МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ

КАЛИНИНГРАДСКОЙ ОБЛАСТИ

ПРОФЕССИОНАЛЬНАЯ ОБРАЗОВАТЕЛЬНАЯ ОРГАНИЗАЦИЯ

«КОЛЛЕДЖ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И СТРОИТЕЛЬСТВА»

(ГБУ КО ПОО «КИТиС»)

|  |  |  |
| --- | --- | --- |
| Курсовой проект  допущен к защите  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (дата)  Зам.директора по УМР  \_\_\_\_\_\_\_\_\_\_\_\_\_Павленко Г.Я.  (подпись) |  | Курсовой проект  защищен с оценкой  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (оценка)  Руководитель работы  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) |

Пояснительная записка к курсовому проекту

по дисциплине: МДК 01.01 Разработка программных модулей

Тема: Разработка приложения по изучению английского языка.

Специальность 09.02.07 «Информационные системы и программирование»

|  |  |
| --- | --- |
| Выполнила: | студентка 3 курса,  группы ИСп 22-1  Глазкова Полина Игоревна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |
| Руководитель: | Большакова-Стрекалова Анна Викторовна, преподаватель ГБУ КО ПОО КИТиС  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

Калининград

2025

Аннотация

Тема курсового проекта заключается в создании приложения для изучения английского языка «ITEnglishApplication», ориентированного на расширение словарного запаса посредством интерактивных тестов на угадывание перевода слов, с использованием среды разработки Android Studio. Структура отчёта включает введение, три главы, заключение, список использованных источников и листинг программ, упомянутых при описании программного продукта.

Во введении обосновывается актуальность предметной области и представляются общие идеи проекта. Первая глава содержит описание предметной области, принцип работы программы, анализ аналогичных продуктов и техническое задание на разработку программного продукта.

Во второй главе описывается процесс разработки программы, включая использованные средства и технологии, интерфейс приложения, логику работы программы и схему взаимодействия компонентов проекта.

Третья глава посвящена установке и началу работы с программой, проведению тестов в процессе работы, итоговому тестированию продукта и анализу результатов работы программы.

В заключении формулируются основные выводы, вытекающие из проведённого.

Ключевые слова: Kotlin, приложение, разработка.

Приложение: «ITEnglishApplication».

Общий объём работы 47 страниц.

Ссылка на GitHub репозиторий: <https://github.com/tkazy/cursach>

Содержание

[Введение 4](#_Toc196730007)

[1 Описание предметной области 6](#_Toc196730008)

[1.1 Аналоги разрабатываемого приложения 6](#_Toc196730009)

[1.2 Техническое задание 10](#_Toc196730010)

[1.3 Описание структуры приложения 12](#_Toc196730011)

[2 Описание разработки приложения 14](#_Toc196730012)

[2.1 Обоснование средств разработки 14](#_Toc196730013)

[2.2 Разработка интерфейса 17](#_Toc196730014)

[2.3. Разработка логики работы приложения. Схема взаимодействия компонентов проекта 21](#_Toc196730015)

[2.3.1 Логика работы приложения 21](#_Toc196730016)

[2.3.2 Схема взаимодействия компонентов 23](#_Toc196730017)

[2.4 Описание переменных, компонентов, классов и подпрограмм 24](#_Toc196730018)

[3 Тестирование и установка программы 26](#_Toc196730019)

[Заключение 32](#_Toc196730020)

[Список использованной литературы 33](#_Toc196730021)

[Приложение А 34](#_Toc196730022)

[Приложение Б 43](#_Toc196730023)

# Введение

В современном мире знание английского языка является не просто преимуществом, а зачастую необходимостью. Глобализация, развитие международных коммуникаций и доступность информации на английском языке делают владение им ключевым фактором успеха в различных сферах жизни. Стремление к изучению английского языка растёт, а вместе с ним и потребность в эффективных и доступных инструментах обучения.

В связи с этим, разработка мобильных приложений для изучения иностранных языков становится все более актуальной. Мобильные приложения обладают рядом преимуществ, таких как доступность в любое время и в любом месте, интерактивность и возможность персонализации обучения. Они позволяют вовлечь пользователей в процесс обучения в интересной и увлекательной форме, что способствует лучшему усвоению материала.

Целью данного курсового проекта является разработка мобильного приложения для изучения английского языка, ориентированного на расширение словарного запаса посредством интерактивных тестов. Основной функционал приложения заключается в предоставлении пользователю возможности проверить свои знания английских слов, угадывая их перевод. Такой подход позволяет не только проверить уже имеющиеся знания, но и активно запоминать новые слова в контексте, что способствует более эффективному и долговременному усвоению материала. Пополнять свой словарный запас английского языка в удобном и доступном формате.

В приложении должен быть удобный интерфейс, включающий создание профиля (Вход/Регистрация), главную страницу и сами тесты.

В ходе работы над проектом используются современные технологии разработки мобильных приложений: язык программирования Kotlin, SQLite (для хранения данных).

В рамках данного проекта будут рассмотрены следующие задачи:

1. Проектирование функциональности мобильного приложения.
2. Разработка пользовательского интерфейса приложения.
3. Создание базы данных английских слов и их переводов.
4. Проведение анализа существующих приложений и методик.
5. Проведение тестирования и отладки приложения.

В заключение, разработка данного мобильного приложения представляет собой перспективное направление, способное внести вклад в повышение уровня владения английским языком у широкой аудитории.

# 1 Описание предметной области

## 1.1 Аналоги разрабатываемого приложения

Duolingo - популярное приложение для изучения языков в формате игры, предлагающее интерактивные уроки по грамматике, словарному запасу, чтению, письму и аудированию. Обучение построено на принципе «байт-сайз» уроков и геймификации.

Преимущества:

* Бесплатность: Основной функционал приложения доступен бесплатно.
* Геймификация: Обучение представлено в игровой форме, что повышает мотивацию и вовлеченность.
* Комплексный подход: Охватывает различные аспекты изучения языка (грамматика, лексика, произношение).
* Интерактивность: Уроки включают в себя разнообразные задания (перевод, аудирование, произношение).

Недостатки:

* Ограниченная глубина: Не всегда подходит для углублённого изучения сложных грамматических конструкций.
* Зависимость от структуры уроков: Обучение строго следует определённой программе, что может быть не всегда удобно для пользователей с разными потребностями.
* Реклама (в бесплатной версии): Наличие рекламы может отвлекать от процесса обучения.



Рисунок 1 – Логотип Duolingo



Рисунок 2 – Интерфейс Duolingo

Memrise - приложение, ориентированное на запоминание слов и фраз с помощью мнемонических приёмов и повторения с интервалами. Пользователи могут создавать собственные курсы и делиться ими с другими.

Преимущества:

* Эффективное запоминание: Использование мнемоники и повторения с интервалами способствует лучшему усвоению информации.
* Большое количество курсов: Огромная база данных курсов, созданных пользователями, по различным темам и уровням.
* Сообщество: Возможность общаться с другими пользователями и делиться опытом.
* Интерактивные видео: Приложение использует видеоролики с носителями языка для лучшего понимания контекста.

Недостатки:

* Ограниченная грамматика: Основной упор делается на запоминание слов и фраз, грамматика освещается в меньшей степени.
* Качество курсов (от пользователей): Качество курсов, созданных пользователями, может варьироваться.
* Платный контент: Некоторые курсы и функции доступны только по подписке.



Рисунок 3 – Логотип Memrise

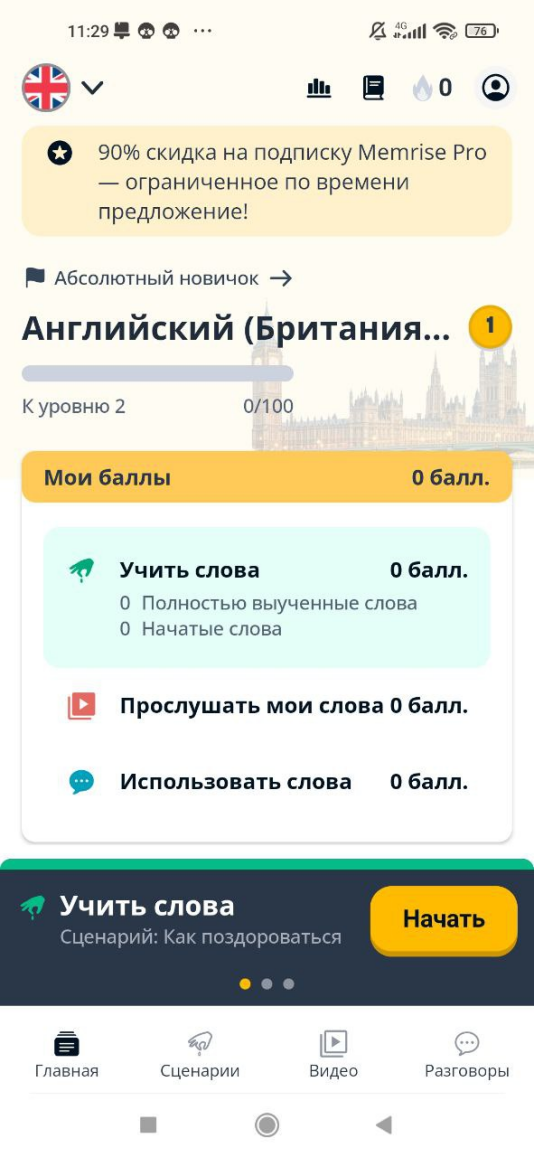


Рисунок 4 – Интерфейс Memrise

Quizlet - приложение для создания и использования карточек для изучения различных предметов, включая английский язык. Пользователи могут создавать свои наборы карточек или использовать готовые.

Преимущества:

* Универсальность: Подходит для изучения любых тем и предметов, в том числе английского языка.
* Гибкость: Пользователи могут создавать свои собственные наборы карточек, адаптируя их к своим потребностям.
* Разнообразие режимов обучения: Предлагает различные режимы обучения (заучивание, письмо, соответствие, тест).
* Бесплатность (ограниченная): Базовый функционал приложения доступен бесплатно.

Недостатки:

* Простота: Может показаться слишком простым для пользователей, ищущих более комплексное обучение.
* Зависимость от контента: Эффективность обучения зависит от качества созданных карточек.
* Реклама и платные функции: Наличие рекламы и платных функций может быть раздражающим.



Рисунок 5 – Логотип Quizlet

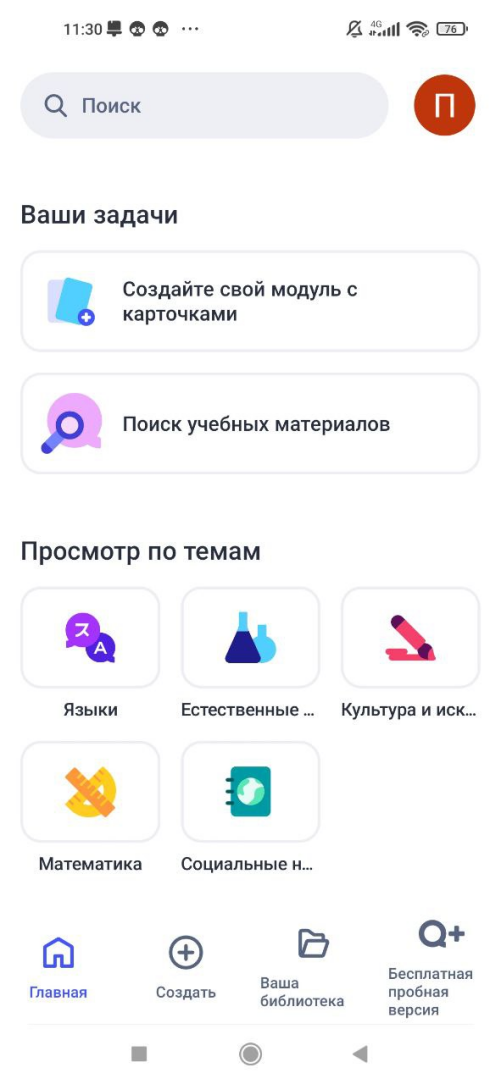


Рисунок 6 – Интерфейс Quizlet

## 1.2 Техническое задание

1. Общая информация:

Наименование проекта: ITEnglishApplication.

Цель проекта: разработка мобильного приложения для изучения английского языка, направленного на расширение словарного запаса посредством интерактивных тестов на угадывание перевода слов.

1. Требования к программе:

Требования к функциональным характеристикам:

Пользовательские функции:

* Авторизация/Регистрация
* Генерация и интерфейс тестов
* Выбор и проверка ответа
* Подсчёт результатов

Требования к надёжности:

* Приложение должно быть стабильным и не содержать критических ошибок.
* Приложение должно корректно обрабатывать ошибки ввода данных.
* Приложение должно предотвращать потерю данных в случае сбоев.

Требования к производительности:

* Быстрый запуск приложения.
* Плавная работа интерфейса.
* Быстрая генерация тестов.

Требования к тестированию:

* Проведение функционального тестирования для проверки всех функциональных возможностей приложения.
* Проведение тестирования пользовательского интерфейса для оценки удобства использования.
* Проведение тестирования производительности для оценки скорости работы приложения.
* Проведение тестирования на различных устройствах и версиях операционной системы.
* Создание тестовой документации (тест-кейсы, отчёты о тестировании).

Требования к сопровождению:

* Устранение обнаруженных ошибок и недочётов.
* Обновление базы данных слов и переводов (опционально).
* Добавление новых функциональных возможностей (опционально).

1. Стадии разработки:

* Анализ требований и планирование
* Проектирование приложения
* Разработка
* Тестирование

## 1.3 Описание структуры приложения

Приложение предназначено для пользователей, изучающих английский язык, независимо от их текущего уровня владения. Основная цель приложения - расширение словарного запаса. Приложение стремится предоставить пользователям удобный и доступный инструмент для запоминания новых слов и проверки уже имеющихся знаний. Оно использует формат интерактивных тестов, который делает процесс обучения более интересным и запоминающимся, по сравнению с традиционными методами заучивания слов.

Основная задача приложения - предоставить пользователю возможность в игровой форме, посредством тестов, угадывать перевод английских слов и, тем самым, активно расширять свой словарный запас.

Реализация этой задачи включает в себя:

* Генерацию тестов: Автоматическое создание тестов, адаптированных под уровень знаний пользователя.
* Предоставление вариантов ответов: Отображение правильного перевода в числе нескольких возможных вариантов, что требует от пользователя активного мышления и знания контекста.
* Проверку и оценку: Немедленную проверку ответов с отображением правильных и неправильных вариантов, а также предоставление общей статистики по результатам теста.

Для повышения эффективности и привлекательности приложения могут быть добавлены следующие дополнительные данные и функциональность:

* База данных слов: Обогащённая база данных английских слов с различными значениями, синонимами и примерами использования. Важно, чтобы база данных постоянно расширялась и обновлялась.
* Уровни сложности: Градация уровней сложности, учитывающая не только сложность слов, но и грамматические конструкции.
* Персонализированные тесты: Алгоритмы, учитывающие слабые места пользователя и предлагающие тесты, направленные на их устранение.
* Произношение слов: Возможность прослушать произношение английского слова.
* Иллюстрации: Добавление иллюстраций к словам для улучшения запоминания.

# 2 Описание разработки приложения

## 2.1 Обоснование средств разработки

Язык программирования: Kotlin.

Преимущества Kotlin:

* Современный и лаконичный синтаксис: Kotlin обладает более выразительным и безопасным синтаксисом, чем Java, что позволяет писать меньше кода и снижает вероятность ошибок. Например, использование data classes для создания моделей данных значительно сокращает объем boilerplate-кода.
* Null-safety: Kotlin имеет встроенную систему предотвращения NullPointerExceptions, что делает код более надёжным и устойчивым к сбоям.
* Корутины для асинхронного программирования: Kotlin предоставляет мощный инструмент корутин, который упрощает написание асинхронного кода и делает его более читаемым и простым в обслуживании. Корутины позволяют избежать проблем с callback hell и улучшить производительность приложения.
* Полная совместимость с Java: Kotlin полностью совместим с Java, что позволяет использовать существующие Java-библиотеки и фреймворки в Kotlin-проектах. Это позволяет постепенно переходить на Kotlin, не переписывая весь код с нуля.
* Extension functions: Kotlin позволяет добавлять новые функции к существующим классам без необходимости их наследования, что делает код более гибким и расширяемым.



Рисунок 7 – Логотип Kotlin

Среда разработки: Android Studio.

Преимущества Android Studio:

* Официальная IDE для Android: Android Studio - это официальная IDE для разработки Android-приложений, разработанная компанией Google. Это гарантирует полную поддержку всех функций Android SDK и наличие необходимых инструментов для разработки, тестирования и отладки приложений.
* Интегрированный эмулятор: Android Studio включает в себя мощный эмулятор Android, который позволяет тестировать приложения на различных устройствах и версиях Android без необходимости использования физического устройства.
* Визуальный редактор макетов: Android Studio предоставляет визуальный редактор макетов (Layout Editor), который упрощает создание пользовательских интерфейсов путём перетаскивания элементов и настройки их свойств.
* Инструменты профилирования: Android Studio включает в себя инструменты для профилирования производительности приложения, которые позволяют выявлять узкие места и оптимизировать код.
* Поддержка Kotlin: Android Studio имеет отличную поддержку Kotlin, включая автодополнение кода, рефакторинг и отладку.
* Интеграция с системами контроля версий: Android Studio интегрирована с системами контроля версий, такими как Git, что упрощает совместную работу над проектом.

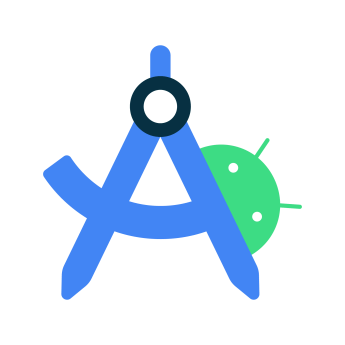


Рисунок 8 - Логотип Android Studio

База данных: SQLite.

Преимущества SQLite:

* Локальное хранение данных: SQLite - это легковесная, автономная и самодостаточная реляционная база данных, которая хранит данные непосредственно на устройстве пользователя в одном файле. Это означает, что приложение может работать без подключения к интернету, что является важным преимуществом для многих пользователей.
* Не требует сервера: SQLite не требует установки и настройки отдельного сервера базы данных, что упрощает развёртывание и использование приложения.
* Простота использования: SQLite имеет простой и понятный SQL-синтаксис, что облегчает разработку и отладку запросов.
* Транзакции: SQLite поддерживает транзакции, что гарантирует целостность данных даже в случае сбоев.
* Бесплатная и открытый исходный код: SQLite распространяется под открытой лицензией и не требует лицензионных отчислений.
* Небольшой размер: SQLite имеет очень небольшой размер, что не влияет на размер приложения.



Рисунок 9 – Логотип SQLite

Для создание макета дизайна интерфейса будет использована программа Figma.

Преимущества Figma:

* Совместная работа

Реальное время: Позволяет нескольким пользователям работать над проектом одновременно, что упрощает командное взаимодействие.

Обратная связь: Участники могут оставлять комментарии прямо на дизайне, облегчая процесс получения отзывов.

* Доступность

Облачное хранилище: Доступ к проектам из любого места с интернетом, без необходимости устанавливать ПО на компьютер.

Платформенная независимость: Работает в браузере и поддерживает различные операционные системы.

* Инструменты

Многофункциональные инструменты: Предоставляет широкий набор инструментов для создания интерфейсов, прототипов и анимаций.

Figma становится все более популярным инструментом для дизайнеров благодаря своим возможностям совместной работы, доступности и функциональности, что делает процесс дизайна более эффективным и гибким.

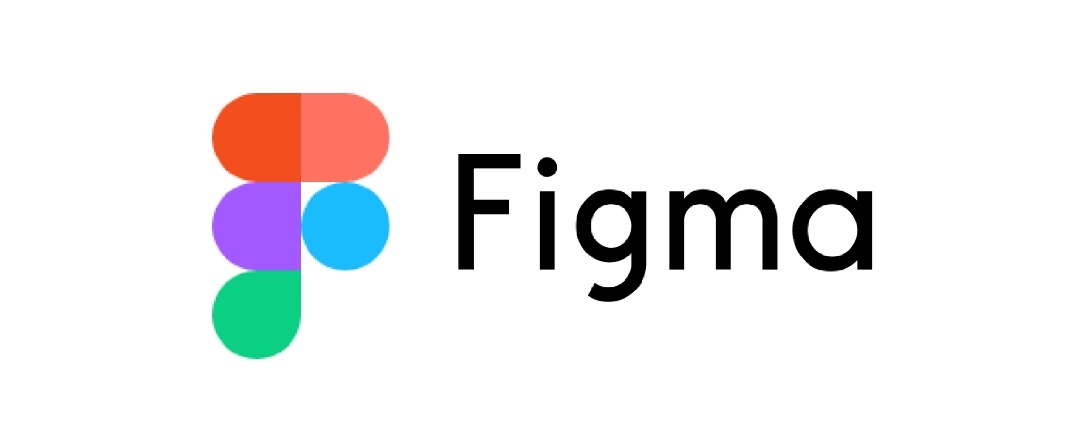


Рисунок 10 – Логотип Figma

## 2.2 Разработка интерфейса

Для начала нужно создать макет в Figma. Макет представлен на рисунках 11 и 12, а также по ссылке <https://www.figma.com/design/kGeFmmVYVaBxW5XT3eZyVB/Untitled?node-id=0-1&m=dev&t=2QaSS5Wz8lbdmchF-1>

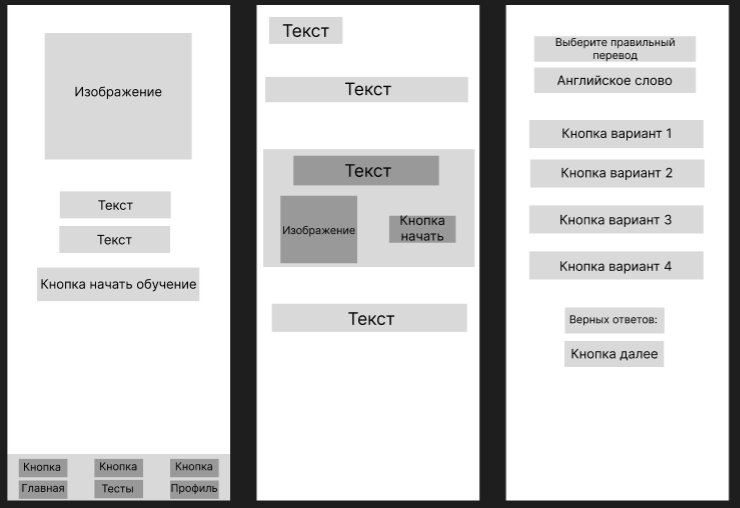


Рисунок 11 – Макет интерфейса

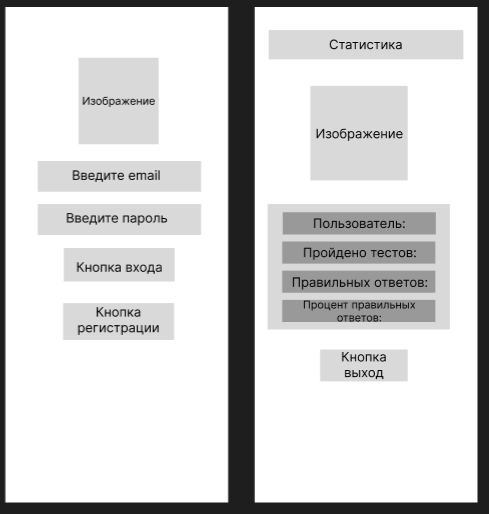


Рисунок 12 – Продолжение макета интерфейса

Далее макеты были доработаны уже в Android Studio и перенесены в код приложения. Итоговый интерфейс представлен на рисунках с 13 по 17.



Рисунок 13 – Главное меню приложения

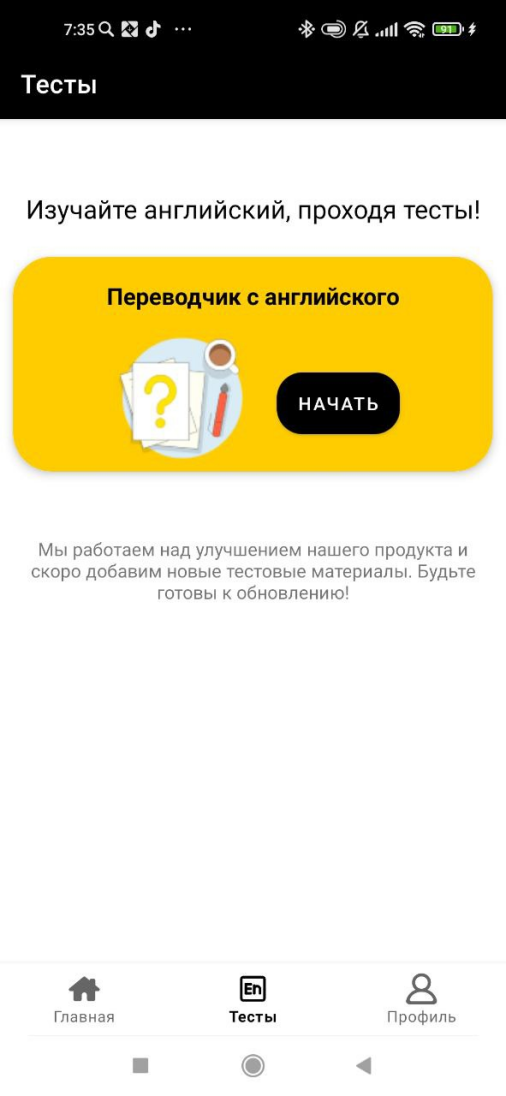


Рисунок 14 – Подготовка к тестам



Рисунок 15 – Тесты

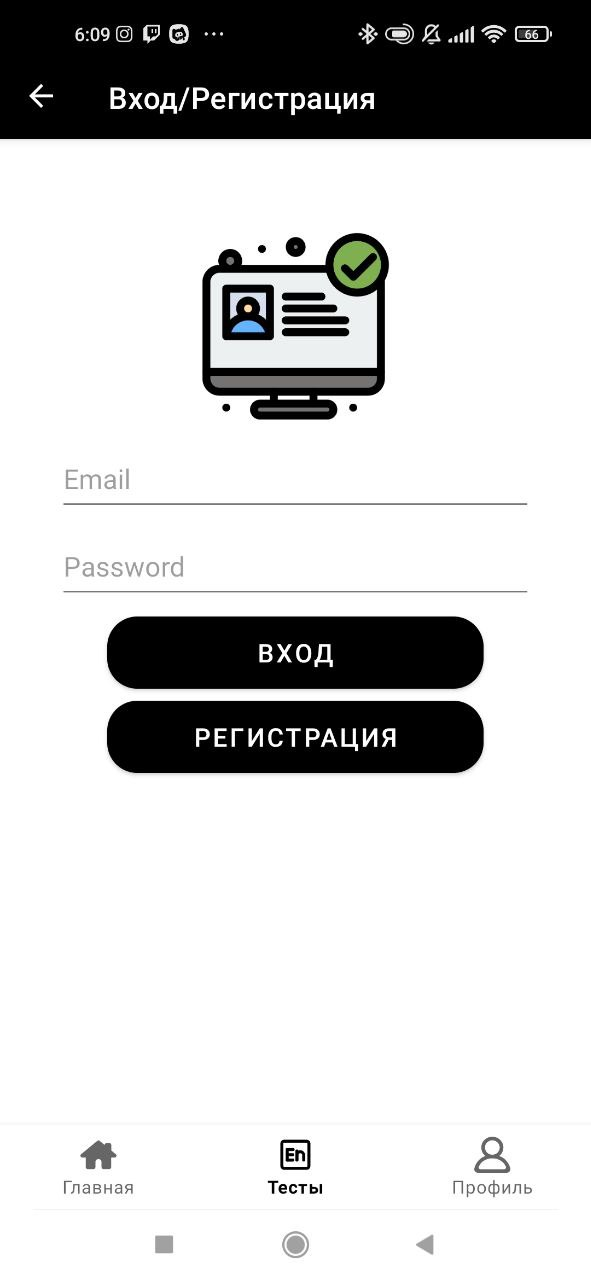


Рисунок 16 – Вход/Регистрация

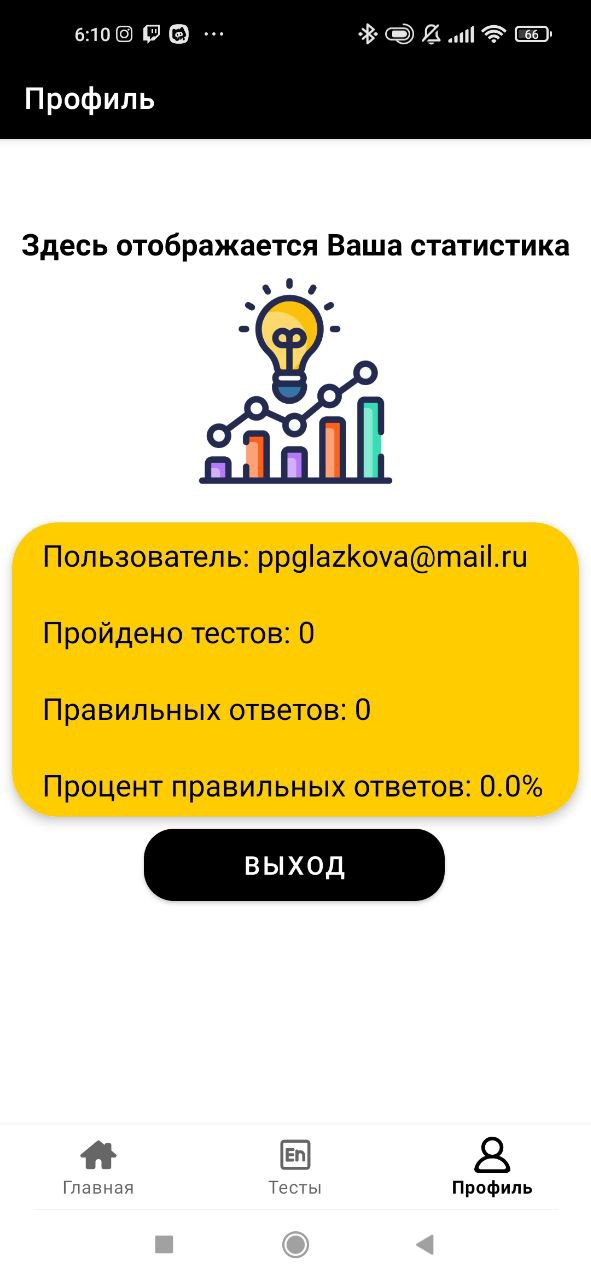


Рисунок 17 – Профиль

## 2.3. Разработка логики работы приложения. Схема взаимодействия компонентов проекта

### 2.3.1 Логика работы приложения

Разработка логики работы программы «ITEnglishApplication» включает в себя проектирование и реализацию алгоритмов и процессов, которые управляют функциональностью приложения. Основные компоненты логики и их взаимодействие описаны ниже:

1. Управление пользователем

Регистрация:

* Получение данных пользователя (email, пароль) через пользовательский интерфейс.
* Валидация введённых данных (проверка формата email, надежности пароля).
* Создание нового пользователя.

Вход:

* Получение данных пользователя (email, пароль) через пользовательский интерфейс.
* Аутентификация пользователя.
* Сохранение информации о текущем пользователе.

Выход:

* Удаление информации о текущем пользователе.

Генерация тестов.

* Получение списка слов и их переводов из базы данных, соответствующих выбранному уровню сложности.

Создание вопросов:

* Выбор случайного слова из списка для создания вопроса.
* Получение правильного перевода слова из базы данных.
* Создание списка вариантов ответов, включающего правильный перевод и несколько случайных неправильных переводов (дистракторов).
* Перемешивание вариантов ответов в случайном порядке.

Формирование теста:

* Создание списка вопросов, составляющих тест.
* Определение порядка вопросов в тесте (случайный порядок).

### 2.3.2 Схема взаимодействия компонентов

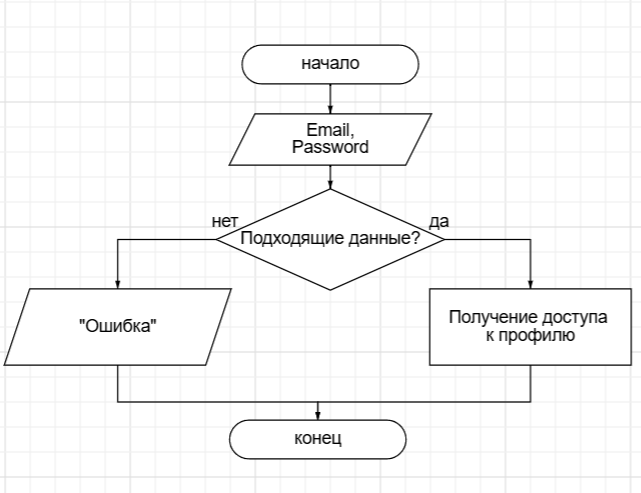


Рисунок 13 – Схема алгоритма регистрации/входу.

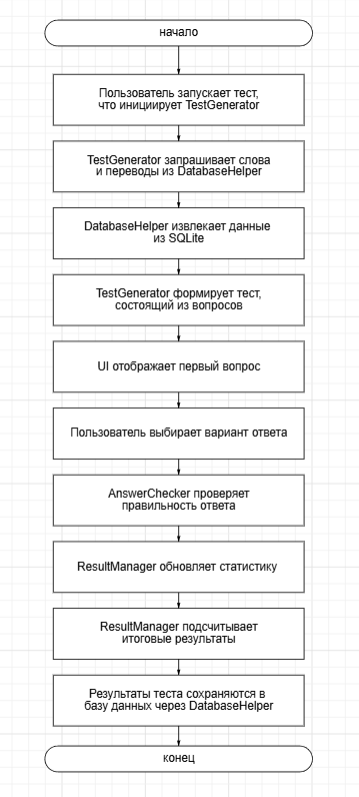


Рисунок 14 – Схема взаимодействия компонентов

## 2.4 Описание переменных, компонентов, классов и подпрограмм

Таблица 1 – Описание переменных, компонентов, классов и подпрограмм.

|  |  |  |
| --- | --- | --- |
| Тип | Название | Описание |
| Классы | MainActivity | Главный экран приложения. |
| MainActivityTest | Класс, который содержит тесты для проверки функциональности MainActivity. |
| ExampleUnitTest | Класс, предназначенный для выполнения юнит-тестов. |
| ExampleInstrumentedTest | Класс, предназначенный для выполнения инструментальных тестов на Android-устройстве. |
| Переменные | activityTest | Правило, обеспечивающее запуск указанной активности перед выполнением тестов. |
| Подпрограммы (Тесты) | checkMenu | Убедиться в наличии кнопок меню |
| checkBtn1 | Нажимает на первый элемент меню, проверяет наличие кнопки "Начать обучение". |
| checkBtn2 | Нажимает на второй элемент меню, проверяет наличие текста "Переводчик с английского". |
| checkBtn3 | Нажимает на третий элемент меню |

Продолжение таблицы 1

|  |  |  |
| --- | --- | --- |
| Тип | Название | Описание |
| Подпрограммы (Тесты) | checkTest | Проверяет наличие вариантов ответа после кнопки "Начать тест". |
| checkHomePage | Нажимает на элемент меню "Главная" и проверяет кнопку "Начать обучение", выполняет клик на кнопку "Начать обучение" и проверяет наличие "Переводчик с английского" в соответствующем фрагменте. |
| checkTestsPage | Нажимает на элемент меню "Панель инструментов" и проверяет, что отображается фрагмент с кнопкой "Начать тест", выполняет клик на кнопку "Начать тест" и проверяет отображение всех вариантов ответа. |
| checkProfilePage | Проверяет наличие следующих элементов: поле ввода электронной почты, поле ввода пароля, кнопка "Войти", кнопка "Регистрация". |
| useAppContext | Проверяет, что контекст приложения получен корректно, и packageName соответствует ожидаемому. |

# 3 Тестирование и установка программы

Установка приложения:

Для установки программы нужно скачать ITEnglishApplication.apk.

Теперь приложение готово к использованию.

Тестирование приложения:

Таблица 2 – Тестирование программы

|  |  |  |  |
| --- | --- | --- | --- |
| № | Название теста | Описание | Результат |
| 1 | Регистрация нового пользователя | Заполнение формы регистрации нового пользователя и подтверждение регистрации. | Пользователь успешно зарегистрирован и может войти в приложение. |
| 2 | Авторизация существующего пользователя | Ввод логина и пароля существующего пользователя и вход в приложение. | Пользователь успешно авторизован и перенаправлен в профиль. |
| 3 | Некорректный ввод данных при регистрации | Попытка регистрации с некорректными данными (например, неверный формат email, пароль менее символов). | Отображается сообщение об ошибке с указанием причины. |
| 4 | Некорректный ввод данных при авторизации | Попытка авторизации с некорректным логином или паролем. | Отображается сообщение об ошибке авторизации. |

Продолжение таблицы 2

|  |  |  |  |
| --- | --- | --- | --- |
| № | Название теста | Описание | Результат |
| 5 | Проверка элементов главного меню | Проверка наличия и видимости элементов навигации главного меню. | Отображаются все элементы навигации (Главная, Тесты, Профиль). |
| 6 | Отображение вариантов ответа | Проверка наличия и видимости вариантов ответа на экране вопроса. | Отображаются все доступные варианты ответа для текущего вопроса. |
| 7 | Проверка правильности ответа | Проверка, является ли выбранный вариант ответа правильным. | Если ответ правильный: отображается уведомление о правильности ответа, увеличивается счетчик правильных ответов.  Если ответ неправильный: отображается уведомление о неправильности ответа. |

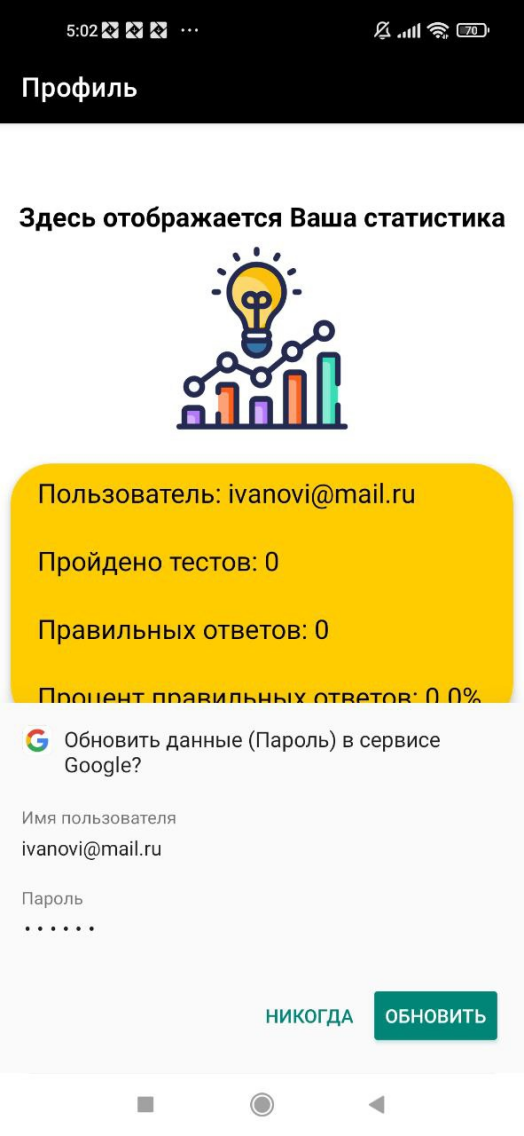


Рисунок 13 – Результат первого теста

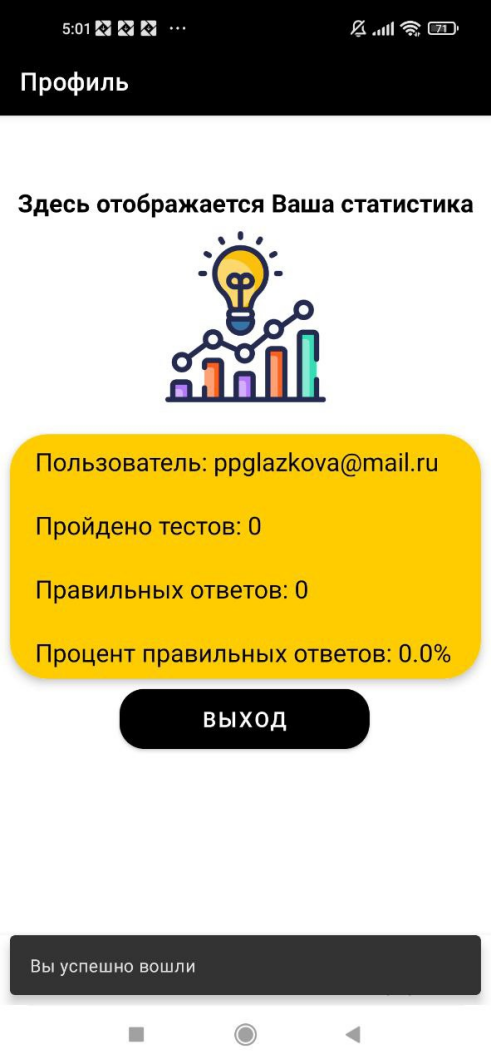


Рисунок 14 – Результат второго теста



Рисунок 15 – Результат третьего теста

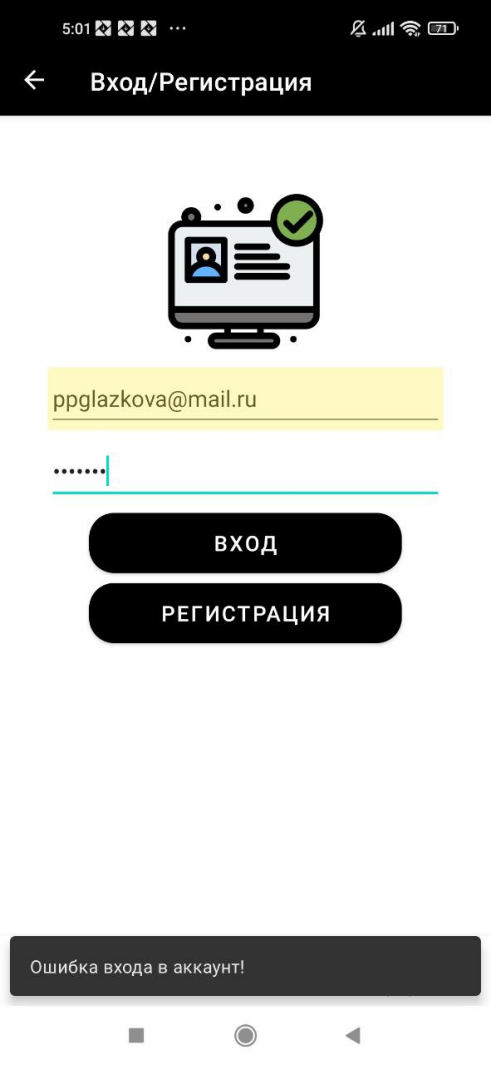


Рисунок 16 – Результат четвертого теста



Рисунок 17 – Результат пятого теста

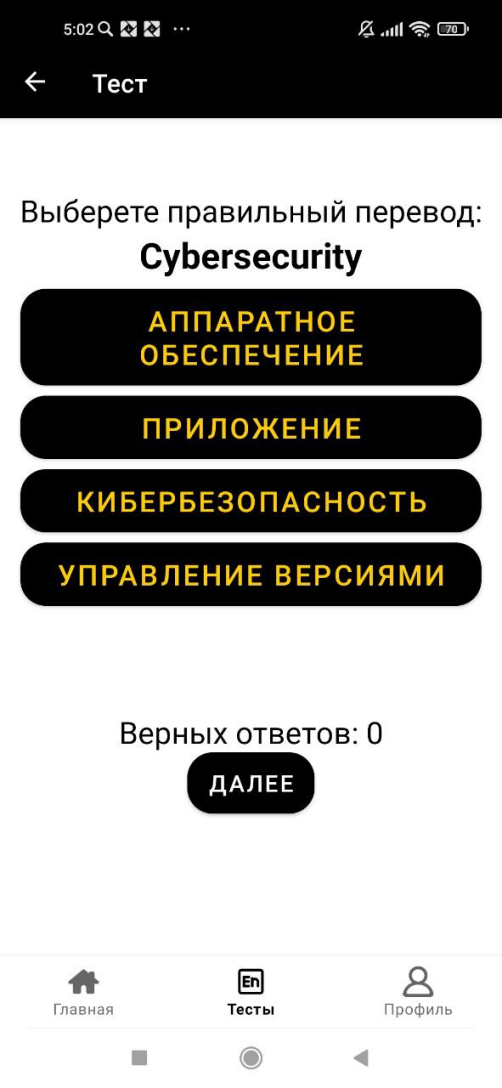


Рисунок 18 – Результат шестого теста

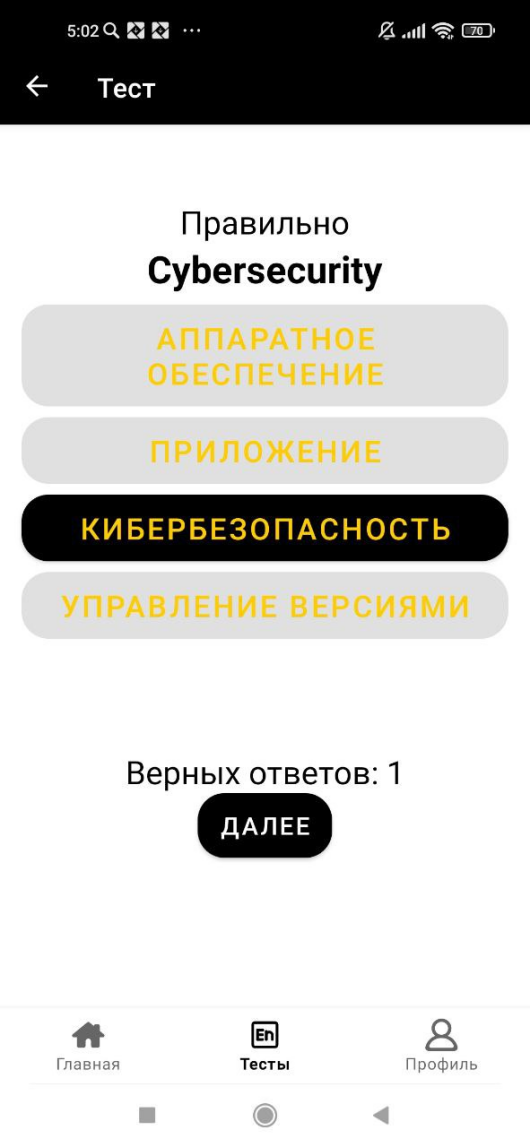


Рисунок 19 – Результат седьмого теста

Вывод: все тесты прошли успешно, ошибок не возникло.

# Заключение

В ходе выполнения курсового проекта «ITEnglishApplication» была разработана и реализована мобильная игра, предназначенная для расширения словарного запаса английского языка у пользователей различных уровней подготовки.

Проект включал в себя следующие основные этапы:

* Анализ предметной области: Изучена предметная область, определена целевая аудитория, выявлены ключевые понятия и функции приложения.
* Проектирование архитектуры приложения: Разработана архитектура приложения, определены основные компоненты и их взаимодействие.
* Разработка пользовательского интерфейса (UI): Спроектирован интуитивно понятный и привлекательный пользовательский интерфейс с учетом принципов UI/UX дизайна. Использовался инструмент Figma для создания макетов и прототипов.
* Реализация функциональности: Реализованы основные функции приложения, включая выбор уровня сложности, генерацию тестов, проверку правильности ответов, подсчёт и отображение результатов. Для хранения данных использовалась локальная база данных SQLite.
* Тестирование приложения: Проведено тестирование приложения для выявления и исправления дефектов.
* Написание документации: Подготовлена документация, описывающая архитектуру приложения, компоненты, классы и функции, а также процесс тестирования.

# Список использованной литературы

1. Документация Android Developers. Официальный сайт. – URL: <https://developer.android.com/docs> (дата обращения: 28.04.2025).
2. Документация Figma. Официальный сайт. – URL: <https://help.figma.com/hc/en-us> (дата обращения: 28.04.2025).
3. Документация Kotlin. Официальный сайт. – URL: <https://kotlinlang.org/docs/home.html> (дата обращения: 28.04.2025).
4. Документация SQLite. Официальный сайт. – URL: <https://www.sqlite.org/docs.html> (дата обращения: 28.04.2025).
5. Крог С. UI/UX дизайн мобильных приложений. – М.: БХВ-Петербург, 2019. – 312 с.
6. Нойес Б., Лукса М. Android Studio 4.0 Development Essentials. – 2020. – 752 с.
7. Сведж Б. Изучаем Kotlin для Android-разработки. – СПб.: Питер, 2021. – 432 с.
8. Соммервилл И. Инженерия программного обеспечения. – 10-е изд. – М.: Вильямс, 2017. – 784 с.
9. Филлипс Б., Харди К., Марвин Дж. Android. Программирование для профессионалов. – 4-е изд. – СПб.: Питер, 2020. – 1104 с.

# Приложение А

Листинг программы

MainActivity.kt

package com.example.application

import android.os.Bundle

import android.util.Log

import androidx.activity.viewModels

import com.google.android.material.bottomnavigation.BottomNavigationView

import androidx.appcompat.app.AppCompatActivity

import androidx.core.os.bundleOf

import androidx.lifecycle.ViewModel

import androidx.lifecycle.ViewModelProvider

import androidx.navigation.findNavController

import androidx.navigation.ui.AppBarConfiguration

import androidx.navigation.ui.setupActionBarWithNavController

import androidx.navigation.ui.setupWithNavController

import androidx.room.Room

import com.example.application.data.Word

import com.example.application.data.WordDatabase

import com.example.application.databinding.ActivityMainBinding

import kotlinx.coroutines.Dispatchers

import kotlinx.coroutines.GlobalScope

import kotlinx.coroutines.launch

class MainActivity : AppCompatActivity() {

private lateinit var binding: ActivityMainBinding

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

binding = ActivityMainBinding.inflate(layoutInflater)

setContentView(binding.root)

val navView: BottomNavigationView = binding.navView

val navController = findNavController(R.id.nav\_host\_fragment\_activity\_main)

// Passing each menu ID as a set of Ids because each

// menu should be considered as top level destinations.

val appBarConfiguration = AppBarConfiguration(

setOf(

R.id.navigation\_home, R.id.navigation\_dashboard, R.id.navigation\_notifications

)

)

setupActionBarWithNavController(navController, appBarConfiguration)

navView.setupWithNavController(navController)

}

}

MainActivityTest.kt

package com.example.application

import androidx.test.espresso.Espresso.onView

import androidx.test.espresso.action.ViewActions.click

import androidx.test.espresso.assertion.ViewAssertions.matches

import androidx.test.espresso.matcher.ViewMatchers.hasDescendant

import androidx.test.espresso.matcher.ViewMatchers.isDisplayed

import androidx.test.espresso.matcher.ViewMatchers.withId

import androidx.test.espresso.matcher.ViewMatchers.withText

import androidx.test.ext.junit.rules.ActivityScenarioRule

import org.junit.Assert.\*

import org.junit.Rule

import org.junit.Test

class MainActivityTest {

@get:Rule

val activityTest = ActivityScenarioRule(MainActivity::class.java)

@Test

fun checkMenu() {

// Проверяем, что приложение открыто

onView(withId(R.id.container)).check(matches(isDisplayed()))

// Проверяем наличие кнопок меню

onView(withId(R.id.navigation\_home)).check(matches(isDisplayed()))

onView(withId(R.id.navigation\_dashboard)).check(matches(isDisplayed()))

onView(withId(R.id.navigation\_notifications)).check(matches(isDisplayed()))

}

@Test

fun checkBtn1() {

// Проверяем, что приложение открыто

onView(withId(R.id.container)).check(matches(isDisplayed()))

// Нажимаем на перый элемент меню

onView(withId(R.id.navigation\_home)).perform(click())

// Проверяем, что после нажатия на элемент меню отображается соответствующий фрагмент

onView(withId(R.id.nav\_host\_fragment\_activity\_main))

.check(matches(hasDescendant(withId(R.id.begin\_studies\_button))));

}

@Test

fun checkBtn2() {

// Проверяем, что приложение открыто

onView(withId(R.id.container)).check(matches(isDisplayed()))

// Нажимаем на второй элемент меню

onView(withId(R.id.navigation\_dashboard)).perform(click())

// Проверяем, что после нажатия на элемент меню отображается соответствующий фрагмент

onView(withId(R.id.nav\_host\_fragment\_activity\_main))

.check(matches(hasDescendant(withText("Переводчик с английского"))));

}

@Test

fun checkBtn3() {

// Проверяем, что приложение открыто

onView(withId(R.id.container)).check(matches(isDisplayed()))

// Нажимаем на третий элемент меню

onView(withId(R.id.navigation\_notifications)).perform(click())

}

@Test

fun checkTest() {

// Проверяем, что приложение открыто

onView(withId(R.id.container)).check(matches(isDisplayed()))

// Нажимаем на элемент меню

onView(withId(R.id.navigation\_dashboard)).perform(click())

// Проверяем, что после нажатия на элемент меню отображается соответствующий фрагмент

onView(withId(R.id.nav\_host\_fragment\_activity\_main))

.check(matches(hasDescendant(withText("Переводчик с английского"))));

onView(withId(R.id.nav\_host\_fragment\_activity\_main))

.check(matches(hasDescendant(withId(R.id.begin\_button))));

onView(withId(R.id.nav\_host\_fragment\_activity\_main))

.check(matches(hasDescendant(withId(R.id.test\_image1))));

// Получаем и нажимаем на кнопку начала теста

onView(withId(R.id.begin\_button)).perform(click());

// Наличие вариантов ответа

onView(withId(R.id.nav\_host\_fragment\_activity\_main))

.check(matches(hasDescendant(withId(R.id.variant1))));

onView(withId(R.id.nav\_host\_fragment\_activity\_main))

.check(matches(hasDescendant(withId(R.id.variant2))));

onView(withId(R.id.nav\_host\_fragment\_activity\_main))

.check(matches(hasDescendant(withId(R.id.variant3))));

onView(withId(R.id.nav\_host\_fragment\_activity\_main))

.check(matches(hasDescendant(withId(R.id.variant4))));

// Наличие остальных элементов

onView(withId(R.id.nav\_host\_fragment\_activity\_main))

.check(matches(hasDescendant(withId(R.id.question))));

onView(withId(R.id.nav\_host\_fragment\_activity\_main))

.check(matches(hasDescendant(withId(R.id.next))));

onView(withId(R.id.nav\_host\_fragment\_activity\_main))

.check(matches(hasDescendant(withId(R.id.result))));

onView(withId(R.id.nav\_host\_fragment\_activity\_main))

.check(matches(hasDescendant(withId(R.id.count))));

onView(withId(R.id.next)).perform(click());

}

@Test

fun checkHomePage() {

// Проверяем, что приложение открыто

onView(withId(R.id.container)).check(matches(isDisplayed()))

// Нажимаем элемент меню

onView(withId(R.id.navigation\_home)).perform(click())

// Проверяем, что после нажатия на элемент меню отображается соответствующий фрагмент

onView(withId(R.id.nav\_host\_fragment\_activity\_main))

.check(matches(hasDescendant(withId(R.id.begin\_studies\_button))));

}

}

ExampleUnitTest.kt

package com.example.application

import org.junit.Test

import org.junit.Assert.\*

/\*\*

\* Example local unit test, which will execute on the development machine (host).

\*

\* See [testing documentation](http://d.android.com/tools/testing).

\*/

class ExampleUnitTest {

@Test

fun addition\_isCorrect() {

assertEquals(4, 2 + 2)

}

}

ExampleInstrumentedTest.kt

package com.example.application

import androidx.test.platform.app.InstrumentationRegistry

import androidx.test.ext.junit.runners.AndroidJUnit4

import org.junit.Test

import org.junit.runner.RunWith

import org.junit.Assert.\*

/\*\*

\* Instrumented test, which will execute on an Android device.

\*

\* See [testing documentation](http://d.android.com/tools/testing).

\*/

@RunWith(AndroidJUnit4::class)

class ExampleInstrumentedTest {

@Test

fun useAppContext() {

// Context of the app under test.

val appContext = InstrumentationRegistry.getInstrumentation().targetContext

assertEquals("com.example.application", appContext.packageName)

}

}

TestFragment.kt

package com.example.application.ui.test

import android.os.Bundle

import android.util.Log

import androidx.fragment.app.Fragment

import android.view.LayoutInflater

import android.view.View

import android.view.ViewGroup

import android.widget.Button

import android.widget.TextView

import androidx.navigation.fragment.findNavController

import androidx.room.Room

import com.example.application.R

import com.example.application.data.Word

import com.example.application.data.WordDatabase

import com.google.firebase.auth.FirebaseAuth

import com.google.firebase.database.FirebaseDatabase

import com.google.firebase.firestore.FirebaseFirestore

import kotlinx.coroutines.Dispatchers

import kotlinx.coroutines.GlobalScope

import kotlinx.coroutines.launch

import kotlin.random.Random

class TestFragment : Fragment() {

private lateinit var database: WordDatabase

private lateinit var questionField: TextView

private lateinit var resultField: TextView

private lateinit var rightAnswer:TextView

private lateinit var count: TextView

private lateinit var variant1: Button

private lateinit var variant2: Button

private lateinit var variant3: Button

private lateinit var variant4: Button

private lateinit var nextBtn: Button

private lateinit var currentWord: Word

private var currentWordIndex = 0

private var rightAnswers = 0

override fun onCreateView(

inflater: LayoutInflater, container: ViewGroup?,

savedInstanceState: Bundle?

): View? {

return inflater.inflate(R.layout.fragment\_test, container, false)

}

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {

super.onViewCreated(view, savedInstanceState)

val firestore = FirebaseFirestore.getInstance()

questionField = view.findViewById(R.id.question)

resultField = view.findViewById(R.id.result)

rightAnswer = view.findViewById(R.id.right\_answer)

resultField.text = "Выберете правильный перевод:"

count = view.findViewById(R.id.count)

count.text = "Верных ответов: " + rightAnswers.toString()

variant1 = view.findViewById(R.id.variant1)

variant2 = view.findViewById(R.id.variant2)

variant3 = view.findViewById(R.id.variant3)

variant4 = view.findViewById(R.id.variant4)

nextBtn = view.findViewById(R.id.next)

GlobalScope.launch(Dispatchers.IO) {

database = Room.databaseBuilder(requireActivity().applicationContext, WordDatabase::class.java, "english\_words")

.createFromAsset("database/english\_words.db")

.build()

val wordDAO = database.wordDAO()

val words = wordDAO.getRandomWords()

Log.d("RRR", words.toString())

requireActivity().runOnUiThread {

updateUI(words)

}

nextBtn.setOnClickListener {

if(currentWordIndex == words.size - 2) {

nextBtn.text = "Завершить"

}

if (currentWordIndex < words.size - 1) {

variant1.isEnabled = true

variant2.isEnabled = true

variant3.isEnabled = true

variant4.isEnabled = true

resultField.text = "Выберете правильный перевод:"

rightAnswer.text = ""

currentWordIndex++

updateUI(words)

} else {

if(FirebaseAuth.getInstance().currentUser != null) {

firestore.collection("users").document(FirebaseAuth.getInstance().currentUser!!.uid).get()

.addOnSuccessListener {

var tests = it.getDouble("tests")!!.toInt()

var answers = it.getDouble("answers")!!.toInt()

tests++

answers += count.text.toString().replace("Верных ответов: ", "").toInt()

firestore.collection("users").document(FirebaseAuth.getInstance().currentUser!!.uid).update(

mapOf(

"tests" to tests,

"answers" to answers

)

)

Log.d("RRR", "$tests $answers")

}

}

findNavController().navigate(R.id.action\_testFragment\_to\_navigation\_dashboard)

}

}

}

variant1.setOnClickListener {

variant2.isEnabled = false

variant3.isEnabled = false

variant4.isEnabled = false

if(variant1.text.toString() == currentWord.wordRussian) {

Log.d("RRR", "Правильно")

resultField.text = "Правильно"

rightAnswers++

} else {

resultField.text = "Неправильно"

rightAnswer.text = "Верно: " + currentWord.wordRussian

}

count.text = "Верных ответов: " + rightAnswers.toString()

}

variant2.setOnClickListener {

variant1.isEnabled = false

variant3.isEnabled = false

variant4.isEnabled = false

if(variant2.text.toString() == currentWord.wordRussian) {

Log.d("RRR", "Правильно")

resultField.text = "Правильно"

rightAnswers++

} else {

resultField.text = "Неправильно"

rightAnswer.text = "Верно: " + currentWord.wordRussian

}

count.text = "Верных ответов: " + rightAnswers.toString()

}

variant3.setOnClickListener {

variant1.isEnabled = false

variant2.isEnabled = false

variant4.isEnabled = false

if(variant3.text.toString() == currentWord.wordRussian) {

Log.d("RRR", "Правильно")

resultField.text = "Правильно"

rightAnswers++

} else {

resultField.text = "Неправильно"

rightAnswer.text = "Верно: " + currentWord.wordRussian

}

count.text = "Верных ответов: " + rightAnswers.toString()

}

variant4.setOnClickListener {

variant1.isEnabled = false

variant2.isEnabled = false

variant3.isEnabled = false

if(variant4.text.toString() == currentWord.wordRussian) {

Log.d("RRR", "Правильно")

resultField.text = "Правильно"

rightAnswers++

} else {

resultField.text = "Неправильно"

rightAnswer.text = "Верно: " + currentWord.wordRussian

}

count.text = "Верных ответов: " + rightAnswers.toString()

}

}

private fun updateUI(words: List<Word>) {

currentWord = words[currentWordIndex]

questionField.text = currentWord.wordEnglish

val filteredList = words.toMutableList().apply { removeAt(currentWordIndex) }

val randomIndices = mutableListOf<Int>()

while (randomIndices.size < 3) {

val randomIndex = Random.nextInt(filteredList.size)

if (!randomIndices.contains(randomIndex)) {

randomIndices.add(randomIndex)

}

}

randomIndices.map { filteredList[it] }.toList()

var answers : List<String> = listOf(filteredList[randomIndices[0]].wordRussian,

filteredList[randomIndices[1]].wordRussian,

filteredList[randomIndices[2]].wordRussian,

currentWord.wordRussian)

answers = answers.shuffled()

Log.d("RRR", answers.toString())

variant1.text = answers[0]

variant2.text = answers[1]

variant3.text = answers[2]

variant4.text = answers[3]

}

}

# Приложение Б

Руководство пользователя

ITEnglishApplication – мобильное приложение, созданное для эффективного расширения вашего словарного запаса английского языка. С помощью интерактивных тестов вы сможете проверять свои знания и учить новые слова. Это руководство поможет вам освоить все основные функции приложения и начать обучение.

Для начала работы с приложением ITEnglishApplication необходимо установить его на вашем смартфоне. Приложение находится в виде файла .apk и может быть запущено на смартфонах под управлением операционной системы Android.

После загрузки исполняемого файла ITEnglishApplication.apk, запустите его, и вы попадёте на главный экран.



Рисунок 1 – Главный экран

Регистрация и авторизация:

При первом запуске приложения вам будет предложено зарегистрироваться или войти в существующий аккаунт.

Регистрация:

Введите email и password (не менее 6 символов).

Нажмите кнопку «Регистрация».

Вход:

Введите email и password.

Нажмите «Вход».

Если у вас уже есть сохранённый аккаунт, то просто нажмите на всплывающую почту, пароль введётся автоматически и вы сразу зайдёте в свой профиль.

Навигация:

Основные элементы управления:

Главный экран: открывается при входе в приложение, при нажатии на кнопку «Начать обучение» вас перекидывает в следующий раздел (рисунок 1).

Тесты: открывается окно, после нажатия на «Начать» вас перекидывает к самим тестам, где дано английское слово и нужно найти перевод из 4 предложенных слов.



Рисунок 2 – Тесты

Профиль: открывается окно «Вход/Регистрация», если вы зашли впервые.

Если вы уже вошли в аккаунт, то показывается количество тестов, правильных ответов и их процент.

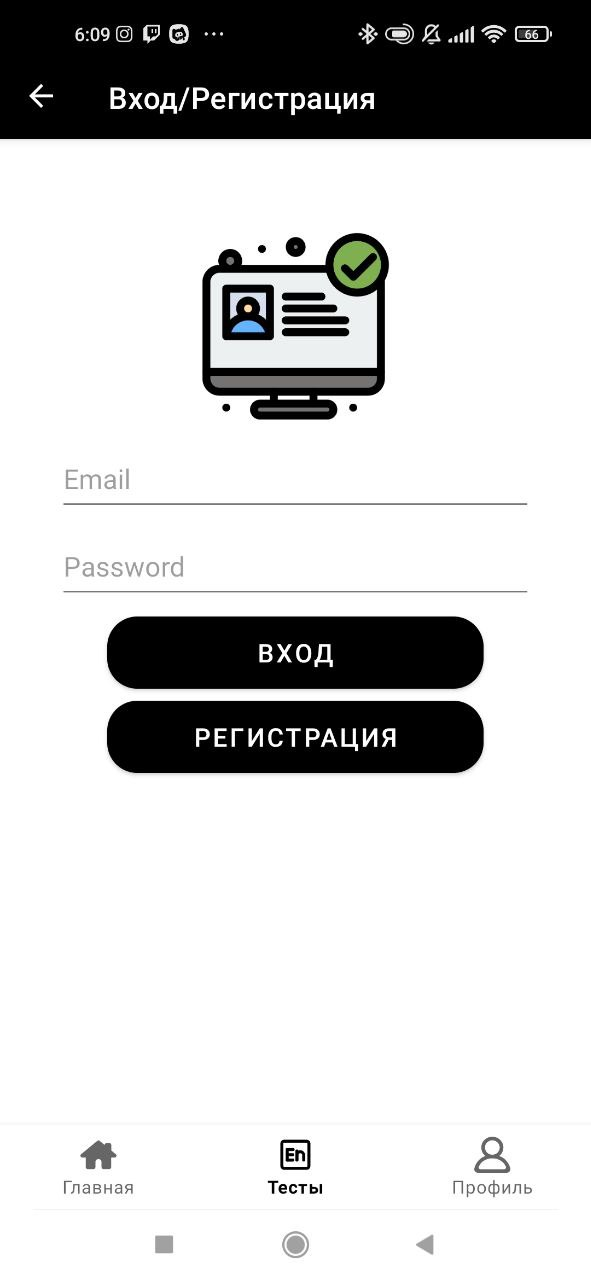


Рисунок 3 – Вход/Регистрация

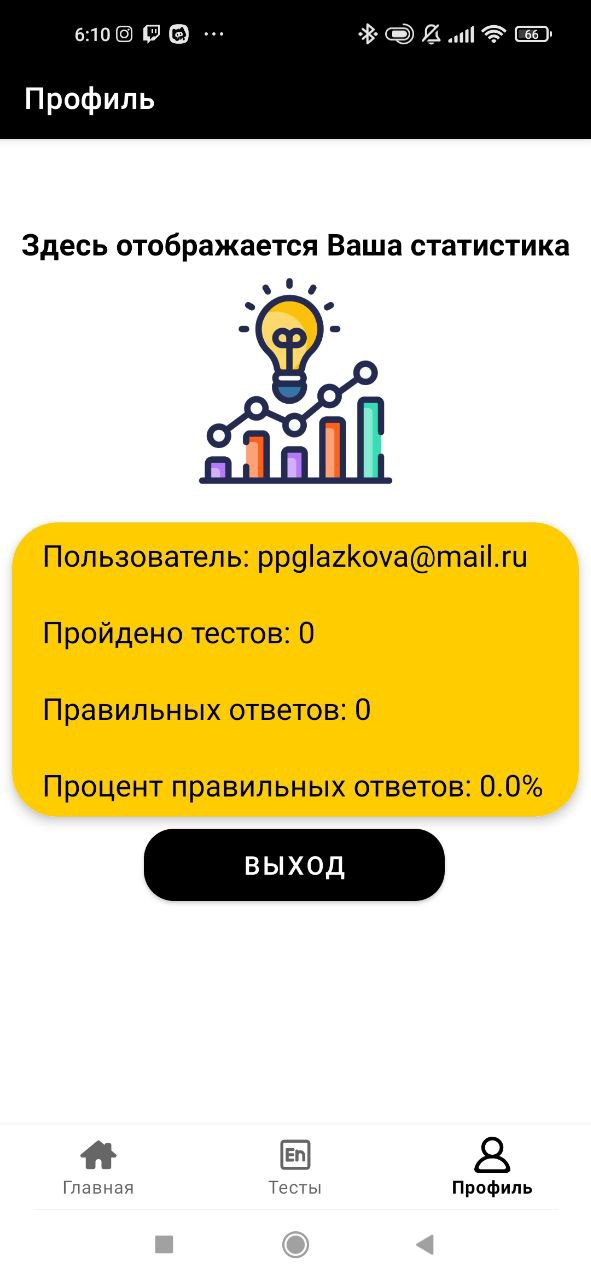


Рисунок 4 – Авторизированный профиль

Выход из приложения:

Нажмите кнопку «Назад» на вашем устройстве несколько раз, пока не вернётесь на главный экран, а затем снова нажмите «Назад», чтобы выйти из приложения.

Дополнительная информация:

Для использования приложения не требуется Интернет.

Есть возможность сохранить логин и пароль вашего профиля, чтобы не вводить их каждый раз при выходе.

Ошибки:

Если при регистрации ввести пароль 5 и меньше символов, то выйдет ошибка и профиль не будет создан.

Если попытаться войти в несуществующий аккаунт, то выйдет ошибка.