МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное
учреждение высшего образования
"Южно-Уральский государственный университет
(национальный исследовательский университет)"
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

ОТЧЕТ

по учебной практике

бакалавра направления 02.03.02 "Фундаментальная информатика и информационные технологии"

Выполнил:_____
студент группы КЭ-201
Гордеев А.С.

Проверил:_____
Докт. физ.-мат. наук, доцент,
Проф. кафедры СП
Макаровских Т.А.
Дата:_____, Оценка:_____

Челябинск-2022

Министерство науки и высшего образования Российской Федерации
Южно-Уральский государственный университет
Кафедра системного программирования

УТВЕРЖДАЮ

Зав. кафедрой
системного программирования

_____ Л.Б. Соколинский

ЗАДАНИЕ
по учебной практике

1. Цель работы

Разработать GUI-приложение, работающее с входной информацией, вводимой пользователем с помощью управляемых элементов формы, либо из текстового файла.

2. Исходные данные к работе

1. База данных пословиц, поговорок, афоризмов, каламбуров и других словесных курьезов.
2. Хранение данных в файле в формате json.
3. Файл хранит такую информацию как: автор, тема и фраза.

3. Перечень подлежащих разработке вопросов

1. Определение структуры приложения (по модулям), структур данных, используемых для хранения основной пользовательской информации.
2. Дизайн оконного интерфейса, анализ структуры входных данных и их защита от некорректного ввода информации.
3. Разработка основного функционала приложения: основных форм и механизмов получения информации из их компонентов и их файлов; основного алгоритма функционирования приложения; тестирование приложения.
4. Подготовка руководства пользователя и документации для программиста.

4. Сроки

Дата выдачи задания: "27" июня 2022 г.
Срок сдачи законченной работы: "23" июля 2022 г.

Руководитель:

Докт. физ-мат наук, проф. кафедры СП _____ Макаровских Т.А.

должность, ученая степень подпись ФИО руководителя

Задание принял к исполнению:

_____ Гордеев А.С.
подпись ФИО студента

ОГЛАВЛЕНИЕ

1. ПОСТАНОВКА ЗАДАЧИ

Требуется написать справочник пословиц, поговорок, афоризмов, каламбуров, других словесных курьезов. Программа выполняет классификацию по авторам и источникам, поиск по темам и автору. Программа должна обеспечивать поиск по базе по заданным критериям, позволять редактировать и дополнять базу. Записи базы данных будут иметь следующие типы (табл.1).

Таблица 1. Переменные, используемые для хранения записей базы данных

Переменная	Тип переменной	Содержательный смысл
id	int	Идентификатор записи
author	QString	Автор (источник)
theme	QString	Тема
phrase	QString	Фраза

Реализовать решение данной задачи можно с помощью такой известной структуры как префиксное дерево, или же бор. Каждый его элемент — вершина дерева, включающая в себя символ поиска строки, номер вершины, хэш-таблицу с номерами следующих вершин в виде: ключ-символ, значение-номер; булеву переменную, является ли вершина листом и массив типа QVector для хранения идентификаторов записей. Само префиксное дерево будет состоять из символов (вершин) строки с названием источника (автора) и темы для их поиска по данным критериям, а каждый лист массив идентификаторов.

Листинг 1. Пример структуры вершины и класса, реализующего поиск в префиксном дереве

```
struct Node {
  QMap<QChar, int> nextNodes;
  bool isLeaf = false;
  QVector<int> ids;
};

class PrefixSearcher {
 public:
```

```
  PrefixSearcher() : countNodes_(1), countStrings_(0) {}
  int getCountStrings() { return countStrings_; }
  void insert(const QString& str, int input_id);
  void erase(const QString& str, int erase_id);
  QVector<int> find(const QString& prefix);

 private:
  QMap<int, Node> trie_;
  int countNodes_;
  int countStrings_;

  bool canGoNode(int vertice, QChar sym);
  void createNode(int vertice, QChar sym);
  int getNextVertice(int vertice, QChar sym);
  bool isLeaf(int vretice);
  void depthFirstSearch(QVector<int>& result, int vertice);
};
```

Разрабатываемое приложение состоит из пяти оконных форм:

1. Главное окно программы.
2. Диалоговое окно добавление записи в базу.
3. Окно редактирования записи.
4. Окно с информацией о приложении.
5. Окно с информацией об авторе.

Каждой из разработанных оконных форм соответствует три файла (*.ui, *.h, *.cpp), приведенные в таблице 2. Там же приведена информация о файле prefixsearcher.h, в котором содержится информация о классе, реализующем работу префиксного дерева.

Таблица 2. Модули создаваемого проекта

Имя файла	Описание информации, содержащейся в нем	Функциональное назначение	Файлы проекта, подключенные к текущему файлу посредством директивы #include и файл разметки интерфейса
prefixsearcher.cpp	Класс реализации поиска в дереве	Работа с префиксном деревом	prefixsearcher.h
tabledata.cpp	Класс, реализующий базу дан-	Хранение и обработка данных	tabledata.h

	ных таблицы	таблицы	
mainwindow.cpp	Класс основного рабочего окна приложения	Работа с основным окном приложения.	mainwindow.h mainwindow.ui
additemdialog.cpp	Класс окна добавления записи	Ввод данных пользователя и добавление в базу	additemdialog.h additemdialog.ui
edititemdialog.cpp	Класс окна редактирования записи	Редактирование данных в базе	edititemdialog.h edititemdialog.ui
helpwindow.cpp	Класс окна с информацией о приложении	Вывод информации о приложении	helpwindow.h helpwindow.ui
aboutwindow.cpp	Класс окна с информацией об авторе	Вывод информации об авторе	aboutwindow.h aboutwindow.ui

На рисунке 1 представлена схема взаимодействия классов и оконных форм в приложении.

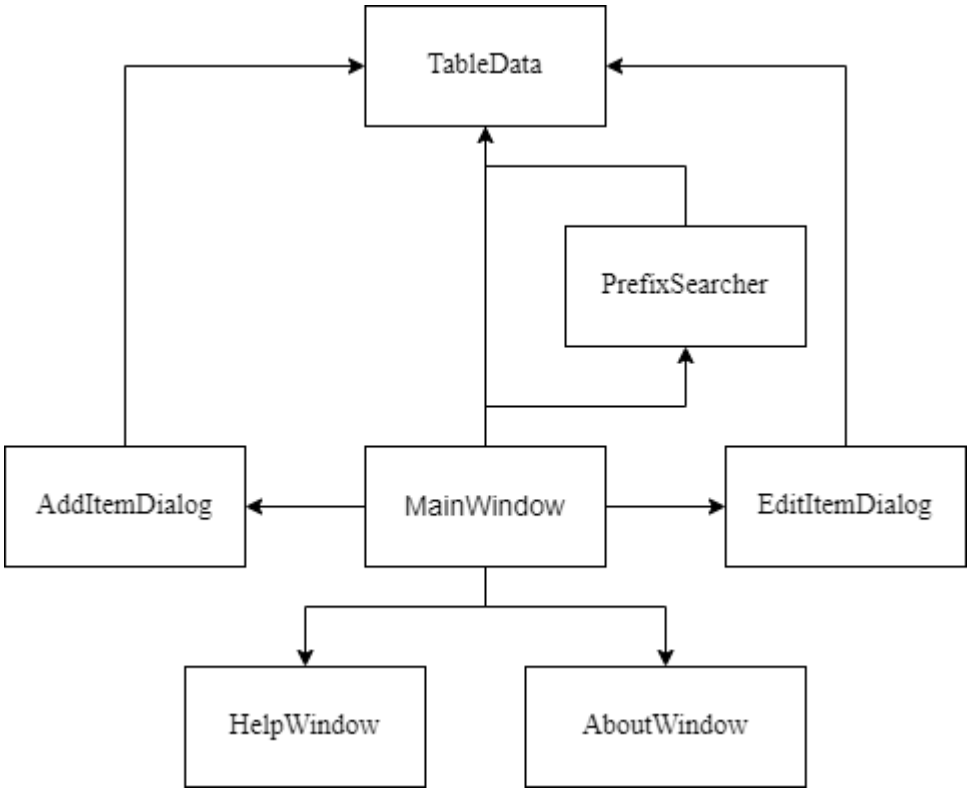

Рисунок 1. Блок-схема взаимодействия классов

2. ДИЗАЙН ОКОННОГО ИНТЕРФЕЙСА

В данном разделе приводится описание всех оконных форм, используемых для функционирования приложения:

– Основная форма для работы с таблицей.

– Форма добавление записи в таблицу.

– Форма редактирования записи таблицы.

– Форма информации о приложении.

– Форма информации об авторе.

– Вспомогательные окна.

При описании интерфейса приводится изображение соответствующей формы и приводится перечень помещенных на нее компонентов для ввода/вывода данных, оформления, и пр.

Для реализации работы с данными можно воспользоваться таблицей, реализованной в классе QTableView. На рисунке 2 представлено оформление основной формы приложения MainWindow класса QMainWindow.

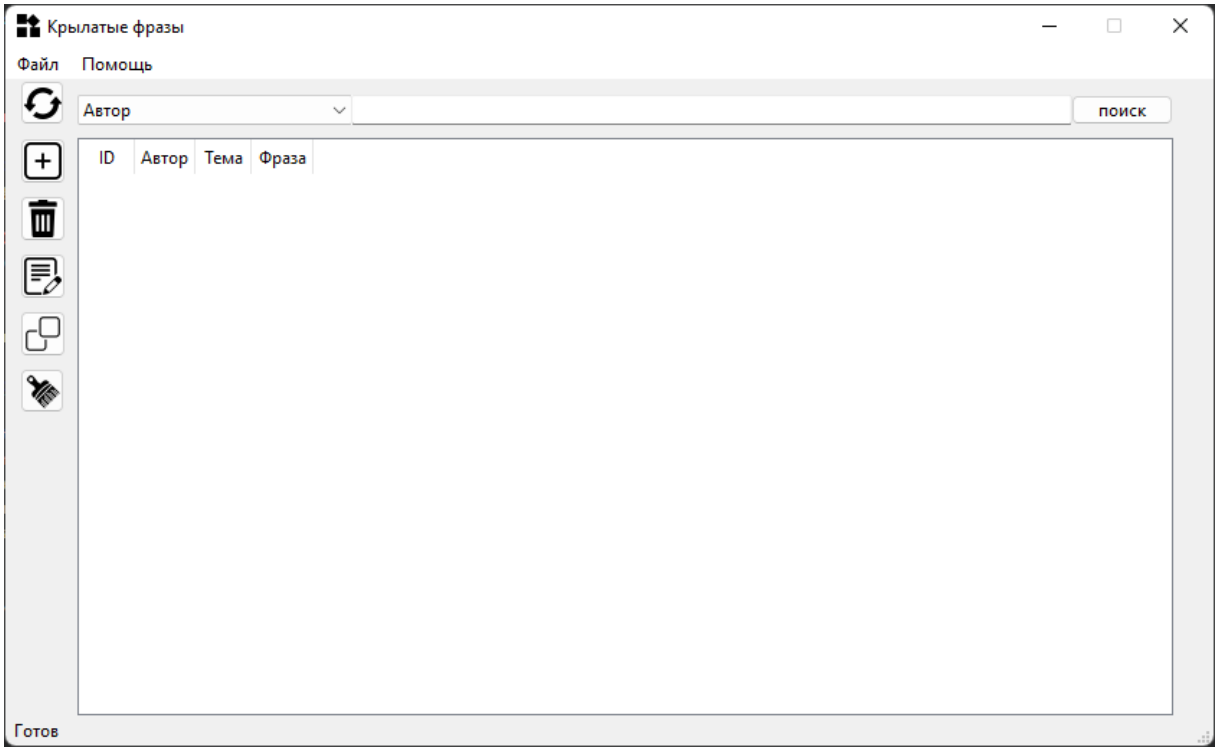

Рисунок 2. Оформление формы основного окна приложения MainWindow

Размещенные на форме компоненты и перечень методов и событий, которые необходимо реализовать приведен в таблице 3.

Таблица 3. Компоненты основного окна приложения

Имя компоненты	Тип	Ограничения для ввода информации	Реализованные события	Функциональное назначение
mainTableView	QTableView	Таблица с данными	Нет	Отображение таблицы из класса TableData
criterionComboBox	QComboBox	Выпадающее меню с пунктами: Автор, Тема	onCriterionComboBoxActivated	Выбор критерия для поиска в таблице
searchLineEdit	QLineEdit	Ввод данных в виде строки	Нет	Пользователь вводит строку для поиска значения из таблицы по заданному критерию
searchButton	QPushButton	Инициализация поиска с заданной строкой	onSearchButtonClicked	Пользователю выводится таблица с заданным критерием поиска
refreshButton	QPushButton	Обновление таблицы	onRefreshButtonClicked	Пользователь обновляет весь интерфейс таблицы
addButton	QPushButton	Вывод формы для добавления записи в таблицу	onAddButtonClicked	При нажатии таблицы открывается форма для заполнения
removeButton	QPushButton	Удаление записи с таблицы	onRemoveButtonClicked	При нажатии на кнопку осуществляется удаление выделенной записи таблицы

editButton	QPushButton	Вывод формы для редакти-рования записи	onEditButtonClicked	При нажатии на кнопку пользователь может редак-тировать вы-деленную за-пись таблицы
copyButton	QPushButton	Копирова-ние записи таблицы	onCopyButtonClicked	При нажатии на кнопку выделенная запись копи-руется в бу-фер обмена
clearButton	QPushButton	Очистка таблицы	onClearButtonClicked	Пользователь может очи-стить таблицу от всех запи-сей
menuBar	QMenuBar	Выбор пункта из выпадаю-щего меню	Нет	Пользователь может обра-титься к пунктам ме-ню
actonSave	QAction	Формат файла json	onSaveFileClicked	Пользователь может сохра-нить свои из-менения в файле
actionLoad	QAction	Формат файла json	onLoadFileClicked	Пользователь может загру-зить необхо-димый ему файл с дан-ными
actionHelp	QAction	Нет	onHelpActionClicked	Отображает окно с помо-щью для ра-боты с про-граммой
actionAbout	QAction	Нет	onAboutActionClicked	Отображает окно с ин-формацией об авторе

Добавление и редактирование записи из таблицы реализованы в по-хожих формах AddItemDialog и EditItemDialog соответственно и имеют класс QDialog. На рисунках 3 и 4 представлено оформление этих форм, а их компоненты в таблицах 4 и 5. Главные отличия в них, это название кно-

пок “Добавить” и “Изменить”, и в том, что данные в форме EditItemDialog берутся из таблицы для редактирования.

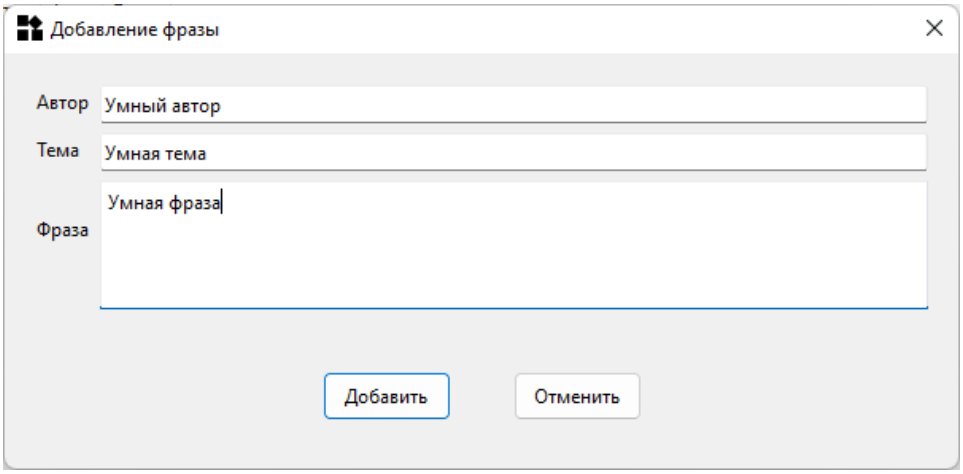

Рисунок 3. Оформление формы добавления записи AddItemDialog

Таблица 4. Компоненты диалогового окна AddItemDialog

Имя компоненты	Тип	Ограничения для ввода информации	Функциональное назначение
authorLabel	QLabel	Текстовая информация	Информация о вводимом поле “Автор”
authorLineEdit	QLineEdit	Ввод строки	Пользователь вводит имя автора или источника
themeLabel	QLabel	Текстовая информация	Информация о вводимом поле ‘Тема”
themeLineEdit	QLineEdit	Ввод строки	Пользователь вводит название темы фразы
phraseLabel	QLabel	Текстовая информация	Информация о вводимом поле “Фраза”
phraseTextEdit	QTextEdit	Ввод многострочных данных	Пользователь вводит фразу
acceptButton	QPushButton	Соглашение о вводимых данных	При нажатии на кнопку пользователь добавляет запись таблицы
cancelButton	QPushButton	Отмена изменений	При нажатии на кнопку пользователь отказывается от вводимых или измененных данных

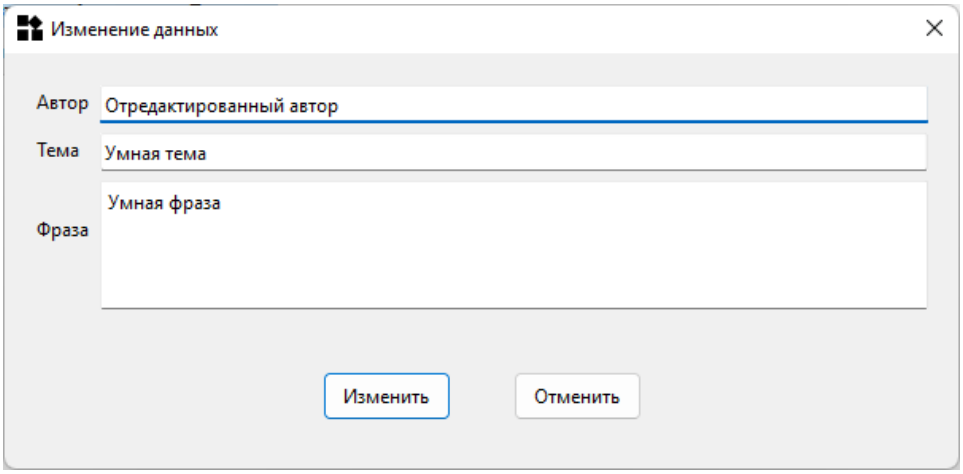

Рисунок 4. Оформление формы редактирования записи EditItemDialog

Таблица 5. Компоненты диалогового окна EditItemDialog

Имя компоненты	Тип	Ограничения для ввода информации	Функциональное назначение
authorLabel	QLabel	Текстовая информация	Информация о редактируемом поле “Автор”
authorLineEdit	QLineEdit	Ввод строки	Пользователь редактирует имя автора или источника
themeLabel	QLabel	Текстовая информация	Информация о вводимом поле ‘Тема”
themeLineEdit	QLineEdit	Ввод строки	Пользователь редактирует название темы
phraseLabel	QLabel	Текстовая информация	Информация о вводимом поле “Фраза”
phraseTextEdit	QTextEdit	Ввод многострочных данных	Пользователь редактирует фразу
acceptButton	QPushButton	Соглашение о вводимых данных	При нажатии на кнопку пользователь изменяет запись таблицы
cancelButton	QPushButton	Отмена изменений	При нажатии на кнопку пользователь отказывается от вводимых или измененных данных

Информационная форма содержит полезные подсказки для пользователя и представлена на рисунке 5, ее компоненты в таблице 6.

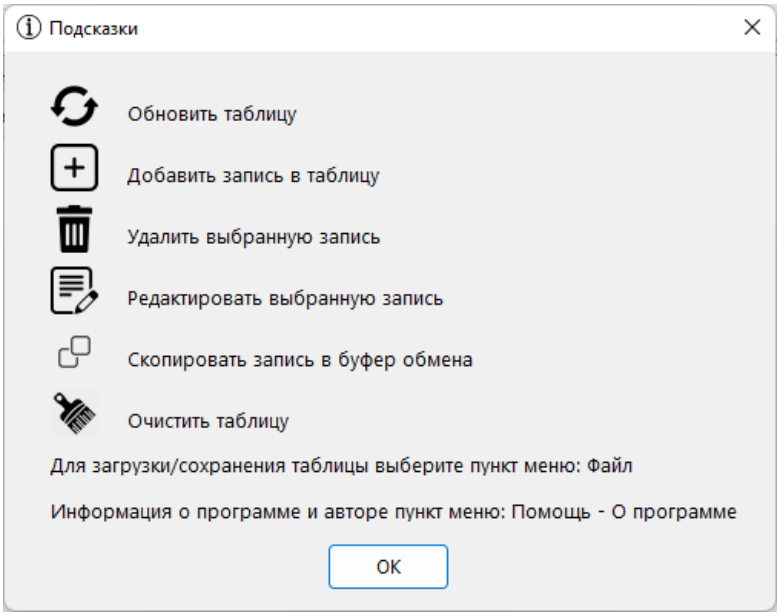

Рисунок 5. Оформление информационной формы HelpWindow

Таблица 6. Компоненты информационной формы HelpWindow

Имя компоненты	Тип	Функциональное назначение
updateIcon	QLabel	Иконка кнопки “Обновить”
updateLabel	QLabel	Подпись иконки “Обновить”
addIcon	QLabel	Иконка кнопки “Добавить”
addLabel	QLabel	Подпись иконки “Добавить”
removeIcon	QLabel	Иконка кнопки “Удалить”
removeLabel	QLabel	Подпись иконки “Удалить”
editIcon	QLabel	Иконка кнопки “Редактировать”
editLabel	QLabel	Подпись иконки “Редактировать”
copyIcon	QLabel	Иконка кнопки “Скопировать”
copyLabel	QLabel	Подпись иконки “Скопировать”
clearIcon	QLabel	Иконка кнопки “Очистить”
clearLabel	QLabel	Подпись иконки “Очистить”
fileLabel	QLabel	Информация о работе с файлами
aboutLabel	QLabel	Информация о том, как вызвать окно “О программе”
okButton	QPushButton	Позволяет выйти из формы

Форма об авторе содержит полезную информацию о программе и ее авторе и представлена на рисунке 6, компоненты в таблице 7.

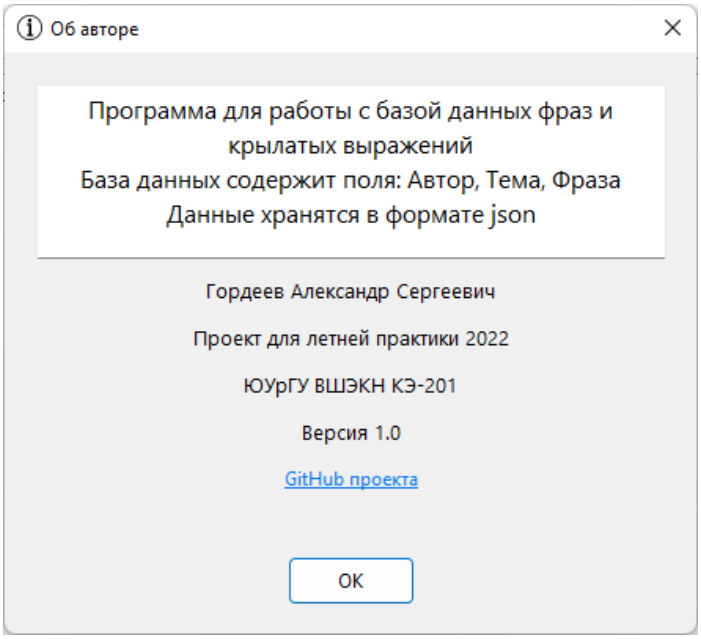

Рисунок 6. Оформление формы об авторе AboutWindow

Таблица 6. Компоненты формы об авторе AboutWindow

Имя компоненты	Тип	Функциональное назначение
aboutProgram	QTextEdit	Информация о программе
aboutLabel	QLabel	Информация об авторе
okButton	QPushButton	Позволяет выйти из формы

Алгоритмы и программная реализация приведенных в этой главе событий и методов приведены в следующем разделе.

3. РАЗРАБОТКА ФУНКЦИОНАЛА ОСНОВНЫХ ФОРМ И МЕХАНИЗМОВ ПОЛУЧЕНИЯ ИНФОРМАЦИИ

Главная форма MainWindow содержит множество кнопок, которые привязаны к событиям QAbstractButton::clicked, а также QAction к событиям QAction::triggered. Здесь приводится информация обо всех событиях и методах.

При нажатии на кнопку refreshButton вызывается метод onRefreshButtonClicked, который обновляет таблицу и сбрасывает состояние поиска. Исходный код представлен в листинге 2.

Листинг 2. Метод onRefreshButtonClicked

```
void MainWindow::onRefreshButtonClicked() {
  QAbstractItemModel* ui_model = ui_->mainTableView->model();
  if (ui_model != tableData_->getModel() && ui_model != nullptr) {
    delete ui_model;
  }
  ui_->mainTableView->setModel(tableData_->getModel());
  ui_->criterionComboBox->setCurrentIndex(0);
  ui_->searchLineEdit->clear();
  statusBar()->showMessage("Таблица обновлена");
}
```

Кнопка addButton вызывает метод onAddButtonClicked, который в свою очередь открывает форму для добавления записи в таблицу AddItemDialog. Исходный код в листинге 3.

Листинг 3. Метод onAddButtonClicked

```
void MainWindow::onAddButtonClicked() {
  AddItemDialog dialog(this);
  dialog.exec();

  if (dialog.result() == QDialog::Accepted) {
    QVector<QString> data = {dialog.getAuthor(), dialog.getTheme(),
                             dialog.getPhrase()};
    tableData_->addRow(data);
    refreshTable();
    statusBar()->showMessage("Запись успешно добавлена в таблицу");
  }
}
```

Кнопка removeButton удаляет выделенную строку в таблице. Удаление происходит по идентификатору (ID) записи. Исходный код представлен в листинге 4.

Листинг 4. Метод onRemoveButtonClicked

```
void MainWindow::onRemoveButtonClicked() {
```

```
  QModelIndexList selected_rows =
      ui_->mainTableView->selectionModel()->selectedRows();
  if (selected_rows.size() != 1) {
    buttonMessageBox("Выберите строку для удаления");
    return;
  }

  int id = selected_rows.first().data().toInt();
  tableData_->removeRow(id);
  refreshTable();
  statusBar()->showMessage("Запись успешно удалена");
}
```

Кнопка editButton вызывает форму редактирования выделенной записи таблицы EditItemDialog. Нужная запись также выбирается по идентификатору. Исходный код в листинге 5.

Листинг 5. Метод onEditButtonClicked

```
void MainWindow::onEditButtonClicked() {
  QModelIndexList selected_rows =
      ui_->mainTableView->selectionModel()->selectedRows();
  if (selected_rows.size() != 1) {
    buttonMessageBox("Выберите строку для редактирования");
    return;
  }

  int id = selected_rows.first().data().toInt();
  QVector<QString> data = tableData_->getRowData(id);
  EditItemDialog dialog(data, this);
  dialog.exec();

  if (dialog.result() == QDialog::Accepted) {
    QVector<QString> data = {dialog.getAuthor(), dialog.getTheme(),
                             dialog.getPhrase()};
    tableData_->editRow(data, id);
    refreshTable();
    statusBar()->showMessage("Запись в таблице успешно изменена");
  }
}
```

Кнопка copyButton копирует выделенную запись таблицы в буфер обмена. Код в листинге 6.

Листинг 6. Метод onCopyButtonClicked

```
void MainWindow::onCopyButtonClicked() {
  QModelIndexList selected_rows =
      ui_->mainTableView->selectionModel()->selectedRows();
  if (selected_rows.size() != 1) {
    buttonMessageBox("Выберите строку для копирования");
    return;
  }

  int id = selected_rows.first().data().toInt();
```

```
  QVector<QString> data = tableData_->getRowData(id);
  QString result = data[1];
  for (int i = 2; i < data.size(); ++i) {
    result += '\t' + data[i];
  }

  QApplication::clipboard()->setText(result);
  statusBar()->showMessage("Строка скопирована");
}
```

Кнопка clearButton полностью очищает таблицу от записей. Исходный код представлен в листинге 7.

Листинг 7. Метод onClearButtonClicked

```
void MainWindow::onClearButtonClicked() {
  QAbstractItemModel* model = tableData_->clearModel();
  if (model) {
    QAbstractItemModel* ui_model = ui_->mainTableView->model();
    if (ui_model != nullptr) {
      delete ui_model;
    }
    ui_->mainTableView->setModel(model);
    statusBar()->showMessage("Таблица успешно очищена");
  } else {
    statusBar()->showMessage("Ошибка очистки таблицы");
  }
}
```

Кнопка searchButton вызывает метод onSearchButtonClicked, который делегирует свою работу методу refreshTable. Исходный код этих двух методов представлен в листинге 8.

Листинг 8. Метод onSearchButtonClicked и refreshTable

```
void MainWindow::onSearchButtonClicked() { refreshTable(); }

void MainWindow::refreshTable() {
  QAbstractItemModel* ui_model = ui_->mainTableView->model();
  if (ui_model != tableData_->getModel() && ui_model != nullptr) {
    delete ui_model;
  }

  QString criterion = ui_->criterionComboBox->currentText();
  QString search_string = ui_->searchLineEdit->text();
  if (search_string.isNull() || search_string.isEmpty()) {
    ui_->mainTableView->setModel(tableData_->getModel());
    return;
  }

  QAbstractItemModel* model = tableData_->search(criterion,
search_string);
  ui_->mainTableView->setModel(model);
}
```

Вызов меню actionSave позволяет выбрать место для сохранения данных таблицы в формате json. Код в листинге 9.

Листинг 9. Метод onSaveFileClicked

```
void MainWindow::onSaveFileClicked() {
  QString path = QFileDialog::getSaveFileName(this, "Сохранить
файл", "", "Фразы (*.json)");
  if (path.isNull() || path.isEmpty()) {
    return;
  }

  if (tableData_->saveToJson(path)) {
    statusBar()->showMessage("Файл успешно сохранен");
  } else {
    statusBar()->showMessage("Ошибка сохранения файла");
  }
}
```

Вызов меню actionLoad открывает окно с выбором файла для загрузки данных таблицы в формате json. Код представлен в листинге 10.

Листинг 10. Метод onLoadFileClicked

```
void MainWindow::onLoadFileClicked() {
  QString path = QFileDialog::getOpenFileName(this, "Загрузить
файл", "", "Фразы (*.json)");
  if (path.isNull() || path.isEmpty()) {
    return;
  }

  QAbstractItemModel* model = tableData_->loadFromJson(path);
  if (model) {
    ui_->mainTableView->setModel(model);
    compl_->setModel(model);
    statusBar()->showMessage("Файл успешно загружен");
  } else {
    statusBar()->showMessage("Ошибка загрузки файла");
  }
}
```

Вызов меню actionHelp открывает окно с подсказками для пользователя HelpWindow. Исходный код вызывающего метода в листинге 11.

Листинг 11. Метод onHelpActionClicked

```
void MainWindow::onHelpActionClicked() {
  HelpWindow wnd(this);
  wnd.exec();
}
```

Вызов меню actionAbout открывает окно AboutWindow с информацией об авторе и самой программе. Код метода в листинге 12.

Листинг 12. Метод onAboutActionClicked

```
void MainWindow::onAboutActionClicked() {
  AboutWindow wnd(this);
  wnd.exec();
}
```

Вспомогательный метод для вызова окон с сообщениями о различных ошибках типа QMessageBox. Исходный код метода в листинге 13.

Листинг 13. Метод buttonMessageBox

```
void MainWindow::buttonMessageBox(const QString& text) {
  QMessageBox msg_box;
  msg_box.setWindowTitle("Таблица");
  msg_box.setWindowIcon(windowIcon());
  msg_box.setIcon(QMessageBox::Information);
  msg_box.setText(text);
  msg_box.exec();
}
```