МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

РАЗРАБОТКА IOS-ПРИЛОЖЕНИЯ ДЛЯ СОВЕРШЕНСТВОВАНИЯ НАВЫКОВ АНГЛИЙСКОГО ЯЗЫКА

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 411 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Вильцева Данилы Денисовича

Научный руководитель
доцент _____ Ю. Н. Кондратова

Заведующий кафедрой
к.ф.-м.н., доцент _____ С. В. Миронов

Саратов 2024г

ВВЕДЕНИЕ

В наши дни всё больше людей предпочитают мобильные приложения для более быстрого доступа к информации. С постоянным увеличением количества смартфонов и планшетов, а также улучшением сетевой инфраструктуры, рынок мобильных приложений стремительно развивается и приобретает все большую значимость.

Глобальные доходы от приложений особенно резко выросли после пандемии Covid-19. Согласно опросу Statista [**?**], еще в 2019 году доход индустрии приложений составил 253 млрд долларов. Статистика показывает, что в 2023 году доходы, полученные от покупок в приложениях, рекламы приложений или платных приложений составят почти 490 миллиардов долларов, а пользователи iPhone и iPad составят 63% этой стоимости.

Мобильные приложения предлагают ряд значимых преимуществ, что делает их неотъемлемой частью современной технологической инфраструктуры. Во-первых, они обеспечивают простоту и удобство использования. Благодаря интуитивному интерфейсу и персонализированным функциям, пользователи могут легко получать необходимую информацию или воспользоваться услугами в несколько касаний.

Во-вторых, мобильные приложения создают возможности для бизнеса и предпринимателей, позволяя им расширять свою аудиторию, улучшать взаимодействие с клиентами и увеличивать продажи.

Современный рынок не может обойти вниманием феномен изучения иностранных языков. Мобильные приложения, предназначенные для изучения иностранных языков, занимают значительную долю на этом рынке и продолжают пользоваться огромным спросом. Это объясняется несколькими факторами.

Во-первых, мобильные приложения предлагают удобство и доступность для пользователей. Они позволяют изучать язык в любое удобное время и в любом месте, просто используя свой смартфон или планшет. Это особенно важно для современных людей, которые часто находятся в движении и имеют ограниченное количество свободного времени.

Во-вторых, мобильные приложения для изучения иностранных языков обычно предлагают разнообразные интерактивные методы обучения. Они могут включать в себя игры, карточки с флэш-картами, аудио- и видеоматериалы, тесты и многое другое. Это помогает сделать процесс изучения языка увлека-

тельным и интересным, что способствует более эффективному усвоению материала.

Всё это доказывает неоспоримый потенциал и перспективность развития в данной области.

Целью данной работы является разработка мобильного приложения под iOS для совершенствования практических навыков английского языка.

Для достижения поставленной цели были поставлены следующие задачи:

- Анализ аналогичных мобильных приложений и их недостатков
- Определение требований к приложению
- Определение необходимых для разработки инструментов и технологий
- Разработка серверной части приложения
- Разработка мобильного приложения под платформу iOS

Структура и объём работы.

Для решения поставленных задач выполнена выпускная квалификационная работа, включающая в себя введение, 2 основные главы, заключение, список использованных источников из 27 наименований и 3 приложений. Работа изложена на 135 страницах, содержит 73 рисунка.

Первая глава имеет название «Анализ готовых приложений» и содержит анализ трех аналогичных мобильных приложений, их преимущества и недостатки, сформулированные функциональные и нефункциональные требования.

Вторая глава имеет название «Разработка мобильного приложения», данная глава содержит подробное описание архитектуры приложения, инструменты и технологии для разработки клиентской и серверной частей.

Выпускная квалификационная работа заканчивается заключением, списком использованных источников, а также приложениями с кодом А-В.

1 Основное содержание работы

1.1 Анализ мобильных приложений

В работе были рассмотрены приложения для изучения английского языка, такие как:

- Duolingo;
- Babbel;
- Tandem.

Были выявлены основные преимущества и недостатки рассмотренных приложений. На основе проведенного анализа были выделены функциональные требования к разрабатываемому приложению, учитывающие как основные преимущества, так и недостатки аналогичных продуктов на рынке.

1.2 Функциональные требования к приложению

Функциональные требования к приложению определяют, каким образом приложение должно функционировать и какие задачи должно выполнять. Они описывают конкретные функции, возможности и поведение приложения. Функциональные требования помогают определить, что приложение должно делать, чтобы удовлетворить потребности пользователей или решить определенную проблему. В ходе анализа были выявлены и выполнены следующие требования:

- Приложение должно обеспечить возможность прохождения тестирования пользователем для определения текущего уровня языка;
- Приложение должно обеспечить возможность общения пользователя с другими людьми на английском языке;
- Приложение должно обеспечить возможность проверки введенных сообщений на грамматические ошибки;
- В приложении должна быть доступна краткая теория по каждой грамматической теме английского языка для различных уровней владения;
- В приложении должны быть реализованы различные упражнения для отработки грамматических и лексических навыков;
- Приложение должно обеспечить возможность изучения новых слов с помощью системы Лейтнера.

1.3 Нефункциональные требования к системе

Нефункциональные требования определяют свойства и ограничения, накладываемые на систему. Для реализации приложения были выявлены следующие требования:

- Мобильное приложение должно быть написано на языке Swift с использованием Xcode и фреймворка SwiftUI;
- Приложение должно работать на платформе iOS версии 15.0 и выше;
- Серверная часть приложения должна представлять собой микросервисную архитектуру с использованием языков Java и Python, а также фреймворков Spring Framework и FastAPI соответственно;
- В качестве базы данных должна использоваться реляционная СУБД PostgreSQL;
- Приложение должно быть контейнеризовано с использованием Docker для обеспечения портабельности и изолированной среды выполнения как на стадии разработки, так и на стадии развертывания;
- Разработка и развертывание приложения должны осуществляться с применением CI/CD практик. Это включает автоматизированные процессы сборки, тестирования и развертывания, а также автоматическое обновление кода на сервере при внесении изменений в репозиторий;
- Для развертывания серверной части приложения требуется использование инфраструктуры, совместимой с Docker контейнерами. Предпочтительным вариантом для этого является облачная платформа Timeweb Cloud, которая поддерживает контейнеризацию и предоставляет инструменты для управления и масштабирования приложений.

1.4 Архитектура приложения

Компонентами архитектуры приложения являются клиент, сервер и база данных.

1. **Клиент — мобильное приложение под iOS**:
   - Мобильное приложение разработано на языке Swift с использованием среды разработки Xcode и фреймворка SwiftUI;
   - Взаимодействие с сервером осуществляется через API (Application Programming Interface), реализованный на серверной стороне приложения;
   - Клиентское приложение обеспечивает удобный пользовательский ин-

терфейс для взаимодействия с функциональными возможностями приложения, включая аутентификацию пользователя, отображение данных, отправку запросов к серверу и обработку ответов.

2. **Сервер**: Серверная часть приложения представляет собой микросервисную архитектуру. В микросервисной архитектуре приложение разбивается на ряд независимо развертываемых сервисов, которые взаимодействуют с помощью API-интерфейсов. Благодаря такому подходу каждый отдельный сервис можно развертывать и масштабировать независимо от других.
   - Основной сервис реализован на языке программирования Java с использованием фреймворка Spring Framework;
   - Сервис упражнений реализован на языке Python с использованием фреймворка FastAPI;
   - Основные задачи сервера включают обработку запросов от клиентских приложений, выполнение бизнес-логики, доступ к базе данных и обеспечение безопасности приложения;
   - Сервер взаимодействует с клиентами посредством API, предоставляя им необходимые данные и функциональность.
3. **База данных**:
   - В качестве базы данных используется PostgreSQL, реляционная система управления базами данных, известная своей надежностью, масштабируемостью и расширяемостью;
   - База данных служит для хранения структурированных данных, таких как информация о пользователях, их профилях, настройках, а также других сущностей, необходимых для работы приложения;
   - Серверное приложение взаимодействует с базой данных через соответствующие запросы, обеспечивая доступ к данным и выполнение операций CRUD (Create, Read, Update, Delete).

1.5 Разработка серверной части

1.5.1 Чаты

Для подбора собеседника и генерации темы диалога на стороне сервера реализован модуль «Dialog», который включает в себя:

Обмен сообщениями в чатах осуществлен с помощью протокола WebSocket, который позволяет устанавливать двустороннее соединение между клиентом и

сервером для передачи данных.

Был реализован класс-конфигурация «WebSocketConfig», который позволяет клиентам устанавливать соединение с сервером по адресу «/ws» и обмениваться сообщениями с помощью протокола STOMP, используя префиксы «/app» и «/topic» для отправки и получения сообщений.

Помимо этого, был создан класс-компонент «WebSocketEventListener», обеспечивающий обработку событий подключения и отключения клиентов WebSocket, а также отправку сообщений об отключении пользователя от канала.

1.5.2 Тестирование

Для проверки текущего уровня знания английского языка на стороне бэкенда был разработан модуль тестирования, включающий в себя:

- controllers.TestController - rest-контроллер, отвечающий за обработку HTTP-запросов и возвращение соответствующих HTTP-ответов;
- model.Answer - модель ответа пользователя;
- model.Question - модель вопроса тестирования;
- repo.TestRepository - репозиторий модуля тестирования, использующийся для выполнения операций базы данных, таких как сохранение, поиск, обновление и удаление данных;
- services.TestService - интерфейс, содержащий методы для тестирования;
- services.TestServiceImplementation - сервис, реализующий методы интерфейса TestService.

1.5.3 Карточки

Для изучения слов английского языка был выбран метод флэш-карточек, позволяющий эффективно изучать новые слова и повторять уже изученные. Для реализации данной возможности на стороне бэкенда был создан модуль «Cards», включающий в себя:

- controllers.CardController - rest-контроллер, отвечающий за обработку HTTP-запросов и возвращение соответствующих HTTP-ответов;
- model.Card - модель карточки слова;
- model.CardSet - модель сета карточек слов;
- repo.CardRepository - репозиторий модуля карточек;
- services.CardService - интерфейс, содержащий методы для реализации функционала флэш-карточек;

— services.CardServiceImplementation - сервис, реализующий методы интерфейса CardService.

1.5.4 Теория

Согласно выявленным в ходе анализа функциональным требованиям к приложению, необходимо обеспечить пользователю доступ к краткой теории по каждой грамматической теме английского языка. Для этого на стороне бэкенда был разработан модуль «Theory», включающий в себя:

— controllers.TheoryController - rest-контроллер, отвечающий за обработку HTTP-запросов и возвращение соответствующих HTTP-ответов;
— model.GrammarTheory - общая модель теории;
— model.Category - модель категории;
— model.Topic - модель темы;
— model.Theory - модель самой теоретической справки;
— model.Subtopic - модель подтемы;
— repo.GrammarTheoryRepository - репозиторий модуля теории;
— scraper.Scraper - сервис, реализующий парсер теории;
— services.TheoryService - интерфейс, содержащий методы для работы с полученной теорией;
— services.TheoryServiceImplementation - сервис, реализующий методы интерфейса TheoryService [**?**].

Для получения теории английского языка был разработан парсер сайта «https://preply.com/en/learn/english/grammar», реализованный в сервисе «Scraper». Метод «fetch» выполняет запрос к указанному URL-адресу и возвращает html-код страницы для получения информации о категориях, темах, подтемах и самой теории.

Метод «scrapeCategories» отвечает за получение категорий грамматических тем, содержащихся на главной странице.

Сначала происходит получение html-кода страницы с помощью метода «fetch». Далее он парсится с использованием библиотеки Jsoup, чтобы его можно было анализировать и извлекать необходимые данные. Используя метод «doc.select(«.stylestopicCardWrapperyd3fU»)», извлекаются все элементы HTML, которые представляют собой категории на странице. Эти элементы содержат информацию о категории, такую как заголовок и описание. Для каждого элемента категории выполняется итерация. Для каждой категории создается объект

класса «Category», в который затем будет добавлена информация о категории. Для каждого элемента категории извлекается заголовок и описание. Заголовок извлекается из ссылки (<a>), а описание из соответствующего элемента на странице.

После извлечения информации о категории вызываются методы «scrapeTopics» и «scrapeTheory» для сканирования списка тем или теорий, соответственно. В зависимости от того, какие данные доступны (темы или теории), создается соответствующий список и добавляется в объект категории. Если при сканировании данных о категории возникает исключение IOException, оно перехватывается и обрабатывается, чтобы предотвратить сбой выполнения программы.

После извлечения информации о категории объект «Category» добавляется в список «categories». В конце метода возвращается список всех категорий, извлеченных из страницы.

Аналогично реализованы методы «scrapeTopics», «scrapeSubtopics», а также «scrapeTheory» для получения данных о темах, подтемах и теории.

Для сохранения данных созданы модели данных «Category», «GrammarTheory», «Subtopic», «Theory» и «Topic».

В интерфейсе «TheoryService» описаны методы для вызова парсера теории «scrapeTheory» и метод её получения из базы данных «getTheory».

1.5.5 Проверка грамматики

Для проверки сообщений на наличие грамматических ошибок был создан модуль «Grammar». В интерфейсе «GrammarService» описан единственный метод получения данных с сервера LanguageTool по API. Он реализован в сервисе «GrammarServiceImplementation».

1.5.6 Упражнения

Согласно функциональным требованиям, в приложении должны быть реализованы упражнения для отработки грамматических и лексических навыков, что позволит пользователю повышать свой уровень владения английским языком. Данный функционал было принято реализовать в отдельном микросервисе на языке Python с использованием фреймворка FastAPI, в виду более удобного взаимодействия с сервисом OpenAI через зависимость «openai».

В проекте имеется три вида упражнений:

— «Грамматический тренинг» - реализован в SentencesExercise.py

— «Ответ на вопрос» - реализован в QuestionsExercise.py
— «Перевод текста» - реализован в TranslationExercise.py

Во всех сервисах есть метод «ask_gpt», который принимает список сообщений в качестве входных данных. Он отправляет эти сообщения модели GPT-3.5 для генерации ответов. Метод «client.chat.completions.create» используется для взаимодействия с API OpenAI.

Ответ от API OpenAI сохраняется в переменной response. Ответ содержит список choice, сгенерированных моделью. Далее извлекается содержимое первого choice (response.choices[0].message.content) и возвращает его в качестве вывода функции «ask_gpt».

Для создания грамматического тренинга была реализована функция «main» в «SentencesExercise.py». В качестве параметра она принимает тему упражнений и использует ее для формулирования запроса к модели GPT-3.5.

Функция формирует вопрос, который будет передан модели для генерации упражнений по указанной теме. Вопрос составлен таким образом, чтобы указать модели требуемый формат ответа и желаемый тип упражнений.

Вопрос упаковывается в структуру данных, которая представляет собой список сообщений, где каждое сообщение содержит роль (в данном случае «user») и контент (вопрос).

Список сообщений передается функции «ask_gpt», которая отправляет его модели для генерации ответа.

Полученный ответ представлен в виде JSON-строки. Функция выполняет десериализацию JSON-строки в Python-структуру данных с помощью функции «json.loads». Обработанные данные по упражнениям возвращаются в качестве вывода функции «main».

1.6 Докеризация

Докеризация - это процесс упаковки приложений и их зависимостей в контейнеры, которые могут быть развернуты и запущены в любой среде, поддерживающей Docker.

Для всех сервисов данного приложения и базы данных были реализованы свои докерфайлы. Определение конфигурации, настройки для развертывания и связывание сервисов и базы данных находится в файле «docker-compose.yml».

1.7 CI/CD

Для упрощения развертывания приложения на сервере при каждом изменении кода была использована DevOps практика CI/CD, которая позволяет свести к минимуму ошибки, повысить темпы сборки и качество разрабатываемого продукта.

В качестве инструмента для автоматизации данного процесса был выбран Github Actions.

Для определения шагов, которые будут выполнены автоматически при определенных событиях, таких как push в репозиторий или создания pull request, был реализован файл «pipeline.yml».

Далее был создан runner на Github Actions для собственного хоста и настроен на сервере с помощью указанной документации.

1.8 Развертывание на сервере

В качестве облачного хостинга, предоставляющего облачные сервера, был выбран «Timeweb Cloud», в виду его разнообразных возможностей и доступных цен на различные конфигурации серверов.

Был выбран облачный сервер со следующей конфигурацией:

- CPU - 1 x 3.3 ГГц;
- RAM - 2 Гб;
- Канал - 200 Мбит/с.

Данной конфигурации достаточно для тестового развертывания приложения и проверки его функциональности.

1.9 Разработка iOS-приложения

Для реализации приложения был использован язык программирования Swift, среда разработки Xcode и вышеописанные инструменты.

Структура проекта выглядит следующим образом:

- «DI» - модуль, отвечающий за инъекцию зависимостей приложения. Включает в себя «ServicesAssembly», где происходит регистрация соответствующих зависимостей (вьюмодели, сервисы и так далее), а также «Assembler», в котором происходит их сборка;
- «App» - модуль, содержащий класс, который является стартовой точкой работы приложения;
- «Resources» - здесь хранятся необходимые данные для отображения интерфейса (изображения, цвета, строки и так далее);
- «Navigation» - модуль, отвечающий за навигацию в приложении. Содержит «MainNavigation», который необходим для навигации между экранами авторизированной части приложения. «StartNavigation» - навигация экранов части приложения до авторизации. Также содержит структуру «AppContainer», определяющую какую View необходимо отобразить.
- «Models» - содержит все модели приложения. Они подразделяются на «LocalModel» и «ServerModel». При ответе сервера приложение получает «ServerModel», далее переводит её с помощью соответствующего «Mapper» в локальную модель, которая затем и используется в работе;
- «Services» - содержит все сервисы приложения, такие как:
  - «SocketService» - отвечает за работу с вебсокетом;

  - «Network» - отвечает за работу с сетью;
  - «Storeges» - отвечает за хранение данных в UserDefaults и паролей в KeyChain;
  - «AuthenticationService» - отвечает за аутентификацию пользователя в приложении.
- «Helpers» - содержит вспомогательные компоненты приложения, например, структуру «FormatDate» для преобразования даты в верный формат;
- «Modules» - содержит все основные модули приложения («Чаты», «Упражнения», «Карточки» и др.). Каждый модуль содержит соответствующую View и ViewModel;
- «Components» - содержит вспомогательные компоненты пользовательского интерфейса.

В ходе разработки мобильного iOS-приложения были реализованы следующие модули:

- Стартовый экран;
- Регистрация пользователя;
- Авторизация пользователя;
- Создание профиля пользователя;
- Тестирование;
- Экран профиля;
- Чаты;
- Словарь;
- Карточки;
- Теория;
- Упражнения.

ЗАКЛЮЧЕНИЕ

В данной работе были рассмотрены мобильные приложения Duolingo, Babbel, Tandem, проанализированы их достоинства и недостатки.

Были сформулированы требования к разрабатываемой системе с учетом недостатков рассмотренных приложений. Также были описаны необходимые инструменты для разработки мобильного приложения для iOS и серверной части приложения.

В бакалаврской работе было разработано приложение для совершенствования навыков владения английским языком. Приложение написано для iOS с использованием языка Swift, фреймворка SwiftUI, архитектуры MVVM и других технологий. Функционал серверной части приложения был разработан на языках программирования Java и Python, с использованием фреймворков Spring и FastAPI, соответственно. В качестве базы данных приложения был выбран PostgreSQL.

Возможности приложения позволяют пользователю:

— Создавать личный аккаунт;
— Проверять текущий уровень владения языком посредством тестирования;
— Воспользоваться встроенным словарем для поиска необходимых слов и выражений;
— Совершенствовать грамматические навыки в упражнениях: «Грамматический тренинг», «Ответ на вопрос» и «Перевод текста»;
— Изучать новые слова и повторять уже изученные в разделе «Карточки»;
— Изучать и повторять теорию грамматики английского языка;
— Общаться с другими пользователями с наиболее подходящим уровнем владения языком;
— Добавлять других пользователей в «Друзья».