Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Институт информационных технологий

Кафедра «Микропроцессорные системы и сети»

|  |  |  |
| --- | --- | --- |
|  |  | **К защите допустить**  Заведующая кафедрой  \_\_\_\_\_\_\_\_\_\_ И. В. Кашникова  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2021 года |

**П О Я С Н И Т Е Л Ь Н А Я З А П И С К А**

**К ДИПЛОМНОМУ ПРОЕКТУ**

**НА ТЕМУ:**

**«****Программное средство по учету персонифицированных данных»**

|  |  |  |
| --- | --- | --- |
| Дипломник |  | А. А. Салов |
|  | (дата, подпись) |  |
| Руководитель |  | А. С. Петроченко |
|  | (дата, подпись) |  |
| Консультанты: по ЕСПД и ЕСКД |  | В.Н. Мухаметов |
|  | (дата, подпись) |  |
| Рецензент |  |  |
|  | (дата, подпись) |  |

МИНСК – 2021

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»**

**Институт информационных технологий**

**Кафедра Микропроцессорные системы и сети**

Специальность I-40 01 73 – Программное обеспечение информационных систем

***УТВЕРЖДАЮ***

Зав. кафедрой \_\_\_\_\_\_\_\_\_\_\_\_/ \_\_\_\_\_\_\_\_\_\_\_\_

(Подпись, Ф.И.О.)

"\_\_\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021 г.

ЗАДАНИЕ

**по дипломному проектированию**

Слушателю *Салову Анатолию Анатольевичу*

1.Тема проекта *Программное средство по учету персонифицированных данных*

утверждена приказом по Институту №  отгода**.**

2. Сроки сдачи слушателем законченного проекта   
 3. Исходные данные к проекту:

*.*

4. Перечень подлежащих разработке вопросов:

*4.1. Анализ существующих решений по теме проекта и постановка задачи.*

*4.2. Методы и модели, положенные в основу проекта*

*4.3. Разработка проекта программного средства*

*4.4. Разработка алгоритмов и реализация программного кода.*

*4.5. Тестирование разработанного программного средства.*

5. Перечень графического материала (с точным указанием обязательных чертежей и графиков):

*5.1. Схема ресурсов системы, ф. А3, чертеж*

*5.2. Схема работы системы ф.А3, чертеж.*

*5.3. Алгоритм авторизации пользователя ф.А3, чертеж.*

*5.4. Алгоритм регистрации нового пользователя ф.А3, чертеж.*

*5.5. Алгоритм экспорта списков ф. А3, чертеж*

6. Консультанты по проекту (с указанием относящихся к ним разделов проекта)

*ЕСКД, ЕСПД – Мухаметов В.Н.*

7. Дата выдачи задания

8. Календарный график работы над проектом на весь период проектирования

Руководитель (подпись, Ф.И.О., должность)  *А. С. Петроченко, старший преподователь*

Задание принял к исполнению (дата и подпись слушателя)

**Содержание**

[Условные обозначения и сокращения 4](#_Toc83246588)

[Введение 5](#_Toc83246589)

[1 Аналитический обзор аналогов. Постановка задачи на дипломное проектирование 7](#_Toc83246590)

[1.1 Аналитический обзор существующих аналогов 7](#_Toc83246591)

[1.2 Постановка цели и задач на дипломное проектирование 15](#_Toc83246592)

[2 Методы и модели, положенные в основу проекта 16](#_Toc83246593)

[2.1 Описание предметной области 16](#_Toc83246594)

[3 Проектирование приложения 20](#_Toc83246595)

[3.1 Функциональный анализ предметной области. 20](#_Toc83246596)

[3.2 Разработка структуры программного средства. 21](#_Toc83246597)

[3.3 Разработка структуры классов при использовании объектно-ориентированного стиля программирования приложений. 24](#_Toc83246598)

[3.4 Разработка структуры базы данных. 26](#_Toc83246599)

[3.5 Реализация подключения базы данных в программу на Qt 28](#_Toc83246600)

[4 Разработка алгоритмов и их программная реализация 30](#_Toc83246601)

[4.1 Алгоритм входа в систему 30](#_Toc83246602)

[4.2 Алгоритм регистрации нового пользователя 30](#_Toc83246603)

[4.3 Алгоритм экспорта списков 30](#_Toc83246604)

[4.4 Алгоритм сохранения клиента в БД 31](#_Toc83246605)

[4.5 Алгоритм удаления клиента в БД 32](#_Toc83246606)

[5 Тестирование разработанного программного средства 33](#_Toc83246607)

[Заключение 47](#_Toc83246608)

[Список использованных источников 48](#_Toc83246609)

[Приложение А (обязательное) Листинг кода 49](#_Toc83246610)

# Условные обозначения и сокращения

|  |  |  |
| --- | --- | --- |
| API | – | Application Programming Interface (программный интерфейс приложения) |
| ALFACRM | – | Cистема учета и хранения персонифицированных данных |
| COM | – | Component Object Model (модель компонентного объекта) |
| CRM | – | Customer Relationship Management (система управления взаимоотношениями с клиентами) |
| EXCEL | – | Microsoft Excel (программа для работы с электронными таблицами) |
| JSON | – | JavaScript Object Notation([текстовый формат](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82) [обмена данными](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BC%D0%B5%D0%BD_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D0%BC%D0%B8), основанный на [JavaScript](https://ru.wikipedia.org/wiki/JavaScript)) |
| OCRM |  | Opereted Customer Relationship Management (оперативное управление взаимоотношениями с клиентами) |
| QT | – | QT (фреймворк для разработки кроссплатформенного программного обеспечения на языке программирования C++) |
| QtScript | – | QtScript (скриптовый язык, который, начиная с версии 4.3.0, является составной частью Qt)  Язык основан на стандарте ECMAScript) |
| REST | – | Representational State Transfer («передача состояния представления») |
| SOA | – | Service-oriented Architecture (сервис-ориентированная архитектура) |
| SOAP | – | Simple Object Access Protocol (простой протокол доступа к объектам) |
| UML | – | Unified Modeling Language (унифицированный язык моделирования) |
| URL | – | Uniform Resource Locator (унифицированный указатель ресурса) |
| WCF | – | Windows Communication Foundation (фреймфорк, входящий в состав .Net Framework) |
| XML | – | eXtensible Markup Language (расширяемый язык разметки) |

# Введение

Для любой современной компании граммотная и своевременная автоматизация полезна, так как она позваляет упорядочить и ускорить бизнес-процессы, повысить их эффективность, точность и, конечно, увеличить прибыль организации.

Автоматизация бизнес-процессов предприятия – это возможность освобождать ресурсы, своевременно реагировать на изменения рынка, быстро принимать решение и распланировать дальнейшие действия. Это важно для любой организации, которая собирается расти и развиваться в современных условиях жесткой конкуренции и в режиме постоянной многозадачности.

В настоящее время существует множество инструментов для автоматизации процессов. Среди них можно выделить программное обеспечение, которое разработано для автоматизации конкретных процессов в компаниях.

***Системы управления проектами*.** [Trello](https://trello.com/), [Jira](https://www.atlassian.com/" \t "_blank), [Asana](https://www.atlassian.com/" \t "_blank) – это сервисы, в которых можно ставить задачи сотрудникам и распределять их между членами команды.

***Системы управления рабочим временем и персоналом***. [Bamboohr](https://www.bamboohr.com/), [Hurma](https://hurma.work/" \t "_blank) – это системы, где сотрудники могут оставлять заявки на отпуск, следить за рабочим временем, количеством рабочих и выходных дней.

***Системы управления клиентами (CRM)***. [AmoCRM](https://www.amocrm.ru/" \t "_blank), [Terrasoft](https://www.terrasoft.ua/ru" \t "_blank), [Bitrix24](https://www.bitrix24.ua/) – это системы учета взаимоотношений с клиентами. Так называют стратегию ведения бизнеса, когда отношения с клиентом выводятся на первый план и ставятся главной целью деятельности предприятия. Программа учета заявок, использующая CRM-стратегию, позволит выявить наиболее «доходных» клиентов, научиться более эффективному менеджменту и предотвратить уход главных клиентов к конкурентам, возможен импорт и экспорт crm базы клиентов в excel. Это современная, простая и удобная методика для управления отношениями с клиентами. Обеспечивает ведение базы данных и учет клиентов, выставленных им счетов, заключенных с ними договоров, других документов, продуктов, сотрудников. CRM-системы необходимы любому бизнесу, который работает напрямую с клиентами и стремится расширять число покупателей.

***ERP-систем****ы*. Это комплексные системы автоматизации всех процессов в компании. Самая известная ERP-система – это [SAP](https://www.sap.com/). Она гибкая, но долго внедряется и настраивается.

*CMS-системы*. [WordPress](https://ru.wordpress.com/" \t "_blank) и [Tilda](https://tilda.cc/ru/" \t "_blank) – тоже инструменты автоматизации бизнес-процессов. С их помощью можно удобно администрировать сайт [1].

И это только часть сервисов и систем, которые компании имеют возможность использовать для автоматизации бизнес-процессов.

Как правило, в крупных компаниях используется несколько систем, поскольку каждая решает определенные задачи.

Организации мелкого формата, как правило, лишены возможности использовать несколько систем для автоматизации бизнес-процессов. Наиболее частый вариант – это использование обычных таблиц Excel, либо использование WEB – приложений.

На сегодняшний день неотъемлемой частью всех информационно-техногенных цивилизаций является автоматизация управления информационными потоками. Связано это, прежде всего, с ростом объема оперируемой человеком информацией и потребностью в высокой скорости ее обработки. В связи с этим актуальна разработка различных приложений, программ для переноса информации с бумажных носителей на цифровые.

Основная задача разрабатываемого приложения заключается в предоставлении пользователю интерфейса для возможности быстрого и удобного выполнения поставленных ему задач. Целью данного дипломного проекта является обеспечение реализующее ведение учета учащихся детского образовательного центра, позволяющее вносить в базу как клиентов-родителей, так и детей, на примере конкретной компании позволяющего облегчить работу по учету клиентов предприятия.

В рамках проекта необходимо выполнить разработку, отладку и тестирование программного средства для учета персонифицированных данных.

# Аналитический обзор аналогов. Постановка задачи на дипломное проектирование

## Аналитический обзор существующих аналогов

ООО «Школа цифровых технологий» является организацией, предоставляющей возможность учащимся приобретать новые навыки в различных сферах: робототехника, программирование, научно-техническое творчество.

Проведем обзор програмных средств по учету и хранению персонифицированных данных. Данные программные средства широко представленны на рынке.

**«Талланто»** – система учета и хранения персонифицированных данных, которая используется для обслуживания школ и кружков, центров деского развития, робототехники и.т.д. Сайт системы: https://tallanto.com.

Данная система позволяет решать следующие задачи:

– обработка и ввод первичной информации о предоставленных услугах и их оплатах;

– учет клиентов образовательных центров;

– регистрация и контроль платежей;

– ведение справочной информации по услугам и ценам.

– расчет платежей по предоставленным услугам;

– формирование списков учащихся и преподавателей;

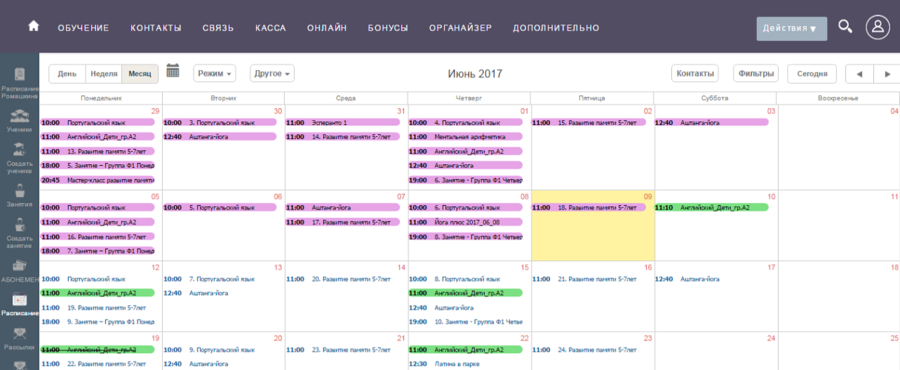
– формирование документов статистической отчетности и информационно - аналитических документов по оказанным услугам, категориям абонентов и прочее.

– создание таблиц, списков учащихся и преподавателей;

– учет посещения занятий;

– визуализация посещения занятий в виде графиков, диаграмм;

– формирование документов статистической отчетности и информационно - аналитических документов по оказанным услугам, категориям абонентов и прочее.



|  |
| --- |
|  |

Рисунок 1.1– Окно программного средства «Талланто»

В настоящее время преподаватели и менеджеры образовательных центров и сотрудники отдела кадров компании ООО «Школа цифровых технологий» для своевременного и корректного выполнения поставленных задач используют несколько внедренных информационных систем.

**«Учет учебного центра»** – система учета для детских центров. Учет клиентов и посещений. Планирование задач и рабочей нагрузки. Учет посещаемости, удобное добавление и удаление учеников, а также Email и SMS рассылки. Сайт системы: http://usu.by/uchet\_uchebnogo\_tsentra.php.

Данная система позволяет решать следующие задачи:

– создание таблиц, списков учащихся и преподавателей;

– учет посещения занятий;

– визуализация посещения занятий в виде графиков, диаграмм;

– учет клиентов образовательных центров;

– ведение справочной информации по услугам и ценам.

– расчет платежей по предоставленным услугам;

– формирование списков учащихся и преподавателей;

– формирование документов статистической отчетности и информационно - аналитических документов по оказанным услугам, категориям абонентов и прочее.

|  |
| --- |
|  |

Рисунок 1.2– Окно программного средства «Учет учебного центра»

**«BASEPLAN»** – разработана специально для автоматизации рабочих процессов в учебных заведениях и тренинговых компаниях. Система позволяет решать задачи формирования, а также управления открытыми и закрытыми мероприятиями. Кроме того, вести учет наполнения групп и прочей отчетности, управлять мероприятием при помощи расписания, прикреплять группы к преподавателям, формировать расписание занятости помещений. Сайт системы: https://baseplan.ru.

Данная система позволяет решать следующие задачи:

– ведение общей базы и импорт существующей, используя CSV-формат;

– инструменты ведения учета посетителей, формирования учебного расписания использования помещений, просмотра отчетов по доходам и зарплатам;

– разграничение доступа к системе для разных сотрудников;

– учет клиентов образовательных центров;

– ведение справочной информации по услугам и ценам.

– расчет платежей по предоставленным услугам;

– формирование списков учащихся и преподавателей;

– формирование документов статистической отчетности и информационно - аналитических документов по оказанным услугам, категориям абонентов и прочее.

|  |
| --- |
|  |

Рисунок 1.3– Окно программного средства «Учет учебного центра»

В настоящее время преподаватели и менеджеры образовательных центров и сотрудники отдела кадров компании ООО «Школа цифровых технологий» для своевременного и корректного выполнения поставленных задач используют несколько внедренных информационных систем.

***«******ALFACRM»*** – комплексная система учета и хранения персонифицированных данных, которая используется для обслуживания школ и кружков, центров деского развития, робототехники и.т.д. Это современная, простая и удобная система для управления отношениями с клиентами. Обеспечивает ведение базы данных и учет клиентов (организаций и лиц), выставленных им счетов, заключенных с ними договоров (сделок), других документов, продуктов (товаров и услуг), сотрудников. Программа предназначена для автоматизации деятельности отдела продаж, финансового отдела и других задач, которые можно гибко настроить. Сайт системы: https://alfacrm.pro/.

Данная система позволяет решать следующие задачи:

– обработка и ввод первичной информации о предоставленных услугах и их оплатах;

– учет клиентов образовательных центров;

– регистрация и контроль платежей;

– ведение справочной информации по услугам и ценам.

– расчет платежей по предоставленным услугам;

– формирование списков учащихся и преподавателей;

– формирование документов статистической отчетности и информационно - аналитических документов по оказанным услугам, категориям абонентов и прочее.

|  |
| --- |
|  |

Рисунок 1.4– Окно «ALFACRM»

***«MICROSOFT EXCEL»*** — программа для работы с электронными таблицами, созданная корпорацией Microsoft для Microsoft Windows, Windows NT и Mac OS, а также Android, iOS и Windows Phone. Сайт программы: https://www.microsoft.com/ru-ru/microsoft-365/excel. Это программный продукт, который относится к категории электронных таблиц, которые предназначены для выполнения разнообразнейших вычислений. Основное рабочее поле электронной таблицы состоит из набора строк и столбцов. Они отображаются на экране в специальном окне, которое можно прокручивать вверх и вниз, вправо и влево. Область, находящаяся на пересечении строки и столбца, называется ячейкой. В ячейке может находится число, текст и формула. С помощью формул можно выполнить вычисления, в которых используются значения, содержащиеся в других ячейках. Можно копировать и перемещать ячейки, а также изменять содержащиеся в них формулы. Большинство специалистов, которые используют в своей работе Excel, задействуют крайне ограниченное количество возможностей этой программы.

Среди основных задач, которые позволяет решать данное приложение, можно выделить следующие:

– создание таблиц, списков учащихся и преподавателей;

– учет посещения занятий;

– визуализация посещения занятий в виде графиков, диаграмм;

– учет клиентов образовательных центров;

– формирование документов статистической отчетности и информационно - аналитических документов по оказанным услугам, категориям абонентов и прочее.

|  |
| --- |
|  |

Рисунок 1.5 – Окно «EXCEL»

В рамках данного дипломного проекта предполагается разработка программного средства, которое позволит отказаться от «ALFACRM» и «EXCEL». Взаимодействие Пользователя (Преподаватель), клиента (родителя) и клиентов (детей) с учетом разрабатываемого программного средства представлено на рисунке 1.6.

|  |
| --- |
|  |

Рисунок 1.6 – Взаимодействие Пользователя (Преподаватель), клиента (родителя) и клиентов (детей) с учетом разрабатываемого программного средства

Программное средство решает следующие задачи:

– добавление клиента с контактными и анкетными данными;

– коррекция ранее внесенного клиента;

– удаление ранее внесенного клиента;

– добавление ребенка к клиенту с документами и анкетными данными;

– коррекция ранее внесенного ребенка;

– присвоение клиенту одной или нескольких категорий.

– ведение справочника категорий.

– ведение справочника городов.

– формирование списков по заданным правилам, с возможностью расширения списков без перекомпиляции программы.

– поиск в базе данных любых сведений по части текста.

– установка парольной защиты на вход в программу.

Сравнение аналогов представлено на рисунке 1.7. Метод сравнения аналогов предполагает сбор и анализ показателей описываемых систем, осуществляющих аналогичный вид деятельности.

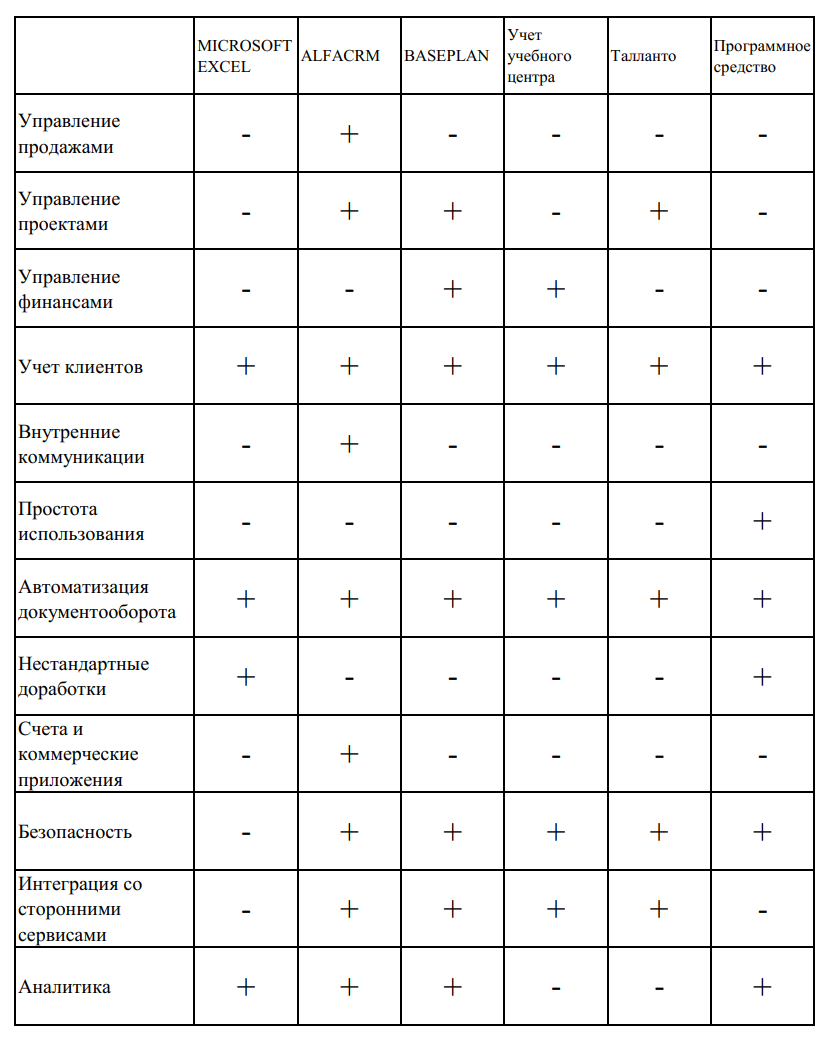


Рисунок 1.7 – Сравнение аналогов

## Постановка цели и задач на дипломное проектирование

В рамках компании ООО «Школа цифровых технологий» было принято решение разработать программное средство реализующее ведение учета учащихся детского образовательного центра, позволяющее вносить в базу как клиентов-родителей, так и детей.

Интерфейс программы и создание списков реализовать через QtScript, для работы и изменения этих функций без перекомпиляции проекта.

Проект выполнить как приложение, хранящее данные в базе SQLite, загружающее их в формате JSON и выводящее в интерфейс, а также позволяющее редактировать, выполнять поиск и формировать списки по имеющимся данным

Сложные объекты хранить как JSON-объекты в виде больших текстовых полей, чтобы упростить сопровождение базы данных, при изменении структуры классов, не нужно было модифицировать структуру таблиц.

Список, порядок и представление данных в интерфейсе задать не кодом на C++, а специальными JSON-объектами — представлениями. Это позволяет модифицировать интерфейс приложения и менять логику представления данных без перекомпиляции самого приложения.

Предусмотреть возможножность расширения функционала программы также за счет создания списков, которые получаются из основной базы путем фильтрации, сортировки и выбора конкретных полей с преобразованием данных. Все эти функцию работают без перекомпиляции проекта, за счет расширения через представления списков и написания функций на ECMAScript.

Основными задачами, решаемыми в дипломном проекте, являются:

– провести функциональный анализ предметной области;

– разработать функциональные требования;

– описать модели и методы, используемые в программном средстве;

– спроектировать программное средство;

– спроектировать и реализовать алгоритмы программного средства на языке С++ с использованием фрэймворка QT;

– протестировать разработанное программное средство;

– подготовить графическую и текстовую части дипломного проекта.

# Методы и модели, положенные в основу проекта

## Описание предметной области

Программные средства по учету персонифицированных данных – это средства для хранения данных и предоставления правильных ответов на вопросы пользователя с целью визуализации и упрощения получения информации. Возможности менеджера включают в себя организационные моменты, по управлению клиентами для данного предприятия исходя из функций, которые ему предоставляет работодатель.

Работа с клиентами организована следующим образом: у каждого клиента, пришедшего в школц, менеджер запрашивает стандартные данные: фамилия, имя, отчество, дата рождения, адрес места жительства. После этого менеджер выясняет у клиента, в какой кружок он хочет пойти. Если в кружке есть свободные места, то ученика записывают в этот кружок.

Информация обычно хранятся в базе данных, особенность программ для хранения персонифицированных данных – невысокая требовательность к личным данным пользователя и несложная объектная модель построения связей.

Персонифицированные данные – представление, которое может содержать дату рождения, электронный почтовый ящик, сведения о месте жительсва.

Многие программы для хранения данных позволяют просматривать информацию и в случае необходимости визуализировать информацию с помощью диаграмм.

Категория запросов пользователя – сущность, которая формирует логичный набор вопросов определенного требования.

Запросы обычно предполагают единственный верный вариант, несколько верных вариантов, а также указание изображения в качестве верного варианта. Представляют собой сущность, содержащую сам вопрос, верные ответы, категорию, и варианты ответов.

Одной из важнейших проблем, существующих на фирмах, является автоматизация деятельности.

В современных условиях работы частных фирм актуальными становиться вопросы быстрое и качественное обслуживание персонала. Автоматизация работы по регистрации учащихся значительно ускоряет процесс регистрации клиентов.

Конечно, это не решит полностью проблем с задержкой по выдаче заявлений и до конца не освободит работников от проведения одинаковых повторяющихся операций. Но система позволит упорядочить хотя бы процесс выдачи заявлений и позволит избавится от рукописных ошибок, описок, неясностей в тексте и в цифрах.

В рамках этого процесса оптимизации работы регистрации клиентов возможно в холле фирмы установить электронный терминал, который помогает автоматизировать регистрацию клиентов

Но, к сожалению, внедрение такой системы не всегда возможно. Во-первых, это связанно с затратами на сам терминал и специализированное программное обеспечение. Лицензионное программное обеспечение довольно-таки дорогое удовольствие, и не каждая организация может себе его позволить, а тем более начинающая.

Во- вторых ограничением является и то, что многие люди, не умеют пользоваться или плохо владеют компьютером, и для освоение сложных и громоздких программ нужно большое количество времени и ресурсов.

В связи с необходимостью автоматизации данной области и приведенными выше ограничениями, оптимальным является автоматизация с помощью программных средств.

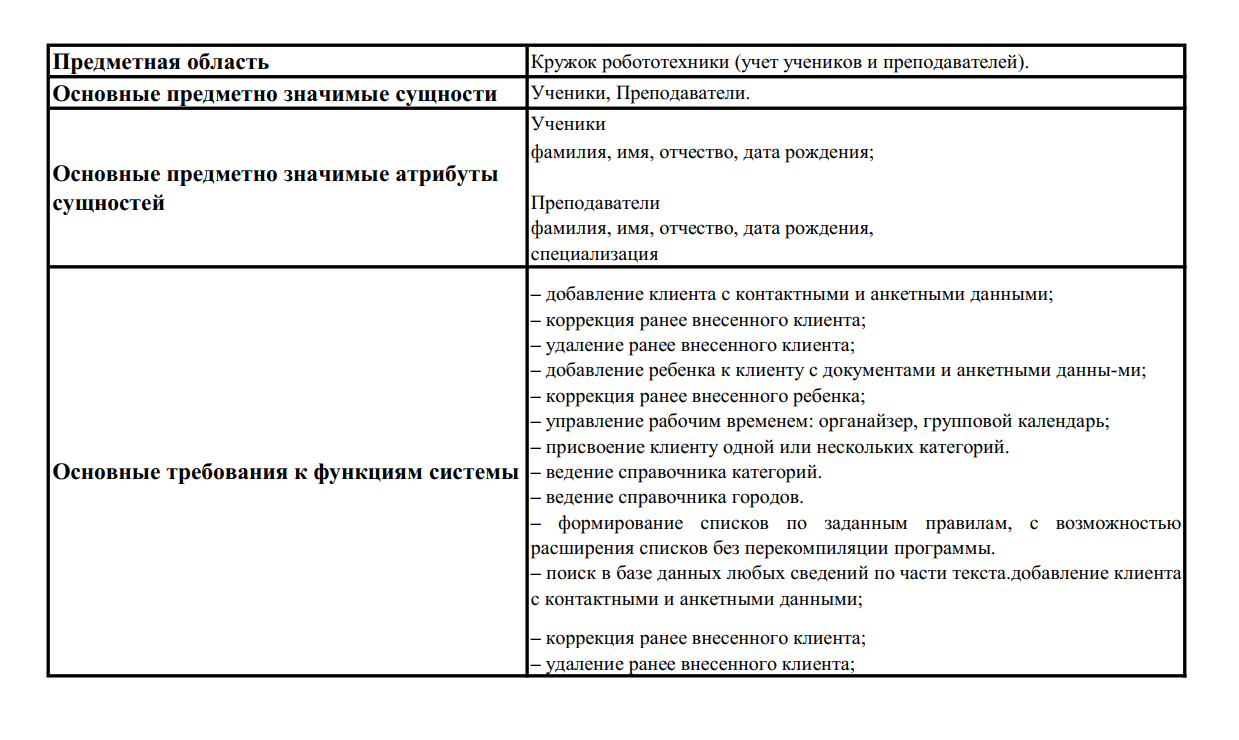
 На основании проведенного бизнес-анализа были выделены сущности предметной области (Рисунок 2.1).

Рисунок 2.1 – Сущности предметной области (без связей)

Разрабатываемое приложение создается с целью предоставления пользователям системы наиболее быстрого и легкого получения информации.

Вход в программу пользователя является одной из первых функций, с которыми встречается пользователь, для начала работы с программой. Страница ввода логина и пароля пользователя доступна сразу после запуска приложения.

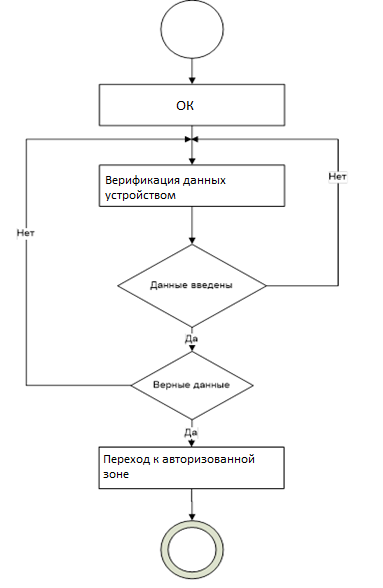
После верификации пользовательских данных пользователь будет перенаправлен на главную часть приложения предназначеную для авторизованных пользователей. Диаграмма логина пользователя представлена на рисунке 2.2.

Рисунок 2.2 – Процесс логина пользователя

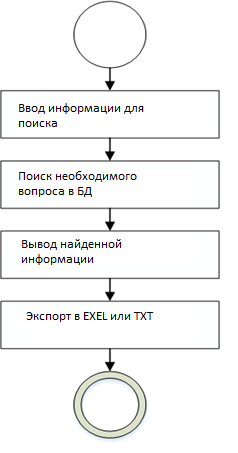
Функция поиска по базе данных доступна любому авторизованному пользователю. Для этого необходимо выбрать в главном меню кнопку поиска. В окне необходимо ввести любое слово и воспользоваться поиском для подбора ответов в форме списка. Диаграмма деятельности по поиску и ответу на вопрос представлена на рисунке 2.3.

Рисунок 2.3 – Процесс перехода на вопрос

# Проектирование приложения

## Функциональный анализ предметной области.

Матрица трассировки — метод визуализации связей между элементами системы в форме таблицы. Матрица трассировки создается путем связывания бизнес-требований с вариантами использования и сценариями тестирования, которые будут использоваться для их проверки. Трассировочная таблица приведена в таблице 3.1.

Таблица 3.1. Трассировочная матрица

|  |  |  |  |
| --- | --- | --- | --- |
| № | Пожелания заказчика  (Stakeholder Needs, User Requirements) | № функции | Функциональное требование (Features, Functional Requirements) |
| 1 | Вход в систему | 1.1  1.2  1.3 | Ввод имени пользователя и пароля  Аутентификация  Оповещения о корректности данных |
| 2 | Просмотр списка семей и его редактирование | 2.1  2.2  2.3  2.4 | Ввод данных  Изменение данных  Сохранение данных  Оповещения о корректности данных |
| 3 | Просмотр списка учеников и его редактирование | 3.1  3.2 | Ввод данных  Изменение данных  Сохранение данных  Оповещения о корректности данных |
| 4 | Настройки пользователя | 4.1  4.2  4.3  4.4  4.5 | Регистрация пользователя  Изменение пароля пользователя  Добавление категории  Изменение категории  Оповещения о корректности данных |

Модель вариантов использования пользователей изображена на рисунке 3.1.

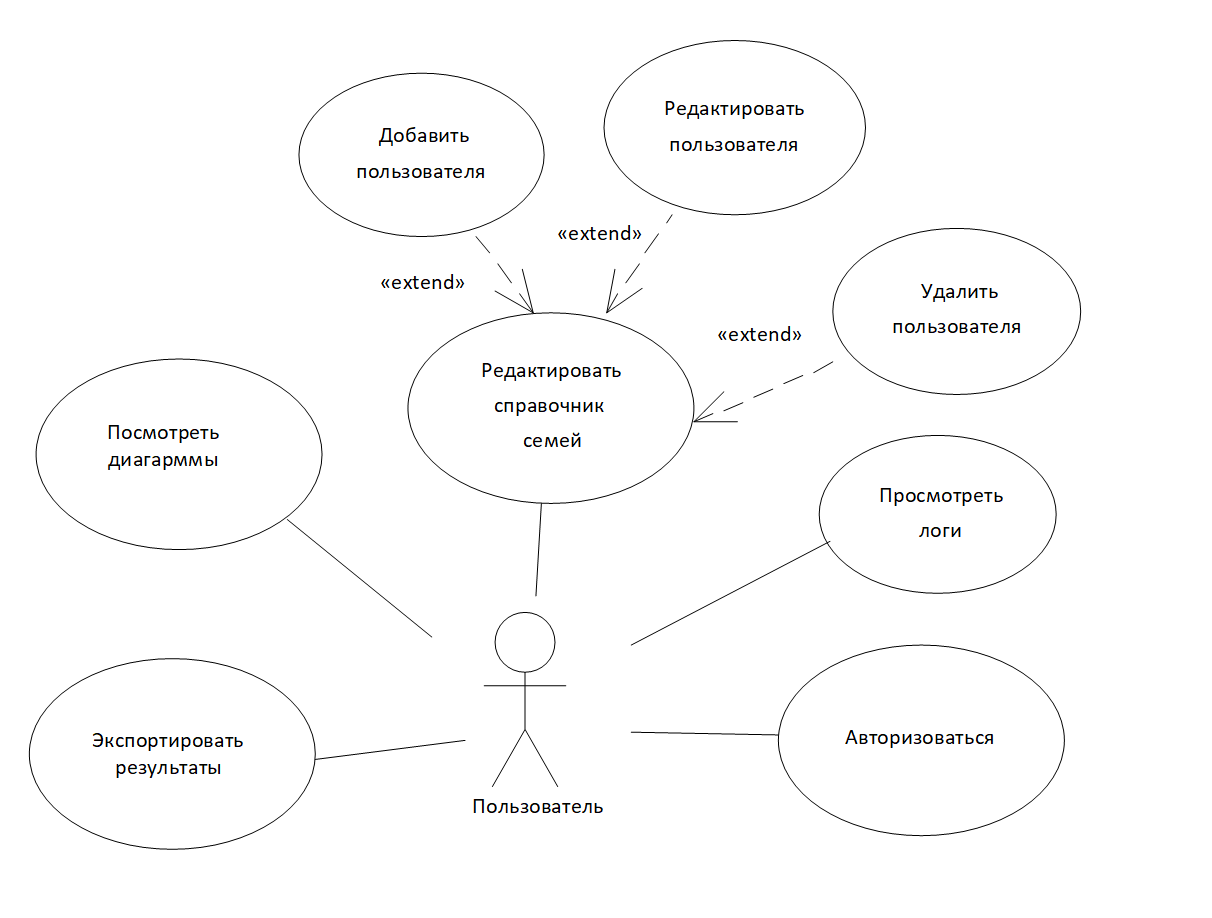


Рисунок 3.1 Модель вариантов использования пользователя

## Разработка структуры программного средства.

Проект устроен как приложение, хранящее данные в базе SQLite, загружающее их в формате JSON и выводящее в интерфейс, а также позволяющее редактировать, выполнять поиск и формировать списки по имеющимся данным.

В базе данных минимум полей — сложные объекты хранятся как JSON-объекты в виде больших текстовых полей. В виде полей оставлены только первичные и внешние ключи, чтобы можно было быстро искать и удалять записи. Такой подход упрощает сопровождение базы данных, при изменении структуры классов, не нужно модифицировать структуру таблиц.

Список, порядок и представление данных в интерфейсе задано не кодом на C++, а специальными JSON-объектами — представлениями. В представлении таблицы указывается список колонок, функция сортировки и список полей, которые могут быть как прямым полем из JSON-объекта, так и вычисляемыми функциями на ECMAScript. Это позволяет модифицировать интерфейс приложения и менять логику представления данных без перекомпиляции самого приложения. В представлении формы ввода указывается список полей JSON-объекта, подлежащих коррекции, заголовок для пользователя к каждому полю, а также тип поля — текст, большой текст или выпадающий список (для последнего можно указать также источник записей). Аналогично, это позволяет изменять редактирование данных без перекомпиляции самого приложения.

Программа может быть расширена также за счет создания списков, которые получаются из основной базы путем фильтрации, сортировки и выбора конкретных полей с преобразованием данных. Например, можно выбрать всех клиентов, у которых не менее заданного числа детей, и отсортировать их по числу детей и по фамилии, а также включить в выходные данные возраст детей, вычисленный из даты рождения. Все эти функцию работают без перекомпиляции проекта, за счет расширения через представления списков и написания функций на ECMAScript. При создании списка указывается правило фильтрации, правило сортировки и список выходных данных.

Представления – это специальные JSON объекты, формат данных которых - ключ-значение, который обычно записывается в фигурных скобках.

Список представлений для таблиц:

view\_children.json — представление для просмотра списка детей клиента.

view\_client.json — представление для просмотра списка клиентов.

view\_search.json — представление для просмотра результатов поиска.

Список представлений для редактора:

editor\_child.json — форма редактирования для ребенка.

editor\_client.json — форма редактирования для клиента.

Список представлений для списков (может быть расширен пользователем):

list\_fiophonechildcount\_bychildcount.json — список клиентов по числу детей.

list\_fiophonechildcount\_cats.json — список клиентов по категориям - ФИО, телефон, число детей.

list\_fiophonechildcount\_city.json — список клиентов по городу — ФИО, телефон, число детей.

Для реализации данной предметной области используем фреймворк Qt 5.14 с языком программирования С++. Qt5 подразумевает двойное лицензирование, а это означает, что он может быть использован в некоммерческих целях для создания приложений с открытым исходным кодом, а также как лицензия для коммерческих клиентов. В дипломном проекте будет использованна некоммерческая лицензия.

В качестве базы данных используем SQLite. SQLite — это внутрипроцессная библиотека, которая реализует автономный, безсерверный, транзакционный механизм базы данных SQL с нулевой конфигурацией. Это база данных с нулевой конфигурацией, что означает, что, как и другие базы данных, вам не нужно настраивать ее в вашей системе.

Графический интерфейс проекта реализован на QtWidgets. QtWidgets содержит классы для классических приложений на основе виджетов, это это базовый класс для всех GUI.

Для представления данных в базе используется формат текстовых данных JSON.

Для преобразования данных, получения списков и управления отчетами без перекомпиляции проекта применяется язык ECMAScript (базовый вариант JavaScript) и компонент QtScript. ECMAScript — это встраиваемый расширяемый не имеющий средств ввода-вывода язык программирования, используемый в качестве основы для построения других скриптовых языков. Стандартизирован международной организацией ECMA в спецификации ECMA-262.

Для реализации перечисленных функциональных моделей будем использовать перечисленные модули.

QtCore – базовый модуль фрэймворка. В нем представлены основные не графические классы: система сигналов и слотов, потоков, разделяемой памяти, регулярных выражений и т. д.

QtGui – модуль дополняет модуль QtCore функциональностью GUI. Содержит компоненты графического интерфейса (элементы управления), основанные на визуальном представлении.

QtScript – Модуль QtScript предоставляет классы для придания приложениям Qt возможности выполнять скрипты.

QtSql – Модуль QtSql помогает обеспечить однородную интеграцию БД в ваши Qt приложения.

## Разработка структуры классов при использовании объектно-ориентированного стиля программирования приложений.

Проект состоит из каталога бинарных файлов и каталога данных.

В каталоге бинарных данных находится исполнимый файл проекта, библиотеки C++ и модули Qt, а также плагины для доступа к базе.

В каталоге данных находится база SQLite, представления просмотра, редактора и списков в формате JSON.

Также в каталоге данных находятся еще два файла style.css и userfunc.js. Первый отвечает за визуальное оформление интерфейса (стилизацию), второй содержит функции на ECMAScript, выполняющие обработку данных для представлений — сортировку, фильтрацию, форматирование, получение списков. Например функция client\_CalcMainFIO(obj) выполняет получение ФИО клиента из отдельных полей, а функция childcountandfamily\_lessThen(left,right) производит сортировку клиентов по числу детей и фамилии.

Классы в программе разбиты по группами:

Классы форм — отвечают за работу с интерфейсом.

Классы модели — реализуют систему «модель-представление-контроллер», где в роли модели выступают данные в формате JSON.

Служебные классы для выполнения поиска по данным, обработки скриптов, построения форм и диалогов правки данных.

Классы форм:

MainWindow – главное окно приложения, содержит таблицы представлений и кнопки управления.

DialogRefEdit – диалог редактирования справочников

DialogViewList – диалог просмотра списков

DialogEditObject – диалог редактирования объекта в формате JSON, с автоматическим построением интерфейса и полей ввода

DialogSelList – диалог выбора значения из списка JSON, позволяет как выбирать одно значение, так и работать в качестве списка на выбор нескольких значений «галочками».

Классы модели

QJsonTableModel — расширяет класс QabstractTableModel, реализует модель для представления массива JSON в табличном виде, использует указанное представление и ECMAScript для получения табличного вида.

QJsonSortProxyModel — расширяет класс QsortFilterProxyModel, позволяет дополнительно фильтровать и сортировать модель QjsonTableModel на основе переданных параметров фильтрации и сортировки, как функций ECMAScript

Служебные классы

QjsonSearchInObject — класс для поиска значений в JSON-массиве, возвращает указания, где и что найдено.

FormBuilder – построитель простых диалогов, позволяет избежать создания в конструкторе простых диалогов вроде авторизации или подтверждения действий.

Phone – класс телефона, содержит цифры и методы форматирования

PhoneTools – класс для обработки телефона, проверка корректности и форматирования

Helper – вспомогательный класс для разных операций, например, чтения файла

MainEngine – класс-переходник для работы с компонентом QtScript, расширяет функции ECMAScript новыми методами, например, для работы с датами.

## Разработка структуры базы данных.

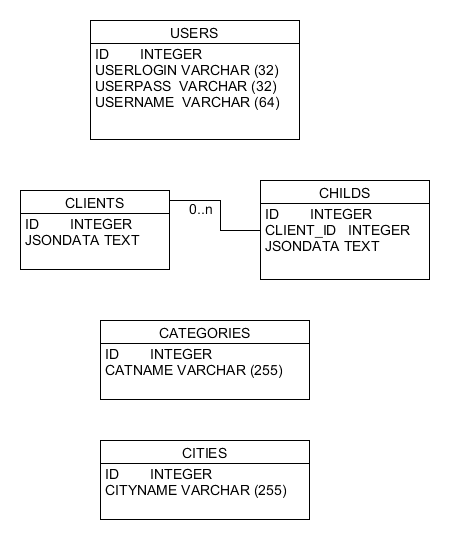
|  |
| --- |
|  |

Рисунок 3.2 – Визуальное представление базы данных

В качестве базы данных используем SQLite. SQLite — это внутрипроцессная библиотека, которая реализует автономный, безсерверный, транзакционный механизм базы данных SQL с нулевой конфигурацией. Это база данных с нулевой конфигурацией, что означает, что, как и другие базы данных, вам не нужно настраивать ее в вашей системе.

В базе данных минимум полей — сложные объекты хранятся как JSON-объекты в виде больших текстовых полей. В виде полей оставлены только первичные и внешние ключи, чтобы можно было быстро искать и удалять записи. Такой подход упрощает сопровождение базы данных, при изменении структуры классов, не нужно модифицировать структуру таблиц.

В каталоге данных находится база SQLite, представления просмотра, редактора и списков в формате JSON.

Рисунок .3 – Схема базы данных с имеющимися связями:

USERS – список пользователей, имеет поля для логина, имени пользователя и его пароля.

CLIENTS – список клиентов, данные клиента хранятся в виде JSON в поле JSONDATA

CHILDS – список детей, данные хранятся в виде JSON в поле JSONDATA. Поле CLIENT\_ID указывает на первичный ключ клиента, к которому относится данный ребенок.

CATEGORIES – справочник категорий, название категории записано в поле CATNAME

CITIES – справочник городов, название города записано в поле CITYNAME

## 3.5 Реализация подключения базы данных в программу на Qt

Для реализации связи объектов и базы используем компоненты QtSQL.

Подключаем библиотеки

#include "QtSql/QSqlDatabase"

#include "QsqlQuery"

В главной форме добавляем объекты базы данных.

QSqlDatabase db

При создании формы программы, пытаемся подключиться к базе данных, указав тип базы, имя базы и проверив результат.

db = QSqlDatabase::addDatabase("QSQLITE");

db.setDatabaseName("childcenter.db");

if (!db.open())

QMessageBox::information(this,"Ошибка открытия базы данных: ",db.lastError().text()) ;

if (!db.tables().contains("USERS")) {

}

Далее, для операций выборки данных из базы, применяем SQL-запрос SELECT, а для изменения данных — запросы INSERT, UPDATE, DELETE.

Для выполнения запросов нужно использовать объект QsqlQuery. Есть два способа сформировать запрос — без параметров и с параметрами. Для запроса без параметров передаем в объект при создании текст запроса, вызываем метод exec() и метод next() перебираем все записи запроса до конца.

Процедура выборки клиентов из таблицы.

QSqlQuery query("SELECT id, jsondata FROM clients") ;

query.exec() ;

while (query.next()) {

...

}

Запрос с параметрами позволяет указать какие-то данные, которые будут отправлены в базу данных. Например, процесс обновления записи ребенка в базе с параметрами первичного ключа и данных JSON – готовим запрос методом prepare, потом вызовами метода bind заполняем параметры, и только потом вызываем метод exec.

QSqlQuery query ;

query.prepare("UPDATE CHILDS SET JSONDATA=:jsondata WHERE id=:id") ;

query.bindValue(":id",id) ;

query.bindValue(":jsondata",QJsonDocument(obj).toJson()) ;

query.exec() ;

Для получения данных из запроса можно использовать метод объекта value, по номеру поля или имени поля. Так, для получения значения первого поля в виде целого числа, используем вызовами.

query.value(0).toInt()

Для получения поля jsondata в виде строки, можно использовать вызов

query.value("jsondata").toString()

# Разработка алгоритмов и их программная реализация

## Алгоритм входа в систему

Для входа в систему требуется ввести имя пользователя и пароль. Если данные корректно введены, то система ищет в базе данных пользователя с введенным email и паролем. Если в системе нет такого пользователя, система сообщает об этом пользователю.

Процесс авторизации реализован с помощью функции authorizeUser() соединенной с сигналом кнопки авторизации. Функция сохраняет пользовательский ввод, составляет из него запрос к БД, проверяет хранимый пароль на соответствие тому, который ввел пользователь и, в случае успеха, открывает MainWindow с параметрами, сохраненными из предыдущей сесии. Класс авторизации будет хранить внутри указатель на пользователей (объект Users). Этот объект должен создаваться где-то снаружи, например в функции main и передаваться всем классам, работающим с пользователями (виджету статистики, например). В случае успешной авторизации класс вырабатывает сигнал auth\_ok, на который должны быть подписаны формы приложения, в которые должен переходить пользователь при успешной авторизации. Схема алгоритма входа в систему представлена на чертеже ГУИР 903321.003.

## Алгоритм регистрации нового пользователя

Процесс авторизации реализован с помощью функции registerUser() соединенной с сигналом кнопки регистрации, расположенной в окне регистрации. Она выполняет проверку пользовательского ввода, получает новый номер записи из БД и регистрирует пользователя. Библиотека QtSql обеспечивает нам доступ к сообщениям с ошибками, которые формируются запросами QSqlQuery. Схема алгоритма входа в систему представлена на чертеже ГУИР 903321.004.

## Алгоритм экспорта списков

Электронные таблицы Microsoft Excel - очень мощное средство создания и ведения различных электронных документов. На сегодняшний день много полезных данных в компаниях продолжает хранится в Excel. При экспорте данных в Excel приложение создает копию списка, а затем сохраняет их в файле, который затем открывает в Excel. Схема алгоритма входа в систему представлена на чертеже ГУИР 903321.006.

## Алгоритм сохранения клиента в БД

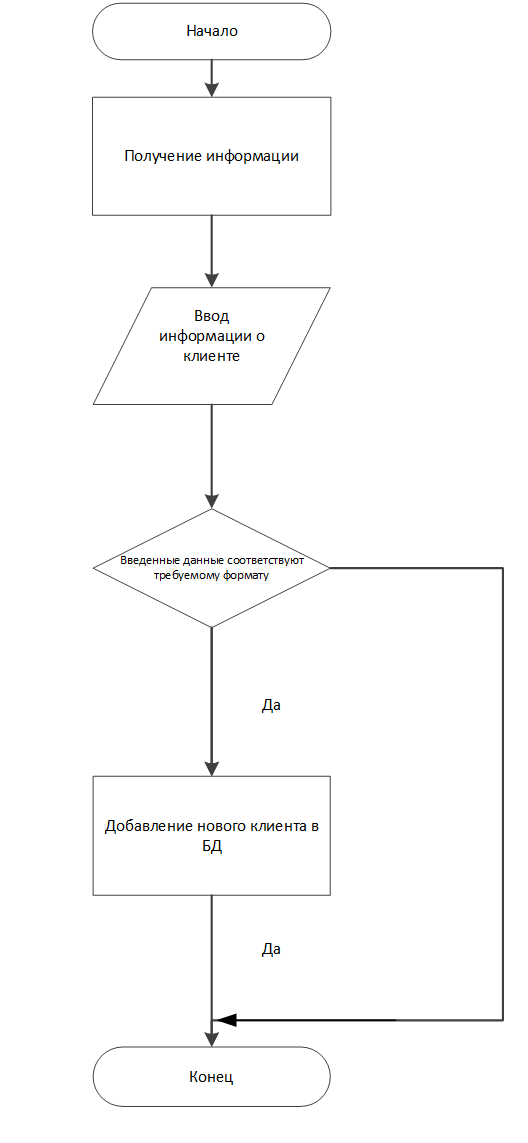
Алгоритм сохранения клиента приведен на рисунке 4.3.

Рисунок 4.1 – Алгоритм сохранения клиента

## Алгоритм удаления клиента в БД

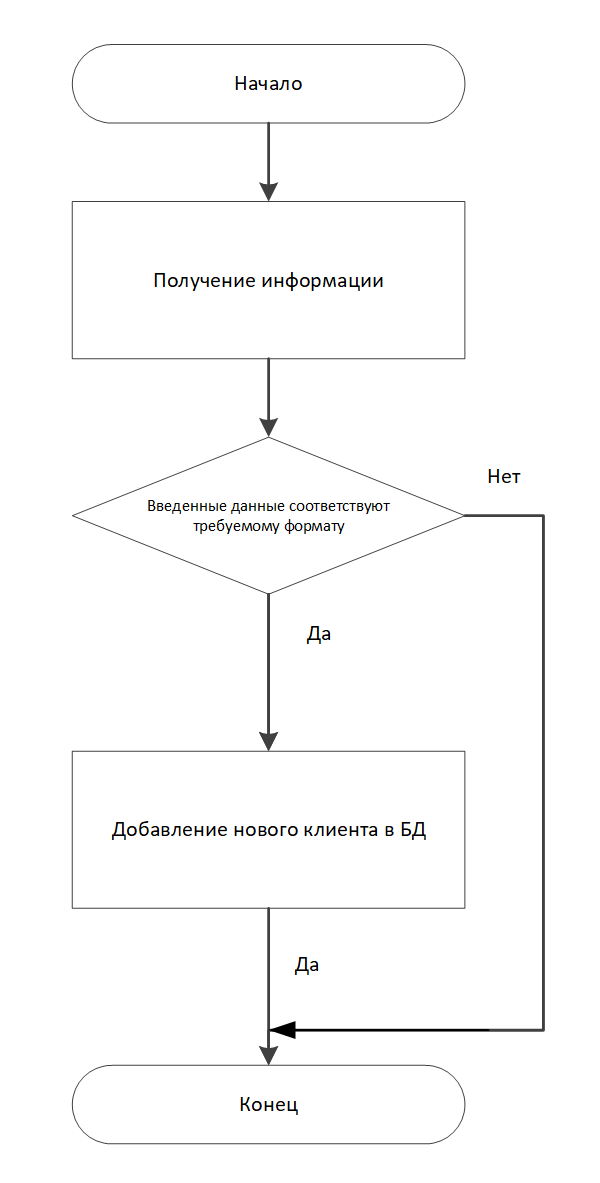
Алгоритм удаления клиента приведен на рисунке 4.4.

Рисунок 4.2 – Алгоритм удаления клиента

# Тестирование разработанного программного средства

Для оценки оценки разработанного программного средства было выбрано и проведено функциональное тестирование, поскольку данный вид тестирования позволил проверить соответствие разработанного программного средства требованиям задания. Оценка качества с точки зрения поведения системы, её быстродействия, удобства использования и т.д., а также поиск путей повышения её показателей тематикой данного проекта не является.

Для того, чтобы провести тестирования использовались Test Cases. Test Cases представляют собой перечень функций программного средства, а также конкретные проверки для них. Каждая проверка включает в себя набор входных значений, предустановленных условий, выполняемых шагов и ожидаемых результатов от их выполнения.

В ходе проведенного тестирования было проверено выполнение программным средством следующих функций:

– поиск пользователя;

– поиск клиента по числу детей;

– поиск клиента по категории «многодетная семья»;

– поиск клиента по «городу»;

– поиск клиента по категории «неполная семья»;

– поиск клиента по части фамилии;

– сохранение в БД записи «Клиент»;

– сохранение в БД записи «Город»;

– сохранение в БД записи «Категория»;

– экспорт списка в .xlsx;

– экспорт списка в .txt;

– удаление клиента из БД;

Выполненный набор Test Cases представлен в таблице 5.1.Таблица 5.1 – Выполненные тесты программного средства (Test Cases)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Модуль** | **Тест кейс описание** | **Ожидаемый результат** | **Комментарии** |
| 1 | QtCore, QtSql, QtScript | **Поиск пользователя:**  1.Считать параметр User\_Login;  2.Осуществить поиск по параметру в БД. | Происходит поиск пользователя, работа приложения продолжается. В случае отсутствия записи, работа приложения прекращается. | Выполнено  успешно |
| 2 | QtCore, QtSql, QtScript | **Поиск клиента по числу детей:**  1. Применяем SQL-запрос SELECT;  2.Осуществить поиск по параметру в БД. | 1 Происходит поиск клиента.  2. Клиент найден, формируется список, работа приложения продолжается.  3.Клиент не найден, сообщение, что результаты не найдены, работа приложения продолжается. | Выполнено  успешно |
| 3 | QtCore, QtSql, QtScript | **Поиск клиента по категории «многодетная семья»;**  1. Применяем SQL-запрос SELECT;  2.Осуществить поиск по параметру в БД. | 1 Происходит поиск клиента.  2. Клиент найден, формируется список, работа приложения продолжается.  3.Клиент не найден, сообщение, что результаты не найдены, работа приложения продолжается. | Выполнено  успешно |
| 4 | QtCore, QtSql, QtScript | **Поиск клиента по «городу»:**  1. Применяем SQL-запрос SELECT;  2.Осуществить поиск по параметру в БД. | 1 Происходит поиск клиента.  2. Клиент найден, формируется список, работа приложения продолжается.  3.Клиент не найден, сообщение, что результаты не найдены, работа приложения продолжается. | Выполнено  успешно |
| 5 | QtCore, QtSql, QtScript | **Поиск клиента по категории «неполная семья»:**  1. Применяем SQL-запрос SELECT;  2.Осуществить поиск по парамет-ру в БД. | 1 Происходит поиск клиента.  2. Клиент найден, формируется список, работа приложения продолжается.  3.Клиент не найден, сообщение, что результаты не найдены, работа приложения продолжается. | Выполнено  успешно |
| 6 | QtCore, QtSql, QtScript | **Поиск клиента по части фамилии**  Предварительные условия:  Заполнение поля «Поиск».  1. Применяем SQL-запрос SELECT.  2. Осуществить поиск в БД. | 1 Происходит поиск клиента.  2. Клиент найден, формируется список, работа приложения продолжается.  3.Клиент не найден, сообщение, что результаты не найдены, работа приложения продолжается. | Выполнено  успешно |
| 7 | QtCore, QtSql, QtScript | **Cохранение в БД записи «Клиент»:**  Предварительные условия:  Заполнение параметров клиента в окне «Клиент».  1. Применяем SQL-запрос INSERT; | 1 Происходит cохранение клиента в БД.  2.Работа приложения продолжается. | Выполнено  успешно |

Продолжение таблицы 5.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 8 | QtCore, QtSql, QtScript | **Cохранение в БД записи «Город»:**  Предварительные условия:  Заполнение названия города в окне «Список городов».  1. Применяем SQL-запрос INSERT; | 1 Происходит cохранение названия города в БД.  2.Работа приложения продолжается. | Выполнено  успешно |
| 9 | QtCore, QtSql, QtScript | **Cохранение в БД записи «Категория»:** Предварительные условия:  Заполнение названия категории в окне «Список категорий».  1. Применяем SQL-запрос INSERT; | 1 Происходит cохранение названия категории в БД.  2.Работа приложения продолжается. | Выполнено  успешно |
| 10 | QtCore, QtSql, QtScript | **Экспорт списка в .xlsx**  Предварительные условия:  Список сформирован, выбрана кодировка.  1.Применяем класс ExcelExportHelper  2. Экспорт проведен. | 1 Происходит сохранение списка в формате .xlsx.  2. Происходит открытие сохраненного файла.  3.Документ открывается, работа приложения продолжается. | Выполнено  успешно |
| 11 | QtCore, QtSql, QtScript | **Экспорт списка в .txt**  Предварительные условия:  Список сформирован, выбрана кодировка.  1.Применяем класс TxtExportHelper  2. Экспорт проведен. | 1 Происходит сохранение списка в формате .txt.  2. Происходит открытие сохраненного файла.  3.Документ открывается, работа приложения продолжается. | Выполнено  успешно |
| 12 | QtCore, QtSql, QtScript | **Удаление клиента из БД**  Предварительные условия:  Клиент в БД существует.  1. Применяем SQL-запрос DELETE | 1 Происходит удаление клиента в БД.  2.Работа приложения продолжается. | Выполнено  успешно |

Тестирование программного средства – это проверка соответствия между реальным и ожидаемым поведением программы, осуществляемая на конечном наборе тестов, выбранном определенным образом. В более широком смысле, тестирование – это одна из техник контроля качества, включающая в себя активности по планированию работ (Test Management), проектированию тестов (Test Design), выполнению тестирования (Test Execution) и анализу полученных результатов (Test Analysis).

В рамках данногой работы проводится тестирование. Запуск и тестирование выполнялось в операционной системе Windows 10.

Первоначально пользователем заполняется логин и пароль в окне «Авторизация», после чего нажимается кнопка «ОК», которая запускает дальнейшую работу приложения.

|  |
| --- |
|  |

Рисунок 5.1 – Заполнение пользователем данных в окне Авторизация»

На данном этапе пользователь видит главное окно приложения.

|  |
| --- |
|  |

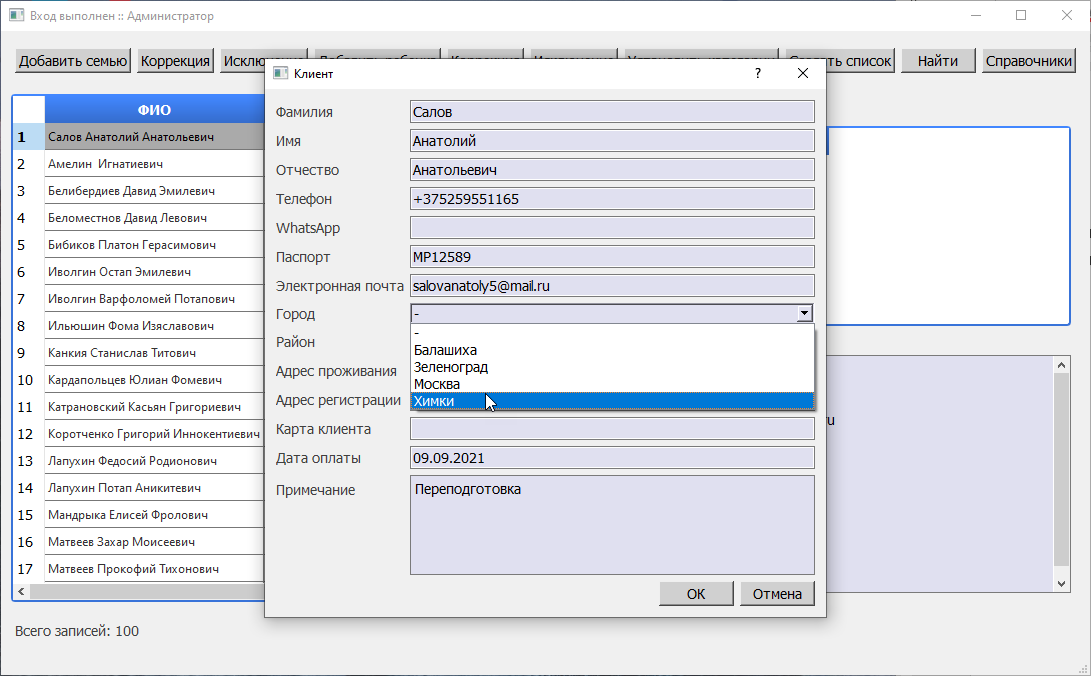
Рисунок 5.2 – Главное окно приложения

После нажатия кнопки «Добавить семью» выводится окно «Клиент» для заполнения данных этой семьи.

|  |
| --- |
|  |

Рисунок 5.3 – Заполнение данных клиента «Семья»

После заполнения формы и нажатия кнопки «ОК» запись добавляется в базу, пользователь может убедиться, посмотрев на счетчик записей в левом нижнем углу и убедиться, что количество записей изменилось.

Рисунок 5.4 – Количество клиентов «Семья» в базе

После того, как клиент «Семья» добавлен в базу, пользователь может произвести коррекцию его данных, выделив клиента и нажав кнопку «Коррекция». Обнаружив, что необходимого города в выподающем списке нету – пользователь должен добавить его в базу городов. Для этого необходимо нажать кнопку «Справочники», а затем перейти в раздел «Города»

|  |
| --- |
|  |

Рисунок 5.5 – Коррекция данных клиента «Семья»

|  |
| --- |
|  |

Рисунок 5.6 – Выбор справочника

В окне «Справочник городов» - пользователь может добавлять, корректировать, удалять города.

|  |
| --- |
|  |

Рисунок 5.7– Добавление города в справочник городов

После добавления нужного города, пользователь может добавить его в данные клиента.

|  |
| --- |
|  |

Рисунок 5.8– Добавление города в справочник городов

Для удаления клиента «Семья» пользователь должен нажать кнопку «Исключение», после чего нажать кнопку «Да» в окне «Подтверждение».

|  |
| --- |
|  |

Рисунок 5.9– Удаление клиента «Семья»

Для каждого клиента «Семья» можно добавить одного или несколько клиентов "Ребенок». Для этого необходимо выделить клиента «Семья» и нажать кнопку «Добавить ребенка».

|  |
| --- |
|  |

Рисунок 5.10– Добавление клиента «Ребенок»

Убедится, что клиент «Ребенок» добавился можно в окне «Дети», выделив нужного нам клиента «Семья».

|  |
| --- |
|  |

Рисунок 5.11– Окно «Дети»

Для каждой семьи можно добавлять или удалять отдельные категории, выделив клиента «Семья» и нажав кнопку «Установить категорию». Проверить категорию можно в столбце «Категория».

|  |
| --- |
|  |

Рисунок 5.12– Добавление категории клиенту «Семья»

Клиент может редактировать список доступных категорий. Для этого необходимо нажать кнопку «Справочники» и выбрать «Категории».

|  |
| --- |
|  |

Рисунок 5.13– Окно «Справочник категорий»

Клиент может производить поиск в базе по необходимым шаблонам. Для этого необходимо нажать кнопку «Создать список», и выбрать необходимой шаблон.

|  |
| --- |
|  |

Рисунок 5.14– Окно «Клиенты по городу»

Предусмотрен экспорт полученных списков в форматы .xlsx и .txt. Для предотвращения ошибок в чтении из-за неправильной кодировки экспортируемых файлов предусмотрен выбор кодировки.

|  |
| --- |
|  |

Рисунок 5.15– Выбор кодировки в окне «Клиенты по городу»

Предусмотрен поиск в базе по любым словам. Для этого необходимо нажать кнопку «Найти». Возможен экспорт полученных списков в форматы .xlsx и .txt. Для предотвращения ошибок в чтении из-за неправильной кодировки экспортируемых файлов предусмотрен выбор кодировки.

|  |
| --- |
|  |

Рисунок 5.16– Окно «Поиск»

Поиск в базе работает по частям слова, выводя все результаты в список.

Полученный список пользователь может экспортировать в форматы .xlsx и .txt.

|  |
| --- |
|  |

Рисунок 5.17– Результат поиска

Пользователь имеет возможность проанализировать качественный состав клиентов построением диаграмм. Диаграммы можно строить нескольких типов: число детей по возрастам, число детей в семьях.

|  |
| --- |
|  |

Рисунок 5.18– Выбор типа диаграммы в окне «Выбор шаблона»

Диаграмма «Число детей по возрастам» показывает количество детей и возрост каждой группы.

|  |
| --- |
|  |

Рисунок 5.19– Диаграмма «Число детей по возрастам»

Диаграмма «Число детей в семьях» показывает количество детей и количество семей в каждом городе. Перед составлением диаграммы необходимо указать город в котором будет производится поиск.

|  |
| --- |
|  |

Рисунок 5.20– Диаграмма «Число детей в семьях»

Для оценки оценки разработанного программного средства было выбрано и проведено функциональное тестирование, поскольку данный вид тестирования позволил проверить соответствие разработанного программного средства требованиям задания. В ходе проведенного тестирования было продемонстрированно выполнение программным средством следующих функций:

– добавление клиента с контактными и анкетными данными;

– коррекция ранее внесенного клиента;

– удаление ранее внесенного клиента;

– добавление ребенка к клиенту с документами и анкетными данными;

– коррекция ранее внесенного ребенка;

– присвоение клиенту одной или нескольких категорий.

– ведение справочника категорий.

– ведение справочника городов.

– формирование списков по заданным правилам, экспорт в различные форматы полученных списков.

– поиск в базе данных любых сведений по части текста.

– визуализация клиентской базы с помощью диаграмм.

# Заключение

В рамках данного дипломного проекта для ведения учета учащихся детского образовательного центра было разработано кроссплатформенное программное средства относящееся к типу CRM-систем на фрэймворке QT и языке программирования «С++».

В ходе разработки были решены и реализованы следующие задачи:

– описана предметная область и разработана функциональная модель, дан перечень используемых средств и технологий, применяемых при разработке программного средства, а также описаны модули входящие в фрэймворк QT и дано краткое их описание;

– проведен функциональный анализ предметной области, а также проведена разработка структуры программного средства.

– разработана структура классов при использовании объектно-ориентированного стиля програм-мирования приложений и описана структура базы данных, а также реализация подключения базы данных в программу с помощью инструментов фрэймворка QT;

– описаны агоритмы входа в систему, регистрации нового пользователя, сохранения клиента в БД, удаления клиента в БД, экспорта списков.

– проведено визуальное и функциональное тестирование разработанного программного средства, описана разработка алгоритмов и их программная реализация;

Таким образом, разработанное программное средство может быть использовано в сферах деятельности, связанных с учетом персонифицированных данных.

# Список использованных источников

1. Уроки по Qt5 [Электронный ресурс]. – 2020. – Режим доступа: https://ravesli.com/uroki-po-qt5/. – Дата доступа: 04.09.2021.
2. Руководство по Qt для начинающих [Электронный ресурс]. – 2014. – Режим доступа: http://knzsoft.ru/qt-bgr/. – Дата доступа: 04.09.2021.
3. Уроки QT Creator [Электронный ресурс]. – 2021. – Режим доступа: https://itproger.com/course/qt-creator. – Дата доступа: 04.09.2021.
4. Learning Hub [Электронный ресурс]. – 2020. – Режим доступа: https://resources.qt.io/learning-hub. – Дата доступа: 04.09.2021.
5. Сайт компании «QT» [Электронный ресурс]. – 2021. – Режим доступа: https://www.qt.io. – Дата доступа: 04.09.2021.
6. Все о кросс платформенном программировании [Электронный ресурс]. – 2021. – Режим доступа: http://www.forum.crossplatform.ru. – Дата оступа: 04.09.2021.
7. Сайт по работе с QT «Qt – EVILEG». – 2021. – Режим доступа: https://evileg.com/en/knowledge/qt/. – Дата доступа: 04.09.2021.
8. Уроки программирования на языке С++ [Электронный ресурс]. – 2021. – Режим доступа: https://ravesli.com/uroki-cpp/. – Дата доступа: 04.09.2021.
9. Язык программирования C++ [Электронный ресурс]. – 2020. – Режим доступа: https://metanit.com/cpp/tutorial/1.1.php. – Дата доступа: 04.09.2021.
10. Qt 5.10. Профессиональное программирование на С++. М. Шлее/ СПб.: БХВ-Петербург, 2018. -1072 с.
11. Тестирование. Фундаментальная теория [Электронный ресурс]. – Режим доступа: https://habrahabr.ru/post/279535/. 2018. – Дата доступа: 09.02.2021.
12. Основы программирования на C++ [Электронный ресурс]. – 2021. – Режим доступа: [https://purecodecpp.com](https://habr.com/ru/company/regionsoft/blog/520242/). – Дата доступа: 04.09.2021.

# Приложение А (обязательное) Листинг кода

Класс mainwindow.cpp

MainWindow::MainWindow(QWidget \*parent)

    : QMainWindow(parent)

    , ui(new Ui::MainWindow)

{

    ui->setupUi(this);

    QDir::setCurrent("../data") ;

    ui->tableClients->horizontalHeader()->setSectionResizeMode(QHeaderView::ResizeToContents) ;

    ui->tableClients->verticalHeader()->setSectionResizeMode(QHeaderView::ResizeToContents) ;

    ui->tableClients->setSelectionBehavior(QAbstractItemView::SelectRows) ;

    ui->tableClients->setSelectionMode(QAbstractItemView::SingleSelection) ;

    ui->tableChildrens->horizontalHeader()->setSectionResizeMode(QHeaderView::ResizeToContents) ;

    //ui->tableChildrens->verticalHeader()->setSectionResizeMode(QHeaderView::ResizeToContents) ;

    ui->tableChildrens->setSelectionBehavior(QAbstractItemView::SelectRows) ;

    ui->tableChildrens->setSelectionMode(QAbstractItemView::SingleSelection) ;

    qApp->setStyleSheet(Helper::readAllText("style.css"));

    connect(QJsonUtils::getGlobalUtils(),

            SIGNAL(sendParseError(const QString &, const QString &)),

            this,

            SLOT(getParseError(const QString &, const QString &))) ;

    QJsonArray clients ;//loadClients() ;

    cities = QJsonUtils::getGlobalUtils()->loadArrayFromFile("cities.json") ;

    categories = QJsonUtils::getGlobalUtils()->loadArrayFromFile("categories.json") ;

    view\_client = QJsonUtils::getGlobalUtils()->loadObjectFromFile("view\_client.json") ;

    view\_children = QJsonUtils::getGlobalUtils()->loadObjectFromFile("view\_children.json") ;

    updateInfo(clients) ;

    engine = new MainEngine(new QScriptEngine()) ;

    engine->getEngine()->evaluate(Helper::readAllText("userfunc.js")) ;

    modelclientsorted = new QJsonSortProxyModel(view\_client,engine->getEngine()) ;

    modelclientsorted->setSourceModel(modelclient = new QJsonTableModel(clients,view\_client,engine->getEngine())) ;

    modelclientsorted->setDynamicSortFilter(true) ;

    ui->tableClients->setModel(modelclientsorted) ;

    modelchildren = new QJsonTableModel(QJsonArray(),view\_children,engine->getEngine()) ;

    ui->tableClients->sortByColumn(0, Qt::AscendingOrder);

    connect(ui->tableClients->selectionModel(),SIGNAL(selectionChanged(const QItemSelection&,const QItemSelection&)),

            this,SLOT(on\_row\_select(const QItemSelection&,const QItemSelection&))) ;

    connect(modelclient,SIGNAL(sendNoSelected(const QString &)),

            this,SLOT(messageNoSelected(const QString &))) ;

    connect(modelclient,SIGNAL(dataUpdated(const QJsonArray &)),

            this,SLOT(saveClients(const QJsonArray &))) ;

    connect(modelclient,SIGNAL(dataChanged(const QModelIndex &,const QModelIndex &,const QVector<int> &)),

            this,SLOT(dataChangedClients(const QModelIndex &,const QModelIndex &,const QVector<int> &))) ;

    ui->butImport->hide() ;

    ui->ButGet->hide() ;

    ui->ButPost->hide() ;

    QTimer::singleShot(1000,this,SLOT(doAuth())) ;

    //if (!doAuth()) qApp->closeAllWindows() ;

}

void MainWindow::updateInfo(const QJsonArray & data)

{

    ui->labInfo->setText("Всего записей: "+QString::number(data.count()));

}

void MainWindow::UpdatePanel(const QModelIndex &index)

{

    if (!modelclient->isCurrent(index)) return ;

    QJsonObject obj = modelclient->getCurrent(index) ;

    QStringList list ;

    list.append("Код:"+QString::number(obj["id"].toInt())) ;

    foreach(auto key, obj.keys()) {

        if (obj[key].isString()) {

            if (obj[key].toString().length()>0)

                list.append(key+": "+obj[key].toString()) ;

        }

    }

    ui->textInfo->setText(list.join("\n")) ;

    //ui->textInfo->setText(engine->getEngine()->evaluate(

    //     "shell\_CalcPanel("+QJsonDocument(obj).toJson()+")").toString()) ;

    ui->textInfo->setText(QJsonDocument(obj).toJson()) ;

    modelchildren->replaceSource(obj["Childrens"].toArray()) ;

    ui->tableChildrens->setModel(modelchildren) ;

}

ClientAPI \*MainWindow::getAPI()

{

    ClientAPI \* api = new ClientAPI(this) ;

    connect(api,SIGNAL(errorAPI(APIErrorType, const QString &, const QString &)),

            this,SLOT(getErrorAPI(APIErrorType, const QString &, const QString &))) ;

    return api ;

}

void MainWindow::reloadClients()

{

    ClientAPI \* api = getAPI() ;

    connect(api,SIGNAL(actionFromAPI(const QJsonValue &)),this,SLOT(setClientsFromAPI(const QJsonValue &))) ;

    api->execGet("getclients",user\_id) ;

}

void MainWindow::doAuth()

{

    if (QFile::exists("passhash")) {

        QJsonObject auth = QJsonUtils::createObjectWith1Field("passhash",Helper::readAllText("passhash").trimmed()) ;

        ClientAPI \* api = getAPI() ;

        connect(api,SIGNAL(actionFromAPI(const QJsonValue &)),this,SLOT(getAuthReplyAPI(const QJsonValue &))) ;

        api->execPost("doauth",user\_id,auth) ;

        return ;

    }

    FormBuilder builder(this,"Авторизация") ;

    QFormLayout \* layout = builder.addFormLayout() ;

    QLineEdit \* eduser = new QLineEdit() ;

    QLineEdit \* edpass = new QLineEdit() ;

    edpass->setEchoMode(QLineEdit::EchoMode::Password) ;

    layout->addRow(new QLabel("Логин"),eduser) ;

    layout->addRow(new QLabel("Пароль"),edpass) ;

    eduser->setFocus() ;

    if (builder.exec()) {

        QJsonObject auth = QJsonUtils::createObjectWith2Field("login",eduser->text().trimmed(),"pass",edpass->text()) ;

        ClientAPI \* api = getAPI() ;

        connect(api,SIGNAL(actionFromAPI(const QJsonValue &)),this,SLOT(getAuthReplyAPI(const QJsonValue &))) ;

        api->execPost("doauth",user\_id,auth) ;

    }

    else

        this->close() ;

}

/\*

QJsonArray MainWindow::loadCliens()

{

     QJsonArray clients = QJsonUtils::getGlobalUtils()->loadArrayFromFile("base.json") ;

     for (int i=0; i<clients.count(); i++) {

         QJsonObject el = clients[i].toObject() ;

         el["id"]=i+1 ;

         clients.replace(i,el) ;

     }

     return clients ;

}

\*/

void MainWindow::dataChangedClients(const QModelIndex &topLeft, const QModelIndex &bottomRight, const QVector<int> &roles)

{

    UpdatePanel(topLeft) ;

}

void MainWindow::on\_row\_select(const QItemSelection &selected, const QItemSelection &deselected)

{

    Q\_UNUSED(deselected) ;

    //ui->labInfo->setText("") ;

    if (selected.indexes().size()==0) return ;

    QModelIndex index = modelclientsorted->mapToSource(selected.indexes()[0]) ;

    UpdatePanel(index) ;

}

void MainWindow::saveClients(const QJsonArray & data)

{

    //QJsonUtils::getGlobalUtils()->saveArrayToFile(DATADIR+"shells.json",data) ;

    updateInfo(data) ;

}

void MainWindow::messageNoSelected(const QString &whatis)

{

    QMessageBox::information(this,"Ошибка","Не выбрана запись: "+whatis) ;

}

void MainWindow::getParseError(const QString &filename, const QString &error)

{

    QMessageBox::information(this,"Ошибка","В файле "+filename+" ошибка "+error) ;

}

MainWindow::~MainWindow()

{

    delete modelclientsorted ;

    delete engine ;

    delete ui;

}

void MainWindow::on\_butImport\_clicked()

{

    return ;

/\*

    QStringList log ;

    QJsonArray clients = loadClients() ;

    QJsonArray import = QJsonUtils::getGlobalUtils()->loadArrayFromFile("result.json") ;

    foreach(auto obj, import) {

        int idxfind = -1 ;

        for (int i=0; i<clients.count(); i++)

            if ((clients[i].toObject()["FamilyM"].toString()==obj["FamilyM"].toString())&&

                (clients[i].toObject()["NameM"].toString()==obj["NameM"].toString())&&

                (clients[i].toObject()["ParentM"].toString()==obj["ParentM"].toString()))

                idxfind=i ;

        if (idxfind!=-1) {

            QJsonDocument doc ;

            doc.setObject(obj.toObject()) ;

            log.append(doc.toJson()) ;

        }

        else

            clients.append(obj) ;

    }

    Helper::writeAllText("log",log.join("\n")) ;

    QJsonUtils::getGlobalUtils()->saveArrayToFile("base.json",clients) ;

    QMessageBox::information(this,"OK","Import OK, skipped: "+QString::number(log.count())) ;

\*/

}

void MainWindow::on\_ButList\_clicked()

{

    QJsonArray files ;

    QDir dir ;

    foreach(auto fi, dir.entryInfoList())

        if (fi.fileName().startsWith("list\_")) {

            QJsonObject cont = QJsonUtils::getGlobalUtils()->loadObjectFromFile(fi.fileName()) ;

            files.append(QJsonUtils::createObjectWith2Field("filename",fi.fileName(),

                                                            "caption",cont["viewname"].toString())) ;

        }

    DialogSelList dsel(files,QJsonUtils::createObjectWith2Field("title","Выбор шаблона","itemcaption","caption"),this) ;

    if (dsel.exec()!=QDialog::Accepted) return ;

    QJsonObject list\_info = QJsonUtils::getGlobalUtils()->loadObjectFromFile(dsel.getSelected()["filename"].toString()) ;

    QJsonObject filter ;

    //qDebug()<<list\_info ;

    if (list\_info.contains("filter")) {

        FormBuilder builder(this,"Параметры списка") ;

        QFormLayout \* layout = builder.addFormLayout() ;

        foreach(auto item, list\_info["filter"].toArray()) {

            QJsonObject filterelem = item.toObject() ;

            QLabel \* lab = new QLabel(filterelem["caption"].toString()) ;

            QLineEdit \* ed = new QLineEdit() ;

            ed->setProperty("filtername",filterelem["name"].toString()) ;

            layout->addRow(lab,ed) ;

        }

        if (builder.exec())

            for (int i=0; i<layout->rowCount(); i++) {

                auto \* item = (QLineEdit\*)layout->itemAt(i,QFormLayout::ItemRole::FieldRole)->widget() ;

                filter[item->property("filtername").toString()]=item->text().trimmed() ;

            }

        else

            return ;

    }

    QJsonArray list\_source ;

    if (list\_info.contains("source")) {

        if (list\_info["source"].toString()=="join") {

            // Строим комбинированный список

            foreach(auto client, modelclient->getSource()) {

                QJsonObject row = client.toObject() ;

                foreach(auto child, client.toObject()["Childrens"].toArray()) {

                    QJsonObject row1 = child.toObject() ;

                    foreach(auto key, row.keys())

                        row1.insert(key,row[key]) ;

                    list\_source.append(row1) ;

                }

            }

        }

        else

            list\_source = modelclient->getSource() ;

    }

    else

        list\_source = modelclient->getSource() ;

    auto modellist = new QJsonTableModel(list\_source,list\_info,engine->getEngine()) ;

    QJsonSortProxyModel \* model = new QJsonSortProxyModel(list\_info,engine->getEngine()) ;

    model->setDynamicSortFilter(true) ;

    model->setFilterObject(filter) ;

    model->setSourceModel(modellist) ;

    DialogViewList d(this) ;

    d.setWindowTitle(list\_info["viewname"].toString()) ;

    d.setModel(model) ;

    d.exec() ;

    delete model ;

}

void MainWindow::on\_ButFind\_clicked()

{

    QJsonArray clients = modelclient->getSource() ;

    QString str = QInputDialog::getText(this,"Поиск","Введите строку") ;

    if (str.length()==0) return ;

    QJsonArray searchres ;

    QJsonSearchInObject search(str) ;

    foreach(auto item, clients) {

        search.findInObject(item.toObject()) ;

        foreach(auto key, search.getWhereFind().keys()) {

            QJsonObject found = item.toObject() ;

            found["fieldname"]=key ;

            found["fieldvalue"]=search.getWhereFind()[key] ;

            searchres.append(found) ;

        }

    }

    if (searchres.count()==0) {

        QMessageBox::information(this,"Ошибка","По тексту <"+str+"> ничего не найдено") ;

        return ;

    }

    QJsonObject list\_info = QJsonUtils::getGlobalUtils()->loadObjectFromFile("view\_search.json") ;

    auto modelsearch = new QJsonTableModel(searchres,list\_info,engine->getEngine()) ;

    QJsonSortProxyModel \* model = new QJsonSortProxyModel(list\_info,engine->getEngine()) ;

    model->setDynamicSortFilter(true) ;

    model->setSourceModel(modelsearch) ;

    DialogViewList d(this) ;

    d.setWindowTitle(list\_info["viewname"].toString()) ;

    d.setModel(model) ;

    d.exec() ;

    delete model ;

    // go to main list

    /\*

    QModelIndex index = modelclientsorted->mapFromSource(modelclient->getIndexByField("id",10)) ;

    ui->tableClients->selectionModel()->select(index,QItemSelectionModel::SelectionFlag::Select) ;

    ui->tableClients->scrollTo(index) ;

    \*/

}

void MainWindow::on\_ButGet\_clicked()

{

}

void MainWindow::on\_ButPost\_clicked()

{

    /\*QJsonArray clients = modelclient->getSource() ;

    QJsonDocument doc ;

    doc.setObject(clients[0].toObject()) ;

    QNetworkRequest req = QNetworkRequest(QUrl(API\_URL+QString("?action=setclient&user\_id=7"))) ;

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

    QNetworkReply \* reply = manager.post(req,doc.toJson());

    loader = new DialogLoader(this) ;

    loader->open() ;\*/

}

void MainWindow::on\_butAddClient\_clicked()

{

}

void MainWindow::setClientsFromAPI(const QJsonValue &result)

{

    modelclient->replaceSource(result.toArray()) ;

}

void MainWindow::getAuthReplyAPI(const QJsonValue &result)

{

    QJsonObject auth = result.toObject() ;

    if (auth["authok"].toBool()) {

        user\_id = auth["user\_id"].toInt() ;

        setWindowTitle("База Единства :: "+auth["username"].toString()) ;

        reloadClients() ;

        //QTimer::singleShot(1000,this,SLOT(reloadClients())) ;

    }

    else {

        QMessageBox::information(this,"Ошибка авторизации",auth["msg"].toString()) ;

        QTimer::singleShot(1000,this,SLOT(doAuth())) ;

    }

}

void MainWindow::getErrorAPI(APIErrorType errtype, const QString &request, const QString &errmsg)

{

    QMessageBox::information(this,"Ошибка API "+request,errmsg) ;

}

Класс dialogviewlist.cpp

DialogViewList::DialogViewList(QWidget \*parent) :

    QDialog(parent),

    ui(new Ui::DialogViewList)

{

    ui->setupUi(this);

    ui->tableList->horizontalHeader()->setSectionResizeMode(QHeaderView::ResizeToContents) ;

    ui->tableList->verticalHeader()->setSectionResizeMode(QHeaderView::ResizeToContents) ;

    ui->tableList->setSelectionBehavior(QAbstractItemView::SelectRows) ;

    ui->tableList->setSelectionMode(QAbstractItemView::SingleSelection) ;

}

DialogViewList::~DialogViewList()

{

    delete ui;

}

void DialogViewList::setModel(QSortFilterProxyModel \* model)

{

    this->model = model ;

    ui->tableList->setModel(model) ;

    ui->tableList->sortByColumn(0, Qt::AscendingOrder);

}

void DialogViewList::on\_ButExport\_clicked()

{

    QStringList res ;

    for (int i=0; i<ui->tableList->model()->rowCount(); i++) {

        QString str = QString::number(i+1)+". " ;

        for (int j=0; j<ui->tableList->model()->columnCount(); j++)

            str+=ui->tableList->model()->data(ui->tableList->model()->index(i,j)).toString()+"  " ;

        res.append(str.trimmed()) ;

    }

    QString filename=QDir::tempPath()+'/'+QUuid::createUuid().toString()+".txt" ;

    Helper::writeAllText(filename,res.join("\r\n")) ;

    QDesktopServices::openUrl(QUrl::fromLocalFile(filename));

}

void DialogViewList::on\_butDel\_clicked()

{

    QJsonTableModel \* sourcemodel = (QJsonTableModel \*)model->sourceModel() ;

    QModelIndex index = model->mapToSource(ui->tableList->currentIndex()) ;

    if (!sourcemodel->isCurrent(index)) return ;

    sourcemodel->deleteFromModel(index) ;

}

void DialogViewList::on\_ButExportCSV\_clicked()

{

    QStringList res ;

    QString str = "№;" ;

    for (int j=0; j<ui->tableList->model()->columnCount(); j++)

        str+="\""+ui->tableList->model()->headerData(j,Qt::Horizontal).toString()+"\";" ;

    res.append(str.trimmed()) ;

    for (int i=0; i<ui->tableList->model()->rowCount(); i++) {

        QString str = QString::number(i+1)+";" ;

        for (int j=0; j<ui->tableList->model()->columnCount(); j++)

            str+="\""+ui->tableList->model()->data(ui->tableList->model()->index(i,j)).toString().replace("\"","")+"\";" ;

        res.append(str.trimmed()) ;

    }

    QString filename=QDir::tempPath()+'/'+QUuid::createUuid().toString()+".csv" ;

    Helper::writeAllText(filename,res.join("\r\n")) ;

    QDesktopServices::openUrl(QUrl::fromLocalFile(filename));

}

Класс dialogeditobject.cpp

DialogEditObject::DialogEditObject(const QJsonObject & obj, const QJsonObject & settings,

QScriptEngine \*engine, QWidget \*parent) :

QDialog(parent),

ui(new Ui::DialogEditObject)

{

ui->setupUi(this);

this->engine = engine ;

this->settings = settings ;

saved = obj ;

ui->buttonBox->button(QDialogButtonBox::Ok)->setText("ОК");

ui->buttonBox->button(QDialogButtonBox::Cancel)->setText("Отмена");

setWindowTitle(settings.value("title").toString()) ;

if (settings.value("inputs").toArray().count()==0)

foreach (QString key, saved.keys())

addRow(saved,key,key,QJsonObject()) ;

else

foreach (QJsonValue v, settings.value("inputs").toArray()) {

QJsonObject optrow = v.toObject() ;

QString key = optrow["field"].toString() ;

if (saved[key].isNull()||saved[key].isUndefined())

if (optrow["default"].isString()||optrow["default"].isDouble()||optrow["default"].isBool())

saved[key] = optrow["default"] ;

addRow(saved,key,optrow["caption"].toString(),optrow) ;

}

connect(ui->buttonBox,SIGNAL(rejected()),this,SLOT(reject())) ;

connect(ui->buttonBox,SIGNAL(accepted()),this,SLOT(accept())) ;

//connect(ui->buttonBox,SIGNAL(accepted()),this,SLOT(saveObject())) ;

}

void DialogEditObject::addRow(const QJsonObject &obj, const QString &key, const QString &caption, const QJsonObject &optrow)

{

QJsonValue data = obj[key] ;

QWidget \* widget = NULL;

if (data.isString()) {

if (optrow["control"].toString()==QString("dateedit")) {

widget = new QDateEdit(MainEngine::str2rusdate(data.toString()),this) ;

((QDateEdit\*)widget)->setCalendarPopup(true) ;

}

else

if (optrow["control"].toString()==QString("memo")) {

widget = new QPlainTextEdit(data.toString(),this) ;

((QPlainTextEdit\*)widget)->setFixedHeight(

optrow["memoheight"].isDouble()?

optrow["memoheight"].toInt():100) ;

}

else

if (optrow["control"].toString()==QString("combo")) {

widget = new QComboBox(this) ;

((QComboBox\*)widget)->insertItem(0,"-") ;

QStringList source = optrow["combosource"].toString().split("::") ;

if (source.length()==2) {

if (source[0]=="[func]") {

QScriptValue list = engine->evaluate(source[1]) ;

for (int i=0; i<list.property("length").toInt32(); i++)

((QComboBox\*)widget)->addItem(list.property(i).toString());

int idx = ((QComboBox\*)widget)->findText(data.toString()) ;

if (idx!=-1) ((QComboBox\*)widget)->setCurrentIndex(idx) ;

else ((QComboBox\*)widget)->setCurrentIndex(0) ;

}

}

}

else

widget = new QLineEdit(data.toString(),this) ;

}

else

if (data.isDouble())

widget = new QLineEdit(QString::number(data.toDouble()),this) ;

else

if (data.isBool()) {

widget = new QCheckBox(this) ;

((QCheckBox\*)widget)->setChecked(data.toBool()) ;

}

if (widget!=NULL) {

widget->setProperty("pkey",key) ;

ui->formLayout->addRow(caption,widget) ;

}

}

DialogEditObject::~DialogEditObject()

{

delete ui;

}

QJsonObject DialogEditObject::getObject() const

{

QJsonObject res = saved ;

for (int i=0; i<ui->formLayout->rowCount(); i++) {

QLayoutItem \* wItem = ui->formLayout->itemAt(i,QFormLayout::FieldRole) ;

QString key = wItem->widget()->property("pkey").toString() ;

if (res[key].isString()) {

if (QDateEdit \* dateedit = qobject\_cast<QDateEdit\*>(wItem->widget())) {

res[key]=dateedit->date().toString("dd.MM.yyyy") ;

}

else

if (QPlainTextEdit \* memo = qobject\_cast<QPlainTextEdit\*>(wItem->widget())) {

res[key]=memo->toPlainText() ;

}

else

if (QComboBox \* combo = qobject\_cast<QComboBox\*>(wItem->widget())) {

res[key]=combo->currentText() ;

}

else

res[key]=((QLineEdit\*)wItem->widget())->text() ;

}

else

if (res[key].isDouble())

res[key]=((QLineEdit\*)wItem->widget())->text().toDouble() ;

else

if (res[key].isBool())

res[key]=((QCheckBox\*)wItem->widget())->isChecked() ;

}

return res ;

}

Класс dialogsellist.cpp

DialogSelList::DialogSelList(const QJsonArray & list, const QJsonObject & view,

                             QWidget \*parent) :

    QDialog(parent),

    ui(new Ui::DialogSelList)

{

    ui->setupUi(this);

    this->view = view ;

    this->list = list ;

    ui->buttonBox->button(QDialogButtonBox::Ok)->setText("ОК");

    ui->buttonBox->button(QDialogButtonBox::Cancel)->setText("Отмена");

    setWindowTitle(view["title"].toString()) ;

    foreach (auto obj, list) {

        QString caption = obj.toObject()[view["itemcaption"].toString()].toString() ;

        auto item = new QListWidgetItem(caption,ui->listWidget) ;

        item->setData(QListWidgetItem::UserType,obj.toObject()) ;

        ui->listWidget->addItem(item) ;

    }

    if (ui->listWidget->count()>0) ui->listWidget->setCurrentRow(0) ;

    connect(ui->buttonBox,SIGNAL(rejected()),this,SLOT(reject())) ;

    connect(ui->buttonBox,SIGNAL(accepted()),this,SLOT(accept())) ;

}

DialogSelList::~DialogSelList()

{

    delete ui;

}

QJsonObject DialogSelList::getSelected()

{

    if (ui->listWidget->currentItem()==NULL) return QJsonObject() ;

    return ui->listWidget->currentItem()->data(QListWidgetItem::UserType).toJsonObject() ;

}

Класс qjsontablemodel.cpp

void QJsonTableModel::rebuildAutoColumns()

{

    autocolumns.clear() ;

    for (int i=0; i<array.count(); i++) {

        QJsonObject row = array.at(i).toObject() ;

        foreach (QString key, row.keys())

            if (row[key].isBool()||row[key].isString()||row[key].isDouble())

                if (!autocolumns.contains(key)) autocolumns.append(key);

    }

}

QJsonTableModel::QJsonTableModel(const QJsonArray &source, const QJsonObject &view,

                                 QScriptEngine \* engine)

{

    this->array = source ;

    this->view = view ;

    columns = view["columns"].toArray() ;

    useautocolumns = columns.count()==0 ;

    rebuildAutoColumns() ;

    this->engine = engine ;

}

int QJsonTableModel::rowCount( const QModelIndex& parent ) const {

    Q\_UNUSED(parent) ;

    return array.count() ;

}

int QJsonTableModel::columnCount( const QModelIndex& parent ) const {

    Q\_UNUSED(parent) ;

    return useautocolumns?autocolumns.count():columns.count() ;

}

QVariant QJsonTableModel::data( const QModelIndex& index, int role ) const {

    if(

        !index.isValid() ||

        array.count() <= index.row())

        return QVariant();

    if (role == Qt::BackgroundRole) {

        if (array.at(index.row()).toObject().contains("inbox\_id"))

            return QColor::fromRgb(200,200,255) ;

        else

            return QVariant() ;

    }

    if (( role != Qt::DisplayRole && role != Qt::EditRole ))

        return QVariant();

    QJsonObject rowobj = array.at(index.row()).toObject() ;

    QJsonValue val ;

    if (useautocolumns) {

        val = rowobj.value(autocolumns[index.column()]) ;

        return QJsonUtils::getGlobalUtils()->getJsonValueAsString(val) ;

    }

    else {

        QJsonObject rowparam = columns.at(index.column()).toObject() ;

        if (rowparam["field"].toString()=="[calc]") {

            QString calc = rowparam["func"].toString() ;

            calc = calc.replace("<current>",QJsonDocument(rowobj).toJson()) ;

            return engine->evaluate(calc).toString() ;

        }

        else {

            val = rowobj[rowparam["field"].toString()] ;

            return QJsonUtils::getGlobalUtils()->getJsonValueAsString(val) ;

        }

    }

    return QVariant() ;

}

bool QJsonTableModel::setData( const QModelIndex& index, const QVariant& value, int role ) {

    return false ;

    /\*

    if( !index.isValid() || role != Qt::EditRole || list->count() <= index.row() )

        return false;

    bool isok = false;

    int n = value.toInt(&isok) ;

    if (index.column()==PLACECOUNT) {

        if (!isok) return false ;

        if (!tickets->at(index.row())->canSetPlaceCount(n)) return false ;

        tickets->at(index.row())->updatePlaceCount(n) ;

        emit dataChanged(index, index.sibling(index.row(),index.column()+1));

    }

    return true ;

    \*/

}

QVariant QJsonTableModel::headerData( int section, Qt::Orientation orientation, int role ) const {

    if( role != Qt::DisplayRole )

        return QVariant() ;

    if( orientation == Qt::Vertical )

        return section+1;

    if (useautocolumns)

        return autocolumns[section] ;

    else

        return columns.at(section).toObject().value("caption") ;

}

Qt::ItemFlags QJsonTableModel::flags( const QModelIndex& index ) const {

    Qt::ItemFlags flags = QAbstractTableModel::flags( index );

    /\*if( index.isValid() ) {

        if( index.column() == PLACECOUNT ) {

            flags |= Qt::ItemIsEditable;

        }

    }

\*/

    return flags;

}

bool QJsonTableModel::isCurrent(const QModelIndex &index) const

{

    if (index.isValid() && (array.size() > index.row()))

        return true ;

    else {

        emit sendNoSelected(view.value("viewname").toString());

        return false ;

    }

}

QJsonObject QJsonTableModel::getCurrent(const QModelIndex &index) const {

    QJsonObject obj ;

    if (index.isValid() && (array.size() > index.row()))

        obj = array.at(index.row()).toObject() ;

    return obj ;

}

void QJsonTableModel::replaceSource(const QJsonArray &source)

{

    this->array = source ;

    beginResetModel();

    rebuildAutoColumns() ;

    endResetModel();

}

void QJsonTableModel::updateModel(const QModelIndex &index, const QJsonObject & newobj) {

    if (index.isValid() && (array.size() > index.row())) {

        array.replace(index.row(),newobj) ;

        emit dataUpdated(array) ;

        emit dataChanged(index.sibling(index.row(),0),

                         index.sibling(index.row(),columnCount(index)-1));

    }

}

void QJsonTableModel::insertToModel(const QJsonObject & newobj) {

    array.append(newobj) ;

    emit dataUpdated(array) ;

    beginResetModel();

    rebuildAutoColumns() ;

    endResetModel();

}

void QJsonTableModel::deleteFromModel(const QModelIndex &index)

{

    if (index.isValid() && (array.size() > index.row())) {

        array.removeAt(index.row()) ;

        emit dataUpdated(array) ;

        beginResetModel();

        rebuildAutoColumns() ;

        endResetModel();

    }

}

QJsonArray QJsonTableModel::getSource() const

{

    return array ;

}

QModelIndex QJsonTableModel::getIndexByField(const QString &field, const QJsonValue &value) const

{

    for (int i=0; i<array.count(); i++)

        if (array[i].toObject().contains(field))

            if (array[i].toObject()[field]==value)

                return index(i,0) ;

}

Класс qjsonsortproxymodel.cpp

QJsonSortProxyModel::QJsonSortProxyModel(const QJsonObject &view, QScriptEngine \* engine)

{

    this->view = view ;

    this->engine=engine ;

}

bool QJsonSortProxyModel::lessThan(const QModelIndex &left, const QModelIndex &right) const

{

    if (view["sortfunc"].isUndefined())

        return false ;

    else {

        QJsonObject leftObj = ((QJsonTableModel \*)sourceModel())->getCurrent(left) ;

        QJsonObject rightObj = ((QJsonTableModel \*)sourceModel())->getCurrent(right) ;

        QString sort = view["sortfunc"].toString() ;

        sort = sort.replace("<left>",QJsonDocument(leftObj).toJson()) ;

        sort = sort.replace("<right>",QJsonDocument(rightObj).toJson()) ;

        QScriptValue res = engine->evaluate(sort).toBool() ;

        if (res.isError())

            qDebug()<<res.toString() ;

        return res.toBool() ;

    }

    /\*

    QVariant leftData = sourceModel()->data(left);

    QVariant rightData = sourceModel()->data(right);

    if ((left.column()==1)&&(right.column()==1)) {

        QDate leftDate = QDate::fromString(leftData.toString(),"dd.MM.yyyy") ;

        if (!leftDate.isValid()) leftDate = QDate::currentDate() ;

        QDate rightDate = QDate::fromString(rightData.toString(),"dd.MM.yyyy") ;

        if (!rightDate.isValid()) rightDate = QDate::currentDate() ;

        return leftDate<rightDate ;

    }

    else {

        return leftData<rightData ;

    }

    \*/

}

QVariant QJsonSortProxyModel::headerData(int section, Qt::Orientation orientation,

                                int role) const {

    return sourceModel()->headerData(section, orientation,

                                     role);

}

bool QJsonSortProxyModel::filterAcceptsRow(int source\_row, const QModelIndex &source\_parent) const

{

    if (view["filterfunc"].isUndefined())

        return true ;

    else {

        QJsonObject obj = ((QJsonTableModel \*)sourceModel())->getCurrent(sourceModel()->index(source\_row,0,source\_parent)) ;

        QString filterfunc = view["filterfunc"].toString() ;

        filterfunc = filterfunc.replace("<obj>",QJsonDocument(obj).toJson()) ;

        filterfunc = filterfunc.replace("<filter>",QJsonDocument(filter).toJson()) ;

        QScriptValue res = engine->evaluate(filterfunc).toBool() ;

        if (res.isError())

            qDebug()<<res.toString() ;

        return res.toBool() ;

    }

}

void QJsonSortProxyModel::setFilterObject(const QJsonObject &filter)

{

    this->filter = filter ;

}

Класс qjsonsearchinobject.cpp

#include "qjsonsearchinobject.h"

#include <QStringList>

QJsonSearchInObject::QJsonSearchInObject(const QString & str)

{

    this->str = str.trimmed() ;

}

void QJsonSearchInObject::recFindInObject(const QJsonObject & obj, const QString & parentpart)

{

    foreach (auto key, obj.keys()) {

        if (obj[key].isString())

            if (obj[key].toString().contains(str,Qt::CaseInsensitive))

                fields.insert(parentpart+key,obj[key].toString()) ;

        if (obj[key].isDouble())

            if (QString::number(obj[key].toDouble()).contains(str,Qt::CaseInsensitive))

                fields.insert(parentpart+key,QString::number(obj[key].toDouble())) ;

        if (obj[key].isObject())

            recFindInObject(obj[key].toObject(),key+"\_") ;

        if (obj[key].isArray())

            foreach(auto item, obj[key].toArray())

                if (item.isObject())

                    recFindInObject(item.toObject(),key+"\_") ;

    }

}

void QJsonSearchInObject::findInObject(const QJsonObject & obj)

{

    fields.clear() ;

    recFindInObject(obj,"") ;

}

QMap<QString,QString> QJsonSearchInObject::getWhereFind() const {

    return fields ;

}

Класс formbuilder.cpp

FormBuilder::FormBuilder(QWidget \* parent, const QString & title)

{

    dlg = new QDialog(parent) ;

    dlg->setWindowTitle(title) ;

    vlayout = new QVBoxLayout() ;

    QDialogButtonBox \* box = new QDialogButtonBox() ;

    box->addButton("OK",QDialogButtonBox::ButtonRole::AcceptRole) ;

    box->addButton("Отмена",QDialogButtonBox::ButtonRole::RejectRole) ;

    QObject::connect(box,SIGNAL(rejected()),dlg,SLOT(reject())) ;

    QObject::connect(box,SIGNAL(accepted()),dlg,SLOT(accept())) ;

    box->setMinimumHeight(40) ;

    vlayout->addWidget(box,0,Qt::AlignBottom) ;

    dlg->setLayout(vlayout) ;

}

FormBuilder::~FormBuilder()

{

    delete dlg ;

}

QFormLayout \*FormBuilder::addFormLayout()

{

    QFormLayout \* layout = new QFormLayout() ;

    vlayout->insertLayout(0,layout) ;

    return layout ;

}

bool FormBuilder::exec()

{

    return dlg->exec()==QDialog::Accepted ;

}

Метод phone.cpp

Phone::Phone()

{

    str="0000000000" ;

    iscorrect=false ;

}

Phone::Phone(const Phone &src)

{

    str=src.str ;

    iscorrect = src.iscorrect ;

}

const QString DIGITS = "0123456789" ;

const int MIN\_PHONE\_LEN = 5;

const int MAX\_PHONE\_LEN = 7;

QString stripExceptDigits(const QString & str)

{

    QString res = "";

    foreach (auto c, str)

        if (DIGITS.indexOf(c)!=-1) res.append(c);

    return res;

}

Phone::Phone(const QString &src)

{

    iscorrect = true ;

    QString ph = stripExceptDigits(src) ;

    if (ph.length() == 10)

        str = ph ;

    else

        if ((ph.length() == 11) && ((ph[0]=='7')||(ph[0]=='8')))

            str = ph.mid(1);

        else

            if ((ph.length() >= MIN\_PHONE\_LEN) && (ph.length() <= MAX\_PHONE\_LEN))

                str = ph ;

            else

                iscorrect=false ;

}

QString Phone::getFormattedStr(const QString & phoneformat) const

{

    QString res = "";

    int p = 0;

    foreach (auto c, phoneformat)

        if (c=='X') {

            if (p>=str.length()) break ;

            res.append(str[p++]);

        }

            else res.append(c);

    return res;

}

bool Phone::isCorrect() const

{

    return iscorrect ;

}

Метод qphonetools.cpp

#include "qphonetools.h"

QPhoneTools::QPhoneTools(QObject \*parent) : QObject(parent)

{

}

bool QPhoneTools::isPhoneCorrect(const QString & phone) const

{

    Phone p(phone) ;

    return p.isCorrect() ;

}

QString QPhoneTools::formatPhone(const QString & phone, const QString & format) const

{

    Phone p(phone) ;

    return p.getFormattedStr(format) ;

}

Метод helper.cpp

Helper::Helper()

{

}

QString Helper::readAllText(const QString & filename, const QString & defvalue) {

    QFile file(filename);

    if (!file.exists()) return defvalue ;

    if (!file.open(QIODevice::ReadOnly)) return defvalue ;

    QString res = file.readAll() ;

    file.close();

    return res ;

}

bool Helper::writeAllText(const QString &filename, const QString &str)

{

    QFile file(filename);

    if (!file.open(QIODevice::WriteOnly)) return false ;

    file.write(str.toUtf8()) ;

    file.close();

    return true ;

}

QByteArray Helper::readAllBytes(const QString & filename) {

    QFile file(filename);

    if (!file.exists()) return QByteArray() ;

    if (!file.open(QIODevice::ReadOnly)) return QByteArray() ;

    QByteArray res = file.readAll() ;

    file.close();

    return res ;

}

bool Helper::writeAllBytes(const QString &filename, const QByteArray &data)

{

    QFile file(filename);

    if (!file.open(QIODevice::WriteOnly)) return false ;

    file.write(data) ;

    file.close();

    return true ;

}

bool Helper::confirm(QWidget \* parent, const QString &msg, const QString & header)

{

    QMessageBox msgbox(QMessageBox::Question,

                (header!="")?header:"Подтверждение",msg,

                QMessageBox::Yes | QMessageBox::No,

                parent);

    msgbox.setButtonText(QMessageBox::Yes, "Да");

    msgbox.setButtonText(QMessageBox::No, "Нет");

    return (msgbox.exec() == QMessageBox::Yes) ;

}