Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Системы обработки информации и управления

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

НА ТЕМУ:

Информационная система обучения английского языка

Студент ИУ5 – 82Б _____ Г.В.Пустовалов
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель ВКРБ _____ В.М.Постников
(Подпись, дата) (И.О.Фамилия)

Консультант _____ М.В.Черненький
(Подпись, дата) (И.О.Фамилия)

Нормоконтролер _____ Ю.Н.Кротов
(Подпись, дата) (И.О.Фамилия)

2025 г.

АННОТАЦИЯ

Расчётно-пояснительная записка квалификационной работы бакалавра содержит 52 страницы. С приложениями объём составляет 69 страниц. Работа включает в себя 23 таблицы и 17 иллюстраций. В процессе выполнения было использовано 25 источников.

Объектом разработки является мобильное приложение для изучения английского языка.

В работе рассматриваются архитектурные, технические и пользовательские аспекты разработки мобильного приложения, предназначенного для индивидуального изучения лексики с помощью карточек. Приложение позволяет пользователям создавать собственные тематические категории, добавлять и редактировать пары слов, отслеживать прогресс изучения, просматривать статистику и получать достижения. Особое внимание уделено реализации хранения данных с использованием Core Data [13], а также интеграции с облачным сервисом Firebase [12] для регистрации и авторизации пользователей. Для повышения отзывчивости интерфейса реализована поддержка многопоточности с использованием GCD и Swift Concurrency.

Целью работы является разработка мобильного приложения, которое упрощает процесс изучения английского языка, обеспечивает пользователям удобный и персонализированный инструмент для формирования и освоения индивидуальных словарей.

В процессе выполнения квалификационной работы бакалавра была изучена предметная область, проанализированы существующие аналоги, сформированы функциональные и технические требования, спроектирована архитектура системы, реализованы основные модули и проведено тестирование приложения [9]. Разработка велась с использованием языка Swift [10], архитектурных паттернов MVC и Clean Architecture [11], а также стандартных средств iOS-платформы.

Пояснительная записка содержит 2 приложения.

СОДЕРЖАНИЕ

СПИСОК СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В данной работе используются следующие сокращения и обозначения:

iOS – iPhone Operating System;

GCD – Grand Central Dispatch;

MVC – Model-View-Controller;

MVVM – Model-View-ViewModel;

MVP – Model-View-Presenter;

VIPER – View-Interactor-Presenter-Entity-Router;

SDK – Software Development Kit;

UI – User Interface;

UX – User Experience;

API – Application Programming Interface;

QoS – Quality of Service;

Swift – Язык программирования от Apple;

Xcode – Среда разработки для iOS/macOS;

ER – Entity-Relationship;

Clean Architecture – Архитектурный подход с разделением слоёв;

Core Data – Фреймворк Apple для локального хранения данных;

Firebase – Облачная платформа от Google;

UserDefaults – Механизм хранения простых настроек iOS.

ВВЕДЕНИЕ

Актуальность разработки обосновывается тем, что изучение иностранных языков становится все более востребованным в современном мире. Это связано с глобализацией, необходимостью международного общения, а также ростом карьерных и образовательных возможностей, которые открываются перед людьми со знанием английского языка. В то же время цифровые технологии проникают во все аспекты жизни, предоставляя удобные и эффективные инструменты для обучения.

Традиционные методы изучения языков, включающие использование бумажных учебников, списков слов и карточек, обладают рядом недостатков: они не всегда удобны, требуют постоянного доступа к физическим материалам и не предлагают интерактивности, что может снижать мотивацию учащихся. Современные мобильные приложения предоставляют уникальную возможность для автоматизации и повышения интереса пользователей за счет своей гибкости.

Особенно актуальной данная тема становится в условиях растущего спроса на мобильные решения, позволяющие учиться в удобное время и месте. Такие приложения играют важную роль в создании комфортной среды для самообразования, повышении эффективности усвоения материала и поддержании высокого уровня вовлеченности пользователей.

1 ПОСТАНОВКА ЗАДАЧ РАЗРАБОТКИ

1.1 Общетехническое обоснование разработки

1.1.1 Постановка задачи проектирования

Для организации современного, удобного и интуитивно понятного взаимодействия пользователя с мобильным приложением необходимо разработать программное решение, соответствующее актуальным требованиям к пользовательскому опыту и технической надёжности. В условиях постоянного роста интереса к индивидуальному обучению и саморазвитию, особенно в области изучения иностранных языков, ключевую роль играет создание инструмента, который не только предоставляет необходимые функции, но и адаптируется под личные предпочтения каждого пользователя.

Разрабатываемая система должна обеспечивать доступ к обучающим материалам в любое время и в любом месте, обладать высокой производительностью, простым и логичным пользовательским интерфейсом, а также надёжным механизмом хранения и обработки данных. Кроме того, особое внимание в рамках проекта уделяется персонализации обучения, которая достигается за счёт гибкой архитектуры, позволяющей пользователю подстраивать процесс под себя: от визуального оформления интерфейса до управления собственным учебным контентом.

Основной акцент в проектируемой информационной системе делается на методику активного запоминания лексики с использованием карточек, в сочетании с механизмом интервального повторения. Такой подход зарекомендовал себя как один из наиболее эффективных способов долговременного запоминания новых слов и выражений, поскольку он основан на повторении информации с оптимально подобранными временными интервалами, что способствует укреплению знаний в долговременной памяти.

Мобильное приложение должно предоставлять пользователю возможность регистрации и входа в систему, а также дальнейшую работу с собственным

учебным материалом: создание тематических категорий слов, добавление и редактирование карточек, проведение тренировок, в которых учитывается уровень прогресса по каждому слову. Это позволяет пользователю отслеживать динамику собственного обучения и своевременно возвращаться к трудным для запоминания элементам.

Дополнительно, система должна предоставлять средства для персонализации интерфейса, в том числе выбор темы оформления (светлая, тёмная, системная), а также возможность отслеживания личных достижений и статистики обучения. Такие элементы не только повышают удобство использования, но и играют важную роль в мотивации, стимулируя пользователя к регулярным занятиям и постепенному улучшению результатов.

1.1.2 Описание предметной области

Предметной областью разрабатываемой информационной системы является процесс индивидуального изучения английского языка с использованием мобильного программного обеспечения, функционирующего на современных мобильных устройствах под управлением операционной системы iOS. Основное назначение системы — обеспечение пользователю комфортного, доступного и эффективного средства освоения и повторения лексического материала посредством цифровых технологий. Особенность данной области заключается в акценте на интерактивное заучивание и систематическое повторение словарных единиц, а также в использовании проверенного в образовательной практике метода двусторонних карточек, с опорой на визуальные средства и игровые элементы, направленные на повышение вовлечённости.

Система ориентирована на широкую аудиторию — от начинающих до продвинутых пользователей, стремящихся к постепенному расширению словарного запаса в удобной и интуитивно понятной среде. В рамках предметной области реализована возможность создавать тематические категории (например, «Животные», «Бизнес», «Путешествия»), в каждую из которых добавляются

парные карточки с русским и английским переводом. Обучение осуществляется в формате интерактивных тренировок, где пользователь самостоятельно оценивает степень знания слова. Каждому слову присваивается параметр progressOfLearning, который отражает текущий уровень его усвоения и служит основой для адаптивной логики повторения.

Важной частью системы является фиксация пользовательской активности, которая в дальнейшем отображается в виде статистических графиков и системы достижений. Это позволяет не только визуализировать прогресс, но и стимулировать регулярные занятия, формируя привычку к постоянному обучению.

Структура системы реализована на основе набора логически связанных сущностей, каждая из которых играет определённую роль в процессе обучения:

- Пользователь (User) — основной субъект системы, обладающий уникальным идентификатором, а также персональными данными (имя, email), необходимыми для авторизации и персонификации контента.
- Категория (Category) — объединение лексических единиц по тематическому признаку; позволяет структурировать материал в соответствии с интересами и задачами пользователя.
- Слово (Word) — базовая обучающая единица, содержащая переводы на двух языках и уровень текущего усвоения.
- Достижение (Achievement) — механизм геймификации, позволяющий отмечать выполнение определённых условий (например, «Выучено 100 слов») и поощрять активность.
- Топ пять слов (TopFiveWords) — отдельная сущность, предназначенная для отображения последних пяти слов, которые ещё не были выучены. Это позволяет пользователю сфокусироваться на актуальных для повторения словах, повышая эффективность обучения.

Графическая структура предметной области представлена в приложении А1, в которой отражены основные сущности и взаимосвязи между ними. Описание сущностей базы данных, необходимые для работы приложения,

описаны в приложении А2. Сущности хранятся локально с использованием Core Data — встроенного фреймворка для управления объектной моделью. Для хранения пользовательской информации, авторизации и синхронизации используется облачная платформа Firebase [22]. Простые пользовательские настройки, такие как выбор темы оформления, сохраняются с помощью механизма UserDefaults [14], что позволяет поддерживать персонализированное взаимодействие даже после перезапуска приложения.

Таким образом, предметная область охватывает все основные компоненты процесса самостоятельного изучения английского языка: от создания и хранения лексических данных до визуализации прогресса и мотивации пользователя. Благодаря гибкой архитектуре и применению современных технологий, система представляет собой надёжный, расширяемый и удобный инструмент, способный адаптироваться под потребности различных категорий пользователей и способствующий эффективному языковому развитию.

1.1.3 Выбор критериев качества

Чтобы понять, насколько разрабатываемое приложение будет полезным и конкурентоспособным, важно сравнить его с существующими аналогами. Для этого я выбрал несколько основных критериев, которые, на мой взгляд, больше всего влияют на удобство и эффективность изучения английского языка:

1) Функциональная полнота;
2) Индивидуальная настройка;
3) Производительность
4) Готовые курсы
5) Отслеживание прогресса

Функциональная полнота: включает в себя наличие всех функций необходимых для эффективного изучения языка. Важен формат обучения (например, это могут быть: карточки или тесты), наличие заготовленных слов и выражений, фильтры и сортировка по темам, прогресс изучения слов и выражений, также критерий включает в себя удобство интерфейса.

Индивидуальная настройка: возможность персонализировать материал и интерфейс для комфортного изучения.

Производительность: скорость загрузки экранов приложения, плавные анимации, стабильность работы.

Готовые курсы: выбор и разнообразие готовых курсов или набора слов для изучения.

Отслеживание прогресса: наличие возможности просмотра статистики, прогресса или достижений пользователя для мотивации пользователя.

Назначим весовые коэффициенты обозначенным критериям, все критерии сравнения – качественные. Результаты приведены в таблице 1.

Таблица 1 - Критерии качества

№ п/п	Название критерия	Обозначение	Весовой коэффициент
1.	Функциональная полнота	K1	5 α
2.	Индивидуальная настройка	K2	4 α
3.	Производительность	K3	3 α
4.	Готовые курсы	K4	1 α
5.	Отслеживание прогресса	K5	2 α

Определим весовые коэффициенты критериев, используя метод балльной оценки и данные из таблицы 1.

Определим величину α, рассчитав значения весовых коэффициентов по формуле 1.

$$\sum_{i=1}^{n} \alpha_i = 1, \quad (1)$$

где n – общее количество весовых коэффициентов;

a_i – весовые коэффициенты.

Подставив значения в формулу (1), получаем:

$$5\alpha + 4\alpha + 3\alpha + 1\alpha + 2\alpha = 15\alpha$$

$$\alpha \approx 0{,}0667$$

1.1.4 Анализ прототипов и аналогов

Мобильное приложение "Duolingo"

"Duolingo" — одно из самых популярных приложений для изучения иностранных языков, предлагающее интерактивные уроки, упражнения и игровые элементы для увлекательного обучения. Оно подходит для пользователей всех уровней подготовки и поддерживает множество языков.

Приложение решает задачи создания удобного способа изучения языка, повышения мотивации через геймификацию и адаптации процесса под уровень пользователя. Среди функциональных возможностей: курсы по темам и уровням сложности, упражнения для развития навыков чтения, письма, аудирования и произношения, а также баллы, достижения и ежедневные задания для удержания интереса.

Пользователи могут отслеживать свой прогресс, видеть количество выученных слов и завершенных уроков, пользоваться графиками успехов. Приложение напоминает о занятиях и адаптирует обучение под цели и интересы.

Мобильное приложение "Lingualeo"

"Lingualeo" — это обучающая платформа для изучения английского языка, ориентированная на развитие словарного запаса и навыков восприятия языка. Приложение использует игровые элементы и персонализированный подход, чтобы сделать обучение эффективным и увлекательным.

Основные задачи: помощь пользователям в улучшении словарного запаса, грамматики и аудирования, а также создание удобной системы обучения для людей с разным уровнем подготовки.

Функционал включает словари с тематическими наборами, упражнения на аудирование и произношение, тренировки по грамматике и возможность изучать

новые слова с помощью ассоциаций и карточек. В приложении реализована система личных целей и напоминаний, которая помогает поддерживать регулярность занятий.

Мобильное приложение "Busuu"

"Busuu" — это универсальный инструмент для изучения иностранных языков, ориентированный на структурированное обучение с использованием уроков, тестов и общения с носителями языка. Поддерживает широкий спектр языков и предоставляет интерактивные упражнения для развития всех языковых навыков.

Основные задачи приложения включают систематизацию обучения, интеграцию общения с носителями и улучшение навыков через пошаговые уроки.

Ключевые функции: грамматические уроки, упражнения для пополнения словарного запаса, диалоги с носителями языка и сертификация уровня владения языком. Пользователи могут устанавливать цели обучения и отслеживать свой прогресс с помощью наглядной статистики.

Варианты систем приведены в таблице 2.

Таблица 2 - Варианты систем

Вариант	Обозначение
Информационная система обучения английского языка	B1
Duolingo	B2
LinguaLeo	B3
Busuu	B4

Представим качественные критерии [1], выделенные выше, в виде количественных, используя вербально-числовую шкалу [2] для того, чтобы корректно сделать расчеты. Результаты расчета переведены в таблицах 3-7.

Таблица 3 - Вербально-числовая шкала для критерия К1 (Функциональная полнота)

Качественная оценка	Количественная оценка
Реализованы все основные функции: карточки или тесты, заготовленные слова/выражения, фильтры и сортировка, прогресс изучения, удобный и интуитивный интерфейс.	1
Реализованы основные функции (карточки, заготовленные слова), частично реализованы вспомогательные (например, нет сортировки или прогресса), интерфейс в целом понятен.	0,7
Только базовые элементы (например, карточки без фильтров и отслеживания), интерфейс требует доработки.	0,3
Большинство необходимых функций отсутствуют, интерфейс перегружен или неудобен.	0

Таблица 4 - Вербально-числовая шкала для критерия К2 (Индивидуальная настройка)

Качественная оценка	Количественная оценка
Пользователь может полностью настраивать учебный материал, добавлять слова, категории слов, адаптировать интерфейс.	1
Частичная настройка (например, можно добавлять слова, но нет группировки по категориям или отсутствие выбора режима обучения).	0,7
Настройка ограничена (только выбор готовых курсов, без добавления своих данных).	0,3
Настроек практически нет.	0

Таблица 5 - Вербально-числовая шкала для критерия К3 (Производительность)

Качественная оценка	Количественная оценка
Мгновенная загрузка данных, не нужно ждать ответа сети для основных задач.	1
В целом быстрая работа, незначительные задержки.	0,7
Задержки при работе с основным функционалом приложения.	0,3
Частые сбои, долгая загрузка, ощутимые лаги.	0

Таблица 6 - Вербально-числовая шкала для критерия К4 (Готовые курсы)

Качественная оценка	Количественная оценка
Есть разнообразные курсы, разбитые по уровням и темам.	1
Есть несколько курсов общего назначения.	0,7
Доступен только один базовый курс.	0,3
Курсы отсутствуют.	0

Таблица 7 - Вербально-числовая шкала для критерия К5 (Отслеживание прогресса)

Качественная оценка	Количественная оценка
Подробная статистика, графики, уведомления, уровни	1
Отображается базовая статистика (например, количество слов)	0,7
Прогресс отслеживается частично или формально	0,3
Нет отслеживания прогресса	0

Определим параметры рассматриваемых программ в таблице 8.

Таблица 8 - Количественные и качественные параметры расширений

Код критерия	Критерий	Значение критерия варианта библиотеки			
		В1	**В2**	**В3**	**В4**
К1	Функциональная полнота	1	0,7	0,7	1
К2	Индивидуальная настройка	0,7	1	0,3	0,3
К3	Производительность	1	0,7	0,7	0,7
К4	Готовые курсы	0,3	1	0,7	1
К5	Отслеживание прогресса	1	0,7	0,7	1

С помощью Парето-оптимальности [3] проведем оценку исходных вариантов в таблице 9.

Таблица 9 - Сравнение вариантов на Парето-оптимальность

Вариант расширения	**Вариант расширения**			
	B1	**B2**	**B3**	**B4**
B1	0	0	1	0
B2	0	0	0	0
B3	0	0	0	0
Результат сравнения	0	0	1	0
Парето-оптимальность	Да	Да	Нет	Да

Варианты B1, B2 и B4 являются Парето-оптимальными, поскольку не доминируются другими. Вариант B3 не входит в множество Парето-оптимальных, так как вариант B1 доминирует над ним по всем критериям.

Таблица 10 – Вес критериев.

Критерий	**Коэффициент важности локального критерия (α_i)**
K1	5 × 0,0667 = 0,3335
K2	4 × 0,0667 = 0,2668
K3	3 × 0,0667 = 0,2001
K4	1 × 0,0667 = 0,0667
K5	2 × 0,0667 = 0,1334

Рассчитаем значения интегральных критериев для каждого варианта (приложения) с помощью метода взвешенной суммы критериев [4], используя формулу (2) и таблицу 10:

$$Y_j = \sum_{i=1}^{n} a_i * k_{ij}, \qquad (2)$$

где k_{ij} – коэффициент важности j варианта по i локальному критерию;

a_i – весовые коэффициенты.

Посчитаем значение интегрального критерия для В1, В2 и В4 по формуле (2):

Вариант В1:

$$Y_1 = (1{,}0 * 0{,}3335) + (0{,}7 * 0{,}2668) + (1.0 * 0{,}2001) + (0{,}3 * 0{,}0667) + (1.0 * 0{,}1334) = 0{,}87377$$

Вариант В2:

$$Y_2 = (0{,}7 * 0{,}3335) + (1.0 * 0{,}2668) + (0{,}7 * 0{,}2001) + (1.0 * 0{,}0667) + (0{,}7 * 0{,}1334) = 0{,}8004$$

Вариант В4:

$$Y_4 = (0{,}7 * 0{,}3335) + (0{,}3 * 0{,}2668) + (0{,}7 * 0{,}2001) + (1.0 * 0{,}0667) + (1 * 0{,}1334) = 0.65366$$

Таблица 11 - Значения локальных критериев методом взвешенной суммы

Код критерия	**Коэффициент важности локального критерия (α_i)**	**Значения локальных критериев**		
		В1	**В2**	**В4**
К1	0,3335	1	0,7	1
К2	0,2668	0,7	1	0,3
К3	0,2001	1	0,7	0,7
К4	0,0667	0,3	1	1
К5	0,1334	1	0,7	1
$Y_j = \sum_i^n \alpha_i \cdot k_{ij}$		0,87377	0,8004	0,65366
$Y_j = max \sum_i^n \alpha_i \cdot k_{ij}$		0,87377 (*)		

(*) – Вариант В1 – наилучший вариант согласно критерию интегральному критерию: взвешенной суммы локальных критериев.

На основе метода Парето-оптимальности только B1, B2 и B4 являются допустимыми. Вариант B3 исключается из множества эффективных решений. Метод взвешенной суммы показывает, что наибольший интегральный показатель (0,87377) у собственного приложения (B1). Это подтверждает актуальность разработки и её преимущество над аналогами по совокупности ключевых критериев.

2 КОНСТРУКТОРСКО-ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

2.1 Конструкторская часть

2.1.1 Функциональные требования

Разрабатываемая система должна выполнять следующие функции:

- Регистрация и авторизация пользователя

a) Регистрация пользователя по email и паролю;

b) Авторизация с использованием сохранённого токена доступа;

- Работа с категориями слов

a) Создание пользовательских категорий для слов (тематических наборов);

b) Удаление существующих категорий;

c) Выбор изображения и названия при создании категории;

d) Просмотр всех категорий с прогрессом изучения.

- Работа со словами внутри категории

a) Добавление карточек с парами "русский — английский";

b) Удаление карточек из категории;

c) Автоматический перевод слов при добавлении (с использованием API);

d) Визуализация прогресса изучения по каждой карточке.

- Режим тренировки слов

a) Отображение карточек с одной стороны (английский или русский);

b) Переворот карточки для просмотра перевода;

c) Отметка "Помню" / "Забыл" с соответствующим увеличением или уменьшением прогресса;

- Персонализация и интерфейс

a) Настройка темы интерфейса (светлая / тёмная / системная);

b) Изменение аватара пользователя;

c) Просмотр информации об аккаунте (логин, email, прогресс).

- Отслеживание прогресса

a) Отображение общего прогресса пользователя по всем категориям;

b) Достижения пользователя, на основе активности в приложении;

c) Статистика соотношения изученных и невыученных слов.

2.1.2 Описание интерфейса

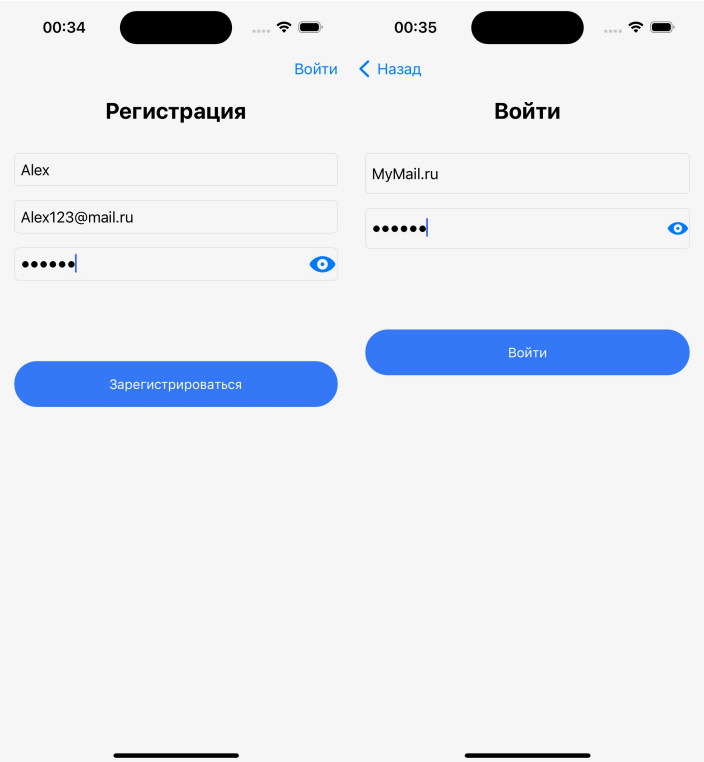


Рисунок 1 – Экран регистрации и авторизации

Регистрация и авторизация пользователей, представлены на рисунке 1. Данные функции являются основными функциональными задачами любого современного мобильного приложения, имеющих уникальный контент для каждого пользователя.

Пользователи смогут зарегистрироваться в приложении через почту. Далее им необходимо будет придумать и написать логин и пароль для своего аккаунта. Если пользователь все сделает правильно, то его аккаунт будет зарегистрирован в системе.

Чтобы войти в свой аккаунт, пользователю необходимо будет ввести свои учетные данные (почту и пароль). Если данные совпадают с записями в базе, то доступ предоставляется. Для удобства пользователей будет реализован

автоматический вход, который использует токен авторизации, сохраняющийся на устройстве.

Регистрация пользователей необходима, ведь учетная запись позволяет сохранять созданные категории, слова, персональные настройки.

Функционал регистрации и авторизации является неотъемлемой частью успешного приложения. Он помогает обеспечить безопасность, предоставить пользователям персонализированный опыт и увеличить их лояльность. При этом важно учитывать удобство и безопасность, чтобы минимизировать возможные недостатки.

Экран просмотра категорий пользователя

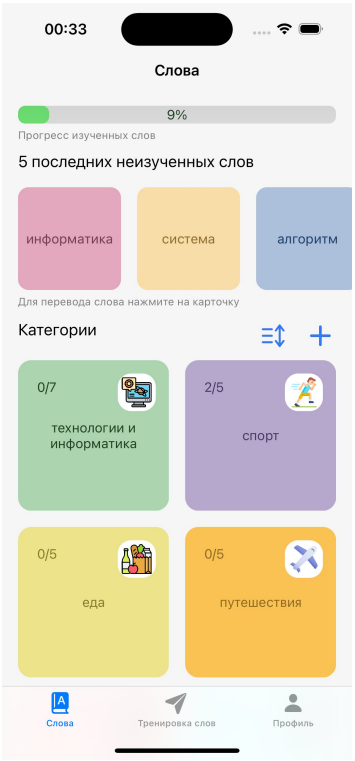


Рисунок 2 – Экран просмотра категорий пользователя

Экран просмотра категорий пользователя, представленный на рисунке 2, отображает список всех категорий, созданных пользователем. Он обеспечивает быстрый и удобный доступ к каждому разделу словаря, сгруппированному по темам, и позволяет визуально оценить прогресс изучения внутри каждой категории.

Основные элементы экрана:

– Прогресс-бар: Визуальное отображение прогресса изучения в виде соотношения выученных и невыученных слов помогает отслеживать успехи.

– Последние невыученные слова: На экране представлены 5 последних невыученных слов, чтобы пользователь мог сразу приступить к повторению.

– Список категорий: Каждая категория отображается с названием, иллюстрацией и количеством выученных и невыученных слов.

Экран просмотра категорий служит центром управления процессом обучения, предоставляя пользователю всю необходимую информацию для планирования и контроля своего прогресса.

Экран создания категории

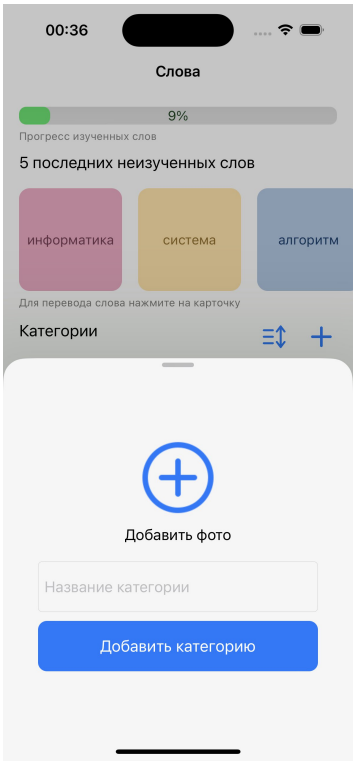


Рисунок 3 – Экран создания категории

Экран создания категории, представленный на рисунке 3, позволяет пользователю создать категорию для изучения слов.

Основные элементы экрана:

– Выбор изображения: Пользователь может выбрать картинку для визуального представления категории, что упрощает поиск и улучшает восприятие интерфейса.

– Поле для ввода названия: Предусмотрено текстовое поле, где пользователь может ввести название категории, чтобы она отражала тему или смысл добавляемых слов.

Экран создания категории обеспечивает гибкость и интуитивность, позволяя настроить категории в соответствии с предпочтениями и потребностями пользователя.

Экран списка слов в категории пользователя

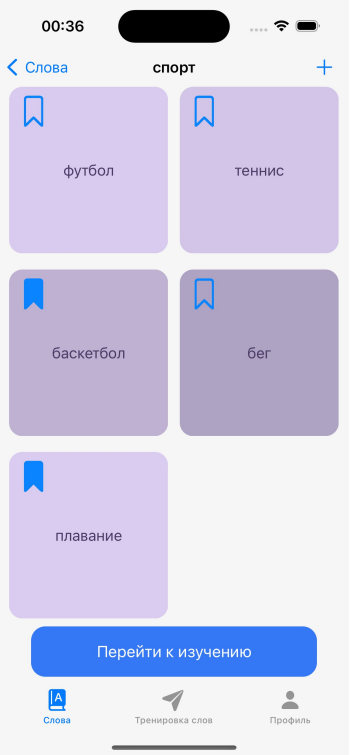


Рисунок 4 – Экран списка слов в категории пользователя

Экран списка слов в категории пользователя, представленный на рисунке 4, предоставляет пользователю доступ ко всем словам, относящимся к выбранной категории, и позволяет управлять процессом их изучения.

Основные элементы экрана:

– Список слов: каждое слово отображается в отдельной ячейке. При нажатии на ячейку слово переворачивается, показывая его перевод на английском языке, при повторном нажатии перевод меняется обратно на русский язык.

– Кнопка "Перейти к изучению": открывает режим карточек, где пользователь может начать тренировку слов, относящихся к категории.

Экран списка слов в категории помогает пользователю быстро просматривать слова и переходить к их изучению.

Экран добавления слов в категорию пользователя

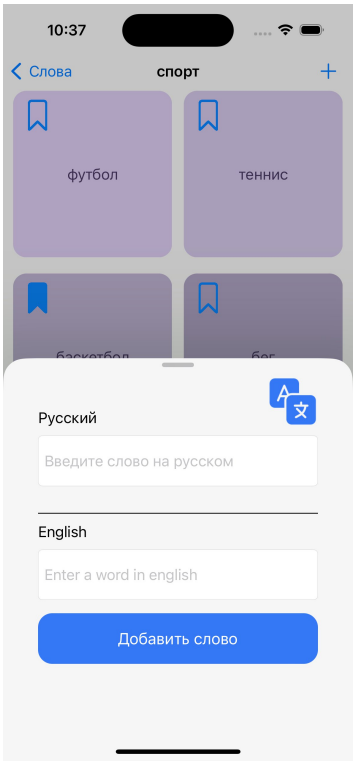


Рисунок 5 – Экран добавления слов в категорию пользователя

Экран добавления слов в категорию пользователя, представленный на рисунке 5, позволяет пользователю пополнять выбранную категорию новыми словами, обеспечивая гибкость и персонализацию процесса обучения.

Основные элементы экрана:

– Поле для ввода на русском языке. Поле для ввода перевода слова.

– Поле для ввода слова на английском: пользователь вводит слово, которое нужно добавить в категорию.

– Кнопка перевода: после ввода слова на одном из языков можно нажать на кнопку с переводом и после выполнения запроса получим перевод этого слова на другом языке.

– Кнопка "Добавить слово":

– После заполнения полей пользователь может нажать кнопку для добавления слова в текущей категории.

Экран добавления слов обеспечивает простой и интуитивный процесс, позволяя пользователю самостоятельно формировать словарь для каждой категории и эффективно расширять базу для изучения.

Экран изучения и освоения новых слов

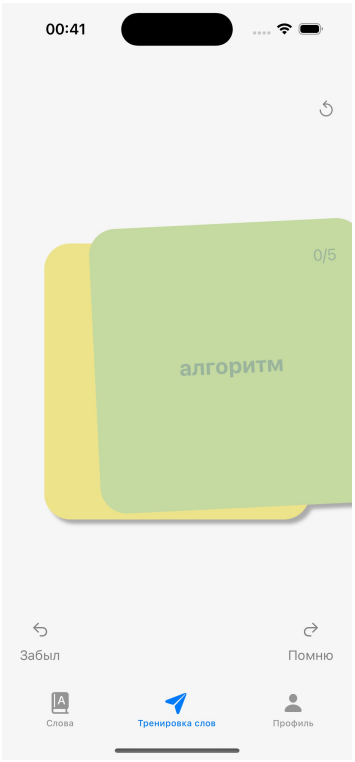


Рисунок 6 – Экран изучения и освоения новых слов

Экран изучения и освоения новых слов, представленный на рисунке 6, предоставляет пользователю возможность тренировать невыученные слова через карточки, обеспечивая эффективное запоминание и повторение материала.

Основные элементы экрана:

– Карточка со словом: На одной стороне карточки отображается слово на английском языке, а на другой — перевод на русский язык. По нажатию пользователь может перевернуть карточку, чтобы увидеть перевод, и таким образом проверить свои знания.

– Кнопки для взаимодействия: "Забыл" и "Помню": при нажатии на одну из кнопок карточка перемещается в соответствующую категорию для дальнейшего повторения. Также можно карточку "перетянуть" в соответствующую зону.

Также важно отметить, что слово считается выученным, если у него прогресс изучения равен 5. При нажатии на кнопку "Помню", счетчик увеличивается на 1, а при нажатии на "Забыл" счетчик уменьшается на 1, но значение не может быть менее 0.

Экран профиля пользователя

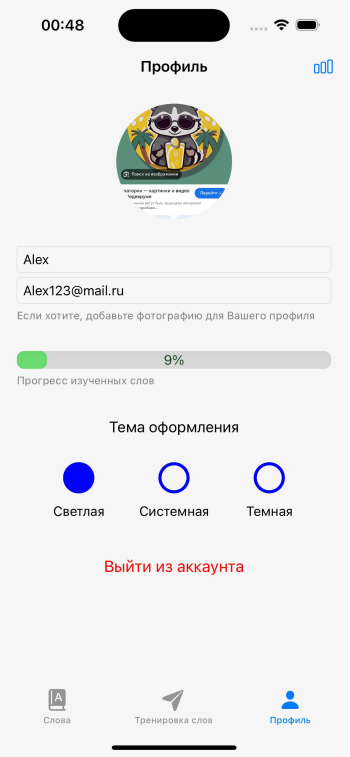


Рисунок 7 – Экран профиля пользователя

Экран профиля пользователя, представленный на рисунке 7, предоставляет пользователю персонализированную информацию о его аккаунте, прогрессе и настройках, создавая комфортную среду для дальнейшего обучения.

Основные элементы экрана:

– Фото профиля: отображается аватар пользователя, который можно изменить.

– Логин и почта

– Прогресс изучения

– Выбор темы: пользователь может выбрать предпочитаемую тему интерфейса: светлую, темную или системную (в зависимости от настроек устройства).

Экран профиля помогает пользователю следить за своими успехами и предоставляет возможность персонализировать приложение под свои предпочтения, делая взаимодействие с ним более удобным и приятным.

Экран достижений пользователя в освоении слов английского языка

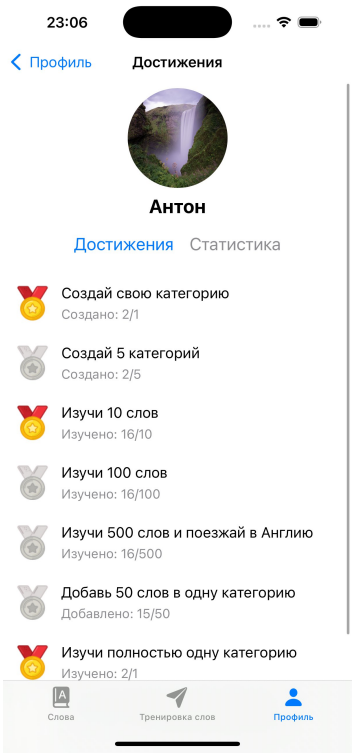


Рисунок 8 – Экран достижений пользователя в освоении слов английского языка

Экран достижений пользователя в освоении слов английского языка, представленный на рисунке 8, предназначен для отображения списка достижений, которые пользователь может получить в процессе использования приложения. Он визуализирует успехи, формирует чувство прогресса и стимулирует продолжать обучение, выполняя при этом важную мотивационную функцию.

Каждое достижение представлено в виде карточки с иконкой, кратким описанием и статусом (получено / не получено). Среди достижений могут быть, например: «Создай свою категорию», «Изучи 100 слов», «Изучи полностью одну

категорию» и другие. При получении достижения может отображается цветное изображение.

Этот экран способствует геймификации учебного процесса и повышает вовлечённость пользователя в регулярные занятия.

Экран статистики пользователя

Рисунок 9 – Экран статистики пользователя

Экран статистики пользователя, представленный на рисунке 9, предназначен для визуального отображения учебного прогресса пользователя с помощью диаграмм и графиков. Он помогает быстро оценить уровень активности, а также прогресс по изучению слов и категорий.

На экране отображаются:

– Столбчатая диаграмма, отражающая количество созданных пользователем категорий относительно слов в категории

– Кольцевая диаграмма (или круговая), показывающая соотношение между выученными словами и словами в процессе изучения.

Статистика обновляется автоматически на основе действий пользователя и позволяет отслеживать динамику обучения в реальном времени. Такой подход

способствует самоконтролю и помогает корректировать учебную стратегию. Благодаря наглядности и простоте восприятия, экран статистики является важным компонентом пользовательского интерфейса, повышающим осознанность и эффективность изучения языка.

2.1.3 Выбор технологий создания системы

Для разработки мобильного приложения был выбран нативный подход с использованием iOS UIKit, , поскольку он предоставляет полный доступ к экосистеме Apple и позволяет создавать приложения, полностью соответствующие гайдлайнам платформы. Нативные инструменты от Apple [6], включая UIKit , SwiftUI [23], [24] и другие фреймворки, гарантируют стабильность, производительность и актуальную документацию, что делает их надежным выбором для iOS-разработки [15], [18], [19].

Кроме того, в качестве альтернативы UIKit существует язык программирования Objective-C, который долгое время был основным инструментом для создания приложений для iOS [7]. Однако, с учетом современных тенденций, Swift [5] и UIKit являются более предпочтительными для разработки новых приложений. Swift [20] обладает современным синтаксисом, высокой производительностью и улучшенной безопасностью благодаря автоматическому управлению памятью, что делает его более удобным и эффективным для разработки приложений.

Кроссплатформенные инструменты, такие как React Native, Flutter и Kotlin Multiplatform Mobile, имеют свои преимущества, включая экономию времени и ресурсов при создании приложений для iOS и Android. Однако у них есть и недостатки:

– React Native: использует JavaScript, но интерфейс состоит из WebView, что может снизить нативность пользовательского опыта.

– Flutter: создает интерфейсы с использованием виджетов на языке Dart, что требует имитации нативных элементов, усложняя интеграцию с платформой.

– Kotlin Multiplatform Mobile: Новый инструмент от Google, который пока еще недостаточно зрел для коммерческого использования.

Выбор в пользу UIKit и Swift обоснован необходимостью высокого качества пользовательского интерфейса, производительности и глубокого взаимодействия с платформенными API. UIKit позволяет создавать приложения с использованием нативных элементов интерфейса, обеспечивая максимальную совместимость с устройствами iOS и улучшенный пользовательский опыт.

Критерии сравнения языков программирования приведены в таблицах 12-16:

Таблица 12 - Поддержка ios платформы

Качественная оценка	**Количественная оценка**
Отсутствует поддержка iOS	1
Поддержка iOS есть, но не полная или устаревшая	2
Полная и актуальная поддержка iOS	3

Таблица 13 - Гибкость настройки интерфейса (UI)

Качественная оценка	**Количественная оценка**
Ограниченные возможности настройки интерфейса, нет нативности	1
Средние возможности, требуются дополнительные библиотеки, нет нативности	2
Высокая гибкость, легко настраиваемый интерфейс	3

Таблица 14 - Производительность

Качественная оценка	Количественная оценка
Низкая производительность, много ограничений	1
Средняя производительность, возможны улучшения	2
Высокая производительность и оптимизация	3

Таблица 15 - Скорость разработки

Качественная оценка	Количественная оценка
Долгий процесс разработки, много проблем	1
Умеренная скорость разработки	2
Быстрая разработка, наличие множества инструментов	3

Таблица 16 - Поддержка оффлайн-функционала

Качественная оценка	Количественная оценка
Нет или ограниченная поддержка	1
Поддержка ограничена, требует дополнительных усилий	2
Полная и удобная поддержка оффлайн-функционала	3

В результате, сформировав критерии сравнения языков программирования, можем выбрать наилучший вариант для разработки системы.

Таблица 17 – Сравнение языков программирования

	Языки программирования				**Весовые коэффициенты**
Критерии сравнения языков программирования	**Swift**	**Objective-C**	**React Native**	**Flutter**	
Поддержка ios платформы	3	3	2	2	0,2
Гибкость настройки интерфейса (UI)	3	2	2	2	0,2
Производительность	3	3	1	1	0,2
Скорость разработки	2	1	2	2	0,2
Масштабируемость	3	3	1	2	0,1
Поддержка оффлайн-функционала	3	3	2	2	0,1
Итоговая оценка $y_j = \sum_{\{i=1\}}^{n} \alpha_{i^*} \ k_{\{ij\}}$	2,8	2,4	1,4	1,8	

2.1.4 Выбор архитектуры

Перед началом проектирование программного обеспечение необходимо тщательно проработать архитектуру будущего мобильного приложения, так как для построение платформы, которая способна создавать уникальные мобильные приложения для различных станций технического обслуживания автомобилей

требуется крепкая основа, в фундаменте которой как раз и будет находится хорошо спроектированная архитектура.

Для создания архитектуры мобильного приложения буду руководствоваться принципами Clean Architecture [8]. Основная задача - обеспечить достаточных уровень абстракции, что позволяет создавать слабо зависимые модули приложения, которые достаточно хорошо поддаются тестированию и удобны в плане дальнейшей поддержки, а также позволяют с минимальными изменениями вне модулей изменять стек технологий в них применяющийся. Общая схема подобного рода архитектуры представлена на рисунке 10:

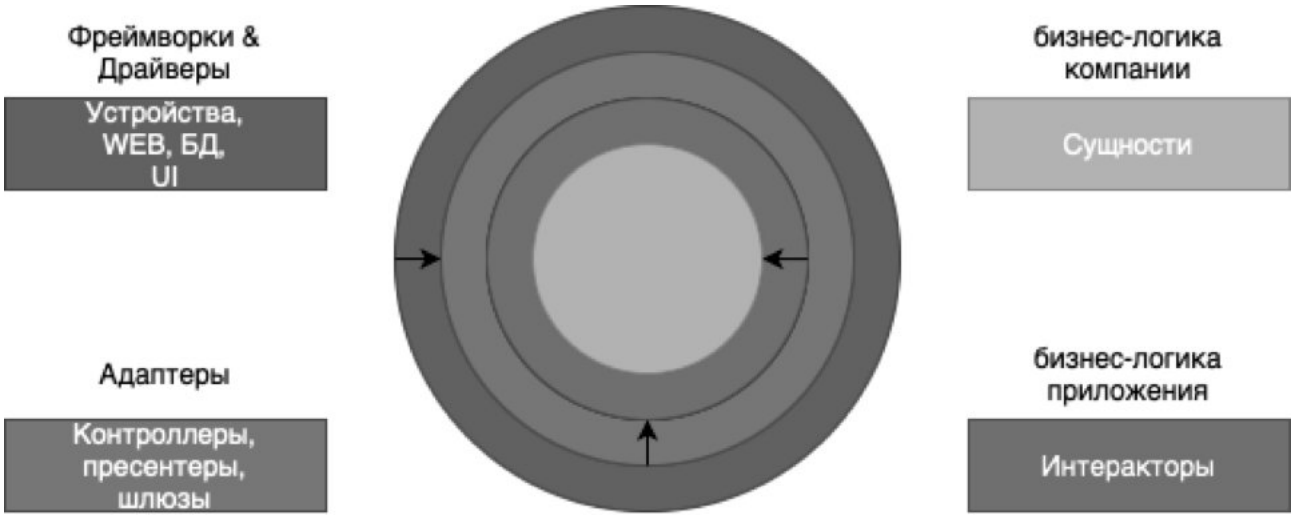


Рисунок 10 – Схема Clean Architecture

Среди основных архитектур, которые применяются при разработке мобильных приложений под платформу iOS можно выделить следующие:

1) MVC – (Model-View-Controller)
2) MVP - (Model-View-Presenter)
3) MVVM - (Model-View-ViewModel)
4) VIPER - (View, Interactor, Presenter, Entity, Router)
5) Clean Swift (VIP) - View, Interactor, Presenter (вариация VIPER)
6) Redux / Flux.

MVC или же Model View Controller [16] является одной из самых распространенных архитектур для платформы iOS. Эта архитектура раньше

применялась компанией Apple при разработки своих приложений. Схема данной архитектуры представлена на рисунке 11:

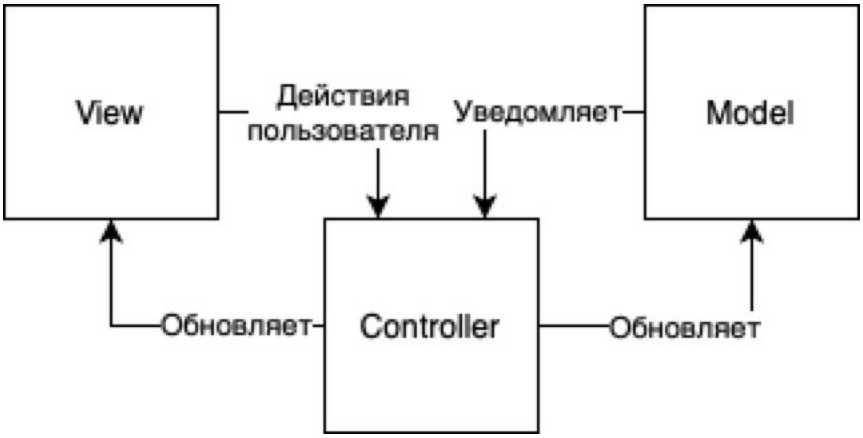


Рисунок 11 – Схема MVC

Классическое приложение будет состоять из нескольких представлений, включающих в себя как целые экраны, так и отдельные компоненты пользовательского интерфейса, а также окна или же веб-страницы. Итак, каждое представление условно делится на три составляющие - модель, которая содержит в себе данные и бизнес логика для того, чтобы с ними взаимодействовать, View, являющееся графическим представлением для данных из модели и отображается на экране, именно с этим слоем взаимодействует пользователь, а также контроллер - он занимается обработкой, а также валидацией при необходимости, пользовательского ввода и передачей намерений к модели. Все это представляет из себя классический MVC [17].

MVP представляет из себя дальнейшее развитие MVC. Причиной его возникновения стало сложность в тестировании классического MVC, поэтому для облегчения данной задачи была создана новая архитектура, именуемая Model View Presenter, схема которой представлена на рисунке 12:

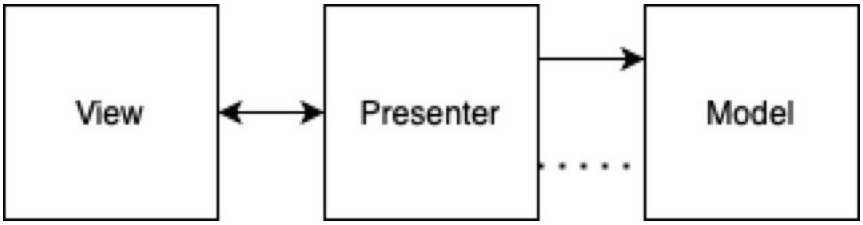


Рисунок 12 – Схема MVP

В случае классического MVP модуля модель будет хранить данные, отображение будет графической частью модуля, которая будет воспроизводиться на экране пользователя, и Presenter - к которому отображение идет за данными и перенаправляет в него пользовательские действия. Благодаря нему реализуется взаимодействие между Model и View.

Архитектура MVVM изначально разработана корпорацией Microsoft для своих Reactive eXtentions. Для ее реализации на платформе требуется наличие какого-либо механизма привязки различных компонентов. Данная архитектура требует от разработчика навыков работы с реактивным стилем программирования.

Чтобы использовать данную архитектуру на платформе iOS требуется ряд библиотек, без которых подобного рода разработка будет хотя и не невозможна, но значительно затруднена. Для языка программирования Objective-C была создана Reactive Cocoa, которая в последствии расширена в библиотеке RxSwift, представляющей собой реализацию паттернов реактивных расширений для языка программирования Swift. В настоящий момент именно она получила наибольшее распространение. А также существует инструмент, созданный именно компанией Apple - фреймворк Combine. Он представляет из себя все тот же поставщик реактивного функционала для iOS приложения, но при этом более ориентирован на особенности данной платформы.

Итак, в отличие от двух описанных до этого архитектурах мобильного приложения для Model View ViewModel не характерно прямое взаимодействие модели и отображения. Для данного подхода предусмотрена технология прямой привязки данных к представлению, за счет этого при любых изменениях в данных, отображение будет моментально об этом уведомлено и переименовано, основываясь на новых данных.

Итак, в слое модели по аналогии с другими, описанными ранее архитектурами, также содержатся данные и бизнес логика. Отображение представляет из себя весь графический интерфейс приложения. Оно

подписывается на события обновления полей, в которых хранятся данные, необходимые для отображения на экране, а также на любые другие необходимые события из вью модели. Вью модель представляет из себя одновременно абстракцию для отображения и обертку для модели.

Для архитектуры VIPER каждый модуль приложения дробиться на пять сущностей, представляющих из себя:

View - как и в других архитектурах представляет все, что связанно непосредственно с отрисовкой на пользовательском экране.

Interactor представляет из себя часть модуля и реализует бизнес логику.

Presenter служит для подготовки данных для отображения.

Data Store хранит в себе Entities: сущности данных, которые по запросу предоставляет в Interactor. Обычно является хранилищем данных, которое на платформе iOS может быть реализовано посредством таких технологий как CoreData, SQLite или Realm.

Router - служит для перехода между экранами.

Таким образом, любой объект в модуле попадает в одну из описанных выше категорий с определенным набором задач.

Схема данной архитектуры представлена на рисунке 13.

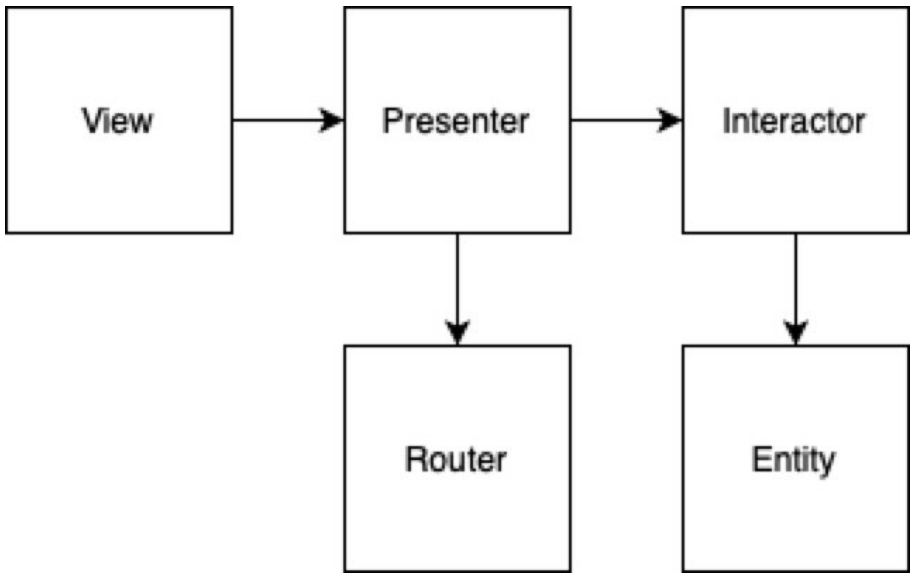


Рисунок 13 – Схема Viper

Clean Swift представляет из себя альтернативу Viper и на первый взгляд очень похож на него. Так, в clean swift взаимодействие между компонентами модуля происходит циклично и однонаправленно. Передача данных основана на протоколах, что в дальнейшем при изменении компонентов системы позволяет осуществить безболезненный переход.

Пример работы модуля можно описать на примере нажатия на кнопку. Сначала пользователь нажал на кнопку вью, контроллер генерирует объект с описанием того, что нажато и что хочет пользователь, и передает его в Interactor. Interactor, отвечающий за обработку логики независимой от пользовательского интерфейса, позволяет решить, что делать с этим запросом дальше. Он может как обратиться на backend с целью обработки запроса или же к внутренней базе данных. Затем полученные данные передаются в Presenter. Он формирует отформатированный объект с данными, которые будут понятны View и затем передает их в Controller.

Controller отвечает за все манипуляции со View, будь то создание, настройка, установка шрифта или верстки. Каждый контроллер в данной архитектуре принимает Input протокол для отображения данных.

Interactor принимает действия пользователя от контроллера с параметрами, такими как новый текст введенных пользователем в поле ввода и прочее, определенными в Input протоколе. После выполнения действий с логикой он должен передать поля готовые для отображения в Presenter.

Presenter обрабатывает данные, готовые для показа в контроллере. Тут, к примеру может быть изменен формат текста или же значения цвета и прочее.

Worker представляет из себя компонент, служащих для упрощения логики работы Interactor. Довольно часто Clean Swift модуль и вовсе может обойтись без данного компонента. В Worker’ы обычно выносится логика взаимодействия с базой данных или же однотипные запросы, которые могут быть использованы в разных модулях.

Router отвечает за переходы и передачу данных между различными модулями. Обычно он хранит в себе Controller, поскольку исторически именно

он отвечал за переходы. Данные между модулями передаются, используя протоколы хранилища каждого отдельного модуля, между которыми происходил переход.

Модели представляет из себя структуры данных для передачи между компонентами и включают в себя Request — для передачи запроса из контроллера в Interactor. Response — ответ Interactor'а для передачи в Presenter. ViewModel — для передачи данных в готовом для отображения виде в Controller. Схема данной архитектуры представлена на рисунке 14.

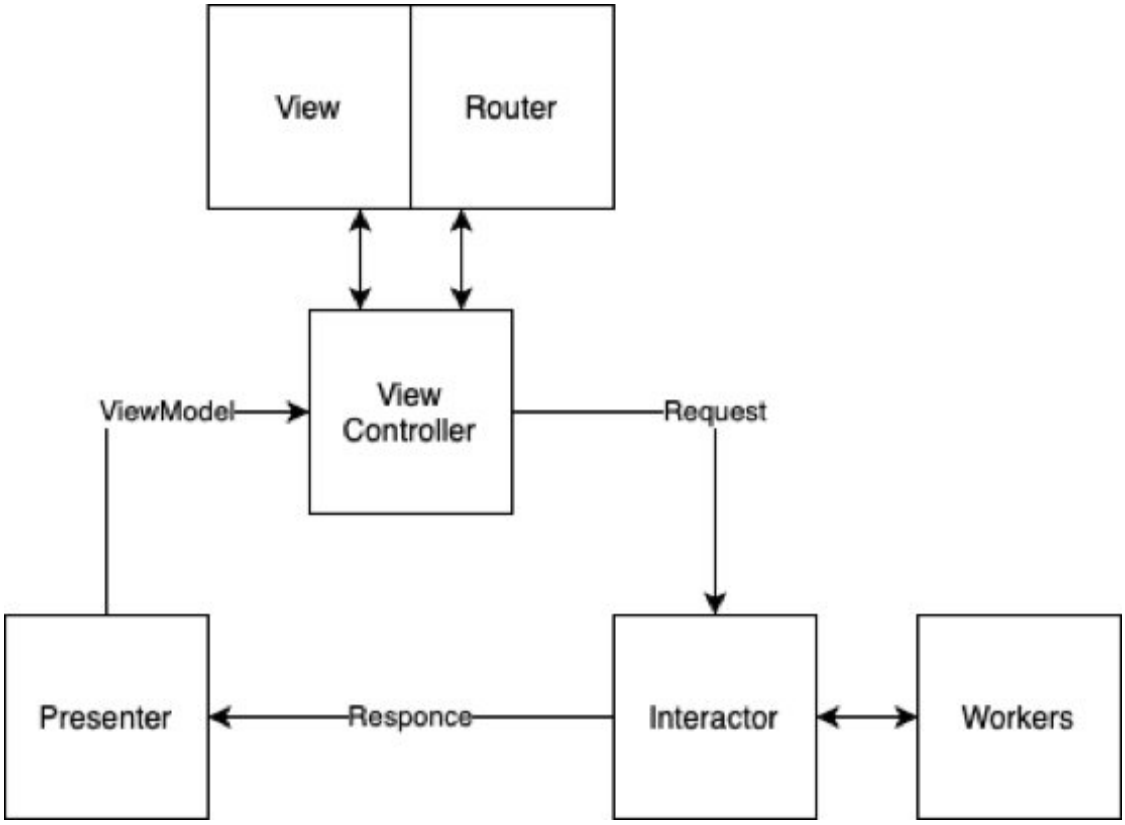


Рисунок 14 – Схема Clean Swift (VIP)

Обоснование выбора архитектурных шаблонов: MVC и Clean Swift

При разработке мобильного приложения для изучения английского языка был выбран гибридный подход, сочетающий два архитектурных шаблона — MVC (Model–View–Controller) и Clean Swift (VIP). Основными причинами такого выбора стали:

– Скорость разработки: архитектура MVC предоставляет простую и понятную структуру, что позволяет быстро реализовать базовую

функциональность и прототип интерфейса. Это критично на ранних этапах проекта, когда важно получить работающий результат в сжатые сроки.

– Опыт работы: в процессе предыдущей разработки был накоплен значительный опыт работы как с MVC, так и с Clean Swift. Это позволило максимально эффективно применять оба подхода на практике, избегая типичных ошибок и ускоряя процесс реализации бизнес-логики.

– Разделение ответственности: Clean Swift (VIP) обеспечивает строгую модульность и изоляцию компонентов, что упрощает поддержку и масштабирование кода. Использование Interactor, Presenter и Router позволяет эффективно обрабатывать пользовательские действия, форматировать данные для представления и управлять переходами между экранами.

Комбинирование MVC и Clean Swift позволило найти баланс между простотой реализации и качеством архитектурных решений, обеспечив как быструю разработку, так и хорошую масштабируемость проекта.

2.1.5 Выбор технологий многопоточности

Многопоточность является неотъемлемой частью разработки мобильных приложений, особенно в тех случаях, когда требуется обеспечить высокую отзывчивость интерфейса, быструю обработку данных и асинхронное взаимодействие с сетью. Современные мобильные приложения всё чаще сталкиваются с необходимостью выполнять ресурсоёмкие задачи (сетевые запросы, обработка данных, работа с БД) без блокировки пользовательского интерфейса. Для решения таких задач в экосистеме iOS предусмотрено несколько механизмов многопоточности, среди которых ключевыми являются Grand Central Dispatch (GCD) и Swift Concurrency. Оба подхода активно используются в разработке исследуемого приложения для изучения английского языка.

Grand Central Dispatch

Grand Central Dispatch (GCD) — это низкоуровневый API, предоставляемый Apple для управления асинхронным выполнением кода. По

своей сути, GCD представляет собой абстракцию над потоками, предлагая разработчику не прямое управление thread, а работу с очередями задач (dispatch queues), в которые помещается логика для выполнения.

GCD управляет пулом потоков, где задачи распределяются по виртуальным потокам, динамически отображаемым на ограниченном количестве физических потоков, соответствующих доступным ядрам процессора. Это позволяет системе более эффективно использовать ресурсы, автоматически балансируя нагрузку и минимизируя простои. Концептуальная модель представлена на рисунке 15.

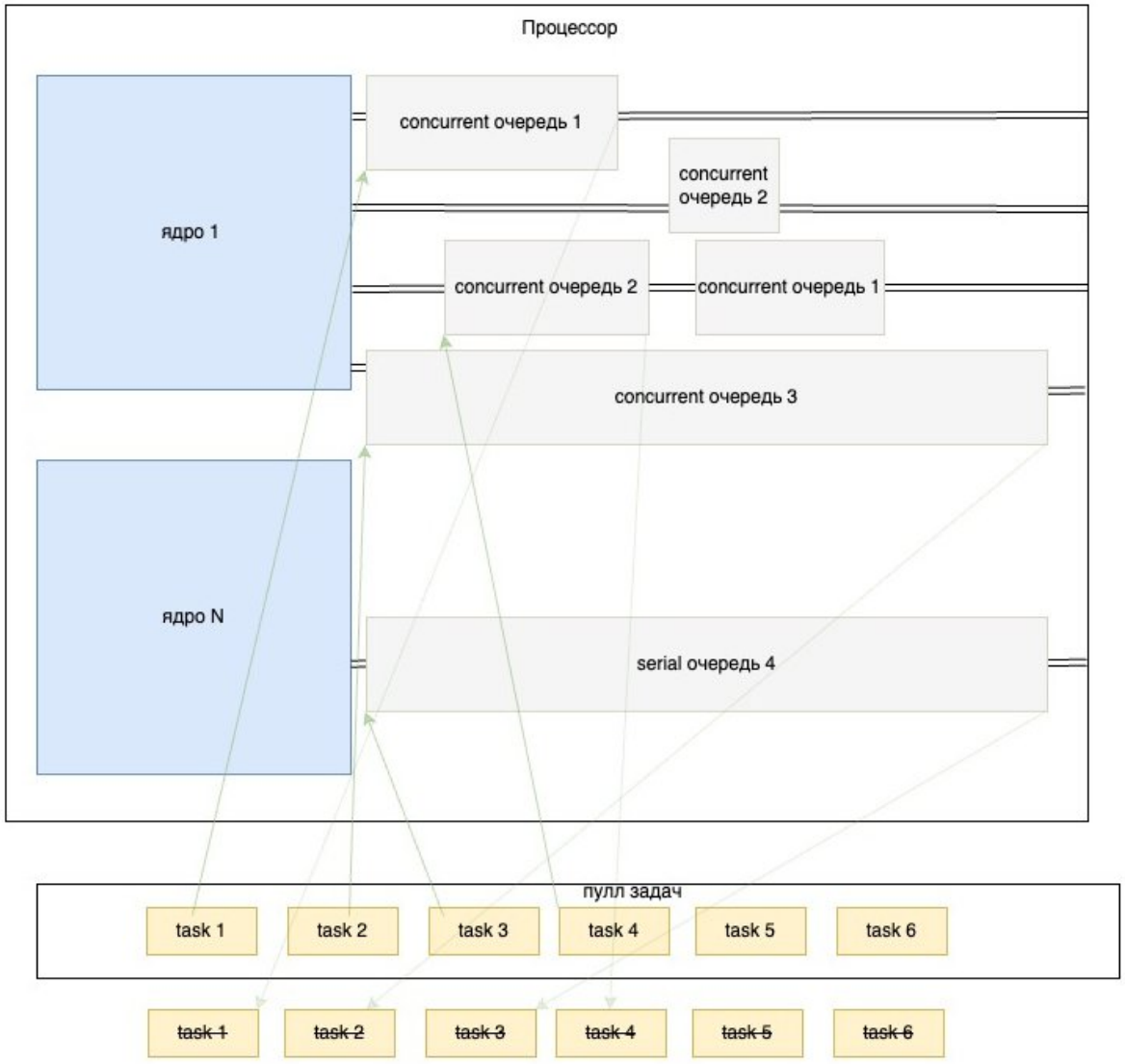


Рисунок 15 – Концептуальная работа GCD

Сами очереди бывают двух типов:

1) Serial (последовательные) — задачи выполняются строго по порядку;
2) Concurrent (параллельные) — задачи могут выполняться одновременно, если ресурсы позволяют.

Кроме того, есть системные глобальные очереди с различным приоритетом (qos — quality of service) и возможность создавать пользовательские очереди. Несмотря на эффективность, GCD не лишён недостатков:

– Контекст переключения (context switching): при интенсивной работе с множеством задач система может тратить значительные ресурсы на переключение между потоками, особенно при высоком уровне параллелизма.

– Memory overhead: каждый поток (в том числе виртуальный), который используется для хранения локальных переменных, параметров функций и информации о вызовах функций требует выделения системных ресурсов — стеков, структур планирования и др. При чрезмерном количестве задач может наблюдаться избыточное потребление памяти.

Тем не менее, GCD остаётся мощным инструментом, особенно для простых и локальных фоновых операций.

Swift Concurrency

Swift Concurrency — это высокоуровневая система управления асинхронным кодом, встроенная в язык Swift, начиная с версии 5.5. Она предоставляет декларативный и безопасный способ работы с многопоточными задачами за счёт ключевых конструкций async, await, Task, actor и других.

Swift Concurrency построена на основе планировщика задач (cooperative task scheduler), который работает совместно с GCD, но предоставляет более предсказуемую и управляемую модель. В отличие от GCD, где задачи жестко распределяются по потокам, Swift Concurrency использует легковесные кооперативные задачи, которые могут быть приостановлены (suspended) и возобновлены без создания новых потоков. Концептуальная модель Swift Concurrency представлена на рисунке 16.

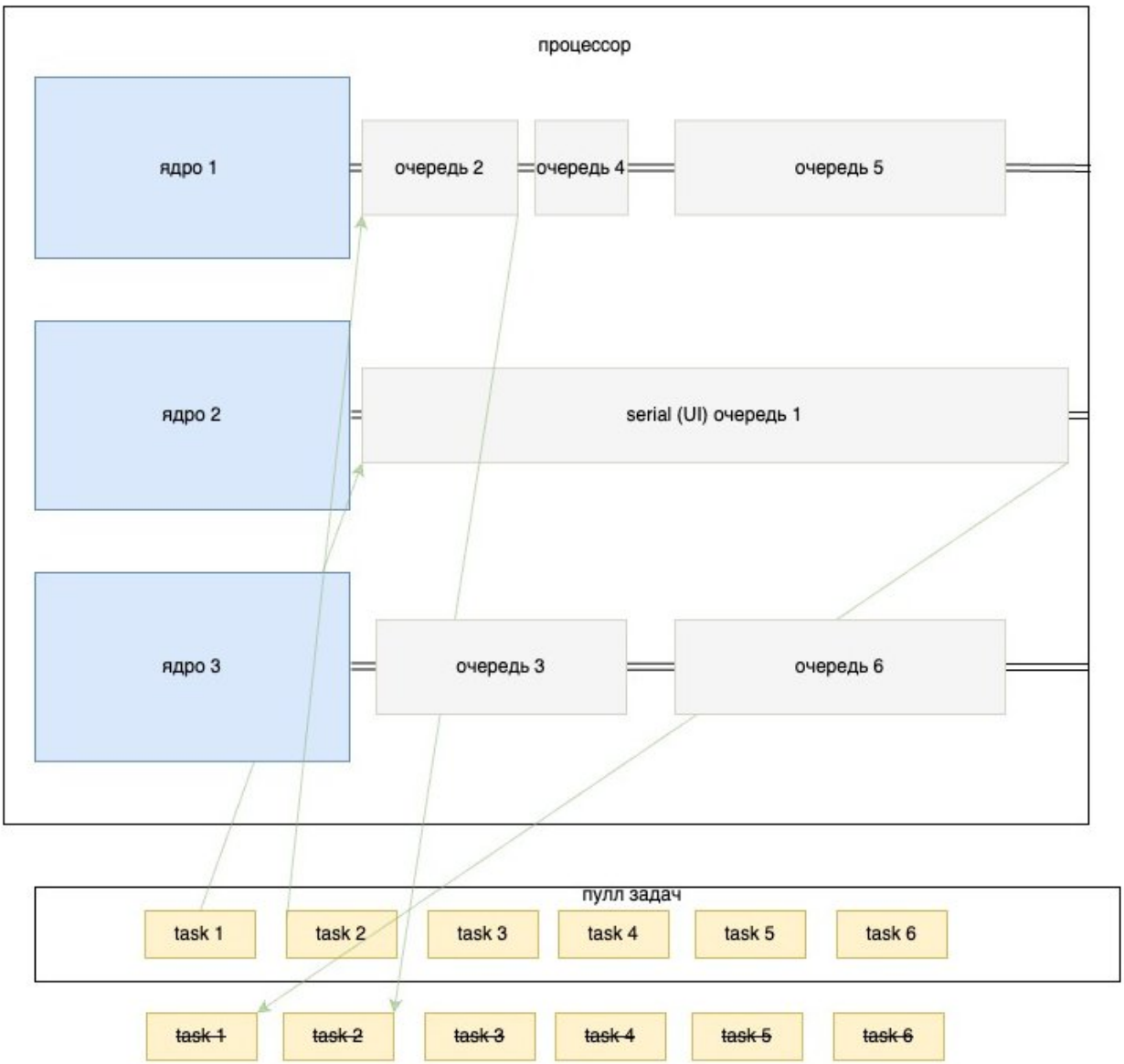


Рисунок 16 – Концептуальная модель Swift Concurrency

Использование GCD и Swift Concurrency в рамках одного проекта позволяет эффективно сочетать низкоуровневое управление задачами с высокоуровневой безопасной асинхронностью. GCD остаётся удобным решением для простых фоновых задач, требующих прямого контроля, тогда как Swift Concurrency предоставляет современную, декларативную модель для построения сложных асинхронных систем. В совокупности эти технологии позволяют обеспечить высокую производительность, стабильность и комфортный пользовательский опыт в мобильном приложении.

2.1.5 Выбор СУБД

Для реализации регистрации и авторизации будет использоваться Firebase, а для локального хранения данных — CoreData, дополнительно, для хранения простых пользовательских настроек в приложении применяется UserDefaults. Архитектура системы представлена на рисунке 17.

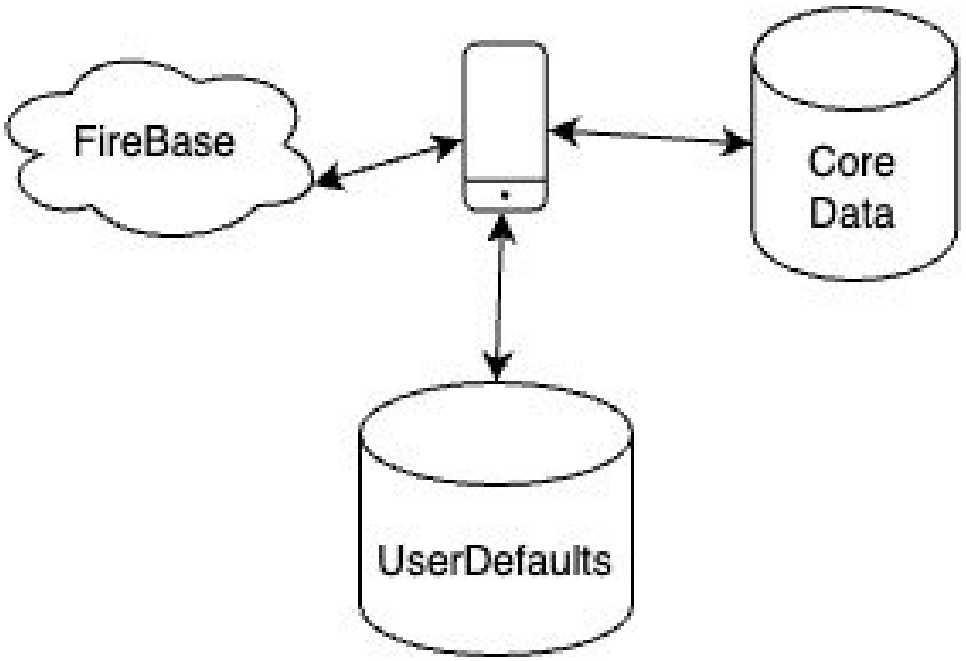


Рисунок 17 - Архитектура системы

Firebase — облачная платформа от Google с широкими возможностями:

- Легкая интеграция: SDK для iOS поддерживает функции аутентификации, хранения данных, аналитики.
- Гибкость аутентификации: регистрация через email, социальные сети, анонимная авторизация.
- Надежность и масштабируемость: построен на инфраструктуре Google, что гарантирует стабильность при росте нагрузки.
- Быстрая разработка: позволяет сосредоточиться на клиентской части.

В рамках разработки мобильного приложения для изучения английского языка в качестве системы хранения данных была выбрана CoreData — предоставляет удобный способ управления данными внутри приложений iOS. Структура хранимых данных отражена на ER диаграмме из приложения А1. Этот

фреймворк можно рассматривать как промежуточный слой между «сырой» базой данных и кодом, написанным на Swift. Core Data позволяет оперировать привычными объектами Swift, не вдаваясь в детали низкоуровневой работы с базами данных.

Под капотом Core Data использует SQLite — надёжную и производительную реляционную СУБД, представленную в виде одного файла. Разработчику не нужно взаимодействовать с SQL-запросами напрямую — все операции выполняются через удобные API, в виде манипуляции экземплярами классов моделей. Это значительно упрощает повседневную работу с данными и снижает риск ошибок, связанных с ручным управлением базой.

Core Data поддерживает автоматическую миграцию. Поскольку структура базы данных (схема) напрямую зависит от модели данных, в процессе развития приложения часто возникает необходимость изменить её: добавить новые свойства, переименовать существующие, изменить связи между объектами и т.д. В случае прямой работы с SQLite, такие изменения потребовали бы ручного вмешательства и могли привести к повреждению базы. Core Data, в свою очередь, предоставляет автоматические и пользовательские миграции, что позволяет безболезненно адаптировать схему к новой структуре модели, не теряя существующих данных.

Core Data также позволяет работать в оффлайн-режиме, что особенно важно для приложений, которые должны функционировать без постоянного подключения к интернету. Все данные сохраняются локально на устройстве пользователя, обеспечивая доступ к изучению английского в любой момент. Это повышает стабильность и удобство использования приложения, особенно в условиях нестабильного или отсутствующего сетевого соединения.

Core Data обладает рядом дополнительных преимуществ:

- автоматическое кэширование объектов и отслеживание их изменений;
- возможность выполнения сложных запросов с использованием NSFetchRequest, NSPredicate и NSSortDescriptor — для фильтрации, сортировки и выборки данных по критериям;

– работа в многопоточном окружении с помощью отдельных контекстов (NSManagedObjectContext), что особенно важно для приложений, где операции с данными выполняются в фоне, параллельно с пользовательским интерфейсом.

– Таким образом, Core Data стала идеальным выбором для мобильного приложения, предполагающего работу с большим объёмом структурированных данных (категории, слова). Она обеспечивает надёжное и удобное хранилище, возможность оффлайн-доступа, снижает сложность реализации функциональности, связанной с обработкой и синхронизацией данных.

CoreData — мощный инструмент для локального хранения данных:

– Оффлайн-доступ: Хранение пользовательских данных на устройстве, доступно без интернета.

– Производительность: эффективно работает с большими объемами данных.

– Интеграция с iOS: поддерживает другие фреймворки Apple, а удобные инструменты в Xcode ускоряют разработку.

UserDefaults также используется в приложении в качестве простого механизма хранения пользовательских настроек. Это встроённое средство системы iOS, позволяющее сохранять пары ключ–значение и автоматически поддерживающее их между сессиями. В рамках данной разработки UserDefaults применяется для хранения флажков, таких как выбранная тема интерфейса (светлая / тёмная / системная) и другие мелкие настройки. Данный способ подходит для быстрого доступа к конфигурационным параметрам, не требующим структурированного хранения или сложной логики обновления.

Критерии сравнения баз данных приведены в таблицах 18-23:

Таблица 18 –Сложность разработки

Качественная оценка	Количественная оценка
Использование требует большого объема ручной настройки и устаревших технологий	1
Присутствуют инструменты автоматизации, но часть функций требует ручной конфигурации	2
Разработка осуществляется быстро за счёт полной документации, шаблонов и встроенной поддержки в Xcode	3

Таблица 19 –Масштабируемость

Качественная оценка	Количественная оценка
Плохо масштабируется, сложна при изменениях или увеличении объема данных	1
При масштабировании требует дополнительных архитектурных решений или сторонних библиотек	2
Изначально построена для масштабируемости, поддерживает расширение без радикальных изменений структуры БД	3

Таблица 20 –Производительность

Качественная оценка	Количественная оценка
Используемые технологии создают нагрузку на ресурсы устройства, наблюдаются задержки	1
Производительность приемлемая, но требует оптимизации кода или структуры данных	2
Справляется с большими объёмами данных, оптимизирована под iOS	3

Таблица 21 –Оффлайн-доступ

Качественная оценка	Количественная оценка
Не поддерживает работу без интернета или требует внешние костыльные решения	1
Есть оффлайн-доступ, но требует ручной реализации синхронизации и кеширования	2
Полноценная нативная поддержка оффлайн-режима, синхронизация и хранение встроены	3

Таблица 22 –Поддержка экосистемы iOS

Качественная оценка	Количественная оценка
Плохо интегрируется с iOS, требует обертки для работы языка Swift между другими языками.	1
Частично поддерживается, но требует дополнительной настройки или адаптации под Apple-фреймворки.	2
Полностью интегрируется в iOS, разработана Apple или под неё, оптимизирована под Swift и Xcode.	3

Сравнение и выбор базы данных для разрабатываемой системы приведено в таблице 23

Таблица 23 – Сравнение баз данных

	Базы данных			
Критерии сравнения баз данных	**Firebase**	**Core data**	**SQLite**	**Весовые коэффициенты**
Сложность разработки	3	2	1	0,1
Масштабируемость	3	2	2	0,1
Производительность	2	3	3	0,2
Оффлайн-доступ	1	3	3	0,3
Поддержка экосистемы iOS	3	3	2	0,3
Итоговая оценка $y_j = \sum_{\{i=1\}}^{n} \alpha_{i*}\ k_{\{ij\}}$	2,2	2,8	2,4	я]

ЗАКЛЮЧЕНИЕ

В ходе разработки мобильного приложения для изучения английского языка были достигнуты следующие результаты:

– Изучена предметная область, охватывающая вопросы индивидуального языкового обучения, методик заучивания лексики, а также применения цифровых технологий в образовательной среде. Особое внимание уделено анализу современных подходов к запоминанию слов, таких как интервальное повторение и обучение с помощью карточек. Проведено исследование существующих решений, таких как Duolingo, Lingualeo и Busuu, с последующей оценкой их функциональных и технических характеристик. Это позволило выявить сильные и слабые стороны аналогов, а также сформировать обоснованные требования к собственной системе.

– Сформулированы и классифицированы требования к разрабатываемому приложению. Учтены как функциональные аспекты — регистрация пользователей, создание и редактирование карточек, прохождение тренировок, учёт прогресса, — так и нефункциональные: производительность, надёжность, стабильность работы в офлайн-режиме, а также удобство пользовательского интерфейса. Особый акцент сделан на возможности персонализации и мотивации пользователя.

– Разработана архитектура системы, включающая клиентскую часть, реализованную на языке Swift, и облачные сервисы авторизации, предоставляемые платформой Firebase [21]. В качестве базы для локального хранения данных выбрана технология Core Data, обеспечивающая удобную работу с объектной моделью данных и возможность масштабирования. В структуре проекта применены принципы Clean Architecture в сочетании с паттерном MVC, что позволило достичь модульности, гибкости и расширяемости системы. Такой подход упрощает сопровождение проекта и добавление нового функционала в будущем.

– Реализована мобильная информационная система, ориентированная на повседневное использование в процессе изучения английского языка. В приложении предусмотрены следующие пользовательские сценарии: создание категорий, добавление карточек, изучение слов с помощью механики свайпов, отслеживание индивидуального прогресса, получение достижений за активность и ведение статистики в виде графиков. Визуальная часть проекта разработана с учётом требований Apple Human Interface Guidelines, что обеспечивает интуитивность и адаптивность интерфейса под различные устройства.

– Обеспечена многопоточность при работе с данными и сетью. Для этого использовались технологии GCD (Grand Central Dispatch) и Swift Concurrency, что позволило избежать блокировок основного потока, повысить отзывчивость интерфейса и обеспечить корректное выполнение операций с базой данных и облачными сервисами. Асинхронное взаимодействие реализовано как при чтении/сохранении объектов в Core Data, так и при выполнении сетевых запросов (регистрация, авторизация). Это особенно важно для стабильности работы приложения в условиях нестабильного интернет-соединения.

– Для хранения простых пользовательских настроек, таких как тема оформления интерфейса и локальные флажки, применён встроенный механизм UserDefaults, обеспечивающий быстрый доступ и устойчивость к сбоям.

Разработанное программное решение обладает потенциалом для дальнейшего развития. Возможность расширения функциональности — за счет внедрения новых режимов тренировки, поддержки мультимедиа, а также адаптации под другие языки и платформы — делает систему конкурентоспособной и перспективной на рынке мобильных образовательных продуктов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Постников В.М. Основы эксплуатации автоматизированных систем обработки информации и управления. Краткий курс: учеб, пособие. М.: Изд- во МГТУ им. Н.Э. Баумана, 2013. 180 с.

2. Постников В.М. Основы эксплуатации АСОИиУ. Часть 1. Техническое обслуживание: учеб, пособие. М.: Изд-во МГТУ им. Н.Э. Баумана. 2015. 192 с.

3. Постников В.М. Основы эксплуатации АСОИнУ. Часть 2. Администрирование и развитие: учеб, пособие. М.: Изд- во МГТУ им. Н.Э. Баумана, 2015. 190 с.

4. Постников В.М. Черненький В.М. Методы принятия решений в системах организационного управления: учеб, пособие. / В.М. Постников. В.М. Черненький. М.: Изд-во МГТУ им. Н.Э. Баумана, 2014. 205 с.

5. Документация Swift [Электронный ресурс]. – Режим доступа: https://docs.swift.org/swift-book/. – Загл. с экрана. – Дата обращения 20.05.2025.

6. Документация Apple [Электронный ресурс]. – Режим доступа: https://developer.apple.com/documentation/. – Загл. с экрана. – Дата обращения 20.05.2025.

7. Christian Keur. iOS Programming: The Big Nerd Ranch Guide [Текст] / Aaron Hillegass. – O'Reilly Media, 2021. – 225 с.

8. Robert C. Martin. Clean Code: A Handbook of Agile Software Craftsmanship [Текст] Dean Wampler. – O'Reilly Media, 2008. – 464 с.

9. Методические рекомендации по подготовке и защите выпускной квалификационной работы бакалавра. / Кротов Ю.Н. [Электронный ресурс] – URL: https://drive.google.com/file/d/1pEcfTr3xDdJ81Hxz2F6GcbtNV1n3dan6/view. (дата обращения: 10.04.2025).

10. Apple Inc. The Swift Programming Language. – 6.1. – Apple Books, 2024. – 500 с. docs.swift.org

11. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. – Boston: Prentice Hall, 2017. – 432 с

12. Firebase Documentation. URL: https://firebase.google.com/docs (дата обращения: 27.05.2025). Firebase+1Google Cloud+1

13. Core Data | Apple Developer Documentation. URL: https://developer.apple.com/documentation/coredata (дата обращения: 27.05.2025). Apple Developer

14. User Defaults in iOS Swift: A Comprehensive Guide. URL: https://medium.com/@kashif00527/user-defaults-in-ios-swift-a-comprehensive-guide-6d6d00675074 (дата обращения: 27.05.2025).

15. Apple Inc. UIKit | Apple Developer Documentation. URL: https://developer.apple.com/documentation/uikit (дата обращения: 27.05.2025).

16. Apple Inc. View Controllers | Apple Developer Documentation. URL: https://developer.apple.com/documentation/uikit/view-controllers (дата обращения: 27.05.2025).Apple Developer

17. Apple Inc. Views and Controls | Apple Developer Documentation. URL: https://developer.apple.com/documentation/uikit/views-and-controls (дата обращения: 27.05.2025).

18. Apple Inc. UIKit Catalog: Creating and Customizing Views and Controls. URL: https://developer.apple.com/documentation/uikit/mac_catalyst/uikit_catalog_creating_and_customizing_views_and_controls (дата обращения: 27.05.2025).Apple Developer

19. Apple Inc. UIKit Updates | Apple Developer Documentation. URL: https://developer.apple.com/documentation/updates/uikit (дата обращения: 27.05.2025).

20. Josh Holtz. Туториал по Встроенным Подпискам в iOS с помощью StoreKit 2 и Swift. Хабр. URL: https://habr.com/ru/articles/707730/ (дата обращения: 27.04.2025).

21. popkovden. Firebase Analytics в KMP: Android, iOS, Desktop (MacOS, Windows). Хабр. URL: https://habr.com/post/818891/comments/ (дата обращения: 20.04.2025).

22. Bazilier. Аутентификация Firebase с помощью Google Sign-In в iOS. Хабр. URL: https://habr.com/post/741408/comments/ (дата обращения: 27.04.2025).

23. batrade1000. SwiftUI уроки (часть 1). Хабр. URL: https://habr.com/ru/articles/796029/ (дата обращения: 27.04.2025).

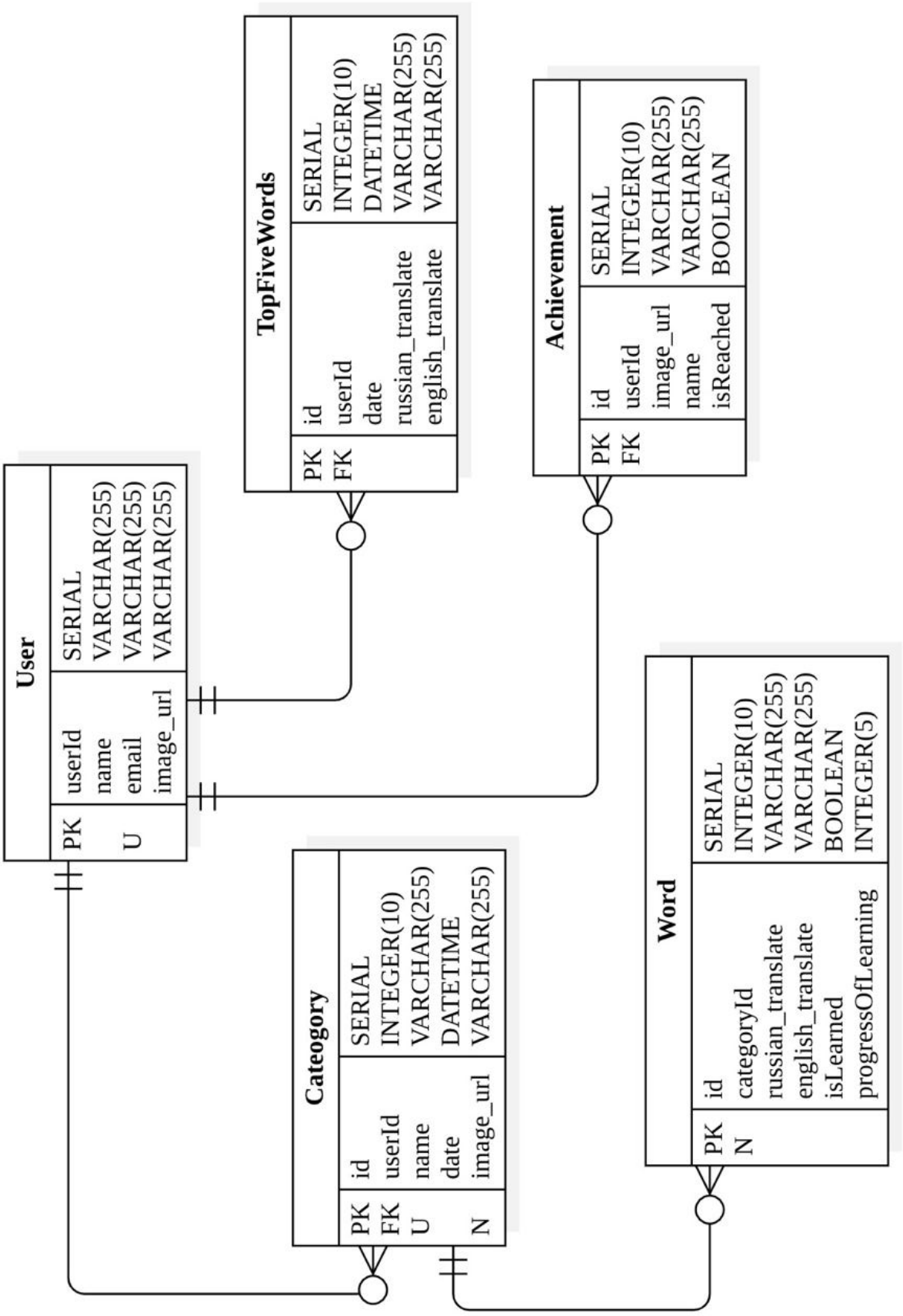
24. batrade1000. SwiftUI уроки (часть 12). Хабр. URL: https://habr.com/en/articles/813437/comments/ (дата обращения: 27.04.2025).

ПРИЛОЖЕНИЕ А. ГРАФИЧЕСКИЕ МАТЕРИАЛЫ

А1. Рисунок 1 «ER диаграмма данных предметной области»

А2. Таблица 1 «Описание сущностей базы данных»

А1. Рисунок 1 – ER диаграмма данных предметной



А2. Таблица 1 «Описание сущностей базы данных»

Таблица	Поле	Тип данных	Ключ	Описание
User	userId	SERIAL	PK	Уникальный идентификатор пользователя
	name	VARCHAR(255)		Имя пользователя
	email	VARCHAR(255)		Уникальная электронная почта пользователя
	image_url	VARCHAR(255)		Ссылка на изображение профиля
Category	id	SERIAL	PK	Уникальный идентификатор категории
	userId	INTEGER(10)	FK	Внешний ключ на пользователя
	name	VARCHAR(255)		Уникальное название категории
	date	DATETIME		Дата создания категории
	image_url	VARCHAR(255)		Ссылка на изображение категории
Word	id	SERIAL	PK	Уникальный идентификатор слова
	categoryId	INTEGER(10)	FK	Внешний ключ на категорию
	russian_translate	VARCHAR(255)		Перевод слова на русский язык
	english_translate	VARCHAR(255)		Слово на английском языке
	isLearned	BOOLEAN		Флаг, показывающий, выучено ли слово
	progressOfLearning	INTEGER(5)		Уровень прогресса изучения
Achievement	id	SERIAL	PK	Уникальный идентификатор достижения
	userId	INTEGER(10)	FK	Внешний ключ на пользователя
	image_url	VARCHAR(255)		Ссылка на изображение достижения
	name	VARCHAR(255)		Название достижения

	isReached	BOOLEAN		Флаг выполнения достижения
Таблица	**Поле**	**Тип данных**	**Ключ**	**Описание**
TopFiveWords	id	SERIAL	PK	Уникальный идентификатор записи
	userId	INTEGER(10)	FK	Внешний ключ на пользователя
	date	DATETIME		Дата добавления
	russian_translate	VARCHAR(255)		Перевод слова на русский язык
	english_translate	VARCHAR(255)		Слово на английском языке

ПРИЛОЖЕНИЕ Б ТЕХНИЧЕСКОЕ ЗАДАНИЕ

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Кафедра «Системы обработки информации и управления»

Утверждаю	Согласовано
Заведующий кафедрой ИУ-5	Научный руководитель
_____ В.И.Терехов	_____ В.М.Постников
"__"_____2025 г.	"__"_____2025 г.

Информационная система обучения английскому языку

Техническое задание
(вид документа)

писчая бумага
(вид носителя)

5
(количество листов)

ИСПОЛНИТЕЛЬ: _____ Пустовалов Григорий Владимирович

"__"_____2025 г.

Москва - 2025

Введение

Актуальность разработки обосновывается тем, что изучение иностранных языков становится все более востребованным в современном мире. Это связано с глобализацией, необходимостью международного общения, а также ростом карьерных и образовательных возможностей, которые открываются перед людьми со знанием английского языка. В то же время цифровые технологии проникают во все аспекты жизни, предоставляя удобные и эффективные инструменты для обучения.

Традиционные методы изучения языков, включающие использование бумажных учебников, списков слов и карточек, обладают рядом недостатков: они не всегда удобны, требуют постоянного доступа к физическим материалам и не предлагают интерактивности, что может снижать мотивацию учащихся. Современные мобильные приложения предоставляют уникальную возможность для автоматизации и повышения интереса пользователей за счет своей гибкости.

1. Основания для разработки

Основанием для разработки является задание на выпускную квалификационную работу, подписанное руководителем выпускной работы и утверждённое заведующим кафедрой ИУ5 МГТУ им. Н.Э. Баумана 15 декабря 2024 года.

2. Назначение разработки

Назначением работы является разработка мобильного приложения, которое упрощает процесс изучения английского языка, обеспечивает пользователям удобный и персонализированный инструмент для создания и изучения слов в английском языке.

3. Требования к программе или программному изделию

3.1. Требования к функциональным характеристикам.

Разрабатываемая система должна выполнять следующие функции:

3.1.1 Авторизация.

3.1.2 Регистрация.

3.1.3 Просмотр категорий слов.

3.1.4 Просмотр онбординга.

3.1.5 Создание категории слов.

3.1.6 Удаление категорий слов.

3.1.7 Сортировка категорий слов.

3.1.8 Просмотр слов в категории.

3.1.9 Добавление слова в категорию.

3.1.10 Возможность добавления перевода слова, используя api переводчика.

3.1.11 Удаление слова из категории.

3.1.12 Тренировка слов внутри категории.

3.1.13 Тренировка слов для изучения.

3.1.14 Просмотр профиля.

3.1.15 Возможность управление цветовой темой приложения.

3.1.16 Просмотр достижений.

3.1.17 Просмотр статистики.

3.2 Требования ко входным данным

3.2.1 Входные данные формируются при регистрации и действиями авторизованного пользователя.

3.2.2 Поддержка загрузки изображений.

3.3 Требования к выходным данным

3.3.1 Система должна сохранять обработанные данные в базу данных.

3.3.2 При добавлении новых слов и категорий должны валидировать текст.

3.3.3 Показ ошибок пользователю в случае неисправности.

3.4 Требования к составу и параметрам технических средств

3.4.1 Минимальная версия IOS 16.0

3.5 Требования к программному обеспечению

3.5.1 Операционная система: macOS.

3.5.2 Среда выполнения: Xcode.

3.5.3 Хранилище данных: Core Data, Firebase.

4. Требования к программной документации

Для представления заказчику разрабатываются следующие документы:

1. Техническое задание.
2. Рабочий материал по выполняемому проекту.
3. Программа и методика испытаний.
4. Графический материал по программному изделию в формате презентации.

5. Технико-экономические показатели

Требования к данному разделу не предъявляются.

6. Стадии и этапы разработки

6.1 Этапы разработки программы:

6.1.1 Выбор инструмента для разработки.

6.1.2 Проектирование системы.

6.1.3 Разработка внутренних алгоритмов обработки данных

6.1.4 Разработка процессов, выполняющих оркестрацию данных

6.1.5 Реализация программы.

6.1.6 Тестирование и отладка программы.

6.1.7 Сравнение производительности системы с аналогами

6.1.8 Оформление документации.

6.2 Этапы предъявления программных документов (таблица 6.2)

Таблица 6.2 Этапы предъявления программных документов

№ п/п	Наименование этапа и содержание работ	Сроки исполнения
1	Техническое задание	11-20 марта 2025 г.
2	Рабочий материал	март 2025 г.
3	Программа и методика испытаний	апрель 2025 г.
4	Графический материал по программному изделию	май 2025 г.

7. Порядок контроля и приёмки

Приём программного изделия в виде испытания функциональных возможностей осуществляется в ходе «Защиты макетов программ – предварительной защиты ВКРБ» в период с 15 по 26 мая 2025 года в соответствие с разработанной программой и методикой испытаний.

На испытание представляются программный продукт, техническое задание, программа и методика испытаний.

8. Дополнительные условия

Данные технические заданные может уточняться в установленном порядке.

ПРИЛОЖЕНИЕ В ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Кафедра «Системы обработки информации и управления»

Утверждаю	Согласовано
Заведующий кафедрой ИУ-5	Научный руководитель
_____В.И.Терехов	_____В.М. Постников
"__"_____2025 г.	"__"_____2025 г.

Информационная система обучения английскому языку

ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ
(вид документа)

писчая бумага
(вид носителя)

8
(количество листов)

ИСПОЛНИТЕЛЬ:

_____ Пустовалов Григорий Владимирович
"__"_____2025 г.

Москва – 2025

1. Объект испытаний

Информационная система обучения английскому языку.

2. Цель испытаний

Цель испытания – проверка функционирования всех указанных в техническом задании функций системы.

3. Требования к программе

Система должна выполнять следующие функции:

3.1. Система должна сохранять обработанные данные в базу данных.

3.2. При добавлении новых слов и категорий должны валидировать текст.

3.3. Показ ошибок пользователю в случае неисправности.

4. Требования к документам

Перед проведением испытаний предъявляются следующие документы:

1. Техническое задание;
2. Программа и методика испытаний.

5. Технические требования

5.1. Требования к техническим средствам

Система должно выполняться на телефоне или симуляторе со следующими характеристиками:

5.1 Оперативная память смартфона не менее 3 ГБ.

5.2 Процессор A13 Bionic или новее.

Для авторизации и регистрации на устройстве должен быть доступ в интернет.

5.2. Требования к программным средствам

5.2.1. Требования к программным средствам для разработки

5.2.1.1. Операционная система: macOS версии не ниже 13.0 (Ventura) или более новой.

5.2.1.2. Среда разработки: Xcode версии не ниже 14.0, поддерживающая SDK для iOS 16.

5.2.1.3. Язык программирования: Swift (актуальная версия, поддерживаемая Xcode).

5.2.1.4. Инструменты контроля качества кода: SwiftLint.

5.2.2. Требования к программным средствам для использования системы

5.2.2.1. Операционная система: iOS версии не ниже 16.0.

5.2.2.2. Поддерживаемые устройства: iPhone SE (2-го поколения) и выше.

5.2.2.3. Необходимые разрешения: доступ к интернету, уведомления , доступ к галерее.

6. Средства и порядок испытаний

Методы испытаний и их последовательность для информационной системы приведены в таблице 1.

Таблица 1 - Методы испытаний и их последовательность

№	Действие	Результат	№ п. ТЗ
1.	1. Предусловие: пользователь не должен быть не авторизован. 2. Открыть приложение и нажать кнопку «Войти»; 3. Ввод почты и пароля. 4. Нажимать на кнопку «Войти».	Пользователь авторизовался. Переход на главную с категориями пользователя.	4.1.1.
2.	1. Предусловие: пользователь не должен быть авторизован. 2. Открыть приложение. 3. Ввести имя, почту, пароль. 4. Нажимать на кнопку «Зарегистрироваться»	Пользователь зарегистрировался. Если пользователь зашел в приложение в первый раз, то отобразиться экран – знакомства с приложением, иначе переход на главную с категориями по умолчанию.	4.1.2.
3.	1. Предусловие: пользователь должен быть авторизован. 2. Отрыть приложение. 3. Перейти к экрану «Слова».	Перед пользователем будет главный экран с категориями слов, прогресс изучения и 5 последних неизученных слов.	4.1.3.
4.	1. Предусловие: пользователь должен быть авторизован. 2. Отрыть приложение. 3. Перейти к экрану «Слова». 4. Нажать на изображение «Лампа».	Пользователь увидит экран знакомства с приложением — краткую инструкцию по работе с приложением.	4.1.4.
5.	1. Предусловие: пользователь должен быть авторизован. 2. Отрыть приложение. 3. Перейти к экрану «Слова». 4. Нажать на «+». 5. Ввести название для новой	Создастся новая категория с изображением по умолчанию, если пользователь не добавит свое.	4.1.5.

№	Действие	Результат	№ п. ТЗ
	категории и добавить фотографию при необходимости.		
6.	1. Предусловие: пользователь должен быть авторизован, должна быть хотя бы одна категория. 2. Отрыть приложение. 3. Перейти к экрану «Слова». 4. На главной найти категорию, для удаления. 5. Удержать нажатие. 6. На плашке подтвердить удаление, нажать на «Удалить категорию».	Удаление выбранной категории слов.	4.1.6.
7.	1. Предусловие: пользователь должен быть авторизован. 2. Отрыть приложение. 3. Перейти к экрану «Слова». 4. Нажать на кнопку для сортировки. 5. Выбрать тип сортировки.	Возможность сортировки категорий слов по недавно добавленным или названию.	4.1.7.
8.	1. Предусловие: пользователь должен быть авторизован. 2. Отрыть приложение. 3. Перейти к экрану «Слова». 4. Открыть категорию для просмотра слов.	Перед пользователем будет экран со словами внутри категории.	4.1.8.
9.	1. Предусловие: пользователь должен быть авторизован. 2. Отрыть приложение 3. Перейти к экрану «Слова» 4. Открыть необходимую	Добавление нового слова в категорию.	4.1.9.

№	Действие	Результат	№ п. ТЗ
	категорию слов. 5. Нажать на иконку плюс. 6. Ввести слово на русском и английском языке. 7. Нажать на кнопку «Добавить слово».		
10.	1. Предусловие: пользователь должен быть авторизован. 2. Отрыть приложение. 3. Перейти к экрану «Слова» 4. Открыть необходимую категорию слов. 5. Нажать на иконку плюс. 6. Ввести слово на русском или на английском языке, в соответствующее поле. 7. Нажать на кнопку перевода.	Возможность перевода слова на противоположный язык, без использования словаря.	4.1.10.
11.	1. Предусловие: пользователь должен быть авторизован, в категории слов должны быть слова. 2. Отрыть приложение. 3. Перейти к экрану «Слова». 4. Открыть необходимую категорию слов. 5. Удержать нажатие. 6. На плашке подтвердить удаление, нажать на «Удалить слово».	Удаление слова из категории.	4.1.11.
12.	1. Предусловие: пользователь должен быть авторизован, в категории слов должны быть слова. 2. Отрыть приложение	Пользователь имеет возможность изучать новые слова в выбранной категории.	4.1.12.

№	Действие	Результат	№ п. ТЗ
	3. Перейти к экрану «Слова» 4. Открыть необходимую для изучения категорию слов. 5. Нажать на кнопку «Перейти к изучению»		
13.	1. Предусловие: пользователь должен быть авторизован. 2. Отрыть приложение 3. Перейти к экрану «Тренировка слов» 4. Начать тренировку, листая вправо или влево в зависимости от знания перевода слова.	Пользователь имеет возможность изучать новые слова.	4.1.13.
14.	1. Предусловие: пользователь должен быть авторизован. 2. Отрыть приложение. 3. Перейти к экрану «Профиль».	Пользователь может посмотреть профиль, в котором есть изображение профиля, имя, почта, прогресс изучения слов, возможность смены темы и кнопка для выхода из аккаунта.	4.1.14.
15.	1. Предусловие: пользователь должен быть авторизован. 2. Отрыть приложение 3. Перейти к экрану «Профиль» 4. Выбрать необходимую цветовую тему.	Пользователь имеет возможность выбрать удобную ему цветовую тему (темную или светлую) или задать системную тему, повторяющую тему, заданную на устройстве.	4.1.15.
16.	1. Предусловие: пользователь должен быть авторизован. 2. Отрыть приложение. 3. Перейти к экрану «Профиль». 4. Нажать на кнопку иконку «Статистика».	Пользователь может посмотреть свои достижения.	4.1.16.
17.	1. Предусловие: пользователь	Пользователь может посмотреть	4.1.17.

№	Действие	Результат	№ п. ТЗ
	должен быть авторизован. 2. Отрыть приложение. 3. Перейти к экрану «Профиль». 4. Нажать на кнопку иконку «Статистика». 5. Перейти к экрану, нажав на «Статистика».	свою статистику количества слов в категории и соотношение изученных слов к словам в процессе изучения.	

7. Результат испытаний

Результатом проведенных испытаний является получение ожидаемых реакций системы на различные действия, совершенные пользователем, в рамках функций системы.