|  |
| --- |
| **Министерство образования Республики Беларусь**  **Учреждение образования**  **«Брестский государственный технический университет»**  **Кафедра ИИТ**  **ПОЯСНИТЕЛЬНАЯ ЗАПИСКА** К КУРСОВОМУ ПРОЕКТУ ПО ДИСЦИПЛИНЕ«Конструирование программного обеспечения» **Тема: «Разработка программы расчета стипендии»**  **КП.ПО-9.1-40-01-01**  **Листов:**  **Выполнил:**  Студент 2-го курса,  ФЭИС,  Группы ПО-9  Харитонович З. С.  **Нормоконтроль:**  Сааков А.В.  **Проверил:**  Сааков А.В.  **Брест 2023** |

# ВВЕДЕНИЕ

В современном мире вопрос учёта, хранения и обработки различных данных занимает очень важную роль. Хотя в мире существует большое множество способов решить этот вопрос, многие хотят иметь самый эффективный, совершенный и простой в использовании способ работы с данными. Простота достигается путём разработки интуитивно понятного графического интерфейса, совершенство достигается путём использования новейших технологий для работы с данными, эффективность достигается путём применения самых оптимальных способов работы с данными.

Поэтому нужно понимать, что главной целью работы с данными является понимание принципов работы не только новейших технологий, но и методы их применения. На данный момент существует множество способов работы с данными, но самым совершенным является база данных. База данных – очень мощный инструмент для работы с данными, благодаря ней можно производить любые операции над данными. Также не стоит забывать о применении графического интерфейса, без которого представить современное приложение не представляется возможным, – наглядного представления того, что происходит в результате действий над программой.

Таким образом данный курсовой проект направлен на глубокое исследование того, каким образом можно эффективно пользоваться технологией работы с данными посредством базы данных, а также на изучение того, каким образом можно преподнести простоту работы с базой данных посредством графического интерфейса.

1. **ПОСТАНОВКА ЗАДАЧИ**

Разработка программы расчета стипендии.

Для расчета стипендии необходимо обработать информацию следующего вида: номер группы, Ф.И.О. студента, форма обучения (платная/бюджетная), зачеты по пяти предметам (зачет/незачет), оценки по 4 предметам, признак участия в общественной работе: 1 – активное участие, 0 – неучастие.

Индивидуальное задание: рассчитать стипендию для студентов-бюджетников. При этом: отличники (отметки 9, 10) и общественники получают 50% надбавку, а просто отличники – 25%. Студенты со средним баллом, равным или ниже 5, стипендию не получают. Базовый размер стипендии вводится с клавиатуры.

Для разработки программы будут использоваться следующая технология: объектно-ориентированное программирование (ООП).

Средствами разработки программы будут: язык программирования Java, JavaFX – .

Среда разработки – JetBrains IntelliJ IDEA.

Тип используемой СУБД – MySQL.

1. **АНАЛИЗ ЗАДАЧИ**

## 2.1 РАЗРАБОТКА МОДУЛЬНОЙ СТРУКТУРЫ ПРОГРАММЫ

Первым этапом работы программы является авторизация. Для этого необходимо создать таблицу, содержащую сведения об учётных записях пользователей следующего вида: login, password, role. В качестве password хранится хеш-значение, а не исходное. Role равняется 1 для администратора и 0 для обычного пользователя. Изначально необходимо вручную записать в таблицу учётную запись администратора.

Вторым этапом работы программы является работа с таблицей данных, которая становится возможной после прохождения авторизации.

Модуль администратора включает следующие подмодули со следующими функциональными возможностями:

1. Управление учётными записями пользователей: просмотреть все учётные записи; добавить учётную запись; отредактировать учётную запись; удалить учётную запись.
2. Работа с данными: просмотреть все данные; добавить запись; удалить запись; редактировать запись; выполнить поиск данных; выполнить сортировку по различным полям; произвести расчет стипендии для студентов-бюджетников.
3. Модуль пользователя включает подмодуль работы с данными со следующими функциональными возможностями: просмотреть все данные; выполнить поиск данных; выполнить сортировку по различным полям; произвести расчет стипендии для студентов-бюджетников.

## ВЫБОР СПОСОБОВ ОПИСАНИЯ И ОБЪЕДИНЕНИЯ ВХОДНЫХ ДАННЫХ

Класс **User** – класс представления пользователя –содержит в себе поля:

* String **login –** логин пользователя;
* String **password** – пароль пользователя;
* boolean **role –** роль пользователя (true – администратор, false – обычный пользователь).

Класс **Student** – класс представления студента – содержит поля:

* int **id** – индентификатор;
* String **NSP** – фамилия, имя, отчество студента;
* String **group** – название группы студента;
* boolean **eduForm** – форма обучения студента (true – бюджетная, false – платная);
* boolean[] **credits** – зачеты студента (true – зачтено, false – не зачтено);
* int[] **exams** – оценки студента по экзаменам;
* boolean **socWork** – признак участия студента в общественной работе;
* double **scholarship** – размер стипендии студента.

Класс **Database** – класс, содержащий методы для работы с базой данных.

Класс **MainApplication** – класс приложения**.**

Класс **AuthenticationController** – модуль авторизации.

Класс **MenuController** – меню выбора подмодуля.

Класс **AccountManagerController** – подмодуль управления учётными записями модуля администратора.

Класс **ScholarshipCalculationController** – подмодуль работы с данными модуля пользователя.

Класс **ScholarshipAdminController** – подмодуль работы с данными модуля администратора (наследует **ScholarshipCalculationController).**

## РАЗРАБОТКА ПЕРЕЧНЯ ПОЛЬЗОВАТЕЛЬСКИХ ФУНКЦИЙ ПРОГРАММЫ

Класс **Database** содержит методы:

* ResultSet **sqlRequest**(String query) – запрос в базу данных (рис. 1);
* void **sqlUpdate**(String query) – внесение изменений в базу данных (рис. 2).

Класс **MainApplication** содержит методы:

* void **changeScene**(String file, String title, int v, int v1) – смена сцены (рис. 3);
* void **start**(Stage stage) – инициализация окна приложения (рис. 4).

Класс **AuthenticationController** содержит методы:

* void **onLoginButtonClick**() – авторизация (рис. 5);
* User **auth**(String login, String password) – проверка авторизации (рис. 6);
* String **stringToHash**(String str) – перевод строки в хеш (рис. 7).

Класс **MenuController** содержит методы:

* void **onAccountManagerButtonClick**() – переход к подмодулю управления учётными записями (рис. 8);
* void **onScholarshipCalculationButtonClick**() – переход к подмодулю работы с данными модуля пользователя (рис. 9);
* void **onExitButtonClick**() – выход из учётной записи, возврат к модулю авторизации (рис. 10).

Класс **AccountManagerController** содержит методы:

* void **onBackButtonClick**() – возврат к меню (рис. 11);
* void **updateTable**() – обновление содержимого таблицы пользователей (рис. 12);
* void **onAddButtonClick**() – добавление записи пользователя (рис. 13);
* void **onDeleteButtonClicked**() – удаление записи пользователя (рис. 14);
* void **onEditButtonClicked**() – редактирование записи пользователя (рис. 15).

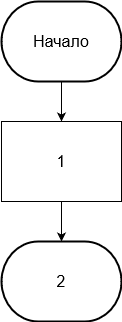
Класс **ScholarshipCalculationController** содержит методы:

* void **onBackButtonClick**() – возврат к меню (рис. 16);
* void **updateTable**() – обновление содержимого таблицы студентов (рис. 17);
* void **onSearchButtonClicked**() – поиск данных (рис. 18);
* void **onCalculateButtonClick**() – расчёт стипендии (рис. 19);
* void **onAdminButtonClicked**() – переход к подмодулю работы с данными модуля администратора (рис. 20).

Класс **ScholarshipAdminController** (наследует класс **ScholarshipCalculationController)** содержит методы:

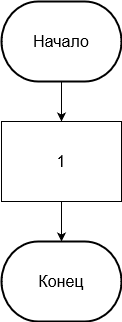
* void **onBackButtonClicked**() – возврат к подмодулю работы с данными модуля пользователя (рис. 21);
* void **onAddButtonClicked**() – добавление записи студента (рис. 22);
* void **onDeleteButtonClicked**() – удаление записи студента (рис. 23);
* void **onEditButtonClicked**() – редактирование записи студента (рис. 24).

# РАЗРАБОТКА АЛГОРИТМОВ РАБОТЫ ПОЛЬЗОВАТЕЛЬСКИХ ФУНКЦИЙ ПРОГРАММЫ



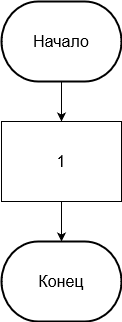
*Рисунок 1 – функция запроса в базу данных. (ResultSet sqlRequest(String query)*)

1. Подключение драйвера базы данных, установление соединения с базой данных, отправка SQL запроса, получение ответа от БД (набор данных).
2. Возврат набора данных.



*Рисунок 2 – функция внесения изменений в базу данных. (void sqlUpdate(String query))*

1. Подключение драйвера базы данных, установление соединения с базой данных, отправка SQL запроса.



*Рисунок 3 – функция смены сцены. (void changeScene(String file, String title,*

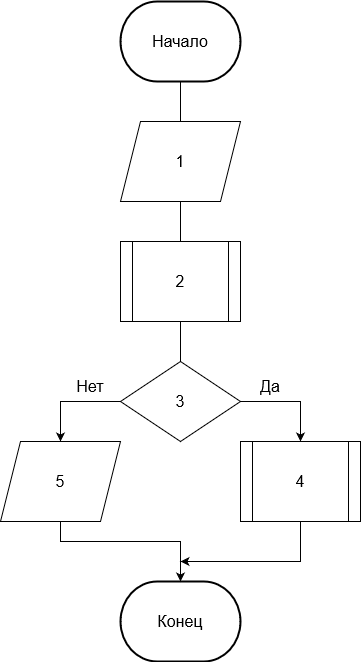
*int v, int v1))*

1. Изменение текущей сцены на **file**, изменение имени окна на **title**, установка размеров окна **v** \* **v1**.



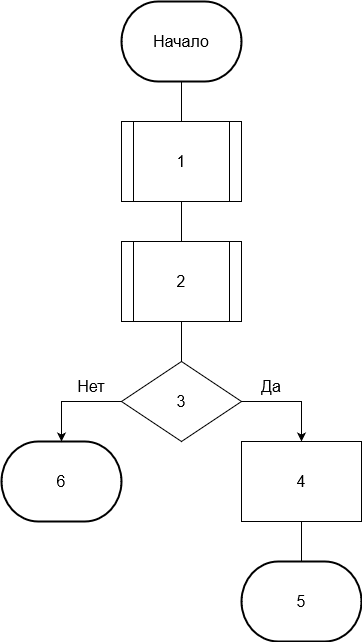
*Рисунок 4 – функция инициализации окна приложения. (void start(Stage stage))*

1. changeScene("authentication-view.fxml", "Авторизация", 320, 240)



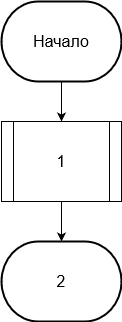
*Рисунок 5 – функция авторизации. (void onLoginButtonClick())*

1. Извлечение логина и пароля из формы.
2. Проверка авторизации (auth(login, password)).
3. Авторизация удалась (user != null)?
4. changeScene("menu-view.fxml", "Меню", 320, 240).
5. Вывод ошибки.



*Рисунок 6 – функция проверки авторизации. (User auth(String login, String password))*

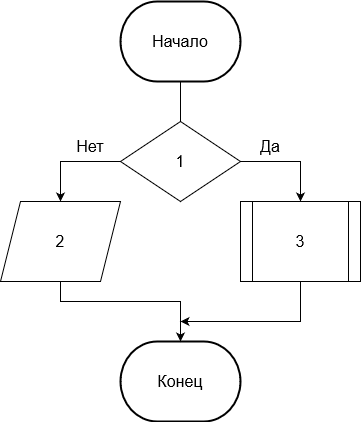
1. Хеширование пароля (stringToHash(password)).
2. Запрос записи пользователя login из базы данных (sqlRequest(query)).
3. Пароли совпадают?
4. Создание объекта user класса User на основе значений, полученных из БД.
5. Возврат объекта user.
6. Возврат null.



*Рисунок 7 – функция перевода строки в хеш.*

*(String stringToHash(String str))*

1. Хеширование строки по стандарту SHA-256, форматирование хеша в строку.
2. Возврат хеша.



*Рисунок 8 – функция перехода к подмодулю управления учётными записями. (void onAccountManagerButtonClick())*

1. Текущий пользователь является администратором?
2. Вывод ошибки.
3. changeScene("account-manager-view.fxml", "Управление учётными записями", 1280, 720).



*Рисунок 9 – функция переход к подмодулю работы с данными модуля пользователя. (void onScholarshipCalculationButtonClick())*

1. changeScene("scholarship-calculation-view.fxml", "Расчёт стипендии", 1280, 720).



*Рисунок 10 – функция выхода из учётной записи. (void onExitButtonClick())*

1. changeScene("authentication-view.fxml", "Авторизация", 320, 240).



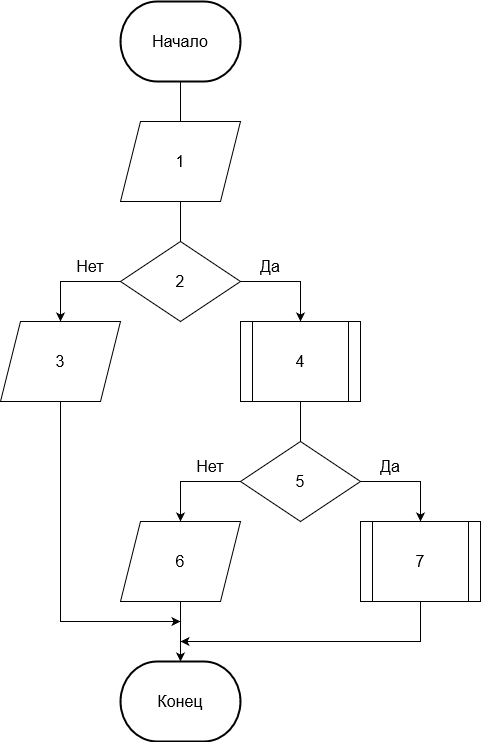
*Рисунок 11 – функция возврата к меню. (void onBackButtonClick())*

1. changeScene("menu-view.fxml", "Меню", 320, 240).



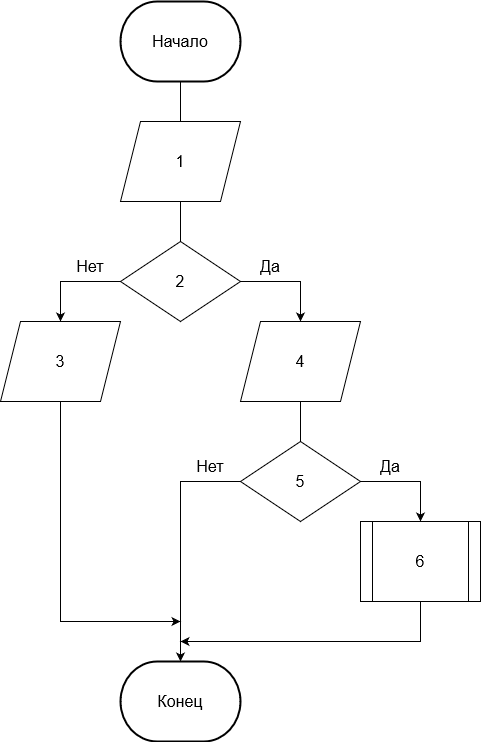
*Рисунок 12 – функция обновления содержимого таблицы пользователей. (void updateTable())*

1. Запрос записей всех пользователей из базы данных (sqlRequest(query)).
2. Перебор записей пользователей.
3. Создание объекта user класса User на основе значений, полученных из БД; добавление объекта user в массив данных data.
4. Вывод содержимого data на экран.



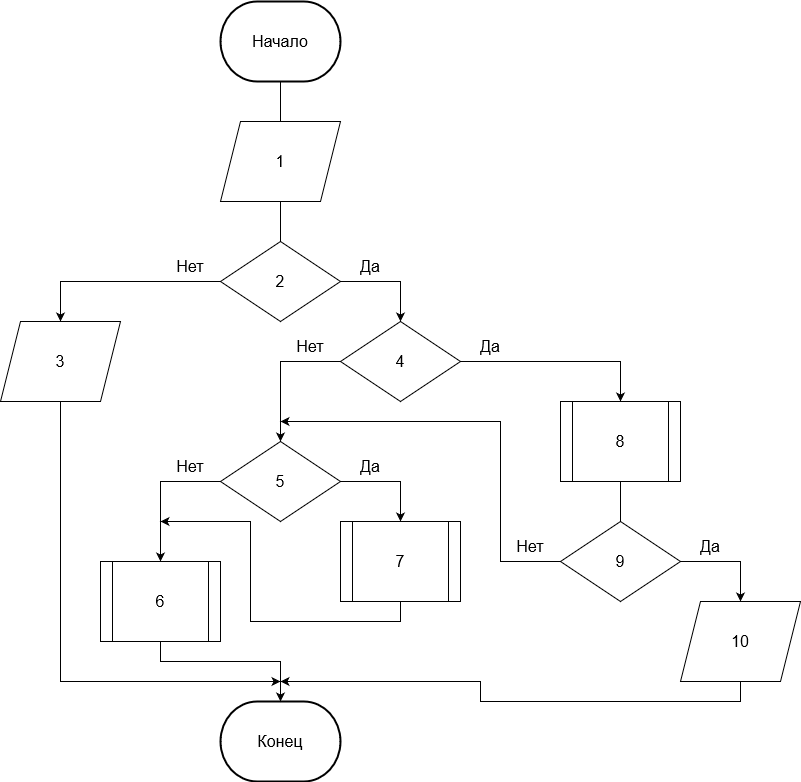
*Рисунок 13 – функция добавления записи пользователя. (void onAddButtonClick())*

1. Извлечение значений из полей формы.
2. Все поля заполнены?
3. Вывод ошибки (заполните все поля).
4. Запрос записи пользователя login из базы данных (sqlRequest(query)).
5. Запись не существует?
6. Вывод ошибки (такой пользователь уже существует).
7. Добавление записи в базу данных (sqlUpdate(query)); updateTable().



*Рисунок 14 – функция удаления записи пользователя. (void onDeleteButtonClicked())*

1. Извлечение выделенной в таблице записи.
2. Запись не пуста?
3. Вывод ошибки (выберите запись).
4. Запрос подтверждения операции.
5. Операция подтверждена?
6. Удаление записи из базы данных (sqlUpdate()); updateTable().



*Рисунок 15 – функция редактирования записи пользователя. (void onEditButtonClicked())*

1. Извлечений значений из полей формы.
2. Поля не пусты и запись в таблице выделена?
3. Вывод ошибки.
4. Логин был изменён?
5. Пароль был изменён?
6. Изменение записи в базе данных (sqlUpdate()).
7. Хеширование нового пароля.
8. Запрос записи пользователя newLogin из базы данных (sqlRequest(query)).
9. Запись не существует?
10. Вывод ошибки.



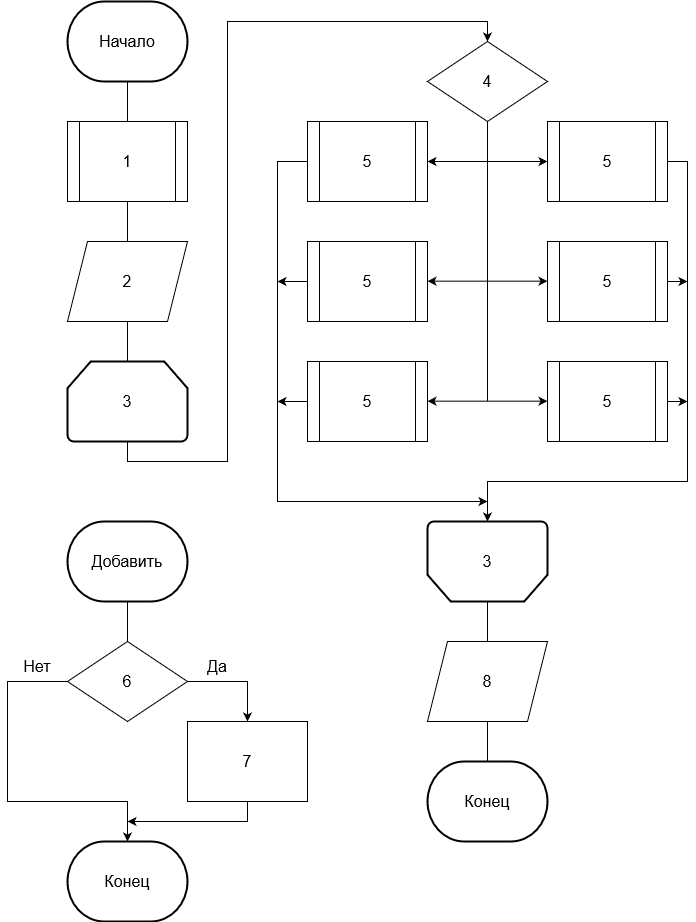
*Рисунок 16 – функция возврата к меню. (void onBackButtonClick())*

1. changeScene("menu-view.fxml", "Меню", 320, 240).



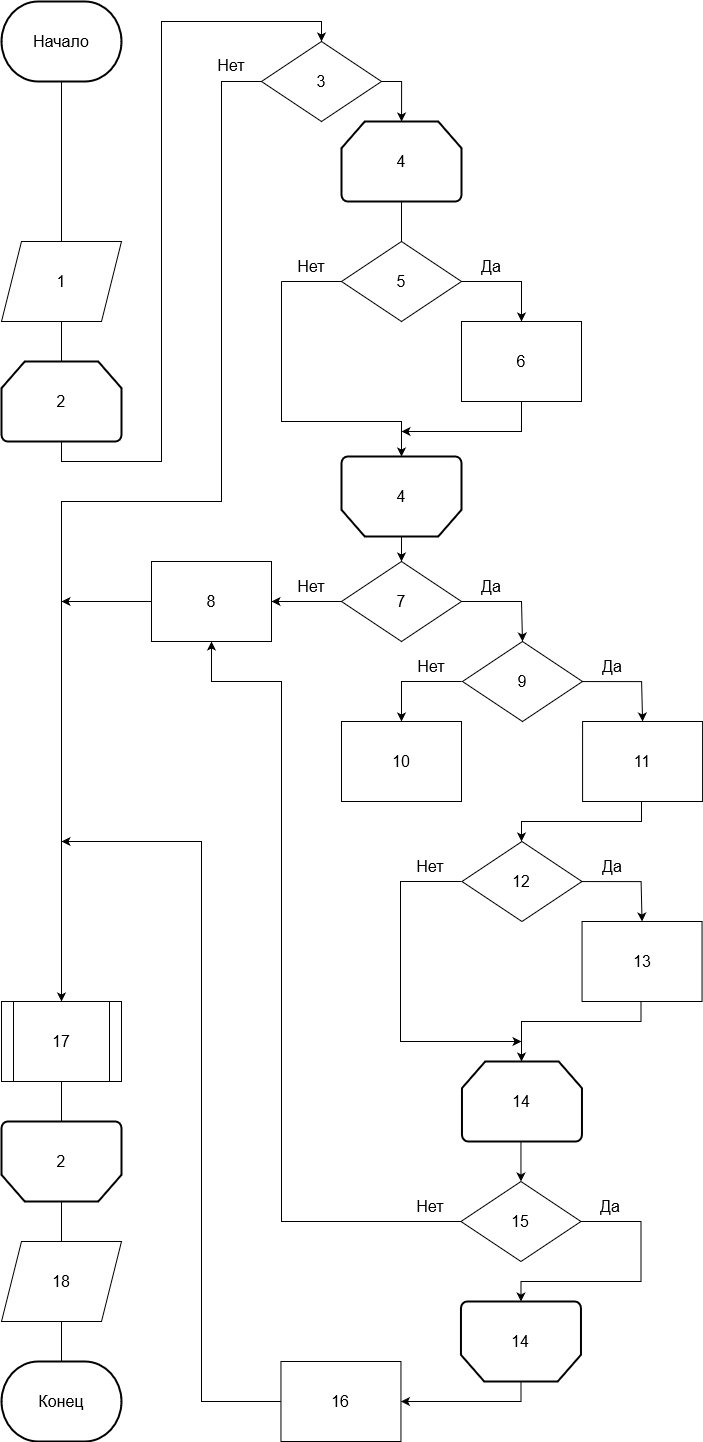
*Рисунок 17 – функция обновления содержимого таблицы студентов. (void updateTable())*

1. Запрос записей всех студентов из базы данных (sqlRequest(query)).
2. Перебор записей студентов.
3. Создание объекта student класса Student на основе значений, полученных из БД; добавление объекта student в массив данных data.
4. Вывод содержимого data на экран.



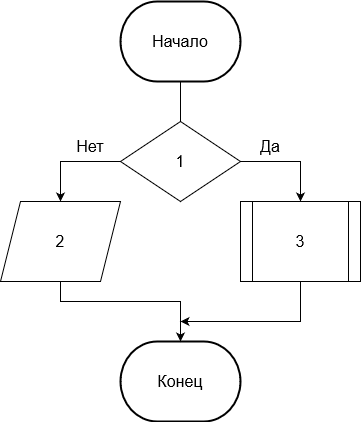
*Рисунок 18 – функция поиска данных. (void onSearchButtonClicked())*

1. updateTable().
2. Извлечение поискового запроса из формы.
3. Перебор всех студентов в таблице.
4. Поле, по которому осуществляется поиск.
5. Вызов «Добавить» для соответствующего поля.
6. Содержится ли поисковой запрос в значении соответствующего поля.
7. Добавить текущего студента в массив данных newData.
8. Вывод содержимого newData на экран.



*Рисунок 19 – функция расчёта стипендии. (void onCalculateButtonClick())*

1. Извлечение значения базового размера стипендии из формы.
2. Перебор всех студентов в таблице (data).
3. Текущий студент является бюджетником?
4. Перебор оценок по экзаменам.
5. Оценка ниже 9?
6. Студент не является отличником (isExc = false).
7. Средний балл не ниже 5?
8. Стипендия не выплачивается (scholarship = 0);
9. Студент является отличником?
10. Коэффициент k = 1.0
11. Коэффициент k = 1.25
12. Студент является общественником?
13. Коэффициент k = 1.5
14. Перебор зачётов.
15. Зачтено?
16. Установить стипендию в размере базовая \* k.
17. Изменение записи студента в базе данных.
18. Вывод содержимого обновлённого data на экран.



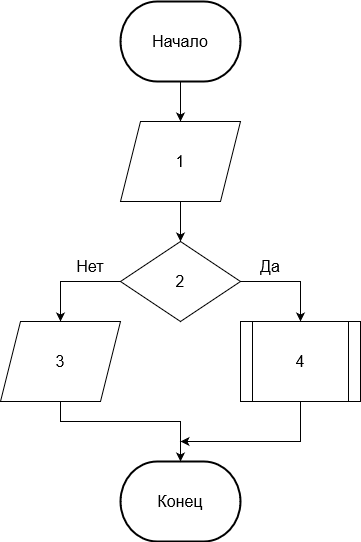
*Рисунок 20 – функция перехода к подмодулю работы с данными модуля администратора. (void onAdminButtonClicked())*

1. Текущий пользователь является администратором?
2. Вывод ошибки.
3. changeScene("scholarship-admin-view.fxml", "Расчёт стипендии", 1280, 720).



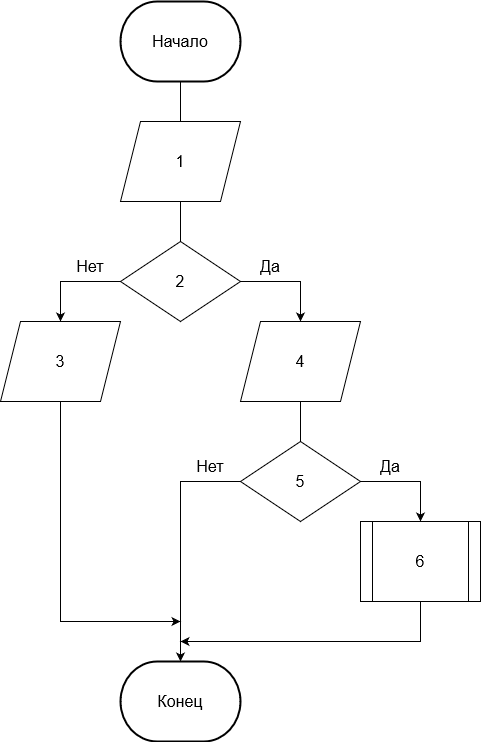
*Рисунок 21 – функция возврата к подмодулю работы с данными модуля пользователя. (void onBackButtonClicked())*

1. changeScene("scholarship-calculation-view.fxml", "Расчёт стипендии", 1280, 720).



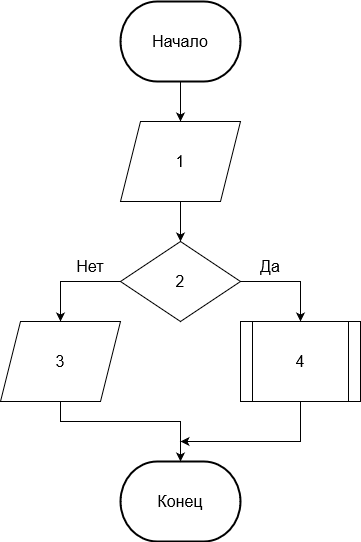
*Рисунок 22 – функция добавление записи студента. (void onAddButtonClicked())*

1. Извлечение значений из полей формы.
2. Поля не пусты?
3. Вывод ошибки.
4. Добавление записи студента в базу данных (sqlUpdate(query)).



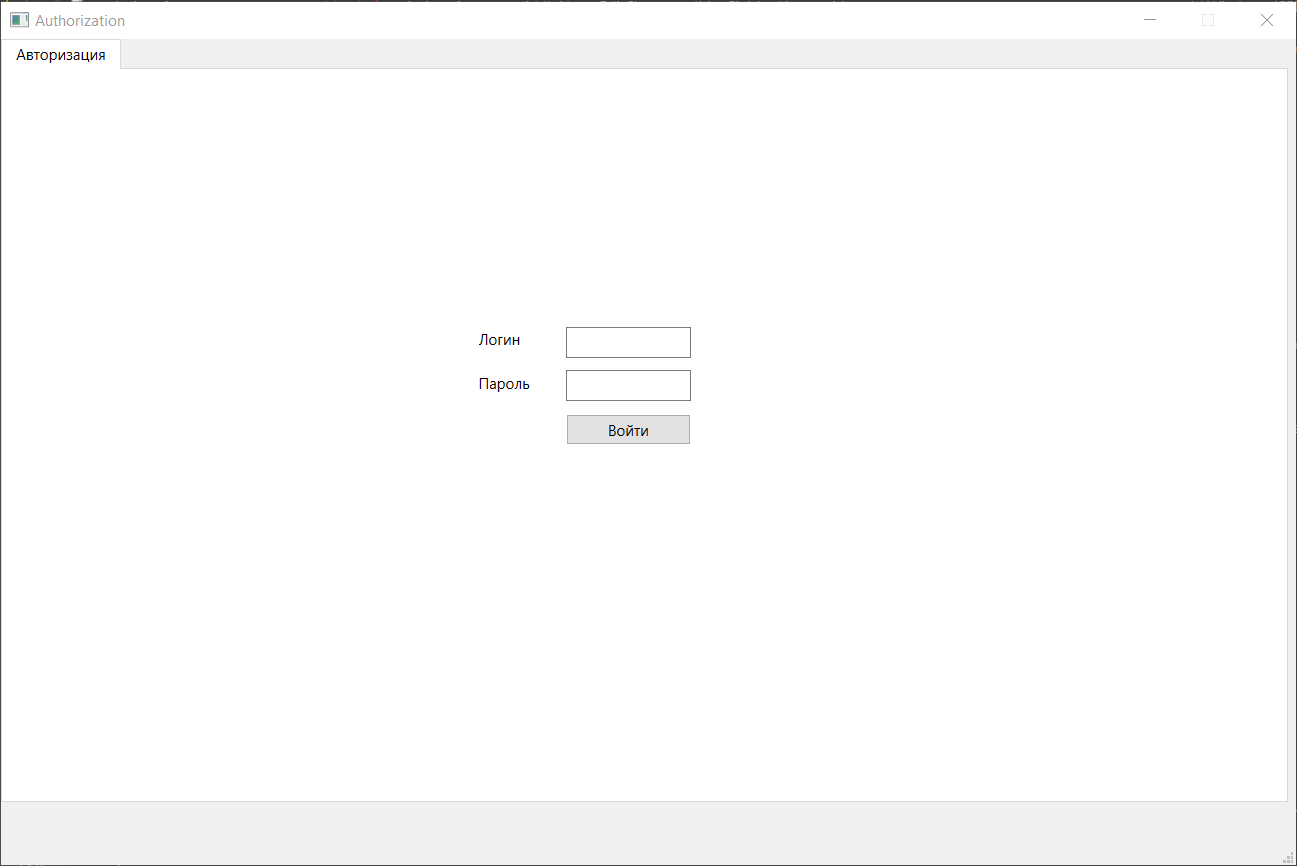
*Рисунок 23 – функция удаления записи студента. (void onDeleteButtonClicked())*

1. Извлечение выделенной в таблице записи.
2. Запись не пуста?
3. Вывод ошибки (выберите запись).
4. Запрос подтверждения операции.
5. Операция подтверждена?
6. Удаление записи студента из базы данных (sqlUpdate()); updateTable().



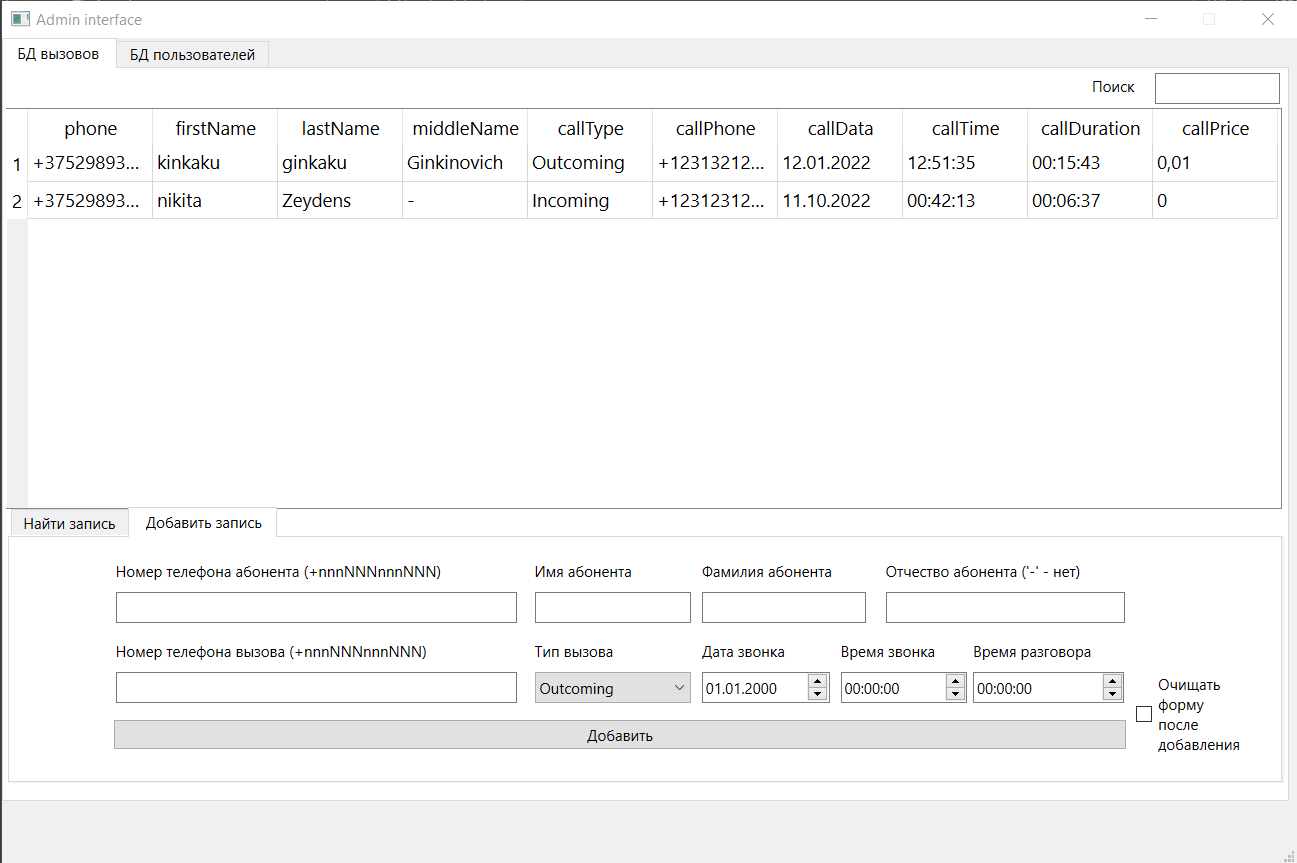
*Рисунок 24 – функция редактирования записи студента. (void onEditButtonClicked())*

1. Извлечений значений из полей формы.
2. Поля не пусты и запись в таблице выделена?
3. Вывод ошибки.
4. Изменение записи студента в базе данных (sqlUpdate()); updateTable().
5. **РАЗРАБОТКА И ОПИСАНИЕ РАБОТЫ ПРОГРАММЫ**

****

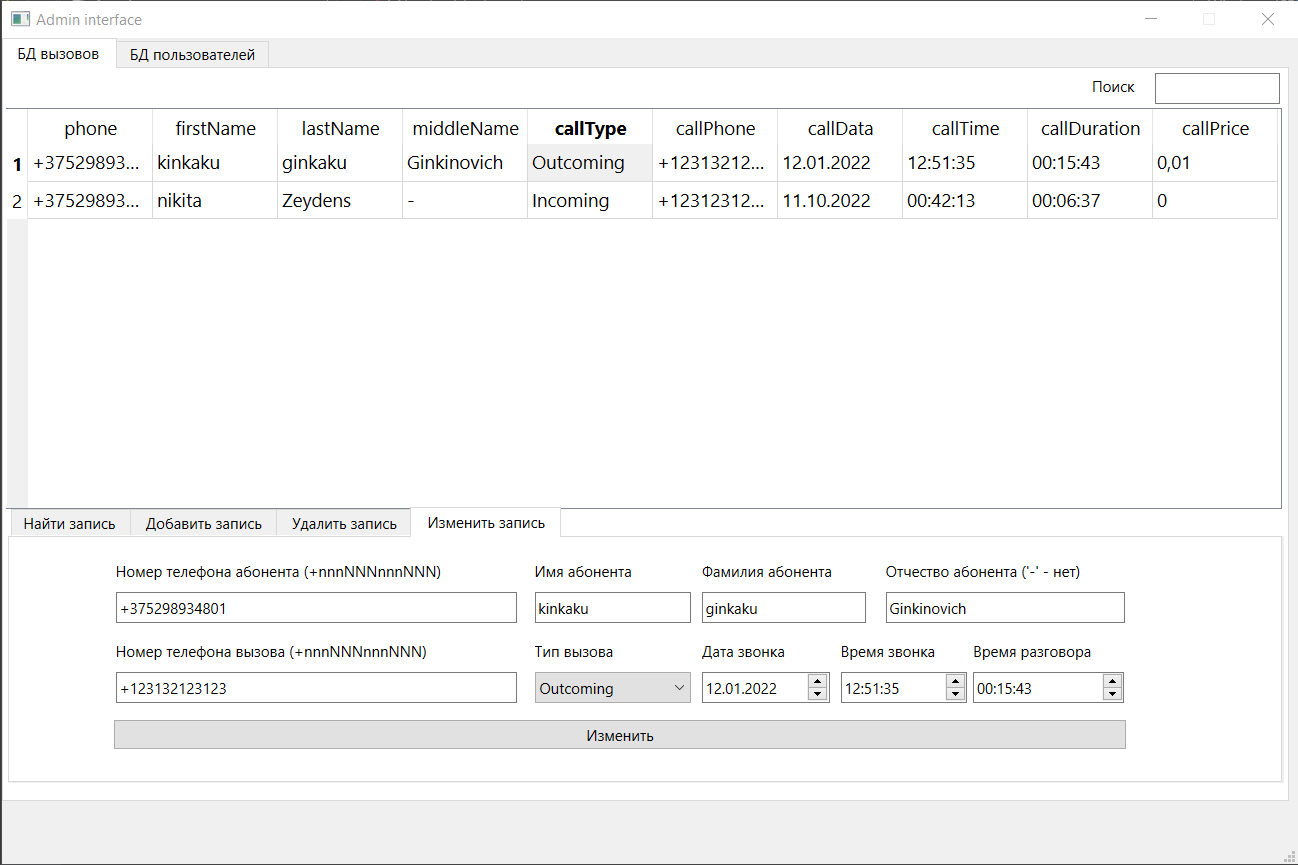
*Рисунок 23 - Окно авторизации*

При запуске программы пользователя встречает окно авторизации (рис. 23). В окне авторизации вводится логин и пароль, если в таблице есть запись с соответствующим логином и паролем, то определяем тип пользователя по третьему полю. Если в таблице нет записи с соответствующим логином или паролем, то высветится надпись о том, что такого пользователя нет. При неправильной записи какого-либо из полей, неправильное поле подсвечивается в любой из форм.



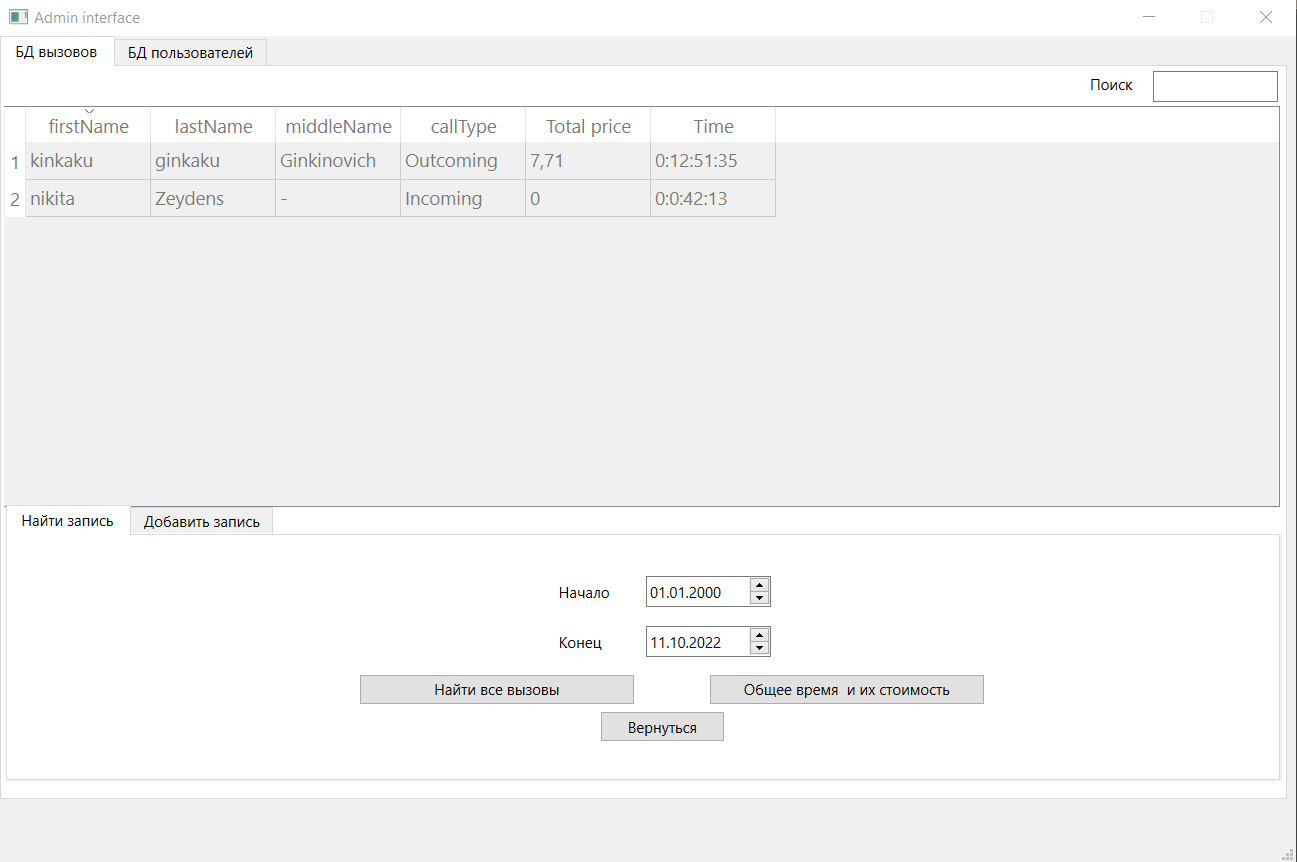
*Рисунок 24 - Интерфейс администратора*

Вторым этапом идёт работа с таблицей. Если пользователь – администратор, то ему открывается возможность добавлять, удалять, изменять, искать по периоду и искать по имени запись в таблице звонков. Также у администратора имеется возможность работы с таблицей пользователей, а именно добавлять, удалять, изменять и искать по логину запись в таблице пользователей (рис. 24).

****

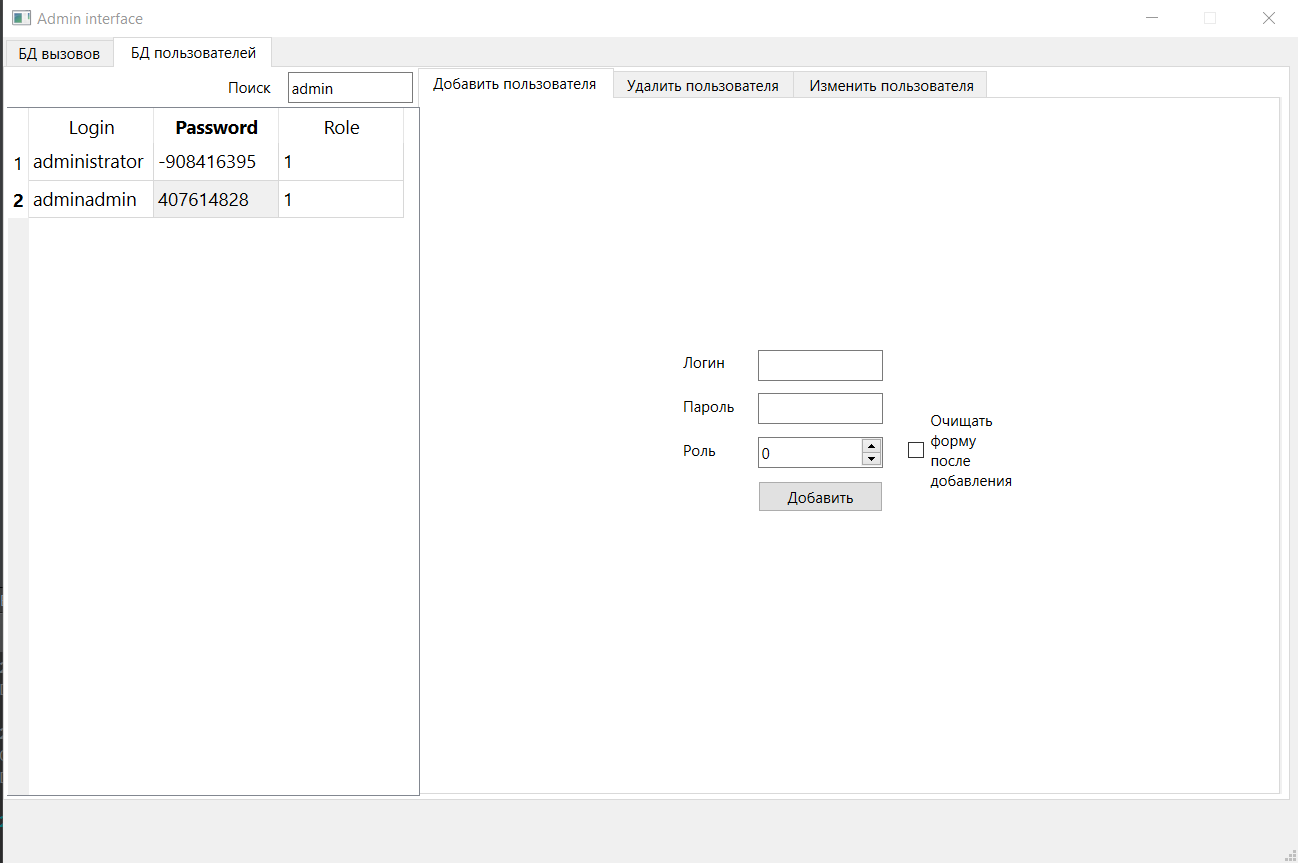
*Рисунок 25 - Выбор записи*

При двойном клике по записи у администратора появляются две возможности: изменить или удалить эту запись (рис. 25). В это время в две эти формы вносится соответствующая информация из столбцов записи в соответствующие поля формы. Далее можно выбрать действие, изменить или удалить эту запись, при изменении текущая запись перезаписывается с новыми данными, а при удалении вовсе удаляется из таблицы. Во время поиска по имени в таблице звонков или по логину в таблице пользователей также имеется возможность удалить или изменить выбранную запись.

****

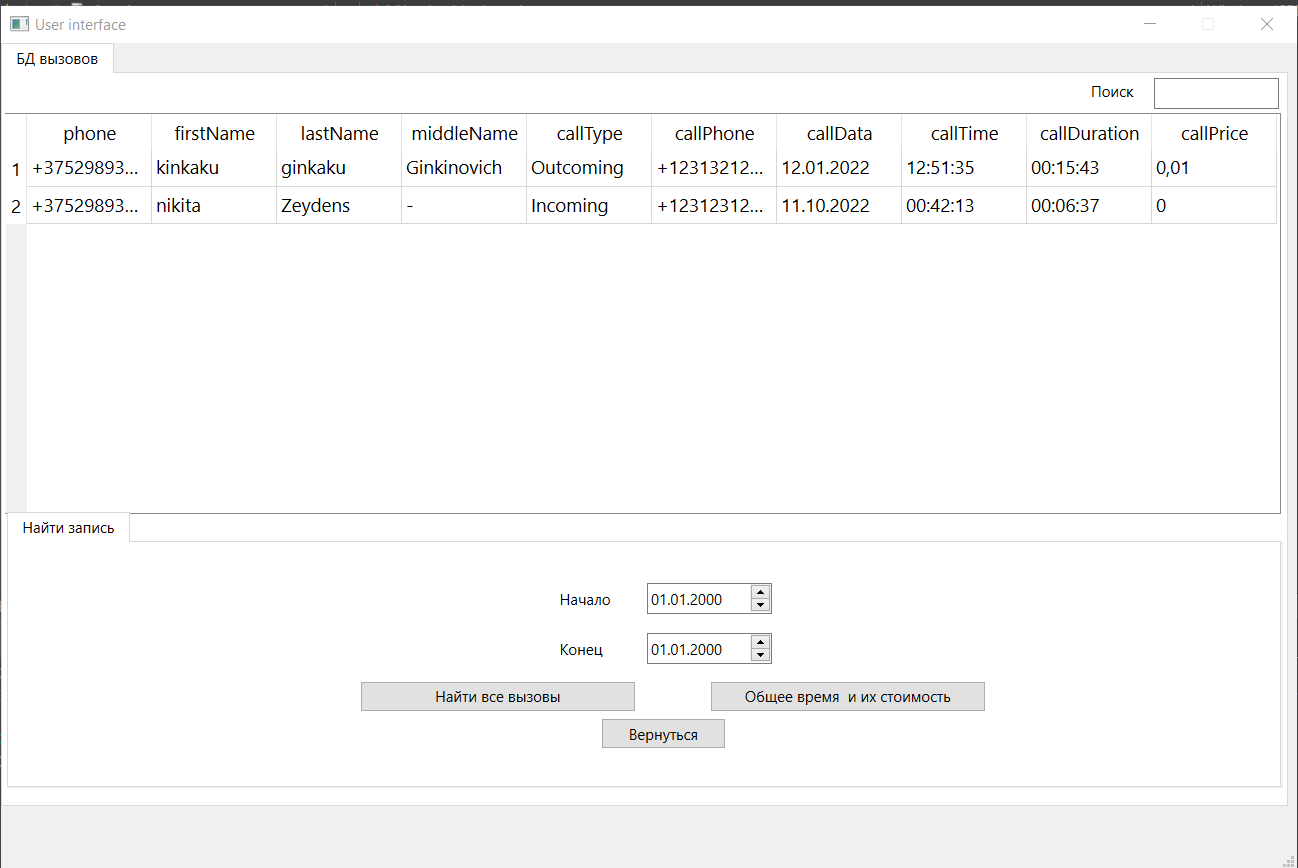
*Рисунок 26 - Поиск по периоду*

Далее рассмотрим работу поиска по периоду. В форме есть два поля ввода даты, по которым мы будем искать (рис. 26). Нажав на кнопку найти все вызовы, в таблице высветятся все записи, которые входят в этот период. Нажав на кнопку общее время и их стоимость в таблице появятся записи, входящие в этот период, сгруппированные по номеру телефона, ФИО и типу звонка, далее появится колонка общая стоимость – стоимость вызова, умноженная на общее время вызовов конкретного абонента в минутах. Во время поиска по периоду выбрать запись нельзя. По нажатии на кнопку вернуться, мы возвращаемся к начальной таблице звонков.

****

*Рисунок 27 - Работа поиска по таблице*

При вводе символов в строке поиска по таблице осуществляется поиск вхождения данного символа без учёта регистра (рис. 27).

****

*Рисунок 28 - интерфейс пользователя*

Если пользователь – не администратор, то у него имеется лишь возможность поиска по имени и по периоду (рис. 28).

**ЗАКЛЮЧЕНИЕ**

В результате выполнения курсовой работы было получено огромное количество опыта при работе с такими технологиями как БД и qt. По ходу выполнения курсовой работы пришлось столкнуться с большим количеством проблем и багов. Начиная с банального непонимания того, как работать с данными технологиями, заканчивая беглым исправлением новых багов.

На практике было получено знание о том, почему данные технологии применения БД при работе с различными типами данных и встраивание функций для работы с БД в графический интерфейс применяются в современном мире повсеместно и ежедневно.

Таким образом данная курсовая работа ещё раз подчёркивает важность обращения, хранения, изменения и корректирования данных посредством различных методов и способов, так как данные операции являются чуть ли не одними из ключевых современном мире.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Grant A. The Definitive Guide to SQLite / A. Grant, M. Owens – New-York: Apress, 2nd edition 2010. – 387с.
2. Lazar G. Mastering Qt 5: Create stunning cross-platform applications using C++ with Qt Widgets and QML with Qt Quick / G. Lazar., R. Penea – Birmingham: Packt Publishing; 2nd Revised edition 2018. – 534с.
3. Lippman S. C++ Primer / S. Lippman, J. Lajoie, B. Moo – Boston:  Addison-Wesley Professional; 5th edition 2012, - 976с.