

Министерство науки и высшего образования Российской Федерации
Муромский институт (филиал)
Федерального государственного бюджетного образовательного учреждения высшего
образования
«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»

Факультет _____ ИТР _____

Кафедра _____ ПИН _____

КУРСОВАЯ РАБОТА

По Разработка приложений для мобильных операционных систем

Тема Мобильное приложение «Доставка продуктов питания»

Руководитель

Колпаков А.А.

(фамилия, инициалы)

(подпись)

(дата)

Студент ПИН - 121
(группа)

Скотников Е.С.

(фамилия, инициалы)

(подпись)

(дата)

Муром 2024

В данной курсовой работе необходимо было спроектировать мобильное приложение для сервиса доставки продуктов сервиса. В качестве средств разработки приложения была использована среда Android Studio. Язык разработки: Kotlin.

In this course work, it was necessary to design a mobile application for a grocery delivery service. The Android Studio environment was used as the application development tools. Development language: Kotlin

Содержание

Введение.....	6
1.Анализ технического задания.....	7
1.1. Функционал	7
1.2. Операционная система и язык программирования	8
1.3. База данных.....	8
1.4. Система контроля версий.....	9
1.5. Этапы разработки.....	9
1.6. Формирование требований к системе.....	10
1.7. Описание предметной области	11
2.Разработка алгоритмов	12
2.1. Общая структура приложения.....	12
2.2. Логика работы с базой данных	12
2.3. Алгоритмы работы приложения.....	13
2.4. Пример пользовательского интерфейса	16
2.5 . Тестирование АИС.....	18
3.Руководство программиста	21
4.Руководство пользователя	26
4.1. Руководство пользователя.....	26
4.2. Руководство администратора	27
Заключение	28
Список литературы:	29
Приложение	30
Приложение 1. Модели баз данных	30
Приложение 2. Снимки окон программы	32
Приложение 3. Код программы	39

					МИВУ 09.03.04 - 0.016 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата				
Разраб.		Скотников Е.С.			Мобильное приложение «Доставка продуктов питания»	Лит.	Лист	Листов
Провер.		Колпаков А.А					4	41
Реценз.						МИ ВлГУ ПИН-121		
Н. Контр.								
Утверд.								

Введение

В современном мире мобильные технологии занимают центральное место в жизни человека, упрощая выполнение повседневных задач и предоставляя быстрый доступ к необходимым услугам. Одной из таких услуг является доставка продуктов питания, спрос на которую продолжает расти благодаря стремлению людей экономить время и силы.

Мобильные приложения, предназначенные для автоматизации процессов доставки продуктов, являются ключевым инструментом для предприятий, работающих в данной сфере.

Целью данной курсовой работы является разработка мобильного приложения «Доставка продуктов питания», которое предоставит пользователям возможность удобно заказывать продукты с доставкой на дом. Приложение должно обеспечивать следующие функции: регистрация пользователей, просмотр каталога товаров, добавление продуктов в корзину, оформление заказов и просмотр истории покупок.

Актуальность разработки обусловлена стремлением предприятий повысить конкурентоспособность и удовлетворить потребности современных пользователей, которые ценят скорость, удобство и персонализированный подход. Реализация данного проекта позволит не только автоматизировать процесс доставки, но и повысить лояльность клиентов за счет качественного и удобного сервиса.

Данная работа охватывает этапы проектирования, разработки и тестирования мобильного приложения, а также включает описание архитектуры системы и методов обработки данных.

					МИВУ 09.03.04 – 0.016 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		6

1. Анализ технического задания

Для успешной реализации проекта необходимо тщательно рассмотреть все ключевые требования и спецификации, чтобы гарантировать соответствие конечного продукта ожиданиям заказчика и пользователям. Данный проект представляет собой мобильное приложение для управления пользователями, заказами и продуктами питания, разрабатываемое для платформы Android с использованием языка Kotlin. Приложение является автономным и использует локальную базу данных SQLite для хранения данных о пользователях, товарах, корзине и заказах.

1.1. Функционал

- Каталог продуктов

Приложение предоставляет пользователям доступ к каталогу продуктов питания. Пользователь может просматривать список доступных товаров, включая их описание, цену и наличие.

- Корзина и оформление заказа

Пользователи могут добавлять выбранные продукты в корзину. Реализована возможность редактирования содержимого корзины (изменение количества или удаление товаров). После подтверждения заказа пользователь завершает процесс оформления, указывая данные для доставки.

- История заказов

Пользователи могут просматривать историю своих покупок, где отображаются сведения о датах, статусах и содержании заказов.

- Регистрация и авторизация пользователей

Для работы с приложением пользователи должны зарегистрироваться и авторизоваться. Реализована поддержка двух ролей пользователей:

					МИВУ 09.03.04 – 0.016 ПЗ	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		

Клиент — доступ к каталогу, корзине и истории заказов.

Администратор — управление данными в каталоге, контроль заказов.

- Интерфейс и адаптивность

Приложение включает интуитивно понятный интерфейс с адаптацией под устройства с различными размерами экранов, что делает его удобным для использования на широком спектре мобильных устройств.

1.2. Операционная система и язык программирования

- Операционная система Android

Основной платформой разработки выбрана Android, что позволяет охватить широкую пользовательскую аудиторию благодаря популярности платформы.

- Язык программирования Kotlin

Использование Kotlin обеспечивает лаконичный, безопасный и производительный код. Язык предоставляет улучшенные возможности работы с асинхронным кодом, что важно для взаимодействия с сервером и обработки запросов пользователей.

1.3. База данных

Система управления базами используется для хранения данных о пользователях, продуктах, заказах и истории покупок. В проекте используется Room — библиотека хранения данных, предоставляемая Google для разработчиков Android. Она представляет собой часть архитектурных компонентов Android Jetpack. Основное назначение Room — упростить работу с базами данных SQLite в приложениях Android, предоставляя более высокоуровневый и удобный интерфейс для работы с данными.

					МИВУ 09.03.04 – 0.016 ПЗ	Лист
						8
Изм.	Лист	№ докум.	Подпись	Дата		

1.4. Система контроля версий

- Git и GitHub

Система контроля версий Git будет использоваться для управления исходным кодом проекта, а репозиторий на GitHub обеспечит удобство совместной работы и возможность отслеживания изменений.

1.5. Этапы разработки

Процесс разработки будет состоять из нескольких ключевых этапов:

- Проектирование

Создание архитектуры приложения, включая структуру базы данных и клиент-серверное взаимодействие.

- Разработка мобильного приложения

Реализация пользовательского интерфейса для клиентов и администраторов. Внедрение функционала корзины, оформления заказов и просмотра истории.

- Разработка серверной части

Реализация API для обработки данных, отправляемых от мобильного приложения.

- Интеграция

Подключение мобильного приложения к серверу и тестирование обмена данными.

- Тестирование и оптимизация

					МИВУ 09.03.04 – 0.016 ПЗ	Лист
						9
Изм.	Лист	№ докум.	Подпись	Дата		

Проведение функционального тестирования на виртуальных устройствах для проверки работоспособности приложения.

1.6. Формирование требований к системе

Функциональные требования описывают поведение системы и ожидаемый результат при выполнении определенных действий.

- Работа с пользователями

Регистрация пользователей с вводом имени, email, пароля и адреса доставки. Авторизация через email и пароль. Разграничение доступа в зависимости от роли:

Клиент: просмотр каталога, оформление заказа, доступ к истории покупок.

Администратор: управление каталогом товаров, просмотр заказов.

- Работа с каталогом товаров

Просмотр каталога с товарами, включая название, цену, описание и наличие. Возможность поиска товаров по ключевым словам. Фильтрация товаров по категориям или цене.

- Управление корзиной

Добавление товаров в корзину с указанием количества. Удаление товаров из корзины или изменение их количества. Автоматический расчет общей стоимости заказа.

- Оформление заказа

Заполнение формы заказа (адрес доставки, контактные данные). Отправка заказа на сервер и получение подтверждения. Изменение статуса заказа (например, «В обработке», «Доставлено»).

					МИВУ 09.03.04 – 0.016 ПЗ	Лист
						10
Изм.	Лист	№ докум.	Подпись	Дата		

- История покупок

Просмотр списка предыдущих заказов с подробной информацией о содержимом, дате оформления и статусе.

- Административный функционал

Добавление, редактирование и удаление товаров из каталога. Просмотр и управление заказами.

1.7. Описание предметной области

Мобильное приложение «Доставка продуктов питания» создается для и упрощения процессов, связанных с продажей и доставкой продуктов питания. Данная система предназначена как для конечных потребителей (клиентов), так и для административного персонала компании, обеспечивающего работу сервиса. В рамках проекта охватываются ключевые аспекты предметной области, связанные с управлением заказами, товарными запасами и взаимодействием с пользователями.

Разработка мобильного приложения «Доставка продуктов питания» требует реализации всех необходимых функций для обеспечения удобства работы пользователей и автоматизации бизнес-процессов предприятия. Проект сочетает современный технологический стек, интуитивный интерфейс и масштабируемую архитектуру, что делает его актуальным и эффективным инструментом для достижения целей автоматизации.

					МИВУ 09.03.04 – 0.016 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		11

2. Разработка алгоритмов

2.1. Общая структура приложения

Приложение построено на платформе Android с использованием языка Kotlin и архитектуры MVVM (Model-View-ViewModel). Оно использует библиотеку Room для работы с локальной базой данных SQLite и стандартные XML-макеты для построения пользовательского интерфейса.

2.2. Логика работы с базой данных

Структура базы данных. Приложение включает следующие таблицы:

- Users: данные о пользователях (ID, имя, пароль, роль).
- Products: данные о товарах (ID, название, цена, количество).
- CartItems: данные о корзине.
- OrderTable: данные о заказах.
- OrderItem: связь между заказами и товарами.

Пример модели OrderTable:

```
@Entity(
    tableName = "order_table",
    foreignKeys = [
        ForeignKey(
            entity = User::class,
            parentColumns = ["id"],
            childColumns = ["userId"],
            onDelete = ForeignKey.CASCADE
        )
    ]
)
data class OrderTable(
```

					МИВУ 09.03.04 – 0.016 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		12

```

@PrimaryKey(autoGenerate = true) val id: Int = 0,
val userId: Int,
val createdAt: Long = System.currentTimeMillis(),
val city: String,
val street: String,
val home: String,
val totalPrice: Double = 0.00,
var status: OrderStatus = OrderStatus.PENDING
)

```

Пример DAO для работы с заказами:

@Dao

```
interface OrderItemDao {
```

```
    @Insert(onConflict = OnConflictStrategy.REPLACE)
```

```
    suspend fun insert(orderItem: OrderItem)
```

```
    @Query("SELECT * FROM order_item WHERE id = :id")
```

```
    suspend fun getOrderItemById(id: kotlin.Long): OrderItem?
```

```
    @Query("UPDATE order_item SET orderId = :orderId, productId = :productId,
price = :price WHERE id = :id")
```

```
    suspend fun updateOrderItem(id: Int, orderId: Int, productId: Int, price: Int)
```

```
}
```

Все модели базы данных находятся в приложении 1.

2.3. Алгоритмы работы приложения

2.3.1 Регистрация и авторизация

Алгоритм регистрации:

					МИВУ 09.03.04 – 0.016 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		13

Пользователь вводит имя, пароль и номер телефона. Приложение выполняет валидацию данных. После успешной проверки данные сохраняются в базе данных.

Пример валидации:

```
loginButton.setOnClickListener {  
    val phone = phoneEditText.text.toString().trim()  
    val password = passwordEditText.text.toString().trim()  
    // Валидация телефона  
    if (phone.isEmpty()) {  
        phoneEditText.error = "Введите номер телефона"  
        return@setOnClickListener  
    }  
    val phoneRegex = Regex("^((\\+7|8)?\\d{10})$")  
    if (!phone.matches(phoneRegex)) {  
        phoneEditText.error = "Неверный формат телефона. Пример:  
+7XXXXXXXXXXXX или 8XXXXXXXXXXXX"  
        return@setOnClickListener  
    }  
    // Валидация пароля  
    if (password.isEmpty()) {  
        passwordEditText.error = "Введите пароль"  
        return@setOnClickListener  
    }  
    if (password.length < 3) {  
        passwordEditText.error = "Пароль должен содержать минимум 3  
символа"  
        return@setOnClickListener  
    }  
}
```

2.3.2 Работа с товарами

Алгоритм добавления товара:

Пользователь вводит данные о товаре (название, цена, количество).

Приложение вызывает метод DAO для сохранения товара.

Пример:

```
fun addProduct(product: Product) {  
    viewModelScope.launch {  
        productDao.insertProduct(product)  
    }  
}
```

2.3.3 Работа с заказами

Алгоритм оформления заказа:

Пользователь заполняет данные о доставке. Приложение создаёт новую запись в таблице OrderTable. Приложение связывает заказ с товарами в корзине через таблицу OrderItem.

Пример:

```
userCartItems.forEach { cartItem ->  
    val product = database.productDao().getProductById(cartItem.productId)  
    if (product != null) {  
        val productHistory = ProductHistory(  
            name = product.name,  
            descrpt = product.descrpt,  
            price = product.price,  
            quantity = cartItem.cartQuantity,  
            image = product.image  
        )  
    }
```

```

        val productHistoryId =
            database.productHistoryDao().insert(productHistory)

        val orderItem = OrderItem(
            orderId = orderId,
            productId = productHistoryId,
            quantity = cartItem.cartQuantity,
            price = productHistory.price * cartItem.cartQuantity
        )
        database.orderItemDao().insert(orderItem)
    }
}

```

2.4. Пример пользовательского интерфейса

Отображение списка товаров

Список отображается с помощью RecyclerView, где каждый элемент включает название, цену и кнопку "Добавить в корзину".

Пример XML-макета элемента:

```

<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <TextView
        android:id="@+id/textViewProductName"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="2"
        android:text="Название товара" />

```



```

<TextView
    android:id="@+id/textViewProductPrice"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:text="Цена товара" />

```

```

<Button
    android:id="@+id/buttonAddToCart"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Добавить" />

```

</LinearLayout>

Листинг адаптера:

```

class ProductAdapter(
    private val products: List<Product>,
    private val onAddToCart: (Product) -> Unit
) : RecyclerView.Adapter<ProductAdapter.ProductViewHolder>() {
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
        ProductViewHolder {
        val view = LayoutInflater.from(parent.context).inflate(R.layout.item_product,
            parent, false)
        return ProductViewHolder(view)
    }

    override fun onBindViewHolder(holder: ProductViewHolder, position: Int) {
        val product = products[position]
        holder.bind(product)
    }
}

```

					МИВУ 09.03.04 – 0.016 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		17

```

override fun getItemCount() = products.size

inner class ProductViewHolder(itemView: View) :
RecyclerView.ViewHolder(itemView) {

    fun bind(product: Product) {
        itemView.findViewById<TextView>(R.id.textViewProductName).text =
product.name
        itemView.findViewById<TextView>(R.id.textViewProductPrice).text =
"P${product.price}"
        itemView.findViewById<Button>(R.id.buttonAddToCart).setOnClickListener
{
            onAddToCart(product)
        }
    }
}
}

```

2.5. Тестирование АИС

Основная цель проведения тестирования состоит в проверке соответствия реализации системы требуемой функциональности. Во время тестирования каждый тест регистрируется и его результаты сравниваются с ожидаемым. Если ожидаемый результат не совпадает с фактическим, это отмечается в протоколе тестирования. В таблице 1 представлена методика проведения тестирования разработанного программного продукта.

					МИВУ 09.03.04 – 0.016 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		18

Выполненное действие	Полученный результат
Запуск исполняемого файла	Отображение формы авторизации.
Переход по кнопке “Зарегистрироваться”	Отображение формы регистрации.
Заполнение данных	Вывод сообщения “регистрация прошла успешно”
Заполнение существующими данными	Вывод сообщения "Номер телефона уже зарегистрирован ”
Авторизация с ролью «ROLE_ADMIN»	Отображение интерфейса администратора.
Авторизация с ролью «ROLE_USER»	Отображение интерфейса пользователя.
Нажатие боковой кнопки	Открытие панели со всеми страницами взаимодействия
Переход по вкладке “Список товаров”	Открытие списка всех товаров, их цена и количество на складе
Нажатие на кнопку «Добавить в корзину»	Товар перенесется к вам в корзину с указанным рядом количеством(при условии, что оно не превышает общее)
Нажатие на кнопку назад	Возвращение на стартовую страницу
Переход по вкладке “Корзина”	Открытие списка добавленных вами товаров и их количество

Таблица 1 – методика тестирования разработанной программы

Результаты, полученные в ходе тестирования разработанного программного продукта, позволяют сделать заключение в том, что разработанная программа соответствует требованиям технического задания.

					МИВУ 09.03.04 – 0.016 ПЗ	Лист
						20
Изм.	Лист	№ докум.	Подпись	Дата		

3. Руководство программиста

Приложение включает в себя работу с базой данных, аутентификацию и регистрацию пользователей, а также функциональность для заказа товаров

Фрагменты

Фрагменты управляют различными экранами и функциями приложения. Вот основные:

- AddProductFragment

Позволяет администратору добавлять новый продукт. Включает поля ввода данных продукта (название, цена, количество, описание).

Методы:

onCreateView: Создаёт и возвращает привязанное представление фрагмента. Инициализирует ViewModel с использованием AddProductViewModelFactory. Устанавливает привязку ViewModel и жизненного цикла для binding.

onViewCreated: Выполняется после создания представления. Устанавливает обработчики событий для кнопок выбора изображения и сохранения продукта.

onDestroyView: Очищает ссылку на binding, чтобы избежать утечек памяти.

- CartFragment

Управляет отображением корзины пользователя. Показывает список товаров, добавленных в корзину, их количество, и позволяет их удалить.

Методы:

onCreateView: Создаёт и возвращает привязанное представление (View) для фрагмента корзины. Использует FragmentCartBinding для удобного доступа к элементам пользовательского интерфейса.

onViewCreated: Выполняется после создания представления. Инициализирует DAO для работы с базой данных. Настраивает RecyclerView

					МИВУ 09.03.04 – 0.016 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		21

для отображения товаров в корзине. Загружает товары в корзину для текущего пользователя. Добавляет обработчик для кнопки "Оформить заказ".

loadCartItems: Загружает данные корзины из базы данных. Если корзина пуста, отображает сообщение пользователю. Если данные успешно загружены, обновляет список товаров в памяти и вызывает обновление UI.

updateTotalPrice: Рассчитывает общую стоимость товаров в корзине для текущего пользователя. Обновляет отображение общей стоимости в интерфейсе.

updateRecyclerView: Генерирует данные для отображения корзины, связывая товары с дополнительной информацией о них. Настраивает адаптер **CartAdapter** для отображения списка товаров.

removeItemFromCart: Удаляет элемент из корзины. Удаляет элемент из базы данных. Увеличивает количество товара в таблице продуктов. Обновляет UI, пересчитывает общую стоимость и обновляет список товаров.

refreshFragment: Перезагружает текущий фрагмент, чтобы обновить его состояние (например, после удаления товара).

handleCheckout: Переход на экран оформления заказа (например, к экрану ввода адреса).

onDestroyView: Освобождает ресурсы **binding**, чтобы избежать утечек памяти.

- **FinishedOrdersFragment**

Отображает завершённые заказы пользователя.

Методы:

onCreateView создаёт и возвращает пользовательский интерфейс фрагмента с помощью привязки **FragmentOrdersBinding**.

onViewCreated инициализирует доступ к базе данных через **AppDatabase**, настраивает DAO для работы с таблицами заказов и их элементов, а также настраивает **RecyclerView** с использованием адаптера **OrderAdapter**. Этот адаптер отвечает за отображение заказов и обработку нажатий на элементы

списка, что позволяет перейти на экран с деталями выбранного заказа с использованием метода `findNavController`.

`loadOrders` отвечает за загрузку данных заказов текущего пользователя из базы данных, фильтруя их по указанным статусам, таким как `CANCELLED` или `DELIVERED`. Если список заказов пуст, `RecyclerView` скрывается, а пользователю выводится сообщение. Если заказы найдены, список обновляется, и они отображаются на экране.

Все операции с базой данных выполняются в фоновом потоке с использованием `CoroutineScope`. Если происходит ошибка при загрузке данных, выводится соответствующее сообщение. Этот класс обеспечивает основной функционал для отображения завершённых заказов и навигации к их деталям.

- `OrderSummaryFragment`

Управляет отображением всех заказов для администратора или пользователя. Позволяет администраторам видеть полный список заказов и менять статус заказа.

- `ProductListFragment`

Показывает список доступных продуктов для пользователей. Позволяет добавлять товары в корзину.

- `AddressInputFragment`

Ввод данных для доставки (город, улица, дом). Используется для создания нового заказа.

- `OrderDetailsFragment`

Показывает детали конкретного заказа.

Адаптеры

Адаптеры используются для отображения списков элементов в `RecyclerView`. Вот ключевые адаптеры:

- `CartAdapter`

Используется для отображения товаров в корзине. Отображает количество, название и цену товара.

					МИВУ 09.03.04 – 0.016 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		23

- OrderAdapter

Управляет отображением списка заказов.

- ProductAdapter

Для отображения списка продуктов.

- AdminProductAdapter

Отображает список продуктов с возможностью редактирования количества.

Активности

Активности предоставляют основную структуру для различных частей приложения. Ключевые активности:

- LoginActivity

Экран авторизации пользователя. Проверяет данные, введенные пользователем, и перенаправляет на главный экран.

- RegisterActivity

Регистрация нового пользователя. Включает ввод номера телефона, пароля и других данных.

- MainActivity

Основная активность приложения, управляющая боковым меню и навигацией между фрагментами. Настраивает отображение элементов навигации в зависимости от роли пользователя (администратор/пользователь).

Основные функции

- Работа с базой данных:

Фрагменты и активности используют DAO для выполнения операций с базой данных. Основные таблицы: User, Product, CartItem, OrderTable, OrderItem. Основной класс: AppDatabase().

AppDatabase(): управляет созданием, доступом и миграцией базы данных приложения, предоставляя DAO (Data Access Object) для взаимодействия с таблицами.

- Навигация:

Реализована через NavigationComponent. Действия между фрагментами и передачей данных (например, ID заказа или продукта).

- UI и взаимодействие:

Используются RecyclerView и адаптеры для отображения списков. Большинство фрагментов включают функциональность для добавления, редактирования или удаления данных.

					МИВУ 09.03.04 – 0.016 ПЗ	Лист
						25
Изм.	Лист	№ докум.	Подпись	Дата		

4. Руководство пользователя

4.1. Руководство пользователя

Руководство пользователя для мобильного приложения. Это руководство поможет вам освоиться с основными функциями и страницами Android-приложения. В нем описаны ключевые действия, которые доступны пользователю.

Рекомендуемые системные требования:

Операционная система Android: 8.1;

Процессор с частотой 1 ГГц;

4 ГБ оперативной памяти;

Не менее 50 МБ места на жёстком диске;

При запуске программы пользователь попадает на страницу авторизации (Приложение 2 рисунок 1), не имея аккаунта можно нажать на кнопку “Зарегистрироваться”, перейдя на страницу регистрации(Приложение 2 рисунок 2), после успешной регистрации вы вернетесь на страницу для авторизации и сможете войти в приложение, вас встретит пустая стартовая страница и боковая карусель, нажав на нее перед вами предстанет весь пользовательский интерфейс (Приложение 2 рисунок 3). Первым делом переходим на страницу с товаром (Приложение 2 рисунок 4). Выбираем товар и его количество нажимаем “В корзину”. После вернувшись к стартовой карусели переходим на страницу “Корзина” (Приложение 2 рисунок 5). Здесь изучив состав своей корзины мы можем перейти к составлению заказа, по нажатию кнопки вы попадете на страницу с вводом адреса(Приложение 2 рисунок 6), после успешного ввода адреса вы можете вернуться на стартовое меню, ваша корзина будет очищена. Из неизученных вкладок в карусели остаются “Заказы” и “История заказов”. На странице “Заказы” (Приложение 2 рисунок 7) вы увидите свои заказы все еще находящиеся в работе, по нажатию на один из них вы увидите подробную информацию о заказе (Приложение 2

					МИВУ 09.03.04 – 0.016 ПЗ	Лист
						26
Изм.	Лист	№ докум.	Подпись	Дата		

рисунок 8). Страница с историей заказов в свою очередь (Приложение 2 рисунок 9) содержит в себе либо отмененные заказа, либо доставленные. На ней так же можно посмотреть подробную информацию по отработанном заказе

4.2. Руководство администратора

Получив номер телефона(логин) и пароль с ролью администратора, пользователь получает ряд возможностей. Интерфейс все так же исполнен с использованием боковой карулеси (Приложение 2 рисунок 10). Администратор может добавить новый товар перейдя по вкладке “Добавление товаров” (Приложение 2 рисунок 11), поменять статус заказов (Приложение 2 рисунок 12), назначить новых администраторов(Приложение 2 рисунок 13) и отредактировать количество товара на складе (Приложение 2 рисунок 14)

					МИВУ 09.03.04 – 0.016 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		27

Заключение

В ходе выполнения курсовой работы была разработана автоматизированная информационная система (АИС) для управления процессами магазина, включающая функциональность для авторизации, регистрации пользователей, работы с товарами, корзиной, заказами и отчетностью. Реализованная система предоставляет удобный интерфейс как для пользователей, так и для администраторов.

Пользователи имеют возможность зарегистрироваться, просматривать доступные товары, добавлять их в корзину, оформлять заказы и отслеживать историю своих покупок. Администраторская панель позволяет управлять пользователями, товарами и заказами, включая редактирование и удаление данных.

В процессе разработки были успешно применены современные технологии и инструменты программирования, такие как Android SDK, Jetpack Navigation, Room для работы с базой данных, а также архитектурные паттерны MVVM и RecyclerView для отображения данных.

Созданная система отличается высокой степенью автоматизации и интерактивности, что способствует повышению эффективности управления данными и улучшению пользовательского опыта.

Таким образом, цель курсовой работы была достигнута: разработана полноценная информационная система, которая решает поставленные задачи

					МИВУ 09.03.04 – 0.016 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		28

Список литературы:

1. Аллен Г. Android. Программирование приложений для смартфонов и планшетов — СПб.: Питер, 2019. — 480 с.
2. Бурд Б. Android. Разработка приложений для чайников — СПб.: Питер, 2017. — 416 с.
3. Гриффитс Дэвид, Гриффитс Дон Head First. Программирование для Android. 2-е изд. — СПб.: Питер, 2018. — 912 с.
4. Дейтел П., Дейтел Х., Уолд А. Android для разработчиков. 3-е изд. — СПб.: Питер, 2016.
5. Колисниченко Д.Н. Программирование для Android 5. Самоучитель. — СПб.: БХВ-Петербург, 2015. — 303 с.

					МИВУ 09.03.04 – 0.016 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		29

Приложение

Приложение 1. Модели баз данных

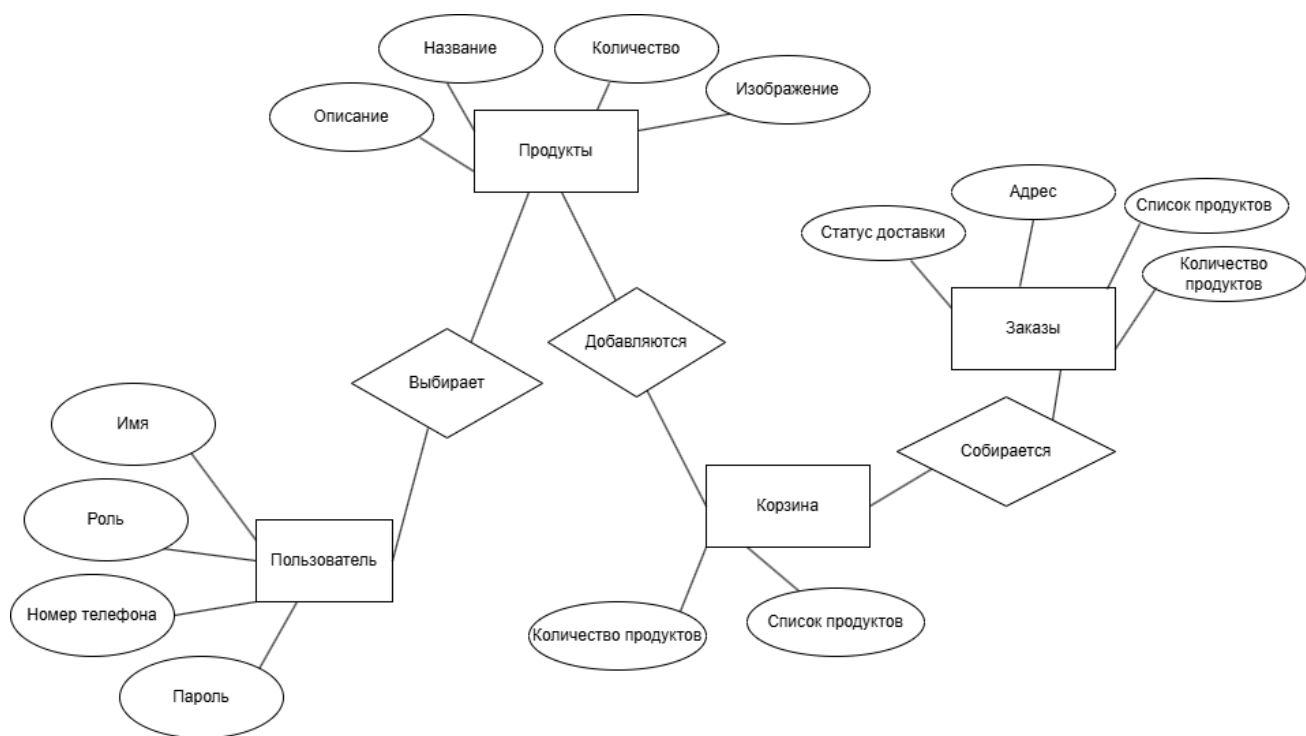


Рисунок 1 - Концептуальная модель данных

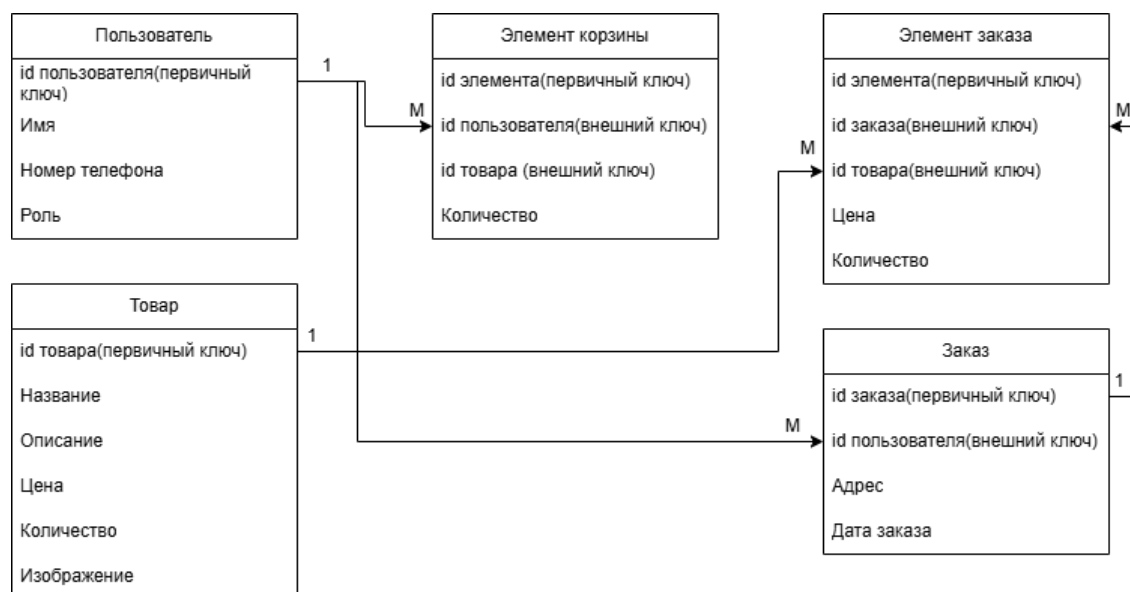


Рисунок 2 – Логическая модель данных

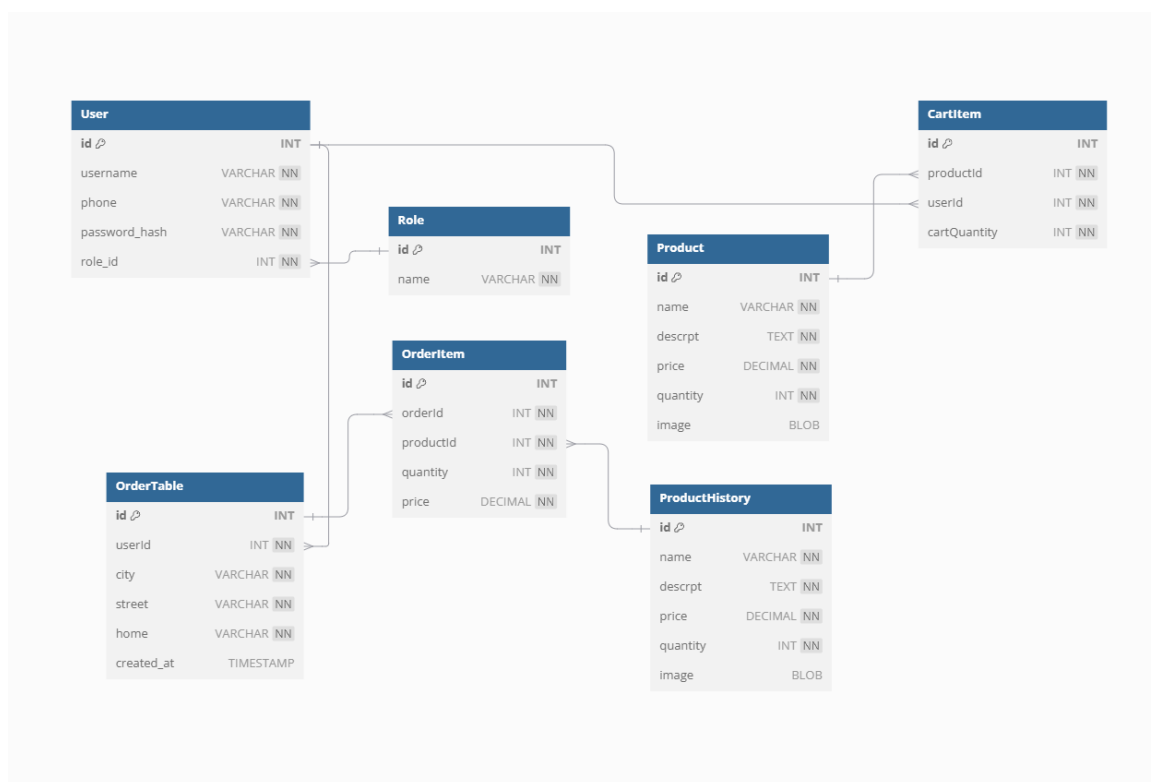


Рисунок 3 – Физическая модель данных

Приложение 2. Снимки окон программы

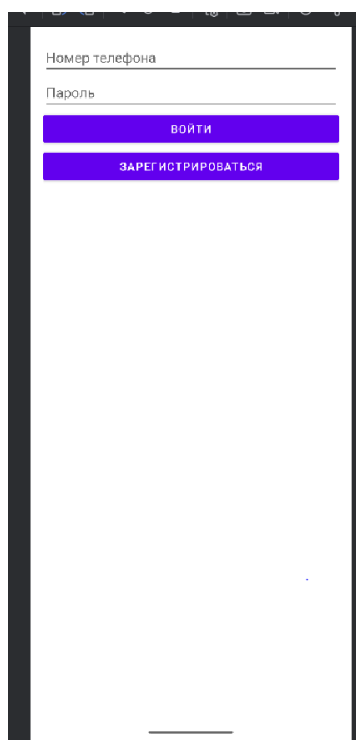


Рисунок 1 – Стартовое окно(окно входа)

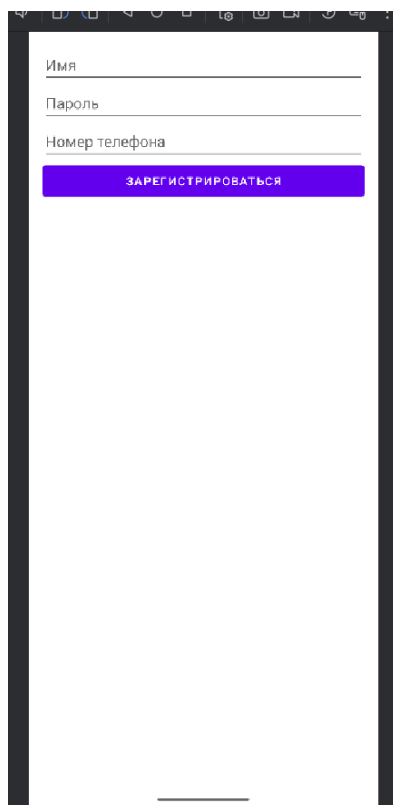


Рисунок 2 – Окно регистрации

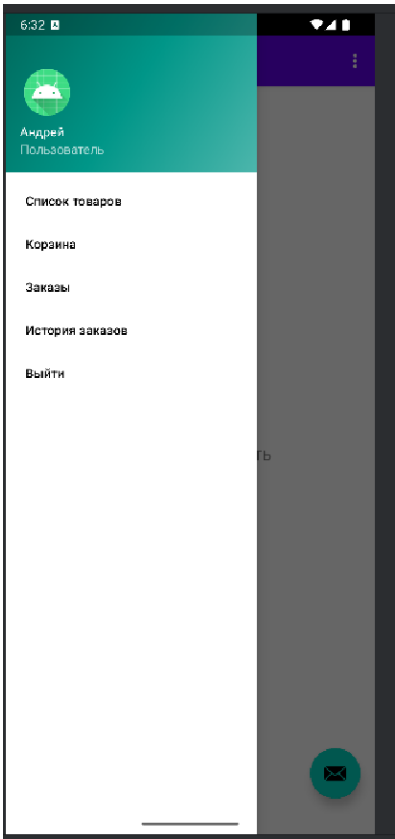


Рисунок 3 – Стартовая страница и боковая карусель

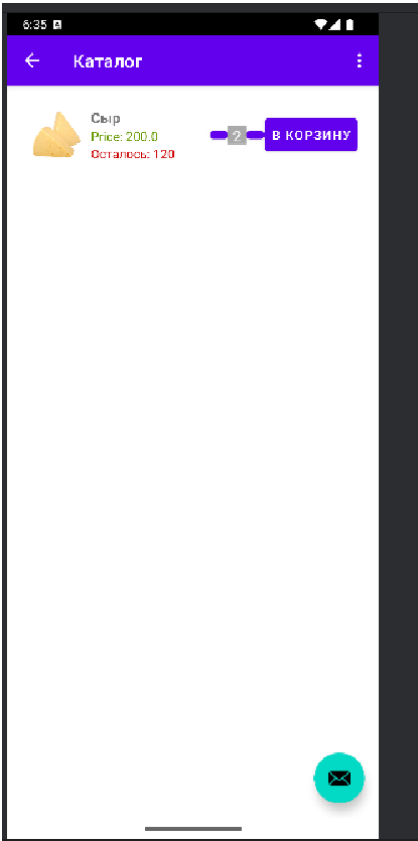


Рисунок 4 – Каталог товаров

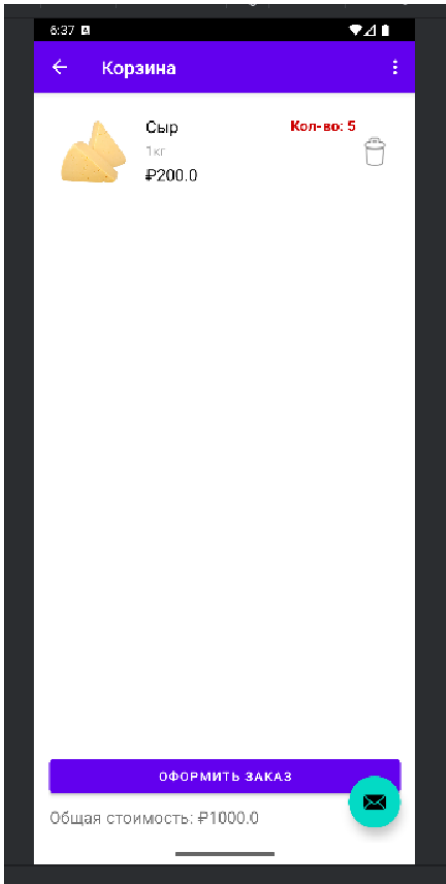


Рисунок 5 – Корзина

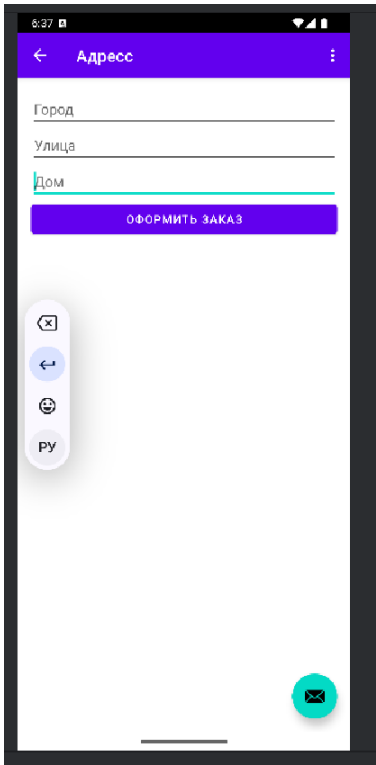


Рисунок 6 – Ввод адреса



Рисунок 7 – Активные заказы

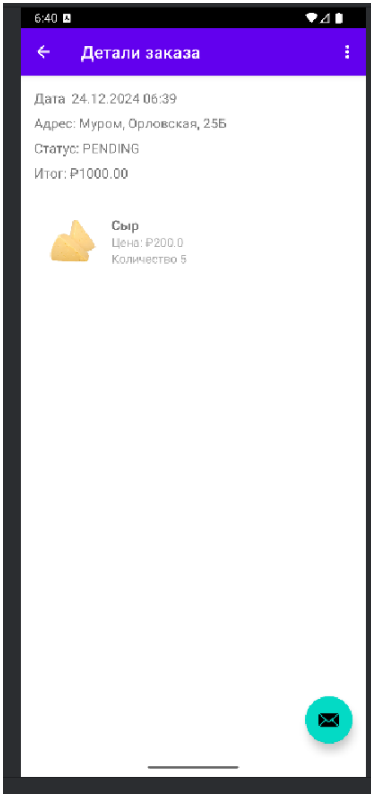


Рисунок 8 – Информация о заказе

					МИВУ 09.03.04 – 0.016 ПЗ	Лист
						35
Изм.	Лист	№ докум.	Подпись	Дата		

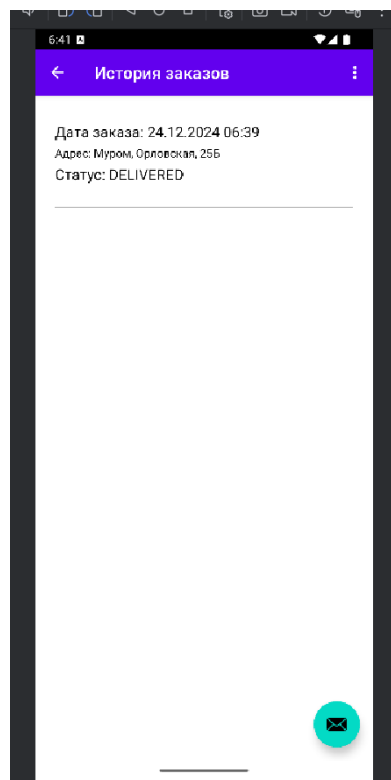


Рисунок 9 – Завершенные заказы

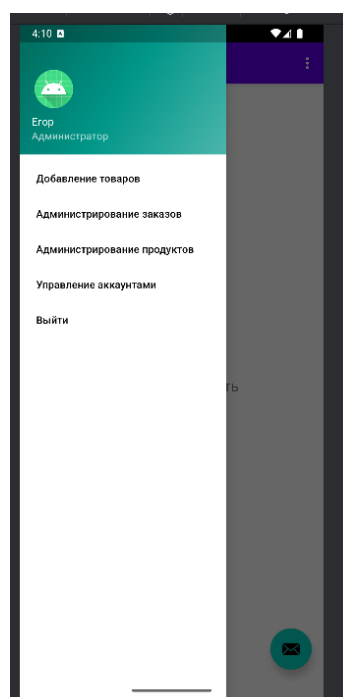


Рисунок 10 – Интерфейс администратора

					МИВУ 09.03.04 – 0.016 ПЗ	Лист
						36
Изм.	Лист	№ докум.	Подпись	Дата		

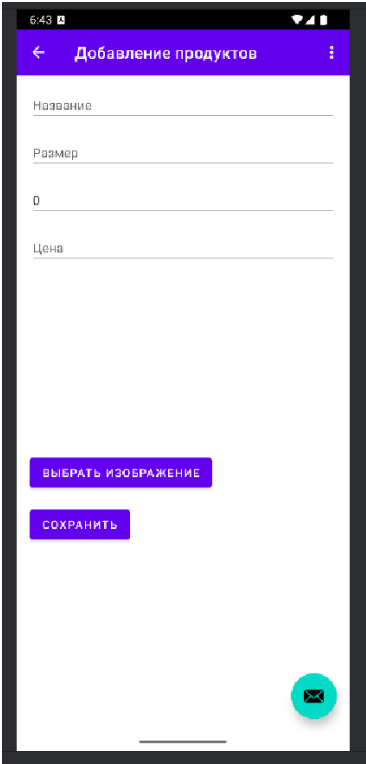


Рисунок 11 – Добавление товара

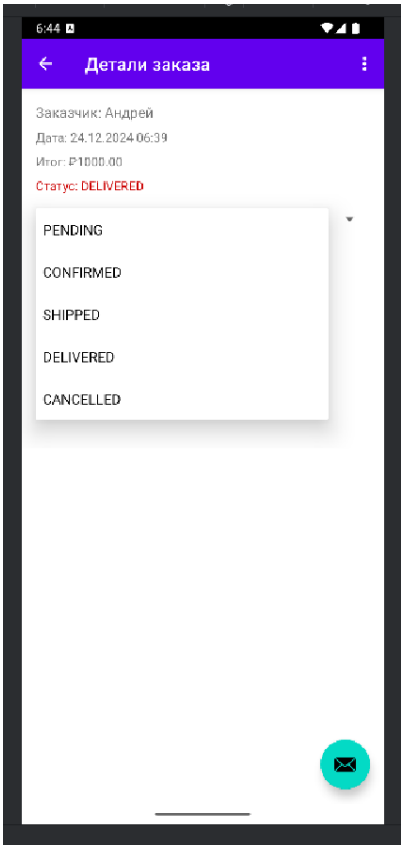


Рисунок 12 – Редактирование статуса заказа

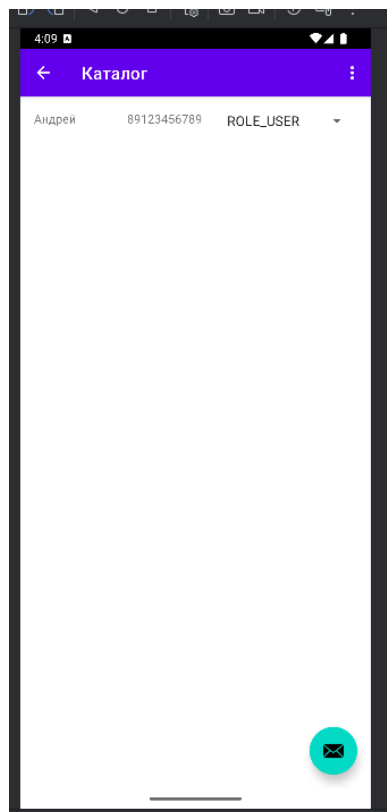


Рисунок 13 – Редактирование ролей на других аккаунтах

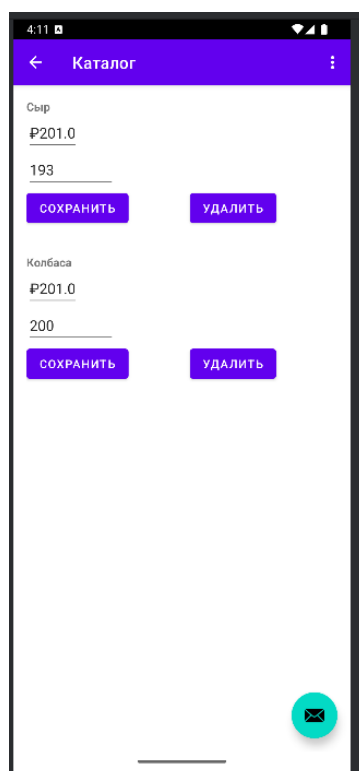


Рисунок 14 – Редактирование количества товара на складе

					МИВУ 09.03.04 – 0.016 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		38

Приложение 3. Код программы

ссылка на репозиторий GitHub с исходным кодом -
<https://github.com/Onzeboy/kotlincourse>

					МИВУ 09.03.04 – 0.016 ПЗ	Лист
						39
Изм.	Лист	№ докум.	Подпись	Дата		