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 ANDROID-ПРИЛОЖЕНИЯ ДЛЯ СОВЕРШЕНСТВОВАНИЯ НАВЫКОВ АНГЛИЙСКОГО ЯЗЫКА

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студентки 4 курса 411 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Мигуновой Анастасии Александровны

Научный руководитель
доцент _____ Ю. Н. Кондратова

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

Саратов 2024

ВВЕДЕНИЕ

В современном мире мобильные приложения стали неотъемлемой частью повседневной жизни, предоставляя пользователям множество удобных решений для различных задач. С ростом технологий и широким распространением смартфонов и планшетов, мобильные приложения находят применение в самых разных сферах, включая образование, здравоохранение, финансы и развлечения. Их актуальность обусловлена не только высокой доступностью и удобством использования, но и способностью адаптироваться к индивидуальным потребностям пользователей.

Пандемия COVID-19 кардинально изменила образ жизни людей по всему миру, внесла значительные коррективы в повседневные привычки и ускорила цифровую трансформацию различных сфер деятельности. В условиях карантина и социальной дистанции мобильные приложения стали незаменимым инструментом для выполнения различных задач, от работы и обучения до общения и развлечений. Они позволили людям оставаться на связи, продолжать обучение и поддерживать продуктивность в условиях изоляции.

Одной из самых значимых областей, где мобильные приложения проявили свою актуальность, является изучение английского языка. В условиях глобализации и растущей международной коммуникации владение английским языком стало необходимым навыком для профессионального и личного роста. Мобильные приложения для изучения английского языка предлагают инновационные подходы к обучению, сочетая интерактивные методы, геймификацию и персонализацию.

Пандемия подчеркнула важность гибкости и доступности в обучении, и мобильные приложения стали ответом на эти вызовы. Они предоставляют возможность изучать язык в любое удобное время и в любом месте, что особенно актуально в условиях ограниченного доступа к традиционным образовательным учреждениям.

Разработка мобильного приложения для изучения английского языка представляет собой важный шаг в направлении предоставления пользователям удобного и эффективного инструмента для обучения. Учитывая современные тенденции, которые стоят перед сферой образования, такое приложение сможет удовлетворить потребности широкого круга пользователей, способствуя их профессиональному и личностному развитию.

Цель данной работы — разработка мобильного приложения под Android для совершенствования практических навыков английского языка.

Задачи данной работы:

- 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 Android.

Структура и объём работы.

Для решения поставленных задач выполнена выпускная квалификационная работа, включающая в себя введение, 4 основные главы, заключение, список использованных источников из 20 наименований и 3 приложений. Работа изложена на 112 страницах, содержит 37 рисунков.

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 «Средства и технологии, используемые для разработки android-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 готовых приложений

Перед тем как приступить к разработке, важно провести анализ рынка аналогичных мобильных приложений для изучения английского языка: Duolingo, Memrise, HelloTalk. Были выявлены основные преимущества и недостатки рассмотренных приложений.

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

1.2 Средства и технологии, используемые для разработки android-приложения

Компонентами архитектуры приложения являются клиент, сервер и база данных.

1. **Клиент — мобильное приложение под Android**:
   - клиентское приложение разработано на языке Kotlin с использованием интегрированной среды разработки Android Studio и XML-файлов для определения визуального интерфейса;
   - взаимодействие с сервером осуществляется через API (Application Programming Interface), реализованный на серверной стороне приложения;
   - клиентское приложение обеспечивает удобный пользовательский интерфейс для взаимодействия с функциональными возможностями приложения, включая отображение данных, отправку запросов к серверу и обработку ответов.
2. **Сервер**: Серверная часть приложения представляет собой микросервисную архитектуру. В этой архитектуре приложение разбивается на ряд независимо развертываемых сервисов, которые взаимодействуют с помощью API-интерфейсов. Благодаря такому подходу каждый 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).

1.3 Разработка серверной части

1.3.1 Подбор собеседника и темы диалога

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

- **controllers.DialogController** — это компонент, обрабатывающий HTTP-запросы, которые начинаются с «/dialog», в соответствие с аннотацией @RequestMapping("/dialog"). Он взаимодействует с сервисным слоем, выполняя бизнес-логику, и формирует соответствующие HTTP-ответы.
- **services.DialogService** — компонент, который отвечает за взаимодействие с базой данных. Он представлен в виде интерфейса и используется для взаимодействия с базой данных, абстрагируя детали реализации, такие как SQL-запросы.
- **services.DialogServiceImplementation** — сервис, осуществляющий обработку запросов от контроллера, использующий репозиторий для получения, сохранения, обновления или удаления данных из базы данных. Реализует методы интерфейса DialogService.

1.3.2 Словарь

Пользователь должен иметь возможность искать и сохранять необходимые слова как на русском, так и на английском языке в словаре. Для этого на стороне сервера реализован модуль «Dictionary», который включает в себя:

— **model.Word** — сущность «Word» для использования в контексте JPA. Она представляет собой запись в таблице «dictionary», где каждый экземпляр класса соответствует строке в таблице. Поля класса «Word» аннотированы с помощью JPA-аннотаций для определения столбцов таблицы:
  1. «@Id» указывает, что поле id является первичным ключом.
  2. «@Column(name = "word")» связывает поле word с одноименным столбцом в таблице.
  3. «@Column(name = "description")» связывает поле description с одноименным столбцом в таблице.

— **controllers.DictionaryController** — это компонент, обрабатывающий HTTP-запросы, которые начинаются с «/dictionary», в соответствие с аннотацией @RequestMapping("/dictionary"). Он взаимодействует с сервисным слоем, выполняя бизнес-логику, и формирует соответствующие HTTP-ответы.

— **repo.DictionaryRepository** — интерфейс репозитория для сущности Word с использованием Spring Data JPA. В этом интерфейсе определены два метода для поиска слов в словаре: findEngWord и findRusWord.

1.3.3 Добавление в друзья

Пользователь должен иметь возможность отправлять запросы на добавление в друзья и принимать или отклонять запросы в друзья от других пользователей. Для этого на стороне сервера реализован модуль «Friends», который включает в себя:

— **model.FriendRequest** — сущность «FriendRequest» представляет собой запись в таблице «friend_requests», где каждый экземпляр класса соответствует строке в таблице. Поля класса аннотированы с помощью JPA-аннотаций для определения столбцов таблицы:
  1. «@Id» указывает, что поле id является первичным ключом.
  2. «@Column(name = "sender_email")» связывает поле senderEmail с одноименным столбцом в таблице.
  3. «@Column(name = "receiver_email")» связывает поле receiverEmail с

одноименным столбцом в таблице.

— **controllers.FriendRequestController** — это компонент, обрабатывающий HTTP-запросы, которые начинаются с «/api/v1/friends», в соответствие с аннотацией @RequestMapping("/api/v1/friends"). Он взаимодействует с сервисным слоем, выполняя бизнес-логику, и формирует соответствующие HTTP-ответы.

— **repo.FriendRequestRepository** — интерфейс репозитория для сущности FriendRequest, который использует Spring Data JPA. В этом интерфейсе определены методы для выполнения операций с сущностью FriendRequest в базе данных:

1. метод existsBySenderEmailAndReceiverEmail проверяет наличие запроса на добавление в друзья от определенного отправителя (senderEmail) к определенному получателю (receiverEmail). Если такой запрос существует, метод возвращает true, в противном случае - false.
2. метод deleteBySenderEmailAndReceiverEmail удаляет запрос на добавление в друзья от определенного отправителя к определенному получателю. Если запрос не найден, никаких действий не выполняется.
3. метод findBySenderEmail находит все запросы на добавление в друзья, отправленные определенным отправителем (senderEmail).
4. метод findByReceiverEmail находит все запросы на добавление в друзья, адресованные определенному получателю (receiverEmail).

— **services.FriendRequestService** — компонент, который отвечает за взаимодействие с базой данных. Он представлен в виде интерфейса и используется для взаимодействия с базой данных, абстрагируя детали реализации, такие как SQL-запросы.

— **services.FriendRequestServiceImplementation** — сервис, осуществляющий обработку запросов от контроллера, использующий репозиторий для получения, сохранения, обновления или удаления данных из базы данных. Реализует методы интерфейса FriendRequestService.

1.3.4 Интересы

Пользователь также должен иметь возможность выбирать интересующие его темы для общения. Для этого на стороне сервера реализован модуль «Intersts», который включает в себя:

— **model.Interest** — сущность «Interest» представляет собой запись в таблице «interests», где каждый экземпляр класса соответствует строке в таблице. Поля класса аннотированы с помощью JPA-аннотаций для определения столбцов таблицы:
    1. «@Id» указывает, что поле id является первичным ключом.
    2. «@Column(name = «interest», unique = true)» связывает поле interest с одноименным столбцом в таблице. Значения в этом столбце должны быть уникальными в пределах таблицы базы данных.
— **controllers.InterestController** — это компонент, обрабатывающий HTTP-запросы, которые начинаются с «/interests», в соответствие с аннотацией @RequestMapping("/interests"). Он взаимодействует с сервисным слоем, выполняя бизнес-логику, и формирует соответствующие HTTP-ответы.
— **repo.InterestRepository** — интерфейс репозитория для сущности Interest, который использует Spring Data JPA.
— **services.InterestService** — компонент, который отвечает за взаимодействие с базой данных. Он представлен в виде интерфейса и используется для взаимодействия с базой данных, абстрагируя детали реализации, такие как SQL-запросы.
— **services.InterestServiceImplementation** — сервис, осуществляющий обработку запросов от контроллера, использующий репозиторий для получения, сохранения, обновления или удаления данных из базы данных. Реализует методы интерфейса InterestService.

1.3.5 Профиль пользователя

Регистрация и авторизация пользователя реализованы на стороне сервера в модуле «Profile», который включает в себя:

— **model.User** — сущность «User» представляет собой запись в таблице «users», где каждый экземпляр класса соответствует строке в таблице. Поля класса аннотированы с помощью JPA-аннотаций для определения столбцов таблицы.
— **model.OnBoardingInfo** — сущность «OnBoardingInfo» представляет собой запись в таблице «users», где каждый экземпляр класса соответствует строке в таблице. Эта модель нужна для установки информации пользователя в поля email, username, dateOfBirth, photo.
— **config.AmazonConfig** — конфигурационный класс для настройки клиента

Amazon S3, который будет взаимодействовать с облачным хранилищем Timeweb.

— **controllers.AuthController** — это компонент, обрабатывающий HTTP-запросы, которые начинаются с «/auth», в соответствие с аннотацией @RequestMapping("/auth").
— **controllers.RegisterController** — это компонент, обрабатывающий HTTP-запросы, которые начинаются с «/register», в соответствие с аннотацией @RequestMapping("/register").
— **controllers.UserController** — это компонент, обрабатывающий HTTP-запросы, которые начинаются с «/api/v1/user», в соответствие с аннотацией @RequestMapping("/api/v1/user"). Он взаимодействует с сервисным слоем, выполняя бизнес-логику, и формирует соответствующие HTTP-ответы.
— **repository.UserRepository** — интерфейс UserRepository, который используется для работы с сущностью User в базе данных. Он расширяет интерфейс JpaRepository и добавляет метод для поиска пользователей по их email-адресу.
— **services.AuthService** — сервис, осуществляющий обработку запросов от контроллера AuthController.
— **services.UserService** — сервис, осуществляющий обработку запросов от контроллеров RegisterController и UserController.
— **utils.Base64DecodedMultipartFile** — класс Base64DecodedMultipartFile реализует интерфейс MultipartFile, который обычно используется в Spring для обработки файлов, загруженных с веб-формы.

1.4 Разработка Android-приложения

Для разработки Android-приложения использован язык программирования Kotlin, среда разработки Android Studio и описанные выше инструменты.

Проект состоит из следующих основных компонентов:

1. **MainActivity**:
   — MainActivity является основным активити (Activity) приложения. Оно запускается при старте приложения и служит контейнером для различных фрагментов (Fragments).
   — Управляет навигацией между фрагментами, обрабатывает основные события пользовательского интерфейса и координирует взаимодействие между фрагментами.

2. **Фрагменты (Fragments)**:
   — Фрагменты представляют собой независимые части пользовательского интерфейса и логики приложения. Они могут быть добавлены, удалены или заменены в MainActivity в зависимости от действий пользователя или состояния приложения [**?**].
3. **Навигация**:
   — Навигация между фрагментами осуществляется через MainActivity, которая использует FragmentManager для добавления, замены или удаления фрагментов.
   — При взаимодействии пользователя с элементами интерфейса в одном фрагменте, MainActivity инициирует переход к другому фрагменту, передавая необходимые данные через аргументы.
4. **Обработка событий**:
   — MainActivity слушает и обрабатывает основные события пользовательского интерфейса, такие как нажатия кнопок и выбор элементов меню. Фрагменты также могут обрабатывать события, относящиеся к их собственной области ответственности.
   — Фрагменты могут взаимодействовать с MainActivity через интерфейсы, обеспечивая слабую связанность и гибкость в обработке событий [**?**].
5. **Ресурсы (Resources)**:
   — Проект содержит ресурсы, такие как строки (strings), стили (styles), макеты (layouts) и изображения (drawables), которые используются в MainActivity и фрагментах для построения пользовательского интерфейса.
   — Все ресурсы хранятся в соответствующих директориях внутри папки res: res/values, res/layout, res/drawable.

Также был создан клиентский API и интерфейс для взаимодействия с сервером через HTTP-запросы с использованием библиотеки Retrofit в Android-приложении. Класс ApiClient создает и настраивает экземпляр Retrofit для подключения к серверу. Интерфейс Service содержит методы для различных операций с сервером.

1. **ApiClient**:
   — Retrofit и Gson: Настраивается объект Retrofit, который использует

GsonConverterFactory для обработки JSON и ScalarsConverterFactory для обработки строковых ответов.

- Базовый URL: Устанавливается базовый URL для всех запросов: "http://90.156.224.51:8081".
- Service: Создается экземпляр интерфейса Service, предоставляющий методы для взаимодействия с сервером.

2. **Service Interface**:
   - Методы в интерфейсе аннотированы для выполнения различных HTTP-запросов, таких как GET и POST.

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

1. Стартовый экран;
2. Регистрация пользователя;
3. Авторизация пользователя;
4. Создание профиля пользователя;
5. Тестирование;
6. Экран профиля;
7. Чаты;
8. Словарь;
9. Карточки;
10. Теория;
11. Упражнения.

ЗАКЛЮЧЕНИЕ

В данной работе были рассмотрены мобильные приложения Duolingo, Memrise, HelloTalk, проанализированы их достоинства и недостатки.

Исходя из выявленных недостатков, были сформулированы требования к разрабатываемой системе. Также описаны необходимые инструменты для разработки мобильного приложения для Android и серверной части.

В результате было разработано приложение для совершенствования навыков владения английским языком. Приложение создано для Android с использованием языка Kotlin, архитектур MVP и других современных технологий. Серверная часть приложения разработана на языках программирования Java и Python с использованием фреймворков Spring и FastAPI соответственно. В качестве базы данных использован PostgreSQL.

Функциональные возможности приложения включают:

- Создание личного аккаунта;
- Проверку текущего уровня владения языком через тестирование;
- Встроенный словарь для поиска необходимых слов и выражений;
- Грамматические упражнения: «Грамматический тренинг», «Ответ на вопрос» и «Перевод текста»;
- Раздел «Карточки» для изучения и повторения новых слов;
- Теоретический материал по грамматике английского языка;
- Возможность общения с пользователями, имеющими подходящий уровень владения языком;
- Добавление других пользователей в список друзей.