Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

(ФГАОУ ВО ПНИПУ)

ЭЛЕКТРОТЕХНИЧЕСКИЙ ФАКУЛЬТЕТ

КАФЕДРА ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И АВТОМАТИЗИРОВАННЫЕ СИСТЕМЫ

УДК 004.7

ОТЧЕТ

ПО ТВОРЧЕСКОЙ РАБОТЕ

По дисциплине «Основы алгоритмизации

и программирования»:

СИСТЕМА АВТОМАТИЗАЦИИ КОНТРОЛЯ УЧЕБНОЙ ДЕЯТЕЛЬНОСТИ СТУДЕНТОВ И ВЫЯВЛЕНИЕ СТУДЕНТОВ-ЛИДЕРОВ

Выполнили:

Студент группы РИС-24-2б

Молочко А. А.

Студент группы РИС-24-2б

Носков Д. С.

Проверил:

Доцент кафедры ИТАС

Полякова О. А.

Пермь 2025

ОГЛАВЛЕНИЕ

1. Введение3
   1. Наименование программы3
2. Назначения и цели3
   1. Решаемые задачи3
   2. Целевая аудитория3
3. Технические особенности3
   1. Архитектура3
   2. Функционал3
4. Требования к системе4
   1. Требования к функциональным характеристикам4
      1. Требования к составу выполняемых функций4
      2. Требования к входным данным4
   2. Аппаратные требования6
      1. Клиентская часть6
      2. Серверная часть6
   3. Особенности реализации6
5. Оформление приложения6
6. Приложение А19
7. Приложение Б20
8. Приложение В21
9. Приложение Г22
10. Приложение Д23
11. Приложение Е25
12. Приложение Ж26
13. Приложение И28
14. Приложение К29
15. Приложение Л30
16. Приложение М31
17. Приложение Н35
18. Приложение П36
19. Приложение Р36
20. **Введение**
    1. **Наименование программы**Наименование программы: «Система автоматизации контроля учебной деятельности студентов и выявление студентов-лидеров». Далее по тексту – АРМ.

Краткая характеристика области применения  
На данный момент контроль успеваемости в высших учебных заведениях традиционно осуществляется в бумажной форме. Результаты лабораторных работ, практических заданий и экзаменов фиксируются вручную. Все данные заносятся в журналы успеваемости и оценки за работы выставляются в письменном виде.

1. **Назначение и цели**
   1. **Решаемые задачи**
      * Устранение влияния человеческого фактора.
      * Защита данных от потери или повреждения.
      * Обеспечение наглядности динамики успеваемости.
      * Сокращение времени анализа успеваемости.
      * Оптимизация поиска перспективных студентов.
   2. **Целевая аудитория**
      * Студенты.
      * Преподаватели.
2. **Технические особенности**
   1. **Архитектура**
      * Клиентская часть: Qt-приложение на C++.
      * Серверная часть: Локальный SQLite-база данных
   2. **Функционал**

|  |  |
| --- | --- |
| Модуль | Описание |
| Авторизация | Вход для студентов, лидеров и преподавателей. |
| Управление учебным процессом | Организация контроля за выполнением лабораторных и проверочных работ. |
| Тестирование студентов | Проведение тестов и сохранение ответов в базу данных. |
| Отслеживание успеваемости | Анализ успеваемости с графическим отображением данных. |
| Отправка файлов по email | Передача файлов пользователям через электронную почту |

1. **Требования к системе**
   1. **Требования к функциональным характеристикам**
      1. **Требования к составу выполняемых функций**

Система должна обеспечивать выполнение следующих функций:

1. **Учет пользователей**

* Регистрация студентов и преподавателей
* Редактирование профилей
* Просмотр списка пользователей

1. **Отслеживание успеваемости студентов**
   * Проверка сдачи лабораторных работ
   * Получение данных о проверочных работах
   * Графическое отображение успеваемости студентов
2. **Отправка файлов по email**
3. **Проведение тестирование студентов**
   * Проверка на лидера
   * Получение типа темперамента студента
     1. **Требования к входным данным (см. Таблицы 1, 2, 3, 4, 5, 6)**

**Таблица 1. Лабораторные работы**

|  |  |  |  |
| --- | --- | --- | --- |
| Параметр | Тип данных | Размер | Описание |
| id | ЦЕЛЫЙ | - | Уникальный идентификатор |
| student\_name | ТЕКСТ | 64 | Имя студента |
| lab\_name | ТЕКСТ | 64 | Название лабораторной работы |
| has\_report | ЦЕЛЫЙ | - | Наличие отчета  (0 – нет, 1-да) |
| has\_diagram | ЦЕЛЫЙ | - | Наличие диаграммы  (0 – нет, 1-да) |
| has\_code\_ | ЦЕЛЫЙ | - | Наличие кода  (0 – нет, 1-да) |

**Таблица 2. Результаты опроса**

|  |  |  |  |
| --- | --- | --- | --- |
| Параметр | Тип данных | Размер | Описание |
| id | ЦЕЛЫЙ | - | Уникальный идентификатор |
| full\_name | ТЕКСТ | 64 | Полное имя участника |
| total\_score | ЦЕЛЫЙ | - | Общий балл |
| temperament | ТЕКСТ | 32 | Тип темперамента |
| survey\_completed | ЦЕЛЫЙ | - | Опрос завершен (0 – нет, 1-да) |
| survey\_taken | ЦЕЛЫЙ | - | Опрос пройден  (0 – нет, 1-да) |

**Таблица 3. Тесты**

|  |  |  |  |
| --- | --- | --- | --- |
| Параметр | Тип данных | Размер | Описание |
| id | ЦЕЛЫЙ | - | Уникальный идентификатор |
| student\_name | ТЕКСТ | 64 | Имя студента |
| test\_topic | ТЕКСТ | 64 | Тема теста |
| grade | ТЕКСТ | - | Оценка |

**Таблица 4. Пользователи**

|  |  |  |  |
| --- | --- | --- | --- |
| Параметр | Тип данных | Размер | Описание |
| id | ЦЕЛЫЙ | - | Уникальный идентификатор |
| name | ТЕКСТ | 64 | Имя студента |
| password | ТЕКСТ | 64 | Пароль пользователя |
| groupname | ТЕКСТ | 64 | Название группы |
| role | ТЕКСТ | 32 | Роль пользователя |
| email | ТЕКСТ | 64 | Электронная почта |

**Таблица 5. Выдача лабораторных работ**

|  |  |  |  |
| --- | --- | --- | --- |
| Параметр | Тип данных | Размер | Описание |
| id | ЦЕЛЫЙ | - | Уникальный идентификатор |
| name | ТЕКСТ | 64 | Имя студента |
| is\_issued | ЛОГИЧЕСКИЙ | - | Флаг выдачи |

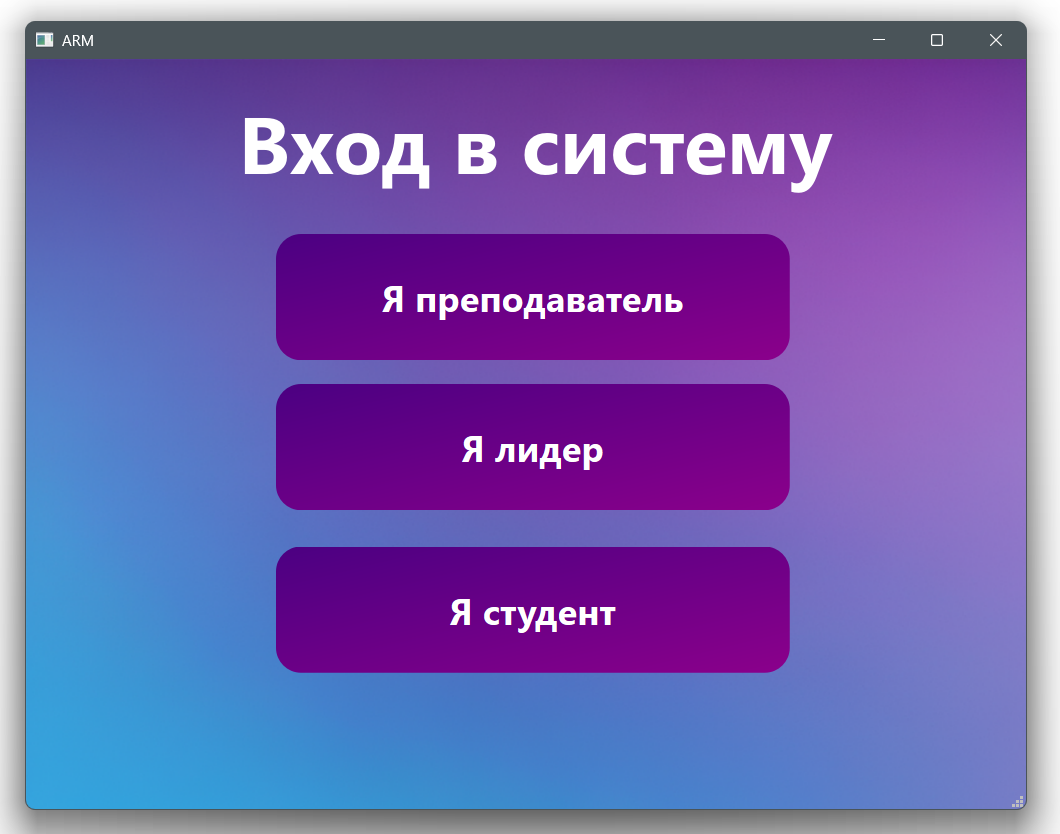
**Таблица 6. Sqlite последовательность**

|  |  |  |  |
| --- | --- | --- | --- |
| Параметр | Тип данных | Размер | Описание |
| name | ТЕКСТ | 64 | Имя таблицы |
| seq | ЦЕЛЫЙ | - | Текущие значение счетчика |

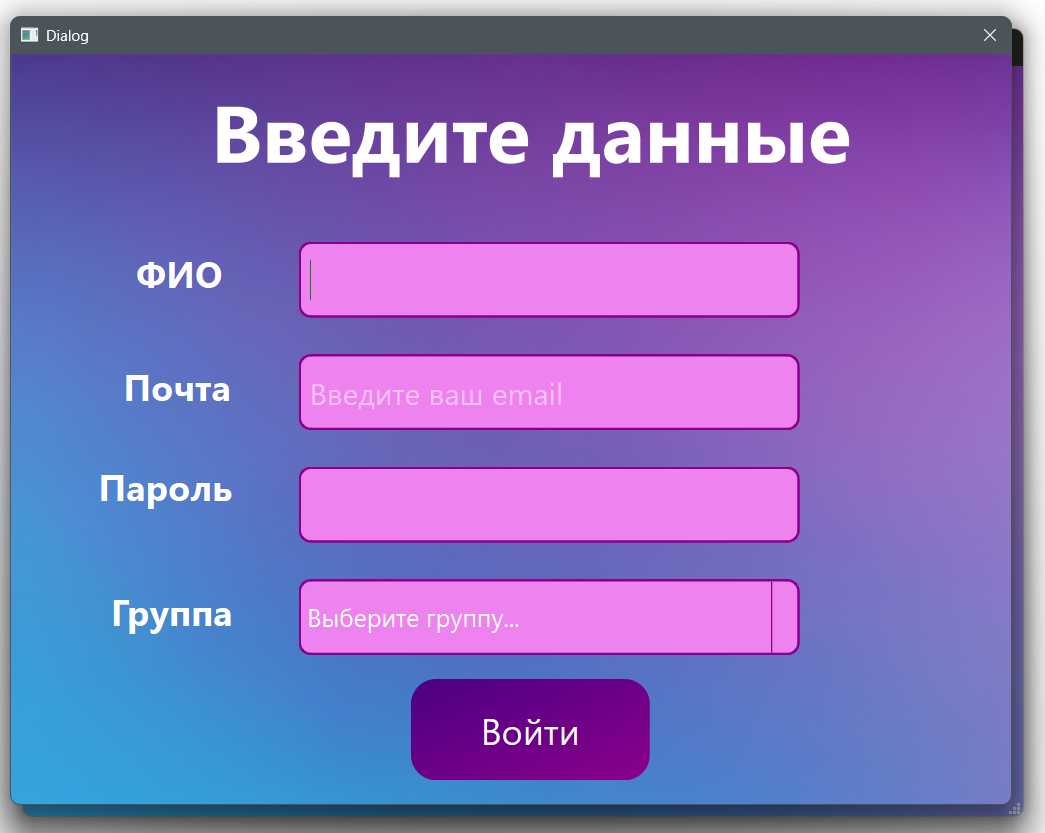
* 1. **Аппаратные требования**
     1. **Клиентская часть**
        + ОС: Windows 10/11
        + Процессор: x64-x86
        + ОЗУ: 4 Гб
     2. **Серверная часть**
        + СУБД: SQLite (Локально)
        + Хранилище: зависит от числа пользователей
  2. **Особенности реализации**
     + - Используемые технологии:
         * Язык программирования: С++17
         * Фреймворк: Qt 6
       - Использовались библиотеки:
         * Qcustomplot
         * SmtpClient

1. **Оформление приложения**

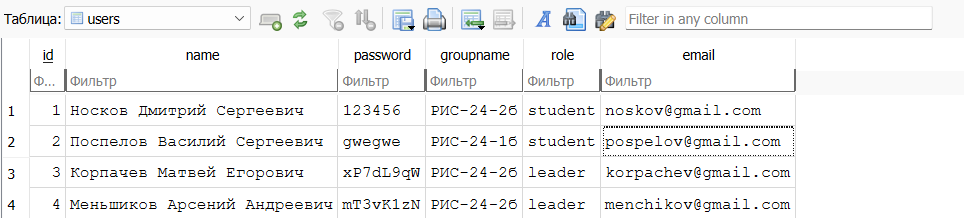
Когда приложение запускается, сначала отображается стартовый экран, на котором происходит распределение функций в соответствии с ролью пользователя в образовательном процессе.



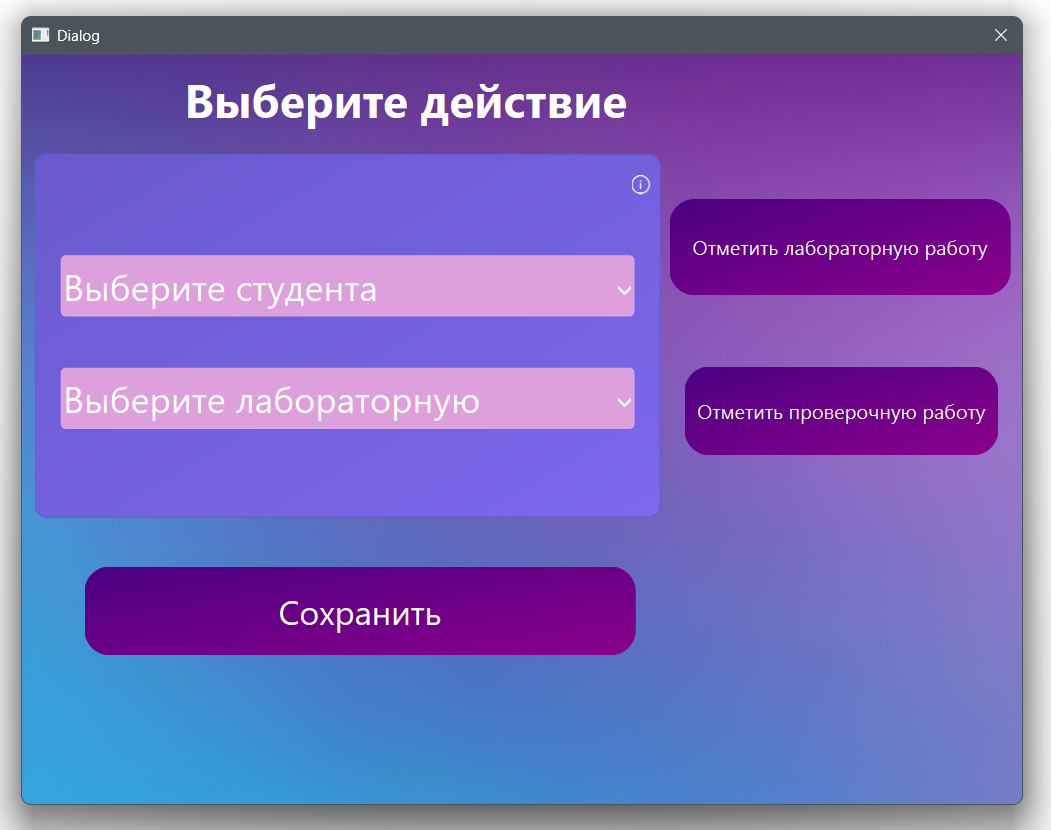
*Рисунок 1 – окно входа*

При нажатии на одну из кнопок мы попадаем на окно авторизации, где необходимо ввести ФИО, электронную почту (e-mail), пароль и указать группу обучения.

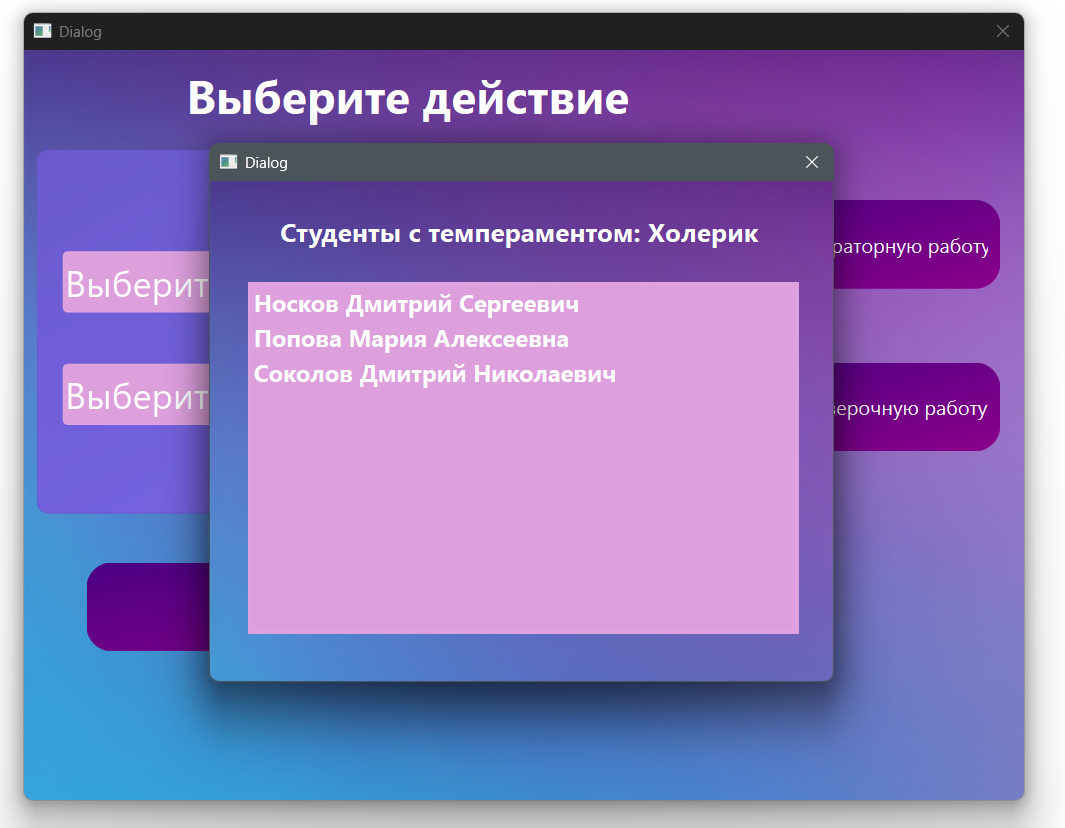
*Рисунок 2 – окно авторизации*

**

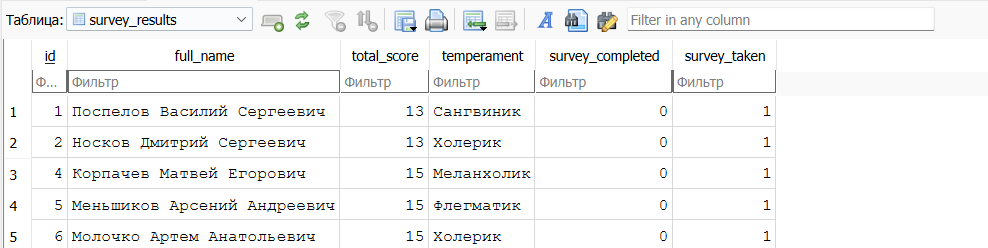
*Рисунок 3 – таблица с данными студентов*

На скриншоте представлен интерфейс системы управления учебным процессом, предназначенный для лидеров групп или преподавателей. Основное назначение окна — организация контроля за выполнением лабораторных и проверочных работ, включая механизмы отметки выполнения и выставления оценок. Интерфейс предоставляет инструменты для выбора студентов и соответствующих работ, а также сохранения результатов проверки.

*Рисунок 4 – окно лидера*

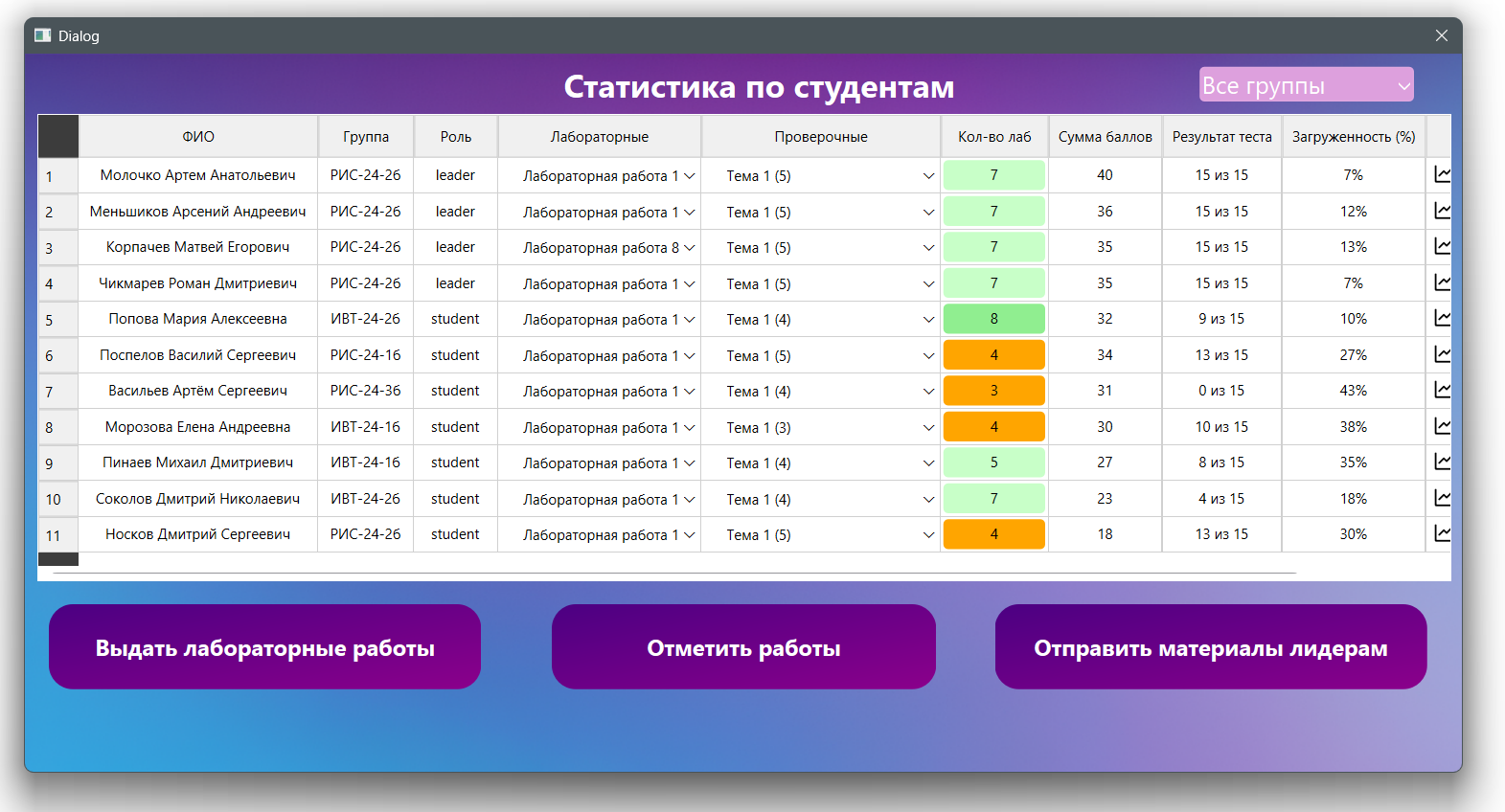
В данном окне реализована функция фильтрации студентов по темпераменту (в примере — "Холерик"). Система выводит список студентов, чей темперамент совпадает с темпераментом лидера, что упрощает распределение задач или формирование групп на основе психологической совместимости.

*Рисунок 5 – окно …*

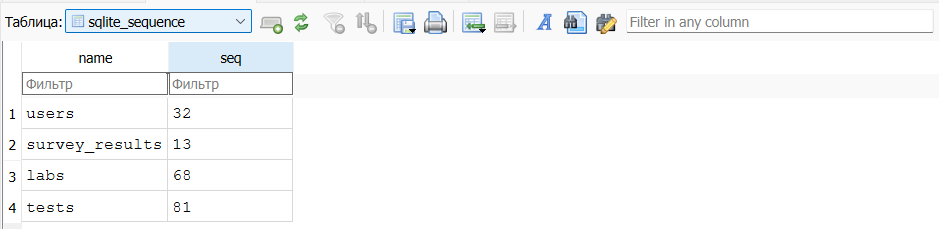
**

*Рисунок 6 – таблица, где храниться темперамент студента*

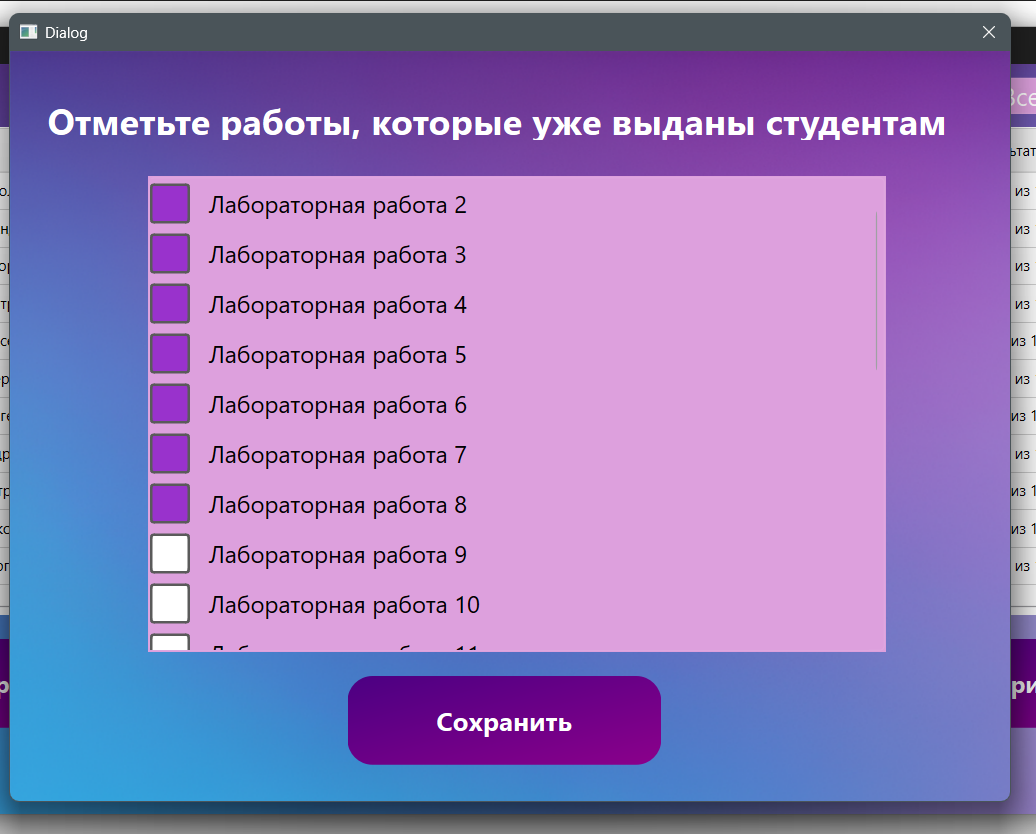
В окне преподавателя отображается сводная таблица с данными студентов (ФИО, группа, успеваемость) и их прогрессом по лабораторным/проверочным работам. Система показывает количество сданных работ, баллы, результаты тестов и уровень загруженности. Нижний блок содержит инструменты для распределения заданий, отметки выполненных работ и взаимодействия с лидерами групп, обеспечивая контроль учебного процесса.



*Рисунок 7 – окно преподавателя*

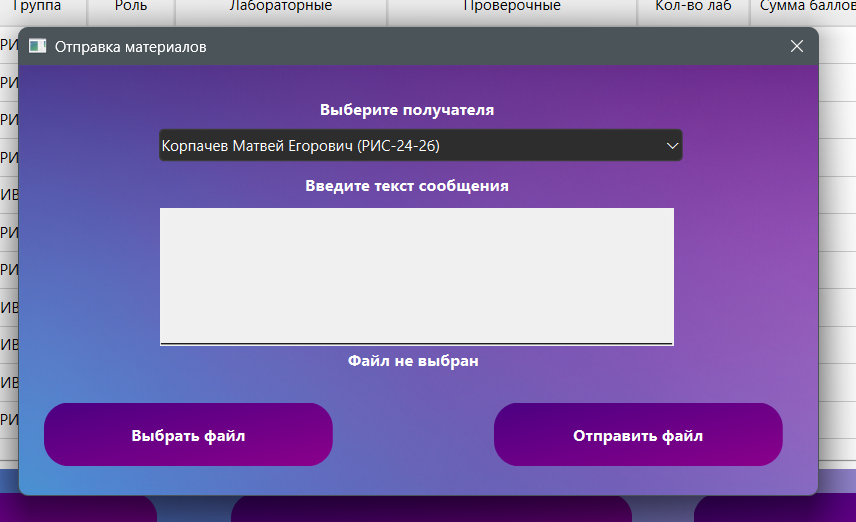
**

*Рисунок 8 – таблица, в которой храниться статистика*

Интерфейс предназначен для учета выданных лабораторных работ. Преподаватель может отметить уже распределенные задания и сохранить изменения. Функционал упрощает контроль за выполнением учебного плана.

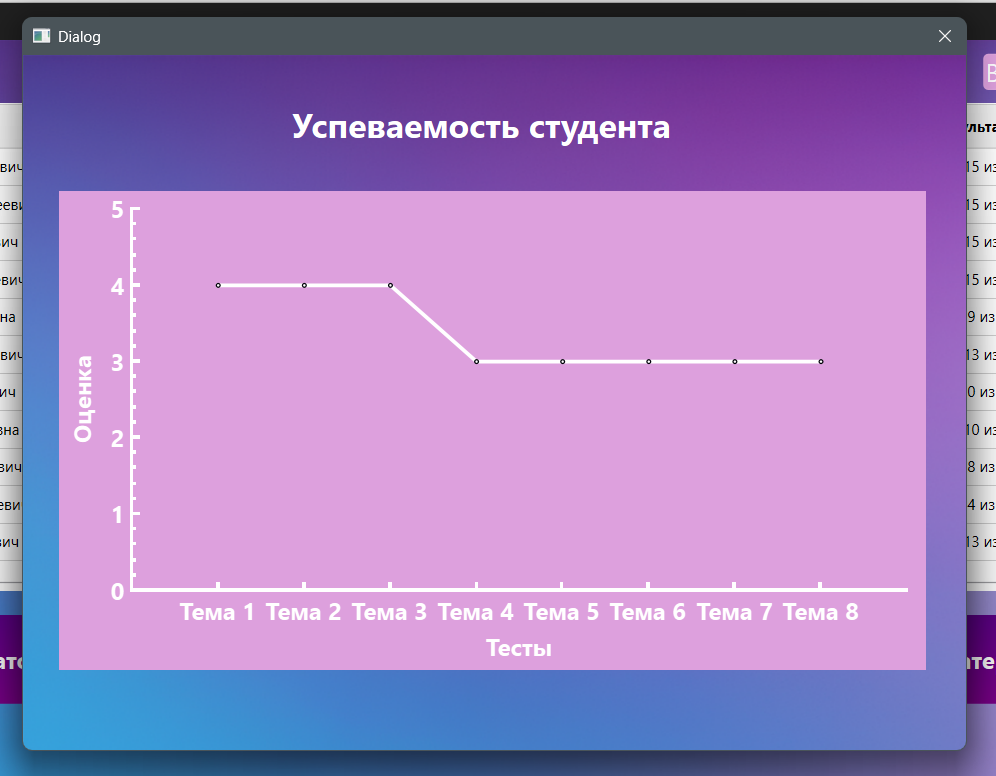
*Рисунок 9 – окно выдачи лабораторных работ*

В данном интерфейсе реализован функционал для отправки учебных материалов лидерам групп. Система позволяет выбрать конкретного получателя из списка, ввести сопроводительный текст, прикрепить необходимый файл через кнопку "Выбрать файл" и отправить материалы с помощью кнопки "Отправить файл". Этот инструмент обеспечивает эффективное взаимодействие между преподавателем и лидерами учебных групп, упрощая процесс распределения учебных материалов и заданий.

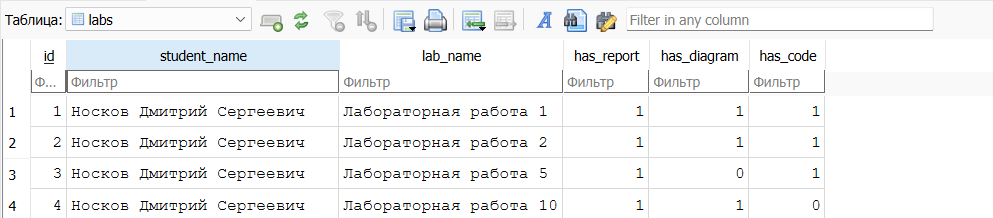


*Рисунок 10 – окно отправки материалов*

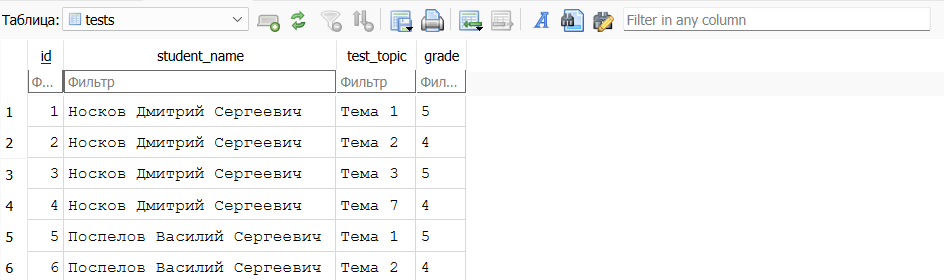
Окно отображает индивидуальную успеваемость студента по темам и тестам. В разделе представлены баллы за выполненные работы, отражающие текущий прогресс. Ниже перечислены темы тестов, по которым оценивается студент. Интерфейс предназначен для мониторинга академических результатов и выявления областей, требующих дополнительного внимания.



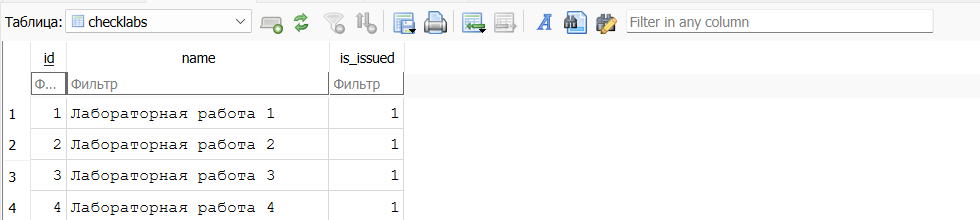
*Рисунок 11 – окно графика успеваемости*

**

*Рисунок 12 – таблица лабораторных работ*

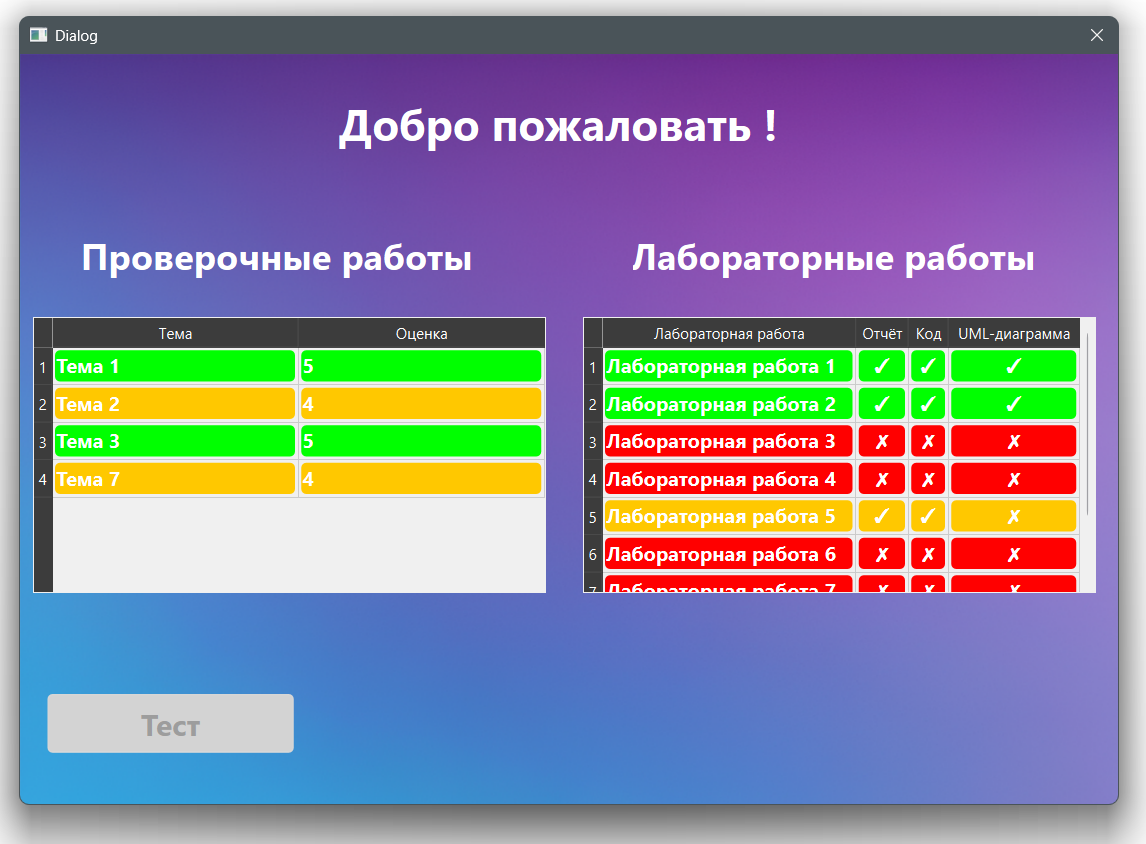
**

*Рисунок 13 – таблица проверочных работ*

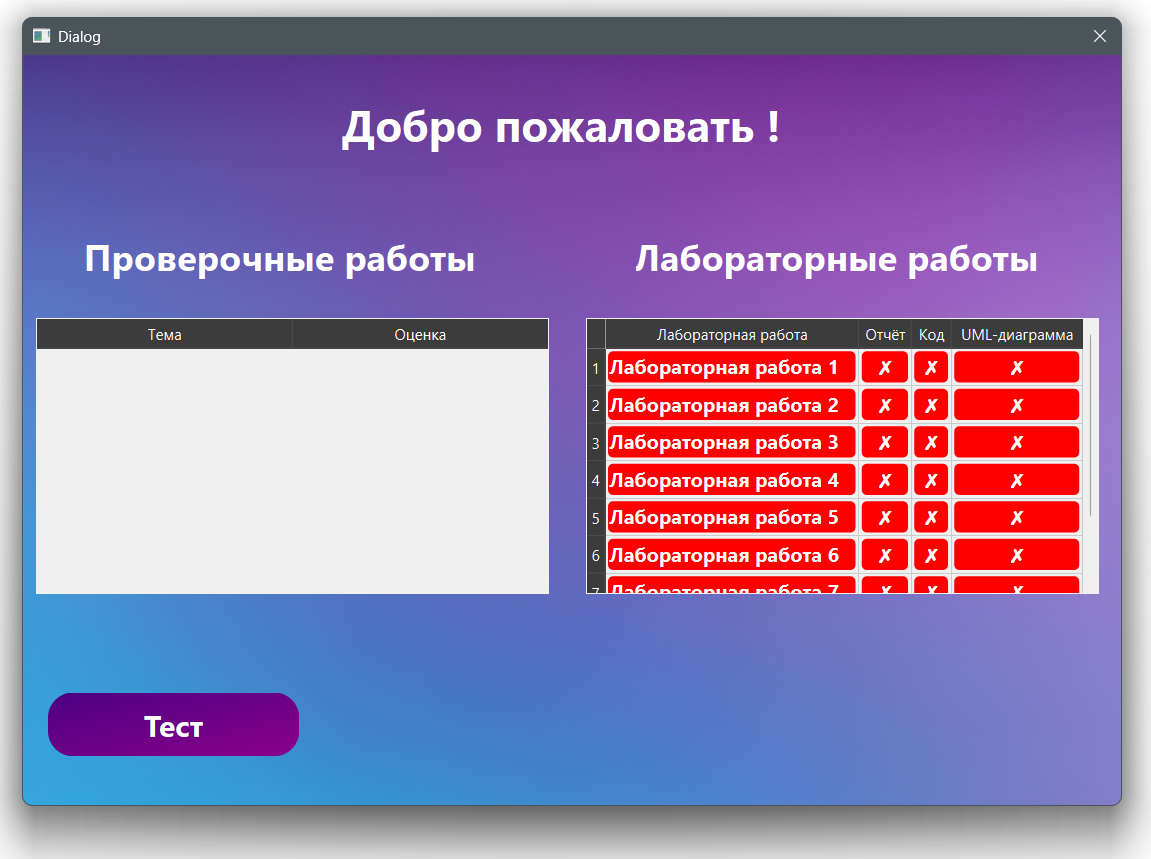


*Рисунок 14 – таблица проверка сдачи лабораторных работ*

В данном интерфейсе отображается персональная учебная статистика студента. Верхний блок содержит результаты проверочных работ по темам с оценками. Основная часть показывает выполнение лабораторных работ с отметками о сдаче отчётов, кода и UML-диаграмм. Внизу предусмотрена возможность прохождения теста. Интерфейс даёт студенту полный обзор своей успеваемости и текущих учебных задач.

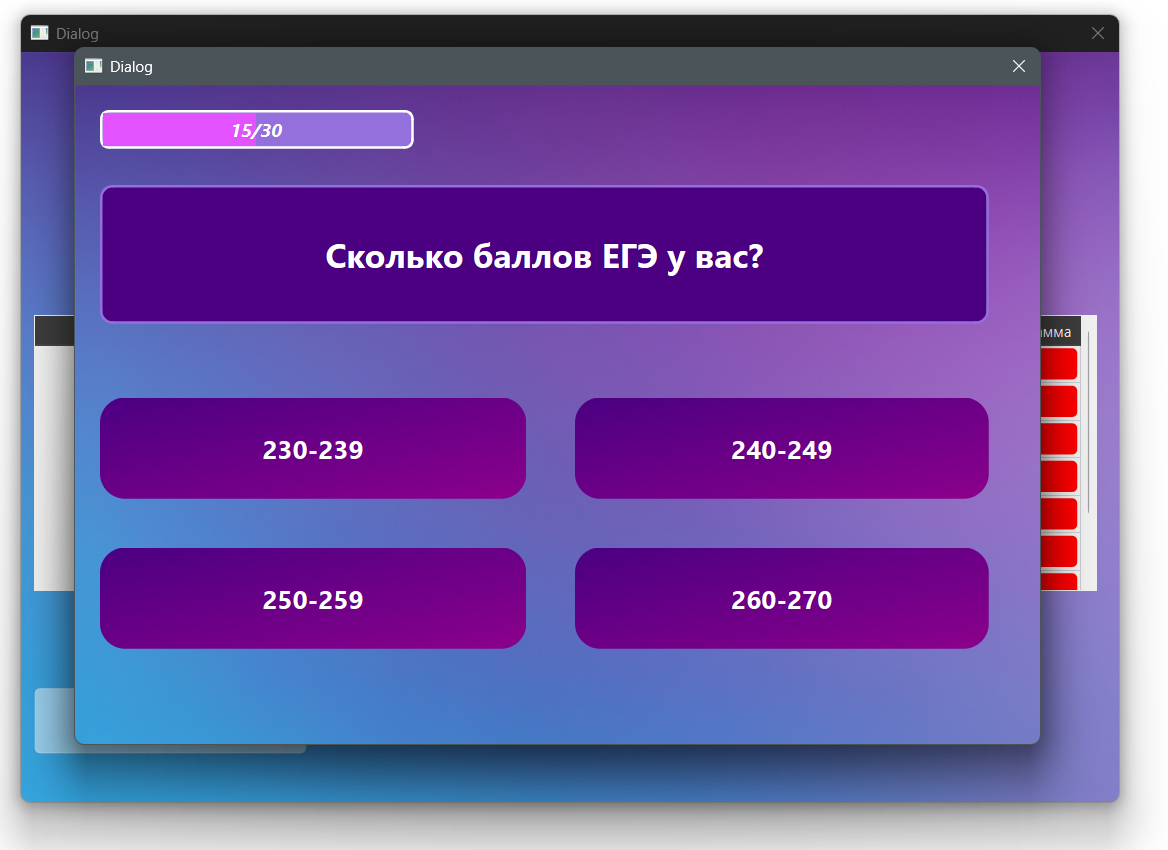


*Рисунок 15 – окно студента (кнопка тест отключена)*



*Рисунок 16 – окно студента (кнопка тест активна)*

На скриншоте представлен один из вопросов тестирования с вариантами ответов в формате диапазонов баллов ЕГЭ (230-239, 240-249, 250-259, 260-270). Вопрос "Сколько баллов ЕГЭ у вас?" предназначен для сбора информации об академических достижениях пользователей в рамках образовательного теста или анкетирования.



*Рисунок 17 – окно тестирования*

**ПРИЛОЖЕНИЕ А**

Файл arm.h

#ifndef ARM\_H

#define ARM\_H

#include <QMainWindow>

// Пространство имён UI, автоматически создаётся Qt Designer'ом

QT\_BEGIN\_NAMESPACE

namespace Ui {

class ARM;

}

QT\_END\_NAMESPACE

// Основной класс окна приложения, наследует QMainWindow

class ARM : public QMainWindow

{

Q\_OBJECT // Макрос Qt, необходимый для использования сигналов и слотов

public slots:

// Слоты для обработки нажатий кнопок

void on\_btnStudent\_clicked(); // Обработка нажатия кнопки "Студент"

void on\_btnLeader\_clicked(); // Обработка нажатия кнопки "Лидер"

void on\_btnTeacher\_clicked(); // Обработка нажатия кнопки "Преподаватель"

public:

// Конструктор и деструктор

ARM(QWidget\* parent = nullptr); // Конструктор с необязательным родителем

~ARM(); // Деструктор

// Оригинальные геометрии кнопок — используются для анимаций

QRect originalGeometryLeader;

QRect originalGeometryStudent;

QRect originalGeometryTeacher;

// Переопределённый метод для фильтрации событий (например, наведение мыши)

bool eventFilter(QObject\* obj, QEvent\* event) override;

// Метод для анимации изменения масштаба кнопки

void animateButtonScale(QWidget\* widget, qreal start, qreal end, const QRect& baseGeometry);

private:

Ui::ARM\* ui; // Указатель на интерфейс, сгенерированный Qt Designer

};

#endif // ARM\_H

**ПРИЛОЖЕНИЕ Б**

Файл givefromtecher.h

#ifndef GIVEFROMTEACHER\_H

#define GIVEFROMTEACHER\_H

#include <QDialog> // Для диалогового окна

#include <QSqlDatabase> // Для работы с базой данных

#include <QSqlQuery> // Для выполнения SQL-запросов

#include <QListWidgetItem> // Для работы с элементами списка

// Пространство имён для UI-класса, автоматически создаётся Qt Designer'ом

namespace Ui {

class GiveFromTeacher;

}

// Класс диалогового окна, предоставляющий интерфейс преподавателя

class GiveFromTeacher : public QDialog

{

Q\_OBJECT // Макрос Qt для поддержки сигналов и слотов

private slots:

// Слот, вызываемый при нажатии кнопки "Сохранить"

void on\_Save\_clicked();

public:

// Конструктор и деструктор

explicit GiveFromTeacher(QWidget\* parent = nullptr); // Конструктор с указанием родительского виджета

~GiveFromTeacher(); // Деструктор

private:

Ui::GiveFromTeacher\* ui; // Указатель на интерфейс, созданный в Qt Designer

QSqlDatabase db; // Объект подключения к базе данных

// Приватные методы

void initDB(); // Инициализация базы данных и создание таблицы, если необходимо

void loadLabs(); // Загрузка лабораторных работ из базы данных в список (QListWidget)

QRect originalGeometry; // Исходная геометрия виджета, используется для анимации

// Метод фильтрации событий (например, наведение мыши на элементы интерфейса)

bool eventFilter(QObject\* obj, QEvent\* event) override;

// Метод анимации масштабирования кнопки или виджета

void animateButtonScale(QWidget\* widget, qreal start, qreal end, const QRect& baseGeometry);

};

#endif // GIVEFROMTEACHER\_H

**ПРИЛОЖЕНИЕ В**

Файл global.h

#ifndef GLOBALS\_H

#define GLOBALS\_H

#include <QString> // Подключение класса QString для работы со строками

// Глобальные переменные, доступные во всей программе

extern QString g\_userName; // Глобальная переменная: имя пользователя

extern QString g\_userEmail; // Глобальная переменная: электронная почта пользователя

#endif // GLOBALS\_H

**ПРИЛОЖЕНИЕ Г**

Файл loginwindow.h

#ifndef LOGINWINDOW\_H

#define LOGINWINDOW\_H

#include <QDialog> // Для создания диалогового окна

#include <QLineEdit> // Для работы с полями ввода текста

// Пространство имён для автоматически сгенерированного класса UI от Qt Designer

namespace Ui {

class LoginWindow;

}

// Класс LoginWindow представляет окно авторизации пользователя

class LoginWindow : public QDialog

{

Q\_OBJECT // Макрос Qt, необходимый для системы сигналов и слотов

public slots:

// Слот, вызываемый при нажатии на кнопку "Войти"

void on\_btnLogin\_clicked();

public:

// Конструктор:

// - role: роль пользователя (например, студент, преподаватель и т.д.)

// - parent: родительский виджет

// - armWindow: указатель на главное окно (ARM), с которым связана авторизация

explicit LoginWindow(const QString& role, QWidget\* parent = nullptr, QWidget\* armWindow = nullptr);

// Деструктор

~LoginWindow();

private:

Ui::LoginWindow\* ui; // Указатель на интерфейс, созданный в Qt Designer

QString currentRole; // Текущая роль пользователя (используется для настройки интерфейса)

void setupUIForRole(); // Метод для настройки элементов UI в зависимости от роли

QWidget\* armWindow; // Указатель на основное окно (например, для возвращения после авторизации)

QRect originalGeometry; // Исходная геометрия кнопки, используется для анимации

// Переопределённый метод фильтрации событий (например, наведение и уход курсора)

bool eventFilter(QObject\* obj, QEvent\* event) override;

// Метод анимации изменения масштаба элемента (например, кнопки)

void animateButtonScale(QWidget\* widget, qreal start, qreal end, const QRect& baseGeometry);

};

#endif // LOGINWINDOW\_H

**ПРИЛОЖЕНИЕ Д**

Файл mainwindowleader.h

#ifndef MAINWINDOWLEADER\_H

#define MAINWINDOWLEADER\_H

#include <QDialog> // Для использования диалогового окна

#include <QComboBox> // Для выпадающих списков

// Пространство имён для класса интерфейса, сгенерированного Qt Designer

namespace Ui {

class MainWindowLeader;

}

// Класс, представляющий окно для пользователя с ролью "Лидер"

class MainWindowLeader : public QDialog

{

Q\_OBJECT // Макрос Qt, необходимый для системы сигналов и слотов

public slots:

void on\_btnLab\_clicked(); // Обработка нажатия кнопки "Лабораторная работа"

void on\_btnTest\_clicked(); // Обработка нажатия кнопки "Тест"

void on\_btnSave\_clicked(); // Обработка нажатия кнопки "Сохранить"

void onLabSelected(const QString& labName); // Слот, вызываемый при выборе лабораторной работы

void on\_btnShowMatchingTemperament\_clicked(); // Показать студентов с совпадающим темпераментом

public:

explicit MainWindowLeader(QWidget\* parent = nullptr); // Конструктор

~MainWindowLeader(); // Деструктор

// Оригинальные геометрии элементов интерфейса, используются для анимации

QRect originalGeometry;

QRect originalGeometryLab;

QRect originalGeometryTest;

QRect originalGeometryComboLab;

QRect originalGeometryComboLab2;

QRect originalGeometryComboTest1;

QRect originalGeometryComboTest2;

QRect originalGeometryComboTest3;

private:

Ui::MainWindowLeader\* ui; // Указатель на интерфейс окна

// Методы для заполнения выпадающих списков

void fillStudentComboBox(); // Заполнение списка студентов

void fillLabComboBox(); // Заполнение списка лабораторных работ

void fillTestComboBoxes(); // Заполнение списков тестов

// Метод фильтрации событий для анимации при наведении и т.п.

bool eventFilter(QObject\* obj, QEvent\* event) override;

// Анимация масштабирования кнопки

void animateButtonScale(QWidget\* widget, qreal start, qreal end, const QRect& baseGeometry);

// Анимация масштабирования комбобокса (выпадающего списка)

void animateComboBoxScale(QComboBox\* comboBox, qreal startScale, qreal endScale, const QRect& originalGeometry);

// Показ диалогового окна с чекбоксами (например, выбор выполнения лабораторной)

void showCheckboxDialog(const QString& studentName, const QString& labName);

};

#endif // MAINWINDOWLEADER\_H

**ПРИЛОЖЕНИЕ Е**

Файл mainwindowstudent.h

#ifndef MAINWINDOWSTUDENT\_H

#define MAINWINDOWSTUDENT\_H

#include <QDialog> // Для создания диалогового окна

#include <QTableWidgetItem> // Для работы с ячейками таблицы

#include <QRect> // Для хранения и работы с геометрией виджетов

#include "test.h" // Заголовок класса тестирования (предположительно, форма с вопросами и ответами)

QT\_BEGIN\_NAMESPACE

namespace Ui {

class MainWindowStudent;

}

QT\_END\_NAMESPACE

// Класс окна для студента, отображающего его прогресс, тесты и лабораторные работы

class MainWindowStudent : public QDialog

{

Q\_OBJECT // Макрос Qt для поддержки сигналов и слотов

public:

explicit MainWindowStudent(QWidget\* parent = nullptr); // Конструктор с необязательным родительским виджетом

~MainWindowStudent(); // Деструктор

protected:

// Метод фильтрации событий, например, для анимации при наведении

bool eventFilter(QObject\* watched, QEvent\* event) override;

private slots:

// Слот для открытия окна с тестом

void openTest();

private:

Ui::MainWindowStudent\* ui; // Указатель на интерфейс, сгенерированный Qt Designer

Test\* test; // Указатель на окно теста

QRect originalGeometry; // Исходная геометрия кнопки/элемента — для анимации

// Метод создания ячейки в таблице с отметкой о выполнении (галочка/крестик и т.д.)

QTableWidgetItem\* createStatusItem(bool done);

// Загрузка данных студента в интерфейс (например, оценки, выполненные задания)

void loadStudentData();

// Метод анимации кнопки (масштабирование при наведении или нажатии)

void animateButtonScale(QWidget\* widget, qreal start, qreal end, const QRect& baseGeometry);

};

#endif // MAINWINDOWSTUDENT\_H

**ПРИЛОЖЕНИЕ Ж**

Файл mainwindowteacher.h

#ifndef MAINWINDOWTEACHER\_H

#define MAINWINDOWTEACHER\_H

#include <QDialog> // Базовый класс для окна преподавателя

#include <QMainWindow> // (не используется напрямую, можно удалить, если не нужно)

#include <QTableWidget> // Таблица для отображения данных студентов

#include <QComboBox> // Выпадающий список (например, для выбора группы)

#include <QPushButton> // Кнопки интерфейса

#include <QStringList> // Список строк

#include <QList> // Общий список объектов

namespace Ui {

class MainWindowTeacher;

}

// Класс основного окна для роли преподавателя

class MainWindowTeacher : public QDialog

{

Q\_OBJECT // Макрос Qt для поддержки сигналов и слотов

public:

explicit MainWindowTeacher(QWidget\* parent = nullptr); // Конструктор

~MainWindowTeacher(); // Деструктор

private:

Ui::MainWindowTeacher\* ui; // Интерфейс, созданный в Qt Designer

// Основные элементы интерфейса

QTableWidget\* stats; // Таблица с данными о студентах

QPushButton\* btnGive; // Кнопка "Выдать лабораторные работы"

QPushButton\* btnMaterials; // Кнопка "Выдать методические материалы"

QComboBox\* groupFilter; // Фильтрация студентов по группе

// Оригинальные геометрии элементов, используются для анимации

QRect originalGeometry1;

QRect originalGeometry2;

QRect originalGeometry3;

QRect originalGeometrygroup;

// Методы настройки и логики интерфейса

void setupUI(); // Инициализация и настройка интерфейса

void populateTable(); // Заполнение таблицы студентами

void applyGroupFilter(); // Применение фильтрации по выбранной группе

void sortByRating(); // Сортировка таблицы студентов по рейтингу

// Анимация элементов интерфейса

bool eventFilter(QObject\* obj, QEvent\* event) override;

void animateButtonScale(QWidget\* widget, qreal start, qreal end, const QRect& baseGeometry);

void animateComboBoxScale(QComboBox\* comboBox, qreal startScale, qreal endScale, const QRect& originalGeometry);

private slots:

// Обработчики событий (слоты)

void onGiveClicked(); // Выдать лабораторные работы студентам

void onMaterialsClicked(); // Выдать методические материалы

void onGroupFilterChanged(); // Смена выбранной группы в фильтре

void onStudentClicked(int row, int column); // Клик по студенту в таблице

void onLabsClicked(); // Клик по кнопке "Лабораторные"

void openGraphForStudent(const QString& studentName); // Открыть график успеваемости конкретного студента

};

#endif // MAINWINDOWTEACHER\_H

**ПРИЛОЖЕНИЕ И**

Файл performancegraphwindow.h

#ifndef PERFORMANCEGRAPHWINDOW\_H

#define PERFORMANCEGRAPHWINDOW\_H

#include <QDialog> // Для создания диалогового окна

#include "qcustomplot.h" // Библиотека для построения графиков (https://www.qcustomplot.com)

namespace Ui {

class PerformanceGraphWindow;

}

// Класс окна, отображающего график успеваемости конкретного студента

class PerformanceGraphWindow : public QDialog

{

Q\_OBJECT // Макрос Qt, необходимый для системы сигналов и слотов

public:

// Конструктор: принимает имя студента и опциональный родительский виджет

explicit PerformanceGraphWindow(const QString& studentName, QWidget\* parent = nullptr);

// Деструктор

~PerformanceGraphWindow();

private:

Ui::PerformanceGraphWindow\* ui; // Интерфейс окна, созданный в Qt Designer

// Метод построения графика успеваемости по имени студента

void drawGraph(const QString& studentName);

};

#endif // PERFORMANCEGRAPHWINDOW\_H

**ПРИЛОЖЕНИЕ К**

Файл sendmailwindow.h

#ifndef SENDEMAILWINDOW\_H

#define SENDEMAILWINDOW\_H

#include <QDialog> // Для создания модального окна

#include <QRect> // Для хранения геометрии кнопок (нужно для анимации)

namespace Ui {

class SendEmailWindow;

}

// Класс окна отправки письма с приложенным файлом

class SendEmailWindow : public QDialog

{

Q\_OBJECT // Макрос Qt, необходимый для работы сигналов и слотов

public:

// Конструктор с необязательным родительским виджетом

explicit SendEmailWindow(QWidget\* parent = nullptr);

// Деструктор

~SendEmailWindow();

protected:

// Переопределение фильтра событий (для анимации при наведении)

bool eventFilter(QObject\* watched, QEvent\* event) override;

private slots:

// Слот для выбора файла (по кнопке "Выбрать файл")

void onSelectFile();

// Слот для отправки письма (по кнопке "Отправить")

void onSendClicked();

private:

// Метод отправки письма на указанный адрес с прикреплённым файлом

void sendEmail(const QString& recipientEmail, const QString& filePath);

// Анимация кнопок при наведении/нажатии

void animateButtonScale(QWidget\* widget, qreal start, qreal end, const QRect& baseGeometry);

// Загрузка списка лидеров (адресатов)

void loadLeaders();

private:

Ui::SendEmailWindow\* ui; // Интерфейс, сгенерированный Qt Designer

QString selectedFilePath; // Путь к выбранному файлу

QRect originalGeometrySend; // Оригинальная геометрия кнопки "Отправить"

QRect originalGeometrySelect;// Оригинальная геометрия кнопки "Выбрать файл"

};

#endif // SENDEMAILWINDOW\_H

**ПРИЛОЖЕНИЕ Л**

Файл temperamentdialog.h

#ifndef TEMPERAMENTDIALOG\_H

#define TEMPERAMENTDIALOG\_H

#include <QDialog> // Базовый класс для модального окна

namespace Ui {

class TemperamentDialog;

}

// Диалоговое окно для отображения списка студентов по типу темперамента

class TemperamentDialog : public QDialog

{

Q\_OBJECT // Макрос Qt, необходимый для сигналов и слотов

public:

// Конструктор: принимает тип темперамента, список студентов и опциональный родительский виджет

explicit TemperamentDialog(const QString& temperament, const QStringList& students, QWidget\* parent = nullptr);

// Деструктор

~TemperamentDialog();

private:

Ui::TemperamentDialog\* ui; // Указатель на интерфейс, созданный в Qt Designer

};

#endif // TEMPERAMENTDIALOG\_H

**ПРИЛОЖЕНИЕ М**

Файл test.h

#ifndef TEST\_H

#define TEST\_H

#include <QDialog>

#include <QVector>

#include <QStringList>

#include <QSqlDatabase>

#include <QSqlQuery>

#include <QSqlError>

#include <QVariant>

#include <QEvent>

namespace Ui {

class Test;

}

// Класс диалогового окна для проведения теста

class Test : public QDialog {

Q\_OBJECT

public:

// Конструктор с указанием родительского виджета по умолчанию

explicit Test(QDialog\* parent = nullptr);

// Деструктор

~Test();

// Исходные геометрии для анимации или позиционирования виджетов (4 штуки)

QRect originalGeometry1;

QRect originalGeometry2;

QRect originalGeometry3;

QRect originalGeometry4;

signals:

// Сигнал, который посылается при завершении теста

void testCompleted();

private slots:

// Слот, вызываемый при нажатии кнопки ответа

void onAnswerButtonClicked();

// Слот, вызываемый при нажатии кнопки закрытия окна

void onCloseButtonClicked();

private:

Ui::Test\* ui; // Указатель на UI, сгенерированный из файла .ui

// Текущий прогресс прохождения теста (например, процент или номер шага)

int currentProgress = 0;

// Индекс текущего вопроса

int currentQuestion = 0;

// Количество правильных ответов, набранных пользователем

int correctAnswersCount = 0;

// Дополнительные оценки, например для ЕГЭ или мотивации (предположительно для аналитики)

int egeScore = 0;

int motivationScore = 0;

// Счётчики для определения типа темперамента

int sanguine = 0;

int choleric = 0;

int melancholic = 0;

int phlegmatic = 0;

// Структура для хранения данных вопроса

struct QuestionData {

QString questionText; // Текст вопроса

QStringList answers; // Варианты ответов

int correctAnswer; // Индекс правильного ответа, -1 если нет правильного (для вопросов без однозначного правильного ответа)

};

QVector<QuestionData> testQuestions; // Список всех вопросов теста

// Метод для инициализации данных теста (заполнение тестовых вопросов)

void initializeTestData();

// Обновление интерфейса для отображения текущего вопроса

void updateQuestion();

// Создание таблицы в базе данных для сохранения результатов

void createTable();

// Сохранение результатов теста в базу данных, используя имя пользователя и количество правильных ответов

void saveResultsToDatabase(const QString& fullName, int correctAnswersCount);

// Метод для вычисления типа темперамента на основе собранных данных

QString calculateTemperamentType() const;

// Переопределение фильтра событий для отслеживания событий (например, для анимаций или кастомного поведения)

bool eventFilter(QObject\* obj, QEvent\* event) override;

// Анимация масштабирования кнопки или другого виджета с заданной геометрией

void animateButtonScale(QWidget\* widget, qreal start, qreal end, const QRect& baseGeometry);

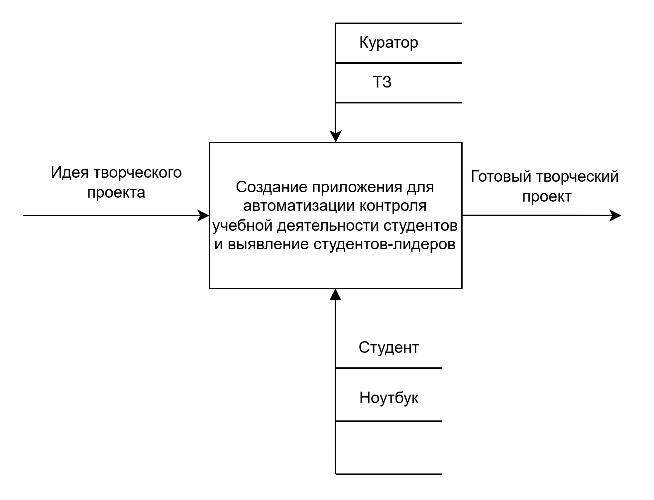
// Показ финального слайда с результатами теста

void showFinalSlide();

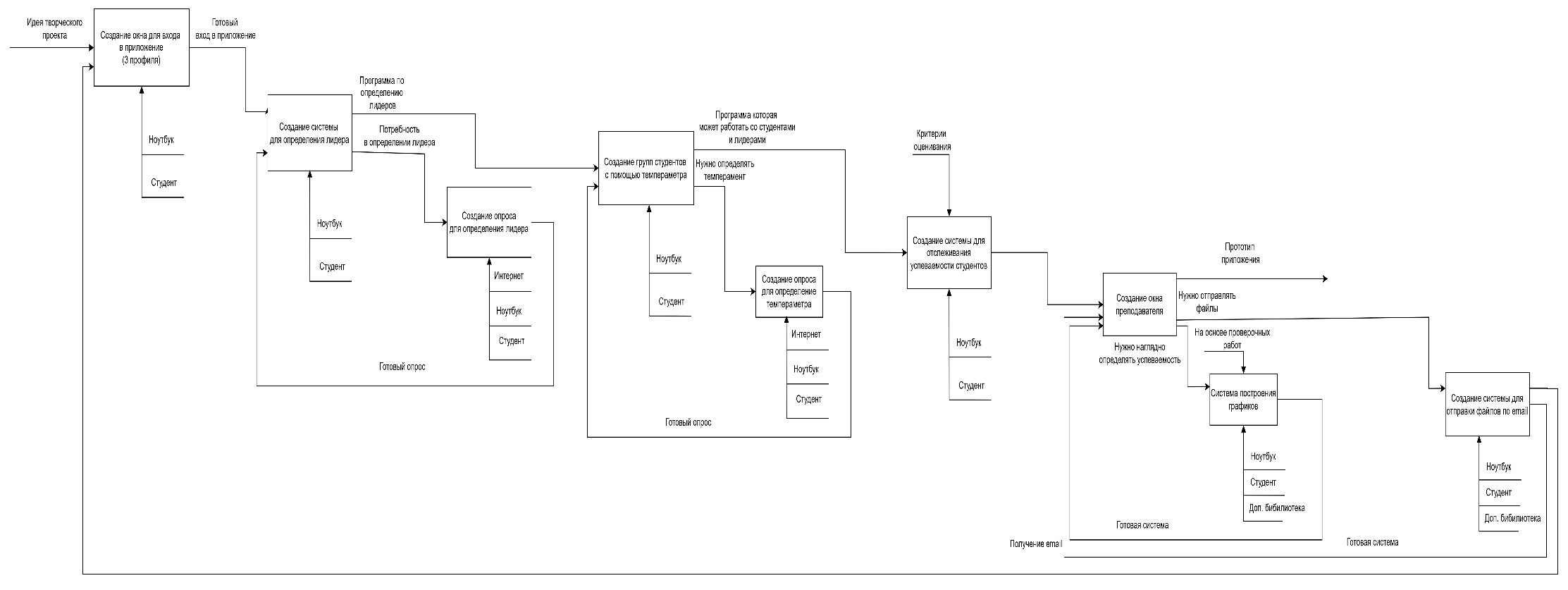
};

#endif // TEST\_H

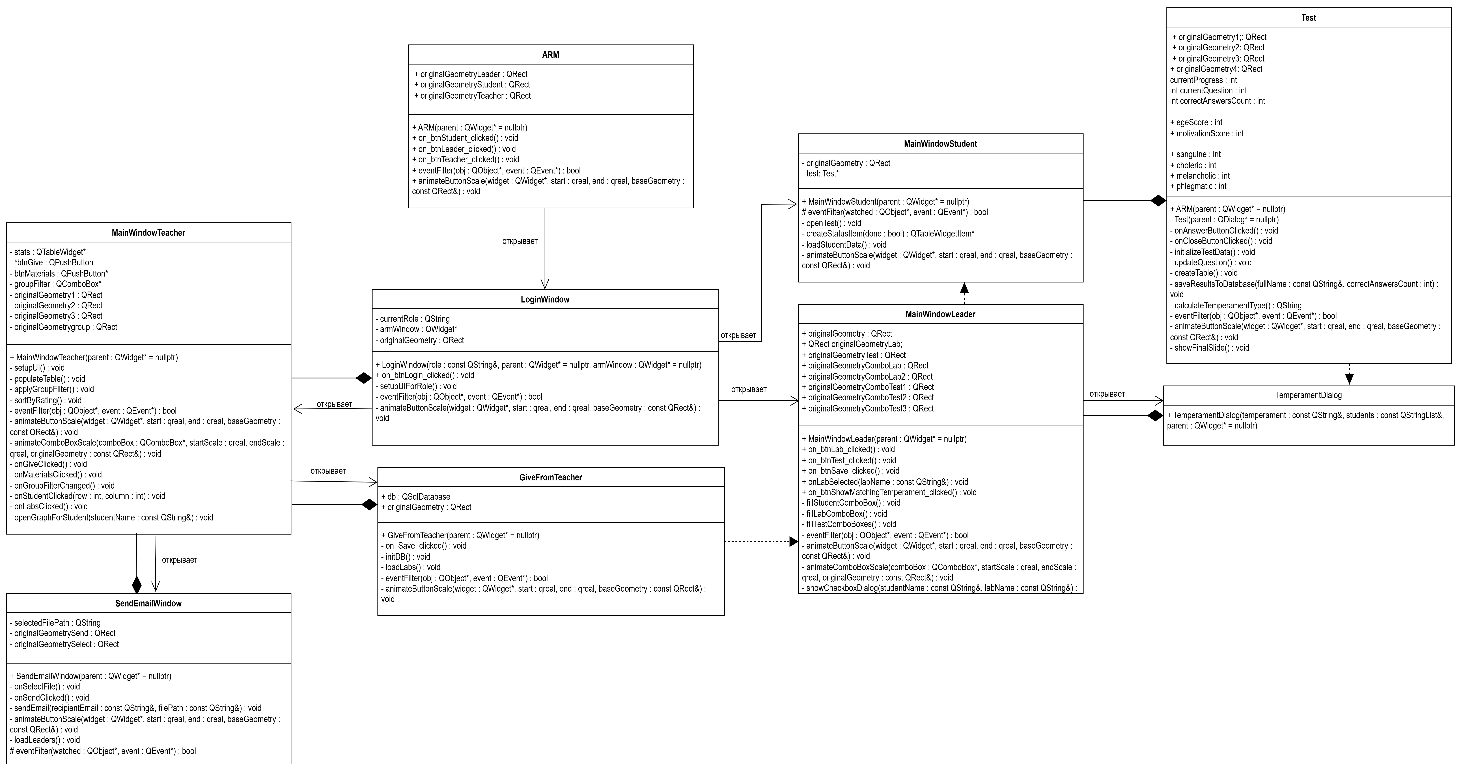
**ПРИЛОЖЕНИЕ Н**

****

****

****

**ПРИЛОЖЕНИЕ П**

****

**ПРИЛОЖЕНИЕ Р**

[**https://github.com/studentARM/ARM**](https://github.com/studentARM/ARM)