

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

Кафедра

вычислительной техники

«Вычислительная техника»

Специальность: 09.05.01 «Применение и эксплуатация
автоматизированных систем специального
назначения»
Специализация: «Эксплуатация вычислительных машин, комплексов,
систем и сетей специального назначения»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
СПЕЦИАЛИСТА

на тему

ANDROID-ПРИЛОЖЕНИЕ ПО ВЫБОРУ
ДОЛЖНОСТЕЙ НА ОСНОВЕ ХАРАКТЕРИСТИКИ
ЛИЧНОСТИ И ПРОФЕССИОНАЛЬНЫХ
НАВЫКОВ

Студент

 20.06.22
(подпись, дата)

Кирячек В.А.
(ФИО полностью)

Руководитель

 20.06.22
(подпись, дата)

Митрохин М.А.
(фамилия, инициалы)

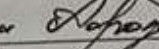
Нормоконтролёр

 20.06.22
(подпись, дата)

Бычков А.С.
(фамилия, инициалы)

Рецензент

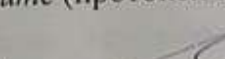
кач. отдела
ИТ/ИТ, Информатика
(должность, место работы)

 22.06.2022
(подпись, дата)

Афонин А.Ю.
(фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 02.06.22 г. № 13)

Заведующий кафедрой

 (подпись)

М.А. Митрохин
(фамилия, инициалы)

Работа защищена с отметкой хорошо (протокол заседания ГЭК от

24 июня 2022 № 12)

Секретарь ГЭК



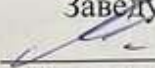
Е.И. Гурин

Пенза 2022

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

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
 М.А. Митрохин
«26» апреля 2022 г.

**ЗАДАНИЕ
НА ДИПЛОМНЫЙ ПРОЕКТ**

1. Студент Кирячек Владислав Анатольевич гр. 17BO2 факультета вычислительной техники.

2. Тема работы «Android-приложение по выбору должностей на основе характеристики личности и профессиональных навыков»

Тема утверждена приказом ПГУ №0357/о от "26" апреля 2022 г.

3. Руководитель работы д.т.н., заведующий кафедрой ВТ ПГУ Митрохин М.А.

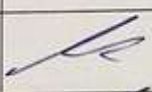
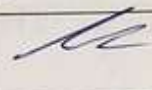
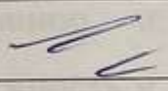
4. Задание на работу (назначение разработки, исходные данные и т.п.)

Android-приложение по выбору должностей на основе характеристики личности и профессиональных навыков предназначена для принятия на должность компетентных кадров. Приложение должно обеспечивать регистрацию, авторизацию пользователей, добавление профилей кандидатов и их фильтрацию на основе характеристики личности и профессиональных навыков.

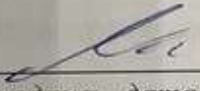
5. Перечень подлежащих разработке вопросов

Требуется произвести анализ предметной области, сформировать техническое задание, выбрать инструменты для разработки, выполнить проектирование системы, разработать Android-приложение по выбору должностей на основе характеристики личности и профессиональных навыков. Программа должна выполнять функции: обеспечивать добавление, изменение и удаление карт и тактических знаков из дополненной реальности; регистрацию и авторизацию пользователей в программе. Также выполнить задание по БЖД и осуществить экономическое обоснование разработки системы

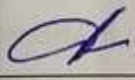
6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области	8%	26.04.2022 – 29.04.2022	
2.	Составление технического задания	6%	30.04.2022 – 02.05.2022	
3.	Выбор инструментов разработки	18%	03.05.2022 – 06.05.2022	
4.	Проектирование системы	32%	07.05.2022 – 12.05.2022	
5.	Выполнение экономического обоснования	10%	02.06.2022 – 06.06.2022	
6.	Выполнение задания по БЖД	8%	07.06.2022 – 10.06.2022	
7.	Оформление пояснительной записки	18%	11.06.2022 – 14.06.2022	

Дата выдачи задания "26" апреля 2022 г.

Руководитель работы  26.04.22
(подпись, дата)

Задание к исполнению принял студент  26.04.22
(подпись, дата)

Работу к защите допустить
Декан факультета  Л. Р. Рыжова
(подпись, дата)
23.06.2022

Реферат

Пояснительная записка содержит 123 листов, 52 рисунка, 8 источников, 4 приложения.

ANDROID, КАНДИДАТ, ПРОФИЛЬ, *HR*, КЛАСС, КОНТЕЙНЕР, ГРУППА, ПРИЛОЖЕНИЕ.

Цель работы – разработать *Android*-приложение по выбору должностей на основе характеристики личности и профессиональных навыков.

Разработанное *Android*-приложение предназначено для выбора кандидатов на должность на основе характеристики личности и профессиональных навыков, оно позволяет вести регистрацию и авторизацию пользователей, добавлять профили кандидатов на должность и проводить их фильтрацию на выбранную должность. Хранение профилей пользователей и кандидатов происходит в базе *Firebase*.

Графическая часть дипломного проекта содержит следующие чертежи:

- диаграммы *UML* – 4 листа формата A1;
- схема взаимодействия программ – 1 лист формата A1.

Разработанная автоматизированная система была реализована на языке программирования *Java* и языке разметки *xml*. Для хранения данных использовалась *NoSql* база данных *Firebase*.

Для инфологического проектирования был использован сервис *DrawIo*.

					ПГУ 1.090501.05.001 ПЗ				
Изм.	Лист	№ докум.	№	Подпись	Дата				
Разраб.		Киричек В.А.				Android-приложение по выбору должностей на основе характеристики личности и профессиональных навыков	Лит.	Лист	Листов
Провер.		Митрохин М.А.						2	123
Н. Контр.		Бычков А.С.							
Утверд.		Митрохин М.А.							
						Пояснительная записка	ФВТ, каф. ВТ гр. 17В02		

Содержание

Перечень сокращений и обозначений	5
Введение	6
1 <i>Android</i> -приложение по выбору должностей на основе характеристики личности и профессиональных навыков.....	7
1.1 Обзор предметной области.....	9
1.2 Обзор способов и средств, доступных для работы с системой подбора должностей.....	9
1.2.1 Определение операционной системы	9
1.2.2 Анализ популярности операционных систем	11
1.2.3 Использование типологии Майер-Бриггс при подборе персонала.....	11
1.2.4 Существующие рекрутинговые системы.....	13
1.3 Постановка задачи.....	17
1.4 Проектирование системы	17
1.4.1 Выбор среды разработки	17
1.5 Разработка приложения для устройств на основе операционной системы <i>Android</i>	20
1.6 Архитектура приложения	48
1.7 Тестирование и отладка.....	49
2 Техничко-экономическое обоснование разработки системы.....	62
2.1 Введение.....	62
2.2 Расчёт затрат на разработку программного обеспечения	63
2.3 Расчёт цены разработанной программы	64
2.4 Расчёт капитальных вложений	65
2.5 Расчёт эксплуатационных расходов.....	66
2.6 Определение расходов на интернет трафик	68
2.7 Техничко-экономические показатели разработки.....	68
3 Безопасность жизнедеятельности. Эргономика рабочего места.....	70
3.1 Характеристика условий труда программиста.....	70

3.2 Требования к параметрам воздушной среды.	71
3.3 Требования к уровню шума и вибрации.	72
3.4 Требования к монтажу производственного оборудования.....	72
3.5 Требования к освещению рабочих мест.	73
3.6 Расчет освещения рабочего места программиста.	73
Заключение.	76
Список использованных источников	77
Приложение А – <i>UML</i> - диаграммы	78
Приложение Б – Схема взаимодействия программ	83
Приложение В - Листинг программы.....	85
Приложение Г - Презентация.....	120

Перечень сокращений и обозначений

БД – база данных

ОС – операционная система

ПО – программное обеспечение

					<i>ПГУ 1.09.05.01.05.001 ПЗ</i>	<i>Лист</i>
<i>Изм.</i>	<i>Лист</i>	<i>№ докум.</i>	<i>Подпись</i>	<i>Дата</i>		5

Введение

В настоящее время, подбор персонала на должность является важной задачей, ведь на каждое место претендует большое количество кандидатов, а их компетенции и вероятность твердо обосноваться на рабочем месте и в коллективе стоят под сомнением.

На данный момент, при подборе сильно влияет человеческий фактор, компетенции сотрудников на должности *HR*, зачастую не столь высоки и по этой причине, они не всегда имеют представление, какой человек в большей или меньшей степени подойдет на должность. Для решения данных проблем можно привлечь современные методы, а именно приложение по автоматическому подбору персонала. Данная работа предлагает использование приложения на смартфоны под управлением *Android*, с помощью которого руководитель сможет получать список наиболее подходящих кандидатов на искомое место.

Разработка данного программного обеспечения представляет собой совокупность пользовательского интерфейса в приложении, занесения в базу кандидатов и алгоритма подбора кандидатов на должность. Поэтому необходимо проанализировать программное обеспечение, базы данных и алгоритмы для успешного выполнения поставленных задач приложением.

1 Android-приложение по выбору должностей на основе характеристики личности и профессиональных навыков

1.1 Обзор предметной области

Отбор персонала в организацию начинается после представления соискателем и изучения специалистом базовых сведений. Собеседование проводится в несколько этапов, в которых часть кандидатов отсеивается, когда становится известным их полное или частичное несоответствие заявленным требованиям. Многоступенчатая организация отбора при найме персонала значительно снижает вероятность совершения ошибки при рассмотрении нового сотрудника.

Грамотный отбор и подбор персонала организации включает следующие этапы:

- анкетирование, предоставление автобиографических сведений, анализ послужного списка;
- собеседование;
- освидетельствование на предмет профессиональной подготовки;
- медицинский осмотр;
- принятие решения о приеме на работу, либо отказ соискателю.

Если кандидат проходит предварительный отбор, то по форме, которая была установлена в учреждении, он так же заполняет анкеты и резюме. Специалисты проверяют деятельность кандидата на предыдущих местах работы [1].

На этапе собеседования, организация может получить данные о личных качествах кандидата. Во время беседы, сотрудник обращает внимание на все мелочи собеседуемого, такие как опрятный внешний вид, умение держаться, культура поведения, манеры, способность формировать и выражать мысли, а так же слушать и проводить конструктивный диалог. По итогам собеседования, не проходят отбор еще так же часть кандидатов и на вакантное

					ПГУ 1.09.05.01.05.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		7

место остаются только те, кто подходят по требованиям для данной должности.

Проверка на профессиональную подготовленность идет так же во время найма нового сотрудника, подходящие результаты освидетельствования подтверждают, что специалист за счет наличия необходимых качеств подходит для искомой должности.

Система отбора персонала зачастую включает в себя прохождение медицинской комиссии. Такого рода освидетельствование повышает шансы на то, что сотрудник в ближайшее время не получит травму или профессиональное заболевание, а его физическая форма позволит работать в конкретной сфере деятельности.

Поэтапный процесс отбора кандидатов дает возможность получить достаточное количество информации о кандидатах. Работодатель может выбирать самую подходящую кандидатуру среди множества собеседуемых. После принятия итогового решения, подготавливается документ о найме кандидата и заключается договор.

Самым тяжелым этапом во всем процессе можно считать именно собеседование, так как оно сильно завязано на человеческом факторе, а именно:

- профессиональная готовность человека, который будет заниматься отбором персонала в компанию
- высокий уровень недоверия у руководителей к психологическим тестам.
- специалист верит в тесты, умеет ими пользоваться, но не может сформулировать, что именно и какими именно методиками нужно мерить и как интерпретировать [2].

Таким образом, выстраивается проблема выбора кандидатов на должность, когда по характеристикам своей личности они не подходят для занимаемого места. Решением этой проблемы может быть автоматический выбор кандидатов на основе полученных данных на этапе анкетирования.

Платформа, на которой будет проводиться подбор должна быть наиболее доступна большинству.

1.2 Обзор способов и средств доступных для работы с системой подбора должностей

1.2.1 Определение операционной системы

Согласно исследованиями *Sensor Tower* потребительские расходы на мобильные приложения и установки приложений значительно выросли за последние несколько лет, в одном только 2020 году денежные затраты пользователей в этой сфере достигли 111 миллиардов долларов во всем мире. Так как рынок развивается стремительно и перспектива именно за ним, то и акцент необходимо делать на этом.

У мобильного приложения есть ряд преимуществ:

- обеспечивает более тесное взаимодействие с пользователем;
- предоставляется прямой канал связи с клиентурой;
- существует функциональность, даже при работе в автономном режиме.

Поэтому перед выбором встают две платформы – *Android* и *IOS*. Так как задачи *Android*-приложения по выбору должностей на основе характеристики личности и профессиональных навыков ориентированы в большей степени на тесное взаимодействие с пользователем:

- ОС *Android* – открытый и бесплатный абстрактный список элементов от *Google*, который включает в себя ОС, ПО и важные приложения для использования на смартфонах и мобильных устройствах.

- ОС *IOS* – закрытая система, которая так же включает в себя список элементов от *Apple* и поддерживается исключительно на устройствах компании, таких как *Iphone*, *Ipad* и смарт часы.

Плюсы и минусы *Android* и *IOS*.

Устройства под руководством операционной системы от *Apple* имеют ряд весомых плюсов:

					ПГУ 1.09.05.01.05.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		9

- удобное интуитивно понятное управление, меню;
- комфортная, продуманная синхронизация с устройствами от компании производителя, как в ручном, так и автоматическом режиме;
- долгосрочная поддержка обновлений устройств с внесением доработок и улучшений гаджетов;
- закрытость операционной системы.

Если рассматривать закрытость системы с положительной стороны, то можно отметить, что такая политика делает систему надежной и неуязвимой, поддерживает стабильную работу и общее качество на высоком уровне.

Из минусов стоит отметить несколько основных:

- возможность сбоев в передаче данных с устройствами других платформ;
- отсутствие бюджетных моделей;
- закрытая система не позволяет вносить доработки пользователям.

Устройства под руководством операционной системы от *Google* так же имеют ряд своих, как положительных, так и отрицательных качеств.

Среди плюсов стоит отметить:

- огромный выбор моделей телефонов для любого бюджета;
- большой выбор позволяет найти устройства с разными возможностями, которые необходимы конкретному потребителю;
- открытая операционная система, позволяет вносить пользователям свои доработки в устройства на данной базе;
- открытая система позволяет устанавливать софт из любых источников.

Основные минусы исходят именно от открытой системы:

- слабая защищенность от вредоносного ПО;
- короткий срок поддержки;
- отсутствие оптимизации системы под все устройства [3].

Данные плюсы и минусы были приведены с позиции того, что важнее в конечном итоге пользователям и что будет более востребованным.

1.2.2 Анализ популярности операционных систем

На основе данных, собранных статистическими сервисами, было выявлено, что 71.43% пользователей используют устройства на базе операционной системы *Android*, в то время, как на базе *IOS* 27.85%. Пользователи в преобладающем большинстве выбирают низкую стоимость, разнообразие моделей и открытую систему [4].

1.2.3 Использование типологии Майер-Бриггс при подборе персонала

Во многих странах при выборе профессии используются различные тесты, но именно тест Майер-Бриггс наиболее распространен, так как имеет не 5-6 градаций, а 16, что позволяет более точно определить предрасположенность человека к определенному роду деятельности.

Существуют две основные группы признаков, влияющих на выбор должностей:

- логика-этика. Эта группа определяет способ принятия решений. Логика опирается на объективные данные и факты. Поэтому они успешнее в тех сферах деятельности, где это необходимо. Этики, в свою очередь, руководствуются принятыми моральными нормами и ценностями, поэтому им подходит социальная и гуманитарная сфера деятельности;

- сенсорика-интуиция. Эти признаки определяют тип мышления. У интуитов оно абстрактное, поэтому им проще понимать смысл в целом, прогнозировать, изобретать. Сенсорика обладает практическим мышлением, поэтому они справляются лучше в тех сферах, где требуется точность и аккуратность.

Распределение по сферам, в целом, строится по следующим значениям:

- логики-интуиты —сфера исследований: наука, техника, программирование, физика, математика, аналитика, генерация идей. Для типов личности *ENTj* и *INTp* более свойственна практическая и организаторская деятельность в этих сферах, так как их сильными сторонами являются деловая логика и интуиция времени, а для *ENTp* и *INTj* — теоретическая деятельность,

связанная с исследованиями и созданием новых идей, потому что у них сильны логика соотношений и интуиция возможностей.

- логики-сенсоры — управленцы и производственники: экономика, финансы, спорт, создание структур и организаций, производство, контроль деятельности. У *ESTp* и *ISTj* это законотворчество и строгий контроль, потому что их сильной стороной является логика соотношений и силовая сенсорика. У *ESTj* и *ISTp* — менее строгий, но направленный на повышение эффективности и удобства. У них сильными сторонами являются деловая логика и сенсорика ощущений;

- этики-интуиты — гуманитарии: преподавание, литература, искусство, религия, история, философия, психология. Для *ENFj* и *INFp* ближе искусство как таковое в контексте эмоций, им вызываемых. Это потому что их сильные стороны этика эмоций и интуиция времени. А *INFj* и *ENFp*, которые лучше разбираются в людях в силу развитых этики отношений и интуиции возможностей — преподавание, психология, религия;

- этики-сенсоры — это социалы: забота о других, медицина, сфера развлечений и агитации, кулинария, забота о внешнем виде: стиль, визаж, дизайн.

Причем то, что относится к вопросам агитации и развлечений больше подходит *ESFj* и *ISFp*, потому что их сильными сторонами является этика эмоций и сенсорика ощущений, а социальная работа и медицина — *ESFp*, *ISFj*, потому что их сильные функции этика отношений и силовая сенсорика;

Экстраверсия-интроверсия существует в двух вариантах – силовая сенсорика и сенсорика ощущений. Первая определяет внутренние ощущения человека (она интровертна), вторая взаимодействие с внешней средой (она экстравертна).

Поскольку экстраверсия нацелена на внешнюю среду, то у экстравертов лучше развиты аспекты отвечающие за взаимодействие с окружающим миром. Уточним для решения каких задач будут эффективны такие экстравертные аспекты:

					ПГУ 1.09.05.01.05.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		12

- деловая логика используется для повышения эффективности;
- этика эмоций для пропаганды, поддержания энтузиазма;
- интуиция возможностей для генерации идей;
- силовая сенсорика для физического воздействия на что-либо.

Экстравертам требуется работа контактная, с людьми, активная деятельность, когда происходит часто что-то новое. Им неплохо быть организаторами, ездить в командировки, проводить собеседования.

У интровертов лучше развиты черты отвечающие за внутренний мир человека. Такие аспекты будут эффективны для других задач:

- логика соотношений — анализ данных и выработка правил поведения;
- этика отношений — симпатии-антипатии, сострадание;
- сенсорика ощущений — создание комфортных для человека условий;
- интуиция времени — выбор подходящего для действий момента.

Интровертам подойдет более спокойная работа, с меньшим кругом общения, побольше возможностей разбираться и прорабатывать накопленный материал, полностью погружаться в свою сферу деятельности, специализироваться в одной конкретной области [6].

1.2.4 Существующие рекрутинговые системы

На данный момент существует множество рекрутинговых сервисов по управлению кандидатами, их основная цель, первичный подбор и управление кадрами внутри подразделений. Они основываются на анкетах кандидатов и уже относительно их заполнения, HR-менеджер распределяет их по должностям. Самыми популярными в России являются:

- Хантфлоу – позволяет отслеживать ход поиска и повышать его эффективность. Реализован автоматический сбор резюме с более 15 job-сайтов, есть возможность публикации собственных вакансий на площадках прямо из системы. Существует встроенный почтовый клиент для переписки с кандидатами, в том числе для запросов согласия на обработку персональных

данных, отправки офферов. Реализована интеграция с рабочими календарями, встроенный планировщик занятости. Открытое *Api* позволяет интегрировать другие сервисы, в том числе карьерный сайт (рисунок 1);

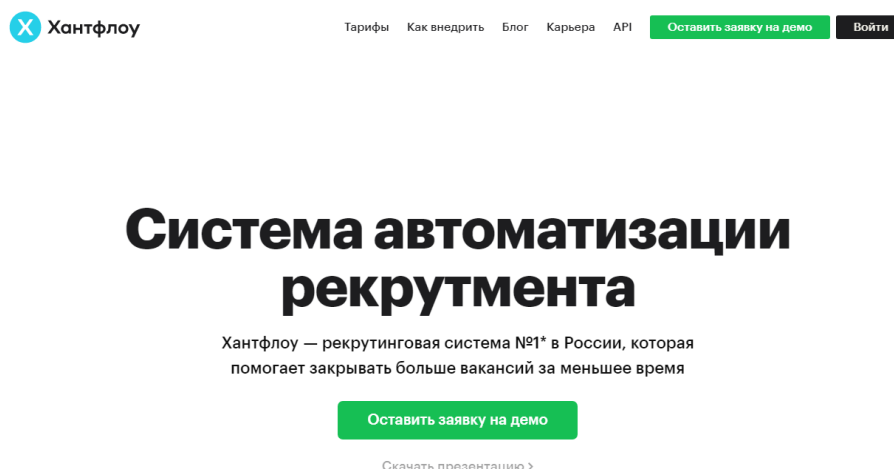


Рисунок 1 – Главная страница сайта Хантфлоу

- *Potok* – обеспечивает хранение базы кандидатов и истории общения с ними. Есть мобильная версия. Поддерживает сбор резюме с более 10 *job* – сайтов. Реализован поиск по базе кандидатов и по ключевым словам в резюме. Есть встроенный карьерный сайт, настроены отклики с него прямо в систему. Возможно подключение рабочей почты в личном кабинете для получения и отправки писем кандидатам, включая запросы согласия на обработку персональных данных. Есть собственный почтовый клиент. Так же существует интеграция с календарем, открытое *API*, позволяющее использовать интеграции с телефонии и современными мессенджерами. Возможна интеграция с система видео-интервью обучения, организационно-штатными структурами и чат ботами. Позволяет настраивать автоматические цепочки согласования кандидата с системой уведомления для кандидатов, заказчиков и рекрутеров (рисунок 2);

					ПГУ 1.09.05.01.05.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		14

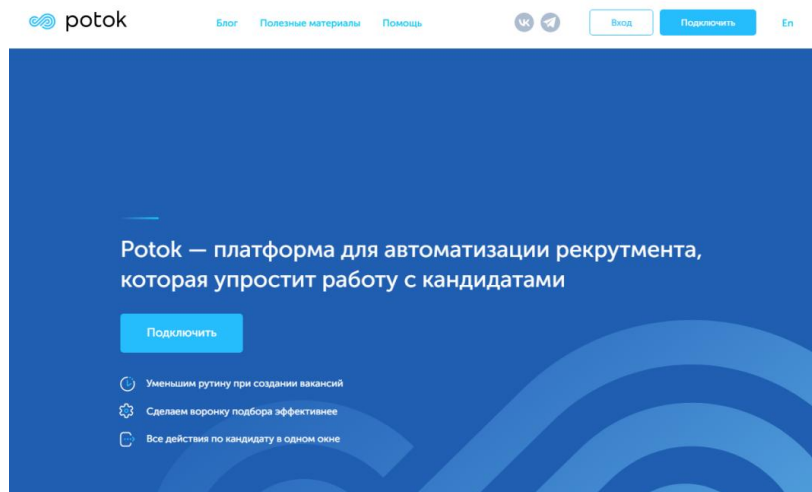


Рисунок 2 – Главная страница сайта *Potok*

- *FriendWork* – доступен с любых видов устройств, есть англоязычная версия. Поддерживает импорт резюме с более 15 *job*-сайтов и из социальных сетей. Реализован гибкий поиск по ключевым словам. Есть свой карьерный портал. Настраивается строгая последовательность этапов для перемещения кандидатов по воронке подбора в запланированном порядке, есть напоминания о событиях для рекрутера, заказчика и кандидата, а также форма обратной связи для руководителей. Доступно публичное *API*, реализованы интеграции с конструктором сайтов Тильда, карьерным сайтом заказчика и *BI*- системами, с почтой и календарями (рисунок 3);

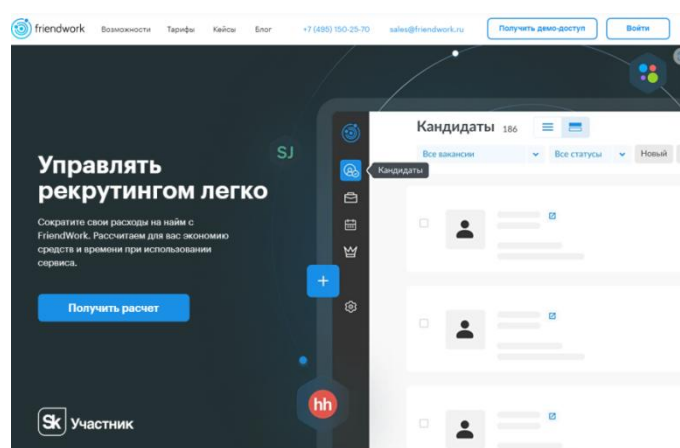


Рисунок 3 – Главная страница сайта *FriendWork*

- *Talantix* – разработанная компанией *HeadHunter* и входит в ее *HR*-систему. Поддерживает импорт резюме с разных платформ и рабочих сайтов. Есть возможность загрузки прежней базы прямо в интерфейсе пользователя. Реализован полнотекстовый поиск кандидатов, а также фильтры по меткам, вакансиям, этапам подбора, региону и далее. Возможна настройка этапов подбора и шаблонов для разных типов вакансий. Ранжирование кандидатов на вакансиях на *AI* и полностью автоматизированный сценарий работы с тестовыми заданиями. Реализованы интеграции с мессенджерами и почтовыми сервисами для проведения собеседований онлайн. Отсутствует публичный *API* [5] (рисунок 4).

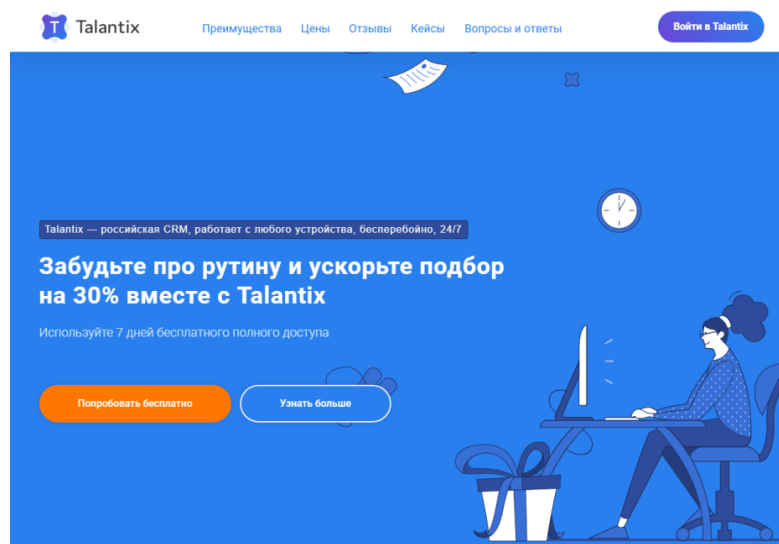


Рисунок 4 – Главная страница сайта *Talantix*

На основе полученных данных можно отметить, что самая подходящая мобильная система для работы с кандидатами будет *Android* от *Google*. Принятие такого решения обусловлено распространенностью устройств на представленной ОС и свободы работы с софтом для нее. В то же время, исследовав рынок рекрутинговых систем следует отметить, что их основной функцией является связь с кандидатами и все функции сервиса направлены именно на это. В то время, как человеческий фактор все так же решает в выборе рекрутера на должность, тем самым обнажая проблемы конечного

выбора персонала. Решением этой проблемы может оказаться система подбора персонала на основе их профессиональных навыков и характеристики личности. Благодаря анализу результатов теста Мейер-Бриггс и сопоставлению их с навыками искомой специальностью, возможно подобрать наиболее подходящих кандидатов на представленные должности.

1.3 Постановка задачи

Необходимо разработать приложение на телефон под управление операционной системы *Android* «Подбор персонала на основе их профессиональных навыков и характеристики личности».

В данной программе будет реализована часть, отвечающая за регистрацию, вход и сохранение, как пользователей, так и администраторов, заполнение информации о кандидатах и занесение всей информации о пользователях, администраторах и кандидатах в базу данных. Кроме того, реализован автоматический подбор пользователей на должность, исходя из занесенной информации об их навыках и типе личности на основе типологии Майерс-Бриггс.

Программа должна быть разработана для работы в ОС *Android* и подходит для версий *Android* от 8.0 и выше.

1.4 Проектирование системы

1.4.1 Выбор среды разработки

Самыми распространенными средами разработки на *Android* на данный момент являются *Eclipse*, *IntelliJ Idea* и *Android Studio*. Каждая из них имеет, как ряд своих преимуществ, так и недостатков.

Компания *IBM* вложила большие силы в то, что бы *Eclipse* была доступным решением для разработчиков. Она является *open source* и каждый может написать под нее плагин и посмотреть исходный код.

Преимущества *Eclipse* заключаются в том, что это *open source* и на данный момент является скорее каркасом, на котором возможно собрать

					ПГУ 1.09.05.01.05.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		17

среду, модифицированную под конкретную задачу. Та же ситуация с поддержкой языков. Благодаря сообществу, есть возможность работать с огромным количеством языков. Так же может работать в групповом режиме.

Минусы *Eclipse* заключаются в том, что используется большее количество системных ресурсов и мощности процессора, встречаются трудности с обратной совместимостью, не редки случаи, когда сторонние плагины, на которых строился проект становятся несовместимы в новых версиях *Eclipse*. Так же проявляются конфликты между несколькими модулями при одновременном использовании. Несовершенная функция отладки, в отличии от платных *IDE* инструментов, а так же неполная интеграция с *Maven* и *Gradle*, что приводит к тому, что преимущество непрерывной компиляции порой не используются [7].

Разработчики называют *IntelliJde* систему «умной» и на то есть весомые причины. Здесь реализовано не просто автоматическое заполнение шаблонов, как в *Emmet*, здесь реализован глубокий семантический анализ написанного для понимания сути кода и предсказания возможных вариантов продолжения.

Преимущества *IntelliJIDE* заключаются в том, что программа имеет большие возможности настройки, обеспечивает поддержку языков программирования на основе *JVM*, таких как *Kotlin*. Имеет встроенную поддержку контроля версий и систему сборки *Gradle*.

Основной недостаток *IntelliJde* продиктован ценой продукта, она достаточно высокая и подходит для крупных коммерческих предприятий. Так же, проблемой является загрузка системных ресурсов и крутая кривая обучения для новых разработчиков.

Google в свою очередь представляет свой продукт, в котором нет принципиальных отличий от выше представленной среды, но так как *Android Studio* представляется от *Google*, то она и является рекомендованной системой для разработки приложений на *Android*.

Преимуществом *Android Studio* является эмуляция устройств на базе ОС *Android*, которое имеет полный функционал копируемого устройства. Среда

содержит в себе встроенный *Version Control System* – набор систем контроля над версиями проекта, который позволяет разработчику постоянно регистрировать изменения в файлах, с которыми он работает для того, что бы в дальнейшем, при необходимости была возможность вернуться к той или иной версии файлов. *Android Studio* предоставляет возможность записи тестов пользовательского интерфейса во время работы приложения. *Instant Run* позволяет разработчику увидеть изменения, которые вносились в код программы или эмулятор, практически в тот же момент, что позволяет быстро оценить эффективность изменения кода и его влияние на производительность. Так же, среда имеет удобный дизайнер пользовательских интерфейсов, позволяющий облегчить визуальное проектирование приложения, *xml* редактор, поддержку рендера средства *GPU*, а так же большую базу сервисов, интегрированных в работу со средой [8].

Недостатки *Android Studio* заключаются в том, что все средства пакета требуют большого количества места на диске, как и высокой производительности для тестирования и эмуляции. Отсутствует возможность написания серверных проектов на *Java*, в отличии от *IntelliJ IDE*, семантический анализ менее сильный и меньшее количество возможностей его настройки.

На основании всех вышеперечисленных плюсов и минусов, можно отметить, что для одиночной разработки в большей степени подходит именно *Android Studio*, так как позволяет вести удобную эмуляцию, является бесплатной и позволяет анализировать написанный код в процессе работы. Так же, важным ее преимуществом оказывается то, что она имеет прямую поддержку сервисов *Google* и сторонних библиотек, которые активно используются в разработке.

1.5 Разработка приложения для устройств на основе операционной системы *Android*

Приложение по подбору персонала на основе характеристики личности

					ПГУ 1.09.05.01.05.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		19

и профессиональных навыков состоит из совокупности двух частей – визуальной и программной:

- визуальная часть:

- 1) страницы приложения;
- 2) изображения;
- 3) визуальные объекты;

- программная часть:

- 1) функции;
- 2) расширяемый язык разметки;
- 3) установление связи с базой данных.

Для создания нового проекта в *Android Studio* необходимо выполнить следующие действия:

- выбрать один из вариантов *Activity* при создании проекта (рисунок 5);
- на странице создания проекта, ввести название проекта, название его архива, место сохранения, язык разработки и версию комплекта для разработки программного обеспечения (рисунок 6).

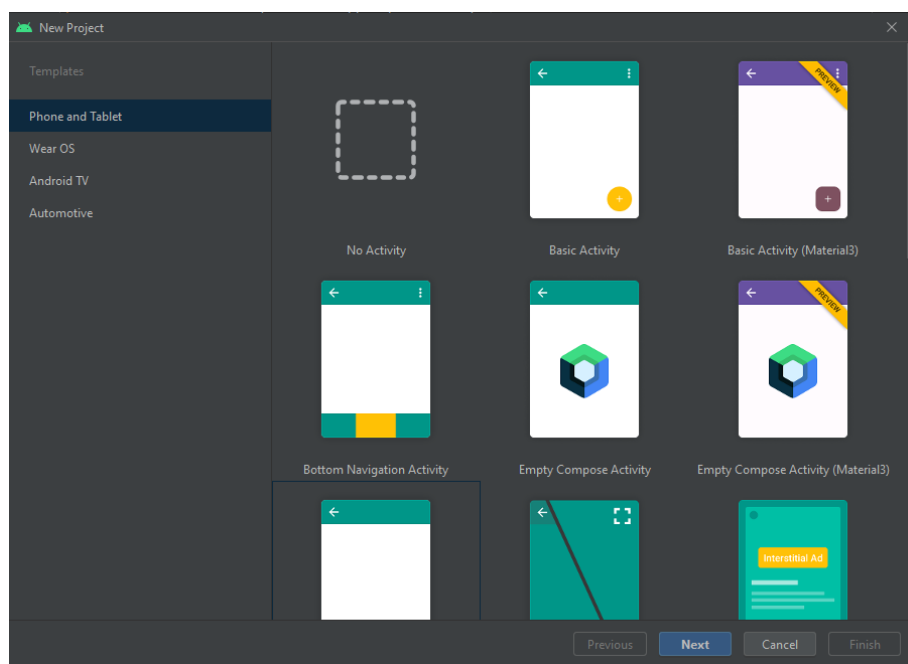


Рисунок 5 – Выбор Activity

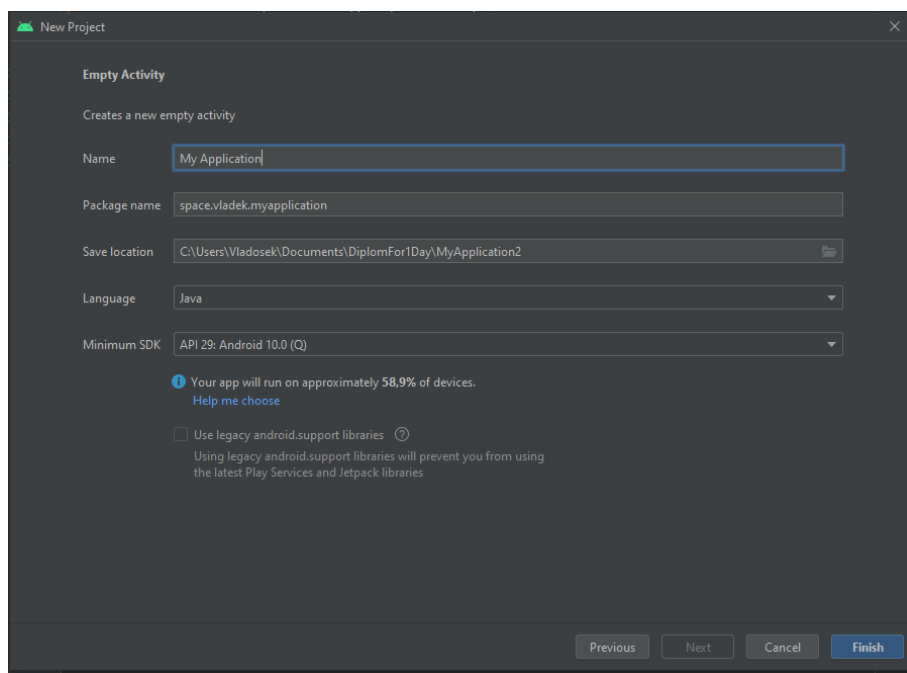


Рисунок 6 – Первоначальные настройки проекта

В левом окне среды разработки, находится дерево файлов проекта, где находятся все основные папки и файлы проекта (рисунок 7).

В дереве файлов проекта заключены следующие модули:

- *manifest* отвечает за инкапсуляцию архитектуры *Android*-приложения, функциональные возможности и конфигурацию;
- в папке *Java* заключены классы и функции приложения, отвечающие за основные процессы работы приложения;
- в папке *res* находятся подпапки и файлы, отвечающие за работу с визуальной составляющей проекта;
- в подпапке *drawable* находятся файлы и изображения, отвечающие за графику в приложении;
- в подпапке *layout* находятся файлы разметки страниц, на которых отображен визуальный интерфейс приложения;
- в подпапке *mipmap* находятся текстуры для значков приложения;
- в подпапке *values* хранятся ресурсы массивов, к которым можно обращаться из кода проекта;

- в подпапке *xml* хранятся отдельные файлы разметки страниц, которые могут быть использованы для отдельных визуальных элементов приложения;
- в пакете *Gradle Scripts* представлены инструкции и свойства автоматической сборки проекта.

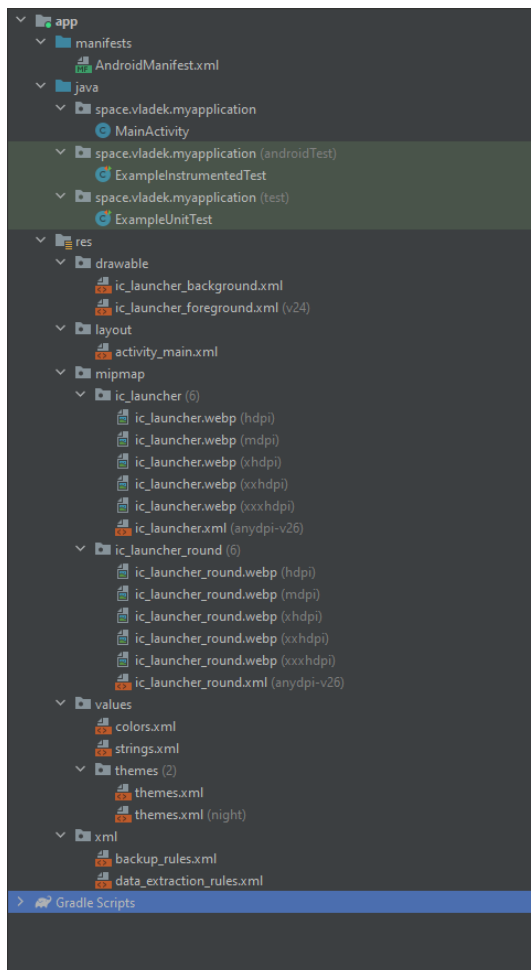


Рисунок 7 – Дерево файлов проекта

В целом, окно среды разработки выглядит следующим образом (рисунок 8):

- сверху меню навигации и настройки среды разработки;
- слева менеджер ресурсов, структуры и дерева проекта;
- снизу окна панель логов компиляции;
- слева окна настройки и запуска виртуальных и физических устройств;
- в центре основное рабочее пространства проекта.

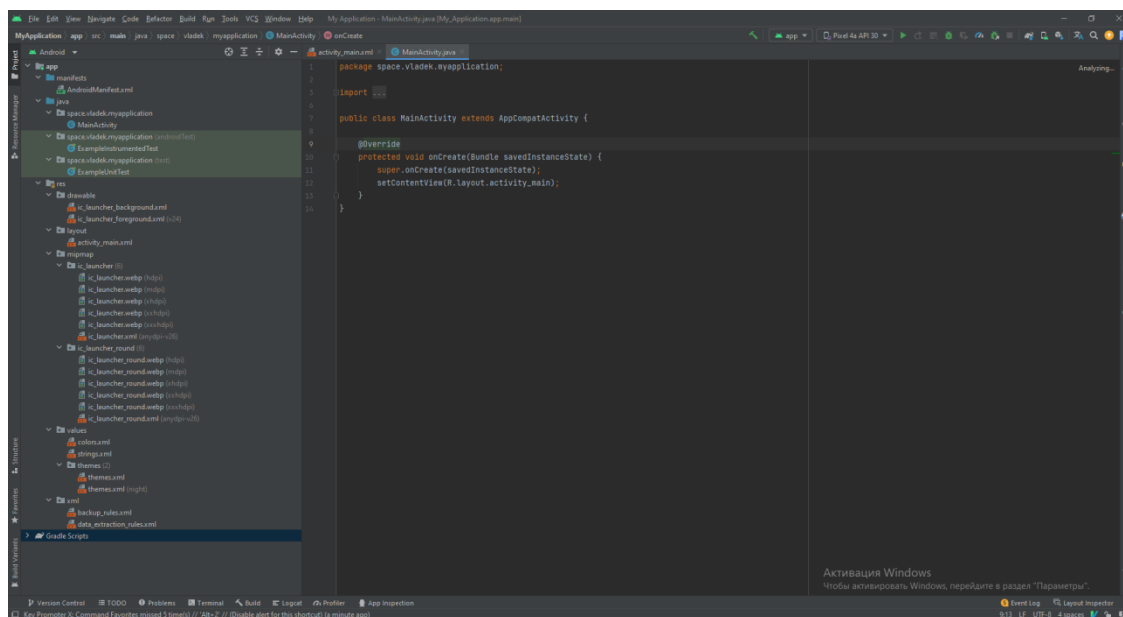


Рисунок 8 – Окно среды разработки проекта

Для создания новых файлов в дереве приложения, необходимо нажать правую кнопку мыши, перевести на *new* и выбрать необходимый формат файла (рисунок 9).

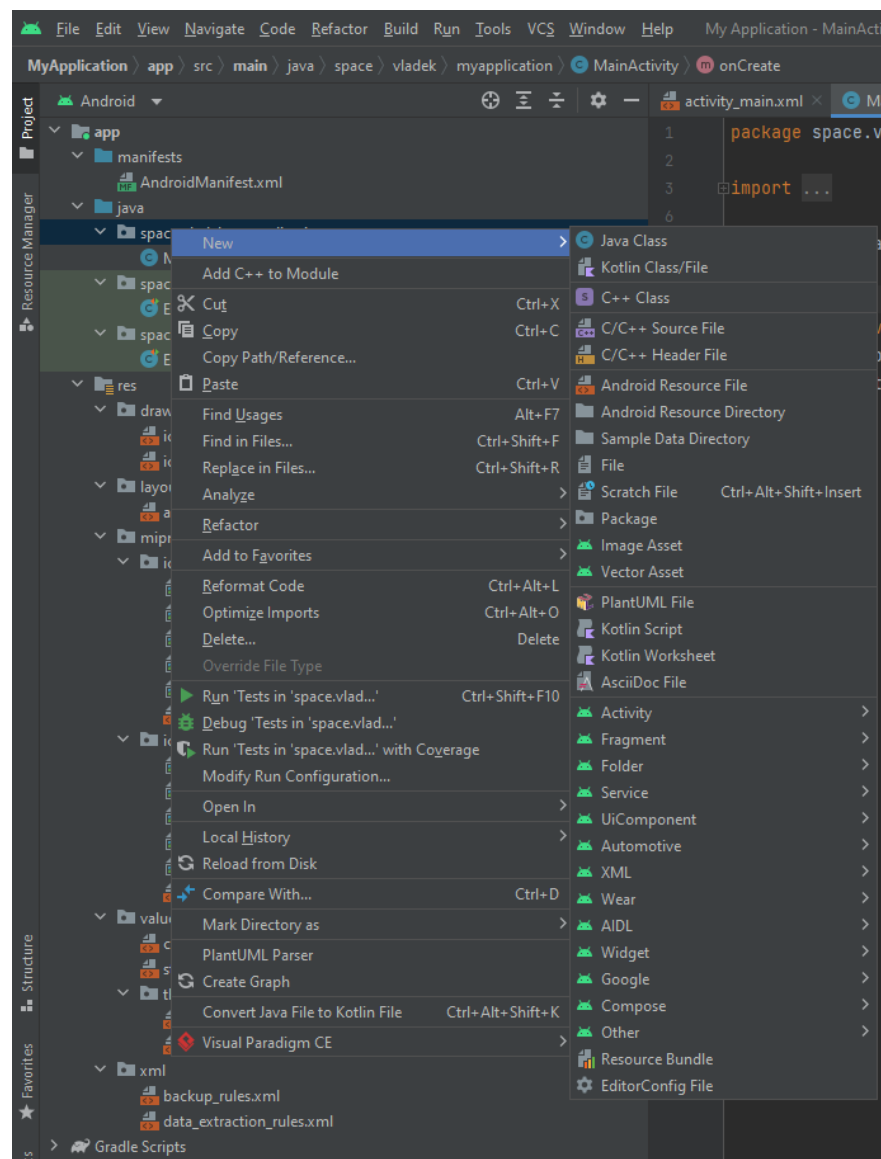


Рисунок 9 – Создание нового файла в приложении

Есть возможность, как создания независимых файлов, так и заранее связанных между собой страниц *Activity*. При создании нового класса в таком случае, будет одновременно создан файл разметки в подпапке *layout* (рисунок 10).

При переключении на файл разметки, есть возможность просмотра и редактирования в трех формах – дизайн, двойной экран и код. Переключение этих форм находится в правом верхнем углу среды разработки (рисунок 11).

При выборе визуального и двойного отображения файла разметки, есть возможность изменять его с помощью графического интерфейса. Для

добавления элемента графического интерфейса необходимо перетащить его из палетки в левом верхнем углу на рабочее пространство (рисунок 12).

После добавления в рабочее пространство, есть возможность выбрать этот объект и отредактировать его свойства средствами визуального интерфейса, которые находятся на правой панели среды разработки (рисунок 13).

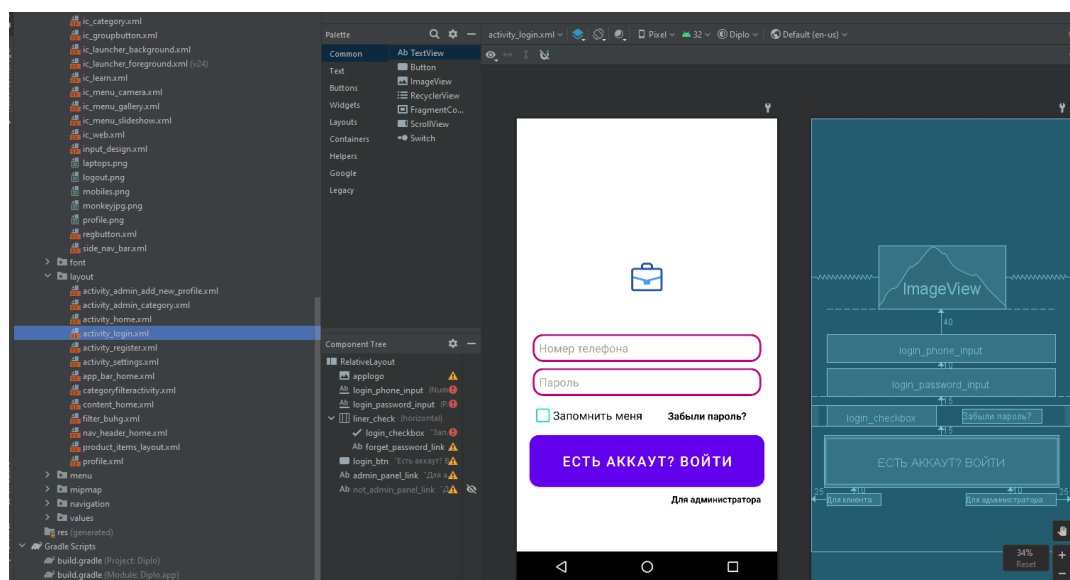


Рисунок 10 – Внешний вид файла XML

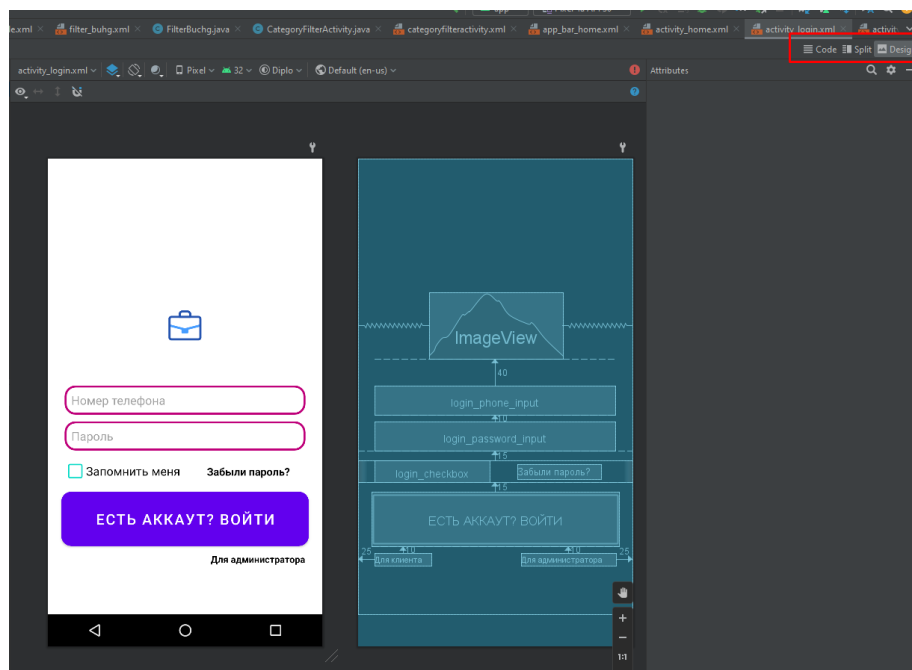


Рисунок 11 – Выбор формы отображения разметки

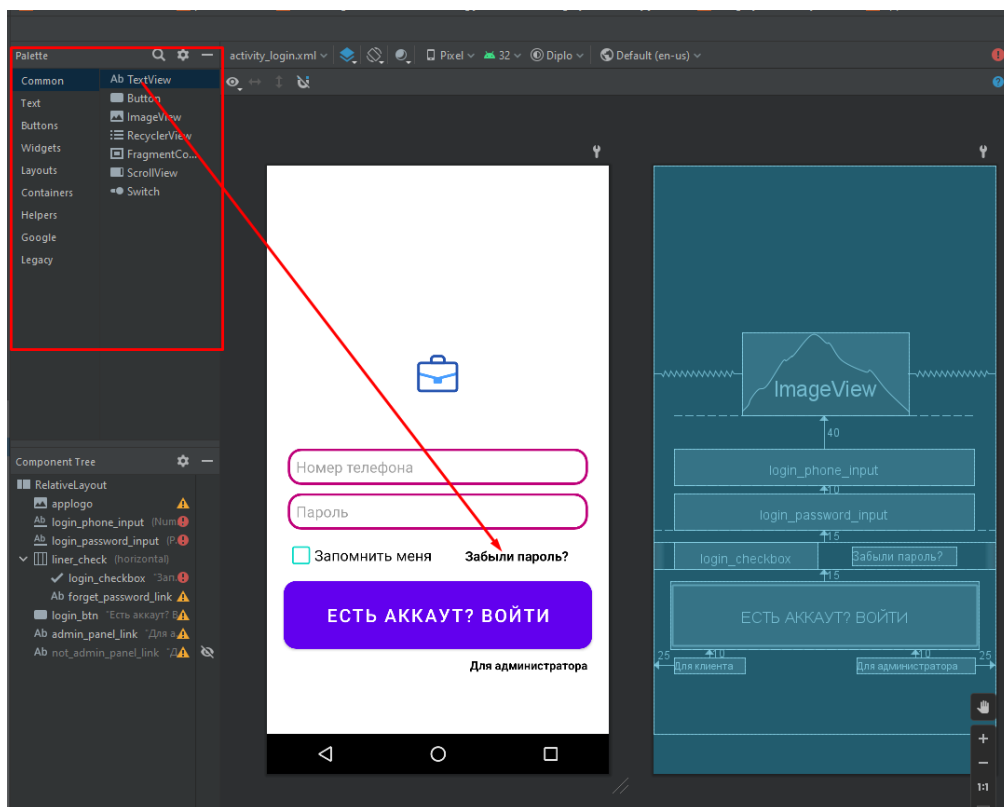


Рисунок 12 – Добавление объекта в файл

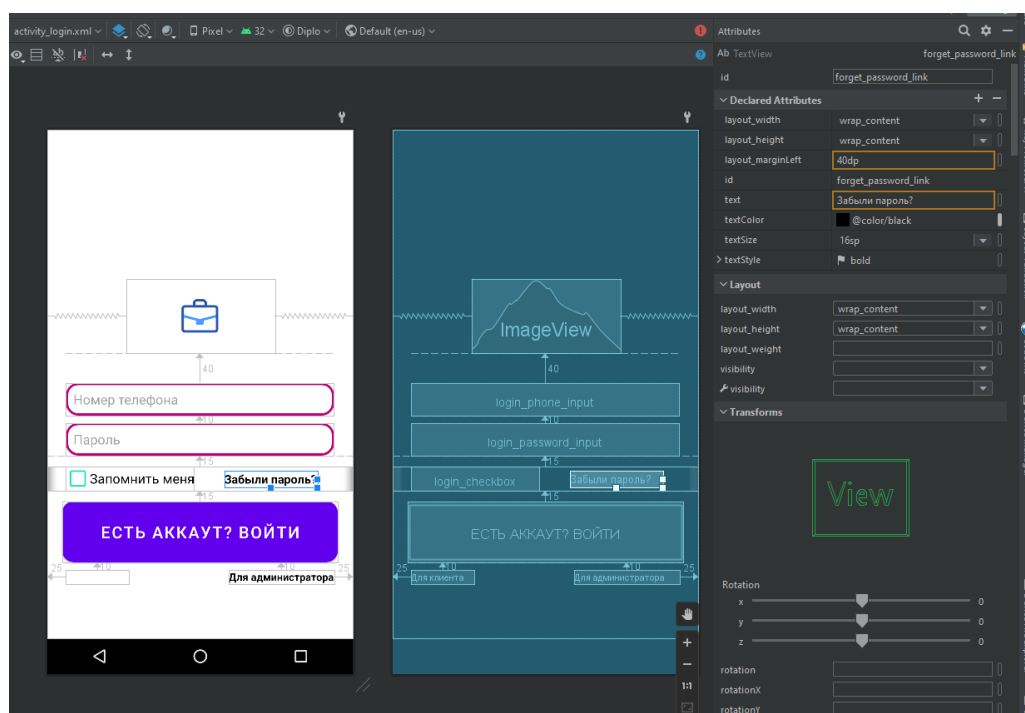


Рисунок 13 – Редактирование свойств объекта

В качестве базы данных использовалась *NoSql Firebase* от *Google*. Для подключения базы к проекту, требуется создать проект внутри *Firebase*. Для этого необходимо нажать на кнопку «*Add a project*» (рисунок 14). Затем ввести название проекта (рисунок 15) и включить или отключить аналитику (рисунок 16).

Предварительно требуется войти в свой аккаунт *Google* в самой среде разработки, для того, что бы была возможна связь базы и приложения (рисунок 17).

После создания проекта, возможно добавление базы к приложению, для этого необходимо нажать на иконку *Android* (рисунок 18).

Затем на странице подключения приложения, необходимо ввести название хранилища и название проекта (рисунок 19). После, скачать и добавить в проект файл *google-services.json* (рисунок 20). Затем, добавить код в инструкции автоматической сборки *gradle* для связи с *Firebase* (рисунок 21). Затем доступно завершение подключения приложения к базе (рисунок 22).

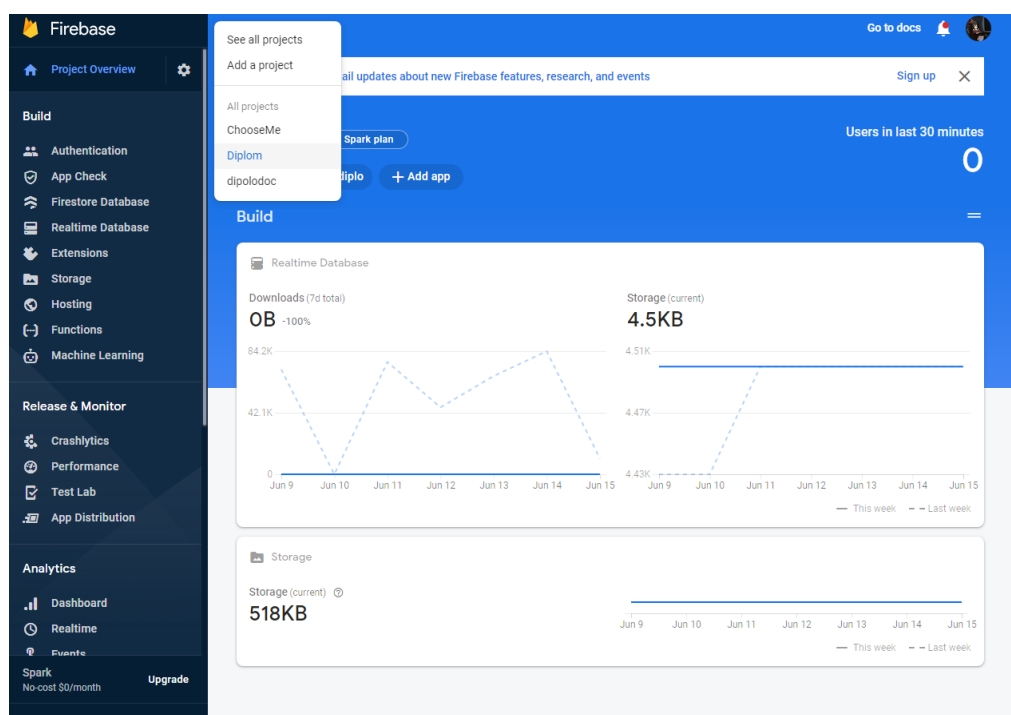


Рисунок 14 – Добавление проекта

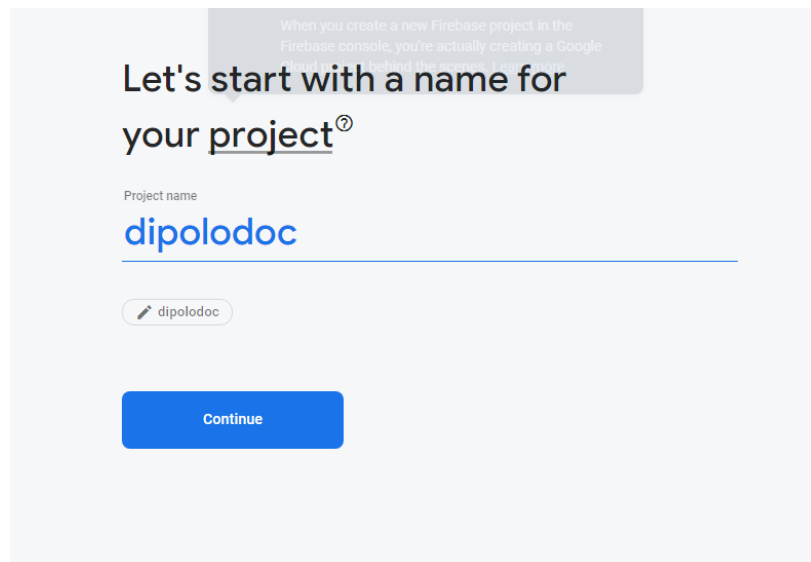


Рисунок 15 – Ввод названия проекта

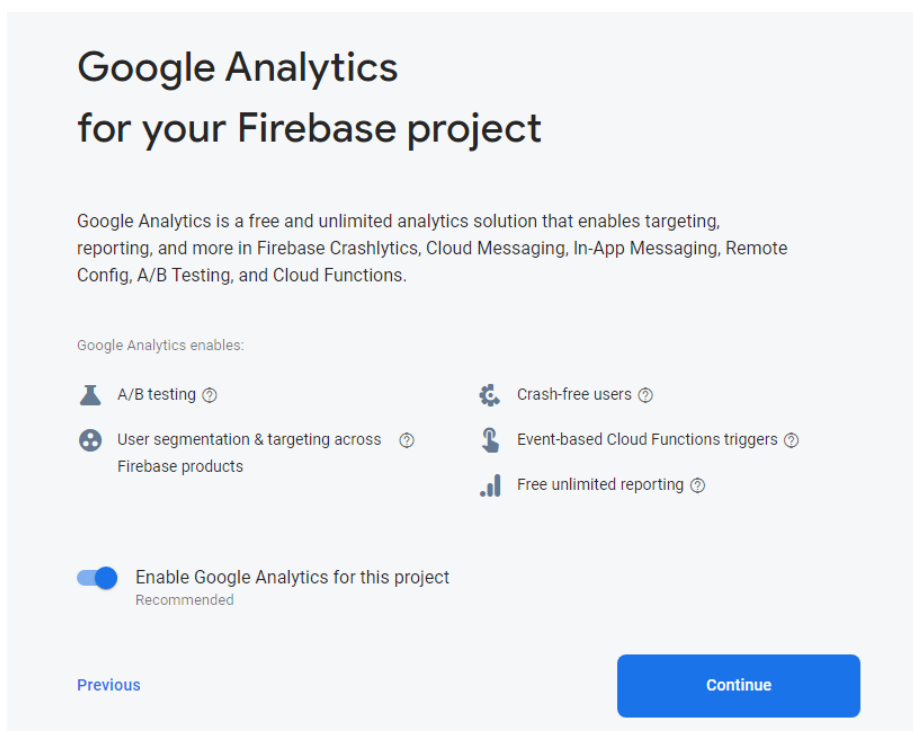


Рисунок 16 – Включение доступа к аналитике

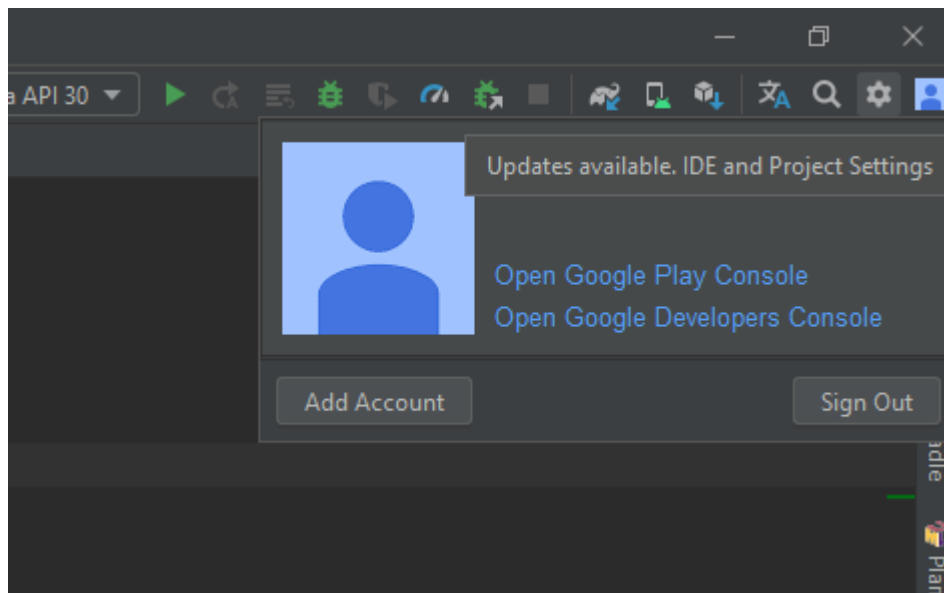


Рисунок 17 – Вход в *Google* аккаунт

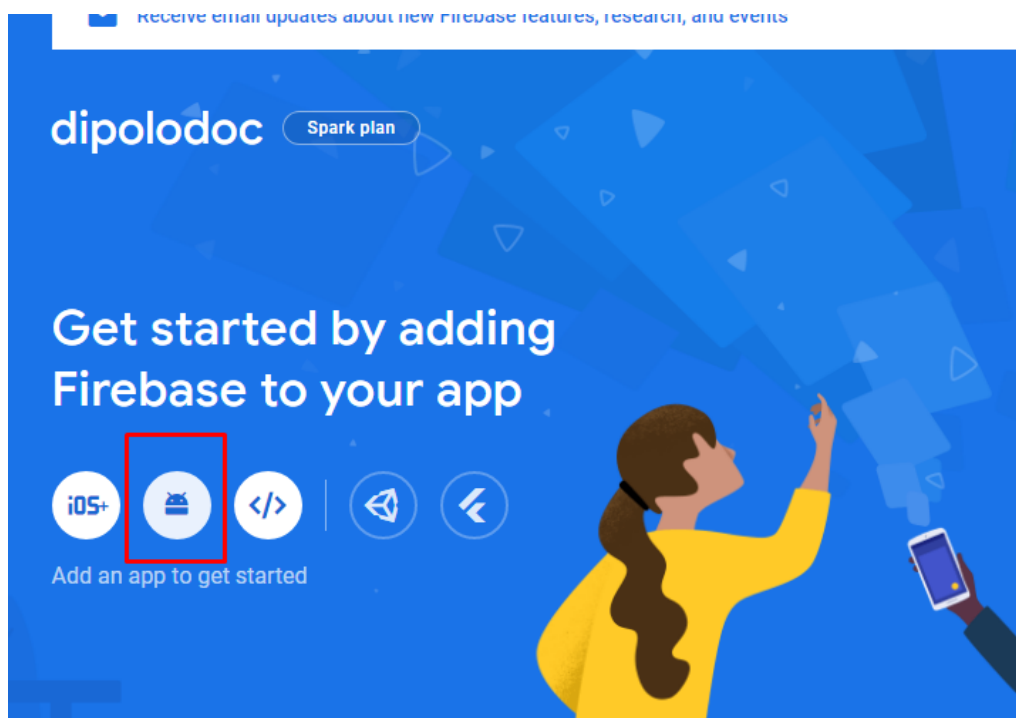


Рисунок 18 – Добавление приложения к проекту

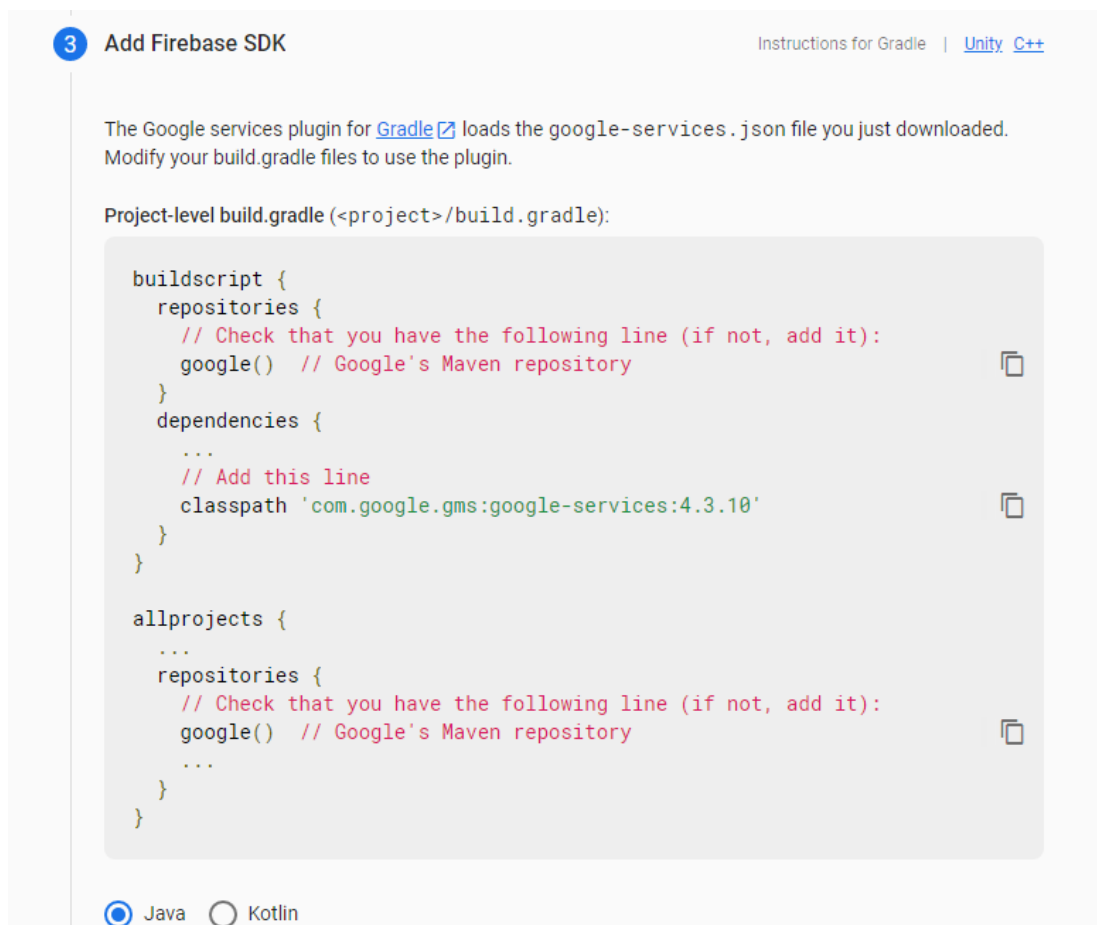


Рисунок 21 – Добавление описания в *Gradle*

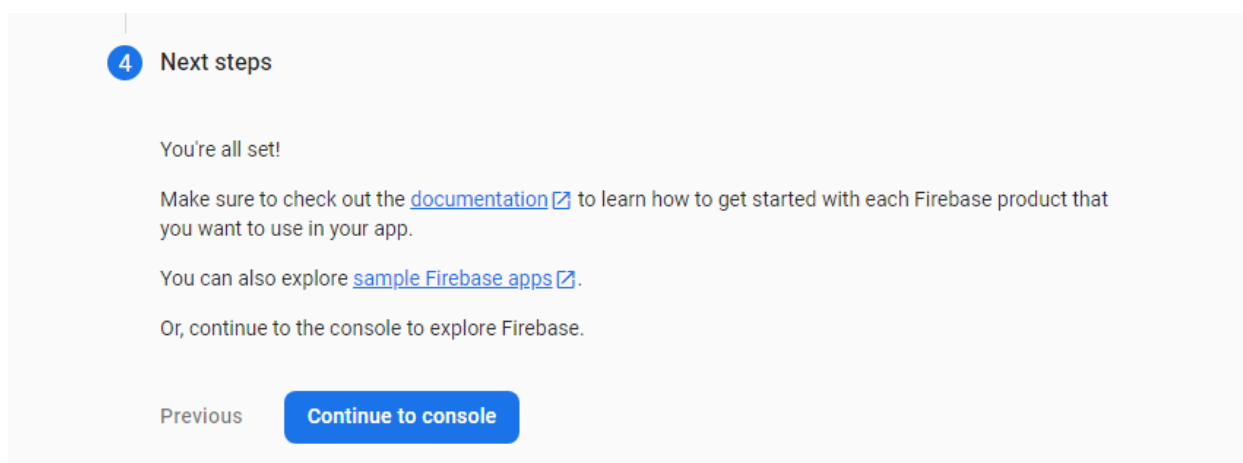


Рисунок 22 – Завершение подключения приложения

На рисунке 23 показана диаграмма классов. Она представляет собой упрощенную модель взаимодействия и иерархии элементов системы и их атрибутов.

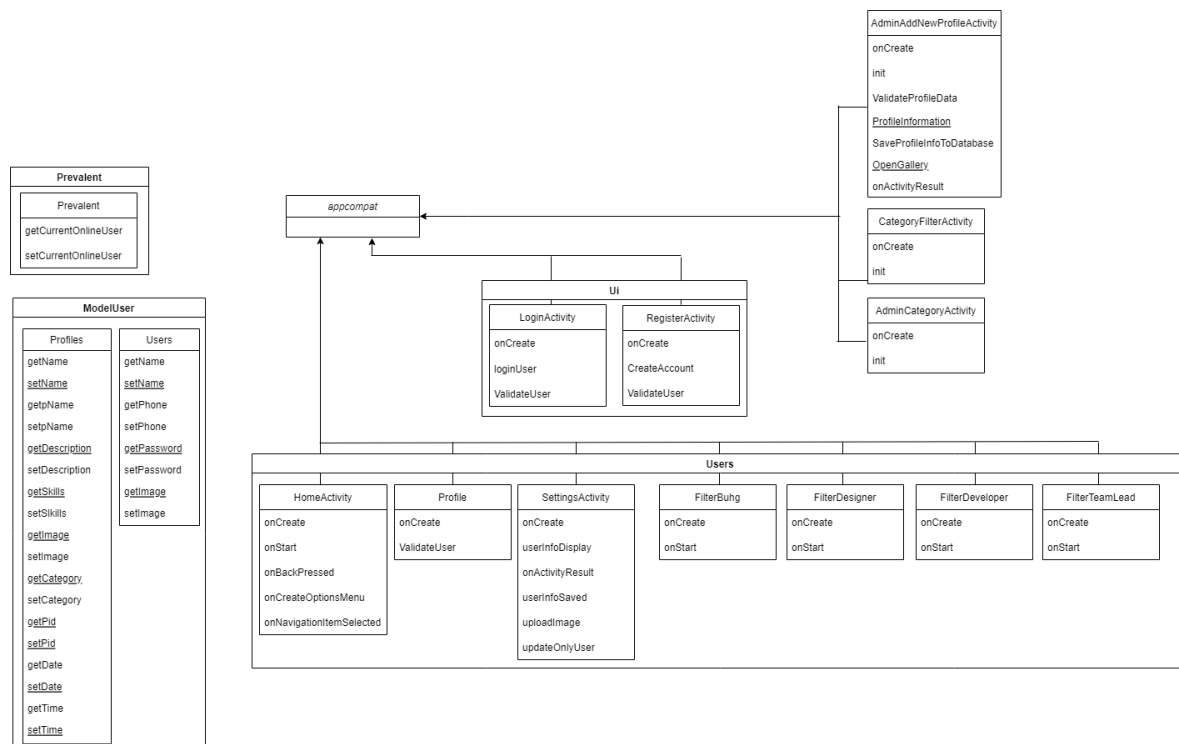


Рисунок 23 – Диаграмма классов

Рассмотрим *Java*-классы.

AppCompat –библиотека совместимости, поддерживающая перенос данных между версиями *Android*.

Profiles–класс, в котором объявлены внешние переменные для обработки информации о кандидатах:

- *Profiles* – объявление переменных с данными о кандидатах;
- *getName* – получение данных об имени кандидата;
- *setName* – занесение информации об имени кандидата;
- *getDescription* – получение данных о типе личности кандидата;
- *setDescription* – занесение информации о типе личности кандидата;
- *getSkills* – получение данных о навыках кандидата;
- *setSkills* – занесение информации о навыках кандидата;
- *getImage* – получение ссылки на изображение;
- *setImage* – добавление ссылки на изображение;
- *getCategory* – получение данных о предполагаемой должности кандидата;

- *setCategory* – внесение в должность кандидата;
- *getPid* – получение идентификатора кандидата;
- *setPid* – добавление идентификатора кандидата;
- *getTime* – получение данных о времени добавления профиля в базу;
- *setTime* – добавление данных о времени внесения профиля в базу;
- *getDate* – получение данных о дате добавления профиля в базу;
- *setDate* – добавление данных о дате внесения профиля в базу.

Users – класс, в котором объявлены внешние переменные для обработки данных о пользователях:

- *Users* – объявление переменных с данными о пользователях;
- *getName* – получение данных об имени пользователя;
- *setName* – добавление данных об имени пользователя;
- *getPhone* – получение данных о номере телефона пользователя;
- *setPhone* – добавление данных о номере телефона пользователя;
- *getPassword* – получение данных о пароле пользователя;
- *setPassword* – добавление пароля пользователя;
- *getImage* – получение ссылки на фото пользователя;
- *setImage* – добавление ссылки на фото пользователя.

Prevalent – класс, связующий пользовательский класс и хранящий данные о верифицированном пользователе.

HomeActivity – класс, в котором содержатся данные о работе начальной страницы при входе от обычного пользователя:

- *onCreate* – связь переменных с объектами xml файла *activity_home*, объявление связи с базой *Firebase*, получение информации о классе пользователя;
- *onStart* – создание списка из данных базы, вызов функции вывода профилей на экран;
- *onBackPressed* – возврат на начальный экран
- *onCreateOptionsMenu* – получение информации о списке меню и

связь с xml файлом home;

- *onNavigationItemSelected* – вывод категорий меню, объявление переходов на другие страницы, внесение в поток данных о пользователе, при переходе на страницы, где происходит взаимодействие с ним.

Profile – класс первоначальной страницы входа в систему, предоставляющий выбор регистрации или авторизации:

onCreate – инициализация объектов *xml*, функции перехода на страницы регистрации и входа;

ValidateUser – проверка верификации пользователя, если он уже существует, то переход сразу на страницу *HomeActivity*.

SettingsActivity – класс редактирования данных о пользователе:

- *onCreate* – инициализация объектов *xml*, ссылки на функции сохранения изменений и выхода, вызов функции изменения изображения;

- *userInfoDisplay* – устанавливается связь с базой данных, запускается функция получения и занесения данных в базу;

- *uploadImage* – загрузка изображения из хранилища телефона;

- *onActivityResult* – получение ссылки на изображение из базы;

- *userInfoSaved* – вывод сообщений о заполнении форм и запуск функции загрузки изображения;

- *updateOnlyUser* – сохранение изменений без учета изображения.

AdminAddNewProfileActivity – класс, отвечающий за добавление нового пользователя в базу данных:

- *onCreate* – вызов функций инициализации, открытия галереи телефона и верификации профиля;

- *init* – инициализация объектов *xml*;

- *ValidateProfileData* – передача данных из формы в переменную и запуск функции внесения в базу;

- *ProfileInformation* – получение даты и времени в системе, установка связи с базой, постановка задачи, загрузка изображения в хранилище и получение ссылки на изображение в хранилище:

- *SaveProfileInfoToDatabase* – загрузка в базу данных о кандидате;
- *OpenGallery* – запуск галереи телефона;
- *onActivityResult* – получение ссылки на изображение.

AdminCategoryActivity – класс, определяющий переход администратора на страницу добавления пользователя:

- *onCreate* – вызов функции инициализации, запуск функций открытия страницы по нажатию на изображение;
- *init* – функция инициализации *xml* объектов на странице.

CategoryFilterActivity – класс, отвечающий за переход на страницу отфильтрованных анкет на определенную должность:

- *onCreate* – вызов функции инициализации, запуск функций открытия страницы по нажатию на изображение;
- *init* – функция инициализации *xml* объектов на странице.

FilterBuhg – класс, отвечающий за вывод отфильтрованных анкет на должность бухгалтера:

- *onCreate* – инициализация *xml* объектов на форме;
- *onStart* – создание списка из данных базы, фильтрация списка.

RegisterActivity – класс, отвечающий за ввод данных пользователем и регистрации этой информации в базе:

- *onCreate* – инициализация объектов *xml* на форме, запуск функции создания аккаунта;
- *CreateAccout* – получение введенных в поля формы значений и занесение их в переменные;
- *ValidatePhone* – подключение к базе данных, вызов функции изменения базы и если не существует пользователя с введенными значениями, тогда в записываются введенные значения.

LoginActivity – класс, отвечающий за авторизацию пользователя в приложении:

- *onCreate* – инициализация объектов *xml*, обработчики нажатия для выбора входа от администратора и пользователя;

- *loginUser* – получение данных введенных в форме, вызов сообщения об успешном или некорректном вводе, вызов функции подтверждения существования пользователя;

- *ValidateUser* – проверка постановления галочки запоминания, если установлена, то занесение в список, установление связи с базой, вызов функции изменения базы.

FilterDevelop – класс, отвечающий за вывод отфильтрованных анкет на должность разработчика:

- *onCreate* – инициализация *xml* объектов на форме;
- *onStart* – создание списка из данных базы, фильтрация списка.

FilterTeamLead – класс, отвечающий за вывод отфильтрованных анкет на должность руководителя проекта:

- *onCreate* – инициализация *xml* объектов на форме;
- *onStart* – создание списка из данных базы, фильтрация списка.

FilterDesigner – класс, отвечающий за вывод отфильтрованных анкет на должность дизайнера:

- *onCreate* – инициализация *xml* объектов на форме;
- *onStart* – создание списка из данных базы, фильтрация списка.

Для демонстрации данных требуется визуальное оформление проекта, которое было реализовано на языке разметки *xml* и связано с классами приложения, которое представлено на диаграмме классов *xml* (рисунок 24).

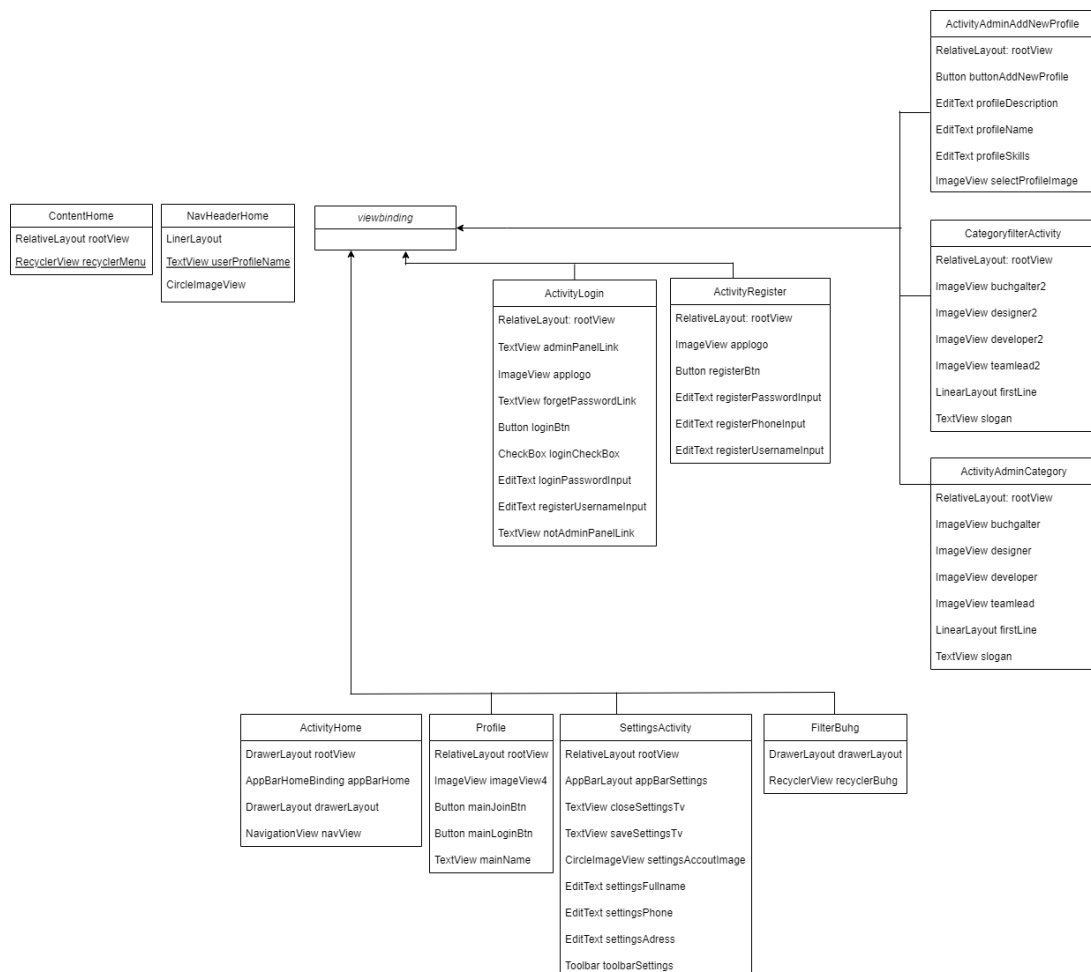


Рисунок 24 – Диаграмма классов *xml* объектов

Рассмотрим *xml*-классы приложения.

ViewBinding – включает в себя все ссылки на *xml* представления.

ActivityHome – *xml* форма класса *HomeActivity* (рисунок 25):

- *DrawerLayout rootView* – контейнер компонентов;
- *AppBarHome appBarHome* – компонент всплывающего маркера;
- *DrawerLayout drawerLayout* – контейнер навигации;
- *NavigationView navView* – компонент окна навигации.

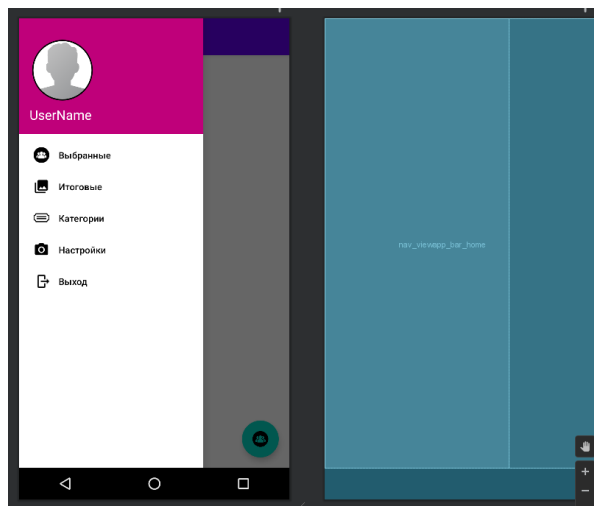


Рисунок 25 - *HomeActivity*

CategoryfilerActivity – xml форма класса *CategoryFilterActivity* (рисунок 26):

- *RelativeLayout rootView* – контейнер компонентов;
- *ImageView buchgalter2* – компонент изображения перехода на страницу кандидатов на должность бухгалтера;
- *ImageView designer2* – компонент изображения перехода на страницу кандидатов на должность дизайнера;
- *ImageView developer2* – компонент изображения перехода на страницу кандидатов на должность разработчика;
- *ImageView teamlead2* – компонент изображения перехода на страницу кандидатов на должность руководителя проекта;
- *LinearLayout firstline* – контейнер для слогана приложения;
- *TextView slogan* – текстовый компонент для представления слогана.

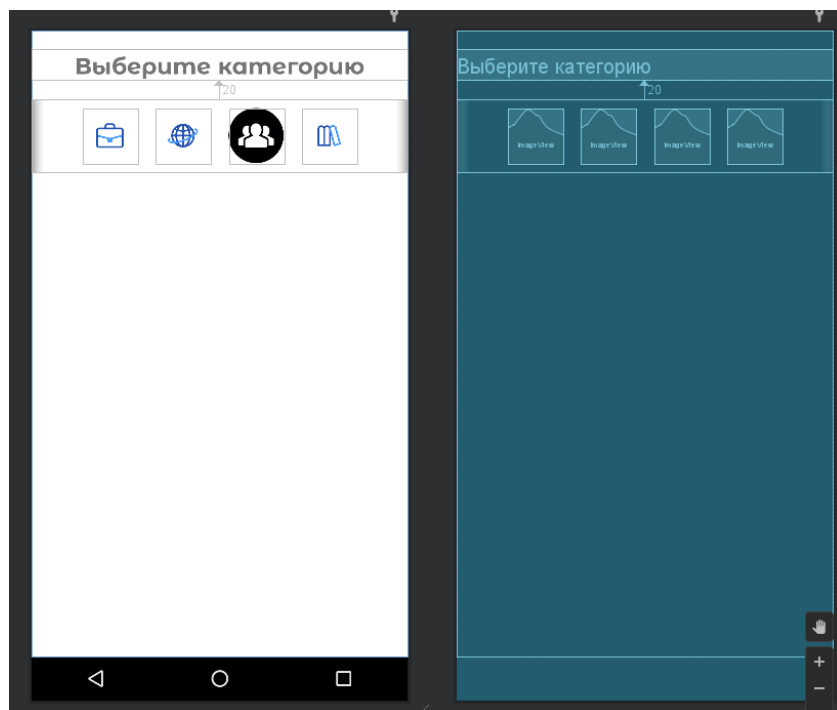


Рисунок 26 – *CategoryFilterActivity*

ActiveAdminAddNewProfile – XML форма класса

AdminAddNewProfileActivity (рисунок 27):

- *RelativeLayout* – контейнер компонентов;
- *Button buttonAddNewProfile* – кнопка записи данных в базу;
- *EditText profileDescription* – форма ввода данных о характеристике;
- *EditText profileName* – форма ввода данных об имени кандидата;
- *EditText profileSkills* – форма ввода данных о навыках кандидата;
- *ImageView selectProfileImage* – компонент отображения и

добавления изображения кандидата.

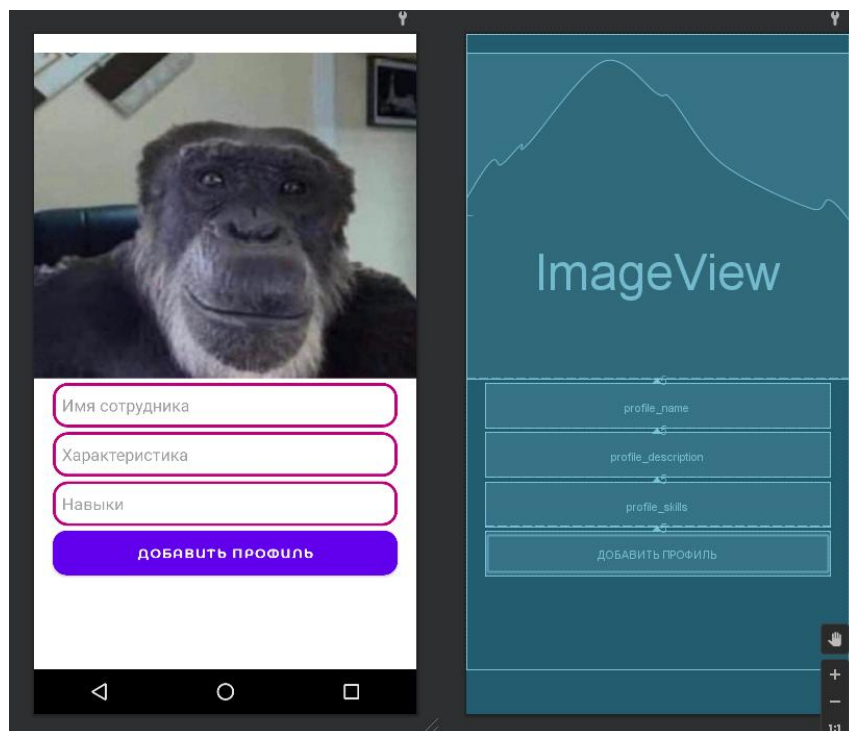


Рисунок 27 - *AdminAddNewProfileActivity*

ActivityAdminCategory – XML форма класса *AdminCategoryActivity* (рисунок 28):

- *RelativeLayout rootView* – контейнер компонентов;
- *ImageView buchgalter* – компонент изображения добавления профиля кандидатов на должность бухгалтера;
- *ImageView designer* – компонент изображения добавления профиля кандидатов на должность дизайнера;
- *ImageView developer* – компонент изображения добавления профиля кандидатов на должность разработчика;
- *ImageView teamlead* – компонент изображения добавления профиля кандидатов на должность руководителя проекта;
- *LinearLayout firstline* – контейнер для слогана приложения;
- *TextView slogan* – текстовый компонент для представления слогана.

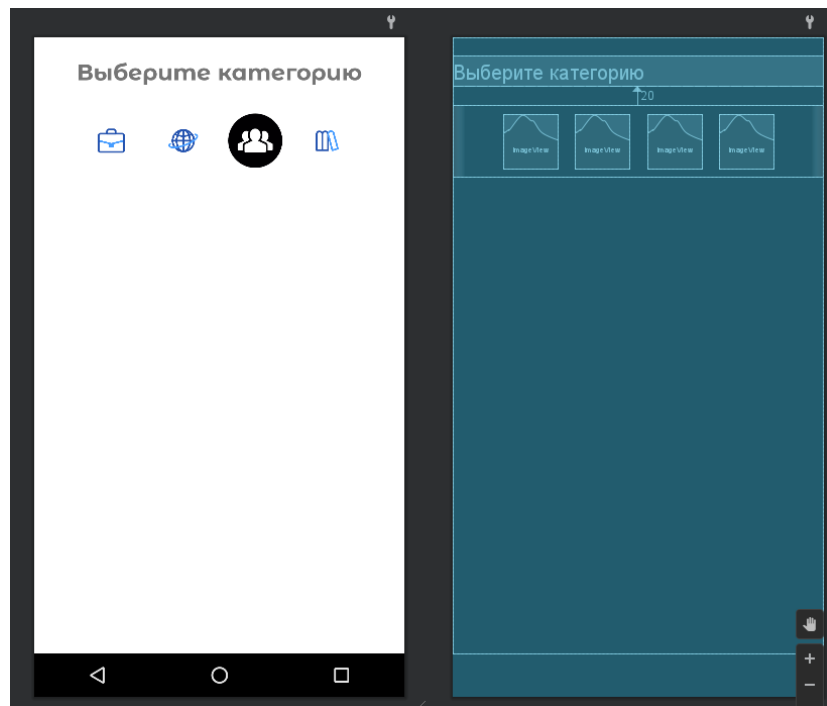


Рисунок 28 – *ActiveAdminCategory*

ContentHome – XML форма класса *HomeActivity* (рисунок 29):

- *RelativeLayout rootView* – контейнер компонентов;
- *RecyclerView recyclerMenu* – группа представлений контента страницы.

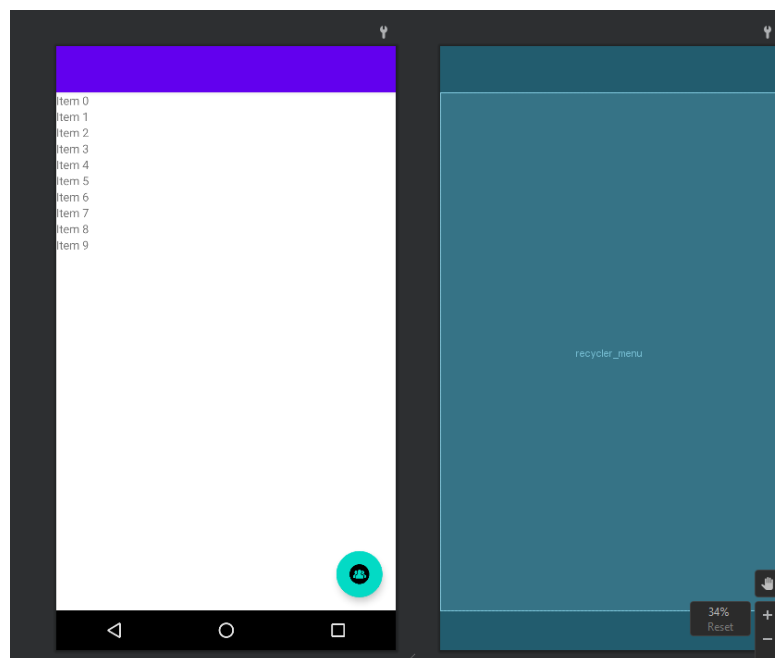


Рисунок 29 – *ContentHome*

ActivityLogin – xml форма класса *LoginActivity* (рисунок 30):

- *RelativeLayout rootView* – контейнер компонентов;
- *TextView adminPanelLink* – текстовый компонент для перехода в вид ввода от администратора;
- *ImageView applogo* – компонент изображения для представления логотипа;
- *TextView forgotPasswordLink* – текстовый компонент указывающий на отсутствие пароля;
- *LinearLayout linerCheck* – контейнер для хранения компонентов заголовка;
- *Button loginBtn* – кнопка ввода данных для авторизации;
- *CheckBox loginCheckBox* – отметка для постоянной верификации пользователя;
- *EditText loginPasswordInput* – компонент текстового ввода пароля пользователя;
- *EditText loginPhoneInput* – компонент текстового ввода номера телефона пользователя;
- *TextView notAdminPanelLink* – текстовый компонент для перехода в вид ввода от пользователя.

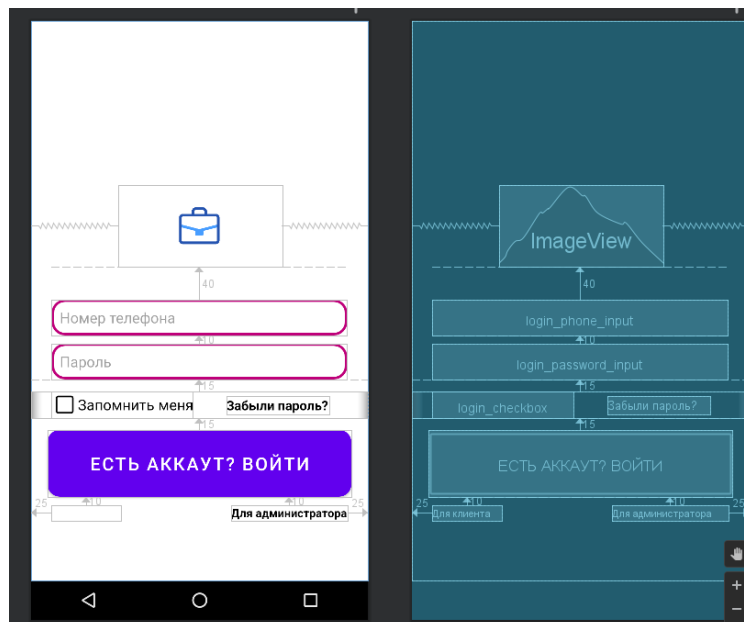


Рисунок 30 – *ActivityLogin*

ActiveRegister – XML форма класса *RegisterActivity* (рисунок 31):

- *RelativeLayout rootView* – контейнер компонентов;
- *ImageView applogo* – компонент изображения для представления логотипа;
- *Button registerBtn* – кнопка ввода данных для регистрации;
- *EditText registerPasswordInput* – компонент текстового ввода пароля пользователя;
- *EditText registerPhoneInput* – компонент текстового ввода номера телефона пользователя;
- *EditText registerUsernameInput* – компонент текстового ввода имени пользователя.

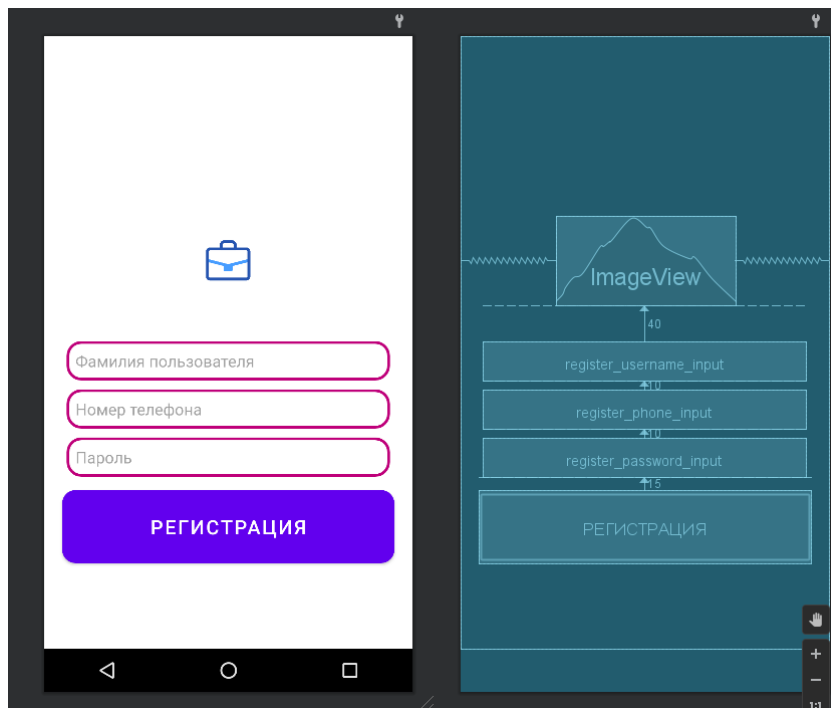


Рисунок 31 - *ActivityRegister*

ActivitySettings – XML форма класса *SettingsActivity* (рисунок 32):

- *RelativeLayout rootView* – контейнер компонентов;
- *AppBarLayout appBarSettings* – контейнер верхних кнопок;
- *TextView closeSettingsTv* – текстовый компонент для выхода со страницы;
- *TextView saveSettingsTv* – текстовый компонент для сохранения и выхода со страницы;
- *CircleImageView settingsAccountImage* – компонент для отображения изображения аккаунта;
- *EditText settingsAdress* – форма ввода адреса пользователей.
- *EditText settingsFullName* – форма ввода имени пользователя;
- *EditText settingsPhone* – форма ввода номера телефона;
- *Toolbar toolbarSettings* – контейнер компонентов перехода.

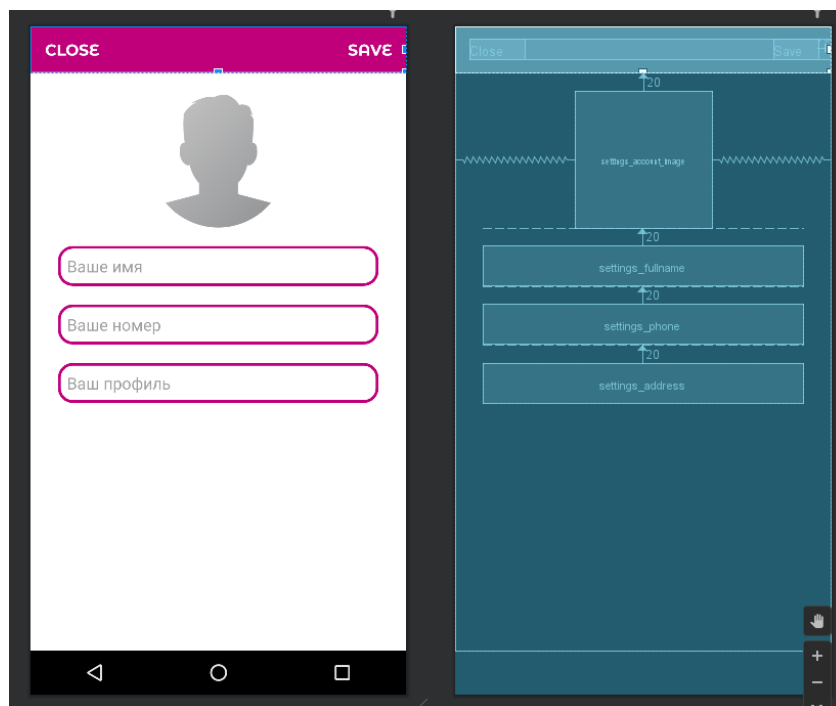


Рисунок 32 – ActivitySettings

NavHeaderHome – XML форма для отображения всплывающей панели на страницах (рисунок 33):

- *LinearLayout* – контейнер компонентов;
- *TextView userProfileName* – текстовый компонент для отображения имени пользователя;
- *CircleImageView settingsAccountImage* – компонент для отображения изображения аккаунта.

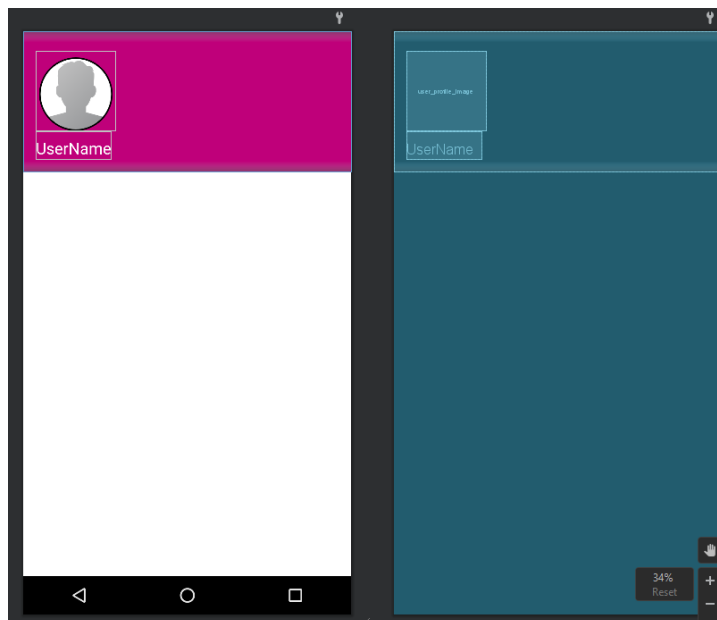


Рисунок 33 – *NavHeader*

Profile – XML форма класса *Profile* (рисунок 34):

- *RecyclerView rootView* – контейнер компонентов;
- *ImageView imageView4* – компонент изображения для отображения логотипа приложения;
- *Button mainJoinBtn* – кнопка для перехода на страницу регистрации;
- *Button mainLoginBtn* – кнопка для перехода на страницу авторизации;
- *TextView mainName* – текстовый компонент для отображения слогана приложения.

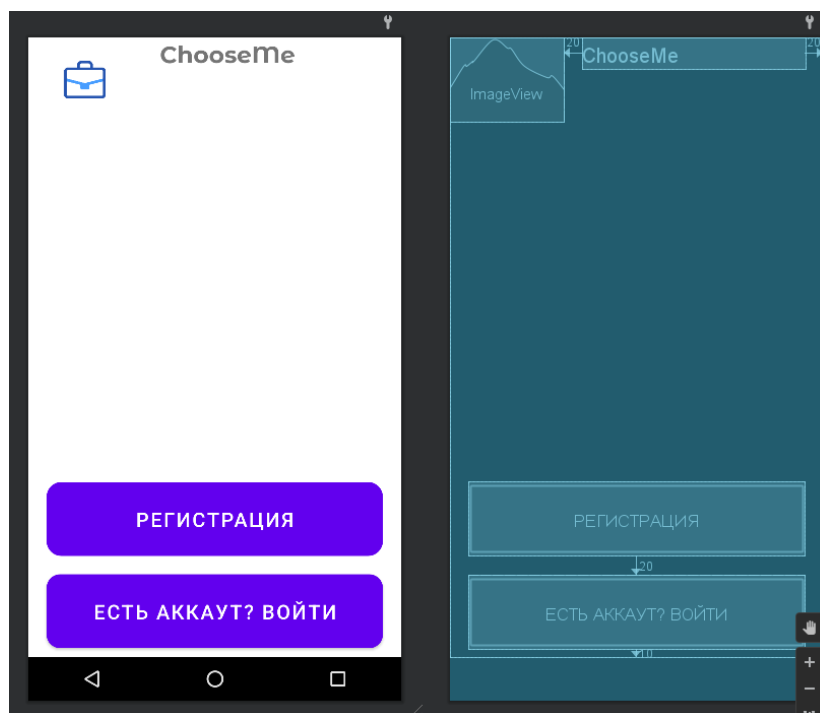


Рисунок 34 – Profile

FilterBuhg – XML форма класса *FilterBuhg* (Рисунок 35):

- *DrawerLayout drawerLayout* – контейнер компонентов;
- *RecyclerView recyclerBuhg* – компонент для отображения списка кандидатов.

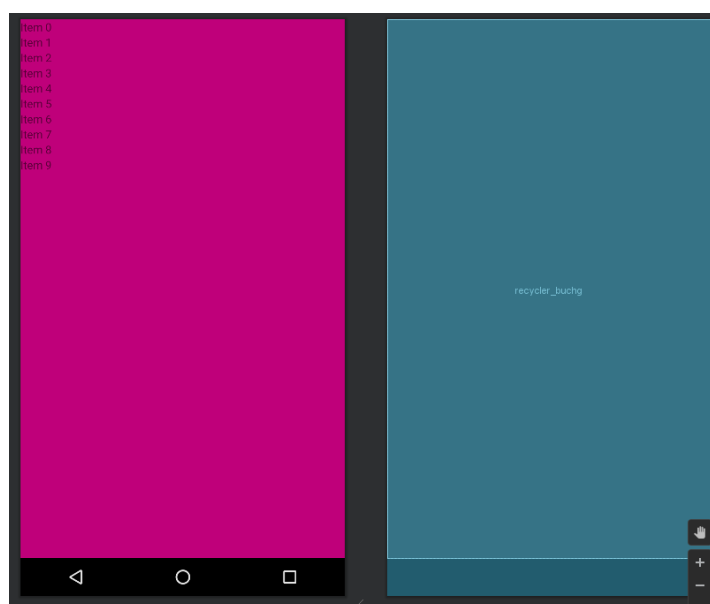


Рисунок 35 – FilterBuhg

1.6 Архитектура приложения

Приложение построено на основе разбития на модули, где каждый из них отвечает за свою зону ответственности. Это позволяет повысить понимание работы программы и структурировать добавляемые страницы.

Приложение состоит из нескольких основных модулей:

- *ModelUser* – отвечает за временное хранилище данных из базы и получение-передачу в пользовательские классы;
- *UI* - классы пользовательского интерфейса, которые отвечают непосредственно за взаимодействие с ним;
- *ViewHolder* – модуль построения структуры профиля;
- *Prevalent* – отвечает за хранение данных о верифицированном пользователе;
- *CloudServices* – внешнее хранилище данных, которое состоит из БД *Firebase*;
- *Layout* – содержит в себе файлы, отвечающие за визуальное представление приложения.

Взаимодействие модулей строится таким образом, что классы *UI* обращаются к *xml* файлам *layout* для визуального взаимодействия с пользователем, в то же время, *Activity* обращаются к базе и получают или записывают данные, затем заносят их в статические методы *ModelUser*, для последующего обращения к ним. *Prevalent* так же использует статические методы для проверки данных о верификации в классах *UI* (рисунок 36).

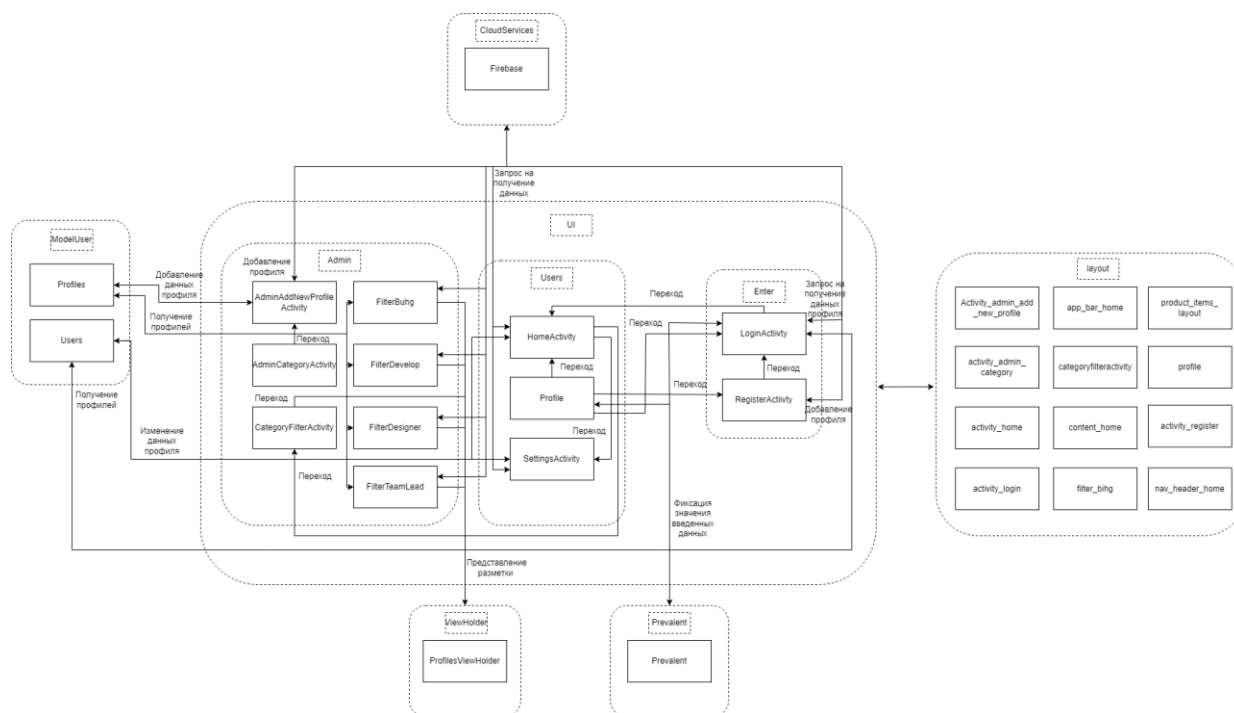


Рисунок 36 – Взаимодействие модулей приложения.

1.7 Тестирование и отладка

Для тестирования программы был использован встроенный эмулятор в среду разработки *Android Studio*, запущенную на ПК со следующими характеристиками:

- видеокарта: *AMD Radeon RX580*;
- процессор: *AMD Ryzen 7 3700X 4.30 GHz*;
- ОЗУ: *16 GB*;
- ОС: *Windows 10 X64*;
- память: не менее *30 GB*.

Тест-кейс №1. Создание учетной записи и выбор кандидатов на должность бухгалтера.

Шаги:

- регистрация учетной записи;
- авторизация с новой учетной записью;
- переход на страницу категорий;
- открытие категории «Бухгалтеры» и успешный вывод списка

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.09.05.01.05.001 ПЗ

Лист

49

кандидатов.

Ожидаемый результат – Успешное выполнение всех действий.

Для тестирования приложения необходимо запустить эмулятор мобильного устройства (рисунок 37).

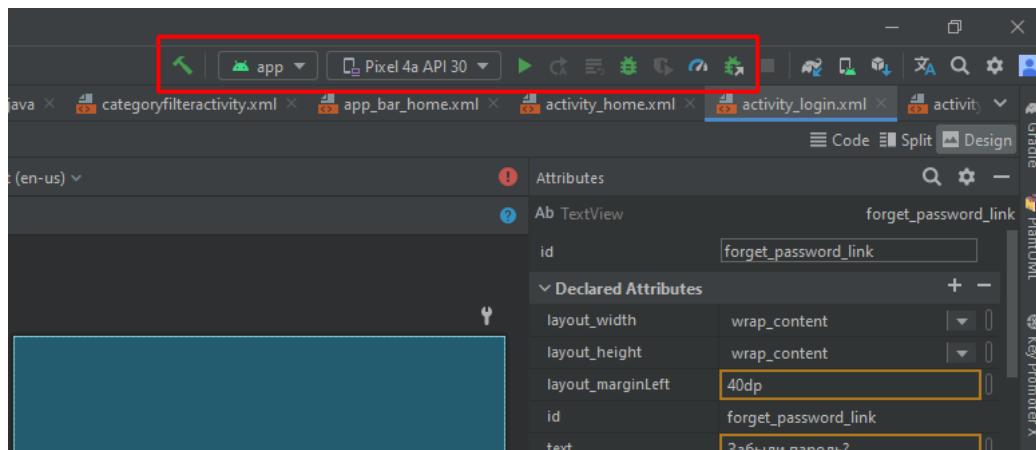


Рисунок 37 – Запуск эмуляции устройства.

При запуске эмулятора будет представлен начальный экран входа в приложение, на котором будут находиться две кнопки – регистрации авторизации пользователя (рисунок 38).

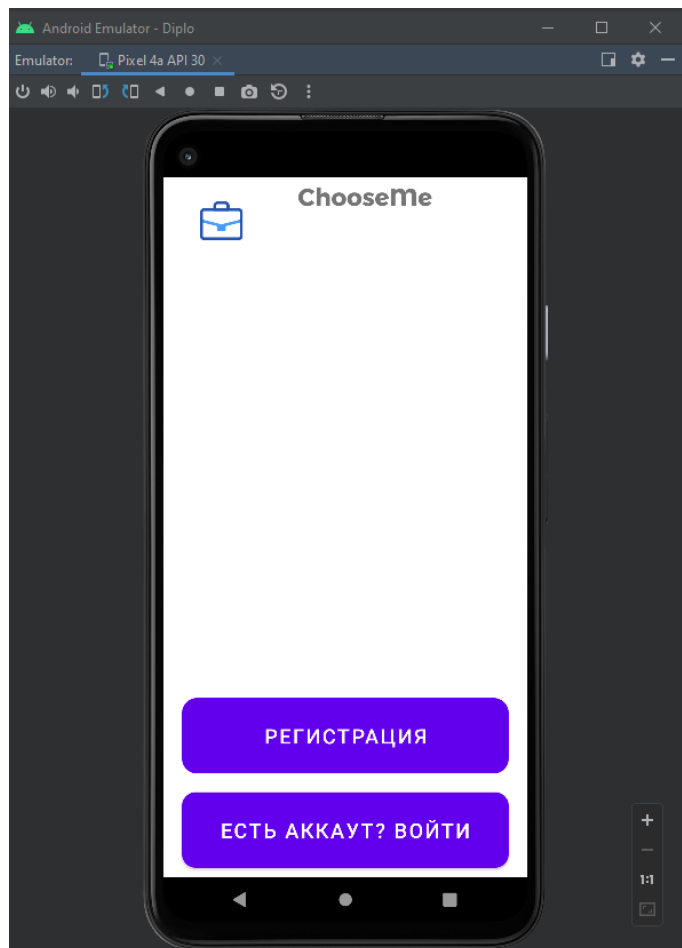


Рисунок 38 – Первоначальный экран.

При нажатии на кнопку «Регистрация» будет выполнен переход на страницу регистрации пользователя, где имеется возможность ввести фамилию, номер телефона и пароль для авторизации (рисунок 39). Введенные значения заносятся в *NoSql* базу *Firebase* по нажатию кнопки регистрации (рисунок 40) Если аккаунт уже существует, то будет выдано соответствующее сообщение об ошибке и выполнен переход на страницу авторизации (рисунок 41).

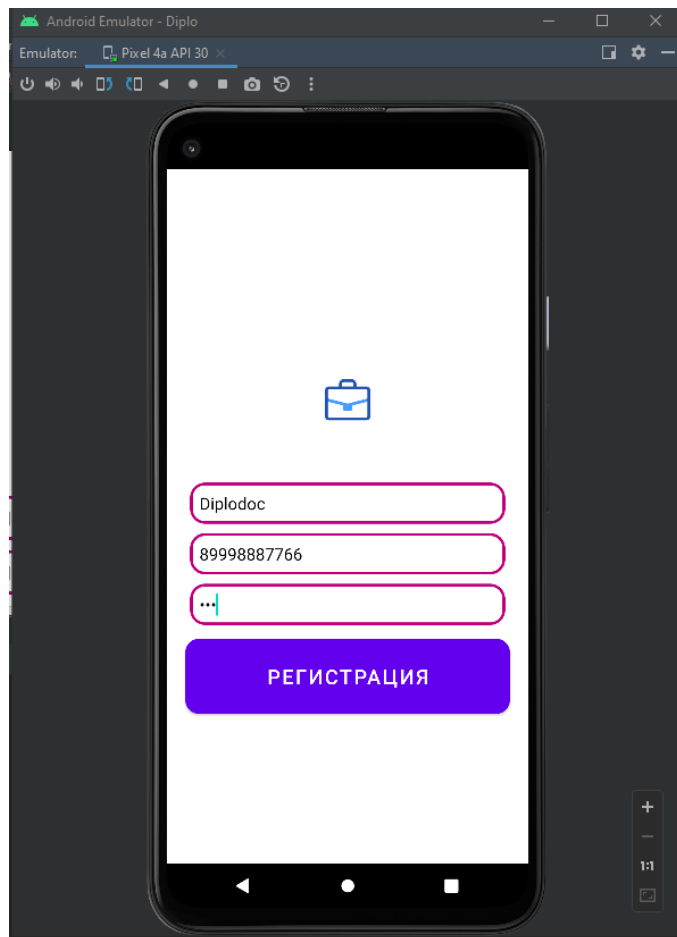


Рисунок 39 – Страница регистрации пользователя

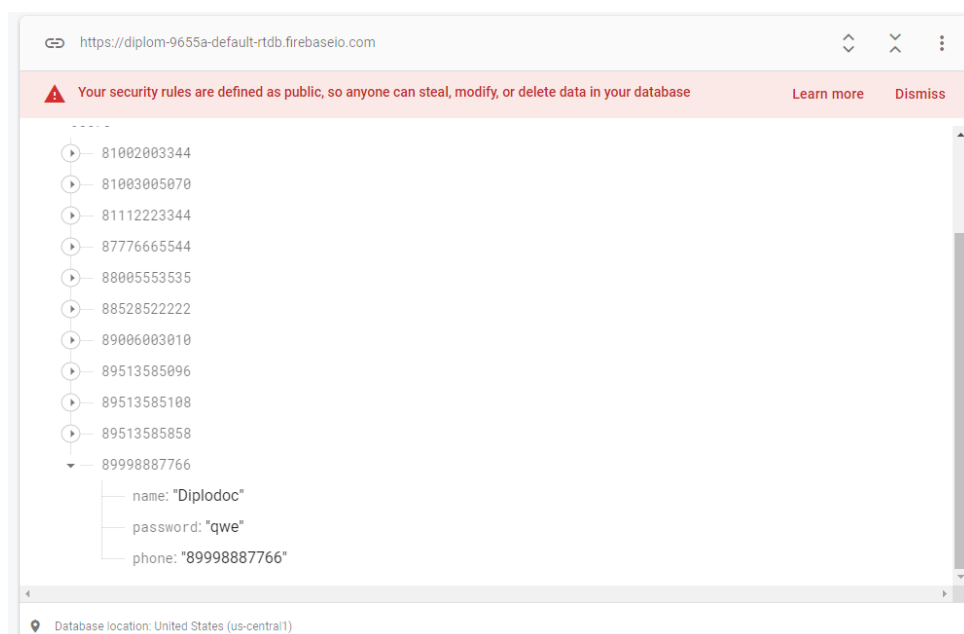


Рисунок 40 – Данные о пользователе в *Firestore*

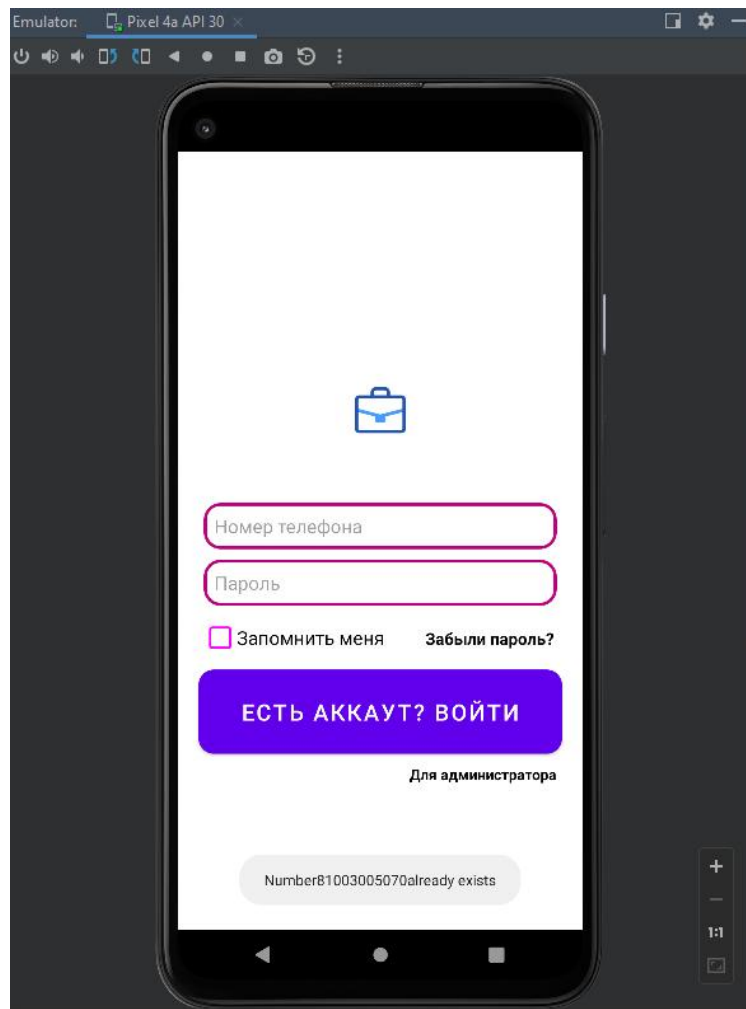


Рисунок 41 – Переход на страницу авторизации пользователя

В случае нажатия на начальном экране (рисунок 38) кнопки «Есть аккаунт? Войти» будет выполнен переход на страницу авторизации (рисунок 40). Вход в аккаунт возможен, как от администратора, так и от пользователя. Изначально дан вход от пользователя, для того, что бы авторизоваться, необходимо ввести номер телефона и пароль, а затем нажать на кнопку «Есть аккаунт? Войти», после чего будет открыта главная страница пользователя (рисунок 43), так же, переход на эту страницу будет осуществляться автоматически при запуске приложения, если выбрать галочку «Запомнить меня». Для того, что бы войти от имени администратора, необходимо нажать на текст «Для администратора», после чего интерфейс визуально изменится (рисунок 42), после чего так же ввести данные, а затем нажать на кнопку «Вход администратора», после чего будет открыта страница выбора категорий

(рисунок 48). Для возвращения к входу от пользователя, надо нажать на текст «Для клиента» (рисунок 42).

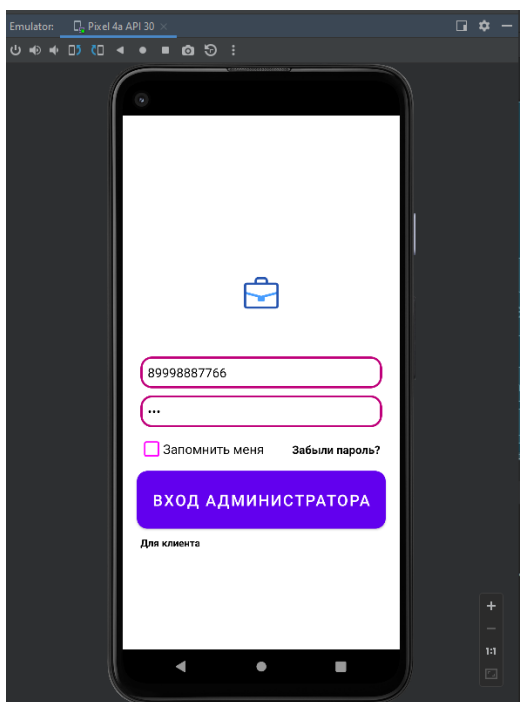


Рисунок 42 – Авторизация от имени администратора

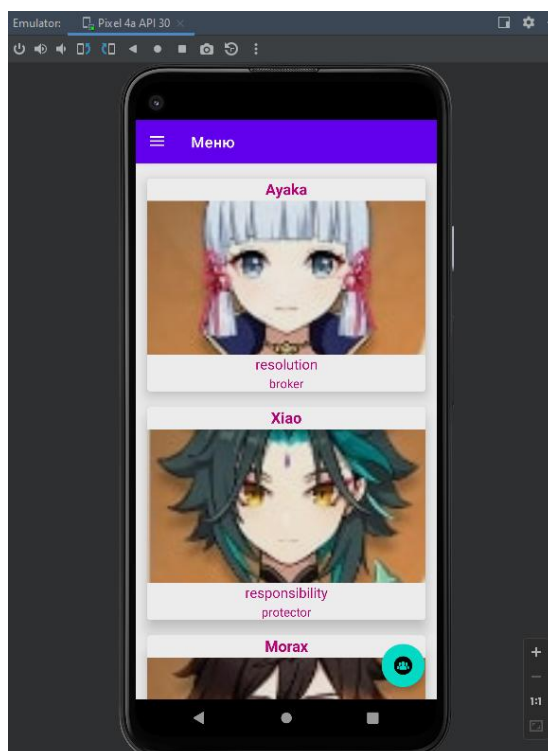


Рисунок 43 – Главная страница пользователя

На главной странице представлены профили кандидатов, у которых указаны их основные профессиональные навыки и характеристика личности по Майер-Бриггс, для перехода в меню необходимо нажать на правую верхнюю кнопку, тогда откроется боковая панель, на которой возможно выбрать один из пунктов (рисунок 44). При нажатии на пункт «Выход» будет выход из профиля в первоначальный экран (рисунок 38), а так же выход из режима запоминания, в случае если во время авторизации была поставлена галочка. В случае нажатия на пункт «Категории», будет открыта страница категорий профилей кандидатов (рисунок 45).

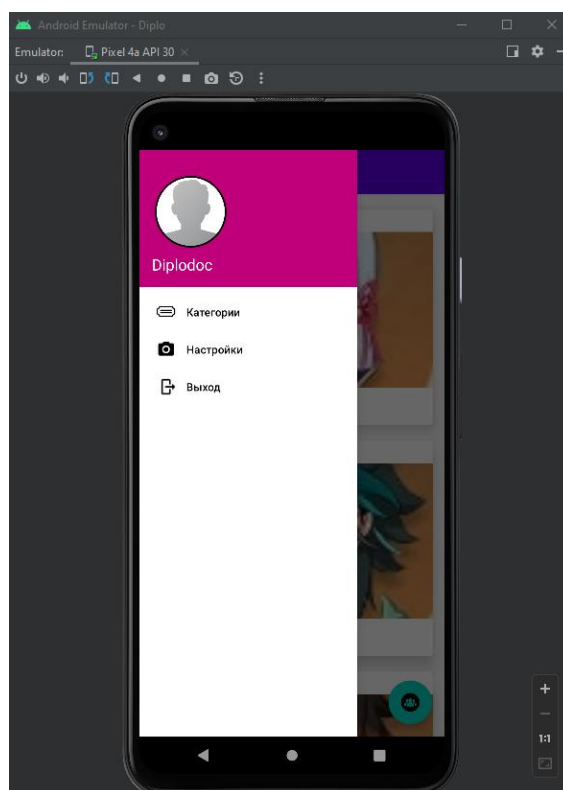


Рисунок 44 – Боковое меню на главной странице

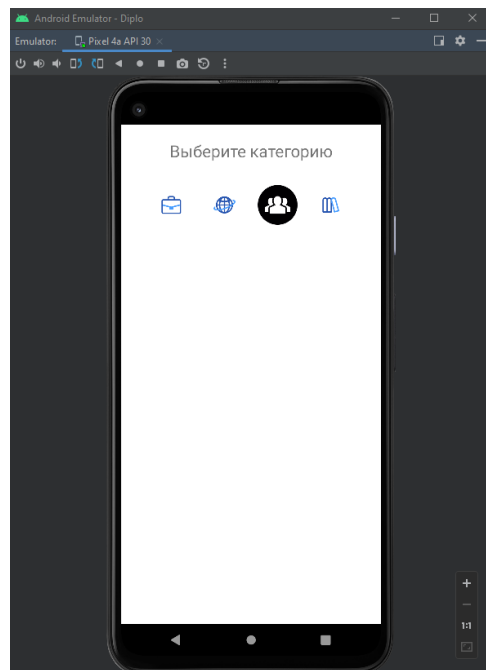


Рисунок 45 – Страница выбора фильтра по категориям

При открытии страницы категорий, есть возможность нажать на одну из предложенных кнопок и тогда будет совершен переход на страницу с отфильтрованными профилями кандидатов на определенную должность (рисунок 48). Фильтрация происходит за счет алгоритма, основанного на условиях, где на должность проходят те пользователи, которые отвечают заданным требованиям. Таким образом, устанавливается связь с базой, а затем ставится условие, по которому на должность бухгалтера проходят те кандидаты, которые отвечают либо характеру личности «защитник» или навыку «полемики», затем в элементы паттерна *holder* выводятся те профили, которые отвечают заданному требованию, а иные скрываются (рисунок 47).

```

@Override
protected void onStart() {
    super.onStart();
    FirebaseRecyclerOptions<Profiles> options = new FirebaseRecyclerOptions.Builder<Profiles>()
        .setQuery(ProductsRef, Profiles.class).build();
    FirebaseRecyclerAdapter<Profiles, ProfilesViewHolder> adapter = new FirebaseRecyclerAdapter<Profiles, ProfilesViewHolder>(options) {
        @Override
        protected void onBindViewHolder(@NonNull ProfilesViewHolder holder, int position, @NonNull Profiles model) {
            if (model.getDescription().equals(poemicist) || model.getDescription().equals(protector))
            {
                holder.txtProfileName.setText(model.getName());
                holder.txtProfilePrice.setText(model.getSkills());
                holder.txtProfileDescription.setText(model.getDescription());
                Picasso.get().load(model.getImage()).into(holder.imageView);
            } else {
                holder.txtProfileName.setVisibility(View.GONE);
                holder.txtProfileDescription.setVisibility(View.GONE);
                holder.txtProfilePrice.setVisibility(View.GONE);
                holder.imageView.setVisibility(View.GONE);
            }
        }
    }
}

```

Рисунок 47 – Код функции фильтрации профилей

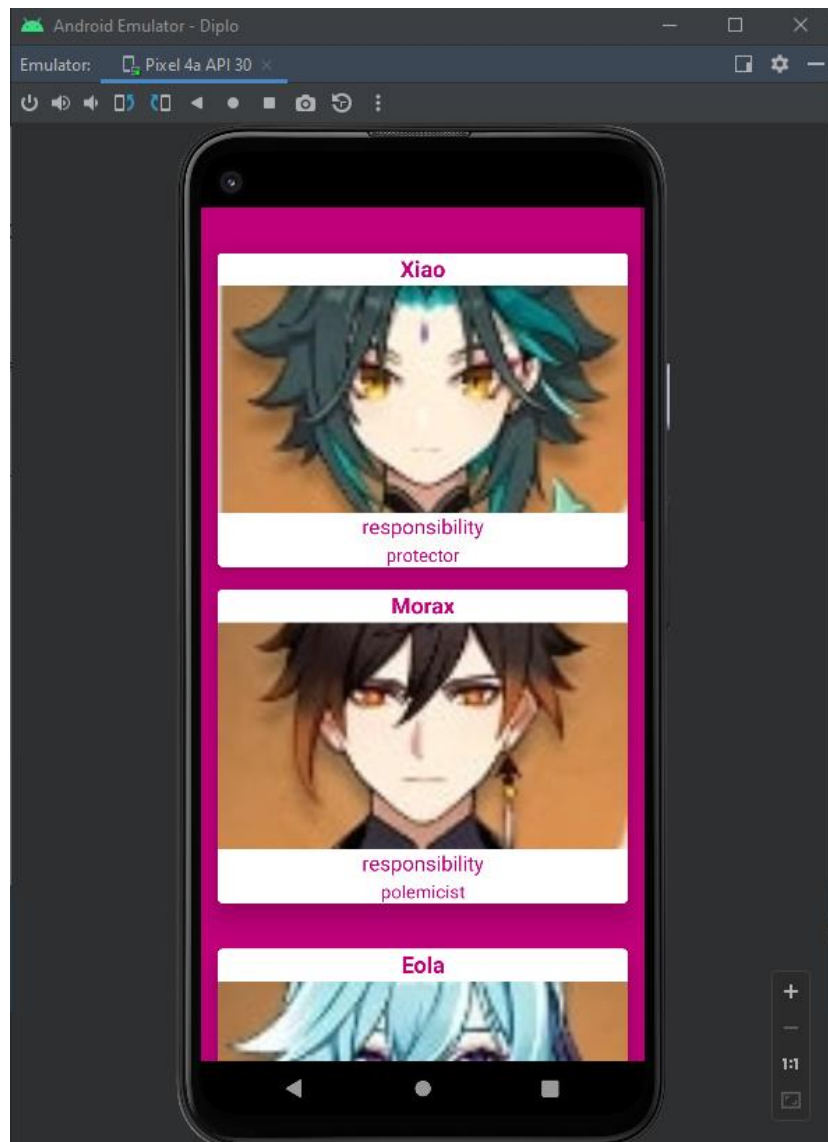


Рисунок 48 – Фильтр кандидатов на должность бухгалтера.

Тест-кейс №2. Добавление нового профиля кандидата.

Шаги:

- авторизация в системе, как администратор;
- переход на страницу добавления профиля;
- создание профиля «Аяка»;
- отображение введенных данных в БД *Firebase*.

Ожидаемый результат – успешное выполнение заданных действий.

Выполняется переход на страницу авторизации и нажимается кнопка «Для администратор», таким образом, страница переходит в режим авторизации администратора (рисунок 42). Когда происходит переход на главную страницу от администратора (рисунок 49), существует только возможность добавления новых профилей кандидатов. Для этого надо нажать на любую из кнопок и будет выполнен переход на страницу добавления нового профиля (рисунок 50).

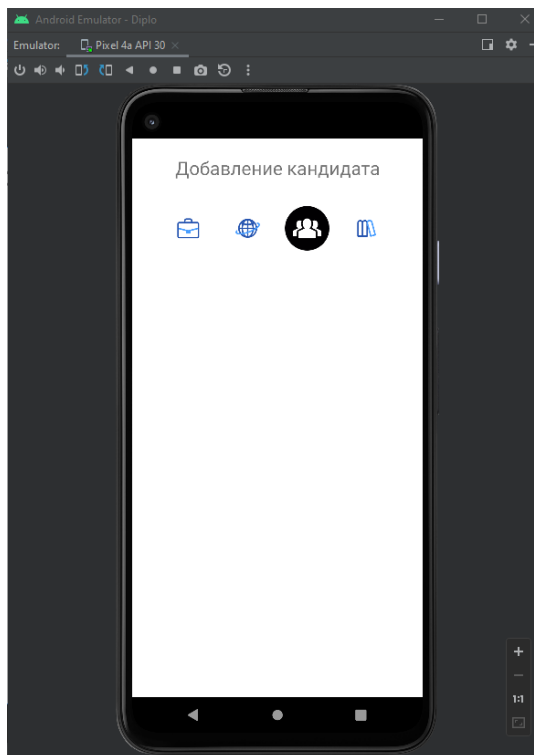


Рисунок 49 – Главная страница добавления кандидатов.

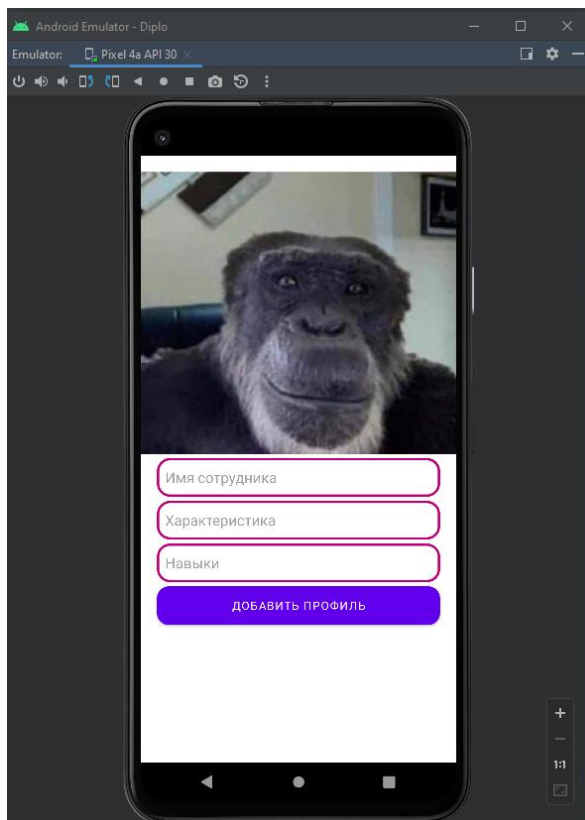


Рисунок 50 – Страница ввода данных профиля

На странице ввода данных (рисунок 50), необходимо ввести имя сотрудника, его характеристику личности по Майер-Бриггс и основной профессиональный навык. Кроме того, необходимо нажать на изображение, тогда откроется память телефона, где в хранилище следует найти подходящее фото для профиля кандидата и выбрать его (рисунок 51). Затем следует нажать кнопку «Добавить профиль» и введенные данные будут занесены в БД *Firebase* (рисунок 52).

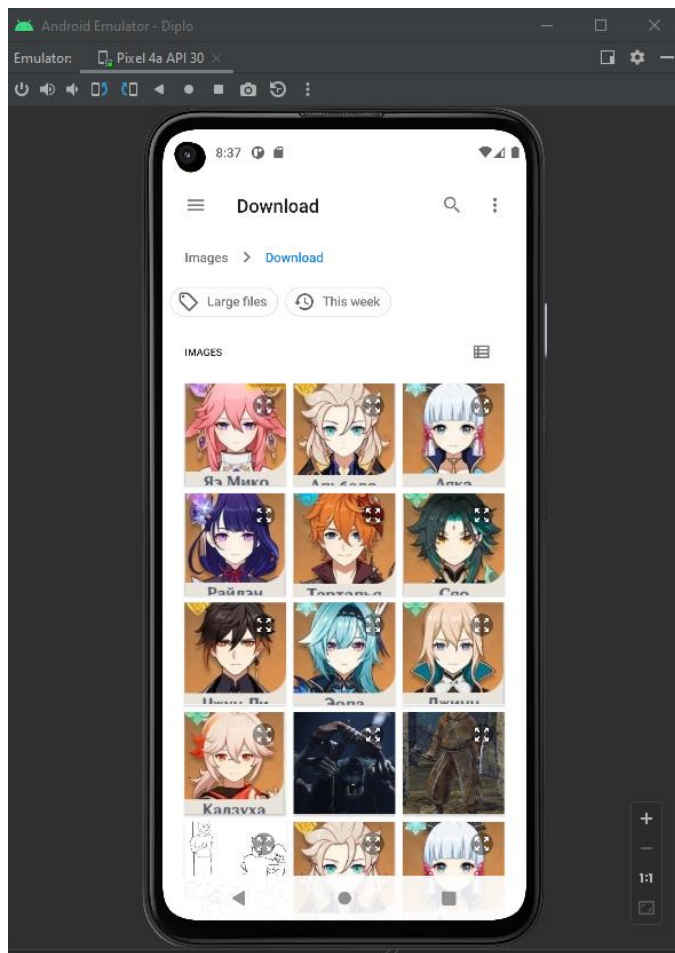


Рисунок 51 – Добавление фото из памяти телефона.

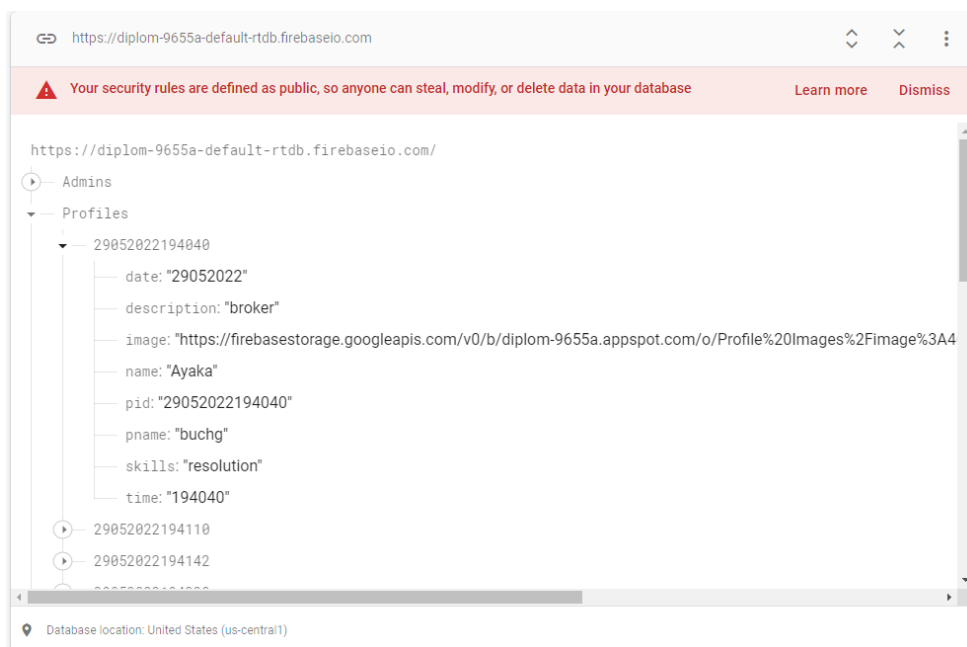


Рисунок 52 – Данные о кандидате в *Firebase*.

Таким образом, приложение выполняет исправно все свои функции, такие как регистрация пользователя, авторизация, добавление профилей кандидатов, фильтрация профилей в соответствии с требованиями для должности. Связь с БД стабильная, приложение работает плавно, без ошибок.

					<i>ПГУ 1.09.05.01.05.001 ПЗ</i>	<i>Лист</i>
<i>Изм.</i>	<i>Лист</i>	<i>№ докум.</i>	<i>Подпись</i>	<i>Дата</i>		61

2 Технико-экономическое обоснование разработки программы

2.1 Введение

Целью технико-экономического обоснования разработки является количественное и качественное доказательство экономической целесообразности разработки и усовершенствования программы, а также определение организационно-экономических условий ее эффективного функционирования. Эффективность программного модуля определяется его качеством и эффективностью процесса разработки и сопровождения. Качество программного изделия определяется тремя составляющими:

- с точки зрения пользователя данного программного продукта;
- с позиции использования ресурсов и их оценки;
- по выполнению требований на программное изделие.

Программное изделие должно быть разработано так, чтобы оно выполняло свои функции без лишних затрат ресурсов (оперативной памяти ПК, машинного времени, пропускной способности каналов передачи данных и др. – на стадии функционирования; времени разработки и денежных ресурсов – на стадии использования программного изделия). Содержание технико-экономического обоснования разработки программы заключается в следующем:

- расчет технико-экономических показателей и выбор базы сравнения;
- определение трудоемкости и стоимости программного обеспечения;
- расчет цены ПО;
- расчет капитальных и эксплуатационных затрат на разработку;
- определение показателей финансово-экономической эффективности.

Приведенные в данном разделе дипломного проекта результирующие таблицы, перечни данных, позволяют сопоставить результаты разработки и затраты на нее, чтобы сделать вывод об эффективности проекта. Исходные

данные для расчета экономических показателей приведены в таблице 2.

Таблица 2 - Исходные данные для расчета экономических показателей

Обозначение	Наименование показателя	Единицы измерения	Значение показателя		
СЭВМ	Стоимость ЭВМ	тыс. руб	120		
ДМ	Среднее количество рабочих дней в месяце	дни	25		
ρ_n	Норматив рентабельности	-	0,24		
ω_d	Коэффициент, учитывающий дополнительную заработную плату разработчика программы	-	0,2		
ω_c	Коэффициент, учитывающий начисления органам социального страхования	-	0,27		
ω_n	Коэффициент, учитывающий накладные расходы организации	-	1,35		
qI	Количество I-задач, решаемых потребителем	Зад. год	60		
$tM.B.I$	Время решения I-ой задачи разработанной программой	Маш. час	1		
nn	Количество организаций, которые приобретут данную программу	шт.	1г	2г	3г
			4	5	8
$ZЭЛ$	Тариф за 1 кВт/час	Руб.	3,44		
$ТС$	Срок службы разработанной программы	год	5		
$НДС$	Налог на добавленную стоимость	%	20		
TP	Количество рабочих дней в году	дн	264		
NCM	Количество смен работы ЭВМ	-	1		
tCM	Продолжительность смены	ч	8		
α	Простои ЭВМ	%	10		
P	Мощность, потребляемая ЭВМ	кВт	0,4		
NCP	Среднее количество ремонтов в год	-	2		
SD	Стоимость деталей, заменяемых при ремонте	Руб.	2500		

2.2 Расчёт затрат на разработку программного обеспечения

Суммарные затраты на разработку проекта рассчитываются по следующей формуле:

$$S_{PP} = S_{ЗП} + S_{НАК}, \quad (1)$$

где $S_{ЗП}$ – затраты по заработной плате инженера-программиста;

$S_{НАК}$ – накладные расходы.

Затраты по заработной плате инженера-программиста рассчитываются по формуле:

$$S_{ЗП} = O_{ЗП} * (1 + \omega_c) * (1 + \omega_d) * tp_i, \quad (2)$$

где $O_{ЗП}$ – основная заработная плата инженера-программиста за месяц (20000 руб.);

tp_i – время, необходимое для разработки программы программистом i -го разряда (чел.-мес.);

ω_d – коэффициент, учитывающий дополнительную заработную плату разработчика программы, в долях к сумме основной заработной платы;

ω_c – коэффициент, учитывающий начисления органам социального страхования на заработную плату разработчика программы, в долях к сумме основной заработной плате разработчика.

Система разрабатывалась 30 дней, если учесть, что в одном месяце 25 рабочих дней, то:

$$tp_{12} = 30/25 = 1,2 \text{ (чел./Мес)}. \quad (3)$$

Таким образом, затраты по заработной плате инженера-программиста составят:

$$S_{ЗП} = 10000 * (1 + 0,268) * (1 + 0,20) * 1,2 = 18258,52. \text{ (руб.)}.$$

Накладные затраты рассчитываются с учетом ω_n – коэффициента, определяющего уровень накладных расходов организации по формуле:

$$S_{НАК} = O_{ЗП} * \omega_n * tp_{12}, \quad (4)$$

$$S_{НАК} = 20000 * 1,35 * 1,2 = 32400 \text{ (руб.)}.$$

Таким образом, суммарные затраты на разработку системы составляют:

$$S_{РП} = 18258,52 + 32400 = 50658,52 \text{ (руб.)}.$$

2.3 Расчет цены разработанной программы

Оптовая цена разработанной программы определяется по следующей формуле:

$$Z_{П} = S_{РП} + П, \quad (5)$$

					ПГУ 1.09.05.01.05.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		64

где Z_{Π} – оптовая цена (цена разработчика) (руб.);

$S_{\Pi\Pi}$ – суммарные затраты на разработку программы (руб.);

Π – прибыль, рассчитанная по формуле:

$$\Pi = \rho_n * S_{\Pi\Pi}, \quad (6)$$

где ρ_n – норматив рентабельности, учитывающий прибыль организации, разрабатывающей данную программу в долях ко всем затратам данной организации на разработку программы.

Итак,

$$Z_{\Pi} = S_{\Pi\Pi} * (1 + \rho_n), \quad (7)$$

$$Z_{\Pi} = 50658,52 * (1 + 0,24) = 62815,92 \text{ (руб.)}.$$

Розничная цена программы рассчитывается с учетом налога на добавленную стоимость (НДС = 20%) по формуле:

$$Z_{\Pi p} = Z_{\Pi} + H_{\text{ДНС}} = Z_{\Pi} + (Z_{\Pi} * 20) / 100 = Z_{\Pi} * (1 + 0,20), \quad (8)$$

$$Z_{\Pi p} = 62815,92 * (1 + 0,20) = 75379,1 \text{ (руб.)}.$$

Выручка от продаж при условии n_{Π} – количество организаций, желающих приобрести программу, составит:

$$B = Z_{\Pi p} * n_n \quad (9)$$

$$B1 = 75379,1 * 4 = 301516,41 \text{ (руб.)},$$

$$B2 = 75379,1 * 5 = 376895,52 \text{ (руб.)},$$

$$B3 = 75379,1 * 8 = 603032,83 \text{ (руб.)}.$$

2.4 Расчет капитальных вложений

Капиталовложения, связанные с работой ЭВМ рассчитываются по формуле:

$$K_{\text{ЭВМ}} = C_{\text{ЭВМ}} + S_3, \quad (10)$$

где $C_{\text{ЭВМ}}$ – стоимость ЭВМ (руб.);

S_3 – стоимость запасных частей (руб.).

Итак, произведем расчет коэффициентов, входящих в формулу расчета величины капиталовложений:

					ПГУ 1.09.05.01.05.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		65

$$S_3 = 0,15 * C_{ЭВМ}; \quad (11)$$

$$S_3 = 0,15 * 120000 = 18000 \text{ (руб.)}.$$

Капиталовложения в ЭВМ и VR периферию составляют:

$$K_{ЭВМ} = 120000 + 18000 = 138000 \text{ (руб.)}.$$

2.5 Расчет эксплуатационных расходов

Эксплуатационные расходы на ЭВМ рассчитываются по формуле:

$$E = (T_{MB} * E_q) + Z_n/T_c, \quad (12)$$

где T_{MB} – машинное время для решения задач с помощью разработанной программы, (маш.час/год);

E_q – эксплуатационные расходы, приходящиеся на 1 час работы ЭВМ;

Z_n – цена, по которой продается программа (руб.);

T_c – срок службы программы (г).

Полезный фонд времени работы ЭВМ рассчитывается по формуле:

$$T_{пол} = T_{общ} * t_{см} * N_{см} * (1 - \alpha/100), \quad (13)$$

где $T_{общ}$ – общий фонд времени работы ЭВМ (дни); $T_{общ} = T_p$; $N_{см}$ – количество смен работы ЭВМ;

$t_{см}$ – время одного рабочего дня (час);

α – простои ЭВМ (в % от общего фонда времени работы ЭВМ). Полезный фонд времени работы ЭВМ получим:

$$T_{пол} = 264 * 8 * 1 * (1 - 10/100) = 1900,8 \text{ (Маш. час /год)}.$$

Машинное время для решения задач с помощью данной программы рассчитывается по формуле:

$$T_{MB} = qI * t_{MBI}, \quad (14)$$

где qI – количество I -задач, решаемых потребителем в год(шт.);

t_{MBI} – время решения I -ой задачи, разработанной программой (Маш. час).

$$T_{MB} = 60 * 1 = 60 \text{ (Маш. час / год)}.$$

Эксплуатационные расходы, приходящиеся на 1 час работы ЭВМ, оцениваются по формуле:

$$E_q = (A_o + S_{zn} + S_{эл} + R_{pm})/T_{пол}, \quad (15)$$

где A_O – амортизационные отчисления (руб.);

$S_{ЗП}$ – затраты по заработной плате инженера в год (руб./год);

$S_{ЭЛ}$ – стоимость потребляемой энергии (руб.);

R_{PM} – затраты на ремонт ЭВМ (руб.);

$T_{ПОЛ}$ – полезный годовой фонд работы ЭВМ, (маш.час/год).

Амортизационные отчисления рассчитываются с учетом нормы амортизации ($A_n = 12,5 \%$);

$$A_O = A_n * K_{ЭВМ} = 0,125 * 138000 = 17250 \text{ (руб.)}. \quad (16)$$

Затраты по заработной плате инженера за год рассчитывается по формуле:

$$S_{ЗП} = (1 + \omega_c) * (1 + \omega_d) * O_{ЗП} * 12, \quad (17)$$

где ω_c – коэффициент, учитывающий начисления органам социального страхования на заработную плату разработчика программы, в долях к сумме основной заработной плате разработчика;

ω_d – коэффициент, учитывающий дополнительную заработную плату разработчика программы, в долях к сумме основной заработной платы;

$O_{ЗП}$ – основная заработная плата инженера за месяц 8-го разряда (20000 руб.);

Рассчитываем годовые затраты по заработной плате и социальным отчислениям для инженера:

$$S_{ЗП} = (1 + 0,268) * (1 + 0,20) * 20000 * 12 = 365184 \text{ (руб./ год)}.$$

Стоимость потребляемой энергии оценивается по формуле:

$$S_{ЭЛ} = P * T_{ПОЛ} * Z_{ЭЛ}, \quad (18)$$

где P – мощность, потребляемая ЭВМ (кВт);

$T_{ПОЛ}$ – полезный годовой фонд работы ЭВМ (маш.час/год);

$Z_{ЭЛ}$ – тариф за 1 кВт/час (руб. /кВт).

Итак, произведем расчет стоимости потребляемой энергии:

$$S_{ЭЛ} = 0,4 * 1900,8 * 3,44 = 2202,7 \text{ (руб.)}.$$

Затраты на ремонт ЭВМ вычисляются по формуле:

$$R_{PM} = N_{CP} * S_d, \quad (19)$$

где N_{CP} – среднее количество ремонтов в год;

S_d – стоимость деталей, заменяемых при одном ремонте, в среднем.

$$R_{PM} = 2 * 2500 = 5000 \text{ (руб.)}.$$

Произведем вычисление эксплуатационных расходов, приходящихся на 1 час работы ЭВМ:

$$E_q = (17250 + 365184 + 2202,7 + 5000) / 1900,8 = 205 \text{ (руб./час)}.$$

Далее вычислим эксплуатационные расходы на ЭВМ:

$$E = (60 * 205) + 62815,92 / 5 = 24863,18 \text{ (руб.)}.$$

2.6 Определение расходов на интернет трафик

Для облегчения работы программиста и повышения его производительности в проекте предусмотрено использование глобальной сети интернет.

В настоящее время на пензенском рынке интернет очень много провайдеров предлагающие свои услуги. Цены, предлагаемые основными провайдерами, сведем в таблицу 3, для анализа и экономически целесообразного выбора поставщика услуги по интернет трафику.

Таблица 3 - Таблица интернет трафиков и провайдеров

Название провайдера	Предлагаемый тариф	Стоимость тарифа(руб.)	Скорость доступа(Мбит/с)
Ростелеком	Пакет-Для Впечатлений	699	100
МТС	Безлимитище	450	100
Дом.РУ	Скорость света	550	100

Т.к. вышеперечисленные провайдеры предлагают безлимитные интернет трафики с одинаковой скоростью доступа в сеть интернет, то экономически целесообразно выбрать трафик с самой низкой стоимостью. Выбираем ПАО "МТС", тариф "Безлимитище", с абонентской платой в месяц 450 рублей.

2.7 Технико-экономические показатели разработки

Обобщенные технико-экономические показатели разработки программы
сведены в таблицу 4

Таблица 4 - Техничко-экономические показатели разработки

Показатель	Значение
Капитальные вложения (руб.)	138000
Эксплуатационные расходы (руб.)	24863,18
Затраты на разработку ПО (руб.)	50658,52
Затраты на интернет трафик (руб.)	450
Затраты на заработную плату программисту (руб.)	30432
Суммарно (руб.)	244403,7

3 Безопасность жизнедеятельности. Эргономика рабочего места программиста

3.1 Характеристика условий труда программиста

Использование последних достижений технологического прогресса привел к изменению необходимых условий и требований производственного труда сотрудников. Работа программистов стала интенсивнее и требующей увеличенных трат физической и умственной энергии. Данная проблема требует комплексного решения вопроса в сферах оборудования рабочих мест, контроля уровня шума, света, воздушной среды.

В наши дни работа с ПК и мобильными телефонами интенсивно применяется во всех сферах работы человека. Такая деятельность подвергает человека влиянию ряда опасных электромагнитных излучений (СВЧ, УВЧ, ВЧ), шума, статики и других.

Взаимодействие с ПК и мобильными телефонами сопровождается увеличенной умственной нагрузкой на пользователя, постоянной зрительной работой и нагрузкой на мышцы шеи и спины при работе за компьютером. Несомненная важность имеет правильное оборудование и размещение рабочих мест пользователей и разработчиков.

Также в процессе своей деятельности работникам следует помнить о правильном соблюдении режимов труда и отдыха. При пренебрежении данного режима у рабочих могут возникать различные симптомы, выражающиеся в головных болях, нарушениях сна, раздражительности, усталости, болях в пояснице, шее и руках.

Одними из критериев, по которым определяется пригодность рабочих пространств работников, являются требования к параметрам воздушной среды, требования к уровню шума и вибрации, требования к освещенности рабочих мест, требования к производственному оборудованию и требования к пользователям мобильных телефонов.

					<i>ПГУ 1.09.05.01.05.001 ПЗ</i>	Лист
Изм.	Лист	№ докум.	Подпись	Дата		70

3.2 Требования к параметрам воздушной среды

Климатические параметры на рабочих местах программистов в комнатах с компьютерами и мобильными телефонами должны удовлетворять требованиям из таблицы 5

Таблица 5 - Климат рабочих помещений

Период года	Температура воздуха, С	Скорость движения воздуха, м/с	Относительная влажность воздуха, %
Холодный	22-24	до 0,1	40-60
Теплый	23-25	0,1-0,2	40-60

Воздух, находящийся в рабочих помещениях работников ЭВМ, должен быть профильтрован от пыли, загрязнений и микроорганизмов. Должно достигаться отсутствие патогенной микрофлоры.

Воздушное кондиционирование должно поддерживать параметры создаваемого микроклимата в установленных пределах во время всех сезонов года, очищать воздух от пыли и загрязнений, создавать необходимое давление в помещении для недопущения поступления неочищенного воздуха. Температура также должна быть не менее 19 градусов.

Температура воздуха в помещении должна регулироваться с учетом теплового потока от оборудования. Предпочтительным выбором оборудования должна быть техника с малой электрической мощностью. Оборудование техники должно быть установлено так, чтобы тепловой поток не был направлен на рабочих. Следует ограничивать количество техники, находящейся в помещении.

Пол в помещениях должен быть ровный, из нескользкого материала, удобным для чистки и влажной уборки. Покрытия рабочих пространств должны обладать антистатическими свойствами и сохранять свои параметры в процессе эксплуатации. В помещениях должна проводиться влажная уборка.

3.3 Требования к уровню шума и вибрации

Главными источниками шума в рабочих комнатах с установленными в них ЭВМ являются принтеры, плоттеры, вентиляторы, кондиционеры, вентиляционные системы, трансформаторы, множительная техника.

Для понижения уровня шума и вибрации в комнатах, находящееся в ней оборудование необходимо монтировать на специальные фундаменты и амортизирующие подкладки, предусмотренные нормативными документами.

Величина шума в рабочих комнатах не должен превышать уровень в 50 дБа. Нормирование шума производится путем применения малошумного оборудования, использования звукопоглощающих материалов (перфорированных панелей). Существует возможность использования акустических подвесных потолков.

Оборудование, уровень издаваемого шума которого превышает допустимые нормы, необходимо монтировать вне рабочих помещений.

3.4 Требования к монтажу производственного оборудования

Для обеспечения электропитанием ЭВМ и компьютеров в комнатах должна быть смонтирована отдельная сеть электроснабжения.

Периферийные устройства компьютеров Должны подключаться исключительно к линии электроснабжения компьютерной сети.

Запрещается включение в линии электроснабжения компьютерной сети энергопотребителей, не относящихся к компьютерам (вентиляторы, лампы и т.п.).

Запрещается использовать оборудование, не подключенное к защитному заземлению, если оно предусмотрено конструкцией оборудования.

Запрещается использовать технику с открытыми корпусами, если это не предусмотрено правилами работы этого оборудования.

					<i>ПГУ 1.09.05.01.05.001 ПЗ</i>	Лист
Изм.	Лист	№ докум.	Подпись	Дата		72

Проведение обслуживания рабочих мест программистов должно проводиться специально подготовленным персоналом, имеющим необходимую квалификацию техника (инженера).

Электрические розетки системы электропитания необходимо располагать так, чтобы кабели электрооборудования на рабочих местах не пересекали его, мешая работе.

3.5 Требования к освещению рабочих мест

Одним из важнейших параметров в мероприятиях создания оптимальных условий труда является уровень световой среды помещения. Это рационализация организации совокупности естественного и искусственного освещения на рабочих местах.

Комнаты, используемые для размещения рабочих мест пользователей ПК и мобильных телефонов, должны иметь как естественное, так и искусственное освещение.

Осуществление искусственного освещения комнат с рабочими местами должно осуществляться с помощью световых проемов, ориентированных на север, либо северо-восток.

В том случае, если ориентация производится в другие стороны, необходимо предусмотреть наличие средств регулирования интенсивности поступающего естественного освещения.

Окна в комнатах с расположенными в них компьютерах и средствами ЭВМ необходимо оборудовать средствами регулирования интенсивности естественного освещения типа занавесок, жалюзи.

3.6 Расчет освещения рабочего места программиста

Расчет освещенности рабочего места сводится к выбору системы освещения, определению необходимого числа светильников, их типа и размещения. Исходя из этого, рассчитаем параметры искусственного освещения. Обычно искусственное освещение выполняется посредством

					<i>ПГУ 1.09.05.01.05.001 ПЗ</i>	Лист
Изм.	Лист	№ докум.	Подпись	Дата		73

электрических источников света двух видов: ламп накаливания и люминесцентных ламп. Будем использовать люминесцентные лампы, которые по сравнению с лампами накаливания имеют ряд существенных преимуществ:

- по спектральному составу света они близки к дневному, естественному свету;
- обладают более высоким КПД (в 1,5-2 раза выше, чем КПД ламп накаливания);
- обладают повышенной светоотдачей (в 3-4 раза выше, чем у ламп накаливания);
- более длительный срок службы.

Расчет освещения производится для комнаты площадью 15м^2 , ширина которой 5м, высота - 3 м. Воспользуемся методом светового потока. Для определения количества светильников определим световой поток, падающий на поверхность по формуле:

$$F = E * K * S * Z / n, \quad (20)$$

где F - рассчитываемый световой поток, Лм;

E - нормированная минимальная освещенность, Лк (определяется по таблице). Работу программиста, в соответствии с этой таблицей, можно отнести к разряду точных работ, следовательно, минимальная освещенность будет $E = 300\text{Лк}$;

S - площадь освещаемого помещения (в нашем случае $S = 20\text{м}^2$);

Z - отношение средней освещенности к минимальной (обычно принимается равным 1,1...1,2, пусть $Z = 1,2$); K - коэффициент запаса, учитывающий уменьшение светового потока лампы в результате загрязнения светильников в процессе эксплуатации (его значение зависит от типа помещения и характера проводимых в нем работ и в нашем случае $K = 1,5$);

n - коэффициент использования, (выражается отношением светового потока, падающего на расчетную поверхность, к суммарному потоку всех ламп и исчисляется в долях единицы; характеризуется коэффициентами отражения от стен (PC) и потолка (PP), значение коэффициентов PC и PP

были указаны выше: $PC = 40\%$, $PP = 60\%$. Значение n определим по таблице коэффициентов использования различных светильников. Для этого вычислим индекс помещения по формуле:

$$I = S / h * (A + B), \quad (21)$$

где S - площадь помещения, $S = 20 \text{ м}^2$;

h - расчетная высота подвеса, $h = 3 \text{ м}$;

A - ширина помещения, $A = 4 \text{ м}$;

B - длина помещения, $B = 5 \text{ м}$.

Подставив значения получаем:

$$I = 20/3 * (4 + 5) = 60.$$

Зная индекс помещения I , по таблице находим $n = 0,22$.

Подставим все значения в формулу для определения светового потока F :

$$F = 300 * 1,5 * 20 * 1,2 / 0,22 = 49090 \text{ Лм}.$$

Для освещения выбираем люминесцентные лампы типа ЛБ40-1, световой поток которых $F = 4320 \text{ Лк}$.

Рассчитаем необходимое количество ламп по формуле:

$$N = F / F_{\text{л}}, \quad (22)$$

где N - определяемое число ламп;

F - световой поток, $F = 49090 \text{ Лм}$;

$F_{\text{л}}$ - световой поток лампы, $F_{\text{л}} = 4320 \text{ Лм}$.

$$N = 49090 / 4320 = 11 \text{ штук}.$$

Заключение

В процессе выполнения данной выпускной квалификационной работы проведены анализ требований, проектирование и разработка, тестирование Android-приложение по выбору должностей на основе характеристики личности и профессиональных навыков.

В процессе анализа требований, были сформированы требования к системе. Проведён анализ вариантов использования. Разработана модель предметной области, представленная диаграммой классов. В результате выполнения выпускной квалификационной работы реализовано Android-приложение по выбору должностей на основе характеристики личности и профессиональных навыков.

Проект реализован в среде Android Studio на языке программирования Java и с использованием языка разметки XML. На основе анализа, полученных результатов, можно сделать вывод, что все функции приложения отрабатываются корректно в штатном режиме.

Таким образом, требования технического задания были выполнены в полном объёме и в установленные сроки.

Список использованных источников

1. Ступени отбора персонала в организацию [Электронный ресурс] URL: <https://osnovap.ru/analytics/stupeni-otbora-personala-v-organizaciyu>.
2. Проблемы отбора персонала и их решение [Электронный ресурс] URL: <https://www.kausgroup.ru/knowledge/300articles/category/personnel/material/133/>.
3. Мобильные операционные системы [Электронный ресурс] URL: <https://www.technodor.info/2019/04/blog-post.html>.
4. Статистика Android vs IOS в 2021 году [Электронный ресурс] URL: <https://gs.statcounter.com/os-market-share/mobile/worldwide>.
5. Исследования рынка рекрутинговых систем: функциональность облачных ATS в России [Электронный ресурс] URL: <https://www.tadviser.ru/index.php>.
6. Типируем сотрудников по тесту 16 personalities: 6 способов использования в работе [Электронный ресурс] URL: <https://peopleforce.io/ru/blog/tipiruem-sotrudnikov-po-testu-16-personalities>.
7. Противостояние Eclipse vs IntelliJ IDEA [Электронный ресурс] URL: <https://webformymself.com/eclipse-vs-intellij-idea/>
8. Что лучше android studio или intellij idea [Электронный ресурс] URL: <https://everychild.ru/rejting/chto-luchshe-android-studio-ili-intellij-idea>.

Приложение А

(обязательное)

Android-приложение по выбору должностей на основе характеристики
личности и профессиональных навыков

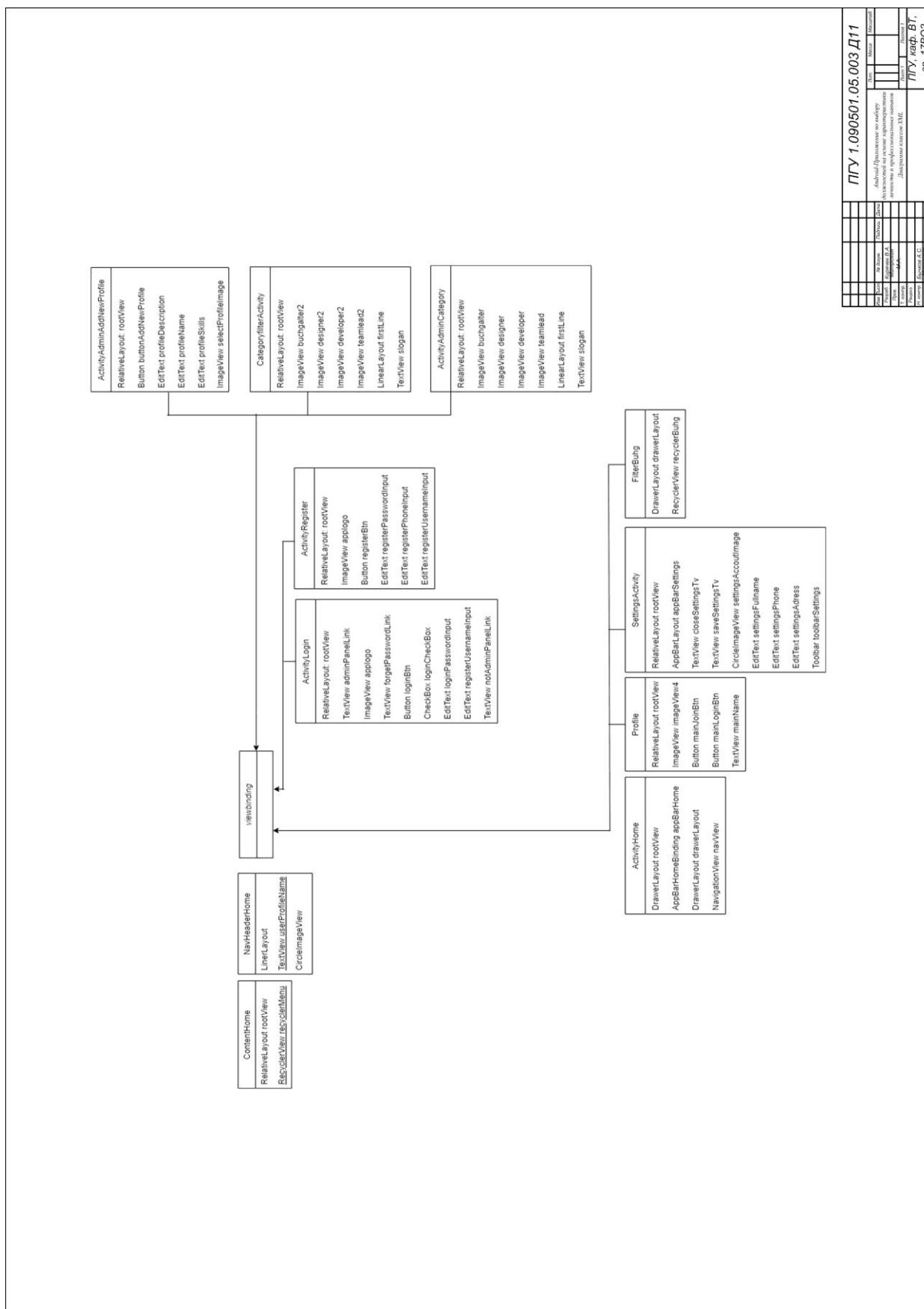
UML - диаграммы

					<i>ПГУ 1.09.05.01.05.001 ПЗ</i>	Лист
Изм.	Лист	№ докум.	Подпись	Дата		78

Приложение А.1 – Диаграмма классов *Java*

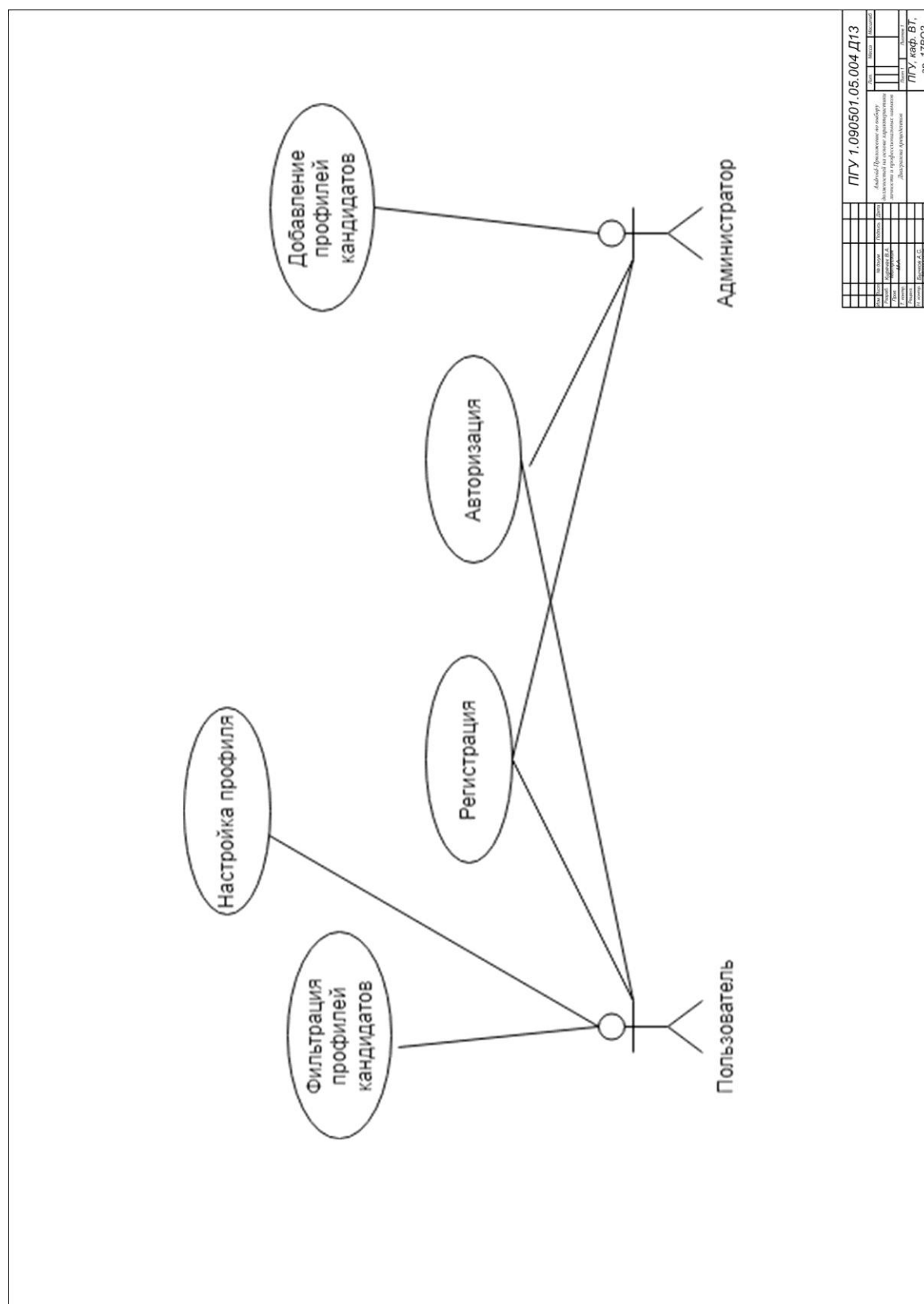
[illegible]

Приложение А.2 – Диаграмма классов XML



ПГУ 1.090501.05.003 Д11									
Имя	Фамилия	Отчество	Дата	Подпись	Подпись	Подпись	Подпись	Подпись	Подпись
Имя	Фамилия	Отчество	Дата	Подпись	Подпись	Подпись	Подпись	Подпись	Подпись
Имя	Фамилия	Отчество	Дата	Подпись	Подпись	Подпись	Подпись	Подпись	Подпись
Имя	Фамилия	Отчество	Дата	Подпись	Подпись	Подпись	Подпись	Подпись	Подпись
Имя	Фамилия	Отчество	Дата	Подпись	Подпись	Подпись	Подпись	Подпись	Подпись
Имя	Фамилия	Отчество	Дата	Подпись	Подпись	Подпись	Подпись	Подпись	Подпись
Имя	Фамилия	Отчество	Дата	Подпись	Подпись	Подпись	Подпись	Подпись	Подпись
Имя	Фамилия	Отчество	Дата	Подпись	Подпись	Подпись	Подпись	Подпись	Подпись
Имя	Фамилия	Отчество	Дата	Подпись	Подпись	Подпись	Подпись	Подпись	Подпись

Приложение А.3 – Диаграмма прецедентов

[illegible]

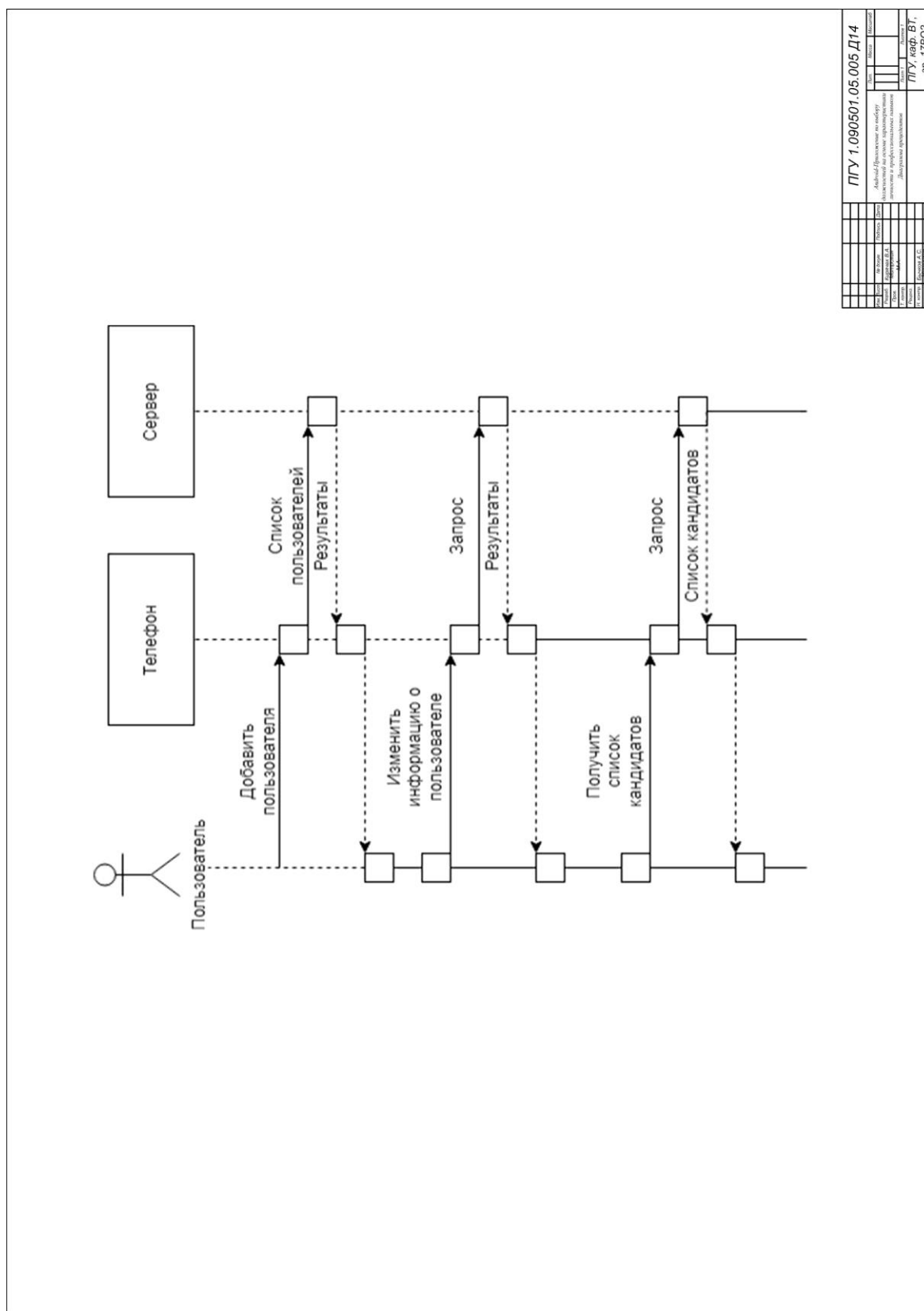
<i>Изм.</i>	<i>Лист</i>	<i>№ докум.</i>	<i>Подпись</i>	<i>Дата</i>

ПГУ 1.09.05.01.05.001 ПЗ

Лист

81

Приложение А.4 – Диаграмма последовательности



Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.09.05.01.05.001 ПЗ

Лист

82

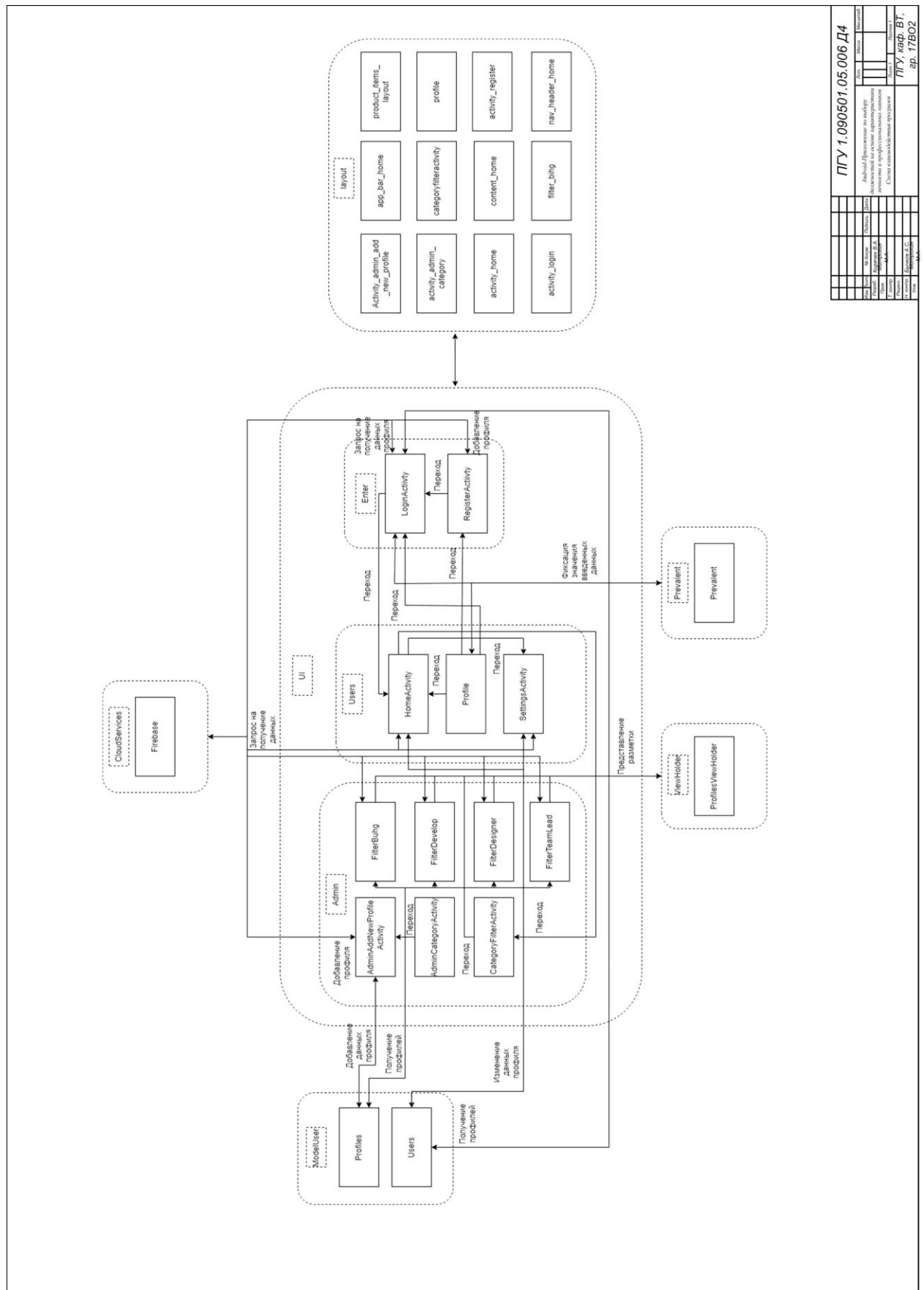
Приложение Б

(обязательное)

Android-приложение по выбору должностей на основе характеристики
личности и профессиональных навыков

Схема взаимодействия программ

					<i>ПГУ 1.09.05.01.05.001 ПЗ</i>	Лист
Изм.	Лист	№ докум.	Подпись	Дата		83



Приложение В

(обязательное)

Android-приложение по выбору должностей на основе характеристики
личности и профессиональных навыков

Листинг программы

					<i>ПГУ 1.09.05.01.05.001 ПЗ</i>	Лист
Изм.	Лист	№ докум.	Подпись	Дата		85

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="space.vladdek.diplo">
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Diplo">
        <activity android:name=".ui.Users.FilterBuchg"
            android:exported="false"/>
        <activity android:name=".ui.Admin.CategoryFilterActivity"
            android:exported="false"/>
        <activity android:name=".ui.Users.SettingsActivity"
            android:exported="false" />
        <activity android:name=".ui.Users.HomeActivity"
            android:exported="false"
            android:label="@string/title_activity_home"
            android:theme="@style/Theme.Diplo.NoActionBar" />
        <activity android:name=".ui.Admin.AdminCategoryActivity"
            android:exported="false" />
        <activity android:name=".ui.Admin.AdminAddNewProfileActivity"
            android:exported="false" />
        <activity android:name=".ui.RegisterActivity"
            android:exported="false" />
        <activity android:name=".ui.LoginActivity"
            android:exported="false" />
        <activity android:name=".CoursePage"
            android:exported="false" />
        <activity
            android:name="com.theartofdev.edmodo.cropper.CropImageActivity"
            android:theme="@style/Base.Theme.AppCompat"/>
        <activity android:name=".ui.Users.Profile"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>

```

					ПГУ 1.09.05.01.05.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		86

```

        </activity>
        <meta-data
            android:name="preloaded_fonts"
            android:resource="@array/preloaded_fonts" />
    </application>
</manifest>

package space.vladek.diplo.Interface;
import android.view.View;
public interface ItemClickListener {
    void onClick(View view, int position, boolean isLongClick);
}
package space.vladek.diplo.ModelUser;
public class Profiles {
    private String pName, Description, Skills, image, category, pid, date,
time, name ;
    public Profiles(){
    }
    public Profiles(String pName, String description, String skills, String
image, String category, String pid, String date, String time, String name) {
        this.pName = pName;
        Description = description;
        Skills = skills;
        image = image;
        this.category = category;
        this.pid = pid;
        this.date = date;
        this.time = time;
        name = name;
    }
    public String getName() {
        return this.name;
    }
    public void setName(final String name) {
        this.name = name;
    }
    public String getpName() {
        return pName;
    }
    public void setpName(String pName) {
        this.pName = pName;
    }
}

```

					ПГУ 1.09.05.01.05.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		87

```

public String getDescription() {
    return Description;
}
public void setDescription(String description) {
    Description = description;
}
public String getSkills() {
    return Skills;
}
public void setSkills(String skills) {
    Skills = skills;
}
public String getImage() {
    return image;
}
public void setImage(String image) {
    this.image = image;
}
public String getCategory() {
    return category;
}
public void setCategory(String category) {
    this.category = category;
}
public String getPid() {
    return pid;
}
public void setPid(String pid) {
    this.pid = pid;
}
public String getDate() {
    return date;
}
public void setDate(String date) {
    this.date = date;
}
public String getTime() {
    return time;
}
public void setTime(String time) {
    this.time = time;
}

```

					<i>ПГУ 1.09.05.01.05.001 ПЗ</i>	Лист
Изм.	Лист	№ докум.	Подпись	Дата		88

```

    }
}
package space.vladek.diplo.ModelUser;
import space.vladek.diplo.Prevalent.Prevalent;
public class Users {
    private String name, phone, password, image;
    public Users()
    {
    }
    public Users(String name, String phone, String password, String image) {
        this.name = name;
        this.phone = phone;
        this.password = password;
        this.image = image;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getPhone() {
        return phone;
    }
    public void setPhone(String phone) {
        this.phone = phone;
    }

    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
    public String getImage() {
        return image;
    }
    public void setImage(String image) {
        this.image = image;
    }
}

```

```

package space.vladek.diplo.Prevalent;
import space.vladek.diplo.ModelUser.Users;
public class Prevalent {
    public Users currentOnlineUser;
    public Users getCurrentOnlineUser() {
        return currentOnlineUser;
    }
    public void setCurrentOnlineUser(Users currentOnlineUser) {
        this.currentOnlineUser = currentOnlineUser;
    }
    public static final String UserPhoneKey = "UserPhone";
    public static final String UserPasswordKey = "UserPassword";
}

package space.vladek.diplo.ui.Admin;
import space.vladek.diplo.R;
public class AdminAddNewProfileActivity extends AppCompatActivity {
    private String categoryName, Description, Skill, Name, saveCurrentDate,
saveCurrentTime, ProfileRandomKey;
    private String downloadImageUrl;
    private ImageView profileImage;
    private EditText profileName, profileDescription, profileSkills;
    private Button addNewProfileButton;
    private static final int GALLERYPICK = 1;
    private Uri ImageUri;
    private StorageReference ProfileImageRef;
    private DatabaseReference ProfilesRef;
    private ProgressDialog loadingBar;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_admin_add_new_profile);
        init();
        profileImage.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                OpenGallery();
            } //вызов функции открытия галереи на телефоне и взятия от туда
фото
        });
        addNewProfileButton.setOnClickListener(new View.OnClickListener() {
            @Override

```

					ПГУ 1.09.05.01.05.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		90

```

        public void onClick(View v) {
            ValidateProfileData();
        } //получение введенных значений и проверка заполнения
    });
}

private void init(){ //инициализация переменных, начиная от заполнения,
заканчивая базой и панелью загрузки
    categoryName = getIntent().getExtras().get("category").toString();
//получение значений из категорий
    profileImage = findViewById(R.id.select_profile_image);
    profileName = findViewById(R.id.profile_name);
    profileDescription = findViewById(R.id.profile_description);
    profileSkills = findViewById(R.id.profile_skills);
    addNewProfileButton = findViewById(R.id.button_add_new_profile);
    ProfileImageRef =
FirebaseStorage.getInstance().getReference().child("Profile Images");
    ProfilesRef =
FirebaseDatabase.getInstance().getReference().child("Profiles");
    loadingBar = new ProgressDialog(this);
}

private void ValidateProfileData(){ //занесение значений в переменные и
проверка заполнения
    Description = profileDescription.getText().toString();
    Skill = profileSkills.getText().toString();
    Name = profileName.getText().toString();
    if (ImageUri == null){
        Toast.makeText(this, "добавьте фото пользователя",
Toast.LENGTH_LONG).show();
    } else if (TextUtils.isEmpty(Description)){
        Toast.makeText(this, "Добавьте характеристику",
Toast.LENGTH_LONG).show();
    } else if (TextUtils.isEmpty(Skill)){
        Toast.makeText(this, "Добавьте навыки пользователя",
Toast.LENGTH_LONG).show();
    } else if (TextUtils.isEmpty(Name)){
        Toast.makeText(this, "Добавьте имя пользователя",
Toast.LENGTH_LONG).show();
    } else {
        ProfileInformation();
    }
}
}

```

```

private void ProfileInformation() {
    loadingBar.setTitle("Загрузка данных");
    loadingBar.setMessage("Пожалуйста, подождите...");
    loadingBar.setCanceledOnTouchOutside(false);
    loadingBar.show();
    Calendar calendar = Calendar.getInstance();
    SimpleDateFormat currentDate = new SimpleDateFormat("ddMMyyyy");
    saveCurrentDate = currentDate.format(calendar.getTime());
    SimpleDateFormat currentTime = new SimpleDateFormat("HHmmss");
    saveCurrentTime = currentTime.format(calendar.getTime());
    ProfileRandomKey = saveCurrentDate + saveCurrentTime;
    final StorageReference filePath =
ProfileImageRef.child(ImageUri.getLastPathSegment() + ProfileRandomKey +
".jpg");
    final UploadTask uploadTask = filePath.putFile(ImageUri);
    uploadTask.addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception e) {
            String message = e.toString();
            Toast.makeText(AdminAddNewProfileActivity.this, "Ошибка: " +
message, Toast.LENGTH_SHORT).show();
            loadingBar.dismiss();
        }
    }).addOnSuccessListener(new
OnSuccessListener<UploadTask.TaskSnapshot>() {
        @Override
        public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
            Toast.makeText(AdminAddNewProfileActivity.this, "Изображение
успешно загружено.", Toast.LENGTH_SHORT).show();
            uploadTask.continueWithTask(new
Continuation<UploadTask.TaskSnapshot, Task<Uri>>() {
                @Override
                public Task<Uri> then(@NonNull
Task<UploadTask.TaskSnapshot> task) throws Exception {
                    if(!task.isSuccessful()){
                        throw task.getException();
                    }
                    downloadImageUrl =
filePath.getDownloadUrl().toString();
                    return filePath.getDownloadUrl();
                }
            });
        }
    });
}

```

```

    }
    }).addOnCompleteListener(new OnCompleteListener<Uri>() {
        @Override
        public void onComplete(@NonNull Task<Uri> task) {
            if(task.isSuccessful()){
                downloadImageUrl = task.getResult().toString();
                Toast.makeText(AdminAddNewProfileActivity.this,
"Фото сохранено", Toast.LENGTH_SHORT).show();
                SaveProfileInfoToDatabase();
            }
        }
    });
}

private void SaveProfileInfoToDatabase(){ //добавление профиля в базу
данных

    HashMap<String,Object> profileMap = new HashMap<>();
    profileMap.put("pid", ProfileRandomKey);
    profileMap.put("date", saveCurrentDate);
    profileMap.put("time",saveCurrentTime);
    profileMap.put("description", Description);
    profileMap.put("skills", Skill);
    profileMap.put("name", Name);
    profileMap.put("pname", categoryName);
    profileMap.put("image", downloadImageUrl);

    ProfilesRef.child(ProfileRandomKey).updateChildren(profileMap).addOnCompleteLis
tener(new OnCompleteListener<Void>() { //в случае успеха добавление профиля и
запуск другого окна, иначе сообщение об ошибке
        @Override
        public void onComplete(@NonNull Task<Void> task) {
            if(task.isSuccessful()){
                Intent intent = new Intent(AdminAddNewProfileActivity.this,
AdminCategoryActivity.class );
                startActivity(intent);
                loadingBar.dismiss();
                Toast.makeText(AdminAddNewProfileActivity.this, "Add profile",
Toast.LENGTH_LONG).show();
            }
            else {
                String message = task.getException().toString();

```

					ПГУ 1.09.05.01.05.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		93

```

        Toast.makeText (AdminAddNewProfileActivity.this, "Error" +
message, Toast.LENGTH_LONG).show();

        loadingBar.dismiss();

    }

}

});

}

private void OpenGallery(){ //открытие галереи телефона и взятие оттуда
изображения

    Intent galleryIntent = new Intent();
    galleryIntent.setAction(Intent.ACTION_GET_CONTENT);
    galleryIntent.setType("image/*");
    startActivityForResult(galleryIntent, GALLERYPICK);

}

@Override

protected void onActivityResult(int requestCode, int resultCode, @Nullable
Intent data) { //занесение изображения из памяти телефона в переменную
    super.onActivityResult(requestCode, resultCode, data);
    if(requestCode == GALLERYPICK && resultCode == RESULT_OK && data !=
null)

    {

        ImageUri = data.getData();
        profileImage.setImageURI(ImageUri);

    }

}

package space.vladek.diplo.ui.Admin;
import space.vladek.diplo.R;
public class AdminCategoryActivity extends AppCompatActivity {
    private ImageView buchg, develop, lead, design;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_admin_category);
        init(); //инициализация категорий
        buchg.setOnClickListener(new View.OnClickListener() { //передача
значения категории в добавление админом

        @Override
        public void onClick(View v) {

            Intent intent = new Intent(AdminCategoryActivity.this,
AdminAddNewProfileActivity.class);

            intent.putExtra("category", "buchg");
            startActivity(intent);

```

```

        }

    });

    develop.setOnClickListener(new View.OnClickListener() {

        @Override

        public void onClick(View v) {

            Intent intent = new Intent(AdminCategoryActivity.this,
AdminAddNewProfileActivity.class);

            intent.putExtra("category", "develop");

            startActivity(intent);

        }

    });

    lead.setOnClickListener(new View.OnClickListener() {

        @Override

        public void onClick(View v) {

            Intent intent = new Intent(AdminCategoryActivity.this,
AdminAddNewProfileActivity.class);

            intent.putExtra("category", "lead");

            startActivity(intent);

        }

    });

    design.setOnClickListener(new View.OnClickListener() {

        @Override

        public void onClick(View v) {

            Intent intent = new Intent(AdminCategoryActivity.this,
AdminAddNewProfileActivity.class);

            intent.putExtra("category", "design");

            startActivity(intent);

        }

    });

}

private void init(){ //функция инициализации
    buchg = findViewById(R.id.buchgalter);
    develop = findViewById(R.id.developer);
    lead = findViewById(R.id.teamlead);
    design = findViewById(R.id.designer);
}

}

package space.vladek.diplo.ui.Admin;
import space.vladek.diplo.R;
import space.vladek.diplo.ui.Users.FilterBuchg;

public class CategoryFilterActivity extends AppCompatActivity {

```

```

private ImageView buchg, develop, lead, design;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.categoryfilteractivity);
    init(); //инициализация категорий
    buchg.setOnClickListener(new View.OnClickListener() { //передача
значения категории в добавление админом
        @Override
        public void onClick(View v) {
            Intent intent = new Intent(CategoryFilterActivity.this,
FilterBuchg.class);
            startActivity(intent);
        }
    });
    develop.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent intent = new Intent(CategoryFilterActivity.this,
AdminAddNewProfileActivity.class);
            intent.putExtra("category", "develop");
            startActivity(intent);
        }
    });
    lead.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent intent = new Intent(CategoryFilterActivity.this,
AdminAddNewProfileActivity.class);
            intent.putExtra("category", "lead");
            startActivity(intent);
        }
    });
    design.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent intent = new Intent(CategoryFilterActivity.this,
AdminAddNewProfileActivity.class);
            intent.putExtra("category", "design");
            startActivity(intent);
        }
    });
}

```

```

    }

    private void init(){ //функция инициализации
        buchg = findViewById(R.id.buchgalter2);
        develop = findViewById(R.id.developer2);
        lead = findViewById(R.id.teamlead2);
        design = findViewById(R.id.designer2);
    }
}

package space.vladek.diplo.ui.Users;
import io.paperdb.Paper;
import space.vladek.diplo.R;
import space.vladek.diplo.ViewHolder.ProfilesViewHolder;
import space.vladek.diplo.ui.Admin.AdminAddNewProfileActivity;
import space.vladek.diplo.ui.Admin.AdminCategoryActivity;
import space.vladek.diplo.ui.Admin.CategoryFilterActivity;
import space.vladek.diplo.ui.LoginActivity;
public class FilterBuchg extends AppCompatActivity {
    private AppBarConfiguration mAppBarConfiguration;
    DatabaseReference ProductsRef;
    StorageReference storageRef;
    private RecyclerView recyclerView;
    RecyclerView.LayoutManager layoutManager;
    private final String parentDbName = "Users";
    private Prevalent prevalent;
    public Users usersData;
    private String image;
    private StorageReference ProfileImageRef;
    private UploadTask uploadTask;
    private String downloadImageUri;
    private Uri uriDown;
    private String broker = "broker", resolution = "resolution", protector =
"protector", responsibility = "responsibility", polemicist = "polemicist";
//    private String ;
    @Override
    protected void onCreate(final Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        this.setContentViews(R.layout.filter_buhg);
        this.ProductsRef =
FirebaseDatabase.getInstance().getReference().child("Profiles");
        ProfileImageRef =
FirebaseStorage.getInstance().getReference().child("Profile Images");
        this.recyclerView = this.findViewById(R.id.recycler_buchg);

```

					ПГУ 1.09.05.01.05.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		97

```

        this.recyclerView.setHasFixedSize(true);
        this.layoutManager = new LinearLayoutManager(this);
        this.recyclerView.setLayoutManager(this.layoutManager);
    }
    @Override
    protected void onStart() {
        super.onStart();
        FirebaseRecyclerOptions<Profiles> options = new
        FirebaseRecyclerOptions.Builder<Profiles>()
            .setQuery(ProductsRef, Profiles.class).build();
        FirebaseRecyclerAdapter<Profiles, ProfilesViewHolder> adapter = new
        FirebaseRecyclerAdapter<Profiles, ProfilesViewHolder>(options) {
            @Override
            protected void onBindViewHolder(@NonNull ProfilesViewHolder holder,
            int position, @NonNull Profiles model) {
                if (model.getDescription().equals(polemicist) ||
                model.getDescription().equals(protector))
                {
                    holder.txtProfileName.setText(model.getName());
                    holder.txtProfilePrice.setText(model.getSkills());

                    holder.txtProfileDescription.setText(model.getDescription());

                    Picasso.get().load(model.getImage()).into(holder.imageView);
                } else {
                    holder.txtProfileName.setVisibility(View.GONE);
                    holder.txtProfileDescription.setVisibility(View.GONE);
                    holder.txtProfilePrice.setVisibility(View.GONE);
                    holder.imageView.setVisibility(View.GONE);
                    //holder.txtProfileName.setLayoutParams(new
                    RecyclerView.LayoutParams(0, 0));
                }
            }
            @NonNull
            @Override
            public ProfilesViewHolder onCreateViewHolder(@NonNull ViewGroup
            parent, int viewType) {
                View view =
                LayoutInflater.from(parent.getContext()).inflate(R.layout.product_items_layout,
                parent, false);

                ProfilesViewHolder holder = new ProfilesViewHolder(view);

```

					ПГУ 1.09.05.01.05.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		98

```

        return holder;
    }
};
recyclerView.setAdapter(adapter);
adapter.startListening();
}
}

package space.vladek.diplo.ui.Users;
import space.vladek.diplo.R;
import space.vladek.diplo.ViewHolder.ProfilesViewHolder;
import space.vladek.diplo.ui.Admin.AdminAddNewProfileActivity;
import space.vladek.diplo.ui.Admin.AdminCategoryActivity;
import space.vladek.diplo.ui.Admin.CategoryFilterActivity;
import space.vladek.diplo.ui.LoginActivity;
public class HomeActivity extends AppCompatActivity implements
NavigationView.OnNavigationItemSelectedListener {
    private AppBarConfiguration mAppBarConfiguration;
    DatabaseReference ProductsRef;
    StorageReference storageRef;
    private RecyclerView recyclerView;
    RecyclerView.LayoutManager layoutManager;
    private final String parentDbName = "Users";
    private Prevalent prevalent;
    public Users usersData;
    private String image;
    private StorageReference ProfileImageRef;
    private UploadTask uploadTask;
    private String downloadImageUri;
    private Uri uriDown;
    @Override
    protected void onCreate(final Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        this setContentView(R.layout.activity_home);
        this.ProductsRef =
        FirebaseDatabase.getInstance().getReference().child("Profiles");
        ProfileImageRef =
        FirebaseStorage.getInstance().getReference().child("Profile Images");
        final Toolbar toolbar = this.findViewById(R.id.toolbar);
        toolbar.setTitle("Меню");
        this.supportActionBar(toolbar);
        final FloatingActionButton fab = this.findViewById(R.id.fab);

```

```

fab.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(final View view) {
        Snackbar.make(view, "Здесь будет переход в корзину",
BaseTransientBottomBar.LENGTH_LONG)
            .setAction("Action", null).show()
    }
});

final DrawerLayout drawer = this.findViewById(R.id.drawer_layout);
final ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
    this, drawer, toolbar, R.string.navigation_drawer_open,
R.string.navigation_drawer_close) {
    @Override
    public void onDrawerSlide(final View drawerView, final float
slideOffset) {
        super.onDrawerSlide(drawerView, slideOffset);
    }
};

drawer.addDrawerListener(toggle);
toggle.syncState();

final NavigationView navigationView = this.findViewById(R.id.nav_view);
navigationView.setNavigationItemSelectedListener(this);

//

final View headerView = navigationView.getHeaderView(0);
final TextView userNameTextView =
headerView.findViewById(R.id.user_profile_name);
final CircleImageView profileImageView =
headerView.findViewById(R.id.user_profile_image);
DatabaseReference rootRef;
rootRef = FirebaseDatabase.getInstance().getReference();
Bundle extras = getIntent().getExtras();
String sessionName = extras.getString("keyName");
String sessionPhone = extras.getString("ketPhone");
String sessionPassword = extras.getString("keyPassword");
String sessionImage = extras.getString("keyImage");
userNameTextView.setText(sessionName);
this.recyclerView = this.findViewById(R.id.recycler_menu);
this.recyclerView.setHasFixedSize(true);
this.layoutManager = new LinearLayoutManager(this);
this.recyclerView.setLayoutManager(this.layoutManager);
}

```

					ПГУ 1.09.05.01.05.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		100

```

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

    FirebaseRecyclerOptions<Profiles> options = new
FirebaseRecyclerOptions.Builder<Profiles>()
        .setQuery(ProductsRef, Profiles.class).build();

    FirebaseRecyclerAdapter<Profiles, ProfilesViewHolder> adapter = new
FirebaseRecyclerAdapter<Profiles, ProfilesViewHolder>(options) {
        @Override
        protected void onBindViewHolder(@NonNull ProfilesViewHolder holder,
int position, @NonNull Profiles model) {
            holder.txtProfileName.setText(model.getName());
            holder.txtProfilePrice.setText(model.getSkills());
            holder.txtProfileDescription.setText(model.getDescription());
            Picasso.get().load(model.getImage()).into(holder.imageView);
        }
        @NonNull
        @Override
        public ProfilesViewHolder onCreateViewHolder(@NonNull ViewGroup
parent, int viewType) {
            View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.product_items_layout,
parent, false);

            ProfilesViewHolder holder = new ProfilesViewHolder(view);
            return holder;
        }
    };
    recyclerView.setAdapter(adapter);
    adapter.startListening();
}
@Override
public void onBackPressed() {
    final DrawerLayout drawerLayout =
this.findViewById(R.id.drawer_layout);
    if (drawerLayout.isDrawerOpen(GravityCompat.START)) {
        drawerLayout.closeDrawer(GravityCompat.START);
    } else {
        super.onBackPressed();
    }
}
@Override

```

					ПГУ 1.09.05.01.05.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		101

```

        public boolean onCreateOptionsMenu(final Menu menu) {
            // Inflate the menu; this adds items to the action bar if it is
present.
            this.getMenuInflater().inflate(R.menu.home, menu);
            return true;
        }
        @Override
        public boolean onNavigationItemSelected(@NonNull final MenuItem item) {
            final int id = item.getItemId();
            if (id == R.id.nav_categories) {
                Intent homeIntent = new Intent(HomeActivity.this,
CategoryFilterActivity.class);
                startActivity(homeIntent);
            } else if (id == R.id.nav_settings) {
                final Intent loginIntent = new Intent(this,
SettingsActivity.class);
                loginIntent.putExtra("keyName", usersData.getName());
                loginIntent.putExtra("keyPhone", usersData.getPhone());
                loginIntent.putExtra("keyPassword", usersData.getPassword());
                loginIntent.putExtra("keyImage", usersData.getImage());
                this.startActivity(loginIntent);
            } else if (id == R.id.nav_logout) {
                Paper.book().destroy();
                final Intent loginIntent = new Intent(this, LoginActivity.class);
                this.startActivity(loginIntent);
            }
            final DrawerLayout drawerLayout =
this.findViewById(R.id.drawer_layout);
            drawerLayout.closeDrawer(GravityCompat.START);
            return false;
        }
    }
}

package space.vladek.diplo.ui.Users;
import space.vladek.diplo.ui.LoginActivity;
import space.vladek.diplo.ModelUser.Users;
import space.vladek.diplo.Prevalent.Prevalent;
import space.vladek.diplo.R;
import space.vladek.diplo.ui.RegisterActivity;
public class Profile extends AppCompatActivity {
    private Button joinButton, regButton;
    private ProgressDialog loadingBar;

```

					<i>ПГУ 1.09.05.01.05.001 ПЗ</i>	Лист
Изм.	Лист	№ докум.	Подпись	Дата		102

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.profile);

    joinButton = (Button) findViewById(R.id.main_login_btn);
    //инициализация кнопки входа
    regButton = (Button) findViewById(R.id.main_jotn_btn); //инициализация
кнопки регистрации
    loadingBar = new ProgressDialog(this);
    Paper.init(this);
    joinButton.setOnClickListener(new View.OnClickListener() { //запуск
страницы входа
        @Override
        public void onClick(View v) {
            Intent loginIntent = new Intent(Profile.this,
LoginActivity.class);
            startActivity(loginIntent);
        }
    });
    regButton.setOnClickListener(new View.OnClickListener() { //запуск
страницы регистрации
        @Override
        public void onClick(View v) {
            Intent registerIntent = new Intent(Profile.this,
RegisterActivity.class);
            startActivity(registerIntent);
        }
    });

    String UserPhoneKey = Paper.book().read(Prevalent.UserPhoneKey);
    //добавление переменной номера и считывание ее значения из записи телефона
    String UserPasswordKey = Paper.book().read(Prevalent.UserPasswordKey);
    //добавление переменной пароля и считывание ее значения из записи телефона
    if(UserPhoneKey != "" && UserPasswordKey != ""){ //проверка на
существование записей
        if (!TextUtils.isEmpty(UserPhoneKey) &&
(!TextUtils.isEmpty(UserPasswordKey))){
            ValidateUser(UserPhoneKey, UserPasswordKey); //если запись
существует, то запуск функции подтверждения входа
            loadingBar.setTitle("Enter application");
            loadingBar.setMessage("Please Wait");
            loadingBar.setCanceledOnTouchOutside(false);
            loadingBar.show();
        }
    }
}

```

```

    }
}

private void ValidateUser(final String phone, final String password){
    final DatabaseReference RootRef; //установка связи с базой данных
    RootRef = FirebaseDatabase.getInstance().getReference();

    RootRef.addListenerForSingleValueEvent(new ValueEventListener() {
//запуск функции проверки номера и пароля
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            if(dataSnapshot.child("Users").child(phone).exists()) //Если в
базе существуют значения
            {
                Users usersData =
dataSnapshot.child("Users").child(phone).getValue(Users.class); //Получение
значения в переменной номера из базы

                if(usersData.getPhone().equals(phone)) //если совпадает со
значением из базы
                {
                    if(usersData.getPassword().equals(password)) // если
совпадает со значение из базы
                    {
                        loadingBar.dismiss();
                        Toast.makeText(Profile.this, "Successful",
Toast.LENGTH_LONG).show();

                        Intent homeIntent = new Intent(Profile.this,
HomeActivity.class);

                        startActivity(homeIntent); //всплывающее сообщение
успешного входа и переход на домашнюю страницу
                    }
                    else {
                        loadingBar.dismiss();
                    }
                }
            } else {
                loadingBar.dismiss();
                Toast.makeText(Profile.this, "Account with number" + phone
+ "doesn't exist", Toast.LENGTH_LONG).show(); //так же , может нужен телефон
                Intent registerIntent = new Intent(Profile.this,

```

```

RegisterActivity.class);

        startActivity(registerIntent); //если аккаунта с таким
номером не существует, то перевод на страницу регистрации
    }

}

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

});

}

}

package space.vladek.diplo.ui.Users

private CircleImageView profileImageView;
private EditText fullNameEditText, userPhoneEditText, addressEditText;
private TextView saveTextBtn, closeTextBtn;
private Uri imageUri;
private String checker = "";
private StorageReference storageProfilePictureRef;
private StorageTask uploadTask;
private Prevalent prevalent;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_settings);
    profileImageView = (CircleImageView)
findViewById(R.id.settings_account_image);
    fullNameEditText = (EditText) findViewById(R.id.settings_fullname);
    userPhoneEditText = (EditText) findViewById(R.id.settings_phone);
    addressEditText = (EditText) findViewById(R.id.settings_address);
    saveTextBtn = (TextView) findViewById(R.id.save_settings_tv);
    closeTextBtn = (TextView) findViewById(R.id.close_settings_tv);
    storageProfilePictureRef =
FirebaseStorage.getInstance().getReference().child("Profile pictures");
    userInfoDisplay(profileImageView, fullNameEditText, userPhoneEditText,
addressEditText );
    closeTextBtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent loginIntent = new Intent(SettingsActivity.this,
HomeActivity.class);
            startActivity(loginIntent);

```

```

    }
    });

//
saveTextBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if(checker.equals("clicked")){
            userInfoSaved();
        }
        else {
            updateOnlyUser();
        }
    }
});

profileImageView.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        checker = "clicked";
        CropImage.activity(imageUri).setAspectRatio(1,
1).start(SettingsActivity.this);
    }
});

}

private void userInfoDisplay(final CircleImageView profileImageView,final
EditText fullNameEditText,final EditText userPhoneEditText,final EditText
addressEditText) {
    String phone = prevalent.getCurrentOnlineUser().getPhone();
    DatabaseReference UsersRef =
FirebaseDatabase.getInstance().getReference().child("Users").child(phone);
    UsersRef.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot dataSnapshot)
        {
            if (dataSnapshot.exists())
            {
                if (dataSnapshot.child("image").exists())
                {
                    String image =
dataSnapshot.child("image").getValue().toString();

                    String name =
dataSnapshot.child("name").getValue().toString();

```

					ПГУ 1.09.05.01.05.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		106

```

        String phone =
dataSnapshot.child("phone").getValue().toString();
        String address =
dataSnapshot.child("address").getValue().toString();

        Picasso.get().load(image).into(profileImageView);
        fullNameEditText.setText(name);
        userPhoneEditText.setText(phone);
        addressEditText.setText(address);
    }
    if (dataSnapshot.child("address").exists())
    {
        String name =
dataSnapshot.child("name").getValue().toString();
        String phone =
dataSnapshot.child("phone").getValue().toString();
        String address =
dataSnapshot.child("address").getValue().toString();

        fullNameEditText.setText(name);
        userPhoneEditText.setText(phone);
        addressEditText.setText(address);
    }
}

@Override
public void onCancelled(DatabaseError databaseError) {

}

});
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent
data)
{
    super.onActivityResult(requestCode, resultCode, data);

    if (requestCode==CropImage.CROP_IMAGE_ACTIVITY_REQUEST_CODE &&
resultCode==RESULT_OK && data!=null)
    {
        CropImage.ActivityResult result =
CropImage.getActivityResult(data);
        imageUri = result.getUri();
    }
}

```

```

        profileImageView.setImageURI(imageUri);
    }
    else
    {
        Toast.makeText(this, "Ошибка", Toast.LENGTH_SHORT).show();

        startActivity(new Intent(SettingsActivity.this,
SettingsActivity.class));
        finish();
    }
}
private void userInfoSaved(){
    if (TextUtils.isEmpty(fullNameEditText.getText().toString()))
    {
        Toast.makeText(this, "Заполните имя.", Toast.LENGTH_SHORT).show();
    }
    else if (TextUtils.isEmpty(addressEditText.getText().toString()))
    {
        Toast.makeText(this, "Заполните адрес", Toast.LENGTH_SHORT).show();
    }
    else if (TextUtils.isEmpty(userPhoneEditText.getText().toString()))
    {
        Toast.makeText(this, "Заполните номер", Toast.LENGTH_SHORT).show();
    }
    else if (checker.equals("clicked"))
    {
        uploadImage();
    }
}

private void uploadImage(){
    final ProgressDialog progressDialog = new ProgressDialog(this);
    progressDialog.setTitle("Обновляемся..");
    progressDialog.setMessage("Пожалуйста, подождите");
    progressDialog.setCanceledOnTouchOutside(false);
    progressDialog.show();
    if (imageUri != null)
    {
        final StorageReference fileRef = storageProfilePictureRef
            .child(prevalent.getCurrentOnlineUser().getPhone() +
".WebP");

```

					ПГУ 1.09.05.01.05.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		108

```

uploadTask = fileRef.putFile(imageUri);

uploadTask.continueWithTask(new Continuation() {
    @Override
    public Object then(@NonNull Task task) throws Exception
    {
        if (!task.isSuccessful())
        {
            throw task.getException();
        }
        return fileRef.getDownloadUrl();
    }
})
.addOnCompleteListener(new OnCompleteListener<Uri>() {
    @Override
    public void onComplete(@NonNull Task<Uri> task)
    {
        if (task.isSuccessful())
        {
            Uri downloadUrl = task.getResult();
            String myUrl = downloadUrl.toString();
            DatabaseReference ref =
FirebaseDatabase.getInstance().getReference().child("Users");
            HashMap<String, Object> userMap = new
HashMap<>();

            userMap. put ("name",
fullNameEditText.getText().toString());
            userMap. put ("address",
addressEditText.getText().toString());
            userMap. put ("phoneOrder",
userPhoneEditText.getText().toString());
            userMap. put ("image", myUrl);
            ref.child(prevalent.getCurrentOnlineUser().getPhone()).updateChildren(userMap);
            progressDialog.dismiss();
            startActivity(new Intent(SettingsActivity.this,
HomeActivity.class));
            Toast.makeText(SettingsActivity.this, "Информация успешно сохранена",
Toast.LENGTH_SHORT).show();

            finish();
        }
        else

```

					ПГУ 1.09.05.01.05.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		109

```

        {
            progressDialog.dismiss();
            Toast.makeText(SettingsActivity.this, "Error.", Toast.LENGTH_SHORT).show();
        }
    }
});

}
else
{
    Toast.makeText(this, "Изображение не выбрано.",
Toast.LENGTH_SHORT).show();
}
}

private void updateOnlyUser() {
    DatabaseReference ref =
FirebaseDatabase.getInstance().getReference().child("Users");

    HashMap<String, Object> userMap = new HashMap<>();
    userMap.put("name", fullNameEditText.getText().toString());
    userMap.put("address", addressEditText.getText().toString());
    userMap.put("phoneOrder", userPhoneEditText.getText().toString());
    ref.child(prevalent.getCurrentOnlineUser().getPhone()).updateChildren(userMap);
    startActivity(new Intent(SettingsActivity.this, HomeActivity.class));
    Toast.makeText(SettingsActivity.this, "Успешно сохранено",
Toast.LENGTH_SHORT).show();
    finish();
}

}

package space.vladek.diplo.ui;
import space.vladek.diplo.R;
import space.vladek.diplo.ui.Admin.AdminAddNewProfileActivity;
import space.vladek.diplo.ModelUser.Users;
import space.vladek.diplo.Prevalent.Prevalent;
import space.vladek.diplo.ui.Admin.AdminCategoryActivity;
import space.vladek.diplo.ui.Users.HomeActivity;
public class LoginActivity extends AppCompatActivity {
    private Button loginBtn;
    private EditText usernameInput, phoneInput, passwordInput;
    private ProgressDialog loadingBar;
    private TextView AdminLink, NotAdminLink;
    private String parentDbName = "Users";
    private CheckBox checkBoxRememberMe;

```

```

public Users usersData;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_login);
    loginBtn = (Button) findViewById(R.id.login_btn); //инициализация
кнопки
    phoneInput = (EditText) findViewById(R.id.login_phone_input);
//инициализация ввода номера
    passwordInput = (EditText) findViewById(R.id.login_password_input);
//инициализация ввода пароля
    loadingBar = new ProgressDialog(this); //прогресс бар
    AdminLink = (TextView) findViewById(R.id.admin_panel_link);
//инициализация входа от админа
    NotAdminLink = (TextView) findViewById(R.id.not_admin_panel_link);
//вход от пользователя
    checkBoxRememberMe = (CheckBox) findViewById(R.id.login_checkbox);
    Paper.init(this);
    AdminLink.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            AdminLink.setVisibility(View.INVISIBLE);
            NotAdminLink.setVisibility(View.VISIBLE);
            loginBtn.setText("Вход администратора");
            parentDbName = "Admins";
        }
    });
    NotAdminLink.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            AdminLink.setVisibility(View.VISIBLE);
            NotAdminLink.setVisibility(View.INVISIBLE);
            loginBtn.setText("Войти");
            parentDbName = "Users";
        }
    });
    loginBtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            loginUser();
        }
    });
}

```

```

    }

    private void loginUser() {
        String phone = phoneInput.getText().toString();
        String password = passwordInput.getText().toString();
        if (TextUtils.isEmpty(phone))
        {
            Toast.makeText(this, "Enter phone", Toast.LENGTH_LONG).show();
        }
        else if (TextUtils.isEmpty(password))
        {
            Toast.makeText(this, "Enter password",
Toast.LENGTH_LONG).show();
        }else
        {
            loadingBar.setTitle("Enter application");
            loadingBar.setMessage("Please Wait");
            loadingBar.setCanceledOnTouchOutside(false);
            loadingBar.show();
            ValidateUser(phone,password);
        }
    }

    private void ValidateUser(String phone, String password) {
        if (checkBoxRememberMe.isChecked()) {
            Paper.book().write(Prevalent.UserPhoneKey, phone);
            Paper.book().write(Prevalent.UserPasswordKey, password);
        } // если пользователь существует, то происходит запись в память
        телефона логина и пароля

        final DatabaseReference RootRef; //установка связи с базой данных
        RootRef = FirebaseDatabase.getInstance().getReference();

        RootRef.addListenerForSingleValueEvent(new ValueEventListener() {
            //запуск функции проверки номера и пароля
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                if(dataSnapshot.child(parentDbName).child(phone).exists())
            //Если в базе существуют значения
                {
                    usersData =
dataSnapshot.child(parentDbName).child(phone).getValue(Users.class);
            //Получение значения в переменной номера из базы

                    if(usersData.getPhone().equals(phone)) //если совпадает

```

					<i>ПГУ 1.09.05.01.05.001 ПЗ</i>	Лист
Изм.	Лист	№ докум.	Подпись	Дата		112

```

со значением из базы
{
    if (usersData.getPassword().equals(password)) //
если совпадает со значение из базы
    {
        if (parentDbName.equals("Users")) {
            loadingBar.dismiss();
            Toast.makeText(LoginActivity.this,
"Successful", Toast.LENGTH_LONG).show();
            Intent homeIntent = new
Intent(LoginActivity.this, HomeActivity.class);
            homeIntent.putExtra("keyName",
usersData.getName());
            homeIntent.putExtra("keyPhone",
usersData.getPhone());
            homeIntent.putExtra("keyPassword",
usersData.getPassword());
            homeIntent.putExtra("keyImage",
usersData.getImage());
            startActivity(homeIntent); //всплывающее
сообщение успешного входа и переход на домашнюю страницу
        }
        else if (parentDbName.equals("Admins")) {
            loadingBar.dismiss();
            Toast.makeText(LoginActivity.this,
"Successful", Toast.LENGTH_LONG).show();
            Intent homeIntent = new
Intent(LoginActivity.this, AdminCategoryActivity.class);
            startActivity(homeIntent); //переход на
страницу администрирования категорий
        }
        else {
            loadingBar.dismiss();
            Toast.makeText(LoginActivity.this,
"WrongPassword", Toast.LENGTH_LONG).show(); //сообщение неправильного ввода
        }
    }
} else {
    loadingBar.dismiss();
    Toast.makeText(LoginActivity.this, "Account with
number" + phone + "doesn't exist", Toast.LENGTH_LONG).show(); //так же , может

```

					ПГУ 1.09.05.01.05.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		113

нужен телефон

```
Intent registerIntent = new Intent(LoginActivity.this,
RegisterActivity.class);

startActivity(registerIntent); //если аккаунта с таким
номером не существует, то перевод на страницу регистрации
    }
}

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

}

});
}
```

```
package space.vladek.diplo.ui;
import java.util.HashMap;
import space.vladek.diplo.R;
public class RegisterActivity extends AppCompatActivity {
private Button registerBtn;
private EditText usernameInput, phoneInput, passwordInput;
private ProgressDialog loadingBar;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_register);
    registerBtn = (Button) findViewById(R.id.register_btn);
    usernameInput = (EditText) findViewById(R.id.register_username_input);
    phoneInput = (EditText) findViewById(R.id.register_phone_input);
    passwordInput = (EditText) findViewById(R.id.register_password_input);
    loadingBar = new ProgressDialog(this);
    registerBtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            CreateAccount();
        }
    });
}

private void CreateAccount(){
    String username = usernameInput.getText().toString();
    String phone = phoneInput.getText().toString();
    String password = passwordInput.getText().toString();
}
```

					ПГУ 1.09.05.01.05.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		114

```

        if (TextUtils.isEmpty(username))
        {
            Toast.makeText(this, "Enter name", Toast.LENGTH_LONG).show();
        }
        else if (TextUtils.isEmpty(phone))
        {
            Toast.makeText(this, "Enter phone", Toast.LENGTH_LONG).show();
        }
        else if (TextUtils.isEmpty(password))
        {
            Toast.makeText(this, "Enter password", Toast.LENGTH_LONG).show();
        }
        else
        {
            loadingBar.setTitle("MakeAccount");
            loadingBar.setMessage("Please Wait");
            loadingBar.setCanceledOnTouchOutside(false);
            loadingBar.show();

            ValidatePhone(username, phone, password);
        }
    }

    private void ValidatePhone(String username, String phone, String password)
    {
        final DatabaseReference RootRef;
        RootRef = FirebaseDatabase.getInstance().getReference();
        RootRef.addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                if (!(dataSnapshot.child("Users").child(phone).exists()))
                {
                    HashMap<String, Object> userDataMap = new HashMap<>();
                    userDataMap.put("phone", phone);
                    userDataMap.put("name", username);
                    userDataMap.put("password", password)

                    RootRef.child("Users").child(phone).updateChildren(userDataMap).addOnCompleteListener(new OnCompleteListener<Void>() {
                        @Override
                        public void onComplete(@NonNull Task<Void> task) {
                            if(task.isSuccessful())
                            {

```

					<i>ПГУ 1.09.05.01.05.001 ПЗ</i>	Лист
Изм.	Лист	№ докум.	Подпись	Дата		115

```

        loadingBar.dismiss();
        Toast.makeText(RegisterActivity.this,
"Registration successful", Toast.LENGTH_LONG).show();
        Intent loginIntent = new
Intent(RegisterActivity.this, LoginActivity.class);
        startActivity(loginIntent);
    } else
    {
        loadingBar.dismiss();
        Toast.makeText(RegisterActivity.this, "Error",
Toast.LENGTH_LONG).show();
    }
    }
    });
    } else
    {
        Toast.makeText(RegisterActivity.this, "Number" + phone +
"already exists", Toast.LENGTH_LONG).show();
        loadingBar.dismiss();
        Intent loginIntent = new Intent(RegisterActivity.this,
LoginActivity.class);
        startActivity(loginIntent);
    }
    }
    @Override
    public void onCancelled(@NonNull DatabaseError error) {
    }
    });
    }
}

```

```

package space.vladek.diplo.ViewHolder;
import space.vladek.diplo.Interface.ItemClickListener;
import space.vladek.diplo.ModelUser.Profiles;
import space.vladek.diplo.R;
public class ProfilesViewHolder extends RecyclerView.ViewHolder implements
View.OnClickListener {
    public TextView txtProfileName, txtProfileDescription, txtProfilePrice;
    public ImageView imageView;
    public ItemClickListener listner;
    public ProfilesViewHolder(View itemView)
    {

```

```

        super(itemView);
        imageView = itemView.findViewById(R.id.profile_image);
        txtProfileName = itemView.findViewById(R.id.profile_name);
        txtProfileDescription =
itemView.findViewById(R.id.profile_description);
        txtProfilePrice = itemView.findViewById(R.id.profile_price);
    }
    public void setItemClickListener(ItemClickListener listner)
    {
        this.listner = listner;
    }
    @Override
    public void onClick(View view)
    {
        listner.onClick(view, getAdapterPosition(), false);
    }
}

```

Приложение Г

(обязательное)

Android-приложение по выбору должностей на основе характеристики
личности и профессиональных навыков

Презентация

					<i>ПГУ 1.09.05.01.05.001 ПЗ</i>	Лист
Изм.	Лист	№ докум.	Подпись	Дата		118

Пензенский государственный университет
Дипломный проект на тему

Android-приложение по выбору должностей на основе характеристики личности и профессиональных навыков

Студент – Кирячек В.А.
Руководитель – Митрохин М.А.

Активация Windows
Чтобы активировать Windows, перейдите

- На данный момент, при подборе сильно влияет человеческий фактор, компетенции сотрудников на должности HR, зачастую не столь высоки и по этой причине, они не всегда имеют представление, какой человек в большей или меньшей степени подойдет на должность. Для решения данных проблем можно привлечь современные методы, а именно приложение по автоматическому подбору персонала. Данная работа предлагает использование приложения на смартфоны под управлением Android, с помощью которого руководитель сможет получать список наиболее подходящих кандидатов на искомое место.



Активация Windows
Чтобы активировать Windows, перейдите

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.09.05.01.05.001 ПЗ

Лист

119

Самым тяжелым этапом во всем процессе можно считать именно собеседование, так как оно сильно завязано на человеческом факторе, а именно:

- Профессиональная готовность человека, который будет заниматься отбором персонала в компанию
- Высокий уровень недоверия у руководителей к психологическим тестам.
- Специалист верит в тесты, умеет ими пользоваться, но не может сформулировать, что именно и какими именно методиками нужно мерить и как интерпретировать.



- - ОС Android – это открытый и бесплатный программный стек Google, который включает в себя операционную систему, промежуточное программное обеспечение, а также ключевые приложения для использования на мобильных устройствах, включая смартфоны.
- - ОС iOS – Apple iPhone OS изначально была разработана для использования на устройствах iPhone. Теперь мобильная операционная система называется iOS и поддерживается на ряде устройств Apple, включая iPhone, iPad и смарт часы. Мобильная операционная система iOS доступна только на устройствах Apple собственного производства, поскольку компания не лицензирует ОС для стороннего оборудования. Apple iOS является производной от операционной системы Apple Mac OS X.



Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.09.05.01.05.001 ПЗ

Лист

120

ОПРОСНИК КЕЙРСИ (ТИПОЛОГИЯ МАЙЕРС-БРИГГС)



Активация Windows
Чтобы активировать Windows, перейдите

Самыми распространенными средами разработки на Android на данный момент являются Eclipse, IntelliJ Idea и Android Studio. Каждая из них имеет, как ряд своих преимуществ, так и недостатков.



eclipse



IntelliJ IDEA

2017.2

Активация Windows
Чтобы активировать Windows, перейдите

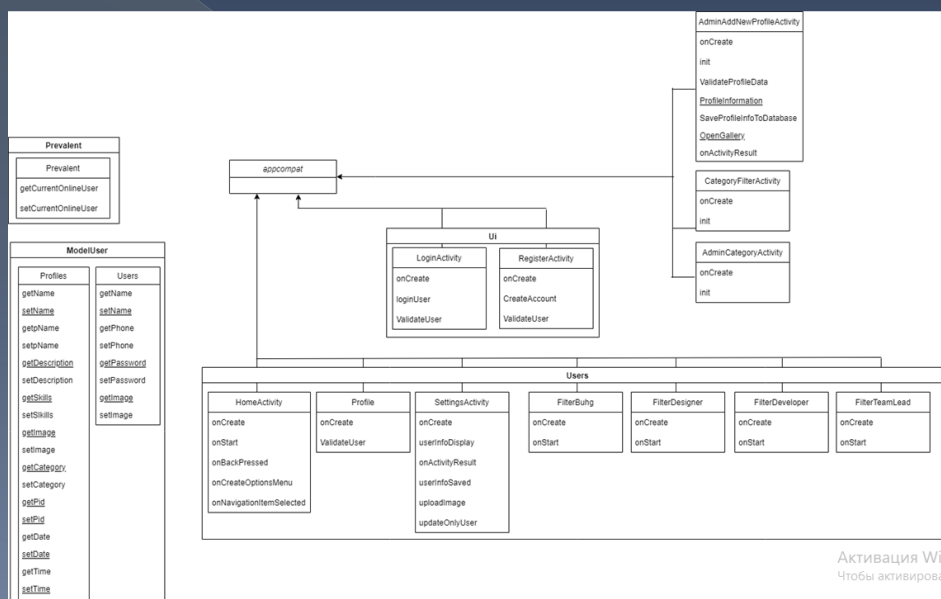
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.09.05.01.05.001 ПЗ

Лист

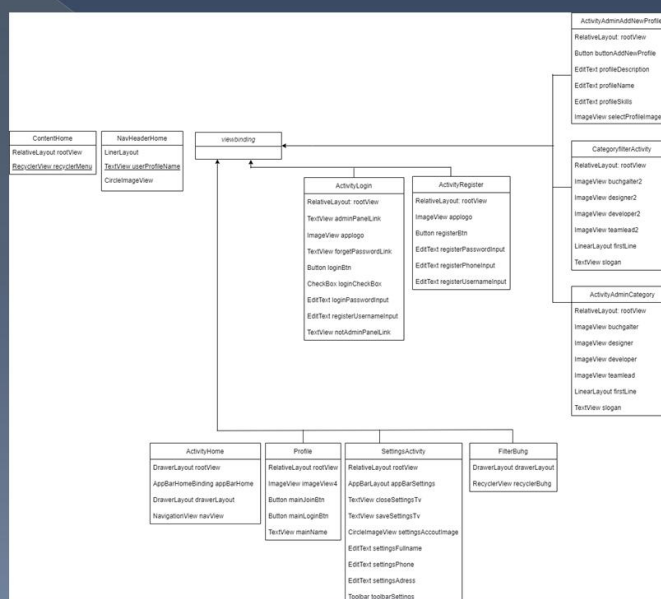
121

Диаграмма Java классов



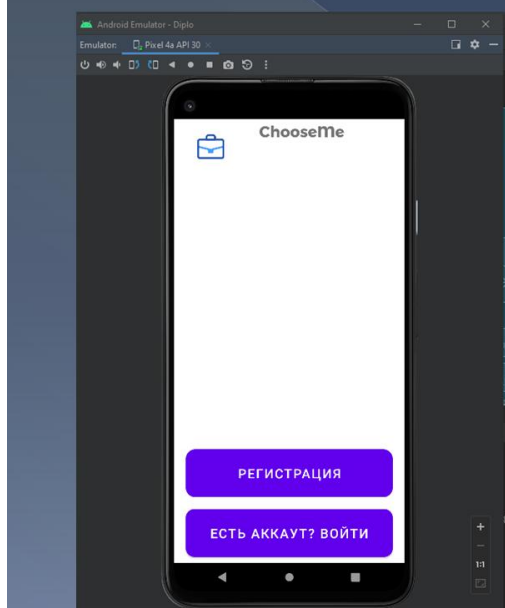
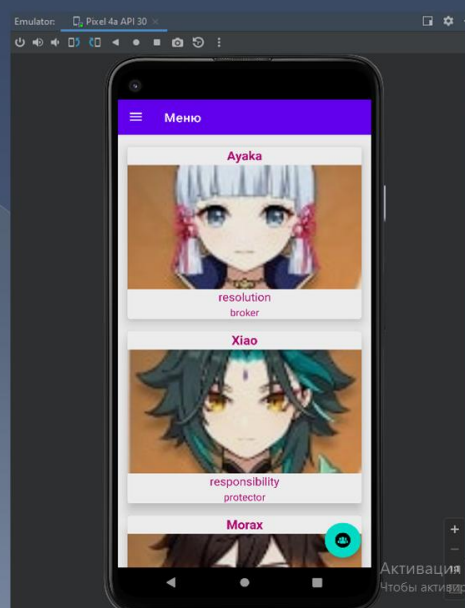
Активация Windows
Чтобы активировать Windows, перейдите на сайт Microsoft.

Диаграмма XML классов



Активация Windows
Чтобы активировать Windows, перейдите на сайт Microsoft.

Демонстрация работы



Активация Windows
Чтобы активировать Windows, перейдите на [microsoft.com/go/winactivate](#)