Министерство образования и молодежной политики Свердловской области



ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

Отчёт по учебной практике

УП 01.02

Выполнила: Яковлева Анастасия Николаевна

Группа: ПР-31

Преподаватель: Мирошниченко Г.В.

2025

Содержание

[1. **Задание №1 Мобильное приложение «Производство мебели»** 3](#_Toc216720776)

[1.1 Описание задачи 3](#_Toc216720777)

[1.2 Структура проекта 4](#_Toc216720778)

[1.3 Описание разработанных функций 6](#_Toc216720779)

[1.4 Алгоритм решения 35](#_Toc216720780)

[1.5 Используемые библиотеки 38](#_Toc216720781)

[1.6 Тестовые случаи 39](#_Toc216720782)

[1.7 Используемые инструменты 41](#_Toc216720783)

[1.8 Описание пользовательского интерфейса 41](#_Toc216720784)

[1.9 Приложение (pr screen экранов) 43](#_Toc216720785)

# 1. **Задание №1 Мобильное приложение «Производство мебели»**

## Описание задачи

Cоздать приложение для Производства мебели (используя примеры сайтов в интернете)

В этой базе заказчик хотел бы хранить информацию о:

* номерах заказов (является первичным ключом)
* различных типах производимой мебели (диваны, столы, шкафы, стулья и т.п) – с фото и возможностью группировки по типу мебели. – информацию запрашивать через API
* различных предметах производимой мебели (например, диванов разных моделей)
* типах деталей, которые необходимы для производства мебели (гайки, шайбы, болты, винты и т.п) с возможностью группировки по типу детали.
* характеристиках каждой поставляемой детали (вес, металл, диаметр и т.п)
* поставщиках деталей
* клиентах (с предоставлением скидок)

Некоторые условия, существенные для проектирования базы данных:

* каждый поставщик может поставлять несколько различных деталей
* одна и та же деталь может поставляться разными поставщиками
* возможна поставка детали одного и того же типа, но с различными характеристиками
* в производстве одного предмета мебели может использоваться
* несколько различных деталей с разными характеристиками

На главной странице должна быть реализована авторизация и выбор режима работы (для работника производства мебели, для клиента и для поставщика деталей), с проверкой на правильность введенных данных (проверка на правильность ввода почты, логина и пароля) и сохранением через SharedPreference в базе данных.

В приложении должно быть организовано не менее 4-5 окон.

Навигацию осуществить наиболее оптимальным образом для приложения (с помощью Carousel with MotionLayout, Navigation) – должны быть использованы не менее 2-х видов навигации.

В приложении должен быть реализован следующий функционал: редактирование, удаление, добавление типов мебели и поставщиков, клиентов.

На главной странице поиск по типу мебели.

Создайте и заполните готовую базу и подключите его в проекте через ROOM.

## Структура проекта

Проект организован по стандартной структуре Android-приложения (Рисунок 1). В папке manifests находится AndroidManifest.xml, который объявляет все Activity и настройки приложения. Папка kotlin содержит исходный код на Kotlin: Activity для каждого экрана, адаптеры, фрагменты и классы данных. В папке Database находятся аннотированные сущности, дао-интерфейсы и сама база данных проекта.

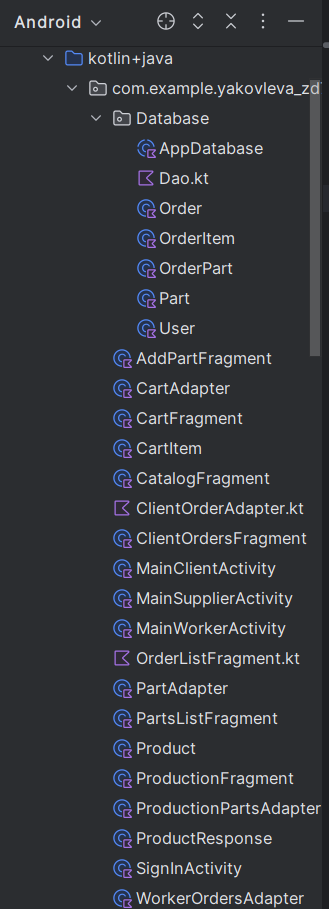


Рисунок 1 - Корневая папка app приложения

В папке res находятся ресурсы проекта: лэйауты, иконки, фигуры, строки, цвета, меню и навигации (Рисунок 2).

Изображение выглядит как текст, снимок экрана, меню, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 2 - Папка res

Файлы сборки Gradle находятся в корне проекта, управляя зависимостями и настройками компиляции (Рисунок 3)

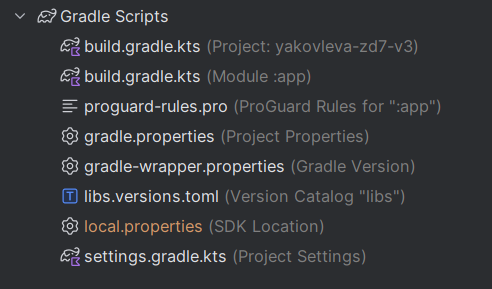


Рисунок 3 - Корневая папка Gradle Scripts

## 1.3 Описание разработанных функций

Методы класса SignInActivity

Метод performRegistration вызывается при нажатии на кнопку «Войти» на экране входа. Он проверяет входные данные на корректность, и, если они валидны, введенные почта и пароль ищутся в базе данных. Если они есть и совпадают, то производится вход на главный экран пользователя в зависимости от его роли (Рисунок 4), иначе проверяется есть ли вообще эта почта: если да, то выводится сообщение, что пароль неверный, а если нет, то регистрируется новый пользователь (Рисунок 5).



Рисунок 4 – performRegistration: проверка входных данных



Рисунок 5 - performRegistration: ошибка входа/регистрация

Функция registerNewUser регистрирует нового пользователя и добавляет его в базу данных (Рисунок 6)

Изображение выглядит как текст, снимок экрана, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 6 - registerNewUser

Функция saveUserSession сохраняет сессию пользователя в SharedPreferences (Рисунок 7)

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 7 - saveUserSession

Метод navigateToMainActivity направляет пользователя на главный экран его роли (Рисунок 8)

Изображение выглядит как текст, снимок экрана, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 8 - navigateToMainActivity

На каждом экране присутствует функция showSnackBar, которая выводит всплывающее сообщение (Рисунок 9).

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, снимок экрана

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 9 - showSnackBar

Методы класса MainClientActivity

MainClientActivity – это контейнер для фрагментов экранов клиента, он содержит только перегрузку метода onCreate, в которой настраивается нижняя навигация ().

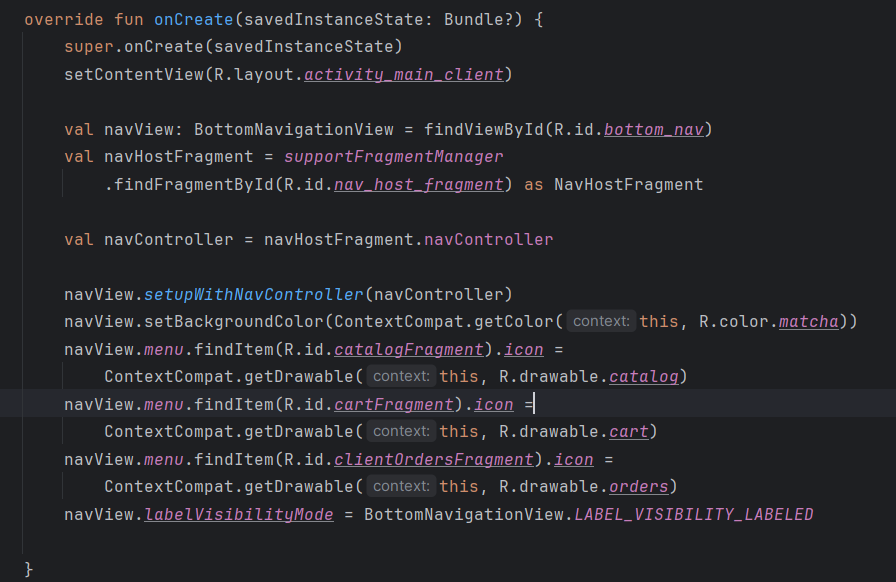


Рисунок 10 - перегрузка метода onCreate в MainClientActivity

Методы класса CatalogFragment

В перегрузке метода onViewCreated настраиваются таб-навигация и карусель карточек товаров, а также обработчики события нажатия кнопок вперед-назад карусели (Рисунок 11)



Рисунок 11 - перегрузка метода onViewCreated в CatalogFragment

Метод setupTabs настраивает таб-навигацию: добавляет табы из списка категорий товаров, устанавливает цвета при различных состояниях и вызывает метод, загружающий товары выбранной категории (Рисунок 12)

Изображение выглядит как текст, снимок экрана, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 12 - setupTabs

Метод setupCarousel настраивает карусель карточек товаров. В ее адаптере присутствует 3 обязательные перегрузки: populate, которая заполняет элементы карусели данными; count, которая возвращает количество элементов карусели и onNewItem, который вызывается при установке нового элемента (Рисунок 13)

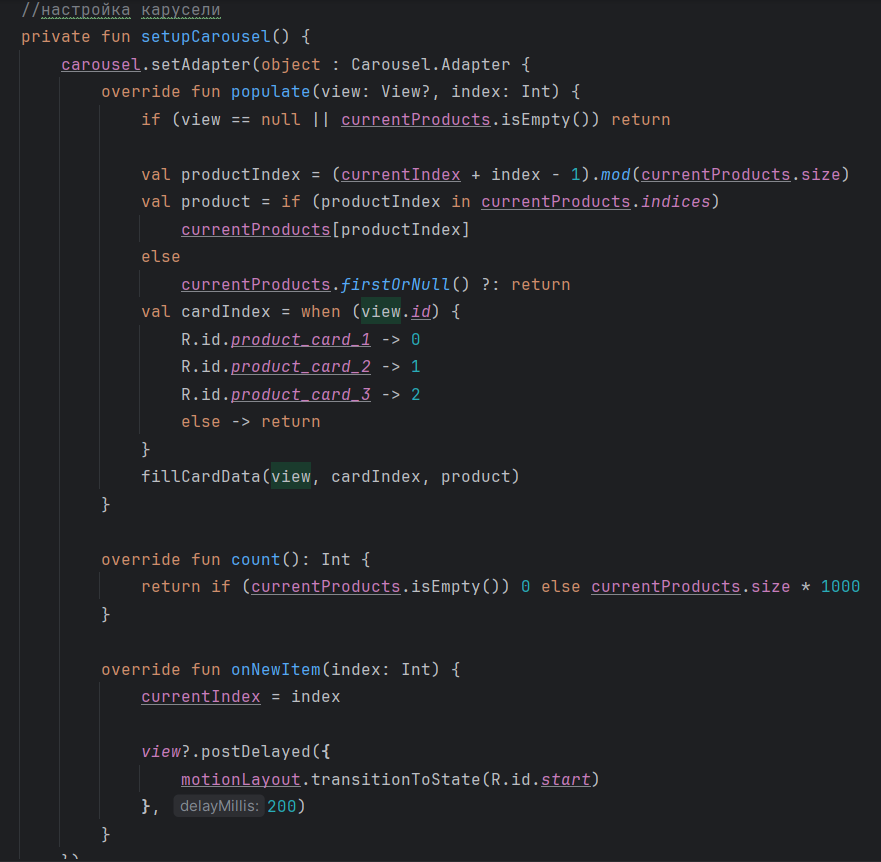


Рисунок 13 - setupCarousel

Процедура fillCardData динамически заполняет карточки товаров в карусели. Она определяет, какая именно карточка (первая, вторая или третья в карусели) должна быть заполнена (Рисунок 14), и подставляет соответствующие данные: название, описание, цену и изображение товара. Если описание слишком длинное - оно обрезается (Рисунок 15), а если изображение недоступно - показывается стандартная иконка. При нажатии кнопки "В корзину" товар добавляется в корзину пользователя (Рисунок 16).



Рисунок 14 - fillCardData: определение заполняемой карточки

Изображение выглядит как текст, снимок экрана, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 15 - fillCardData: установка описания товара

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 16 - fillCardData: установка изображения товара и добавление в корзину

Метод loadProducts загружает товары в каталог, предотвращая повторные запросы при активной загрузке (Рисунок 17).

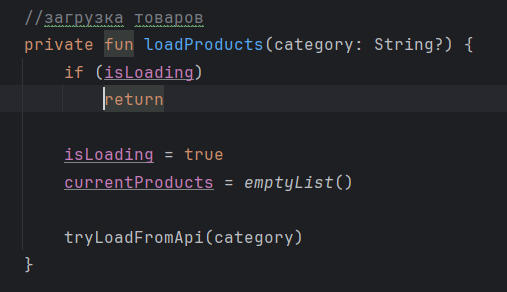


Рисунок 17 - loadProducts

Метод tryLoadFromApi пытается загрузить товары через API Ecom Furniture по категории (Рисунок 18), если что-то идет не так, загружаются тестовые товары (Рисунок 19)

Изображение выглядит как текст, снимок экрана, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 18 - tryLoadFromApi: загрузка товаров по категории

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 19 - tryLoadFromApi: обработка ошибок

Функция loadTestProducts вызывается в случае, если не удалось получить данные из API. Она создает список тестовых товаров (Рисунок 20)

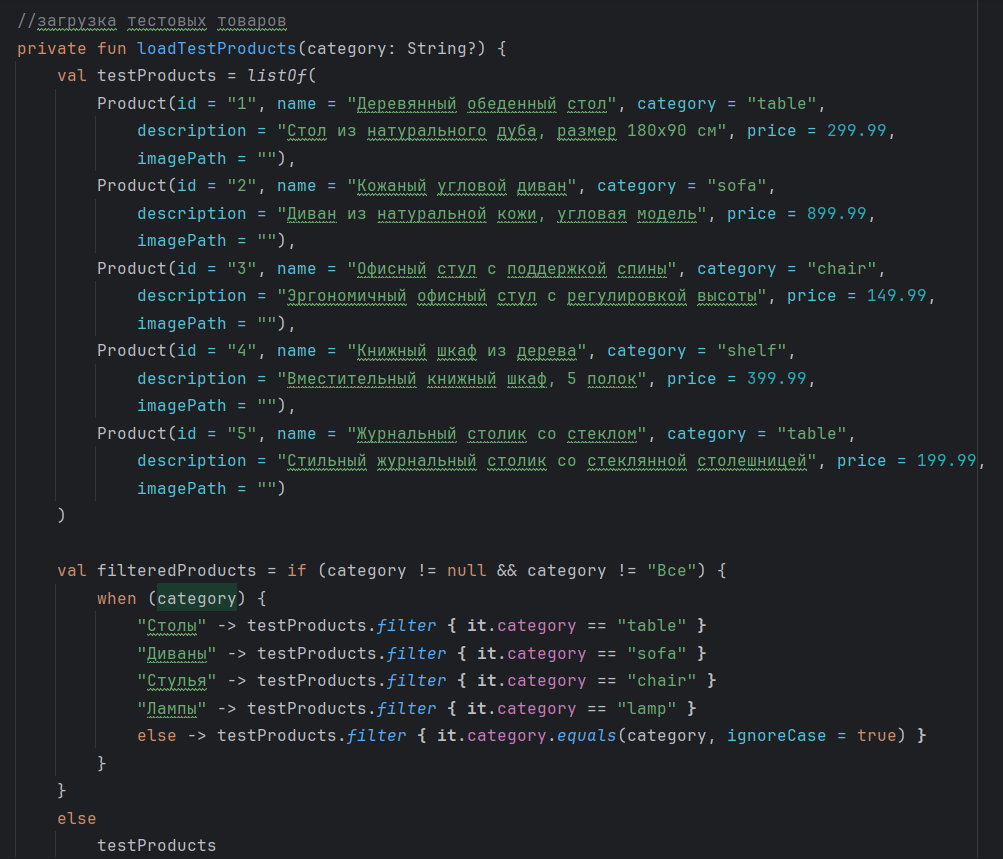


Рисунок 20 - loadTestProducts

Метод addToCart добавляет товар в корзину зашедшего пользователя: программа получает его id из сохраненной в SharedPreferences сессии и открывает файл его корзины cart\_userId. Если такой товар уже есть в корзине, то он увеличивает его количество на 1, инае добавляет новую запись о товаре. Все данные сохраняются в формате "ID|Название|Цена|Количество", что позволяет легко хранить и восстанавливать состояние корзины между сеансами приложения (Рисунок 21).

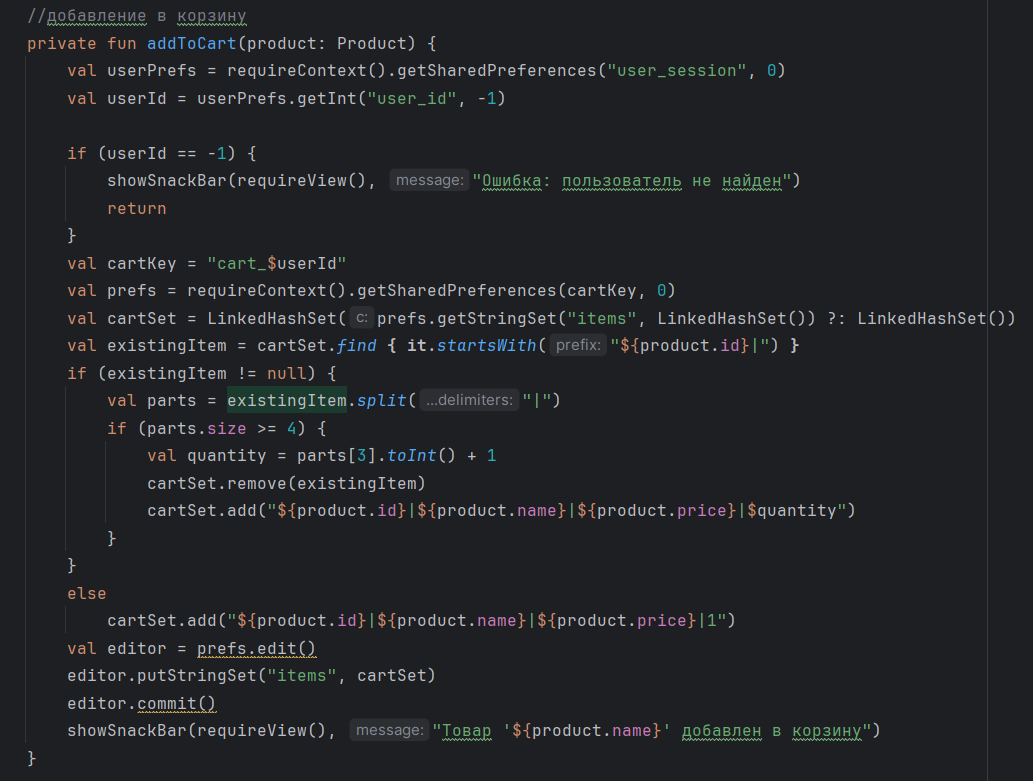


Рисунок 21 - addToCart

Методы класса CartFragment

Процедура loadCartItems загружает корзину вошедшего пользователя по его id (Рисунок 22).

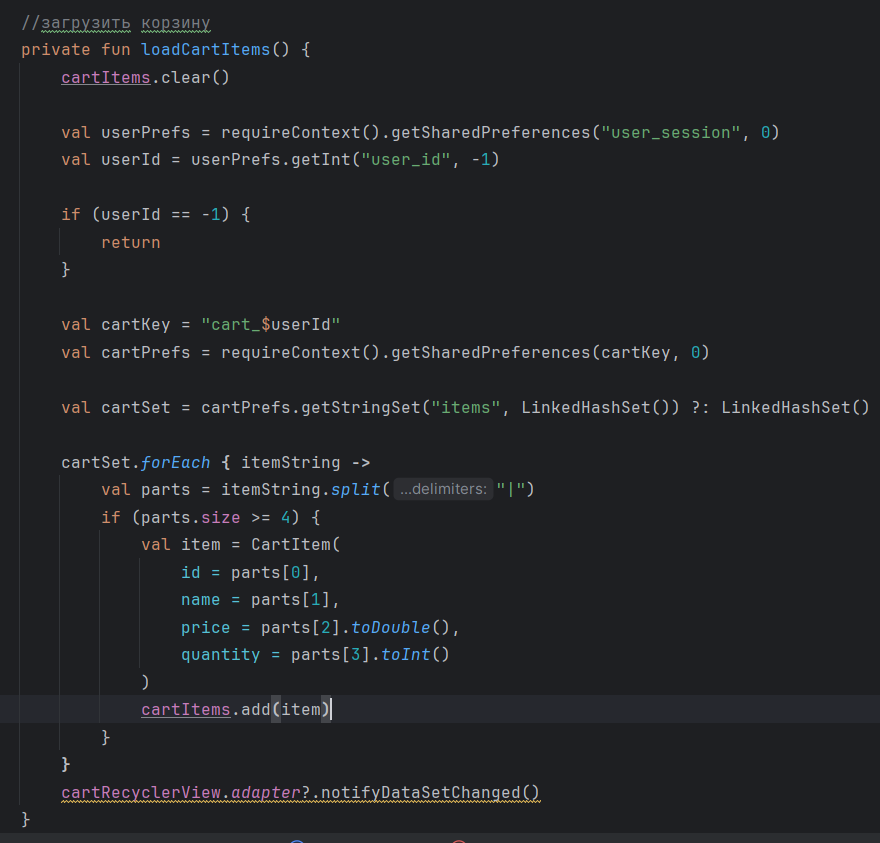


Рисунок 22 - loadCartItems

Метод saveCartItems сохраняет корзину пользователя в SharedPreferences (Рисунок 23)

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 23 - saveCartItems

Процедуры increaseQuanity и decreaseQuanity увеличивают и уменьшают количество определенного товара в корзине соответственно (Рисунок 24). Если вызвать декремент, когда в корзине всего 1 товар, он удалится.

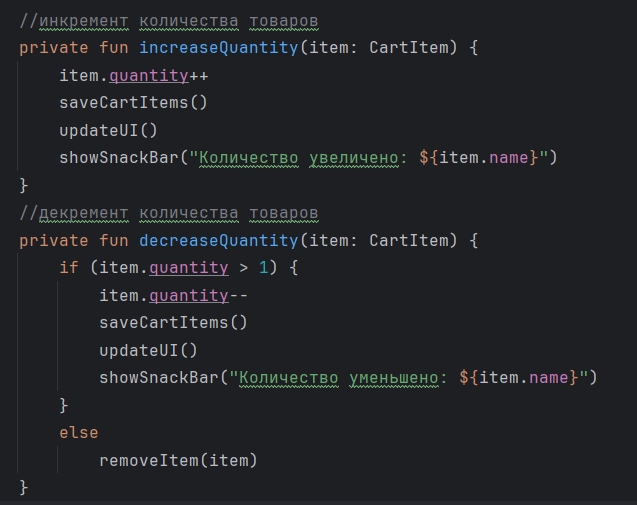


Рисунок 24 - increaseQuanity и decreaseQuanity

Процедура removeItem удаляет товар из корзины (Рисунок 25)

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 25 - removeItem

Процедура updateUI обновляет пользовательский интерфейс: меняется итоговая сумма заказа, в зависимости от заполненности корзины отображается либо список товаров и их количество в скобках на кнопке «Оформить заказ», либо надпись «Корзина пуста» (Рисунок 26).



Рисунок 26 - updateUI

Метод checkout оформляет заказ из корзины вошедшего пользователя (Рисунок 27), создавая постоянную запись в базе данных. Он проверяет наличие товаров в корзине, вычисляет общую сумму и генерирует уникальный номер заказа. Затем в отдельном потоке сохраняет заказ и все его товары в базу данных, определяя категории товаров по их названиям (Рисунок 28). После успешного сохранения он полностью очищает корзину пользователя в SharedPreferences и обновляет интерфейс, показывая сообщение об успешном оформлении заказа (Рисунок 29).

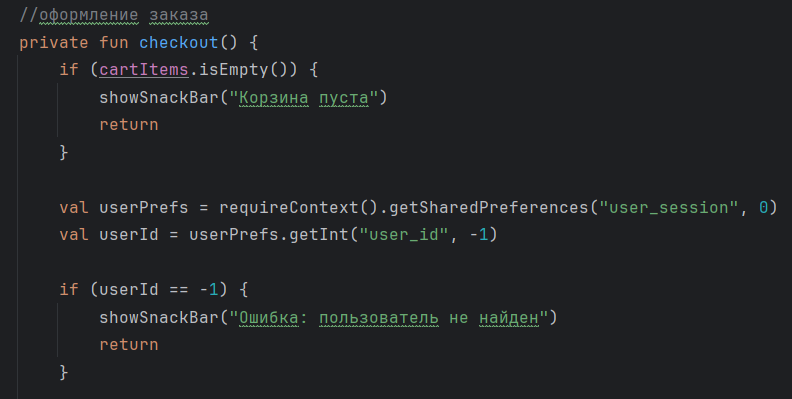


Рисунок 27 - checkout: полуение сессии пользователя

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 28 - checout: добавление заказа в базу данных

Изображение выглядит как текст, снимок экрана, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 29 - checkout: очистка корзины

Методы класса ClientOrdersFragment

Процедура loadOrders загружает заказы пользователя по его id (Рисунок 30), затем в отдельном потоке запрашивает из Room все заказы текущего клиента вместе с их товарами. Полученные данные сортируются по дате создания и отображаются в RecyclerView. Если заказов нет - показывается сообщение "Заказов пока нет" (Рисунок 31).

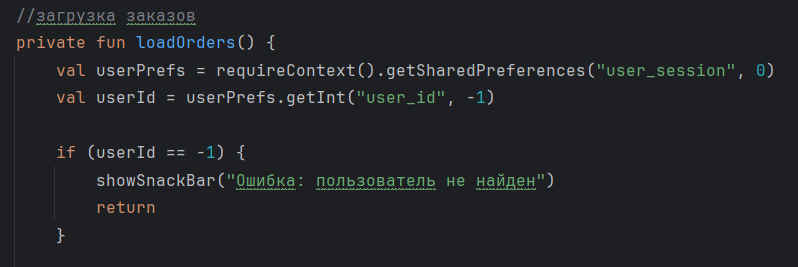


Рисунок 30 - loadOrders: сессия пользователя

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 31 - loadOrders: загрузка заказов из базы данных

Методы класса MainSupplierActivity такие же, как и у клиента (Рисунок 10). Класс является контейнером для фрагментов поставщика деталей.

Методы класса PartsListFragment

Настройка таб-навигации такая же, как у клиента, только категории не товаров, а деталей (Рисунок 12)

Процедура loadParts проверяет авторизацию поставщика (Рисунок 32), затем в отдельном потоке запрашивает из базы данных только его детали. Если выбрана конкретная категория - дополнительно фильтрует по типу детали. Результаты отображаются в списке: если деталей нет - показывается соответствующее сообщение, если есть - отображается список с возможностью редактирования или удаления (Рисунок 33).

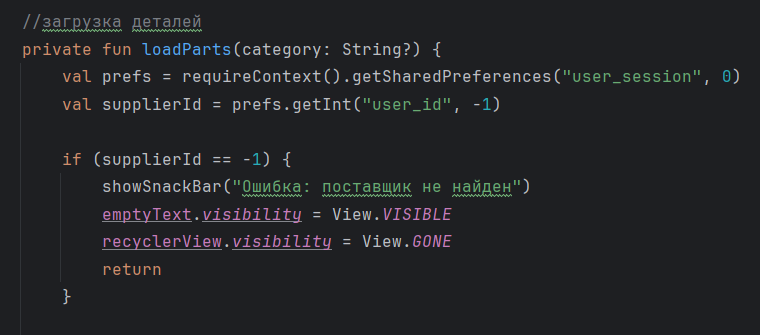


Рисунок 32 - loadParts: сессия пользователя

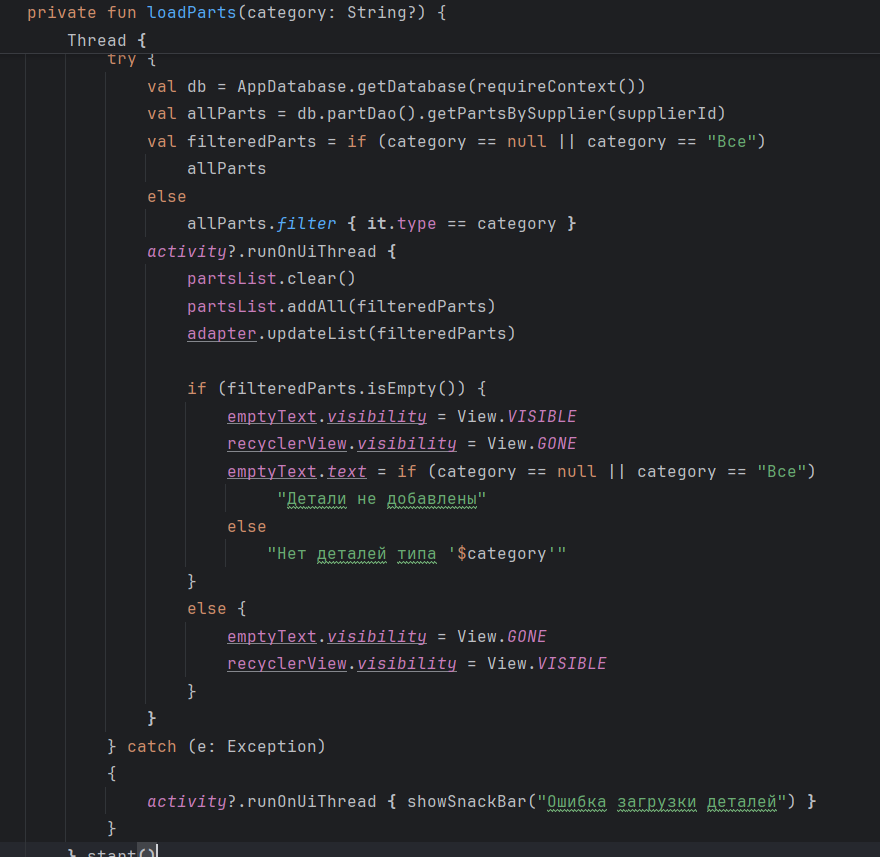


Рисунок 33 - loadParts: загрузка деталей

Метод открывает экран редактирования детали, передавая ID выбранной детали в следующий фрагмент. Он создает Bundle с параметром partId и выполняет навигацию к экрану AddPartFragment, который автоматически загрузит данные детали для редактирования (Рисунок 34).

Изображение выглядит как текст, Шрифт, снимок экрана

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 34 - editPart

Процедура deletePart удаляет деталь из базы данных (Рисунок 35)

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 35 - deletePart

Методы класса AddPartFragment

Метод setupSpinner настраивает спиннер с категориями деталей (Рисунок 36)

Изображение выглядит как текст, Шрифт, снимок экрана

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 36 - setupSpinner

Метод setupButton обрабатывает сохранение или обновление детали поставщика. Он проверяет заполнение полей и корректность ввода цены (Рисунок 37). Затем определяет, создается ли новая деталь или редактируется существующая по currentPartId (-1 – создается, >= 0 - редактируется). В отдельном потоке сохраняет данные в базу данных: либо добавляет новую запись (Рисунок 38), либо обновляет существующую (Рисунок 39), привязывая деталь к текущему поставщику через его ID. После успешного сохранения очищает форму, показывает подтверждение и возвращает пользователя назад к списку деталей.

Изображение выглядит как текст, снимок экрана

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 37 - setupButton: проверка валидности введенных данных

Изображение выглядит как текст, снимок экрана, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 38 - setupButton: создание новой детали

Изображение выглядит как текст, снимок экрана

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 39 - setupButton: редактирование существующей детали

Метод loadPartData загружает данные детали для редактирования (Рисунок 40). Он проверяет, был ли передан partId в аргументах фрагмента - если да, значит это режим редактирования. Затем в отдельном потоке запрашивает из базы данных деталь по этому ID и заполняет форму её данными: название, характеристики, цена и тип. Если ID не передан, форма остается пустой для создания новой детали.



Рисунок 40 - loadPartData

Процедура clearForm очищает поля ввода детали (Рисунок 41)

Изображение выглядит как текст, Шрифт, снимок экрана, дизайн

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 41 - clearForm

Класс MainWorkerActivity – контейнер для фрагментов экранов работника производства. Он не был изменен при создании.

Методы класса OrdersListFragment

Процедура setupTabs настраивает таб-навигацию для работника по статусам заказов: «Новый», «В производстве», «Готов» (Рисунок 42)

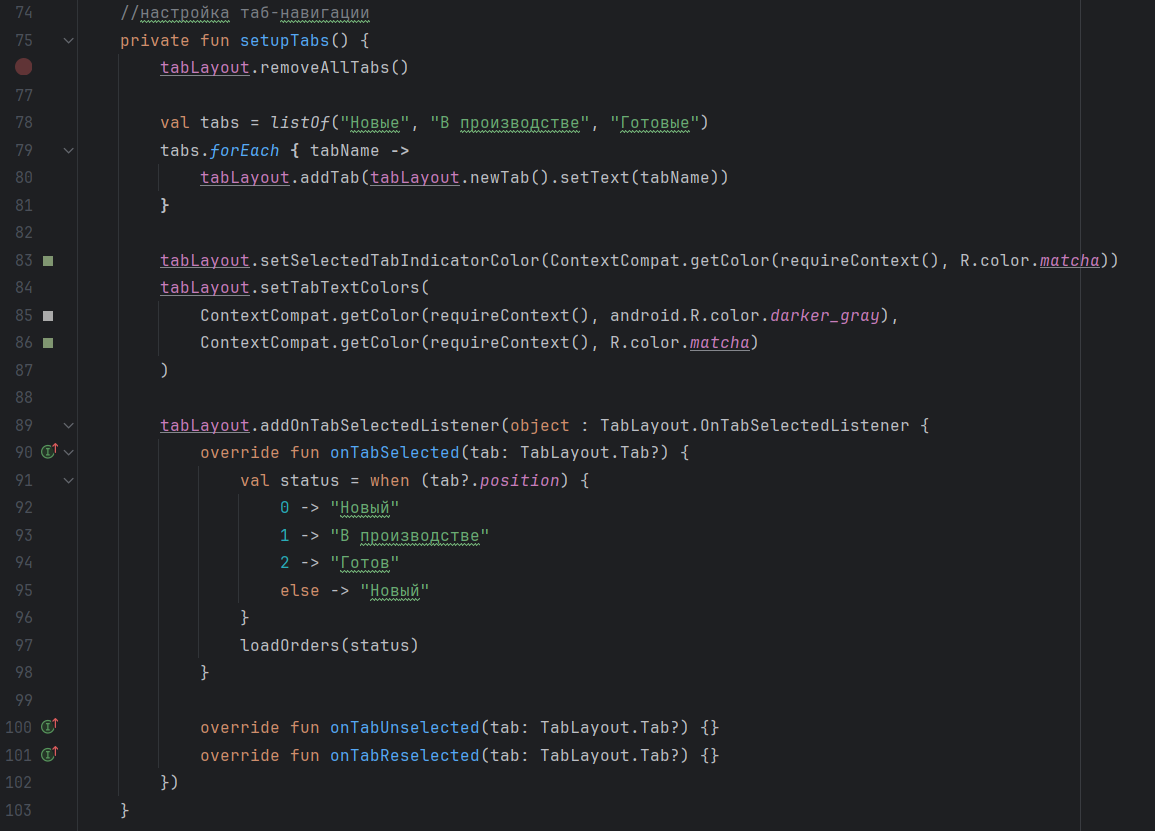


Рисунок 42 - setupTabs в OrdersListFragment

Метод openProduction переходит на экран производства для конкретного заказа. Он упаковывает Id заказа в Bundle и передает его через навигационную систему. Получающий фрагмент ProductionFragment извлекает этот ID и загружает все данные заказа: товары, клиента и доступные детали для производства (Рисунок 43)

Изображение выглядит как текст, Шрифт, снимок экрана

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 43 - openProduction

Метод loadOrders загружает все заказы из базы данных и расфасовывает их по статусу в соответствующие табы (Рисунок 44)

Изображение выглядит как текст, снимок экрана, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 44 - loadOrders

Метод startProduction запускает производство заказа, меняя его статус с "Новый" на "В производстве" и назначая работника. Он проверяет авторизацию работника, затем в отдельном потоке обновляет статус заказа в базе данных и назначает ему ID текущего работника. После успешного обновления переходит на экран производства с этим заказом и обновляет список заказов, чтобы отобразить изменения в реальном времени (Рисунок 45)

Изображение выглядит как текст, снимок экрана

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 45 - startProduction

Методы класса ProductionFragment

Метод загружает все данные для производства заказа в одном потоке. Он последовательно получает из базы данных: сам заказ, его товары, информацию о клиенте, все доступные детали и уже выбранные для этого заказа детали. Затем объединяет эти данные, отмечая какие детали уже выбраны и в каком количестве, после чего обновляет интерфейс - показывает информацию о заказе и список деталей для выбора (Рисунок 46)

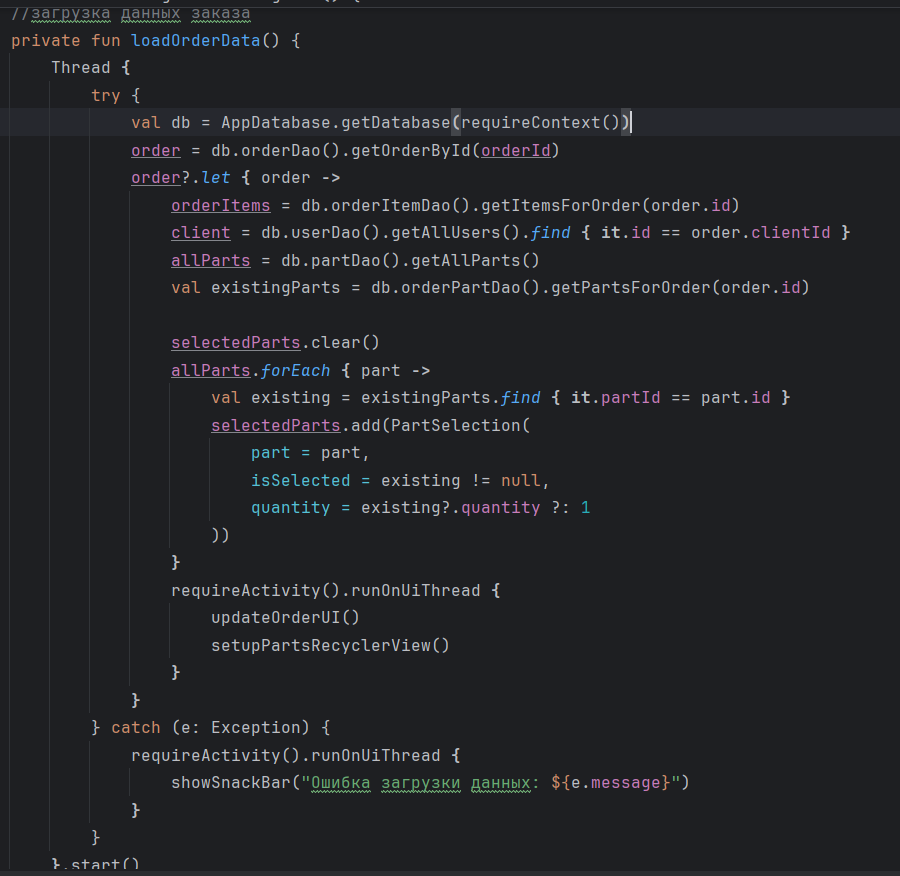


Рисунок 46 - loadOrderData

Процедура updateOrderUI обновляет интерфейс экрана производства, отображая всю информацию о заказе. Он показывает номер заказа, данные клиента, текущий статус и общую сумму, а также список всех товаров в заказе с их количеством и стоимостью (Рисунок 47). В зависимости от статуса заказа меняется цвет текста и кнопки: "Новый" - зелёный (matcha), "В производстве" – красно-коричневый (chai), "Готов" - коричневый (carob). Текст и активность кнопки также адаптируются под статус: "Начать производство", "Завершить производство" или неактивная "Заказ готов" (Рисунок 48)

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 47 - updateOrderUI: отображение информации о заказе

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 48 - updateOrderUI: настройка цвета и текста

Метод setupPartsRecyclerView настраивает список деталей для выбора в производстве (Рисунок 49).

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 49 - setupPartsRecyclerView

Метод savePartSelection сохраняет выбор детали для заказа в базе данных. Когда работник отмечает чекбокс детали - метод добавляет связь между заказом и деталью в таблицу order\_part с указанием количества. Если связь уже существует - обновляет количество. Если чекбокс снят - удаляет связь. Работник получает уведомление о действии (Рисунок 50)

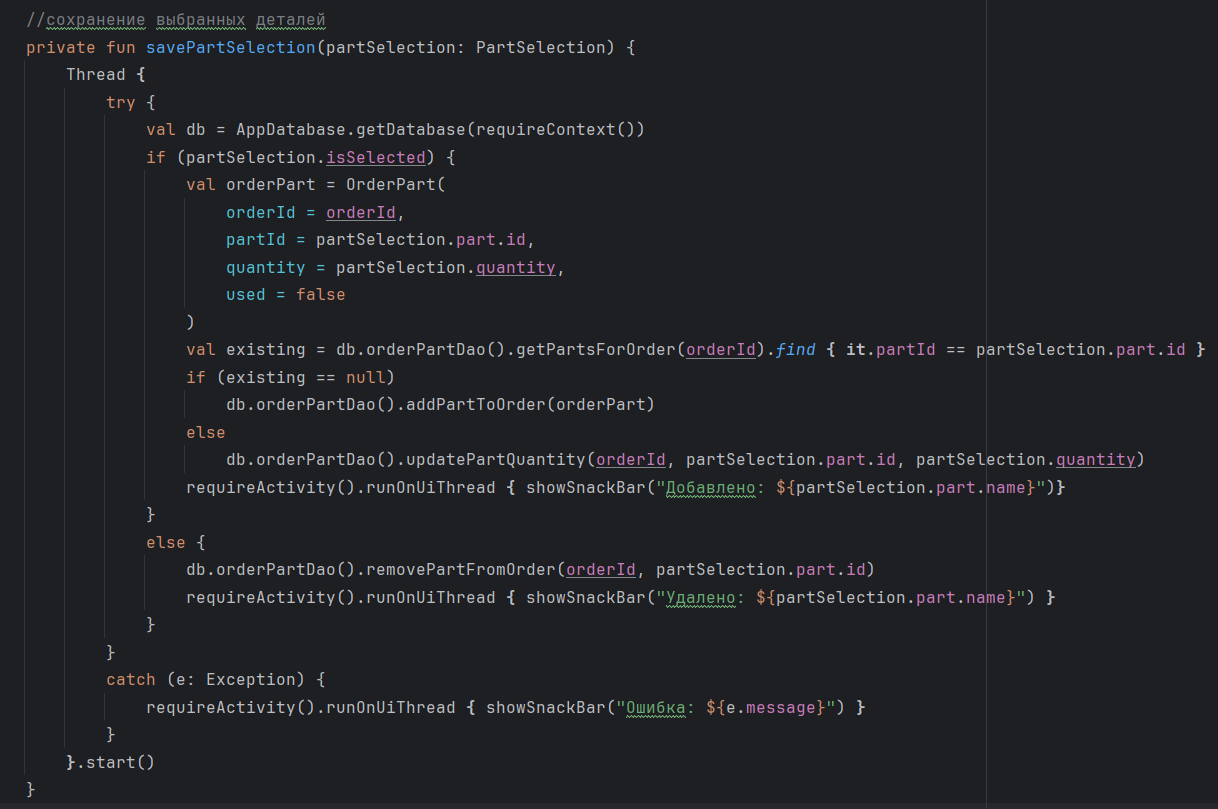


Рисунок 50 - savePartSelection

Метод setupCompleteButton настраивает обработчик основной кнопки на экране производства, которая меняет свое действие в зависимости от статуса заказа. Для заказа со статусом "Новый" кнопка запускает производство, меняя статус на "В производстве". Для заказа "В производстве" кнопка завершает производство, меняя статус на "Готов" и отмечая все выбранные детали как использованные. Для других статусов показывает сообщение об ошибке (Рисунок 51).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 51 - setupCompleteButton

Метод startProduction запускает производственный процесс для заказа. Он проверяет авторизацию работника, затем в отдельном потоке обновляет статус заказа в базе данных с "Новый" на "В производстве" и назначает ему Id текущего работника. После успешного сохранения обновляет интерфейс (Рисунок 52)

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 52 - startProduction

Метод completeProduction завершает производство заказа. Он проходит по всем выбранным деталям и помечает их как использованные в таблице order\_part, затем меняет статус заказа на "Готов" и фиксирует время завершения. После успешного сохранения обновляет интерфейс, показывая завершенный статус, и автоматически возвращает работника назад к списку заказов, где этот заказ теперь отображается во вкладке "Готовые" (Рисунок 53).

Изображение выглядит как текст, снимок экрана, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 53 - completeProduction

## 1.4 Алгоритм решения

Блоксхема для метода performRegistration (Рисунок 54)

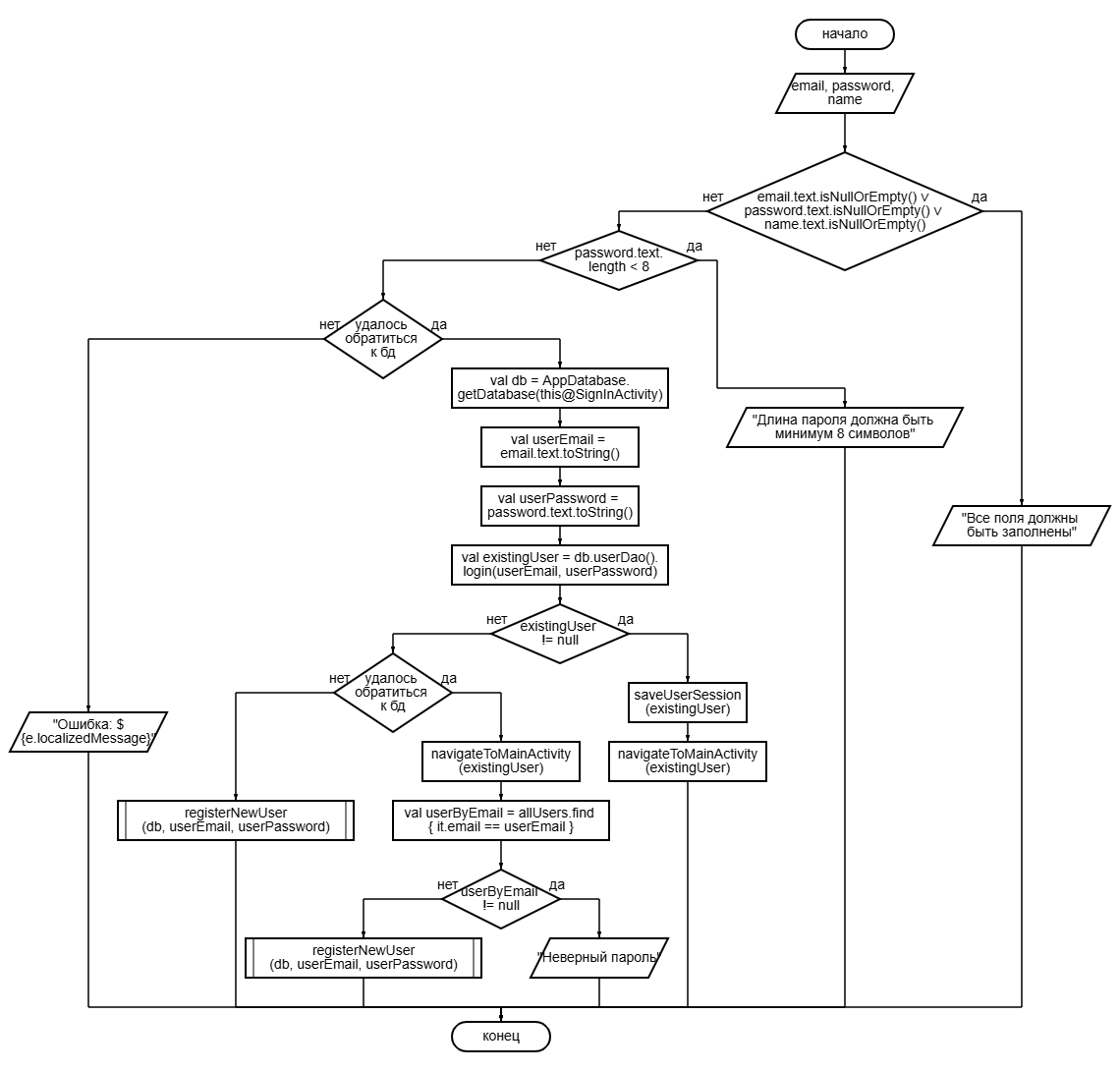


Рисунок 54 - Блоксхема к методу performRegistration

Блоксхема для метода navigateToMainActivity (Рисунок 55)

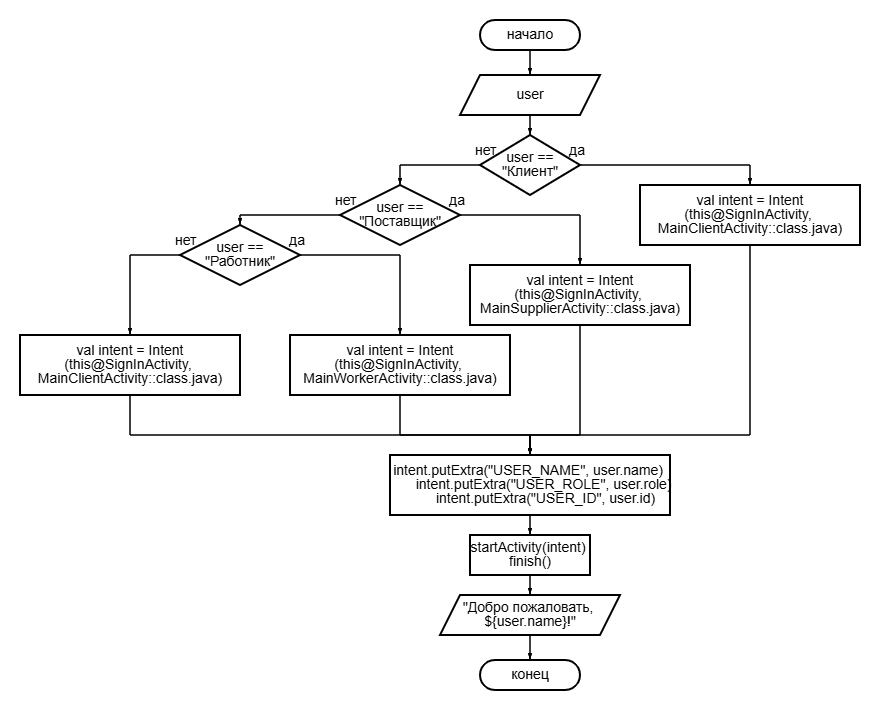


Рисунок 55 - Блоксхема к метоу navigateToMainActivity

Блоксхема для метода fillCardData (Рисунок 56)

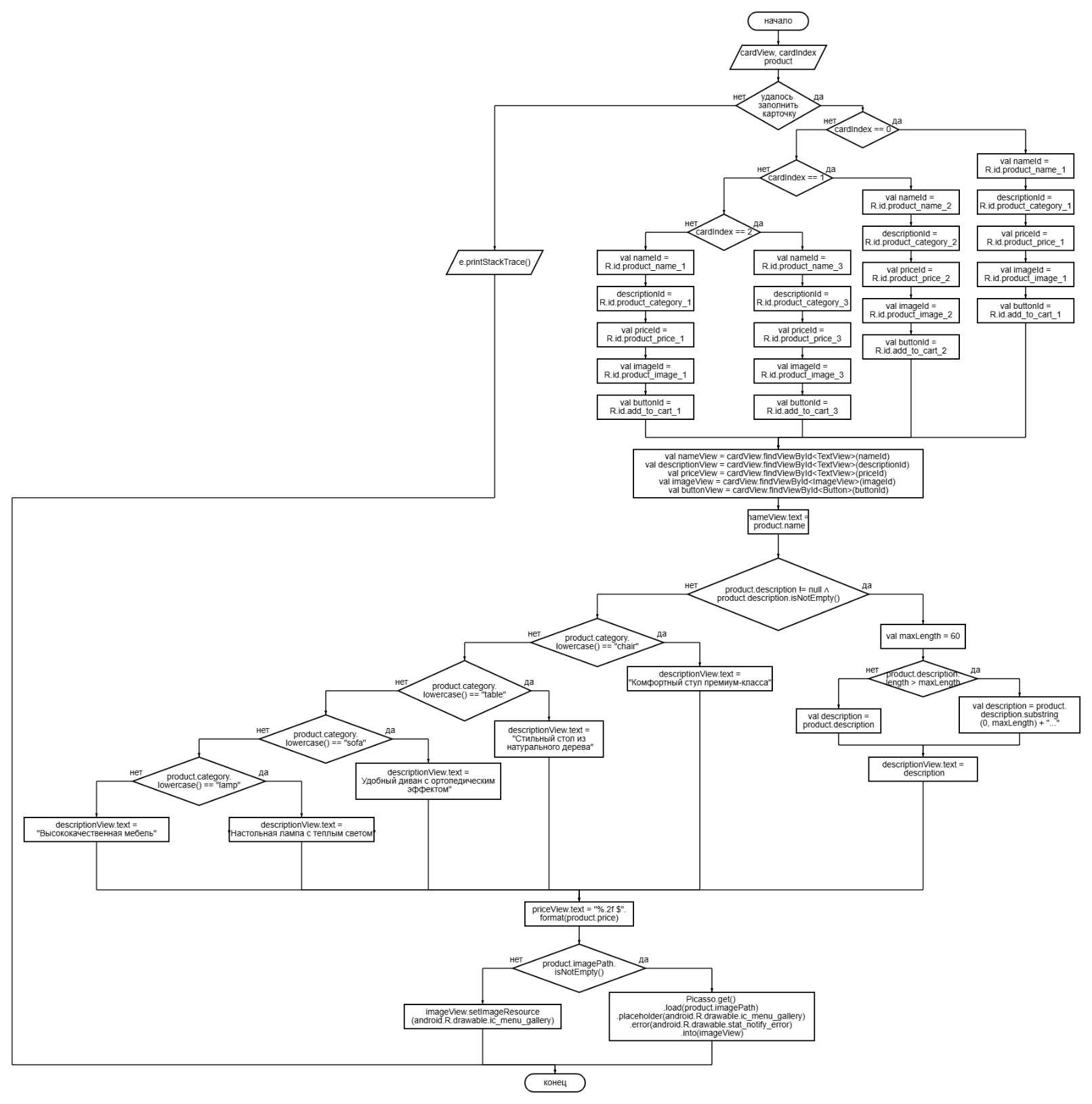


Рисунок 56 - Блоксхема к метоу fillCardData

## 1.5 Используемые библиотеки

* Room - локальная база данных для хранения пользователей, заказов и деталей.
* Picasso - загрузка и отображение фотографий мебели из API.
* Volley - отправка HTTP-запросов для получения каталога мебели.
* Gson - преобразование JSON-ответов API в объекты Kotlin.
* Material Design - современный UI с готовыми компонентами интерфейса.
* Navigation - управление переходами между экранами приложения.
* ConstraintLayout - гибкая верстка сложных интерфейсов.
* AppCompat - поддержка новых функций Android на старых версиях.

Описанные библиотеки приведены на рисунке 56 (Рисунок 57).

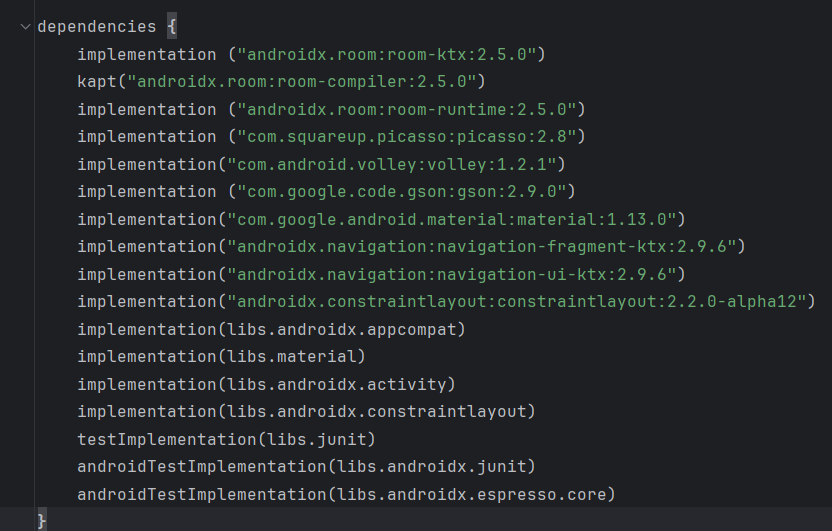


Рисунок 57 - Используемые библиотеки

## 1.6 Тестовые случаи

На экране входа присутствует проверка на пустые поля (Рисунок 58) и длину пароля (Рисунок 59). Если введенная почта присутствует в базе данных, а пароль к ней не подходит программа выводит всплывающее сообщение (Рисунок 60).

|  |  |  |
| --- | --- | --- |
| Рисунок 58 - Экран входа: пустые поля | Рисунок 59 - Экран входа: короткий пароль | Рисунок 60 - Экран входа: неверный пароль |

На экране добавления/редактирования деталей у поставщика присутствуют проверки на пустые обязательные поля (Рисунок 61) и корректную цену (Рисунок 62)

|  |  |
| --- | --- |
| Рисунок 61 - Экран добавить деталь: пустые обязательные поля | Рисунок 62 - Экран добавить деталь: некорректная цена |

## 1.7 Используемые инструменты

Приложение написано на языке программирования Kotlin в среде разработки Android Studio

## 1.8 Описание пользовательского интерфейса

Экран входа (SignInActivity) содержит поля для ввода почты, имени и пароля, спиннер с выбором роли и кнопку входа. При нажатии проверяется заполненность полей и соответствие пароля почте: если введенной почты нет в базе данных – регистрируется новый пользователь. После успешного входа происходит переход на главный экран в зависимости от роли.

Экраны клиента (MainClientActivity) содержат нижнюю навигацию.

Каталог (CatalogFragment) содержит карточки товаров, перелистывающиеся в карусели с Motion Layout. На карточке товара присутствует его фотография, название, описание, цена и кнопка «Добавить в корзину». Сверху таб-навигация по категориям товаров: «Все», «Столы», «Стулья», «Диваны», «Лампы».

К каждому клиенту по его id привязана локальная корзина (CartFragment). При первом заходе корзина пуста, это отображается на экране. Снизу плашка с итоговой суммой и кнопкой «Корзина пуста» или «Оформить заказ». Когда клиент добавляет товар в корзину, в корзине сверху отображается плашка с его названием, ценой, итоговой ценой, количество, кнопки уменьшения и увеличения количества и кнопка «Удалить».

К каждому клиенту по его id привязан список заказов (ClientOrdersFragment). На экране «Мои заказы», если список пуст, отображается надпись «Заказов пока нет». Когда заказ оформляется, на этом экране отображается его id, дата и время, сумма, список товаров и статус. Если статус «Новый», то у него коричневый фон, если «В производстве» - красно-коричневый, а если «Готов» - зеленый.

Экраны поставщика деталей (MainSupplierActivity) содержат нижнюю навигацию.

Экран «Детали» (PartsListFragment) содержит таб-навигацию по типам деталей: «Все», «Болт», «Гайка», «Винт», «Шайба», «Шуруп», «Крепеж», «Другое». У каждого поставщика в списке отображаются только его детали. Если деталей нет отображается надпись «Детали не добавлены», а при навигации между табами – «Нет деталей типа ‘тип’». У каждой детали в списке отображается название, тип, характеристика, цена за единицу, а также зеленая кнопка «Редактировать» и красно-коричневая кнопка «Удалить».

Экран «Добавить» (AddPartFragment) содержит название экрана («Добавить деталь» или «Редактировать») в зависимости от режима, а также поля для ввода названия, характеристики и цены, спиннер для выбора типа и кнопка «Сохранить».

Навигация между экранами работника производства (mainWorkerActivity) осуществлена через интенты.

Экран заказов (OrdersListFragment) содержит таб-навигацию по статусам заказов: «Новый», «В производстве» и «Готов». Если заказов нет, в зависимости от таба, на котором находится пользователь, отображается надпись «Нет новых заказов», «Нет заказов в работе» или «Нет готовых заказов». В списке заказ имеет id, имя клиента, дату и время, список товаров, сумму, статус в зеленой плашке и кнопку. В зависимости от статуса заказа текст и фон кнопки меняется: если заказ новый, то «Произвести» на зеленом фоне, если в производстве – «Продолжить» на красно-коричневом фоне и если готов, то «Просмотр» на красно-коричневом фоне.

При нажатии на «Произвести» или «Продолжить» открывается экран производства (ProductionFragment). Сверху отображается информация о заказе: id, имя клиента, статус, товары и сумма. В самом низу кнопка «Завершить производство», в центре список из деталей для производства. у каждой детали из списка есть чекбокс (используется/не используется в производстве), название, цена за единицу, тип, характеристика, id поставщика и поле ввода для количества деталей.

## 1.9 Приложение (pr screen экранов)

Экраны приложения представлены на рисунках ниже (*Рисунок 63*-*Рисунок 86*).

|  |  |  |
| --- | --- | --- |
| Рисунок 63 - Экран входа | Рисунок 64 - Каталог: "Все" | Рисунок 65 - Каталог: "Столы" |
| Рисунок 66 - Каталог: "Стулья" | Рисунок 67 - Каталог: "Диваны" | Рисунок 68 - Каталог: "Лампы" |
| Рисунок 69 - Корзина: пустая корзина | Рисунок 70 - Корзина: список товаров | Рисунок 71 - Корзина: оформление заказа |
| Рисунок 72 - Мои заказы: нет заказов | Рисунок 73 - Мои заказы: список заказов | Рисунок 74 - Детали: нет деталей |
| Рисунок 75 - Детали: "Все" | Рисунок 76 - Детали: список деталей по типу | Рисунок 77 - Детали: нет деталей определенного типа |
| Рисунок 78 - Добавить деталь | Рисунок 79 - Редактировать деталь | Рисунок 80 - Новые заказы: нет заказов |
| Рисунок 81 - Заказы а производстве: нет заказов | Рисунок 82 - Новые заказы: список заказов | Рисунок 83 - Заказы в производстве: список заказов |
| Рисунок 84 - Готовые заказы | Рисунок 85 - Производство: производство начато | Рисунок 86 - Производство |