МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ «ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНО-ЭКОНОМИЧЕСКИЙ КОЛЛЕДЖ»

Специальность 2-40 01 01 «Программное обеспечение

информационных технологий»

**КУРСОВОЙ ПРОЕКТ**

по дисциплине: «Основы алгоритмизации и программирования»

**на тему: «Разработка мобильного приложения «Учим английские слова для IT»**

Пояснительная записка

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Разработал |  |  |  |  | Иванов С. С. |
|  | (дата) |  | (подпись) |  | (ФИО) |
| Руководитель |  |  |  |  | Глушенок А. В. |
|  | (дата) |  | (подпись) |  | (ФИО) |

Гомель, 2022

СОДЕРЖАНИЕ

[Введение 4](#_Toc169474547)

[1 Символьный и строковый тип данных 5](#_Toc169474548)

[2 Описание алгоритмов и технологии решения 8](#_Toc169474549)

[2.1 Требование к программному продукту 8](#_Toc169474550)

[2.2 Описание работы алгоритмов функций 8](#_Toc169474551)

[2.3 Проектирование интерфейса и технологии решения 9](#_Toc169474552)

[3 Выполнение практической части 16](#_Toc169474553)

[3.1 Описание интерфейса программы 16](#_Toc169474554)

[3.2 Описание алгоритмов работы программы 21](#_Toc169474555)

[4 Результаты работы 26](#_Toc169474556)

[Заключение 43](#_Toc169474557)

[Список использованных источников 44](#_Toc169474558)

[Приложение А 45](#_Toc169474559)

# Введение

Целью данного проекта является разработка мобильного приложения «Учим английские слова».

Мобильное приложение — программное изделие, разновидность прикладного программного обеспечения, предназначенная для работы на смартфонах, планшетах и других мобильных устройствах.

Мобильного приложения «Учим английские слова» представляет собой направление в области образования. Современный мир характеризуется повышенным интересом к изучению иностранных языков, а мобильные приложения стали незаменимым инструментом в этом процессе благодаря своей доступности и удобству использования. Проект ориентирован на разработку приложения, которое поможет пользователям улучшить свой уровень владения английским языком.

Для достижения цели необходимо выполнить ряд задач:

* Анализ требований;
* Разработка теоретического материала;
* Проектирование структуры базы данных приложения;
* Проектирование интерфейса;
* Разработка функциональности приложения;
* Тестирование и отладка.

Среда разработки, используемая в процессе создания программного продукта – «Android Studio». «Android Studio» — интегрированная среда разработки для работы с платформой «Android», анонсированная 16 мая 2013 года на конференции «Google I/O». В последней версии «Android Studio» поддерживается «Android» 4.1 и выше.

Язык, используемый для разработки – «Kotlin». «Kotlin» — статически типизированный, объектно-ориентированный язык программирования, работающий на основе «Java Virtual Machine» и разрабатываемый компанией «JetBrains».

Результатом курсового проекта станет разработанная информационно-поисковая система, предоставляющая теоретический материал, задания разных типов для закрепления пройденного материала и статистка прохождения заданий.

# 1 СИМВОЛЬНЫЙ И СТРОКОВЫЙ ТИП ДАННЫХ

Символьный тип — тип данных, предназначенный для хранения одного символа (управляющего или печатного) в определённой кодировке. Может являться как однобайтовым (для стандартной таблицы символов), так и многобайтовым (к примеру, для Юникода).

Строковый тип — тип данных, значениями которого является произвольная последовательность символов алфавита. Каждая переменная такого типа может быть представлена фиксированным количеством байтов либо иметь произвольную длину.

Основные проблемы в машинном представлении строкового типа:

* строки могут иметь достаточно существенный размер, до нескольких десятков мегабайтов;
* изменяющийся со временем размер — возникают трудности с добавлением и удалением символов.

В представлении строк в памяти компьютера существует два принципиально разных подхода: представление массивом символов, метод «завершающего байта», в виде списка.

При представлении массивом символов размер массива хранится в отдельной области. От названия языка «Pascal», где этот метод был впервые реализован, данный метод получил название «Pascal strings».

Более оптимизированным вариантом этого метода является т. н. формат «c-addr u». В отличие от «Pascal strings», здесь размер массива хранится не совместно со строковыми данными, а является частью указателя на строку.

Преимущества представления в виде массива символов:

* программа в каждый момент времени содержит сведения о размере строки, поэтому операции добавления символов в конец, копирования строки и собственно получения размера строки выполняются достаточно быстро;
* строка может содержать любые данные;
* возможно на программном уровне следить за выходом за границы строки при её обработке;
* возможно быстрое выполнение операции вида «взятие N-ого символа с конца строки».

Недостатки представления в виде массива символов:

* проблемы с хранением и обработкой символов произвольной длины;
* увеличение затрат на хранение строк — значение «длина строки» также занимает место и в случае большого количества строк маленького размера может существенно увеличить требования алгоритма к оперативной памяти;
* ограничение максимального размера строки. В современных языках программирования это ограничение скорее теоретическое, так как обычно размер строки хранится в 32-битовом поле, что даёт максимальный размер строки в 4 294 967 295 байт (4 гигабайта);
* при использовании алфавита с переменным размером символа (например, «UTF-8»), в размере хранится не количество символов, а именно размер строки в байтах, поэтому количество символов необходимо считать отдельно.

Метод «завершающего байта» заключается в том, что одно из возможных значений символов алфавита (как правило, это символ с кодом 0) выбирается в качестве признака конца строки, и строка хранится как последовательность байтов от начала до конца. Есть системы, в которых в качестве признака конца строки используется не символ 0, а байт «0xFF» (255) или код символа «$».

Метод имеет три названия — «ASCIIZ», «C-strings» и метод нуль-терминированных строк.

Преимущества метода «завершающего байта»:

* отсутствие дополнительной служебной информации о строке (кроме завершающего байта);
* возможность представления строки без создания отдельного типа данных;
* отсутствие ограничения на максимальный размер строки;
* экономное использование памяти;
* простота получения суффикса строки;
* простота передачи строк в функции (передаётся указатель на первый символ);

Недостатки метода «завершающего байта»:

* долгое выполнение операций получения длины и конкатенации строк;
* отсутствие средств контроля за выходом за пределы строки, в случае повреждения завершающего байта возможность повреждения больших областей памяти, что может привести к непредсказуемым последствиям — потере данных, краху программы и даже всей системы;
* невозможность использовать символ завершающего байта в качестве элемента строки;
* невозможность использовать некоторые кодировки с размером символа в несколько байт (например, «UTF-16»), так как во многих таких символах, например «Ā» (0x0100), один из байтов равен нулю (в то же время, кодировка «UTF-8» свободна от этого недостатка).

Метод представление в виде списка делает язык более «теоретически элегантным» за счёт соблюдения ортогональности в системе типов, но приносит существенные потери быстродействия.

Процедуры и функции:

Простейшие операции со строками:

* получение символа по индексу — в большинстве языков это тривиальная операция;
* конкатенация строк.

Производные операции:

* получение подстроки по индексам начала и конца;
* проверка вхождения одной строки в другую (поиск подстроки в строке);
* проверка на совпадение строк (с учётом или без учёта регистра символов);
* получение длины строки;
* замена подстроки в строке.

Операции при трактовке строк как списков:

* свёртка;
* отображение одного списка на другой;
* фильтрация списка по критерию.

Продвинутые операции:

* нахождение минимальной надстроки, содержащей все указанные строки;
* поиск в двух массивах строк совпадающих последовательностей.

До появления стандарта Юникод в 1991 году, один символ обычно кодировался одним байтом из 8 двоичных битов или меньше — 7-битные, 6-битные. 8-битые кодировки позволяли представлять 256 возможных значений. Однако для полноценного представления символов алфавитов нескольких языков 256 символов недостаточно. Для решения этой проблемы применялись разные подходы.

Переключение языка управляющими кодами. Метод не стандартизирован и лишает текст самостоятельности (то есть последовательность символов без управляющего кода в начале теряет смысл);

Использование двух или более байт для представления каждого символа («UTF-16», «UTF-32»). Главным недостатком этого метода является потеря совместимости с предыдущими библиотеками для работы с текстом при представлении строки как «ASCIIZ». Например, концом строки должен считаться уже не байт со значением 0, а два или четыре подряд идущих нулевых байта, в то время как одиночный байт «0» может встречаться в середине строки, что сбивает библиотек.

Использование кодировки с переменным размером символа. Например, в «UTF-8» часть символов представляется одним байтом, часть двумя, тремя или четырьмя. Этот метод позволяет сохранить частичную совместимость со старыми библиотеками, но приводит к невозможности прямой адресации символа в памяти по номеру его позиции в строке.

# 2 Описание алгоритмов и технологии решения

2.1 Требование к программному продукту

Программное средство должно обрабатывать базу данных, иметь возможность создания новой записи или просмотра существующих записей.

Функционал программного продукта должен включать в себя функции:

* создания записи;
* чтения записи;
* обновления записи;
* удаления записи;
* поиска;
* сортировки;
* представление теоретического материала;
* выполнения заданий разных типов для закрепления пройденного материала;
* отслеживание статистки прохождения заданий;

Программа должна иметь 2 формы:

* Приветственная с названием курсового проекта и автором;
* Форма обработки файла;

В программе должна осуществляться валидация введённых пользователем данных

База данных содержит таблицы:

* «words\_table» содержащая следующие поля:
  1. «ID» — уникальный не пустой целочисленный автоинкрементный основной столбец характеризующий номер строки в базе данных;
  2. «ENGLISH NAME» ­­­­­­­­­­— уникальный не пустой строковый столбец, хранящий английское представление слова;
  3. «RUSSIAN NAME» ­­­­­­­­­­— уникальный не пустой строковый столбец, хранящий русское представление слова;
  4. «WORD DEFENITION» — строковый столбец, хранящий определение слова.
* «test find translation table», «test find word table», «test true false table» содержащие следующие поля:
  1. «ID» — целочисленное основное не пустое уникальное автоинкрементное поле;
  2. «RESULT» — не пустое текстовое поле, содержащее результаты прохождения теста;
  3. «RESULT PROCENT» — не пустое целочисленное поле, содержащее результаты прохождения теста в виде процентного соотношения.

2.2 Описание работы алгоритмов функций

Для создания записи необходимо нажать на кнопку и форма измениться на форму добавления записи, после этого необходимо будет ввести данные, нажать на кнопку, при этом данные из изменяемых полей проверяются на содержание в базе данных, в случае если они присутствуют пользователь получит сообщение об этом, в случае если поля пустые пользователь получит сообщение об ошибке, в ином случае поля будут добавлены в новую запись в базе данных и пользователь получит сообщение о успешном, а запись будет добавлена в базу данных и форма измениться на предыдущую.

Для чтения записи необходимо выбрать соответствующую запись в списке и изменяемые поля заполнятся содержанием записи.

Для обновления записи необходимо нажать на соответствующую кнопку и содержимое записи с соответствующим номером будет проверено на содержание в базе данных и если оно не содержится, то заменено на данные из изменяемых полей, в ином случае пользователь получит сообщение об ошибке удаления записи.

Для поиска по теоретическому материалу необходимо нажать на кнопку и ввести символы, содержащиеся в искомом слове, при этом все записи в базе данных проверяются на содержание введённых символов и в случае, если они содержат, то содержимое списка записей обновляется, в списке отображаются только поля, содержащие последовательность введённых символов, в случае ни одной записи не найдено пользователь получит сообщение.

Для сортировки списка необходимо нажать на соответствующую кнопку и порядок содержимого списка будет обновлен в соответствии с видом сортировки.

Для просмотра теоретического материала пользователю необходимо нажать на кнопку, после этого на экране отобразится новая форма с теоретическим материалом, при этом с начала отображается верхнее меню и содержание выпадающего меню. Для отображения теоретического материала в списке все поля берутся из базы данных и отображаются в списке.

Для выполнения заданий разных типов для закрепления пройденного материала необходимо в выпадающем меню выбрать нужный вариант и на экране отобразится приветственная форма и параметры теста, для начала теста необходимо нажать на соответствующую кнопку и форма измениться на тестовые задания, при этом содержание параметров теста применяется к тесту, для выбора ответа необходимо нажать на кнопку, при этом ответ записывается в временное хранилище, после завершения теста результаты теста рассчитываются и вносятся в базу данных.

Для отслеживания статистки прохождения заданий необходимо выбрать соответствующий пункт и форма измениться на форму с историей прохождения тестов, при этом содержание базы данных заносятся в список пройденных тестов.

2.3 Проектирование интерфейса и технологии решения

В ходе разработки проекта были разработаны следующие формы.

Встречная форма, изображена на рисунке 2.1. Данная форма представляет из себя приветственное окно с названием проекта, именем разработчика и кнопкой для начала использования приложения.

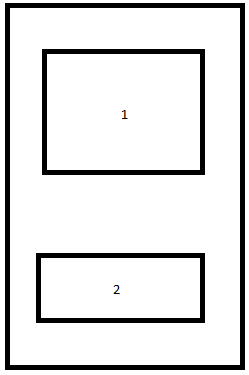


Рисунок 2.1 – Встречная форма: 1 – панель с встречной информации; 2 – панель для кнопки начала;

Главная форма, изображен на рисунке 2.2. Данная форма представляет из себя выпадающее меню, а в нем уже располагаются все формы приложения.

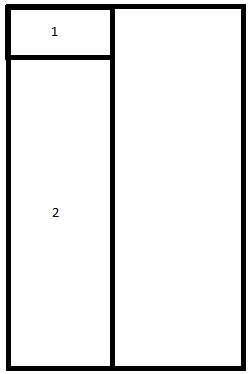


Рисунок 2.2. – Главная форма: 1 – панель с названием приложения; 2 – панель навигации по приложению;

Форма добавления записи, изображен на рисунке 2.3. Данная форма представляет из себя форму для добавления новых записей в базу данных.

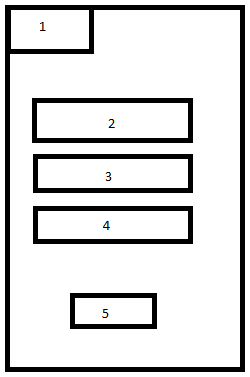


Рисунок 2.3 - Форма добавления записи: 1 – панель для кнопки выхода; 2 – панель для английского представления слова; 3 – панель для русского представления слова; 4 – панель для определения слова; 5 – панель для кнопки добавления;

Форма изменения базы данных приложения, изображена на рисунке 2.4. Данная форма представляет из себя форму для изменения базы данных.

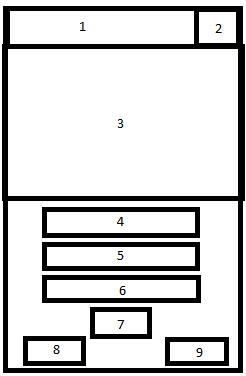


Рисунок 2.4 – Форма изменения базы данных приложения; 1 – панель поиска; 2 – панель для кнопки обновления списка; 3 – панель для списка записей; 4 – панель для английского определения слова; 5 – панель для русского определения слова; 6 – панель для определения слова; 7 – панель для кнопки сохранения изменений; 8 – кнопка для удаления записи; 9 – кнопка для добавления записи;

Форма теста, изображен на рисунке 2.5. Данная форма представляет из себя форму для настройки и начала теста. Представляет из себя название теста, поле для введения количества раундов и кнопки начала теста.

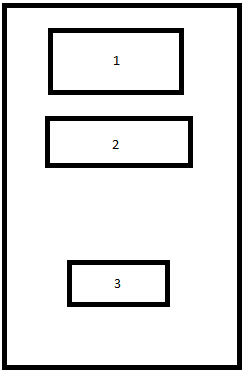


Рисунок 2.5 – Форма теста; 1 – панель с названием теста; 2 – панель для ввода количества раундов; 3 – панель для запуска теста;

Форма прохождения тестов, изображенная на рисунке 2.6. Данная форма представляет из себя форму для прохождения тестов. Представляет из себя слово в английской или русской версии и кнопки выбора вариантов этого слова.

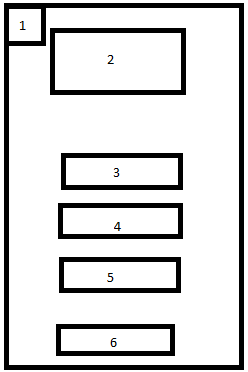


Рисунок 2.6 - Форма прохождения тестов; 1 – панель для кнопки выхода; 2 – панель для вопроса теста; 3 – панель для первого варианта выбора; 4 – панель для второго варианта выбора; 5 - панель для третьего варианта выбора; 6- панель для ответа;

Форма словаря, изображенная на рисунке 2.3.7. Данная форма представляет из себя форму, содержащую весь теоретический материал.

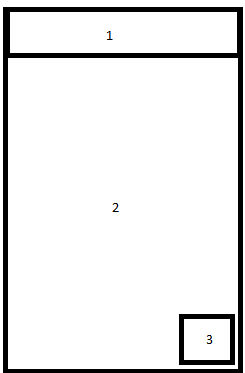


Рисунок 2.7 - Форма словаря; 1 – панель поиска; 2 – панель для списка записей; 3 – панель для добавления записи;

Форма статистики, изображенная на рисунке 2.8. Данная форма представляет из себя форму содержащую статистику прохождения тестов.

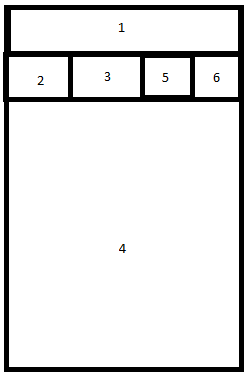


Рисунок 2.8 - Форма статистики: 1 – панель с названием раздела; 2, 3, 5, 6 – панели с названиями тестов; 4 – панель с историей теста;

Фрагмент, изображенный на рисунке 2.9. Представляет из себя элемент списка, представляющий запись из базы данных;

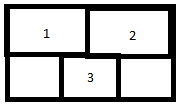


Рисунок 2.9 - Фрагмент списка: 1 – панель для английского слова; 2 – панель для русского слова; 3 – панель для определения слова;

Используемые компоненты:

* «Activity» – представляет собой формы приложения;
* «Fragment» – представляет собой элементы форм;
* «EditText» – поле для текстового ввода пользователем;
* «Button» – кнопка для выполнения действия при нажатии;
* «TextView» – отображение текста на экране;
* «LinearLayout» – размещение элементов в один столбец или строку;
* «Toolbar» – панель инструментов в верхней части экрана с кнопками и заголовком;
* «NavigationView» – панель навигации с боковым выдвижным меню;
* «RecyclerView» – отображение списка элементов с возможностью прокрутки;
* «ConstraintLayout» – размещение элементов с помощью ограничений, например, сверху, снизу, слева, справа;
* «FloatingActionButton» – кнопка с плавающим действием;
* «SearchView» – поле для поиска с возможностью фильтрации данных.

Для работы с базой данных будет использована библиотека «android database sqlite».

Библиотека «android database sqlite» в «Kotlin» предоставляет набор классов и интерфейсов для работы с базами данных «SQLite» на «Android». Она позволяет создавать, открывать, читать, записывать и удалять данные в базе данных «SQLite».

Основные компоненты:

* «SQLiteOpenHelper» – этот класс является базовым классом для работы с базами данных SQLite. Он обеспечивает автоматическое создание и обновление базы данных, а также предоставляет методы для получения доступа к объекту SQLiteDatabase;
* «SQLiteDatabase» – этот класс представляет собой открытое соединение с базой данных SQLite. Он позволяет выполнять SQL-запросы, добавлять, обновлять и удалять данные;
* «Cursor» – этот класс представляет собой набор строк из результатов SQL-запроса. Он позволяет перебирать строки и получать доступ к значениям столбцов.

# 3 Выполнение практической части

3.1 Описание интерфейса программы

В ходе разработки программы был реализован следующий пользовательский интерфейс.

Встречная форма, изображенная на рисунке 3.1. Данная форма представляет из себя приветственное окно, на котором расположено название проекта, указан разработчик, ссылка на исходный код и кнопка для начала использования приложения.

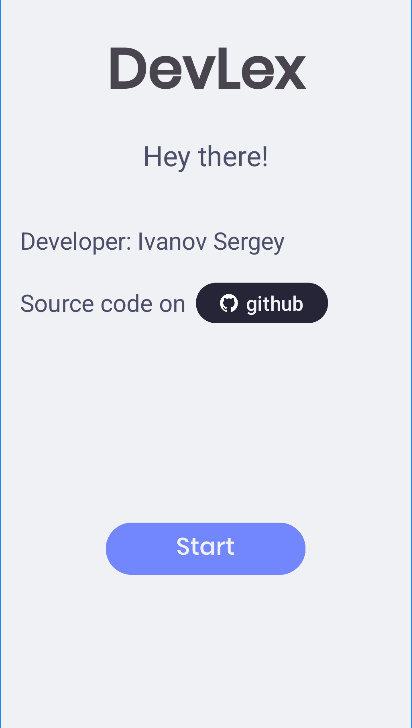


Рисунок 3.1 – Встречная форма

Главная форма, изображённая на рисунке 3.2. Данная форма представляет из себя выпадающее меню, изначально установлен фрагмент меню «Lexicon» на котором расположен весь теоретический материал.

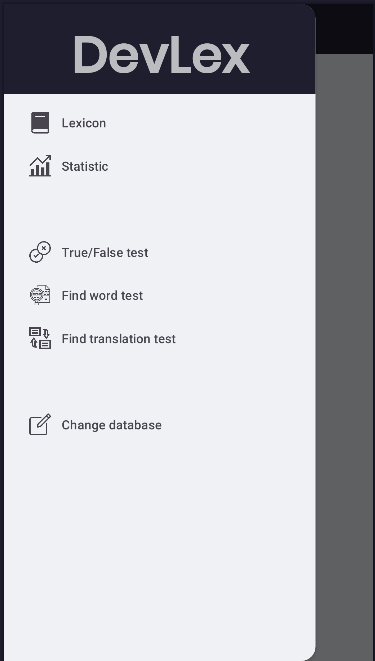


Рисунок 3.2 – Главная форма

Форма добавления новой записи, изображённая на рисунке 3.3. Данная форма представляет из себя форму для добавления новых записей в базу данных. Содержит 3 поля для ввода информации и две кнопки: назад и добавить.

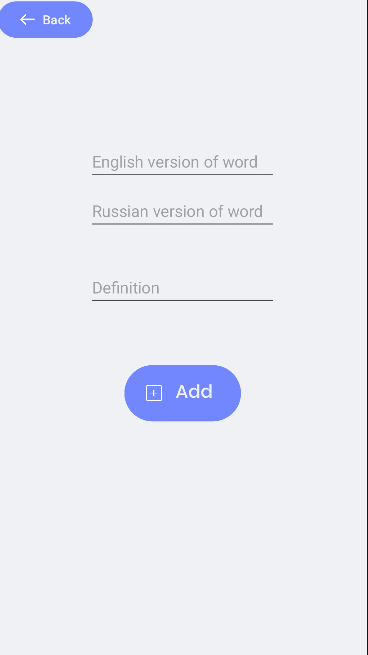


Рисунок 3.3 – Форма добавления новой записи

Фрагмент изменения базы данных, изображенный на рисунке 3.4. Данный фрагмент представляет из себя форму для изменения базы данных. Пользователю представляется список теоретического материала, поисковая строка, кнопка обновления списка и пустые текстовые поля. При нажатии на строку в списке текстовые поля заполняются её содержимым, и пользователь может с ним взаимодействовать.

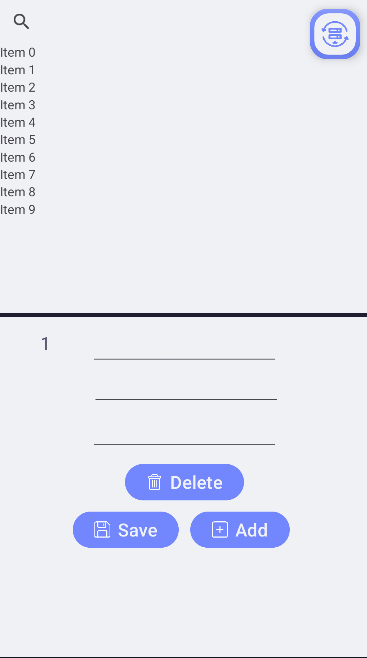


Рисунок 3.4 – Фрагмент изменения базы данных

Фрагмент «Найди перевод», изображен на рисунке 3.5. Данный фрагмент представляет из себя форму для настройки и начала теста «Найди перевод». Пользователю представляется название теста, поле для ввода числовых значений характеризующих количество раундов и кнопка для начала теста. Фрагменты «Найди слово» и «Правда/Ложь» выглядят идентично.

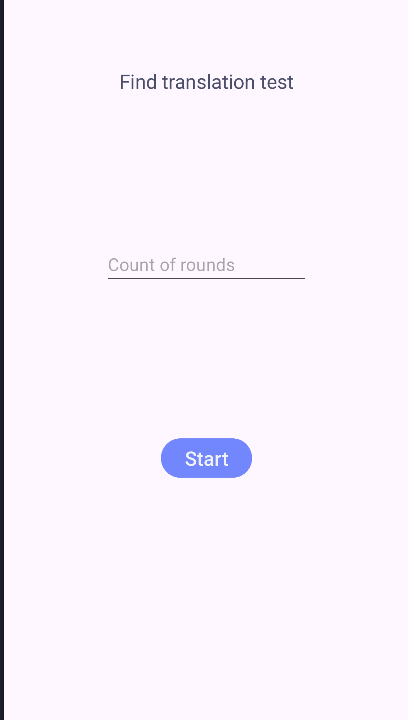


Рисунок 3.5 - Фрагмент «fragment\_find\_translation\_test»

Форма «activity\_find\_tranlation\_test», изображенная на рисунке 3.6. Данная форма представляет из себя форму для прохождения теста «Find translation test». Пользователю представляется случайное английское слово, для которого он должен выбрать правильный перевод из представленных на кнопках.

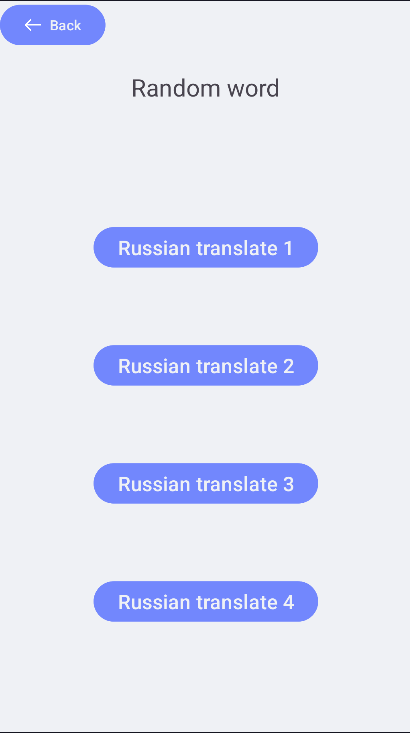


Рисунок 3.6 - Форма «activity\_find\_tranlation\_test»

Фрагмент «fragment\_find\_word\_test», изображенный на рисунке 3.7. Данный фрагмент представляет из себя форму для прохождения теста «Find word test». Пользователю представляется случайное значение слово, если оно существует и 4 английских слова, пользователь должен выбрать соответствующее определению слово.

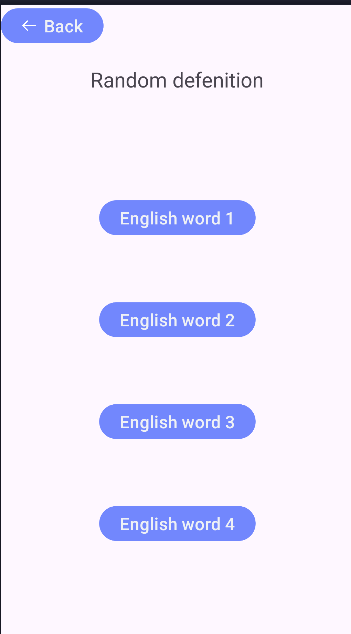


Рисунок 3.7 - Форма «activity\_find\_word\_test»

Фрагмент «fragment\_lexicon», изображен на рисунке 3.8. Данный фрагмент представляет из себя форму, содержащую весь теоретический материал. Пользователю представляется поисковая строка, кнопка для обновления списка, кнопка для добавления новой записи и сам список содержащий теоретический материал.

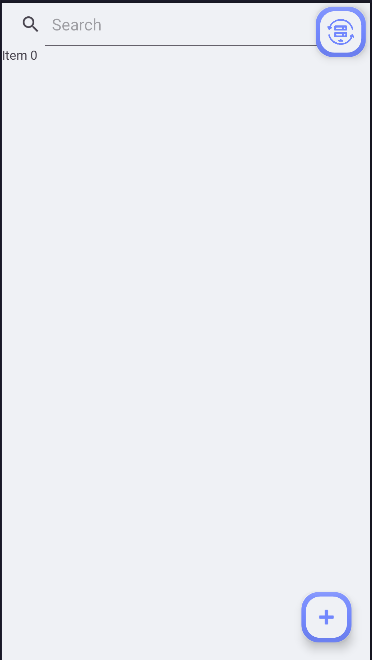


Рисунок 3.8 - Фрагмент «fragment\_lexicon»

Фрагмент «fragment\_statistic», изображен на рисунке 3.9. Данный фрагмент представляет из себя форму содержащую статистику прохождения тестов. Пользователю представляется списки, содержащие историю прохождения тестов, и средние значения прохождения тестов.

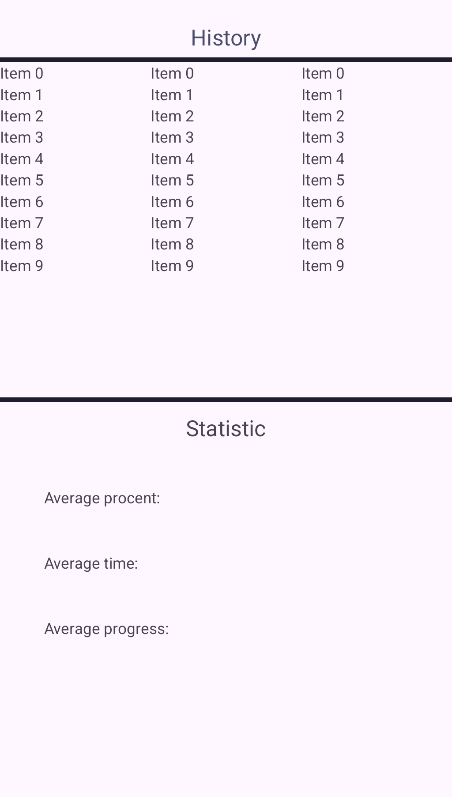


Рисунок 3.9 - Форма «fragment\_statistic»

Форма «activity\_true\_false\_test», изображен на рисунке 3.10. Данная форма представляет из себя форму прохождения теста «Find word test». Пользователю представляется случайное английское слово и случайное русское слово, пользователю должен выбрать соответствует ли перевод.

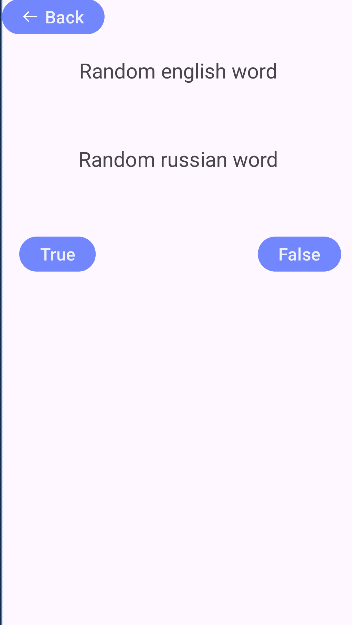


Рисунок 3.10 - Форма «activity\_true\_false\_test»

3.2 Описание алгоритмов работы программы

Разработана процедура, обрабатывающая нажатие пользователя на кнопки, код процедуры изображен на рисунке 3.11. Все последующие обработки нажатий на кнопки реализованы похожим образом.

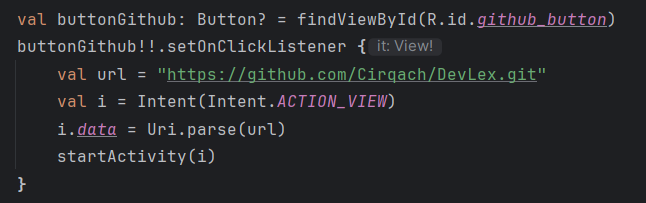


Рисунок 3.11 – Код процедуры нажатия на кнопку.

Для использования базы данных на устройстве разработана процедура, отвечающая за копирование базы данных в директорию приложения для её использования. Код процедуры представлен на рисунке 3.12.

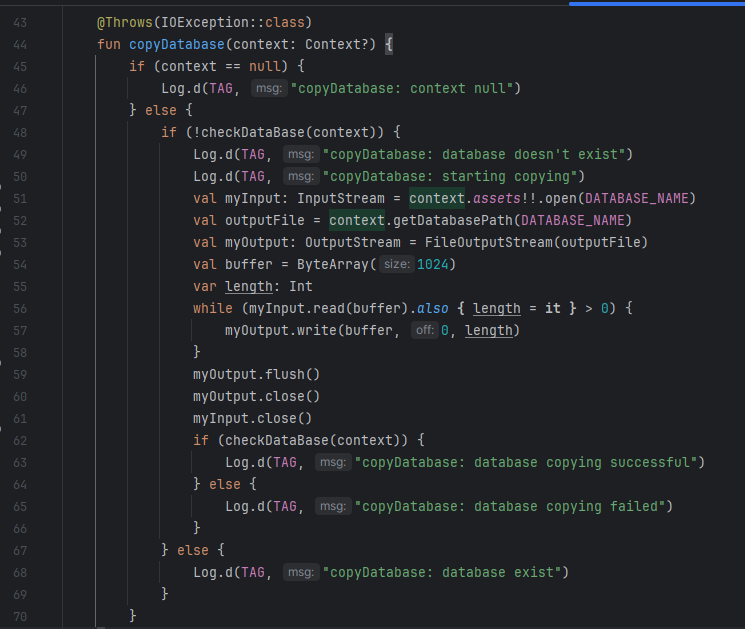


Рисунок 3.12 – Код процедуры копирования базы данных

Разработана функция, отвечающий за добавление данных в таблицу словаря, код функции представлен на рисунке 3.13. Для таблиц тестов используется похожий код.

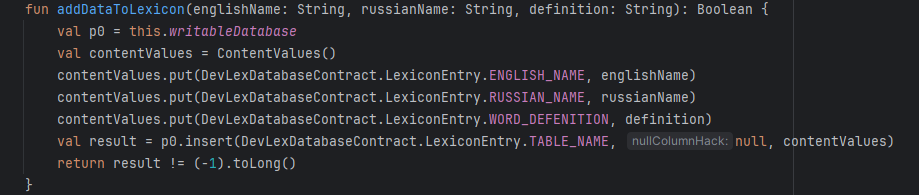


Рисунок 3.13 – Код функции добавления данных в таблицу словаря

Для чтения всех данных из таблицы создана функция, которая использует «SQL» код для запроса к таблице, передаваемой в параметрах функции. Код представлен на рисунке 3.14.

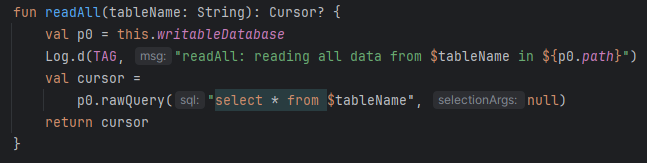


Рисунок 3.14 – Код функция чтение всех данных из таблицы

Для удаления данных из таблицы по его уникальному идентификатору разработана функция, которая принимает название таблицы и уникальный идентификатор. Код функции представлен на рисунке 3.15.

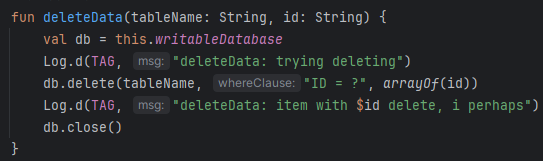


Рисунок 3.15 – Функция удаление данных из таблицы по «ID»

Разработана функция сохранения данных в таблицу словаря, функция принимает все поля таблицы, в качестве указателя для сохранения данных используется передаваемый уникальный идентификатор. Код функции представлен на рисунке 3.16. Для сохранения результатов тестов используется похожие функции.

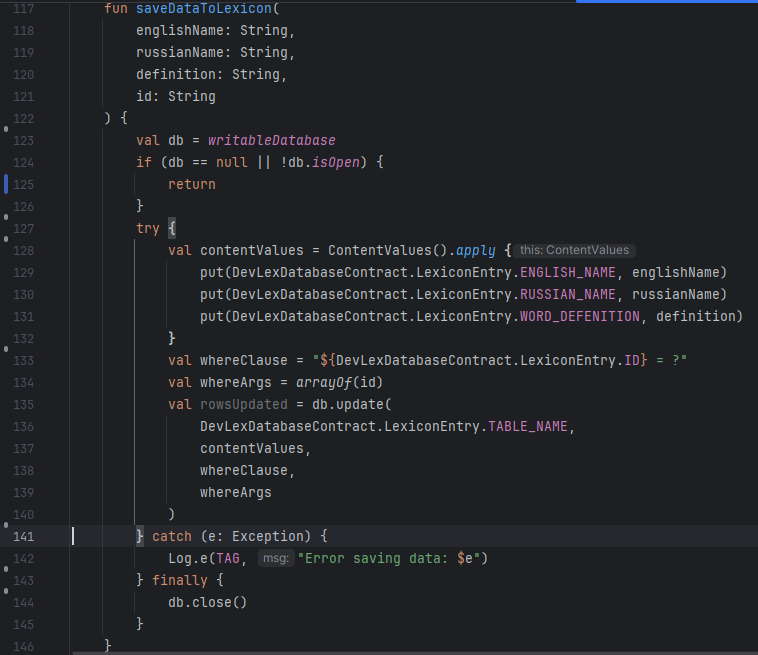


Рисунок 3.16 – Функция сохранения данных в словарь

Для получения конкретных данных по уникальному идентификатору создана функция, код функции изображен на рисунке 3.17. Для получения данных из таблиц тестов используются похожие функции.

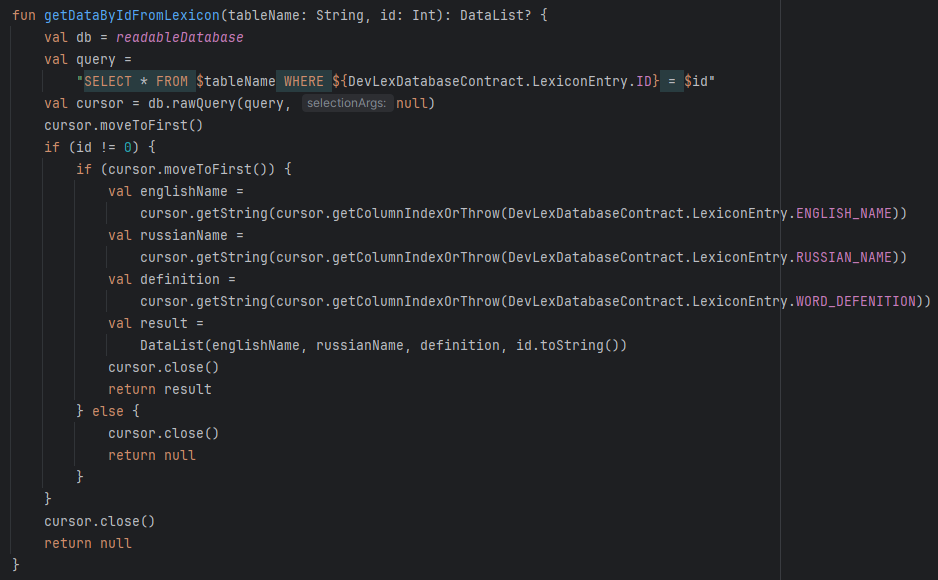


Рисунок 3.17 – Взятие данные из словаря по «ID»

Для проверки данных на существование создана функция, принимающая название таблицы и уникальный идентификатор на проверку. Код функции представлен на рисунке 3.18.

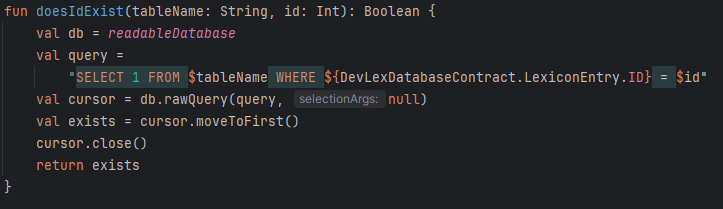


Рисунок 3.18 – Проверка данных на существование

Для валидации данных, введенных пользователем создан следующий фрагмент кода, он проверяет не пусты ли обязательные поля и не существуют ли введенные данные. Фрагмент кода изображен на рисунке 3.19.

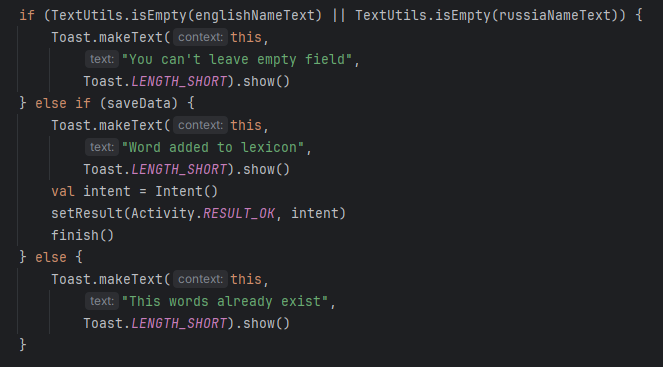


Рисунок 3.19 – Фрагмент кода валидации введенных данных

Полный код программы приведен в приложении А.

# 4 Результаты работы

При открытии программы появляется начальная форма, содержащая информацию о программе и кнопку входа на основную форму. На рисунке 4.1 представлена появившиеся форма.

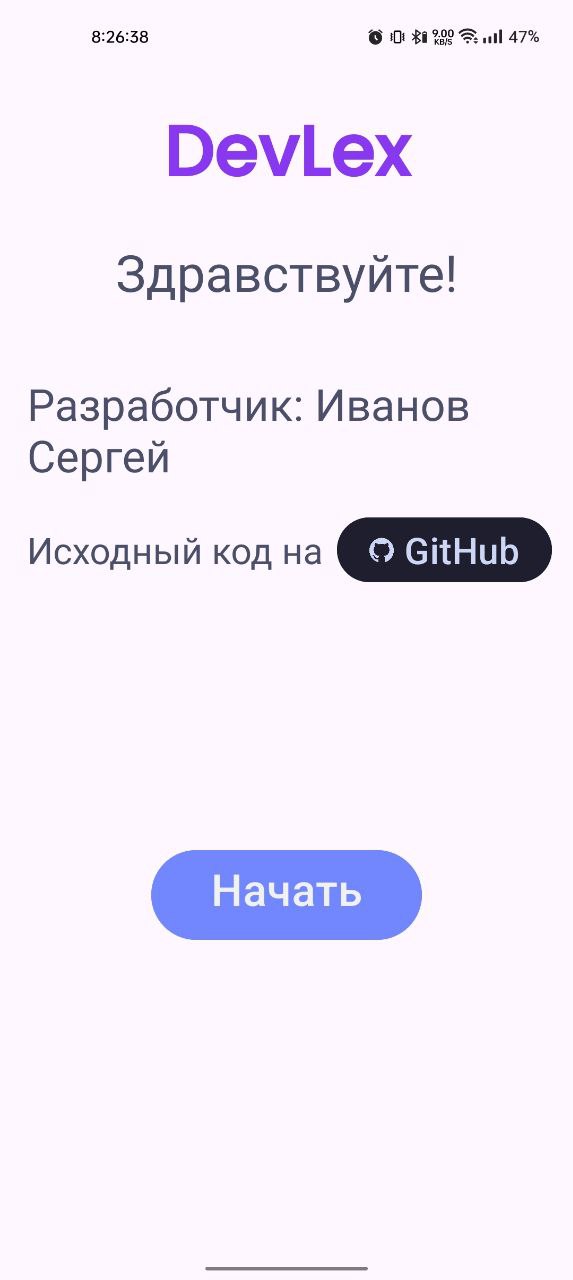


Рисунок 4.1 — Начальная форма

При нажатии кнопки «Начать» открывается основная форма, она представлена на рисунке 4.2.



Рисунок 4.2 — Основная форма

Основная форма представляет графический интерфейс для изучения теоретического материала.

На форме теоретический материал можно отсортировать по английскому и русскому алфавиту. Результаты сортировки по английскому алфавиту в обратном порядке представлены на рисунке 4.3.

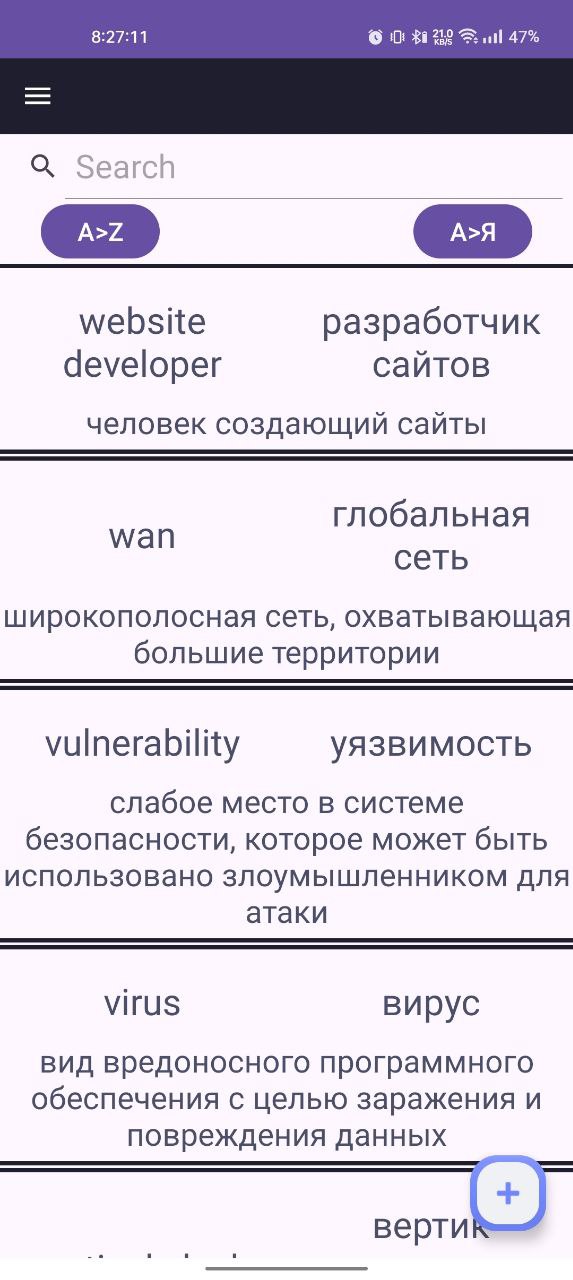


Рисунок 4.3 – Сортировка по английскому алфавиту в обратном порядке

Результаты сортировки по русскому алфавиту представлены на рисунке 4.4.



Рисунок 4.4 – Сортировка по русскому алфавиту

Результаты сортировки по русскому языку в обратном порядке представлены на рисунке 4.5.

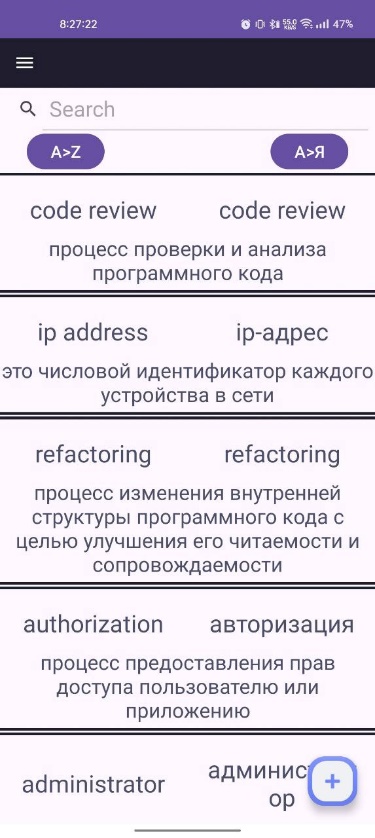


Рисунок 4.5 – Сортировка по русскому алфавиту в обратном порядке

Для поиска по материалу можно воспользоваться поиском, результат работы поиска изображен на рисунке 4.6.

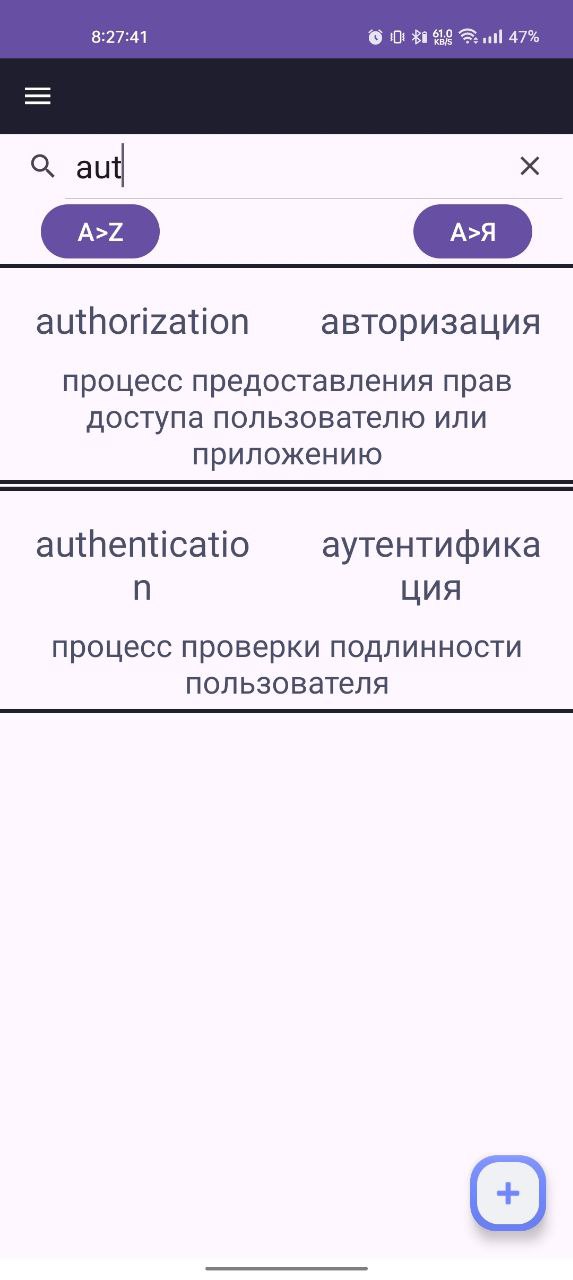


Рисунок 4.6 – Результат работы поиска

Для добавления записей в список можно воспользоваться функцией добавления с помощью кнопки в правом нижнем углу. При нажатии открывается форма для добавления, изображена на рисунке 4.7.

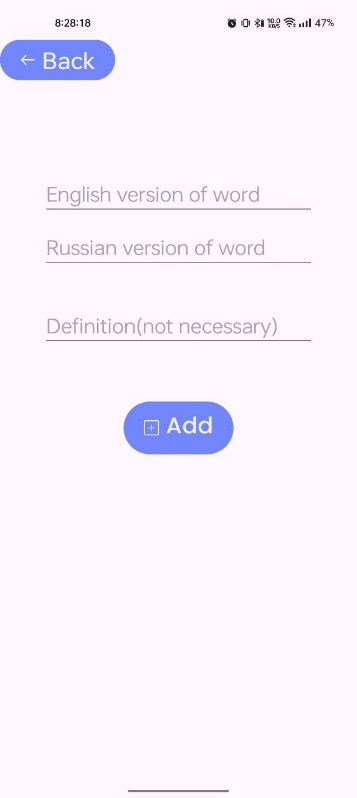


Рисунок 4.7 – Форма добавления новых записей

При неправильном заполнении данных, обязательное поле отсутствует или запись с такими данными уже существует, отображается уведомление, изображено на рисунке 4.8.



Рисунок 4.8 – Уведомление о некорректности введённых данных

Если все данные заполнены верно, то запись добавляется в базу данных, а форма меняется на основную, результат изображен на рисунке 4.9.

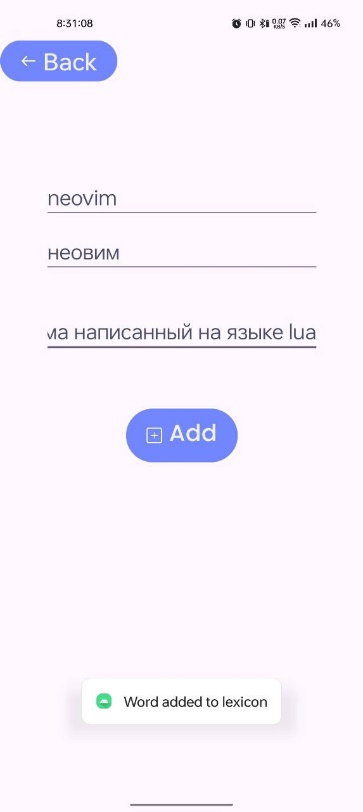


Рисунок 4.9 – Уведомление о добавлении новой записи

Результат добавления записи в словарь изображен на рисунке 4.10.

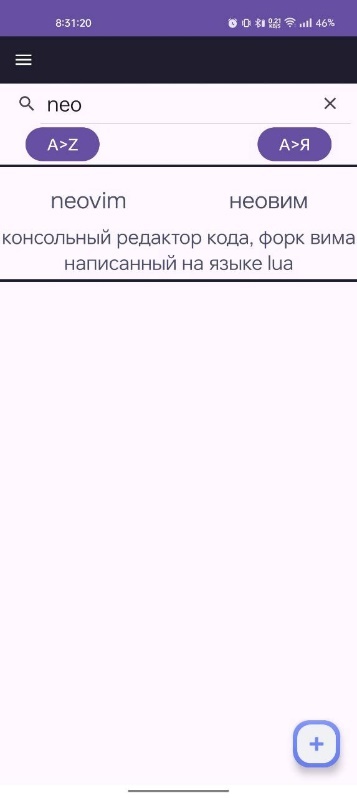


Рисунок 4.10 – Результат добавления записи

Для перемещения по другим формам приложения реализовано выпадающее меню, результат изображен на рисунке 4.11.

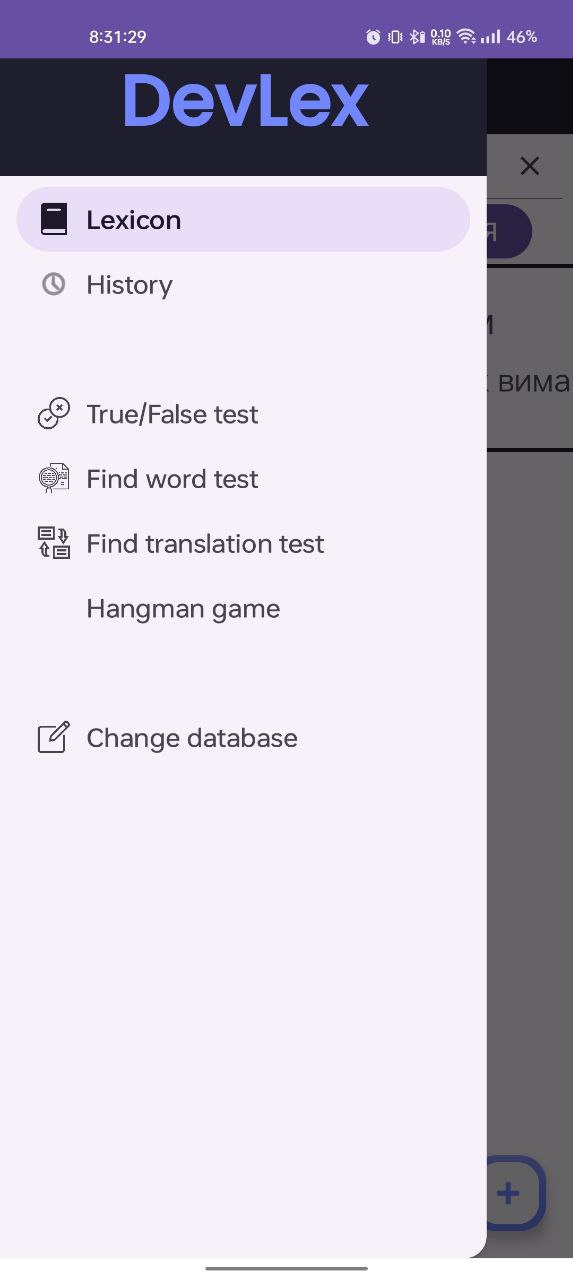


Рисунок 4.11 – Выпадающее меню

Для изменения словаря реализована форма перейти к которой можно с помощью выпадающего меню, форма изображена на рисунке 4.12.

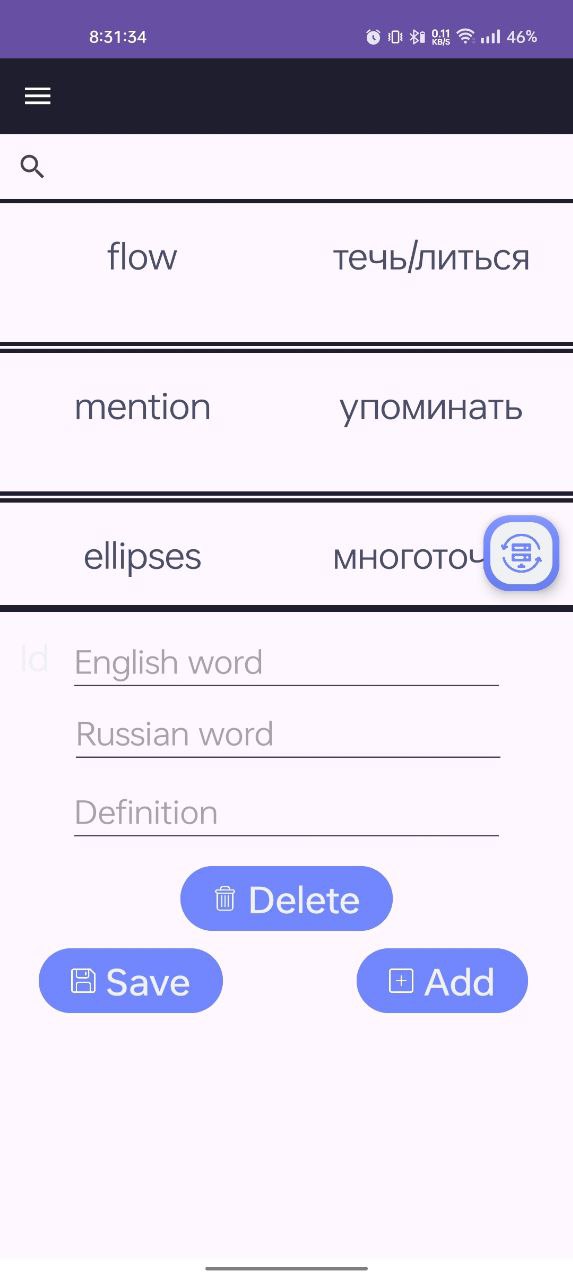


Рисунок 4.12 – Форма изменения словаря

Для изменения записей реализованы функции удаления и сохранения изменений. Для удаления необходимо нажать на запись и нажать кнопку «Удалить», результат изображен на рисунке 4.13.

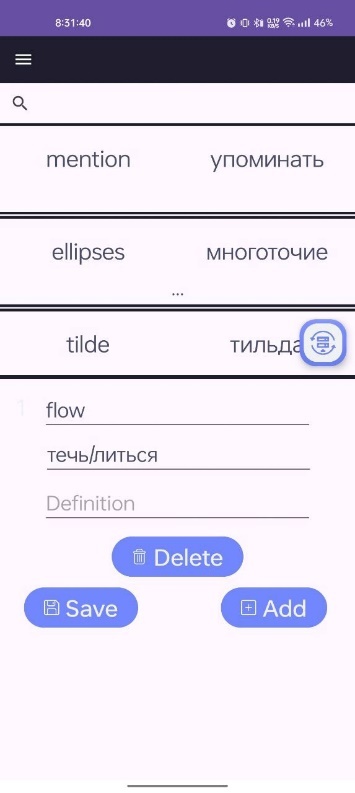


Рисунок 4.13 – Удаление записи «flow»

Для изменения записи необходимо нажать на запись, данные занесутся в поля и их можно будет изменить, для сохранения изменений необходимо нажать «Сохранить», результат изображен на рисунке 4.14.

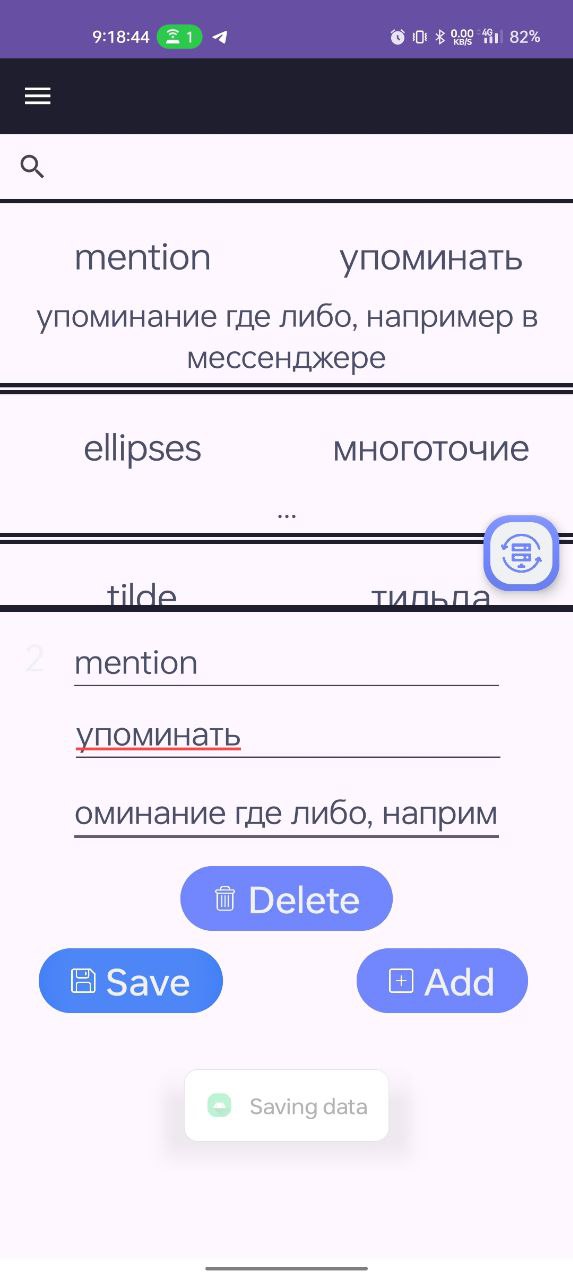


Рисунок 4.14 – Сохранение изменённых данных

Для перехода к тесту необходимо воспользоваться пунктами из выпадающего меню.

Первый тест – «Правда/Ложь», приветственная форма изображена на рисунке 4.15.

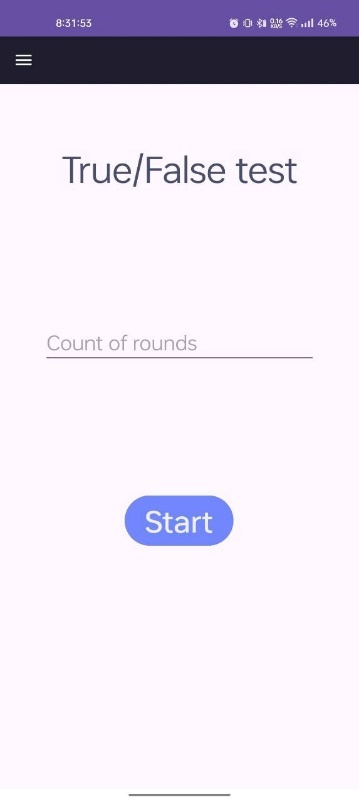


Рисунок 4.15 – Приветственная форма теста

Пользователь может ввести количество вопросов, в случае если количество не указано, то количество вопросов равно количествам записей в базе данных. При нажатии кнопки «Начать» форма меняется на форму теста, форма изображена на рисунке 4.16.

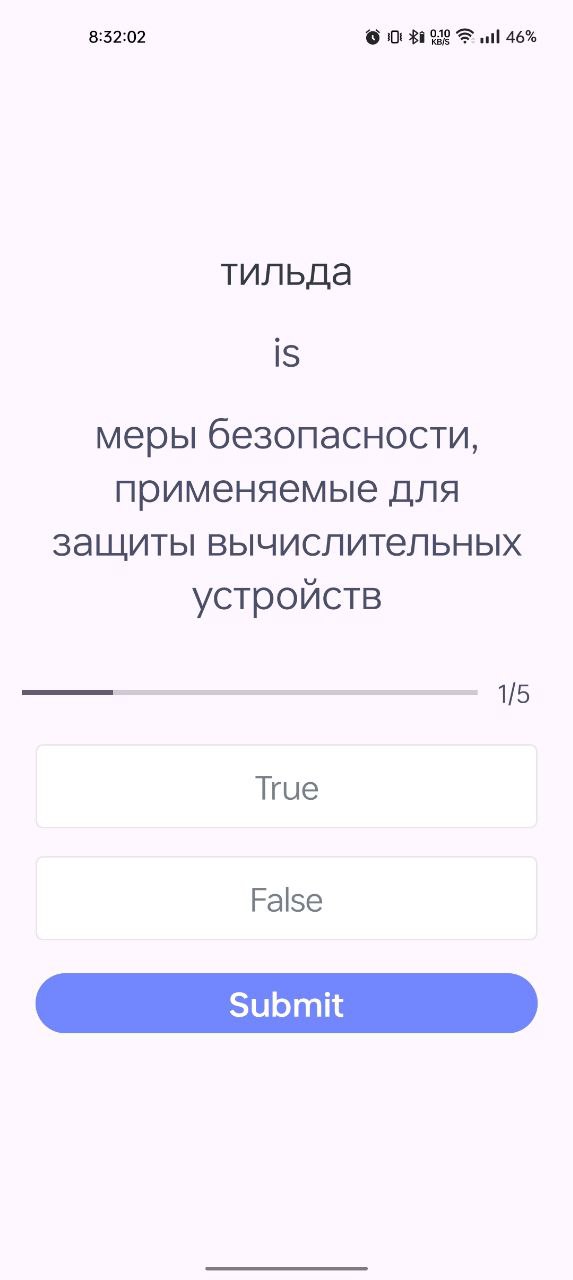


Рисунок 4.16 – Форма теста

Для ответа пользователю нужно выбрать пункт и нажать кнопку «Ответить», верный ответ становиться зеленым, изображено на рисунке 4.17.

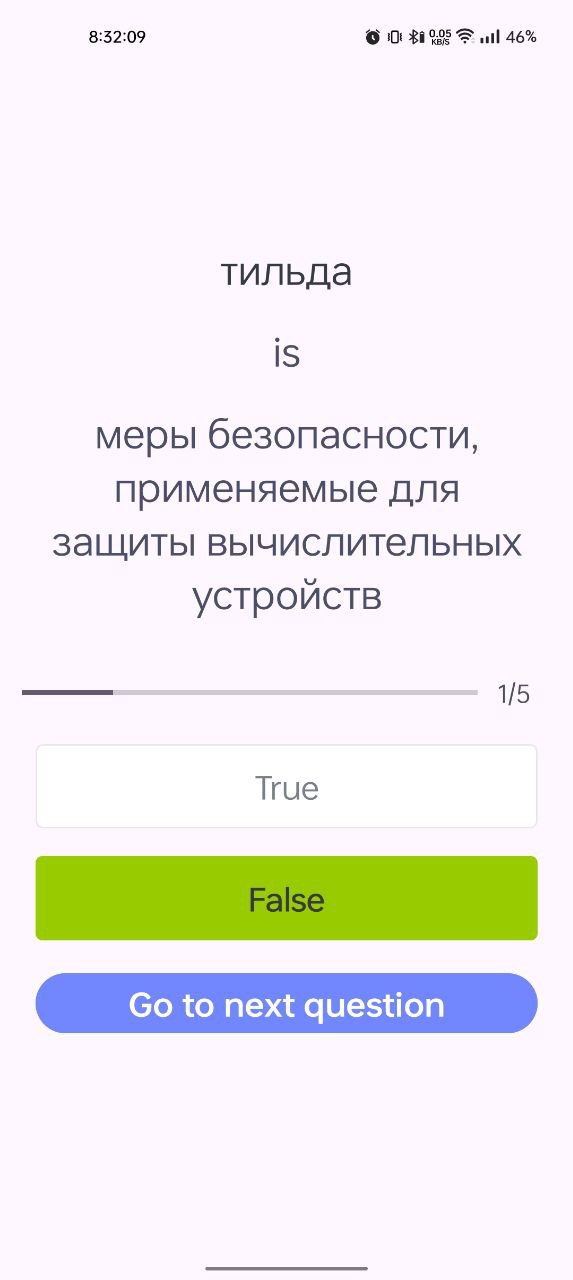


Рисунок 4.17 – Отображение верного ответа

По окончанию теста форма меняется на форму результата и пользователю представляется результат прохождения теста, а также результат заносится в историю прохождения тестов, результат изображен на рисунке 4.18.

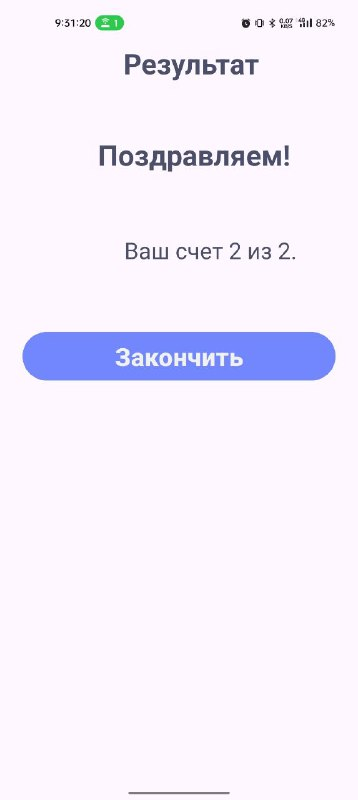


Рисунок 4.18 – Форма результата

Формы второго и третьего теста похожи на первый.

Четвертый тест «Hangman», представляет из себя игру, в которой необходимо угадывать слово по буквам. В игре необходимо ввести букву в поле для ввода и нажать «Ответить», если эта буква присутствует в загаданном слове, изображено на рисунке 4.19, то места пропусков в которых стоит эта буква заменятся ею, в случае если этой буквы нет в слове, то изображение измениться.

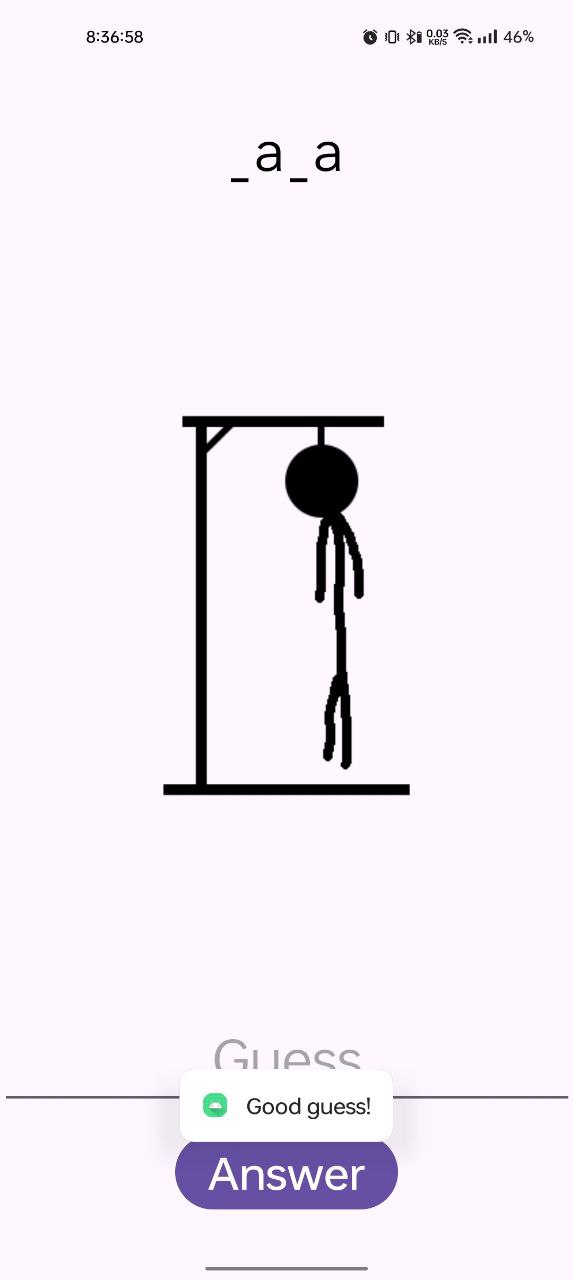


Рисунок 4.19 – Открытие букв в загаданном слове

Всего у пользователя есть 10 попыток, пока изображение не исчезнет. При отгадывании всех букв слова, появиться всплывающее окно с выбором между закончить тест или начать заново. Если был выбран закончить тест, форма измениться на форму с результатом, а данные прохождения теста занесутся в базу данных, результат изображен на рисунке 4.20.



Рисунок 4.20 – Окно выигрыша

Для просмотра истории результатов необходимо выбрать соответствующий пункт в выпадающем меню. Изображение раздела истории изображено на рисунке 4.21, разделы для других трёх тестов идентичны.

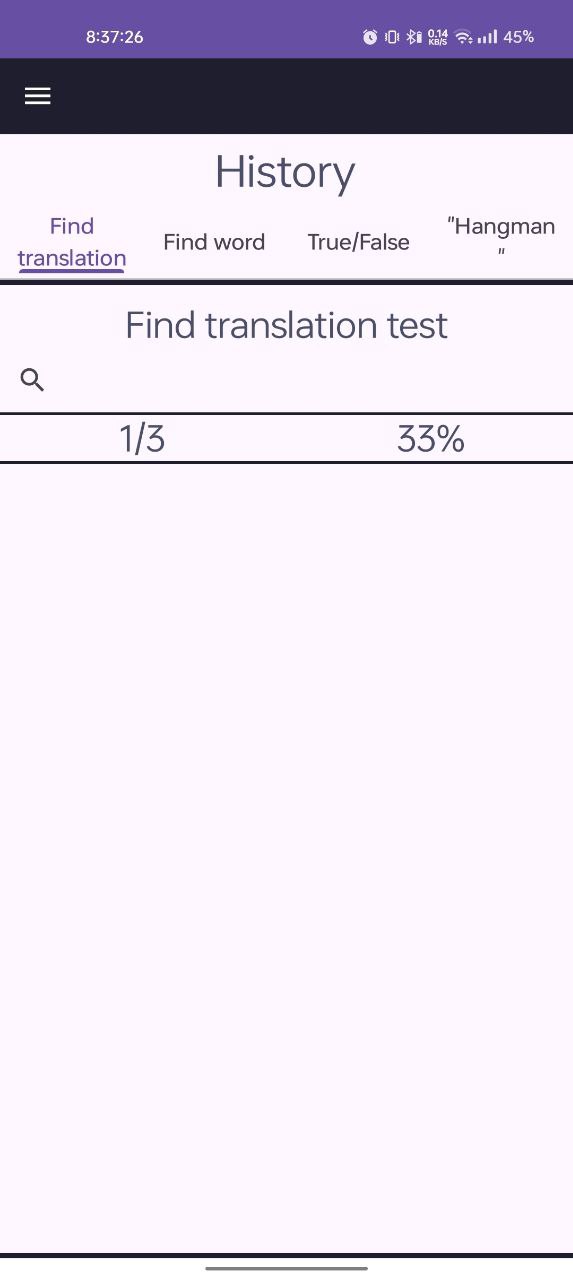


Рисунок 4.21 – История прохождения теста

# Заключение

В результате выполнения курсового проекта было разработано мобильное приложение «Учим английские слова для IT».

При написании курсового проекта был раскрыт теоретический вопрос индивидуального задания по теме «Символьный и строковый типы данных».

Для работы системы с данными была организована работа с базой данных «SQLite». При разработке использовался язык «Kotlin» и среда разработки «Android Studio».

Функциональные возможности программы позволяют изучать теоретический материал, закреплять теоретический материал с помощью тестов, изучать результаты прохождения тестов, изменять записи в таблице базы данных, добавлять записи таблицу базы данных, удалять записи из таблицы базы данных.

Был реализован простой и понятный интерфейс мобильного приложения.

Для предотвращения ошибок со стороны пользователя были организованы проверки на ввод данных и предупреждения при некорректности данных или программных ошибках.

В результате курсового проекта было реализовано приложение, которое упрощает обучение английским словам. В программе были разработаны все поставленные задачи. В приложении реализован удобный интерфейс для изучения теоретического материала, несколько тестов для закрепления изученного материала и история прохождения тестов для отслеживания прогресса изучения.

# Список использованных источников

1. «Developers» Тренировочные курсы [Электронный ресурс]. – «Essentials» – «Training courses»: – <https://developer.android.com/courses>.
2. «Kotlin» Документация [Электронный ресурс]. – «Docs» - «Home»: - https://kotlinlang.org/docs/home.html
3. Пьер-Ив Симон. Волшебство «Kotlin»: Учеб. Пособие / В. М. Стасышин. - Новосибирск: Изд-во ДМК Пресс, 2020 — 535 c.
4. Паул Сандерсон. «SQLite Forensics»: Учеб. Пособие / Изд-во «Amazon», 2018 ­­– 315 с.
5. Филлипс Б., Стюарт К., Марсикано К., Гарднер Б. «Android». Программирование Для Профессионалов / Изд-во «Big Nerd Ranch Guides»: 2019 – 624 с.
6. Брюс Э., Светлана И. «Atomic Kotlin» / Изд-во «Mindview LLC»: 2021 – 636 с.
7. Срибаца Э. «SQLite for Mobile Apps Simplified» / Изд-во «Amazon»: 2014 – 122 c.

# Приложение А

(обязательное)

**Листинг приложения**

Модуль встречной формы:

package github.cirqach.devlex  
  
import android.content.Intent  
import android.net.Uri  
import android.os.Bundle  
import android.widget.Button  
import androidx.activity.enableEdgeToEdge  
import androidx.appcompat.app.AppCompatActivity  
import androidx.core.view.ViewCompat  
import androidx.core.view.WindowInsetsCompat  
import github.cirqach.devlex.app\_pages.RealMainActivity  
import github.cirqach.devlex.database.DevLexDBHelper  
  
  
class MainActivity : AppCompatActivity() {  
  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 enableEdgeToEdge()  
 setContentView(R.layout.activity\_main)  
 ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) **{** v, insets **->** val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())  
 v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)  
 insets  
 **}** val buttonStart: Button? = findViewById(R.id.start\_button)  
 buttonStart!!.setOnClickListener **{** val dbh = DevLexDBHelper(this)  
 dbh.copyDatabase(this)  
  
 val toRealActivityIntent = Intent(this, RealMainActivity::class.java)  
 startActivity(toRealActivityIntent)  
 **}** val buttonGithub: Button? = findViewById(R.id.github\_button)  
 buttonGithub!!.setOnClickListener **{** val url = "https://github.com/Cirqach/DevLex.git"  
 val i = Intent(Intent.ACTION\_VIEW)  
 i.data = Uri.parse(url)  
 startActivity(i)  
 **}** }  
  
}

Модуль основной формы:

package github.cirqach.devlex.app\_pages  
  
import android.os.Bundle  
import android.view.MenuItem  
import androidx.appcompat.app.ActionBarDrawerToggle  
import androidx.appcompat.app.AppCompatActivity  
import androidx.appcompat.widget.Toolbar  
import androidx.core.view.GravityCompat  
import androidx.drawerlayout.widget.DrawerLayout  
import com.google.android.material.navigation.NavigationView  
import github.cirqach.devlex.R  
import github.cirqach.devlex.app\_pages.fragments.ChangeDatabaseFragment  
import github.cirqach.devlex.app\_pages.fragments.StatisticPage.StatisticFragment  
import github.cirqach.devlex.app\_pages.fragments.tests.HangmanGameFragment  
  
class RealMainActivity : AppCompatActivity(), NavigationView.OnNavigationItemSelectedListener {  
 private lateinit var drawerLayout: DrawerLayout  
  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.activity\_real\_main)  
 drawerLayout = findViewById(R.id.drawer\_layout)  
  
 val toolbar = findViewById<Toolbar>(R.id.toolbar)  
 setSupportActionBar(toolbar)  
  
 val action = supportActionBar  
 action!!.title = ""  
  
 val navigationView = findViewById<NavigationView>(R.id.nav\_view)  
 navigationView.setNavigationItemSelectedListener(this)  
  
 val toggle = ActionBarDrawerToggle(  
 this,  
 drawerLayout,  
 toolbar,  
 R.string.open\_nav,  
 R.string.close\_nav  
 )  
  
 drawerLayout.addDrawerListener(toggle)  
 toggle.syncState()  
  
 if (savedInstanceState == null) {  
 supportFragmentManager.beginTransaction()  
 .replace(  
 R.id.fragment\_container,  
 github.cirqach.devlex.app\_pages.fragments.LexiconFragment()  
 ).commit()  
 navigationView.setCheckedItem(R.id.nav\_words)  
 }  
  
  
 }  
  
 override fun onNavigationItemSelected(item: MenuItem): Boolean {  
 when (item.itemId) {  
 R.id.nav\_words -> supportFragmentManager.beginTransaction()  
 .replace(  
 R.id.fragment\_container,  
 github.cirqach.devlex.app\_pages.fragments.LexiconFragment()  
 ).commit()  
  
 R.id.nav\_statistic -> supportFragmentManager.beginTransaction()  
 .replace(  
 R.id.fragment\_container,  
 StatisticFragment()  
 ).commit()  
  
 R.id.nav\_true\_false\_test -> supportFragmentManager.beginTransaction()  
 .replace(  
 R.id.fragment\_container,  
 github.cirqach.devlex.app\_pages.fragments.tests.TrueFalseTestFragment()  
 ).commit()  
  
 R.id.nav\_find\_word\_test -> supportFragmentManager.beginTransaction()  
 .replace(  
 R.id.fragment\_container,  
 github.cirqach.devlex.app\_pages.fragments.tests.FindWordTestFragment()  
 ).commit()  
  
 R.id.nav\_find\_translation\_test -> supportFragmentManager.beginTransaction()  
 .replace(  
 R.id.fragment\_container,  
 github.cirqach.devlex.app\_pages.fragments.tests.FindTranslationTestFragment()  
 ).commit()  
  
 R.id.nav\_edit\_database -> supportFragmentManager.beginTransaction()  
 .replace(  
 R.id.fragment\_container,  
 ChangeDatabaseFragment()  
 ).commit()  
  
 R.id.hangman\_test -> supportFragmentManager.beginTransaction()  
 .replace(  
 R.id.fragment\_container,  
 HangmanGameFragment()  
 ).commit()  
 }  
 drawerLayout.closeDrawer(GravityCompat.START)  
 return true  
 }  
  
 @Deprecated("Deprecated in Java")  
 override fun onBackPressed() {  
 super.onBackPressed()  
 if (drawerLayout.isDrawerOpen(GravityCompat.START)) {  
 drawerLayout.closeDrawer(GravityCompat.START)  
 } else {  
 onBackPressedDispatcher.onBackPressed()  
 }  
 }  
  
}

Модуль изменения словаря:

package github.cirqach.devlex.app\_pages  
  
import android.content.Intent  
import android.database.Cursor  
import android.os.Build  
import android.os.Bundle  
import android.util.Log  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import android.widget.Button  
import android.widget.EditText  
import android.widget.SearchView  
import android.widget.TextView  
import android.widget.Toast  
import androidx.annotation.RequiresApi  
import androidx.fragment.app.Fragment  
import androidx.recyclerview.widget.LinearLayoutManager  
import androidx.recyclerview.widget.RecyclerView  
import com.google.android.material.floatingactionbutton.FloatingActionButton  
import github.cirqach.devlex.R  
import github.cirqach.devlex.database.DataList  
import github.cirqach.devlex.database.DevLexAdapter  
import github.cirqach.devlex.database.DevLexDBHelper  
import github.cirqach.devlex.database.DevLexDatabaseContract  
import java.util.Locale  
  
  
class ChangeDatabaseFragment : Fragment() {  
  
 private lateinit var dbh: DevLexDBHelper  
 private lateinit var newArray: ArrayList<DataList>  
 private lateinit var recyclerView: RecyclerView  
 private lateinit var searchView: SearchView  
 private lateinit var adapter: DevLexAdapter  
 private lateinit var russianEditText: EditText  
 private lateinit var englishEditText: EditText  
 private lateinit var definitionEditText: EditText  
 private lateinit var deleteButton: Button  
 private lateinit var saveButton: Button  
 private lateinit var addButton: Button  
 private lateinit var idTextView: TextView  
  
 override fun onCreateView(  
 inflater: LayoutInflater, container: ViewGroup?,  
 savedInstanceState: Bundle?  
 ): View? {  
 return inflater.inflate(R.layout.fragment\_change\_database, container, false)  
 }  
  
 @RequiresApi(Build.VERSION\_CODES.S)  
 override fun onViewCreated(view: View, savedInstanceState: Bundle?) {  
 super.onViewCreated(view, savedInstanceState)  
  
 val refreshDatabaseButton: FloatingActionButton =  
 view.findViewById(R.id.crud\_refresh\_database\_button)  
 refreshDatabaseButton.setOnClickListener **{** displayWord()  
 **}** recyclerView = view.findViewById(R.id.crud\_recycler\_view)  
  
 russianEditText = view.findViewById(R.id.crud\_russian\_name\_edit\_text\_view)  
 englishEditText = view.findViewById(R.id.crud\_english\_name\_edit\_text\_view)  
 definitionEditText = view.findViewById(R.id.crud\_definition\_edit\_text\_view)  
 idTextView = view.findViewById(R.id.crud\_id\_text\_view)  
  
 if (view.context != null) {  
 dbh = DevLexDBHelper(view.context)  
 }  
 recyclerView.layoutManager = LinearLayoutManager(view.context)  
 recyclerView.setHasFixedSize(true)  
 displayWord()  
  
  
  
 searchView = view.findViewById(R.id.crud\_search\_view)  
 searchView.setOnQueryTextListener(object : SearchView.OnQueryTextListener {  
 override fun onQueryTextSubmit(query: String?): Boolean {  
 return false  
 }  
  
 override fun onQueryTextChange(newText: String?): Boolean {  
 filterList(newText)  
 return true  
 }  
  
 })  
  
 deleteButton = view.findViewById(R.id.crud\_delete\_button)  
 saveButton = view.findViewById(R.id.crud\_save\_button)  
 addButton = view.findViewById(R.id.crud\_add\_button)  
 deleteButton.setOnClickListener **{** dbh.deleteData(  
 DevLexDatabaseContract.LexiconEntry.TABLE\_NAME,  
 idTextView.text.toString()  
 )  
 displayWord()  
 **}** addButton.setOnClickListener **{** val intent = Intent(requireContext(), AddWordActivity::class.java)  
 startActivity(intent)  
 **}** saveButton.setOnClickListener **{** Toast.makeText(view.context, "Saving data", Toast.LENGTH\_SHORT).show()  
 Log.d(  
 "save\_button\_click", "Data to save: " +  
 "english = ${englishEditText.text} , " +  
 "russian = ${russianEditText.text}, " +  
 "definition = ${definitionEditText.text} ," +  
 "ID = ${idTextView.text}"  
 )  
 dbh.saveDataToLexicon(  
 englishEditText.text.toString(),  
 russianEditText.text.toString(),  
 definitionEditText.text.toString(),  
 idTextView.text.toString()  
 )  
  
 displayWord()  
 **}** }  
  
  
 private fun filterList(query: String?) {  
 if (query != null) {  
 val filteredList = ArrayList<DataList>()  
 for (i in newArray) {  
 if (i.englishName.lowercase(Locale.ROOT).contains(query) ||  
 i.russianName.lowercase(Locale.ROOT).contains(query) ||  
 i.definition.lowercase(Locale.ROOT).contains(query)  
 ) {  
 filteredList.add(i)  
 }  
 }  
 if (filteredList.isEmpty()) {  
 Toast.makeText(requireView().context, "No data found", Toast.LENGTH\_SHORT).show()  
 } else {  
 adapter.setFilteredList(filteredList)  
  
 }  
 }  
 }  
  
 private fun displayWord() {  
 val newCursor: Cursor? = dbh.readAll(DevLexDatabaseContract.LexiconEntry.TABLE\_NAME)  
 newArray = ArrayList()  
 while (newCursor!!.moveToNext()) {  
 val uEnglishName = newCursor.getString(1)  
 val uRussianName = newCursor.getString(2)  
 val uDefinition = newCursor.getString(3)  
 val uid = newCursor.getString(0)  
 newArray.add(DataList(uEnglishName, uRussianName, uDefinition, uid))  
 }  
 recyclerView.adapter = DevLexAdapter(newArray)  
 adapter = DevLexAdapter(newArray)  
 recyclerView.adapter = adapter  
  
  
 adapter.onItemClick = **{** position **->** russianEditText.setText(position.russianName)  
 englishEditText.setText(position.englishName)  
 definitionEditText.setText(position.definition)  
 idTextView.text = position.id  
 **}** }  
  
}

Модуль добавления записей:

package github.cirqach.devlex.app\_pages  
  
import android.app.Activity  
import android.content.Intent  
import android.os.Bundle  
import android.text.TextUtils  
import android.widget.Button  
import android.widget.EditText  
import android.widget.Toast  
import androidx.activity.enableEdgeToEdge  
import androidx.appcompat.app.AppCompatActivity  
import androidx.core.view.ViewCompat  
import androidx.core.view.WindowInsetsCompat  
import github.cirqach.devlex.R  
import github.cirqach.devlex.database.DevLexDBHelper  
  
class AddWordActivity : AppCompatActivity() {  
  
 private lateinit var englishName: EditText  
 private lateinit var russianName: EditText  
 private lateinit var definition: EditText  
 private lateinit var addButton: Button  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 enableEdgeToEdge()  
 setContentView(R.layout.activity\_add\_word)  
 ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) **{** v, insets **->** val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())  
 v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)  
 insets  
 **}** val goBackButton = findViewById<Button>(R.id.go\_back\_button)  
 goBackButton.setOnClickListener **{** onBackPressed()  
 **}** englishName = findViewById(R.id.english\_name\_edit\_text)  
 russianName = findViewById(R.id.russian\_name\_edit\_text)  
 definition = findViewById(R.id.definition\_edit\_text)  
 addButton = findViewById(R.id.add\_button)  
  
 val db = DevLexDBHelper(this)  
  
 addButton.setOnClickListener **{** val englishNameText = englishName.text.toString()  
 val russiaNameText = russianName.text.toString()  
 val definitionText = definition.text.toString()  
 val saveData = db.addDataToLexicon(englishNameText, russiaNameText, definitionText)  
  
 if (TextUtils.isEmpty(englishNameText) || TextUtils.isEmpty(russiaNameText)) {  
 Toast.makeText(this,  
 "You can't leave empty field",  
 Toast.LENGTH\_SHORT).show()  
 } else if (saveData) {  
 Toast.makeText(this,  
 "Word added to lexicon",  
 Toast.LENGTH\_SHORT).show()  
 val intent = Intent()  
 setResult(Activity.RESULT\_OK, intent)  
 finish()  
 } else {  
 Toast.makeText(this,  
 "This words already exist",  
 Toast.LENGTH\_SHORT).show()  
 }  
 **}** }  
  
}

Модуль результатов:

package github.cirqach.devlex.app\_pages  
  
import android.content.Intent  
import android.os.Bundle  
import androidx.activity.enableEdgeToEdge  
import androidx.appcompat.app.AppCompatActivity  
import androidx.core.view.ViewCompat  
import androidx.core.view.WindowInsetsCompat  
import github.cirqach.devlex.R  
import github.cirqach.devlex.databinding.ActivityResultBinding  
  
class ActivityResult : AppCompatActivity() {  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 *enableEdgeToEdge*()  
 setContentView(R.layout.*activity\_result*)  
 ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.*main*)) **{** v, insets **->** val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())  
 v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)  
 insets  
 **}** val totalQuestions = *intent*.getIntExtra("TOTAL\_QUESTIONS", 0)  
 val correctAnswers = *intent*.getIntExtra("CORRECT\_ANSWERS", 0)  
  
 val binding = ActivityResultBinding.inflate(*layoutInflater*)  
 setContentView(binding.*root*)  
  
 binding.tvScore.*text* =  
 getString(  
 R.string.*your\_score\_is\_out\_of*,  
 correctAnswers.toString(),  
 totalQuestions.toString()  
 )  
  
 binding.btnFinish.setOnClickListener **{** startActivity(Intent(this@ActivityResult, RealMainActivity::class.*java*))  
 **}** }  
}

Модуль «Правда/Ложь»:

package github.cirqach.devlex.app\_pages.tests  
  
import android.content.Intent  
import android.graphics.Color  
import android.graphics.Typeface  
import android.os.Bundle  
import android.util.Log  
import android.view.View  
import android.widget.TextView  
import androidx.activity.enableEdgeToEdge  
import androidx.appcompat.app.AppCompatActivity  
import androidx.core.content.ContextCompat  
import androidx.core.view.ViewCompat  
import androidx.core.view.WindowInsetsCompat  
import github.cirqach.devlex.R  
import github.cirqach.devlex.app\_pages.ActivityResult  
import github.cirqach.devlex.database.DevLexDBHelper  
import github.cirqach.devlex.database.DevLexDatabaseContract  
import github.cirqach.devlex.database.TrueFalseTest.QuestionBaseHelper  
import github.cirqach.devlex.databinding.ActivityTrueFalseTestBinding  
  
class TrueFalseTestActivity : AppCompatActivity(), View.OnClickListener {  
  
 private var mCurrentPosition: Int = 1 // Default and the first question position  
 private var mQuestionsList: MutableList<QuestionBaseHelper.Question?> = mutableListOf()  
 private var mSelectedOptionPosition: Int = 0  
 private var mCorrectAnswers: Int = 0  
  
  
 private lateinit var binding: ActivityTrueFalseTestBinding  
 private var dbh: DevLexDBHelper? = null  
 private var questionHelper: QuestionBaseHelper = QuestionBaseHelper()  
 private var questionsList: MutableList<QuestionBaseHelper.Question?> =  
 mutableListOf() // Set to hold all questions  
 private var questionsCount: Int = 0  
 private val tag: String = "true false test"  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 enableEdgeToEdge()  
 setContentView(R.layout.activity\_true\_false\_test)  
 ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) **{** v, insets **->** val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())  
 v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)  
 insets  
 **}** dbh = DevLexDBHelper(this)  
 questionsCount = intent.getIntExtra(  
 "count\_of\_test",  
 DevLexDBHelper.getTableRowCount(  
 DevLexDatabaseContract.LexiconEntry.TABLE\_NAME,  
 dbh!!  
 )  
 )  
  
 mQuestionsList = questionHelper.getQuestionsList(dbh, questionsCount) Log.d(tag, "onCreate: question base is $questionsList")  
 binding = ActivityTrueFalseTestBinding.inflate(layoutInflater)  
 setContentView(binding.root)  
  
 Log.d(tag, "onCreate: ${questionHelper.getQuestionsList(dbh, questionsCount)}")  
  
 setQuestion()  
 binding.progressBar.max = questionsCount  
 binding.tvOptionTrue.setOnClickListener(this)  
 binding.tvOptionFalse.setOnClickListener(this)  
 binding.btnSubmit.setOnClickListener(this)  
 }  
  
 private fun setQuestion() {  
 val question = mQuestionsList[mCurrentPosition - 1]  
  
 defaultOptionsView()  
  
 if (mCurrentPosition == mQuestionsList.size) {  
 binding.btnSubmit.text = getString(R.string.finish)  
 } else {  
 binding.btnSubmit.text = getString(R.string.submit)  
 }  
  
 binding.progressBar.progress = mCurrentPosition  
 binding.tvProgress.text = getString(  
 R.string.putting\_result\_to\_database,  
 mCurrentPosition.toString(),  
 mQuestionsList.size.toString()  
 )  
  
  
 if (question != null) {  
 binding.tvQuestion.text = question.russianOrEnglishName  
 binding.tvDefinition.text = question.definition  
 binding.tvOptionFalse.text = getString(R.string.false\_answer)  
 binding.tvOptionTrue.text = getString(R.string.true\_answer)  
 }  
  
 binding.tvOptionTrue.isEnabled = true  
 binding.tvOptionFalse.isEnabled = true  
 }  
  
 private fun defaultOptionsView() {  
  
 val options = ArrayList<TextView>()  
 options.add(0, binding.tvOptionTrue)  
 options.add(1, binding.tvOptionFalse)  
  
 for (option in options) {  
 option.setTextColor(Color.parseColor("#7A8089"))  
 option.typeface = Typeface.DEFAULT  
 option.background = ContextCompat.getDrawable(  
 this@TrueFalseTestActivity,  
 R.drawable.default\_option\_border\_bg  
 )  
 }  
 }  
  
 override fun onClick(v: View?) {  
  
 when (v?.id) {  
  
 R.id.tv\_option\_true -> {  
  
 selectedOptionView(binding.tvOptionTrue, true)  
 }  
  
 R.id.tv\_option\_false -> {  
  
 selectedOptionView(binding.tvOptionFalse, false)  
 }  
  
 R.id.btn\_submit -> {  
  
 if (mSelectedOptionPosition == 0) {  
  
 mCurrentPosition++  
  
 when {  
  
 mCurrentPosition <= mQuestionsList.size -> {  
  
 setQuestion()  
 }  
  
 else -> {  
  
 val intent =  
 Intent(this@TrueFalseTestActivity, ActivityResult::class.java)  
 intent.putExtra("CORRECT\_ANSWERS", mCorrectAnswers)  
 intent.putExtra("TOTAL\_QUESTIONS", mQuestionsList.size)  
 if (dbh != null) {  
 if (dbh?.saveDataToTest(  
 DevLexDatabaseContract.Tables.TRUE\_FALSE\_TABLE,  
 mCorrectAnswers,  
 (mCorrectAnswers.toDouble() / mQuestionsList.size \* 100).toInt(),  
 questionsCount  
 ) == true  
 ) {  
 Log.d(  
 tag,  
 "onClick: data saved $mCorrectAnswers + ${  
 if (mQuestionsList.size != 0 && mCorrectAnswers != 0) {  
 (mQuestionsList.size / mCorrectAnswers \* 100)  
 } else {  
 "0/${mQuestionsList.size}"  
 }  
 }"  
 )  
 } else {  
 Log.d(tag, "onClick: data not saved")  
 }  
 } else {  
 Log.d(tag, "onClick: dbh is null")  
 }  
  
  
 startActivity(intent)  
 finish()  
 // END  
 }  
 }  
 } else {  
 val question = mQuestionsList[mCurrentPosition - 1]  
  
 if (!question!!.answer) {  
 answerView(mSelectedOptionPosition, R.drawable.wrong\_option\_border\_bg)  
 } else {  
 mCorrectAnswers++  
 }  
  
 answerView(  
 if (question.answer) {  
 1  
 } else {  
 2  
 }, R.drawable.correct\_option\_border\_bg  
 )  
  
 if (mCurrentPosition == mQuestionsList.size) {  
 binding.btnSubmit.text = getString(R.string.finish)  
 } else {  
 binding.btnSubmit.text = getString(R.string.go\_to\_next\_question)  
 }  
  
 mSelectedOptionPosition = 0  
  
 binding.tvOptionTrue.isEnabled  
 binding.tvOptionFalse.isEnabled = false  
 }  
 }  
 }  
 }  
  
 private fun answerView(answer: Int, drawableView: Int) {  
  
 when (answer) {  
  
 1 -> {  
 binding.tvOptionTrue.background = ContextCompat.getDrawable(  
 this@TrueFalseTestActivity,  
 drawableView  
 )  
 }  
  
 2 -> {  
 binding.tvOptionFalse.background = ContextCompat.getDrawable(  
 this@TrueFalseTestActivity,  
 drawableView  
 )  
 }  
 }  
 }  
  
 private fun selectedOptionView(tv: TextView, selectedOption: Boolean) {  
  
 defaultOptionsView()  
  
 mSelectedOptionPosition = if (selectedOption) 1 else 2  
  
 tv.setTextColor(  
 Color.parseColor("#363A43")  
 )  
 tv.setTypeface(tv.typeface, Typeface.BOLD)  
 tv.background = ContextCompat.getDrawable(  
 this@TrueFalseTestActivity,  
 R.drawable.selected\_option\_border\_bg  
 )  
 }  
}

Модуль «Hangman»:

package github.cirqach.devlex.app\_pages.tests  
  
import android.os.Bundle  
import android.view.View  
import android.widget.Button  
import android.widget.EditText  
import android.widget.ImageView  
import android.widget.TextView  
import android.widget.Toast  
import androidx.activity.enableEdgeToEdge  
import androidx.appcompat.app.AlertDialog  
import androidx.appcompat.app.AppCompatActivity  
import androidx.core.view.ViewCompat  
import androidx.core.view.WindowInsetsCompat  
import github.cirqach.devlex.R  
import github.cirqach.devlex.database.DevLexDBHelper  
import github.cirqach.devlex.database.HangmanGame.HangmanHelper  
import java.util.Locale  
import java.util.Random  
  
class HangmanGame : AppCompatActivity() {  
 private var kill = 0  
 private var secretWord = ""  
 private var secretDisplay = ""  
 private val correctGuesses = mutableListOf<String>()  
 private lateinit var toBeGuessed: TextView  
 private lateinit var hangmanDrawing: ImageView  
 private lateinit var playerGuess: EditText  
 private lateinit var tryButton: Button  
 private lateinit var copySecretWord: String  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 enableEdgeToEdge()  
 setContentView(R.layout.activity\_hangman\_game)  
 ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) **{** v, insets **->** val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())  
 v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)  
 insets  
 **}** val extras = intent.extras  
 if (extras != null) {  
 secretWord = extras.getString("secretWord").toString()  
 }  
 copySecretWord = secretWord  
 toBeGuessed = findViewById(R.id.toBeGuessed)  
 hangmanDrawing = findViewById(R.id.hangmanDrawing)  
 tryButton = findViewById(R.id.tryButton)  
 playerGuess = findViewById(R.id.playerGuess)  
 hangmanDrawing.setImageResource(R.drawable.hangman10)  
 prepGame()  
 }  
  
 fun guessTry(click: View) {  
 val tryButton = findViewById<Button>(R.id.tryButton)  
 if (click === tryButton) {  
 val pGuess = playerGuess.text.toString().lowercase(Locale.ROOT)  
 playerGuess.text = null  
  
 if (pGuess.length == 1) {  
 if (pGuess in secretWord.lowercase(Locale.ROOT)) {  
  
 correctGuesses.add(pGuess)  
 refactorSecret()  
  
 Toast.makeText(applicationContext, getString(R.string.good\_gues), Toast.LENGTH\_SHORT).show()  
 checkWin()  
 return  
 }  
 }  
  
 if (pGuess.length > 1) {  
 if (pGuess.equals(secretWord, ignoreCase = true)) {  
 winDialogPopUp(true)  
 return  
 }  
 }  
  
 kill++  
 when (kill) {  
 1 -> hangmanDrawing.setImageResource(R.drawable.hangman9)  
 2 -> hangmanDrawing.setImageResource(R.drawable.hangman8)  
 3 -> hangmanDrawing.setImageResource(R.drawable.hangman7)  
 4 -> hangmanDrawing.setImageResource(R.drawable.hangman6)  
 5 -> hangmanDrawing.setImageResource(R.drawable.hangman5)  
 6 -> hangmanDrawing.setImageResource(R.drawable.hangman4)  
 7 -> hangmanDrawing.setImageResource(R.drawable.hangman3)  
 8 -> hangmanDrawing.setImageResource(R.drawable.hangman2)  
 9 -> hangmanDrawing.setImageResource(R.drawable.hangman1)  
 10 -> {  
 hangmanDrawing.setImageResource(R.drawable.hangman0)  
 winDialogPopUp(false)  
 val dbh = DevLexDBHelper(this)  
 dbh.saveHangmanResult("lose", copySecretWord)  
 }  
 }  
 }  
 }  
  
 private fun refactorSecret() {  
 secretDisplay = ""  
 secretWord.forEach **{** s **->** secretDisplay += (checkIfGuessed(s.toString()))  
 **}** toBeGuessed.text = secretDisplay  
 }  
  
 private fun checkIfGuessed(s: String): String {  
 return when (correctGuesses.contains(s.lowercase(Locale.ROOT))) {  
 true -> s  
 false -> "\_"  
 }  
 }  
  
 private fun checkWin() {  
 var everythingGuessed = true  
 secretWord.lowercase(Locale.ROOT).forEach **{** c **->** if (!correctGuesses.contains(c.toString()))  
 everythingGuessed = false  
 **}** if (everythingGuessed) {  
 winDialogPopUp(true)  
 val dbh = DevLexDBHelper(this)  
 dbh.saveHangmanResult("win", copySecretWord)  
 }  
 }  
  
 private fun winDialogPopUp(won: Boolean) {  
 val builder = AlertDialog.Builder(this@HangmanGame)  
 if (won) {  
 builder.setTitle(getString(R.string.congratulations))  
 } else {  
 builder.setTitle(getString(R.string.boo\_you\_hanged\_a\_man))  
 }  
 builder.setMessage(getString(R.string.do\_you\_want\_to\_play\_again))  
  
 builder.setPositiveButton(getString(R.string.let\_s\_go)) **{** \_, \_ **->** val dbHelper = DevLexDBHelper(this)  
 val db = HangmanHelper()  
 secretWord = db.getQuestionsList(dbHelper, 10)[Random().nextInt(10) + 1].toString()  
 prepGame()  
  
 Toast.makeText(applicationContext,  
 getString(R.string.new\_game\_started), Toast.LENGTH\_SHORT).show()  
  
 **}** builder.setNegativeButton(getString(R.string.no)) **{** \_, \_ **->** finish()  
 **}** val dialog: AlertDialog = builder.create()  
 dialog.show()  
  
  
 }  
  
 private fun prepGame() {  
 hangmanDrawing.setImageResource(R.drawable.hangman10)  
 kill = 0  
 secretDisplay = ""  
 correctGuesses.clear()  
  
 repeat(secretWord.length) **{** secretDisplay += "\_"  
 **}** toBeGuessed.text = secretDisplay  
 val dbHelper = DevLexDBHelper(this)  
 val db = HangmanHelper()  
 secretWord = db.getQuestionsList(dbHelper, 10)[Random().nextInt(10)].toString()  
 copySecretWord = secretWord  
 }  
}

Модуль статистики:

package github.cirqach.devlex.app\_pages.fragments  
  
import android.os.Bundle  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import androidx.fragment.app.Fragment  
import github.cirqach.devlex.R  
  
  
class StatisticFragment : Fragment() {  
  
 override fun onCreateView(  
 inflater: LayoutInflater, container: ViewGroup?,  
 savedInstanceState: Bundle?  
 ): View? {  
 // Inflate the layout for this fragment  
 return inflater.inflate(R.layout.*fragment\_statistic*, container, false)  
 }  
  
  
}

Модуль словаря:

package github.cirqach.devlex.app\_pages.fragments  
  
import android.app.Activity  
import android.content.Intent  
import android.database.Cursor  
import android.os.Bundle  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import android.widget.Button  
import android.widget.SearchView  
import android.widget.Toast  
import androidx.fragment.app.Fragment  
import androidx.recyclerview.widget.LinearLayoutManager  
import androidx.recyclerview.widget.RecyclerView  
import com.google.android.material.floatingactionbutton.FloatingActionButton  
import github.cirqach.devlex.R  
import github.cirqach.devlex.app\_pages.AddWordActivity  
import github.cirqach.devlex.database.DataList  
import github.cirqach.devlex.database.DevLexAdapter  
import github.cirqach.devlex.database.DevLexDBHelper  
import github.cirqach.devlex.database.DevLexDatabaseContract  
import java.util.Locale  
  
class LexiconFragment : Fragment() {  
  
  
 private lateinit var dbh: DevLexDBHelper  
 private lateinit var newArry: ArrayList<DataList>  
 private lateinit var recyclerView: RecyclerView  
 private lateinit var searchView: SearchView  
 private lateinit var adapter: DevLexAdapter  
  
 @Deprecated("Deprecated in Java")  
 override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {  
 super.onActivityResult(requestCode, resultCode, data)  
  
 if (requestCode == ADD\_WORD\_REQUEST\_CODE && resultCode == Activity.RESULT\_OK) {  
 // Word added successfully, update the list  
 fillArray()  
 adapter.notifyDataSetChanged()  
 }  
 }  
  
 override fun onCreateView(  
 inflater: LayoutInflater, container: ViewGroup?,  
 savedInstanceState: Bundle?  
 ): View? {  
 // Inflate the layout for this fragment  
 return inflater.inflate(R.layout.fragment\_lexicon, container, false)  
 }  
  
 companion object {  
 const val ADD\_WORD\_REQUEST\_CODE = 1  
 }  
  
 override fun onViewCreated(view: View, savedInstanceState: Bundle?) {  
 super.onViewCreated(view, savedInstanceState)  
 val floatingActionButton: FloatingActionButton =  
 view.findViewById(R.id.floatingActionButton)  
 floatingActionButton.setOnClickListener **{** val intent = Intent(  
 requireContext(),  
 AddWordActivity::class.java  
 )  
 startActivityForResult(intent, ADD\_WORD\_REQUEST\_CODE)  
 **}** recyclerView = view.findViewById(R.id.lexiconRecyclerView)  
 dbh = DevLexDBHelper(view.context)  
 recyclerView.layoutManager = LinearLayoutManager(view.context)  
 recyclerView.setHasFixedSize(true)  
  
 fillArray()  
 adapter = DevLexAdapter(newArry)  
 recyclerView.adapter = adapter  
 sortDisplayWordAscendingEnglish()  
 displayList()  
  
 searchView = view.findViewById(R.id.searchView)!!  
 searchView.setOnQueryTextListener(object : SearchView.OnQueryTextListener {  
 override fun onQueryTextSubmit(query: String?): Boolean {  
 return false  
 }  
  
 override fun onQueryTextChange(newText: String?): Boolean {  
 filterList(newText)  
 return true  
 }  
  
 })  
 val sortButton = view.findViewById<Button>(R.id.sortButtonEnglish)  
 var isReverseSort = false  
 sortButton.setOnClickListener **{** if (isReverseSort) {  
 sortDisplayWordAscendingEnglish()  
 displayList()  
 } else {  
 sortDisplayWordDescendingEnglish()  
 displayList()  
 }  
 isReverseSort = !isReverseSort  
 **}** var isReverseSortRussian = false  
 val sortButtonRussian = view.findViewById<Button>(R.id.sortButtonRussian)  
 sortButtonRussian.setOnClickListener **{** if (isReverseSortRussian) {  
 sortDisplayWordAscendingRussian()  
 displayList()  
 } else {  
 sortDisplayWordDescendingRussian()  
 displayList()  
 }  
 isReverseSortRussian = !isReverseSortRussian  
 **}** }  
  
 private fun fillArray() {  
 val newCursor: Cursor? = dbh.readAll(DevLexDatabaseContract.LexiconEntry.TABLE\_NAME)  
 newArry = ArrayList()  
 while (newCursor!!.moveToNext()) {  
 val uEnglishName = newCursor.getString(1)  
 val uRussianName = newCursor.getString(2)  
 val uDefinition = newCursor.getString(3)  
 val uid = newCursor.getString(0)  
 newArry.add(  
 DataList(  
 uEnglishName.lowercase(Locale.ROOT),  
 uRussianName.lowercase(Locale.ROOT),  
 uDefinition,  
 uid  
 )  
 )  
 }  
 newCursor.close()  
  
 }  
  
 private fun filterList(query: String?) {  
 if (query.isNullOrEmpty()) {  
 adapter.setFilteredList(newArry)  
 return  
 }  
  
 val filteredList = ArrayList<DataList>()  
 for (i in newArry) {  
 if (i.englishName.lowercase(Locale.ROOT).contains(query) ||  
 i.russianName.lowercase(Locale.ROOT).contains(query) ||  
 i.definition.lowercase(Locale.ROOT).contains(query)  
 ) {  
 filteredList.add(i)  
 }  
 }  
  
 if (filteredList.isEmpty()) {  
 Toast.makeText(requireView().context, "No data found", Toast.LENGTH\_SHORT).show()  
 }  
  
 adapter.setFilteredList(filteredList)  
 }  
  
 private fun displayList() {  
 adapter.setFilteredList(newArry)  
 }  
  
 private fun sortDisplayWordAscendingEnglish() {  
 newArry.sortWith(compareBy **{** it.englishName.lowercase() **}**)  
 }  
  
 private fun sortDisplayWordDescendingEnglish() {  
 newArry.sortWith(compareByDescending **{** it.englishName.lowercase() **}**)  
 }  
  
 private fun sortDisplayWordAscendingRussian() {  
 newArry.sortWith(compareBy **{** it.russianName.lowercase() **}**)  
 }  
  
 private fun sortDisplayWordDescendingRussian() {  
 newArry.sortWith(compareByDescending **{** it.russianName.lowercase() **}**)  
 }  
}  
Модуль изменения словаря:

package github.cirqach.devlex.app\_pages.fragments  
  
import android.content.Intent  
import android.database.Cursor  
import android.os.Build  
import android.os.Bundle  
import android.util.Log  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import android.widget.Button  
import android.widget.EditText  
import android.widget.SearchView  
import android.widget.TextView  
import android.widget.Toast  
import androidx.annotation.RequiresApi  
import androidx.fragment.app.Fragment  
import androidx.recyclerview.widget.LinearLayoutManager  
import androidx.recyclerview.widget.RecyclerView  
import com.google.android.material.floatingactionbutton.FloatingActionButton  
import github.cirqach.devlex.R  
import github.cirqach.devlex.app\_pages.AddWordActivity  
import github.cirqach.devlex.database.DataList  
import github.cirqach.devlex.database.DevLexAdapter  
import github.cirqach.devlex.database.DevLexDBHelper  
import github.cirqach.devlex.database.DevLexDatabaseContract  
import java.util.Locale  
  
  
class ChangeDatabaseFragment : Fragment() {  
  
 lateinit var dbh: DevLexDBHelper  
 private lateinit var newArray: ArrayList<DataList>  
 private lateinit var recyclerView: RecyclerView  
 private lateinit var searchView: SearchView  
 private lateinit var adapter: DevLexAdapter  
 private lateinit var russianEditText: EditText  
 private lateinit var englishEditText: EditText  
 private lateinit var definitionEditText: EditText  
 private lateinit var deleteButton: Button  
 private lateinit var saveButton: Button  
 private lateinit var addButton: Button  
 private lateinit var idTextView: TextView  
  
 override fun onCreateView(  
 inflater: LayoutInflater, container: ViewGroup?,  
 savedInstanceState: Bundle?  
 ): View? {  
 // Inflate the layout for this fragment  
 return inflater.inflate(R.layout.*fragment\_change\_database*, container, false)  
 }  
  
 @RequiresApi(Build.VERSION\_CODES.*S*)  
 override fun onViewCreated(view: View, savedInstanceState: Bundle?) {  
 super.onViewCreated(view, savedInstanceState)  
  
 if (view.*context* != null) {  
 dbh = DevLexDBHelper(view.*context*)  
 }  
  
 val refreshDatabaseButton: FloatingActionButton =  
 view.findViewById(R.id.*crud\_refresh\_database\_button*)  
 refreshDatabaseButton.setOnClickListener **{** displayWord()  
 **}** recyclerView = view.findViewById(R.id.*crud\_recycler\_view*)  
  
 russianEditText = view.findViewById(R.id.*crud\_russian\_name\_edit\_text\_view*)  
 englishEditText = view.findViewById(R.id.*crud\_english\_name\_edit\_text\_view*)  
 definitionEditText = view.findViewById(R.id.*crud\_definition\_edit\_text\_view*)  
 idTextView = view.findViewById(R.id.*crud\_id\_text\_view*)  
  
  
 recyclerView.*layoutManager* = LinearLayoutManager(view.*context*)  
 recyclerView.setHasFixedSize(true)  
 displayWord()  
  
  
  
 searchView = view.findViewById(R.id.*crud\_search\_view*)  
 searchView.setOnQueryTextListener(object : SearchView.OnQueryTextListener {  
 override fun onQueryTextSubmit(query: String?): Boolean {  
 return false  
 }  
  
 override fun onQueryTextChange(newText: String?): Boolean {  
 filterList(newText)  
 return true  
 }  
  
 })  
  
 deleteButton = view.findViewById(R.id.*crud\_delete\_button*)  
 saveButton = view.findViewById(R.id.*crud\_save\_button*)  
 addButton = view.findViewById(R.id.*crud\_add\_button*)  
 deleteButton.setOnClickListener **{** deleteValues()  
 **}** addButton.setOnClickListener **{** val intent = Intent(requireContext(), AddWordActivity::class.*java*)  
 startActivity(intent)  
 **}** saveButton.setOnClickListener **{** Toast.makeText(view.*context*, "Saving data", Toast.*LENGTH\_SHORT*).show()  
 Log.d(  
 "save\_button\_click", "Data to save: " +  
 "english = ${englishEditText.*text*} , " +  
 "russian = ${russianEditText.*text*}, " +  
 "definition = ${definitionEditText.*text*} ," +  
 "ID = ${idTextView.*text*}"  
 )  
 dbh.saveDataToLexicon(  
 englishEditText.*text*.toString(),  
 russianEditText.*text*.toString(),  
 definitionEditText.*text*.toString(),  
 idTextView.*text*.toString()  
 )  
  
 displayWord()  
 **}** }  
  
 private fun deleteValues() {  
 dbh.deleteData(  
 DevLexDatabaseContract.LexiconEntry.TABLE\_NAME,  
 idTextView.*text*.toString()  
 )  
 displayWord()  
 }  
  
  
 private fun filterList(query: String?) {  
 if (query != null) {  
 val filteredList = ArrayList<DataList>()  
 for (i in newArray) {  
 if (i.englishName.*lowercase*(Locale.*ROOT*).*contains*(query) ||  
 i.russianName.*lowercase*(Locale.*ROOT*).*contains*(query) ||  
 i.definition.*lowercase*(Locale.*ROOT*).*contains*(query)  
 ) {  
 filteredList.add(i)  
 }  
 }  
 if (filteredList.isEmpty()) {  
 Toast.makeText(requireView().*context*, "No data found", Toast.*LENGTH\_SHORT*).show()  
 } else {  
 adapter.setFilteredList(filteredList)  
  
 }  
 }  
 }  
  
 private fun displayWord() {  
 val newCursor: Cursor? = dbh.readAll(DevLexDatabaseContract.LexiconEntry.TABLE\_NAME)  
 newArray = ArrayList()  
 while (newCursor!!.moveToNext()) {  
 val uEnglishName = newCursor.getString(1)  
 val uRussianName = newCursor.getString(2)  
 val uDefinition = newCursor.getString(3)  
 val uid = newCursor.getString(0)  
 newArray.add(DataList(uEnglishName, uRussianName, uDefinition, uid))  
 }  
 recyclerView.*adapter* = DevLexAdapter(newArray)  
 adapter = DevLexAdapter(newArray)  
 recyclerView.*adapter* = adapter  
  
  
 adapter.onItemClick = **{** position **->** // Get the DataList object at the clicked position  
  
 // Set the values of the EditText fields  
 russianEditText.setText(position.russianName)  
 englishEditText.setText(position.englishName)  
 definitionEditText.setText(position.definition)  
 idTextView.*text* = position.id  
 **}** }  
  
  
}

Модуль встречного меню тестов:

package github.cirqach.devlex.app\_pages.fragments.tests  
  
import android.content.Intent  
import android.os.Bundle  
import android.util.Log  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import android.widget.Button  
import android.widget.EditText  
import android.widget.Toast  
import androidx.fragment.app.Fragment  
import github.cirqach.devlex.R  
import github.cirqach.devlex.app\_pages.tests.TrueFalseTestActivity  
  
  
class TrueFalseTestFragment : Fragment() {  
  
 override fun onCreateView(  
 inflater: LayoutInflater, container: ViewGroup?,  
 savedInstanceState: Bundle?  
 ): View? {  
 // Inflate the layout for this fragment  
 return inflater.inflate(R.layout.*fragment\_true\_false\_test*, container, false)  
 }  
  
 override fun onViewCreated(view: View, savedInstanceState: Bundle?) {  
 val button = view.findViewById<Button>(R.id.*tft\_fragment\_start\_button*)  
 button.setOnClickListener **{** val tag = "True false test fragment"  
 Log.d(tag, "onViewCreated: handle button click")  
 val questionsCountEditText =  
 view.findViewById<EditText>(R.id.*true\_false\_test\_fragment\_edit\_text*)  
 if (questionsCountEditText.*text*.toString().*toInt*() <= 0) {  
 Toast.makeText(view.*context*, getString(R.string.*cant\_use\_zero*), Toast.*LENGTH\_SHORT*)  
 .show()  
 } else {  
 val questionsCount = questionsCountEditText.*text*.toString().*toIntOrNull*()  
 Log.d(tag, "onViewCreated: question count = $questionsCount")  
 val intent = Intent(  
 view.*context*,  
 TrueFalseTestActivity::class.*java* )  
 Log.d(tag, "onViewCreated: put extra $questionsCount")  
 intent.putExtra("count\_of\_test", questionsCount)  
 startActivity(intent)  
 }  
 **}** }  
  
}

Модуль адаптера для списков истории:

package github.cirqach.devlex.app\_pages.fragments.StatisticPage  
  
import androidx.fragment.app.Fragment  
import androidx.fragment.app.FragmentManager  
import androidx.lifecycle.Lifecycle  
import androidx.viewpager2.adapter.FragmentStateAdapter  
  
class TabLayoutAdapter(  
 fragmentManager: FragmentManager,  
 lifecycle: Lifecycle  
) : FragmentStateAdapter(fragmentManager, lifecycle) {  
 override fun getItemCount(): Int {  
 return 4  
 }  
  
 override fun createFragment(position: Int): Fragment {  
  
 return when (position) {  
 0 -> FragmentFindTranslationTabLayout()  
 1 -> FragmentFindWordTabLayout()  
 2 -> FragmentTrueFalseTabLayout()  
 3 -> FragmentHangmanGameTabLayout()  
 else -> {  
 FragmentFindTranslationTabLayout()  
 }  
 }  
 }  
}

Модуль истории:

package github.cirqach.devlex.app\_pages.fragments.StatisticPage  
  
import android.os.Bundle  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import androidx.fragment.app.Fragment  
import androidx.viewpager2.widget.ViewPager2  
import com.google.android.material.tabs.TabLayout  
import github.cirqach.devlex.R  
  
class StatisticFragment : Fragment() {  
  
 private lateinit var tabLayout: TabLayout  
 private lateinit var viewPager2: ViewPager2  
 private lateinit var tabLayoutAdapter: TabLayoutAdapter  
  
  
 override fun onCreateView(  
 inflater: LayoutInflater, container: ViewGroup?,  
 savedInstanceState: Bundle?  
 ): View? {  
 // Inflate the layout for this fragment  
 return inflater.inflate(R.layout.*fragment\_statistic*, container, false)  
 }  
  
 override fun onViewCreated(view: View, savedInstanceState: Bundle?) {  
 super.onViewCreated(view, savedInstanceState)  
  
  
  
 tabLayout = view.findViewById(R.id.*StatisticTestTabLayout*)  
 viewPager2 = view.findViewById(R.id.*viewPagerForTests*)  
  
 tabLayoutAdapter = TabLayoutAdapter(*parentFragmentManager*, lifecycle)  
  
 tabLayout.addTab(tabLayout.newTab().setText(getString(R.string.*find\_translation\_test3*)))  
 tabLayout.addTab(tabLayout.newTab().setText(getString(R.string.*find\_word\_test2*)))  
 tabLayout.addTab(tabLayout.newTab().setText(getString(R.string.*true\_false\_test2*)))  
 tabLayout.addTab(tabLayout.newTab().setText(getString(R.string.*hangman\_game2*)))  
  
  
 viewPager2.*adapter* = tabLayoutAdapter  
  
 tabLayout.addOnTabSelectedListener(object : TabLayout.OnTabSelectedListener {  
 override fun onTabSelected(tab: TabLayout.Tab?) {  
 if (tab != null) {  
 viewPager2.*currentItem* = tab.*position* }  
 }  
  
 override fun onTabUnselected(p0: TabLayout.Tab?) {  
 }  
  
 override fun onTabReselected(p0: TabLayout.Tab?) {  
 }  
  
 })  
 viewPager2.registerOnPageChangeCallback(object : ViewPager2.OnPageChangeCallback() {  
 override fun onPageSelected(position: Int) {  
 super.onPageSelected(position)  
 tabLayout.selectTab(tabLayout.getTabAt(position))  
 }  
 })  
  
 }  
  
}

Модули вкладок тестов в истории:

package github.cirqach.devlex.app\_pages.fragments.StatisticPage  
  
import android.database.Cursor  
import android.os.Bundle  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import android.widget.SearchView  
import android.widget.Toast  
import androidx.fragment.app.Fragment  
import androidx.recyclerview.widget.LinearLayoutManager  
import androidx.recyclerview.widget.RecyclerView  
import github.cirqach.devlex.R  
import github.cirqach.devlex.app\_pages.fragments.StatisticPage.adapters.TestResultAdapter  
import github.cirqach.devlex.database.DevLexDBHelper  
import github.cirqach.devlex.database.DevLexDatabaseContract  
import github.cirqach.devlex.database.TestDataList  
import java.util.Locale  
  
  
private lateinit var *dbh*: DevLexDBHelper  
private lateinit var *newArry*: ArrayList<TestDataList>  
private lateinit var *recyclerView*: RecyclerView  
private lateinit var *adapter*: TestResultAdapter  
  
class FragmentTrueFalseTabLayout : Fragment() {  
  
 override fun onCreateView(  
 inflater: LayoutInflater, container: ViewGroup?,  
 savedInstanceState: Bundle?  
 ): View? {  
 // Inflate the layout for this fragment  
 return inflater.inflate(R.layout.*fragment\_true\_false\_tab\_layout*, container, false)  
 }  
  
 companion object;  
  
 override fun onViewCreated(view: View, savedInstanceState: Bundle?) {  
 super.onViewCreated(view, savedInstanceState)  
  
 *recyclerView* = view.findViewById(R.id.*tfRecyclerView*)  
 *dbh* = DevLexDBHelper(view.*context*)  
 *recyclerView*.*layoutManager* = LinearLayoutManager(view.*context*)  
 *recyclerView*.setHasFixedSize(true)  
 displayWord()  
  
 *adapter* = TestResultAdapter(*newArry*)  
 *recyclerView*.*adapter* = *adapter* val searchView: SearchView = view.findViewById(R.id.*tfSearchView*)  
 searchView.setOnQueryTextListener(object : SearchView.OnQueryTextListener {  
 override fun onQueryTextSubmit(query: String?): Boolean {  
 return false  
 }  
  
 override fun onQueryTextChange(newText: String?): Boolean {  
 filterList(newText)  
 return true  
 }  
  
 })  
  
  
 }  
  
 private fun filterList(query: String?) {  
 if (query != null) {  
 val filteredList = ArrayList<TestDataList>()  
 for (i in *newArry*) {  
 if (i.score.*lowercase*(Locale.*ROOT*).*contains*(query) ||  
 i.resultPercent.toString().*lowercase*(Locale.*ROOT*).*contains*(query)  
 ) {  
 filteredList.add(i)  
 }  
 }  
 if (filteredList.isEmpty()) {  
 Toast.makeText(requireView().*context*, getString(R.string.*no\_data\_found*), Toast.*LENGTH\_SHORT*).show()  
 } else {  
 *adapter*.setFilteredList(filteredList)  
  
 }  
 }  
 }  
  
 private fun displayWord() {  
 val newCursor: Cursor? = *dbh*.readAll(DevLexDatabaseContract.Tables.TRUE\_FALSE\_TABLE)  
 *newArry* = ArrayList()  
 while (newCursor!!.moveToNext()) {  
 val result = newCursor.getInt(1)  
 val resultProcent = newCursor.getInt(2)  
 val questionCount = newCursor.getInt(3)  
 *newArry*.add(TestDataList("$result/$questionCount", resultProcent))  
 }  
 *recyclerView*.*adapter* = TestResultAdapter(*newArry*)  
 }  
}

Модуль «Data base helper»:

package github.cirqach.devlex.database  
  
import android.content.ContentValues  
import android.content.Context  
import android.database.Cursor  
import android.database.sqlite.SQLiteDatabase  
import android.database.sqlite.SQLiteException  
import android.database.sqlite.SQLiteOpenHelper  
import android.util.Log  
import java.io.File  
import java.io.FileOutputStream  
import java.io.IOException  
import java.io.InputStream  
import java.io.OutputStream  
  
  
class DevLexDBHelper(context: Context?) :  
 SQLiteOpenHelper(context, DATABASE\_NAME, null, DATABASE\_VERSION) {  
  
  
 companion object {  
 const val TAG: String = "DatabaseHelper"  
  
 private const val DATABASE\_NAME = "DevLex.db"  
  
 private const val DATABASE\_VERSION = 1  
  
 fun getTableRowCount(tableName: String, dbh: DevLexDBHelper): Int {  
 val db = dbh.readableDatabase  
 val cursor = db.query(tableName, null, null, null, null, null, null)  
 val count = cursor.count  
 cursor.close()  
 Log.d(TAG, "getTableRowCount: returning $count")  
 return count  
 }  
 }  
  
 override fun onCreate(db: SQLiteDatabase) {  
 Log.d(TAG, "onCreate: called OnCreate")  
 }  
  
  
 @Throws(IOException::class)  
 fun copyDatabase(context: Context?) {  
 if (context == null) {  
 Log.d(TAG, "copyDatabase: context null")  
 } else {  
 if (!checkDataBase(context)) {  
 Log.d(TAG, "copyDatabase: database doesn't exist")  
 Log.d(TAG, "copyDatabase: starting copying")  
 val myInput: InputStream = context.assets!!.open(DATABASE\_NAME)  
 val outputFile = context.getDatabasePath(DATABASE\_NAME)  
 val myOutput: OutputStream = FileOutputStream(outputFile)  
 val buffer = ByteArray(1024)  
 var length: Int  
 while (myInput.read(buffer).also **{** length = it **}** > 0) {  
 myOutput.write(buffer, 0, length)  
 }  
 myOutput.flush()  
 myOutput.close()  
 myInput.close()  
 if (checkDataBase(context)) {  
 Log.d(TAG, "copyDatabase: database copying successful")  
 } else {  
 Log.d(TAG, "copyDatabase: database copying failed")  
 }  
 } else {  
 Log.d(TAG, "copyDatabase: database exist")  
 }  
 }  
 }  
  
 private fun checkDataBase(context: Context?): Boolean {  
 val dataBasePath: String? = context?.applicationInfo?.dataDir?.let **{** "$it/assets/" **}** var checkDataBase = false  
 try {  
 val myPath = dataBasePath + DATABASE\_NAME  
 val dataBaseFile = File(myPath)  
 checkDataBase = dataBaseFile.exists()  
 } catch (e: SQLiteException) {  
 Log.d(TAG, "checkdatabase: ${e.stackTrace}")  
 }  
 return checkDataBase  
 }  
  
 override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {  
 Log.d(TAG, "onUpgrade: CALLED ON UPGRADE")  
 }  
  
 // add data to Lexicon  
 fun addDataToLexicon(englishName: String, russianName: String, definition: String): Boolean {  
 val p0 = this.writableDatabase  
 val contentValues = ContentValues()  
 contentValues.put(DevLexDatabaseContract.LexiconEntry.ENGLISH\_NAME, englishName)  
 contentValues.put(DevLexDatabaseContract.LexiconEntry.RUSSIAN\_NAME, russianName)  
 contentValues.put(DevLexDatabaseContract.LexiconEntry.WORD\_DEFENITION, definition)  
 val result = p0.insert(DevLexDatabaseContract.LexiconEntry.TABLE\_NAME, null, contentValues)  
 return result != (-1).toLong()  
 }  
  
 fun readAll(tableName: String): Cursor? {  
 val p0 = this.writableDatabase  
 Log.d(TAG, "readAll: reading all data from $tableName in ${p0.path}")  
 val cursor =  
 p0.rawQuery("select \* from $tableName", null)  
 return cursor  
 }  
  
 fun deleteData(tableName: String, id: String) {  
 val db = this.writableDatabase  
 Log.d(TAG, "deleteData: trying deleting")  
 db.delete(tableName, "ID = ?", arrayOf(id))  
 Log.d(TAG, "deleteData: item with $id delete, i perhaps")  
 db.close()  
 }  
  
 fun saveDataToLexicon(  
 englishName: String,  
 russianName: String,  
 definition: String,  
 id: String  
 ) {  
 val db = writableDatabase  
 if (db == null || !db.isOpen) {  
 return  
 }  
 try {  
 val contentValues = ContentValues().apply **{** put(DevLexDatabaseContract.LexiconEntry.ENGLISH\_NAME, englishName)  
 put(DevLexDatabaseContract.LexiconEntry.RUSSIAN\_NAME, russianName)  
 put(DevLexDatabaseContract.LexiconEntry.WORD\_DEFENITION, definition)  
 **}** val whereClause = "${DevLexDatabaseContract.LexiconEntry.ID} = ?"  
 val whereArgs = arrayOf(id)  
 val rowsUpdated = db.update(  
 DevLexDatabaseContract.LexiconEntry.TABLE\_NAME,  
 contentValues,  
 whereClause,  
 whereArgs  
 )  
 } catch (e: Exception) {  
 Log.e(TAG, "Error saving data: $e")  
 } finally {  
 db.close()  
 }  
 }  
  
 fun saveDataToTest(  
 tableName: String,  
 result: Int,  
 resultProcent: Int,  
 countOfQuestions: Int  
 ): Boolean {  
 Log.d(TAG, "saveDataToTest: saving $tableName + $result + $resultProcent")  
 val p0 = this.writableDatabase  
 val contentValues = ContentValues()  
 contentValues.put(DevLexDatabaseContract.Test.RESULT, result)  
 contentValues.put(DevLexDatabaseContract.Test.RESULT\_PROCENT, resultProcent)  
 contentValues.put(DevLexDatabaseContract.Test.QUESTION\_COUNT, countOfQuestions)  
 val returnResult = p0.insert(tableName, null, contentValues)  
 return returnResult != (-1).toLong()  
 }  
  
 fun getDataByIdFromLexicon(tableName: String, id: Int): DataList? {  
 val db = readableDatabase  
 val query =  
 "SELECT \* FROM $tableName WHERE ${DevLexDatabaseContract.LexiconEntry.ID} = $id"  
 val cursor = db.rawQuery(query, null)  
 cursor.moveToFirst()  
 if (id != 0) {  
 if (cursor.moveToFirst()) {  
 val englishName =  
 cursor.getString(cursor.getColumnIndexOrThrow(DevLexDatabaseContract.LexiconEntry.ENGLISH\_NAME))  
 val russianName =  
 cursor.getString(cursor.getColumnIndexOrThrow(DevLexDatabaseContract.LexiconEntry.RUSSIAN\_NAME))  
 val definition =  
 cursor.getString(cursor.getColumnIndexOrThrow(DevLexDatabaseContract.LexiconEntry.WORD\_DEFENITION))  
 val result =  
 DataList(englishName, russianName, definition, id.toString())  
 cursor.close()  
 return result  
 } else {  
 cursor.close()  
 return null  
 }  
 }  
 cursor.close()  
 return null  
 }  
  
 fun doesIdExist(tableName: String, id: Int): Boolean {  
 val db = readableDatabase  
 val query =  
 "SELECT 1 FROM $tableName WHERE ${DevLexDatabaseContract.LexiconEntry.ID} = $id"  
 val cursor = db.rawQuery(query, null)  
 val exists = cursor.moveToFirst()  
 cursor.close()  
 return exists  
 }  
  
 fun saveHangmanResult(  
 result: String,  
 word: String  
 ): Boolean {  
 val p0 = this.writableDatabase  
 val contentValues = ContentValues()  
 contentValues.put(DevLexDatabaseContract.HangmanGame.RESULT, result)  
 contentValues.put(DevLexDatabaseContract.HangmanGame.WORD, word)  
 val returnResult =  
 p0.insert(DevLexDatabaseContract.HangmanGame.TABLE\_NAME, null, contentValues)  
 return returnResult != (-1).toLong()  
 }  
}  
Контракт с базой данных:

package github.cirqach.devlex.database  
  
import android.provider.BaseColumns  
  
class DevLexDatabaseContract {  
  
 object LexiconEntry : BaseColumns {  
 const val TABLE\_NAME = "lexicon\_table"  
 const val ID = "ID"  
 const val WORD\_DEFENITION = "WORD\_DEFENITION"  
 const val ENGLISH\_NAME = "ENGLISH\_NAME"  
 const val RUSSIAN\_NAME = "RUSSIAN\_NAME"  
 }  
  
 object Test : BaseColumns {  
 const val RESULT = "RESULT"  
 const val RESULT\_PROCENT = "RESULT\_PROCENT"  
 const val QUESTION\_COUNT = "QUESTION\_COUNT"  
 }  
  
 object Tables : BaseColumns {  
 const val FIND\_WORD\_TABLE = "test\_find\_word\_table"  
 const val FIND\_TRANSLATION\_TABLE = "test\_find\_translation\_table"  
 const val TRUE\_FALSE\_TABLE = "test\_true\_false\_table"  
 }  
  
 object HangmanGame : BaseColumns {  
 const val TABLE\_NAME = "hangman\_game"  
 const val ID = "id"  
 const val WORD = "word"  
 const val RESULT = "result"  
 }  
}

Модуль адаптер для словаря:

package github.cirqach.devlex.database  
  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import android.widget.TextView  
import androidx.recyclerview.widget.RecyclerView  
import github.cirqach.devlex.R  
  
class DevLexAdapter(private var wordList: ArrayList<DataList>) :  
 RecyclerView.Adapter<DevLexAdapter.DevLexViewHolder>() {  
  
 var onItemClick: ((DataList) -> Unit)? = null  
  
 class DevLexViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {  
 val tEnglishName: TextView = itemView.findViewById(R.id.*english\_name\_textview*)  
 val tRussianName: TextView = itemView.findViewById(R.id.*russian\_name\_text\_view*)  
 val tDefinition: TextView = itemView.findViewById(R.id.*word\_definition\_textview*)  
 val tid: TextView = itemView.findViewById(R.id.*id\_text\_view*)  
 }  
  
 override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): DevLexViewHolder {  
 val itemView =  
 LayoutInflater.from(parent.*context*)  
 .inflate(R.layout.*lexicon\_recyclerview\_item*, parent, false)  
 return DevLexViewHolder(itemView)  
 }  
  
 override fun getItemCount(): Int {  
 return wordList.size  
 }  
  
 override fun onBindViewHolder(holder: DevLexViewHolder, position: Int) {  
 if (position < wordList.size) {  
 val currentItem = wordList[position]  
 holder.tEnglishName.*text* = currentItem.englishName  
 holder.tRussianName.*text* = currentItem.russianName  
 holder.tDefinition.*text* = currentItem.definition  
 holder.tid.*text* = currentItem.id  
 }  
  
 holder.itemView.setOnClickListener **{** if (position < wordList.size) {  
 onItemClick?.invoke(wordList[position])  
 }  
 **}** }  
  
 fun setFilteredList(wordList: ArrayList<DataList>) {  
 this.wordList = wordList  
 notifyDataSetChanged() // Notify that the data has changed  
 }  
  
  
}

Модуль помощник для тестов:

package github.cirqach.devlex.database.TrueFalseTest  
  
import android.util.Log  
import github.cirqach.devlex.database.DataList  
import github.cirqach.devlex.database.DevLexDBHelper  
import github.cirqach.devlex.database.DevLexDatabaseContract  
import kotlin.random.Random  
  
class QuestionBaseHelper {  
 private val tag = "true false test"  
  
 data class Question(  
 val russianOrEnglishName: String,  
 val definition: String,  
 val answer: Boolean // Change the type to Boolean  
 )  
  
  
 fun getQuestionsList(dbh: DevLexDBHelper?, questionsCount: Int): MutableList<Question?> {  
 if (dbh == null) {  
 Log.d("getQuestionsList", "dbh is null")  
 return *mutableListOf*(null) // Return an empty list with a single null element to indicate failure  
 }  
  
 val tableRowCount = DevLexDBHelper.getTableRowCount(  
 DevLexDatabaseContract.LexiconEntry.TABLE\_NAME,  
 dbh  
 )  
  
 if (tableRowCount == 0) {  
 Log.d("getQuestionsList", "No questions found in the database")  
 return *mutableListOf*(null) // Return an empty list with a single null element to indicate failure  
 }  
  
 val questionsList = *mutableListOf*<Question?>()  
 var retrievedCount = 0  
  
 while (retrievedCount < questionsCount) {  
 val question = getRandomQuestion(dbh)  
 if (question != null && question.definition.*isNotEmpty*()) {  
 questionsList.add(question)  
 retrievedCount++  
 }  
 }  
  
 if (questionsList.size < questionsCount) {  
 Log.d(  
 "getQuestionsList",  
 "Retrieved only ${questionsList.size} questions out of requested $questionsCount"  
 )  
 }  
  
 return questionsList  
 }  
  
  
 private fun getRandomQuestion(dbh: DevLexDBHelper?): Question? {  
 if (dbh == null) {  
 Log.d("getRandomQuestion", "dbh is null")  
 return null  
 }  
  
 val tableRowCount = DevLexDBHelper.getTableRowCount(  
 DevLexDatabaseContract.LexiconEntry.TABLE\_NAME,  
 dbh  
 )  
  
 if (tableRowCount == 0) {  
 Log.d("getRandomQuestion", "No questions found in the database")  
 return null  
 }  
  
 // Try retrieving a random question up to 3 times (adjustable)  
 for (attempt in 1..2) {  
 try {  
 val randomIndex = Random.nextInt(0, tableRowCount)  
 val questionData = dbh.getDataByIdFromLexicon(  
 DevLexDatabaseContract.LexiconEntry.TABLE\_NAME,  
 randomIndex + 1 // Adjust for 0-based indexing  
 )  
 Log.d(tag, "getRandomQuestion: question data = $questionData")  
  
 if (questionData != null) { // Check if data is retrieved successfully  
 val options = *mutableListOf*(  
 questionData,  
 getRandomQuestionExceptId(dbh, questionData.id.*toInt*(), tableRowCount)  
 ?: continue,  
 )  
  
 options.*shuffle*()  
  
 val russianOrEnglishName = Random.nextBoolean()  
 val definition = if (Random.nextBoolean()) {  
 options[0].definition  
 } else {  
 options[1].definition  
 }  
  
 return Question(  
 if (russianOrEnglishName) {  
 questionData.englishName  
 } else {  
 questionData.russianName  
 },  
 definition,  
 questionData.definition == definition  
 )  
 }  
  
 } catch (e: Exception) {  
 Log.e("getRandomQuestion", "Error getting random question: ${e.message}")  
 }  
 }  
  
 Log.d("getRandomQuestion", "Failed to retrieve a random question after 3 attempts")  
 return null // Return null after all attempts fail  
 }  
  
  
 private fun getRandomQuestionExceptId(  
 dbh: DevLexDBHelper,  
 id: Int,  
 tableRowCount: Int  
 ): DataList? {  
 while (true) {  
 val randomNumber = Random.nextInt(1, tableRowCount + 1) // Include upper bound  
 if (randomNumber != id && dbh.doesIdExist(  
 DevLexDatabaseContract.LexiconEntry.TABLE\_NAME,  
 randomNumber  
 )  
 ) {  
 return dbh.getDataByIdFromLexicon(  
 DevLexDatabaseContract.LexiconEntry.TABLE\_NAME,  
 randomNumber  
 )  
 }  
 }  
 }  
  
  
}