МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет вычислительной техники

Кафедра МО и ПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

БАКАЛАВРСКАЯ РАБОТА

на тему

Мобильное приложение банка

Студент _____ Щеваев Михаил Владимирович
(подпись, дата) (ФИО полностью)

Руководитель _____ Балашова И.Ю.
(подпись, дата) (фамилия, инициалы)

Нормоконтролер _____ Такташкин Д.В
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____№_____)

Заведующий кафедрой _____ Козлов А.Ю.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____№____)

Секретарь ГЭК _____ Попова Н.А.
(подпись) (фамилия, инициалы)

Пенза, 2023

Федеральное государственное бюджетное образовательное учреждение высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет	*Кафедра*
Вычислительной техники	МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____А.Ю. Козлов
«__» _____2023 г

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА

1. Студент Щеваев Михаил Владимирович гр. 19ВП2 факультета ВТ
<Фамилия, имя, отчество полностью>
2.Тема работы Мобильное приложение банка

Тема утверждена приказом ПГУ №_____от "___"_____2023 г.

3. Руководитель работы к.т.н., доцент кафедры «МО и ПЭВМ» Балашова И.Ю.

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

1) *Назначение: мобильное приложение банка с возможностью приобретения и хранения криптовалюты.*
2) *Среда выполнения: операционная система Android.*
Языковые средства разработки: Java, UML.
Инструментальные средства: Android Studio, StarUML, Parse android, Retrofit, Fragment API.
Технологии разработки: ООП, принципы RUP, низкоуровневые шаблоны проектирования.
3) *Основные функции приложения:*
а) добавить банковский счет;
б) добавить банковскую карту;
в) изменить информацию о себе в личном кабинете;
г) совершить перевод;
д) посмотреть историю операций;
е) купить криптовалюту;
е) оплатить жилищные услуги;
ж) оплатить кредит;
з) зарегистрироваться в системе;
и) авторизоваться в системе.

5. Перечень подлежащих разработке вопросов:

1) *Анализ предметной области и постановка задачи;*

2) *Проектирование программного обеспечения;*
3) *Реализация;*
4) *Контроль качества программного продукта;*
5) *Планирование и расчет бюджета;*
6) *Оформление пояснительной записки.*

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и постановка задачи	10%	12.04.2023 – 20.04.2023	
2	Проектирование программного обеспечения	25%	20.04.2023 – 12.05.2023	
3	Реализация	30%	12.05.2023 – 30.05.2023	
4	Контроль качества программного продукта	10%	30.05.2023 – 02.06.2023	
5	Планирование и расчет бюджета	5%	02.06.2023 – 04.06.2023	
6	Оформление пояснительной записки	20%	04.06.2023 – 20.06.2023	

Дата выдачи задания «22» ____декабря____2022 г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан факультета _____
(подпись, дата)

Реферат

Пояснительная записка содержит 97 листов, 33 рисунка, 8 таблиц, 8 использованных источников и 4 приложения.

ГРАФИЧЕСКОЕ ПРЕДСТАВЛЕНИЕ, КЛИЕНТСКАЯ ЧАСТЬ, СЕРВЕРНАЯ ЧАСТЬ, БАЗА ДАННЫХ, UML, REST.

Объектом проектирования и разработки является мобильное приложение банка, предоставляющее доступ к покупке криптовалюты.

Цель выпускной квалификационной работы – разработка мобильного приложения, предоставляющего услуги интернет-банкинга с возможностью взаимодействия с криптовалютой.

Технологии разработки – разработка клиентской и серверной части проводилась на языке программирования Java, для взаимодействия с данными использовалась платформа с открытым исходным кодом Parse android. В качестве среды разработки была выбрана интегрированная среды разработки Android Studio.

В результате выполнения выпускной квалификационной работы было разработано универсальное мобильное приложения банка, взаимодействующее с криптовалютой.

Содержание

Введение 5
1. Анализ решений со схожими возможностями 6
1.1 Анализ предметной области и постановка задачи 6
1.2 Анализ аналогичных мобильных приложений 8
1.3 Сравнительный анализ аналогов 9
1.4 Анализ функциональных требований 11
1.5 Анализ нефункциональных требований 13
2 Проектирование сервиса оказания услуг интернет-банкинга 15
2.1 Архитектура мобильного приложения 15
2.2 Проектирование структуры базы данных 16
2.3 Проектирование программных средств 17
2.4 Проектирование пользовательского интерфейса 18
3 Реализация мобильного приложения 35
3.1 Реализация серверной части 35
3.2 Реализация клиентской части 36
4 Контроль качества программного продукта 38
4.1 Тестирование приложения черным ящиком 38
4.1.1 Классы эквивалентности 38
4.1.2 Граничные условия 38
4.1.3 Тест-кейсы 39
4.2 Функциональное тестирование белым ящиком 39
4.3 Планирование и расчет бюджета разрабатываемого приложения 41
Заключение 44
Список использованных источников 45
Приложение А. Глоссарий 46
Приложение Б. Исходный код серверной части 48
Приложение В. Исходный код клиентской части 69
Приложение Г. Результаты тестирования 88

Введение

За последние пару десятилетий банковская система Российской Федерации претерпела значительные изменения.

Цифровизация сферы услуг привела к появлению альтернативы фиатным денежным средствам. Этой альтернативой стала криптовалюта. Ученые и разработчики наладили инфраструктуру цифровой среды, что позволило использовать криптовалюту как средство обмена во время сделок. Люди начали покупать и продавать виртуальную валюту на специализированных биржах. Но, несмотря на стремительные темпы развития этой альтернативы, рядовые пользователи останавливаются перед использованием криптовалюты в повседневной жизни. На это влияют особенности хранения виртуальных средств. Если держать криптовалюту на бирже, то ее обвал приведет к потере всех накоплений. Горячие кошельки требуют отдельного приложения или наличие аккаунта на специализированном сайте. Холодные кошельки подвержены сбоям в работе из-за отсутствия постоянного подключения к интернету. Аппаратные кошельки требуют отдельный носитель информации. Преодолеть эти трудности в использовании криптовалюты, может помочь универсальное мобильное приложение банка. Знакомый большинству функционал перевода средств, оплаты кредитов и жилищных услуг, сочетая в себе простой интерфейс взаимодействия с криптовалютой, предоставит новые возможности пользователям, сэкономив их время и деньги.

Исходя из вышесказанного, можно понять, что целью данной работы является реализация приложения для мобильных устройств, предоставляющего возможности интернет-банкинга с простым доступом к криптовалюте.

1. Анализ решений со схожими возможностями

1.1 Анализ предметной области и постановка задачи

В данный момент времени доступ к финансовым услугам является важной частью жизни общества. Мобильные приложения стали главным средством привлечения новых пользователей. Возросший поток клиентов привел к расширению сферы услуг. Это явление породило массу узконаправленных программных решений. Из-за своей узкой направленности большинство приложений никак не взаимодействуют между собой, что приводит к путанице и бесполезной трате времени клиентов.

В свою очередь Правительство Российской Федерации и Банк России продолжают оказывать поддержку развития цифровизации финансового рынка. В рамках распоряжения Правительства РФ «Об утверждении Стратегии развития финансового рынка РФ до 2023 года» внимание будет направлено на создание условий для внедрения инноваций на российском финансовом рынке. Планируется выработать подходы по использованию децентрализованных финансовых инструментов и регулированию операций с цифровыми активами (криптовалютой) [1].

Исходя из быстрого развития финансового сектора и активного содействия со стороны государства можно сделать вывод, что разработка мобильного приложения, предметная область которого представляет сферу оказания услуг интернет-банкинга, является актуальной и востребованной.

Для того, чтобы решить задачу объединения разрозненных сервисов, специализирующихся на оказании платежных услуг, необходимо проанализировать вероятные варианты использования разрабатываемого приложения.

Разрабатывая приложение мобильного банка, взаимодействующего с криптовалютой, важно помнить об удобстве обслуживания данной системы. Для этого необходимо структурировать информацию по каждому пользователю. Эту задачу можно выполнить путем добавления в базу данных

сведений о клиентах, хранящихся в файлах формата JSON. От каждого нового пользователя будет требоваться следующий набор параметров при регистрации:

– полное имя;
– идентификатор учетной записи;
– пароль;
– номер телефона;
– адрес проживания;
– выбор рода деятельности из выпадающего списка.

На основном экране необходимо реализовать добавление банковских счетов и функцию перевода денежных средств между ними, оформление кредитных карт, просмотр истории операций.

Для оформления кредита клиенту нужно будет ввести необходимую сумму и срок погашения займа в месяцах.

От экрана с криптовалютой требуется вывод доступных позиций виртуальной валюты, предоставление возможности покупки необходимого количества цифровых активов. Купленная криптовалюта будет привязана к данным конкретного пользователя, что упростит ее дальнейшее использование на других сервисах.

Чтобы упростить документирование оплаченных жилищно-коммунальных услуг, следует сохранять на экране оплаты счетов электронные чеки с названием услуги, суммой оплаты и датой оформления.

В личном кабинете должна присутствовать возможность изменения персональной информации. База данных, использующая хэш-таблицы, позволит быстро добавить внесенные корректировки.

На каждом этапе работы приложения с данными, функции, отдельно взятых классов, будут искать, добавлять или удалять пары данных вида «ключ-значение». Такой подход минимизирует риск потери информации при сбоях интернет-соединения.

Описанный выше анализ вероятных вариантов использования дает понять, что целью данной работы является создание мобильного приложения, предоставляющего простое взаимодействие с криптовалютой и оказывающего услуги интернет-банкинга.

1.2 Анализ аналогичных мобильных приложений

Во время исследования рынка финансовых приложений были выявлены следующие программные продукты-аналоги: «СберБанк Онлайн», «Тинькофф онлайн банк», «Альфа-Банк».

«СберБанк Онлайн» является мобильным приложением, предоставляющим свои клиентам следующий набор услуг:

– просмотр информации по счетам;
– пополнение счета;
– осуществление переводов между картами и счетами;
– оплата кредитов;
– оплата услуг юридических лиц;
– оплата коммунальных услуг.

Приложение «СберБанк Онлайн» входит в крупную экосистему «Сбера», которая насчитывает более 40 компаний и сервисов. Для того, чтобы пользовательский опыт взаимодействия с таким количеством услуг не вызывал неудобств, был введен единый идентификатор клиента. Эта технология упростила обработку платежей и переводов потребителей [2].

«Тинькофф онлайн банк» – приложение для смартфонов, обладающее простым в использовании интерфейсом. Удобная лента событий отображает историю финансовых операций. Отдельного внимания заслуживает реализация бонусных предложений от партнеров банка. Чтобы получить бонусы в виде кешбэка, нужно выбрать интересующие категории спецпредложений и перейти к покупкам [3].

«Альфа-Банк» – решение для мобильных устройств, которое предлагает все виды финансовых услуг, включая инвестиции, накопительные счета,

кредитование, управление активами и участие в партнерских программах. Присутствуют гибкие инструменты учета средств [4].

1.3 Сравнительный анализ аналогов

Для лучшего понимания технических возможностей продуктов-аналогов выполним сравнительный анализ этих сервисов.

Выберем критерии оценки и выделим коэффициенты значимости для каждого критерия оценки возможностей рассматриваемых программ. Наименования критериев и их коэффициенты значимости представлены в таблице 1.

Таблица 1 – Критерии оценки мобильных приложений банков

№	Наименование
1	Просмотр информации по счетам
2	Выполнение переводов на сторонние счета
3	Возможность просмотра истории операций
4	Оплата коммунальных услуг
5	Редактирование данных пользователя в личном кабинете
6	Предоставление интерфейса для услуг кредитования
7	Формирование и документирование чеков по оказанным услугам
8	Составление индивидуальных предложений от партнеров сервиса
9	Расширенные возможности администрирования
10	Оперативное добавление информации в базу данных
11	Поддержка других операционных систем
12	Адаптация интерфейса приложения под размеры экрана устройства
13	Простой для понимания дизайн сервиса
14	Интерфейс взаимодействия с криптовалютой

Вышеописанные критерии будем оценивать по трехбалльной системе, где 0 – «полное несоответствие», а 2 – «полное соответствие». Таблица 2 содержит описание характеристик оценки требований с соответствующим количеством баллов.

Таблица 2 – Характеристики оценки требований

Балл	Характеристика оценки
0	функционал нереализован
1	функционал реализован частично
2	функционал полностью реализован

Для выявления сильных сторон программных возможностей продуктов-аналогов, было проведено их сравнение. Сравнительному анализу подверглись следующие программные продукты-аналоги:

– «СберБанк Онлайн»;

– «Тинькофф онлайн банк»;

– «Альфа-Банк».

Основываясь на особенностях критериев оценки функционала приложений, заполним таблицу 3. Для этого выполним расчет аддитивной суммы интегральной оценки для отобранных сервисов по формуле:

$$О_{пп} = \sum_{i=1}^{n} K_i,$$

где $О_{пп}$ – интегральная оценка продукта-аналога;

n – количество критериев оценки;

K_i – значение выполнения критерия (0 – не выполнен, 2 – выполнен);

Таблица 3 – Критериальная оценка продуктов-аналогов

№	Критерий	Продукт-аналог №		
		1	2	3
1	Просмотр информации по счетам	2	2	2
2	Выполнение переводов на сторонние счета	2	2	2
3	Возможность просмотра истории операций	2	2	2
4	Оплата коммунальных услуг	2	2	2
5	Редактирование данных пользователя в личном кабинете	2	1	1
6	Предоставление интерфейса для услуг кредитования	2	2	2
7	Формирование и документирование чеков по оказанным услугам	2	2	1
8	Составление индивидуальных предложений от партнеров сервиса	2	2	2
9	Расширенные возможности администрирования	2	1	1
10	Оперативное добавление информации в базу данных	1	1	1
11	Поддержка других операционных систем	2	2	2
12	Адаптация интерфейса приложения под размеры экрана устройства	2	2	2
13	Простой для понимания дизайн сервиса	1	2	1
14	Интерфейс взаимодействия с криптовалютой	0	0	0
	Интегральная оценка	24	22	21

Несмотря на широкий спектр оказываемых услуг, вышеописанные мобильные приложения банков не имеют интерфейса взаимодействия с криптовалютой, что значительно ограничивает пользователей современной сферы финансов. Решить эту проблему поможет универсальное мобильное приложение, разработка которого является целью данной работы.

1.4 Анализ функциональных требований

Функциональные требования играют важную роль в определении поведения программного продукта. Они описывают возможности программы, которые должен спроектировать и разработать программист. Каждое требование к функционалу системы повышает ее удобство использования пользовтелем.

На рисунке 1 изображена диаграмма вариантов использования, которая была составлена в соответствии с функциональными требованиями. Диаграмма показывает последовательность порядка выполнения ролью «Пользователь» варианта использования «Зарегистрироваться в системе», предназначенный для создания учетной записи клиента.

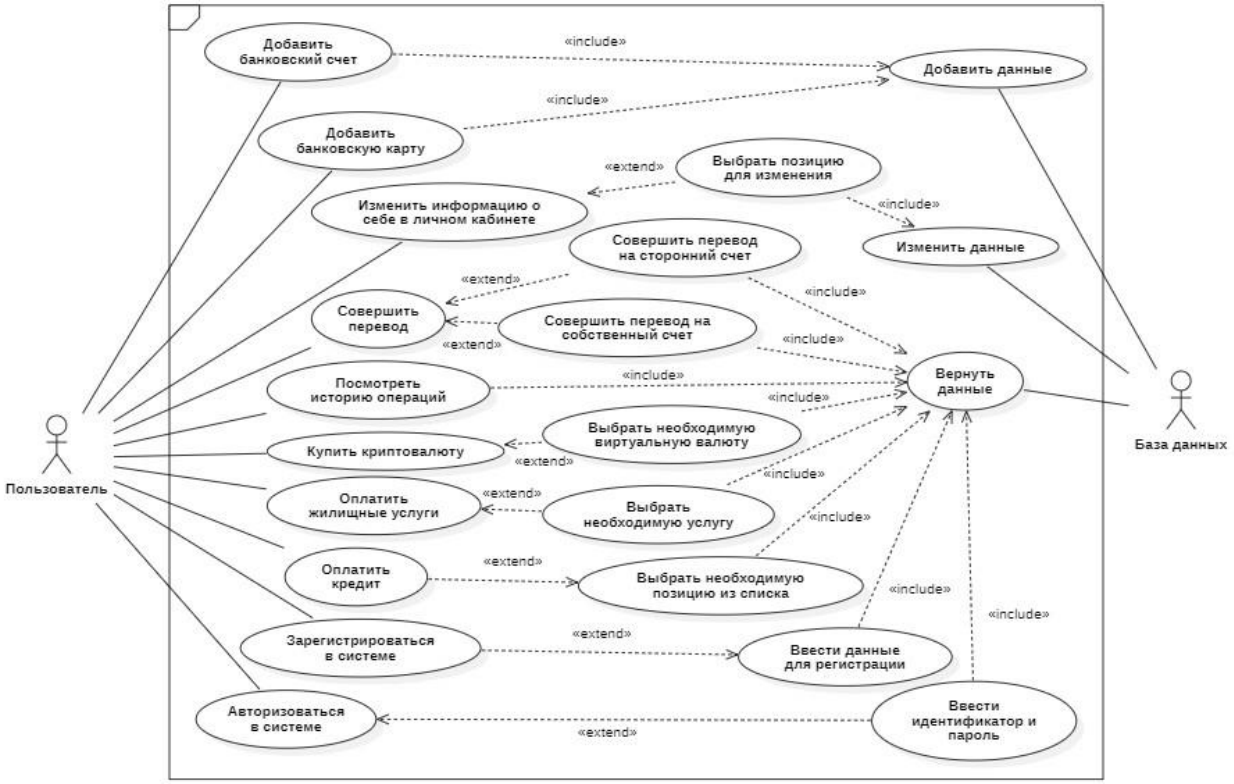


Рисунок 1 – Диаграмма вариантов использования

На рисунке 2 представлена последовательность выполнения ролью «Пользователь» варианта использования «Зарегистрироваться в системе». Эта функция играет важную роль в реализации индивидульного подходе к клиентам. В данном приложении формы регистрации предлагают ввести

пользователю данные, начиная от его полного имени до выбора вида деятельности. Подробный сбор личной информации помогает составить точный портрет потребителя, что упрощает составление индивидуальных предложений под конкретного клиента.

Прецедент: Зарегистрироваться в системе.
ID: 1
Описание: Пользователь создает новую учетную запись в приложении.
Главные актеры: Пользователь.
Второстепенные актеры: Отсутствуют.
Предусловия: 1. Пользователь открыл мобильное приложение. 2. Пользователь перешел к экрану регистрации.
Основной поток: 1. Пользователь ввел все необходимые данные. 2. Пользователь нажал на кнопку регистрации.
Постусловия: 1. Приложение отображает главный экран новой учетной записи.
Альтернативные потоки: Отсутствуют.

Рисунок 2 – Специализация для варианта использования «Зарегистрироваться в системе»

На рисунке 3 представлена последовательность выполнения ролью «Пользователь» варианта использования «Совершить перевод».

Прецедент: Совершить перевод.
ID: 3
Описание: Пользователь совершает денежный перевод по номеру счета.
Главные актеры: Пользователь.
Второстепенные актеры: Отсутствуют.
Предусловия: 1. Пользователь открыл выпадающее окно перевода средств.
Основной поток: 1. Пользователь выбрал счет для перевода. 2. Пользователь выбрал с какого банковского счета будут списаны деньги. 3. Пользователь ввел сумму перевода. 4. Пользователь подтвердил перевод.
Постусловия: 1. Приложение перевело денежные средства по номеру счета. 2. Приложение задокументировало перевод денег в истории операций.
Альтернативные потоки: Отсутствуют.

Рисунок 3 – Спецификация для варианта использования «Совершить перевод»

На рисунке 4 приведено описание порядка выполнения ролью «Пользователь» варианта использования «Купить криптовалюту».

Прецедент: Купить криптовалюту.
ID: 4
Описание: Пользователь совершает покупку криптовалюты.
Главные актеры: Пользователь.
Второстепенные актеры: Отсутствуют.
Предусловия: 1. Пользователь открыл окно покупки криптовалюты.
Основной поток: 1. Пользователь выбрал необходимую ему криптовалюту. 2. Пользователь выбрал нужное количество виртуальной валюты. 3. Пользователь подтвердил приобретение.
Постусловия: 1. Приложение добавило требуемое количество криптовалюты в криптовалютный кошелек пользователя. 2. Приложение задокументировало покупку криптовалюты в истории операций.
Альтернативные потоки: Отсутствуют.

Рисунок 4 – Спецификация для варианта использования «Купить криптовалюту»

В результате анализа функциональных требований были смоделированы бизнес-процессы, позволяющие обслуживать пользователей финансовой сферы, обеспечивая при этом простой и поятный доступ к криптовалюте.

1.5 Анализ нефункциональных требований

Разработка данного приложения должна быть выполнена в интегрированной среде разработки Android Studio на языке программирования Java. Программа Android Studio содержит обширный набор иструментов разарботки. Редактор графического представления упрощает компановку элементов программы. Возможность использования языка разметки гипертесткта заметно облегчает настройку отствупов графических объектов относительно экрана устройства. Реализовать работу с данными требуется с использованием платформы Parse android. От мобильных приложений

требуется высокая оптимизация, что обеспечивает широкий охват пользователей. Платформа Parse android решает проблему оптимизации хранения и извлечения данных. Для взаимодействия с клиентской частью необходимо воспользоваться технологий Retrofit. При создании интерфейса приложения нужно использовать Fragment API, что значительно упростит компоновку модулей с информацией на экране мобильного устройства. Создание визуальных элементов для приложения необходимо выполнить в многофункциональном графическом редакторе Adobe Photoshop. Мобильное приложение должно работать на операционной системе Android.

2 Проектирование сервиса оказания услуг интернет-банкинга

2.1 Архитектура мобильного приложения

При создании мобильного приложения была использована многоуровневая архитектура. Схематичное представление данной архитектуры приведено на рисунке 5.

Рисунок 5 – Архитектура клиент-сервер

Многоуровневая архитектура содержит три слоя:

– презентационный слой;

– слой бизнес-логики;

– слой доступа к данным.

Презентационный слой представляет визуальный функционал приложения. При создании этого слоя разработчики программного обеспечения уделяют особое внимание дизайну пользовательского интерфейса. Работа над дизайном приложения заключается в выстраивании интуитивно понятной навигации сервиса, создании визуальных компонентов взаимодействия. Слой бизнес-логики отвечает за обработку событий и обмен данными. На этом уровне система выполняет сценарии ее использования, к ним относятся: кэширование, передача данных, обработка исключений, выполнение различных операций. Слой доступа к данным отвечает за обработку и сохранение информации. Компоненты этого слоя обеспечивают функциональный доступ к данным [5].

Передача состояний от презентационного слоя к слою бизнес-логики осуществляется по протоколу HTTP при помощи REST API.

2.2 Проектирование структуры базы данных

При проектировании структуры базы данных были рассмотрены возможности современных подходов к разработке хранилищ информации, отвечающих за взаимодействие с ней. К одному из таких подходов относится составление данных ввиде таблиц. Базы такого типа называются реляционными. Перед использованием данного решения необходимо заранее определить схему, по которой программа будет записывать информацию. Особенность этого подхода ограничивает гибкий подход к разработке программного обеспечения, усложняеет структуру кода и снижает скорость работы программы. Противоположностью данного решения являются нереляционные базы данных. В них отсутствуют ограничения формирования информаци, что позволяет быстро адаптировать базу к меняющимся типам данным в конкретный момент работы приложения.

Основываясь на анализе возможностей нереляционных баз данных, выбрана платформа с открытым исходным кодом, которая называется Parse android. Этот сервис решает проблему написания серверной части вручную. Программное решение Parse android позволяет разработчикам быстро создавать мобильные приложения с общими данными, без написания пользовательских API. Подход к составлению логически структурированных моделий данных в Parse andoid отличается от релиационного подхода к формированию хранилищ информации. Данная платформа позоляет добавлять произольные поля информации без предварительной декларации изменения всей таблицы. Любое изменение состояния данных в базе сводится к редактированию структуры кода, отвечающего за взаимодействие с ней.

Parse android – это программа, которая создает интерфейс для автоматического сбора и хранения информации. Благодаря объединению инструментов разработки для мобильных устройств и гибкой серверной части Parse android обладает следующими возможностями:

– регистрация пользователя с последующей аутентификацией;
– создание, изменение и удаление данных;

– упрощение отправки уведомлений;
– загрузка файлов на сервер для общего доступа.

2.3 Проектирование программных средств

Основываясь на требованиях к функционалу разрабатываемого приложения, была создана диаграмма компонентов, описывающая структуру проекта. Краткая спецификация компонентов программного продукта представлена на рисунке 6.

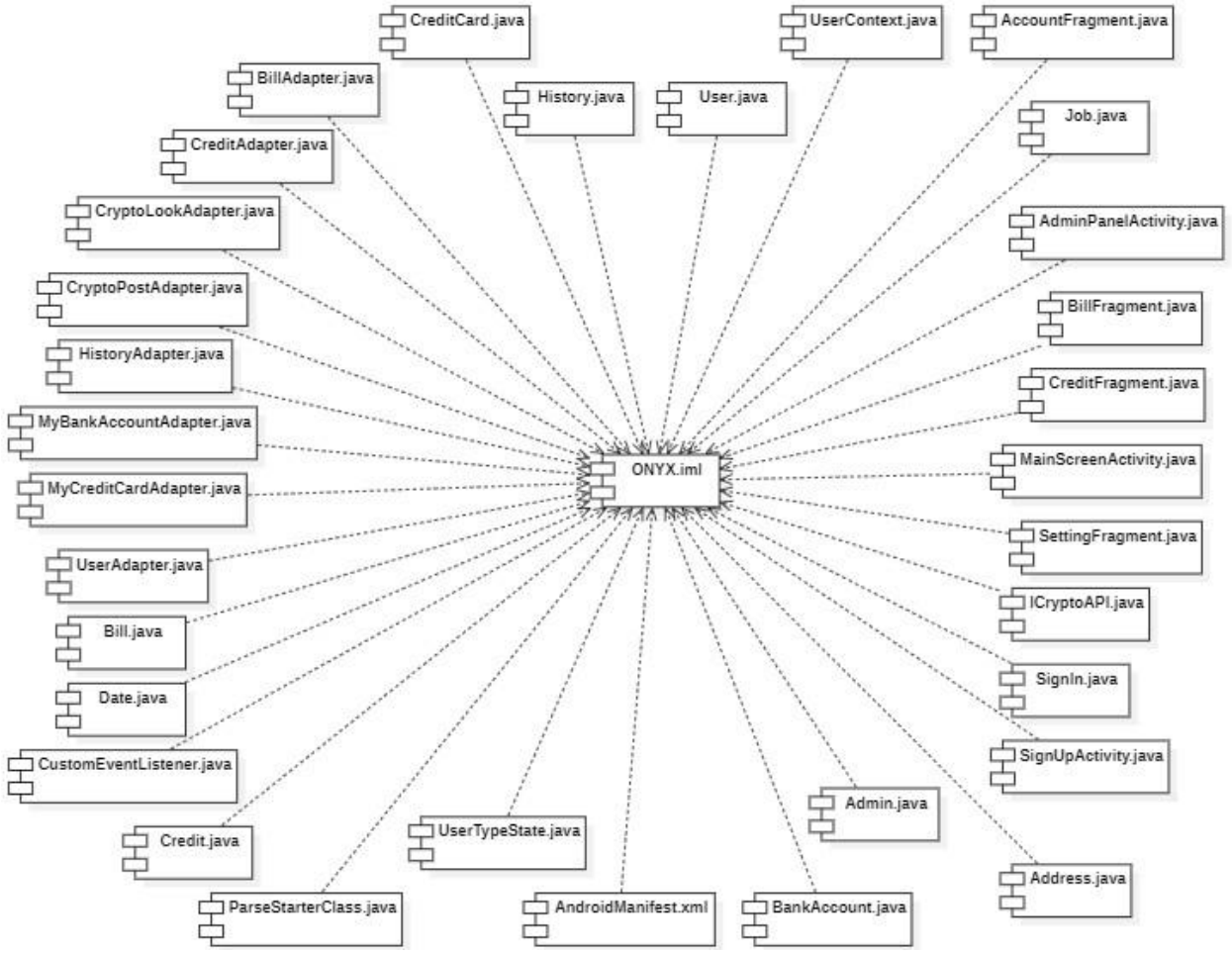


Рисунок 6 – Диаграмма компонентов

Рассмотрим описание назначений наиболее значимых классов с файлов регистрации и авторизации пользователей. Класс SignUpActivity реализует функции записи вводимой клиентом информации. Каждому новому пользователю присваивается идентификатор учетной записи, который служит

ключом ко всем остальным его данным. Класс SignIn проверяет пароль и идентификатор при входе в систему. За отображение навигационной панели внизу экрана устройства и переходы между активностями отвечает класс MainScreenActivity. Верхняя панель главного экрана отображает полное имя пользователя, общую сумму денежных средств, текущую дату. За вывод этой информации и предоставление основных финансовых услуг отвечает класс AccountFragment. Класс CreditFragment отвечает за оформление и оплату кредита. Проверку параметров подключения к стороннему источнику информации за актуальномым курсам криптовалюты отвечает класс ICryptoAPI. Формированием списка доступных криптовалют занимается класс CryptoLookAdapter. Покупку виртуальной валюты реализует класс CryptoPostAdapter. Используя логику класса BillAdapter, класс BillFragment реализует оплату и документирование жилищных услуг. За доступ к данным со стороны администрации сервиса отвечает класс AdminPanelActivity. Класс ParseStarterClass выполняет обработку и сохранение данных в базе. Работа с функционалом личного кабинета происходит в классе UserAdapter. За настройку кабинета отвечает класс UserAdapter.

2.4 Проектирование пользовательского интерфейса

Исходя из диаграммы вариантов использования, был разработан графический интерфейс приложения. Пользовательский интерфейс должен быть прост и понятен в использовании.

Для проектирования визуально части сервиса был выбран многофункциональный редактор растровой графика Adobe Photoshop. Редактор обладает большими возможностями в сфере проектирования и всерстки программных продуктов. Широкий выбор инструментов данной программы позволил спроектировать внешний вид клиентской части, еще до ее функциональной реализации, что позволило сократить время разарботки.

В данном программном обеспечении знакомство с сервисом начинается с экрана авторизации и регистрации. На рисунке 7 показан экран авторизации с подробным описанием его элементов.

Рисунок 7 – Экран авторизации

Основные элементы интерфейса, показанного на рисунке 7:

1 – поле ввода идентификатора пользователя, передающее в базу данных уникальное числовое значение учетной записи клиента;

2 – поле ввода пароля от аккаунта;

3 – кнопка, подтверждающая вход;

4 – кнопка перехода к экрану регистрации.

При регистрации новой учетной записи от пользователя требуется ввести обширный список данных. Данная информация заносится в соответствующие поля файла типа JSON, что позволяет осуществить реализацию изменений личных данных в кабинете клиента. Часть введенных данных должна использоваться при авторизации. На рисунке 8 изображен интерфейс регистрации.

Рисунок 8 – Экран регистрации

Элементы экрана регистрации рисунка 8:

1 – поле ввода полного имени пользователя;

2 – поле ввода идентификатора учетной записи;

3 – поле ввода пароля;

4 – поле ввода номера телефона;

5 – выпадающий список полей ввода адреса проживания клиента;

6 – список, предлагающий выбрать один из видов деятельности;

7 – кнопка, подтверждающая регистрацию.

Главный экран приложения представлен на рисунке 9. Эта часть сервиса выполняет основные услуги интернет-банкинга. Удобное разделение функциональных частей этого экрана обеспечивает простоту использования возможностей приложения.

Рисунок 9 – Главный экран

К элементам главного экрана разрабатываемого приложения, представленного на рисунке 9, относятся:

1 – общая информация;

2 – совокупность денежных средств всех банковских счетов пользователя;

3 – кнопка запроса денежных средств;

4 – кнопка, позволяющая перевести деньги на собственный и сторонний счет;

5 – кнопка, позволяющая посмотреть историю операций;

6 – кнопка добавления нового банковского сета;

7 – раздел банковских счетов;

8 – кнопка, позволяющая добавить новую банковскую карту;

9 – раздел банковских карт;

10 – панель навигации по основным экранам приложения.

Рассмотрим подробное проектирование процесса перевода средств стороннему пользователю. Моделирование процессов данного функционала играет важную роль в формировании общего сценария совершения финансовых транзакций между клиентами системы.

На рисунке 10 показано окно выбора действия. Пользователь может перевести денежные средства со одного счета на другой.

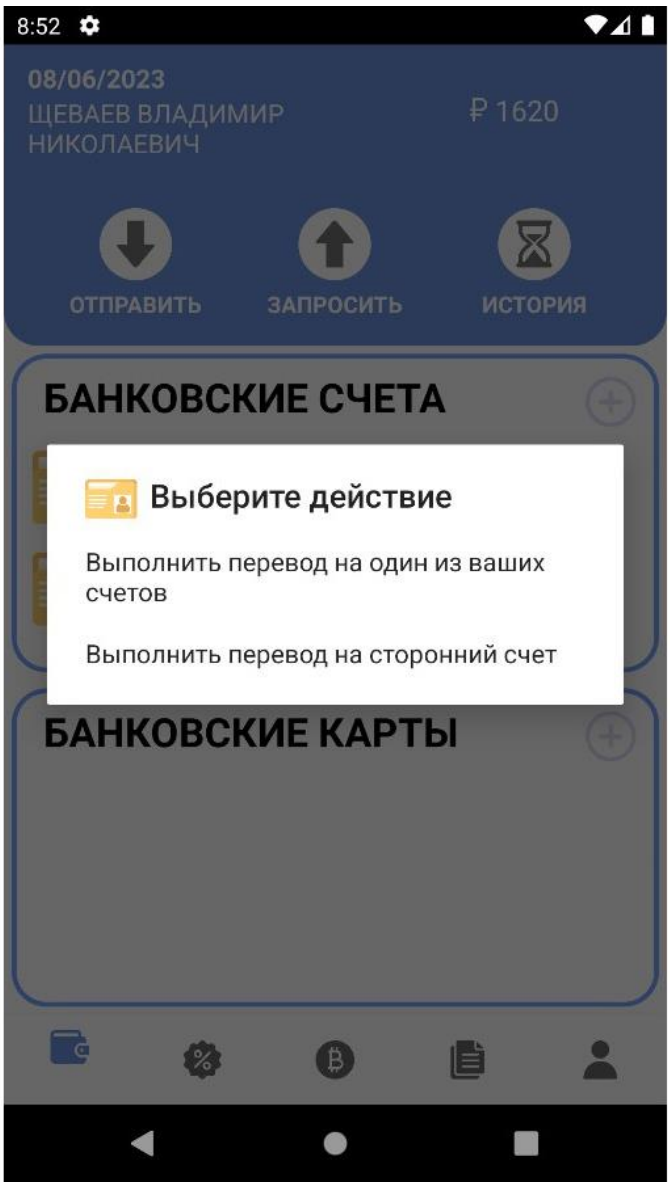


Рисунок 10 – Окно выбора действия

Программируя процесс перевода, средствами среды разработки, ограничим ввод данных цифровым значением. Внешний вид данной процедуры изображен на рисунке 11.

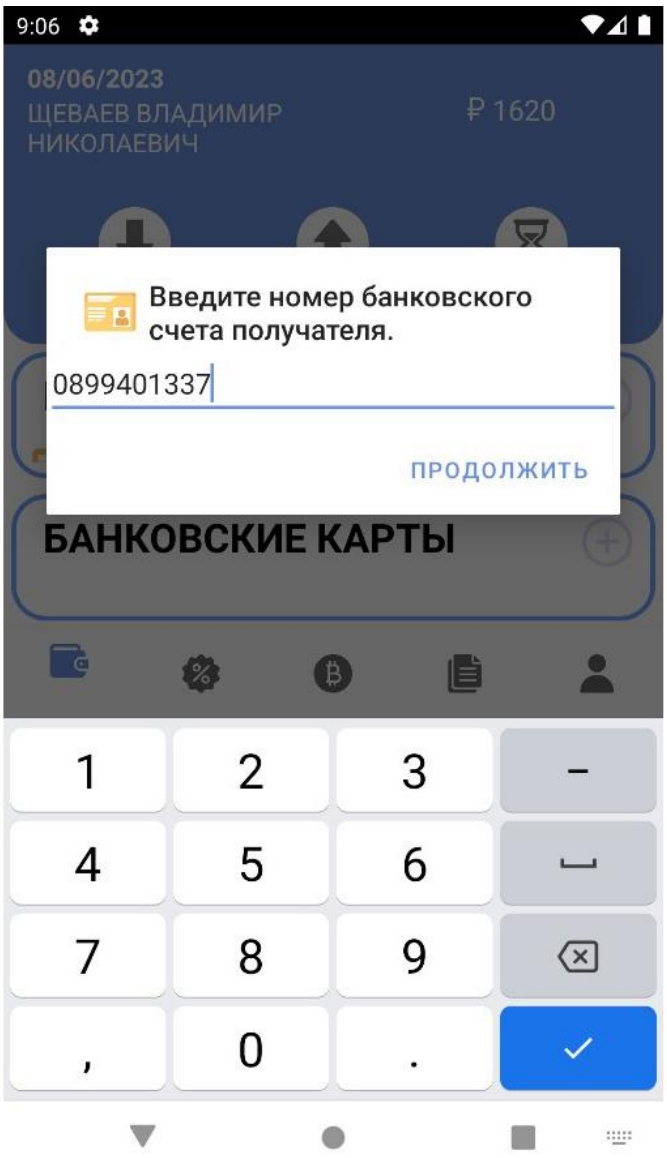


Рисунок 11 – Процесс ввода номера счета

Для улучшения эргономики, разрабатываемого интерфейса, смоделируем внешний вид выбора пользователем банковского счета. Окна цепочки событий перевода денежных активов не должны вызывать затруднений в использовании. Для этого ограничим количество необходимих действий на каждом этапе.

Индивидуальный подход к клиентам можно реализовать, используя вопросительные предложения. Так приложение, посторенное на жесткой логике бизнес-процессов, стает верным помошником в решении финансовых операций. Этот положительный опыт использования сервиса может сказаться

на удержании новой аудитории. Один из пример данного подхода представлен на рисунке 12.

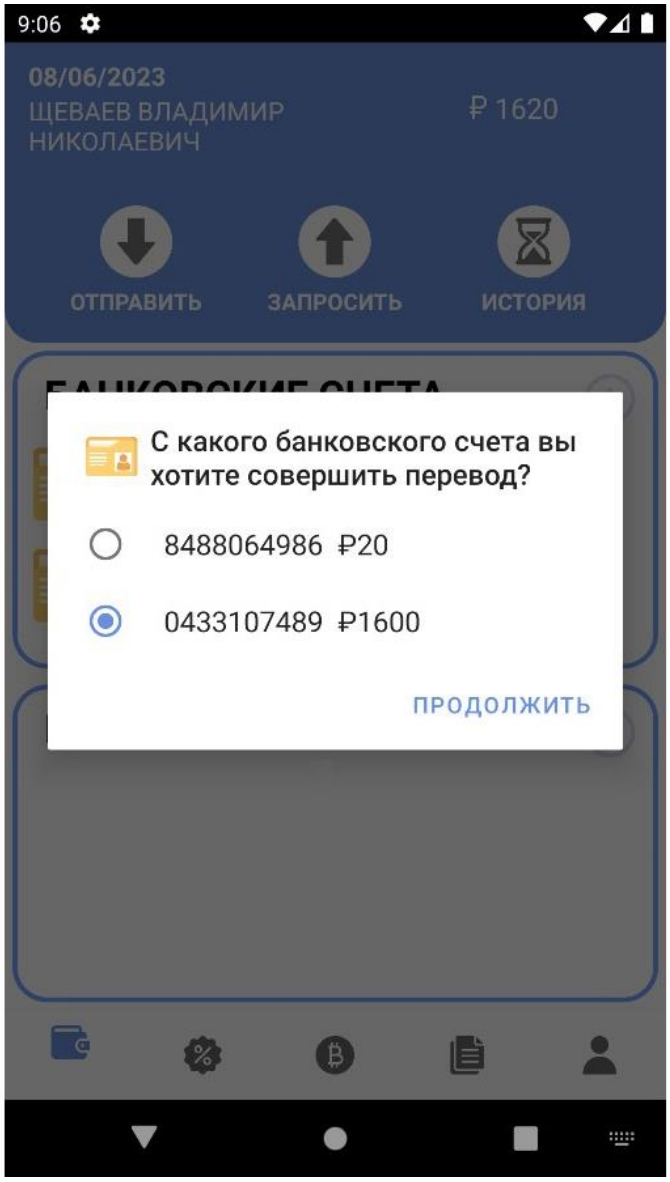


Рисунок 12 – Процесс выбора пользователей одного из его счетов

Возможности среды разработки Android Studio предоставляют доступ к настройке отображения графических ресурсов. Встроенные стили и нстройка компонентов создают простор для реализации шаблонов, созданых в графическом редакторе. Смоделируем внешний вид процесса ввода переводимой суммы. Для этого воспользуемся ограничение количества действий пользователя на одном экране. Изобразим цифровую клавиатуру операционной системы Android, добавим поле ввода, обозначим переход к

следующему действия кнопкой с соответствующей надписью. Затемним задий фон, оставив в зоне видимости основную информацию. Подробное представление проектируемого процесса показано на рисунке 13.

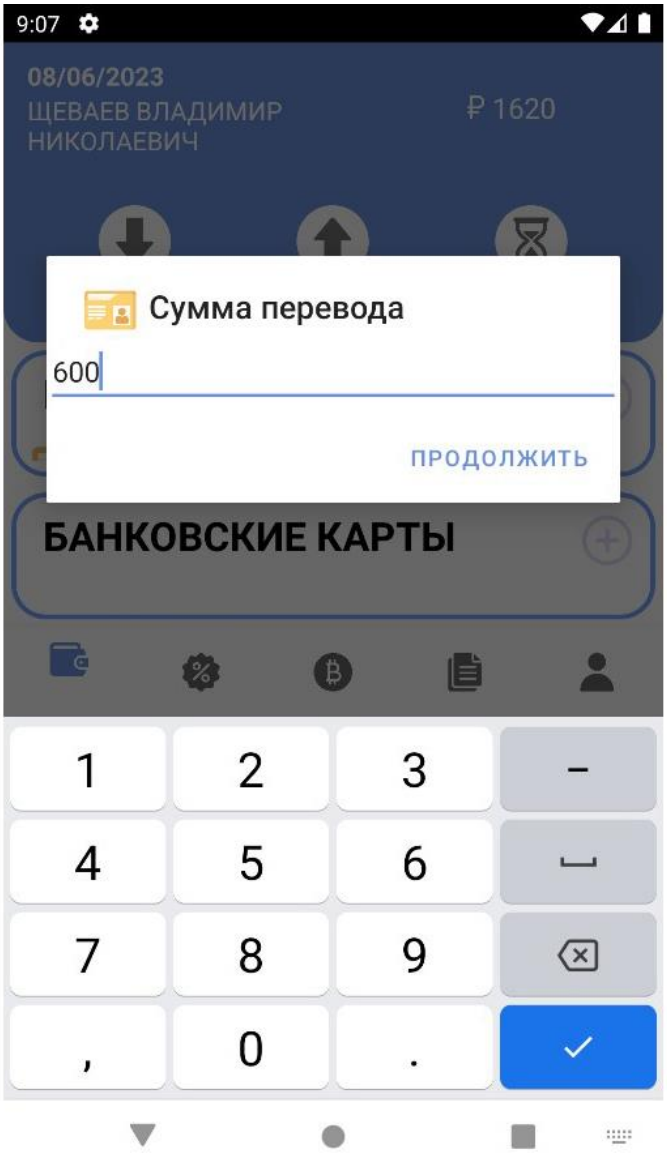


Рисунок 13 – Процесс ввода суммы перевода

На главном экране спроектируем списание денежных средств после процедуры перевода. Для этого добавим текст суммы за вычетом отправленных активов. Изменим баланс банковского счета, участвовавшего в процедуре перевода.

Важно помнить, что все процессы передачи данных аккаунта пользователя должны проходить обработку серверной частью и выводиться на

стороне клиента. Одним из инструментов отображения подобной нформации является окно истории операций. В этом окне должно выводиться название операции и дата ее проведения. В приложении необходимо реализовать возможность пролистывания списка истории до самой первой операции.

Окно истории изображено на рисунке 14. Пользователь может просмотреть список, интересующих его операций.

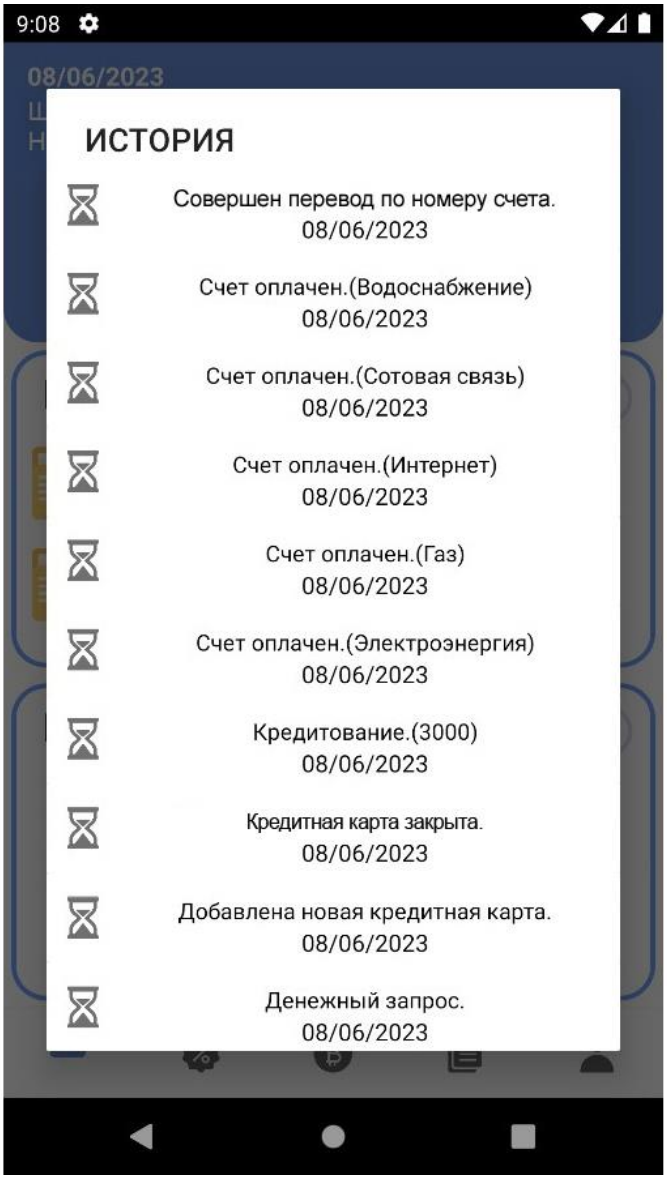


Рисунок 14 – История операций

В среде растрового редактора смоделируем модель поведения работы клиентской и серверной части. От этой цепочки событий ожидается результат ввиде зачисления денежных средств на банковский счет, номер которого был

использован при переводе. Программная реализация этого графического представления должна будет заключаться в синхронизации результат операции между двумя аккаунтами. Для наглядной демонстрации успеха операции, создадим профиль получателя. Отобразим общую сумму со всех счетов данного пользовтаеля и изобразим внешний вид окна банковских счетов. Изображение конечного этапа процесса перевода денежных средств представлено на рисунке 15

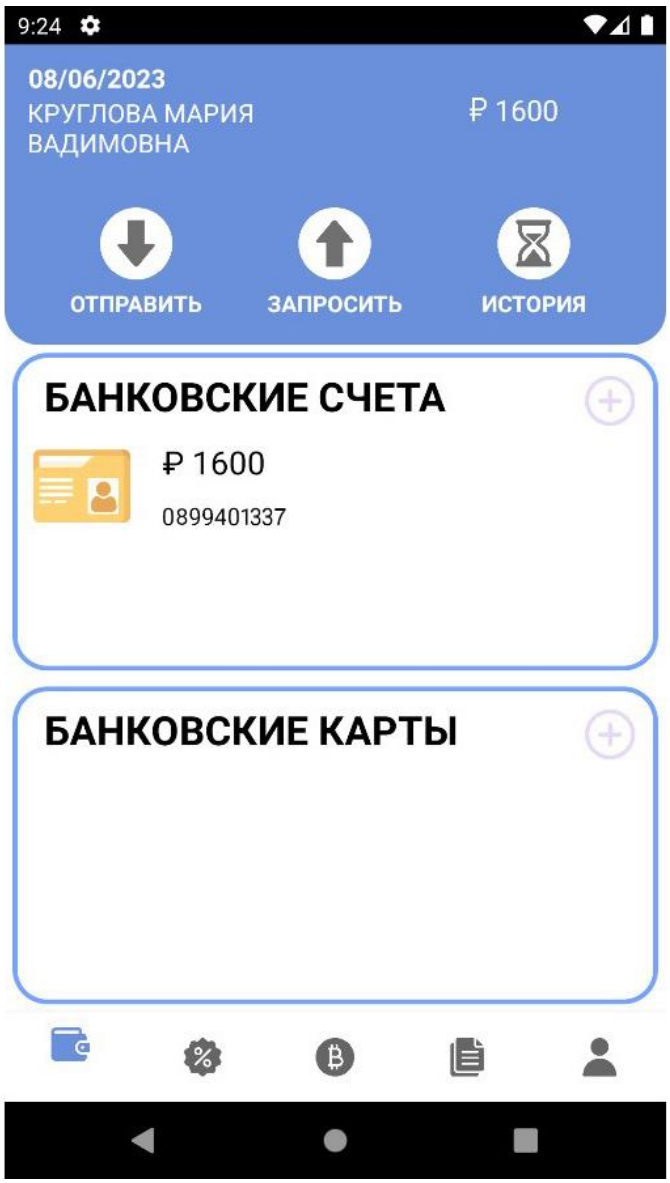


Рисунок 15 – Главный экран получателя денежного перевода

Одной из услуг разрабатываемого приложения является кредитование. Для реализации данного бизнес-процесса требуется спроектировать

отдельный экран с удобной и понятной навигацией. Кнопка в правом нижнем углу данного раздела должна будет выводить окно оформления кредита. Важно отобразить максимальные значения суммы займа и срока выплаты, оформить поля ввода необходимой суммы и количества месяцев, отведенных на погашение кредита. Подтверждение оформления обозначить кнопкой. Экран кредитования изображен на рисунке 16.

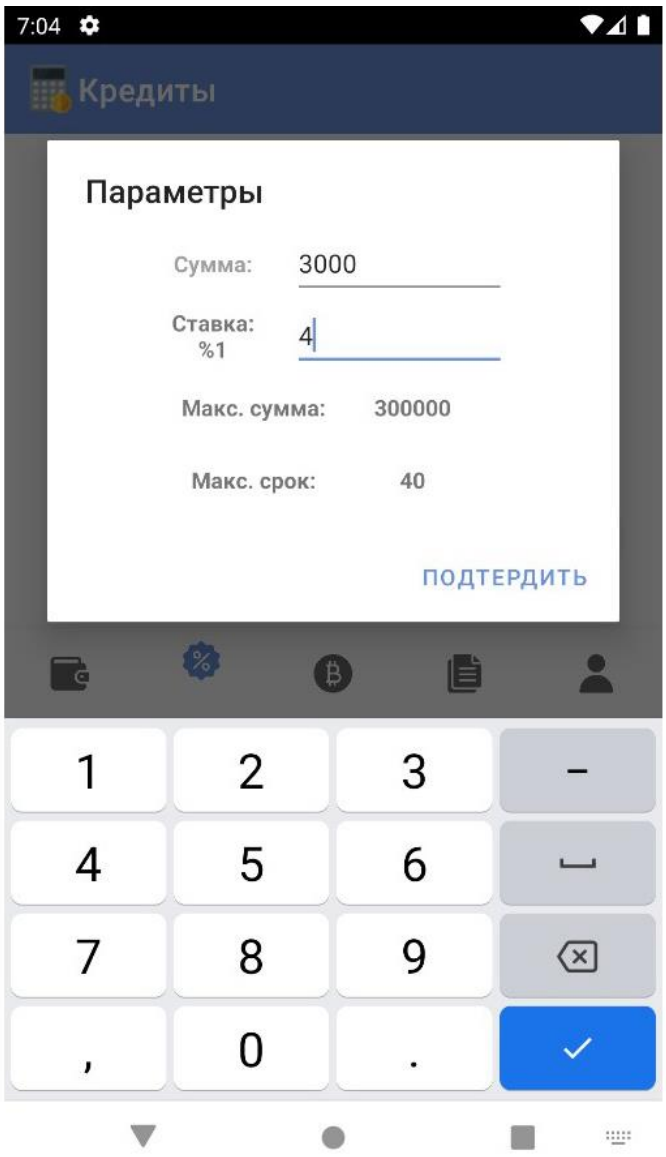


Рисунок 16 – Экран оказания услуг кредитования

Ключевой особенностью услуги кредитования является формирование общей суммы кредита, исходя из рода деятельности клиента. Во время регистрации пользователь может выбрать вид текущей занятости, что

определит величину процентой ставки. Если клиент сменит свой вид деятельности, то изменить данный пунк он сможет в личном кабинете. Вслед за этим значение процентной ставки тоже измениться. Данная функция удобна тем, что пользователь заранее может расчитать свои возможности относительно заработка.

На рисунке 17 изображено окно кредитования с пронумерованными элементами графического интерфейса.

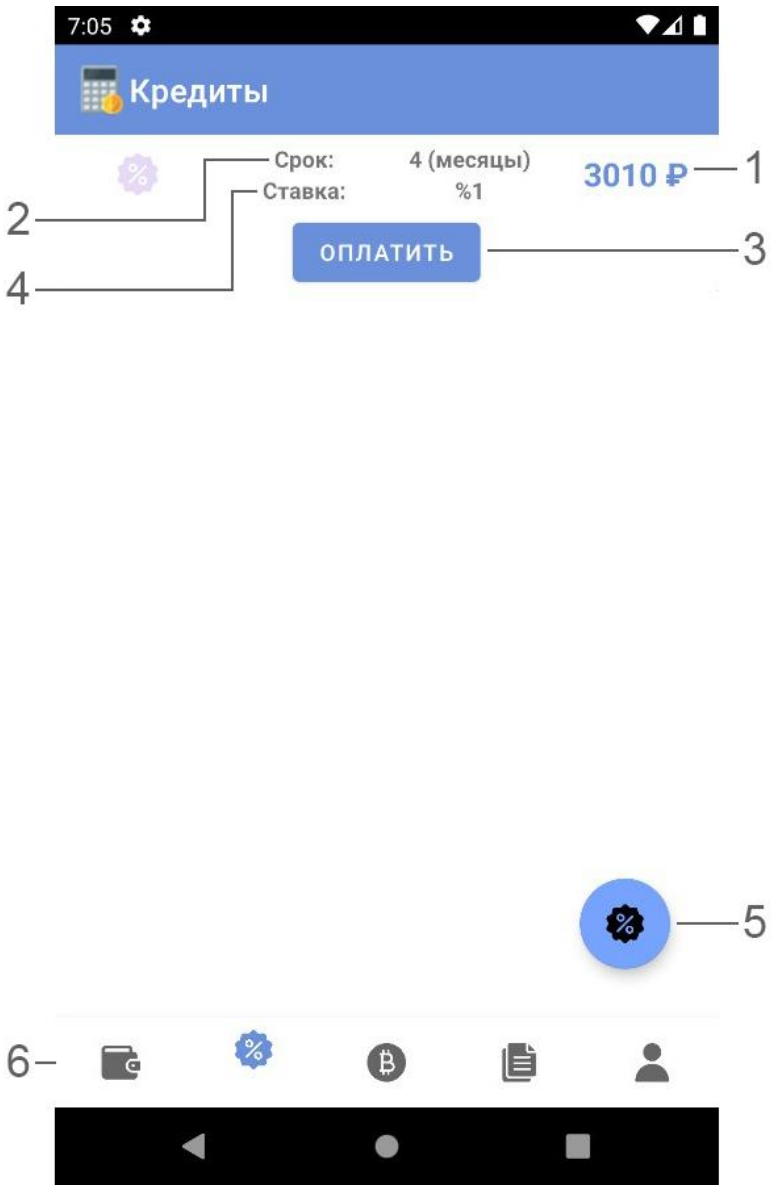


Рисунок 17 – Экран кредитования

Основные элементы экрана кредитования:

1 – общая сумма выплаты;

2 – скрок погашения кредита;

3 – кнопка оплаты кредита;

4 – значение процентной ставки;

5 – кнопка, отвечающая за оформление кредита;

6 – панель навигации по основным экранам приложения;

За покупку и хранение виртуальных активов отвечает функционал экрана криптовалюты. Рисунок 18 демонстрирует оформление этого раздела программы.

Рисунок 18 – Экран криптовалюты

Для современного клиента оплата жилищно-коммунальных услуг стала естественным процессом. Современные технологии упростили это занятие, высвободив большой объем времени. Перед реализацией данного функционала создадим графическое представление окна оплаты счетов. Спроектируем окно выбора необходимой услуги с указание суммы оплаты, добавим кнопку для открытия списка услуг. Создадим шаблон вывода на экран оплаченных и задокументированных услуг.

Раздел оплаты и документирования коммунальных услуг представлен на рисунке 19.

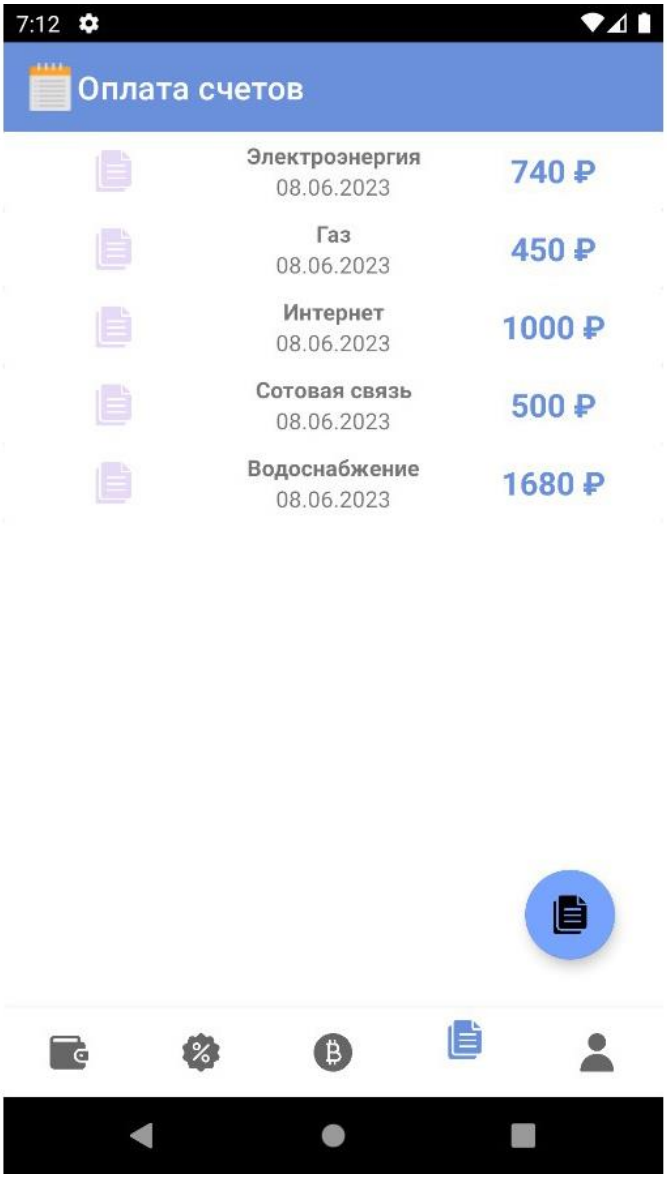


Рисунок 19 – Экран оплаты счетов

Важным элементом любого финансового приложения является личный кабинет пользователься. В нем он всегда может проверить и изменить интересующую его информацию. Данный элемент системы тесно связан с хранилищем данных.

Планируется вывод информации из файлов типа JSON в соответствующие поля на экране. Редактирование будет реализовано в отдельных окнах. После изменения или отмены изменеия необходимо организовать вывод информационного сообщения о результате действия. Предполагается отображение уникального идентификатора клиента на экран устройства. На рисунке 20 представлен доступный функционал редактирования пунктов личной информации.

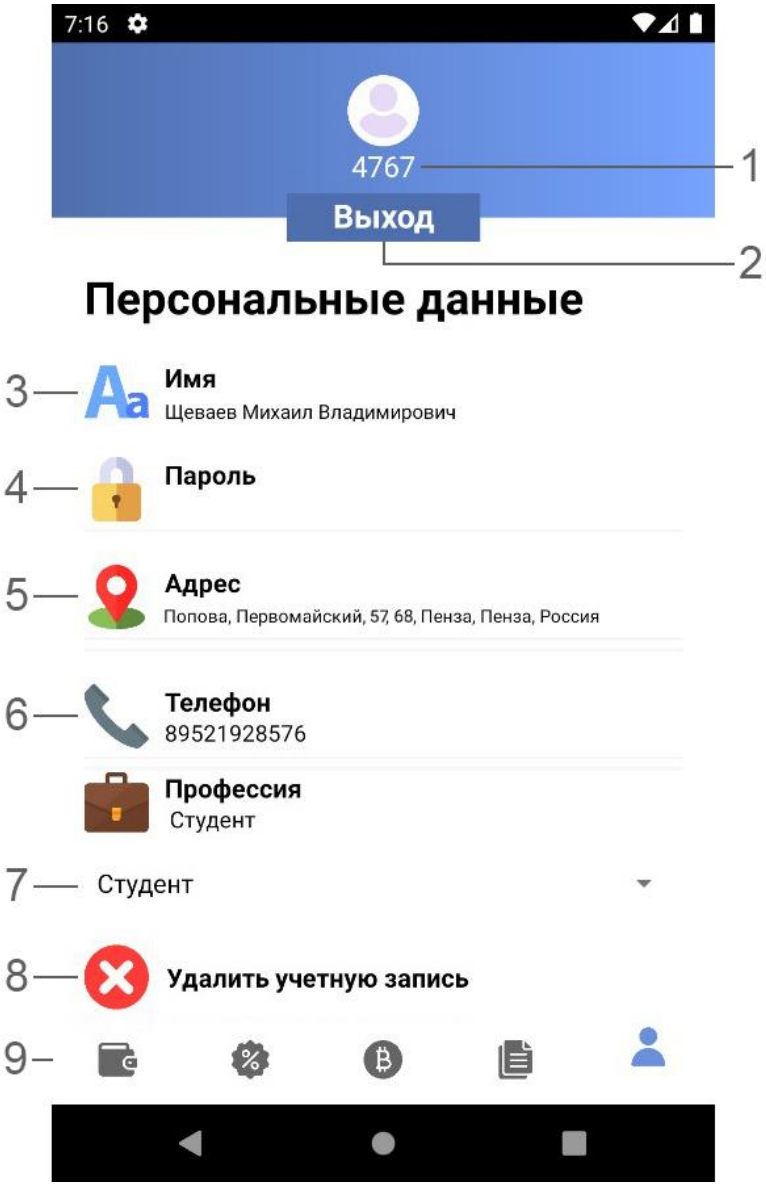


Рисунок 20 – Личный кабинет пользователя

Основные элементы экрана личного кабинета пользователя:

1 – поле редактирования уникального идентификатора учетной записи;

2 – кнопка выхода из аккаунта;

3 – поле редактирования полного имени пользователя;

4 – поле, которое служит для изменения пароля;

5 – поле редактирования адреса проживания;

6 – поле для редактирования номера телефона;

7 – поле выпадающего списка, которое позволяет сменить вид деятельности;

8 – кнопка, отвечающая за удаление учетной записи из базы данных;

9 - панель навигации по основным экранам приложения.

Проектирование графического интерфейса на ранних этаках разработки позволяет продумать цепочки процессов бизнес-логики, используя наглядные примеры. Пользуясь инструментами интегрированной среды разаработки, можно быстро составить структуру экранов приложения.

Исходя из вышеописанных моделей проектирования пользовательского интерфейса, можно понять, что графическое представление проекта реализует все функциональные требования.

3 Реализация мобильного приложения

3.1 Реализация серверной части

Серверная часть приложения была выполнена в интегрированной среде разработки Android Studio с использованием языка программирования Java. В проекте за работу с данными отвечают подключаемые инструменты разработки платформы Parse android.

Настройка Parse android начинается с создания собственного экземпляра Parse на хостинге платформы. Для работы с ней необходимо добавить несколько сетевых разрешений в файл манифеста. Инициализация проекта разрабатываемой программы в системе хостинга представлена на рисунке 21.

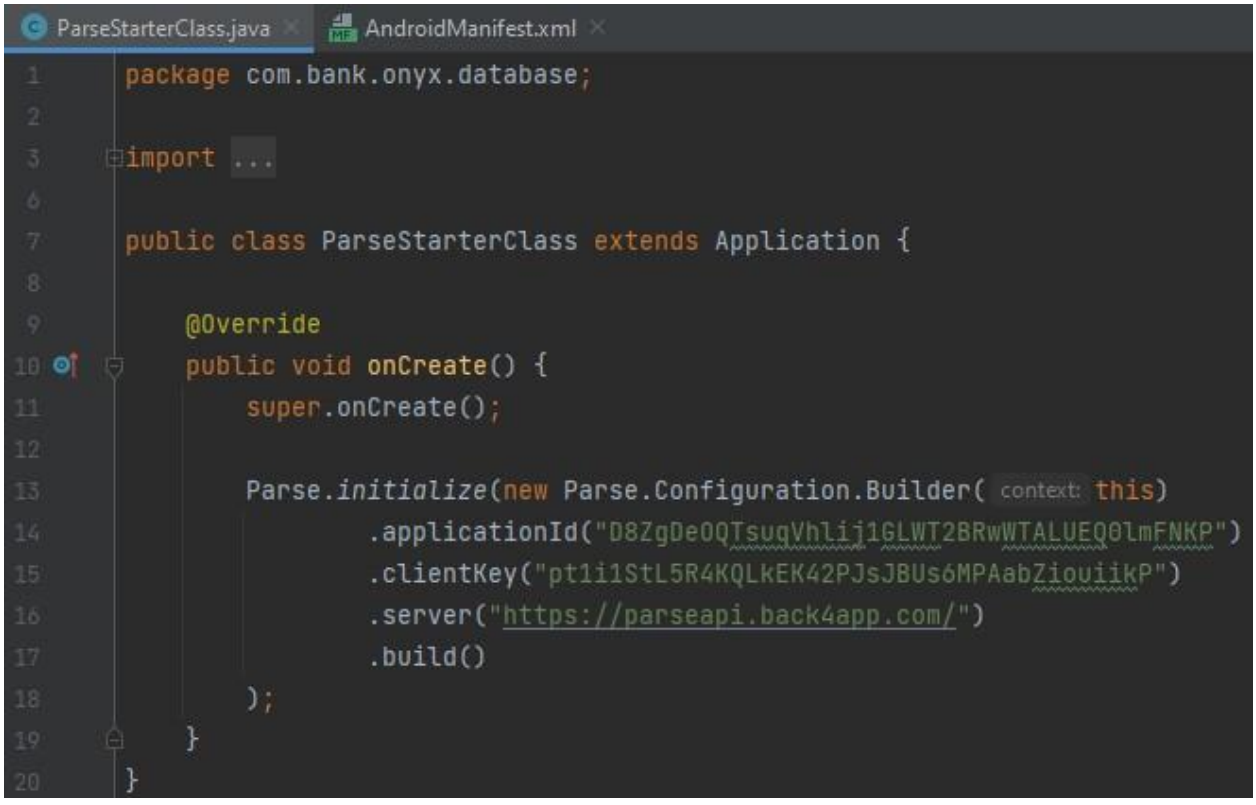


Рисунок 21 – Параметры подключения

Для взаимодействия с учетными записями пользователей у Parse android есть собственный класс Parseuser. Добавить нового пользователя в базу можно, создав объект ParseUser и вызвав метод signUpInBackground. За вход пользователя в систему и передачу его данных отвечает метод

logInBackground. Встроенные элементы Parse android позволяют в любое время получить доступ к копии результатов запроса клиентской части, что упрощает отладку процессов регистрации и входа.

Хранение данных в Parse android построено вокруг сущности ParseObject. Каждый ParseObject содержит пары данных типа «ключ-значение», совместных с форматом текстовых данных JSON. У каждой сущности ParseObject есть собственное имя класса, которое используется для маркировки типов данных. За сохранение моделей данных в базе отвечает метод saveInBackground.

Общий шаблон работы с запросами заключается в использовании экземпляров класса ParseQuery и наложении на них условий для извлечения соответствующих объектов с помощью метода findInBackground. Используя запрос getQuery, можно получить данные объекта целиком, если известен его идентификатор [6].

Благодаря использованию данной технологии удалось автоматизировать работу с данными, что значительно упростило разработку приложения и снизило нагрузку на клиентскую часть.

3.2 Реализация клиентской части

Исходя из функциональных требованй, в клиентской части были реализованны основыне процессы фионансовых операций, процессы взаимодействия с криптовалютой и редактирование личного кабинета пользователя.

За логику работы клиента отвечает большое количество классов, наиболее значимым из них является класс AccountFragment. Именно он выполняет обработку входных данных и вывод на экран активностей приложения. Методы данного класса структурированны согласно низкоуровневым шаблонам платформы Parse android. Языковые коностркции оформлены таким образом, чтобы обращаться к сущностям ParseObject за необходимой информацие. Обработка файлов типа JSON настроена на работу

с данными типа «ключ-значение». Такой подход позволяет быстро выводить на экран получаемую информацию. На рисунке 22 предсталена одна из языковых структур, использованных в написании кода программы.

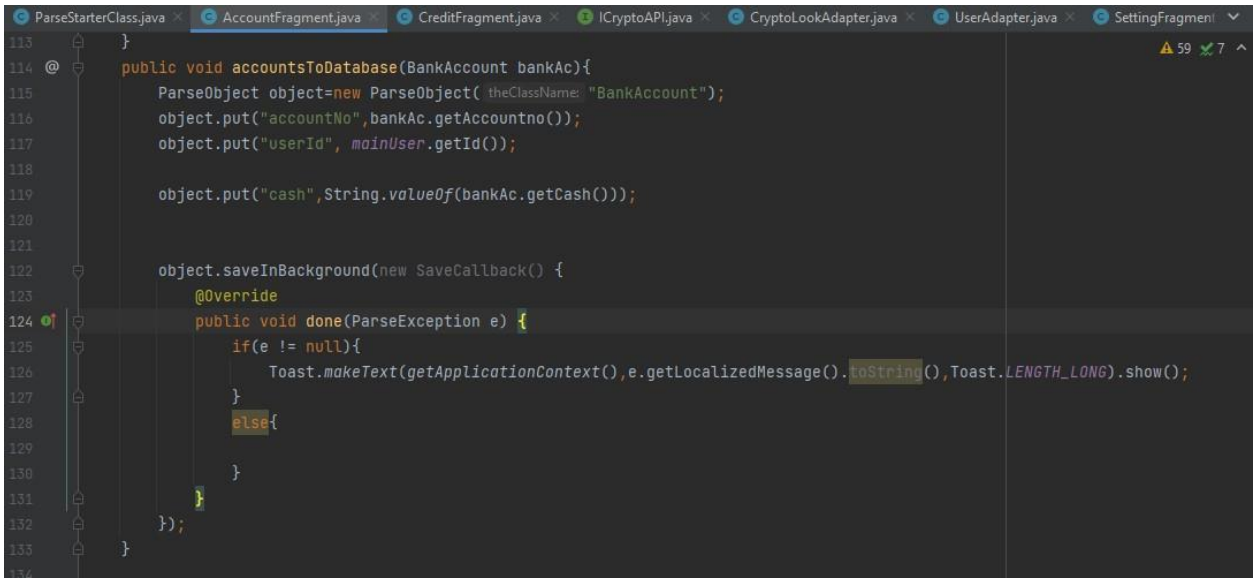


Рисунок 22 – Пример языковой структуры клиентской части.

Благодаря своевременному проектированию графического интерфейса, удалось снизить трудозатраты на разработку, что значительно упростило процесс работы над клиентской частью. Использование низкоуровневых шаблонов проектирования, основанных на технологиях обработки данных платформы Parse android, позволило реализовать весь заявленный функционал.

4 Контроль качества программного продукта

4.1 Тестирование приложения черным ящиком

Особенность тестирования черным ящиком заключается в том, что вместо свойств составных частей системы, изучается ее реакция на изменение условий эксплуатации. Под стратегией данного подхода к тестированию понимается создание тестов для тестовых наборов.

4.1.1 Классы эквивалентности

Под классами эквивалентности понимается набор входных данных, который обрабатывается программой по одному алгоритму или приводит к единственному результату. Для разрабатываемого приложения классы эквивалентности представлены в таблице 4.

Таблица 4 – Классы эквивалентности

	Верные классы эквивалентности	Неверные классы эквивалентности
Длина пароля	5-12 (1)	0-4 (2) 13 - ∞ (3)
Структура пароля	Начинается со специального символа (4)	Начинается не со специального символа (5)
Содержание пароля	Цифры (6) Латинские буквы (7) Символы (8)	Остальные символы (9)

Данные в таблице 4 необходимы для корректной работы метода проверки пароля от учетной записи.

4.1.2 Граничные условия

Граничное тестирование использует принцип, при котором один класс эквивалентности переходит в другой. Именно на границе перехода чаще всего возникают ошибки. К граничным значениям относится крайнее значение, начальное-конечное, нижнее-верхнее и максимальное-минимальное. Тип и описание граничные значения для разрабатываемого продукта представлены в таблице 5.

Таблица 5 – Граничные значения

Границы	Описание
Физические	Поле не может быть пустым
Технологические	Ограничения оперативной памяти устройства
Произвольные	Пароль не более 12 символов

Эти граничные условия помогут избежать сбоев системы на переходах функционала одного класса к другому.

4.1.3 Тест-кейсы

По созданным классам эквивалентности и граничным условиям был создан набор тест-кейсов. Такой подход к тестированию представляет собой алгоритм действий по которому проводится тестирование уже написанной программы. Тест-кейсы представлены в таблице 6.

Таблица 6 – Набор тест-кейсов

№	Класс эквивалентности	Входные данные	Выходные данные
1	Все правильные(1,4,5,8,9, 10)	4767	Успешно
2	(2,4,5,8,10)	46Sq67	Недопустимый идентификатор
3	(3,4,5,8,9,10)	Wrt22@34opt9wq12Xcb	Больше 12 символов
4	(1,5,6,8,9,10)	@23Wp40_1	Ошибка, начинается со специального символа

Написанные тест-кейсы помогут выявить недостатки работы системы и предотвратят сбои в ее работе.

4.2 Функциональное тестирование белым ящиком

Тестирование программного обеспечения белым ящиком – это метод тестирования, который предполагает, что структура кода и особенности системы заранее известны тестировщику. При создании тест-кейсов важно избегать любых зависимостей, потому что один из них может быть удален из-за ненадобности, тогда сломается логика проверки зависимых от него тест-кейсов [7]. Результаты функционального тестирования регистрации и авторизации приведены в таблице 7.

Таблица 7 – Функциональное тестирование

Вариант использования	Тест и его входные параметры	Результат
Зарегистрироваться в системе	Ввод верных данных: Имя пользователя: Щеваев Михаил Владимирович; Идентификатор: 4767; Пароль: As@45Tm80; Номер телефона: 89521928576; Адрес: Попова, Первомайский, 57 68, Пенза, Россия.	Тест пройден (рисунок Г.1)
	Ввод занятого идентификатора: Имя пользователя: Щеваев Михаил Владимирович; Идентификатор: 4790; Пароль: As@45Tm80; Номер телефона: 89521928576; Адрес: Попова, Первомайский, 57 68, Пенза, Россия.	Тест пройден (рисунок Г.2)
	Ввод неверного пароля: Имя пользователя: Щеваев Михаил Владимирович; Идентификатор: 4767; Пароль: As@45Tm80; Номер телефона: 89521928576; Адрес: Попова, Первомайский, 57 68, Пенза, Россия.	Тест пройден (рисунок Г.3)
	Ввод некорректного номера телефона: Имя пользователя: Щеваев Михаил Владимирович; Идентификатор: 4767; Пароль: As@45Tm80; Номер телефона: 89521928576; Адрес: Попова, Первомайский, 57 68, Пенза, Россия.	Тест пройден (рисунок Г.4)
	Ввод некорректного адреса проживания: Имя пользователя: Щеваев Михаил Владимирович; Идентификатор: 4767; Пароль: As@45Tm80; Номер телефона: 89521928576; Адрес: Попова, Первомайский, 57 68, Пенза, Россия.	Тест пройден (рисунок Г.5)
Авторизоваться в системе	Ввод верных данных: Идентификатор: 4767; Пароль: As@45Tm80.	Тест пройден (рисунок Г.6)
	Ввод неверного идентификатора: Идентификатор: 4767; Пароль: As@45Tm80.	Тест пройден (рисунок Г.7)
	Ввод неверного пароля: Идентификатор: 4767; Пароль: As@45Tm80.	Тест пройден (рисунокГ.8)

Функциональное тестирование показало корректность работы клиентской части мобильного приложения. Проверку прошел функционал регистрации и авторизации пользователя в системе.

4.3 Планирование и расчет бюджета разрабатываемого приложения

Для планирования и расчета бюджета разрабатываемого приложения была использована программа Microsoft Project. Данный программный продукт обеспечивает разработку планов, анализ объемов работ, отслеживание прогресса деятельности [8]. Реализация проекта была начата 1 марта 2023 года.

Во время планирования разработки было выявлено 30 задач. Наименования и сроки их выполнения представлены на рисунке 23.

	Название задачи	Длительнс	Начало	Окончани	Предшественн
1	Начало	0 дней	Ср 3/1/23	Ср 3/1/23	
2	**Анализ требований**	**9 дней**	**Ср 3/1/23**	**Пн 3/13/23**	
3	Изучение предметной области	5 дней	Ср 3/1/23	Вт 3/7/23	1
4	Анализ продуктов-аналогов	4 дней	Ср 3/8/23	Пн 3/13/23	3
5	Анализ требований завершен	0 дней	Пн 3/13/23	Пн 3/13/23	4
6	**Выбор средств реализации проекта**	**5 дней**	**Ср 3/8/23**	**Вт 3/14/23**	
7	Анализ возможных программных средств	4 дней	Ср 3/8/23	Пн 3/13/23	3
8	Выбор средств реализации проекта	1 день	Вт 3/14/23	Вт 3/14/23	7
9	Выбор средств завершен	0 дней	Вт 3/14/23	Вт 3/14/23	8
10	**Реализация проекта**	**13 дней**	**Ср 3/15/23**	**Пт 3/31/23**	
11	Разработка структуры данных	3 дней	Ср 3/15/23	Пт 3/17/23	5,9
12	**Разработка серверной части приложения**	**10 дней**	**Сб 3/18/23**	**Чт 3/30/23**	
13	Программирование моделей данных	3 дней	Сб 3/18/23	Вт 3/21/23	11
14	Разработка API	3 дней	Ср 3/22/23	Пт 3/24/23	13
15	Реализация API	4 дней	Пн 3/27/23	Чт 3/30/23	14
16	**Разработка пользовательского интерфейса**	**10 дней**	**Сб 3/18/23**	**Чт 3/30/23**	
17	Разработка графических элементов	3 дней	Сб 3/18/23	Вт 3/21/23	11
18	Разработка пользовательского интерфейса	3 дней	Ср 3/22/23	Пт 3/24/23	17
19	Настройка взаимодействия клиентской части с серверной частью	4 дней	Пн 3/27/23	Чт 3/30/23	18,14
20	Реализация проекта завершена	0 дней	Чт 3/30/23	Чт 3/30/23	15,19
21	**Контроль качества продукта**	**15 дней**	**Сб 4/1/23**	**Чт 4/20/23**	
22	Тестирование серверной части	5 дней	Пн 4/3/23	Пт 4/7/23	20
23	Тестирование клиентской части	5 дней	Пн 4/3/23	Пт 4/7/23	20
24	Изправление ошибок	5 дней	Пн 4/10/23	Пт 4/14/23	22,23
25	Тестирование завершено	0 дней	Пт 4/14/23	Пт 4/14/23	24
26	**Внедрение приложения на локальный сервер**	**8 дней**	**Пн 4/17/23**	**Ср 4/26/23**	
27	Развертывание приложения на сервре	5 дней	Пн 4/17/23	Пт 4/21/23	25
28	Поиск и испраление ошибок	3 дней	Пн 4/24/23	Ср 4/26/23	27
29	Внедрение завершено	0 дней	Ср 4/26/23	Ср 4/26/23	28
30	Завершение проекта	0 дней	Ср 4/26/23	Ср 4/26/23	29

Рисунок 23 – Список задач проекта

В таблице 8 представлены типы основных ресурсов проекта и их стоимость.

Таблица 8 – Список ресурсов

Наименование ресурса	Тип	Стандартная ставка	Ставка сверхурочных
Разработчик	Трудовой	675,00 ₽/ч	1350,00 ₽/ч
Дизайнер	Трудовой	575,00 ₽/ч	1150,00 ₽/ч

На рисунке 24 изображена диаграмма Ганта. Этот инструмент представляет собой горизонтальную гистограмму, которая отражает сроки задач и зависимости между ними.

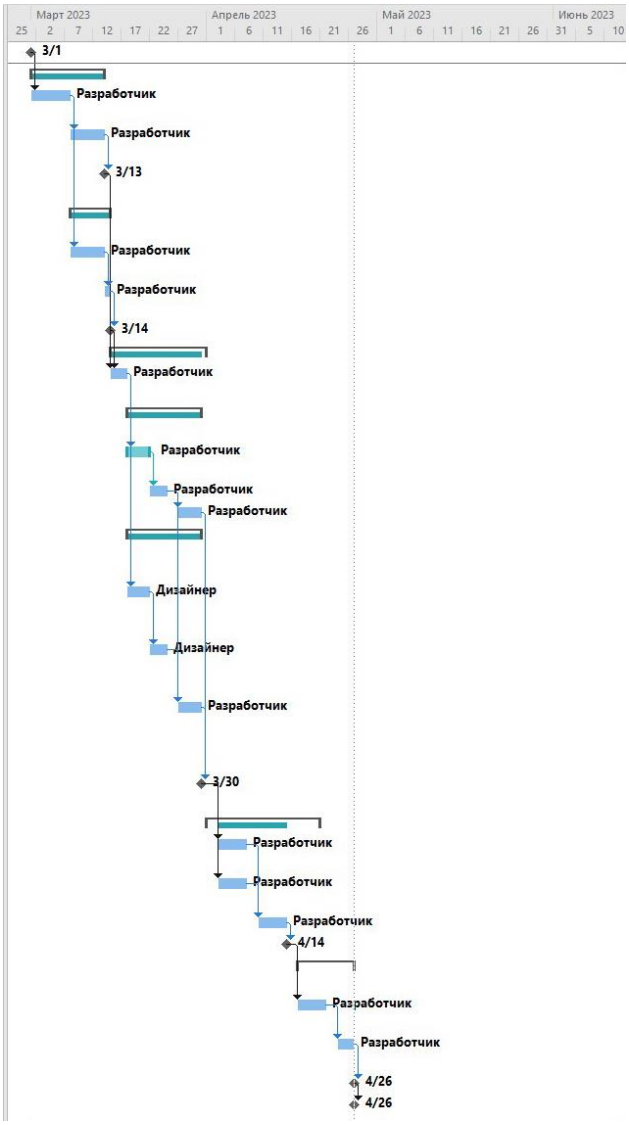
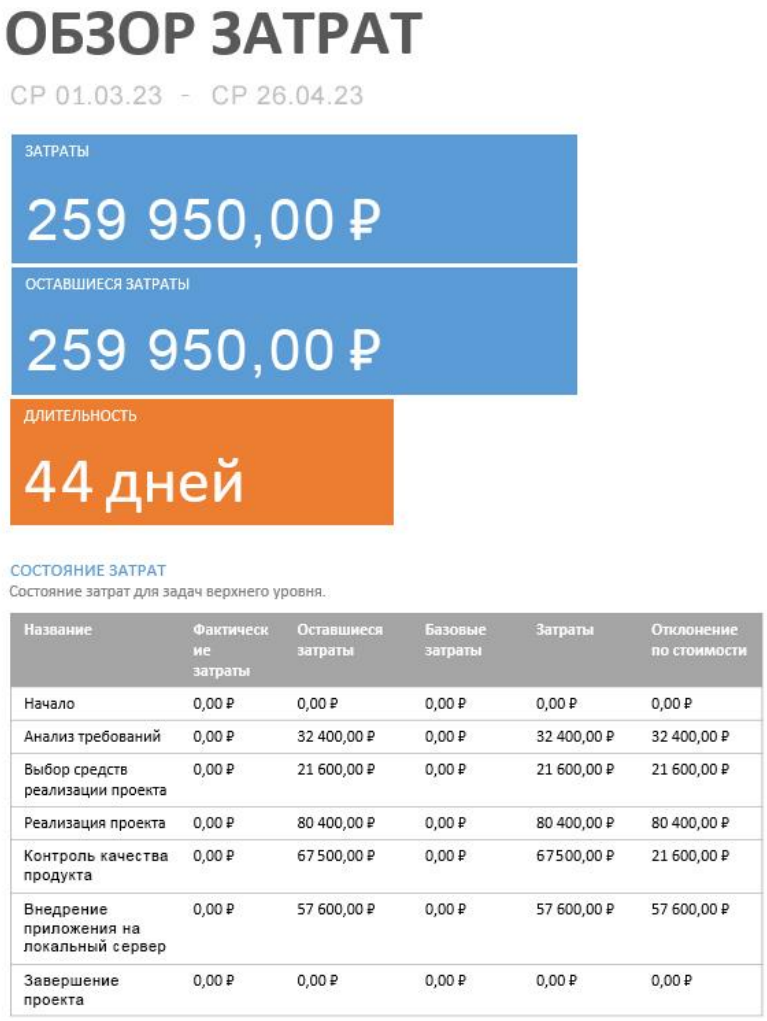


Рисунок 24 – Диаграмма Ганта

Обзорный отчет проекта представлен на рисунке 25.



ОБЗОР ЗАТРАТ

СР 01.03.23 - СР 26.04.23

ЗАТРАТЫ

259 950,00 ₽

ОСТАВШИЕСЯ ЗАТРАТЫ

259 950,00 ₽

ДЛИТЕЛЬНОСТЬ

44 дней

СОСТОЯНИЕ ЗАТРАТ

Состояние затрат для задач верхнего уровня.

Название	Фактические затраты	Оставшиеся затраты	Базовые затраты	Затраты	Отклонение по стоимости
Начало	0,00 ₽	0,00 ₽	0,00 ₽	0,00 ₽	0,00 ₽
Анализ требований	0,00 ₽	32 400,00 ₽	0,00 ₽	32 400,00 ₽	32 400,00 ₽
Выбор средств реализации проекта	0,00 ₽	21 600,00 ₽	0,00 ₽	21 600,00 ₽	21 600,00 ₽
Реализация проекта	0,00 ₽	80 400,00 ₽	0,00 ₽	80 400,00 ₽	80 400,00 ₽
Контроль качества продукта	0,00 ₽	67 500,00 ₽	0,00 ₽	67500,00 ₽	21 600,00 ₽
Внедрение приложения на локальный сервер	0,00 ₽	57 600,00 ₽	0,00 ₽	57 600,00 ₽	57 600,00 ₽
Завершение проекта	0,00 ₽	0,00 ₽	0,00 ₽	0,00 ₽	0,00 ₽

Рисунок 25 – Обзорный отчет проекта

Отчет, показанный на рисунке 25, дает понять, что предполагаемые расходы на разработку данного приложения составляют 259 950 рублей. Общий срок разработки при этом равен 44 дням.

Заключение

В результате выполнения выпускной квалификационной работы были выполнены основные этапы разработки универсального мобильного приложения банка.

На стадии анализа программных продуктов-аналогов были сформулированы функциональные и нефункциональные требования к разрабатываемому приложению.

На стадии проектирования сервиса была разработана его архитектура, организована структура базы данных, спроектирован пользовательский интерфейс.

Серверная часть приложения была выполнена в интегрированной среде разработки Android Studio с использованием языка программирования Java. За работу с данными отвечают подключаемые инструменты разработки платформы Parse android.

Было выполнено планирование и расчет бюджета для разрабатываемого продукта.

Исходя из вышесказанного, можно сделать вывод, что за время выполнения выпускной квалификационной работы были выполнены все поставленные задачи.

Список использованных источников

1. Распоряжение Правительства РФ от 29.12.2022 № 4355-р «Об утверждении Стратегии развития финансового рынка РФ до 2030 года». [Электронный ресурс]. URL: https://www.consultant.ru/document/cons_doc_LAW_436693/9138a38feb7c43b1e13ce17ab3ebf26f37646356/ (дата обращения: 18.04.2023).

2. Описание возможностей мобильного приложения СберБанк Онлайн. [Электронный ресурс]. URL: https://www.sberbank.com/ru/ecs (дата обращения: 19.04.2023).

3. Возможности интернет-банка Тинькофф [Электронный ресурс]. URL: https://finuslugi.ru/navigator/nakopit-i-sohranit/stat_vozmozhnosti_internet_banka_tinkoff (дата обращения 22.04.2023).

4. Подробный обзор Альфа-банка. [Электронный ресурс]. URL: https://bankiras.ru/alfa-bank/ (дата обращения: 22.04.2023).

5. Подробное руководство по созданию архитектуры мобильных приложений. [Электронный ресурс]. URL: https://scand.com/ru/company/blog/the-ultimate-guide-to-mobile-application-architecture/ (дата обращения: 25.04.2023).

6. Создание приложений на базе технологии Parse. [Электронный ресурс]. URL: https://github.com/codepath/android_guides/wiki/Building-Data-driven-Apps-with-Parse (дата обращения: 27.04.2023).

7. Канер С. Тестирование программного обеспечения. – Санкт-Петербург.: Питер, 2018. – ISBN 979-5-4462-0773-7. – 129 с.

8. Скороход С. Управление проектами средствами Microsoft Project. Курс лекций. – Москва.: Интуит НОУ, 2016. – 319 с.

Приложение А

ГЛОССАРИЙ

(справочное)

UML (Unified Modeling Language) – язык графического описания, использующийся для моделирования бизнес-процессов и системного проектирования.

JSON (JavaScript Object Notation) – текстовый формат обмена данными, использующий JavaScript. Формат JSON можно использовать в разработке, независимо от языка программирования JavaScript.

API (Application Programming Interface) – набор программных решений, решающий проблему разных по своей функциональности программ.

HTTP (HyperText Transfer Protocol) – протокол передачи данных, предназначенный для передачи гипертекстовых документов.

REST API – это архитектурный стиль организации компонентов распределенного приложения в сети. Является согласованным набором ограничений, учитываемых при проектировании.

Приложение Б

ЛИСТИНГ ПРИЛОЖЕНИЯ

(рекомендуемое)

Листинг Б.1 – Файл SignUpActivity.java

```
package com.bank.onyx.Sign;

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;

import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

import com.bank.onyx.Job.Contractor;
import com.bank.onyx.Job.Doctor;
import com.bank.onyx.Job.Driver;
import com.bank.onyx.Job.Engineer;
import com.bank.onyx.Job.Entrepreneur;
import com.bank.onyx.Job.Farmer;
import com.bank.onyx.Job.Job;
import com.bank.onyx.Job.Police;
import com.bank.onyx.Job.Soldier;
import com.bank.onyx.Job.Sportsman;
import com.bank.onyx.Job.Student;
import com.bank.onyx.Job.Teacher;
import com.bank.onyx.Job.Waiter;
import com.bank.onyx.Job.Worker;
import com.bank.onyx.MainScreen.MainScreenActivity;
import com.bank.onyx.R;
import com.bank.onyx.UserInfo.Address;
import com.bank.onyx.UserInfo.User;
import com.parse.ParseException;
import com.parse.ParseObject;
import com.parse.ParseUser;
import com.parse.SaveCallback;
import com.parse.SignUpCallback;

import static com.bank.onyx.Sign.SignIn.mainUser;

public class SignUpActivity extends AppCompatActivity {

    private EditText userNameText , userPassText ,userIdText,userPhoneText;

    private Spinner spinner;
    private ArrayAdapter<String> jobArrayAdapter;
    private String [] jobs;
    private Job [] defaultJobs;
    public Job tempJob ;
    public String job;
    private ImageView imageView;

    private Address newAddress;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_sign_up);
        userNameText=findViewById(R.id.edittext_id_name_sign_up);
```

```
        userPassText=findViewById(R.id.edittext_user_password_sign_up);
        userIdText=findViewById(R.id.edittext_id_number_sign_up);
        userPhoneText=findViewById(R.id.edittext_phone_sign_up);
        spinner = findViewById(R.id.jobSpinner);
        defineJobSpinner();
    }
    public void defineJobSpinner(){
        defaultJobs = new Job[]{new Contractor(),new Doctor(),new
Driver(),new Engineer(),new Entrepreneur(),
        new Farmer(),new Police(),new Soldier(),new Sportsman(),new
Student(),new Teacher(),new Waiter(),new Worker()};
        jobs = new String[]
{"Подрядчик","Врач","Водитель","Инженер","Предприниматель","Фермер","Полицейс
кий","Военный",
                "Спортсмен","Студент","Учитель","Официант","Строитель"};
        jobArrayAdapter = new
ArrayAdapter(this,R.layout.support_simple_spinner_dropdown_item,jobs);
        spinner.setAdapter(jobArrayAdapter);
        spinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> adapterView, View view,
int i, long l) {
                job = adapterView.getSelectedItem().toString();
            }
            @Override
            public void onNothingSelected(AdapterView<?> adapterView) {
            }
        });
    }
    public void signUp(View view){
        mainUser=new User(userNameText.getText().toString(),
                userIdText.getText().toString(),
                userPassText.getText().toString(),
                userPhoneText.getText().toString(),newAddress
                );
        ParseUser parseUser=new ParseUser();
        parseUser.setUsername(mainUser.getId());
        parseUser.setPassword(mainUser.getPass());
        ParseObject object=new ParseObject("UserInfo");
        object.put("userRealName",mainUser.getName());
        object.put("phone",mainUser.getPhoneNumber());
        object.put("address",mainUser.addressWrite());
        object.put("username", mainUser.getId());
        for(Job x:defaultJobs){
            if(x.getName().equals(job)){
                tempJob=x;
                break;
            }
        }
        mainUser.setJob(tempJob);
        object.put("job",tempJob.getName());
        object.put("maxCreditAmount",tempJob.getMaxCreditAmount());
        object.put("maxCreditInstallment",tempJob.getMaxCreditInstallment());
        object.put("interestRate",tempJob.getInterestRate());
        object.put("address",mainUser.addressWrite());
        object.saveInBackground(new SaveCallback() {
            @Override
            public void done(ParseException e) {
                if(e != null){

Toast.makeText(getApplicationContext(),e.getLocalizedMessage().toString(),Toa
st.LENGTH_LONG).show();
                }
```

```
            }
        });
       parseUser.signUpInBackground(new SignUpCallback() {
            @Override
            public void done(ParseException e) {
                if(e != null){

Toast.makeText(getApplicationContext(),e.getLocalizedMessage(),Toast.LENGTH_L
ONG).show();
                }
                else{
                    Toast.makeText(getApplicationContext(),"Новый
пользователь" ,Toast.LENGTH_LONG).show();
                    Intent intent=new Intent(getApplicationContext(),
MainScreenActivity.class);
                    startActivity(intent);
                }
            }
       });
    }
    public void createAddress(View v){
        AlertDialog.Builder ad = new AlertDialog.Builder(this);
        ad.setTitle("Адрес");
        ad.setIcon(R.drawable.ic_address);
        LayoutInflater inflater = getLayoutInflater();
        View dialogView= inflater.inflate(R.layout.settings_address_popup,
null);
        ad.setView(dialogView);
        ad.setPositiveButton("ПРОДОЛЖИТЬ", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                TextView
street=dialogView.findViewById(R.id.setting_address_street);
                TextView
floor=dialogView.findViewById(R.id.setting_address_floor);
                TextView
houseNo=dialogView.findViewById(R.id.setting_address_house);
                TextView
country=dialogView.findViewById(R.id.setting_address_country);
                TextView district
=dialogView.findViewById(R.id.setting_address_district);
                TextView
town=dialogView.findViewById(R.id.setting_address_town);
                TextView locality
=dialogView.findViewById(R.id.setting_address_locality);
                if(street.getText().toString() !=null
&&district.getText().toString()!=null && floor.getText().toString()!=null &&
houseNo.getText().toString()!=null && town.getText().toString()!=null &&
locality.getText().toString()!=null && country.getText().toString()!=null){
                    newAddress= new
Address(street.getText().toString(),district.getText().toString(),Integer.par
seInt(floor.getText().toString()),
town.getText().toString(),locality.getText().toString(),country.getText().toS
tring());
                }else{
                    Toast.makeText(getApplicationContext(),"Заполните все
поля",Toast.LENGTH_SHORT).show();
                }
            }
        });
        ad.setNegativeButton("ОТМЕНИТЬ", new
DialogInterface.OnClickListener() {
            @Override
```

```
            public void onClick(DialogInterface dialog, int which) {

Toast.makeText(getApplicationContext(),"Отменено",Toast.LENGTH_SHORT).show();
            }
        });
        ad.create().show();
    }
}
```

Листинг Б.2 – Файл SignIn.java

```
package com.bank.onyx.Sign;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

import com.bank.onyx.Bill.Bill;
import com.bank.onyx.Bill.Date;
import com.bank.onyx.Credit.Credit;
import com.bank.onyx.Job.Job;
import com.bank.onyx.R;
import com.bank.onyx.UserInfo.Address;
import com.bank.onyx.UserInfo.BankAccount;
import com.bank.onyx.UserInfo.CreditCard;
import com.bank.onyx.UserInfo.History;
import com.bank.onyx.UserInfo.User;
import com.bank.onyx.splashScreen.splashScreen;
import com.parse.FindCallback;
import com.parse.GetDataCallback;
import com.parse.LogInCallback;
import com.parse.ParseException;
import com.parse.ParseFile;
import com.parse.ParseObject;
import com.parse.ParseQuery;
import com.parse.ParseUser;

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

public class SignIn extends AppCompatActivity {
    private EditText userName,userPass;
    public static User mainUser;
    public static ArrayList<User> allUsers;
    private String billType;
    private String billAmount;
    private String billDate;
    private ArrayList<Bill> bills;
    private ArrayList<BankAccount> bankAccounts;
    private Stack<History> history;
    private ArrayList<CreditCard> creditCards;
    private String bankCash,bankAccountNo;
    private String cardNo, cardLimit;
    private Intent intent ;
    private ArrayList<Credit> credits;
    private String creditAmount;
```

```
    private String creditInstallment;
    private String creditInterestRate;
    private String creditPayAmount;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
           setContentView(R.layout.activity_sign_in);//load screen
           userName=findViewById(R.id.edittext_id_number_sign_in);
           userPass=findViewById(R.id.edittext_user_password_sign_in);
    }
    public void signUp(View view){
        Intent signUp=new Intent(SignIn.this, SignUpActivity.class);
        startActivity(signUp);
    }
    public void getUserBills(){
        ParseQuery<ParseObject> queryBill=ParseQuery.getQuery("Bill");
        queryBill.whereEqualTo("username",SignIn.mainUser.getId());
        queryBill.findInBackground(new FindCallback<ParseObject>() {
            @Override
            public void done(List<ParseObject> objects, ParseException e) {
                if(e!=null){
                    e.printStackTrace();
                }else{
                    bills = new ArrayList<>();
                    if(objects.size()>0){
                        for(ParseObject object:objects){
                            billType=object.getString("type");
                            billAmount=object.getString("amount");
                            billDate=object.getString("date");
                            String [] date = billDate.split("/");
                            Date tempdate = new
Date(date[0],date[1],date[2]);
                            Bill tempBill = new
Bill(billType,Integer.parseInt(billAmount),tempdate);
                            bills.add(tempBill);
                        }
                    }
                    SignIn.mainUser.setUserbills(bills);
                }
            }
        });
    }
    public void getUserCredits(){
        ParseQuery<ParseObject> queryBill=ParseQuery.getQuery("Credit");
        queryBill.whereEqualTo("username",SignIn.mainUser.getId());
        queryBill.findInBackground(new FindCallback<ParseObject>() {
            @Override
            public void done(List<ParseObject> objects, ParseException e) {
                if(e!=null){
                    e.printStackTrace();
                }else{
                    credits = new ArrayList<>();
                    if(objects.size()>0){
                        for(ParseObject object:objects){
                            creditAmount = object.getString("amount");
                            creditInstallment =
object.getString("installment");
                            creditInterestRate =
object.getString("interestRate");
                            creditPayAmount = object.getString("payAmount");
                            Credit tempCredit = new
Credit(Integer.parseInt(creditAmount),Integer.parseInt(creditInstallment),
```

```
Integer.parseInt(creditInterestRate),Integer.parseInt(creditPayAmount));
                            credits.add(tempCredit);
                        }
                    }
                    SignIn.mainUser.setCredits(credits);
                }
            }
        });
    }

    public void getCreditCards(User user){
        ParseQuery<ParseObject>
queryBankAccount=ParseQuery.getQuery("CreditCard");
        queryBankAccount.whereEqualTo("userId", user.getId());
        queryBankAccount.findInBackground(new FindCallback<ParseObject>() {
            @Override
            public void done(List<ParseObject> objects, ParseException e) {
                if(e!=null){
                    e.printStackTrace();
                }else{
                    creditCards = new ArrayList<>();
                    if(objects.size()>0){
                        for(ParseObject object:objects){
                            cardNo=object.getString("creditCardNo");
                            cardLimit=object.getString("limit");
                            creditCards.add(new
CreditCard(cardNo,Integer.parseInt(cardLimit)));
                        }
                    }
                    user.setCreditcards(creditCards);
                }
            }
        });
    }
    public void getBankAccounts(User user){
        ParseQuery<ParseObject>
queryBankAccount=ParseQuery.getQuery("BankAccount");
        queryBankAccount.whereEqualTo("userId", user.getId());
        queryBankAccount.findInBackground(new FindCallback<ParseObject>() {
            @Override
            public void done(List<ParseObject> objects, ParseException e) {
                if(e!=null){
                    e.printStackTrace();
                }else{
                    bankAccounts = new ArrayList<>();
                    if(objects.size()>0){
                        for(ParseObject object:objects){

                            bankAccountNo=object.getString("accountNo");
                            bankCash=object.getString("cash");
                            bankAccounts.add(new
BankAccount(bankAccountNo,Integer.parseInt(bankCash)));
                        }
                    }
                    user.setBankAccounts(bankAccounts);
                }
            }
        });
    }
    public void getHistory(){
        ParseQuery<ParseObject>
queryBankAccount=ParseQuery.getQuery("History");
        if (!mainUser.getId().equals("9999")){
```

```
            queryBankAccount.whereEqualTo("userId", SignIn.mainUser.getId());
        }
        else{
            getAllUsers();
        }
        queryBankAccount.findInBackground(new FindCallback<ParseObject>() {
            @Override
            public void done(List<ParseObject> objects, ParseException e) {
                if(e!=null){
                    e.printStackTrace();
                }else{
                    history = new Stack<>();
                    if(objects.size()>0){
                        for(ParseObject object:objects){
                            String id = object.getString("userId");
                            String process = object.getString("process");
                            java.util.Date date = object.getDate("date");
                            history.push(new History(id,process,date));
                        }
                    }
                    SignIn.mainUser.setHistory(history);
                }
            }
        });
    }
    public void getAllUsers(){
        ParseQuery<ParseObject>
queryBankAccount=ParseQuery.getQuery("UserInfo");
        queryBankAccount.findInBackground(new FindCallback<ParseObject>() {
            @Override
            public void done(List<ParseObject> objects, ParseException e) {
                if(e!=null){
                    e.printStackTrace();
                }else{
                    allUsers = new ArrayList<>();
                    if(objects.size()>0){
                        for(ParseObject object:objects){
                            String name=object.getString("userRealName");
                            String phone=object.getString("phone");
                            String userId=object.getString("username");
                            String address_string=
object.getString("address");
                            String[] str = address_string.split(" ");
                            Address address = new
Address(str[0],str[1],Integer.parseInt(str[2]), str[5],str[6],str[7]);
                            String jobName = object.getString("job");
                            String maxCreditAmount =
object.getString("maxCreditAmount");
                            String interestRate =
object.getString("interestRate");
                            String maxCreditInstallment =
object.getString("maxCreditInstallment");
                            Job tempJob = new
Job(jobName,maxCreditAmount,maxCreditInstallment,interestRate);
                            User tempUser = new User(name,userId,
phone,address,tempJob);

                            getBankAccounts(tempUser);
                            getCreditCards(tempUser);
                            ParseFile
parseFile=(ParseFile)object.get("images");
                            if( parseFile!=null){
                                parseFile.getDataInBackground(new
GetDataCallback() {
```

```
                                        @Override
                                        public void done(byte[] data,
ParseException e) {
                                            if(data!=null && e==null){
                                                Bitmap downloadedImage=
BitmapFactory.decodeByteArray(data,0,data.length);

tempUser.setPhoto(downloadedImage);
                                            }
                                        }
                                    });
                                }
                                allUsers.add(tempUser);
                            }
                        }
                    }
                }
            });
    }
    public void signIn(View view){
        ParseUser.logInInBackground(userName.getText().toString(),
userPass.getText().toString(), new LogInCallback() {
            @Override
            public void done(ParseUser user, ParseException e) {
                if(e !=null){
Toast.makeText(getApplicationContext(),e.getLocalizedMessage(),Toast.LENGTH_L
ONG).show();
                }else{
                    ParseQuery<ParseObject>
query=ParseQuery.getQuery("UserInfo");

query.whereEqualTo("username",userName.getText().toString());
                    query.findInBackground(new FindCallback<ParseObject>() {
                        @Override
                        public void done(List<ParseObject> objects,
ParseException e) {
                            if(e!=null){
                                e.printStackTrace();
                            }else{
                                if(objects.size()>0){
                                    for(ParseObject object:objects){
                                        String
name=object.getString("userRealName");
                                        String
phone=object.getString("phone");
                                        String
userId=object.getString("username");
                                        String address_string=
object.getString("address");
                                        String[] str = address_string.split("
");
                                        Address address = new
Address(str[0],str[1],Integer.parseInt(str[2]), str[5],str[6],str[7]);
                                        String jobName =
object.getString("job");
                                        String maxCreditAmount =
object.getString("maxCreditAmount");
                                        String maxCreditInstallment =
object.getString("maxCreditInstallment");
                                        String interestRate =
object.getString("interestRate");
                                        Job tempJob = new
Job(jobName,maxCreditAmount,maxCreditInstallment,interestRate);
```

```
                                        mainUser = new User(name,userId,
phone,address,tempJob);
                                        ParseFile
parseFile=(ParseFile)object.get("images");
                                       if( parseFile!=null){
                                           parseFile.getDataInBackground(new
GetDataCallback() {
                                               @Override
                                               public void done(byte[] data,
ParseException e) {
                                                   if(data!=null && e==null){
                                                       Bitmap
downloadedImage= BitmapFactory.decodeByteArray(data,0,data.length);
                                                  mainUser.setPhoto(downloadedImage);
                                                   }
                                               }
                                           });
                                       }

Toast.makeText(getApplicationContext(),"Здравствуйте,
"+name,Toast.LENGTH_LONG).show();
                                        getBankAccounts(mainUser);
                                        getCreditCards(mainUser);
                                        getHistory();
                                        getUserBills();
                                        getUserCredits();
                                    }
                                }
                            }
                        }
                    });
                    intent = new Intent(SignIn.this, splashScreen.class);
                    startActivity(intent);
                }
            }
        });
    }
}
```

Листинг Б.3 – Файл ParseStarterClass.java

```
package com.bank.onyx.database;

import android.app.Application;

import com.parse.Parse;

public class ParseStarterClass extends Application {

    @Override

    public void onCreate() {
        super.onCreate();
        Parse.initialize(new Parse.Configuration.Builder(this)
                .applicationId("D8ZgDeOQTsuqVhlij1GLWT2BRwWTALUEQ0lmFNKP")
                .clientKey("pt1i1StL5R4KQLkEK42PJsJBUs6MPAabZiouiikP")
                .server("https://parseapi.back4app.com/")
                .build()
        );
    }
}
```

Листинг Б.4 – Файл MainScreenActivity.java

```
package com.bank.onyx.MainScreen;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;

import android.os.Bundle;
import android.view.MenuItem;
import android.widget.ImageView;
import android.widget.Toast;

import com.bank.onyx.MainScreen.FinanceScreen.CryptoModel;
import com.bank.onyx.MainScreen.FinanceScreen.FinanceFragment;
import com.bank.onyx.R;
import com.bank.onyx.service.ICryptoAPI;
import com.google.android.material.bottomnavigation.BottomNavigationView;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

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

import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;
import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;

public class MainScreenActivity extends AppCompatActivity {

    private BottomNavigationView bottomNavigationView;

    Fragment fragment1 = new AccountFragment();
    final Fragment fragment2 = new CreditFragment();
    final Fragment fragment4 = new BillFragment();
    final Fragment fragment5 = new SettingFragment();

    private Fragment tempFragment=fragment1;

    final FragmentManager fm = getSupportFragmentManager();

    private ArrayList<CryptoModel> cryptoModels;
    private final String BASE_URL = "https://api.nomics.com/v1/";
    private Retrofit retrofit;
    private ImageView downloadingImageView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main_screen);

        Gson gson=new GsonBuilder().setLenient().create();

        retrofit=new Retrofit.Builder().
                baseUrl(BASE_URL).
                addConverterFactory(GsonConverterFactory.create(gson)).
                build();
        loadData();

        bottomNavigationView = findViewById(R.id.bottom_navigation);
```

```
fm.beginTransaction().add(R.id.fragment_container,fragment5,"5").hide(fragmen
t5).commit();
fm.beginTransaction().add(R.id.fragment_container,fragment4,"4").hide(fragmen
t4).commit();
fm.beginTransaction().add(R.id.fragment_container,fragment2,"2").hide(fragmen
t2).commit();
   fm.beginTransaction().add(R.id.fragment_container,fragment1,"1").commit();
        bottomNavigationView.setOnNavigationItemSelectedListener(new
BottomNavigationView.OnNavigationItemSelectedListener() {
            @Override
            public boolean onNavigationItemSelected(@NonNull MenuItem item) {
                switch (item.getItemId()){
                    case R.id.menu1:
                        fragment1 = new AccountFragment();
fm.beginTransaction().add(R.id.fragment_container,fragment1,"1").commit();
fm.beginTransaction().hide(tempFragment).show(fragment1).commit();
                        tempFragment = fragment1;
                        break;
                    case R.id.menu2:
           fm.beginTransaction().hide(tempFragment).show(fragment2).commit();
                        tempFragment = fragment2;
                        break;
                    case R.id.menu3:
getSupportFragmentManager().beginTransaction().hide(tempFragment).add(R.id.fr
agment_container,tempFragment=new
FinanceFragment(cryptoModels)).show(tempFragment).commit();
                        break;
                    case R.id.menu4:
           fm.beginTransaction().hide(tempFragment).show(fragment4).commit();
                        tempFragment = fragment4;
                        break;
                    case R.id.menu5:
fm.beginTransaction().hide(tempFragment).show(fragment5).commit();
                        tempFragment = fragment5;
                        break;
                }
                return true;
            }
        });
    }
    private void loadData(){
        ICryptoAPI cryptoAPI=retrofit.create(ICryptoAPI.class);
        Call<List<CryptoModel>> call=cryptoAPI.getData();
        call.enqueue(new Callback<List<CryptoModel>>() {
            @Override
            public void onResponse(Call<List<CryptoModel>> call,
Response<List<CryptoModel>> response) {
                if(response.isSuccessful()){
                    List<CryptoModel> responseList=response.body();
                    cryptoModels=new ArrayList<>(responseList);
                }
            }

            @Override
            public void onFailure(Call<List<CryptoModel>> call, Throwable t)
{
                Toast.makeText(getApplicationContext(),
t.getLocalizedMessage(), Toast.LENGTH_LONG).show();
            }
        });
    }
}
```

Листинг Б.5 – Файл CreditFragment.java

```
package com.bank.onyx.MainScreen;

import android.content.DialogInterface;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.bank.onyx.Credit.Credit;
import com.bank.onyx.Adapters.CreditAdapter;
import com.bank.onyx.Credit.CustomEventListener;
import com.bank.onyx.R;
import com.bank.onyx.Sign.SignIn;
import com.bank.onyx.UserInfo.BankAccount;
import com.bank.onyx.UserInfo.History;
import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.parse.FindCallback;
import com.parse.ParseException;
import com.parse.ParseObject;
import com.parse.ParseQuery;
import com.parse.SaveCallback;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.List;

import static com.bank.onyx.Sign.SignIn.mainUser;
import static com.parse.Parse.getApplicationContext;

public class CreditFragment extends Fragment {

    private Toolbar toolbarCredit;
    private RecyclerView recyclerViewCredit;
    private ArrayList<Credit> list,searchList;
    private CreditAdapter creditAdapter;
    private FloatingActionButton floatingActionButtonCredit;
    private Credit credit;
    private AdapterView.OnItemClickListener mListener;

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable
ViewGroup container, @Nullable Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_2,container,false);
    }
```

```
    @Override
    public void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }
    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle
savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);
        floatingActionButtonCredit =
getView().findViewById(R.id.floatingActionButton_credit);
        toolbarCredit = getView().findViewById(R.id.toolbar_credit);
        toolbarCredit.setTitle("Кредиты");
        toolbarCredit.setLogo(R.drawable.icon_credit);
((AppCompatActivity)getActivity()).setSupportActionBar(toolbarCredit);
        setHasOptionsMenu(true);
        recyclerViewCredit =
getView().findViewById(R.id.recyclerView_credit);
        recyclerViewCredit.setHasFixedSize(true);
        recyclerViewCredit.setLayoutManager(new
LinearLayoutManager(getContext()));
        list = SignIn.mainUser.getCredits();
        //creditAdapter = new CreditAdapter(getContext(),list);
        creditAdapter = new CreditAdapter(getContext(),list);
        creditAdapter.setListener(new CustomEventListener() {
            @Override
            public void MyEventListener() {
                list= mainUser.getCredits();
                creditAdapter = new CreditAdapter(getContext(),list);
                creditAdapter.setListener(new CustomEventListener() {
                    @Override
                    public void MyEventListener() {
                        list= mainUser.getCredits();
                        creditAdapter = new CreditAdapter(getContext(),list);
                        recyclerViewCredit.setAdapter(creditAdapter);
                    }
                });
                recyclerViewCredit.setAdapter(creditAdapter);
            }
        });
        recyclerViewCredit.setAdapter(creditAdapter);
        floatingActionButtonCredit.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view) {
                AlertDialog.Builder creditPopup = new
AlertDialog.Builder(getContext());
                creditPopup.setTitle("Параметры");
                LayoutInflater inflater = getActivity().getLayoutInflater();
                View dialogView=
inflater.inflate(R.layout.credit_screen_credit_first_step_popup, null);
                creditPopup.setView(dialogView);
                EditText creditAmount =
dialogView.findViewById(R.id.editText_credit_amount) ;
                TextView interestRate =
dialogView.findViewById(R.id.textView_credit_interestRate);
                EditText installment =
dialogView.findViewById(R.id.editText_credit_installment);

                TextView staticMaxAmount =
dialogView.findViewById(R.id.textView_static_credit_max_amount);
                TextView staticMaxInstallment =
dialogView.findViewById(R.id.textView_static_credit_max_installment);
                interestRate.setText("Ставка:
%"+SignIn.mainUser.getJob().getInterestRate());
```

```
staticMaxAmount.setText(SignIn.mainUser.getJob().getMaxCreditAmount());

staticMaxInstallment.setText(SignIn.mainUser.getJob().getMaxCreditInstallment
());
                creditPopup.setNegativeButton("Подтердить", new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialogInterface, int
i) {
                       if(!(creditAmount.getText().toString().matches("") ||
installment.getText().toString().matches(""))){
                            int maxAmount =
Integer.parseInt(SignIn.mainUser.getJob().getMaxCreditAmount());
                            int maxInstallment =
Integer.parseInt(SignIn.mainUser.getJob().getMaxCreditInstallment());
                            int currentAmount =
Integer.parseInt(creditAmount.getText().toString());
                            int currentInstallment =
Integer.parseInt(installment.getText().toString());
                            if(maxAmount>=currentAmount &&
maxInstallment>=currentInstallment){
                                AlertDialog.Builder creditPopupSecond = new
AlertDialog.Builder(getContext());
                                creditPopupSecond.setTitle("Условия
кредитования");
                                LayoutInflater inflaterSecond =
getActivity().getLayoutInflater();
                                View dialogViewSecond=
inflaterSecond.inflate(R.layout.credit_screen_credit_second_step_popup,
null);
                                creditPopupSecond.setView(dialogViewSecond);
                                TextView amountSecond =
dialogViewSecond.findViewById(R.id.textView_taken_amount);
                                TextView payAmountSecond =
dialogViewSecond.findViewById(R.id.textView_pay_amount);

amountSecond.setText(String.valueOf(currentAmount));

payAmountSecond.setText(String.valueOf(currentAmount +
((currentAmount/Integer.parseInt(SignIn.mainUser.getJob().getInterestRate()))
*currentInstallment)/1200));

creditPopupSecond.setNegativeButton("Подтердить", new
DialogInterface.OnClickListener() {
                                    @Override
                                    public void onClick(DialogInterface
dialogInterface, int i) {
                                        int payCurrentAmount = currentAmount +
((currentAmount*Integer.parseInt(SignIn.mainUser.getJob().getInterestRate()))
*currentInstallment)/1200;
                                        Credit tempCredit = new
Credit(currentAmount,currentInstallment,Integer.parseInt(SignIn.mainUser.getJ
ob().getInterestRate()),payCurrentAmount);
                                        receiveCredit(tempCredit);
                                    }
                                });
                                creditPopupSecond.create().show();
                            }
                            else {
                                Toast.makeText(getContext(), "Неправильное
значение", Toast.LENGTH_SHORT).show();
                            }
                        }
```

```
                        else {
                            Toast.makeText(getContext(), "Неправильное
значение", Toast.LENGTH_SHORT).show();
                        }
                    }
                });
                creditPopup.create().show();
            }
        });
    }
    public void updateBankAccount(BankAccount bankac){
        ParseQuery<ParseObject>
queryBankAccount=ParseQuery.getQuery("BankAccount");
        queryBankAccount.whereEqualTo("accountNo", bankac.getAccountno());
        queryBankAccount.findInBackground(new FindCallback<ParseObject>() {
            @Override
            public void done(List<ParseObject> objects, ParseException e) {
                if(e!=null){
                    e.printStackTrace();
                }else{
                    if(objects.size()>0){
                        for(ParseObject object:objects){
                            object.deleteInBackground();
                            accountsToDatabase(bankac);
                        }
                    }
                }
            }
        });
    }
    public void accountsToDatabase(BankAccount bankAc){
        ParseObject object=new ParseObject("BankAccount");
        object.put("accountNo",bankAc.getAccountno());
        object.put("userId", SignIn.mainUser.getId());
        object.put("cash",String.valueOf(bankAc.getCash()));
        object.saveInBackground(new SaveCallback() {
            @Override
            public void done(ParseException e) {
                if(e != null){

//Toast.makeText(getApplicationContext(),e.getLocalizedMessage().toString(),T
oast.LENGTH_LONG).show();
                }
                else{
                }
            }
        });
    }

    public void creditToDatabase(Credit tempCredit){
        ParseObject object=new ParseObject("Credit");
        object.put("amount",String.valueOf(tempCredit.getAmount()));
        object.put("username", SignIn.mainUser.getId());
       object.put("installment",String.valueOf(tempCredit.getInstallment()));
     object.put("interestRate",String.valueOf(tempCredit.getInterestRate()));
        object.put("payAmount",String.valueOf(tempCredit.getPayAmount()));
        object.saveInBackground(new SaveCallback() {
            @Override
            public void done(ParseException e) {
                if(e != null){

//Toast.makeText(getApplicationContext(),e.getLocalizedMessage().toString(),T
oast.LENGTH_LONG).show();
                }
```

```
                else{
                }
            }
        });
    }
    public java.util.Date getDateRealTime(){
        SimpleDateFormat format = new SimpleDateFormat("dd/MM/yyyy");
        Date currentTime = Calendar.getInstance().getTime();
        return currentTime;
    }
    public void historyToDatabase(History history){
        ParseObject object=new ParseObject("History");
        object.put("process",history.getProcess());
        object.put("userId", mainUser.getId());
        object.put("date",history.getDateDate());
        object.saveInBackground(new SaveCallback() {
            @Override
            public void done(ParseException e) {
                if(e != null){

Toast.makeText(getApplicationContext(),e.getLocalizedMessage().toString(),Toa
st.LENGTH_LONG).show();
                }
                else{
                }
            }
        });
    }
    public void receiveCredit(Credit tempCredit){
        int max=Integer.MIN_VALUE;
        int index=-1;
        for(int i =0;i<SignIn.mainUser.getBankAccounts().size();i++){
            if(SignIn.mainUser.getBankAccounts().get(i).getCash()>max){
                max=SignIn.mainUser.getBankAccounts().get(i).getCash();
                index=i;
            }
        }
        if(SignIn.mainUser.getBankAccounts().size()>0 ){

SignIn.mainUser.getBankAccounts().get(index).setCash(SignIn.mainUser.getBankA
ccounts().get(index).getCash()+tempCredit.getAmount());
            updateBankAccount(SignIn.mainUser.getBankAccounts().get(index));

Toast.makeText(getApplicationContext(),"Одобрено",Toast.LENGTH_LONG).show();
            creditToDatabase(tempCredit);
            list.add(tempCredit);
            creditAdapter = new CreditAdapter(getContext(),list);
            creditAdapter.setListener(new CustomEventListener() {
                @Override
                public void MyEventListener() {
                    list= mainUser.getCredits();
                    creditAdapter = new CreditAdapter(getContext(),list);
                    creditAdapter.setListener(new CustomEventListener() {
                        @Override
                        public void MyEventListener() {
                            list= mainUser.getCredits();
                            creditAdapter = new
CreditAdapter(getContext(),list);
                            recyclerViewCredit.setAdapter(creditAdapter);
                        }
                    });
                    recyclerViewCredit.setAdapter(creditAdapter);
                }
            });
```

```
            recyclerViewCredit.setAdapter(creditAdapter);
            History hs = new History(mainUser.getId(),"Кредитование.(" +
tempCredit.getAmount()+")",getDateRealTime() );
            mainUser.getHistory().push(hs);
            historyToDatabase(hs);
        }
        else {
            Toast.makeText(getApplicationContext(),"У вас нет банковского
счета.",Toast.LENGTH_LONG).show();
        }
    }
}
```

Листинг Б.6 – Файл CryptoLookAdapter.java

```
package com.bank.onyx.Adapters;

import android.app.Activity;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.ahmadrosid.svgloader.SvgLoader;
import com.bank.onyx.MainScreen.FinanceScreen.CryptoModel;
import com.bank.onyx.R;
import com.squareup.picasso.Picasso;

import java.util.ArrayList;

public class CryptoLookAdapter extends
RecyclerView.Adapter<CryptoLookAdapter.PostHolder> {
    private Context context;
    private ArrayList<CryptoModel> models;
    private Activity activity;
    class PostHolder extends RecyclerView.ViewHolder{
        private  Context context;
        private TextView
cryptoNameText,cryptoSymbolText,cryptoAmountText,cryptoPriceText;
        private ImageView cryptoImageView;
        public PostHolder(@NonNull View itemView, Context context) {
            super(itemView);
            this.context=context;
            cryptoNameText=itemView.findViewById(R.id.crypto_symbol_text);
            cryptoAmountText=itemView.findViewById(R.id.crypto_name_text);
            cryptoImageView=itemView.findViewById(R.id.crypto_imageview);

cryptoPriceText=itemView.findViewById(R.id.crypto_price_text_view);
        }
    }
    public CryptoLookAdapter(ArrayList<CryptoModel> models, Activity
activity, Context context) {
        this.models = models;
        this.activity=activity;
        this.context=context;
    }
    @NonNull
    @Override
```

```
    public CryptoLookAdapter.PostHolder onCreateViewHolder(@NonNull ViewGroup
parent, int viewType) {
        LayoutInflater
layoutInflater=LayoutInflater.from(parent.getContext());
        View view =layoutInflater.inflate(R.layout.custom_view,parent,false);
        return new CryptoLookAdapter.PostHolder(view,context);
    }
    @Override
    public void onBindViewHolder(@NonNull PostHolder holder, int position) {

holder.cryptoNameText.setText(models.get(position).getCurrencyName());
        holder. cryptoAmountText.setText(models.get(position).getAmount());
        holder. cryptoPriceText.setText(models.get(position).getPrice()+"
USD");

if(models.get(position).getLogoUrl().substring(models.get(position).getLogoUr
l().length()-3).equalsIgnoreCase("svg")){
            SvgLoader.pluck()
                    .with(activity)
                    .setPlaceHolder(R.mipmap.ic_launcher,
R.mipmap.ic_launcher)

.load(models.get(position).getLogoUrl(),holder.cryptoImageView);
        }else{

Picasso.get().load(models.get(position).getLogoUrl()).into(holder.cryptoImage
View);
        }
    }
    @Override
    public int getItemCount() {
        return models.size();
    }
}
```

Листинг Б.7 – Файл CryptoPostAdapter.java

```
package com.bank.onyx.Adapters;

import android.annotation.SuppressLint;
import android.app.Activity;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import android.widget.ImageView;
import android.widget.TextView;

import androidx.annotation.NonNull;

import androidx.recyclerview.widget.RecyclerView;

import com.ahmadrosid.svgloader.SvgLoader;
import com.bank.onyx.MainScreen.FinanceScreen.CryptoModel;
import com.bank.onyx.MainScreen.FinanceScreen.ItemClickListener;
import com.bank.onyx.R;
import com.squareup.picasso.Picasso;

import java.util.ArrayList;
public class CryptoPostAdapter  extends
RecyclerView.Adapter<CryptoPostAdapter.PostHolder> {
    private Context context;
```

```
    private ArrayList<CryptoModel> models;
    private Activity activity;
    private ItemClickListener itemClickListener;
    class PostHolder extends RecyclerView.ViewHolder{
        private  Context context;
        private TextView cryptoNameText,cryptoSymbolText,cryptoPriceText;
        private ImageView cryptoImageView;
        public PostHolder(@NonNull View itemView, Context context) {
            super(itemView);
            this.context=context;
            cryptoNameText=itemView.findViewById(R.id.crypto_name_text);
            cryptoSymbolText=itemView.findViewById(R.id.crypto_symbol_text);

cryptoPriceText=itemView.findViewById(R.id.crypto_price_text_view);
            cryptoImageView=itemView.findViewById(R.id.crypto_imageview);
        }
    }
    public CryptoPostAdapter(ArrayList<CryptoModel> models, Activity
activity, Context context, ItemClickListener itemClickListener) {
        this.models = models;
        this.activity=activity;
        this.context=context;
        this.itemClickListener=itemClickListener;
    }
    @NonNull
    @Override
    public PostHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
        LayoutInflater
layoutInflater=LayoutInflater.from(parent.getContext());
        View view =layoutInflater.inflate(R.layout.custom_view,parent,false);
        return new PostHolder(view,context);
    }
    @Override
    public void onBindViewHolder(@NonNull PostHolder holder,
@SuppressLint("RecyclerView") int position) {

holder.cryptoNameText.setText(models.get(position).getCurrencyName());

holder.cryptoSymbolText.setText(models.get(position).getCurrencySymbol());
        holder.cryptoPriceText.setText(models.get(position).getPrice());
        //for .svg image file

if(models.get(position).getLogoUrl().substring(models.get(position).getLogoUr
l().length()-3).equalsIgnoreCase("svg")){
            SvgLoader.pluck()
                    .with(activity)
                    .setPlaceHolder(R.mipmap.ic_launcher,
R.mipmap.ic_launcher)

.load(models.get(position).getLogoUrl(),holder.cryptoImageView);
        }else{

Picasso.get().load(models.get(position).getLogoUrl()).into(holder.cryptoImage
View);
        }
        models.get(position).setImage(holder.cryptoImageView);
        final CryptoModel cryptoModel = models.get(position);
        // set click listener
        holder.itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                itemClickListener.onItemClicked(holder, cryptoModel,
position);
```

```
            }
        });
    }
    @Override
    public int getItemCount() {
        return models.size();
    }
}
```

Приложение В

ЛИСТИНГ ПРИЛОЖЕНИЯ

(рекомендуемое)

Листинг В.1 – Файл UserAdapter.java

```
package com.bank.onyx.Adapters;

import android.app.Activity;

import android.app.AlertDialog;
import android.content.DialogInterface;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;
import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;
import com.bank.onyx.R;
import com.bank.onyx.UserInfo.User;

import java.util.ArrayList;

public class UserAdapter extends RecyclerView.Adapter<UserAdapter.ViewHolder>
{
    ArrayList<User> users;
    Activity context;
    public UserAdapter(ArrayList<User> myMovieData,Activity activity) {
        this.users = myMovieData;
        this.context = activity;
    }
    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
        LayoutInflater layoutInflater =
LayoutInflater.from(parent.getContext());
        View view =
layoutInflater.inflate(R.layout.user_admin_card_view,parent,false);
        ViewHolder viewHolder = new ViewHolder(view);
        return viewHolder;
    }
    @NonNull
    @Override
    public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
        final User account = users.get(position);
        holder.textviewmoney.setText(account.getName());
        holder.textviewbankno.setText(account.getId());
        holder.photo.setImageBitmap(account.getPhoto());
        holder.itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                AlertDialog.Builder buyCryptoPopup=new
AlertDialog.Builder(context);
                buyCryptoPopup.setTitle("ИНФОРМАЦИЯ ПОЛЬЗОВАТЕЛЯ");
                LayoutInflater inflater = context.getLayoutInflater();
                View dialogView=
inflater.inflate(R.layout.admin_screen_user_popup, null);
                buyCryptoPopup.setView(dialogView);
                TextView phone=
dialogView.findViewById(R.id.text_view_phone_number);
                TextView adres=
dialogView.findViewById(R.id.text_view_adres);
                TextView job= dialogView.findViewById(R.id.text_view_job);
                TextView bankcount=
dialogView.findViewById(R.id.text_view_bank_account_counter);
```

```
                TextView creditcount=
dialogView.findViewById(R.id.text_view_credit_card_counter);
                TextView interestRate=
dialogView.findViewById(R.id.text_view_interest_rate);
                interestRate.setText("Кредитная ставка: " +
account.getJob().getInterestRate() +"\n" + "Максимальная сумма кредита: " +
account.getJob().getMaxCreditAmount());
                phone.setText("Телефон: +" + account.getPhoneNumber());
                adres.setText("Адрес: " + account.addressWrite());
                job.setText("Профессия: " + account.getJob().getName());
                String tempStr = "Банковские счета:\n";
                for (int i = 0; i < account.getBankAccounts().size(); i++) {
                    tempStr+= account.getBankAccounts().get(i).getAccountno()
+ ": " + account.getBankAccounts().get(i).getCash() + "₽";
                    if (account.getBankAccounts().size()-1!=i){
                        tempStr+= "\n";
                    }
                }
                String tempCC = "Кредитные карты:\n";
                for (int i = 0; i < account.getCreditcards().size(); i++) {
                    tempCC+=
account.getCreditcards().get(i).getCreditCardNo() + ": " +
account.getCreditcards().get(i).getLimit() + "₽";
                    if (account.getCreditcards().size()-1!=i){
                        tempCC+= "\n";
                    }
                }
                bankcount.setText(tempStr);
                creditcount.setText(tempCC);
                buyCryptoPopup.setNegativeButton("ЗАКРЫТЬ", new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                    }
                });
                buyCryptoPopup.create().show();
            }
        });
    }
    @Override
    public int getItemCount() {
        return users.size();
    }
    public class ViewHolder extends RecyclerView.ViewHolder{
        TextView textviewbankno;
        TextView textviewmoney;
        ImageView photo;
        public ViewHolder(@NonNull View itemView) {
            super(itemView);
            textviewbankno =
itemView.findViewById(R.id.text_view_bank_account_no);
            textviewmoney =
itemView.findViewById(R.id.text_view_bank_account_money);
            photo = itemView.findViewById(R.id.bank_account_ImageView);
        }
    }
}
```

Листинг В.2 – Файл BillFragment.java

```
package com.bank.onyx.MainScreen;

import android.content.DialogInterface;
```

```
import android.os.Bundle;
import android.text.InputType;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.EditText;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.bank.onyx.Bill.Bill;
import com.bank.onyx.Adapters.BillAdapter;
import com.bank.onyx.Bill.ElectricBill;
import com.bank.onyx.Bill.GasBill;
import com.bank.onyx.Bill.InternetBill;
import com.bank.onyx.Bill.PhoneBill;
import com.bank.onyx.Bill.WaterBill;
import com.bank.onyx.R;
import com.bank.onyx.Sign.SignIn;
import com.bank.onyx.UserInfo.BankAccount;
import com.bank.onyx.UserInfo.History;
import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.parse.FindCallback;
import com.parse.ParseException;
import com.parse.ParseObject;
import com.parse.ParseQuery;
import com.parse.SaveCallback;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.List;

import static com.bank.onyx.Sign.SignIn.mainUser;
import static com.parse.Parse.getApplicationContext;

public class BillFragment extends Fragment{
    private Toolbar toolbar;
    private RecyclerView recyclerView;
    private ArrayList<Bill> list;
    private BillAdapter billAdapter;
    private FloatingActionButton floatingActionButtonBill;
    private Bill bill;
    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable
ViewGroup container, @Nullable Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_4,container,false);
    }
    @Override
    public void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }
    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle
savedInstanceState) {
```

```
        super.onViewCreated(view, savedInstanceState);
        floatingActionButtonBill =
getView().findViewById(R.id.floatingActionButton_bill);
        toolbar = getView().findViewById(R.id.toolbar_bill);
        toolbar.setTitle("Оплата счетов");
        toolbar.setLogo(R.drawable.icon_receipt);
        ((AppCompatActivity)getActivity()).setSupportActionBar(toolbar);
        setHasOptionsMenu(true);
        recyclerView = getView().findViewById(R.id.recyclerView_bill);
        recyclerView.setHasFixedSize(true);
        recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));
        list = SignIn.mainUser.getUserbills();
        billAdapter = new BillAdapter(getContext(),list);
        recyclerView.setAdapter(billAdapter);
        floatingActionButtonBill.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view) {
                final EditText editText = new EditText(getContext());
                editText.setHint("введите сумму");
                editText.setInputType(InputType.TYPE_CLASS_NUMBER);
                AlertDialog.Builder ad = new
AlertDialog.Builder(getContext());
                ad.setTitle("Выберите услугу");
                ad.setIcon(R.drawable.icon_bill);
                ad.setView(editText);
                String[] items = {"Электроэнергия","Газ","Интернет","Сотовая
связь","Водоснабжение"};
                final int[] checkedItem = {0};
                ad.setSingleChoiceItems(items, checkedItem[0], new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialogInterface, int
i) {
                        switch (i) {
                            case 0:
                                checkedItem[0] =i;
                                break;
                            case 1:
                                checkedItem[0] =i;
                                break;
                            case 2:
                                checkedItem[0] =i;
                                break;
                            case 3:
                                checkedItem[0] =i;
                                break;
                            case 4:
                                checkedItem[0] =i;
                                break;
                        }
                    }
                });
                ad.setNegativeButton("ОПЛАТИТЬ", new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialogInterface, int
i) {
                        i= checkedItem[0];
                        switch (i) {
                            case 0:
                                try {
                                    bill=new ElectricBill();
                                }catch (NumberFormatException e){
```

```
                                }
                                break;
                            case 1:
                                try {
                                    bill=new GasBill();
                                }catch (NumberFormatException e){
                                }
                                break;
                            case 2:
                                try {
                                    bill=new InternetBill();
                                }catch (NumberFormatException e){
                                }
                                break;
                            case 3:
                                try {
                                    bill=new PhoneBill();
                                }catch (NumberFormatException e){
                                }
                                break;
                            case 4:
                                try {
                                    bill=new WaterBill();
                                }catch (NumberFormatException e){
                                }
                                break;
                        }
                        try {

bill.setAmount(Integer.parseInt(editText.getText().toString()));
                            setDate(bill);
                            payBill(bill);
                        }catch (NumberFormatException e){
                        }
                    }
                });
                ad.create().show();
            }
        });
    }
    public void payBill(Bill bill){
        int max=Integer.MIN_VALUE;
        int index=-1;
        for(int i =0;i<SignIn.mainUser.getBankAccounts().size();i++){
            if(SignIn.mainUser.getBankAccounts().get(i).getCash()>max){
                max=SignIn.mainUser.getBankAccounts().get(i).getCash();
                index=i;
            }
        }
        if(SignIn.mainUser.getBankAccounts().size()>0 &&

bill.getAmount()<=SignIn.mainUser.getBankAccounts().get(index).getCash()){
            SignIn.mainUser.getBankAccounts().get(index).

setCash(SignIn.mainUser.getBankAccounts().get(index).getCash()-
bill.getAmount());
            updateBankAccount(SignIn.mainUser.getBankAccounts().get(index));
            Toast.makeText(getApplicationContext(),"Счет
оплачен",Toast.LENGTH_LONG).show();
            billToDatabase(bill);
            list.add(bill);
            billAdapter = new BillAdapter(getContext(),list);
            recyclerView.setAdapter(billAdapter);
```

```
            History hs = new History(mainUser.getId(),"Счет оплачен.(" +
bill.getType()+")",getDateRealTime() );
            mainUser.getHistory().push(hs);
            historyToDatabase(hs);
        }
        else {
            Toast.makeText(getApplicationContext(),"Недостаточно
средств.",Toast.LENGTH_LONG).show();
        }
    }
    public void updateBankAccount(BankAccount bankac){
        ParseQuery<ParseObject>
queryBankAccount=ParseQuery.getQuery("BankAccount");
        queryBankAccount.whereEqualTo("accountNo", bankac.getAccountno());
        queryBankAccount.findInBackground(new FindCallback<ParseObject>() {
            @Override
            public void done(List<ParseObject> objects, ParseException e) {
                if(e!=null){
                    e.printStackTrace();
                }else{
                    if(objects.size()>0){
                        for(ParseObject object:objects){
                            object.deleteInBackground();
                            accountsToDatabase(bankac);
                        }
                    }
                }
            }
        });
    }
    public void accountsToDatabase(BankAccount bankAc){
        ParseObject object=new ParseObject("BankAccount");
        object.put("accountNo",bankAc.getAccountno());
        object.put("userId", SignIn.mainUser.getId());
        object.put("cash",String.valueOf(bankAc.getCash()));
        object.saveInBackground(new SaveCallback() {
            @Override
            public void done(ParseException e) {
                if(e != null){

Toast.makeText(getApplicationContext(),e.getLocalizedMessage().
                            toString(),Toast.LENGTH_LONG).show();
                }
                else{
                }
            }
        });
    }
    public void billToDatabase(Bill bill){
        ParseObject object=new ParseObject("Bill");
        object.put("type",bill.getType());
        object.put("username", SignIn.mainUser.getId());
        object.put("amount",String.valueOf(bill.getAmount()));

object.put("date",bill.getDate().getDay()+"/"+bill.getDate().getMonth()+
                "/"+bill.getDate().getYear());
        object.saveInBackground(new SaveCallback() {
            @Override
            public void done(ParseException e) {
                if(e != null){
                }
                else{
                }
            }
```

```
        });
    }
    public void setDate(Bill newBill){
        SimpleDateFormat format = new SimpleDateFormat("dd/MM/yyyy");
        java.util.Date currentTime = Calendar.getInstance().getTime();
        String str = format.format(currentTime);
        String [] date = str.split("/");
        newBill.getDate().setDay(date[0]);
        newBill.getDate().setMonth(date[1]);
        newBill.getDate().setYear(date[2]);
    }
    public void historyToDatabase(History history){
        ParseObject object=new ParseObject("History");
        object.put("process",history.getProcess());
        object.put("userId", mainUser.getId());
        object.put("date",history.getDateDate());
        object.saveInBackground(new SaveCallback() {
            @Override
            public void done(ParseException e) {
                if(e != null){
                }
                else{
                }
            }
        });
    }
    public java.util.Date getDateRealTime(){
        SimpleDateFormat format = new SimpleDateFormat("dd/MM/yyyy");
        Date currentTime = Calendar.getInstance().getTime();
        return currentTime;
    }
}
```

Листинг В.3 – Файл BillAdapter.java

```
package com.bank.onyx.Adapters;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.cardview.widget.CardView;
import androidx.recyclerview.widget.RecyclerView;
import com.bank.onyx.Bill.Bill;
import com.bank.onyx.R;
import java.util.List;

public class BillAdapter extends
RecyclerView.Adapter<BillAdapter.CardViewObjectHolder> {
    private Context mContext;
    private List<Bill> list;
    public BillAdapter(Context mContext, List<Bill> list) {
        this.mContext = mContext;
        this.list = list;
    }
    public class CardViewObjectHolder extends RecyclerView.ViewHolder{
        public TextView textViewBillType;
        public TextView textViewBillAmount;
        public TextView textViewBillDate;
        public CardView cardViewBill;
```

```
        public CardViewObjectHolder(View view){
            super(view);
            textViewBillType = view.findViewById(R.id.textView_bill_type);
            textViewBillAmount =
view.findViewById(R.id.textView_bill_amount);
            textViewBillDate = view.findViewById(R.id.textView_bill_date);
            cardViewBill = view.findViewById(R.id.cardView_bill);
        }
    }
    @NonNull
    @Override
    public CardViewObjectHolder onCreateViewHolder(@NonNull ViewGroup parent,
int viewType) {
        View itemView = LayoutInflater.from(parent.getContext())
                .inflate(R.layout.bill_cardview,parent,false);
        return new CardViewObjectHolder(itemView);
    }
    @Override
    public void onBindViewHolder(@NonNull CardViewObjectHolder holder, int
position) {
        Bill bill = list.get(position);
        holder.textViewBillType.setText(bill.getType());
        holder.textViewBillAmount.setText(bill.getAmount()+" ₽");
        String dateStr = bill.getDate().getDay() +"."+
bill.getDate().getMonth() +
                "."+ bill.getDate().getYear();
        holder.textViewBillDate.setText(dateStr);
    }
    @Override
    public int getItemCount() {
        return list.size();
    }
}
```

Листинг В.4 – Файл SettingFragment.java

```
package com.bank.onyx.MainScreen;

import android.Manifest;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.graphics.Bitmap;
import android.graphics.ImageDecoder;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.os.Handler;
import android.provider.MediaStore;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.RelativeLayout;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
```

```
import androidx.appcompat.app.AlertDialog;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.fragment.app.Fragment;
import static android.app.Activity.RESULT_OK;

import com.bank.onyx.Job.Contractor;
import com.bank.onyx.Job.Doctor;
import com.bank.onyx.Job.Driver;
import com.bank.onyx.Job.Engineer;
import com.bank.onyx.Job.Entrepreneur;
import com.bank.onyx.Job.Farmer;
import com.bank.onyx.Job.Job;
import com.bank.onyx.Job.Police;
import com.bank.onyx.Job.Soldier;
import com.bank.onyx.Job.Sportsman;
import com.bank.onyx.Job.Student;
import com.bank.onyx.Job.Teacher;
import com.bank.onyx.Job.Waiter;
import com.bank.onyx.Job.Worker;
import com.bank.onyx.R;
import com.bank.onyx.Sign.SignIn;
import com.bank.onyx.UserInfo.Address;
import com.parse.DeleteCallback;
import com.parse.FindCallback;
import com.parse.LogOutCallback;
import com.parse.ParseException;
import com.parse.ParseFile;
import com.parse.ParseObject;
import com.parse.ParseQuery;
import com.parse.ParseUser;
import com.parse.SaveCallback;

import java.io.ByteArrayOutputStream;
import java.util.List;

import static com.bank.onyx.Sign.SignIn.mainUser;
import static com.parse.Parse.getApplicationContext;

public class SettingFragment extends Fragment {
   private TextView name,phone,userId,userAdress,prof,logOut;
    private Bitmap selectedImage;
    private ImageView imageView;
    private Spinner spinner;
    private ArrayAdapter<String> jobArrayAdapter;
    private  RelativeLayout
relativeLayoutMobileRow,relativeLayoutNameRow,relativeLayoutAddressRow,relati
veLayoutPassRow,relativeLayoutDeleteRow;
    private String newChangeItem ="";
    private Address newAddress;
    private String [] jobs;
    private Job [] defaultJobs;
    private  String job;
    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable
ViewGroup container, @Nullable Bundle savedInstanceState) {
        final View rootView = inflater.inflate(R.layout.activity_profile,
container, false);
        name = (TextView)
rootView.findViewById(R.id.settings_userNameTextView);
        phone=(TextView)rootView.findViewById(R.id.settings_phoneTextView);
        userId=(TextView)rootView.findViewById(R.id.setting_userId_Big);
```

```
userAdress=(TextView)rootView.findViewById(R.id.settings_addressTextView);
        prof=(TextView)rootView.findViewById(R.id.settings_profTextView);
        imageView=rootView.findViewById(R.id.fragment5_ImageView);

relativeLayoutMobileRow=rootView.findViewById(R.id.account_mobile_row);

relativeLayoutNameRow=rootView.findViewById(R.id.account_setting_name_row);

relativeLayoutAddressRow=rootView.findViewById(R.id.settings_addres_row);

relativeLayoutPassRow=rootView.findViewById(R.id.settings_change_pass_row);

relativeLayoutDeleteRow=rootView.findViewById(R.id.settings_account_delete_ro
w);
        logOut=rootView.findViewById(R.id.setting_logout_textView);
        spinner = rootView.findViewById(R.id.jobSpinner);
        name.setText(mainUser.getName());
        phone.setText(mainUser.getPhoneNumber());
        userId.setText(mainUser.getId());
        userAdress.setText(mainUser.addressWrite());
        prof.setText(mainUser.getJob().getName());
        if(mainUser.getPhoto()!=null){
            imageView.setImageBitmap(mainUser.getPhoto());
        }
        defineJobSpinner();
        return rootView;
    }
    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle
savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);
        relativeLayoutNameRow.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                changeName(v);
            }
        });
        relativeLayoutMobileRow.setOnClickListener(new View.OnClickListener()
{
            @Override
            public void onClick(View v) {
                changePhone(v);
            }
        });
        imageView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                selectImage(v);
            }
        });
        relativeLayoutPassRow.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                changePass(v);
            }
        });
relativeLayoutAddressRow.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        changeAddress(v);
    }
});
relativeLayoutDeleteRow.setOnClickListener(new View.OnClickListener() {
```

```
    @Override
    public void onClick(View v) {
        deleteAccount(v);
    }
});
logOut.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        logOut(v);
    }
});
    }
    public void changeName(View v){
        final EditText editText = new EditText(getContext());
        editText.setHint("Введите имя");
        AlertDialog.Builder ad = new AlertDialog.Builder(getContext());
        ad.setTitle("Изменить имя");
        ad.setIcon(R.drawable.icon_user_id);
        ad.setView(editText);
        ad.setPositiveButton("ПОДТВЕРДИТЬ", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                newChangeItem=editText.getText().toString();
                change( newChangeItem,"userRealName");
                name.setText( newChangeItem);
                mainUser.setName( newChangeItem);
            }
        });
        ad.setNegativeButton("ОТМЕНИТЬ", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {

Toast.makeText(getContext(),"Отменено",Toast.LENGTH_SHORT).show();
            }
        });
        ad.create().show();
    }
    public void changePass(View v){
        final EditText editText = new EditText(getContext());
        editText.setHint("Введите пароль");
        AlertDialog.Builder ad = new AlertDialog.Builder(getContext());
        ad.setTitle("Изменить пароль");
        ad.setIcon(R.drawable.icon_lock);
        ad.setView(editText);
        ad.setPositiveButton("ПОДТВЕРДИТЬ", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                newChangeItem=editText.getText().toString();
                ParseUser user=ParseUser.getCurrentUser();
               user.setPassword(newChangeItem);
               user.saveInBackground(new SaveCallback() {
                   @Override
                   public void done(ParseException e) {
                       if(e ==null ) {
                           Toast.makeText(getContext(),"Пароль
изменен",Toast.LENGTH_SHORT).show();
                       } else {

Toast.makeText(getContext(),e.getLocalizedMessage(),Toast.LENGTH_SHORT).show(
);
                       }
```

```
                    }
                });
            }
        });
        ad.setNegativeButton("ОТМЕНИТЬ", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {

Toast.makeText(getContext(),"Отменено",Toast.LENGTH_SHORT).show();
            }
        });
        ad.create().show();
    }
    public void changeAddress(View v){
        AlertDialog.Builder ad = new AlertDialog.Builder(getContext());
        ad.setTitle("Изменить адрес");
        ad.setIcon(R.drawable.ic_address);
        LayoutInflater inflater = getActivity().getLayoutInflater();
        View dialogView= inflater.inflate(R.layout.settings_address_popup,
null);
        ad.setView(dialogView);
        ad.setPositiveButton("ПОДТВЕРДИТЬ", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                TextView
street=dialogView.findViewById(R.id.setting_address_street);
                TextView
floor=dialogView.findViewById(R.id.setting_address_floor);
                TextView
houseNo=dialogView.findViewById(R.id.setting_address_house);
                TextView
country=dialogView.findViewById(R.id.setting_address_country);
                TextView district
=dialogView.findViewById(R.id.setting_address_district);
                TextView
town=dialogView.findViewById(R.id.setting_address_town);
                TextView locality
=dialogView.findViewById(R.id.setting_address_locality);
                if(street.getText().toString() !=null &&
district.getText().toString()!=null && floor.getText().toString()!=null &&
houseNo.getText().toString()!=null && town.getText().toString()!=null &&
locality.getText().toString()!=null && country.getText().toString()!=null){
                    newAddress= new Address(street.getText().toString(),
district.getText().toString(),Integer.parseInt(floor.getText().toString()),
town.getText().toString(),
locality.getText().toString(),country.getText().toString());
                    mainUser.setAddress(newAddress);
                    userAdress.setText(mainUser.addressWrite());
                    change(mainUser.addressWrite(),"address");
                }else{
                    Toast.makeText(getContext(),"Заполните все
поля",Toast.LENGTH_SHORT).show();
                }
            }
        });
        ad.setNegativeButton("ОТМЕНИТЬ", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {

Toast.makeText(getContext(),"Отменено",Toast.LENGTH_SHORT).show();
            }
```

```
        });
        ad.create().show();
    }
    public void changePhone(View v){
        final EditText editTextPhone = new EditText(getContext());
        editTextPhone.setHint("Введите телефон");
        AlertDialog.Builder ad = new AlertDialog.Builder(getContext());
        ad.setTitle("Изменить телефон");
        ad.setIcon(R.drawable.ic_mobile);
        ad.setView(editTextPhone);
        ad.setPositiveButton("ПОДТВЕРДИТЬ", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                newChangeItem =editTextPhone.getText().toString();
                change(newChangeItem,"phone");
                mainUser.setPhoneNumber(newChangeItem);
                phone.setText(newChangeItem);
            }
        });
        ad.setNegativeButton("ОТМЕНИТЬ", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {

Toast.makeText(getContext(),"Отменено",Toast.LENGTH_SHORT).show();
            }
        });
        ad.create().show();
    }
    private void deleteAccount(View v){
        ParseUser user = ParseUser.getCurrentUser();
        ParseQuery<ParseObject>
queryBankAccount=ParseQuery.getQuery("BankAccount");
        queryBankAccount.whereEqualTo("userId", mainUser.getId());
        queryBankAccount.findInBackground(new FindCallback<ParseObject>() {
            @Override
            public void done(List<ParseObject> objects, ParseException e) {
                if(e!=null){
                    e.printStackTrace();
                }else{
                    if(objects.size()>0){
                        ParseObject.deleteAllInBackground(objects);
                    }
                }
            }
        });
        ParseQuery<ParseObject> queryDel=ParseQuery.getQuery("Bill");
        queryDel.whereEqualTo("username", mainUser.getId());
        queryDel.findInBackground(new FindCallback<ParseObject>() {
            @Override
            public void done(List<ParseObject> objects, ParseException e) {
                if(e!=null){
                    e.printStackTrace();
                }else{
                    if(objects.size()>0){
                        ParseObject.deleteAllInBackground(objects);
                    }
                }
            }
        });
        ParseQuery<ParseObject> queryDel2=ParseQuery.getQuery("Credit");
        queryDel2.whereEqualTo("username", mainUser.getId());
        queryDel2.findInBackground(new FindCallback<ParseObject>() {
```

```
            @Override
            public void done(List<ParseObject> objects, ParseException e) {
                if(e!=null){
                    e.printStackTrace();
                }else{
                    if(objects.size()>0){
                        ParseObject.deleteAllInBackground(objects);
                    }
                }
            }
        });
        ParseQuery<ParseObject> queryDel3=ParseQuery.getQuery("CreditCard");
        queryDel3.whereEqualTo("userId", mainUser.getId());
        queryDel3.findInBackground(new FindCallback<ParseObject>() {
            @Override
            public void done(List<ParseObject> objects, ParseException e) {
                if(e!=null){
                    e.printStackTrace();
                }else{
                    if(objects.size()>0){
                        ParseObject.deleteAllInBackground(objects);
                    }
                }
            }
        });
        ParseQuery<ParseObject>
queryDel4=ParseQuery.getQuery("CryptoAmount");
        queryDel4.whereEqualTo("username",mainUser.getId());
        queryDel4.findInBackground(new FindCallback<ParseObject>() {
            @Override
            public void done(List<ParseObject> objects, ParseException e) {
                if(e!=null){
                    e.printStackTrace();
                }else{
                    if(objects.size()>0){
                        ParseObject.deleteAllInBackground(objects);
                    }
                }
            }
        });
        ParseQuery<ParseObject> queryDel5=ParseQuery.getQuery("History");
        queryDel5.whereEqualTo("userId", mainUser.getId());
        queryDel5.findInBackground(new FindCallback<ParseObject>() {
            @Override
            public void done(List<ParseObject> objects, ParseException e) {
                if(e!=null){
                    e.printStackTrace();
                }else{
                    if(objects.size()>0){
                        ParseObject.deleteAllInBackground(objects);
                    }
                }
            }
        });
        ParseQuery<ParseObject> queryDel6=ParseQuery.getQuery("UserInfo");
        queryDel6.whereEqualTo("username", mainUser.getId());
        queryDel6.findInBackground(new FindCallback<ParseObject>() {
            @Override
            public void done(List<ParseObject> objects, ParseException e) {
                if(e!=null){
                    e.printStackTrace();
                }else{
                    if(objects.size()>0){
                        ParseObject.deleteAllInBackground(objects);
```

```
                    }
                }
            }
        });
        new Handler().postDelayed(new Runnable() {
            @Override
            public void run() {
                user.deleteInBackground(new DeleteCallback() {
                    @Override
                    public void done(com.parse.ParseException e) {
                        if (e == null) {
                            Toast.makeText(getContext(), user.getUsername() +
" удален", Toast.LENGTH_SHORT).show();
                        } else {
                            Toast.makeText(getContext(), user.getUsername() +
"не удален", Toast.LENGTH_SHORT).show();
                        }
                    }
                });
            }
            },1000);
        Intent intent=new Intent(getActivity(),SignIn.class);
        startActivity(intent);
    }
    private void change(String changeItem,String changeColumnName){
        ParseQuery<ParseObject> query = ParseQuery.getQuery("UserInfo");
        query.whereEqualTo("username", SignIn.mainUser.getId());
        query.findInBackground(new FindCallback<ParseObject>() {
            @Override
            public void done(List<ParseObject> objects, ParseException e) {
                if (e != null) {
                    e.printStackTrace();
                } else {
                    if (objects.size() > 0) {
                        for (ParseObject object : objects) {
                            ParseObject user = objects.get(0);
                            user.put(changeColumnName,changeItem );
                            user.saveInBackground(new SaveCallback() {
                                @Override
                                public void done(ParseException e) {
                                    if (e != null) {
                                        Toast.makeText(getContext(),
e.getLocalizedMessage().toString(), Toast.LENGTH_SHORT).show();
                                    }
                                }
                            });
                        }
                    }
                }
            }
        });
    }
    public void selectImage(View view){
        if(ContextCompat.checkSelfPermission(getActivity(),
Manifest.permission.READ_EXTERNAL_STORAGE)!=
PackageManager.PERMISSION_GRANTED){
            ActivityCompat.requestPermissions(getActivity(),new
String[]{Manifest.permission.READ_EXTERNAL_STORAGE},1);
        }
        else{
            Intent intent=new Intent(Intent.ACTION_PICK,
MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
            startActivityForResult(intent,2);
```

```
        }
    }
    public void defineJobSpinner(){
        defaultJobs = new Job[]{new Contractor(),new Doctor(),new
Driver(),new Engineer(),new Entrepreneur(),
                new Farmer(),new Police(),new Soldier(),new Sportsman(),new
Student(),new Teacher(),new Waiter(),new Worker()};
        jobs = new String[]
{"Подрядчик","Врач","Водитель","Инженер","Предприниматель","Фермер","Полицейс
кий","Военный",
                "Спортсмен","Студент","Учитель","Официант","Строитель"};
        jobArrayAdapter = new
ArrayAdapter(getContext(),R.layout.support_simple_spinner_dropdown_item,jobs)
;
        spinner.setAdapter(jobArrayAdapter);
        spinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> adapterView, View view,
int i, long l) {
                job = adapterView.getSelectedItem().toString();
                change( job,"job");
                prof.setText( job);
            }
            @Override
            public void onNothingSelected(AdapterView<?> adapterView) {
            }
        });
    }
    public void logOut(View view){
        ParseUser.logOutInBackground(new LogOutCallback() {
            @Override
            public void done(ParseException e) {
                if(e !=null){

Toast.makeText(getApplicationContext(),e.getLocalizedMessage(),Toast.LENGTH_L
ONG).show();
                }else{
                    Intent intent=new Intent(getApplicationContext(),
SignIn.class);
                    startActivity(intent);
                }
            }
        });
    }
    @Override
    public void onActivityResult(int requestCode, int resultCode, @Nullable
Intent data) {
        if (requestCode == 2 && resultCode == RESULT_OK && data != null) {
            Uri dataUri = data.getData();
            try {
                if (Build.VERSION.SDK_INT >= 28) {
                    ImageDecoder.Source source =
ImageDecoder.createSource(getActivity().getContentResolver(), dataUri);
                    selectedImage = ImageDecoder.decodeBitmap(source);
                    imageView.setImageBitmap(selectedImage);
                } else {
                    selectedImage =
MediaStore.Images.Media.getBitmap(getActivity().getContentResolver(),
dataUri);
                    imageView.setImageBitmap(selectedImage);
                }
                mainUser.setPhoto(selectedImage);
                Bitmap smallerImage = makeSmallerImage(selectedImage, 300);
```

```
                ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream();
                smallerImage.compress(Bitmap.CompressFormat.PNG, 50,
byteArrayOutputStream);
                byte[] bytes = byteArrayOutputStream.toByteArray();
                ParseQuery<ParseObject> query =
ParseQuery.getQuery("UserInfo");
                query.whereEqualTo("username", mainUser.getId());
                query.findInBackground(new FindCallback<ParseObject>() {
                    @Override
                    public void done(List<ParseObject> objects,
ParseException e) {
                        if (e != null) {
                            e.printStackTrace();
                        } else {
                            if (objects.size() > 0) {
                                for (ParseObject object : objects) {
                                    ParseObject userInfo = objects.get(0);
                                        ParseFile file = new
ParseFile("image.png", bytes);
                                        userInfo.put("images", file);
                                        object.saveInBackground(new
SaveCallback() {
                                            @Override
                                            public void done(ParseException
e) {
                                                if (e != null) {

Toast.makeText(getContext(), e.getLocalizedMessage().toString(),
Toast.LENGTH_SHORT).show();
                                                } else {

Toast.makeText(getContext(), "Uploaded", Toast.LENGTH_SHORT).show();
                                                }
                                            }
                                        });
                                }
                            }
                        }
                    }
                });
            }catch (Exception e){

Toast.makeText(getActivity().getApplicationContext(),e.getLocalizedMessage(),
Toast.LENGTH_SHORT).show();
            }
        }
        super.onActivityResult(requestCode, resultCode, data);
    }
    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions, @NonNull int[] grantResults) {
        if(requestCode==1 &&
grantResults[0]==PackageManager.PERMISSION_GRANTED && permissions.length>0){
            Intent intent=new Intent(Intent.ACTION_PICK,
MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
            startActivityForResult(intent,2);
        }
        super.onRequestPermissionsResult(requestCode, permissions,
grantResults);
    }
    private Bitmap makeSmallerImage(Bitmap image, int maximumSize) {
        int width = image.getWidth();
        int height = image.getHeight();
```

```
        float bitmapRatio = (float) width / (float) height;
        if (bitmapRatio > 1) {
            width = maximumSize;
            height = (int) (width / bitmapRatio);
        } else {
            height = maximumSize;
            width = (int) (height * bitmapRatio);
        }
        return Bitmap.createScaledBitmap(image,width,height,true);
    }
}
```

Приложение Г

РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

(рекомендуемое)

Рисунок Г.1 – Результат функционального тестирования «Ввод верных данных»

Рисунок Г.2 – Результат функционального тестирования «Ввод занятого идентификатора»

Рисунок Г.3 – Результат функционального тестирования «Ввод неверного пароля»

Рисунок Г.4 – Результат функционального тестирования «Ввод некорректного номера телефона»

Рисунок Г.5 – Результат функционального тестирования «Ввод некорректного адреса проживания»

Рисунок Г.6 – Результат функционального тестирования «Ввод верных данных»

Рисунок Г.7 – Результат функционального тестирования «Ввод неверного идентификатора»

Рисунок Г.8 – Результат функционального тестирования «Ввод неверного пароля»