

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Программное обеспечение средств вычислительной
техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

на тему

ПРОГРАММА ОЦЕНКИ УРОВНЯ ЗНАНИЙ
ЯЗЫКА JAVA ДЛЯ ОТБОРА КАНДИДАТОВ

Студент

(подпись, дата)

Отставнов Александр Михайлович

04.06.23

Руководитель

(подпись, дата)

Митрохин Максим Александрович

01.06.23

Нормоконтролёр

(подпись, дата)

Бычков Андрей Станиславович

04.06.23

Работа допущена к защите (протокол зас. кафедры от 02.06.2023г. № 14)

Заведующий кафедрой

(подпись)

М.А. Митрохин

(фамилия, инициалы)

Работа защищена с отметкой

(протокол заседания ГЭК от

№

5)

Секретарь ГЭК

(подпись)

Е.И. Гурин

(фамилия, инициалы)

Пенза 2023

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
« 26 » апреля 2023г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Отставнов Александр Михайлович гр. 19ВВП1 факультета вычислительной техники

2. Тема работы «Программа оценки уровня знаний языка Java для отбора кандидатов»

Тема утверждена приказом ПГУ №458/о от "26" апреля 2023 г

3. Руководитель бакалаврской работы д.т.н., профессор, зав. каф. ВТ ПГУ Митрохин М.А.

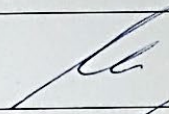
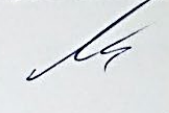
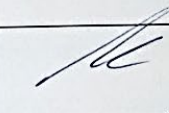
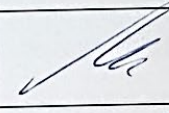
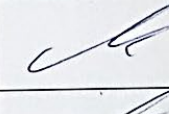
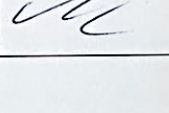
4. Задание на работу (назначение разработки, исходные данные и т.п.)

Разработать приложение «Программа оценки уровня знаний языка Java для отбора кандидатов», позволяющее оптимизировать процесс отбора кандидатов. Приложение должно обладать графическим и интуитивно понятным интерфейсом, для взаимодействия с пользователем. В данном приложении должна быть реализована часть, которая отвечает за регистрацию, вход, тестирование, заполнение и занесение информации о кандидатах в базу данных. На основе полученных результатов тестирования программа автоматически должна подбирать должность, которую может предложить организация для прохождения дальнейшего собеседования.

5. Перечень подлежащих разработке вопросов

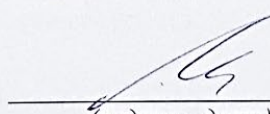
Требуется произвести анализ предметных областей: изучить существующие аналоги, выделить их достоинства и недостатки. Выбрать операционную систему, под управлением которой будет работать приложение. Определить архитектуру приложения и вспомогательные средства разработки. Изучить функционал библиотек для взаимодействия с базой данных реального времени.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области	10 %	26.04.2023 - 29.04.2023	
2.	Выбор архитектуры приложения и технологий, применяемых при проектировании и разработке	15 %	30.04.2023 - 04.05.2023	
3.	Разработка графического интерфейса	20 %	05.05.2023 - 09.05.2023	
4.	Разработка функционала приложения	25 %	10.05.2023 - 21.05.2023	
5.	Тестирование и отладка	15 %	22.05.2023 - 24.05.2023	
6.	Выполнение задания по БЖД	5 %	25.05.2023 - 27.05.2023	
7.	Оформление выпускной работы	10 %	28.05.2023 - 31.05.2023	

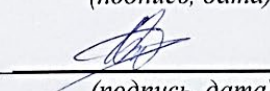
Дата выдачи задания "26" апреля 2023 г.

Руководитель бакалаврской работы


(подпись, дата)

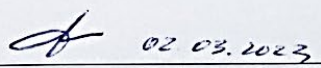
Митрохин М.А.

Задание к исполнению принял студент


(подпись, дата)

Отставнов А.М.

Работу к защите допустить
Декан факультета


(подпись, дата)

Фионова Л.Р.

Реферат

Бакалаврская работа содержит 96 листов, 34 рисунка, 1 таблицу, 5 источников, 3 приложения.

ANDROID-ПРИЛОЖЕНИЕ, ТЕСТИРОВАНИЕ КАНДИДАТОВ, УРОВЕНЬ ЗНАНИЙ, JAVA, MVVM, FIREBASE.

Цель работы – разработать *Android*-приложение, позволяющее определить уровень знания языка *Java* у кандидатов посредством тестирования.

Объектом разработки является приложение, обладающее следующим функционалом – регистрация, авторизация, редактирование личных данных, прохождение тестирования, просмотр результатов тестирования. Приложение должно обладать удобным графическим пользовательским интерфейсом.

Разработка проводилась в среде программирования *Android Studio* 2019. В качестве языка программирования был выбран *Java*

В результате проведенной работы разработано приложение, удовлетворяющее поставленной задаче. Созданное приложение может быть внедрено в *IT*-компанию для оптимизации процесса по подбору персонала.

					ПГУ1.090301.25.001 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата				
Разраб.		Отставнов А.М.			Программа оценки уровня знаний языка Java для отбора кандидатов Пояснительная записка	Лит.	Лист	Листов
Провер.		Митрохин М.А.					2	96
Н. Контр		Бычков А.С.				ФВТ, каф ВТ		
Утверд.		Митрохин М.А.				гр. 19ВВП1		

Содержание

Перечень сокращений и обозначений.....	5
Введение.....	6
1 Программа оценки уровня знаний языка <i>Java</i> для отбора кандидатов.....	7
1.1 Анализ предметной области	7
1.1.1 Описание предметной области	7
1.1.2 Актуальность предметной области	7
1.1.3 Анализ аналогов	8
1.1.3.1 Приложение « <i>Java Quiz</i> ».....	8
1.1.3.2 Приложение « <i>Java Test Prep 2023 Ed</i> »	10
1.1.3.3 Онлайн-сервис <i>itProger.com</i>	13
1.1.3.4 Вывод.....	15
1.1.4 Выбор операционной системы	15
1.1.5 Постановка задачи.....	17
1.1.6 Выбор среды разработки	18
1.1.6.1 Среда <i>Eclipse</i>	19
1.1.6.2 Среда <i>Android Studio</i>	19
1.1.6.3 База данных <i>Firebase Realtime Database</i>	20
1.2 Реализация приложения	22
1.2.1 Архитектура приложения.....	22
1.2.2 Создание проекта в <i>Android Studio</i>	24
1.2.3 Реализация аутентификации пользователей	27
1.2.4 Реализация работы с <i>Firebase Realtime Database</i>	31

1.2.5 Реализация тестирования	32
1.2.6 Интерфейс приложения	39
2 Безопасность жизнедеятельности.....	43
2.1 Рекомендации по безопасному использованию ПК	43
2.2 Рекомендации по безопасному использованию смартфона	44
Заключение	47
Список используемых источников.....	48
Приложение А – <i>UML</i> -диаграммы	49
Приложение Б – Листинг программы	54
Приложение В – Презентация	79

Перечень сокращений и обозначений

В настоящей работе применяют следующие сокращения и обозначения

БД – база данных

ОС – операционная система

ПК – персональный компьютер

Фреймворк – программная платформа, упрощающая разработку программного продукта, определяет структуру проекта

IT (Information Technology) – информационные технологии

MVVM (Model View ViewModel) – шаблон проектирования архитектуры приложения

UML (Unified Modeling Language) – унифицированный язык моделирования

					ПГУ1.090301.16.001 ПЗ	Лист
						5
Изм.	Лист	№ докум.	Подпись	Дата		

Введение

За последние 10 лет смартфон стал неотъемлемой частью нашей жизни. Мобильные приложения прямо сейчас являются одной из самых перспективных направлений разработки по нескольким причинам:

- большое количество пользователей. С каждым годом все больше людей используют мобильные устройства, и это открывает новые возможности для разработчиков мобильных приложений;

- удобство использования. Мобильные приложения легко скачиваются и устанавливаются на устройство, а также имеют более удобный интерфейс, чем веб-сайты;

- разнообразие функций. Мобильные приложения могут предлагать множество функций, таких как интеграция с социальными сетями, оповещения, местоположение и др.;

- возможность монетизации. Мобильные приложения могут приносить доход разработчикам через рекламу, покупки в приложении или подписки на услуги;

- постоянное развитие технологий. Развитие технологий позволяет создавать более функциональные и удобные мобильные приложения, что приводит к росту популярности данного направления разработки.

Актуальность данной темы заключается в том, что в настоящее время *Java* является одним из самых популярных языков программирования в мире, и многие компании и организации ищут специалистов, которые могут работать с этим языком. Программа оценки уровня знаний языка *Java* для отбора кандидатов позволяет работодателям быстро и эффективно выбирать наиболее подходящих кандидатов, что экономит время и ресурсы компании. Кроме того, такая программа позволяет кандидатам оценить свой уровень знаний языка *Java* и подготовиться к собеседованию, что повышает их шансы на успешное прохождение отбора.

1 Программа оценки уровня знаний языка *Java* для отбора кандидатов

1.1 Анализ предметной области

1.1.1 Описание предметной области

Создаваемое приложение предназначено для кандидатов при поступлении на работу. Данное приложение позволяет получить информацию об уровне знаний языка *Java* кандидатов посредством тестирования. На основе полученных результатов тестирования программа автоматически подберет должность, которую может предложить организация для прохождения дальнейшего собеседования. Для сохранения информации в БД и аутентификации, приложение использует интернет.

1.1.2 Актуальность предметной области

Перед тем как принять соискателя на работу, организация должна получить и проанализировать информацию о кандидате. Как правило, для этого существует собеседование, которое проводится в несколько этапов, в которых часть кандидатов отсеивается, когда становится известным их полное или частичное несоответствие заявленным требованиям. Одним из таких этапов является тестирование. Тесты при собеседовании измеряют потенциал человека для выполнения той или иной задачи, которая требует специальных навыков.

Проведение теста для определения уровня знаний перед основным собеседованием при устройстве на работу имеет такие преимущества как:

- экономия времени: проведение теста позволяет определить уровень знаний кандидата заранее, что позволяет сэкономить время на основном собеседовании и сосредоточиться на более глубоких вопросах. Если же при прохождении теста кандидат не набирает нужное количество баллов, то нет смысла продолжать собеседование, так как знания тестируемого не отвечают требованиям вакансии;

					ПГУ1.090301.16.001 ПЗ	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		

- объективность: тестирование позволяет работодателю получить более объективную оценку уровня знаний кандидата, чем при собеседовании. В отличие от собеседования, где кандидат может ответить на вопросы не полностью или некорректно из-за стресса или недостатка времени, тестирование дает возможность показать свои знания в более спокойной обстановке.

В целом, проведение теста для определения уровня знаний языка Java до основного собеседования при устройстве на работу является эффективным инструментом для работодателей, который позволяет определять квалификацию кандидата, экономить время и ресурсы, получать более объективную оценку и улучшать качество найма.

1.1.3 Анализ аналогов

Перед началом создания собственного проекта, было принято решение изучить аналогичные приложения и сайты, чтобы посмотреть, как с данной задачей справились другие разработчики. Таким образом были выделены слабые и сильные стороны существующих приложений и сайтов для того, чтобы улучшить будущий продукт.

В качестве аналогов будущего проекта были выбраны приложения «Java Quiz», «Java Test Prep 2023 Ed» и онлайн-сервис «itProger.com».

1.1.3.1 Приложение «Java Quiz»

Приложение «Java Quiz» встречает пользователя списком разделов языка Java для прохождения тестирования. При выборе любой из категорий сразу же начинается тестирование. По окончании тестирования пользователю показывают результат тестирования в процентах, список правильных и не правильных ответов. После чего пользователь может пройти тест повторно или же вернуться на главную страницу. Интерфейс данного приложения представлен ниже на рисунках 1-3.

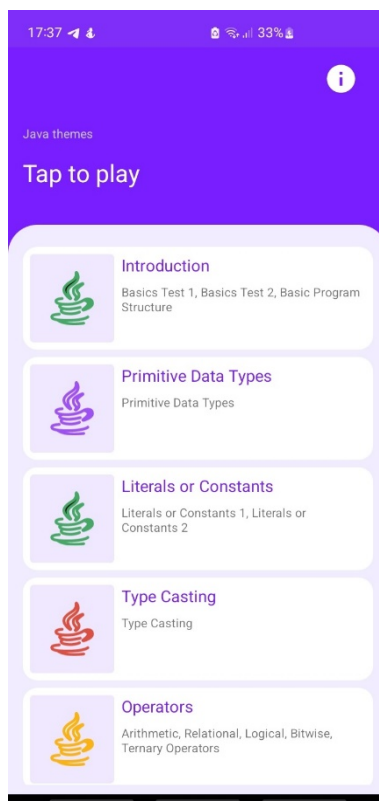


Рисунок 1 – Стартовая страница приложения «Java Quiz»

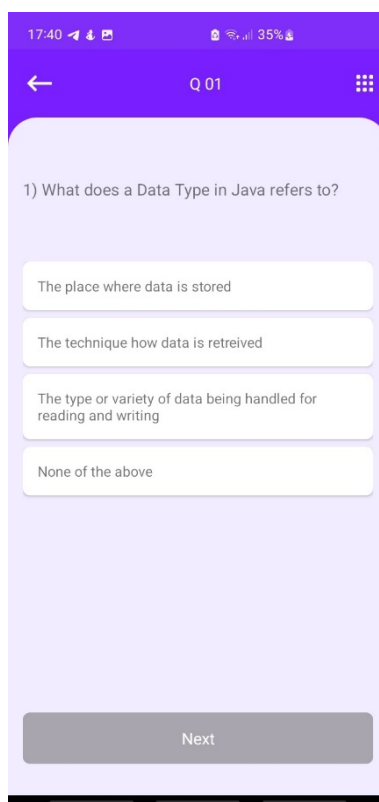


Рисунок 2 – Экран тестирования в приложении «Java Quiz»

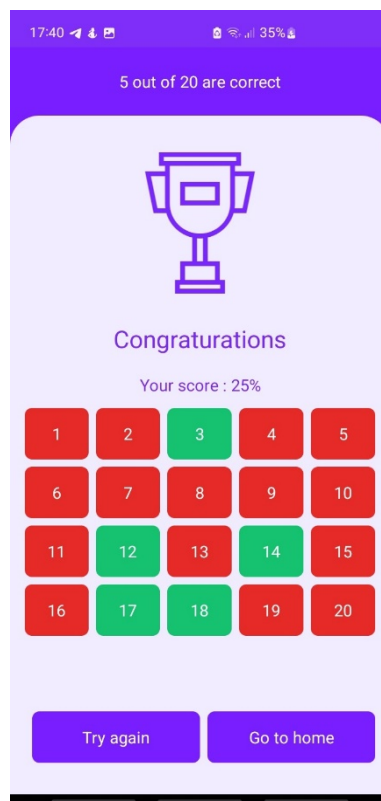


Рисунок 3 – Экран результатов тестирования в приложении «*Java Quiz*»

К достоинствам «*Java Quiz*» можно отнести:

- большая база вопросов, разделенная на категории;
- приятный и понятный интерфейс.

Недостатками данного приложения являются:

- отсутствие авторизации;
- отсутствие русской локализации.

1.1.3.2 Приложение «*Java Test Prep 2023 Ed*»

Другое приложение - «*Java Test Prep 2023 Ed*» встречает пользователя понятным интерфейсом, который позволяет в пару кликов начать тестирование. Перед началом тестирования пользователь может выбрать количество вопросов в тесте. При изменении количества вопросов изменяется и время тестирования. Интерфейс данного приложения представлен ниже на рисунках 4-7.

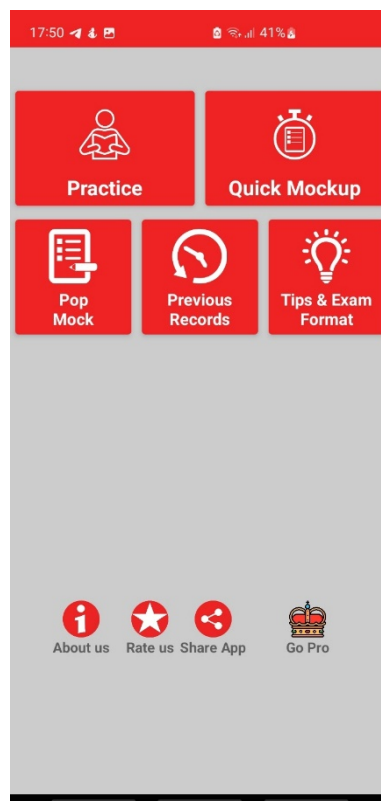


Рисунок 4 – Стартовая страница приложения «Java Test Prep 2023 Ed»

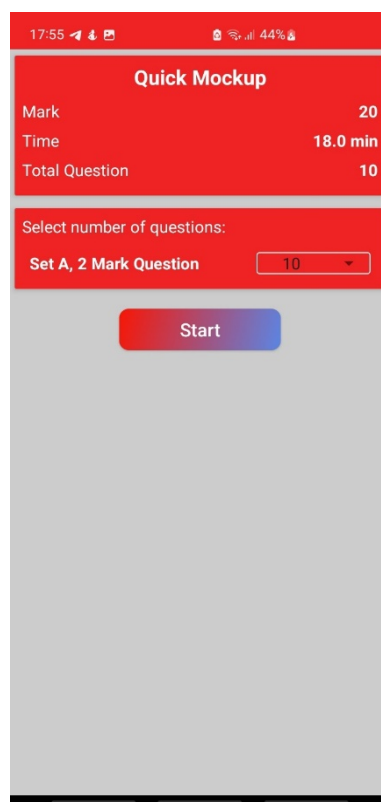


Рисунок 5 – Экран конструктора теста в «Java Test Prep 2023 Ed»

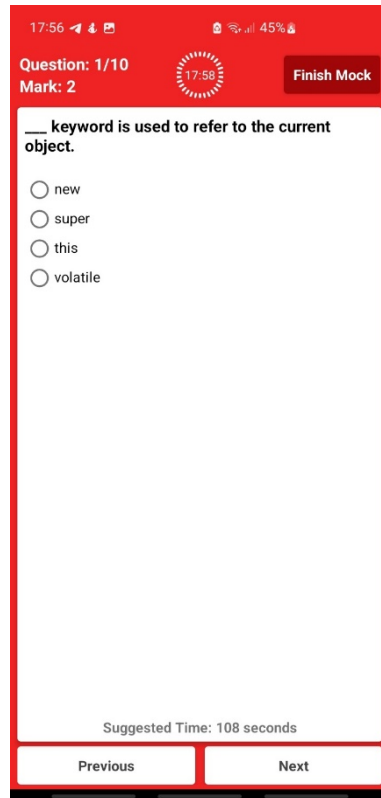


Рисунок 6 – Экран тестирования в «Java Test Prep 2023 Ed»



Рисунок 7 – Экран результатов тестирования в «Java Test Prep 2023 Ed»

					ПГУ1.090301.16.001 ПЗ	Лист
						12
Изм.	Лист	№ докум.	Подпись	Дата		

К достоинствам «*Java Test Prep 2023 Ed*» можно отнести:

- наличие конструктора теста.

Недостатками данного приложения являются:

- отсутствие авторизации;
- отсутствие русской локализации;
- отсутствие разделения вопросов по категориям.

1.1.3.3 Онлайн-сервис *itProger.com*

Следующий аналог – онлайн-сервис «*itProger.com*». Данный сервис направлен на обучение веб программированию, созданию игр и разработке ПО. В разделе «Тесты и практика» есть возможность прохождения тестирования на уровень знания языка *Java* на 3 уровнях сложности – начальный, средний и сложный. После выбора сложности начинается тестирование.

По окончании тестирования пользователь может увидеть количество набранных баллов, а также посмотреть вопросы, в которых допустил ошибки. Интерфейс онлайн-сервиса представлен ниже на рисунках 8-10.

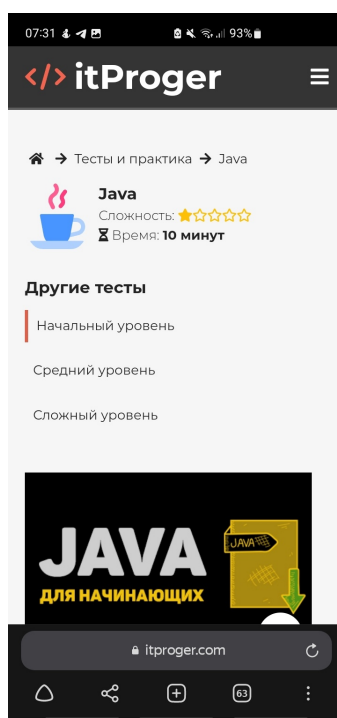


Рисунок 8 – Раздел тестирования *Java* на сервисе *itProger.com*

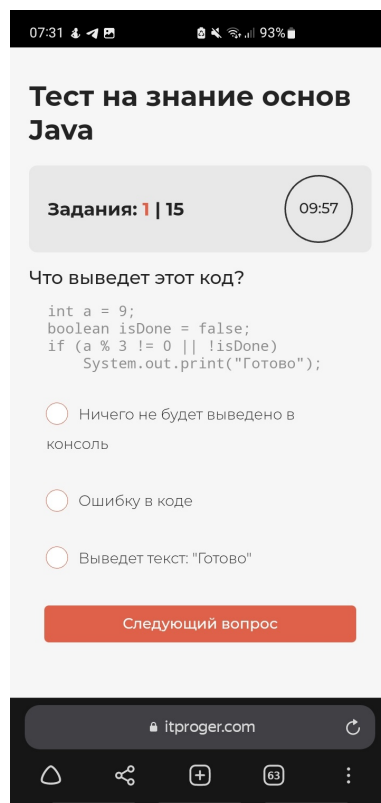


Рисунок 9 – Страница теста на сервисе *itProger.com*

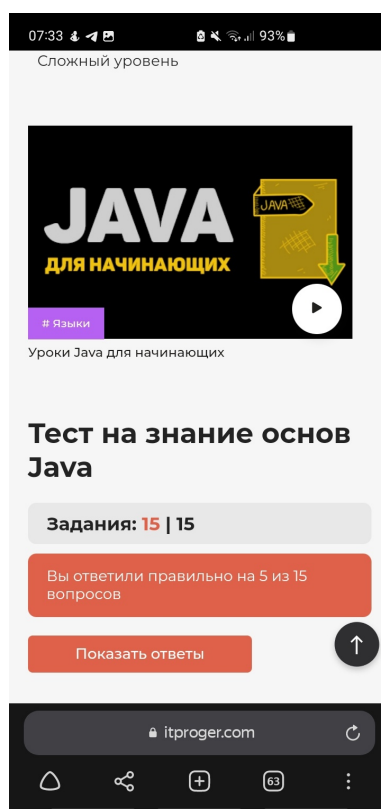


Рисунок 10 – Результат тестирования на сервисе *itProger.com*

					ПГУ1.090301.16.001 ПЗ	Лист
						14
Изм.	Лист	№ докум.	Подпись	Дата		

К достоинствам *itProger.com* можно отнести:

- наличие выбора сложности;
- понятный интерфейс;
- наличие авторизации;
- наличие русской локализации.

Недостатками данного сервиса являются:

- отсутствие разделения вопросов по категориям;
- небольшое количество вопросов в базе.

1.1.3.4 Вывод

Результаты анализа существующих аналогов в таблице 1, показывающей минусы и плюсы рассматриваемых приложений и онлайн-сервисов.

Таблица 1 – Достоинства и недостатки аналогов

Функционал	<i>Java Quiz</i>	<i>Java Test Prep 2023 Ed</i>	<i>itProger.com</i>
Авторизация	-	-	+
Русская локализация	-	-	+
Разделение на категории	+	-	-
Большая база вопросов	+	+	-
Возможность просмотра ответов	+	+	+

1.1.4 Выбор операционной системы

Сейчас на рынке мобильных операционных систем установилось двоевластие. Аудиторию пользователей поделили между собой *iOS* и *Android*, вытеснив практически всех конкурентов. По данным *StatCounter* на март 2023 года, доля пользователей ОС *Android* составляет 70,88%, а доля пользователей ОС *IOS* - 28,42%. Рассмотрим достоинства и недостатки каждой из операционных систем.

Достоинства ОС *Android*:

- открытая платформа с открытым исходным кодом, что позволяет разработчикам создавать более функциональные приложения;
- большой выбор устройств с разными характеристиками и ценами;
- наличие большого количества бесплатных приложений и игр в *Google Play*;
- поддержка многих сервисов *Google*, таких как *Google Maps*, *Gmail*, *Google Drive* и других;
- более гибкая настройка и персонализация интерфейса пользователя.

Недостатки ОС *Android*:

- некоторые устройства не получают обновления операционной системы, что приводит к проблемам с безопасностью и совместимостью;
- некоторые приложения могут работать нестабильно из-за различий в аппаратном обеспечении устройств;
- наличие различных версий операционной системы на разных устройствах может приводить к различиям в функциональности и интерфейсе;
- некоторые устройства могут иметь проблемы с оптимизацией энергопотребления, что может привести к быстрому разряду батареи;
- возможность установки приложений из неофициальных источников может привести к установке вредоносных программ.

Достоинства ОС *IOS*

- высокое качество и производительность устройств *Apple*, что обеспечивает быструю работу операционной системы;
- единая операционная система на всех устройствах *Apple*, что обеспечивает единый интерфейс и функциональность;
- большое количество качественных приложений и игр в *App Store*.
- регулярные обновления операционной системы, что обеспечивает безопасность и стабильность работы устройства;

- хорошая оптимизация энергопотребления, что обеспечивает длительную работу устройства на одной зарядке.

Недостатки ОС *IOS*:

- ограничения при настройке интерфейса и персонализации устройства;
- небольшой выбор устройств с высокой стоимостью;
- ограниченный доступ к файловой системе, что затрудняет работу с файлами;
- невозможность установки приложений из неофициальных источников;
- наличие проприетарной технологии *Apple*, что затрудняет совместимость с устройствами других производителей.

Проанализировав достоинства и недостатки обеих операционных систем, было принято решение разрабатывать приложения под ОС *Android*.

1.1.5 Постановка задачи

Необходимо разработать приложение на телефон под управлением операционной системы *Android* «Программа оценки уровня знаний языка *Java* для отбора кандидатов».

В данной программе будет реализована часть, которая отвечает за регистрацию, вход, тестирование, заполнение и занесение информации о кандидатах в базу данных. Кроме того, на основе полученных результатов тестирования программа автоматически подберет должность, которую может предложить организация для прохождения дальнейшего собеседования.

Используя данное приложение, пользователь(кандидат) сможет:

- регистрироваться;
- авторизовываться;
- проходить тестирование;
- просматривать результаты тестирования;
- редактировать личные данные.

Администратор в свою очередь сможет:

- регистрироваться;

- авторизовываться;
- редактировать вопросы для тестирования;
- просматривать список зарегистрированных пользователей;
- удалять учетные записи пользователей.

Все эти возможности отражены на *UML*-диаграмме вариантов использования на рисунке 11.



Рисунок 11 – *UML*-диаграмма вариантов использования

Программа должна быть разработана для работы в ОС *Android* и подходит для версий *Android* от 10.0 («*Queen cake*») и выше.

1.1.6 Выбор среды разработки

На данный момент существует множество сред разработки для *Android* приложений, но наиболее популярными являются следующие:

- *Eclipse*;
- *Android Studio*.

1.1.6.1 Среда *Eclipse*

Eclipse — это бесплатная *IDE*, разработанная некоммерческой компанией *Eclipse Foundation*. Эта программа является базой, которая регулирует процессы создания приложений.

К достоинствам «*Eclipse*» можно отнести:

- *Eclipse* является бесплатной и открытой средой разработки, которая предоставляет богатый набор инструментов для разработки приложений для *Android*;
- *Eclipse* имеет огромное сообщество пользователей и разработчиков, что обеспечивает быстрый доступ к решениям проблем и поддержке;
- *Eclipse* обеспечивает высокую производительность и качество приложений, благодаря мощным средствам тестирования и отладки;
- *Eclipse* может быть интегрирован с другими инструментами разработки, такими как *Git*, *SVN*, и т.д.;
- *Eclipse* предоставляет широкий набор плагинов и расширений, которые могут быть использованы для расширения функциональности среды разработки.

Очень важным недостатком *Eclipse* является то, что данная среда разработки устарела, из-за чего *Google* в 2016 году отказались от поддержки ее плагинов для *Android*-разработки.

1.1.6.2 Среда *Android Studio*

Android Studio – интегрированная среда разработки для работы с платформой *Android*, разработанная *Google*. Она являет собой набор интегрированных инструментов для эффективной разработки, отладки и тестирования программ, позволяя создать проект любой сложности.

- интегрированная среда разработки (*IDE*) *Android Studio* предоставляет богатый набор инструментов для разработки приложений для *Android*;

- интуитивный интерфейс пользователя и удобная интеграция с системой контроля версий (*VCS*) делают *Android Studio* одной из лучших сред разработки в мире;

- быстрое исправление ошибок и поддержка быстрой разработки обеспечивают высокую производительность и качество приложений;

- *Android Studio* обеспечивает поддержку всех последних технологий и функций, которые можно использовать для разработки приложений для *Android*, таких как *Android Jetpack*, *Kotlin*, *Android Wear*, *Android TV*, и т.д.;

- *Android Studio* предоставляет богатый набор инструментов для тестирования и отладки приложений, включая визуализацию макета и тестирование на различных устройствах;

- почти все *Android*-девелоперы используют *Android Studio*. Благодаря этому можно легко найти интересующую по данной среде информацию в интернете.

Единственный недостаток *Android Studio* - для ее работы нужен достаточно мощный ПК. Поэтому проанализировав достоинства и недостатки представленных сред разработки, было принято решение использовать *Android Studio*.

1.1.6.3 База данных *Firebase Realtime Database*

Поскольку основной задачей приложения является сбор и хранение информации, было необходимо найти удобную и безопасную БД для её хранения. Выбор на *Firebase*, которая отвечает всем этим требованиям.

Firebase — это платформа для разработки приложений, которая предоставляет различные инструменты и сервисы, в том числе базу данных в реальном времени, хостинг, аутентификацию пользователей, облачное хранилище файлов, аналитику, уведомления и другие функции. *Firebase* была разработана компанией *Google* и позволяет разработчикам быстро создавать и разворачивать приложения для веба, *iOS* и *Android*.

Особенности *Firebase*:

- база данных реального времени. *Firebase* предоставляет базу данных в реальном времени, которая позволяет синхронизировать данные между пользователями в режиме реального времени;

- хостинг. *Firebase* предоставляет хостинг, который позволяет размещать веб-сайты и приложения на серверах *Google*;

- аутентификация. *Firebase* обеспечивает безопасную аутентификацию пользователей через социальные сети, электронную почту и пароли;

- хранение файлов. *Firebase* позволяет хранить файлы, такие как изображения, видео и аудиофайлы, в облаке *Google*;

- машинное обучение. *Firebase* включает в себя мощные инструменты машинного обучения, такие как *ML Kit*, который позволяет разработчикам создавать приложения с функциями распознавания изображений и голоса;

- инструменты аналитики. *Firebase* предоставляет множество инструментов аналитики, которые позволяют разработчикам отслеживать поведение пользователей и улучшать опыт использования приложения;

- интеграция с другими сервисами *Google*. *Firebase* интегрируется с другими сервисами *Google*, такими как *Google Cloud Platform*, *AdMob* и *Google Analytics*, что позволяет разработчикам создавать более мощные приложения.

В целом, *Firebase* предоставляет широкий спектр инструментов и функций, которые позволяют разработчикам создавать мощные и инновационные приложения.

1.2 Реализация приложения

1.2.1 Архитектура приложения

Выбор хорошей архитектуры приложения является едва ли не важнейшей частью при создании приложения, поскольку правильно созданная архитектура позволяет вносить изменения с минимальными затратами ресурсов.

При выборе архитектуры приложения было принято решение использовать *MVVM* (*Model-View-ViewModel*). Одним из главных преимуществ *MVVM* является ее способность к тестированию. Каждый компонент приложения может быть протестирован отдельно, что позволяет разработчикам быстро находить и исправлять ошибки. Кроме того, *MVVM* обеспечивает лучшую отделенность компонентов приложения друг от друга, что упрощает поддержку и расширение приложения в будущем. Это особенно важно для больших проектов, где множество разработчиков работает над одним приложением.

Разработанная архитектура будущего приложения представлена на рисунке 12.

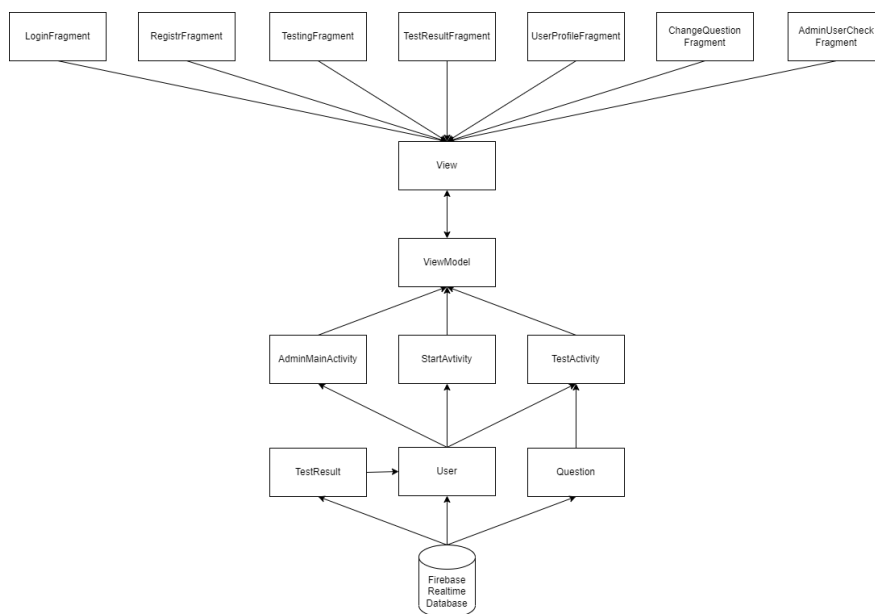


Рисунок 12 – Архитектура приложения

Модель (*Model*) представляет собой слой данных приложения. Она содержит все данные, необходимые для работы приложения, а также методы для получения и обновления этих данных. Модель не зависит от других компонентов приложения и может использоваться в любом контексте. В нашем приложении моделью являются классы *User*, *Question*, *TestResult* и БД *Firebase Realtime Database*.

Представление (*View*) отвечает за отображение данных пользователю. Оно содержит элементы пользовательского интерфейса, такие как кнопки, текстовые поля и т.д. Представление не содержит бизнес-логики и не взаимодействует напрямую с моделью. В данном случае за представление отвечают фрагменты (*Fragments*):

- *LoginFragment* отвечает за авторизацию;
- *RegistrFragment* отвечает за регистрацию;
- *TestingFragment* отвечает за тестирование;
- *TestResultFragment* отвечает за результаты тестирования;
- *UserProfileFragment* отвечает за профиль пользователя;
- *ChangeQuestionFragment* отвечает за изменение вопросов (для администратора);
- *AdminUserCheckFragment* отвечает за просмотр данных пользователя (для администратора).

Модель представления (*ViewModel*) является связующим звеном между моделью и представлением. Она содержит бизнес-логику приложения и методы для обработки пользовательских действий. Модель представления также отвечает за обновление представления при изменении данных в модели. В данном случае моделью представления выступают активности (*Activities*):

- *StartActivity* отвечает за фрагменты регистрации и авторизации;
- *TestActivity* отвечает за фрагменты профиля пользователя, тестирования, результатов тестирования;

- *AdminMainActivity* отвечает за просмотр всех учетных записей кандидатов и редактирование вопросов.

UML-диаграмма классов и *UML*-диаграмма развертывания разработанного приложения представлены на рисунках 13 и 14 соответственно.

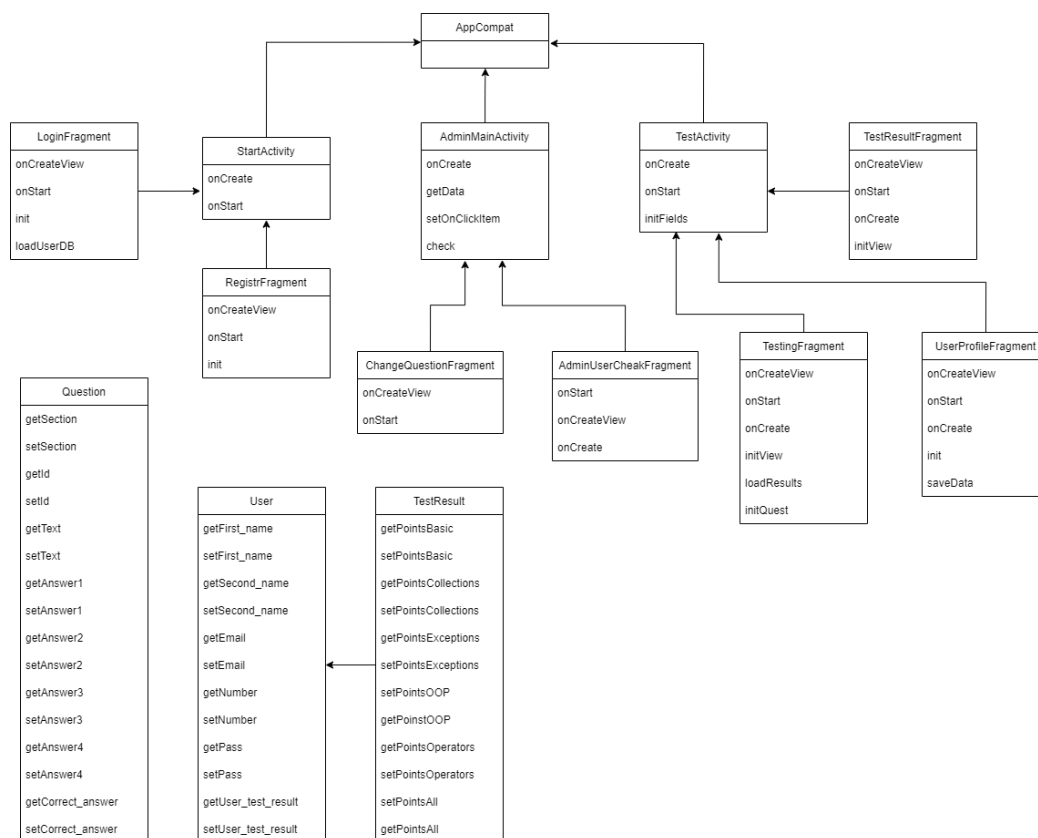


Рисунок 13 - *UML*-диаграмма классов



Рисунок 14 - *UML*-диаграмма развертывания

1.2.2 Создание проекта в *Android Studio*

Чтобы создать новый проект в *Android Studio* необходимо выполнить следующие действия:

- выбрать вариант *Activity* при создании (рисунок 15);
- далее в окне создания проекта требуется придумать название проекта, название его пакета, расположение, язык программирования и указать минимально допустимую версию операционной системы, под которую будет разрабатываться приложение (рисунок 16).

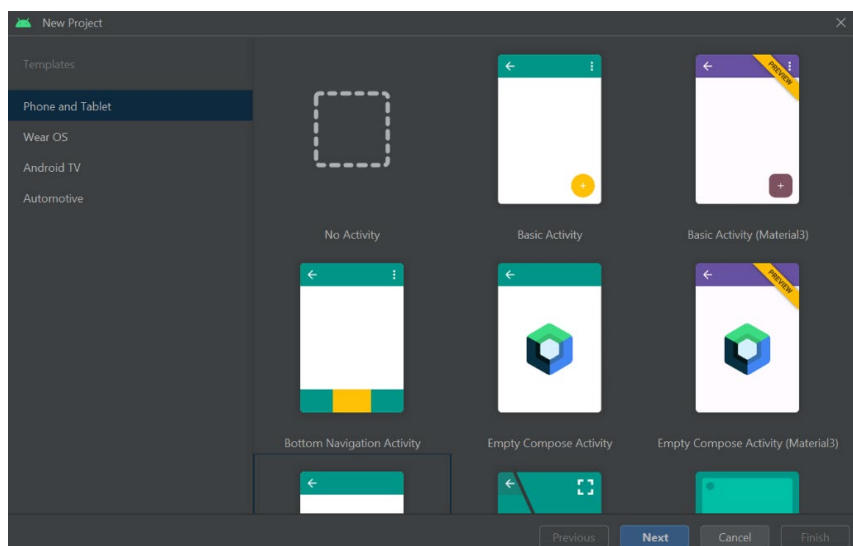


Рисунок 15 – Окно выбора *Activity*

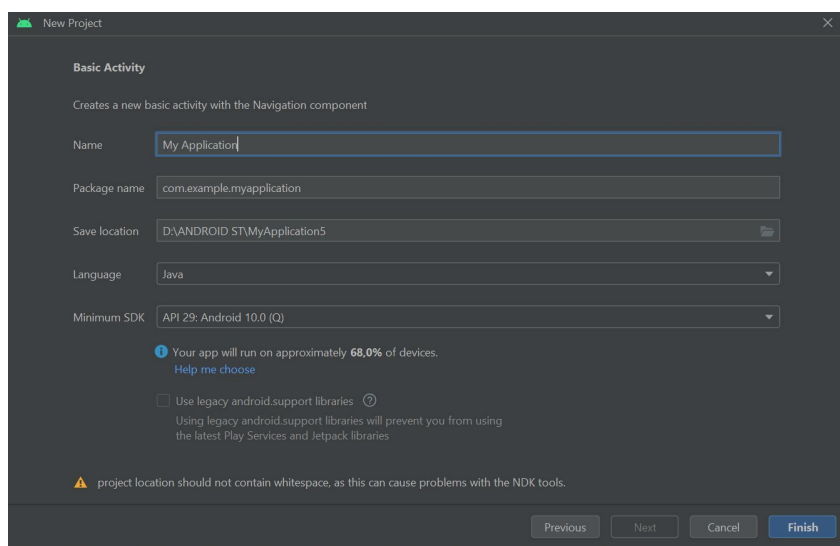


Рисунок 16 – Окно предварительной настройки проекта

В левом окне среды разработки расположено дерево файлов проекта, где находятся все основные папки и файлы текущего проекта (рисунок 17).

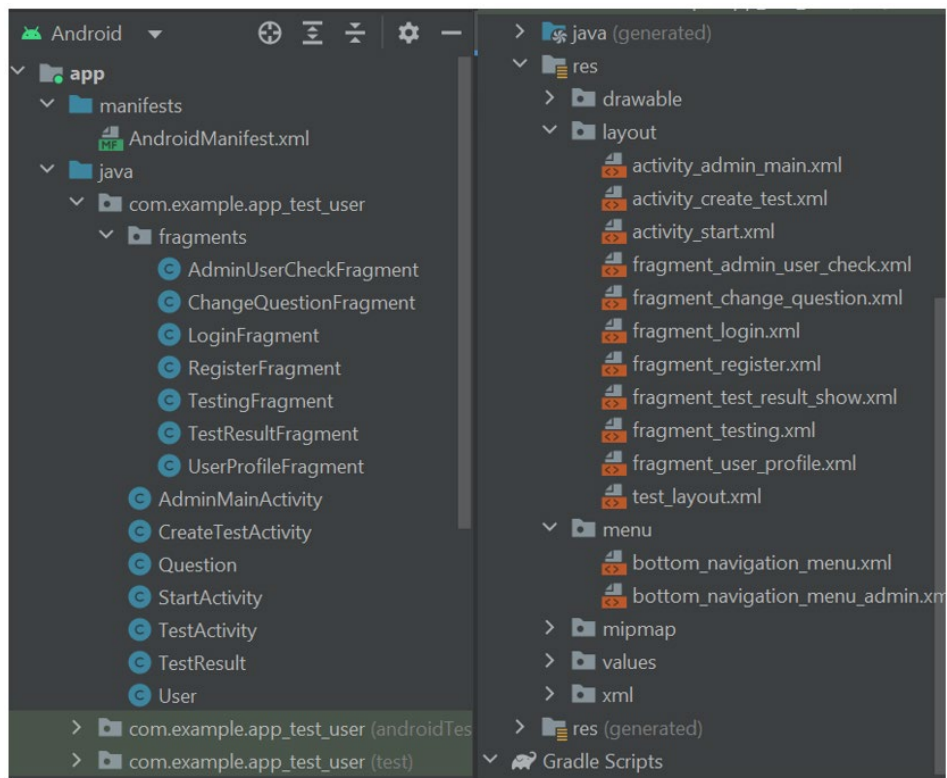


Рисунок 17 – Дерево файлов проекта

Рассмотрим каждый модуль подробнее:

- модуль *Manifest* содержит файл *AndroidManifest.xml*, который является главным файлом манифеста приложения. В этом файле определяются все компоненты приложения, такие как *Activity*, *Service*, *BroadcastReceiver* и *ContentProvider*. Также в этом файле указываются разрешения, которые требуются для работы приложения, и метаданные, которые могут использоваться другими компонентами приложения;

- модуль *Java* содержит исходный код приложения на языке программирования *Java*. Здесь находятся все классы, которые определяют поведение приложения. В этом модуле можно создавать новые классы, интерфейсы, абстрактные классы и т.д. Также здесь можно использовать библиотеки и фреймворки, которые помогают упростить разработку приложения;

- модуль *Res* содержит все ресурсы приложения, такие как изображения, макеты, строки, цвета и т.п. Тут находятся все файлы, которые используются при создании пользовательского интерфейса приложения;

- модуль *Gradle Scripts* содержит в себе скрипты *Gradle*, которые используются для сборки проекта. Здесь можно настроить сборку проекта, указать зависимости от других библиотек и фреймворков, настроить подпись приложения и т.д.

1.2.3 Реализация аутентификации пользователей

Система аутентификации была построена на основе *Firebase Authentication*. *Firebase* предоставляет различные методы аутентификации для приложений *Android*, включая аутентификацию по электронной почте и паролю, аутентификацию через социальные сети, такие как *Facebook* и *Google*, и аутентификацию через телефон. Дело в том, что *Firebase Authentication* создана на основе токенов, что обеспечивает легкую интеграцию с большинством платформ.

При использовании аутентификации *Firebase*, пользователи могут создавать учетные записи в нашем приложении, а также входить в систему с помощью своих учетных записей. *Firebase* обрабатывает все аспекты аутентификации, включая хранение учетных записей пользователей, проверку подлинности учетных записей и управление сессиями пользователей.

Для более удобного способа работы с данными пользователей были разработаны два класса – *User* и *TestResult*. Класс *User* содержит в себе поля *first_name* и *second_name*, отвечающие за имя и фамилию пользователя соответственно, поле *email* – почта, поле *number* – номер телефона пользователя, поле *pass* – пароль пользователя, а также поле *user_test_result*, которое хранит в себе результаты тестирования.

Аутентификация использует метод *signInWithEmailAndPassword()*, который принимает на вход 2 значения – *email* и *password*. Значение *password*

					ПГУ1.090301.16.001 ПЗ	Лист
						27
Изм.	Лист	№ докум.	Подпись	Дата		

берется из второго поля *EditText*. Значение *email* получается путем загрузки данных пользователя после нажатия на кнопку «Войти» - после нажатия на кнопку берется значение из первого поля *EditText*, куда вводится номер телефона, и выполняется загрузка данных из *Firebase Realtime Database*, где номер телефона является уникальным идентификатором. А для того, чтобы случайно не считать вместо номера или пароля пользователя пустое поле, выполняется специальная проверка.

Созданный интерфейс для аутентификации пользователей приведён на рисунке 18.

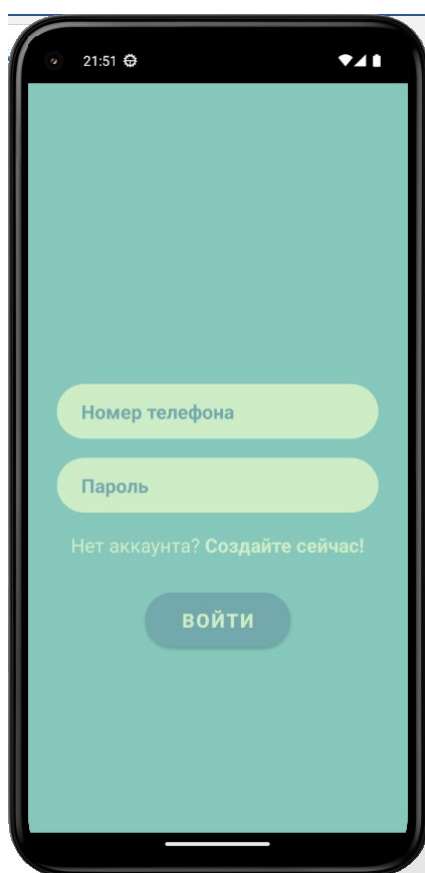


Рисунок 18 – Страница авторизация приложения

Код функции для авторизации пользователей представлен ниже:

```
btn_SingIn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
```

```

String idUserNumber = edLog.getText().toString();
TempList = new ArrayList<>();
FirebaseDatabase
.getInstance("https://otstavnovdiploma-default-rtdb.europe-
west1.firebaseio.com/app/")
.getReference()
.child("Users")
.child(idUserNumber).addListenerForSingleValueEvent(new
 ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        tempUserrrrr = snapshot.getValue(User.class);
        TempList.add(tempUserrrrr);
        User userTemp = TempList.get(1);
        email = userTemp.getEmail();
        String pass = edPass.getText().toString();
        if (TextUtils.isEmpty(email))
            Toast.makeText(getActivity(), "Введите логин",
Toast.LENGTH_SHORT).show();
        if (TextUtils.isEmpty(pass))
            Toast.makeText(getActivity(), "Введите пароль",
Toast.LENGTH_SHORT).show();
        usersAuth.signInWithEmailAndPassword(email, pass)
            .addOnSuccessListener(new OnSuccessListener<AuthResult>() {
                @Override
                public void onSuccess(AuthResult authResult) {
                    Intent intent = new Intent(getActivity(),
TestActivity.class);
                    intent.putExtra("userFName", userTemp.getFirst_name());
                    intent.putExtra("userSName", userTemp.getSecond_name());
                    intent.putExtra("userEmail", userTemp.getEmail());
                    intent.putExtra("userNumber", userTemp.getNumber());
                    intent.putExtra("userPass", userTemp.getPass());
                    intent.putExtra("pointsAll",

```

```

userTemp.user_test_result.getPointsAll());
    intent.putExtra("pointsBasic",
userTemp.user_test_result.getPointsBasic());
    intent.putExtra("pointsCol",
userTemp.user_test_result.getPointsCollections());
    intent.putExtra("pointsExc",
userTemp.user_test_result.getPointsExceptions());
    intent.putExtra("pointsOOP",
userTemp.user_test_result.getPointsOOP());
    intent.putExtra("pointsOper",
userTemp.user_test_result.getPointsOperators());
    Toast.makeText(getActivity(), "Успешная авторизация",
Toast.LENGTH_SHORT).show();
    startActivity(intent);
}
}).addOnFailureListener(new OnFailureListener() {
@Override
public void onFailure(@NonNull Exception e) {
    Toast.makeText(getActivity(), "Ошибка авторизации",
Toast.LENGTH_SHORT).show();}}});});});
});

```

Для отслеживания нажатия на кнопку был использован метод *setOnClickListener*. Чтобы авторизовать пользователя через *FirebaseAuth* используется метод *signInWithEmailAndPassword*, несмотря на то что пользователь при авторизации указывает номер телефона.

После авторизации производится переход на *TestActivity*, в которой пользователь может пройти тест, посмотреть результаты тестирования, если уже проходил его или же изменить личные данные. Учётные записи пользователей, которые успешно прошли регистрацию, можно увидеть в консоли *Firebase* как в разделе *Authentication*, представленном на рисунке 19, так и в разделе *Realtime Database*, представленном на рисунке 20.

<div> <div> <div>🔍</div> <div>Search by email address, phone number, or user UID</div> </div> <div>Add user</div> <div>↻</div> <div>⋮</div> </div>				
Identifier	Providers	Created ↑	Signed In	User UID
alex13otst58...	✉	Apr ...	Apr ...	2CeeBJJ8OZMsSy...
uuuu@mail.co...	✉	Apr ...	Apr ...	QOvY6l2DqhTA2it2...
eeeeeeee@m...	✉	Apr ...	Apr ...	3d03SULJYbcW6t...
trap65@mail.ru	✉	Apr ...	Apr ...	GSikcSbYdnRjXUJ...

Рисунок 19 – Список учётных записей в разделе *Authentication*

Realtime Database	
Data	Rules Backups Usage Extensions NEW
https://otstavnovdiploma-default-rtdb.europe-west1.firebaseio.com	
▶	59640590396
▶	87654787654
▶	88005553535
▶	88888888888
▶	89022050046
▶	89022050066
▶	89034567778
▶	99999999999

Рисунок 20 – Список учётных записей в разделе *Realtime Database*

1.2.4 Реализация работы с *Firebase Realtime Database*

Firebase Realtime Database – это база данных в реальном времени, которая позволяет разработчикам создавать приложения, которые могут обмениваться данными в режиме реального времени. Это означает, что приложения могут обновлять данные на сервере и получать обновления с сервера без необходимости перезагрузки страницы или приложения. *Firebase Realtime Database* использует модель данных *JSON*, что делает его очень

гибким и удобным для использования. Разработчики могут легко создавать и обновлять данные в базе данных, используя *API Firebase*.

Для обращения в тот или иной узел БД используются 2 метода - *getInstance()* и *getReference()*.

Метод *getInstance()* используется для получения экземпляра базы данных *Firebase*. Этот метод возвращает объект *FirebaseApp*, который представляет собой точку входа в *Firebase*.

Метод *getReference()* используется для получения ссылки на базу данных *Firebase*. Код, который используется для обращения к узлу “*Users*” представлен ниже:

```
FirebaseDatabase db = FirebaseDatabase.getInstance(
    "https://otstavnovdiploma-default-rtdb.europe-
west1.firebaseiodatabase.app");
DatabaseReference usersRef = db.getReference("Users");
```

1.2.5 Реализация тестирования

Тестирование затрагивает 5 разделов языка *Java* – Базовый (*Basic*), Коллекции (*Collections*), Исключения (*Exceptions*), Объектно-ориентированное программирование (*OOP*) и Операторы (*Operators*). Каждый раздел включает в себя по 5 вопросов. На рисунке 21 представлен список разделов, которые хранятся в *Firebase Realtime Database*.

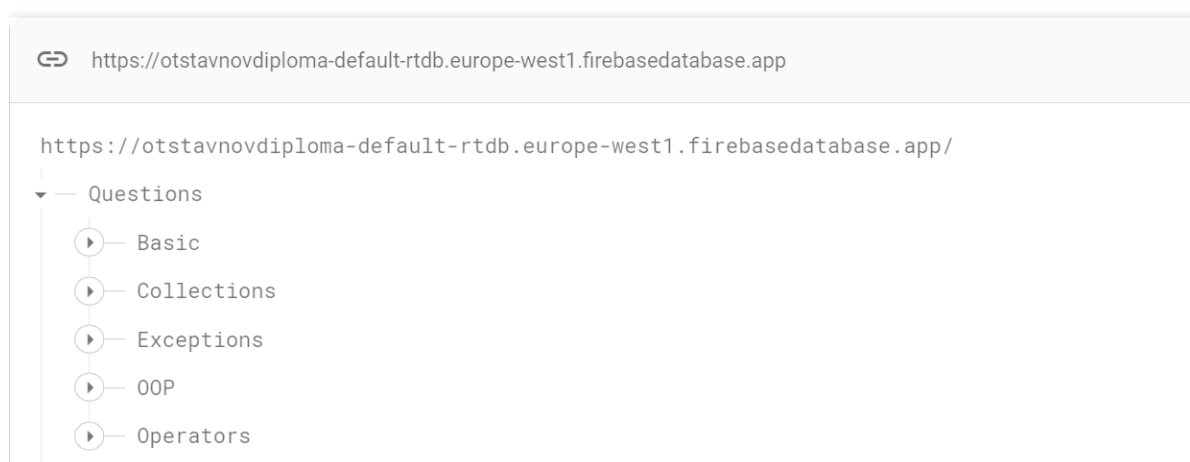


Рисунок 21 – Список разделов с вопросами в *Realtime Database*

Для более удобного способа работы с вопросами был разработан класс *Question*, код которого без конструкторов представлен ниже:

```
public class Question {
    public String section, id, text, answer1, answer2,
    answer3, answer4, correct_answer;},
```

где поле *section* отвечает за раздел, к которому относится вопрос, поле *id* – идентификатор, поле *text* – текст вопроса, поля *answer1*, *answer2*, *answer3*, *answer4* – варианты ответов, поле *correct-answer* – правильный ответ. Представление вопроса в БД показано на рисунке 22:

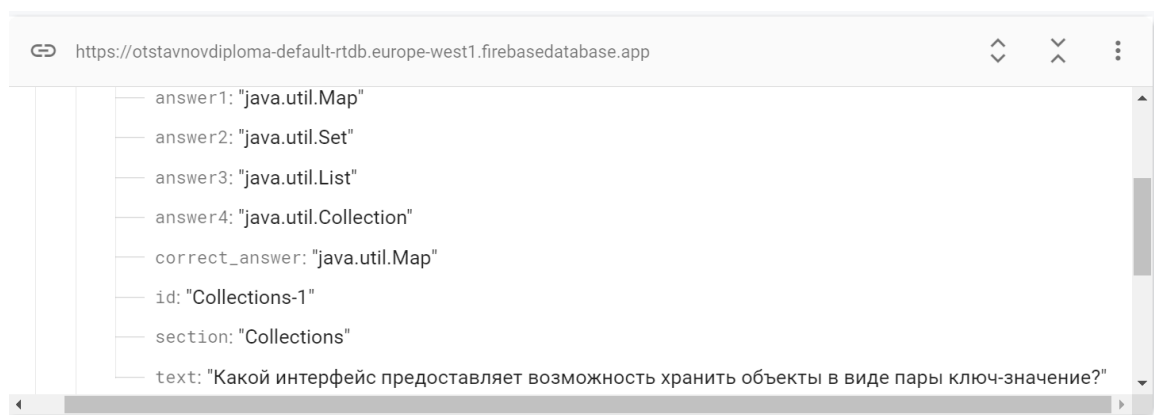


Рисунок 22 – Список разделов с вопросами в *Realtime Database*

Загрузка из БД и инициализация списка вопросов происходит в функции *initQuest()*, код которой представлен ниже. Каждую пятую итерацию меняется раздел БД, из которого загружаются вопросы:

```
public void initQuest() {
    questionList = new ArrayList<>();
    String idSectionBasic = "Basic";
    String idSectionCollections = "Collections";
    String idSectionExceptions = "Exceptions";
    String idSectionOOP = "OOP";
    String idSectionOperators = "Operators";
    String id_temp = "";
    String sec_temp = "";
```

```

for (int i = 2; i <= 25; i++) {
    sec_temp = idSectionBasic;
    id_temp = sec_temp + "-" + i;
    int temp;

    if (i >= 6 && i <= 10) {
        temp = i - 5;
        sec_temp = idSectionCollections;
        id_temp = sec_temp + "-" + temp;
    }

    if (i >= 11 && i <= 15) {
        temp = i - 10;
        sec_temp = idSectionExceptions;
        id_temp = sec_temp + "-" + temp;
    }

    if (i >= 16 && i <= 20) {
        temp = i - 15;
        sec_temp = idSectionOOP;
        id_temp = sec_temp + "-" + temp;
    }

    if (i >= 21) {
        temp = i - 20;
        sec_temp = idSectionOperators;
        id_temp = sec_temp + "-" + temp;
    }

    FirebaseDatabase
        .getInstance("https://otstavnovdiploma-default-
rtdb.europe-west1.firebaseio.com/")
        .getReference()
        .child("Questions")
        .child(sec_temp)
        .child(id_temp).addListenerForSingleValueEvent(new
ValueEventListener() {

```

```

        @Override
        public void onDataChange(@NonNull
DataSnapshot snapshot) {
            tempQuestion =
snapshot.getValue(Question.class);
            questionList.add(tempQuestion);
        }
        @Override
        public void onCancelled(@NonNull
DatabaseError error) {
        }
    });
}

```

Далее текст вопроса выводится в *TextView*, а варианты ответов в соответствующие кнопки *Button*. Интерфейс фрагмента тестирования представлен на рисунке 23.

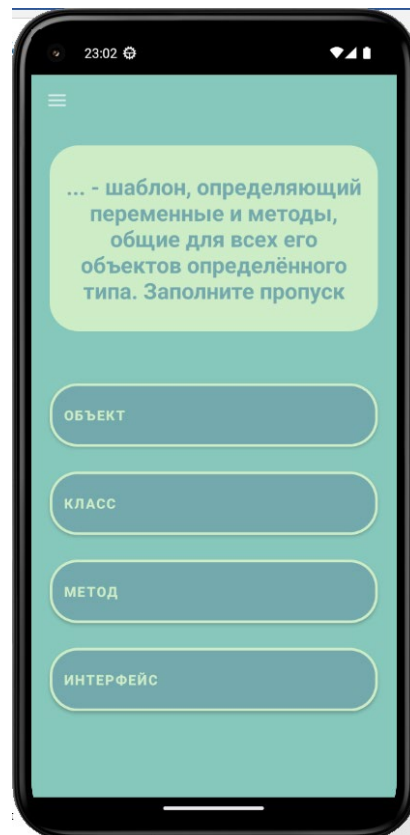


Рисунок 23 – Интерфейс фрагмента тестирования

На каждую кнопку прикреплен слушатель *setOnClickListener(new View.OnClickListener())*, код которого представлен ниже. При нажатии текст, который выведен на кнопке, сравнивается со значением переменной *correct_answer*, и в том случае, когда значения оказываются одинаковыми, счётчик баллов текущего раздела увеличивается.

```
ans2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if (correct_ans.equals(ans2.getText().toString())) {
            switch (section) {
                case "OOP":
                    pointOPP++;
                    break;
                case "Collections":
                    pointCollections++;
                    break;
                case "Basic":
                    pointBasic++;
                    break;
                case "Exceptions":
                    pointExceptions++;
                    break;
                case "Operators":
                    pointOperators++;
                    break;
            }
        }
        iter++;
        if (iter >= 25) {
            userTestResult = new TestResult(pointBasic,
            pointCollections, pointExceptions, pointOPP, pointOperators);
            loadResults();
            getParentFragmentManager().beginTransaction().addToBackStack(null)
            .replace(R.id.home_main_container, new
```

```

    TestResultFragment (userTestResult)).commit();
}
if (iter < 25) {
    questTemp = questionList.get(iter);
    txtView.setText(questTemp.getText());
    ans1.setText(questTemp.getAnswer1());
    ans2.setText(questTemp.getAnswer2());
    ans3.setText(questTemp.getAnswer3());
    ans4.setText(questTemp.getAnswer4());
    correct_ans = questTemp.getCorrect_answer();
    section = questTemp.getSection();
}
}
});

```

Для получения информации об уровне знаний кандидата был сделан опрос, алгоритм работы которого представлен на рисунке 24.

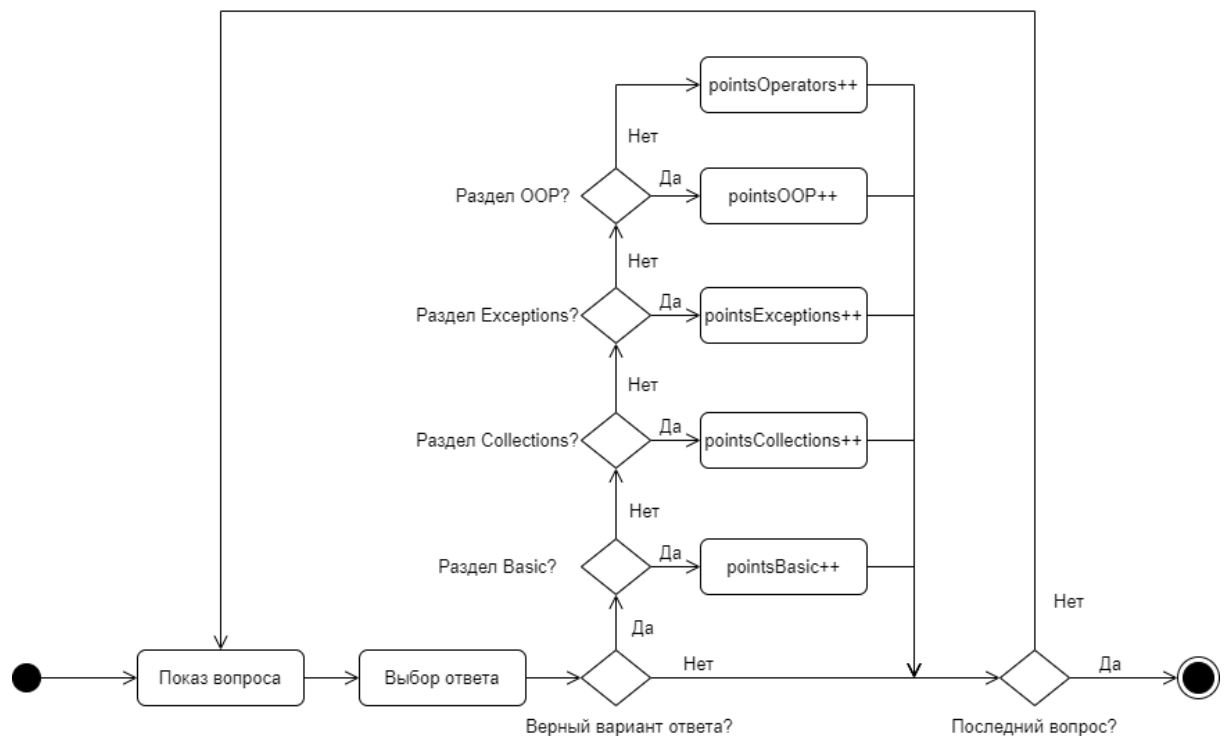


Рисунок 24 – *UML*-диаграмма деятельности опроса

Когда счётчик нажатий будет равен количеству вопросов в тесте,

результаты тестирования сохранятся в БД (рис.25), произойдет переключение на фрагмент, который показывает количество набранных баллов за тестирование, а также вакансии, на которые компания может предложить продолжить собеседование. Результат тестирования представлен на рисунке 26. При нажатии на поле «Общее количество набранных баллов» на экран выводятся количество баллов, которое было набрано в каждом разделе.

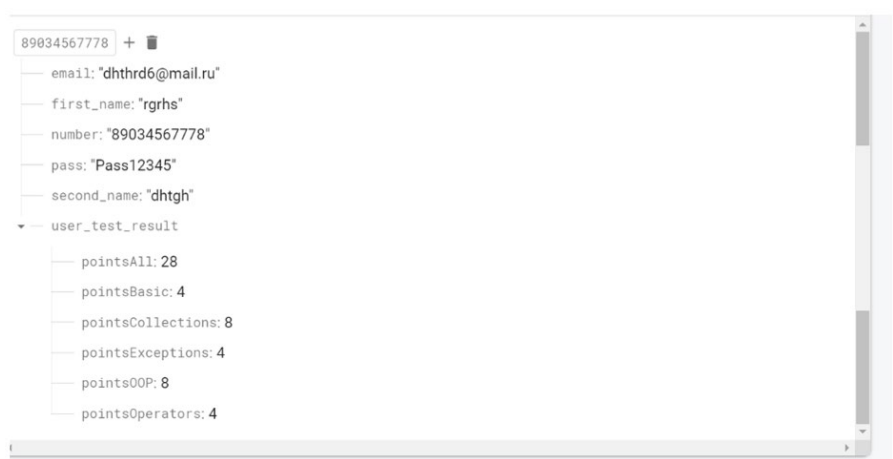


Рисунок 25 – Результат тестирования в БД



Рисунок 26 – Результат тестирования в приложении

В зависимости от количества набранных баллов, пользователю будет предложено продолжить собеседование на ту или иную должность:

- менее 50 – не предлагается продолжить собеседование;
- от 50 до 70 – предлагается продолжить собеседование на должность *Junior-разработчика Java*;
- от 71 до 85 – предлагается продолжить собеседование на должность *Junior-разработчика Java, Middle-разработчика Java*;
- от 85 и выше – предлагается продолжить собеседование на должность *Junior-разработчика Java, Middle-разработчика Java, Senior-разработчика Java*.

1.2.6 Интерфейс приложения

Создание интерфейса для проекта играет очень важную роль. Дизайн одновременно должен быть удобным для использования, простым для понимания, но и предоставлять всю нужную информацию. Для достижения этих целей было принято решение использовать фрагменты и *BottomNavigationBar*, вариации которого можно увидеть на рисунках 27-28.



Рисунок 27 – *BottomNavigationBar* для кандидатов



Рисунок 28 – *BottomNavigationBar* для администратора

Fragment — является модульной частью *activity*, у которой есть свой жизненный цикл, а также свои обработчики различных событий. Схема перехода между экранами представлена на рисунке 29. Интерфейсы всех фрагментов представлены ниже на рисунках 30-34.

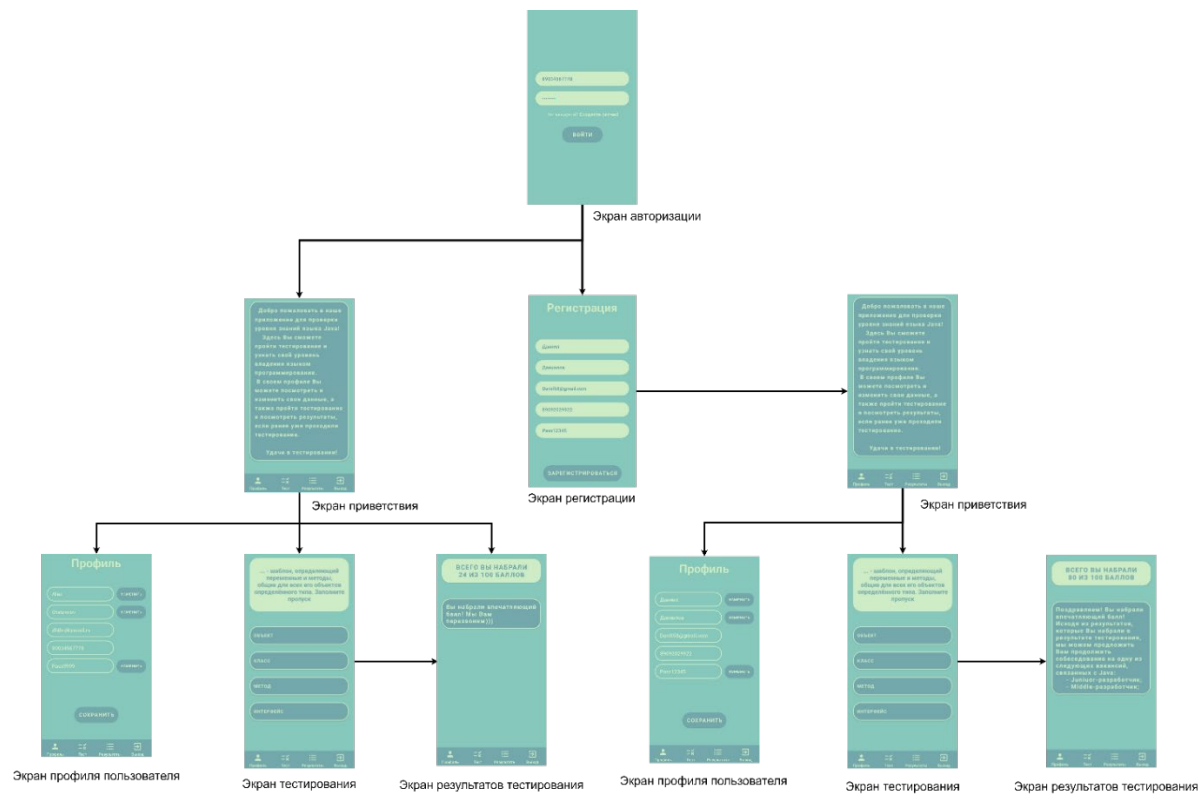


Рисунок 29 – Схема переходов между экранами приложения

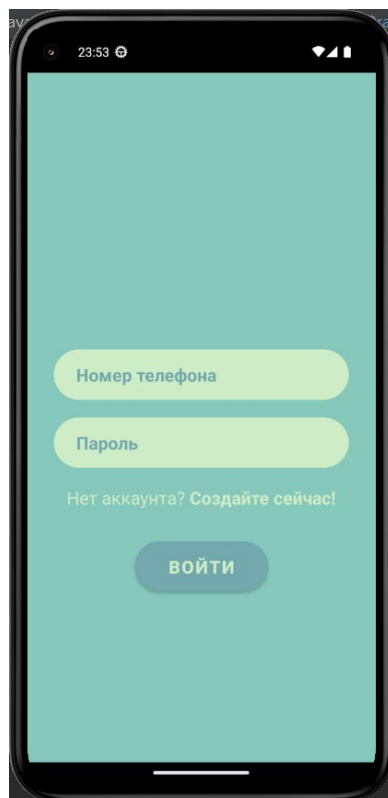


Рисунок 30 – Фрагмент авторизации

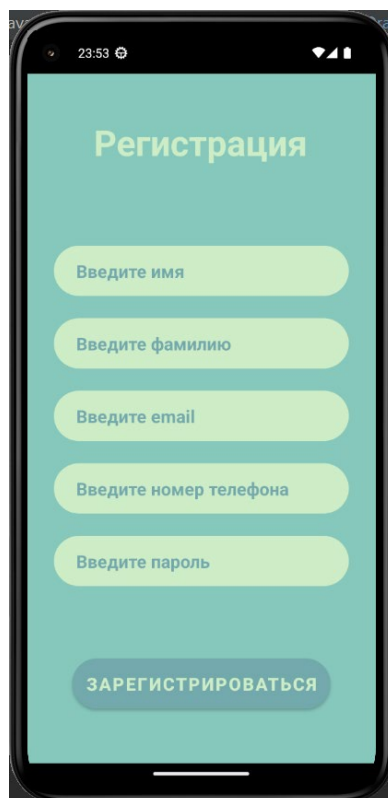


Рисунок 31 – Фрагмент регистрации

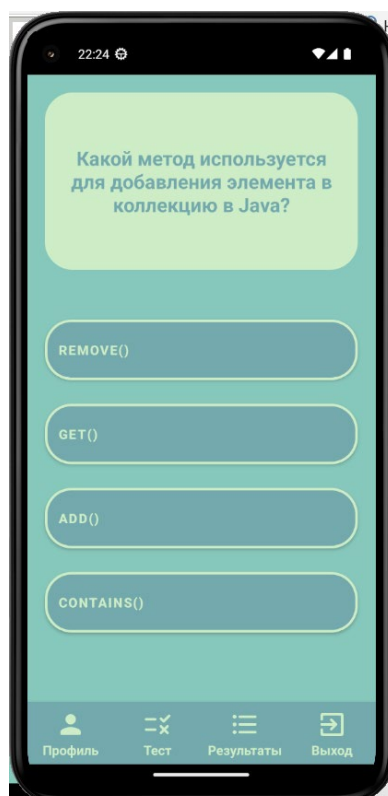


Рисунок 32 – Фрагмент тестирования

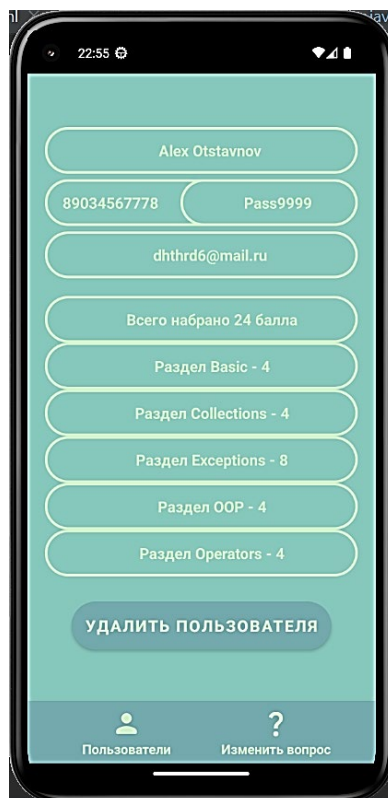


Рисунок 33 – Фрагмент просмотра данных пользователя

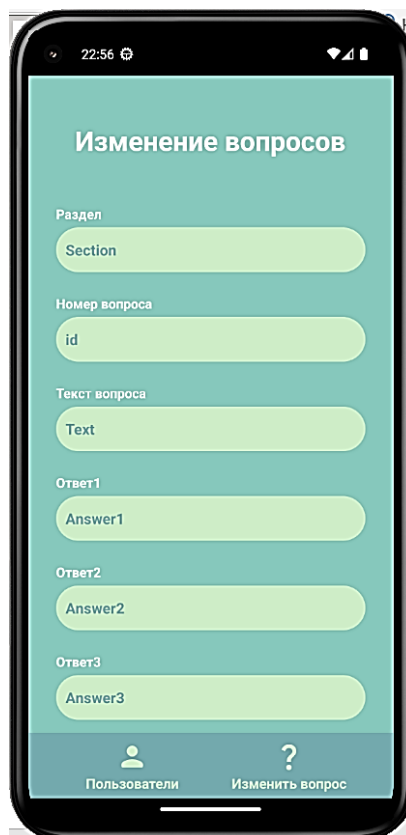


Рисунок 34 – Фрагмент для изменения вопросов теста

2 Безопасность жизнедеятельности

2.1 Рекомендации по безопасному использованию ПК

Длительное времяпровождение за компьютером может повлечь за собой накопительное вредное воздействие на здоровье человека. Другими словами, результат такого воздействия проявляется не сразу, а может постепенно и незаметно ухудшать состояние организма. Одновременно одними из самых распространенных и одними из самых опасных для человека являются следующие заболевания:

- синдром «сухого глаза», вызванный высокой нагрузкой на глаза;
- различные заболевания суставов, которые появляются из-за неправильной позы во время длительной работы за компьютером.

Однако, если соблюдать несколько несложных правил продолжительной работы за компьютером, то можно минимизировать это негативное влияние.

- нужно находиться на безопасном расстоянии до монитора, чтобы снизить нагрузку на глаза. Как правило это расстояние должно составлять 50-70 см. Можно рассмотреть возможность использования специальных защитных экранов или очков с фильтром синего света;
- для минимизации нагрузки шейный отдел рекомендуется соблюдать угол экранного наклона, равный примерно 30 градусам;
- чтобы уменьшить риск возникновения болей в кистях, клавиатура должна находиться в 20-30 см от края стола. Расположение клавиатуры и мыши должно быть на уровне, удобном для работы в натуральной и расслабленной позе. Дополнительно можно использовать подставку для запястий для предотвращения развития туннельного синдрома;
- очень важно, чтобы стул или кресло обеспечивали прямую осанку, при которой спина немного упирается в спинку;

- чтобы позвоночник находился в естественном положении, ноги не должны быть согнуты под стул или кресло, а должны быть выпрямлены вперед с упором в твердую поверхность, т.е. ноги в коленях должны быть согнуты под углом 90 градусов;

- регулярные паузы во время работы за компьютером должны быть обязательно. Примерно каждый час нужно организовывать небольшой перерывы, в течение которых следует сделать короткие упражнения для глаз и шеи. Это поможет снизить напряжение и усталость.

При соблюдении нескольких вышеописанных правил, риск ущерба здоровью и имуществу значительно снижается.

2.2 Рекомендации по безопасному использованию смартфона

В современном обществе мобильные телефоны и смартфоны являются неотъемлемой частью нашей жизни. Во всем мире постоянно увеличивается число людей, которые активно используют смартфоны, чтобы решать бытовых задач. Вместе с этим растет и количество несчастных случаев, связанных с непосредственно с этими устройствами. Существует несколько простых правил, соблюдая которые, можно значительно снизить вредное воздействие смартфона на организм:

- рекомендуется подносить смартфон к уху лишь после того, как собеседник ответит. Дело в том, что в момент вызова телефон работает на максимуме своей мощности. Однако уже через 10–20 секунд после начала вызова излучаемая мощность снижается до минимально допустимого значения;

- периодически нужно делать перерывы от использования смартфона. Можно сделать растяжку, упражнения для глаз. Это поможет снизить напряжение и усталость от длительного использования смартфона;

- держать трубку мобильного телефона нужно на расстоянии от уха, за нижнюю ее часть и вертикально. Затухание радиоволн происходит

пропорционально квадрату пройденного расстояния, поэтому, отодвинув трубку от уха всего на сантиметр и увеличив, таким образом, расстояние до мозга вдвое, можно понизить мощность, излучения воздействия на мозг, в четыре раза;

- обязательно нужно следить за своей осанкой и положением тела во время использования смартфона. Смартфон желательно расположить на уровне глаз, чтобы снизить нагрузку на шею и спину;

- синий свет, излучаемый экраном смартфона, может сказываться на качестве сна. Перед сном стоит ограничить использование смартфона и включить ночной режим или фильтр синего света, чтобы уменьшить его воздействие;

- если владелец смартфона носит очки с металлической оправой, то перед разговором рекомендуется их снять, так как металл может увеличить интенсивность электромагнитного поля;

- чтобы отказаться от лишнего облучения, нужно перестать ложиться спать рядом со смартфоном;

- следует выделить в доме зоны без электронных устройств. Регулярно проводите периоды без использования смартфона, чтобы дать своему мозгу и глазам отдохнуть;

- существуют приложения, которые помогают отслеживать и ограничивать время использования смартфона. Следует установить такие приложения, чтобы контролировать время, проводимое перед экраном, и впоследствии уменьшить его;

- длительность телефонного разговора не должна быть более двух минут, а перерыв между звонками должен составлять минимум 15 минут;

- отключите все уведомления приложений, кроме самых важных. Если полностью отключить уведомления невозможно, рекомендуется выделить определенное время в течение дня, когда они не будут вас беспокоить;

- следует установить оптимальную яркость экрана смартфона, чтобы снизить напряжение глаз и избежать излишнего утомления, а также включить режим ночного режима или фильтр синего света, чтобы сократить его негативное воздействие на сон;

- отключить опцию «получать новые письма» на электронной почте. Благодаря этому просмотр почты будет возможен только при открытии приложения;

- ограничение времени, проводимого перед экраном смартфона. Постоянное использование смартфона может привести к перенапряжению глаз и снижению активности. Можно установить себе лимиты на использование экрана и придерживаться их.

При соблюдении вышеописанных правил, риск ущерба собственному здоровью значительно снижается.

					ПГУ1.090301.16.001 ПЗ	Лист
						46
Изм.	Лист	№ докум.	Подпись	Дата		

Заключение

В результате выполнения данной выпускной квалификационной работы было создано мобильное приложение с удобным и понятным пользовательским интерфейсом. Созданный проект дает возможность получать информацию об уровне знаний языка *Java* кандидатов посредством тестирования.

Для получения требуемой функциональности приложения использовались такие технологии, как язык программирования *Java*, база данных реального времени *Firebase Realtime Database*, среда разработки *Android Studio*. На основе этих технологий, было успешно разработано и протестировано приложение, с функционалом, который включает в себя:

- авторизацию в приложении;
- регистрацию в приложении;
- прохождение тестирования;
- просмотр результатов теста;
- чтение, редактирование и сохранение данных о пользователях в БД;
- автоматический подбор должности, на которую организация может предложить продолжить собеседование.

В процессе работы над данным проектом были улучшены навыки в разработке мобильных приложений под управлением ОС *Android*.

На основе анализа полученных результатов можно сделать вывод, что все функции приложения отрабатываются корректно в штатном режиме.

В заключение можно сказать, что разработка приложения оценки уровня знаний языка *Java* для отбора кандидатов определенно повысит уровень оптимизации подбора кадров. Результаты работы показали, что приложение успешно выполняет поставленные задачи и может быть использовано в сфере *IT*-рекрутинга.

Список используемых источников

1. Гамма Э., Хелм., Джонсон Р., Влиссидес. Приемы объектно-ориентированного проектирования. Паттерны проектирования. – СПб.: Питер, 2018.
2. Дейтел П., Дейтел Х., Уолд А. *Android* для разработчиков. 3-е изд. – СПб.: Питер, 2017.
3. Программирование под *Android* на *Java* [Электронный ресурс] / Режим доступа: <https://metanit.com/java/android> / (дата обращения: 27.04.2023).
4. *Firebase – Realtime Database* [Электронный ресурс] / Режим доступа: <https://firebase.google.com/docs/database> / свободный. (дата обращения: 01.05.2023).
5. *Firebase – Authentication* [Электронный ресурс] / Режим доступа: <https://firebase.google.com/docs/auth> / свободный. (дата обращения: 03.05.2023).

Приложение А

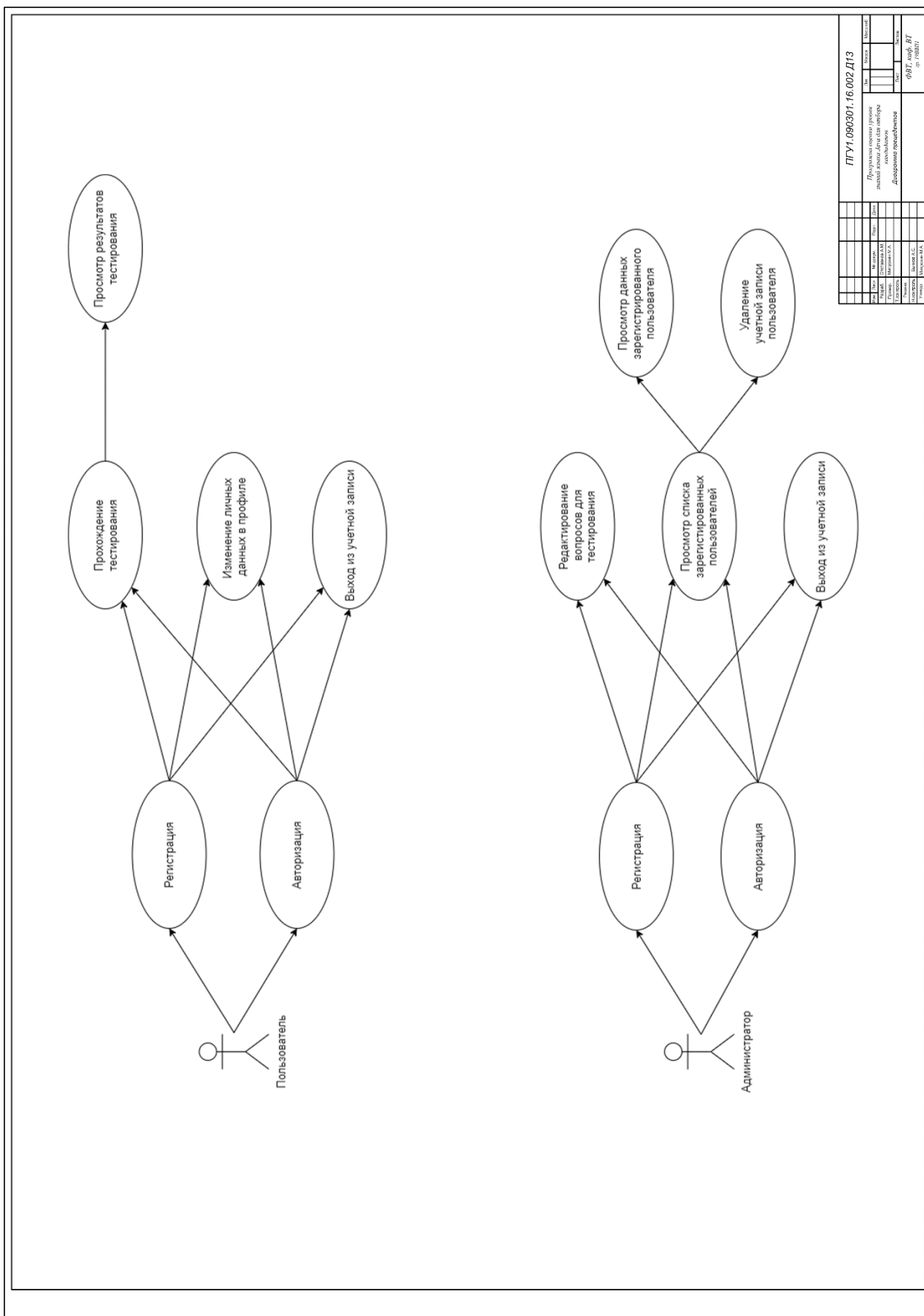
(Обязательное)

Программа оценки уровня знаний языка Java для отбора кандидатов

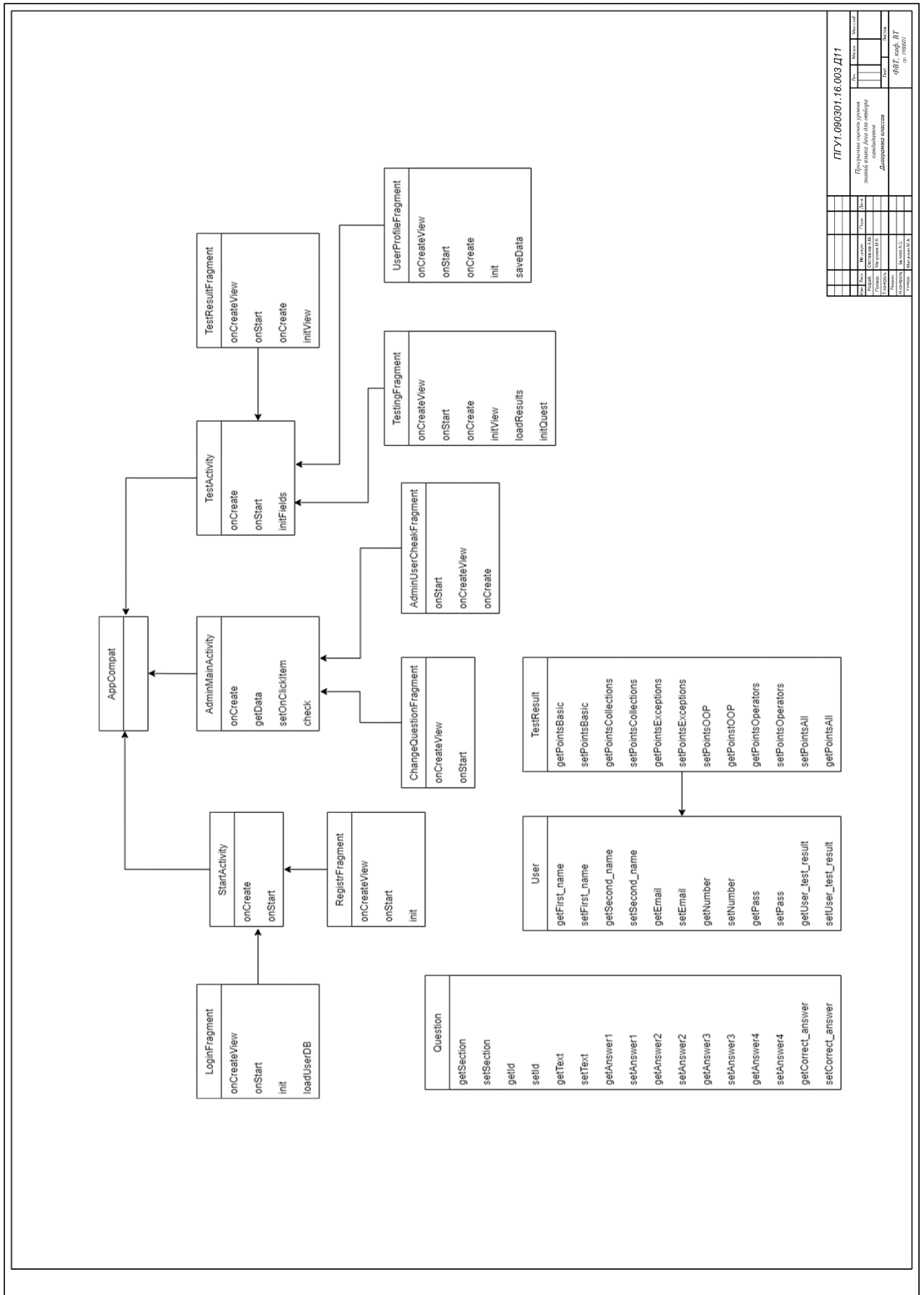
UML-диаграммы

					ПГУ1.090301.16.001 ПЗ	Лист
						49
Изм.	Лист	№ докум.	Подпись	Дата		

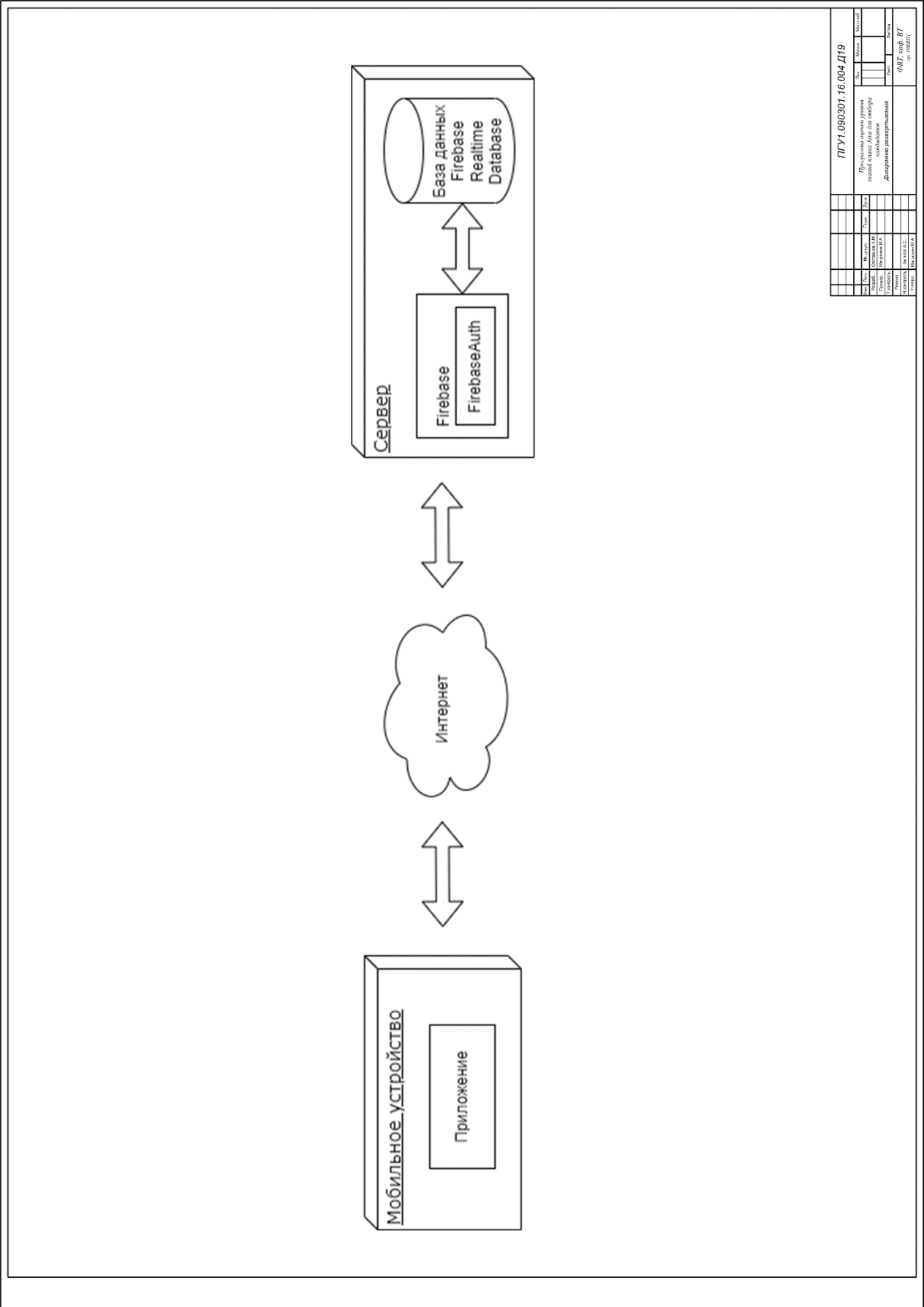
Приложение А.1 – *UML* – диаграмма прецедентов



Приложение А.2 – *UML* – диаграмма классов



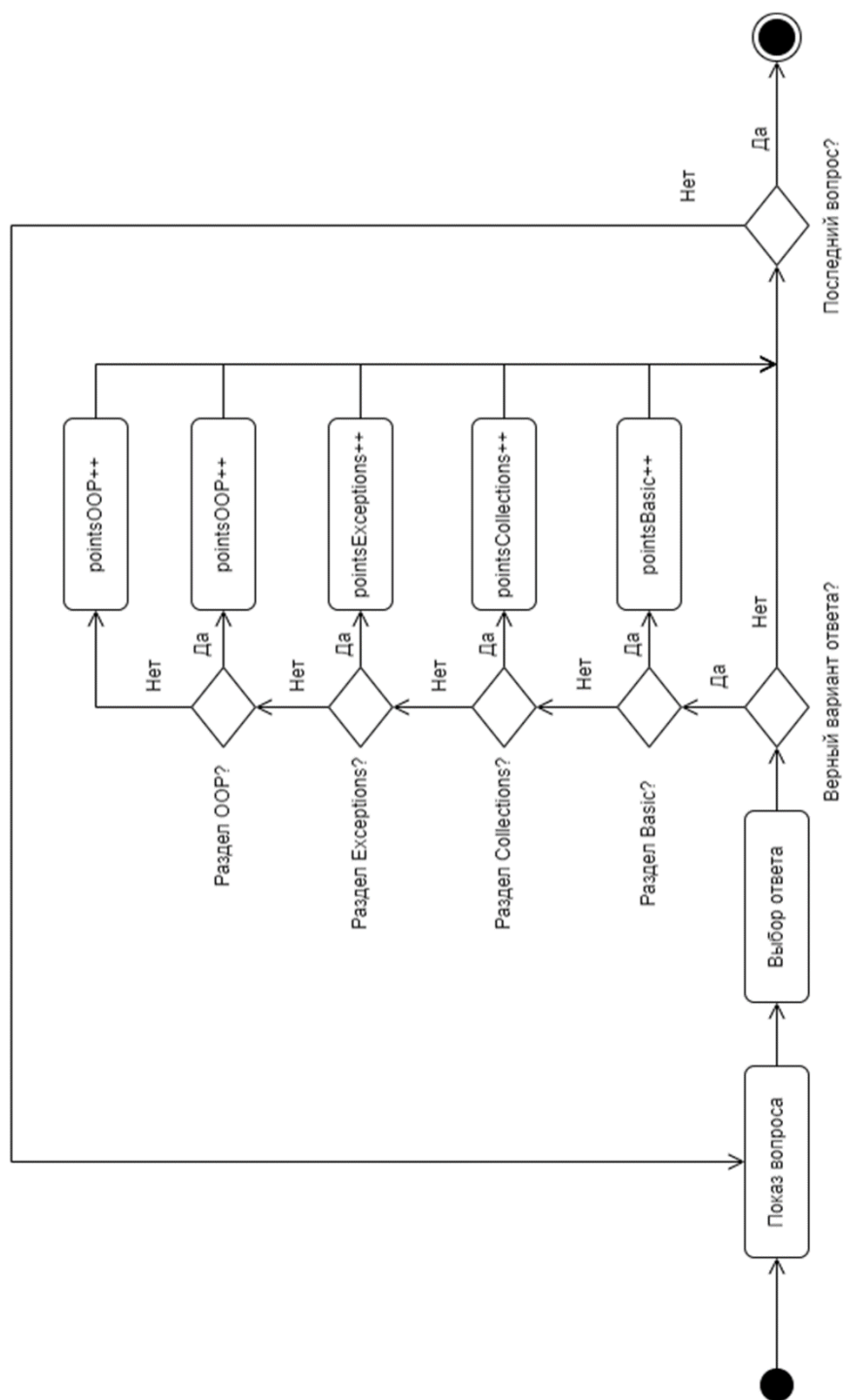
Приложение А.3 – UML – диаграмма развертывания



ПГУ1.090301.16.004 Д19									
Программа оценки уровня					Дата				
этикетки на документах					Дата				
Диагностика развертывания					Дата				
ФОРМ. код. ВТ					Ф. И. О.				
					Подпись				
					М.П.				

Изм.	Лист	№ докум.	Подпись	Дата

Приложение А.4 – *UML* – диаграмма деятельности опроса

[illegible]

Приложение Б

(Обязательное)

Программа оценки уровня знаний языка Java для отбора кандидатов

Листинг программы

					ПГУ1.090301.16.001 ПЗ	Лист
						54
Изм.	Лист	№ докум.	Подпись	Дата		

Файл «LoginFragment.java»

```

package com.example.app_test_user.fragments;

import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;

import com.example.app_test_user.AdminMainActivity;
import com.example.app_test_user.R;
import com.example.app_test_user.TestActivity;
import com.example.app_test_user.User;
import
com.example.app_test_user.databinding.FragmentLoginBinding;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.util.ArrayList;
import java.util.List;

public class LoginFragment extends Fragment {

    EditText edLog, edPass;
    Button btn_SingIn;
    private FragmentLoginBinding mBinding = null;
    private FirebaseAuth usersAuth;
    private FirebaseDatabase db;
    private DatabaseReference usersRef;
    private final String USER_K = "Users";

    User userTemp;

    String email;

```

```

private List<User> TempList;
private User tempUserrrrr = null;
private User noUser;
private int m = 0;

public LoginFragment() {
    super(R.layout.fragment_login);
}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup
container, Bundle savedInstanceState) {

    View view = inflater.inflate(R.layout.fragment_login,
container, false);
    edLog = view.findViewById(R.id.edLoginLog);
    edPass = view.findViewById(R.id.edPassLog);
    btn_SingIn = view.findViewById(R.id.btnSignIn);

    //      edLog.setText("89022050046");
    //      edPass.setText("AdminPass12345");

    btn_SingIn.setOnClickListener(new View.OnClickListener()
{
    @Override
    public void onClick(View view) {
        if (TextUtils.isEmpty(edLog.getText()))
            Toast.makeText(getActivity(), "Введите
логин", Toast.LENGTH_SHORT).show();
        else if (TextUtils.isEmpty(edPass.getText()))
            Toast.makeText(getActivity(), "Введите
пароль", Toast.LENGTH_SHORT).show();
        else {
            loadUserDB();
        }
    }
});

    return view;
}

@Override
public void onStart() {
    super.onStart();
    mBinding =
FragmentLoginBinding.inflate(getLayoutInflater());
    init();
}

```

```

//

getView().findViewById(R.id.btnGoToReg).setOnClickListener(new
View.OnClickListener() {
    @Override
    public void onClick(View view) {

getParentFragmentManager().beginTransaction().addToBackStack(nul
l).replace(R.id.start_main_container, new
RegisterFragment()).commit();
    }
});

}

public void init() {
    usersAuth = FirebaseAuth.getInstance();
    db =
FirebaseDatabase.getInstance("https://otstavnovdiploma-default-
rtdb.europe-west1.firebaseio.com");
    usersRef = db.getReference(USER_K);
}

private void loadUserDB() {

    String idUserNumber = edLog.getText().toString();

    TempList = new ArrayList<>();

    FirebaseDatabase
        .getInstance("https://otstavnovdiploma-default-
rtdb.europe-west1.firebaseio.com")
        .getReference()
        .child("Users")

        .child(idUserNumber).addListenerForSingleValueEvent(new
ValueEventListener() {
            @Override
            public void onDataChange(@NonNull
DataSnapshot snapshot) {
                tempUserrrrr =
snapshot.getValue(User.class);
                TempList.add(tempUserrrrr);

                User userTemp = TempList.get(1);
                email = userTemp.getEmail();
                String pass =
edPass.getText().toString();

```

```

usersAuth.signInWithEmailAndPassword(email, pass)
                                .addOnSuccessListener(new
OnSuccessListener<AuthResult>() {
                                @Override
                                public void
onSuccess(AuthResult authResult) {
                                if
                                (userTemp.email.equals("admin@mail.ru") &&
                                userTemp.pass.equals("AdminPass12345")) {
                                Intent intent1 = new
                                Intent(getActivity(), AdminMainActivity.class);
                                startActivity(intent1);
                                } else {
                                Intent intent = new
                                Intent(getActivity(), TestActivity.class);

                                intent.putExtra("userFName", userTemp.getFirst_name());
                                intent.putExtra("userSName", userTemp.getSecond_name());
                                intent.putExtra("userEmail", userTemp.getEmail());
                                intent.putExtra("userNumber", userTemp.getNumber());
                                intent.putExtra("userPass", userTemp.getPass());
                                intent.putExtra("pointsAll",
                                userTemp.user_test_result.getPointsAll());
                                intent.putExtra("pointsBasic",
                                userTemp.user_test_result.getPointsBasic());
                                intent.putExtra("pointsCol",
                                userTemp.user_test_result.getPointsCollections());
                                intent.putExtra("pointsExc",
                                userTemp.user_test_result.getPointsExceptions());
                                intent.putExtra("pointsOOP",
                                userTemp.user_test_result.getPointsOOP());
                                intent.putExtra("pointsOper",
                                userTemp.user_test_result.getPointsOperators());
                                startActivity(intent);
                                }

                                Toast.makeText(getActivity(), "Успешная авторизация",

```

```

Toast.LENGTH_SHORT).show();

    }

    }).addOnFailureListener(new

OnFailureListener() {

        @Override
        public void

onFailure(@NonNull Exception e) {

Toast.makeText(getActivity(), "Ошибка авторизации",
Toast.LENGTH_SHORT).show();

    }

    });

}

@Override
public void onCancelled(@NonNull
DatabaseError error) {

}

});

}

}

```

Файл «*RegistrFragment.java*»

```

package com.example.app_test_user.fragments;

import android.content.Intent;
import android.os.Bundle;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;

import android.text.TextUtils;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.example.app_test_user.R;
import com.example.app_test_user.TestActivity;
import com.example.app_test_user.TestResult;
import com.example.app_test_user.User;
import
com.example.app_test_user.databinding.FragmentRegisterBinding;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;

```

```

import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;

public class RegisterFragment extends Fragment {

    EditText edFirstName, edSecName, edEmail, edNumber, edPass;
    Button btnReg;

    private FirebaseAuth usersAuth;
    private FirebaseDatabase db;
    private DatabaseReference usersRef;
    private final String USER_K = "Users";

    private FragmentRegisterBinding mBinding = null;

    public RegisterFragment() {
        super(R.layout.fragment_register);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container, Bundle savedInstanceState) {

        View view = inflater.inflate(R.layout.fragment_register,
container, false);

        edFirstName = view.findViewById(R.id.edFirstNameReg);
        edSecName = view.findViewById(R.id.edSecNameReg);
        edEmail = view.findViewById(R.id.edEmailReg);
        edNumber = view.findViewById(R.id.edNumberReg);
        edPass = view.findViewById(R.id.edPassReg);
        btnReg = view.findViewById(R.id.btn_reg);

        btnReg.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(getActivity(),
TestActivity.class);

                String fname = edFirstName.getText().toString();
                String sname = edSecName.getText().toString();
                String email = edEmail.getText().toString();
                String number = edNumber.getText().toString();
                String pass = edPass.getText().toString();

                intent.putExtra("numKey", number);
            }
        });
    }
}

```

```

        if (TextUtils.isEmpty(fname)) {
            Toast.makeText(getActivity(), "Пожалуйста,
введите имя", Toast.LENGTH_SHORT).show();
        } else if (TextUtils.isEmpty(sname)) {
            Toast.makeText(getActivity(), "Пожалуйста,
введите фамилию", Toast.LENGTH_SHORT).show();
        } else if (TextUtils.isEmpty(email)) {
            Toast.makeText(getActivity(), "Пожалуйста,
введите почту", Toast.LENGTH_SHORT).show();
        } else if (TextUtils.isEmpty(number)) {
            Toast.makeText(getActivity(), "Пожалуйста,
введите номер телефона", Toast.LENGTH_SHORT).show();
        } else if (TextUtils.isEmpty(sname)) {
            Toast.makeText(getActivity(), "Пожалуйста,
введите фамилию", Toast.LENGTH_SHORT).show();
        } else if (TextUtils.isEmpty(pass)) {
            Toast.makeText(getActivity(), "Пожалуйста,
введите пароль", Toast.LENGTH_SHORT).show();
        } else if (pass.length() < 6) {
            Toast.makeText(getActivity(), "Минимальная
длина пароля - 6 символов", Toast.LENGTH_SHORT).show();
        } else {

usersAuth.createUserWithEmailAndPassword(email, pass)
            .addOnSuccessListener(new
OnSuccessListener<AuthResult>() {
                @Override
                public void onSuccess(AuthResult
authResult) {

                    User newUser = new
User(fname, sname, email, number, pass);
                    newUser.user_test_result =
new TestResult();

usersRef.child(newUser.getNumber()).setValue(newUser)

.addOnSuccessListener(new OnSuccessListener<Void>() {
                            @Override
                            public void
onSuccess(Void unused) {

intent.putExtra("userFName", newUser.getFirst_name());

intent.putExtra("userSName", newUser.getSecond_name());

intent.putExtra("userEmail", newUser.getEmail());

intent.putExtra("userNumber", newUser.getNumber());

intent.putExtra("userPass", newUser.getPass());

```

```

Toast.makeText(getActivity(), "Регистрация прошла успешно",
Toast.LENGTH_SHORT).show();

startActivity(intent);

    }

    });

    }

    }).addOnFailureListener(new

OnFailureListener() {

    @Override
    public void onFailure(@NonNull

Exception e) {

    }

    });

    }

    });

    return view;

}

@Override
public void onStart() {
    super.onStart();
    init();
}

public void init() {
    usersAuth = FirebaseAuth.getInstance();
    db =
FirebaseDatabase.getInstance("https://otstavnovdiploma-default-
rtbd.europe-west1.firebaseio.com");
    usersRef = db.getReference(USER_K);
}

}

```

Файл «*TestingFragment.java*»

```

package com.example.app_test_user.fragments;

import android.os.Bundle;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;

import android.provider.ContactsContract;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.TextView;

```

```

import android.widget.Toast;

import com.example.app_test_user.Question;
import com.example.app_test_user.R;
import com.example.app_test_user.TestResult;
import com.example.app_test_user.User;
import com.google.android.gms.common.data.DataBuffer;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class TestingFragment extends Fragment {

    private TextView txtView;
    private Button ans1, ans2, ans3, ans4;
    private Question q1;
    private Question tempQuestion = null;
    private List<Question> questionList;
    private Question questTemp;
    public int iter = 0;
    private String correct_ans;
    private String section;
    private int pointOPP, pointBasic, pointCollections,
pointExceptions, pointOperators;
    private TestResult userTestResult;
    private User user;

    public TestingFragment(User loadUser) {
        this.user = loadUser;
    }

    @Override
    public void onStart() {
        super.onStart();

        initQuest();
        initView();
        questTemp = questionList.get(0);
        txtView.setText(questTemp.getText());
        ans1.setText(questTemp.getAnswer1());
        ans2.setText(questTemp.getAnswer2());
        ans3.setText(questTemp.getAnswer3());
        ans4.setText(questTemp.getAnswer4());
        correct_ans = questTemp.getCorrect_answer();

```

```

        section = questTemp.getSection();

        ans1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if
(correct_ans.equals(ans1.getText().toString())) {
                    switch (section) {
                        case "OOP":
                            pointOPP++;
                            break;
                        case "Collections":
                            pointCollections++;
                            break;
                        case "Basic":
                            pointBasic++;
                            break;
                        case "Exceptions":
                            pointExceptions++;
                            break;
                        case "Operators":
                            pointOperators++;
                            break;
                    }
                }
                iter++;
                if (iter >= 25) {
                    userTestResult = new TestResult(pointBasic,
pointCollections, pointExceptions, pointOPP, pointOperators);
                    loadResults();

getParentFragmentManager().beginTransaction().addToBackStack(nul
l).replace(R.id.home_main_container, new
TestResultFragment(userTestResult)).commit();
                }

                if (iter < 25) {
                    questTemp = questionList.get(iter);
                    txtView.setText(questTemp.getText());
                    ans1.setText(questTemp.getAnswer1());
                    ans2.setText(questTemp.getAnswer2());
                    ans3.setText(questTemp.getAnswer3());
                    ans4.setText(questTemp.getAnswer4());
                    correct_ans = questTemp.getCorrect_answer();
                    section = questTemp.getSection();
                }
            }
        });

        ans2.setOnClickListener(new View.OnClickListener() {
            @Override

```

```

        public void onClick(View view) {
            if
(correct_ans.equals(ans2.getText().toString())) {
                switch (section) {
                    case "OOP":
                        pointOPP++;
                        break;
                    case "Collections":
                        pointCollections++;
                        break;
                    case "Basic":
                        pointBasic++;
                        break;
                    case "Exceptions":
                        pointExceptions++;
                        break;
                    case "Operators":
                        pointOperators++;
                        break;
                }
            }
            iter++;
            if (iter >= 25) {
                userTestResult = new TestResult(pointBasic,
pointCollections, pointExceptions, pointOPP, pointOperators);
                loadResults();

getParentFragmentManager().beginTransaction().addToBackStack(nul
l).replace(R.id.home_main_container, new
TestResultFragment(userTestResult)).commit();
            }

            if (iter < 25) {
                questTemp = questionList.get(iter);
                txtView.setText(questTemp.getText());
                ans1.setText(questTemp.getAnswer1());
                ans2.setText(questTemp.getAnswer2());
                ans3.setText(questTemp.getAnswer3());
                ans4.setText(questTemp.getAnswer4());
                correct_ans = questTemp.getCorrect_answer();
                section = questTemp.getSection();
            }
        }
    });

    ans3.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            if
(correct_ans.equals(ans3.getText().toString())) {
                switch (section) {

```

```

        case "OOP":
            pointOPP++;
            break;
        case "Collections":
            pointCollections++;
            break;
        case "Basic":
            pointBasic++;
            break;
        case "Exceptions":
            pointExceptions++;
            break;
        case "Operators":
            pointOperators++;
            break;
    }
    }
    iter++;
    if (iter >= 25) {
        userTestResult = new TestResult(pointBasic,
pointCollections, pointExceptions, pointOPP, pointOperators);
        loadResults();

getParentFragmentManager().beginTransaction().addToBackStack(nul
l).replace(R.id.home_main_container, new
TestResultFragment(userTestResult)).commit();
    }

    if (iter < 25) {
        questTemp = questionList.get(iter);
        txtView.setText(questTemp.getText());
        ans1.setText(questTemp.getAnswer1());
        ans2.setText(questTemp.getAnswer2());
        ans3.setText(questTemp.getAnswer3());
        ans4.setText(questTemp.getAnswer4());
        correct_ans = questTemp.getCorrect_answer();
        section = questTemp.getSection();
    }
    }
});

ans4.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if
(correct_ans.equals(ans4.getText().toString())) {
            switch (section) {
                case "OOP":
                    pointOPP++;
                    break;
                case "Collections":

```

```

        pointCollections++;
        break;
    case "Basic":
        pointBasic++;
        break;
    case "Exceptions":
        pointExceptions++;
        break;
    case "Operators":
        pointOperators++;
        break;
    }
}
iter++;
if (iter >= 25) {
    userTestResult = new TestResult(pointBasic,
pointCollections, pointExceptions, pointOPP, pointOperators);
    loadResults();

getParentFragmentManager().beginTransaction().addToBackStack(nul
l).replace(R.id.home_main_container, new
TestResultFragment(userTestResult)).commit();
}

    if (iter < 25) {
        questTemp = questionList.get(iter);
        txtView.setText(questTemp.getText());
        ans1.setText(questTemp.getAnswer1());
        ans2.setText(questTemp.getAnswer2());
        ans3.setText(questTemp.getAnswer3());
        ans4.setText(questTemp.getAnswer4());
        correct_ans = questTemp.getCorrect_answer();
        section = questTemp.getSection();
    }
}
});

}

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup
container, Bundle savedInstanceState) {

    View view = inflater.inflate(R.layout.fragment_testing,

```

```

container, false);
    return inflater.inflate(R.layout.fragment_testing,
container, false);
}
public void initQuest() {
    questionList = new ArrayList<>();

    String idSectionBasic = "Basic";
    String idSectionCollections = "Collections";
    String idSectionExceptions = "Exceptions";
    String idSectionOOP = "OOP";
    String idSectionOperators = "Operators";

    String id_temp = "";
    String sec_temp = "";

    for (int i = 2; i <= 25; i++) {
        sec_temp = idSectionBasic;
        id_temp = sec_temp + "-" + i;
        int temp;

        if (i >= 6 && i <= 10) {
            temp = i - 5;
            sec_temp = idSectionCollections;
            id_temp = sec_temp + "-" + temp;
        }

        if (i >= 11 && i <= 15) {
            temp = i - 10;
            sec_temp = idSectionExceptions;
            id_temp = sec_temp + "-" + temp;
        }
        if (i >= 16 && i <= 20) {
            temp = i - 15;
            sec_temp = idSectionOOP;
            id_temp = sec_temp + "-" + temp;
        }
        if (i >= 21) {
            temp = i - 20;
            sec_temp = idSectionOperators;
            id_temp = sec_temp + "-" + temp;
        }
        FirebaseDatabase
            .getInstance("https://otstavnovdiploma-
default-rtdb.europe-west1.firebaseio.com/")
            .getReference()
            .child("Questions")
            .child(sec_temp)

            .child(id_temp).addListenerForSingleValueEvent(new
ValueEventListener() {

```

```

        @Override
        public void onDataChange(@NonNull
DataSnapshot snapshot) {
            tempQuestion =
snapshot.getValue(Question.class);
            questionList.add(tempQuestion);
        }

        @Override
        public void onCancelled(@NonNull
DatabaseError error) {
        }
    });
}

q1 = new Question("Basic",
    "Basic-1",
    "... - шаблон, определяющий переменные и методы,
общие для всех его объектов определённого типа. Заполните
пропуск",
    "Объект",
    "Класс",
    "Метод",
    "Интерфейс",
    "Класс");
questionList.add(q1);
}

private void initView() {
    txtView = (TextView)
getActivity().findViewById(R.id.txt_TestQuest);
    ans1 = (Button)
getActivity().findViewById(R.id.btn_answer1);
    ans2 = (Button)
getActivity().findViewById(R.id.btn_answer2);
    ans3 = (Button)
getActivity().findViewById(R.id.btn_answer3);
    ans4 = (Button)
getActivity().findViewById(R.id.btn_answer4);
}

private void loadResults() {
    User newUser = user;
    newUser.user_test_result = userTestResult;
    FirebaseDatabase dataBase =
FirebaseDatabase.getInstance("https://otstavnovdiploma-default-
rtdb.europe-west1.firebaseio.com");
    DatabaseReference usersRef =
dataBase.getReference("Users");
    usersRef.child(newUser.getNumber()).setValue(newUser)

```

```

        .addOnSuccessListener(new
        OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void unused) {

                Toast.makeText(getActivity(), "Загрузка
результатов прошла успешно", Toast.LENGTH_SHORT).show();

            }
        });
    }
}

```

Файл «*TestActivity.java*»

```

package com.example.app_test_user;

import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

import com.example.app_test_user.databinding.TestLayoutBinding;
import com.example.app_test_user.fragments.TestResultFragment;
import com.example.app_test_user.fragments.TestingFragment;
import com.example.app_test_user.fragments.UserProfileFragment;

public class TestActivity extends AppCompatActivity {

    private TestLayoutBinding mBinding = null;
    private Button startTest;
    private User curUser;
    boolean btn_prof = false;
    boolean btn_test = false;
    boolean btn_result = false;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        initFields();
        mBinding =
TestLayoutBinding.inflate(getLayoutInflater());
        setContentView(mBinding.getRoot());
    }
}

```

```

mBinding.bottomNavigationView.setOnItemSelectedListener(item ->
{
    switch (item.getItemId()) {
        case R.id.profile:
            if (!btn_prof) {

getSupportFragmentManager().beginTransaction()

.replace(R.id.home_main_container, new
UserProfileFragment(curUser))

                .commit();
                btn_prof = true;
                btn_test = false;
                btn_result = false;
            }
            break;

        case R.id.test:

            if (!btn_test) {

getSupportFragmentManager().beginTransaction()

.replace(R.id.home_main_container, new TestingFragment(curUser))

                .commit();
                btn_prof = false;
                btn_test = true;
                btn_result = false;

            }
            break;
        case R.id.results:
            if (curUser.user_test_result.pointsAll > 0)
{
                if (!btn_result) {
getSupportFragmentManager().beginTransaction()

.replace(R.id.home_main_container, new
TestResultFragment(curUser.getUser_test_result()))

                    .commit();
                    btn_prof = false;
                    btn_test = false;
                    btn_result = true;
                }
            } else {
                Toast.makeText(this, "Для просмотра
результата пройдите тест", Toast.LENGTH_SHORT).show();
            }
            break;
        case R.id.exit:

```

```

        Intent intent = new Intent(this,
StartActivity.class);
        startActivity(intent);
        break;
    }
    return true;
});

}

@Override
protected void onStart() {
    super.onStart();
}

private void initFields() {

    Bundle arguments = getIntent().getExtras();

    curUser = new User();
    curUser.first_name =
arguments.get("userFName").toString();
    curUser.second_name =
arguments.get("userSName").toString();
    curUser.email = arguments.get("userEmail").toString();
    curUser.number = arguments.get("userNumber").toString();
    curUser.pass = arguments.get("userPass").toString();
    curUser.user_test_result = new TestResult();
    curUser.user_test_result.pointsAll =
arguments.getInt("pointsAll");
    curUser.user_test_result.pointsBasic =
arguments.getInt("pointsBasic");
    curUser.user_test_result.pointsCollections =
arguments.getInt("pointsCol");
    curUser.user_test_result.pointsExceptions =
arguments.getInt("pointsExc");
    curUser.user_test_result.pointsOOP =
arguments.getInt("pointsOOP");
    curUser.user_test_result.pointsOperators =
arguments.getInt("pointsOper");

    }
}

```

Файл «User.java»

```

package com.example.app_test_user;

public class User {

    public String  first_name, second_name, email, number, pass;

```

```

public TestResult user_test_result;
public User() {
}

public User(String first_name, String sec_name, String
email, String number, String pass) {
    this.first_name = first_name;
    this.second_name = sec_name;
    this.email = email;
    this.number = number;
    this.pass = pass;
}

public TestResult getUser_test_result() {
    return user_test_result;
}

public void setUser_test_result(TestResult user_test_result)
{
    this.user_test_result = user_test_result;
}

public String getFirst_name() {
    return first_name;
}

public String getSecond_name() {
    return second_name;
}

public String getEmail() {
    return email;
}

public String getNumber() {
    return number;
}

public String getPass() {
    return pass;
}

public void setFirst_name(String first_name) {
    this.first_name = first_name;
}

public void setSecond_name(String second_name) {
    this.second_name = second_name;
}

public void setEmail(String email) {

```

```

        this.email = email;
    }

    public void setNumber(String number) {
        this.number = number;
    }

    public void setPass(String pass) {
        this.pass = pass;
    }
}

```

Файл «*UserProfileFragment.java*»

```

package com.example.app_test_user.fragments;

import android.os.Bundle;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.example.app_test_user.R;
import com.example.app_test_user.User;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;

public class UserProfileFragment extends Fragment {

    private User currentUser;
    private Button btn_fname, btn_sname, btn_pass, btn_save;
    private EditText
edFirstNameProfile,edSecNameProfile,edEmailProfile,edNumberProfi
le,edPassProfile;

    public UserProfileFragment(User user) {
        currentUser = user;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {

```

```

        super.onCreate(savedInstanceState);
    }

    @Override
    public void onStart() {
        super.onStart();
        init();

        edFirstNameProfile.setText(currentUser.getFirst_name());
        edSecNameProfile.setText(currentUser.getSecond_name());
        edEmailProfile.setText(currentUser.getEmail());
        edNumberProfile.setText(currentUser.getNumber());
        edPassProfile.setText(currentUser.getPass());

        btn_fname.setOnClickListener(new View.OnClickListener()
        {
            @Override
            public void onClick(View view) {
                edFirstNameProfile.setEnabled(true);
            }
        });
        btn_sname.setOnClickListener(new View.OnClickListener()
        {
            @Override
            public void onClick(View view) {
                edSecNameProfile.setEnabled(true);
            }
        });
        btn_pass.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                edPassProfile.setEnabled(true);
            }
        });
        btn_save.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                edFirstNameProfile.setEnabled(false);
                edSecNameProfile.setEnabled(false);
                edPassProfile.setEnabled(false);

                currentUser.first_name =
edFirstNameProfile.getText().toString();
                currentUser.second_name =
edSecNameProfile.getText().toString();
                currentUser.pass =
edPassProfile.getText().toString();

                saveData();
            }
        });
    }
}

```

```

    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container,
                                Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_user_profile,
container, false);
    }
    private void init(){
        edFirstNameProfile = (EditText)
getActivity().findViewById(R.id.edFirstNameProfile);
        edSecNameProfile = (EditText)
getActivity().findViewById(R.id.edSecondNameProfile);
        edEmailProfile = (EditText)
getActivity().findViewById(R.id.edEmailProfile);
        edNumberProfile = (EditText)
getActivity().findViewById(R.id.edNumberProfile);
        edPassProfile = (EditText)
getActivity().findViewById(R.id.edPassProfile);
        btn_fname = (Button)
getActivity().findViewById(R.id.btn_fname_change);
        btn_sname = (Button)
getActivity().findViewById(R.id.btn_sname_change);
        btn_pass = (Button)
getActivity().findViewById(R.id.btn_pass_change);
        btn_save = (Button)
getActivity().findViewById(R.id.btn_save_user_info);

    }

    private void saveData() {
        User newUser = currentUser;
        FirebaseDatabase dataBase =
FirebaseDatabase.getInstance("https://otstavnovdiploma-default-
rttdb.europe-west1.firebaseio.com");
        DatabaseReference usersRef =
dataBase.getReference("Users");
        usersRef.child(newUser.getNumber()).setValue(newUser)
            .addOnSuccessListener(new
OnSuccessListener<Void>() {
                @Override
                public void onSuccess(Void unused) {

                    Toast.makeText(getActivity(), "Изменения
сохранены", Toast.LENGTH_SHORT).show();
                }
            });
        FirebaseUser user =
FirebaseAuth.getInstance().getCurrentUser();
    }

```

```

        user.updatePassword(currentUser.getPass());
    }
}

```

Файл «*Question.java*»

```

package com.example.app_test_user;

public class Question {

    public String section, id, text, answer1, answer2, answer3,
    answer4, correct_answer;

    public Question() {
        this.section = null;
        this.id = null;
        this.text = null;
        this.answer1 = null;
        this.answer2 = null;
        this.answer3 = null;
        this.answer4 = null;
        this.correct_answer = null;
    }

    public Question(String section, String id, String text,
    String answer1, String answer2, String answer3, String answer4,
    String correct_answer) {
        this.section = section;
        this.id = id;
        this.text = text;
        this.answer1 = answer1;
        this.answer2 = answer2;
        this.answer3 = answer3;
        this.answer4 = answer4;
        this.correct_answer = correct_answer;
    }

    public void setSection(String section) {
        this.section = section;
    }

    public void setId(String id) {
        this.id = id;
    }

    public void setText(String text) {
        this.text = text;
    }

    public void setAnswer1(String answer1) {
        this.answer1 = answer1;
    }
}

```

```

public void setAnswer2(String answer2) {
    this.answer2 = answer2;
}

public void setAnswer3(String answer3) {
    this.answer3 = answer3;
}

public void setAnswer4(String answer4) {
    this.answer4 = answer4;
}

public void setCorrect_answer(String correct_answer) {
    this.correct_answer = correct_answer;
}

public String getSection() {
    return section;
}

public String getId() {
    return id;
}

public String getText() {
    return text;
}

public String getAnswer1() {
    return answer1;
}

public String getAnswer2() {
    return answer2;
}

public String getAnswer3() {
    return answer3;
}

public String getAnswer4() {
    return answer4;
}

public String getCorrect_answer() {
    return correct_answer;
}
}

```

Приложение В

(Обязательное)

Программа оценки уровня знаний языка Java для отбора кандидатов

Презентация

					ПГУ1.090301.16.001 ПЗ	Лист
						79
Изм.	Лист	№ докум.	Подпись	Дата		

Тема работы: «Программа оценки уровня знаний языка Java для отбора кандидатов»

Автор работы студент группы 19ВВП1 Отставнов А.М.
Научный руководитель д.т.н., профессор, зав. каф. ВТ ПГУ Митрохин М.А.

Пенза 2023

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.16.001 ПЗ

Лист
80

Цель и объект работы

Цель работы:

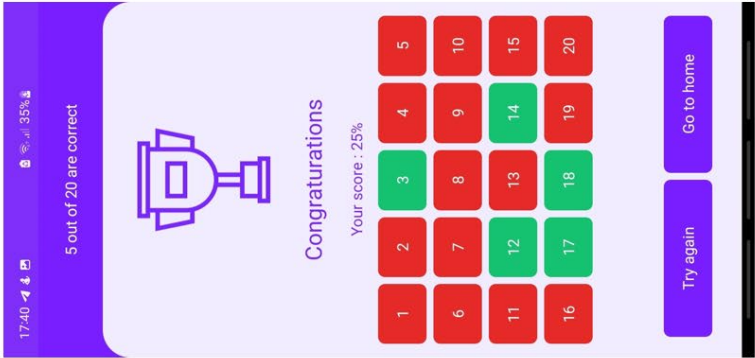
разработка Android-приложения, позволяющего определить уровень знания языка Java у кандидатов посредством тестирования.

Объект работы:

приложение, обладающее следующим пользовательским функционалом:

- регистрация;
- авторизация;
- редактирование личных данных;
- прохождение тестирования;
- просмотр результатов тестирования.

Аналоги



Аналоги

17:55

17:55

44%

Quick Mockup

Mark20

Time18.0 min

Total Question10

Select number of questions:

Set A, 2 Mark Question

10

Start

17:56

17:56

45%

Question: 1/10

Mark: 2

Finish Mock

keyword is used to refer to the current object.

☐ new

☐ super

☐ this

☐ volatile

Previous

Next

Suggested Time: 108 seconds

17:56

17:56

45%

100.0 %

0.0 %

Correct

Wrong

Mark Obtained0.0

Time Elapsed0:5 min

Result : Failed

See Answers

Аналоги

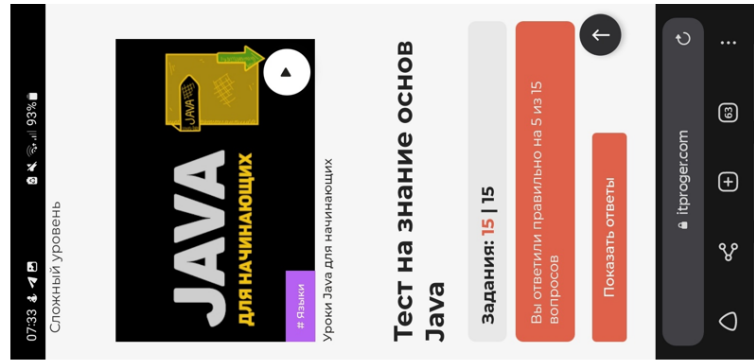
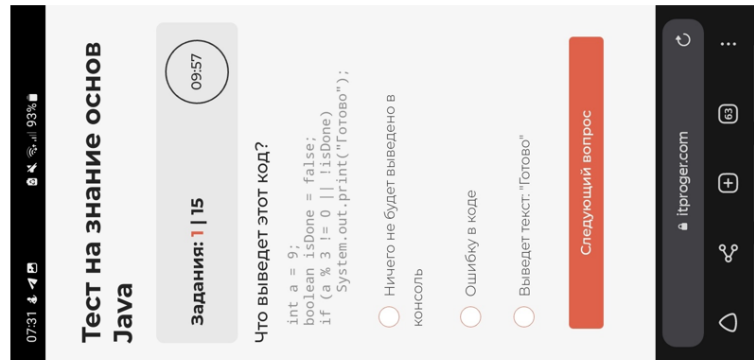
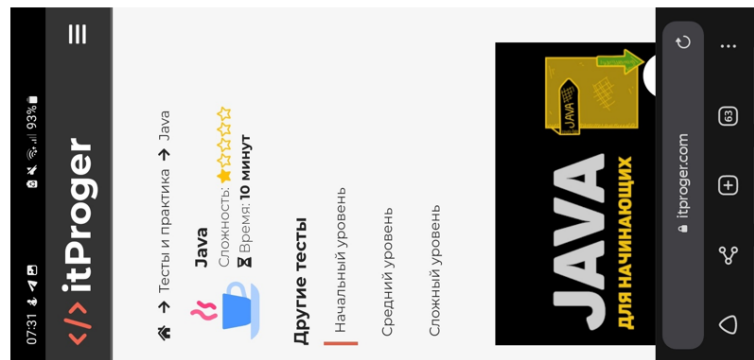
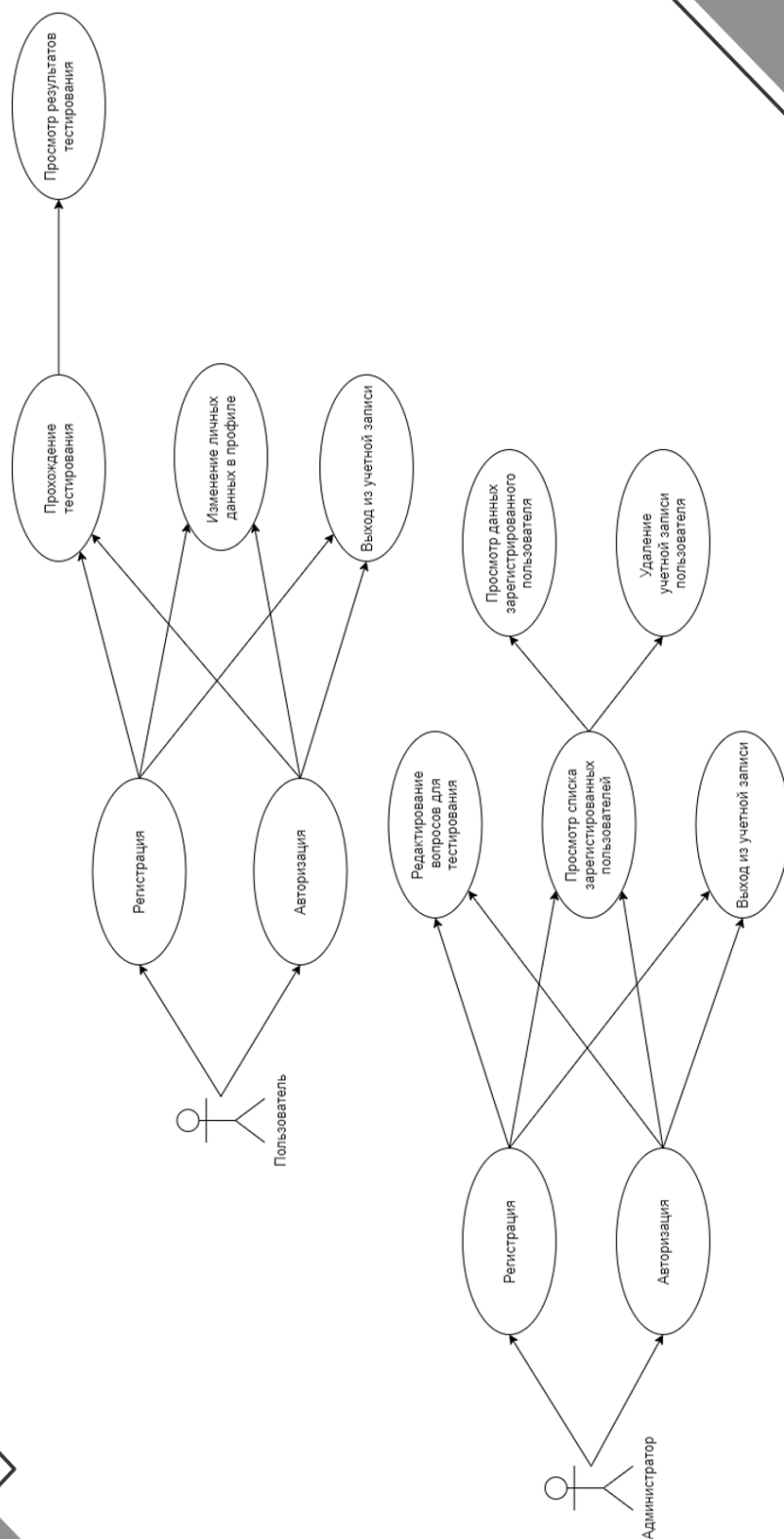


Диаграмма прецедентов



Android Studio

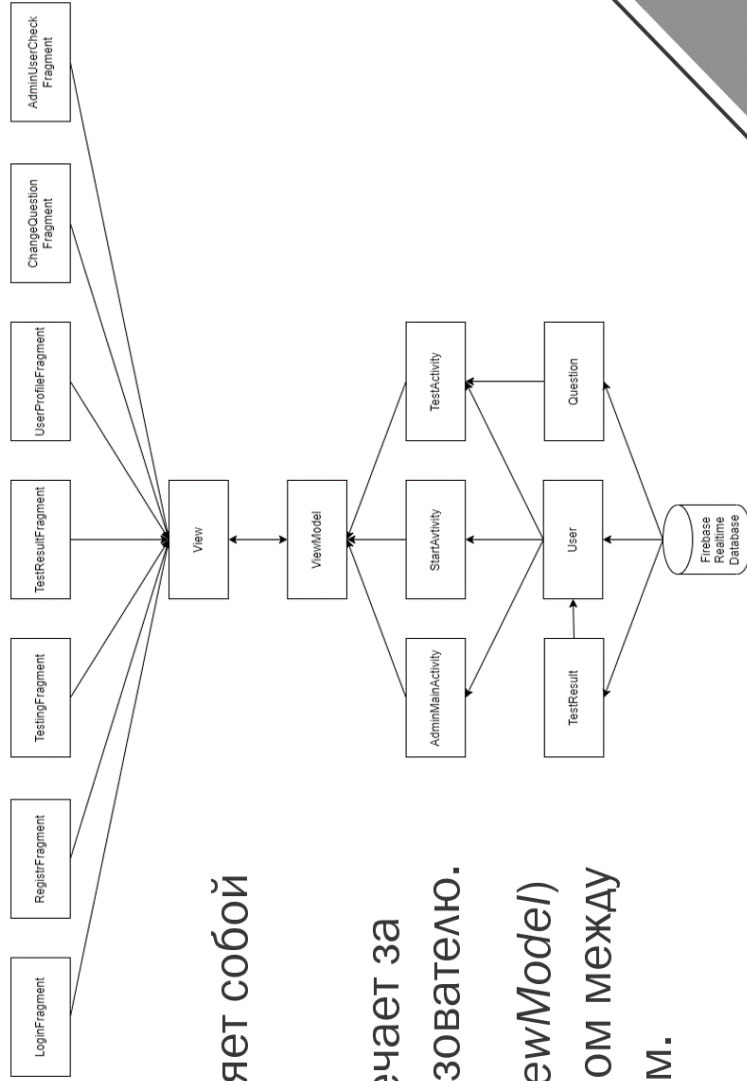


Стек технологий



Изм.	Лист	№ докум.	Подпись	Дата

Архитектура приложения



Модель (*Model*) представляет собой слой данных приложения.

Представление (*View*) отвечает за отображение данных пользователю.

Модель представления (*ViewModel*) является связующим звеном между моделью и представлением.

Диаграмма классов

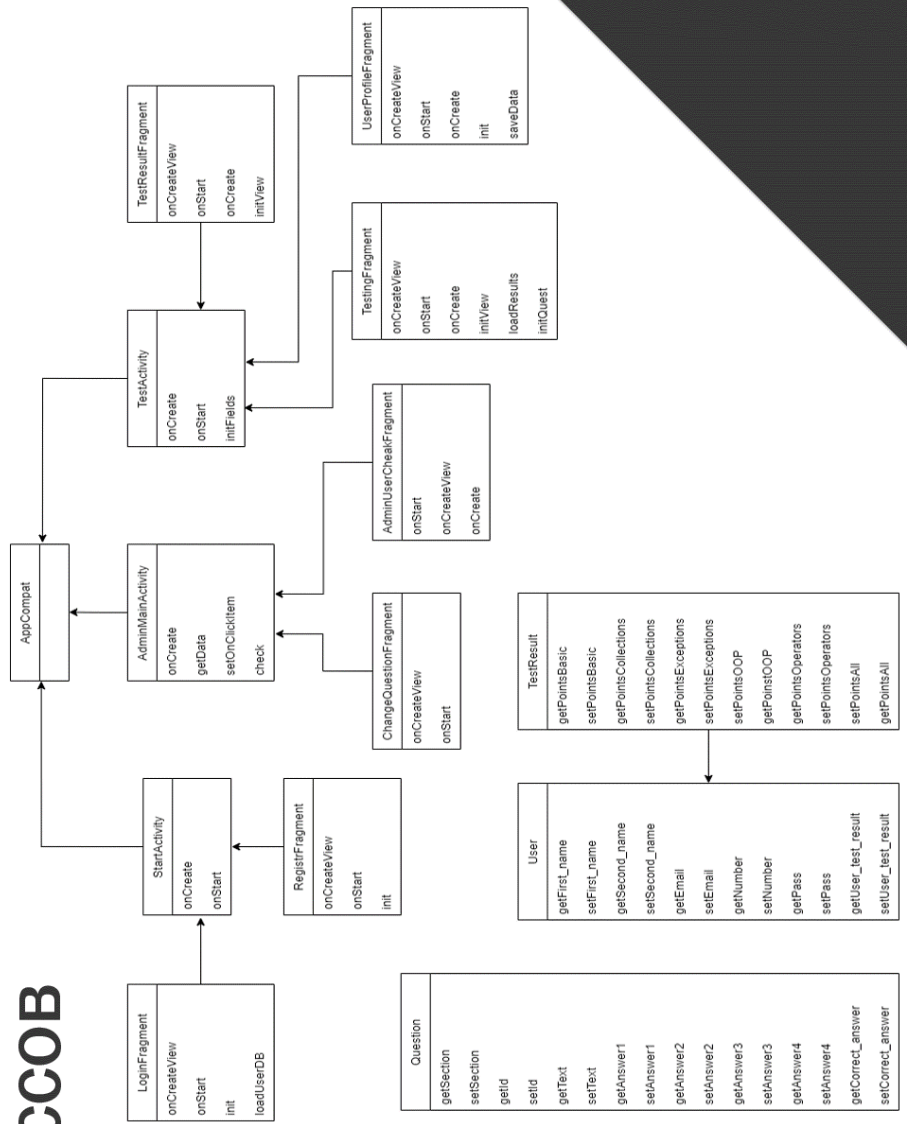
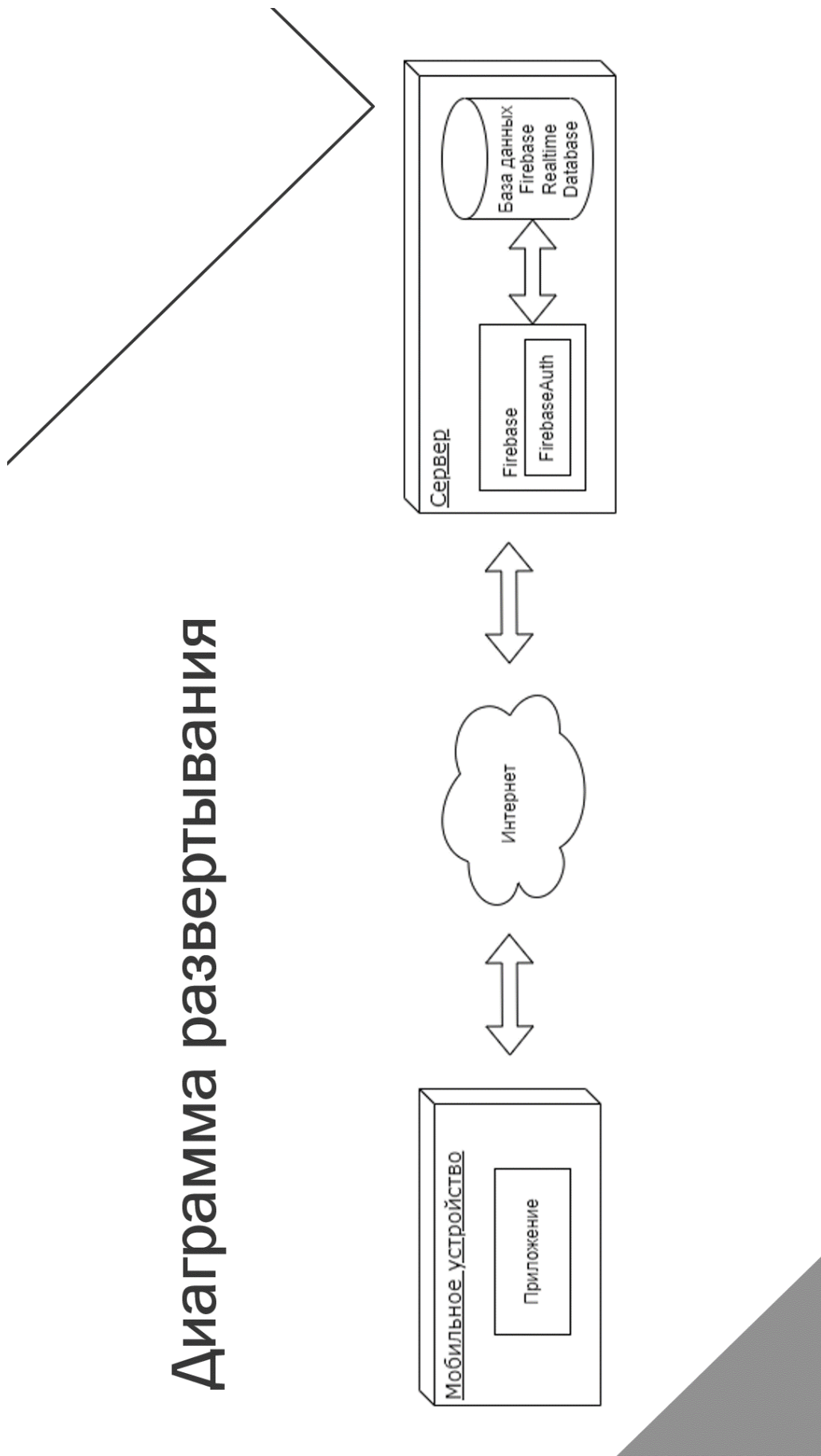
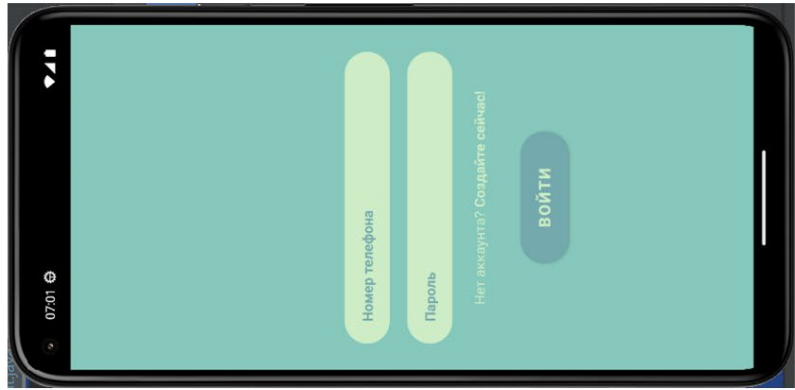
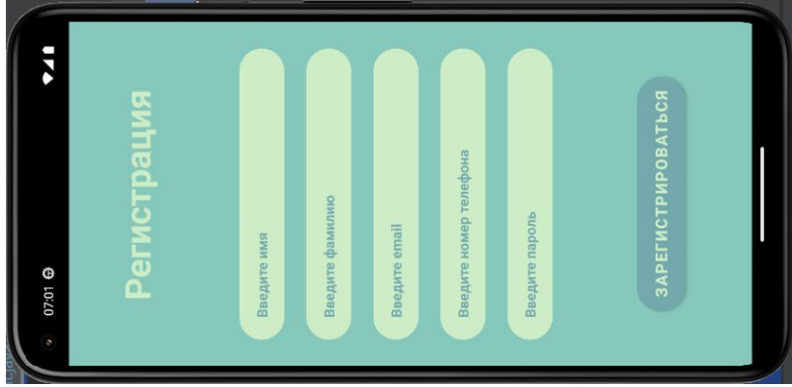


Диаграмма развертывания



Интерфейс приложения



Интерфейс приложения

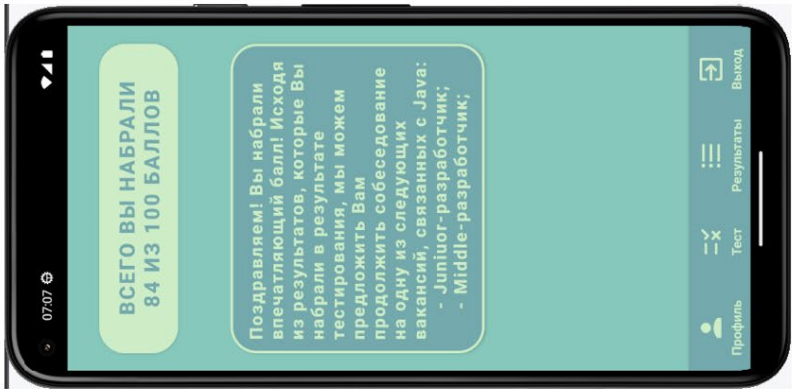
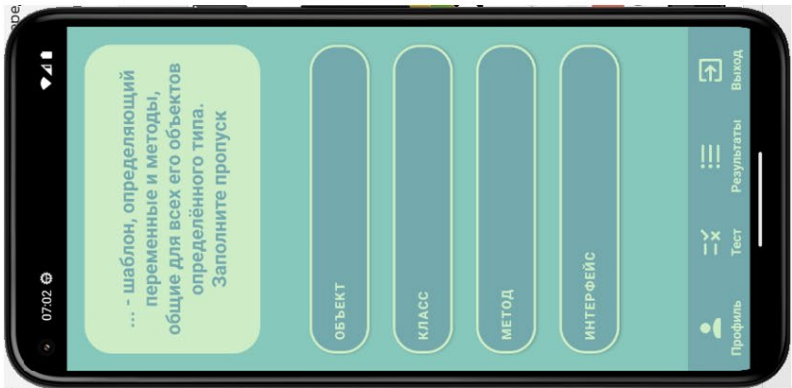
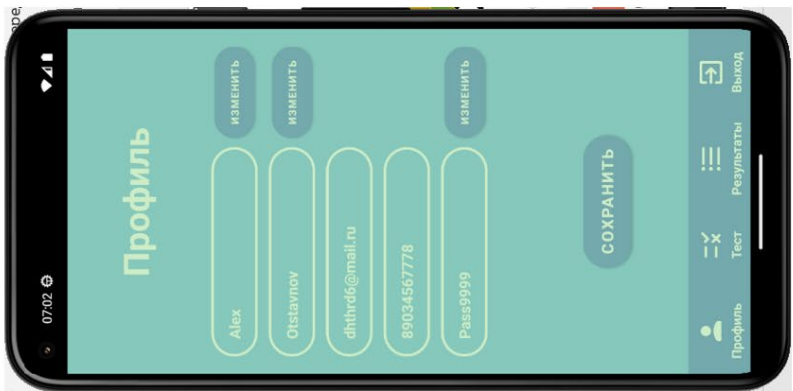


Диаграмма деятельности опроса

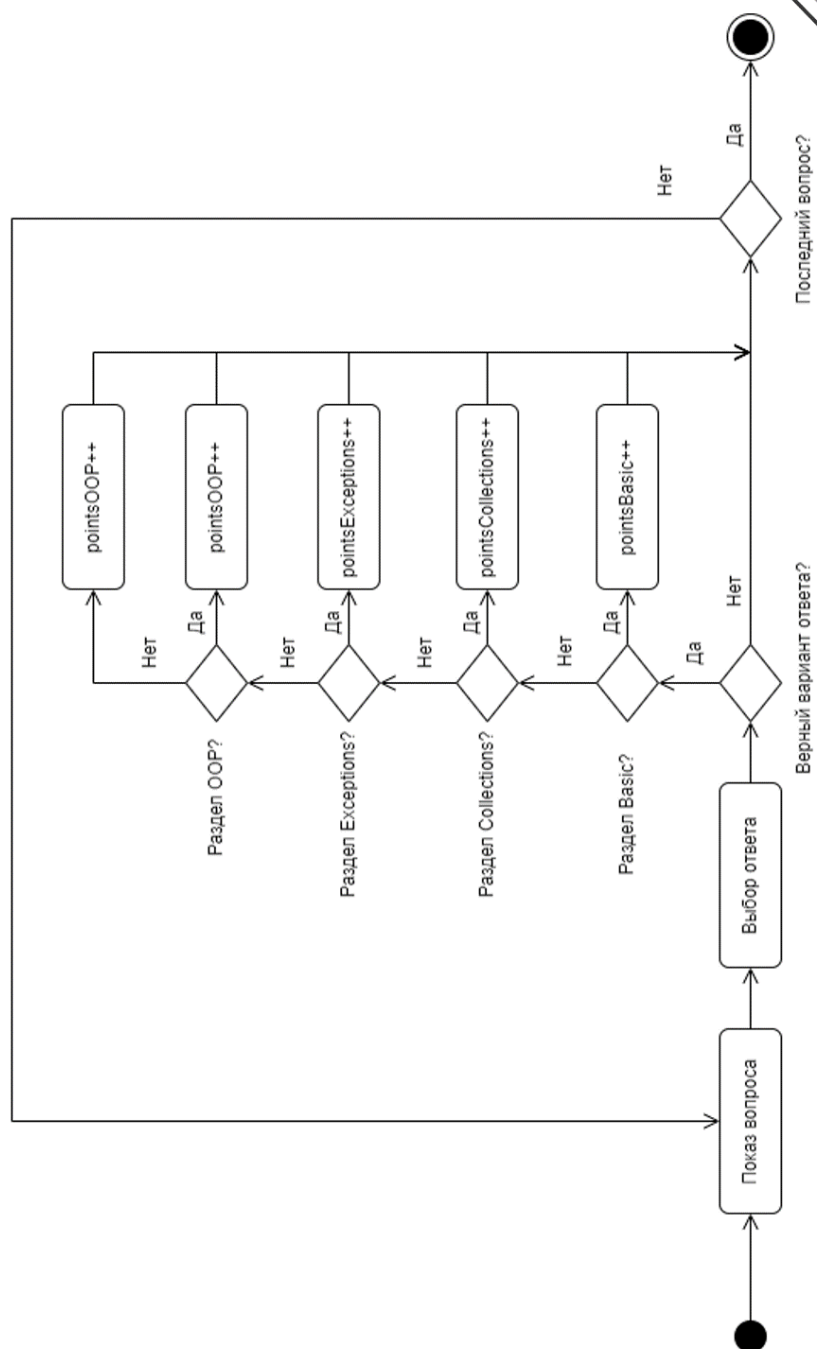
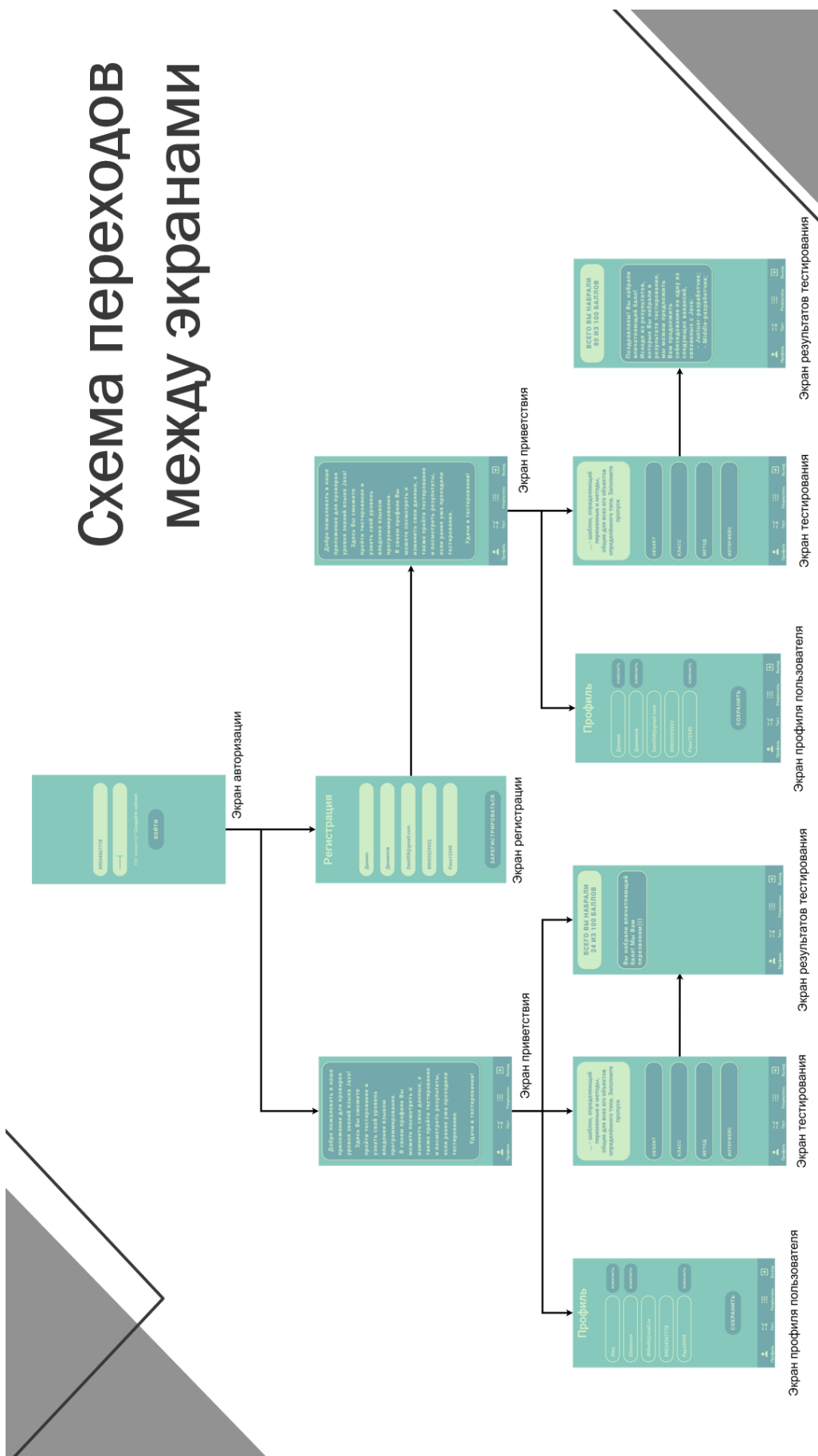
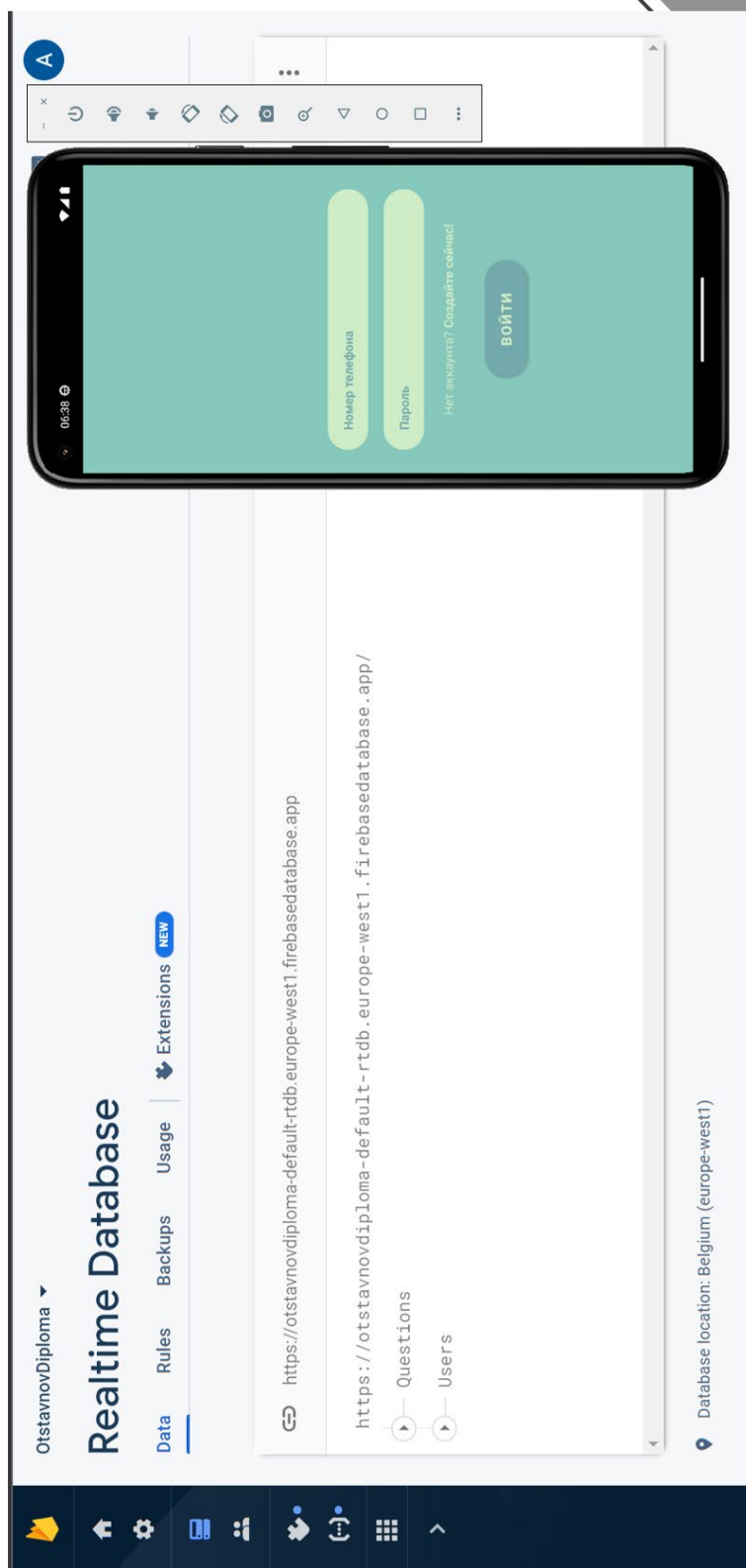


Схема переходов между экранами



Демонстрация



Заключение

В результате выполнения данной выпускной квалификационной работы было создано мобильное приложение с удобным и понятным пользовательским интерфейсом. Созданный проект дает возможность получать информацию об уровне знаний языка Java кандидатов посредством тестирования.

Разработка приложения оценки уровня знаний языка Java для отбора кандидатов определено может повысить уровень оптимизации подбора кадров. Результаты работы показали, что приложение успешно выполняет поставленные задачи и может быть использовано в сфере IT-рекрутинга.

Спасибо за внимание!

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.16.001 ПЗ

Лист

96