**Министерство науки и высшего образования РФ**

**Федеральное государственное автономное образовательное учреждение  
 высшего образования**

***Национальный исследовательский   
Нижегородский государственный университет им. Н.И. Лобачевского***

**Факультет повышения квалификации и профессиональной переподготовки**

Программа дополнительного профессионального образования  
 «Профессиональное программирование»

**ВЫПУСКНАЯ РАБОТА**

**«Репетитор английских слов «Мой словарь»»**

**Выполнил:**

**Скрипов Е.В.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Руководитель:**

**к.ф.-м.н., доцент**

**Городецкий С.Ю.**

**Нижний Новгород**

**2019**

Содержание

1. Введение 3

2. Постановка задачи 4

3. Выбранные программные средства для выполнения разработки 4

4. Модель данных и общая структура программы 4

5. Подходы к решению специальных задач, возникших при разработке 4

6. Примеры использования 4

7. Выводы 4

Литература 4

Приложение 4

# 1. Введение

Приложение «Мой словарь» предназначено для тренировки английских слов и повышения уровня их усвоения с помощью визуального восприятия.

Зрительная память – это запоминание информации, воспринимаемой органами зрения.

При изучении иностранных языков зрительная память играет одну из главных ролей. Проще запомнить то, что видел, и что оставило хоть какой-то эмоциональный отпечаток.

Одним из самых эффективных методов выучить иностранные слова является обклеивание окружающих предметов бирками с их названиями на иностранном языке. В течение долгого времени человек видит на этих предметах их название на иностранном языке, и они невольно отпечатываются в его памяти.

В данном приложении используется похожий метод. К любому слову, которое пользователь хочет запомнить, можно прикрепить изображение, так или иначе ассоциирующееся со словом.

Отличительными особенностями программы являются возможность добавления изображений, ассоциирующихся со словами, а так же автоматическое распределение слов по успеваемости пользователя. Новые и плохо изученные слова всплывают чаще, но иногда будут появляться и хорошо отработанные, для дополнительного контроля.

Использование приложения подразумевает создание пользователем своей собственной базы слов, которую ему необходимо отрабатывать.

# 

# 2. Постановка задачи

# Для начала программа должна запрашивать количество слов, которое пользователь хочет отработать, и режим тренировки. Под режимом подразумевается, на каком языке будет высвечено слово и на каком принимается ответ.

При запуске тренировки, в окне будут поочередно показываться слова, которые нужно перевести. В случае необходимости пользователь может посмотреть изображение (при удержании нажатой левой кнопки мыши на активном элементе), ассоциирующееся с активным словом.

Программа должна:

* Уметь сохранять новые и удалять изученные слова с изображениями.
* Отображать в списке, имеющийся словарный запас, а так же сортировать его.
* Для эффективного обучения – сортировать слова по успеваемости, а значит и вести статистику по каждому слову.
* Иметь приятный и интуитивно-понятный интерфейс.

# 3. Выбранные программные средства для выполнения разработки

Приложение будет написано на языке С++.

В качестве среды разработки была выбрана программа Qt Creator. Визуальное программирование облегчается благодаря встроенному в неё дизайнеру. Так же в Qt содержатся все компоненты, необходимые для реализации поставленных задач.

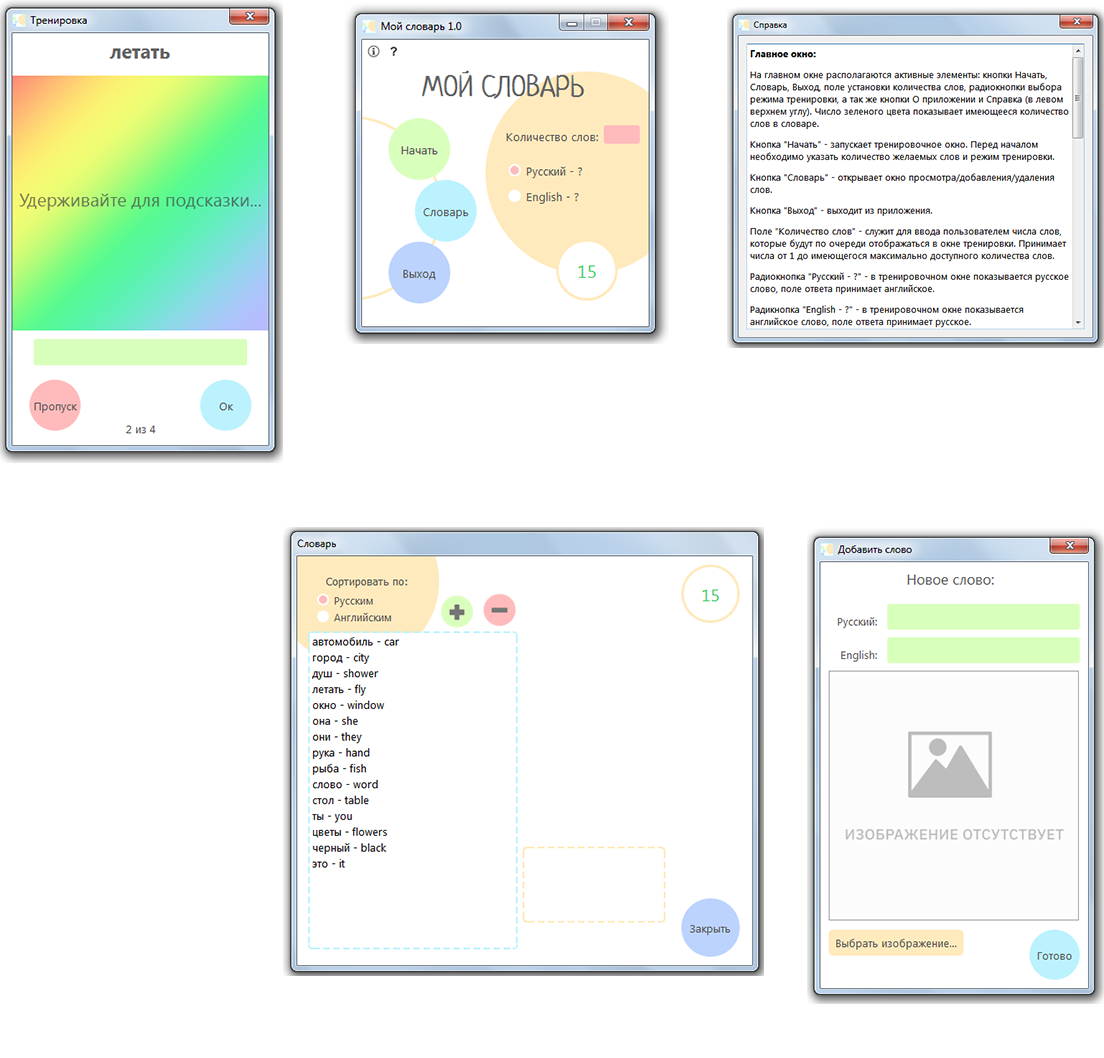
В качестве хранилища данных используется реляционная, не имеющая сервер, база данных SqlLite.

# 4. Модель данных и общая структура программы

***4.1 Структура программы***

В программе реализовано 4 основных окна и одно дополнительное:

* Главное окно (MainWindow)
* Окно тренировки (TrainingWindow)
* Окно словаря (WordsListWindow)
* Окно добавления нового слова (DialogAddWord)
* Окно справки – дополнительное.



Окно справки

Окно словаря

Окно тренировки

Окно добавления нового слова

Главное окно

Рис. 1. Общий вид структуры программы

***4.1 Реализация класса MainWindow***

Класс MainWindow – класс, с которого начинается работа программы.

При запуске приложения объект класса MainWindow проверяет наличие базы данных, а при её отсутствии – создает файл БД с одним словом.

Далее считывается информация о количестве имеющихся слов и отображается в окне, после чего программа ждет действий пользователя.

Ниже представлен заголовочный файл mainwindow.h:

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include "wordslistwindow.h"

#include "trainingwindow.h"

#include "helpdialog.h"

namespace **Ui** {

class **MainWindow**;

}

class **MainWindow** : public QMainWindow

{

Q\_OBJECT

public:

explicit **MainWindow**(QWidget \*parent = nullptr);

~***MainWindow***();

private slots:

void **on\_btnDictionary\_clicked**();//обработчик открытия словаря

void **showWordsCountSlot**();//слот показа кол-ва доступных слов и отображения главного окна

void **on\_btnStart\_clicked**();//обработчик начала тренировки

void **on\_btnExit\_clicked**();//кнопка выхода из программы

void **on\_aboutApplication\_triggered**();//слот О приложении

void **on\_actionhelp\_triggered**();//слот Справка

private:

Ui::MainWindow \*ui;

QString fullDBName = QCoreApplication::applicationDirPath() + QDir::separator() + "db\_name.sqlite";//адрес и имя БД

int wordsCount;//количество доступных слов

int userWordsCount;//желаемое количество слов

WordsListWindow\* wordListDialog;//указатель на словарь

TrainingWindow\* trainingDialog;//указатель на окно тренировки

bool **dataBaseIsOK**();//Проверка открытия или создание БД и запись количества слов в ней

bool **showWordsCount**();//показать кол-во доступных слов

void **animationShow**();//анимация появления

};

#endif // MAINWINDOW\_H

В файле mainwindow.ui располагаются активные элементы: кнопки «Начать», «Словарь», «Выход», поле установки количества слов, радиокнопки выбора режима тренировки, а так же кнопки «О приложении» и «Справка» (в левом верхнем углу). Число зеленого цвета показывает имеющееся количество слов в словаре.

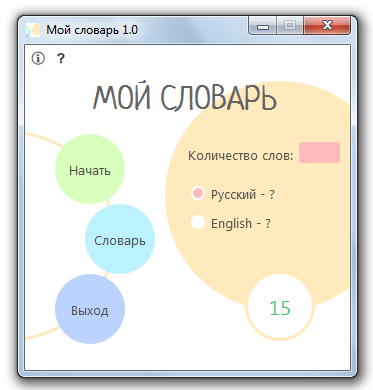


Рис. 2. Главное окно

***4.2 Реализация класса WordsListWindow***

Класс WordsListWindow считывает и отображает информацию о словах из базы данных, а так же производит все операции добавления и удаления слов.

Главными элементами класса являются два контейнера «mapRU» и «mapENG», представляющие собой объекты класса QMap – шаблона, хранящего пары (ключ, значение) и обеспечивающего быстрый поиск значения, связанного с ключом. Быстрый поиск позволяет за короткое время найти и отобразить всю информацию о выбранном слове.

В QMap элементы хранятся в упорядоченном виде (по ключу), что играет нам на руку и избавляет от написания метода сортировки при каждом отображении списка после добавления/удаления слов.

Считывание из БД происходит сразу в оба контейнера. В «mapRU» ключом является русское слово, а в «mapENG» - английское. В зависимости от желания пользователя, при отображении список слов берется либо из «mapRU», либо из «mapENG».

Ниже представлен заголовочный файл wordlistwindow.h:

#ifndef WORDSLISTWINDOW\_H

#define WORDSLISTWINDOW\_H

#include <QDialog>

#include <QMap>

#include <QVector>

#include <QtSql>

#include <QListWidgetItem>

#include "dialogaddword.h"//Заголовочный диалогового окна добавления

namespace **Ui** {

class **WordsListWindow**;

}

class **WordsListWindow** : public QDialog

{

Q\_OBJECT

public:

explicit **WordsListWindow**(QWidget \*parent = nullptr);

~***WordsListWindow***();

signals:

void **provideNewImageNameSignal**(QString);//предоставить имя новой картинки

void **showWordsCountSignal**();//сигнал показа в MainWindow кол-ва доступных слов

private slots:

void **saveRuEngNumberSlot**(QString, QString, int);//слот изменения и предоставления имени новой картинки

void **on\_btnAddWord\_clicked**();//слот добавления нового слова

void **on\_btnDelWord\_clicked**();//слот удаления выделенного слова

void **on\_radioButtonRu\_Eng\_toggled**(bool checked);//если включено, то отображаем Рус - Англ

void **on\_radioButtonEng\_Ru\_toggled**(bool checked);//если включено, то отображаем Англ - Рус

void **on\_listWords\_itemClicked**(QListWidgetItem \*item);//слот отображения картинки выделенного элемента

void **on\_btnClose\_clicked**();//кнопка закрытия окна

private:

bool **readDB**();//Считывание БД

void **showList**();//Отображаем список слов и картинку

bool **addWordInDB**(QString, QString, int);//добавление нового слова в БД и мапы, а так же отображение

bool **addWordInMaps**(QString, QString, int);//добавление нового слова в Мапы

void **deleteWord**(int);//удалить слово

bool **delWordFromDB**(QString);//удалить слово из базы данных

bool **delWordFromMaps**(QString, QString);//удалить слово из Мапов

bool **delImageFromDir**(QString);//удалить картинку из папки

void **clearStat**();//очистить статистику и картинку

private:

Ui::WordsListWindow \*ui;

QMap<QString, QVector<QString>>\* mapRU;

QMap<QString, QVector<QString>>\* mapENG;

DialogAddWord\* addWordDialog = nullptr;//указатель на диал.окно добавления

int numberImg = 0;//номер последней картинки

int wordsCount;//количество доступных слов

QString fullDBName = QCoreApplication::applicationDirPath() + QDir::separator() + "db\_name.sqlite";//адрес и имя БД

};

#endif // WORDSLISTWINDOW\_H

В файле wordlistwindow.ui располагаются: поле со списком доступных слов, радиокнопки сортировки, поля отображения картинки и информации о выделенном слове, кнопки добавления и удаления слова и кнопка «Закрыть». Число зеленого цвета показывает имеющееся количество слов в словаре.

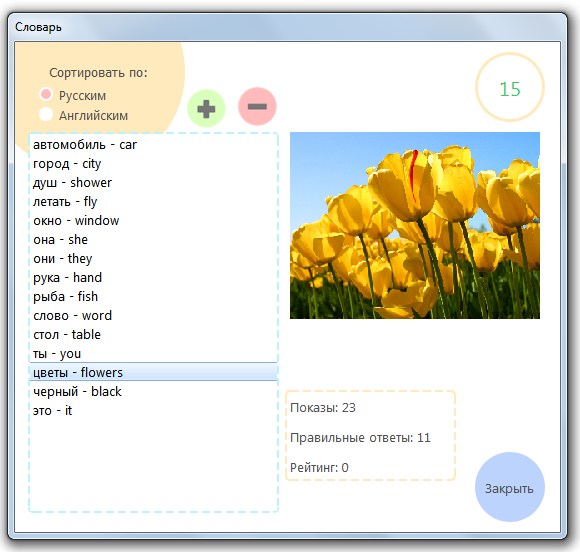


Рис. 3. Окно словаря

***4.3 Реализация класса DialogAddWord***

Класс DialogAddWord принимает информацию о новом слове, а так же сохраняет копию загруженной картинки в отдельную папку с изображениями «saved\_image».

Данный класс никак не контактирует с БД, а только передаёт введенные пользователем данные объекту класса WordListWindow через сигнал *getRuEngAddressSignal* для их сохранения. Кроме того, если была загружена новая картинка, то из объекта WordListWindow испускается сигнал, передающий новое имя для неё. Слот *saveNewImageSlot* принимает его и сохраняет изображение.

Когда пользователь не хочет загружать изображение, то таким словом будет высвечиваться стандартное, с надписью «изображение отсутствует».

Ниже представлен заголовочный файл dialogaddword.h:

#ifndef DIALOGADDWORD\_H

#define DIALOGADDWORD\_H

#include <QDialog>

namespace **Ui** {

class **DialogAddWord**;

}

class **DialogAddWord** : public QDialog

{

Q\_OBJECT

public:

explicit **DialogAddWord**(QWidget \*parent = nullptr);

~***DialogAddWord***();

signals:

void **getRuEngAddressSignal**(QString, QString, int);//сигнал запроса имени новой картинки

private slots:

void **saveNewImageSlot**(QString);//слот установки имени новой картинки

void **on\_btnChangeImage\_clicked**();//добавить свою картинку

void **on\_btnClearImage\_clicked**();//очистить добавленную картинку

void **on\_btnFinishAddWord\_clicked**();//Готово

private:

Ui::DialogAddWord \*ui;

QPixmap savedPic;//сохраняемая картинка

};

#endif // DIALOGADDWORD\_H

В файле dialogaddword.ui располагаются: поля для ввода русского и английского слов (каждое поле принимает только соответствующее ему значение), поле отображения картинки, ассоциирующейся со словом, кнопки «Выбрать изображение» и «Готово».

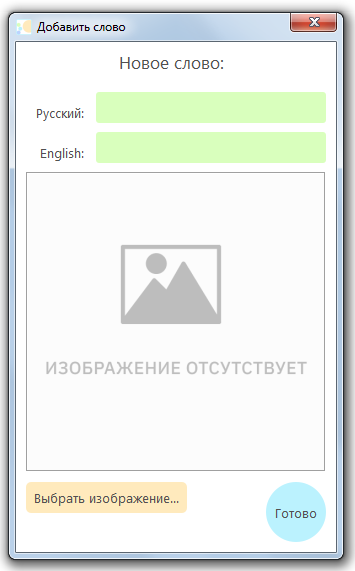


Рис. 4. Окно добавления нового слова

***4.4 Реализация класса TrainingWindow***

Класс TrainingWindow формирует список отображаемых слов, запускает тренировку и сохраняет результаты в БД. При каждом ответе он запоминается, а в окне отображается индикатор правильного или ошибочного ответа.

Конструктор класса принимает входными параметрами: количество слов в словаре, число введенное пользователем в главном окне и булево значение, которое характеризует режим тренировки.

После последнего ответа выводится результат тренировки.

Ниже представлен заголовочный файл trainingwindow.h:

#ifndef TRAININGWINDOW\_H

#define TRAININGWINDOW\_H

#include <QtSql>

#include <QDialog>

#include "word.h"

namespace **Ui** {

class **TrainingWindow**;

}

class **TrainingWindow** : public QDialog

{

Q\_OBJECT

public:

explicit **TrainingWindow**(QWidget \*parent = nullptr, int c = 0, int u = 0, bool rn = true);

~***TrainingWindow***();

signals:

void **showMainWindowSignal**();//сигнал показа главного окна

private slots:

void **on\_btnShowImg\_pressed**();//обработчик удержания кнопки

void **on\_btnShowImg\_released**();//обработчик отпускания кнопки показа изобр.

void **on\_btnOk\_clicked**();//обработчик кнопки ОК

void **on\_btnSkip\_clicked**();//обработчик кнопки пропуска

void **nextOrShowResult**();//слот следующего слова или выхода

void **on\_btnExit\_clicked**();//обработчик кнопки выхода

private:

Ui::TrainingWindow \*ui;

bool ru\_eng;//рус-англ тест

int count;//количество показываемых слов

double userWordsCount;//желаемое количество слов

int currentNumWord;//номер текущего тестируемого слова

int seventy;//70%

int thirty;//30%

int cor = 0;//количество верных ответов

int wr = 0;//количество Не верных ответов

int skip = 0;//количество пропусков

QString fullDBName = QCoreApplication::applicationDirPath() + QDir::separator() + "db\_name.sqlite";//адрес и имя БД

QPixmap clickImg;//картинка с указанием кликнуть

QPixmap hiddenImg;//скрытая картинка

QPixmap answerTrue;//индикатор верного ответа

QPixmap answerFalse;//индикатор НЕверного ответа

QString dir = QCoreApplication::applicationDirPath() + QDir::separator() + "saved\_image";//адрес папки с картинками

QList<Word\*>\* questionList;//указатель на список указателей на слова

QList<Word\*>::iterator it;//итератор списка указателей на слова

Word\* intermediateWord;//указатель на слово-посредник

void **calcPercentWords**();//вычисление процентов слов

bool **readToList**();//считывание из БД в список

void **read70FromDB**();//считывание необходимых 70% слов из БД

void **read30FromDB**();//считывание необходимых 30% слов из БД

void **startTest**();//запуск тестирования

void **showWord**();//отобразить слово

void **answerCorrectly**();//действия при верном ответе

void **answerWrong**();//действия при НЕверном ответе

bool **saveCorrectly**();//сохранили верный ответ

bool **saveWrong**();//сохранили неверный ответ

void **showResult**();//показать результат

void ***TrainingWindow***::***closeEvent***(QCloseEvent \*event);//событие закрытия окна

};

#endif // TRAININGWINDOW\_H

В файле trainingwindow.ui описаны: поле отображения картинки (по необходимости), номер слова и количество отображаемых слов, и кнопки «Пропуск» и «Ок». Поле отображения картинки - показывает стандартное изображение с призывом "Удерживать" левую клавишу мыши на нем для отображения скрытой картинки.

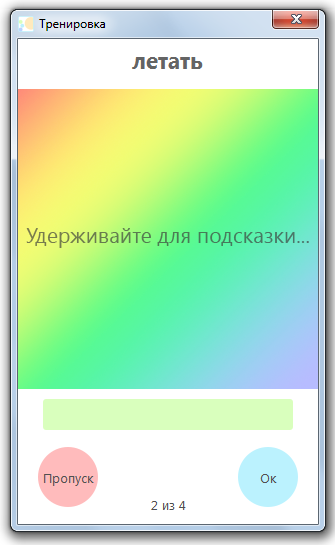


Рис. 4. Окно тренировки

***4.5 Реализация класса Word***

Для формирования тренировочного списка, изменения и сохранения результатов используются объекты класса Word.

Класс Word – содержит поля для хранения информации о слове, а так же методы для её предоставления и изменения.

Ниже представлен заголовочный файл word.h:

#ifndef WORD\_H

#define WORD\_H

#include <QString>

class **Word**

{

public:

**Word**();//конструктор

**Word**(QString, QString, QString, int, int, int);//конструктор от параметров

~**Word**(){}//деструктор

QString **getRu**();//получить русское слово

QString **getEng**();//получить английское слово

QString **getnameImg**();//получить имя картинки

int **getShowing**();//получить количество показов

int **getCorrectly**();//получить количество правильных ответов

int **getRating**();//получить рейтинг

void **addShowing**();//увеличить показы

void **addCorrectly**();//увеличить правильные ответы

void **addRating**();//увеличить рейтинг

void **downRating**();//уменьшить рейтинг

private:

QString ru;//русское

QString eng;//английское

QString nameImg;//имя картинки

int showing;//показы

int correctly;//верные ответы

int rating;//рейтинг

};

#endif // WORD\_H

***4.6 Хранение данных в БД SqlLite.***

Все данные о словах хранятся в файле базы данных "db\_name.sqlite".

В нем находится таблица words\_table, состоящая из колонок:

* ru – слово на русском
* eng – слово на английском
* numberImg – имя (номер) картинки
* showing – показы
* correctly – количество верных ответов
* rating – рейтинг

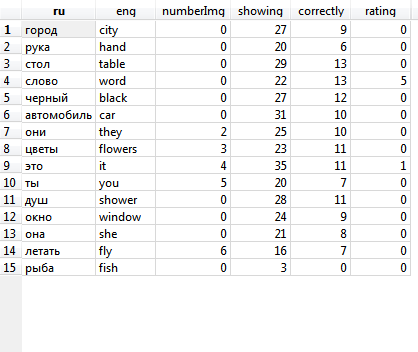


Рис. 5. Пример заполненной таблицы данных

# 5. Подходы к решению специальных задач, возникших при разработке

После ввода пользователем желаемого количества слов для отработки и нажатия кнопки «Начать», как было описано раннее, в объекте класса TrainingWindow формируется список слов для тренировки.

Последовательное заполнение списка словами из БД нам не подходит, т.к. слово, находящееся в самом конце, будет отображаться только в случае ввода пользователем числа, равного количеству элементов в словаре. А если он не хочет отрабатывать их все?

Выходом из ситуации может показаться «рандомное» заполнение списка. Но и в этом случае существует вероятность, что какие-нибудь слова и вовсе никогда не попадут в него, либо будут отображаться слишком часто.

Подход, описанный далее, исключает оба этих недостатка. Смыслом его является разбиение желаемого количества слов на 70% и 30%.

Каждое слово, хранящееся в словаре, имеет три дополнительных целочисленных характеристики: «Количество показов», «Правильные ответы» и «Рейтинг». При добавлении нового слова все они равны нулю.

С количеством показов и правильными ответами всё понятно. А что означает Рейтинг?

Рейтинг – это количество правильных ответов подряд. Он может иметь максимальное значение, равное 5. Но если пользователь после одного или нескольких верных ответов ошибся, то рейтинг сбрасывается до 0 (если он был меньше или равен 3) или до 3 (если его значение было от 3 до 5).

Основываясь на этих характеристиках мы можем выбирать слова, которые необходимо показывать в первую очередь. Ими могут быть слова с нулевым рейтингом, новые слова или слова с маленьким количеством верных ответов.

При формировании списка объект TrainingWindow обращается к базе данных через метод read70FromDB с запросом предоставить определенное количество слов (70%) с сортировкой сначала по рейтингу, затем по показам и количеству верных ответов. Затем следует второй запрос через метод read30FromDB с требованием предоставить несколько (30%) «рандомных» слов, не равных раннее внесенным в список. Таким образом, хорошо отработанные слова тоже будут появляться, для дополнительного контроля.

Ниже представлен код методов.

Метод *read70FromDB:*

void TrainingWindow::**read70FromDB**()//считывание необходимых 70% слов из БД

{

QSqlQuery obj\_query;//объект для управления запросами

QString sev = QString::number(seventy);//перевели в строку

QString str = "SELECT ru, eng, numberImg, showing, correctly, rating FROM words\_table ORDER BY rating, showing, correctly LIMIT " + sev + ";";

if(!obj\_query.exec(str))

qDebug()<<"Не удалось открыть таблицу БД";

QSqlRecord rec = obj\_query.record();

while(obj\_query.next())

{

QString ru = obj\_query.value(rec.indexOf("ru")).toString();

QString eng = obj\_query.value(rec.indexOf("eng")).toString();

QString n = obj\_query.value(rec.indexOf("numberImg")).toString();

int s = obj\_query.value(rec.indexOf("showing")).toInt();

int c = obj\_query.value(rec.indexOf("correctly")).toInt();

int r = obj\_query.value(rec.indexOf("rating")).toInt();

intermediateWord = new Word(ru, eng, n, s, c, r);//создали указатель на объект слова

questionList->push\_back(intermediateWord);//добавили указатель на слово в список

}

}

Метод и *read30FromDB:*

void TrainingWindow::**read30FromDB**()//считывание необходимых 30% слов из БД

{

QString thit = QString::number(userWordsCount - seventy);//оставшиеся 30% в строку

QStringList tmp;//список для проверки по ru

QList<Word\*>::iterator i;

for(i = questionList->begin(); i!=questionList->end(); ++i){

tmp.push\_back("'" + (\*i)->getRu() + "'");//вставляем в tmp русские слова из questionList

}

QString s = tmp.join(", ");//преобразовали в строку с разделителями

QSqlQuery obj\_query;//объект для управления запросами

QString str = "SELECT ru, eng, numberImg, showing, correctly, rating FROM words\_table WHERE ru NOT IN (" + s + ") ORDER BY RANDOM() LIMIT " + thit + ";";

if(!obj\_query.exec(str))

qDebug()<<"Не удалось открыть таблицу БД";

QSqlRecord rec = obj\_query.record();

while(obj\_query.next())

{

QString ru = obj\_query.value(rec.indexOf("ru")).toString();

QString eng = obj\_query.value(rec.indexOf("eng")).toString();

QString n = obj\_query.value(rec.indexOf("numberImg")).toString();

int s = obj\_query.value(rec.indexOf("showing")).toInt();

int c = obj\_query.value(rec.indexOf("correctly")).toInt();

int r = obj\_query.value(rec.indexOf("rating")).toInt();

intermediateWord = new Word(ru, eng, n, s, c, r);//создали указатель на объект слова

questionList->push\_back(intermediateWord);//добавили указатель на слово в список

}

}

При каждом ответе характеристики слов изменяются и следующий список уже не будет таким как предыдущие.

# 7. Выводы

В ходе выполнения работы было создано приложение, позволяющее хранить в себе базу, необходимых пользователю слов, и отрабатывать их.

Разработан и реализован алгоритм формирования тренировочного списка, благодаря которому пользователь подвергается равномерной нагрузке.

Разработан приятный дизайн и интуитивно-понятный интерфейс приложения.

# Литература

1. Капитанов Д.В. Лекции по С++.
2. Городецкий С.Ю. Лекции по Qt Creator.
3. <http://doc.crossplatform.ru/qt/>

# Приложение

mainwindow.cpp:

#include <QMessageBox>

#include "mainwindow.h"

#include "ui\_mainwindow.h"

MainWindow::**MainWindow**(QWidget \*parent) : QMainWindow(parent), ui(new Ui::MainWindow)

{

ui->setupUi(this);

ui->mainToolBar->addAction(ui->aboutApplication);

ui->mainToolBar->addAction(ui->actionhelp);

dataBaseIsOK();//Проверка открытия или создание БД и запись количества слов в ней

ui->labelWordsCount->setText(QString::number(wordsCount));

ui->lineWordCount->setFocus();//фокус на ввод кол-ва слов

animationShow();//плавное появление главного окна

}

MainWindow::~***MainWindow***()

{

delete ui;

}

void MainWindow::**animationShow**()//анимация появления

{

setWindowOpacity(0);

QPropertyAnimation\* animation = new QPropertyAnimation(this, "windowOpacity");

animation->setDuration(300);

animation->setStartValue(0);

animation->setEndValue(1);

animation->start();

}

void MainWindow::**dataBaseIsOK**()//Проверка открытия или создание БД и запись количества слов в ней

{

//-------------SQLLite----------------

// qDebug()<<fullDBName;

QSqlDatabase dbase = QSqlDatabase::addDatabase("QSQLITE");//создали объект БД;

bool isDB\_Open = false;//открыта или закрыта БД

if(!QFile(fullDBName).exists())//если файла не существует

{

dbase.setDatabaseName(fullDBName);//утанавливаем адрес с именем базы

if (!dbase.open())//база создаётся и если не открывается

{

qDebug() << "Ошибка при создании + открытии БД";

}

qDebug() << "Создали новую БД";

QString str = "CREATE TABLE words\_table (ru text, eng text, numberImg integer, showing integer, correctly integer, rating integer);";//строка запроса создания таблицы

QSqlQuery obj\_query;//объект для управления запросами

bool q = obj\_query.exec(str);//выполнился ли запрос?

if(!q)

{

qDebug() << "Не создали таблицу";

}

qDebug() << "Добавили таблицу в БД";

str = "INSERT INTO words\_table VALUES ('кот', 'cat', 0, 0, 0, 0);";//строка запроса вставки строки

q = obj\_query.exec(str);//выполнился ли запрос?

if(!q)

{

qDebug() << "Не внесли в таблицу строку";

}

isDB\_Open = true;//база открыта

qDebug() << "Внесли в таблицу строку";

}

if(!isDB\_Open){//елси файл существует

dbase.setDatabaseName(fullDBName);//утанавливаем адрес с именем базы

if (!dbase.open())//если база не открывается

{

qDebug() << "Ошибка при открытии файла БД";

}

isDB\_Open = true;//база открыта

qDebug()<<"Нашли и открыли файл БД";

}

QSqlQuery obj\_query;//объект для управления запросами

if(!obj\_query.exec("SELECT COUNT(numberImg) FROM words\_table;"))

qDebug()<<"Не удалось открыть таблицу БД";

obj\_query.first();

wordsCount = obj\_query.value(0).toInt();//записали количество слов в базе

dbase.close();//закрыли БД

}

void MainWindow::**showWordsCountSlot**()//слот показа кол-ва доступных слов и отображения главного окна

{

showWordsCount();//показали кол-во доступных слов

show();//показали окно

}

void MainWindow::**showWordsCount**()//показать кол-во доступных слов

{

QSqlDatabase dbase = QSqlDatabase::addDatabase("QSQLITE");//создали объект БД;

dbase.setDatabaseName(fullDBName);//утанавливаем адрес с именем базы

if (!dbase.open())//если база не открывается

{

qDebug() << "Ошибка при открытии файла БД";

}

QSqlQuery obj\_query;//объект для управления запросами

if(!obj\_query.exec("SELECT COUNT(numberImg) FROM words\_table;"))

qDebug()<<"Не удалось открыть таблицу БД";

obj\_query.first();

wordsCount = obj\_query.value(0).toInt();//записали количество слов в базе

dbase.close();//закрыли БД

ui->labelWordsCount->setText(QString::number(wordsCount));

}

void MainWindow::**on\_btnDictionary\_clicked**()//кнопка открытия словаря

{

wordListDialog = new WordsListWindow();//создали диалоговое окно для добавления новых слов

wordListDialog->setAttribute(Qt::WA\_DeleteOnClose);//очистка памяти при закрытии диалогового окна

//соединение сигнала и слота обновления кол-ва слов в главном окне и показа главного окна

connect(wordListDialog, SIGNAL(showWordsCountSignal()), this, SLOT (showWordsCountSlot()));

wordListDialog->show();//показали окно словаря

hide();//скрыли главное окно

}

void MainWindow::**on\_btnStart\_clicked**()//слот запуска окна тренировок

{

if(ui->lineWordCount->text().isEmpty())//проверка на ввод желаемого числа слов

{

QMessageBox msgBox;

msgBox.setWindowTitle("Информация");

msgBox.setText("Введите желаемое количество слов (не более " + QString::number(wordsCount) + ")");

msgBox.*exec*();

return;

}

QRegExp num("^[0-9]+$");//регулярное выражение на проверку на ЧИСЛО желаемых слов

if(!num.exactMatch(ui->lineWordCount->text()))

{

ui->lineWordCount->clear();

return;

}

if(ui->lineWordCount->text().toInt() <= 0)//проверка на ввод желаемого числа слов

{

QMessageBox msgBox;

msgBox.setWindowTitle("Информация");

msgBox.setText("Ха-ха-ха... Очень смешно.");

msgBox.*exec*();

return;

}

userWordsCount = ui->lineWordCount->text().toInt();//желаемое число слов

if (userWordsCount > wordsCount)//проверка, чтобы введенное число не превышало кол-во доступных слов

{

QMessageBox msgBox;

msgBox.setWindowTitle("Информация");

msgBox.setText("Число должно быть не больше " + QString::number(wordsCount));

msgBox.*exec*();

return;

}

bool ru\_eng = true;//по умолчанию рус-англ тест

if(ui->radioEng\_Ru->isChecked()) ru\_eng = false;//если стоит другой флаг, то англ-рус тест

trainingDialog = new TrainingWindow(nullptr, wordsCount, userWordsCount, ru\_eng);//создали диалоговое окно для тренировки

trainingDialog->setAttribute(Qt::WA\_DeleteOnClose);//очистка памяти при закрытии диалогового окна

//соединение сигнала и слота обновления кол-ва слов в главном окне и показа главного окна

connect(trainingDialog, SIGNAL(showMainWindowSignal()), this, SLOT (showWordsCountSlot()));

trainingDialog->show();//показали тренировочное окно

hide();//скрыли главное окно

}

void MainWindow::**on\_btnExit\_clicked**()//слот выхода из программы

{

qApp->quit();

}

void MainWindow::**on\_aboutApplication\_triggered**()//слот о приложении

{

QMessageBox::about(this, tr("О приложении Мой словарь"),

tr("<h2>Мой словарь 1.0</h2>"

"<p>Приложение \"Мой словарь\" предназначено для "

"тренировки английских слов.</p>"

"<p>Отличительными особенностями программы являются "

"возможность добавления изображений, ассоциирующихся со словами, а так же автоматическое распределение слов "

"по успеваемости пользователя. Новые и плохо изученные слова всплывают чаще, но иногда будут "

"появляться и хорошо отработанные, для дополнительного контроля.</p>"));

}

void MainWindow::**on\_actionhelp\_triggered**()//обработчик кнопки Справка

{

helpDialog\* help = new helpDialog(this);

help->setAttribute(Qt::WA\_DeleteOnClose);//очистка памяти при закрытии диалогового окна

help->show();

}

trainingwindow.cpp:

#include <QDebug>

#include <iterator>

#include <QMessageBox>

#include <QTimer>

#include <QCloseEvent>

#include "trainingwindow.h"

#include "ui\_trainingwindow.h"

TrainingWindow::**TrainingWindow**(QWidget \*parent, int c, int u, bool rn) : QDialog(parent), ui(new Ui::TrainingWindow)

{

ui->setupUi(this);

setWindowFlags(windowFlags () & (~Qt::WindowContextHelpButtonHint));//окно будет без вопростельного знака

ru\_eng = rn;//вариант тестирования

count = c;//количество слов

userWordsCount = u;//задаем количество желаемых слов от параметра "u"

currentNumWord = 0;//номер текущего слова

questionList = new QList<Word\*>();

clickImg.load(":/res/img/hoverIMG.png");

answerTrue.load(":/res/img/answerT.png");

answerFalse.load(":/res/img/answerF.png");

ui->btnExit->hide();//скрыли картику выхода

int w = ui->labelImg->width();//ширина лейбла картинки

int h = ui->labelImg->height();//высота лейбла картинки

ui->labelImg->setPixmap(clickImg.scaled(w, h, Qt::KeepAspectRatio));//положили главную картинку в лейбл

calcPercentWords();//вычисление процентов слов

readToList();//считываем желаемое кол-во слов в список для тренировки

startTest();

showWord();

}

TrainingWindow::~***TrainingWindow***()

{

delete ui;

QList<Word\*>::iterator i;//итератором бежим по списку указателей на Word(ы) и удаляем каждый

for(i = questionList->begin(); i!=questionList->end(); ++i){

delete \*i;

}

delete questionList;//удаляем список

}

void TrainingWindow::**calcPercentWords**()//вычисление процентов слов

{

seventy = (userWordsCount/100)\*70;//70%

thirty = userWordsCount - seventy;//30%

}

void TrainingWindow::**readToList**()//считывание слов из БД

{

QSqlDatabase dbase = QSqlDatabase::addDatabase("QSQLITE");//создали объект БД;

dbase.setDatabaseName(fullDBName);//утанавливаем адрес с именем базы

if (!dbase.open())//если база не открывается

{

qDebug() << "Ошибка при открытии файла БД";

}

read70FromDB();//считываем 70%

read30FromDB();//считываем 30%

dbase.close();

}

void TrainingWindow::**read70FromDB**()//считывание необходимых 70% слов из БД

{

QSqlQuery obj\_query;//объект для управления запросами

QString sev = QString::number(seventy);//перевели в строку

QString str = "SELECT ru, eng, numberImg, showing, correctly, rating FROM words\_table ORDER BY rating, showing, correctly LIMIT " + sev + ";";

if(!obj\_query.exec(str))

qDebug()<<"Не удалось открыть таблицу БД";

QSqlRecord rec = obj\_query.record();

while(obj\_query.next())

{

QString ru = obj\_query.value(rec.indexOf("ru")).toString();

QString eng = obj\_query.value(rec.indexOf("eng")).toString();

QString n = obj\_query.value(rec.indexOf("numberImg")).toString();

int s = obj\_query.value(rec.indexOf("showing")).toInt();

int c = obj\_query.value(rec.indexOf("correctly")).toInt();

int r = obj\_query.value(rec.indexOf("rating")).toInt();

intermediateWord = new Word(ru, eng, n, s, c, r);//создали указатель на объект слова

questionList->push\_back(intermediateWord);//добавили указатель на слово в список

}

}

void TrainingWindow::**read30FromDB**()//считывание необходимых 30% слов из БД

{

QString thit = QString::number(userWordsCount - seventy);//оставшиеся 30% в строку

QStringList tmp;//список для проверки по ru

QList<Word\*>::iterator i;

for(i = questionList->begin(); i!=questionList->end(); ++i){

tmp.push\_back("'" + (\*i)->getRu() + "'");//вставляем в tmp русские слова из questionList

}

QString s = tmp.join(", ");//преобразовали в строку с разделителями

QSqlQuery obj\_query;//объект для управления запросами

QString str = "SELECT ru, eng, numberImg, showing, correctly, rating FROM words\_table WHERE ru NOT IN ("

+ s + ") ORDER BY RANDOM() LIMIT " + thit + ";";

if(!obj\_query.exec(str))

qDebug()<<"Не удалось открыть таблицу БД";

QSqlRecord rec = obj\_query.record();

while(obj\_query.next())

{

QString ru = obj\_query.value(rec.indexOf("ru")).toString();

QString eng = obj\_query.value(rec.indexOf("eng")).toString();

QString n = obj\_query.value(rec.indexOf("numberImg")).toString();

int s = obj\_query.value(rec.indexOf("showing")).toInt();

int c = obj\_query.value(rec.indexOf("correctly")).toInt();

int r = obj\_query.value(rec.indexOf("rating")).toInt();

intermediateWord = new Word(ru, eng, n, s, c, r);//создали указатель на объект слова

questionList->push\_back(intermediateWord);//добавили указатель на слово в список

}

}

void TrainingWindow::**startTest**()//запуск тестирования

{

it = questionList->begin();//устанавливаем итератор на начало списка

}

void TrainingWindow::**showWord**()//отобразить слово и установаить скрытую картинку

{

ui->lineAnswer->clear();//очистили поле ввода

if(ru\_eng) ui->labelTest->setText((\*it)->getRu());//отображаем тестируемое русское слово

else ui->labelTest->setText((\*it)->getEng());//отображаем тестируемое английское слово

if((\*it)->getnameImg()=="0"){

hiddenImg.load(":/res/img/noImg.png");//если имя картинки "0", то загружаем пустую

}

else {

hiddenImg.load(dir + QDir::separator() + (\*it)->getnameImg() + ".png");//иначе загружаем из папки

}

setStyleSheet("#TrainingWindow{background-color: white;}");

ui->lineAnswer->show();//показали поле для ответа

ui->lineAnswer->setFocus();//установили фокус на поле ввода

ui->labelTrueFalse->clear();//чистим индикатор ответа

ui->btnOk->show();//показали кнопку ОК

ui->btnSkip->show();//показали кнопку Пропуск

currentNumWord++;

ui->labelNumWord->setText(QString::number(currentNumWord) + " из " + QString::number(userWordsCount));//отображение номера текущего слова из userWordsCount слов

}

void TrainingWindow::**on\_btnShowImg\_pressed**()//обработчик удержания кнопки

{

int w = ui->labelImg->width();//ширина лейбла картинки

int h = ui->labelImg->height();//высота лейбла картинки

ui->labelImg->setPixmap(hiddenImg.scaled(w, h, Qt::KeepAspectRatio));//показали скрытую картинку

}

void TrainingWindow::**on\_btnShowImg\_released**()//обработчик отпускания кнопки показа изображения

{

int w = ui->labelImg->width();//ширина лейбла картинки

int h = ui->labelImg->height();//высота лейбла картинки

ui->labelImg->setPixmap(clickImg.scaled(w, h, Qt::KeepAspectRatio));//картинка с призывом нажать

}

void TrainingWindow::**on\_btnOk\_clicked**()//обработчик кнопки ОК

{

if(ui->lineAnswer->text().isEmpty())//проверка на пустой ответ

{

QMessageBox msgBox;

msgBox.setWindowTitle("Осечка");

msgBox.setText("Введите ответ. Желательно правильный.");

msgBox.*exec*();

ui->lineAnswer->setFocus();//установили фокус на поле ввода

return;

}

ui->lineAnswer->hide();//скрыли поле для ответа

ui->btnOk->hide();//скрыли кнопку ОК

ui->btnSkip->hide();//скрыли кнопку Пропуск

if(ru\_eng)//если тест рус-англ

{

if(ui->lineAnswer->text().toLower() == (\*it)->getEng().toLower())//проверка на правильный ответ рус-англ

answerCorrectly();//действия при верном ответе

else

answerWrong();//действия при НЕверном ответе

}

else//если тест англ-рус

{

if(ui->lineAnswer->text().toLower() == (\*it)->getRu().toLower())//проверка на правильный ответ англ-рус

answerCorrectly();//действия при верном ответе

else

answerWrong();//действия при НЕверном ответе

}

QTimer::singleShot(1000, this, SLOT(nextOrShowResult()));//по таймеру следующее слово или выход

}

void TrainingWindow::**answerCorrectly**()//действия при верном ответе

{

int w = ui->labelTrueFalse->width();//ширина лейбла картинки

int h = ui->labelTrueFalse->height();//высота лейбла картинки

ui->labelTrueFalse->setPixmap(answerTrue.scaled(w, h, Qt::KeepAspectRatio));//индикатор - верно

setStyleSheet("#TrainingWindow{background-color: rgb(217,255,189);}");//зеленый фон

cor++;//увеличили количество верных ответов

saveCorrectly();//верный ответ

}

void TrainingWindow::**answerWrong**()//действия при НЕверном ответе

{

int w = ui->labelTrueFalse->width();//ширина лейбла картинки

int h = ui->labelTrueFalse->height();//высота лейбла картинки

ui->labelTrueFalse->setPixmap(answerFalse.scaled(w, h, Qt::KeepAspectRatio));//индикатор -НЕ верно

setStyleSheet("#TrainingWindow{background-color: rgb(255,187,188);}");//красный фон

wr++;//увелеичили количество Не верных ответов

saveWrong();//НЕ верный ответ

}

void TrainingWindow::**saveCorrectly**()//сохранили верный ответ

{

qDebug()<<"Ответ верный";

(\*it)->addShowing();

(\*it)->addCorrectly();

(\*it)->addRating();

QSqlDatabase dbase = QSqlDatabase::addDatabase("QSQLITE");//создали объект БД;

dbase.setDatabaseName(fullDBName);//утанавливаем адрес с именем базы

if (!dbase.open())//если база не открывается

{

qDebug() << "Ошибка при открытии файла БД";

}

QString ru = (\*it)->getRu();

QString eng = (\*it)->getEng();

QString s = QString::number((\*it)->getShowing());//преобразуем число в строку

QString c = QString::number((\*it)->getCorrectly());//преобразуем число в строку

QString r = QString::number((\*it)->getRating());//преобразуем число в строку

QSqlQuery obj\_query;//объект для управления запросами

QString str = "UPDATE words\_table SET showing = " + s + ", correctly = " + c + ", rating = " + r + " WHERE ru = '" + ru + "' AND eng = '" + eng + "';";

if(!obj\_query.exec(str))

qDebug()<<"Не удалось изменить статистику БД";

dbase.close();

}

void TrainingWindow::**saveWrong**()//сохранили неверный ответ

{

qDebug()<<"Ответ НЕ верный";

(\*it)->addShowing();

(\*it)->downRating();

QSqlDatabase dbase = QSqlDatabase::addDatabase("QSQLITE");//создали объект БД;

dbase.setDatabaseName(fullDBName);//утанавливаем адрес с именем базы

if (!dbase.open())//если база не открывается

{

qDebug() << "Ошибка при открытии файла БД";

}

QString ru = (\*it)->getRu();

QString eng = (\*it)->getEng();

QString s = QString::number((\*it)->getShowing());//преобразуем число в строку

QString r = QString::number((\*it)->getRating());//преобразуем число в строку

QSqlQuery obj\_query;//объект для управления запросами

QString str = "UPDATE words\_table SET showing = " + s + ", rating = " + r + " WHERE ru = '" + ru + "' AND eng = '" + eng + "';";

if(!obj\_query.exec(str))

qDebug()<<"Не удалось изменить статистику БД";

dbase.close();

}

void TrainingWindow::**on\_btnSkip\_clicked**()//обработчик кнопки пропуск

{

ui->lineAnswer->hide();//скрыли поле для ответа

ui->btnOk->hide();//скрыли кнопку ОК

ui->btnSkip->hide();//скрыли кнопку Пропуск

int w = ui->labelTrueFalse->width();//ширина лейбла картинки

int h = ui->labelTrueFalse->height();//высота лейбла картинки

ui->labelTrueFalse->setPixmap(answerFalse.scaled(w, h, Qt::KeepAspectRatio));//индикатор -НЕ верно

setStyleSheet("#TrainingWindow{background-color: rgb(255,187,188);}");//красный фон

skip++;//увеличили количество пропусков

saveWrong();//засчитывается как не верный ответ

qDebug()<<"Пропустили";

QTimer::singleShot(1000, this, SLOT(nextOrShowResult()));//по таймеру следующее слово или выход

}

void TrainingWindow::**nextOrShowResult**()//слот следующего слова или выхода

{

it++;//переходим на следующее слово

if(it == questionList->end())//если следующего слова нет

{

qDebug()<<"Закончили";

showResult();//отображаем результат

return;

}

showWord();//отображаем слово

}

void TrainingWindow::**showResult**()//показать результат

{

setStyleSheet("#TrainingWindow{background-color: white;}");//делаем фон белым

//скрываем лишние элементы

ui->labelTest->hide();

ui->btnShowImg->hide();

ui->lineAnswer->hide();

ui->labelNumWord->hide();

ui->labelTrueFalse->hide();

ui->btnSkip->hide();

ui->btnOk->hide();

//чистим центральный лейбл и отображаем в нем резульат

ui->labelImg->clear();

ui->labelImg->setText("Количество верных ответов: " + QString::number(cor) + "\n" +

"Количество ошибок: " + QString::number(wr) + "\n" +

"Количество пропусков: " + QString::number(skip));

ui->btnExit->show();//показали кнопку выхода

}

void TrainingWindow::**on\_btnExit\_clicked**()//обработчик кнопки выхода

{

close();

}

void TrainingWindow::***closeEvent***(QCloseEvent \*event)//событие закрытия окна тренировки

{

showMainWindowSignal();//сигнал показа главного окна

event->accept(); //подтверждаем событие - добросовестно закрываем приложение

}

wordslistwindow:

#include "wordslistwindow.h"

#include "ui\_wordslistwindow.h"

#include <QPixmap>

#include <QDir>

#include <QMessageBox>

#include <QMouseEvent>

WordsListWindow::**WordsListWindow**(QWidget \*parent) :

QDialog(parent),

ui(new Ui::WordsListWindow)

{

ui->setupUi(this);

setWindowFlags(Qt::Dialog | Qt::CustomizeWindowHint | Qt::WindowTitleHint /\*| Qt::FramelessWindowHint\*/);//окно будет без кнопок

mapRU = new QMap<QString, QVector<QString>>();

mapENG = new QMap<QString, QVector<QString>>();

readDB();//Считывание БД

showList();//Отображаем список слов

ui->labelWordsCount->setText(QString::number(wordsCount));

}

WordsListWindow::~***WordsListWindow***()

{

delete ui;

delete mapRU;

delete mapENG;

}

void WordsListWindow::**readDB**()//Считывание БД

{

QSqlDatabase dbase = QSqlDatabase::addDatabase("QSQLITE");//создали объект БД;

dbase.setDatabaseName(fullDBName);//утанавливаем адрес с именем базы

if (!dbase.open())//если база не открывается

{

qDebug() << "Ошибка при открытии файла БД";

}

QSqlQuery obj\_query;//объект для управления запросами

if(!obj\_query.exec("SELECT MAX(numberImg) FROM words\_table;"))

qDebug()<<"Не удалось открыть таблицу БД";

obj\_query.first();

numberImg = obj\_query.value(0).toInt();//прочитали номер самой последней картинки

if(!obj\_query.exec("SELECT ru, eng, numberImg, showing, correctly, rating FROM words\_table;"))

qDebug()<<"Не удалось открыть таблицу БД";

QSqlRecord rec = obj\_query.record();

while(obj\_query.next())

{

QVector<QString> tmpVecRU;//создали временный вектор RU

QVector<QString> tmpVecENG;//создали временный вектор ENG

tmpVecRU.push\_back(obj\_query.value(rec.indexOf("eng")).toString());//вставляем слова в вектор tmpVecRU.push\_back(obj\_query.value(rec.indexOf("numberImg")).toString());//вставляем слова в вектор tmpVecRU.push\_back(obj\_query.value(rec.indexOf("showing")).toString());//вставляем статистику tmpVecRU.push\_back(obj\_query.value(rec.indexOf("correctly")).toString());//вставляем статистику tmpVecRU.push\_back(obj\_query.value(rec.indexOf("rating")).toString());//вставляем статистику tmpVecENG.push\_back(obj\_query.value(rec.indexOf("ru")).toString());//вставляем слова в вектор tmpVecENG.push\_back(obj\_query.value(rec.indexOf("numberImg")).toString());//вставляем слова в вектор

tmpVecENG.push\_back(obj\_query.value(rec.indexOf("showing")).toString());//вставляем статистику tmpVecENG.push\_back(obj\_query.value(rec.indexOf("correctly")).toString());//вставляем статистику tmpVecENG.push\_back(obj\_query.value(rec.indexOf("rating")).toString());//вставляем статистику

mapRU->insert(obj\_query.value(rec.indexOf("ru")).toString(), tmpVecRU);//заносим информацию в мапRU

mapENG->insert(obj\_query.value(rec.indexOf("eng")).toString(), tmpVecENG);//заносим информацию в мапENG

}

dbase.close();

wordsCount = mapRU->count();//запоминаем количество слов

}

void WordsListWindow::**showList**()//Отображаем список слов

{

QMap<QString, QVector<QString>>::iterator i;

if(ui->radioButtonRu\_Eng->isChecked())

{

ui->listWords->clear();//чистим список

for (i = mapRU->begin(); i != mapRU->end(); ++i){

ui->listWords->addItem(i.key() + " - " + i.value()[0]);

}

}

else if(ui->radioButtonEng\_Ru->isChecked())

{

ui->listWords->clear();//чистим список

for (i = mapENG->begin(); i != mapENG->end(); ++i){

ui->listWords->addItem(i.key() + " - " + i.value()[0]);

}

}

}

void WordsListWindow::**addWordInDB**(QString ru, QString eng, int number)//добавление нового слова в БД

{

QSqlDatabase dbase = QSqlDatabase::addDatabase("QSQLITE");//создали объект БД;

dbase.setDatabaseName(fullDBName);//утанавливаем адрес с именем базы

if (!dbase.open())//если база не открывается

{

qDebug() << "Ошибка при открытии файла БД";

}

QString newName = QString::number(number);//преобразуем число в строку

QSqlQuery obj\_query;//объект для управления запросами

QString str = "INSERT INTO words\_table VALUES('" + ru + "', '" + eng + "', "+ newName +", 0, 0, 0);";

if(!obj\_query.exec(str))

qDebug()<<"Не удалось занести строку в БД";

dbase.close();

}

void WordsListWindow::**addWordInMaps**(QString ru, QString eng, int number)//добавление нового слова в мапы

{

QString newName = QString::number(number);//преобразуем число в строку

QVector<QString> tmpVecRU;//создали временный вектор RU

QVector<QString> tmpVecENG;//создали временный вектор ENG

tmpVecRU.push\_back(eng);//вставляем слова в вектор

tmpVecRU.push\_back(newName);//вставляем слова в вектор

tmpVecRU.push\_back("0");//вставляем статистику

tmpVecRU.push\_back("0");//вставляем статистику

tmpVecRU.push\_back("0");//вставляем статистику

tmpVecENG.push\_back(ru);//вставляем слова в вектор

tmpVecENG.push\_back(newName);//вставляем слова в вектор

tmpVecENG.push\_back("0");//вставляем статистику

tmpVecENG.push\_back("0");//вставляем статистику

tmpVecENG.push\_back("0");//вставляем статистику

mapRU->insert(ru, tmpVecRU);//заносим информацию в мапRU

mapENG->insert(eng, tmpVecENG);//заносим информацию в мапENG

}

void WordsListWindow::**on\_btnAddWord\_clicked**()//обработчик добавления нового слова

{

addWordDialog = new DialogAddWord(this);//создали диалоговое окно для добавления новых слов

addWordDialog->setAttribute(Qt::WA\_DeleteOnClose);//очистка памяти при закрытии диалогового окна

//соединение сигнала и слота сохранения новых слов и картинки с адресом

connect(addWordDialog, SIGNAL(getRuEngAddressSignal(QString, QString, int)), this, SLOT (saveRuEngNumberSlot(QString, QString, int)));

//соединение сигнала и слота предоставленияимени новой картинки

connect(this, SIGNAL(provideNewImageNameSignal(QString)), addWordDialog, SLOT (saveNewImageSlot(QString)));

addWordDialog->show();//показали форму

}

void WordsListWindow::**delWordFromDB**(QString keyRu)//удалить слово из базы данных

{

QSqlDatabase dbase = QSqlDatabase::addDatabase("QSQLITE");//создали объект БД;

dbase.setDatabaseName(fullDBName);//утанавливаем адрес с именем базы

if (!dbase.open())//если база не открывается

{

qDebug() << "Ошибка при открытии файла БД";

}

QSqlQuery obj\_query;//объект для управления запросами

QString str = "DELETE FROM words\_table WHERE ru = '" + keyRu + "';";

if(!obj\_query.exec(str))

qDebug()<<"Не удалось удалить строку из БД";

dbase.close();

}

void WordsListWindow::**deleteWord**(int itemInt)//удалить слово

{

QString str = ui->listWords->item(itemInt)->text();//текст выбранной строки

QStringList lst = str.split(" ");//Делим текст на слова через пробел

QString keyRu;//ключ Ru

QString keyEng;//ключ Eng

if(ui->radioButtonRu\_Eng->isChecked())

{

keyRu = lst.at(0);//ключ Ru - первая строка из lst

keyEng = lst.at(2);//ключ Eng - третья строка из lst

}

else if(ui->radioButtonEng\_Ru->isChecked())

{

keyRu = lst.at(2);//ключ Ru - третья строка из lst

keyEng = lst.at(0);//ключ Eng - первая строка из lst

}

delWordFromDB(keyRu);//удалить слово из базы данных

delWordFromMaps(keyRu, keyEng);//удалить слово из Мапов

ui->listWords->takeItem(itemInt);//удалить слово из списка

clearStat();//очищаем картинку и статистику

wordsCount--;

ui->labelWordsCount->setText(QString::number(wordsCount));

}

void WordsListWindow::**clearStat**()//очистить статистику и картинку

{

ui->labelImg->clear();//очистить картинку

ui->labelShowing->clear();//очистить статистику

ui->labelCorrectly->clear();//очистить статистику

ui->labelRating->clear();//очистить статистику

}

void WordsListWindow::**delWordFromMaps**(QString keyRu, QString keyEng)//удалить слово из Мапов

{

QString nameDelImage = mapRU->value(keyRu)[1];//имя удаляемой картинки

mapRU->remove(keyRu);//удаляем значение из русского

mapENG->remove(keyEng);//удаляем значение из английского

if(nameDelImage != "0")

delImageFromDir(nameDelImage);//удалить картинку из папки

}

void WordsListWindow::**delImageFromDir**(QString name)//удалить картинку из папки

{

QString fullNImg = QCoreApplication::applicationDirPath() + QDir::separator() + "saved\_image" +

QDir::separator() + name + ".png";//полное имя файла с картинкой

QFile file(fullNImg);

file.remove();//удаляем файл картинки

}

void WordsListWindow::**on\_btnDelWord\_clicked**()//обработчик кнопки удаления слова

{

int itemInt = -1;

itemInt = ui->listWords->currentRow();//номер выбранной строки

if(itemInt == -1) return;

if(ui->listWords->count() <= 3)

{

QMessageBox msgBox;

msgBox.setWindowTitle("Информация");

msgBox.setText("Словарь должен содержать минимум 3 слова.");

msgBox.*exec*();

return;

}

QMessageBox msgBox;

msgBox.setWindowTitle("Важно");

msgBox.setText("Удалить слово без возможности восстановления?");

msgBox.setStandardButtons(QMessageBox::Ok | QMessageBox::Cancel);

msgBox.setIcon(QMessageBox::Question);

msgBox.setDefaultButton(QMessageBox::Ok);

int res = msgBox.*exec*();

if (res == QMessageBox::Ok) //нажата кнопка Ok

deleteWord(itemInt);

else //отмена

return;

}

void WordsListWindow::**saveRuEngNumberSlot**(QString ru, QString eng, int name)//слот изменения и предоставления имени новой картинки

{

if(name != 0)//если картинка новая, то создаем адрес с новым именем и отправляем сигналом

{

numberImg++;

QString newName = QString::number(numberImg);//преобразуем число в строку

emit provideNewImageNameSignal(newName);//сигнал передачи адреса и имени новой картинки

name = numberImg;

}

addWordInDB(ru, eng, name);

addWordInMaps(ru, eng, name);

clearStat();//очищаем картинку и статистику

wordsCount++;

ui->labelWordsCount->setText(QString::number(wordsCount));

showList();

}

void WordsListWindow::**on\_listWords\_itemClicked**(QListWidgetItem \*item)//слот отображения картинки выделенного элемента и его статистики

{

QString key = item->text();//берем текст из выделенного элемента

QStringList lst = key.split(" ");//Делим строку на слова через пробел

key = lst.at(0);//ключ - первая строка из lst

QString imageName;

if(ui->radioButtonRu\_Eng->isChecked())

{

imageName = mapRU->value(key)[1];//значение последнего элемента (вектора по ключу)

ui->labelShowing->setText("Показы: " + mapRU->value(key)[2]);

ui->labelCorrectly->setText("Правильные ответы: " + mapRU->value(key)[3]);

ui->labelRating->setText("Рейтинг: " + mapRU->value(key)[4]);

}

else if(ui->radioButtonEng\_Ru->isChecked())

{

imageName = mapENG->value(key)[1];//значение последнего элемента (вектора по ключу)

ui->labelShowing->setText("Показы: " + mapENG->value(key)[2]);

ui->labelCorrectly->setText("Правильные ответы: " + mapENG->value(key)[3]);

ui->labelRating->setText("Рейтинг: " + mapENG->value(key)[4]);

}

QString fullFN =":/res/img/noImg.png";

if(imageName != "0")

{

fullFN = QCoreApplication::applicationDirPath() + QDir::separator() + "saved\_image" +

QDir::separator() + imageName + ".png";//полное имя файла с картинкой

}

QPixmap img(fullFN);

int w = ui->labelImg->width();//ширина лейбла картинки

int h = ui->labelImg->height();//высота лейбла картинки

ui->labelImg->setPixmap(img.scaled(w, h, Qt::KeepAspectRatio));//положили картинку в лейбл

}

void WordsListWindow::**on\_radioButtonRu\_Eng\_toggled**(bool checked)//отображаем Рус - Англ

{

clearStat();

showList();

}

void WordsListWindow::**on\_radioButtonEng\_Ru\_toggled**(bool checked)//отображаем Англ - Рус

{

clearStat();

showList();

}

void WordsListWindow::**on\_btnClose\_clicked**()//кнопка закрытия окна

{

emit showWordsCountSignal();//сигнал показа в MainWindow кол-ва доступных слов

close();

}

dialogaddword.cpp:

#include <QFileDialog>//диалоговое окно выбора файла

#include <QDir>//Для поиска дирректории и разделителей

#include <QRegExp>

#include <QMessageBox>

#include "dialogaddword.h"

#include "ui\_dialogaddword.h"

DialogAddWord::**DialogAddWord**(QWidget \*parent) : QDialog(parent), ui(new Ui::DialogAddWord)

{

ui->setupUi(this);

setWindowFlags(windowFlags () & (~Qt::WindowContextHelpButtonHint));//окно будет без вопростельного знака

savedPic.load(":/res/img/noImg.png");//пустая картинка

int w = ui->labelNewImage->width();//ширина лейбла новой картинки

int h = ui->labelNewImage->height();//высота лейбла новой картинки

ui->labelNewImage->setPixmap(savedPic.scaled(w, h, Qt::KeepAspectRatio));//положили пустую картинку в лейбл новой картинки

if (savedPic == QPixmap(":/res/img/noImg.png"))

ui->btnClearImage->hide();

ui->lineAddRus->setFocus();

}

DialogAddWord::~***DialogAddWord***()

{

delete ui;

}

void DialogAddWord::**saveNewImageSlot**(QString newName)//слот установки имени и сохранения новой картинки

{

QString dir = QCoreApplication::applicationDirPath() + QDir::separator() + "saved\_image";//адрес папки с картинками

if(!QDir(dir).exists())//если папки с картинками не существует

QDir().mkpath(dir);//то создать

ui->labelNewImage->pixmap()->save(dir + QDir::separator() + newName + ".png", "png");//сохраняем картинку

}

void DialogAddWord::**on\_btnChangeImage\_clicked**()//Добавление своей картинки

{

QString dirPath = QCoreApplication::applicationDirPath();//адрес открытия диалогового окна

QString nameFile = QFileDialog::getOpenFileName(this,

tr("Выбор изображения"),//Заголовок

dirPath, //Настройка на нужную папку.

tr("Файлы изображений (\*.png \*.jpg \*.bmp)\n"

"Изображения в формате PNG (\*.png)\n"

"Изображения в формате JPG (\*.jpg)\n"

"Изображения в формате BMP (\*.bmp)"));

if (nameFile.isEmpty()) return;//Файл не выбран

savedPic.load(nameFile);//загружаем выбранную картинку в сохраняемое изображение

int w = ui->labelNewImage->width();//ширина лейбла новой картинки

int h = ui->labelNewImage->height();//высота лейбла новой картинки

ui->labelNewImage->setPixmap(savedPic.scaled(w, h, Qt::KeepAspectRatio));//положили пустую картинку в лейбл новой картинки

ui->btnClearImage->show();//показали форму

}

void DialogAddWord::**on\_btnClearImage\_clicked**()//обработка кнопки очистки изображения

{

if(savedPic != QPixmap(":/res/img/noImg.png"))//если в сохраняемой картинке не лежит пустое изображение

{

savedPic.load(":/res/img/noImg.png");

int w = ui->labelNewImage->width();//ширина лейбла картинки

int h = ui->labelNewImage->height();//высота лейбла картинки

ui->labelNewImage->setPixmap(savedPic.scaled(w, h, Qt::KeepAspectRatio));//положили пустую картинку в лейбл новой картинки

ui->btnClearImage->hide();//скрыли форму

}

}

void DialogAddWord::**on\_btnFinishAddWord\_clicked**()//обработка кнопки Готово

{

if(ui->lineAddRus->text().isEmpty() || (ui->lineAddEng->text().isEmpty()))//если одна или другая строка в добавлении пуста

{

return;

}

QRegExp simbolsRu("^[[а-яА-ЯёЁ]+$");//регулярное выражение на проверку строки (только на русские символы)

QRegExp simbolsEng("^[A-Za-z]+$");//регулярное выражение на проверку строки (только на английские символы)

if(!simbolsRu.exactMatch(ui->lineAddRus->text()) || !simbolsEng.exactMatch(ui->lineAddEng->text()))

{

QMessageBox msgBox;

msgBox.setWindowTitle("Введены не корректные данные");

msgBox.setText("Слова должны соответствовать своим полям. Любые символы кроме русских и английских букв не допускаются.");

msgBox.*exec*();

return;

}

int name = 0;

if(savedPic != QPixmap(":/res/img/noImg.png"))//если в сохраняемой картинке не лежит пустое изображение

{

name = 1;

}

emit getRuEngAddressSignal(ui->lineAddRus->text(), ui->lineAddEng->text(), name);//сигнал запроса имени новой картинки

close();

}