast<QStandardItemModel\*>(tableView->model());

if (!model) {

QMessageBox::warning(this, "Ошибка", "Невозможно получить модель данных.");

return;

}

int id = model->item(selectedRow, 0)->text().toInt();

QMessageBox::StandardButton reply;

reply = QMessageBox::question(this, "Удаление", "Вы уверены, что хотите удалить строку #" + QString::number(id) + "?",

QMessageBox::Yes | QMessageBox::No);

if (reply == QMessageBox::Yes) {

QNetworkRequest request(QUrl("https://gvev.ru/ulsu/api/engine"));

request.setHeader(QNetworkRequest::ContentTypeHeader, "application/json");

request.setRawHeader("Authorization", "Bearer " + authToken.toUtf8());

QJsonObject postData;

postData["method"] = "deleteTechins";

postData["id"] = id;

QNetworkReply \*networkReply = networkManager->post(request, QJsonDocument(postData).toJson());

connect(networkReply, &QNetworkReply::finished, this, &TechinsWindow::onDeleteNetworkReplyFinished);

}

}

void TechinsWindow::onEditButtonClicked()

{

QModelIndexList selectedRows = tableView->selectionModel()->selectedRows();

if (selectedRows.isEmpty()) {

QMessageBox::warning(this, "Ошибка", "Пожалуйста, выберите строку для редактирования.");

return;

}

int selectedRow = selectedRows.first().row();

QStandardItemModel \*model = qobject\_cast<QStandardItemModel\*>(tableView->model());

if (!model) {

QMessageBox::warning(this, "Ошибка", "Невозможно получить модель данных.");

return;

}

int id = model->item(selectedRow, 0)->text().toInt();

QString date = model->item(selectedRow, 1)->text();

QString vehicleId = model->item(selectedRow, 2)->text();

QString mileage = model->item(selectedRow, 6)->text();

QString status = model->item(selectedRow, 7)->text();

AddTechinsWindow \*editWindow = new AddTechinsWindow(this, authToken, id, date, vehicleId, mileage, status);

editWindow->setWindowTitle("Редактировать запись");

connect(editWindow, &AddTechinsWindow::techinsAdded, this, &TechinsWindow::updateTable);

editWindow->show();

}

void TechinsWindow::updateTable()

{

QNetworkRequest request(QUrl("https://gvev.ru/ulsu/api/engine"));

request.setHeader(QNetworkRequest::ContentTypeHeader, "application/json");

request.setRawHeader("Authorization", "Bearer " + authToken.toUtf8());

QJsonObject postData;

postData["method"] = "getTechins";

QNetworkReply \*reply = networkManager->post(request, QJsonDocument(postData).toJson());

connect(reply, &QNetworkReply::finished, this, &TechinsWindow::onNetworkReplyFinished);

}

void TechinsWindow::onNetworkReplyFinished()

{

QNetworkReply \*reply = qobject\_cast<QNetworkReply\*>(sender());

if (reply->error() == QNetworkReply::NoError) {

QByteArray response = reply->readAll();

QJsonDocument jsonResponse = QJsonDocument::fromJson(response);

QJsonArray jsonArray = jsonResponse.array();

QStandardItemModel \*model = new QStandardItemModel(jsonArray.size(), 8, this);

model->setHeaderData(0, Qt::Horizontal, "№ стр.");

model->setHeaderData(1, Qt::Horizontal, "Дата");

model->setHeaderData(2, Qt::Horizontal, "ID Автомобиля");

model->setHeaderData(3, Qt::Horizontal, "Производитель");

model->setHeaderData(4, Qt::Horizontal, "Модель");

model->setHeaderData(5, Qt::Horizontal, "№ кузова");

model->setHeaderData(6, Qt::Horizontal, "Пробег");

model->setHeaderData(7, Qt::Horizontal, "Статус");

for (int i = 0; i < jsonArray.size(); ++i) {

QJsonObject jsonObject = jsonArray[i].toObject();

model->setItem(i, 0, new QStandardItem(QString::number(jsonObject["id"].toInt())));

model->setItem(i, 1, new QStandardItem(jsonObject["date"].toString()));

model->setItem(i, 2, new QStandardItem(QString::number(jsonObject["vehicle\_id"].toInt())));

model->setItem(i, 3, new QStandardItem(jsonObject["vehicle\_make"].toString()));

model->setItem(i, 4, new QStandardItem(jsonObject["vehicle\_model"].toString()));

model->setItem(i, 5, new QStandardItem(jsonObject["vehicle\_number\_body"].toString()));

model->setItem(i, 6, new QStandardItem(QString::number(jsonObject["mileage"].toInt())));

model->setItem(i, 7, new QStandardItem(jsonObject["status"].toString()));

}

tableView->setModel(model);

} else {

QMessageBox::warning(this, "Ошибка", "Не удалось получить данные: " + reply->errorString());

}

reply->deleteLater();

}

void TechinsWindow::onDeleteNetworkReplyFinished()

{

QNetworkReply \*reply = qobject\_cast<QNetworkReply\*>(sender());

if (reply->error() == QNetworkReply::NoError) {

QMessageBox::information(this, "Успех", "Запись успешно удалена.");

updateTable();

} else {

QMessageBox::warning(this, "Ошибка", "Не удалось удалить запись: " + reply->errorString());

}

reply->deleteLater();

}

// userswindow.ui

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>UsersWindow</class>

<widget class="QWidget" name="UsersWindow">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>400</width>

<height>300</height>

</rect>

</property>

<property name="windowTitle">

<string>Form</string>

</property>

</widget>

<resources/>

<connections/>

</ui>

// userswindow.h

#ifndef USERSWINDOW\_H

#define USERSWINDOW\_H

#include <QMainWindow>

#include <QNetworkAccessManager>

#include <QNetworkReply>

#include <QTableView>

#include <QPushButton>

#include <QVBoxLayout>

#include <QJsonObject>

#include <QJsonDocument>

#include <QJsonArray>

#include <QStandardItemModel>

#include <QMessageBox>

namespace Ui {

class UsersWindow;

}

class UsersWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit UsersWindow(QWidget \*parent = nullptr, const QString &token = "", const QJsonObject &userPermissions = QJsonObject());

~UsersWindow();

private slots:

void onAddButtonClicked();

void onEditButtonClicked();

void onDeleteButtonClicked();

void updateTable();

void onNetworkReplyFinished();

void onDeleteNetworkReplyFinished();

private:

Ui::UsersWindow \*ui;

QString authToken;

QJsonObject permissions;

QNetworkAccessManager \*networkManager;

QTableView \*tableView;

QPushButton \*addButton;

QPushButton \*editButton;

QPushButton \*deleteButton;

};

#endif // USERSWINDOW\_H

// userswindow.cpp

#include "userswindow.h"

#include "ui\_userswindow.h"

#include "adduserwindow.h"

UsersWindow::UsersWindow(QWidget \*parent, const QString &token, const QJsonObject &userPermissions) :

QMainWindow(parent),

ui(new Ui::UsersWindow),

authToken(token),

permissions(userPermissions),

networkManager(new QNetworkAccessManager(this))

{

ui->setupUi(this);

tableView = new QTableView(this);

addButton = new QPushButton("Добавить", this);

deleteButton = new QPushButton("Удалить", this);

editButton = new QPushButton("Изменить", this);

addButton->setEnabled(permissions.value("can\_vehicle\_insert").toInt() == 1);

deleteButton->setEnabled(permissions.value("can\_vehicle\_delete").toInt() == 1);

editButton->setEnabled(permissions.value("can\_vehicle\_update").toInt() == 1);

QVBoxLayout \*layout = new QVBoxLayout;

layout->addWidget(tableView);

layout->addWidget(addButton);

layout->addWidget(deleteButton);

layout->addWidget(editButton);

QWidget \*container = new QWidget;

container->setLayout(layout);

setCentralWidget(container);

connect(addButton, &QPushButton::clicked, this, &UsersWindow::onAddButtonClicked);

connect(editButton, &QPushButton::clicked, this, &UsersWindow::onEditButtonClicked);

connect(deleteButton, &QPushButton::clicked, this, &UsersWindow::onDeleteButtonClicked);

updateTable();

}

UsersWindow::~UsersWindow() {

delete ui;

}

void UsersWindow::updateTable()

{

QNetworkRequest request(QUrl("https://gvev.ru/ulsu/api/engine"));

request.setHeader(QNetworkRequest::ContentTypeHeader, "application/json");

request.setRawHeader("Authorization", "Bearer " + authToken.toUtf8());

QJsonObject postData;

postData["method"] = "getUsers";

QNetworkReply \*reply = networkManager->post(request, QJsonDocument(postData).toJson());

connect(reply, &QNetworkReply::finished, this, &UsersWindow::onNetworkReplyFinished);

}

void UsersWindow::onNetworkReplyFinished()

{

QNetworkReply \*reply = qobject\_cast<QNetworkReply\*>(sender());

if (reply->error() == QNetworkReply::NoError) {

QByteArray response = reply->readAll();

QJsonDocument jsonResponse = QJsonDocument::fromJson(response);

QJsonArray jsonArray = jsonResponse.array();

QStringList headers = {

"ID",

"Имя",

"Фамилия",

"Логин",

"Пароль",

"Люди | Смотреть",

"Люди | Удалять",

"Люди | Добавлять",

"Люди | Изменять",

"Тех.осмотры | Смотреть",

"Тех.осмотры | Удалять",

"Тех.осмотры | Добавлять",

"Тех.осмотры | Изменять",

"Автомобили | Смотреть",

"Автомобили | Удалять",

"Автомобили | Добавлять",

"Автомобили | Изменять",

"Сотрудники | Смотреть",

"Сотрудники | Удалять",

"Сотрудники | Добавлять",

"Сотрудники | Изменять"

};

QStandardItemModel \*model = new QStandardItemModel(jsonArray.size(), headers.size(), this);

for (int col = 0; col < headers.size(); ++col) {

model->setHeaderData(col, Qt::Horizontal, headers[col]);

}

for (int i = 0; i < jsonArray.size(); ++i) {

QJsonObject jsonObject = jsonArray[i].toObject();

model->setItem(i, 0, new QStandardItem(QString::number(jsonObject["id"].toInt())));

model->setItem(i, 1, new QStandardItem(jsonObject["fname"].toString()));

model->setItem(i, 2, new QStandardItem(jsonObject["lname"].toString()));

model->setItem(i, 3, new QStandardItem(jsonObject["login"].toString()));

model->setItem(i, 4, new QStandardItem(jsonObject["pass"].toString()));

QStringList keys = {

"can\_people\_select",

"can\_people\_delete",

"can\_people\_insert",

"can\_people\_update",

"can\_techins\_select",

"can\_techins\_delete",

"can\_techins\_insert",

"can\_techins\_update",

"can\_vehicle\_select",

"can\_vehicle\_delete",

"can\_vehicle\_insert",

"can\_vehicle\_update",

"can\_users\_select",

"can\_users\_delete",

"can\_users\_insert",

"can\_users\_update"

};

for (int j = 0; j < keys.size(); ++j) {

bool value = jsonObject.contains(keys[j]) ? (jsonObject[keys[j]].toInt() == 1) : false;

QString boolString = value ? "Да" : "Нет";

model->setItem(i, 5 + j, new QStandardItem(boolString));

}

}

tableView->setModel(model);

} else {

QMessageBox::warning(this, "Ошибка", "Не удалось получить данные: " + reply->errorString());

}

reply->deleteLater();

}

void UsersWindow::onAddButtonClicked()

{

AddUserWindow \*addWindow = new AddUserWindow(this, authToken);

connect(addWindow, &AddUserWindow::userAdded, this, &UsersWindow::updateTable);

addWindow->show();

}

void UsersWindow::onDeleteButtonClicked()

{

QModelIndexList selection = tableView->selectionModel()->selectedRows();

if (selection.isEmpty()) {

QMessageBox::warning(this, "Ошибка", "Выберите строку для удаления.");

return;

}

int row = selection.first().row();

QStandardItemModel \*model = qobject\_cast<QStandardItemModel\*>(tableView->model());

int id = model->item(row, 0)->text().toInt();

QMessageBox::StandardButton reply;

reply = QMessageBox::question(this, "Подтверждение удаления",

"Вы уверены, что хотите удалить строку #" + QString::number(id) + "?",

QMessageBox::Yes|QMessageBox::No);

if (reply == QMessageBox::Yes) {

QNetworkRequest request(QUrl("https://gvev.ru/ulsu/api/engine"));

request.setHeader(QNetworkRequest::ContentTypeHeader, "application/json");

request.setRawHeader("Authorization", "Bearer " + authToken.toUtf8());

QJsonObject postData;

postData["method"] = "deleteUser";

postData["id"] = id;

QNetworkReply \*networkReply = networkManager->post(request, QJsonDocument(postData).toJson());

connect(networkReply, &QNetworkReply::finished, this, &UsersWindow::onDeleteNetworkReplyFinished);

}

}

void UsersWindow::onEditButtonClicked()

{

QModelIndexList selectedRows = tableView->selectionModel()->selectedRows();

if (selectedRows.isEmpty()) {

QMessageBox::warning(this, "Ошибка", "Пожалуйста, выберите строку для редактирования.");

return;

}

int selectedRow = selectedRows.first().row();

QStandardItemModel \*model = qobject\_cast<QStandardItemModel\*>(tableView->model());

if (!model) {

QMessageBox::warning(this, "Ошибка", "Невозможно получить модель данных.");

return;

}

int id = model->item(selectedRow, 0)->text().toInt();

QString name = model->item(selectedRow, 1)->text();

QString lname = model->item(selectedRow, 2)->text();

QString login = model->item(selectedRow, 3)->text();

QString pass = model->item(selectedRow, 4)->text();

bool cps = model->item(selectedRow, 5)->text() == "Да" ? true : false;

bool cpd = model->item(selectedRow, 6)->text() == "Да" ? true : false;

bool cpi = model->item(selectedRow, 7)->text() == "Да" ? true : false;

bool cpu = model->item(selectedRow, 8)->text() == "Да" ? true : false;

bool cts = model->item(selectedRow, 9)->text() == "Да" ? true : false;

bool ctd = model->item(selectedRow, 10)->text() == "Да" ? true : false;

bool cti = model->item(selectedRow, 11)->text() == "Да" ? true : false;

bool ctu = model->item(selectedRow, 12)->text() == "Да" ? true : false;

bool cvs = model->item(selectedRow, 13)->text() == "Да" ? true : false;

bool cvd = model->item(selectedRow, 14)->text() == "Да" ? true : false;

bool cvi = model->item(selectedRow, 15)->text() == "Да" ? true : false;

bool cvu = model->item(selectedRow, 16)->text() == "Да" ? true : false;

bool cus = model->item(selectedRow, 17)->text() == "Да" ? true : false;

bool cud = model->item(selectedRow, 18)->text() == "Да" ? true : false;

bool cui = model->item(selectedRow, 19)->text() == "Да" ? true : false;

bool cuu = model->item(selectedRow, 20)->text() == "Да" ? true : false;

AddUserWindow \*editWindow = new AddUserWindow(this, authToken, id, name, lname, login, pass, cps, cpd, cpi, cpu, cts, ctd, cti ,ctu, cvs, cvd, cvi, cvu, cus, cud, cui, cuu);

editWindow->setWindowTitle("Редактировать запись");

editWindow->setSubmitButtonText("Изменить");

connect(editWindow, &AddUserWindow::userAdded, this, &UsersWindow::updateTable);

editWindow->show();

}

void UsersWindow::onDeleteNetworkReplyFinished()

{

QNetworkReply \*reply = qobject\_cast<QNetworkReply\*>(sender());

if (reply->error() == QNetworkReply::NoError) {

QMessageBox::information(this, "Успех", "Строка успешно удалена.");

updateTable();

} else {

QMessageBox::warning(this, "Ошибка", "Не удалось удалить строку: " + reply->errorString());

}

reply->deleteLater();

}