МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
"Южно-Уральский государственный университет"
(национальный исследовательский университет)
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

РАБОТА ПРОВЕРЕНА

Рецензент
Ген. дир. ООО "АппБит Софтвейр"

_____ Д.А. Диков

“___”_____ 2017 г.

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой,
д.ф.-м.н., профессор

_____ Л.Б. Соколинский

“___”_____ 2017 г.

РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ НА ANDROID «ИЗУЧЕНИЕ ИНОСТРАННЫХ СЛОВ»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 02.03.02.2017.13-001-1382.КР

Научный руководитель:
преподаватель
_____К. Г. Чупахин

Автор работы:
студент группы КЭ-401
_____Е. В. Арямнов

Ученый секретарь
(нормоконтролер)

_____ О.Н. Иванова

“___”_____ 2017 г.

Челябинск-2017

ОГЛАВЛЕНИЕ

ОГЛАВЛЕНИЕ 4
ВВЕДЕНИЕ 6
1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 9
1.1. Обзор аналогичных проектов 9
1.2. Обзор существующих средств разработки для платформы Android 11
1.3. Обзор технологий для создания распределенного приложения 13
1.3.1. Концепция REST 13
1.3.2. Технология WebSocket 13
2. АНАЛИЗ ТРЕБОВАНИЙ 15
2.1. Функциональные требования к проектируемой системе 15
2.2. Нефункциональные требования к проектируемой системе 15
2.3. Диаграмма вариантов использования 16
3. АРХИТЕКТУРА 18
3.1. Архитектура мобильного приложения 18
3.1.1. Проектирование базы данных 18
3.1.2. Компоненты мобильного приложения 19
3.2. Архитектура сервера 22
3.2.1. Проектирование базы данных 22
3.2.2. Архитектура REST-сервиса 23
3.2.3. Проектирование WebSocket-сервера 23
4. РЕАЛИЗАЦИЯ 25
4.1. Реализация серверной части 25
4.1.1. База данных 25
4.1.2. Реализация взаимодействия с базой данных 25
4.1.3. Реализация REST-сервиса 26
4.1.4. Реализация Socket-сервера 29
4.2. Реализация клиентской части 31
4.2.1. Реализация компонента Presenter 32
4.2.2. Реализация компонента Model 36

4.2.3. Реализация компонента View 39
5. ТЕСТИРОВАНИЕ 41
5.1. Модульное тестирование 41
5.1.1. Модульное тестирование мобильного приложения 41
5.1.2. Модульное тестирование сервера 44
5.2. Автоматизированное UI тестирование 45
5.3. Функциональное тестирование 46
ЗАКЛЮЧЕНИЕ 47
ЛИТЕРАТУРА 48
ПРИЛОЖЕНИЯ 51
Приложение 1 51
Приложение 2 54
Приложение 3 58

ВВЕДЕНИЕ

Актуальность задачи

В последние десятилетия иностранные языки являются объектом пристального внимания и изучения не только ученых, но и людей, не связанных с научной деятельностью. При этом отмечается возрастающая роль, которую играют иностранные языки, в осуществлении влияния на сознание и деятельность широких слоев населения [17].

В современном мире нельзя недооценивать возрастающее влияние информационных технологий на повседневную жизнь и рабочую среду, где знание иностранных языков необходимо для полноценной и грамотной работы. Знание иностранного языка позволяет общаться с иностранцами, что способствует сотрудничеству и деловым связям, то есть расширению международных связей в целом, их укреплению [20].

Широкое распространение игр и развитие Интернета привели к возникновению геймификации как новой формы обучения и профессиональной подготовки. Геймификация – это способ воздействия на обучающихся. Идея геймификации состоит в использовании игрового подхода для того, чтобы сделать преподавание и обучение более занимательными, так как в играх приобретается необходимый опыт, устанавливаются безопасные границы, в пределах которых можно исследовать явления, обдумывать их и практиковаться, не боясь совершать ошибки, так как всегда можно победить в следующей игре. Игра – идеальная обучающая среда со встроенным разрешением на ошибку [21]. В электронном образовании игры могут заменить типовые задания, а при традиционном обучении разнообразить сложившийся формат занятий. Настоящая ценность геймификации состоит в том, чтобы игровой принцип способствовал созданию осмысленного учебного опыта. Для геймификации процесса обучения чаще всего используются [18]:

1) элементы соревнования – какие-либо игры между двумя и более игроками, доски лидеров и так далее;

2) очки опыта – то, что показывает прогресс пользователя в обучении. Это может быть реализовано в виде очков, которые выдаются за выполнение тех или иных действий, опыта, уровней и так далее;

3) награды. Они выдаются за выполнение тех или иных элементов приложения.

В связи с этим, в настоящее время широкое распространение получили мобильные программы для изучения иностранных языков и запоминания иностранных слов. Включение же элементов игры в процесс изучения иностранного языка существенно повышает мотивацию к процессу обучения.

Цель и задачи работы

Целью данной работы является разработка Android-приложения для изучения иностранных слов с элементами игры.

Задачи:

- изучить особенности разработки приложений для ОС Android;
- определить требования к программе;
- спроектировать архитектуру разрабатываемого приложения;
- разработать серверную часть;
- разработать мобильное приложение;
- выполнить тестирование.

Структура и объем работы

Работа состоит из введения, четырех разделов, заключения, библиографии и приложения. Объем работы составляет 50 страниц, объем библиографии – 32 источника, объем приложений – 11 страниц.

Раздел «Анализ предметной области» описывает анализ предметной области, в рамках которой проводилась работа.

Раздел «Анализ требований» описывает этап определения требований к разрабатываемой системе. Он включает в себя анализ функциональных и нефункциональных требований к системе, а также вариантов использования системы.

Раздел «Архитектура» описывает архитектуру системы. В данной главе подробно описана архитектура как мобильного приложения, так и сервера.

Раздел «Реализация» посвящен описанию всех компонентов системы, как мобильного приложения, так и сервера.

Раздел «Тестирование» описывает результаты модульного, автоматизированного UI и функционального тестирований.

В заключении описываются основные результаты работы.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Целью данной работы является разработка мобильного приложения для изучения иностранных языков, в частности для запоминания слов с элементами игры в процессе обучения. В рамках текущей работы предлагается следующее решение поставленной задачи: приложение, в функционал которого входит создание и редактирование коллекций слов, упражнения со словами из выбранной коллекции, возможность выполнять упражнения с другими людьми по сети Интернет.

1.1. Обзор аналогичных проектов

Проведен анализ уже используемых приложений для платформы Android, созданных для изучения иностранных слов. Для анализа было выбрано несколько наиболее популярных приложений.

LinguaLeo

Приложение состоит из нескольких частей. В «джунглях» доступны тысячи разнообразных текстов, основная задача которых – научить пользователей новым словам. Перевод того или иного слова можно получить, кликнув по нему [22, 28]. Слова можно добавлять в специальные списки для дальнейшего запоминания.

В Lingua Leo существуют курсы по грамматике. Однако, доступ к большинству открывается только за дополнительную плату. В наличии также различные тематические словари, упражнения по правильному произношению, запоминанию слов, а также грамматические тесты.

Достоинства:

- большое количество различных упражнений;
- возможность изучать грамматику;
- большое количество статей обучающих видео;
- геймификация процесса обучения.

Недостатки:

- часть контента платная;
- возможность изучения только английского языка.

Anki

Anki [26] – открытая мультиплатформенная программа для запоминания слов, основывающаяся на методике интервального повторения. Интервальное повторение – это техника обучения, при которой повторение материала (новых слов, формул и т.д.) совершается через определенные промежутки времени, таким образом, обеспечивая долговременное запоминание.

Достоинства:

– возможность изучения любого языка с помощью этого приложения (слово и его значение может внести сам пользователь);

– пользователь сам может добавить перевод слова.

Недостатки:

– отсутствие упражнений.

EngCards

В приложении порядка 3500 карточек в виде английских слов и картинок с профессиональным произношением от носителя языка. Слова подобраны для всех уровней знания английского языка в соответствии с одноименными курсами: Elementary, Pre-intermediate, Intermediate и UpperIntermediate, а также бонус в виде основных слов (существительные, прилагательные, числительные и глаголы).

Основная особенность программы – это настраиваемые периоды повторения слов. Если слово было выучено в упражнениях «тест» (нужно выполнить на 5 звезд) или «написать самому» (нужно выполнить на 4 звезды), все правильно выбранные и написанные слова будут считаться выученными. Слова исчезнут из обучения на 2 часа, потом вернутся через 2 часа. При повторном заучивании слово уже исчезнет на день, потом неделю, 2 месяца и год в соответствии с настройками [22].

Достоинства:

– поддержка 79 языков;

– стандартные наборы слов.

Недостатки:

– пользователь не может добавить слова, которых нет в коллекциях.

1.2. Обзор существующих средств разработки для платформы Android

На данный момент существует несколько платформ для создания приложений для Android:

1) Android Studio [25], основанная на языке Java и Android SDK [23];

2) Eclipse [27], поддерживающий язык Java и Android Development Tools [1];

3) Xamarin [30], основанная на языке программирования C# и платформе .NET.

Android SDK

Android SDK включает в себя разнообразные библиотеки, документацию и инструменты, которые помогают разрабатывать мобильные приложения для платформы Android.

1) API Android SDK – API библиотеки Android, предоставляемые для разработки приложений;

2) Документация SDK – включает обширную справочную информацию, детализирующую, что включено в каждый пакет и класс и как это использовать при разработке приложений;

3) AVD (Android Virtual Device) – интерактивный эмулятор мобильного устройства Android. Используя эмулятор, можно запускать и тестировать приложения без использования реального Android устройства;

4) Development Tools – SDK включает несколько инструментальных средств для разработки, которые позволяют компилировать и отлаживать создаваемые приложения;

5) Sample Code – Android SDK предоставляет типовые приложения, которые демонстрируют некоторые из возможностей Android, и простые программы, которые показывают, как использовать индивидуальные особенности API в вашем коде.

Android Studio

Android Studio – это интегрированная среда разработки (IDE) для работы с платформой Android, анонсированная 16 мая 2013 года на конференции Google I/O. Является официальной средой разработки приложений для платформы Android.

Особенности:

– расширенный редактор макетов: WYSIWYG, способность работать с UI компонентами при помощи Drag-and-Drop, функция предпросмотра макета на нескольких конфигурациях экрана;

– различные виды сборок и генерация нескольких .apk файлов;

– рефакторинг кода;

– статический анализатор кода (Lint), позволяющий находить проблемы производительности, несовместимости версий и другое;

– шаблоны основных макетов и компонентов Android;

– поддержка разработки приложений для Android Wear и Android TV;

– встроенная поддержка Google Cloud Platform, которая включает в себя интеграцию с сервисами Google Cloud Messaging и App Engine.

Eclipse

Eclipse – свободная интегрированная среда разработки кроссплатформенных приложений. Android Development Tools (ADT) – это плагин для Eclipse IDE, позволяющий разрабатывать приложения для платформы Android. Данная среда разработки позволяет создавать настолько же эффективные приложения для Android, но т.к. Android Studio является инструментом, специально созданным для мобильной разработки, то она позволяет проще и быстрее создавать приложения по сравнению с Eclipse.

Xamarin

Xamarin – это фреймворк для кроссплатформенной разработки мобильных приложений с использованием языка C#. Xamarin обладает следующими преимуществами и недостатками:

1) позволяет создавать кроссплатформенные мобильные приложения (iOS, Android, Windows Phone);

2) не требует знаний стандартных средств реализации Android приложений (основан на C# и платформе .NET);

3) платная лицензия;

4) неполная поддержка стандартных API платформы;

5) относительно низкая производительность.

Учитывая преимущества и недостатки рассмотренных инструментов, для реализации приложения была выбрана среда разработки Android Studio.

1.3. Обзор технологий для создания распределенного приложения

1.3.1. Концепция REST

REST [14] – это стиль построения архитектуры распределенного приложения. Данные в REST передаются посредством протокола HTTP в виде сообщений в формате HTML, XML, JSON и др. В основе REST лежат следующие принципы:

1) разделение ответственности клиента и сервера;

2) информация, содержащаяся в запросе, должна быть достаточной для его обработки и не требовать знаний о предыдущих запросах;

3) каждый источник данных (ресурс) обладает уникальным идентификатором, по которому можно получить или изменить состояние ресурса.

Для взаимодействия приложения с сервером используется API на основе REST. Такая архитектура имеет следующие плюсы:

1) поддержка многих форматов данных (HTML, XML, JSON и др.);

2) простота создания, относительно SOAP-сервисов;

3) независимость от платформы.

1.3.2. Технология WebSocket

WebSocket [29] – протокол полнодуплексной связи (может передавать и принимать одновременно) поверх TCP-соединения, предназначенный для обмена сообщениями между клиентом и веб-сервером в режиме реального времени.

WebSocket разработан для создания приложений в web-браузерах и web-серверах, но он может быть использован для любого клиентского или серверного приложения. Протокол WebSocket – это независимый протокол, основанный на протоколе TCP.

Для установления соединения WebSocket клиент и сервер используют протокол, похожий на HTTP. Клиент формирует особый HTTP-запрос, на который сервер отвечает определенным образом.

Преимущества WebSocket:

1) двустороннее соединение;
2) снижение используемого трафика;
3) безопасность;
4) минимальная задержка передачи данных.

Вывод

В ходе анализа предметной области были рассмотрены аналогичные проекты, а также средства и технологии разработки. В результате было решено использовать Android Studio, Android SDK и язык программирования Java для реализации мобильного приложения. Для реализации сервера было принято решение использовать технологии REST и WebSocket.

2. АНАЛИЗ ТРЕБОВАНИЙ

2.1. Функциональные требования к проектируемой системе

Функциональные требования определяют функциональность ПО, которую нужно построить, чтобы пользователи смогли выполнить свои задачи. В ходе анализа были выявлены следующие требования:

1) приложение LangLearn должно создавать и редактировать коллекции слов;

2) приложение LangLearn должно создавать различные виды заданий для указанной коллекции;

3) приложение LangLearn должно проверять ответы на задания;

4) приложение LangLearn должно регистрировать пользователя через e-mail либо через Facebook;

5) приложение LangLearn должно добавлять друзей;

6) приложение LangLearn должно выполнять задания с друзьями;

7) приложение LangLearn должно показывать лидеров по набранным очкам (опыту);

8) приложение LangLearn должно загружать коллекции слов с сервера;

9) приложение LangLearn должно отправлять коллекции слов на сервер для того, чтобы поделиться с другими пользователями.

2.2. Нефункциональные требования к проектируемой системе

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

1) приложение LangLearn должно быть написано на языке Java с использованием Android SDK;

2) приложение LangLearn должно работать на платформе Android версии 4.0 и выше;

3) сервер должен быть написан на языке Java с использованием фреймворка Spring Framework.

2.3. Диаграмма вариантов использования

В ходе анализа был выделен основной актер системы – *Пользователь* (рис. 1). Он может использовать все функции мобильного приложения, которые представлены ниже:

1) Выполнить вход: пользователь может осуществить вход в приложение;

2) Зарегистрироваться: пользователь может осуществить регистрацию в приложении через социальную сеть Facebook или электронную почту;

3) Создать коллекцию: пользователь может создать коллекцию слов;

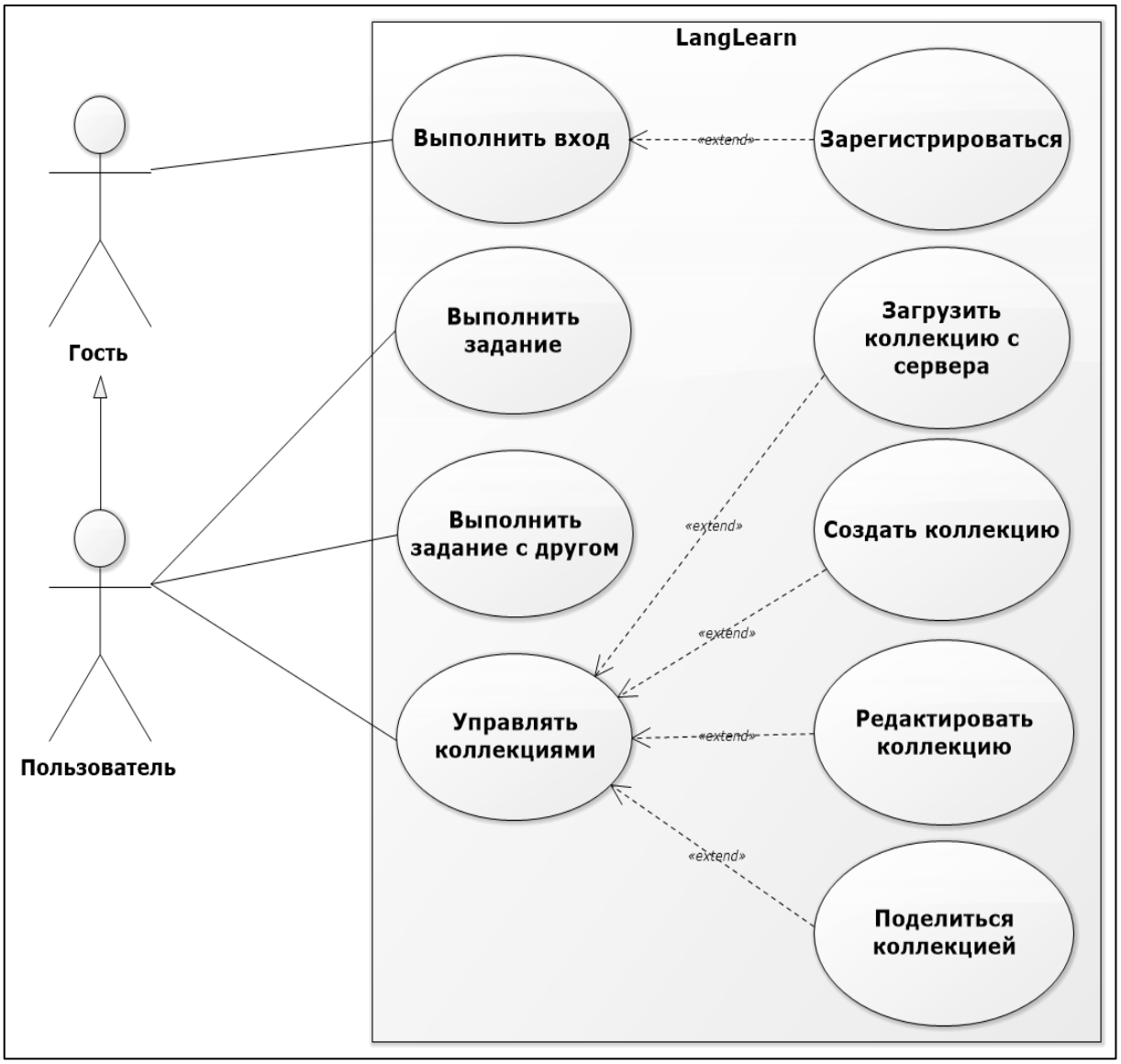


Рис. 1. Диаграмма вариантов использования

4) Редактировать коллекцию: пользователь может добавлять новые слова в коллекцию и удалять из нее старые. Можно удалить коллекцию целиком;

5) Выполнить задание: пользователь может выполнить с какой-либо определенной коллекцией, а может выполнить задание со случайными словами;

6) Выполнить задание с другом: пользователь может пригласить друга выполнить задание;

7) Загрузить коллекцию с сервера: пользователь может загрузить коллекцию с сервера;

8) Поделиться коллекцией: пользователь может загрузить на сервер свою коллекцию, чтобы к ней получили доступ другие пользователи;

9) Управлять коллекциями: пользователь может изменять коллекции.

Вывод

В ходе анализа были установлены основные функциональные и нефункциональные требования к системе, определены пользователь и варианты использования системы.

3. АРХИТЕКТУРА

На рис. 2 представлена архитектура системы. Ее основными компонентами являются сервер и клиентское приложение.

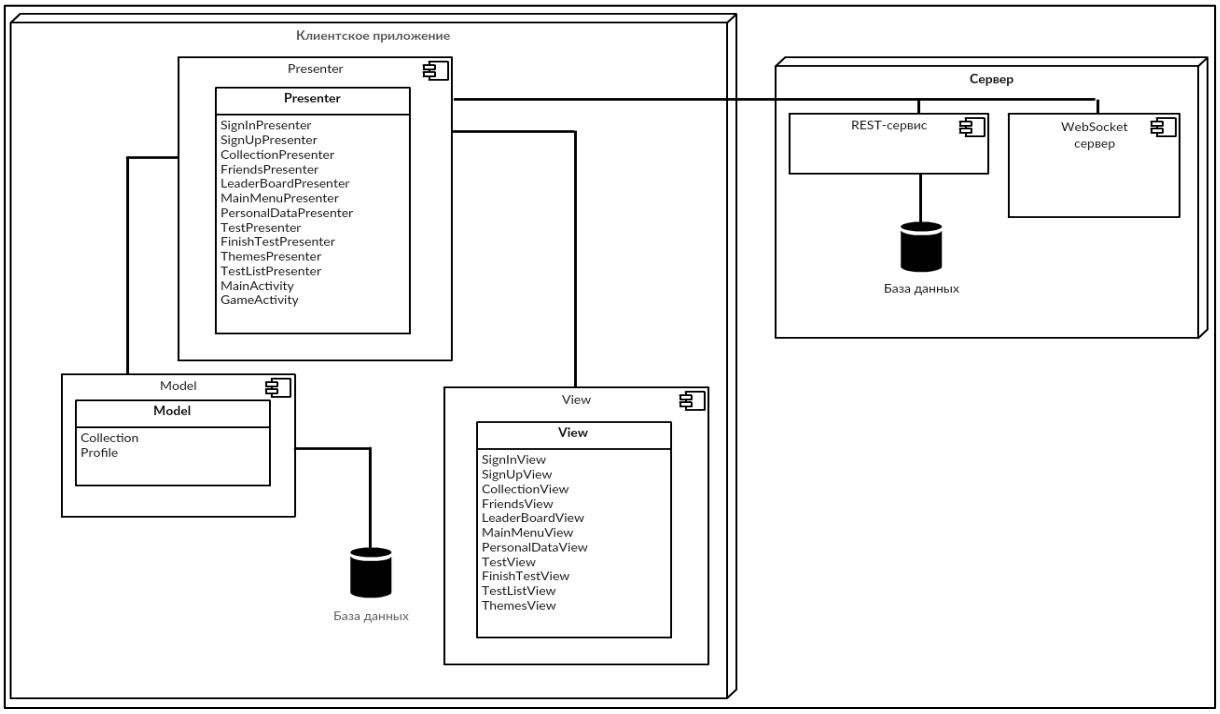


Рис. 2. Архитектура системы

3.1. Архитектура мобильного приложения

Мобильное приложение состоит из двух основных частей: базы данных и программных компонентов.

3.1.1. Проектирование базы данных

Для мобильного приложения необходимо создать базу данных для хранения пользовательских коллекций слов, чтобы у пользователя была возможность пользоваться приложением без подключения к сети интернет.

На рис. 3 приведена схема базы данных мобильного приложения.

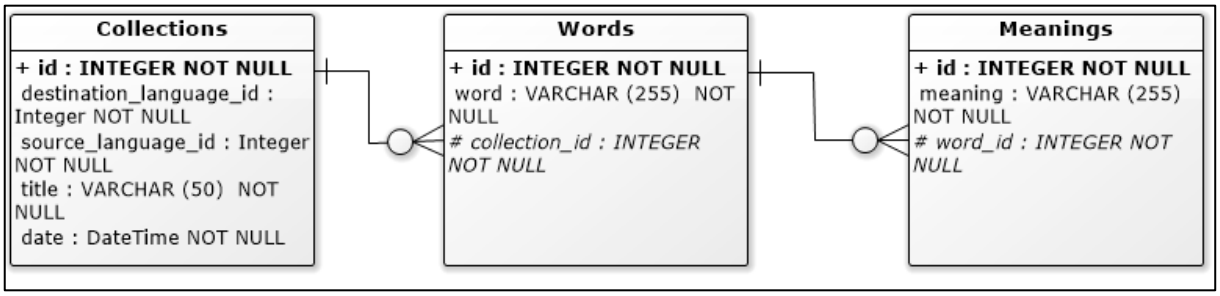


Рис. 3. Схема пользовательской базы данных

База данных состоит из трех таблиц: Collections, Words и Users. В таблице Collections хранится список коллекций пользователя, а также информация о ней. В таблице Words хранится список слов, а в таблице Meanings хранится список значений этих слов. В приложении 1 представлено подробное описание данных таблиц.

3.1.2. Компоненты мобильного приложения

Архитектура приложения будет использовать паттерн проектирования Model-View-Presenter [6] (рис. 2). Основная идея данного паттерна заключается в том, чтобы разделить модель данных приложения, пользовательский интерфейс и взаимодействие с пользователем на три отдельных компонента таким образом, чтобы модификация одного из компонентов оказывала минимальное воздействие на остальные. Данный паттерн предполагает наличие 3 основных типов классов:

1) Model – классы, отвечающие за хранение данных приложения, а также методы работы с этими данными;
2) View – классы, отвечающие за интерфейс приложения. Они отображает данные, которые передаются классами слоя Presenter, а также реагируют на действия пользователя (например, нажатие кнопки), передавая управление классу слоя Presenter;
3) Presenter – классы, обеспечивающие связь между Model и View. Реагируют на изменения в Model, а также обрабатывают действия пользователя, которые им передали классы слоя View.

В разрабатываемом приложении компонент View включает следующие экраны:

1) SignInView. Экран авторизации;
2) SignUpView. Экран регистрации;
3) CollectionView. Набор экранов для изменения коллекций слов;
4) TestView. Экран с текущим вопросом;
5) FinishTestView. Экран, показывающий результат теста;

6) FriendsView. Экран, показывающий результаты выполнения задания;

7) PersonalDataView. Экран, показывающий информацию о профиле;

8) MainMenuView. Главное меню приложения;

9) LeaderBoardView. Экран с доской лидеров;

10) TestListView. Экран со списком тестов;

11) ThemesView. Экран для выбора тем.

Компонент Presenter состоит из:

1) SignInPresenter. Отвечает за авторизацию;

2) SignInPresenter. Отвечает за регистрацию;

3) CollectionsPresenter. Обеспечивает редактирование коллекций;

4) FriendsPresenter. Обеспечивает добавление и удаление друзей;

5) LeaderBoardPresenter. Показывает лидеров;

6) MainMenuPresenter. Обрабатывает действия в главном меню;

7) MainActivity. Основная активность приложения;

8) GameActivity. Активность во время режима соревнования;

9) PersonalDataPresenter. Обеспечивает отображение данных пользователя;

10) TestPresenter. Обеспечивает обработку действий пользователя во время теста;

11) FinishTestPresenter. Обеспечивает обработку завершения теста;

12) TestListPresenter. Обеспечивает обработку действий пользователя в меню списка тестов;

13) ThemesPresenter. Обеспечивает обработку действий пользователя при выборе тем.

Компонент Model состоит из:

1) Collection. Отвечает за хранение коллекций пользователя;

2) Profile. Отвечает за хранение информации о пользователе.

Архитектура режима соревнования

Отдельная часть приложения – режим соревнования. Для него создается отдельная активность GameActivity. В нее передаются данные, необходимые для создания теста, имя и уникальный идентификатор соперника, в ней же устанавливается соединение с Socket-сервером. Для отображения данных используются следующие классы: TestView и FinishTestView. Для обработки пользовательских действий используются классы: TestPresenter и FinishTestPresenter. Для создания тестов используется класс TestCreator. На рис. 4 изображена диаграмма классов данного режима.

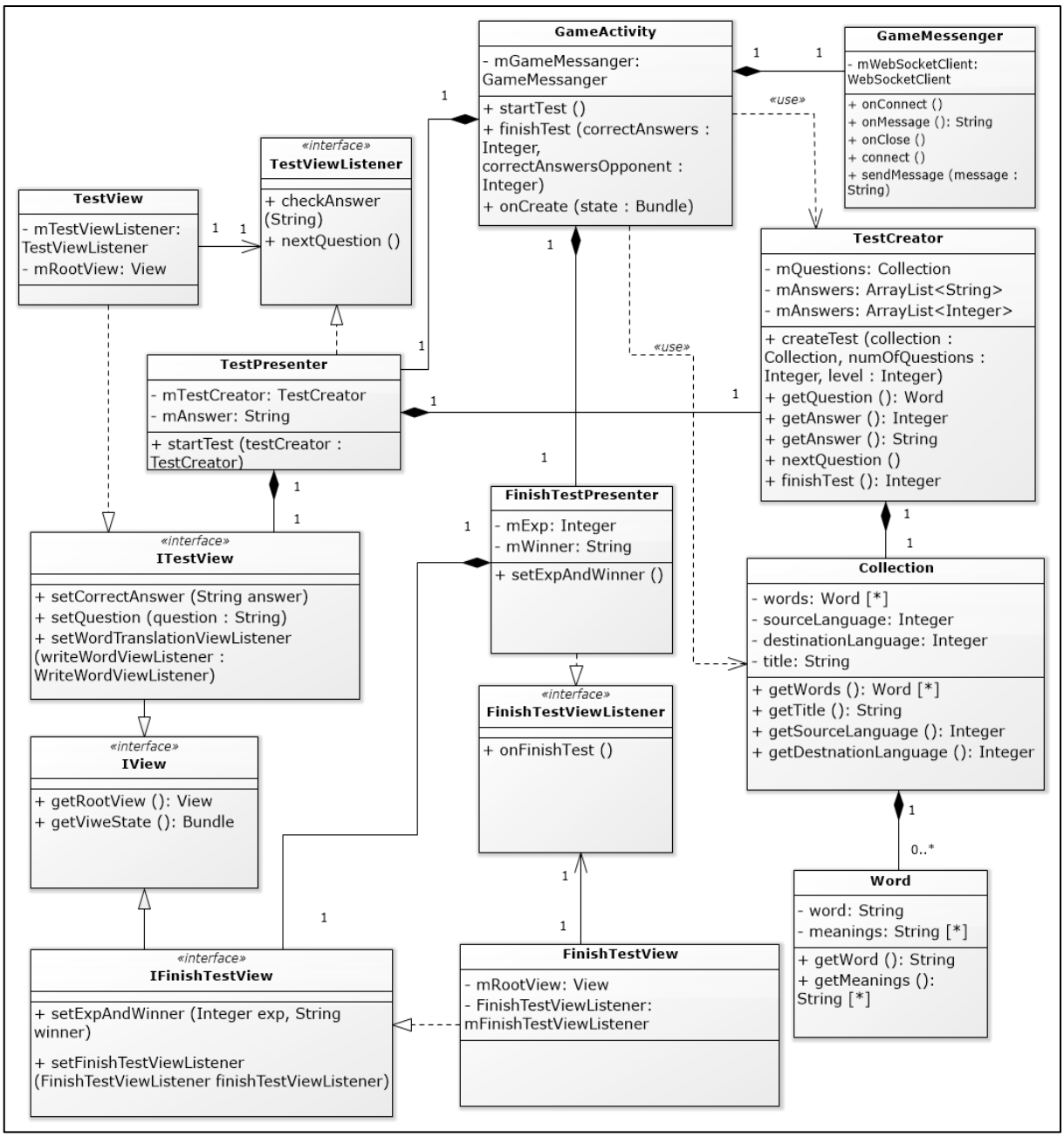


Рис. 4. Схема режима соревнования

3.2. Архитектура сервера

Сервер состоит из трех основных компонентов: база данных, REST-сервис и WebSocket-сервер.

3.2.1. Проектирование базы данных

Для реализации сервера необходимо создать базу данных для хранения информации о пользователях и о коллекциях слов. На рис. 5 представлена схема базы данных сервера.

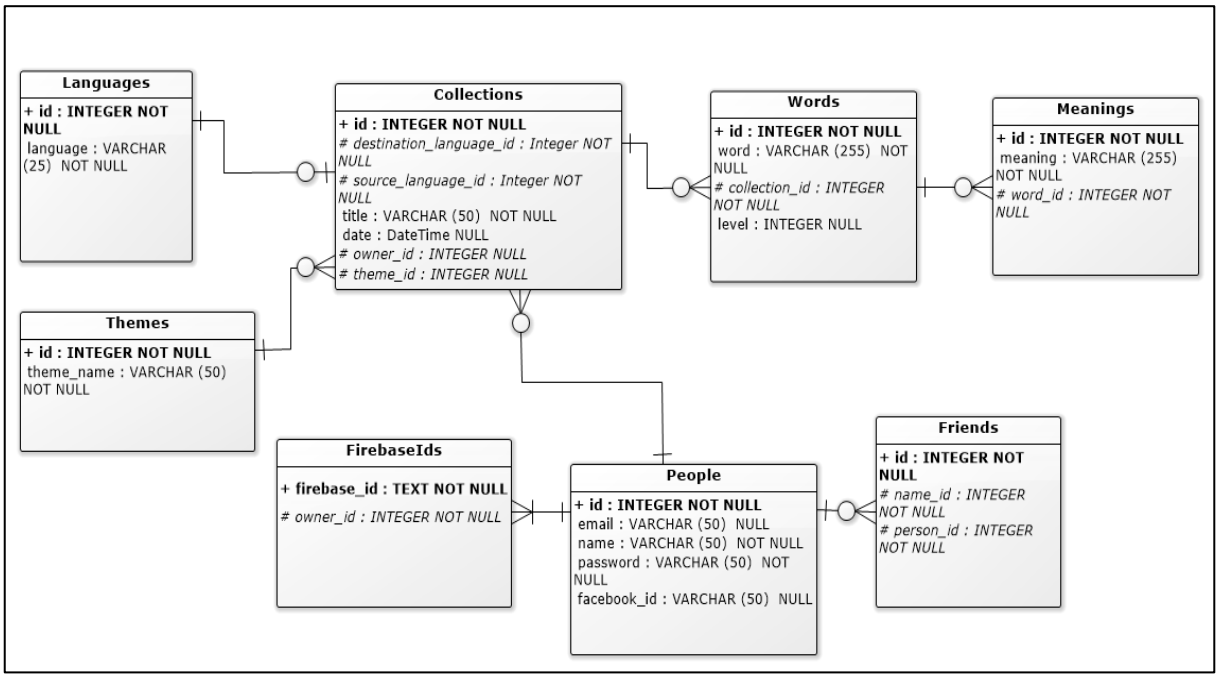


Рис. 5. Схема глобальной базы данных

База данных сервера содержит восьми таблиц: Collections, Words, Meanings, Themes, People, Friends, FirebaseIds и Languages. В таблице Collections хранится список коллекций сервера, а также информация о ней. В таблице Words хранится список слов, а в таблице Meanings хранится список значений этих слов. В таблице Themes хранятся темы коллекций (еда, мебель и тому подобное). В таблице People хранится список пользователей и информация о них, в таблице Friends – список друзей пользователей, в таблице FirebaseIds хранятся id пользовательских устройств для отправки Push-уведомлений, а в таблице Languages – список языков. В приложении 1 представлено подробное описание данных таблиц.

3.2.2. Архитектура REST-сервиса

REST-сервис обрабатывает http-запросы клиента. Такие запросы приходят на один из следующих классов (в зависимости от адреса запроса): CollectionsController, RegistrationController, UserController и LeadersController. Они получают данные из http-запросов и передают их классам CollectionsService и PersonService для дальнейшей обработки. CollectionsService и PersonService – сервис классы для реализации бизнес-логики. Они используют класс DAO<T> для доступа к базе данных. Классы entity являются данными, которые хранятся в базе данных проекта. На рис. 6 изображена диаграмма классов REST-сервиса.

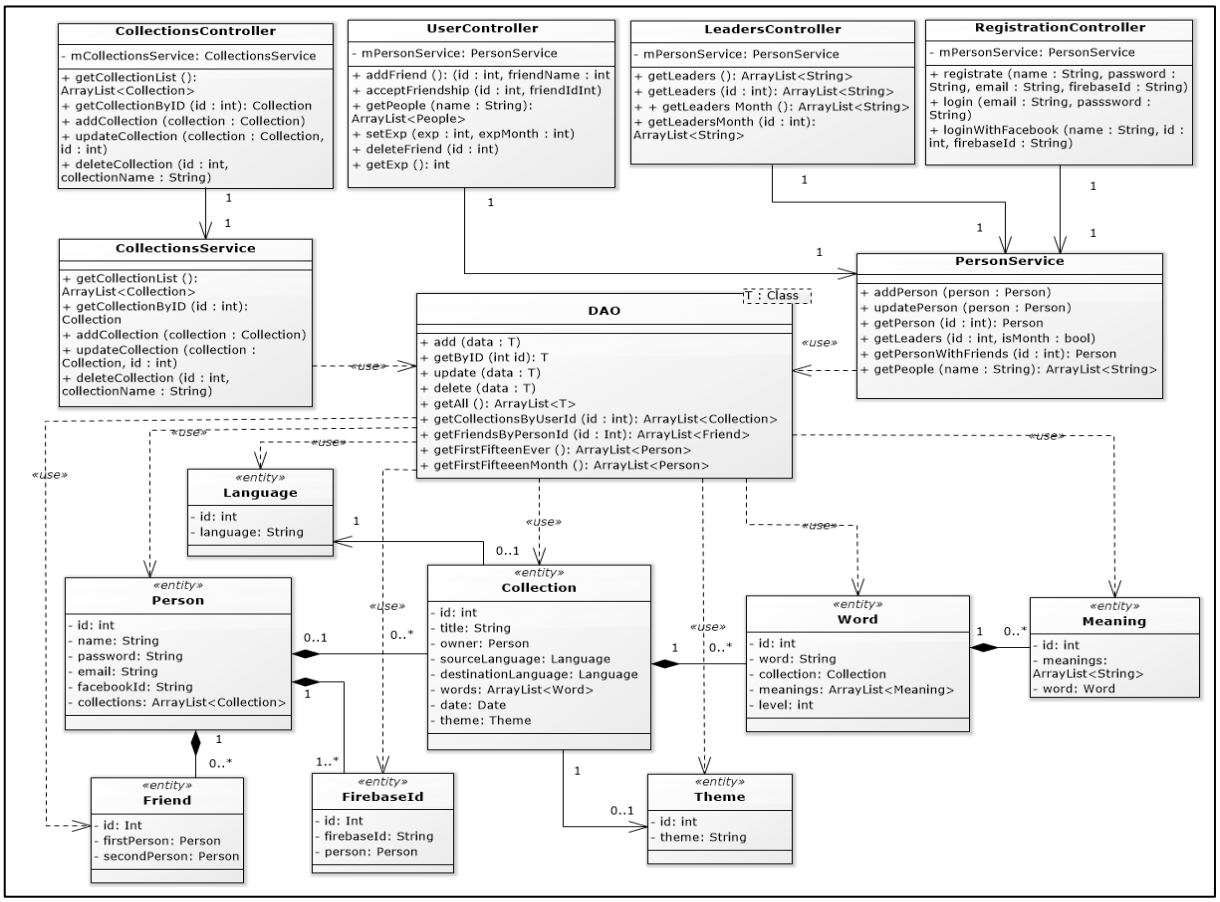


Рис. 6. Диаграмма классов REST-сервиса

3.2.3. Проектирование WebSocket-сервера

WebSocket-сервер необходим для реализации режима соревнования. К нему подключаются пользователи, которые захотят выполнить то или иное задание вместе с другом. Он состоит из следующих основных клас-

сов: SocketServer, который открывает, хранит и завершает сессии пользователей, а также принимает сообщения от них. При поступлении нового сообщения он передает управление классу MessageManager, который отвечает на это сообщение.

Вывод

В результате была спроектирована архитектура мобильного приложения и сервера системы. Мобильное приложение включает в себя компоненты Model, View и Presenter, а также базу данных. Сервер включает в себя REST-сервис, WebSocket-сервер и базу данных.

4. РЕАЛИЗАЦИЯ

4.1. Реализация серверной части

4.1.1. База данных

Создание базы данных, ее поддержка и обеспечение доступа пользователей к ней осуществляется централизованно с помощью специального программного инструментария – системы управления базами данных.

Система управления базами данных (СУБД) [24] – это комплекс программных и языковых средств, необходимых для создания баз данных, поддержания их в актуальном состоянии и организации поиска в них необходимой информации.

Для реализации базы данных на сервере было принято решение использовать СУБД MySQL [11].

4.1.2. Реализация взаимодействия с базой данных

Для реализации взаимодействия с базой данных было решено использовать специальную ORM-библиотеку. ORM (англ. Object-Relational Mapping) [16] – технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных». Цель: работа с данными в терминах классов, а не таблиц. Для этого была выбрана библиотека Hibernate. *Hibernate* [3] – библиотека для языка программирования Java, предназначенная для решения задач объектно-реляционного отображения (ORM).

Преимущества Hibernate:

1) Hibernate устраняет повторяющийся код, скрывает от разработчика код, необходимого для управления ресурсами и позволяет сосредоточиться на бизнес логике;

2) Hibernate поддерживает ленивую инициализацию, используя proxy объекты, и выполняет запросы к базе данных только по необходимости;

3) Hibernate может использовать SQL, для оптимизации запросов;

4) поддержка кэширования.

Для работы с библиотекой Hibernate необходимо, в первую очередь, создать файл с описанием базы данных: hibernate.cfg.xml. В нем содержится основная информация о базе данных, а также сами таблицы, которые входят в состав БД. Далее необходимо разработать классы, которые соответствуют таблицам в базе данных для работы с таблицами, а именно для сохранения и получения данных, поля которых соответствуют колонкам таблиц базы данных

Кроме того, для получения доступа к базе данных необходимы классы DAO (Data Access Object – объект доступа к данным). Они позволяют добавлять, обновлять и удалять данные из соответствующих таблиц. Для этого был создан универсальный класс DAO<Type>, который позволяет к определенной таблице в базе данных в зависимости от переданного параметра Type. На рис. 7 приведен метод класса DAO<Type> add.

В данном методе в первую очередь открывается сессия для доступа к базе данных, затем происходит транзакция, в ходе которой сохраняется нужное значение в нужной таблице.

```
public void add(Type value) throws Exception {
    Session session;
    try {
        session = HibernateU-
til.getSessionFactory().getCurrentSession();
        session.beginTransaction();
        session.save(value);
        session.getTransaction().commit();
    } catch (Exception e) {
        throw e;
    }
}
```

Рис. 7. Метод add класса DAO<Type>

4.1.3. Реализация REST-сервиса

Для реализации REST-сервиса используется платформа Spring. Платформа Spring [19] – популярная платформа приложений с открытым кодом, предназначенная для упрощения разработки для J2EE. Она состоит

из контейнера, платформы управления элементами и набора интегрируемых служб для веб-интерфейсов пользователя, транзакций и сохранения состояния. В состав платформы Spring входит Spring Web – расширяемая платформа MVC для создания веб-приложений. Spring MVC построен вокруг центрального сервлета (DispatcherServlet), который распределяет запросы по контроллерам, а также предоставляет другие возможности при разработке веб приложений. Сервлет (Servlet) [31] – это java-программы, которые выполняются на серверной стороне Web-приложения. На рис. 8 приведен класс инициализации приложения и сервлета. В данном классе также указываются адреса, которые будут обрабатываться сервлетом.

```
public class Initializer implements WebApplicationInitializer {

    private static final String DISPATCHER_SERVLET_NAME = "dis-
patcher";

    @Override
    public void onStartup(ServletContext servletContext) throws
ServletException {
        AnnotationConfigWebApplicationContext ctx = new Annota-
tionConfigWebApplicationContext();
        ctx.register(WebAppConfig.class);
        servletContext.addListener(new ContextLoaderListen-
er(ctx));
        ctx.setServletContext(servletContext);
        ServletRegistration.Dynamic servlet = servletCon-
text.addServlet(DISPATCHER_SERVLET_NAME, new DispatcherServ-
let(ctx));
        servlet.addMapping("/");
        servlet.setLoadOnStartup(1);
    }
}
```

Рис. 8. Класс инициализации приложения и сервлета

После того, как запрос был проанализирован сервлетом, он перенаправляется на соответствующий контроллер, в зависимости от адреса. Контроллер (Controller) [13] – это ключевой интерфейс в Spring. Контроллер обрабатывает запросы пользователей, взаимодействуя с уровнем обслуживания, обновляя модель и направляя пользователей на соответствующие представления в зависимости от результатов выполнения. Для реализации контроллера нужно создать класс и указать специальные аннотации: @RestController и @RequestMapping. Первая указывает, что класс яв-

ляется контроллером, который возвращает объект в формате JSON или XML, во второй указывается адрес, запросы с которого будут обрабатываться этим классом. Кроме того, аннотация @RequestMapping указывается и у методов класса-контроллера. Кроме адреса, для метода могут быть указаны определенные параметры, например, тип http запроса. На рис. 9 показана реализация обработки запроса на получение списка коллекций с сервера. На сервер приходит запрос get, сервер обращается к базе данных, получает список коллекций, затем формирует ответное сообщение и отправляет его клиенту.

Для реализации аутентификации пользователей используется Spring Security. Spring Security [9] – это Java/JavaEE framework, предоставляющий механизмы построения систем аутентификации и авторизации, а также другие возможности обеспечения безопасности для корпоративных приложений, созданных с помощью Spring Framework. Алгоритм аутентификации:

1) пользователю предлагается войти в систему, предоставив имя (логин или email) и пароль. Имя пользователя и пароль объединяются в экземпляр класса UsernamePasswordAuthenticationToken(экземпляр интерфейса Authentication) после чего он передается экземпляру AuthenticationManager для проверки;

```
@RequestMapping(method = RequestMethod.GET)
public ArrayList<Collection> getCollectionList() {
    List<CollectionPojo> pojoCollections = new Ar-
rayList<CollectionPojo>();
    ArrayList<Collection> collections = null;
    try {
        pojoCollections = mCollectionService.getAllCollections();
        collections = new ArrayList<Collection>();
        for (int i = 0; i < pojoCollections.size(); i++) {
            Collection collection = pojoCollec-
tions.get(i).toCollection();
            collections.add(collection);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return collections;
}
```

Рис. 9. Метод для получения списка коллекций слов с сервера

2) если пароль не соответствует имени пользователя, будет выброшено исключение BadCredentialsException с сообщением “Bad Credentials”;

3) если аутентификация прошла успешно возвращает полностью заполненный экземпляр Authentication;

4) для пользователя устанавливается контекст безопасности путем вызова метода SecurityContextHolder.getContext().setAuthentication(…), куда передается объект, который вернул AuthenticationManager.

На рис. 10 представлен класс с настройками для Spring Security. Данный класс получает доступ к базе данных (mDataSource) и позволяет пользователям аутентифицироваться по идентификатору (который хранится в клиентском приложении) и паролю, либо по электронному адресу и паролю. В базе данных на сервере пароли хранятся в зашифрованном виде (для шифрования используется алгоритм bcrypt), поэтому, создается и объект класса PasswordEncoder для шифрования приходящего пароля и последующей проверки, которая определена в методе configureGlobal. В методе configure определяются адреса, к которым ограничивает доступ система защиты.

4.1.4. Реализация Socket-сервера

Для создания Socket-сервера используется WebSocket API в Java EE 7 для обеспечения связи между клиентом и сервером в реальном времени. Ниже приведены некоторые из особенностей, которые делают эту библиотеку удобной для реализации Socket-сервера [4]:

1) возможность использования аннотаций для создания POJO-классов для управления жизненным циклом сервера;

2) совместимость с другими Java EE технологиями;

3) удобство при написании сервера.

Для создания WebSocket-сервера необходимо создать класс и указать для него аннотацию: @ServerEndpoint и указать URI в качестве параметра. Эта аннотация показывает, что данный класс является классом-терминалом и может принимать сообщения по протоколу WebSocket.

```
@Configuration
@EnableWebSecurity
public class AppSecurityConfig extends WebSecurityConfigurerAdapter {

    @Autowired
    DataSource mDataSource;

    @Autowired
    public void configureGlobal(AuthenticationManagerBuilder auth) throws
Exception {
        auth.jdbcAuthentication().dataSource(mDataSource)
                .passwordEncoder(passwordEncoder())
                .usersByUsernameQuery("select id,password,enabled from
people where id=?")
                .authoritiesByUsernameQuery("select id,role from people
where id=?");
        auth.jdbcAuthentication().dataSource(mDataSource)
                .passwordEncoder(passwordEncoder())
                .usersByUsernameQuery("select email,password,enabled from
people where email=?")
                .authoritiesByUsernameQuery("select email,role from peo-
ple where email=?");

    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests()

.antMatchers("/collections/**").access("hasRole('ROLE_REGISTRATED')")

.antMatchers("/user/**").access("hasRole('ROLE_REGISTRATED')")
                .and().formLogin();
    }

    @Bean
    public PasswordEncoder passwordEncoder() {
        PasswordEncoder encoder = new BCryptPasswordEncoder();
        return encoder;
    }
}
```

Рис. 10. Настройка Spring Security

В WebSocket-сервере реализованы методы onMessage(String message, Session session), onOpen(Session session) и onClose(Session session), а также метод getQueryMap для обработки запроса при подключении нового клиента. При подключении нового клиента к серверу, вызывается метод onOpen, который добавляет новую сессию в список сессий mSessions. После этого, при получении сообщений от клиента, вызывается метод onMessage. В нем входящее сообщение обрабатывается специальным объектом mMessageManager класса MessageManager. Этот класс принимает JSON-

сообщения от клиентов и отправляет ответы в зависимости от полученного сообщения. Метод onClose вызывается при завершении сеанса: он удаляет текущую сессию из списка сессий mSessions. На рис. 11 показан методы WebSocket-сервера.

```
@OnOpen
public void onOpen(Session session) {
    Map<String, String> queryParams = getQuery-
Map(session.getQueryString());
    String name = "";
    if (queryParams.containsKey("name")) {
        name = queryParams.get("name");
        try {
            name = URLDecoder.decode(name, "UTF-8");
        } catch (UnsupportedEncodingException e) {}
        mSessions.put(name, session);
        try {
            session.getBasicRemote().sendText("Connected");
        } catch (IOException e) {}
    } else {
        try {
            session.close();
        } catch (IOException e) {}
    }
}

@OnMessage
public void onMessage(String message, Session session) {
    mMessageManager.onMessage(mSessions, message, session, mNameSession-
Pair);
}

@OnClose
public void onClose(Session session) {
    mSessions.remove(session);
}
```

Рис. 11. Методы WebSocket-сервера

4.2. Реализация клиентской части

При разработке приложения для платформы Android, возникает вопрос выбора минимального поддерживаемого уровня API. Уровень API – числовое значение, которое уникально идентифицирует версию API фреймворка платформы Android [15]. С уровнем API повышается количество удобных средств разработки и различных функций, которые может выполнять приложение, но уменьшается количество устройств, которые будут его поддерживать.

В связи со статистикой, показанной на рис. 12, было решено выбрать

15-ый уровень API для разработки, т.к. это минимальный уровень, поддерживаемый всеми наиболее используемыми версиями.

В результате стало возможным использование некоторых достаточно удобных функциональных возможностей, а приложение осталось доступным для большинства устройств.

Для хранения данных пользователя используется база данных SQLite [10]. Поддержка приложениями базы данных этого типа встроена в платформу Android.

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	1.0%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	1.1%
4.1.x	Jelly Bean	16	4.0%
4.2.x		17	5.9%
4.3		18	1.7%
4.4	KitKat	19	22.6%
5.0	Lollipop	21	10.1%
5.1		22	23.3%
6.0	Marshmallow	23	29.6%
7.0	Nougat	24	0.5%
7.1		25	0.2%

Рис. 12. Относительное число устройств, использующих определенную версию ОС Android (по состоянию на 9 января 2017 г.)

4.2.1. Реализация компонента Presenter

Компонент Presenter представляет собой набор классов, которые контролируют ввод данных пользователем и используют модель и представление для реализации необходимой реакции. В ходе создания приложения были реализованы следующие классы этого типа:

1) SignInOrUpPresenter. Он отвечает за переход к экранам авторизации и регистрации по нажатию соответствующей кнопки;

2) SignInPresenter. Отвечает за процесс авторизации. Когда пользователь ввел данные и нажал кнопку «Войти», объект данного класса отправит запрос REST-сервису с помощью класса AuthenticationManager. Класс AuthenticationManager является потомком класса MessageManager. Класс MessageManager и его наследники используются для обращения к REST-сервису;

3) SignUpPresenter. Отвечает за процесс регистрации. На рис. 13 изображен метод onSignUp данного класса;

```
@Override
public void onSignUp(String userName, String password, String password-
Again, String email) {
    if (userName.length() < 5) {
        mSignUpView.setHintName(getString(R.string.tooShortName));
        return;
    }
    if (password.length() < 6) {
        mSig-
nUpView.setHintPassword(getString(R.string.tooSimplePassword));
        return;
    }
    if (!password.equals(passwordAgain)) {
        mSig-
nUpView.setHintPasswordAgain(getString(R.string.notEqualPasswords));
        return;
    }
    if (!validate(email)) {
        mSignUpView.setHintEmail(getString(R.string.notEmail));
        return;
    }
    SharedPreferences pref = getActivi-
ty().getApplicationContext().getSharedPreferences(Config.SHARED_PREF, 0);
    String regId = pref.getString(REG_ID, null);
    mAuthenticationManger.sendRegistrateMessage(userName,password,email,
regId,this);
}
```

Рис. 13. Метод onSignUp

4) ThemesPresenter. Класс, отвечающий за обработку действий, связанных выбором тем;

5) MainMenuPresenter. Класс, обрабатывающий дейсвия пользователя в главном меню. На рис. 14 показан метод onRepeatWords данного класса, который создает тест для повторения слов. В данном методе из базы данных берутся слова 4-8 уровня, объект RepeatingTestCreator, который яв-

ляется реализацией интерфейса ITestCreator, который отвечает за создание тестов. После этого вызывается WordTranslationPresenter для создания соответствующего представления;

```
@Override
public void onRepeatWords() {
    WordsCRUD wordsCRUD = new WordsCRUD(getActivity());
    Collection collection = new Collection("");
    collection.setCollection(wordsCRUD.getAllWords());
    RepeatingTestCreator repeatingTestCreator = new RepeatingTestCrea-
tor(getActivity());

    try {
        repeatingTestCreator.createTest(collection, 20, 4, 8);
        WordTranslationPresenter wordTranslationPresenter = new Word-
TranslationPresenter();
        changeFragment(wordTranslationPresenter,TEST_LIST_TAG);
        wordTranslationPresenter.startTest(repeatingTestCreator);
    } catch (TooSmallNumberOfWords tooSmallNumberOfWords) {
        Toast.makeText(getActivity(), getActivi-
ty().getResources().getString(R.string.noWordsToRepeat),
Toast.LENGTH_SHORT).show();
    }
}
```

Рис. 14. Метод onRepeatWords

6) WordTranslationPresenter и WriteWordPresenter. Отвечают за обработку ответов пользователя и передачу соответствующему представлению вопросы и ответы. На рис. 15 приведены некоторые из методов класса WordTranslationpresenter (методы проверки ответа и перехода к следующему вопросу);

7) FinishTestPresenter. Отвечает за сохранение результатов теста и отправления полученного опыта на сервер, при условии наличия Интернет-соединения;

8) TestListPresenter. Взаимодействует с представлением TestListView и обеспечивает отображение доступных тестов для выбранной коллекции;

9) CollectionListPresneter, WordListPresenter, WordPresneter и DownloadCollectionPresenter. Отвечают за добавление, удаление и редактирование коллекций, слов и значений слов;

```
public void checkAnswerOrNextQuestion(int num, boolean check) {
    if (check) {
        mWordTranslationView.changeIntoRed(num);
        int correctAnswer = mTestCreator.getAnswer(num - 1);
        mWordTranslationView.changeIntoGreen(correctAnswer + 1);
    } else {
        mWordTranslationView.changeIntoUsualColor(getResources());
        nextQuestion();
    }

}

public void nextQuestion() {
    if (mTestCreator.
            moveToNextQuestion()) {
        Word word = mTestCreator.getQuestion();
        if (word.getMeanings().size() > 3) {
            mWordTransla-
tionView.setQuestion(word.getWord(),word.getMeanings().get(0),word.getM
eanings().get(1),

word.getMeanings().get(2),word.getMeanings().get(3));
        } else {
            mTestCreator.previousQuestion();
            continueBigTest(mTestCreator);
        }
    } else {
        int exp = mTestCreator.finishTest();
        mActivity.finishTest(exp, null);
    }

}
```

Рис. 15. Методы перехода к следующему вопросу класса WordTranslationPresenter

10) PersonalDataPresenter. Взаимодействует с классом PersonalDataView для отображения данных пользователя;

11) FriendsPresenter. Взаимодействует с классом FriendsListView для отображения, добавления и удаления друзей. Для того, чтобы удалить друга, необходимо зажать его имя и нажать в появившемся меню кнопку «Удалить». Данное меню реализовано с помощью ContextMenu;

12) LeaderBoardPresenter. Взаимодействует с классом LeaderBoardView для отображения списка лидеров;

13) MainActivity. Основная активность приложения. Отвечает за начальную загрузку приложения, за обработку клавиши назад, за работу выдвигающегося меню для навигации по приложению;

14) GameActivity. Вторая активность приложения. Отвечает за работу приложения во время режима соревнования: устанавливает соединение с Socket-сервером, обрабатывает нажатия на кнопку назад и т.д.

4.2.2. Реализация компонента Model

Компонент Model предоставляет данные и методы работы с этими данными.

Модель Collection отвечает за хранение коллекций слов пользователя. Для работы с этой моделью реализованы следующие классы: Collection и Word. У каждой коллекции (Collection) может быть несколько слов (Word) (или не одного), слова не могут существовать без коллекции. Также у слова присутствуют поля: само слово, уникальный идентификатор слова, значения слова, уровень слова, дата последнего обновления слова в базе данных и идентификатор коллекции.

Для хранения этих данных используется база данных SQLite. Для доступа к базе данных реализован класс CollectionsDb, который является наследником класса SQLiteOpenHelper. Для работы этого класса необходимо реализовать методы onCreate и onUpgrade. В методе onCreate (рис. 16) описана структура базы данных приложения. Он вызывается при создании базы данных. Метод onUpgrade содержит изменения в базе данных, и вызывается, если новая версия базы данных не соответствует текущей. В базе данных приложения присутствует три таблицы: Collections, Words и Meanings. Для доступа к таблицам баз данных реализованы соответствующие классы: CollectionsCRUD, WordsCRUD и MeaningsCRUD. В них реализованы методы добавления, обновления, получения и удаления строк соответствующих таблиц. На рис. 17 представлен метод добавления пустой коллекции.

Для создания тестов реализован набор классов, которые являются реализациями интерфейса ITestCreator (абстрактный класс TestCreator, класс для создания тестов «Слово - перевод» WordTranslationCreator и т.д.).

В нем описаны метод для создания теста, перехода к следующему вопросу, получения ответа на текущий вопрос и т.д.

```
@Override
public void onCreate(SQLiteDatabase db) {
    db.execSQL("create table " + COLLECTIONS + "("
            + "id integer primary key autoincrement,"
            + "name text not null unique,"
            + "source_language integer,"
            + "destination_language integer,"
            + "date long" + ");");
    db.execSQL("create table " + WORDS + "("
            + "id integer primary key autoincrement,"
            + "name text not null,"
            + "level integer,"
            + "collection_id integer not null,"
            + "date long,"
            + "unique (name,collection_id)"
            + "FOREIGN KEY (collection_id) REFERENCES"
            + COLLECTIONS + "(id));");
    db.execSQL("create table " + MEANINGS + "("
            + "id integer primary key autoincrement,"
            + "word_id integer not null,"
            + "meaning text"
            + "FOREIGN KEY (word_id) REFERENCES"
            + WORDS + "(id));");
}
```

Рис. 16. Метод onCreate класса CollectionsDb

```
public void insertEmptyCollection(Collection collection) throws Al-
readyExistingCollection {
    ContentValues cv = new ContentValues();
    CollectionsDb collectionsDb = new CollectionsDb(mContext);
    SQLiteDatabase db = collectionsDb.getWritableDatabase();
    cv.put(NAME, collection.getTitle());
    cv.put(SOURCE_LANGUAGE, collection.getSourceLanguageId());
    cv.put(DESTINATION_LANGUAGE, collec-
tion.getDestinationLangugaeId());
    try {
        db.insert(COLLECTIONS, null, cv);
    } catch (Exception e) {
        throw new AlreadyExistingCollec-
tion(ALREADY_EXISTING_COLLECTION);
    }
    db.close();
    collectionsDb.close();
}
```

Рис. 17. Метод insertEmptyCollection класса CollectionsCRUD

Для доступа к REST-сервису реализован класс MessageManager, а также классы, которые его используют для получения данных с сервера (PersonalDataMessageManager, CollectionsMessageManager и т.д.). В классе

MessageManager реализованы методы GET, PUT, POST и DELETE. На рис. 18 представлена реализация метода GET (sendGetRequest) класса MessageManager.

```
protected String sendGetRequest(final String urlString, final
String urlParameters) {
    java.net.URL url;
    HttpURLConnection httpURLConnection;
    BufferedReader bufferedReader = null;
    String line;
    String result = "";
    try {
        url = new URL(urlString);
        httpURLConnection = (HttpURLConnection)
url.openConnection();
        httpURLConnection.setRequestProperty("Cookie", urlParame-
ters);
        httpURLConnection.setRequestMethod("GET");
        bufferedReader = new BufferedReader(new InputStreamRead-
er(httpURLConnection.getInputStream()));
        while ((line = bufferedReader.readLine()) != null) {
            result += line;
        }
    } catch (Exception e) {
        Log.e(ERROR, e.toString());
    } finally {
        try {
            if (bufferedReader != null)
                bufferedReader.close();
        } catch (IOException e) {}
    }
    return result;
}
```

Рис. 18. Метод sendGetRequest класса MessageManger

Для хранения данных пользователя (имя, опыт ,заработанный до последней синхронизации с сервером и т.д.) был реализован класс Profile. Он использует класс SharedPreferences для хранения необходимых данных. На рис. 19 представлены методы сохранения и получения опыта пользователя.

```
public void saveExp(int exp, int expMonth) {
    mSharedPreferences.edit()
            .putString(EXP, Encryptor.encrypt(exp))
            .putString(EXP_MONTH, Encryptor.encrypt(expMonth))
            .putString(MONTH, (new GregorianCalen-
dar()).get(Calendar.MONTH) + "." + (new GregorianCalen-
dar()).get(Calendar.YEAR))
            .commit();
}
public int getExp() {
    return Encryptor.decrypt(mSharedPreferences.getString(EXP,""));
}
```

Рис. 19. Методы сохранения и получения опыта пользователя

4.2.3. Реализация компонента View

Компонент View представляет собой набор экранов, с которыми взаимодействует пользователь. Для создания интерфейса разработчик может как перетащить из списка на экран, либо заполнить специальный файл формата XML. Конкретные объекты, которые получают сообщения, например, кнопки, на которые может нажать пользователь, указываются в коде приложения.

Разработанное приложение включает в себя 16 экранов. На рис. 20 представлена схема всех экранов и связей между ними, скриншоты самих экранов – в приложении 2.

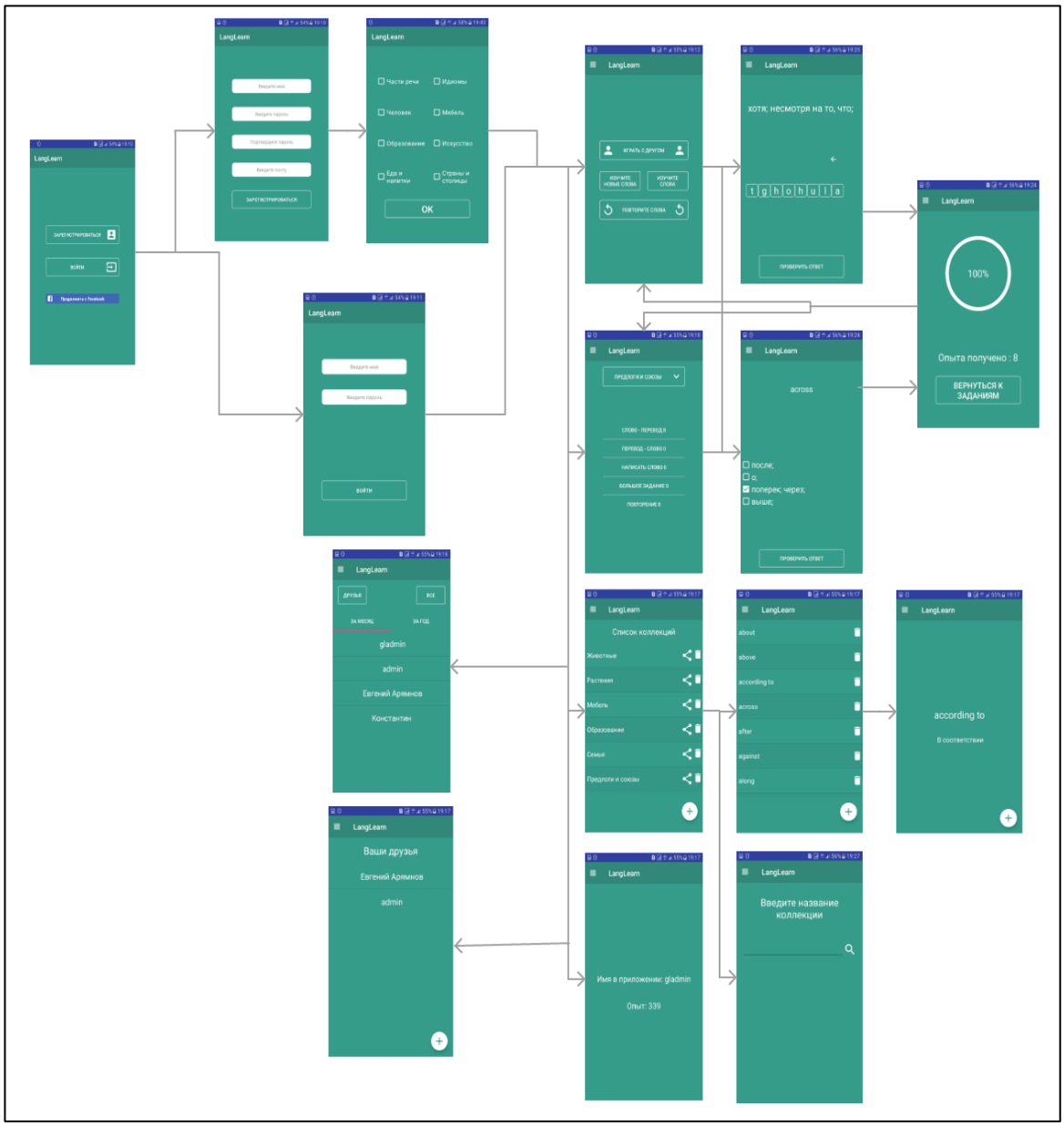


Рис. 20. Схема экранов и связей между ними

Вывод

В результате была выполнена реализация всех компонентов системы в соответствии со всеми требованиями к системе. В том числе были реализованы мобильное приложение для ОС Android и сервер на языке программирования Java.

5. ТЕСТИРОВАНИЕ

Тестирование программного обеспечения [32] – проверка соответствия между реальным и ожидаемым поведением программы, осуществляемая на конечном наборе тестов, выбранном определенным образом. Для тестирования приложения использовались следующие методы:

- модульное тестирование;
- автоматизированное UI-тестирование;
- функциональное тестирование.

5.1. Модульное тестирование

Модульное тестирование [7] – это вид тестирования, который позволяет проверить на корректность отдельные компоненты программы. Данный вид тестирования позволяет быстро проверить, не привело ли очередное изменение кода к появлению ошибок в уже оттестированных местах программы, а также облегчает обнаружение и устранение ошибок.

Для модульного тестирования использовались библиотеки JUnit, Robolectric [8], Mockito [5] и фрэймворк Spring. Модульный тест – это отдельный метод, помеченный аннотацией @Test. Для того, чтобы тесты запустились, нужно классу, содержащему тестовые методы, указать в аннотации RunWith специальный класс, который запустит эти тесты.

5.1.1. Модульное тестирование мобильного приложения

Для модульного тестирования мобильного приложения использовался специальный класс RobolectricTestRunner. Это позволяло создать Mock-объекты классов Activity, Context и так далее, необходимых для работы приложения.

При тестировании классов для работы с базой данных создавались коллекции, добавлялись в базу данных (база данных использовалась SQLite, которая создается для таких тестов фрэймворком Robolectric) и затем шла проверка корректности данных. На рис. 21 показан один из тестов классов для работы с базой данных. Для создания базы данных вызывается метод SetUp, рис 22.

При тестировании классов для взаимодействия с сервером создавал-

ся Mock-объект класса MessageManager (класс, реализующий Get, Post, Put и Delete запросы), который имитировал поведение сервера, возвращая данные (или не возвращая, в зависимости от теста).

```
@Test
public void collectionDBTest() throws Exception{
    Collection collection = new Collection(COLLECTION_NAME,5,7);
    for (int i = 0; i < 15; i++) {
        Random random = new Random();
        StringBuilder stringBuilder = new StringBuilder();
        for (int k = 0; k < 10; k++) {
            stringBuilder.append(new Charac-
ter((char)random.nextInt(255)));
        }
        String s = stringBuilder.toString();
        Word word = new Word(s,random.nextInt(10));
        ArrayList<String> meanings = new ArrayList<>();
        for (int j = 0; j < 5; j++) {
            meanings.add(s);
        }
        word.setMeanings(meanings);
        collection.addWord(word);
    }
    assertNotNull(mCollectionListCRUD);
    mCollectionListCRUD.deleteCollectionByName(collection.getTitle());
    mCollectionListCRUD.insertCollection(collection);
    Collection testCollection = mCollectionList-
CRUD.getEntireCollectionByName(COLLECTION_NAME);
    assertNotNull(testCollection);
    compareCollections(collection,testCollection);
    testCollection = mCollectionList-
CRUD.getCollectionById(testCollection.getId());
    compareCollections(collection,testCollection);
    mCollectionListCRUD.deleteCollectionByName(collection.getTitle());
    as-
sertNull(mCollectionListCRUD.getEntireCollectionByName(collection.getTitl
e()));
}
```

Рис. 21. Тест добавления коллекций в базу данных

```
@Before
public void setUp() throws IOException {
    mCollectionListCRUD = new CollectionList-
CRUD(RuntimeEnvironment.application);
    mWordsCRUD = new WordsCRUD(RuntimeEnvironment.application);
}
```

Рис. 22. Создание тестовой базы данных

При тестировании классов-реализаций интерфейса ITestCreator для каждого класса создавалась случайная коллекция слов, а затем имитировался процесс ответов на вопросы и проверки ответов, а также завершения

теста. В ходе теста шел подсчет количества верных ответов (ответы были случайными), а затем сравнивался с результатом, который предоставлял объект тестируемого класса.

При тестировании Presenter-классов также использовался специальный класс для проведения тестов RobolectricTestRunner. Для создания Mock-объектов модели и представления использовалась аннотация @Mock рядом с соответствующим полем тестового класса, а для инициализации команда MockitoAnnotations.initMocks(this). В данных тестах проверялось логика обработки данных, предоставленных Mock-объектами модели и действий, которые имитировали Mock-объекты представления. Модульное тестирование слоя View

При тестировании View-классов также использовался специальный класс для проведения тестов RobolectricTestRunner. Для создания Mock-объектов слоя Presenter использовалась аннотация @Mock рядом с соответствующим полем тестового класса, а для инициализации команда MockitoAnnotations.initMocks(this). В данных тестах проверялась правильность работы слоя View (видимость на тех или иных объектов, их кликабельность, а также методы классов-реализаций IView). На рис. 23 представлены методы тестирования класса PersonalDataView.

```
@Before
public void setUp() throws Exception {
    LayoutInflater layoutInflater = LayoutInflat-
er.from(RuntimeEnvironment.application);
    MockitoAnnotations.initMocks(this);
    mPersonalDataView = new PersonalDataView(layoutInflater,null);
    mPersonalDataView.setPersonalDataListener(mPersonalDataListener);
    mExp = (TextView) mPersonalData-
View.getRootView().findViewById(R.id.experienceText);
    mName = (TextView) mPersonalData-
View.getRootView().findViewById(R.id.nameText);
}
@Test
public void personalDataViewTest() {
    mPersonalDataView.setExp(11);
    mPersonalDataView.setName("name");
    assertTrue(mName.getVisibility() == View.VISIBLE);
    assertTrue(mExp.getVisibility() == View.VISIBLE);
    assertTrue(mName.getText().toString().equals(R.string.name+" :
name"));
    assertTrue(mExp.getText().toString().equals("Опыт: 11"));
}
```

Рис. 23. Тестирование класса PersonalDataView

В результате было написано 110 модульных теста, все из них выполнились успешно, и результаты совпали с ожидаемыми. Кроме того, после написания тестов система оказалась покрыта тестами на 76 %.

5.1.2. Модульное тестирование сервера

Для тестирования классов-обработчиков запросов (CollectionsController, RegistrationController, UserController и LeadersController) использовался специальный класс MockitoJUnitRunner, который позволял смоделировать работу той или иной части сервера командой standaloneSetup. После этого происходило обращение Mock-клиента по тому или иному адресу для последующей проверки результатов работы. Для тестирования остальных классов использовался специальный класс JUnit4. На рис. 24 представлен тест метода getCollectionById() класса CollectionsController.

```
@Test
public void getCollectionByIdTest() throws Exception{
    CollectionPojo collection = new CollectionPo-
jo(COLLECTION_NAME,5,7,33);
    for (int i = 0; i < 3; i++)
        collection.addWord(new WordPojo(i+""));

when(mCollectionService.getCollectionById(anyInt())).thenReturn(collectio
n);
    mMockMvc.perform(get("/collections/21"))
            .andExpect(status().isOk())
            .andExpect(content().contentType(APPLICATION_JSON))
            .andExpect(jsonPath("$.collection.title", is("name")))
            .andExpect(jsonPath("$.collection.sourceLanguageId", is(5)))
            .andExpect(jsonPath("$.collection.destinationLangugaeId",
is(7)))
            .andExpect(jsonPath("$.collection.collection[0].word",
is("0")))
            .andExpect(jsonPath("$.collection.collection[1].word",
is("1")))
            .andExpect(jsonPath("$.collection.collection[2].word",
is("2")));
    verify(mCollectionService,times(1)).getCollectionById(anyInt());
    verifyNoMoreInteractions(mCollectionService);
}
```

Рис. 24. Тестирование метода getCollectionById() класса

В результате было написано 43 модульных теста, все из них выполнились успешно, и результаты совпали с ожидаемыми. Кроме того, после написания тесто, система оказалась покрыта тестами на 73 %.

5.2. Автоматизированное UI тестирование

Автоматизированное UI тестирование [2] – вид тестирования, который проводится с целью проверки работы системы на соответствие техническому заданию с помощью эмулирования работы реальных пользователей. Для реализации данного вида тестирования использовались библиотеки Espresso [12] и JUnit, которые позволяют проводить тесты как на реальных Android-устройствах, так и на эмуляторах. Тестами имитировались возможные действия пользователей, а затем проверялись результаты. На рис. 25 представлен тест выполнения задания. Для запуска Activity используется ActivityTestRule<MainActivity>, а для работы с компонентами (кнопками и т.п.) используется методы библиотеки Espresso, такие как onView, onData и onPressBack.

```
@Test
public void doWordsTask() {
    onView(withText(R.string.learnWords))perform(click());
    Random random = new Random();
    for (int i = 0; i < 15; i++) {
        onView(withText(R.string.checkAnswer))
                .check(matches(isDisplayed()));
        switch (random.nextInt(4)) {
            case 0:

onView(withId(R.id.variant1)).perform(scrollTo(),click());
                break;
            case 1:

onView(withId(R.id.variant2)).perform(scrollTo(),click());
                break;
            case 2:

onView(withId(R.id.variant3)).perform(scrollTo(),click());
                break;
            case 3:

onView(withId(R.id.variant4)).perform(scrollTo(),click());
                break;
        }
        onView(withText(R.string.checkAnswer))
                .perform(click());

onView(withText(R.string.nextQuestion)).check(matches(isDisplayed()));
        onView(withText(R.string.nextQuestion)).perform(click());
    }
    onView(withText(R.string.backToTasks).check(matches(isDisplayed()));
    pressBack();
    onView(withText("Изучите новые слова")).check(matches(isDisplayed()));
}
```

Рис. 25. Тестирование выполнения задания

Всего в ходе тестирования было создано 17 тестов, все из них выполнились успешно, и результаты совпали с ожидаемыми. В результате была проверена работа приложения на соответствие требованиям с помощью эмулирования действий пользователей.

5.3. Функциональное тестирование

Функциональное тестирование – это тестирование программного обеспечения в целях проверки реализуемости функциональных требований, то есть способности программного обеспечения в определенных условиях решать задачи, нужные пользователям. Функциональные требования определяют, что именно делает программное обеспечение, какие задачи оно решает. Набор тестов на функциональность представлен в приложении 3.

Вывод

В результате было выполнено модульное тестирование основных компонентов системы, было выполнено автоматизированное UI-тестирование, также функциональное тестирование. Все тесты были выполнены успешно.

ЗАКЛЮЧЕНИЕ

Целью данной работы являлась разработка Android-приложения для изучения иностранных слов с элементами игры.

В ходе работы были выполнены следующие задачи:

- изучены особенности разработки приложений для ос Android;
- определены требования к программе;
- спроектирована архитектура приложения;
- разработана серверную часть;
- разработано мобильное приложение;
- выполнено тестирование.

ЛИТЕРАТУРА

1. Android Development Tools for Eclipse. [Электронный ресурс] URL: https://marketplace.eclipse.org/content/android-development-tools-eclipse (дата обращения: 12.03.2017).

2. Automating User Interface Tests. [Электронный ресурс] URL: https://developer.android.com/training/testing/ui-testing/index.html (дата обращения: 16.05.2017).

3. Hibernate ORM 5.2.9.Final User Guide. [Электронный ресурс] https://docs.jboss.org/hibernate/orm/current/userguide/html_single/Hibernate_User_Guide.html (дата обращения: 12.04.2017).

4. Java EE 7: Building Web Applications with WebSocket, JavaScript and HTML5. [Электронный ресурс] URL: http://www.oracle.com/webfolder/technetwork/tutorials/obe/java/HomeWebsocket/WebsocketHome.html (дата обращения: 24.04.2017).

5. Mockito. [Электронный ресурс] URL: http://site.mockito.org/ (дата обращения: 16.05.2017).

6. MVP and MVC Architectures in Android. [Электронный ресурс] URL: http://www.techyourchance.com/mvp-mvc-android-1/ (дата обращения: 12.03.2017).

7. Olan M. Unit testing: test early, test often. // Journal of Computing Sciences in Colleges, 2003. – Vol. 19. – No 2. – P. 319–328.

8. Robolectric. [Электронный ресурс] URL: http://robolectric.org/ (дата обращения: 16.05.2017).

9. Spring Security Reference. [Электронный ресурс] URL: https://docs.spring.io/spring-security/site/docs/current/reference/htmlsingle (дата обращения: 23.04.2017).

10. SQLite – a software library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. [Электронный ресурс] URL: https://sqlite.org (дата обращения: 25.04.2017).

11. SQLite vs MySQL vs PostgreSQL: сравнение систем управления базами данных. [Электронный ресурс] URL http://devacademy.ru/posts/sqlite-vs-mysql-vs-postgresql/ (дата обращения: 12.04.2017).

12. Testing UI for a Single App. [Электронный ресурс] URL: https://developer.android.com/training/testing/ui-testing/espresso-testing.html (дата обращения: 16.05.2017).

13. Web MVC framework. [Электронный ресурс] URL: https://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html (дата обращения: 23.04.2017).

14. Webber J., Parastatidis S., Robonson I. REST in Practice. – USA: O'Reilly Media, 2010. – 448 p.

15. What is API Level? [Электронный ресурс] URL:http://developer.android.com/guide/topics/manifest/uses-sdk-element.html-#ApiLevels (дата обращения:25.04.2017).

16. What is Object/Relational Mapping? [Электронный ресурс] URL: http://hibernate.org/orm/what-is-an-orm/ (дата обращения: 12.04.2017).

17. Актуальность изучения иностранных языков в социально-экономических условиях современной России. [Электронный ресурс] URL: http://www.ibl.ru/konf/140509/61.html (дата обращения: 12.03.2017)

18. Варенина Л. П. Геймификация в образовании. // Историческая и образовательная мысль. 2014. – № 6-2. – С. 314-317.

19. Введение в веб-модель MVC Spring. [Электронный ресурс] URL: https://netbeans.org/kb/docs/web/quickstart-webapps-spring_ru.html (дата обращения: 23.04.2017).

20. Зенина Л.В. Возможности современных компьютерных программ в процессе обучения иностранным языкам в вузе. / Л.В Зенина., Н.А. Каменева. // Вестник Московского государственного гуманитарного университета им. М.А. Шолохова. – Педагогика и психология, 2013. – № 3 – С. 57-60.

21. Краснова Т.И. Геймификация обучения иностранному языку. // Молодой ученый, 2015. – № 11. – С. 1373-1375.

22. Обзор мобильных приложений для изучения английского языка. [Электронный ресурс] URL:https://habrahabr.ru/post/239985/ (дата обращения: 12.03.2017).

23. Общие сведения о платформе Android. [Электронный ресурс] URL: https://developer.android.com/guide/index.html (дата обращения: 12.03.2017).

24. Основные понятия баз данных. [Электронный ресурс] URL: http://inf.susu.ac.ru/Klinachev/lc_sga_26.htm (дата обращения: 12.04.2017)

25. Сайт Android Studio. [Электронный ресурс] URL: https://developer.android.com/studio/features.html (дата обращения: 12.03.2017).

26. Сайт Anki. [Электронный ресурс] URL: http://ankisrs.net/ (дата обращения: 12.03.2017).

27. Сайт Eclipse. [Электронный ресурс] URL: https://eclipse.org/ (дата обращения: 12.03.2017).

28. Сайт LinguaLeo. [Электронный ресурс] URL: http://lingualeo.com/ru (дата обращения: 12.03.2017).

29. Сайт WebSocket. [Электронный ресурс] URL: https://www.websocket.org/aboutwebsocket.html (дата обращения: 12.03.2017).

30. Сайт Xamarin. [Электронный ресурс] URL: https://www.xamarin.com/ (дата обращения: 12.03.2017).

31. Сервлеты. Введение. [Электронный ресурс] URL: http://www.java2ee.ru/servlets/ (дата обращения: 23.04.2017).

32. Тестирование программного обеспечения - основные понятия и определения. [Электронный ресурс] URL: http://www.protesting.ru/testing/ (дата обращения: 16.05.2017).

ПРИЛОЖЕНИЯ

Приложение 1

В таблицах 1-11 представлено описание таблиц баз данных клиентской и серверной части.

Табл. 1. Таблица Collections клиентской части

№	Атрибут	Семантика	Тип
1	id	Уникальный идентификатор	INTEGER
2	destination_language_id	Идентификатор языка перевода	INTEGER
3	source_language_id	Идентификатор языка оригинала.	INTEGER
4	title	Название коллекции	VARCHAR(50)
5	date	Дата последнего использования	DATETIME

Табл. 2. Таблица Words клиентской части

№	Атрибут	Семантика	Тип
1	id	Уникальный идентификатор	INTEGER
2	word	Само слово или словосочетание	VARCHAR(255)
3	collection_id	Внешний ключ. Идентификатор коллекции, к которой принадлежит слово	INTEGER

Табл. 3. Таблица Meanings клиентской части

№	Атрибут	Семантика	Тип
1	id	Уникальный идентификатор	INTEGER
2	meaning	Само значение	VARCHAR(255)
3	word_id	Внешний ключ. Идентификатор слова	INTEGER

Табл. 4. Таблица Collections базы данных сервера

№	Атрибут	Семантика	Тип
1	id	Уникальный идентификатор	INTEGER
2	destination_language_id	Идентификатор языка перевода	INTEGER

3	source_language_id	Идентификатор языка оригинала	INTEGER
4	title	Название коллекции	TEXT
5	date	Дата последнего использования	DATETIME
6	owner_id	Внешний ключ. Идентификатор владельца коллекции	INTEGER
7	theme_id	Внешний ключ. Идентификатор темы коллекции	INTEGER

Табл. 5. Таблица Words базы данных сервера

№	Атрибут	Семантика	Тип
1	id	Уникальный идентификатор	INTEGER
2	word	Само слово или словосочетание	VARCHAR(255)
3	collection_id	Внешний ключ. Идентификатор коллекции, к которой принадлежит слово	INTEGER
4	level	Степень изученности слова пользователем	INTEGER

Табл. 6. Таблица Meanings базы данных сервера

№	Атрибут	Семантика	Тип
1	id	Уникальный идентификатор	INTEGER
2	meaning	Само значение	VARCHAR(255)
3	Word_id	Внешний ключ. Идентификатор слова	INTEGER

Табл. 7. Таблица People базы данных сервера

№	Атрибут	Семантика	Тип
1	id	Уникальный идентификатор	INTEGER
2	name	Имя пользователя	VARCHAR(50)
3	password	Пароль в зашифрованном виде	VARCHAR(50)
4	email	Электронный адрес пользователя	VARCHAR(50)
5	Facebook_id	Id в Facebook	VARCHAR(50)

Табл. 8. Таблица Themes базы данных сервера

№	Атрибут	Семантика.	Тип
1	id	Уникальный идентификатор	INTEGER
2	theme_name	Название	VARCHAR(50)

Табл. 9. Таблица FirebaseIds базы данных сервера

№	Атрибут	Семантика	Тип
1	firebase_id	Уникальный идентификатор. Используется для отправки Push-уведомлений	INTEGER
2	owner_id	Внешний ключ. Id пользователя	INTEGER

Табл. 10. Таблица Friends базы данных сервера

№	Атрибут	Семантика	Тип
1	id	Уникальный идентификатор	INTEGER
2	name_id	Внешний ключ. Идентификатор пользователя	INTEGER
3	person_id	Внешний ключ. Идентификатор друга пользователя	INTEGER

Табл. 11. Таблица Languages базы данных сервера

№	Атрибут	Семантика	Тип
1	id	Уникальный идентификатор	INTEGER
2	language	Язык	VARCHAR(25)

Приложение 2

На рисунках 1-8 представлены скриншоты экранов приложения.

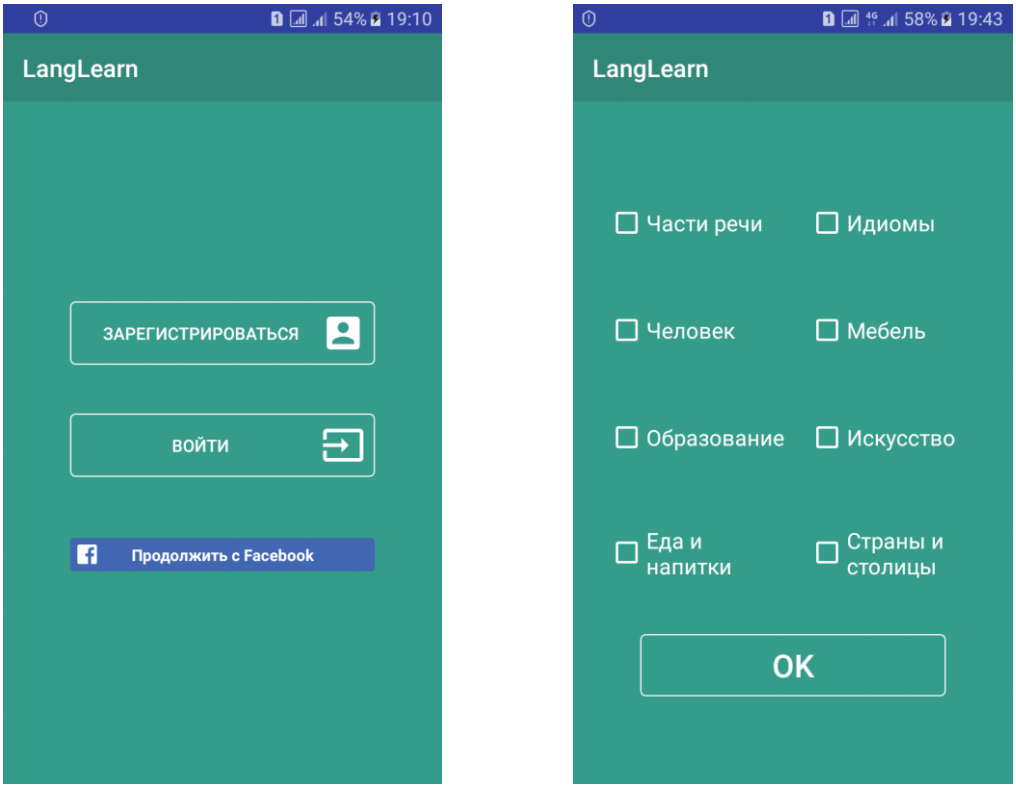


Рис. 1. Стартовый экран (слева) и экран выбора тем слов (справа)

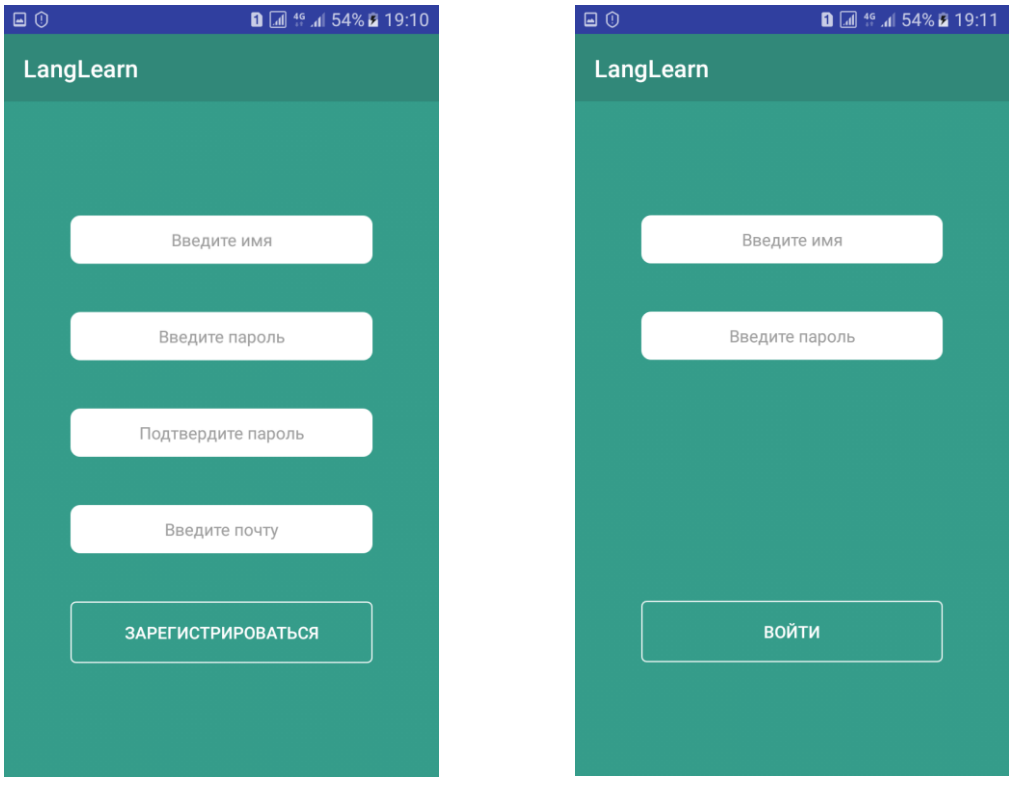


Рис. 2. Экран регистрации (слева) и экран аутентификации (справа)

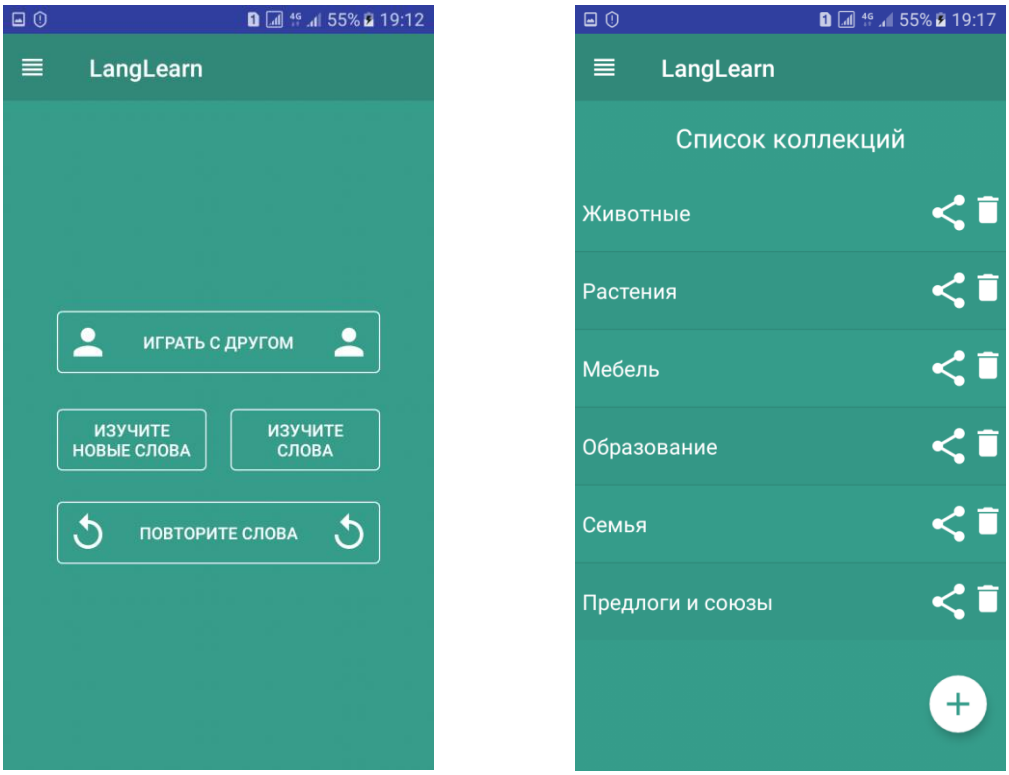


Рис. 3. Главное меню (слева) и экран со списком коллекций пользователя (справа)

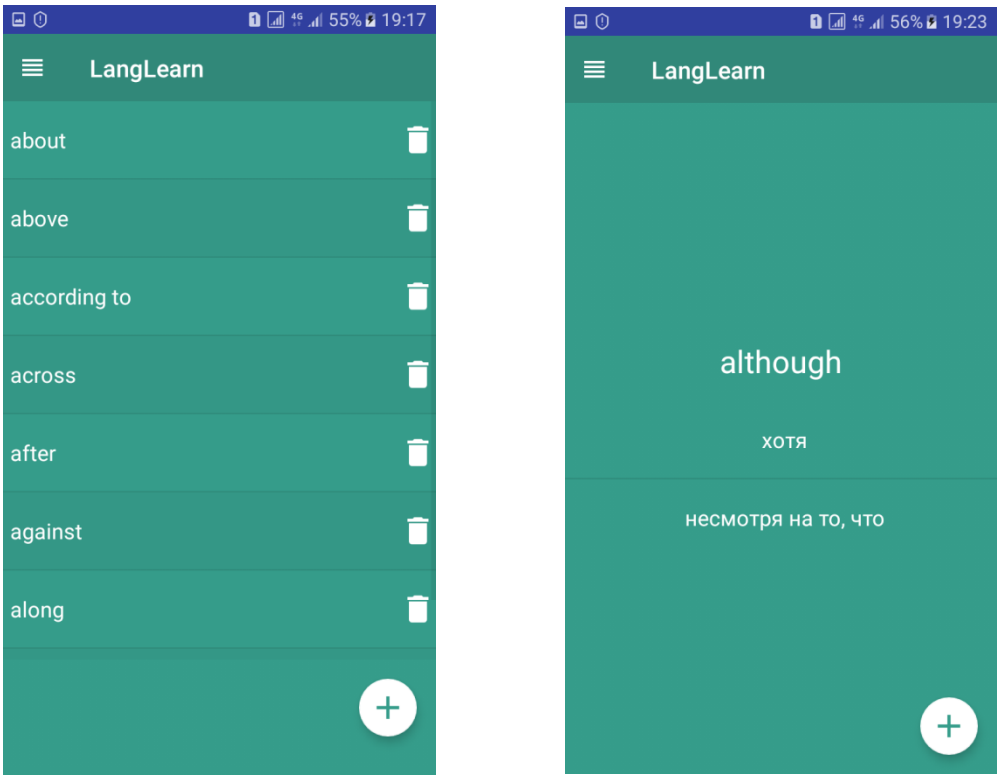


Рис. 4. Экран списка слов (слева) и экран слова (справа)

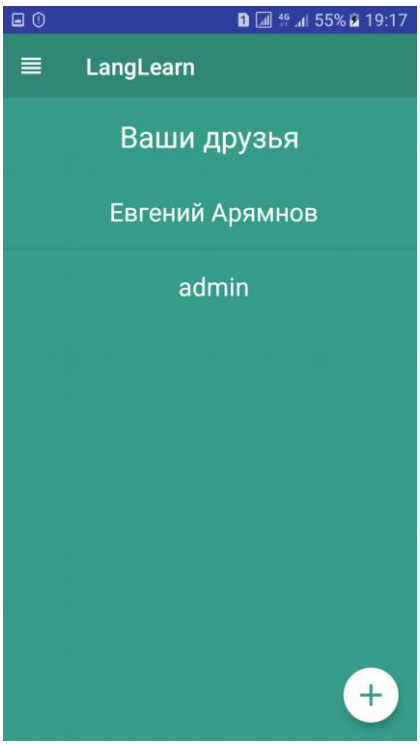


Рис. 5. Экран личных данных (слева) и экран списка друзей (справа)

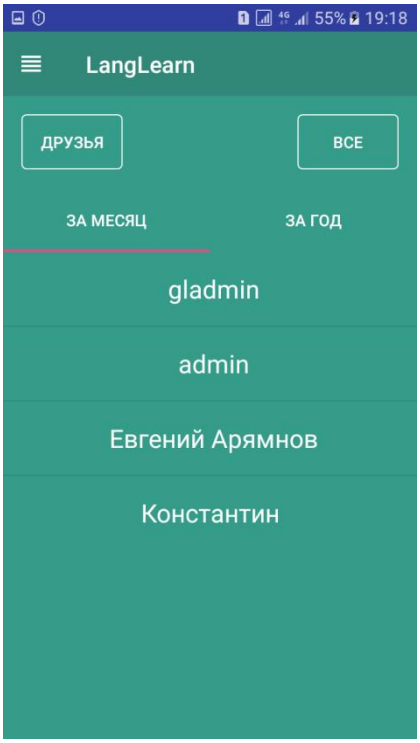
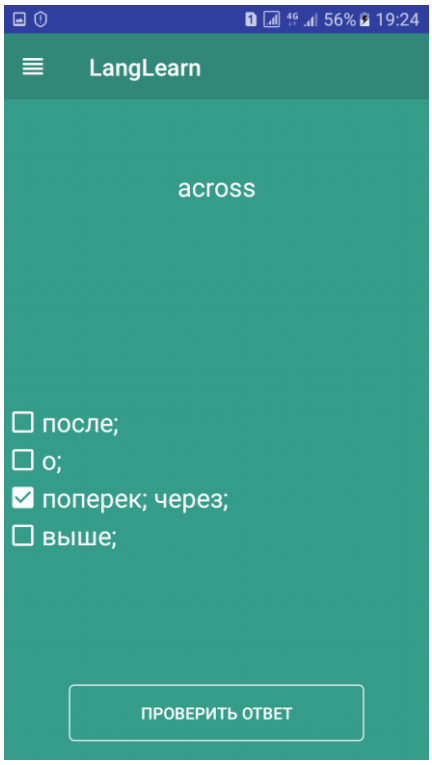


Рис. 6. Экран списка лидеров (слева) и экран списка заданий (справа)



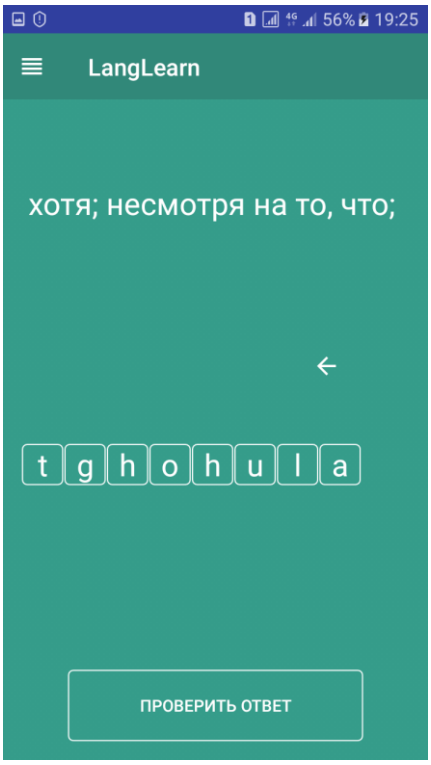
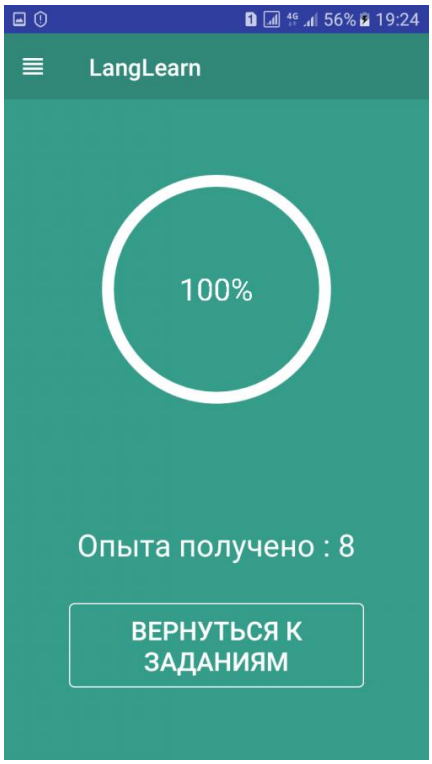


Рис. 7. Экран тестового задания (слева) и экран задания на составление слова (справа)



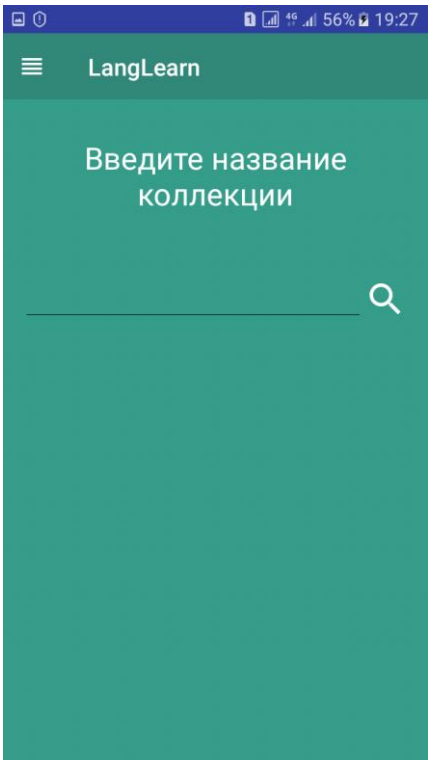


Рис. 8. Экран результата выполнения задания (слева) и экран поиска коллекций (справа)

Приложение 3

В таблице 1 представлен набор функциональных тестов приложения.

Табл. 1. Тестирование системы

№	Название	Шаги	Ожидаемый результат	Тест пройден?
1	Регистрация	1. На странице входа нажать кнопку «Регистрация». 2. Ввести имя в приложении, пароль, повторить пароль и ввести электронный адрес. 3. Нажать кнопку «Зарегистрироваться».	В базе данных сервера появился новый пользователь, на экране отобразилась страница выбора тем.	Да
2	Аутентификация	1. На странице входа нажать кнопку «Войти». 2. Ввести электронный адрес и пароль. 3. Нажать кнопку «Войти».	Пользователь аутентифицировался, на экране отобразилось главное меню, если электронный адрес и пароль верные.	Да
3	Аутентифиация с помощью Facebook	На странице входа нажать кнопку «Продолжить с Facebook»	Пользователь аутентифицировался, на экране отобразилось главное меню, либо зарегистрировался, и на экране отобразилась страница выбора тем.	Да
4	Выбор тем	На странице выбора тем выбрать интересующие темы путем нажатия на них.	Произошла загрузка коллекций в зависимости от тем, выбранных пользователем, на экране отобразилось главное меню.	Да

5	Добавление пустой коллекции	1. В боковом меню нажать на кнопку «Коллекции». 2. Нажать на кнопку с изображением плюсика. 3. Нажать на кнопку для добавления пустой коллекции. 4. В появившемся диалоговом окне ввести название коллекции. 5. Нажать на кнопку «ОК».	В базу данных добавлена новая коллекция	Да
6	Добавление слова	1. На странице списка коллекций выбрать коллекцию. 2. Нажать на кнопку с изображением плюсика. 3. В появившемся диалоговом окне ввести слово. 4. Нажать кнопку «ОК».	В базе данных появилось новое слово.	Да
7	Удаление слова	На странице списка слов нажать на кнопку удаления напротив нужного слова.	Слово удалено из базы данных.	Да
8	Удаление коллекции	На странице списка коллекций нажать на кнопку удаления напротив нужной коллекции.	Коллекция удалена из базы данных.	Да
9	Поделиться коллекции	На странице списка коллекций нажать на кнопку поделиться напротив нужной коллекции.	В базе данных сервера появилась новая общедоступная коллекция слов (если у пользователя есть интернет-соединение).	Да

10	Загрузка коллекции с сервера	1. На странице со списком коллекций нажать на кнопку с изображением плюсика. 2. Нажать на кнопку для загрузки коллекций. 3. В поле для ввода названия ввести название (или не вводить для получения полного списка). 4. Выбрать нужную коллекцию.	В базе данных появилась новая коллекция.	Да
11	Добавление друга	1. На странице со списком друзей нажать на кнопку с изображением плюсика. 2. Ввести имя друга в появившемся диалоговом окне. 3. Нажать кнопку «ОК».	Если человек, которому предложили стать друзьями, соглашается, то у пользователя появился новый друг в приложении.	Да
12	Просмотр списка лидеров	На странице списка лидеров выбрать нужные параметры.	На экране отобразился список лидеров.	Да
13	Выполнить задание	1. В главном меню нажать на одну из кнопок: «Изучить новые слова», «Учить слова» либо «Повторить слова». 2. В появившемся экране отвечать на вопросы, пока не откроется экран результатов.	Пользователь выполнил задание, результаты сохранены.	Да

14	Участвовать в режиме соревнования	1. В главном меню нажать на кнопку «Играть с другом». 2. На появившейся странице выбрать из списка друга, нажав на его имя. 3. На появившейся странице выбрать задание. 4. Дождаться согласия друга. 5. В появившемся экране отвечать на вопросы, пока не откроется экран результатов.	Пользователь принял участие в режиме соревнования, результаты сохранены.	Да