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](#_Toc106902805)

[1 Символьный и строковый тип данных 5](#_Toc106902806)

[2 Описание алгоритмов и технологии решения 7](#_Toc106902807)

[2.1 Требование к программному продукту 7](#_Toc106902808)

[2.2 Описание алгоритмов и технологий решений 7](#_Toc106902809)

[2.3 Организация диалога с пользователем 8](#_Toc106902810)

[3 Выполнение практической части 13](#_Toc106902811)

[3.1 Интерфейс приложения 13](#_Toc106902812)

[3.2 Программная реализация 16](#_Toc106902813)

[3.3 Контроль корректности вводимых данных 22](#_Toc106902814)

[4 Результаты работы 25](#_Toc106902815)

[Заключение 35](#_Toc106902816)

[Список использованных источников 36](#_Toc106902817)

[Приложение А Листинг программы 37](#_Toc106902818)

# 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:

* «CRUD»: «Create» — создания записи, «Read» — чтения записи, «Update» — обновления записи, «Delete» — 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 алгоритмов и технологий решений

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 Разработанное программное обеспечение

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.

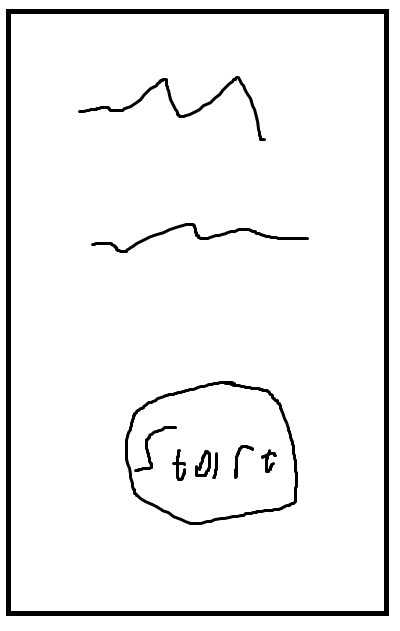


Рисунок 2.1 - Форма «activity\_main»

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

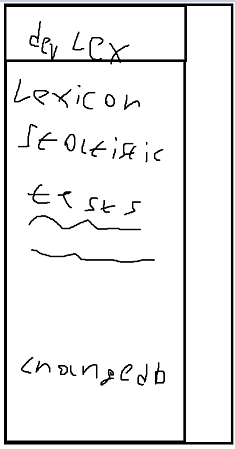


Рисунок 2.2. – Главная форма

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

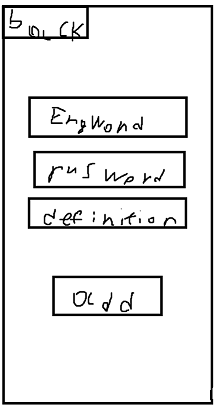


Рисунок 2.3 - Форма добавления записи

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

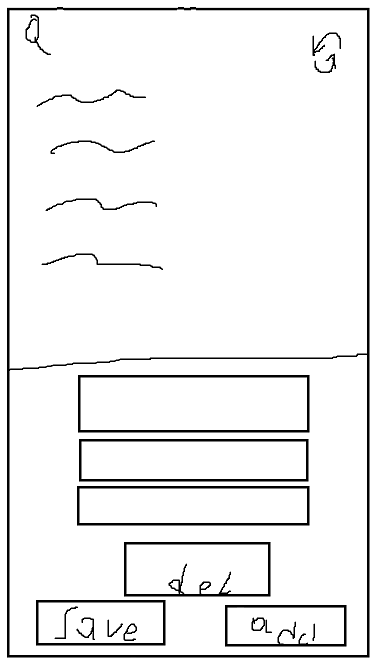


Рисунок 2.4 - Форма изменения базы данных приложения

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.

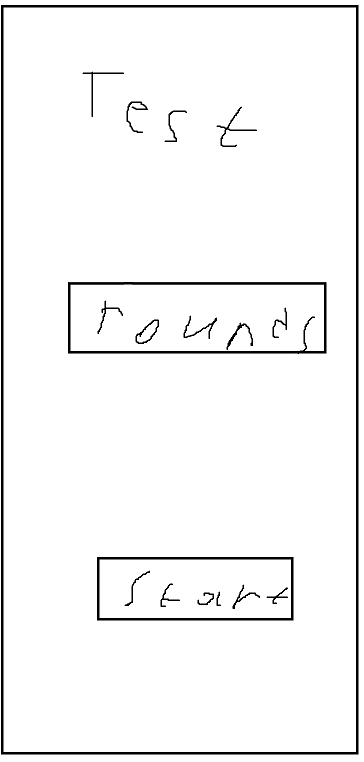


Рисунок 2.5 - Форма теста

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.

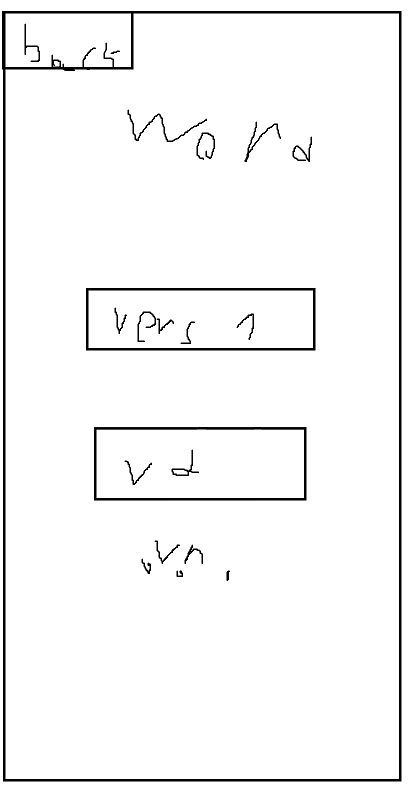


Рисунок 2.6 - Форма прохождения тестов

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

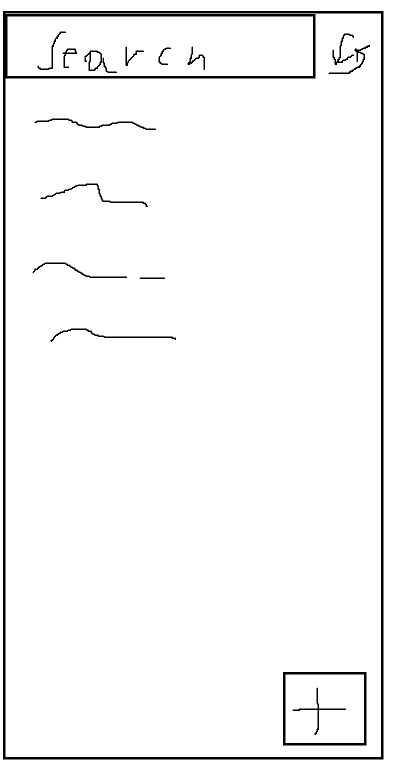


Рисунок 2.7 - Форма словаря

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

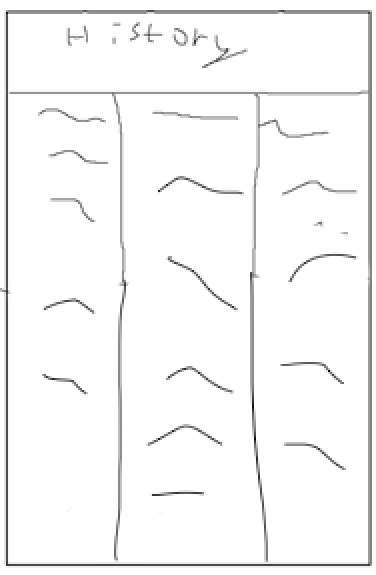


Рисунок 2.8 - Форма статистики

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

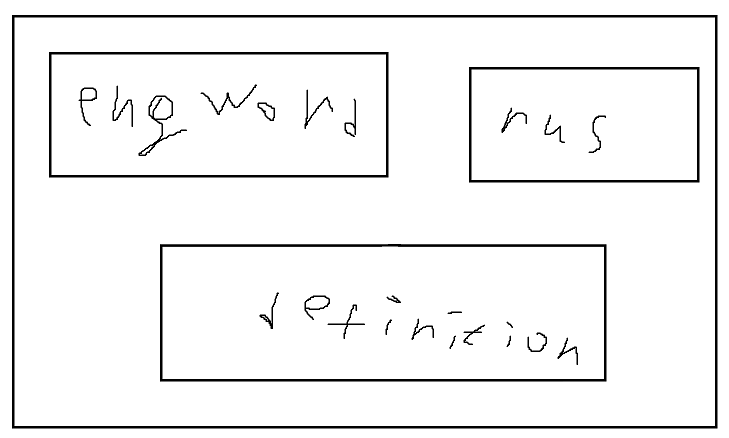


Рисунок 2.9 - Фрагмент списка

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.

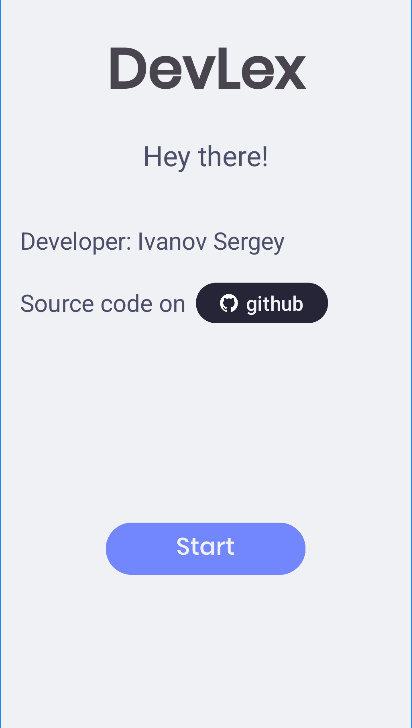


Рисунок 3.1 – Встречная форма

Главная форма, изображённая на рисунке 3.2. Данная форма представляет из себя выпадающее меню, в первую очередь пользователю представиться первый фрагмент меню «Lexicon» на котором расположен весь теоретический материал.

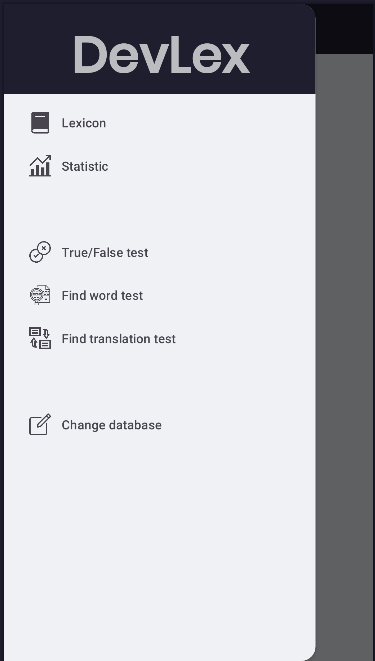


Рисунок 3.2 – Главная форма

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.

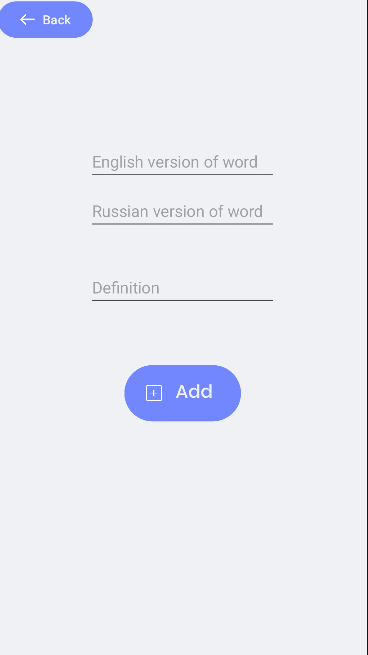


Рисунок 3.3 - Форма добавления новой записи

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.

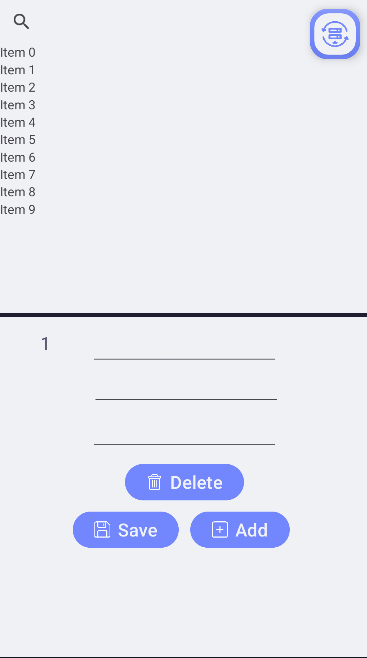


Рисунок 3.4 - Фрагмент изменения базы данных

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.

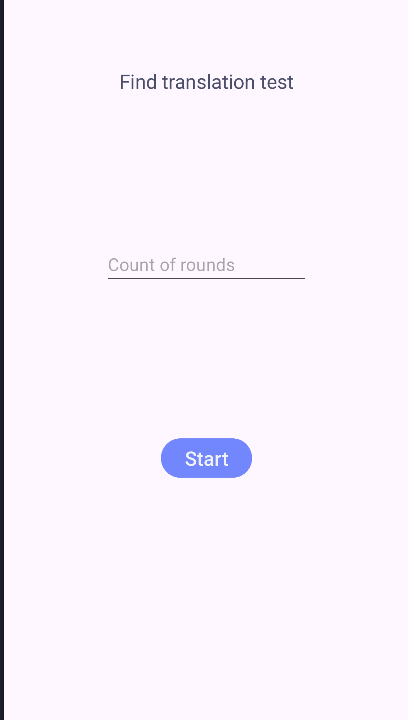


Рисунок 3.5 - Фрагмент «fragment\_find\_translation\_test»

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.

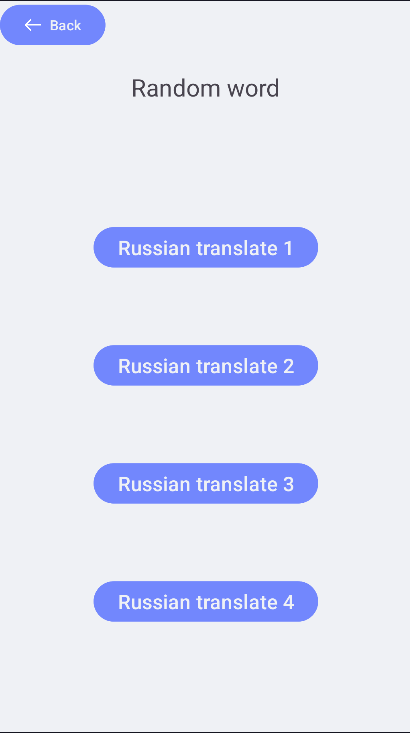


Рисунок 3.6 - Форма «activity\_find\_tranlation\_test»

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.

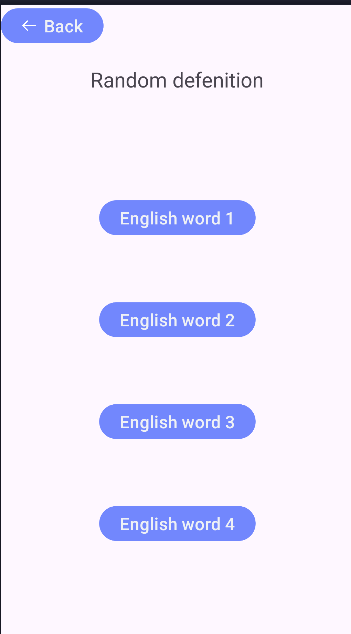


Рисунок 3.7 - Форма «activity\_find\_word\_test»

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.

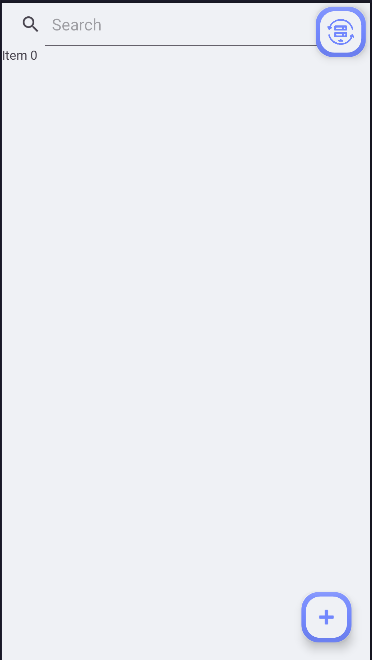


Рисунок 3.8 - Фрагмент «fragment\_lexicon»

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.

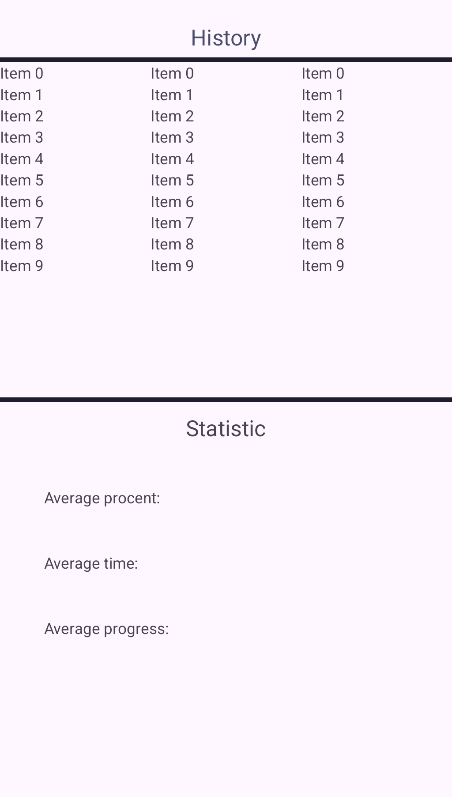


Рисунок 3.9 - Форма «fragment\_statistic»

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.

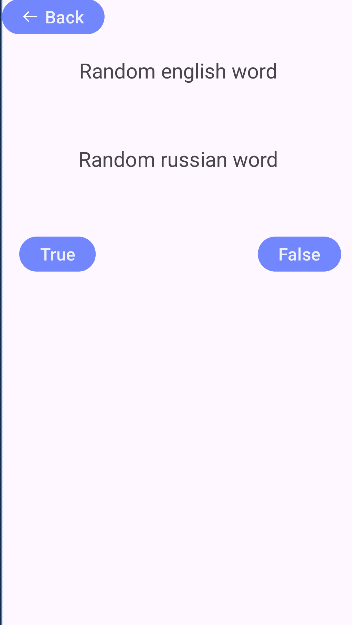


Рисунок 3.10 - Форма «activity\_true\_false\_test»

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.

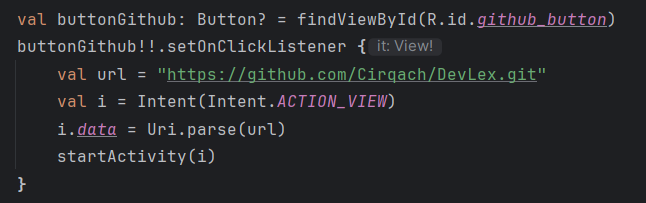


Рисунок 3.11 – Код процедуры нажатия на кнопку.

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.

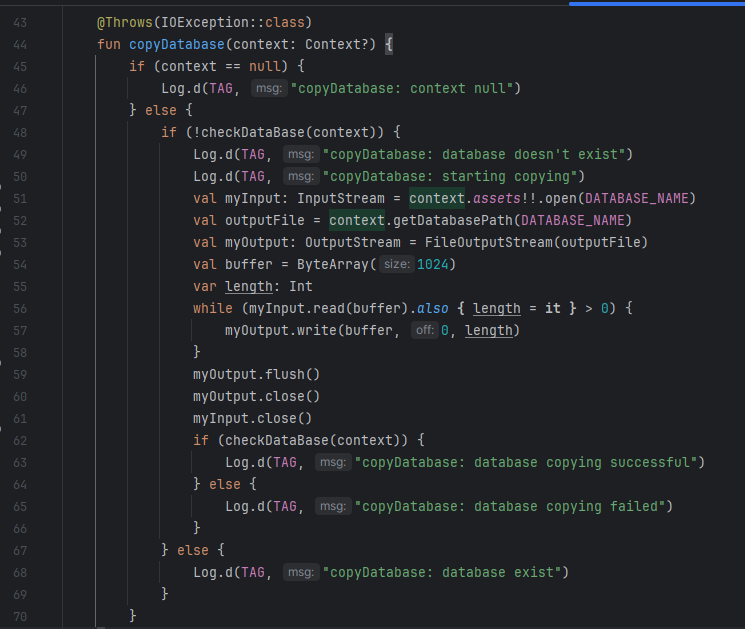


Рисунок 3.12 – Код процедуры копирования базы данных

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

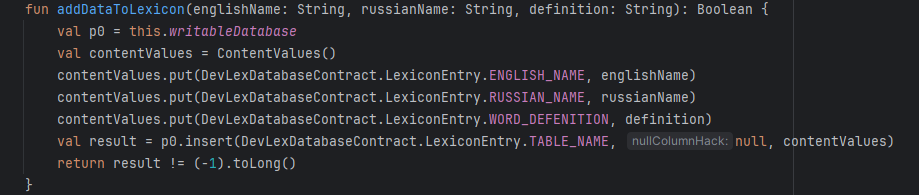


Рисунок 3.13 – Код функции добавления данных в таблицу словаря

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.

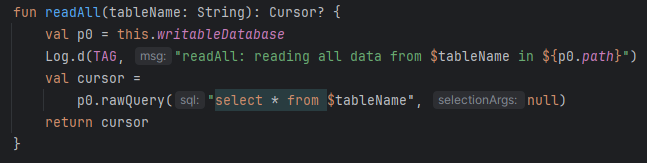


Рисунок 3.14 – Код функция чтение всех данных из таблицы

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.

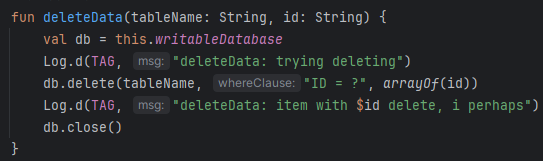


Рисунок 3.15 – Функция удаление данных из таблицы по «ID»

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.

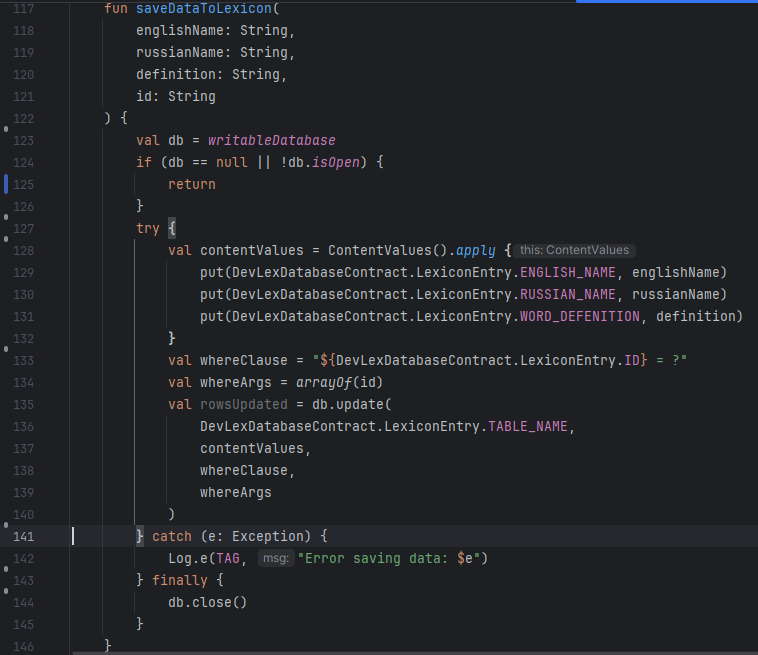


Рисунок 3.16 – Функция сохранения данных в словарь

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.

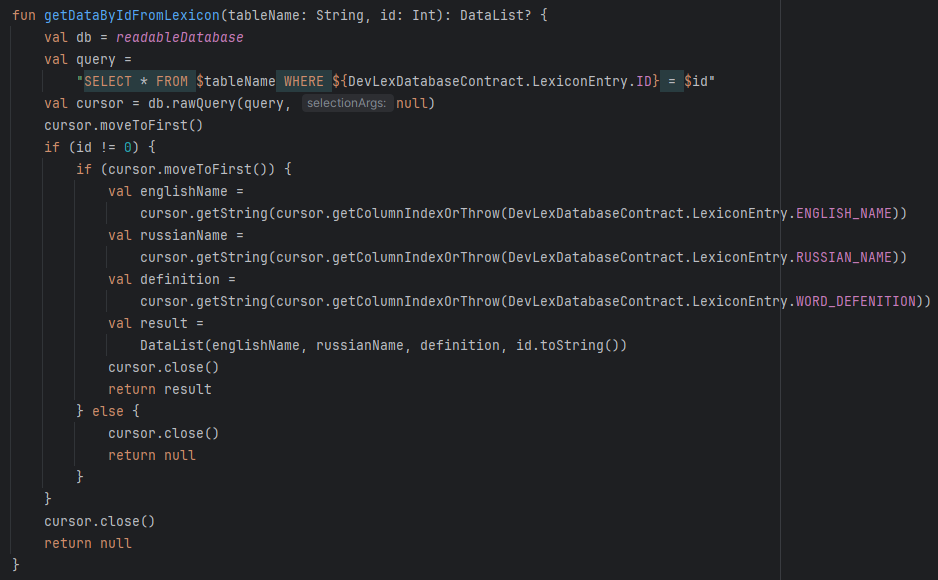


Рисунок 3.17 – «Взятие данные из словаря по ID»

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

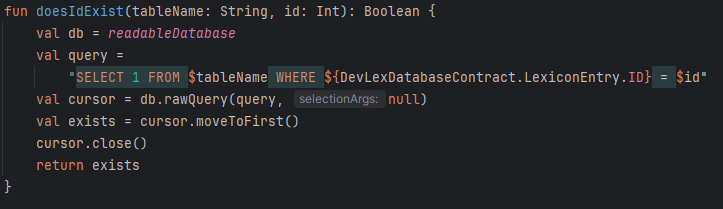


Рисунок 3.18 – «Проверка данных на существование»

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.

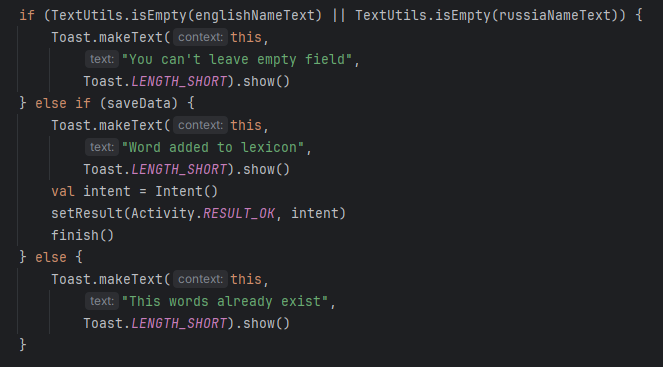


Рисунок 3.19 – Фрагмент кода валидации введенных данных

Полный код программы приведен в приложении А.

# 4 Результаты работы

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

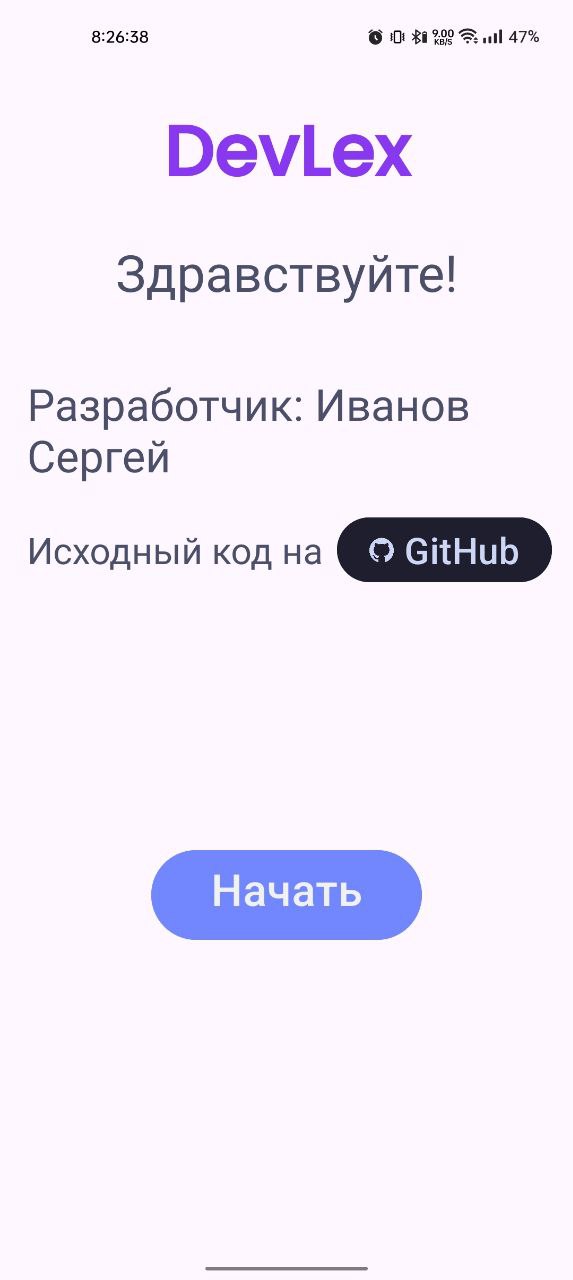


Рисунок 4.1 — Начальная форма

При нажатии кнопки «Начать» открывается основная форма. На рисунке 4.2. представлена основная форма.



Рисунок 4.2 — Основная форма

Основная форма представляет графический интерфейс для изучения теоретического материала.

На основной форме теоретический материал можно отсортировать по английскому и русскому порядку в алфавите. Результаты сортировки представлены на рисунках 4.3, 4.4, 4.5.

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

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, при неправильном заполнении данных, обязательное поле отсутствует или запись с такими данными уже существует, отображается уведомление, изображено на рисунке 4.8. Если все данные заполнены верно, то запись добавляется в базу данных, а форма меняется на основную, результат изображен на рисунке 4.9.

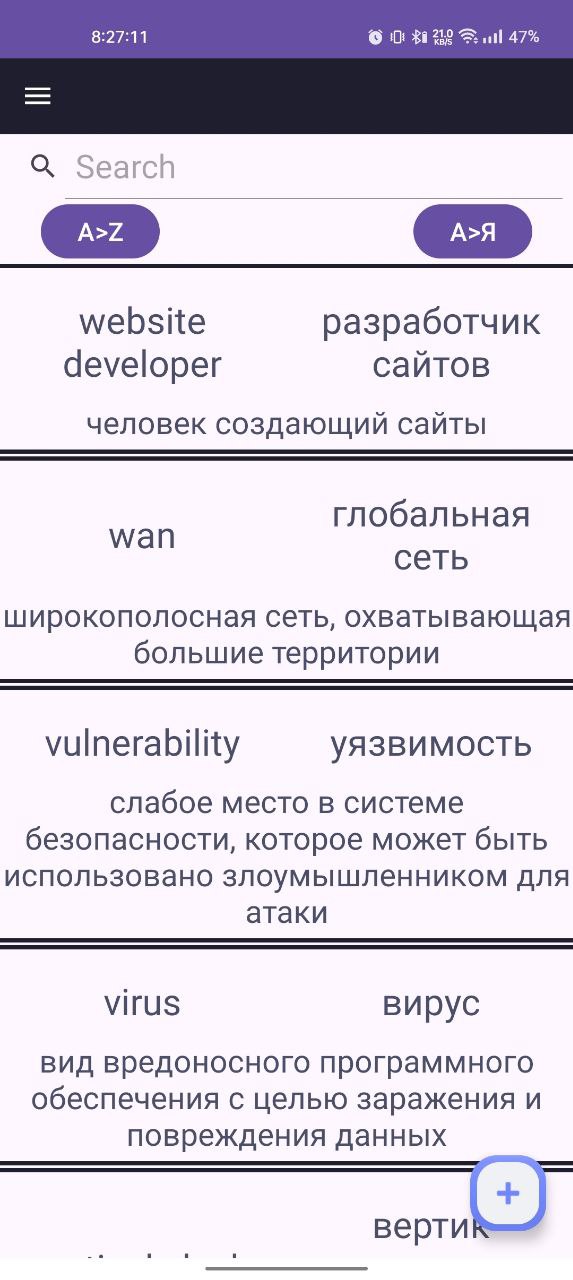


Рисунок 4.3 – Сортировка по английскому алфавиту в обратном порядке



Рисунок 4.4 – Сортировка по русскому алфавиту

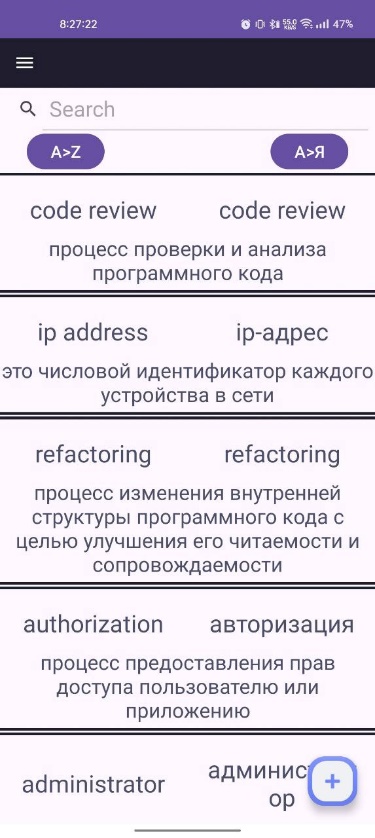


Рисунок 4.5 – Сортировка по русскому алфавиту в обратном порядке

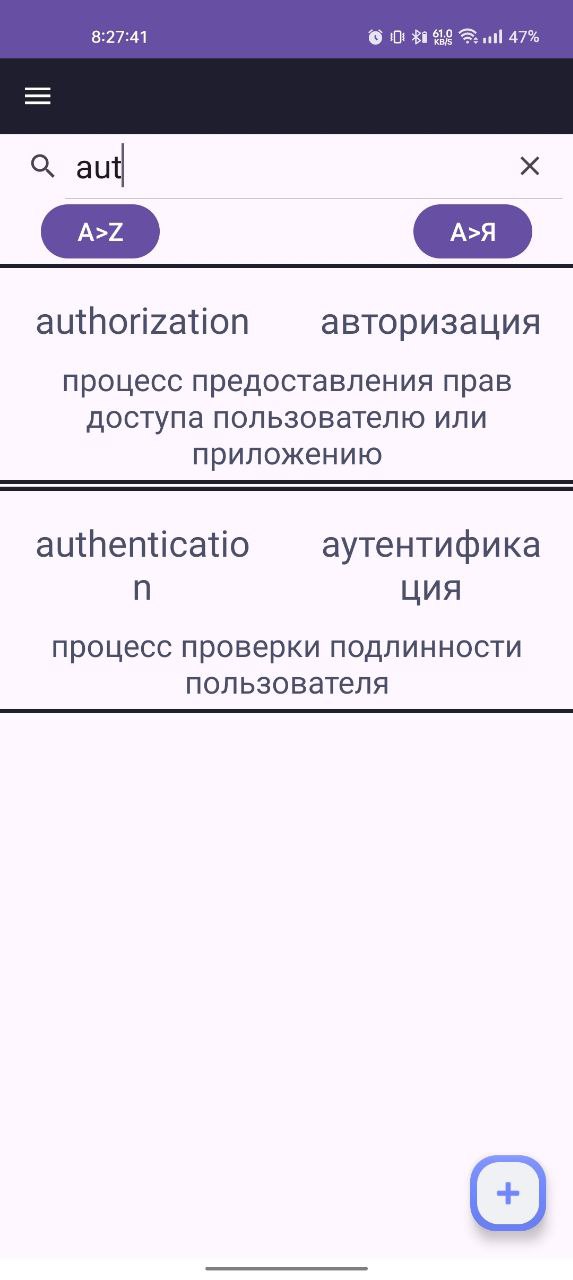


Рисунок 4.6 – Результат работы поиска

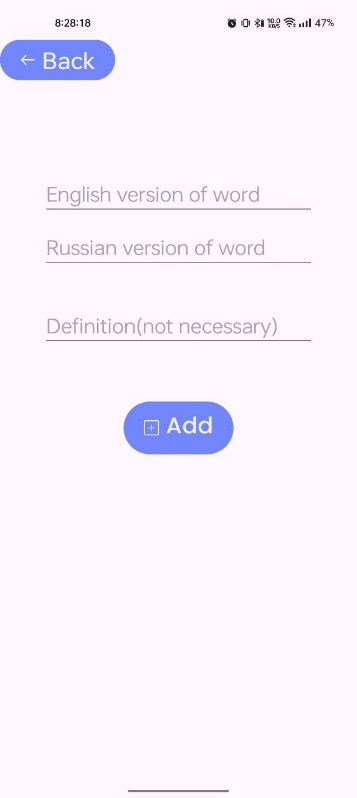


Рисунок 4.7 – Форма добавления новых записей



Рисунок 4.8 – Уведомление о некорректности введённых данных

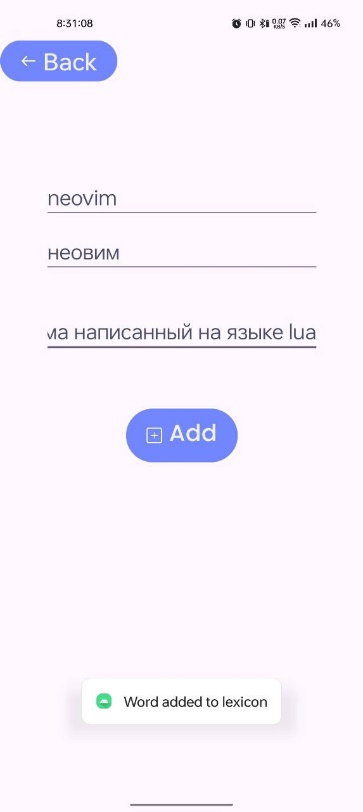


Рисунок 4.9 – Уведомление о добавлении новой записи

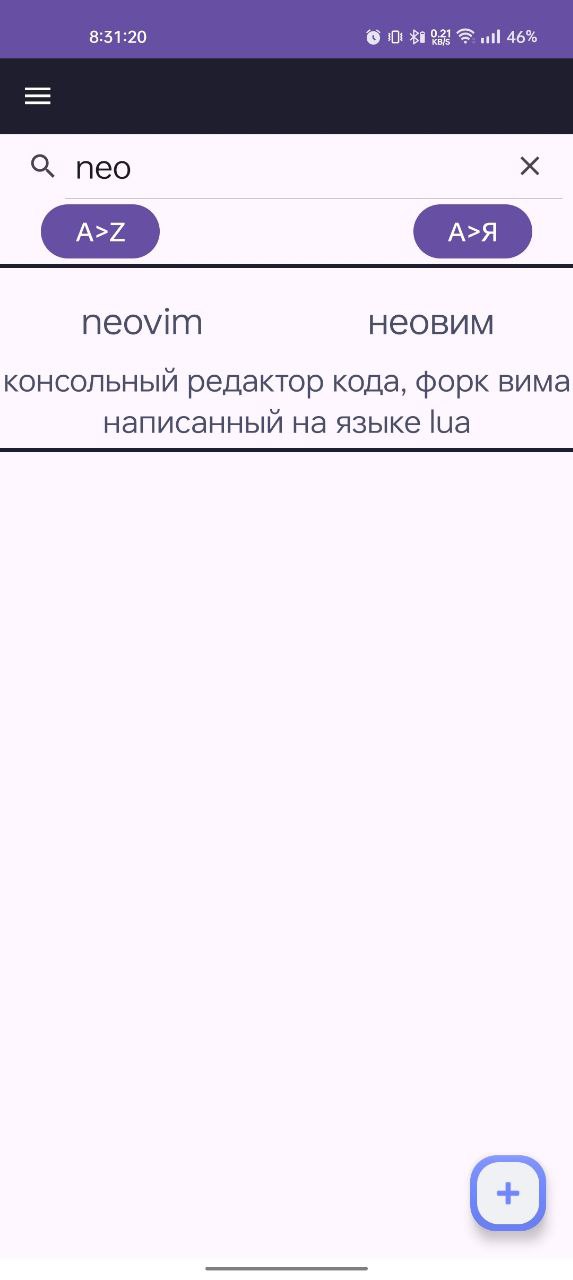


Рисунок 4.10 – Результат добавления записи

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

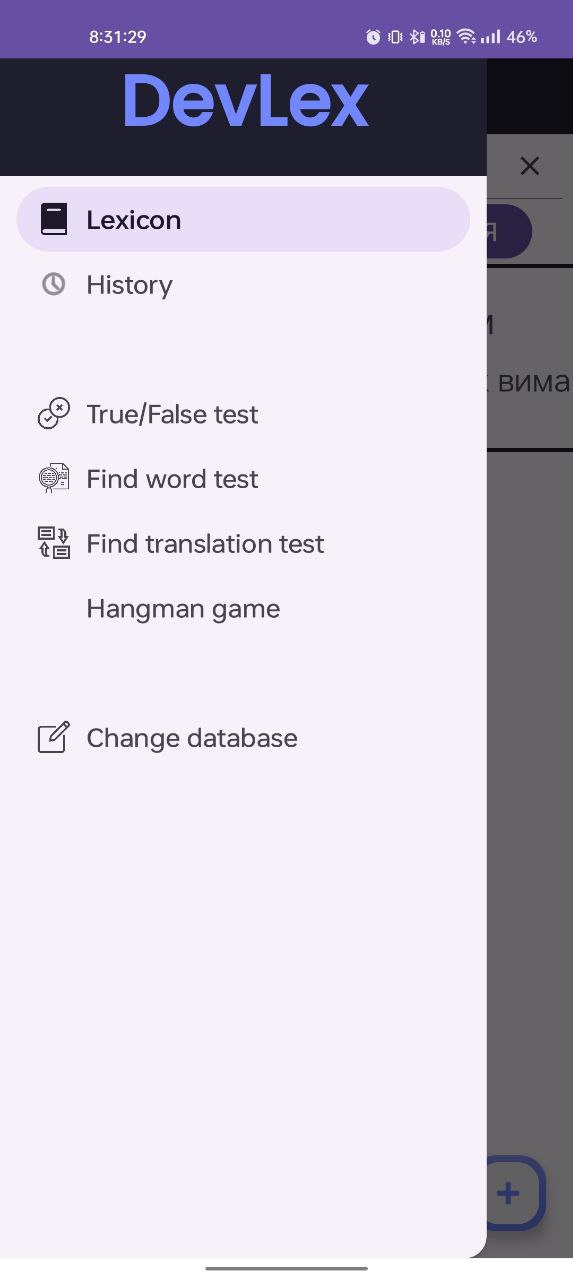


Рисунок 4.11 – Выпадающее меню

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

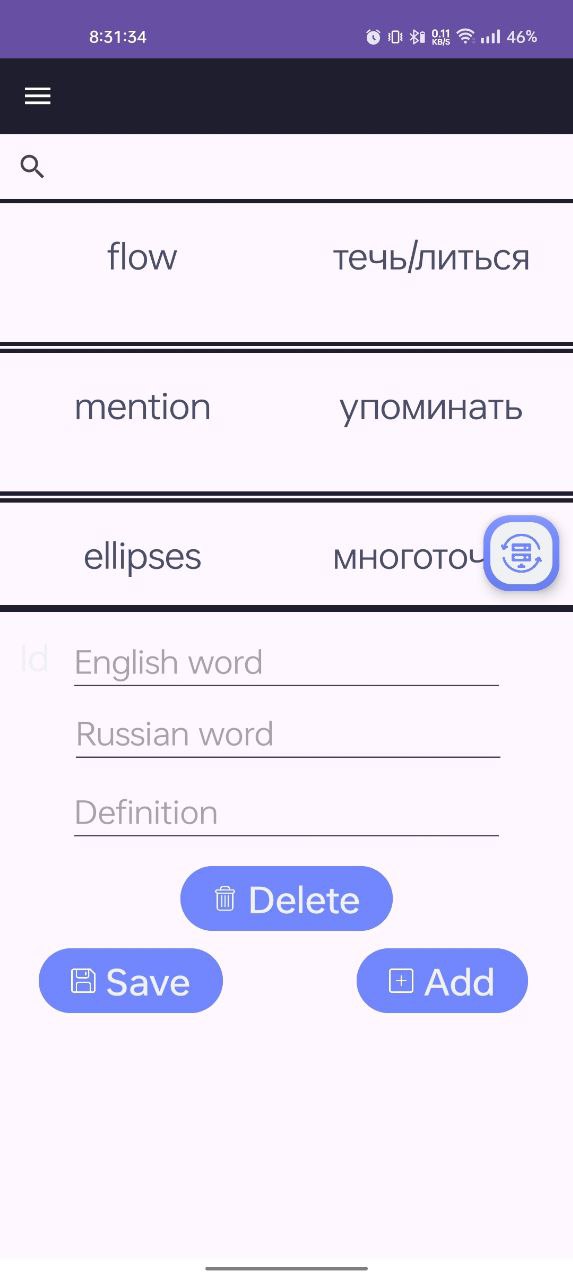


Рисунок 4.12 – Форма изменения словаря

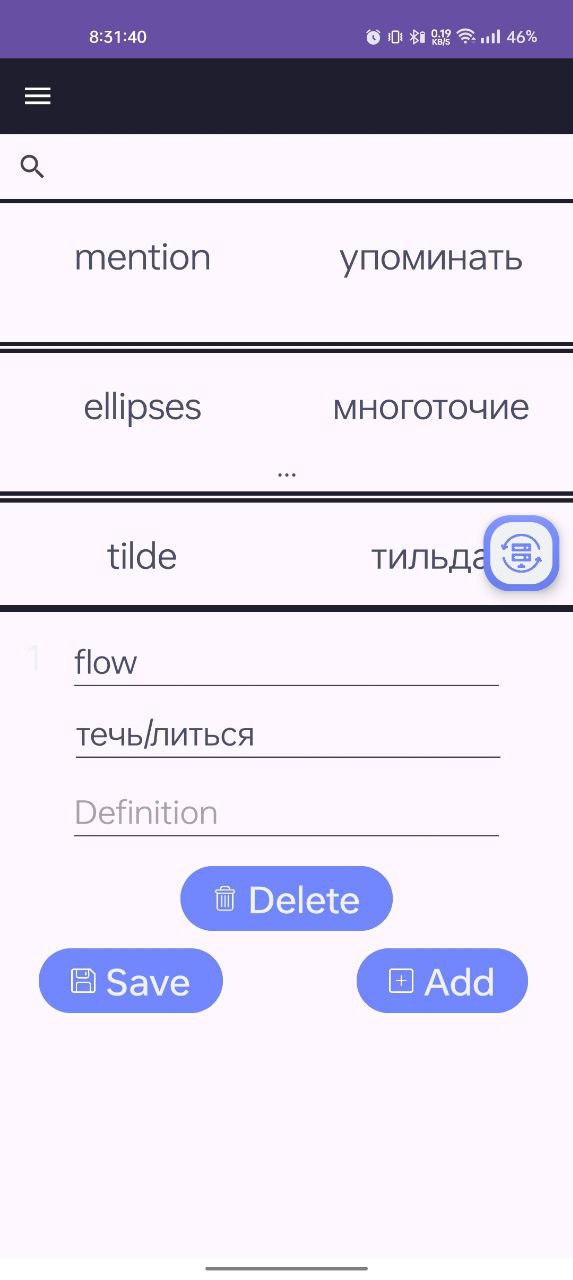


Рисунок 4.13 – Удаление записи «flow»

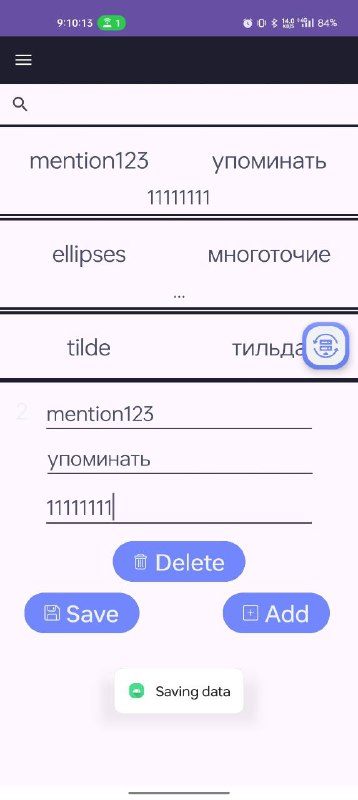


Рисунок 4.14 – Сохранение изменённых данных

Для перехода к тесту необходимо воспользоваться пунктами из выпадающего меню.

Первый тест - «Правда/Ложь», приветственная форма изображена на рисунке 4.15. 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, для ответа пользователю нужно выбрать пункт и нажать кнопку «Ответить», верный ответ становиться зеленым, изображено на рисунке 4.17. 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.

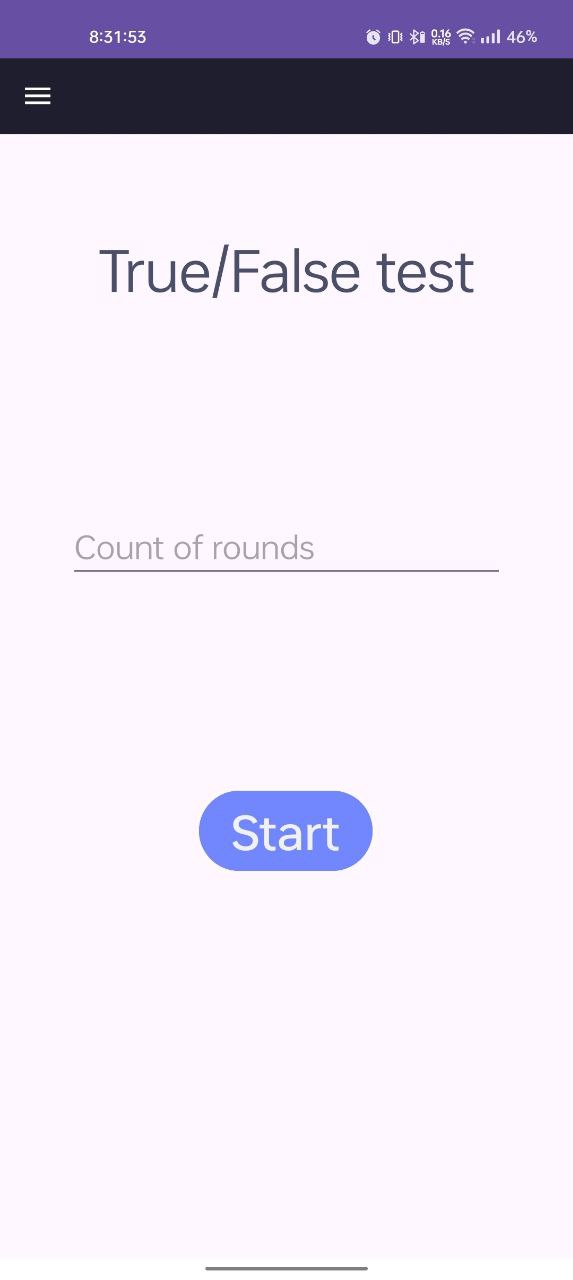


Рисунок 4.15 – Приветственная форма теста

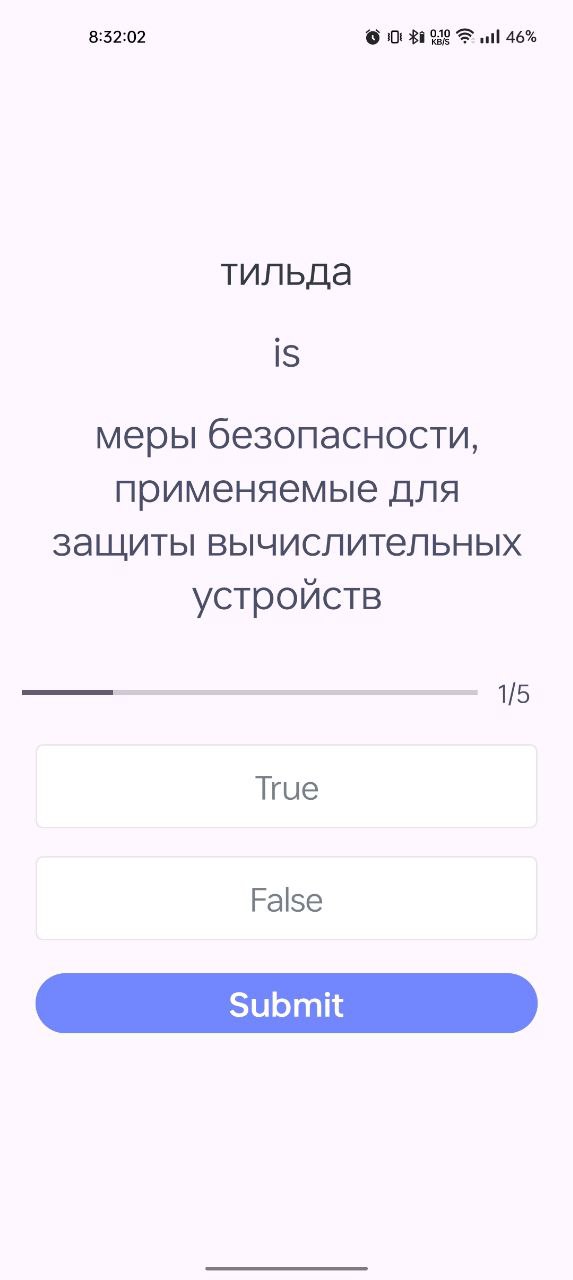


Рисунок 4.16 – Форма теста

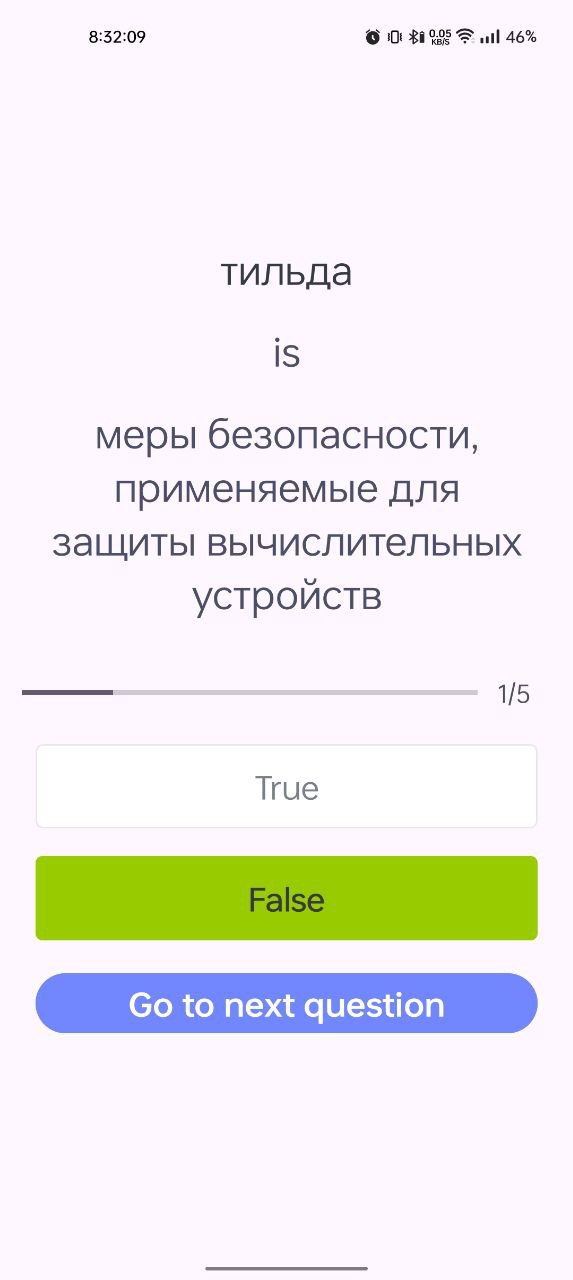


Рисунок 4.17 – Отображение верного ответа

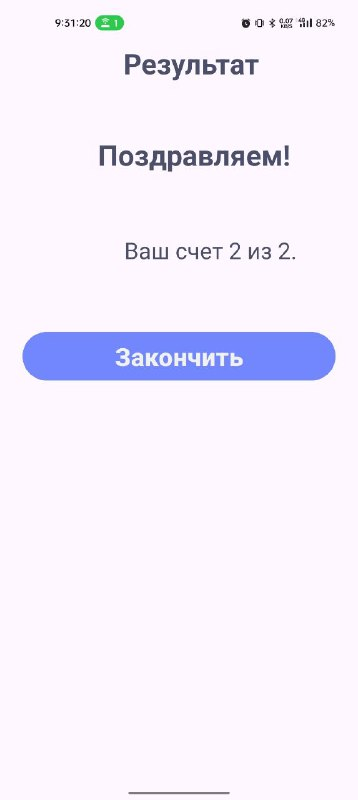


Рисунок 4.18 – Форма результата

Формы второго и третьего теста похожи на первый.

Четвертый тест «Hangman», представляет из себя игру в которой необходимо угадывать слово по буквам, 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. 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.

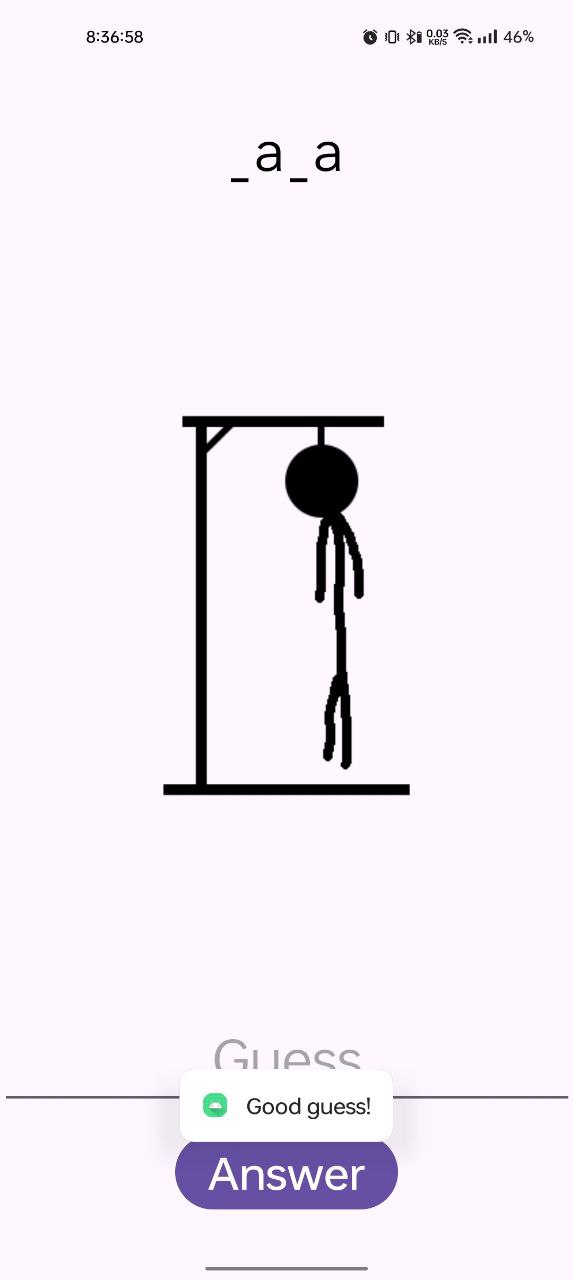


Рисунок 4.19 – Открытие букв в загаданном слове



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.

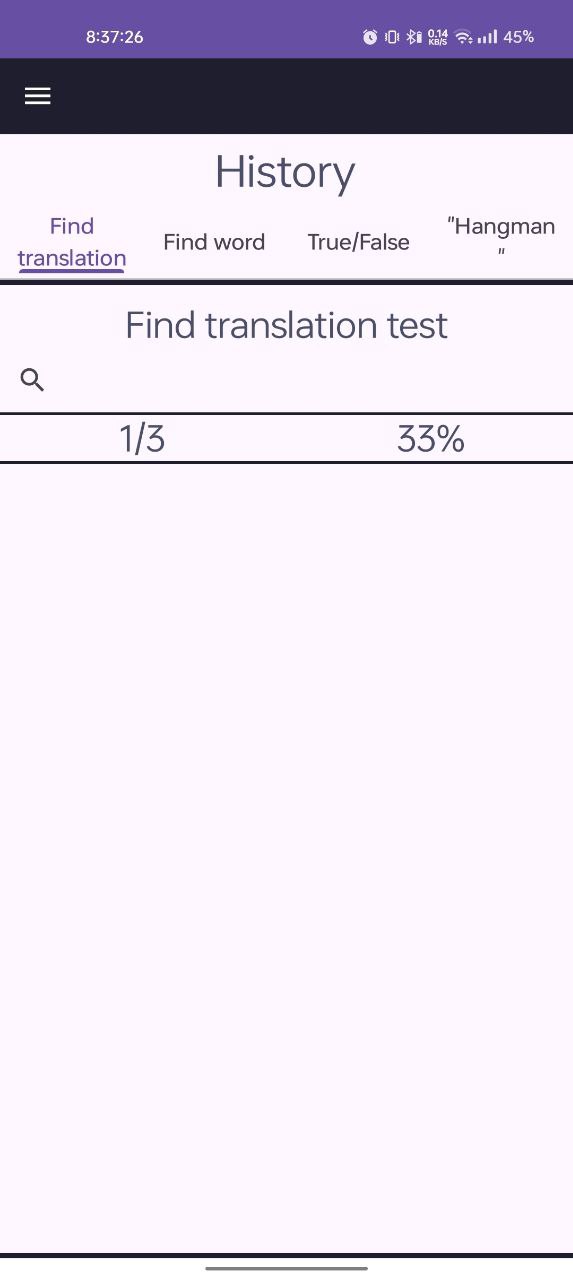


Рисунок 4.21 – История прохождения теста

# Заключение

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

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.

Для работы системы с данными была организована работа с базой данных «SQLite».

При проектировании программы были изучены теоретические сведения по теме «Символьный и строковый типы данных», были закреплены знания о символьных и строковых типах данных и организована работа с базой 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.

# Список использованных источников

1. Пьер-Ив Симон. Волшебство «Kotlin»: Учеб. Пособие / В. М. Стасышин. - Новосибирск: Изд-во ДМК Пресс, 2020 — 535 c.
2. Мэтт Вайсфельд, Объектно-ориентированное мышление / Мэтт Вайсфельд. - Москва: Питер, 2014. — 304 с.
3. Кадырова, Г. Р. Основы алгоритмизации и программирования: учебное пособие / Г. Р. Кадырова. – Ульяновск: УлГТУ, 2014. — 93 с.
4. Паттерны объектно-ориентированного проектирования. / Под ред. Ральф Джонсона. - СПб: Питер, 2001. — 368 с.
5. Практикум по объектно-ориентированному программированию / И. А. Бабушкина, С. М. Окулов. — 2-е изд. —М.: БИНОМ. Лаборатория знаний, 2009. — 366 с.
6. Почувствуй класс / Мейер Б.; пер. с англ. под ред. В.А. Биллига. —М.: Национальный Открытый Университет «ИНТУИТ»: БИНОМ. Лаборатория знаний, 2011 — 775 с.
7. ГОСТ 2.105-95 Общие требования общие требования к текстовым документам [Электронный ресурс]. - Электронные данные. – Режим доступа: – http://gostexpert.ru/gost/gost-2.105-95. - Дата доступа 20.01.2012.

# Приложение А

(обязательное)

**Текст программы (листинг) ИПС «Работа тату-салона»**

Текст модуля Unit1.pas

unit Unit1;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.ExtCtrls,

Vcl.Imaging.jpeg;

type

TForm1 = class(TForm)

Button1: TButton;

Button2: TButton;

Image1: TImage;

Panel1: TPanel;

Panel2: TPanel;

Panel3: TPanel;

procedure Button1Click(Sender: TObject);

procedure Button2Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form1: TForm1;

implementation

{$R \*.dfm}

uses Unit2;

procedure TForm1.Button1Click(Sender: TObject);

begin

form2.show;

form1.hide;

end;

procedure TForm1.Button2Click(Sender: TObject);

begin

form1.Close

end;

end.

Текст модуля Unit2.pas

unit Unit2;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.Imaging.jpeg, Vcl.ExtCtrls, DateUtils,

Vcl.Grids, Vcl.Menus, Vcl.StdCtrls, Vcl.ComCtrls, Vcl.Mask;

type

TForm2 = class(TForm)

Image1: TImage;

StringGrid1: TStringGrid;

MainMenu1: TMainMenu;

g1: TMenuItem;

f1: TMenuItem;

fd1: TMenuItem;

N1: TMenuItem;

Button1: TButton;

Button2: TButton;

Button3: TButton;

Edit1: TEdit;

N2: TMenuItem;

N4: TMenuItem;

SaveDialog1: TSaveDialog;

OpenDialog1: TOpenDialog;

N5: TMenuItem;

O1: TMenuItem;

N6: TMenuItem;

ComboBox1: TComboBox;

N3: TMenuItem;

N7: TMenuItem;

N8: TMenuItem;

N9: TMenuItem;

N10: TMenuItem;

N11: TMenuItem;

N12: TMenuItem;

N13: TMenuItem;

N14: TMenuItem;

N15: TMenuItem;

N16: TMenuItem;

N17: TMenuItem;

N18: TMenuItem;

N19: TMenuItem;

N20: TMenuItem;

N21: TMenuItem;

N22: TMenuItem;

N23: TMenuItem;

Button4: TButton;

ComboBox2: TComboBox;

Button5: TButton;

DateTimePicker1: TDateTimePicker;

ComboBox3: TComboBox;

Button6: TButton;

Edit2: TEdit;

N24: TMenuItem;

N25: TMenuItem;

procedure FormCreate(Sender: TObject);

procedure Button1Click(Sender: TObject);

procedure FormClose(Sender: TObject; var Action: TCloseAction);

procedure Button2Click(Sender: TObject);

procedure N4Click(Sender: TObject);

procedure Button3Click(Sender: TObject);

procedure f1Click(Sender: TObject);

procedure N6Click(Sender: TObject);

procedure N2Click(Sender: TObject);

procedure N5Click(Sender: TObject);

procedure O1Click(Sender: TObject);

procedure Edit1Change(Sender: TObject);

procedure Button4Click(Sender: TObject);

procedure Button5Click(Sender: TObject);

procedure ComboBox1KeyPress(Sender: TObject; var Key: Char);

procedure ComboBox3KeyPress(Sender: TObject; var Key: Char);

procedure Button6Click(Sender: TObject);

procedure N24Click(Sender: TObject);

procedure N25Click(Sender: TObject);

procedure N18Click(Sender: TObject);

procedure N19Click(Sender: TObject);

procedure N20Click(Sender: TObject);

procedure N21Click(Sender: TObject);

procedure N7Click(Sender: TObject);

procedure N8Click(Sender: TObject);

procedure N14Click(Sender: TObject);

procedure N15Click(Sender: TObject);

procedure N16Click(Sender: TObject);

procedure N17Click(Sender: TObject);

procedure N22Click(Sender: TObject);

procedure N1Click(Sender: TObject);

procedure Edit2KeyPress(Sender: TObject; var Key: Char);

procedure ComboBox1Change(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

type

zapis = record

fio\_master : string[20];

fio\_client : string[20];

number : string[20];

service : string[20];

cost : Double;

date : string[20];

time : string[20];

end;

type

TFakeGrid = class(TStringGrid);

var

Form2: TForm2;

f : file of zapis;

a : array[1..100] of zapis;

file\_name : string;

i, j, n : integer;

implementation

{$R \*.dfm}

uses Unit1, Unit3, Unit4, Unit5, Unit6, Unit7, Unit8;

procedure sortirovka(aSg : TStringGrid; const aCol : Integer; const vid : integer); //procedura sotrirovki

var

Grid : TFakeGrid;

begin

//vid = 1 ot a do ya

//vid = -1 ot ya do a

Grid := TFakeGrid(aSg);

with asg do begin

for i := 1 to RowCount - 1 do

for j := 1 to RowCount - 1 do begin

if vid = 1 then begin

if AnsiLowerCase(Cells[acol,i]) < AnsiLowerCase(Cells[acol,j]) then

Grid.MoveRow(i,j);

end

else

if AnsiLowerCase(Cells[acol,i]) > AnsiLowerCase(Cells[acol,j]) then

Grid.MoveRow(i,j);

end;

end;

end;

procedure TForm2.Button1Click(Sender: TObject); //dobavlenie zapisi

begin

Form3.ShowModal;

end;

procedure TForm2.Button2Click(Sender: TObject); //izmenenie zapisi

begin

if StringGrid1.Row = 0 then

ShowMessage('Эту строку нельзя изменять')

else

Form4.ShowModal;

end;

procedure TForm2.Button3Click(Sender: TObject); //delete

begin

with StringGrid1 do begin;

if Row = 0 then

ShowMessage('Эту строку нельзя удалить')

else begin

Rows[Row].Clear;

for i := Row to RowCount - 2 do

Rows[i] := Rows[i + 1];

RowCount := RowCount - 1;

//save

if file\_name <> '' then begin

AssignFile(f, file\_name);

Rewrite(f);

n := form2.Stringgrid1.RowCount - 1;

with form2.StringGrid1 do begin

for i := 1 to n do begin

a[i].fio\_master := Cells[0,i];

a[i].fio\_client := Cells[1,i];

a[i].number := Cells[2,i];

a[i].service := Cells[3,i];

a[i].cost := strtofloat(Cells[4,i]);

a[i].date := Cells[5,i];

a[i].time := Cells[6,i];

Write(f,a[i]);

end;

end;

CloseFile(f);

end

else

ShowMessage('Файл не удалось сохранить, так как нет файла для взаимодействия, попробуйте сохранить файл через "Сохранить как".');

//end save

end;

end;

end;

procedure TForm2.Button4Click(Sender: TObject); //delete proshloe

begin

if file\_name <> '' then begin

AssignFile(f, file\_name);

Reset(f);

n := FileSize(f);

j := 1;

with StringGrid1 do begin

RowCount := 1;

for i := 1 to n do begin

Read(f,a[i]);

if not(strtoint(copy(a[i].date, 7, 4)) < YearOf(now)) then begin

Cells[0,j] := a[i].fio\_master;

Cells[1,j] := a[i].fio\_client;

Cells[2,j] := a[i].number;

Cells[3,j] := a[i].service;

Cells[4,j] := floattostr(a[i].cost);

Cells[5,j] := a[i].date;

Cells[6,j] := a[i].time;

j := j + 1;

RowCount := RowCount + 1;

end;

end;

CloseFile(f);

end;

//save

AssignFile(f, file\_name);

Rewrite(f);

n := form2.Stringgrid1.RowCount - 1;

with form2.StringGrid1 do begin

for i := 1 to n do begin

a[i].fio\_master := Cells[0,i];

a[i].fio\_client := Cells[1,i];

a[i].number := Cells[2,i];

a[i].service := Cells[3,i];

a[i].cost := strtofloat(Cells[4,i]);

a[i].date := Cells[5,i];

a[i].time := Cells[6,i];

Write(f,a[i]);

end;

end;

CloseFile(f);

//end save

end

else

ShowMessage('Файл не выбран');

end;

procedure TForm2.Button5Click(Sender: TObject); //zapisi mastera

var date, master : string;

month, year : integer;

begin

if file\_name <> '' then begin

if ComboBox2.Text <> 'Мастер' then begin

date := FormatDateTime('mm.yyyy',DateTimePicker1.Date);

month := strtoint(copy(date,1,2));

year := strtoint(copy(date,4,4));

master := combobox2.Text;

AssignFile(f, file\_name);

Reset(f);

n := FileSize(f);

j := 1;

with StringGrid1 do begin

RowCount := 1;

for i := 1 to n do begin

Read(f,a[i]);

if a[i].fio\_master = master then

if strtoint(copy(a[i].date, 7, 4)) = year then

if strtoint(copy(a[i].date, 4, 2)) = month then begin

Cells[0,j] := a[i].fio\_master;

Cells[1,j] := a[i].fio\_client;

Cells[2,j] := a[i].number;

Cells[3,j] := a[i].service;

Cells[4,j] := floattostr(a[i].cost);

Cells[5,j] := a[i].date;

Cells[6,j] := a[i].time;

j := j + 1;

RowCount := RowCount + 1;

end;

end;

CloseFile(f);

end;

ComboBox2.Text := 'Мастер';

end

else

ShowMessage('Выберите мастера');

end

else

ShowMessage('Файл не выбран');

end;

procedure TForm2.Button6Click(Sender: TObject); //yvelichenie na procent

var procent : Double;

new\_cost, error : String;

begin

error := 'Не выбранны следующие данные: ';

if StringGrid1.RowCount <> 1 then begin

if Edit2.Text = '' then

error := error + 'не введен процент, ';

if ComboBox3.Text = 'Вид услуги' then

error := error + 'не выбран вид услуги, ';

if error = 'Не выбранны следующие данные: ' then begin

procent := strtofloat(Edit2.text);

procent := procent / 100;

with StringGrid1 do begin

for j := 1 to RowCount - 1 do

if Cells[3,j] = ComboBox3.Text then begin

new\_cost := floattostr(strtofloat(Cells[4,j]) + strtofloat(Cells[4,j]) \* procent);

if pos(',', new\_cost) <> 0 then

Cells[4,j] := copy(new\_cost, 1, pos(',', new\_cost) + 2)

else

Cells[4,j] := copy(new\_cost, 1, length(new\_cost));

end;

end;

//save

if file\_name <> '' then begin

AssignFile(f, file\_name);

Rewrite(f);

n := form2.Stringgrid1.RowCount - 1;

with form2.StringGrid1 do begin

for i := 1 to n do begin

a[i].fio\_master := Cells[0,i];

a[i].fio\_client := Cells[1,i];

a[i].number := Cells[2,i];

a[i].service := Cells[3,i];

a[i].cost := strtofloat(Cells[4,i]);

a[i].date := Cells[5,i];

a[i].time := Cells[6,i];

Write(f,a[i]);

end;

end;

CloseFile(f);

end

else

ShowMessage('Файл не удалось сохранить, так как нет файла для взаимодействия, попробуйте сохранить файл через "Сохранить как".');

//end save

Edit2.Clear;

ComboBox3.Text := 'Вид услуги';

end

else begin

delete(error, length(error) - 1, 2);

showmessage(error + '.');

end;

end

else

ShowMessage('Нет данных для изменения');

end;

procedure TForm2.ComboBox1Change(Sender: TObject);

begin

StringGrid1.Repaint;

Edit1.Text := '';

end;

procedure TForm2.ComboBox1KeyPress(Sender: TObject; var Key: Char); //zapret vvod v combobox

begin

if ComboBox1.Text <> '' then

key := #0;

end;

procedure TForm2.ComboBox3KeyPress(Sender: TObject; var Key: Char); //zapret vvod v combobox

begin

if ComboBox1.Text <> '' then

key := #0;

end;

procedure TForm2.Edit1Change(Sender: TObject); //POISK

var key : string;

Rect: TRect;

z : integer;

flag : boolean;

begin

key := Edit1.Text;

key := AnsiLowerCase(key);

if key <> '' then begin

if (ComboBox1.Text = 'По всем') and (ComboBox1.Text <> '') then begin //poisk po vsemy

with StringGrid1 do begin;

for j := 1 to RowCount -1 do begin

flag := false;

for i := 0 to ColCount - 1 do

if pos(key, AnsiLowerCase(Cells[i, j])) <> 0 then

flag := true;

if flag then

for z := 0 to ColCount-1 do begin

rect := CellRect(z,j);

Canvas.Brush.Color := ClScrollBar;

Canvas.fillRect(Rect);

Canvas.TextOut(Rect.left,Rect.Top,StringGrid1.Cells[z,j]);

end

else

for z := 0 to ColCount-1 do begin

rect := CellRect(z,j);

Canvas.Brush.Color := ClWhite;

Canvas.fillRect(Rect);

Canvas.TextOut(Rect.Left,Rect.top,StringGrid1.Cells[z,j]);

end;

end;

end;

end

//poisk po stolbcam

else begin

z := StrToInt(combobox1.Text) - 1;

with StringGrid1 do begin;

for j := 1 to RowCount - 1 do begin

if pos(key, AnsiLowerCase(Cells[z, j])) <> 0 then

for i := 0 to ColCount-1 do begin

rect := CellRect(i,j);

Canvas.Brush.Color := ClScrollBar;

Canvas.fillRect(Rect);

Canvas.TextOut(Rect.Left,Rect.Top,StringGrid1.Cells[i,j]);

end

else

for i := 0 to ColCount-1 do begin

rect := CellRect(i,j);

Canvas.Brush.Color:= ClWhite;

Canvas.fillRect(Rect);

Canvas.TextOut(Rect.Left,Rect.Top,StringGrid1.Cells[i,j]);

end;

end;

end;

end

end //yborka cveta

else

for i:= 1 to StringGrid1.RowCount-1 do

for j:=0 to StringGrid1.ColCount-1 do begin

StringGrid1.Canvas.Brush.Color:=clWhite;

rect := StringGrid1.CellRect(j,i);

StringGrid1.Canvas.FillRect(rect);

StringGrid1.Canvas.TextOut(rect.Left,rect.Top,stringgrid1.Cells[j,i]);

end;

end;

procedure TForm2.Edit2KeyPress(Sender: TObject; var Key: Char);

begin

if not(key in['0'..'9', ',', #08]) then

key := #0;

if (length(Edit2.Text) > 3) and (key <>#08) then

key := #0;

end;

procedure TForm2.f1Click(Sender: TObject); //open file

begin

if OpenDialog1.Execute then begin

file\_name := OpenDialog1.FileName;

AssignFile(f, file\_name);

Reset(f);

n := FileSize(f);

with StringGrid1 do begin

RowCount := n + 1;

for i := 1 to n do begin

Read(f,a[i]);

Cells[0,i] := a[i].fio\_master;

Cells[1,i] := a[i].fio\_client;

Cells[2,i] := a[i].number;

Cells[3,i] := a[i].service;

Cells[4,i] := floattostr(a[i].cost);

Cells[5,i] := a[i].date;

Cells[6,i] := a[i].time;

end;

end;

CloseFile(f);

end;

end;

procedure TForm2.FormClose(Sender: TObject; var Action: TCloseAction); //close form1 when form2 close

begin

Form1.Close;

end;

procedure TForm2.FormCreate(Sender: TObject); //sozdanie formi sg

begin

with StringGrid1 do begin

ColCount := 7;

ColWidths[0] := 160;

ColWidths[1] := 160;

ColWidths[2] := 120;

ColWidths[3] := 130;

ColWidths[4] := 130;

ColWidths[5] := 90;

ColWidths[6] := 60;

Cells[0,0] := 'ФИО мастера';

Cells[1,0] := 'ФИО клиента';

Cells[2,0] := 'Телефон клиента';

Cells[3,0] := 'Вид услуги';

Cells[4,0] := 'Стоимость, BYN';

Cells[5,0] := 'Дата записи';

Cells[6,0] := 'Время';

end;

end;

procedure TForm2.N14Click(Sender: TObject); //sort fio\_client a > ya

begin

sortirovka(StringGrid1, 1, 1);

end;

procedure TForm2.N15Click(Sender: TObject); //sort fio\_client ya > a

begin

sortirovka(StringGrid1, 1, -1);

end;

procedure TForm2.N16Click(Sender: TObject); //sort vid yslygi a > ya

begin

sortirovka(StringGrid1, 2, 1);

end;

procedure TForm2.N17Click(Sender: TObject); //sort vid yslygi ya > a

begin

sortirovka(StringGrid1, 2, -1);

end;

procedure TForm2.N18Click(Sender: TObject); //sort cost po ymensh

var

Grid : TFakeGrid;

begin

Grid := TFakeGrid(StringGrid1);

with StringGrid1 do begin

for i := 1 to RowCount - 1 do

for j := 1 to RowCount - 1 do

if strtofloat(Cells[4,i]) > strtofloat(Cells[4,j]) then begin

Grid.MoveRow(i,j);

end;

end;

end;

procedure TForm2.N19Click(Sender: TObject); //sort cost po vozrast

var

Grid : TFakeGrid;

begin

Grid := TFakeGrid(StringGrid1);

with StringGrid1 do begin

for i := 1 to RowCount - 1 do

for j := 1 to RowCount - 1 do

if strtofloat(Cells[4,i]) < strtofloat(Cells[4,j]) then begin

Grid.MoveRow(i,j);

end;

end;

end;

procedure TForm2.N1Click(Sender: TObject);

begin

Form8.ShowModal;

end;

procedure TForm2.N20Click(Sender: TObject); //sort date po ymensh

var

Grid : TFakeGrid;

begin

Grid := TFakeGrid(StringGrid1);

with StringGrid1 do begin

for i := 1 to RowCount - 1 do

for j := 1 to RowCount -1 do

if CompareDate(strtodate(stringgrid1.Cells[5,i]),strtodate(stringgrid1.Cells[5,j ])) = 1 then

Grid.MoveRow(i,j);

end;

end;

procedure TForm2.N21Click(Sender: TObject); //sort date po vozrast

var

Grid : TFakeGrid;

begin

Grid := TFakeGrid(StringGrid1);

with StringGrid1 do begin

for i := 1 to RowCount - 1 do

for j := 1 to RowCount -1 do

if CompareDate(strtodate(stringgrid1.Cells[5,i]),strtodate(stringgrid1.Cells[5,j ])) = -1 then

Grid.MoveRow(i,j);

end;

end;

procedure TForm2.N22Click(Sender: TObject); //sort time po ymensh

var

Grid : TFakeGrid;

hour\_i, min\_i, hour\_j, min\_j : integer;

begin

Grid := TFakeGrid(StringGrid1);

with Stringgrid1 do begin

for i := 1 to RowCount - 1 do begin

hour\_i := strtoint(copy(Cells[6,i], 1, 2));

min\_i := strtoint(copy(Cells[6,i], 4, 2));

for j := 1 to RowCount -1 do begin

hour\_j := strtoint(copy(Cells[6,j], 1, 2));

min\_j := strtoint(copy(Cells[6,j], 4, 2));

if (hour\_i < hour\_j) then

Grid.MoveRow(i,j)

else begin

if (hour\_i = hour\_j) then begin

if (min\_i < min\_j) then

Grid.MoveRow(i,j)

end;

end;

end;

end;

end;

end;

procedure TForm2.N24Click(Sender: TObject); //obnovit file

begin

if file\_name <> '' then begin

AssignFile(f, file\_name);

Reset(f);

n := FileSize(f);

with StringGrid1 do begin

RowCount := n + 1;

for i := 1 to n do begin

Read(f,a[i]);

Cells[0,i] := a[i].fio\_master;

Cells[1,i] := a[i].fio\_client;

Cells[2,i] := a[i].number;

Cells[3,i] := a[i].service;

Cells[4,i] := floattostr(a[i].cost);

Cells[5,i] := a[i].date;

Cells[6,i] := a[i].time;

end;

end;

CloseFile(f);

end

else

ShowMessage('Файл не выбран');

end;

procedure TForm2.N25Click(Sender: TObject); //o programme

begin

Form7.Showmodal;

end;

procedure TForm2.N2Click(Sender: TObject); //save open file

begin

if file\_name <>'' then begin

AssignFile(f, file\_name);

Rewrite(f);

n := Stringgrid1.RowCount - 1;

with StringGrid1 do begin

for i := 1 to n do begin

a[i].fio\_master := Cells[0,i];

a[i].fio\_client := Cells[1,i];

a[i].number := Cells[2,i];

a[i].service := Cells[3,i];

a[i].cost := strtofloat(Cells[4,i]);

a[i].date := Cells[5,i];

a[i].time := Cells[6,i];

Write(f,a[i]);

end;

end;

CloseFile(f);

end

else

ShowMessage('Файл не удалось сохранить, так как нет файла для взаимодействия, попробуйте сохранить файл через "Сохранить как".');

end;

procedure TForm2.N4Click(Sender: TObject); //vihod

begin

Form1.Close;

end;

procedure TForm2.N5Click(Sender: TObject); //open sozdanie file

begin

Form5.ShowModal;

end;

procedure TForm2.N6Click(Sender: TObject); //save as

begin

if SaveDialog1.Execute then begin

assignfile(f, SaveDialog1.FileName + '.dat');

rewrite(f);

n := Stringgrid1.RowCount - 1;

with StringGrid1 do begin

for i := 1 to n do begin

a[i].fio\_master := cells[0,i];

a[i].fio\_client := cells[1,i];

a[i].number := cells[2,i];

a[i].service := cells[3,i];

a[i].cost := strtofloat(cells[4,i]);

a[i].date := cells[5,i];

a[i].time := cells[6,i];

write(f,a[i]);

end;

end;

closefile(f);

end;

end;

procedure TForm2.N7Click(Sender: TObject); //sort fio\_master a > ya

begin

sortirovka(StringGrid1, 0, 1);

end;

procedure TForm2.N8Click(Sender: TObject); //sort fio\_master ya > a

begin

sortirovka(StringGrid1, 0, -1);

end;

procedure TForm2.O1Click(Sender: TObject); //open syshestv file

begin

form6.showmodal;

end;

end.

Текст модуля Unit3.pas

unit Unit3;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.Mask, Vcl.ComCtrls;

type

TForm3 = class(TForm)

Label1: TLabel;

Label2: TLabel;

Label3: TLabel;

Edit2: TEdit;

Label4: TLabel;

MaskEdit1: TMaskEdit;

ComboBox1: TComboBox;

Label6: TLabel;

Label7: TLabel;

Label8: TLabel;

Label9: TLabel;

Button1: TButton;

DateTimePicker1: TDateTimePicker;

Edit3: TEdit;

ComboBox2: TComboBox;

MaskEdit2: TMaskEdit;

Button2: TButton;

procedure Button1Click(Sender: TObject);

procedure Edit2KeyPress(Sender: TObject; var Key: Char);

procedure Edit3KeyPress(Sender: TObject; var Key: Char);

procedure Button2Click(Sender: TObject);

procedure ComboBox2KeyPress(Sender: TObject; var Key: Char);

procedure ComboBox1KeyPress(Sender: TObject; var Key: Char);

procedure Edit3Change(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure FormClose(Sender: TObject; var Action: TCloseAction);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form3: TForm3;

const Rus: String = 'АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯабвгдеёжзийклмнопрстуфхцчшщъыьэюя';

implementation

{$R \*.dfm}

uses Unit2;

procedure TForm3.Button1Click(Sender: TObject); //dobavlenie zapisi

var n, hour, minute, i :integer;

error : string;

f : file of zapis;

begin

error := 'Данные введены не правильно : '; //PROVERKA

if Combobox2.Text = 'Мастер' then

error := error + 'не выбран мастер, ';

if Edit2.text = '' then

error := error + 'не введено ФИО клиента, ';

if pos(' ', MaskEdit1.Text) <> 0 then

error := error + 'не введен номер телефона, ';

if Combobox1.Text = 'Вид услуги' then

error := error + 'не выбран вид услуги, ';

if Edit3.text = '' then

error := error + 'не введена стоимость, '

else

if Edit3.Text[1] = ',' then

error := error + 'неверно введена стоимость, ';

if pos(' ', MaskEdit2.Text) <> 0 then

error := error + 'не введено время, '

else

if (strtoint(copy(MaskEdit2.Text, 1, pos(':', MaskEdit2.Text)- 1)) > 23) or

(strtoint(copy(MaskEdit2.Text, pos(':', MaskEdit2.Text) + 1, 2)) > 59) then

error := error + 'введено не коректное время, ';

if error = 'Данные введены не правильно : ' then begin

with form2.StringGrid1 do begin

rowcount:= rowcount + 1;

n := rowcount - 1;

cells[0,n] := combobox2.text; //fio mastera

cells[1,n] := edit2.text; //fio clienta

cells[2,n] := maskedit1.text;

//telephone

cells[3,n] := combobox1.text; //vid yslygi

cells[4,n] := edit3.text; //stoimostb

cells[5,n] := FormatDateTime('dd.mm.yyyy',DateTimePicker1.Date); //data

cells[6,n] := maskedit2.Text; //vremia

end;

//save

if file\_name <> '' then begin

AssignFile(f, file\_name);

Rewrite(f);

n := form2.Stringgrid1.RowCount - 1;

with form2.StringGrid1 do begin

for i := 1 to n do begin

a[i].fio\_master := Cells[0,i];

a[i].fio\_client := Cells[1,i];

a[i].number := Cells[2,i];

a[i].service := Cells[3,i];

a[i].cost := strtofloat(Cells[4,i]);

a[i].date := Cells[5,i];

a[i].time := Cells[6,i];

Write(f,a[i]);

end;

end;

CloseFile(f);

end

else

ShowMessage('Файл не удалось сохранить, так как нет файла для взаимодействия, попробуйте сохранить файл через "Сохранить как".');

//end save

ComboBox2.Text := 'Мастер';

Edit2.Clear;

MaskEdit1.Clear;

ComboBox1.Text := 'Вид услуги';

Edit3.Clear;

MaskEdit2.Clear;

end

else begin

delete(error, length(error) - 1, 2);

showmessage(error + '.');

end;

end;

procedure TForm3.Button2Click(Sender: TObject); //otmena

begin

ComboBox2.Text := 'Мастер';

Edit2.Clear;

MaskEdit1.Clear;

ComboBox1.Text := 'Вид услуги';

Edit3.Clear;

MaskEdit2.Clear;

Form3.Close;

end;

procedure TForm3.ComboBox1KeyPress(Sender: TObject; var Key: Char); //zapret vvod v combobox

begin

if ComboBox1.Text <> '' then

key := #0;

end;

procedure TForm3.ComboBox2KeyPress(Sender: TObject; var Key: Char); //zapret vvod v combobox

begin

if ComboBox1.Text <> '' then

key := #0;

end;

procedure TForm3.Edit2KeyPress(Sender: TObject; var Key: Char); //block for fio

begin

if (pos(key,rus) = 0) and (not(key in['-', #08, '.', #32])) then

key := #0;

if (length(Edit2.Text) > 19) and (key <>#08) then

key := #0;

end;

procedure TForm3.Edit3Change(Sender: TObject);

var str : string;

begin

str := edit3.Text;

delete(str, 1, pos(',', str));

if pos(',', str) <> 0 then

Edit3.Text := '';

end;

procedure TForm3.Edit3KeyPress(Sender: TObject; var Key: Char); //block for cost

begin

if not(key in['0'..'9', ',', #08]) then

key := #0;

if (length(Edit3.Text) > 6) and (key <>#08) then

key := #0;

end;

procedure TForm3.FormClose(Sender: TObject; var Action: TCloseAction);

begin

ComboBox2.Text := 'Мастер';

Edit2.Clear;

MaskEdit1.Clear;

ComboBox1.Text := 'Вид услуги';

Edit3.Clear;

MaskEdit2.Clear;

Form3.Close;

end;

procedure TForm3.FormCreate(Sender: TObject);

begin

ComboBox1.Text := 'Вид услуги';

end;

end.

Текст модуля Unit4.pas

unit Unit4;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.ComCtrls, Vcl.Mask;

type

TForm4 = class(TForm)

Label2: TLabel;

Label1: TLabel;

Label3: TLabel;

Label5: TLabel;

Label6: TLabel;

Label7: TLabel;

Label8: TLabel;

Label9: TLabel;

Button1: TButton;

Button2: TButton;

ComboBox1: TComboBox;

Edit1: TEdit;

MaskEdit1: TMaskEdit;

ComboBox2: TComboBox;

Edit2: TEdit;

DateTimePicker1: TDateTimePicker;

MaskEdit2: TMaskEdit;

procedure FormShow(Sender: TObject);

procedure Button1Click(Sender: TObject);

procedure Button2Click(Sender: TObject);

procedure Edit1KeyPress(Sender: TObject; var Key: Char);

procedure Edit2KeyPress(Sender: TObject; var Key: Char);

procedure ComboBox1KeyPress(Sender: TObject; var Key: Char);

procedure ComboBox2KeyPress(Sender: TObject; var Key: Char);

procedure Edit2Change(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form4: TForm4;

implementation

{$R \*.dfm}

uses Unit2, Unit3;

procedure TForm4.Button1Click(Sender: TObject);

var error : string;

f : file of zapis;

i :integer;

begin

error := 'Данные введены не правильно : '; //PROVERKA

if Edit1.text = '' then

error := error + 'не введено ФИО клиента, ';

if pos(' ', MaskEdit1.Text) <> 0 then

error := error + 'не введен номер телефона, ';

if Edit2.text = '' then

error := error + 'не введена стоимость, '

else

if Edit2.Text[1] = ',' then

error := error + 'неверно введена стоимость, ';

if pos(' ', MaskEdit2.Text) <> 0 then

error := error + 'не введено время, '

else

if (strtoint(copy(MaskEdit2.Text, 1, pos(':', MaskEdit2.Text)- 1)) > 23) or (strtoint(copy(MaskEdit2.Text, pos(':', MaskEdit2.Text) + 1, 2)) > 59) then

error := error + 'введено не коректное время, ';

if error = 'Данные введены не правильно : ' then begin

with Form2.StringGrid1 do begin

i := Row;

Cells[0,i] := Combobox1.text; //fio mstera

Cells[1,i] := Edit1.text; //fio clienta

Cells[2,i] := MaskEdit1.text; //telephone

Cells[3,i] := Combobox2.text; //vid yslygi

Cells[4,i] := Edit2.text; //stoimostb

Cells[5,i] := FormatDateTime('dd.mm.yyyy',DateTimePicker1.Date); //data

Cells[6,i] := MaskEdit2.Text; //time

end;

//save

if file\_name <> '' then begin

AssignFile(f, file\_name);

Rewrite(f);

n := form2.Stringgrid1.RowCount - 1;

with form2.StringGrid1 do begin

for i := 1 to n do begin

a[i].fio\_master := Cells[0,i];

a[i].fio\_client := Cells[1,i];

a[i].number := Cells[2,i];

a[i].service := Cells[3,i];

a[i].cost := strtofloat(Cells[4,i]);

a[i].date := Cells[5,i];

a[i].time := Cells[6,i];

Write(f,a[i]);

end;

end;

CloseFile(f);

end

else

ShowMessage('Файл не удалось сохранить, так как нет файла для взаимодействия, попробуйте сохранить файл через "Сохранить как".');

//end save

Form4.Close;

end

else begin

delete(error, length(error) - 1, 2);

showmessage(error + '.');

end;

end;

procedure TForm4.Button2Click(Sender: TObject);

begin

Form4.Close;

end;

procedure TForm4.ComboBox1KeyPress(Sender: TObject; var Key: Char);

begin

if ComboBox1.Text <> '' then

key := #0;

end;

procedure TForm4.ComboBox2KeyPress(Sender: TObject; var Key: Char);

begin

if ComboBox1.Text <> '' then

key := #0;

end;

procedure TForm4.Edit1KeyPress(Sender: TObject; var Key: Char); //block for fio

begin

if (pos(key, rus) = 0) and (not(key in['-', #08, '.', #32])) then

key := #0;

if (length(Edit1.Text) > 19) and (key <>#08) then

key := #0;

end;

procedure TForm4.Edit2Change(Sender: TObject);

var str : string;

begin

str := edit2.Text;

delete(str, 1, pos(',', str));

if pos(',', str) <> 0 then

Edit2.Text := '';

end;

procedure TForm4.Edit2KeyPress(Sender: TObject; var Key: Char); //block for cost

begin

if not(key in['0'..'9', ',', #08]) then

key := #0;

if (length(Edit2.Text) > 6) and (key <>#08) then

key := #0;

end;

procedure TForm4.FormShow(Sender: TObject); //dobav infi for izmenenii

begin

with Form2.StrinGgrid1 do begin

i := Row;

Combobox1.Text := Cells[0, i];

Edit1.Text := Cells[1, i];

MaskEdit1.Text := Cells[2, i];

Combobox2.Text := Cells[3, i];

Edit2.Text := Cells[4, i];

DateTimePicker1.Date := StrToDate(Cells[5, i]);

MaskEdit2.Text := Cells[6, i];

end;

end;

end.

Текст модуля Unit5.pas

unit Unit5;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls;

type

TForm5 = class(TForm)

Edit1: TEdit;

Button1: TButton;

Button2: TButton;

procedure Button1Click(Sender: TObject);

procedure Button2Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form5: TForm5;

implementation

{$R \*.dfm}

uses Unit2, Unit4;

procedure TForm5.Button1Click(Sender: TObject);

begin

if not (edit1.text = '') then begin

unit2.file\_name := edit1.text + '.dat';

assignfile(unit2.f, unit2.file\_name);

Rewrite(unit2.f);

closefile(unit2.f);

with form2.stringgrid1 do begin

rowcount := 1;

cells[0,0] := 'ФИО мастера';

cells[1,0] := 'ФИО клиента';

cells[2,0] := 'Номер телефона клиента';

cells[3,0] := 'Вид услуги';

cells[4,0] := 'Стоимость';

cells[5,0] := 'Дата записи';

cells[6,0] := 'Время';

end;

form5.Close;

end

else

showmessage('Введите название файла');

end;

procedure TForm5.Button2Click(Sender: TObject);

begin

form5.Close;

end;

end.

Текст модуля Unit6.pas

unit Unit6;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls;

type

TForm6 = class(TForm)

ComboBox1: TComboBox;

Button1: TButton;

Button2: TButton;

procedure Button2Click(Sender: TObject);

procedure Button1Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form6: TForm6;

implementation

{$R \*.dfm}

uses Unit2;

procedure TForm6.Button1Click(Sender: TObject);

var f : file of unit2.zapis;

i : integer;

begin

if FileExists(combobox1.Text + '.dat') then begin

file\_name := Combobox1.Text + '.dat';

AssignFile(f, Combobox1.Text + '.dat');

Reset(f);

n := FileSize(f);

with form2.StringGrid1 do begin

RowCount := n + 1;

for i := 1 to n do begin

Read(f,a[i]);

Cells[0,i] := a[i].fio\_master;

Cells[1,i] := a[i].fio\_client;

Cells[2,i] := a[i].number;

Cells[3,i] := a[i].service;

Cells[4,i] := floattostr(a[i].cost);

Cells[5,i] := a[i].date;

Cells[6,i] := a[i].time;

end;

end;

CloseFile(f);

end

else

ShowMessage('Такого файла не существует.');

end;

procedure TForm6.Button2Click(Sender: TObject);

begin

form6.Close;

end;

end.

Текст модуля Unit7.pas

unit Unit7;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls;

type

TForm7 = class(TForm)

Label1: TLabel;

Label2: TLabel;

Button1: TButton;

Label3: TLabel;

procedure Button1Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form7: TForm7;

implementation

{$R \*.dfm}

procedure TForm7.Button1Click(Sender: TObject);

begin

Form7.Close;

end;

end.

Текст модуля Unit8.pas

unit Unit8;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.ComCtrls, DateUtils;

type

TForm8 = class(TForm)

ComboBox1: TComboBox;

Label1: TLabel;

Label2: TLabel;

ComboBox2: TComboBox;

Button1: TButton;

Button2: TButton;

Edit1: TEdit;

Edit2: TEdit;

Label3: TLabel;

Label4: TLabel;

CheckBox1: TCheckBox;

DateTimePicker1: TDateTimePicker;

DateTimePicker2: TDateTimePicker;

procedure ComboBox1Change(Sender: TObject);

procedure CheckBox1Click(Sender: TObject);

procedure Button1Click(Sender: TObject);

procedure Edit1KeyPress(Sender: TObject; var Key: Char);

procedure Edit2KeyPress(Sender: TObject; var Key: Char);

procedure Edit2Change(Sender: TObject);

procedure Edit1Change(Sender: TObject);

procedure ComboBox1KeyPress(Sender: TObject; var Key: Char);

procedure FormShow(Sender: TObject);

procedure Button2Click(Sender: TObject);

procedure FormClose(Sender: TObject; var Action: TCloseAction);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form8: TForm8;

index : integer;

implementation

{$R \*.dfm}

uses Unit2;

procedure TForm8.Button1Click(Sender: TObject);

var j, z : integer;

error, date, ot\_date, do\_date : string;

flag : boolean;

cost, ot\_cost , do\_cost : double;

begin

error := '';

if form2.StringGrid1.RowCount <> 1 then begin

if ComboBox1.Text = 'Данные для фильтрации' then

error := error + 'Не выбраны данные для фильтрации.'

else

if (ComboBox2.Text = 'Фильтр') and (CheckBox1.Checked = False) then

error := error + 'Не выбран фильтр.';

if error = '' then begin

with form2.StringGrid1 do begin

j := 1;

repeat

if CheckBox1.Checked = False then begin

if AnsiLowerCase(Cells[index,j]) <> AnsiLowerCase(ComboBox2.Text) then begin

Rows[j].Clear;

for z := j to RowCount - 2 do

Rows[z] := Rows[z + 1];

RowCount := RowCount - 1;

j := j - 1;

end;

end

else begin

if index = 4 then begin

cost := strtofloat(Cells[index,j]);

ot\_cost := strtofloat(Edit1.text);

do\_cost := strtofloat(Edit2.text);

if (cost < ot\_cost) or (cost >= do\_cost) then begin

Rows[j].Clear;

for z := j to RowCount - 2 do

Rows[z] := Rows[z + 1];

RowCount := RowCount - 1;

j := j - 1;

end;

end;

if index = 5 then begin

date := Cells[index,j];

ot\_date := FormatDateTime('dd.mm.yyyy',DateTimePicker1.Date);

do\_date := FormatDateTime('dd.mm.yyyy',DateTimePicker2.Date);

if (CompareDate(strtodate(date),strtodate(ot\_date)) = -1) or (CompareDate(strtodate(date),strtodate(do\_date)) = 1) then begin

Rows[j].Clear;

for z := j to RowCount - 2 do

Rows[z] := Rows[z + 1];

RowCount := RowCount - 1;

j := j - 1;

end;

end;

end;

inc(j);

until j = RowCount ;

end;

ComboBox1.Text := 'Данные для фильтрации';

ComboBox2.Clear;

ComboBox2.Text := 'Фильтр';

Edit1.Clear;

Edit2.Clear

end

else begin

showmessage(error);

end;

end

else

ShowMessage('Нет данных для изменения');

end;

procedure TForm8.Button2Click(Sender: TObject);

begin

Form8.Close;

end;

procedure TForm8.CheckBox1Click(Sender: TObject);

begin

if CheckBox1.Checked then begin

ComboBox2.Enabled := False;

Label3.Visible := true;

Label4.Visible := true;

if ComboBox1.ItemIndex = 4 then begin

Edit1.Visible := true;

Edit2.Visible := true;

end;

if ComboBox1.ItemIndex = 5 then begin

DateTimePicker1.Visible := true;

DateTimePicker2.Visible := true;

end;

end

else begin

ComboBox2.Enabled := True;

Edit1.Visible := false;

Edit2.Visible := false;

Label3.Visible := false;

Label4.Visible := false;

DateTimePicker1.Visible := false;

DateTimePicker2.Visible := false;

end;

end;

procedure TForm8.ComboBox1Change(Sender: TObject);

var j, z : integer;

flag : boolean;

begin

Combobox2.Text := 'Фильтр';

index := ComboBox1.ItemIndex;

ComboBox2.Items.Clear;

CheckBox1.Checked := False;

if (index = 4) or (index = 5) then

CheckBox1.Visible := true

else

begin

CheckBox1.Checked := false;

CheckBox1.Visible := false;

Edit1.Visible := false;

Edit2.Visible := false;

Label3.Visible := false;

Label4.Visible := false;

end;

with Form2.StringGrid1 do begin

for j := 1 to RowCount - 1 do begin

flag := true;

for z := 0 to ComboBox2.Items.Count - 1 do

if Cells[index,j] = ComboBox2.Items[z] then

flag := false;

if flag then

ComboBox2.Items.Add(Cells[index,j]);

end;

end;

end;

procedure TForm8.ComboBox1KeyPress(Sender: TObject; var Key: Char);

begin

if ComboBox1.Text <> '' then

key := #0;

end;

procedure TForm8.Edit1Change(Sender: TObject);

var str : string;

begin

str := edit1.Text;

delete(str, 1, pos(',', str));

if pos(',', str) <> 0 then

Edit1.Text := '';

end;

procedure TForm8.Edit1KeyPress(Sender: TObject; var Key: Char);

begin

if not(key in['0'..'9', ',', #08]) then

key := #0;

if (length(Edit1.Text) > 6) and (key <>#08) then

key := #0;

end;

procedure TForm8.Edit2Change(Sender: TObject);

var str : string;

begin

str := edit2.Text;

delete(str, 1, pos(',', str));

if pos(',', str) <> 0 then

Edit2.Text := '';

end;

procedure TForm8.Edit2KeyPress(Sender: TObject; var Key: Char);

begin

if not(key in['0'..'9', ',', #08]) then

key := #0;

if (length(Edit2.Text) > 6) and (key <>#08) then

key := #0;

end;

procedure TForm8.FormClose(Sender: TObject; var Action: TCloseAction);

begin

ComboBox1.Text := 'Данные для фильтрации';

ComboBox2.Clear;

ComboBox2.Text := 'Фильтр';

Edit1.Clear;

Edit2.Clear

end;

procedure TForm8.FormShow(Sender: TObject);

begin

ComboBox1.Text := 'Данные для фильтрации';

index := -1;

CheckBox1.Checked := false;

CheckBox1.Visible := false;

Edit1.Visible := false;

Edit2.Visible := false;

Label3.Visible := false;

Label4.Visible := false;

end;

end.