**СОДЕРЖАНИЕ**

**ВВЕДЕНИЕ 4**

**1. АНАЛИТИЧЕСКАЯ ЧАСТЬ 13**

**1.1. Постановка задачи 15**

**1.2. Анализ существующих разработок и выбор стратегии автоматизации 17**

**1.3. Организация проектных решений 19**

**2. ПРОЕКТНАЯ ЧАСТЬ 30**

**2.1. Обоснование и выбор языка и среды разработки 35**

**2.2. Функциональная модель 21**

**2.3. Логическая модель данных 27**

**2.4. Реализация программных модулей и интерфейса 32**

**2.5. Тестирование ПО 50**

**2.6. Руководство пользователя 53**

**3. ОЦЕНКА КАЧЕСТВА И НАДЕЖНОСТИ ПРОГРАММНОГО ПРОДУКТА 60**

**ЗАКЛЮЧЕНИЕ. 65**

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

**ПРИЛОЖЕНИЯ 68**

**ВВЕДЕНИЕ**

В настоящее время ЭВМ широко применяется во многих отраслях деятельности человека. Ни одно учреждение не может обойтись в своей работе без применения специальных программ.

Технология – в широком смысле – объем знаний, которые можно  
использовать для производства товаров и услуг из экономических ресурсов.  
Технология – в узком смысле – способ преобразования вещества, энергии,  
информации в процессе изготовления продукции, обработки и переработки  
материалов, сборки готовых изделий, контроля качества управления.

**Актуальность** моего проекта заключается в том, что, основная масса банковских компаний используют информационные системы в процессе своей работы, например, как вычислительная техника способна в разы ускорить процесс обработки информации и получения итога. Однако в наши дни остаются фирмы, главы которых не принимают решение вводить информационные системы в силу их дороговизны или же собственной неосведомлённости в предоставленной сфере.

**Целью дипломного проекта** является информационное мобильное приложение "Колибри" для обеспечения взаимодействия клиентов и банков. Предполагается создать приложение по выбору банковских услуг, в котором клиент компании может легко выбрать и заказать услугу или товар, а менеджер компании сможет легко и просто передать клиента в банк на особых условиях без скрытых услуг и подводных камней Приложение предполагает предварительную регистрацию участников или, если пользователь уже был зарегистрирован, просто авторизацию. Пользователи могут выбрать товар или услугу и оформить заказ. Проект будет легок в обращении, позволит хранить большое количество сведений в одной базе данных, а также обеспечит простой доступ к нему с мобильного устройства. В основу проекта будет положено два принципа: простота и удобство. Мобильное приложение будет работать с любой мобильной операционной системой.

**1. АНАЛИТИЧЕСКАЯ ЧАСТЬ**

**1.1. Постановка задачи**

ИМП “Колибри” предназначено для быстрого и удобного выбора банковских услуг и последующего заказа этих услуг или товаров. Основная задача мобильного приложения “Колибри” это поместить услуги банков в одном месте, а так-же сопроводить клиента в получении услуги быстро и без скрытых условий. В мобильном приложении “Колибри”, помимо выбора и заказа услуг, можно отправлять сообщения в общем чате. Для этого нужно выбрать соответствующий пункт меню и отправить свое сообщение, другой пользователь сможет его прочитать так-же он увидит кто написал в чате и дату этого сообщения . В разделе акции и предложения содержится информация о самых лучших предложениях банков на самых выгодных условиях

. Для защиты приложения используется три метода авторизации такие как Google Authentication, авторизация по номеру телефона, или с использованием почты и пароля

Важной особенностью мобильного приложения “Колибри” является высокий уровень безопасности данных. Все файлы пароли и данные пользователей шифруются, что обеспечивает защиту персональных данных пользователей. Кроме того, приложение позволяет управлять списком предложений и товаров, что обеспечивает гибкую настройку и возможность добавлять большое количество данных в приложение.

ИМП “Колибри” может использоваться как в крупных компаниях, так и в небольших организациях. Оно позволяет значительно ускорить рабочие процессы, улучшить коммуникацию между сотрудниками и увеличить эффективность работы всей команды в целом. Кроме того, приложение позволяет сократить затраты на связь так как по большей части вместо классического обзвона клиентов они сами видят список услуг и звонят уже с конкретными намерениями заказать услугу

ИМП “Колибри” также имеет удобный и интуитивно

понятный интерфейс, который не требует особых навыков для использования. Пользователи могут быстро находить нужные функции . Для удобства работы сотрудников, приложение “Колибри” доступно для установки на различные мобильные устройства, включая смартфоны и планшеты на базе операционных систем Android . Благодаря этому, могут быстро заказать услугу или товар не выходя из дома, а администратор быстро обработает заявку или заказ.

В целом, ИМП “Колибри” представляет собой мощный инструмент для организации коммуникации как между клиентами компании, так и между сотрудниками . Оно обеспечивает высокую безопасность данных, удобный интерфейс, гибкие настройки и поддержку на различных устройствах. Это делает приложение идеальным решением для компаний любого масштаба и направления деятельности, которые стремятся к улучшению эффективности работы своих сотрудников и повышению качества обслуживания клиентов . ИМП “Колибри” создано для всех сотрудников компании. В нем нет рекламы. Оно доступно для скачивания на любой мобильный телефон и требует регистрации или авторизации. Его можно будет установить на несколько телефонов. Приложение позволяет клиентам в любое время суток подобрать товар или услугу, а сотрудник как только освободится быстро обработает заказ. Также приложение дает возможность клиентам делать заказы и позвонить менеджеру для консультации по номеру телефона в пару кликов. Оно обладает всем необходимым функционалом для клиента и сотрудника компании

**1.2. Анализ существующих разработок и выбор стратегии автоматизации**

Прежде чем разрабатывать мобильное приложение, разберем несколько ключевых понятий. Базовым термином, является «интегрированная среда разработки» (IDE) – комплекс программных средств, используемый программистами для разработки программного обеспечения (далее – ПО).

Среда разработки включает в себя:

* текстовый редактор,
* транслятор (компилятор и/или интерпретатор),
* средства автоматизации сборки,
* отладчик.

Средства разработки программного обеспечения – совокупность приемов, методов, методик, а также набор инструментальных программ (компиляторы, прикладные/системные библиотеки и т.д.), используемых разработчиком для создания программного кода программы, отвечающего заданным требованиям. Разработка программ – сложный процесс, основной целью которого является создание, сопровождение программного кода, обеспечивающего необходимый уровень надежности и качества. Для достижения основной цели разработки программ используются средства разработки программного обеспечения. Основные средства, используемые на различных этапах разработки. В зависимости от задач, которые стоят перед разработчиками, разработка программного продукта может представлять собой многогранный процесс, в котором задействовано множество участников и различных сервисов.

Основными этапами, которые представляют наибольший интерес, являются:

* проектирование приложения;
* реализация программного кода;
* тестирование приложения.

Средства проектирования приложений.

На этапе проектирования приложения в зависимости от сложности разрабатываемого программного продукта, напрямую зависящего от

предъявляемых требований, выполняются следующие задачи проектирования:

* анализ предъявляемых требований;
* разработка архитектуры;
* разработка отдельных компонентов приложения;
* разработка макета взаимодействия пользователя для использования программного обеспечения (пользовательского интерфейса). Результатом проектирования является «Эскизный проект» (Software Design Document) или «Технический проект» (Software Architecture Document)

**1.3. Организация проектных решений**

* **Проектные решения по техническому обеспечению**

Техническое обеспечение – совокупность технических средств, компьютерной техники, средств передачи информации, используемых в автоматизированных системах управления и в информационных системах.

Для решения поставленной задачи был выбран компьютер с операционной системой MAC OS Sonoma и установленным программным обеспечением Android Studio для разработки мобильного приложения на JAVA. Операционная система MAC OS является одной из наиболее надежных и безопасных систем, которая обеспечивает стабильную работу и защиту от внешних угроз.

* Процессор: Apple M2
* Материнская плата: Apple motherboard
* Оперативная память: NAND DDR5 16GB
* Видеокарта: Apple M2
* Дисковый накопитель: NAND SSD SAMSUNG 512GB
* Сетевой адаптер – 1000 Мбит.

Выбранный компьютер с операционной системой MAC OS Sonoma и установленным программным обеспечением Android Studio обеспечивает высокую производительность и масштабируемость, что позволяет решать сложные задачи в области написания и создания ПП.

На основании проведенного анализа была выбрана модель компьютера с операционной системой MAC OS и установленным программным обеспечением Android Studio, так как она обладает высокой производительностью, масштабируемостью и надежностью, что является критически важным для решения поставленной задачи.

* **Проектные решения по информационному обеспечению**

Входной информацией в данном мобильном приложении являются:

* Номер телефона пользователя;
* Личная информация пользователя;
* Файлы, почта, заявки пользователя.

Выходной информацией в данном случае являются:

* Заказ пользователя ;
* Отправленные заявки, сообщения.
* **Проектные решения по программному обеспечению**

При проектировании и разработке системы необходимо максимально эффективным образом использовать ранее закупленное программное обеспечение, как серверное, так и для рабочих станций.

Программное обеспечение:

* Android 10 и выше;
* MAC OS sonoma;
* Microsoft Office 2016;
* Антивирусное ПО;
* Android Studio.
* **Проектные решения по математическому обеспечению**

Подсчёт количества товара в корзине

Используется Формула K=T\*S

Где:

K – количество товара добавленного в корзину

S- Кол-во единиц товара

T- Единица товара

Подсчет цены товара

Используется формула

I=P\*K

Где:

К-Кол-во покупаемого товара

P-Цена за единицу товара

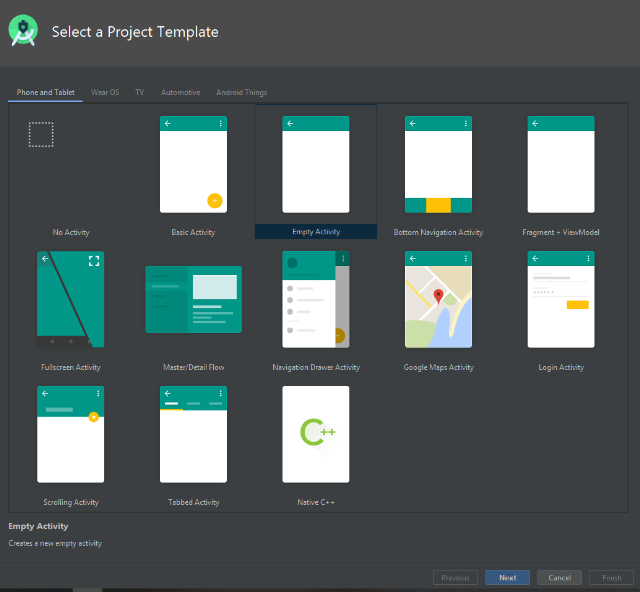
I-Итоговая сумма, которую заплатит покупатель в случае покупки

**2. ПРОЕКТНАЯ ЧАСТЬ**

**2.1. Обоснование и выбор языка и среды разработки**

Существует большое количество универсальных интегрированных сред  
разработки (IDE), которые позволяют разрабатывать программы на разных языках и на одном конкретном специализирующем языке программирования.  
Для дипломную работу выбрана среда разработки Android Studio. Данная программа подходит для разработки мобильного приложения и всех его составляющих. Она состоит из следующих компонентов: Android SDK, графического дизайна, приложения для загрузки компонентов всех версий Android, эмулятора мобильного устройства для запуска приложения.

Важным моментом при выборе Android Studio стало то, что она имеет  
возможность разрабатывать программы практически для всех версий  
операционной системы Android. Существует инструмент для оценки внешнего  
вида программы для различных устройств. Многоцветный код упрощает  
навигацию в больших объемах кода. Программа включает в себя  
Google Cloud Messaging, целью которой является настройка отправлений  
уведомлений для приложений, при котором задействуются облачные сервисы под управлением операционной системы Android, Firebase который мы будем использовать.



Можно создавать многозадачные программы. Одним из  
преимуществ данной среды является детальное тестирование приложения. Данное программное обеспечение призвано предоставить пользователям

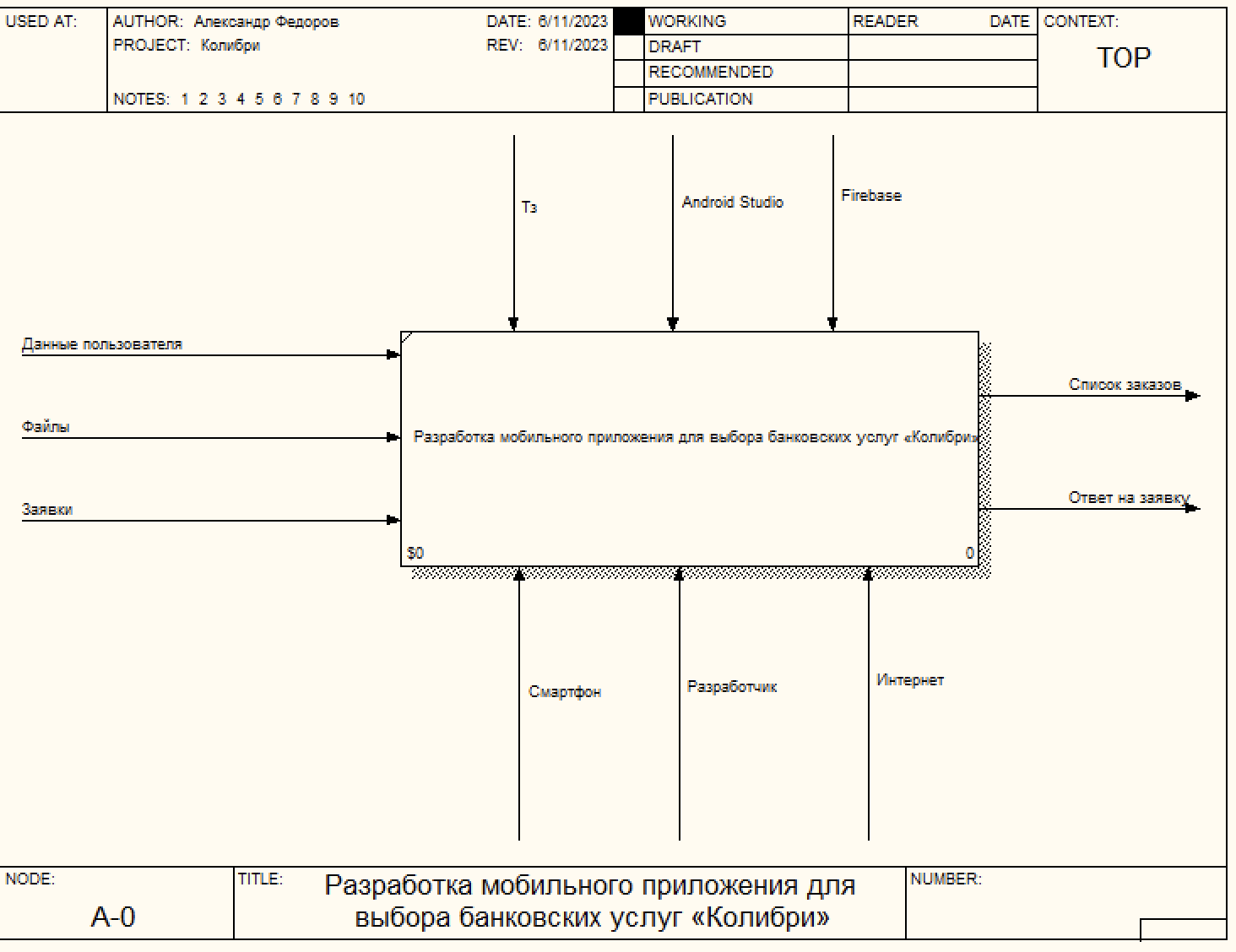
возможность для коммуникации, просмотра ленты актуальных новостей, а также доступа к Google-картам. Разрабатываемое приложение преимущественно носит справочный характер, в связи с этим доступ к основным функциям приложения возможен без аутентификации пользователей. Так как Android Studio предлагает нам разработку только на двух языках программирования Java и Kotlin, выбор пал на Kotlin, так как рекомендации разработчиков начинать изучения лучше всего с Kotlin.

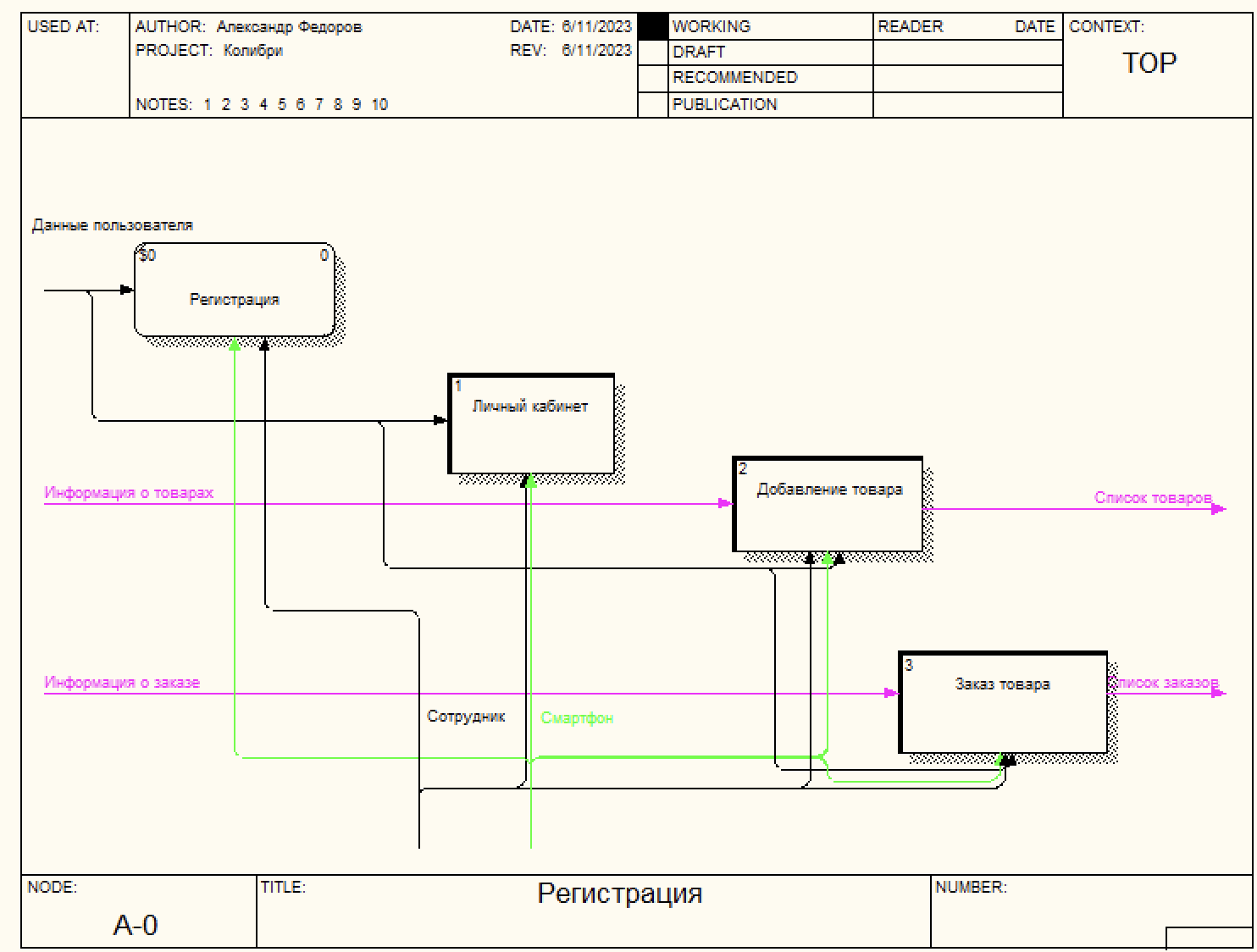
Kotlin является более современным языком программирования, который обладает более удобным синтаксисом и более высокой производительностью по сравнению с Java. Это делает Kotlin идеальным выбором для разработки многозадачных приложений, которые должны работать быстро и эффективно.

Еще одним из преимуществ Kotlin является его возможность взаимодействовать с Java-кодом. Это означает, что разработчики могут использовать существующий Java-код в своих проектах на Kotlin, что упрощает процесс разработки и ускоряет время выпуска приложения на рынок.

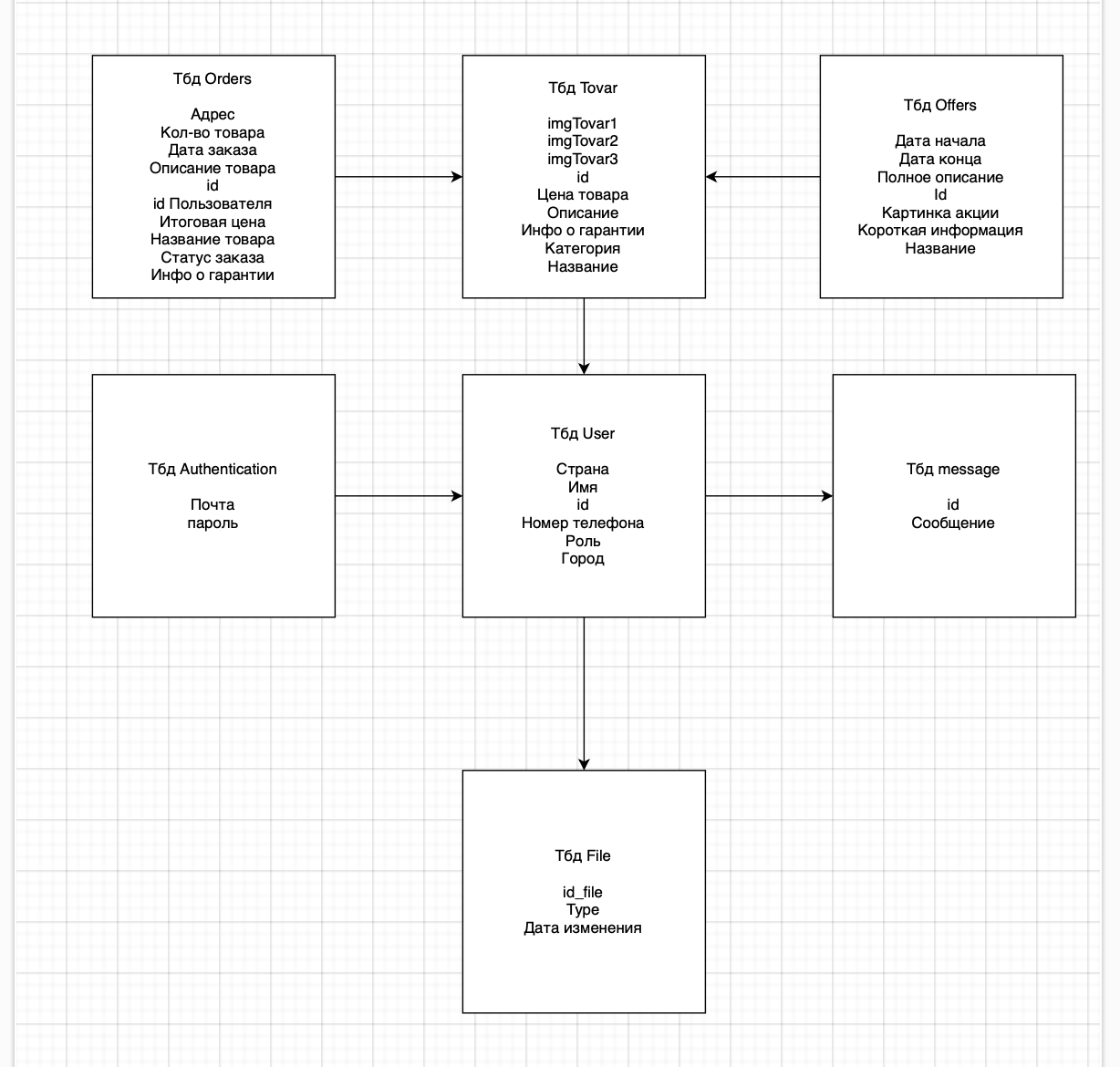
Таким образом, выбор Kotlin для разработки приложения имеет множество преимуществ, таких как удобный синтаксис, высокая производительность, возможность взаимодействия с Java-кодом и т.д. Это делает Kotlin идеальным выбором для разработки многозадачных приложений, таких как приложение, которое мы описываете.

**2.2. Функциональная модель**





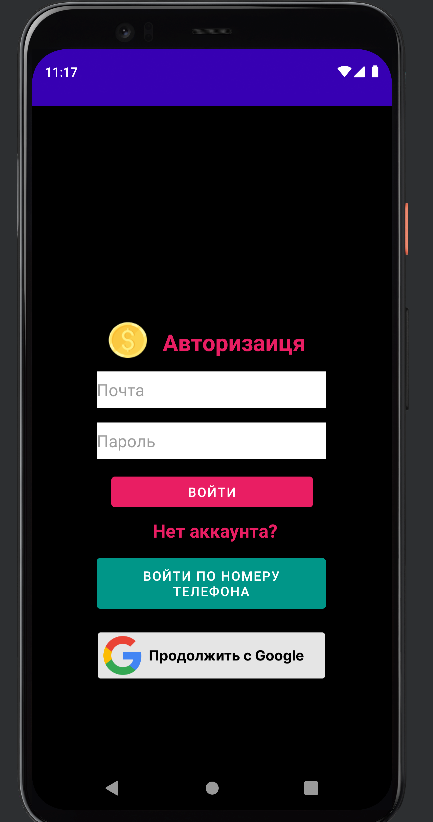
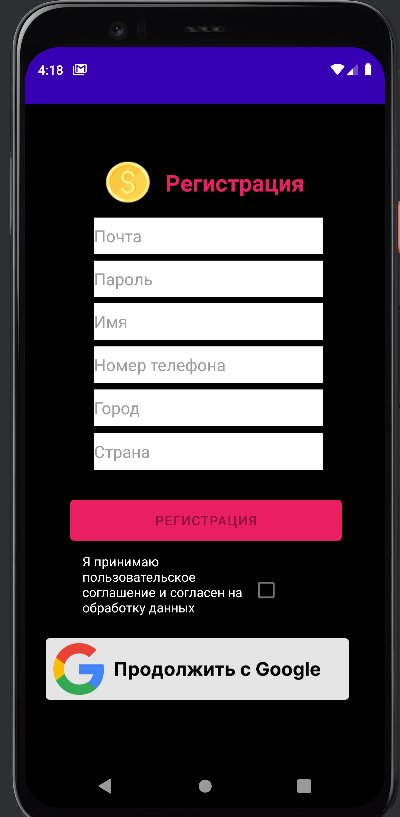
**2.3. Логическая модель данных**



**2.4. Реализация программных модулей и интерфейса**

1-Авторизация\Регистрация

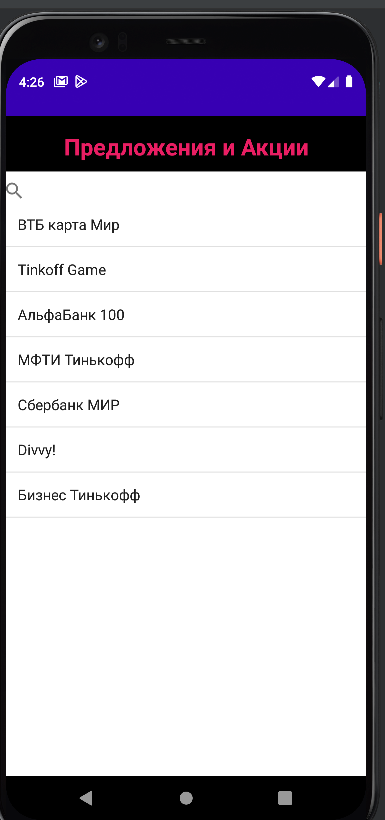
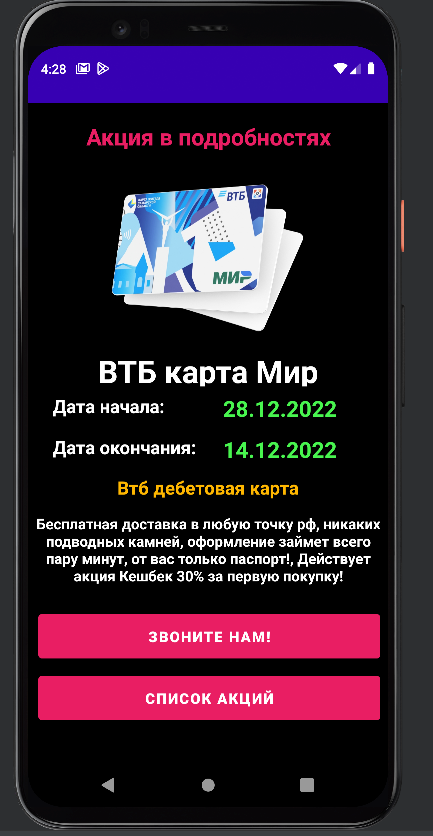
С помощью этого модуля пользователь может зарегистрироваться или авторизоваться

Модуль предложения

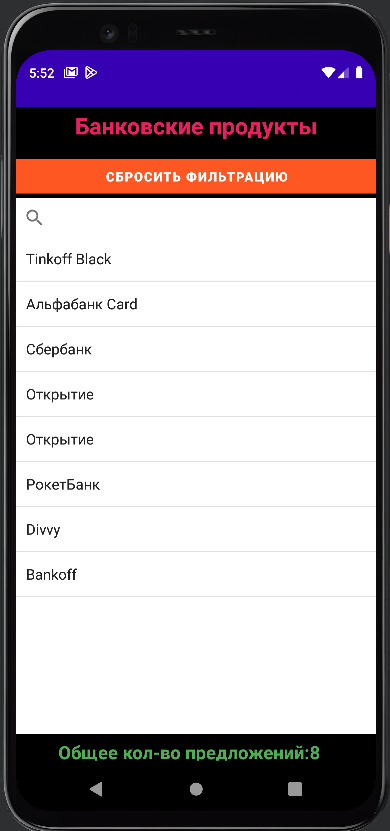
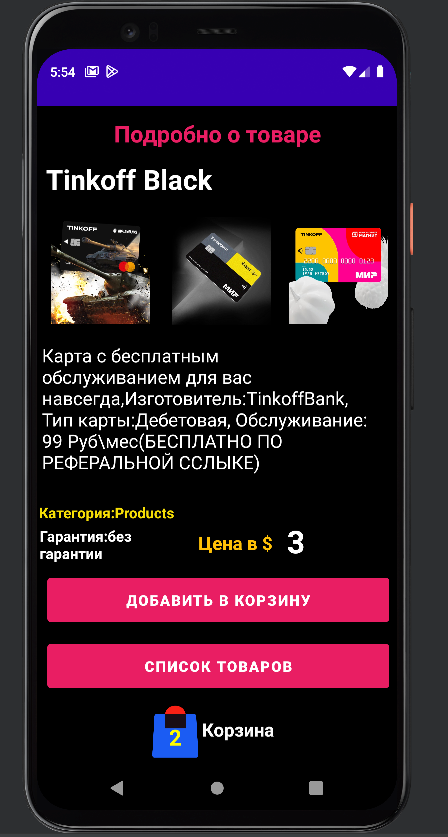
Тут пользователь видит информацию о продуктах компании и банков

При нажатии на предложения появится полная информация о продукте

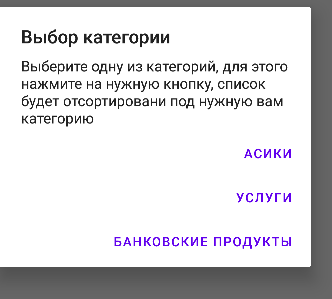
****

**Модуль Товар**

При нажатии на товар появится полная информация о товаре.

**** ****

Так- же есть фильтр по категориям товаров

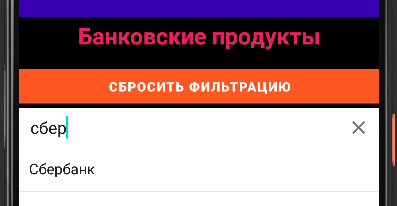
****

Ниже есть отображения общего кол-ва товаров в списке

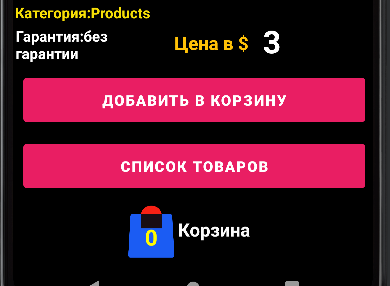
****

****

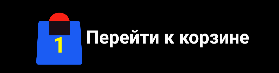
Есть функция поиска товаров

****

Любой товар можно добавить в корзину товаров

****

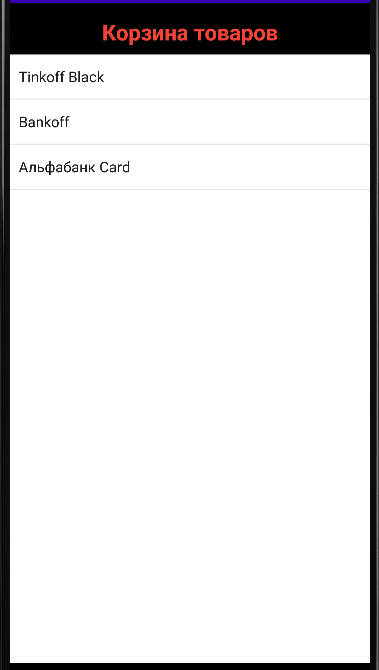
После добавления корзина отображает колличество товара, и теперь можно перейти к корзине

****

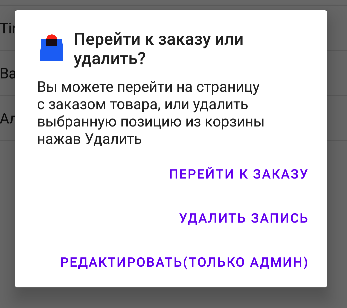
Можно вернуться назад к выбору товара, и добавить в корзину одну, или еще несколько разных позиций

****

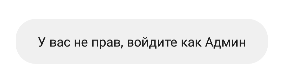
Модуль Заказ\Корзина товаров

****

При нажатии на любую позицию в корзине пользователь может либо удалить запись, либо оформить заказ, администратору доступно редактирование заказов клиента

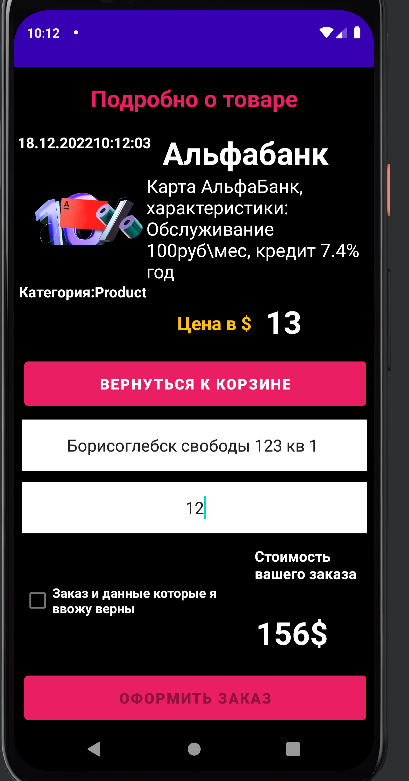
****

При попытке пользователя использовать возможности Админа появится сообщение

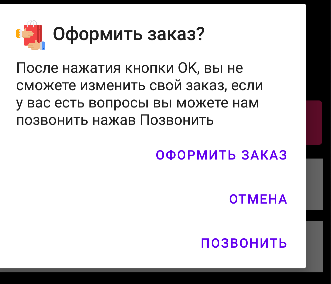
****

При удалении записи выбранная позиция будет удалена

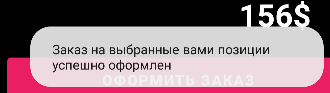
Переход к заказу

****

Пользователь вводит адрес, на который будет доставлен товар, в завсисимости от кол-ва товаров меняется цена заказа, если пользователь ввел верные данные и готов оформить заказ, он долен поставить галочку в Checkbox

****

Клиент может позвонить в компанию нажав на кнопку звонка, верунться к заказу и корзине, после нажатия кнопки заказа информация оданной позиции перейдет в таблицу Orders, а из корзины будет удалена

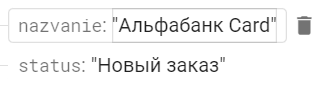
****

После оформления у покупателя появится сообщение

И появится возможность вернутьзя к корзине для заказа других товаров

****

По умолчанию статус заказа новый

****

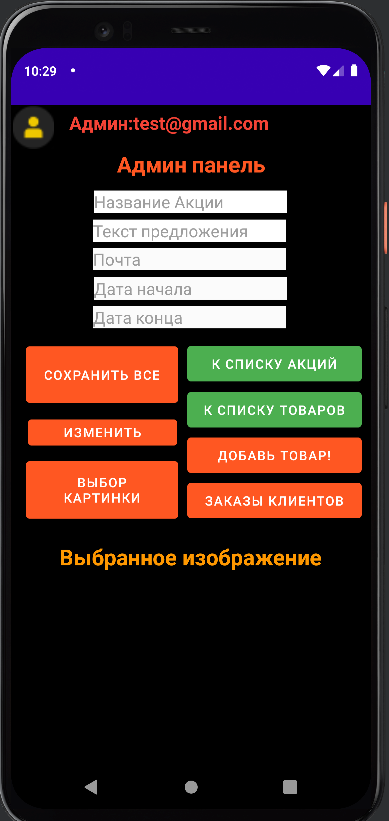
**Модуль панель Администратора**

В приложении есть проверка ролей Зеленый цвеь = пользователь

****

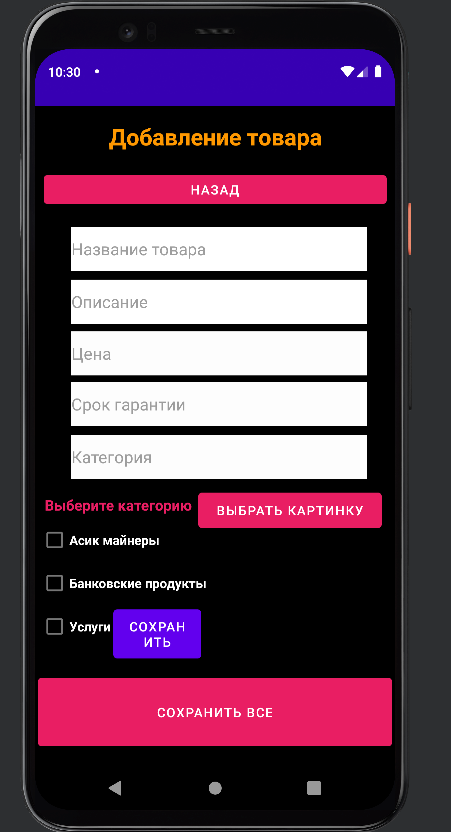
Красный=Администратор

****

****

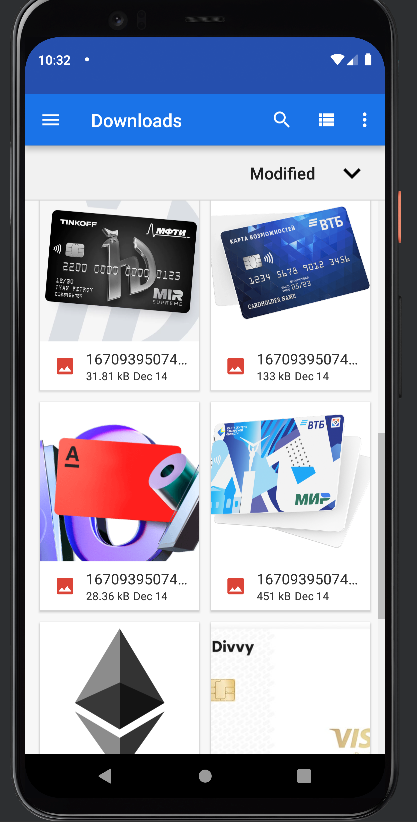
**Основные возможности Админ панели:**

1-Добавление товара

****

Администратор должен заполнить информацию о товаре, далее он может выбрать от 1, до 3х картинок, которые будут загружены в базу данных, загрузить товар с отсутствующим полем или с отсутствующими картинками не выйдет

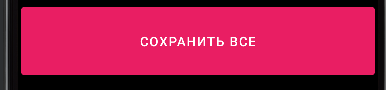
Выбор картинки

****

После заполнения полей и выбора картинки нужно выбрть категорию

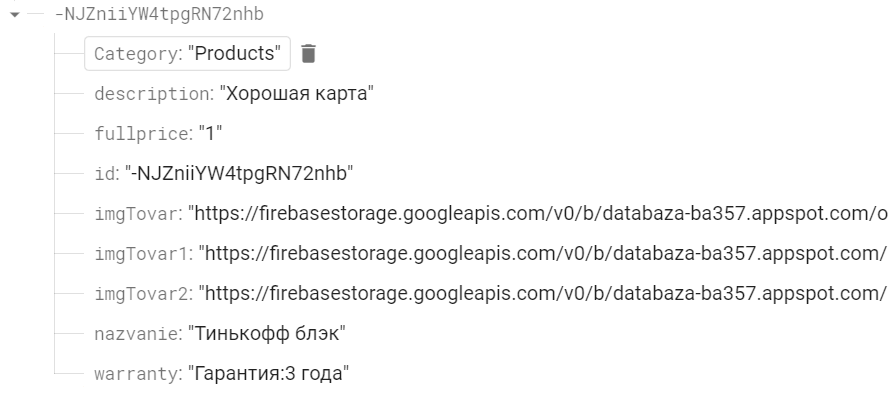
****

Записать сразу в две категории не получится, галочку можно поставить только в одной из них, если администратор считает, что одной картинки хватит, он может нажать сохранить

****

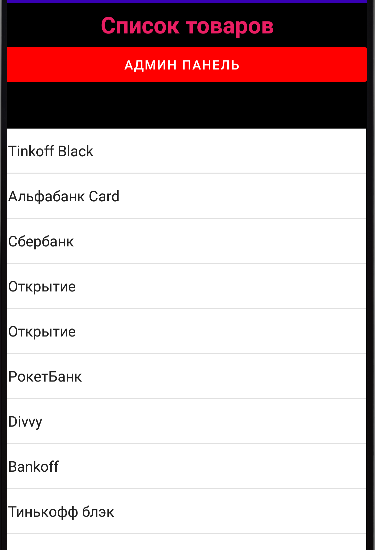
Если требуется три картинки, их можно добавить аналогично первой

Запись успешно добавлена в базу данных

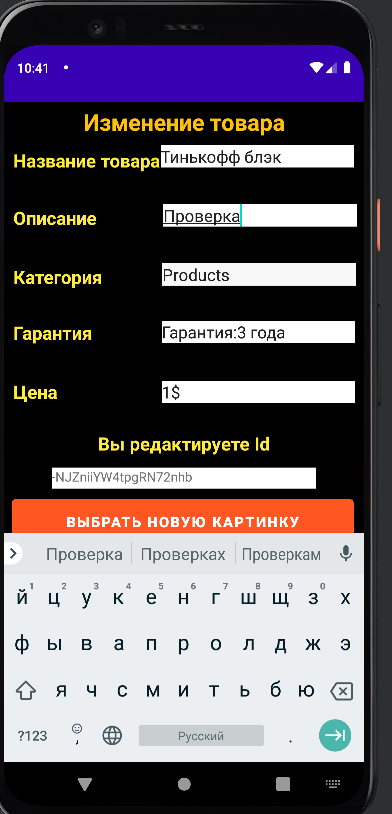
****

**Изменение товара**

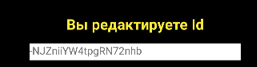
Администартор при нажатии на кнопку изменить перейдет к списку товаров

****

По нажатию на любую позицию администратор может ее просматривать как и клиент, и если нужно отредактировать

****

Показывается Идентификатор записи, которая выбрана в данный момент

****

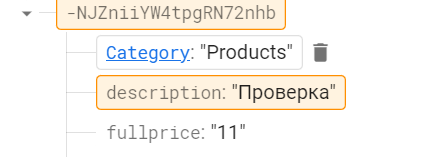
Изменим некоторые данные, выберем другую картинку, заменим некоторые строки

**** вместо Хорошая карта

Сохраним изменения

****

Запись успешно зменена в БД

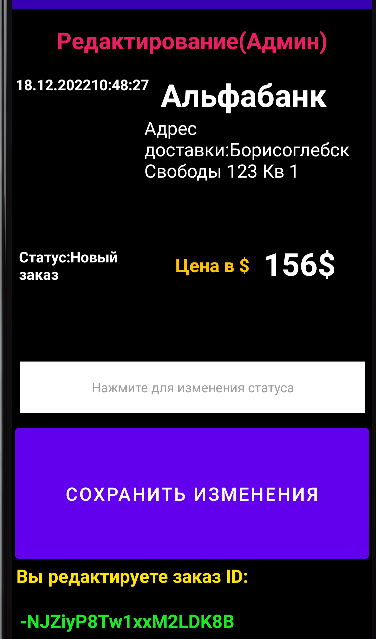
****

Недавно измененные записи подсвечиваются другим цветом

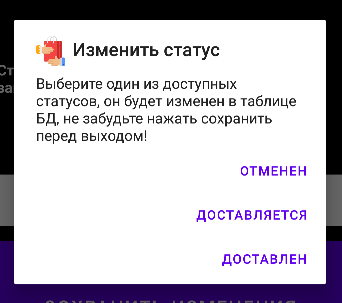
Заказы клиентов

****

При нажатии на позицию в списке Администратор видит Id Заказа который редактирует, Видит что заказ клиент, видит статус, дату\врем

****

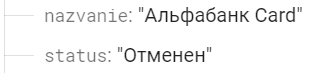
Основная возможность это редактирование статуса

****

После выбора нажимает сохранить

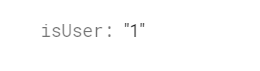
****

В бд изменения сохранены

****

**1-Проверка на администратора(Код)**

По умолчанию все кто регистрируется = Пользователи, администратора можно назначить из БД, заменив isUser на isAdmin

****

Изменение товара имеет Аналогичный код, разница в получении идетнтификатора записи которую нужно изменить(Код)

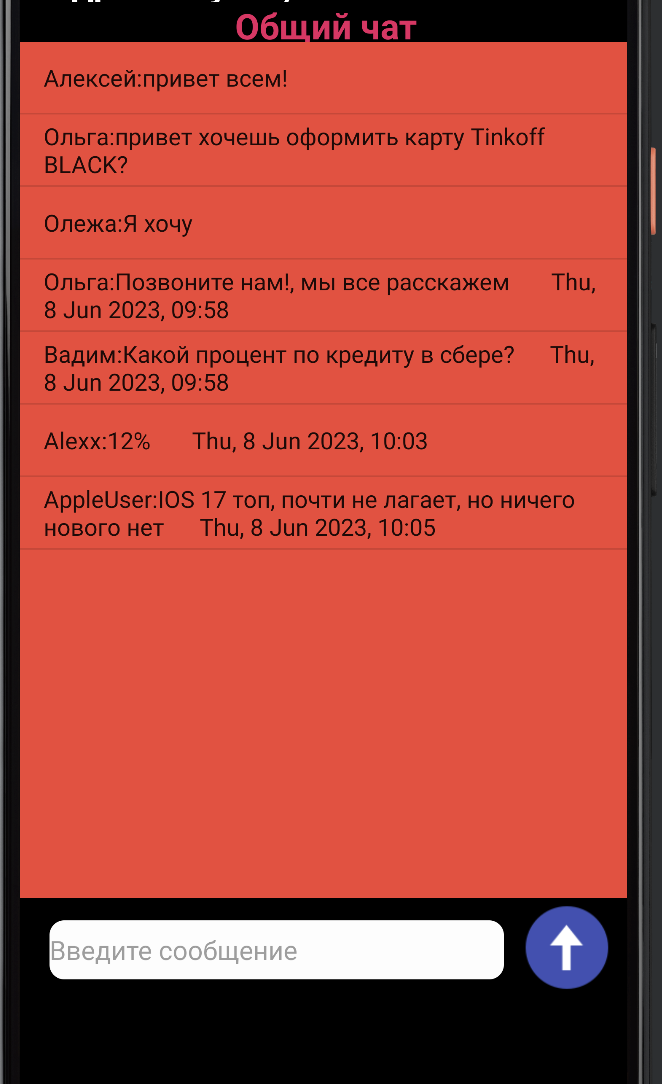
String id = mbase.getKey().Push();

String id = TovarList.getCurrentItem; используется вместо первой строчки

Получение Id записи с товаром из Listview

**Модуль чат**

**Здесь пользователи могут обмениваться сообщениями**

****

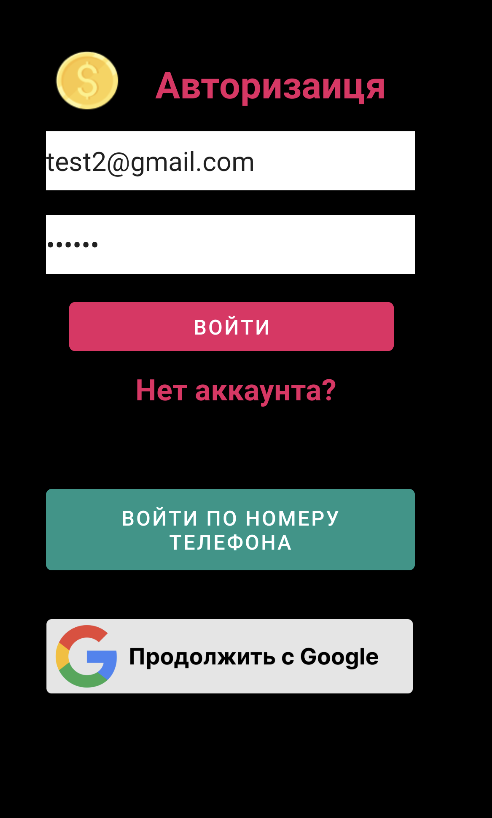
**2.5. Тестирование ПО**

План разработки тест кейсов (тестовых вариантов) предлагается следующий:

1. Анализ требований.
2. Определение набора тестовых данных.
3. Разработка шаблона теста на основании**.**
4. Написание тест кейсов на основании первоначальных требований, тестовых данных и шагов теста.

**Пример:** протестировать функциональность формы авторизации:

|  |  |  |
| --- | --- | --- |
| **Элемент** | **Тип элемента** | **Требования** |
| Текстовое поле login\_input\_email » | EditText | Обязательно для заполнения |
| Текстовое поле «login\_input\_pswd» | EditText | Обязательно для заполнения  Скрывать вводимые данные |
| Кнопка "login\_btn\_next" | button | При заполнении полей выполняется переход |



Программный код в приложении.

Определение набора тестовых требований:

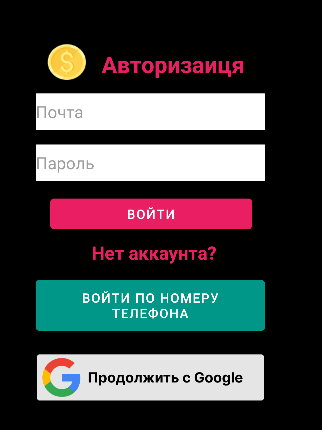
|  |  |
| --- | --- |
| **Действие** | **Результат** |
| 1. Переходим на страницу «Авторизация» | Страница открыта, формы по умолчанию пусты |
| 2. Заполняем текстовые поля:  Логин (register\_input\_code  Пароль (register\_btn\_next) | Текстовые поля для заполнения |
| 3. Проверка кода происходит автоматически | При вводе номера телефона и кода происходит переход на главную страницу. |

Позитивныйтест кейса (все поля **OK**):

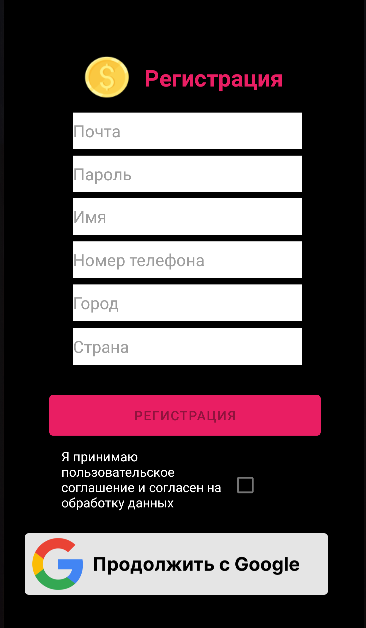
|  |  |
| --- | --- |
| **Действие** | **Результат** |
| 1.Переходим на страницу «Авторизация» | Страница открыта, формы по умолчанию пусты |
| 2. Заполняем текстовые поля:  Текстовое поле «register\_input\_code» | Текстовые поля не заполнены |
| 3.Нажимаем на кнопку «register\_btn\_next» | Появление сообщения – «Введите номер телефона», переход на проверку кода не осуществляется. |

**2.6. Руководство пользователя**

Для начала работы с приложением нужно Авторизоваться

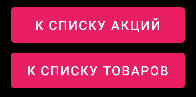


Для этого нужно использовать один из методов, в случае если аккаунта нет, нужно зарегистрироваться



Кнопка регистрация

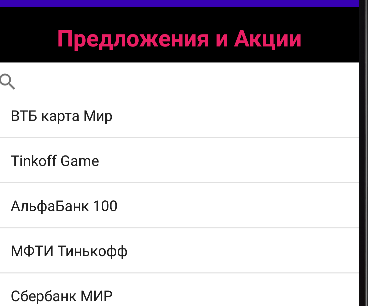
После у пользователя доступно 2 основных функции



Товары\Акции

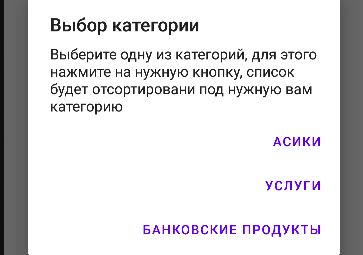
В акциях расположены акции, есть поиск, при нажатии на акцию можно просматривать ее целиком



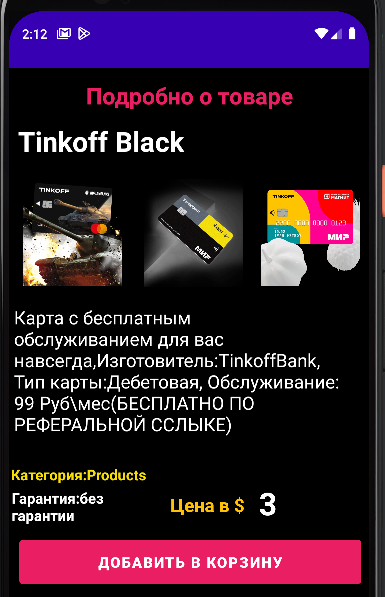


Точно так-же и с товаром

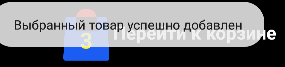
В товарах есть поиск, так-же можно фильтровать товары по категориям



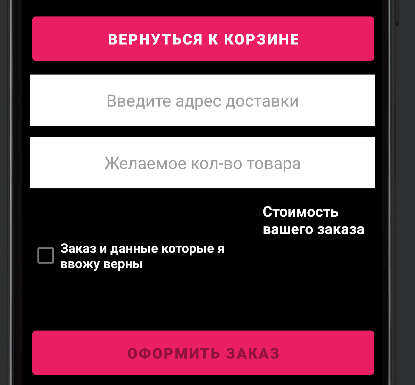
По нажатию на товар полная информация



Далее можно добавить товар в корзину Кол-во товара изменится и появится перейти к корзине



По нажатию на любой товар в корзине можно перейти к заказу

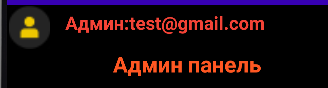


После оформления остается только ждать прихода посылки

* 1. Руководство администратора

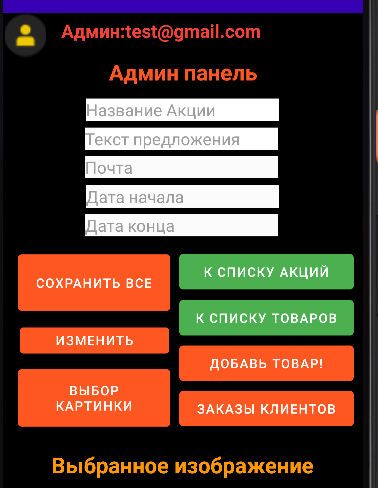
В задачи админа входит добавление\изменение\удаление товара

Для начала как и пользователю нужно авторизоваться, при входе С ролью Админ вы встретите следующие надписи

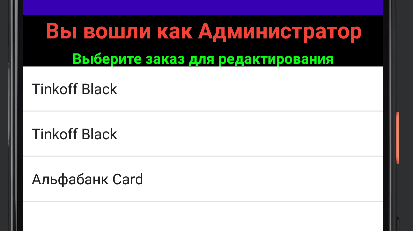


Это значит что вы Админ

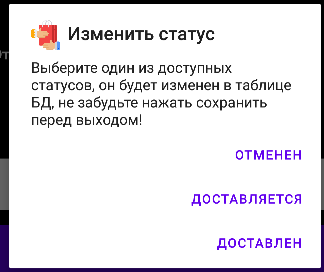
Вид Админ панели



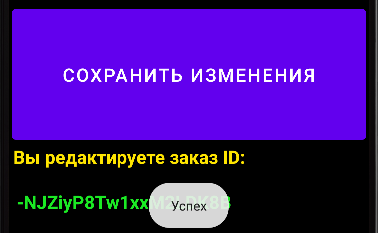
Для редактирования заказа нужно выбрать Заказы клиентов



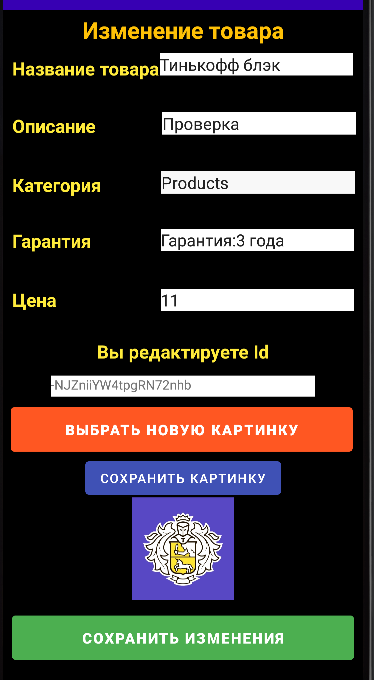
Далее выбрать нужный заказ, статус для него



И нажать кнопку сохранить

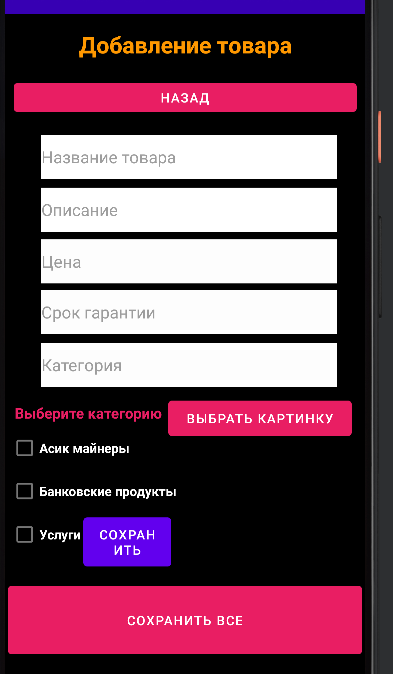


Далее вы можете изменить любой товар



Аналогично статусу жмем изменение в админ панели, выбираем товар, вносим изменение, сохраняем

Добавление товара



Админ может добавить товар, для этого нужно заполнить все поля, выбрать категорию, нажать сохранить категорию, выбрать от 1 до 3х картинок и нажать сохранить все

**3. ОЦЕНКА КАЧЕСТВА И НАДЕЖНОСТИ ПРОГРАММНОГО ПРОДУКТА**

|  |  |
| --- | --- |
| Характеристика | Оценка (соответствует, не соответствует, частично соответствует) |
| **Функциональность** — *Набор атрибутов, характеризующий соответствие функциональных возможностей ПО набору требуемой пользователем функциональности. Детализируется следующими подхарактеристиками:* | |
| Пригодностью для применения | Соответствует |
| Корректностью | Соответствует |
| Способностью к взаимодействию | Соответствует |
| Защищенностью | Соответствует |
| [**Надёжность**](http://www.sbup.com/w/index.php?title=%D0%9D%D0%B0%D0%B4%D1%91%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D1%8C&action=edit&redlink=1) — *Набор атрибутов, относящихся к способности ПО сохранять свой уровень качества функционирования в установленных условиях за определенный период времени. Детализируется следующими подхарактеристикам:* | |
| Уровнем завершенности | Соответствует |
| Устойчивостью к дефектам | Частично соответствует |
| Восстанавливаемостью | Соответствует |
| Доступностью | Соответствует |
| Готовностью | Частично соответствует |
| [**Практичность**](http://www.sbup.com/w/index.php?title=%D0%9F%D1%80%D0%B0%D0%BA%D1%82%D0%B8%D1%87%D0%BD%D0%BE%D1%81%D1%82%D1%8C&action=edit&redlink=1)**(применимость)** — *Набор атрибутов, относящихся к объему работ, требуемых для исполнения и индивидуальной оценки такого исполнения определенным или предполагаемым кругом пользователей. Детализируется следующими подхарактеристиками:* | |
| Понятностью | Соответствует |
| Простотой использования | Соответствует |
| Изучаемостью | Соответствует |
| Привлекательностью | Соответствует |
| [**Эффективность**](http://www.sbup.com/w/index.php?title=%D0%AD%D1%84%D1%84%D0%B5%D0%BA%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D1%81%D1%82%D1%8C&action=edit&redlink=1) — *Набор атрибутов, относящихся к соотношению между уровнем качества функционирования ПО и объемом используемых ресурсов при установленных условиях. Детализируется следующими подхарактеристиками:* | |
| Временной эффективностью | Соответствует |

|  |  |
| --- | --- |
| Используемостью ресурсов | Соответствует |
| [**Сопровождаемость**](http://www.sbup.com/w/index.php?title=%D0%A1%D0%BE%D0%BF%D1%80%D0%BE%D0%B2%D0%BE%D0%B6%D0%B4%D0%B0%D0%B5%D0%BC%D0%BE%D1%81%D1%82%D1%8C&action=edit&redlink=1) — *Набор атрибутов, относящихся к объему работ, требуемых для проведения конкретных изменений (модификаций). Детализируется следующими под характеристиками:* | |
| Удобством для анализа | Соответствует |
| Изменяемостью | Частично соответствует |
| Стабильностью | Частично соответствует |
| Тестируемостью | Соответствует |
| [**Мобильность**](http://www.sbup.com/w/index.php?title=%D0%9C%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C&action=edit&redlink=1) — *Набор атрибутов, относящихся к способности ПО быть перенесенным из одного окружения в другое. Детализируется следующими под характеристиками:* | |
| Адаптируемостью | Соответствует |
| Простотой установки | Соответствует |
| Сосуществованием | Частично соответствует |
| Замещаемостью | Соответствует |

**ЗАКЛЮЧЕНИЕ**

Результатом работы является разработанное мобильное приложение «Колибри». Приложение позволяет эффективно взаимодействовать между собой клиентам и сотрудникам компании, что позволяет оптимизировать работу как отдельного пользователя, так и целой компании. В целом «Колибри» — это эффективное решение для любого предприятия, которое хочет использовать современные мобильные технологии в своей работе.

**В данной дипломной работе были выполнены следующие задачи:** создание базы данных, содержащей информацию о пользователях и товарах, а также разработан интерфейс для взаимодействия пользователя с системой и возможность сделать заказ товара или услуги . Создан и разработан простой и понятный интерфейс где разберется каждый в считаные минуты, приложение позволяет эффективно заказать услугу пользователю, а администратору быстро обработать заказ клиента

Была спроектирована удобная система регистрации которая содержит сразу несколько вариантов таких как GoogleAuthentication, вход по номеру телефона или почте и паролю. В результате было создано мобильное приложение “Колибри”, позволяющая автоматизировать работу небольшой компании.

Основным выводом данной дипломной работы следует считать тот факт, что для повышения эффективности работы предприятия и его подразделений, для начала, нужно разработать эффективную систему для взаимодействия пользователя и сотрудника компании . В результате работы было разработано приложение которое объединяет в себя все выше перечисленные функции Таким образом, цель данной дипломной работы достигнута, все задачи, поставленные в проекте, полностью выполнены.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Льюис, Ш. Нативная разработка мобильных приложений: перекрестный справочник для iOS и Android : справочник / Ш. Льюис, М. Данн ; пер. с англ. А. Н. Киселева. - Москва : ДМК Пресс, 2020. - 376 с. - ISBN 978-5-97060-845-6. - Текст : электронный. - URL:
2. Гриффитс, Д. Head First. Программирование для Android : практическое руководство / Д. Гриффитс, Д. Гриффитс. - 2-е изд. - Санкт-Петербург : Питер, 2018. - 912 с. - (Серия «Head First O’Reilly»). - ISBN 978-5-4461-0708-7. - Текст : электронный. - URL: https://znanium.com/catalog/product/1766374
3. Пирская, Л. В. Разработка мобильных приложений в среде Android Studio : учебное пособие / Л. В. Пирская ; Южный федеральный университет. - Ростов-на-Дону ; Таганрог : Издательство Южного федерального университета, 2019. - 123 с. - ISBN 978-5-9275-3346-6. - Текст : электронный. - URL: https://znanium.com/catalog/product/1894469
4. Разработка приложений под мобильную платформу Android : лабораторный практикум / Д. В. Кравцов, М. А. Лосева, Е. А. Леонов [и др.]. - Москва : ФЛИНТА, 2018. - 72 с. - ISBN 978-5-9765-4014-9. - Текст :электронный. - URL: https://znanium.com/catalog/product/1860040
5. Скин, Д. Kotlin. Программирование для профессионалов : практическое руководство / Д. Скин, Гринхол Дэвид. - Санкт-Петербург : Питер, 2020. - 464 с. - (Серия «Для профессионалов»). - ISBN 978-5-4461-1243-2. - Текст : электронный. - URL: https://znanium.com/catalog/product/1756155
6. Гиматдинов, Р. C., Тенденции развития сферы разработки мобильных приложений / Скиф. 2019. №6 (34). – 272 с. - Текст: электронный. URL: https://cyberleninka.ru/article/n/tendentsii-razvitiya-sfery-razrabotki-android
7. Допира, Р. И., Попова, Н. В. Разработка приложения в среде AndroidStudio / Наука, техника и образование. 2020. №3 (33). – 142 с. - Текст: электронный. URL: https://cyberleninka.ru/article/n/razrabotka-prilozheniya-v-srede-androidstudio

ПРИЛОЖЕНИЯ

**Регистрация\Авторизация**

package com.example.kolibripractice;  
  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.annotation.SuppressLint;  
import android.content.Intent;  
import android.graphics.Color;  
import android.net.Uri;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.Button;  
import android.widget.CheckBox;  
import android.widget.CompoundButton;  
import android.widget.EditText;  
import android.widget.ImageView;  
import android.widget.ListView;  
import android.widget.TextView;  
import android.widget.Toast;  
import android.widget.VideoView;  
  
import com.bumptech.glide.Glide;  
import com.google.android.gms.tasks.OnCompleteListener;  
import

com.google.android.gms.tasks.Task;  
import com.google.firebase.auth.AuthResult;  
import com.google.firebase.auth.FirebaseAuth;  
import com.google.firebase.auth.FirebaseUser;  
import com.google.firebase.firestore.DocumentReference;  
import com.google.firebase.firestore.FirebaseFirestore;  
import com.google.firebase.firestore.Source;  
  
import java.util.HashMap;  
import java.util.Map;  
  
public class ClassicRegisterActivity extends AppCompatActivity {  
private EditText edEmail,edPass,Surname,DateOfBirth,Town,Name;  
private TextView regText,Licenge;  
private ImageView imageViewn1,imageViewn2,imageViewn3,countinue2to3,imageView8,next,mailic,imgKey,imageView6,Nameicon,Dateicon,townIcon,bot;  
private CheckBox Checker;  
private Button Completed,completed2;

@SuppressLint("MissingInflatedId")  
 @Override

protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_classic\_register);  
 edEmail = findViewById(R.id.edEmail);  
 mAuth = FirebaseAuth.getInstance();  
 fstore = FirebaseFirestore.getInstance();  
 edPass = findViewById(R.id.edPass);  
 Licenge = findViewById(R.id.Licenge);  
 regText = findViewById(R.id.regText);  
 imageViewn1 = findViewById(R.id.imageViewn1);  
 imgKey = findViewById(R.id.imgKey);  
 mailic = findViewById(R.id.mailic);  
 countinue2to3 = findViewById(R.id.countinue2to3);  
 imageViewn2 = findViewById(R.id.imageViewn2);  
 imageView6 = findViewById(R.id.

findViewById(R.id.imageViewn3);  
 imageView8 = findViewById(R.id.imageView8);  
 townIcon = findViewById(R.id.townIcon);  
 Dateicon = findViewById(R.id.Dateicon);  
 Nameicon = findViewById(R.id.Nameicon);  
 Checker = findViewById(R.id.Checker);  
 Completed = findViewById(R.id.Completed);  
 completed2 = findViewById(R.id.completed2);  
 next = findViewById(R.id.next);  
 tab1();  
 //3tab  
 Name = findViewById(R.id.Name);  
 Surname = findViewById(R.id.Surname);  
 DateOfBirth = findViewById(R.id.DateOfBirth);  
 Town = findViewById(R.id.Town);  
Completed = findViewById(R.id.Completed);  
 Name.setVisibility(View.GONE);  
 Surname.setVisibility(View.GONE);  
 DateOfBirth.setVisibility(View.GONE);

Checker.setVisibility(View.GONE);  
 Licenge.setVisibility(View.GONE);  
 Completed.setVisibility(View.GONE);  
 completed2.setVisibility(View.GONE);  
 Glide.with(this)  
 .load(R.drawable.giphy)  
 .into(bot);  
bot.setVisibility(View.GONE);  
 fstore = FirebaseFirestore.getInstance();  
 mAuth = FirebaseAuth.getInstance();  
  
  
init();  
init2();  
 Completed.setOnClickListener(new View.OnClickListener()  
 {  
 @Override  
  
 public void onClick(View view) {  
  
 if (edEmail.getText().toString().isEmpty()|| edPass.getText().toString().isEmpty()){  
 Toast.makeText(ClassicRegisterActivity.this,"Поля пустые",Toast.LENGTH\_SHORT).show();  
 }else{

mAuth.createUserWithEmailAndPassword(edEmail.getText().toString(), edPass.getText().toString())  
 .addOnCompleteListener(new OnCompleteListener<AuthResult>() {  
 @Override  
 public void onComplete(@NonNull Task<AuthResult> task) {  
 FirebaseUser users = mAuth.getCurrentUser();  
 DocumentReference df = fstore.collection("Users").document(users.getUid());  
 Map<String,Object> userinfo = new HashMap<>();  
 userinfo.put("Mail",users.getEmail().toString());  
 userinfo.put("FullName",Name.getText().toString());  
 userinfo.put("Surname",Surname.getText().toString());  
 userinfo.put("Date",DateOfBirth.getText().toString());  
 userinfo.put("Town",Town.getText().toString());  
 userinfo.put("Bio","Bio......").toString();  
  
df.set(userinfo);  
 if

intent = new Intent(ClassicRegisterActivity.this, LoginActivity.class);  
 FirebaseUser cUser = mAuth.getCurrentUser();  
 intent.putExtra("UserData","Пользователь:"+cUser.getEmail().toString());  
 startActivity(intent);  
  
 }else{  
 Toast.makeText(ClassicRegisterActivity.this,"Ошибка",Toast.LENGTH\_SHORT).show();  
 }  
 }  
 });  
 }  
  
 }  
 });  
  
  
 }  
 public void init (){  
 Checker.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {  
 @Override  
 public void onCheckedChanged(CompoundButton compoundButton, boolean b) {  
 if (Checker.isChecked()){  
 Button bg = (Button)Completed;

bg.setEnabled(false);  
  
 }  
 }  
 });  
 }  
public void init2(){  
 Button b = (Button)Completed;  
 b.setEnabled(false);  
 mailic.setVisibility(View.GONE);  
 imgKey.setVisibility(View.GONE);  
 Nameicon.setVisibility(View.GONE);  
 townIcon.setVisibility(View.GONE);  
 Dateicon.setVisibility(View.GONE);  
  
  
  
}  
 public void Bold(View view) {  
 edEmail.setHintTextColor(Color.parseColor("#090909"));  
 }  
 public void tab1(){  
 edPass.setVisibility(View.GONE);  
 countinue2to3.setVisibility(View.GONE);

}  
 public void tab2(){  
 imageView8.setVisibility(View.GONE);  
 countinue2to3.setVisibility(View.VISIBLE);  
 edPass.setVisibility(View.VISIBLE);  
 edEmail.setVisibility(View.GONE);  
 regText.setText("Введите свой пароль");  
 imageViewn2.setImageResource(R.drawable.blackline);  
 imageViewn1.setImageResource(R.drawable.whiteline);  
 imageView6.setVisibility(View.GONE);  
 imgKey.setVisibility(View.VISIBLE);  
  
 }  
 public void tab3(){  
 imageViewn3.setImageResource(R.drawable.blackline);  
 imageViewn2.setImageResource(R.drawable.whiteline);  
 imageViewn1.setImageResource(R.drawable.whiteline);  
 regText.setText("Завершение регистрации");

edPass.setVisibility(View.GONE);  
 countinue2to3.setVisibility(View.GONE);  
 next.setVisibility(View.GONE);  
 Completed.setVisibility(View.VISIBLE);  
  
 Name.setVisibility(View.VISIBLE);  
 Surname.setVisibility(View.VISIBLE);  
 DateOfBirth.setVisibility(View.VISIBLE);  
 Town.setVisibility(View.VISIBLE);  
 Checker.setVisibility(View.VISIBLE);  
 Licenge.setVisibility(View.VISIBLE);  
 imgKey.setVisibility(View.GONE);  
 Nameicon.setVisibility(View.VISIBLE);  
 townIcon.setVisibility(View.VISIBLE);  
 Dateicon.setVisibility(View.VISIBLE);  
 imageView6.setVisibility(View.VISIBLE);  
  
 }

view) {  
 tab1();  
 }  
  
 public void twotothree(View view) {  
  
 tab3();  
  
 }  
 public void LastStep(){  
 regText.setText("Верификация");  
 edPass.setVisibility(View.GONE);  
 countinue2to3.setVisibility(View.GONE);  
 next.setVisibility(View.GONE);  
 Completed.setVisibility(View.GONE);  
 edEmail.setVisibility(View.GONE);  
 Name.setVisibility(View.GONE);  
 Surname.setVisibility(View.GONE);  
 DateOfBirth.setVisibility(View.GONE);  
 Town.setVisibility(View.GONE);

Checker.setVisibility(View.GONE);  
 Licenge.setVisibility(View.VISIBLE);  
 imgKey.setVisibility(View.GONE);  
 Nameicon.setVisibility(View.GONE);  
 townIcon.setVisibility(View.GONE);  
 Dateicon.setVisibility(View.GONE);  
 imageView6.setVisibility(View.GONE);  
 bot.setVisibility(View.VISIBLE);  
 completed2.setVisibility(View.VISIBLE);  
 Licenge.setText("Для полной функциональности подтвердите что вы не робот,аккаунтом уже можно пользоваться в ограниченном режиме");  
 }  
  
  
 public void reglic(View view) {  
 Intent intent = new Intent(ClassicRegisterActivity.this,Licenge.class);  
 startActivity(intent);  
 }  
}

**Загрузочный экран**

protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_loading\_screen);  
 Handler handler = new Handler();  
 handler.postDelayed(new Runnable() {  
 @Override  
 public void run() {  
 Intent intent = new Intent(LoadingScreen.this, MainActivity.class);  
 startActivity(intent);  
 finish();  
 }  
 }, 3000);  
 }  
}

**Чат**

@Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_group\_chat);  
 message\_input = findViewById(R.id.message\_input);

messageList = findViewById(R.id.messageList);  
  
 listdata = new ArrayList<>();  
 adapter = new ArrayAdapter<>(this, android.R.layout.simple\_list\_item\_1,listdata);  
 messageList.setAdapter(adapter);  
 mBase = FirebaseDatabase.getInstance().getReference(GROUPKEY);  
 getDataFromDB();  
 Intent i = getIntent();  
 name = i.getStringExtra("Pname");  
 nameup = findViewById(R.id.nameup);  
 nameup.setText("Здравствуйте,"+i.getStringExtra("Pname"));  
  
  
 }  
 private void getDataFromDB(){  
 ValueEventListener vListener = new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (listdata.size()>0)listdata.clear();

for (DataSnapshot ds:snapshot.getChildren()) {  
 MessagesClass msg = ds.getValue(MessagesClass.class);  
 assert msg !=null;  
 listdata.add(msg.message);  
 }  
 adapter.notifyDataSetChanged();  
  
 }  
  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 };  
  
 mBase.addValueEventListener(vListener);  
 }  
  
 private void saveData() {  
 String id = mBase.push().getKey();  
 String message =name+ message\_input.getText().toString();  
  
  
 MessagesClass msgadd = new MessagesClass(id,message);  
 if (!TextUtils.isEmpty(message)) {

if (message != null) mBase.child(id).setValue(msgadd);  
  
  
 } else {  
 Toast.makeText(GroupChat.this, "Возможно некоторые поля пустые!", Toast.LENGTH\_SHORT).show();  
 }  
 }  
  
 public void Send(View view) {  
 saveData();  
 }  
  
}

**Профиль**

private void updateData() {  
  
 updBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 String Old = edName2.getText().toString();  
 String New = edName2h.getText().toString();  
  
 String Old2 = edSurname2.getText().toString();  
 String New2 = edSurnameh.getText().toString();  
  
 String Old3 = edBirth2.getText().toString();  
 String New3 = edBirthh.getText().toString();

String Old5 = edTown2.getText().toString();  
 String New5 = edTownh.getText().toString();  
  
 String Old4 = edPochta.getText().toString();  
 String New4 = edPochtah.getText().toString();  
  
 String Old6 = info.getText().toString();  
 String New6 = edBio.getText().toString();  
  
 updateData(Old,New);  
 updateData2(Old2,New2);  
 updateData3(Old3,New3);  
 updateData4(Old4,New4);  
 updateData5(Old5,New5);  
 updateData6(Old6,New6);  
  
  
 }  
  
 private void updateData(String old, String New) {  
 Map<String,Object>userDetail = new HashMap<>();  
 userDetail.put("FullName",New);  
  
  
 fstore.collection("Users")  
 .whereEqualTo("FullName",old)  
  
 .get().addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {  
 @Override  
 public void onComplete(@NonNull Task<QuerySnapshot> task) {  
 if (task.isSuccessful()&& !task.getResult().isEmpty()){

DocumentSnapshot documentSnapshot = task.getResult().getDocuments().get(0);  
 String DocumentID = documentSnapshot.getId();  
 fstore.collection("Users")  
 .document(DocumentID)  
 .update(userDetail)  
 .addOnSuccessListener(new OnSuccessListener<Void>() {  
 @Override  
 public void onSuccess(Void unused) {  
 Toast.makeText(ProfileActivity.this, "Пацан к успеху шел", Toast.LENGTH\_SHORT).show();  
 Intent intent = new Intent(ProfileActivity.this,ProfileActivity.class);  
 startActivity(intent);  
 }  
 }).addOnFailureListener(new OnFailureListener() {  
 @Override  
 public void onFailure(@NonNull Exception e) {  
 Toast.makeText(ProfileActivity.this, "Error", Toast.LENGTH\_SHORT).show();  
 }  
 });  
 }else{  
  
 }  
 }  
 });  
 }  
 private void updateData2(String Old2, String New2) {

Map<String,Object>userDetail = new HashMap<>();  
 userDetail.put("Surname",New2);  
  
  
 fstore.collection("Users")  
 .whereEqualTo("Surname",Old2)  
  
 .get().addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {  
 @Override  
 public void onComplete(@NonNull Task<QuerySnapshot> task) {  
 if (task.isSuccessful()&& !task.getResult().isEmpty()){  
 DocumentSnapshot documentSnapshot = task.getResult().getDocuments().get(0);  
 String DocumentID = documentSnapshot.getId();  
 fstore.collection("Users")  
 .document(DocumentID)  
 .update(userDetail)  
 .addOnSuccessListener(new OnSuccessListener<Void>() {  
 @Override  
 public void onSuccess(Void unused) {  
 Toast.makeText(ProfileActivity.this, "Пацан к успеху шел", Toast.LENGTH\_SHORT).show();  
 Intent intent = new Intent(ProfileActivity.this,ProfileActivity.class);  
 startActivity(intent);

}  
 }).addOnFailureListener(new OnFailureListener() {  
 @Override  
 public void onFailure(@NonNull Exception e) {  
 Toast.makeText(ProfileActivity.this, "Error", Toast.LENGTH\_SHORT).show();  
 }  
 });  
 }else{  
  
 }  
 }  
 });  
 }  
 private void updateData3(String Old3, String New3) {  
 Map<String,Object>userDetail = new HashMap<>();  
 userDetail.put("Date",New3);  
  
  
 fstore.collection("Users")  
 .whereEqualTo("Date",Old3)  
  
 .get().addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {  
 @Override  
 public void onComplete(@NonNull Task<QuerySnapshot> task) {  
 if (task.isSuccessful()&& !task.getResult().isEmpty()){  
 DocumentSnapshot documentSnapshot = task.getResult().getDocuments().get(0);  
 String DocumentID

= documentSnapshot.getId();  
 fstore.collection("Users")  
 .document(DocumentID)  
 .update(userDetail)  
 .addOnSuccessListener(new OnSuccessListener<Void>() {  
 @Override  
 public void onSuccess(Void unused) {  
 Toast.makeText(ProfileActivity.this, "Пацан к успеху шел", Toast.LENGTH\_SHORT).show();  
 Intent intent = new Intent(ProfileActivity.this,ProfileActivity.class);  
 startActivity(intent);  
 }  
 }).addOnFailureListener(new OnFailureListener() {  
 @Override  
 public void onFailure(@NonNull Exception e) {  
 Toast.makeText(ProfileActivity.this, "Error", Toast.LENGTH\_SHORT).show();  
 }  
 });  
 }else{  
  
 }  
 }  
 });  
 }  
 private void updateData4(String Old4, String New4) {  
 Map<String,Object>userDetail = new HashMap<>();  
 userDetail.put("Mail",New4);

fstore.collection("Users")  
 .whereEqualTo("Mail",Old4)  
  
 .get().addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {  
 @Override  
 public void onComplete(@NonNull Task<QuerySnapshot> task) {  
 if (task.isSuccessful()&& !task.getResult().isEmpty()){  
 DocumentSnapshot documentSnapshot = task.getResult().getDocuments().get(0);  
 String DocumentID = documentSnapshot.getId();  
 fstore.collection("Users")  
 .document(DocumentID)  
 .update(userDetail)  
 .addOnSuccessListener(new OnSuccessListener<Void>() {  
 @Override  
 public void onSuccess(Void unused) {  
 Toast.makeText(ProfileActivity.this, "Пацан к успеху шел", Toast.LENGTH\_SHORT).show();  
 Intent intent = new Intent(ProfileActivity.this,ProfileActivity.class);  
 startActivity(intent);  
 }  
 }).addOnFailureListener(new OnFailureListener() {  
 @Override

public void onFailure(@NonNull Exception e) {  
 Toast.makeText(ProfileActivity.this, "Error", Toast.LENGTH\_SHORT).show();  
 }  
 });  
 }else{  
  
 }  
 }  
 });  
 }

**OTP**

@Override  
protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_send\_otp);  
 final EditText inputMobile = findViewById(R.id.inputMobile);  
 Button buttonGetOtp= findViewById(R.id.btnVerifyс);  
 final ProgressBar progressBar = findViewById(R.id.progressBar);  
 buttonGetOtp.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 if (inputMobile.getText().toString().trim().isEmpty()){  
 Toast.makeText(sendOtpActivity.this,"Введите номер",Toast.LENGTH\_SHORT).show();  
 return;  
 }  
 progressBar.setVisibility(View.VISIBLE);  
 buttonGetOtp.setVisibility(View.VISIBLE);

PhoneAuthProvider.getInstance().verifyPhoneNumber("+7"+inputMobile.getText().toString(),  
 60,  
 TimeUnit.SECONDS,  
 sendOtpActivity.this,  
 new PhoneAuthProvider.OnVerificationStateChangedCallbacks(){  
 @Override  
 public void onVerificationCompleted(@NonNull PhoneAuthCredential phoneAuthCredential) {  
 progressBar.setVisibility(view.GONE);  
 buttonGetOtp.setVisibility(View.VISIBLE);  
 }  
  
 @Override  
 public void onVerificationFailed(@NonNull FirebaseException e) {  
 progressBar.setVisibility(view.GONE);  
 buttonGetOtp.setVisibility(View.VISIBLE);  
 Toast.makeText(sendOtpActivity.this,e.getMessage(),Toast.LENGTH\_SHORT).show();  
 }  
  
 @Override  
 public void onCodeSent(@NonNull String verificationId, @NonNull PhoneAuthProvider.ForceResendingToken forceResendingToken) {  
 progressBar.setVisibility(View.GONE);  
 progressBar.setVisibility(View.VISIBLE);  
 Intent intent = new Intent(getApplicationContext(), VerifyActivity.class);

intent.putExtra("Номер", inputMobile.getText().toString());  
 intent.putExtra("verificationId",verificationId);  
 startActivity(intent);  
 }  
 });  
  
  
 }  
 });  
}  
}

**Заявки**

public void init(){  
  
 listdata = new ArrayList<>();  
 listTemp = new ArrayList<ClassAction2>();  
 adapter = new ArrayAdapter<>(this, android.R.layout.simple\_list\_item\_1,listdata);  
 aclist2.setAdapter(adapter);  
 mBase = FirebaseDatabase.getInstance().getReference(GROUPKEY);  
 getDataFromDB();  
 setOnclickItem();  
  
 }  
  
  
 private void getDataFromDB(){  
 ValueEventListener vListener = new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapsэ

hot) {  
 if (listdata.size()>0)listdata.clear();  
 if (listTemp.size()<0)listTemp.clear();  
 for (DataSnapshot ds:snapshot.getChildren()) {  
 ClassAction2 msg = ds.getValue(ClassAction2.class);  
 assert msg !=null;  
 listdata.add(msg.nazvanie);  
 listTemp.add(msg);  
 }  
 adapter.notifyDataSetChanged();  
  
  
  
  
  
 }  
  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 };  
  
 mBase.addValueEventListener(vListener);  
 }  
 private void setOnclickItem(){  
 aclist2.setOnItemClickListener(new AdapterView.OnItemClickListener() {  
 @Override  
 public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {

ClassAction2 tovarAdd =listTemp.get(i);  
 Intent is = new

Intent(Zayavki.this, FullZayavka.class);  
 is.putExtra("img",tovarAdd.uploadUri);  
 is.putExtra("id",tovarAdd.id);  
 is.putExtra("nazvar",tovarAdd.nazvanie);  
 is.putExtra("pricar",tovarAdd.price);  
 is.putExtra("opisakar",tovarAdd.opisanie);  
 is.putExtra("imyar",tovarAdd.imya);  
 is.putExtra("familyr",tovarAdd.family);  
 is.putExtra("numberr",tovarAdd.number);  
 startActivity(is);  
  
 }  
 });  
 }  
}