Министерство образования и молодежной политики Свердловской области



ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

Отчёт по программе «**Приложение – Производство мебели**»

Выполнил: Меньшиков А.А.

Группа: ПР-32

Преподаватель: Мирошниченко Г.В

2025

**Входные данные:**

Пользовательские данные: Логин, пароль, информация о клиенте или поставщике.

Данные о товарах: Название, описание, цена, тип.

Данные о типах товаров: Название типа товара.

Данные о деталях: Вес, материал, диаметр.

**Выходные данные:**

Списки товаров: Отображение товаров.

Списки клиентов/поставщиков: Отображение информации, редактирование и удаление.

Управление товарами и деталями: Добавление, редактирование и удаление товаров и деталей.

**Код программы:**

База данных:

package data  
  
import android.annotation.SuppressLint  
import android.content.ContentValues  
import android.content.Context  
import android.database.Cursor  
import android.database.sqlite.SQLiteDatabase  
import android.database.sqlite.SQLiteOpenHelper  
import android.util.Log  
import kotlin.Pair as Pair  
  
class DatabaseHelper(context: Context) : SQLiteOpenHelper(context, DATABASE\_NAME, null, DATABASE\_VERSION) {  
  
 companion object {  
 private const val DATABASE\_NAME = "furniture\_db"  
 private const val DATABASE\_VERSION = 1  
 }  
  
 override fun onCreate(db: SQLiteDatabase) {  
 // Создание таблиц  
  
 db.execSQL("""  
 CREATE TABLE Clients (  
 client\_id INTEGER PRIMARY KEY AUTOINCREMENT,  
 name TEXT,  
 discount REAL  
 )  
 """)  
  
 db.execSQL("""  
 CREATE TABLE FurnitureTypes (  
 furniture\_type\_id INTEGER PRIMARY KEY AUTOINCREMENT,  
 name TEXT,  
 photo\_url TEXT  
 )  
 """)  
 db.execSQL("""  
 CREATE TABLE FurnitureItems (  
 furniture\_item\_id INTEGER PRIMARY KEY AUTOINCREMENT,  
 furniture\_type\_id INTEGER,  
 model TEXT,  
 FOREIGN KEY (furniture\_type\_id) REFERENCES FurnitureTypes(furniture\_type\_id)  
 )  
 """)  
 db.execSQL("""  
 CREATE TABLE PartTypes (  
 part\_type\_id INTEGER PRIMARY KEY AUTOINCREMENT,  
 name TEXT  
 )  
 """)  
 db.execSQL("""  
 CREATE TABLE Parts (  
 part\_id INTEGER PRIMARY KEY AUTOINCREMENT,  
 part\_type\_id INTEGER,  
 name TEXT,  
 FOREIGN KEY (part\_type\_id) REFERENCES PartTypes(part\_type\_id)  
 )  
 """)  
 db.execSQL("""  
 CREATE TABLE PartCharacteristics (  
 characteristic\_id INTEGER PRIMARY KEY AUTOINCREMENT,  
 part\_id INTEGER,  
 weight REAL,  
 material TEXT,  
 diameter REAL,  
 FOREIGN KEY (part\_id) REFERENCES Parts(part\_id)  
 )  
 """)  
 db.execSQL("""  
 CREATE TABLE Orders (  
 order\_id INTEGER PRIMARY KEY AUTOINCREMENT,  
 client\_id INTEGER,  
 furniture\_type\_id INTEGER,  
 order\_date DATETIME,  
 status TEXT,  
 FOREIGN KEY (client\_id) REFERENCES Clients(client\_id),  
 FOREIGN KEY (furniture\_type\_id) REFERENCES FurnitureTypes(furniture\_type\_id)  
 )  
 """)  
  
 db.execSQL("""  
 CREATE TABLE Users (  
 user\_id INTEGER PRIMARY KEY AUTOINCREMENT,  
 login TEXT UNIQUE,  
 password TEXT,  
 email TEXT,  
 role TEXT  
 )  
 """)  
 db.execSQL("""  
 CREATE TABLE Details (  
 detail\_id INTEGER PRIMARY KEY AUTOINCREMENT,  
 part\_id INTEGER,  
 name TEXT,  
 FOREIGN KEY (part\_id) REFERENCES Parts(part\_id)  
 )  
 """)  
 addSampleData(db)  
 }  
  
 private fun addSampleData(db: SQLiteDatabase) {  
 val furnitureTypes = *listOf*(  
 "Стол",  
 "Стул",  
 "Диван",  
 "Кровать",  
 "Шкаф",  
 "Тумбочка"  
 )  
  
 for (type in furnitureTypes) {  
 val values = ContentValues().*apply* **{**  
put("name", type)  
 put("photo\_url", "") // Если у вас есть URL для фото, добавьте его  
 **}**  
db.insert("FurnitureTypes", null, values)  
 }  
  
 // Добавление предметов мебели с конкретными моделями  
 val items = *listOf*(  
 Pair(1, "Деревянный стол"),  
 Pair(1, "Стеклянный стол"),  
 Pair(2, "Кожаный стул"),  
 Pair(2, "Деревянный стул"),  
 Pair(3, "Угловой диван"),  
 Pair(4, "Двуспальная кровать"),  
 Pair(5, "Шкаф-купе"),  
 Pair(5, "Шкаф угловой"),  
 Pair(6, "Тумбочка сундук"),  
 Pair(6, "Тумбочка")  
 )  
  
 for (item in items) {  
 val values = ContentValues().*apply* **{**  
put("furniture\_type\_id", item.first)  
 put("model", item.second)  
 **}**  
db.insert("FurnitureItems", null, values)  
 }  
  
 // Добавление типов деталей  
 val partTypes = *listOf*("Ножка", "Столешница", "Спинка", "Матрас", "Дверца", "Полка", "Ящик")  
  
 for (partType in partTypes) {  
 val values = ContentValues().*apply* **{**  
put("name", partType)  
 **}**  
db.insert("PartTypes", null, values)  
 }  
  
 // Добавление деталей с уникальными материалами для каждого предмета мебели  
 val parts = *listOf*(  
 Pair(1, "Столешница стола"),  
 Pair(1, "Ножка стола"),  
 Pair(2, "Спинка стула"),  
 Pair(2, "Ножка стула"),  
 Pair(3, "Спинка дивана"),  
 Pair(4, "Матрас кровати"),  
 Pair(5, "Дверца шкафа"),  
 Pair(5, "Полка шкафа"),  
 Pair(5, "Ящик шкафа"),  
 Pair(6, "Ящик тумбочки"),  
 Pair(6, "Полка тумбочки")  
 )  
  
 for (part in parts) {  
 val values = ContentValues().*apply* **{**  
put("part\_type\_id", part.first) // Используем id типа детали  
 put("name", part.second)  
 **}**  
val partId = db.insert("Parts", null, values)  
  
 // Добавление характеристик для деталей с разнообразными материалами  
 val characteristics = when (part.second) {  
 "Столешница стола" -> *listOf*(  
 PartCharacteristic(partId, 5.0, "Дерево", 1.0),  
 PartCharacteristic(partId, 4.0, "Стекло", 0.0),  
 PartCharacteristic(partId, 0.5, "Ламинат", 0.0),  
 PartCharacteristic(partId, 3.0, "МДФ", 0.0),  
 PartCharacteristic(partId, 2.5, "Керамика", 0.0)  
 )  
 "Ножка стола" -> *listOf*(  
 PartCharacteristic(partId, 2.5, "Дерево", 0.0),  
 PartCharacteristic(partId, 1.5, "Металл", 0.0),  
 PartCharacteristic(partId, 0.3, "Пластик", 0.0),  
 PartCharacteristic(partId, 1.0, "Композитный материал", 0.0)  
 )  
 "Спинка стула" -> *listOf*(  
 PartCharacteristic(partId, 1.0, "Кожа", 0.0),  
 PartCharacteristic(partId, 0.8, "Ткань", 0.0),  
 PartCharacteristic(partId, 0.5, "Экокожа", 0.0)  
 )  
 "Матрас кровати" -> *listOf*(  
 PartCharacteristic(partId, 3.0, "Пенопласт", 0.0),  
 PartCharacteristic(partId, 2.5, "Латекс", 0.0),  
 PartCharacteristic(partId, 3.5, "Пружинный блок", 0.0)  
 )  
 "Дверца шкафа" -> *listOf*(  
 PartCharacteristic(partId, 2.0, "Дерево", 0.0),  
 PartCharacteristic(partId, 1.0, "Стекло", 0.0),  
 PartCharacteristic(partId, 1.5, "МДФ", 0.0)  
 )  
 "Полка шкафа" -> *listOf*(  
 PartCharacteristic(partId, 3.0, "Дерево", 0.0),  
 PartCharacteristic(partId, 1.5, "Стекло", 0.0),  
 PartCharacteristic(partId, 2.0, "Ламинат", 0.0)  
 )  
 "Ящик шкафа" -> *listOf*(  
 PartCharacteristic(partId, 2.0, "Дерево", 0.0),  
 PartCharacteristic(partId, 0.5, "Металл", 0.0),  
 PartCharacteristic(partId, 0.3, "Пластик", 0.0)  
 )  
 "Ящик тумбочки" -> *listOf*(  
 PartCharacteristic(partId, 1.5, "Дерево", 0.0),  
 PartCharacteristic(partId, 0.3, "Металл", 0.0),  
 PartCharacteristic(partId, 0.2, "Пластик", 0.0)  
 )  
 else -> *emptyList*()  
 }  
  
 for (characteristic in characteristics) {  
 val values = ContentValues().*apply* **{**  
put("part\_id", partId) // Используем ID детали  
 put("weight", characteristic.weight)  
 put("material", characteristic.material)  
 put("diameter", characteristic.diameter)  
 **}**  
db.insert("PartCharacteristics", null, values)  
 }  
 }  
 }  
 @SuppressLint("Range")  
 fun getAllFurnitureItemsWithIds(): List<Pair<Int, String>> {  
 val furnitureItems = *mutableListOf*<Pair<Int, String>>()  
 val db = *readableDatabase*  
val cursor: Cursor? = db.rawQuery("""  
 SELECT FurnitureItems.furniture\_item\_id, FurnitureItems.model, FurnitureTypes.name AS furniture\_type\_name   
FROM FurnitureItems   
INNER JOIN FurnitureTypes ON FurnitureItems.furniture\_type\_id = FurnitureTypes.furniture\_type\_id  
 """, null)  
  
 cursor?.*let* **{**  
if (**it**.moveToFirst()) {  
 do {  
 val furnitureItemId = **it**.getInt(**it**.getColumnIndexOrThrow("furniture\_item\_id"))  
 val model = **it**.getString(**it**.getColumnIndexOrThrow("model"))  
  
 // Добавляем в список пару (ID, Название)  
 furnitureItems.add(Pair(furnitureItemId, model))  
 } while (**it**.moveToNext())  
 } else {  
 Log.d("DatabaseHelper", "No furniture items found") // Логируем отсутствие данных  
 }  
 **it**.close()  
 **}**  
db.close()  
 return furnitureItems  
 }  
 @SuppressLint("Range")  
 fun getAllDetails(): List<Detail> {  
 val details = *mutableListOf*<Detail>()  
 val db = *readableDatabase*  
val cursor: Cursor? = db.rawQuery("SELECT detail\_id, name FROM Details", null) // Извлекаем только id и имя  
  
 cursor?.*let* **{**  
if (**it**.moveToFirst()) {  
 do {  
 val detailId = **it**.getInt(**it**.getColumnIndexOrThrow("detail\_id"))  
 val name = **it**.getString(**it**.getColumnIndexOrThrow("name"))  
 details.add(Detail(detailId, name)) // Создаем объект Detail с id и именем  
 } while (**it**.moveToNext())  
 }  
 **it**.close()  
 **}**  
db.close()  
 return details  
 }  
  
  
 override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {  
 if (oldVersion < 2) { // Если вы обновляете с версии 1 до 2  
 db.execSQL("ALTER TABLE Orders ADD COLUMN furniture\_type\_id INTEGER")  
 }  
 db.execSQL("DROP TABLE IF EXISTS Orders")  
 db.execSQL("DROP TABLE IF EXISTS PartCharacteristics")  
 db.execSQL("DROP TABLE IF EXISTS Parts")  
 db.execSQL("DROP TABLE IF EXISTS PartTypes")  
 db.execSQL("DROP TABLE IF EXISTS FurnitureItems")  
 db.execSQL("DROP TABLE IF EXISTS FurnitureTypes")  
 db.execSQL("DROP TABLE IF EXISTS Clients")  
 onCreate(db)  
 }  
  
  
  
 @SuppressLint("Range")  
 fun getAllPartTypes(): List<PartType> {  
 val partTypes = *mutableListOf*<PartType>()  
 val db = *readableDatabase*  
val cursor: Cursor? = db.rawQuery("SELECT \* FROM PartTypes", null)  
  
 cursor?.*let* **{**  
if (**it**.moveToFirst()) {  
 do {  
 val partTypeId = **it**.getInt(**it**.getColumnIndexOrThrow("part\_type\_id"))  
 val name = **it**.getString(**it**.getColumnIndexOrThrow("name"))  
 partTypes.add(PartType(partTypeId, name)) // Предполагается, что у вас есть класс PartType  
 } while (**it**.moveToNext())  
 }  
 **it**.close()  
 **}**  
db.close()  
 return partTypes  
 }  
 @SuppressLint("Range")  
 fun getFurnitureItemById(itemId: Int): Pair<FurnitureItem?, List<PartCharacteristics>> {  
 val db = *readableDatabase*  
val cursor: Cursor? = db.rawQuery("""  
 SELECT FurnitureItems.furniture\_item\_id, FurnitureItems.model, FurnitureTypes.name AS furniture\_type\_name   
FROM FurnitureItems   
INNER JOIN FurnitureTypes ON FurnitureItems.furniture\_type\_id = FurnitureTypes.furniture\_type\_id  
 WHERE FurnitureItems.furniture\_item\_id = ?  
 """, *arrayOf*(itemId.toString()))  
  
 var furnitureItem: FurnitureItem? = null  
 cursor?.*let* **{**  
if (**it**.moveToFirst()) {  
 val furnitureItemId = **it**.getInt(**it**.getColumnIndexOrThrow("furniture\_item\_id"))  
 val model = **it**.getString(**it**.getColumnIndexOrThrow("model"))  
 val furnitureTypeName = **it**.getString(**it**.getColumnIndexOrThrow("furniture\_type\_name"))  
  
 furnitureItem = FurnitureItem(furnitureItemId, model, furnitureTypeName)  
 }  
 **it**.close() // Закрываем курсор  
 **}**  
db.close() // Закрываем базу данных  
  
 // Получение деталей для предмета мебели  
 val partsCharacteristics = getPartsForFurnitureItem(itemId)  
  
 return Pair(furnitureItem, partsCharacteristics)  
 }  
 fun addDetail(partTypeId: Int, name: String): Long {  
 val db = *writableDatabase*  
val values = ContentValues().*apply* **{**  
put("part\_id", partTypeId) // Указываем id типа детали  
 put("name", name) // Указываем имя детали  
 **}**  
return db.insert("Details", null, values) // Вставляем данные в таблицу Details  
 }  
  
 @SuppressLint("Range")  
 fun getPartsForFurnitureItem(furnitureItemId: Int): List<PartCharacteristics> {  
 val partsCharacteristics = *mutableListOf*<PartCharacteristics>()  
 val db = *readableDatabase*  
val cursor: Cursor? = db.rawQuery("""  
 SELECT DISTINCT PartCharacteristics.material   
FROM Parts   
INNER JOIN PartCharacteristics ON Parts.part\_id = PartCharacteristics.part\_id  
 INNER JOIN FurnitureItems ON FurnitureItems.furniture\_item\_id = ?  
 """, *arrayOf*(furnitureItemId.toString()))  
  
 cursor?.*let* **{**  
if (**it**.moveToFirst()) {  
 do {  
 val material = **it**.getString(**it**.getColumnIndexOrThrow("material"))  
 // Добавляем только материал, без других характеристик  
 partsCharacteristics.add(PartCharacteristics(0, 0, "", 0.0, material, 0.0))  
 } while (**it**.moveToNext())  
 }  
 **it**.close() // Закрываем курсор  
 **}**  
db.close() // Закрываем базу данных  
 return partsCharacteristics  
 }  
  
 fun deleteDetail(detailId: Int): Boolean {  
 val db = *writableDatabase*  
val result = db.delete("Details", "detail\_id = ?", *arrayOf*(detailId.toString()))  
 db.close()  
 return result > 0 // Возвращает true, если удаление прошло успешно  
 }  
  
 // Методы для управления заказами  
 fun addOrder(clientId: Int, furnitureTypeId: Int, orderDate: String, status: String): Long {  
 val db = *writableDatabase*  
val values = ContentValues().*apply* **{**  
put("client\_id", clientId)  
 put("furniture\_type\_id", furnitureTypeId)  
 put("order\_date", orderDate)  
 put("status", status)  
 **}**  
  
// Логируем данные, которые мы пытаемся вставить  
 Log.d("DatabaseHelper", "Adding order: clientId=$clientId, furnitureTypeId=$furnitureTypeId, orderDate=$orderDate, status=$status")  
  
 val newRowId = db.insert("Orders", null, values)  
 db.close()  
  
 // Проверяем, был ли добавлен заказ  
 if (newRowId == -1L) {  
 Log.e("DatabaseHelper", "Failed to insert order")  
 } else {  
 Log.d("DatabaseHelper", "Order added with ID: $newRowId")  
 }  
  
 return newRowId  
 }  
  
 @SuppressLint("Range")  
 fun getAllOrders(): List<Order> {  
 val orders = *mutableListOf*<Order>()  
 val db = *readableDatabase*  
val cursor: Cursor? = db.rawQuery("SELECT \* FROM Orders", null)  
  
 cursor?.*let* **{**  
if (**it**.moveToFirst()) {  
 do {  
 val orderId = **it**.getInt(**it**.getColumnIndexOrThrow("order\_id"))  
 val clientId = **it**.getInt(**it**.getColumnIndexOrThrow("client\_id"))  
 val furnitureTypeId = **it**.getInt(**it**.getColumnIndexOrThrow("furniture\_type\_id"))  
 val orderDate = **it**.getString(**it**.getColumnIndexOrThrow("order\_date"))  
 val status = **it**.getString(**it**.getColumnIndexOrThrow("status"))  
  
 orders.add(Order(orderId, clientId, furnitureTypeId, orderDate, status))  
 } while (**it**.moveToNext())  
 }  
 **it**.close() // Закрываем курсор  
 **}**  
db.close() // Закрываем базу данных  
 return orders  
 }  
  
 fun addUser (login: String, password: String, email: String, role: String): Long {  
 val db = *writableDatabase*  
val values = ContentValues().*apply* **{**  
put("login", login)  
 put("password", password)  
 put("email", email)  
 put("role", role)  
 **}**  
val newRowId = db.insert("Users", null, values)  
 db.close()  
 return newRowId  
 }  
  
 // Метод для проверки существующего логина  
 fun isLoginExists(login: String): Boolean {  
 val db = *readableDatabase*  
val cursor: Cursor? = db.rawQuery("SELECT \* FROM Users WHERE login = ?", *arrayOf*(login))  
 val exists = cursor != null && cursor.*count* > 0  
 cursor?.close()  
 db.close()  
 return exists  
 }  
 fun checkUser (login: String, password: String, role: String): Boolean {  
 val db = *readableDatabase*  
val cursor: Cursor? = db.rawQuery(  
 "SELECT \* FROM Users WHERE login = ? AND password = ? AND role = ?",  
 *arrayOf*(login, password, role)  
 )  
 val exists = cursor != null && cursor.*count* > 0  
 cursor?.close()  
 db.close()  
 return exists  
 }  
  
 @SuppressLint("Range")  
 fun getFurnitureTypeById(furnitureTypeId: Int): FurnitureType? {  
 val db = *readableDatabase*  
val cursor: Cursor? = db.rawQuery("SELECT \* FROM FurnitureTypes WHERE furniture\_type\_id = ?", *arrayOf*(furnitureTypeId.toString()))  
  
 val furnitureType: FurnitureType? = if (cursor != null && cursor.moveToFirst()) {  
 val name = cursor.getString(cursor.getColumnIndexOrThrow("name"))  
 FurnitureType(furnitureTypeId, name) // Убедитесь, что у вас есть класс FurnitureType  
 } else {  
 null  
 }  
  
 cursor?.close()  
 db.close()  
 return furnitureType  
 }  
  
 @SuppressLint("Range")  
 fun getClientIdByLogin(login: String): Int? {  
 val db = this.*readableDatabase*  
val cursor = db.rawQuery("SELECT user\_id FROM Users WHERE login = ?", *arrayOf*(login))  
  
 return if (cursor.moveToFirst()) {  
 val clientId = cursor.getInt(cursor.getColumnIndex("user\_id"))  
 cursor.close()  
 clientId  
 } else {  
 cursor.close()  
 null // Если клиент не найден  
 }  
 }  
  
 fun deleteOrder(orderId: Int): Boolean {  
 val db = *writableDatabase*  
val result = db.delete("Orders", "order\_id = ?", *arrayOf*(orderId.toString()))  
 db.close()  
 return result > 0 // Возвращает true, если удаление прошло успешно  
 }  
 @SuppressLint("Range")  
 fun getAllFurnitureItems(): List<FurnitureItem> {  
 val furnitureItems = *mutableListOf*<FurnitureItem>()  
 val db = *readableDatabase*  
val cursor: Cursor? = db.rawQuery("""  
 SELECT FurnitureItems.furniture\_item\_id, FurnitureItems.model, FurnitureTypes.name AS furniture\_type\_name   
FROM FurnitureItems   
INNER JOIN FurnitureTypes ON FurnitureItems.furniture\_type\_id = FurnitureTypes.furniture\_type\_id  
 """, null)  
  
 cursor?.*let* **{**  
if (**it**.moveToFirst()) {  
 do {  
 val furnitureItemId = **it**.getInt(**it**.getColumnIndexOrThrow("furniture\_item\_id"))  
 val model = **it**.getString(**it**.getColumnIndexOrThrow("model"))  
 val furnitureTypeName = **it**.getString(**it**.getColumnIndexOrThrow("furniture\_type\_name"))  
  
 furnitureItems.add(FurnitureItem(furnitureItemId, model, furnitureTypeName))  
 } while (**it**.moveToNext())  
 } else {  
 Log.d("DatabaseHelper", "No furniture items found") // Логируем отсутствие данных  
 }  
 **it**.close()  
 **}**  
db.close()  
 return furnitureItems  
 }  
  
  
  
}  
  
// Классы для представления данных  
data class Order(val orderId: Int, val clientId: Int, val furnitureTypeId: Int, val orderDate: String, val status: String)  
data class FurnitureItem(val furnitureItemId: Int, val model: String, val furnitureTypeName: String)  
data class FurnitureType(val furnitureTypeId: Int, val name: String)  
data class PartType(val partTypeId: Int, val name: String) {  
 override fun toString(): String {  
 return name // Это будет отображаться в Spinner  
 }  
}  
data class Detail(  
 val detailId: Int,  
 val name: String  
)  
data class PartCharacteristic(  
 val partId: Long,  
 val weight: Double,  
 val material: String,  
 val diameter: Double  
)  
data class PartCharacteristics(  
 val partId: Int,  
 val partTypeId: Int,  
 val name: String,  
 val weight: Double,  
 val material: String,  
 val diameter: Double  
)

Код программы:

package com.example.mebel  
  
import android.content.Intent  
import android.os.Bundle  
import android.text.Editable  
import android.text.TextWatcher  
import android.view.View  
import android.widget.EditText  
import android.widget.ListView  
import androidx.appcompat.app.AppCompatActivity  
import data.DatabaseHelper  
import data.FurnitureItem  
  
class ClientMainActivity : AppCompatActivity() {  
  
 private lateinit var furnitureAdapter: FurnitureAdapter  
 private lateinit var allFurnitureItems: List<FurnitureItem> // Сохраняем все предметы мебели  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_client\_main*)  
  
 // Получение данных о мебели  
 val dbHelper = DatabaseHelper(this)  
 allFurnitureItems = dbHelper.getAllFurnitureItems() // Получите все элементы мебели  
 furnitureAdapter = FurnitureAdapter(this, allFurnitureItems)  
  
 // Установка адаптера для ListView  
 val listView: ListView = findViewById(R.id.*furnitureRecyclerView*)  
 listView.*adapter* = furnitureAdapter  
  
 // Обработка нажатий на элементы списка  
 listView.setOnItemClickListener **{** \_, \_, position, \_ **->**  
val selectedItem = furnitureAdapter.getItem(position) as FurnitureItem  
 val intent = Intent(this, ItemDetailsActivity::class.*java*)  
 intent.putExtra("ITEM\_ID", selectedItem.furnitureItemId)  
 startActivity(intent)  
 **}**  
  
// Настройка поля поиска  
 val searchEditText: EditText = findViewById(R.id.*searchEditText*)  
 searchEditText.addTextChangedListener(object : TextWatcher {  
 override fun afterTextChanged(s: Editable?) {  
 val query = s.*toString*().*toLowerCase*()  
 filterFurnitureItems(query)  
 }  
  
 override fun beforeTextChanged(s: CharSequence?, start: Int, count: Int, after: Int) {}  
 override fun onTextChanged(s: CharSequence?, start: Int, before: Int, count: Int) {}  
 })  
 }  
  
 private fun filterFurnitureItems(query: String) {  
 val filteredItems = allFurnitureItems.*filter* **{**  
 **it**.model.*toLowerCase*().*contains*(query)  
 **}**  
furnitureAdapter.updateData(filteredItems)  
 }  
  
 fun oformit(view: View) {  
 startActivity(Intent(this, ClientOrderActivity::class.*java*))  
 }  
}

package com.example.mebel  
  
import androidx.appcompat.app.AppCompatActivity  
import android.os.Bundle  
import android.util.Log  
import android.widget.ArrayAdapter  
import android.widget.Button  
import android.widget.EditText  
import android.widget.Spinner  
import android.widget.Toast  
import data.DatabaseHelper  
import java.text.SimpleDateFormat  
import java.util.Locale  
  
class ClientOrderActivity : AppCompatActivity() {  
 private lateinit var databaseHelper: DatabaseHelper  
 private lateinit var furnitureTypeSpinner: Spinner  
 private lateinit var orderButton: Button  
 private lateinit var clientIdEditText: EditText  
 private lateinit var orderDateEditText: EditText  
 private lateinit var statusEditText: EditText  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_client\_order*)  
  
 databaseHelper = DatabaseHelper(this)  
  
 furnitureTypeSpinner = findViewById(R.id.*furnitureTypeSpinner*)  
 orderButton = findViewById(R.id.*orderButton*)  
 orderDateEditText = findViewById(R.id.*orderDateEditText*)  
 statusEditText = findViewById(R.id.*statusEditText*)  
  
 // Получаем ID клиента из SharedPreferences  
 val sharedPreferences = getSharedPreferences("user\_prefs", *MODE\_PRIVATE*)  
 val clientId = sharedPreferences.getInt("user\_id", -1) // -1 - значение по умолчанию, если не найден  
  
 // Заполняем Spinner доступными типами мебели  
 loadFurnitureTypes()  
  
 orderButton.setOnClickListener **{**  
val selectedItem = furnitureTypeSpinner.*selectedItem*.toString()  
 val furnitureTypeId = selectedItem.*split*(":")[0].*trim*().*toInt*() // Получаем ID выбранного типа мебели  
 val orderDate = orderDateEditText.*text*.toString()  
 val status = statusEditText.*text*.toString()  
  
 if (clientId != -1 && furnitureTypeId != -1 && orderDate.*isNotEmpty*() && status.*isNotEmpty*()) {  
 // Проверка корректности даты  
 if (isValidDate(orderDate)) {  
 try {  
 val newOrderId = databaseHelper.addOrder(clientId, furnitureTypeId, orderDate, status)  
 Toast.makeText(this, "Заказ оформлен", Toast.*LENGTH\_SHORT*).show()  
 } catch (e: Exception) {  
 Log.e("ClientOrderActivity", "Error adding order: ${e.message}")  
 Toast.makeText(this, "Ошибка при добавлении заказа: ${e.message}", Toast.*LENGTH\_SHORT*).show()  
 }  
 } else {  
 Toast.makeText(this, "Введите корректную дату в формате дд.мм.гггг", Toast.*LENGTH\_SHORT*).show()  
 }  
 } else {  
 Toast.makeText(this, "Пожалуйста, заполните все поля корректно", Toast.*LENGTH\_SHORT*).show()  
 }  
 **}**  
}  
  
 private fun loadFurnitureTypes() {  
 val allFurnitureItems = databaseHelper.getAllFurnitureItemsWithIds() // Получаем все предметы мебели с их ID  
 val adapter = ArrayAdapter(this, android.R.layout.*simple\_spinner\_item*, allFurnitureItems.*map* **{** "${**it**.first}: ${**it**.second}" **}**) // Форматируем строку как "ID: Название"  
 adapter.setDropDownViewResource(android.R.layout.*simple\_spinner\_dropdown\_item*)  
 furnitureTypeSpinner.*adapter* = adapter  
 }  
  
 private fun isValidDate(date: String): Boolean {  
 return try {  
 val format = SimpleDateFormat("dd.MM.yyyy", Locale.getDefault())  
 format.*isLenient* = false // Установите в false, чтобы исключить нечеткие даты  
 format.parse(date) != null // Если дата корректная, метод parse вернет непустое значение  
 } catch (e: Exception) {  
 false // Если возникло исключение, дата некорректная  
 }  
 }  
}

package com.example.mebel  
  
import android.content.Context  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import android.widget.ArrayAdapter  
import android.widget.BaseAdapter  
import android.widget.TextView  
import data.FurnitureItem  
  
class FurnitureAdapter(private val context: Context, private var furnitureItems: List<FurnitureItem>) : BaseAdapter() {  
  
 override fun getCount(): Int = furnitureItems.size  
  
 override fun getItem(position: Int): Any = furnitureItems[position]  
  
 override fun getItemId(position: Int): Long = position.toLong()  
  
 override fun getView(position: Int, convertView: View?, parent: ViewGroup): View {  
 val itemView = convertView ?: LayoutInflater.from(context).inflate(R.layout.*item\_furniture*, parent, false)  
 val furnitureItem = getItem(position) as FurnitureItem  
 val modelTextView = itemView.findViewById<TextView>(R.id.*modelTextView*)  
 modelTextView.*text* = furnitureItem.model  
 return itemView  
 }  
  
 fun updateData(newData: List<FurnitureItem>) {  
 furnitureItems = newData  
 notifyDataSetChanged()  
 }  
}

package com.example.mebel  
  
import android.os.Bundle  
import android.util.Log  
import android.widget.ListView  
import android.widget.TextView  
import androidx.appcompat.app.AppCompatActivity  
import data.DatabaseHelper  
import data.PartCharacteristics  
  
class ItemDetailsActivity : AppCompatActivity() {  
  
 private lateinit var partsAdapter: PartsAdapter  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_item\_details*)  
  
 val itemId = *intent*.getIntExtra("ITEM\_ID", -1)  
  
 val dbHelper = DatabaseHelper(this)  
 // Получаем предмет мебели и его детали  
 val (furnitureItem, parts) = dbHelper.getFurnitureItemById(itemId)  
  
 val itemDetailsTextView: TextView = findViewById(R.id.itemDetailsTextView)  
 itemDetailsTextView.text = furnitureItem?.model ?: "Неизвестный предмет"  
 Log.d("ItemDetailsActivity", "Parts Count: ${parts.size}")  
 // Убедитесь, что parts - это список PartCharacteristics  
 val partsListView: ListView = findViewById(R.id.partsListView)  
 partsAdapter = PartsAdapter(this, parts) // Здесь parts - это список PartCharacteristics  
 partsListView.adapter = partsAdapter  
 }  
}

package com.example.mebel  
  
import androidx.appcompat.app.AppCompatActivity  
import android.os.Bundle  
import android.content.Intent  
import android.widget.\*  
import data.DatabaseHelper  
  
class MainActivity : AppCompatActivity() {  
 private lateinit var databaseHelper: DatabaseHelper  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.activity\_main)  
  
 val loginButton: Button = findViewById(R.id.loginButton)  
 val registerButton: Button = findViewById(R.id.registerButton)  
 val roleSpinner: Spinner = findViewById(R.id.roleSpinner)  
 databaseHelper = DatabaseHelper(this)  
  
 // Создаем массив ролей  
 val roles = arrayOf("Работник", "Клиент", "Поставщик")  
  
 // Создаем адаптер для Spinner  
 val adapter = ArrayAdapter(this, android.R.layout.simple\_spinner\_item, roles)  
 adapter.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item)  
 roleSpinner.adapter = adapter  
  
 loginButton.setOnClickListener **{**  
val login = findViewById<EditText>(R.id.loginEditText).text.toString()  
 val password = findViewById<EditText>(R.id.passwordEditText).text.toString()  
 val role = roleSpinner.selectedItem.toString()  
  
 if (login.isNotEmpty() && password.isNotEmpty()) {  
 // Проверяем пользователя  
 if (databaseHelper.checkUser (login, password, role)) {  
 // Получаем ID пользователя  
 val userId = databaseHelper.getClientIdByLogin(login) // Убедитесь, что этот метод возвращает правильный ID для всех ролей  
  
 // Сохраняем ID в SharedPreferences  
 val sharedPreferences = getSharedPreferences("user\_prefs", MODE\_PRIVATE)  
 with(sharedPreferences.edit()) **{**  
putInt("user\_id", userId ?: -1) // Сохраняем ID или -1, если не найден  
 apply()  
 **}**  
  
// Переход на экран работника/клиента/поставщика в зависимости от роли  
 val intent = when (role) {  
 "Работник" -> Intent(this, WorkerMainActivity::class.java)  
 "Клиент" -> Intent(this, ClientMainActivity::class.java)  
 "Поставщик" -> Intent(this, SupplierMainActivity::class.java)  
 else -> null  
 }  
  
 // Запускаем активити, если intent не null  
 intent?.let **{**  
startActivity(it)  
 **}**  
  
} else {  
 Toast.makeText(this, "Неверный логин или пароль", Toast.LENGTH\_SHORT).show()  
 }  
 } else {  
 Toast.makeText(this, "Заполните все поля", Toast.LENGTH\_SHORT).show()  
 }  
 **}**  
  
registerButton.setOnClickListener **{**  
startActivity(Intent(this, RegistrationActivity::class.java))  
 **}**  
}  
}

package com.example.mebel  
  
import android.content.Context  
import android.util.Log  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import android.widget.ArrayAdapter  
import android.widget.TextView  
import data.PartCharacteristics  
  
class PartsAdapter(context: Context, private val parts: List<PartCharacteristics>) : ArrayAdapter<PartCharacteristics>(context, 0, parts) {  
  
 override fun getView(position: Int, convertView: View?, parent: ViewGroup): View {  
 val part = getItem(position)  
 val view = convertView ?: LayoutInflater.from(*context*).inflate(R.layout.*part\_item*, parent, false)  
  
 val partMaterialTextView: TextView = view.findViewById(R.id.*partMaterialTextView*)  
 Log.d("PartsAdapter", "Displaying Part Material: ${part?.material}")  
 // Отображаем только материал детали  
 partMaterialTextView.*text* = part?.material ?: "Неизвестный материал"  
  
 return view  
 }  
}

package com.example.mebel  
  
import android.annotation.SuppressLint  
import androidx.appcompat.app.AppCompatActivity  
import android.os.Bundle  
import android.widget.ArrayAdapter  
import android.widget.Button  
import android.widget.EditText  
import android.widget.Spinner  
import android.widget.Toast  
import data.DatabaseHelper  
  
class RegistrationActivity : AppCompatActivity() {  
 private lateinit var databaseHelper: DatabaseHelper  
  
 @SuppressLint("CutPasteId")  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.activity\_registration)  
 databaseHelper = DatabaseHelper(this)  
  
 val registerSubmitButton: Button = findViewById(R.id.registerSubmitButton)  
 val roleSpinner: Spinner = findViewById(R.id.registerRoleSpinner)  
 val roles = arrayOf("Работник", "Клиент", "Поставщик")  
 val adapter = ArrayAdapter(this, android.R.layout.simple\_spinner\_item, roles)  
 adapter.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item)  
 roleSpinner.adapter = adapter  
 registerSubmitButton.setOnClickListener **{**  
val login = findViewById<EditText>(R.id.registerLoginEditText).text.toString()  
 val password = findViewById<EditText>(R.id.registerPasswordEditText).text.toString()  
 val email = findViewById<EditText>(R.id.registerEmailEditText).text.toString()  
 val role = findViewById<Spinner>(R.id.registerRoleSpinner).selectedItem.toString()  
 if (login.isNotEmpty() && password.isNotEmpty() && email.isNotEmpty()) {  
 if (databaseHelper.isLoginExists(login)) {  
 Toast.makeText(this, "Логин уже существует", Toast.LENGTH\_SHORT).show()  
 } else {  
 // Логика регистрации пользователя  
 databaseHelper.addUser (login, password, email, role)  
 Toast.makeText(this, "Пользователь зарегистрирован ", Toast.LENGTH\_SHORT).show()  
 finish() // Закрыть экран регистрации  
 }  
 } else {  
 Toast.makeText(this, "Заполните все поля", Toast.LENGTH\_SHORT).show()  
 }  
 **}**  
}  
}

package com.example.mebel  
  
import android.os.Bundle  
import android.widget.ArrayAdapter  
import android.widget.Button  
import android.widget.EditText  
import android.widget.Spinner  
import android.widget.Toast  
import androidx.appcompat.app.AppCompatActivity  
import data.DatabaseHelper  
import data.PartType  
  
class SupplierDetailsActivity : AppCompatActivity() {  
  
 private lateinit var dbHelper: DatabaseHelper  
 private lateinit var partTypeSpinner: Spinner  
 private lateinit var detailNameEditText: EditText  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.activity\_supplier\_details)  
  
 dbHelper = DatabaseHelper(this)  
 partTypeSpinner = findViewById(R.id.partTypeSpinner)  
 detailNameEditText = findViewById(R.id.detailNameEditText)  
 val addButton: Button = findViewById(R.id.addButton)  
  
 // Загружаем типы деталей в Spinner  
 loadPartTypes()  
  
 addButton.setOnClickListener **{**  
val name = detailNameEditText.text.toString()  
 val selectedPartType = partTypeSpinner.selectedItem as PartType  
  
 if (name.isNotEmpty()) {  
 // Добавляем деталь в новую таблицу Details  
 val newDetailId = dbHelper.addDetail(selectedPartType.partTypeId, name) // Обновите метод для добавления детали  
 if (newDetailId != -1L) {  
 Toast.makeText(this, "Деталь добавлена: $name", Toast.LENGTH\_SHORT).show()  
 finish() // Закрыть экран после добавления детали  
 } else {  
 Toast.makeText(this, "Ошибка при добавлении детали", Toast.LENGTH\_SHORT).show()  
 }  
 } else {  
 Toast.makeText(this, "Введите название детали", Toast.LENGTH\_SHORT).show()  
 }  
 **}**  
}  
  
 private fun loadPartTypes() {  
 val partTypes = dbHelper.getAllPartTypes() // Получаем все типы деталей из БД  
 val adapter = ArrayAdapter(this, android.R.layout.simple\_spinner\_item, partTypes)  
 adapter.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item)  
 partTypeSpinner.adapter = adapter  
 }  
}

package com.example.mebel  
  
import android.content.Intent  
import android.os.Bundle  
import android.view.View  
import android.widget.ArrayAdapter  
import android.widget.ListView  
import android.widget.Toast  
import androidx.appcompat.app.AppCompatActivity  
import data.DatabaseHelper  
import data.Detail // Убедитесь, что вы импортировали класс Detail  
  
class SupplierMainActivity : AppCompatActivity() {  
  
 private lateinit var dbHelper: DatabaseHelper  
 private lateinit var detailsListView: ListView  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.activity\_supplier\_main)  
  
 dbHelper = DatabaseHelper(this)  
 detailsListView = findViewById(R.id.detailsListView)  
  
 loadDetails() // Загружаем детали при создании активности  
 }  
  
 private fun loadDetails() {  
 val details = dbHelper.getAllDetails() // Получаем все детали из базы данных  
 val adapter = ArrayAdapter(this, android.R.layout.simple\_list\_item\_1, details.map **{** it.name **}**)  
 detailsListView.adapter = adapter  
  
 detailsListView.setOnItemClickListener **{** \_, \_, position, \_ **->**  
val selectedDetail = details[position]  
 Toast.makeText(this, "Выбранная деталь: ${selectedDetail.name}", Toast.LENGTH\_SHORT).show()  
  
 // Удаление детали  
 val isDeleted = dbHelper.deleteDetail(selectedDetail.detailId) // Используем detailId для удаления  
 if (isDeleted) {  
 Toast.makeText(this, "Деталь удалена", Toast.LENGTH\_SHORT).show()  
 loadDetails() // Обновляем список после удаления  
 } else {  
 Toast.makeText(this, "Ошибка при удалении детали", Toast.LENGTH\_SHORT).show()  
 }  
 **}**  
}  
  
 fun ADD(view: View) {  
 startActivity(Intent(this, SupplierDetailsActivity::class.java))  
 }  
}

package com.example.mebel  
  
import androidx.appcompat.app.AppCompatActivity  
import android.os.Bundle  
import android.view.View  
import android.widget.AdapterView  
import android.widget.ArrayAdapter  
import android.widget.ListView  
import android.widget.Toast  
import data.DatabaseHelper  
import data.Order  
  
class WorkerMainActivity : AppCompatActivity() {  
 private lateinit var databaseHelper: DatabaseHelper  
 private lateinit var ordersListView: ListView  
 private lateinit var orderDescriptions: List<String>  
 private lateinit var orders: List<Order> // Предполагается, что Order - это ваша модель данных заказа  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.activity\_worker\_main)  
  
 databaseHelper = DatabaseHelper(this)  
 ordersListView = findViewById(R.id.ordersRecyclerView)  
  
 // Загрузка заказов при создании активности  
 loadOrders()  
  
 // Обработчик нажатия на элемент списка  
 ordersListView.setOnItemClickListener **{** parent, view, position, id **->**  
val selectedOrder = orders[position]  
 deleteOrder(selectedOrder)  
 **}**  
}  
  
 // Метод для загрузки заказов  
 private fun loadOrders() {  
 orders = databaseHelper.getAllOrders() // Получаем все заказы  
 orderDescriptions = orders.map **{** order **->**  
val furnitureType = databaseHelper.getFurnitureTypeById(order.furnitureTypeId)  
 val furnitureTypeName = furnitureType?.name ?: "Неизвестный тип"  
 "Order ID: ${order.orderId}, Status: ${order.status}, Furniture Type: $furnitureTypeName, Order Date: ${order.orderDate}"  
 **}**  
  
val adapter = ArrayAdapter(this, android.R.layout.simple\_list\_item\_1, orderDescriptions)  
 ordersListView.adapter = adapter  
 }  
  
 // Метод для удаления заказа  
 private fun deleteOrder(order: Order) {  
 // Удаление заказа из базы данных  
 val isDeleted = databaseHelper.deleteOrder(order.orderId) // Предполагается, что вы добавите этот метод в DatabaseHelper  
  
 if (isDeleted) {  
 Toast.makeText(this, "Заказ с удален", Toast.LENGTH\_SHORT).show()  
 loadOrders() // Обновление списка заказов  
 } else {  
 Toast.makeText(this, "Ошибка при удалении заказа", Toast.LENGTH\_SHORT).show()  
 }  
 }  
}

<?xml version="1.0" encoding="utf-8"?>  
<shape xmlns:android="http://schemas.android.com/apk/res/android">  
 <solid android:color="@android:color/white"/>  
 <corners android:radius="4dp"/>  
 <stroke android:width="1dp" android:color="@android:color/darker\_gray"/>  
</shape>

<?xml version="1.0" encoding="utf-8"?>  
<shape xmlns:android="http://schemas.android.com/apk/res/android">  
 <solid android:color="@android:color/white"/>  
 <corners android:radius="4dp"/>  
 <stroke android:width="1dp" android:color="@android:color/darker\_gray"/>  
</shape>

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 android:padding="16dp"  
 android:background="@android:color/white">  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:textSize="24sp"  
 android:text="Каталог мебели"  
 android:textColor="@android:color/black"  
 android:paddingBottom="16dp"/>  
  
 <EditText  
 android:id="@+id/searchEditText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Поиск по мебели"  
 android:background="@drawable/edit\_text\_background"  
 android:padding="10dp"/>  
  
 <ListView  
 android:id="@+id/furnitureRecyclerView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="0dp"  
 android:layout\_weight="1"  
 android:divider="@android:color/darker\_gray"  
 android:dividerHeight="1dp"/>  
  
 <Button  
 android:id="@+id/orderButton"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Оформить заказ"  
 android:onClick="oformit"  
 android:background="@color/colorAccent"  
 android:textColor="@android:color/white"  
 android:padding="10dp"  
 android:layout\_marginTop="16dp"/>  
</LinearLayout>

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 android:padding="16dp"  
 android:background="@android:color/white">  
  
 <Spinner  
 android:id="@+id/furnitureTypeSpinner"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:background="@drawable/spinner\_background"  
 android:layout\_marginTop="150dp"  
 android:padding="10dp"/>  
  
 <EditText  
 android:id="@+id/orderDateEditText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Введите дату заказа (дд.мм.гггг)"  
 android:background="@drawable/edit\_text\_background"  
 android:padding="10dp"  
 android:layout\_marginTop="16dp"/>  
  
 <EditText  
 android:id="@+id/statusEditText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Введите статус заказа"  
 android:background="@drawable/edit\_text\_background"  
 android:padding="10dp"  
 android:layout\_marginTop="16dp"/>  
  
 <Button  
 android:id="@+id/orderButton"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Оформить заказ"  
 android:background="@color/colorAccent"  
 android:textColor="@android:color/white"  
 android:padding="10dp"  
 android:layout\_marginTop="200dp"/>  
</LinearLayout>

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 android:padding="16dp"  
 android:background="@android:color/white">  
  
 <TextView  
 android:id="@+id/itemDetailsTextView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:textSize="24sp"  
 android:textColor="@android:color/black"  
 android:paddingBottom="16dp"/>  
  
 <ListView  
 android:id="@+id/partsListView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="0dp"  
 android:layout\_weight="1"  
 android:divider="@android:color/darker\_gray"  
 android:dividerHeight="1dp"/>  
</LinearLayout>

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 android:padding="16dp"  
 android:background="@android:color/white">  
  
 <EditText  
 android:id="@+id/loginEditText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Логин"  
 android:background="@drawable/edit\_text\_background"  
 android:layout\_marginTop="200dp"  
 android:padding="10dp"/>  
  
 <EditText  
 android:id="@+id/passwordEditText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Пароль"  
 android:inputType="textPassword"  
 android:background="@drawable/edit\_text\_background"  
 android:padding="10dp"  
 android:layout\_marginTop="16dp"/>  
  
 <Spinner  
 android:id="@+id/roleSpinner"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:background="@drawable/spinner\_background"  
 android:padding="10dp"  
 android:layout\_marginTop="16dp"/>  
  
 <Button  
 android:id="@+id/loginButton"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Войти"  
 android:background="@color/colorAccent"  
 android:textColor="@android:color/white"  
 android:padding="10dp"  
 android:layout\_marginTop="16dp"/>  
  
 <Button  
 android:id="@+id/registerButton"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Зарегистрироваться"  
 android:background="@color/colorAccent"  
 android:textColor="@android:color/white"  
 android:padding="10dp"  
 android:layout\_marginTop="16dp"/>  
</LinearLayout>

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 android:padding="16dp"  
 android:background="@android:color/white">  
  
 <EditText  
 android:id="@+id/registerLoginEditText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Логин"  
 android:background="@drawable/edit\_text\_background"  
 android:layout\_marginTop="200dp"  
 android:padding="10dp"/>  
  
 <EditText  
 android:id="@+id/registerPasswordEditText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Пароль"  
 android:inputType="textPassword"  
 android:background="@drawable/edit\_text\_background"  
 android:padding="10dp"  
 android:layout\_marginTop="16dp"/>  
  
 <EditText  
 android:id="@+id/registerEmailEditText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Email"  
 android:background="@drawable/edit\_text\_background"  
 android:padding="10dp"  
 android:layout\_marginTop="16dp"/>  
  
 <Spinner  
 android:id="@+id/registerRoleSpinner"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:background="@drawable/spinner\_background"  
 android:padding="10dp"  
 android:layout\_marginTop="16dp"/>  
  
 <Button  
 android:id="@+id/registerSubmitButton"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Зарегистрироваться"  
 android:background="@color/colorAccent"  
 android:textColor="@android:color/white"  
 android:padding="10dp"  
 android:layout\_marginTop="16dp"/>  
</LinearLayout>

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 android:padding="16dp"  
 android:background="@android:color/white">  
  
 <Spinner  
 android:id="@+id/partTypeSpinner"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:background="@drawable/spinner\_background"  
 android:layout\_marginTop="150dp"  
 android:padding="10dp"/>  
  
 <EditText  
 android:id="@+id/detailNameEditText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Введите название детали"  
 android:background="@drawable/edit\_text\_background"  
 android:padding="10dp"  
 android:layout\_marginTop="16dp"/>  
  
 <Button  
 android:id="@+id/addButton"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Добавить"  
 android:background="@color/colorAccent"  
 android:textColor="@android:color/white"  
 android:padding="10dp"  
 android:layout\_marginTop="260dp"/>  
</LinearLayout>

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 android:padding="16dp"  
 android:background="@android:color/white">  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Список поставляемых деталей"  
 android:textSize="24sp"  
 android:textColor="@android:color/black"  
 android:paddingBottom="16dp"/>  
  
 <ListView  
 android:id="@+id/detailsListView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="0dp"  
 android:layout\_weight="1"  
 android:divider="@android:color/darker\_gray"  
 android:dividerHeight="1dp"/>  
  
 <Button  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:onClick="ADD"  
 android:text="Добавить"  
 android:background="@color/colorAccent"  
 android:textColor="@android:color/white"  
 android:padding="10dp"  
 android:layout\_marginTop="16dp"/>  
</LinearLayout>

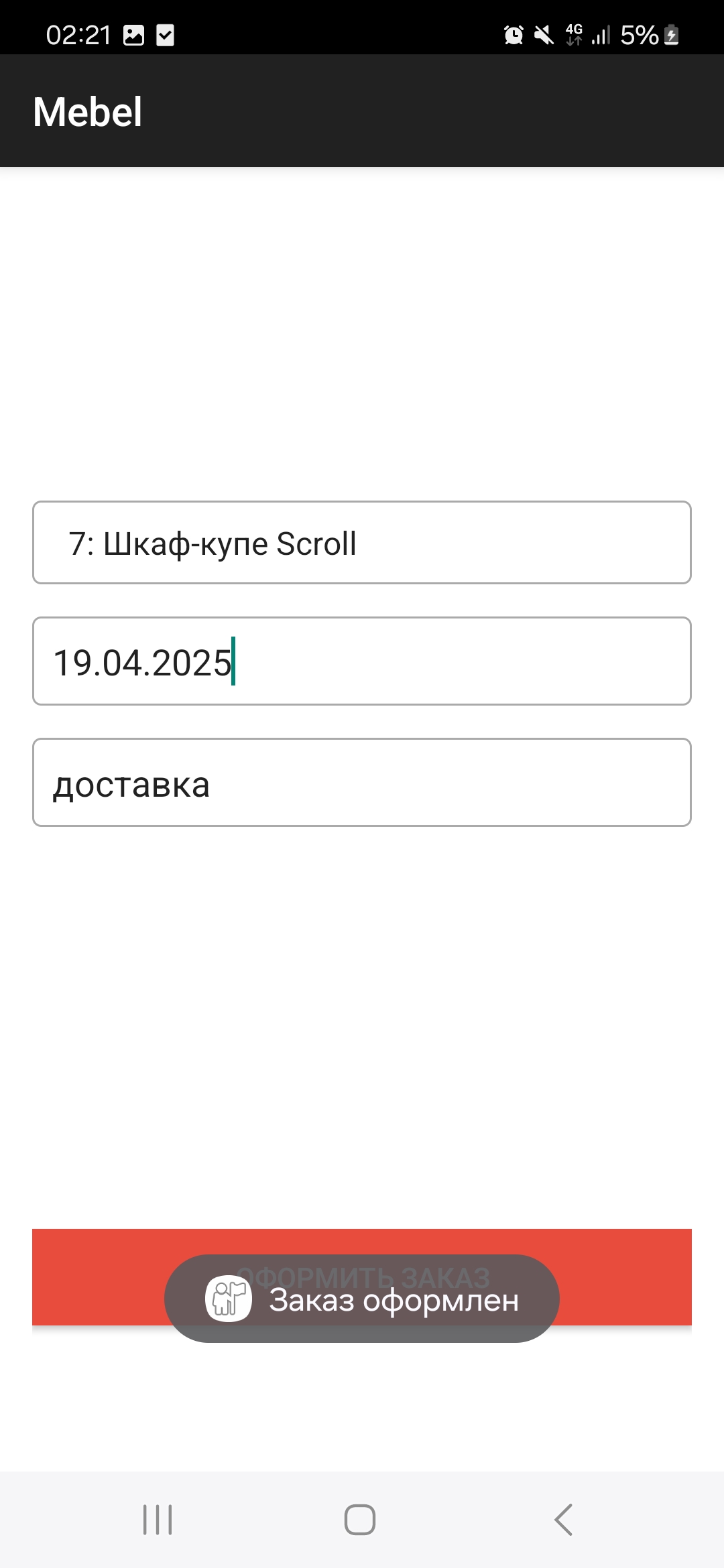
<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 android:padding="16dp"  
 android:background="@android:color/white">  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Список заказов"  
 android:textSize="24sp"  
 android:textColor="@android:color/black"  
 android:paddingBottom="16dp"/>  
  
 <ListView  
 android:id="@+id/ordersRecyclerView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="0dp"  
 android:layout\_weight="1"  
 android:divider="@android:color/darker\_gray"  
 android:dividerHeight="1dp"/>  
</LinearLayout>

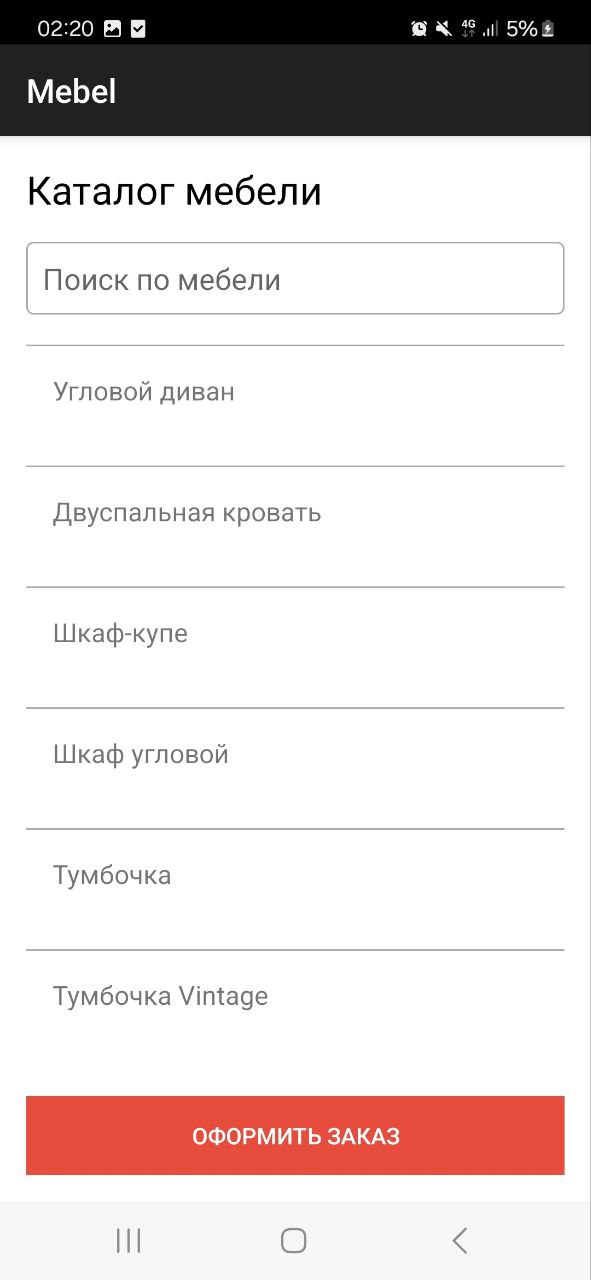
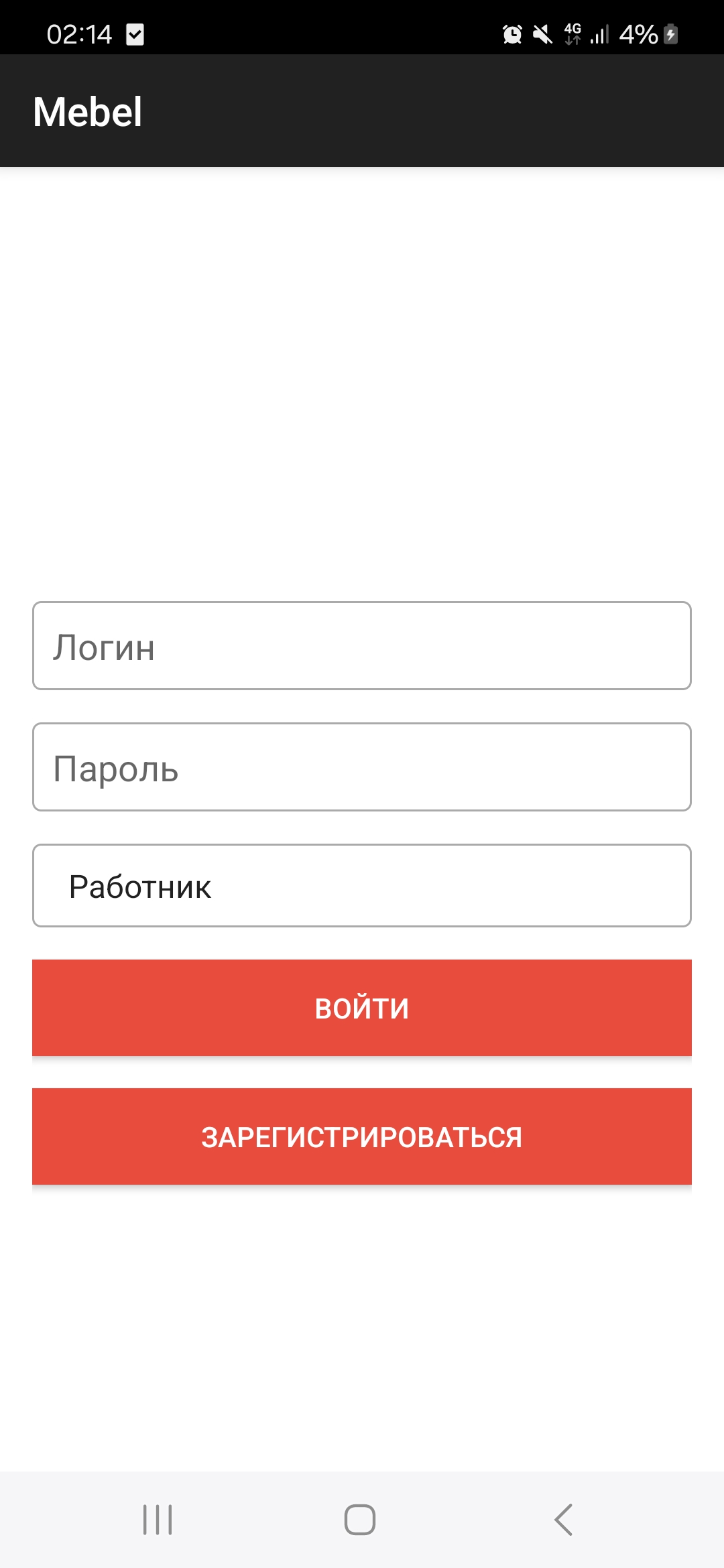
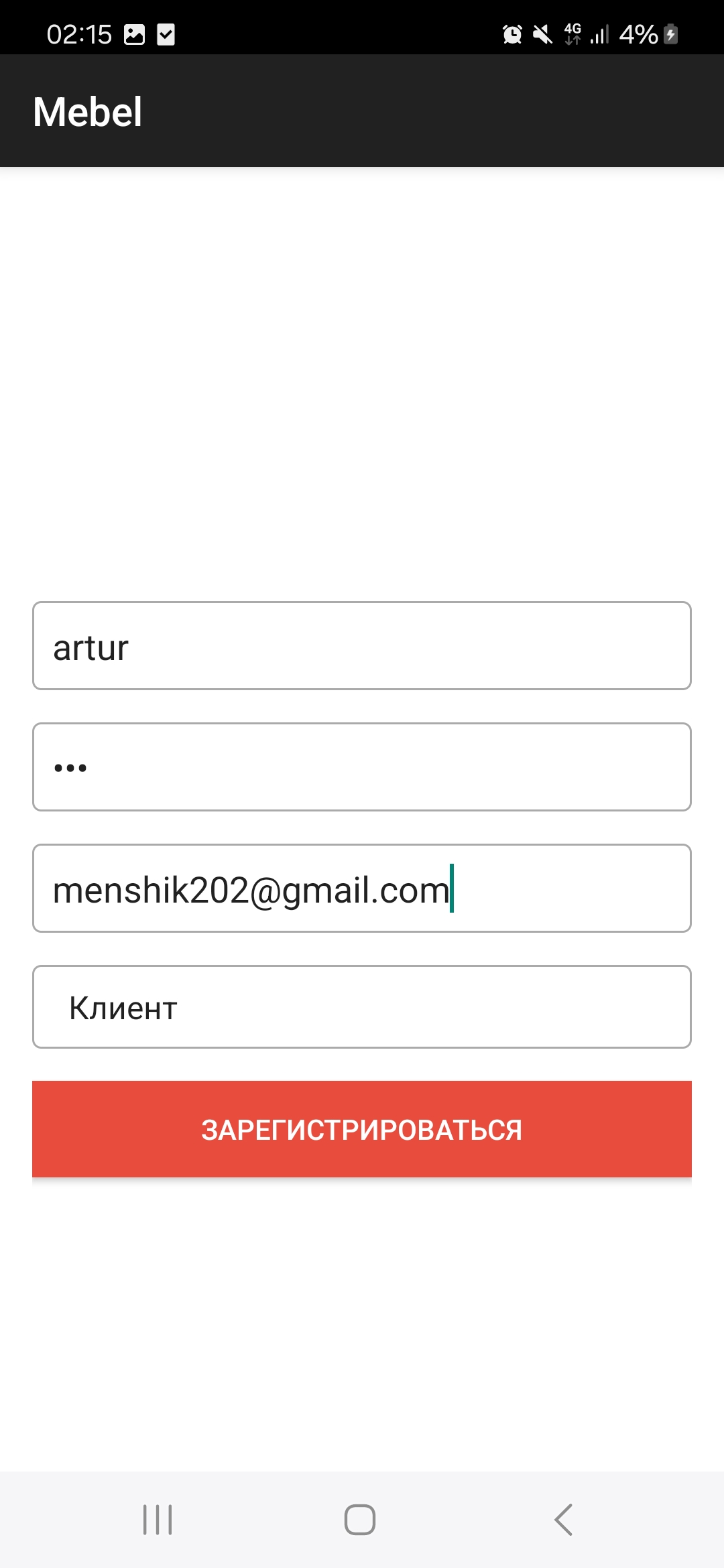
<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 android:padding="16dp">  
  
 <TextView  
 android:id="@+id/modelTextView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:textSize="16sp" />  
  
 <TextView  
 android:id="@+id/typeTextView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:textSize="14sp"  
 android:textColor="#888888" />  
</LinearLayout>

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 android:padding="8dp">  
  
 <TextView  
 android:id="@+id/partMaterialTextView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:textSize="16sp" />  
</LinearLayout>

<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools">  
  
 <application  
 android:allowBackup="true"  
 android:dataExtractionRules="@xml/data\_extraction\_rules"  
 android:fullBackupContent="@xml/backup\_rules"  
 android:icon="@android:drawable/ic\_menu\_myplaces"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:supportsRtl="true"  
 [android:theme="@style/Theme.Mebel](mailto:android:theme="@style/Theme.Mebel)"  
 tools:targetApi="31">  
 <activity  
 android:name=".ItemDetailsActivity"  
 android:exported="false" />  
 <activity  
 android:name=".SupplierDetailsActivity"  
 android:exported="false" />  
 <activity  
 android:name=".ClientOrderActivity"  
 android:exported="false" />  
 <activity  
 android:name=".SupplierMainActivity"  
 android:exported="false" />  
 <activity  
 android:name=".ClientMainActivity"  
 android:exported="false" />  
 <activity  
 android:name=".WorkerMainActivity"  
 android:exported="false" />  
 <activity  
 android:name=".RegistrationActivity"  
 android:exported="false" />  
 <activity  
 android:name=".MainActivity"  
 android:exported="true">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 </activity>  
 </application>  
  
</manifest>

**Экраны:**





**Тестовые ситуации:**

Тестирование приложения может включать следующие сценарии:

1. Авторизация и доступ: Проверка входа, регистрации, обработки неверных данных.
2. Управление товарами: Добавление, редактирование и удаление товаров, проверка отображения изменений.
3. Фильтрация и поиск: Проверка фильтрации товаров по категориям, поиска по названию.
4. Управление пользователями: Просмотр и редактирование данных клиентов и поставщиков, удаление.
5. Управление деталями: Добавление, редактирование и удаление деталей, включая проверку ввода данных и фотографий.