Министерство образования и молодежной политики Свердловской области



ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

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

Выполнил: Максимов К.Э.

Группа: ПР-31

Преподаватель: Мирошниченко Г.В

2023

**Входные данные:**

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

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

Данные о типах товаров: Название типа товара.

Данные о деталях: Вес, материал, диаметр, фото.

**Выходные данные:**

Списки товаров: Отображение товаров с возможностью фильтрации по категориям.

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

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

**Код программы:**

База данных:

package com.example.zd5\_v8.database  
  
import android.content.Context  
import androidx.room.Database  
import androidx.room.Room  
import androidx.room.RoomDatabase  
  
@Database(entities = [Order::class, FurnitureType::class, Furniture::class, PartType::class, Part::class, Supplier::class, Client::class, Worker::class], version = 4)  
abstract class AppDatabase : RoomDatabase() {  
 abstract fun orderDao(): *OrderDao* abstract fun furnitureTypeDao(): *FurnitureTypeDao* abstract fun furnitureDao(): *FurnitureDao* abstract fun partTypeDao(): *PartTypeDao* abstract fun partDao(): *PartDao* abstract fun supplierDao(): *SupplierDao* abstract fun clientDao(): *ClientDao* abstract fun workerDao(): *WorkerDao* companion object {  
 @Volatile  
 private var INSTANCE: AppDatabase? = null  
  
 fun getDatabase(context: Context): AppDatabase {  
 val tempInstance = INSTANCE  
 if (tempInstance != null) {  
 return tempInstance  
 }  
 synchronized(this) {  
 val instance = Room.databaseBuilder(  
 context.*applicationContext*,  
 AppDatabase::class.*java*,  
 "app\_database"  
 ).fallbackToDestructiveMigration().build()  
 INSTANCE = instance  
 return instance  
 }  
 }  
 }  
}

package com.example.zd5\_v8.database  
  
import androidx.room.Entity  
import androidx.room.PrimaryKey  
  
@Entity(tableName = "clients")  
data class Client(  
 @PrimaryKey(autoGenerate = true) val clientId: Int = 0,  
 val email: String,  
 val password: String,  
 val role: String = "Клиент",  
 val name: String,  
 val discount: Double  
)

package com.example.zd5\_v8  
  
import android.view.LayoutInflater  
import android.view.ViewGroup  
import android.widget.Button  
import android.widget.TextView  
import androidx.recyclerview.widget.RecyclerView  
import com.example.zd5\_v8.database.Client  
  
class ClientAdapter(  
 private var clients: *List*<Client>,  
 private val onEditClick: (Client) -> Unit,  
 private val onDeleteClick: (Client) -> Unit  
) : RecyclerView.Adapter<ClientAdapter.ViewHolder>() {  
  
 class ViewHolder(view: ViewGroup, onDelete: (Int) -> Unit) : RecyclerView.ViewHolder(view) {  
 val nameTextView: TextView = view.findViewById(R.id.*tvClientName*)  
 val emailTextView: TextView = view.findViewById(R.id.*tvClientEmail*)  
 val editButton: Button = view.findViewById(R.id.*btnEdit*)  
  
 init {  
 itemView.setOnClickListener {  
 onDelete(*adapterPosition*)  
 }  
 }  
 }  
  
 override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {  
 val view = LayoutInflater.from(parent.*context*).inflate(R.layout.*item\_client*, parent, false) as ViewGroup  
 return ViewHolder(view, { pos -> onDeleteClick(clients[pos]) })  
 }  
  
 override fun onBindViewHolder(holder: ViewHolder, position: Int) {  
 val client = clients[position]  
 holder.nameTextView.*text* = client.name  
 holder.emailTextView.*text* = client.email  
 holder.editButton.setOnClickListener { onEditClick(client) }  
 }  
  
 override fun getItemCount(): Int = clients.size  
  
 fun updateData(newClients: *List*<Client>) {  
 clients = newClients  
 notifyDataSetChanged()  
 }  
}

package com.example.zd5\_v8.database  
  
import androidx.room.Dao  
import androidx.room.Delete  
import androidx.room.Insert  
import androidx.room.Query  
import androidx.room.Update  
  
@Dao  
interface ClientDao {  
 @Insert  
 fun insert(client: Client)  
  
 @Update  
 fun update(client: Client)  
  
 @Delete  
 fun delete(client: Client)  
  
 @Query("SELECT \* FROM clients")  
 fun getAllClients(): List<Client>  
  
 @Query("SELECT \* FROM clients WHERE email = :email")  
 fun findClientByEmail(email: String): Client?  
  
 @Query("SELECT \* FROM clients WHERE clientId = :clientId")  
 fun getClientById(clientId: Int): Client?  
}

package com.example.zd5\_v8.database  
  
import androidx.room.Entity  
import androidx.room.PrimaryKey  
  
@Entity(tableName = "furnitures")  
data class Furniture(  
 @PrimaryKey(autoGenerate = true) val furnitureId: Int = 0,  
 val typeId: Int,  
 val name: String,  
 val description: String,  
 val cost: Double,  
 val photo: String  
)

package com.example.zd5\_v8.database  
  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import android.widget.ImageView  
import android.widget.TextView  
import androidx.recyclerview.widget.RecyclerView  
import com.example.zd5\_v8.R  
import com.squareup.picasso.Picasso  
  
class FurnitureAdapter(  
 private var furnitureList: *List*<Furniture>,  
 private val onItemClick: (Furniture) -> Unit,  
 private val onItemLongClick: (Furniture) -> Unit  
) : RecyclerView.Adapter<FurnitureAdapter.ViewHolder>() {  
  
 class ViewHolder(view: View) : RecyclerView.ViewHolder(view) {  
 val nameTextView: TextView = view.findViewById(R.id.*furniture\_name*)  
 val descriptionTextView: TextView = view.findViewById(R.id.*furniture\_description*)  
 val costTextView: TextView = view.findViewById(R.id.*furniture\_cost*)  
 val photoImageView: ImageView = view.findViewById(R.id.*furniture\_photo*)  
 }  
  
 override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {  
 val view = LayoutInflater.from(parent.*context*)  
 .inflate(R.layout.*item\_furniture*, parent, false)  
 return ViewHolder(view)  
 }  
  
 override fun onBindViewHolder(holder: ViewHolder, position: Int) {  
 val furniture = furnitureList[position]  
  
 holder.itemView.setOnClickListener { onItemClick(furniture) }  
 holder.itemView.setOnLongClickListener { onItemLongClick(furniture); true }  
  
 holder.nameTextView.*text* = furniture.name  
 holder.descriptionTextView.*text* = furniture.description  
 holder.costTextView.*text* = "${furniture.cost} РУБ"  
  
 if (!furniture.photo.*isNullOrEmpty*()) {  
 Picasso.get()  
 .load(furniture.photo)  
 .fit()  
 .centerCrop()  
 .into(holder.photoImageView)  
 } else {  
 holder.photoImageView.setImageResource(R.drawable.*no\_image*)  
 }  
 }  
  
 override fun getItemCount(): Int = furnitureList.size  
  
 fun updateData(newFurnitureList: *List*<Furniture>) {  
 furnitureList = newFurnitureList  
 notifyDataSetChanged()  
 }  
}

package com.example.zd5\_v8.database  
  
import androidx.room.Dao  
import androidx.room.Delete  
import androidx.room.Insert  
import androidx.room.Query  
import androidx.room.Update  
  
@Dao  
interface *FurnitureDao* {  
 @Insert  
 fun insert(furniture: Furniture)  
  
 @Update  
 fun update(furniture: Furniture)  
  
 @Delete  
 fun delete(furniture: Furniture)  
  
 @Query("SELECT \* FROM furnitures")  
 fun getAllFurnitures(): *List*<Furniture>  
  
 @Query("SELECT \* FROM furnitures WHERE furnitureId = :id")  
 fun getFurnitureById(id: Int): Furniture?  
  
 @Query("SELECT \* FROM furnitures WHERE typeId = :typeId")  
 fun getFurnitureByType(typeId: Int): *List*<Furniture>  
}

package com.example.zd5\_v8.database  
  
import androidx.room.Entity  
import androidx.room.PrimaryKey  
  
@Entity(tableName = "furniture\_types")  
data class FurnitureType(  
 @PrimaryKey(autoGenerate = true) val typeId: Int,  
 val typeName: String  
)

package com.example.zd5\_v8.database  
  
import androidx.room.Dao  
import androidx.room.Delete  
import androidx.room.Insert  
import androidx.room.Query  
import androidx.room.Update  
  
@Dao  
interface FurnitureTypeDao {  
 @Insert  
 fun insert(furnitureType: FurnitureType)  
  
 @Update  
 fun update(furnitureType: FurnitureType)  
  
 @Delete  
 fun delete(furnitureType: FurnitureType)  
  
 @Query("SELECT \* FROM furniture\_types")  
 fun getAllFurnitureTypes(): List<FurnitureType>  
}

package com.example.zd5\_v8.database  
  
import androidx.room.Entity  
import androidx.room.PrimaryKey  
  
@Entity(tableName = "orders")  
data class Order(  
 @PrimaryKey(autoGenerate = true) val orderId: Int,  
)

package com.example.zd5\_v8.database  
  
import androidx.room.Dao  
import androidx.room.Delete  
import androidx.room.Insert  
import androidx.room.Query  
import androidx.room.Update  
  
@Dao  
interface *OrderDao* {  
 @Insert  
 fun insert(order: Order)  
  
 @Update  
 fun update(order: Order)  
  
 @Delete  
 fun delete(order: Order)  
  
 @Query("SELECT \* FROM orders")  
 fun getAllOrders(): *List*<Order>  
}

package com.example.zd5\_v8.database  
  
import androidx.room.Entity  
import androidx.room.PrimaryKey  
  
@Entity(tableName = "parts")  
data class Part(  
 @PrimaryKey(autoGenerate = true) val partId: Int = 0,  
 val partTypeId: Int,  
 val name: String,  
 val weight: Double,  
 val material: String,  
 val diameter: Double,  
 val photo: String  
)

package com.example.zd5\_v8.database  
  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import android.widget.ImageView  
import android.widget.TextView  
import androidx.recyclerview.widget.RecyclerView  
import com.example.zd5\_v8.R  
import com.squareup.picasso.Picasso  
  
class PartAdapter(  
 private var partsList: List<Part>,  
 private val onItemClick: (Part) -> Unit,  
 private val onItemLongClick: (Part) -> Unit  
) : RecyclerView.Adapter<PartAdapter.ViewHolder>() {  
  
 class ViewHolder(view: View) : RecyclerView.ViewHolder(view) {  
 val photoImageView: ImageView = view.findViewById(R.id.parts\_photo)  
 val nameTextView: TextView = view.findViewById(R.id.parts\_name)  
 val materialTextView: TextView = view.findViewById(R.id.parts\_material)  
 val diameterTextView: TextView = view.findViewById(R.id.parts\_diameter)  
 val weightTextView: TextView = view.findViewById(R.id.parts\_weight)  
 }  
  
 override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {  
 val view = LayoutInflater.from(parent.context)  
 .inflate(R.layout.item\_parts, parent, false)  
 return ViewHolder(view)  
 }  
  
 override fun onBindViewHolder(holder: ViewHolder, position: Int) {  
 val part = partsList[position]  
 holder.nameTextView.text = part.name  
 holder.materialTextView.text = "Материал: ${part.material}"  
 holder.diameterTextView.text = "Диаметр: ${part.diameter}"  
 holder.weightTextView.text = "Вес: ${part.weight}"  
  
 holder.itemView.setOnLongClickListener {  
 onItemLongClick(part)  
 true  
 }  
 holder.itemView.setOnClickListener { onItemClick(part) }  
  
  
 if (part.photo.isNotEmpty()) {  
 Picasso.get()  
 .load(part.photo)  
 .fit()  
 .centerCrop()  
 .into(holder.photoImageView)  
 } else {  
 holder.photoImageView.setImageResource(R.drawable.no\_image)  
 }  
 }  
  
 override fun getItemCount() = partsList.size  
  
 fun updateParts(parts: List<Part>) {  
 this.partsList = parts  
 notifyDataSetChanged()  
 }  
}

package com.example.zd5\_v8.database  
  
import androidx.room.\*  
  
@Dao  
interface PartDao {  
 @Insert  
 fun insert(part: Part): Long  
  
 @Update  
 fun update(part: Part)  
  
 @Delete  
 fun delete(part: Part)  
  
 @Query("SELECT \* FROM parts")  
 fun getAllParts(): List<Part>  
  
 @Query("SELECT \* FROM parts WHERE partTypeId = :partTypeId")  
 fun getPartsByType(partTypeId: Int): List<Part>  
  
 @Query("SELECT \* FROM parts WHERE partId = :partId")  
 fun getPartById(partId: Int): Part?  
  
 @Insert  
 fun insertType(partType: PartType): Long  
  
 @Query("SELECT \* FROM part\_types")  
 fun getAllPartTypes(): List<PartType>  
}

package com.example.zd5\_v8.database  
  
import androidx.room.Entity  
import androidx.room.PrimaryKey  
  
@Entity(tableName = "part\_types")  
data class PartType(  
 @PrimaryKey(autoGenerate = true) val partTypeId: Int = 0,  
 val partName: String  
)

package com.example.zd5\_v8.database  
  
import androidx.room.Dao  
import androidx.room.Delete  
import androidx.room.Insert  
import androidx.room.Query  
import androidx.room.Update  
  
@Dao  
interface PartTypeDao {  
 @Insert  
 fun insert(partType: PartType)  
  
 @Update  
 fun update(partType: PartType)  
  
 @Delete  
 fun delete(partType: PartType)  
  
 @Query("SELECT \* FROM part\_types")  
 fun getAllPartTypes(): List<PartType>  
}

package com.example.zd5\_v8.database  
  
import androidx.room.Entity  
import androidx.room.PrimaryKey  
  
@Entity(tableName = "suppliers")  
data class Supplier(  
 @PrimaryKey(autoGenerate = true) val supplierId: Int = 0,  
 val email: String,  
 val password: String,  
 val role: String = "Поставщик",  
 val name: String,  
 val address: String  
)

package com.example.zd5\_v8  
  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import android.widget.Button  
import android.widget.TextView  
import androidx.recyclerview.widget.RecyclerView  
import com.example.zd5\_v8.database.Supplier  
  
class SupplierAdapter(  
 private var suppliers: List<Supplier>,  
 private val onEditClick: (Supplier) -> Unit,  
 private val onDeleteClick: (Supplier) -> Unit  
) : RecyclerView.Adapter<SupplierAdapter.ViewHolder>() {  
  
 class ViewHolder(view: View, onEdit: (Int) -> Unit, onDelete: (Int) -> Unit) : RecyclerView.ViewHolder(view) {  
 val nameTextView: TextView = view.findViewById(R.id.tvSupplierName)  
 val emailTextView: TextView = view.findViewById(R.id.tvSupplierEmail)  
 val editButton: Button = view.findViewById(R.id.btnEdit)  
  
 init {  
 itemView.setOnClickListener { onDelete(adapterPosition) }  
 editButton.setOnClickListener { onEdit(adapterPosition) }  
 }  
 }  
  
 override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {  
 val view = LayoutInflater.from(parent.context).inflate(R.layout.item\_supplier, parent, false)  
 return ViewHolder(view, { pos -> onEditClick(suppliers[pos]) }, { pos -> onDeleteClick(suppliers[pos]) })  
 }  
  
 override fun onBindViewHolder(holder: ViewHolder, position: Int) {  
 val supplier = suppliers[position]  
 holder.nameTextView.text = supplier.name  
 holder.emailTextView.text = supplier.email  
 }  
  
 override fun getItemCount(): Int = suppliers.size  
  
 fun updateData(newSuppliers: List<Supplier>) {  
 suppliers = newSuppliers  
 notifyDataSetChanged()  
 }  
}

package com.example.zd5\_v8.database  
  
import androidx.room.Dao  
import androidx.room.Delete  
import androidx.room.Insert  
import androidx.room.Query  
import androidx.room.Update  
  
@Dao  
interface SupplierDao {  
 @Insert  
 fun insert(supplier: Supplier)  
  
 @Update  
 fun update(supplier: Supplier)  
  
 @Delete  
 fun delete(supplier: Supplier)  
  
 @Query("SELECT \* FROM suppliers")  
 fun getAllSuppliers(): List<Supplier>  
  
 @Query("SELECT \* FROM suppliers WHERE email = :email")  
 fun findSupplierByEmail(email: String): Supplier?  
  
 @Query("SELECT \* FROM suppliers WHERE supplierId = :supplierId")  
 fun getSupplierById(supplierId: Int): Supplier?  
}

package com.example.zd5\_v8.database  
  
import androidx.room.Entity  
import androidx.room.PrimaryKey  
  
@Entity(tableName = "workers")  
data class Worker(  
 @PrimaryKey(autoGenerate = true) val workerId: Int = 0,  
 val email: String,  
 val password: String,  
 val name: String  
)

package com.example.zd5\_v8.database  
  
import androidx.room.Dao  
import androidx.room.Delete  
import androidx.room.Insert  
import androidx.room.Query  
import androidx.room.Update  
  
@Dao  
interface WorkerDao {  
 @Insert  
 fun insert(worker: Worker)  
  
 @Update  
 fun update(worker: Worker)  
  
 @Delete  
 fun delete(worker: Worker)  
  
 @Query("SELECT \* FROM workers")  
 fun getAllWorkers(): List<Worker>  
  
 @Query("SELECT \* FROM workers WHERE email = :email")  
 fun findWorkerByEmail(email: String): Worker?  
}

Код программы:

package com.example.zd5\_v8  
  
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 com.example.zd5\_v8.database.AppDatabase  
import com.example.zd5\_v8.database.Furniture  
import com.example.zd5\_v8.database.FurnitureType  
import kotlinx.coroutines.CoroutineScope  
import kotlinx.coroutines.Dispatchers  
import kotlinx.coroutines.launch  
import kotlinx.coroutines.withContext  
  
class AddFurnitureActivity : AppCompatActivity() {  
  
 private lateinit var typeSpinner: Spinner  
 private lateinit var furnitureTypes: *List*<FurnitureType>  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_add\_furniture*)  
  
 val nameEditText = findViewById<EditText>(R.id.*etFurnitureName*)  
 val descriptionEditText = findViewById<EditText>(R.id.*etFurnitureDescription*)  
 val costEditText = findViewById<EditText>(R.id.*etFurnitureCost*)  
 val photoEditText = findViewById<EditText>(R.id.*etFurniturePhoto*) // URL or path of photo  
 typeSpinner = findViewById(R.id.*spinnerFurnitureType*)  
  
 loadFurnitureTypes()  
  
 val btnSave = findViewById<Button>(R.id.*btnSaveFurniture*)  
 btnSave.setOnClickListener {  
 if (nameEditText.*text*.*isEmpty*() || descriptionEditText.*text*.*isEmpty*() || costEditText.*text*.*isEmpty*()) {  
 Toast.makeText(this, "Заполните все поля", Toast.*LENGTH\_SHORT*).show()  
 } else {  
 if (typeSpinner.*selectedItem* != null) {  
 val selectedPosition = typeSpinner.*selectedItemPosition* val selectedFurnitureType = furnitureTypes[selectedPosition]  
 val cost = costEditText.*text*.toString().*toDoubleOrNull*() ?: 0.0  
  
 val furniture = Furniture(  
 typeId = selectedFurnitureType.typeId,  
 name = nameEditText.*text*.toString(),  
 description = descriptionEditText.*text*.toString(),  
 cost = cost,  
 photo = photoEditText.*text*.toString()  
 )  
 saveFurniture(furniture)  
 } else {  
 Toast.makeText(this, "Выберите тип мебели", Toast.*LENGTH\_SHORT*).show()  
 }  
 }  
 }  
 }  
  
 private fun loadFurnitureTypes() {  
 *CoroutineScope*(Dispatchers.IO).*launch* {  
 val db = AppDatabase.getDatabase(this@AddFurnitureActivity)  
 furnitureTypes = db.furnitureTypeDao().getAllFurnitureTypes()  
 withContext(Dispatchers.Main) {  
 updateFurnitureTypesSpinner()  
 }  
 }  
 }  
  
 private fun updateFurnitureTypesSpinner() {  
 val adapter = ArrayAdapter(this, android.R.layout.*simple\_spinner\_item*, furnitureTypes.*map* { **it**.typeName })  
 adapter.setDropDownViewResource(android.R.layout.*simple\_spinner\_dropdown\_item*)  
 typeSpinner.*adapter* = adapter  
 }  
  
 private fun saveFurniture(furniture: Furniture) {  
 *CoroutineScope*(Dispatchers.IO).*launch* {  
 val db = AppDatabase.getDatabase(this@AddFurnitureActivity)  
 db.furnitureDao().insert(furniture)  
 withContext(Dispatchers.Main) {  
 finish()  
 }  
 }  
 }  
}

package com.example.zd5\_v8  
  
import android.os.Bundle  
import android.widget.Button  
import android.widget.EditText  
import android.widget.Toast  
import androidx.appcompat.app.AppCompatActivity  
import com.example.zd5\_v8.database.AppDatabase  
import com.example.zd5\_v8.database.FurnitureType  
import kotlinx.coroutines.CoroutineScope  
import kotlinx.coroutines.Dispatchers  
import kotlinx.coroutines.launch  
import kotlinx.coroutines.withContext  
  
class AddFurnitureTypeActivity : AppCompatActivity() {  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_add\_furniture\_type*)  
  
 val etFurnitureName = findViewById<EditText>(R.id.*etFurnitureName*)  
 val btnSaveFurniture = findViewById<Button>(R.id.*btnSaveFurniture*)  
  
 btnSaveFurniture.setOnClickListener {  
 val typeName = etFurnitureName.*text*.toString().*trim*()  
  
 if (typeName.*isNotEmpty*()) {  
 val furnitureType = FurnitureType(0, typeName)  
 saveFurnitureType(furnitureType)  
 } else {  
 Toast.makeText(this, "Введите название типа мебели", Toast.*LENGTH\_SHORT*).show()  
 }  
 }  
 }  
  
 private fun saveFurnitureType(furnitureType: FurnitureType) {  
 *CoroutineScope*(Dispatchers.IO).*launch* {  
 val db = AppDatabase.getDatabase(this@AddFurnitureTypeActivity)  
  
 val existingTypes = db.furnitureTypeDao().getAllFurnitureTypes()  
  
 val exists = existingTypes.*any* { **it**.typeName.*equals*(furnitureType.typeName, ignoreCase = true) }  
  
 if (!exists) {  
 db.furnitureTypeDao().insert(furnitureType)  
 withContext(Dispatchers.Main) {  
 Toast.makeText(this@AddFurnitureTypeActivity, "Тип мебели добавлен", Toast.*LENGTH\_SHORT*).show()  
 finish()  
 }  
 } else {  
 withContext(Dispatchers.Main) {  
 Toast.makeText(this@AddFurnitureTypeActivity, "Тип мебели уже существует", Toast.*LENGTH\_SHORT*).show()  
 }  
 }  
 }  
 }  
  
}

package com.example.zd5\_v8  
  
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 com.example.zd5\_v8.database.AppDatabase  
import com.example.zd5\_v8.database.Part  
import com.example.zd5\_v8.database.PartType  
import kotlinx.coroutines.CoroutineScope  
import kotlinx.coroutines.Dispatchers  
import kotlinx.coroutines.launch  
import kotlinx.coroutines.withContext  
  
class AddPartActivity : AppCompatActivity() {  
  
 private lateinit var spinnerPartType: Spinner  
 private lateinit var etPartWeight: EditText  
 private lateinit var etPartName: EditText  
 private lateinit var etPartMaterial: EditText  
 private lateinit var etPartDiameter: EditText  
 private lateinit var etPartPhoto: EditText  
 private lateinit var partTypes: List<PartType>  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.activity\_add\_part)  
  
 spinnerPartType = findViewById(R.id.spinnerPartType)  
 etPartWeight = findViewById(R.id.etPartWeight)  
 etPartName = findViewById(R.id.etPartName)  
 etPartMaterial = findViewById(R.id.etPartMaterial)  
 etPartDiameter = findViewById(R.id.etPartDiameter)  
 etPartPhoto = findViewById(R.id.etPartPhoto)  
  
 loadPartTypes()  
  
 findViewById<Button>(R.id.btnSavePart).setOnClickListener {  
 val selectedPosition = spinnerPartType.selectedItemPosition  
 if (selectedPosition >= 0) {  
 val selectedPartType = partTypes[selectedPosition]  
 val name = etPartName.text.toString()  
 val weightText = etPartWeight.text.toString()  
 val material = etPartMaterial.text.toString()  
 val diameterText = etPartDiameter.text.toString()  
 val photo = etPartPhoto.text.toString()  
  
 // Проверяем, что поля введены  
 if (name.isEmpty() || weightText.isEmpty() || material.isEmpty() || diameterText.isEmpty()) {  
 Toast.makeText(this, "Пожалуйста, заполните все поля", Toast.LENGTH\_SHORT).show()  
 return@setOnClickListener  
 }  
  
 // Конвертируем в числа с проверкой  
 val weight = weightText.toDoubleOrNull()  
 val diameter = diameterText.toDoubleOrNull()  
  
 if (weight == null || diameter == null) {  
 Toast.makeText(this, "Пожалуйста, введите корректные числовые значения", Toast.LENGTH\_SHORT).show()  
 return@setOnClickListener  
 }  
  
 val part = Part(  
 partTypeId = selectedPartType.partTypeId,  
 name = name,  
 weight = weight,  
 material = material,  
 diameter = diameter,  
 photo = photo  
 )  
 savePart(part)  
 } else {  
 Toast.makeText(this, "Выберите тип детали", Toast.LENGTH\_SHORT).show()  
 }  
 }  
 }  
  
 private fun loadPartTypes() {  
 CoroutineScope(Dispatchers.IO).launch {  
 val db = AppDatabase.getDatabase(this@AddPartActivity)  
 partTypes = db.partTypeDao().getAllPartTypes()  
 withContext(Dispatchers.Main) {  
 val adapter = ArrayAdapter(this@AddPartActivity, android.R.layout.simple\_spinner\_item, partTypes.map { it.partName })  
 adapter.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item)  
 spinnerPartType.adapter = adapter  
 }  
 }  
 }  
  
 private fun savePart(part: Part) {  
 CoroutineScope(Dispatchers.IO).launch {  
 val db = AppDatabase.getDatabase(this@AddPartActivity)  
 db.partDao().insert(part)  
 withContext(Dispatchers.Main) {  
 Toast.makeText(this@AddPartActivity, "Деталь добавлена", Toast.LENGTH\_SHORT).show()  
 finish()  
 }  
 }  
 }  
}

package com.example.zd5\_v8  
  
import android.os.Bundle  
import android.widget.Button  
import android.widget.EditText  
import android.widget.Toast  
import androidx.appcompat.app.AppCompatActivity  
import com.example.zd5\_v8.database.AppDatabase  
import com.example.zd5\_v8.database.PartType  
import kotlinx.coroutines.CoroutineScope  
import kotlinx.coroutines.Dispatchers  
import kotlinx.coroutines.launch  
import kotlinx.coroutines.withContext  
  
class AddPartTypeActivity : AppCompatActivity() {  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.activity\_add\_part\_type)  
  
 val etPartTypeName = findViewById<EditText>(R.id.etPartTypeName)  
 val btnSavePartType = findViewById<Button>(R.id.btnSavePartType)  
  
 btnSavePartType.setOnClickListener {  
 val typeName = etPartTypeName.text.toString().trim()  
  
 if (typeName.isNotEmpty()) {  
 val partType = PartType(0, typeName)  
 savePartType(partType)  
 } else {  
 Toast.makeText(this, "Введите название типа детали", Toast.LENGTH\_SHORT).show()  
 }  
 }  
 }  
  
 private fun savePartType(partType: PartType) {  
 CoroutineScope(Dispatchers.IO).launch {  
 val db = AppDatabase.getDatabase(this@AddPartTypeActivity)  
  
 val existingTypes = db.partTypeDao().getAllPartTypes()  
  
 val exists = existingTypes.any { it.partName.equals(partType.partName, ignoreCase = true) }  
  
 if (!exists) {  
 db.partTypeDao().insert(partType)  
 withContext(Dispatchers.Main) {  
 Toast.makeText(this@AddPartTypeActivity, "Тип детали добавлен", Toast.LENGTH\_SHORT).show()  
 finish()  
 }  
 } else {  
 withContext(Dispatchers.Main) {  
 Toast.makeText(this@AddPartTypeActivity, "Тип детали уже существует", Toast.LENGTH\_SHORT).show()  
 }  
 }  
 }  
 }  
}

package com.example.zd5\_v8  
  
import androidx.appcompat.app.AppCompatActivity  
import android.os.Bundle  
import android.view.View  
import android.widget.AdapterView  
import android.widget.ArrayAdapter  
import android.widget.Spinner  
import androidx.appcompat.app.AlertDialog  
import androidx.recyclerview.widget.LinearLayoutManager  
import androidx.recyclerview.widget.RecyclerView  
import com.example.zd5\_v8.database.AppDatabase  
import com.example.zd5\_v8.database.Furniture  
import com.example.zd5\_v8.database.FurnitureAdapter  
import kotlinx.coroutines.CoroutineScope  
import kotlinx.coroutines.Dispatchers  
import kotlinx.coroutines.launch  
import kotlinx.coroutines.withContext  
  
class ClientActivity : AppCompatActivity() {  
 private lateinit var recyclerView: RecyclerView  
 private lateinit var furnitureAdapter: FurnitureAdapter  
 private lateinit var categorySpinner: Spinner  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.activity\_client)  
  
 recyclerView = findViewById(R.id.rvFurniture)  
 recyclerView.layoutManager = LinearLayoutManager(this)  
  
 categorySpinner = findViewById(R.id.categorySpinner)  
 loadCategoriesIntoSpinner()  
  
 furnitureAdapter = FurnitureAdapter(listOf(), this::onItemClick, {})  
 recyclerView.adapter = furnitureAdapter  
  
 loadFurnitureFromDb()  
 }  
 private fun onItemClick(furniture: Furniture) {  
 AlertDialog.Builder(this)  
 .setTitle("Подтверждение")  
 .setMessage("Вы хотите купить этот товар?")  
 .setPositiveButton("Да") { \_, \_ -> deleteFurniture(furniture) }  
 .setNegativeButton("Нет", null)  
 .show()  
 }  
 private fun deleteFurniture(furniture: Furniture) {  
 CoroutineScope(Dispatchers.IO).launch {  
 val db = AppDatabase.getDatabase(applicationContext)  
 db.furnitureDao().delete(furniture)  
 loadFurnitureFromDb()  
 }  
 }  
  
 private fun loadCategoriesIntoSpinner() {  
 CoroutineScope(Dispatchers.IO).launch {  
 val db = AppDatabase.getDatabase(this@ClientActivity)  
 val furnitureTypes = db.furnitureTypeDao().getAllFurnitureTypes()  
 val categories = listOf("Все категории") + furnitureTypes.map { it.typeName }  
 withContext(Dispatchers.Main) {  
 val adapter = ArrayAdapter(  
 this@ClientActivity,  
 android.R.layout.simple\_spinner\_dropdown\_item,  
 categories  
 )  
 categorySpinner.adapter = adapter  
 categorySpinner.onItemSelectedListener = object : AdapterView.OnItemSelectedListener {  
 override fun onItemSelected(adapterView: AdapterView<\*>, view: View, position: Int, id: Long) {  
 if (position == 0) {  
 loadFurnitureFromDb()  
 } else {  
 filterFurnitureByCategory(furnitureTypes[position - 1].typeId)  
 }  
 }  
 override fun onNothingSelected(adapterView: AdapterView<\*>) {}  
 }  
 }  
 }  
 }  
  
 private fun filterFurnitureByCategory(typeId: Int) {  
 CoroutineScope(Dispatchers.IO).launch {  
 val db = AppDatabase.getDatabase(this@ClientActivity)  
 val filteredFurniture = db.furnitureDao().getFurnitureByType(typeId)  
 withContext(Dispatchers.Main) {  
 furnitureAdapter.updateData(filteredFurniture)  
 }  
 }  
 }  
  
 private fun loadFurnitureFromDb() {  
 CoroutineScope(Dispatchers.IO).launch {  
 val db = AppDatabase.getDatabase(this@ClientActivity)  
 val furniture = db.furnitureDao().getAllFurnitures()  
 withContext(Dispatchers.Main) {  
 furnitureAdapter.updateData(furniture)  
 }  
 }  
 }  
}

package com.example.zd5\_v8  
  
import android.os.Bundle  
import android.widget.Button  
import android.widget.EditText  
import android.widget.Toast  
import androidx.appcompat.app.AppCompatActivity  
import androidx.core.widget.doOnTextChanged  
import com.example.zd5\_v8.database.AppDatabase  
import com.example.zd5\_v8.database.Client  
import kotlinx.coroutines.CoroutineScope  
import kotlinx.coroutines.Dispatchers  
import kotlinx.coroutines.launch  
import kotlinx.coroutines.withContext  
  
class EditClientActivity : AppCompatActivity() {  
 private lateinit var emailEditText: EditText  
 private lateinit var passwordEditText: EditText  
 private lateinit var nameEditText: EditText  
 private lateinit var btnSave: Button  
 private var clientId: Int = 0  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.activity\_edit\_client)  
  
  
 emailEditText = findViewById(R.id.emailEditText)  
 passwordEditText = findViewById(R.id.passwordEditText)  
 nameEditText = findViewById(R.id.nameEditText)  
 btnSave = findViewById(R.id.btnSave)  
  
  
 emailEditText.doOnTextChanged { text, \_, \_, \_ ->  
 if (text != null && !isValidEmail(text.toString())) {  
 emailEditText.error = "Неверный формат email"  
 }  
 }  
  
 passwordEditText.doOnTextChanged { text, \_, \_, \_ ->  
 if (text != null && text.length < 6) {  
 passwordEditText.error = "Пароль должен быть не менее 6 символов"  
 }  
 }  
  
 nameEditText.doOnTextChanged { text, \_, \_, \_ ->  
 if (text != null && !startsWithCapital(text)) {  
 nameEditText.error = "Имя должно начинаться с заглавной буквы"  
 }  
 }  
  
  
 clientId = intent.getIntExtra("CLIENT\_ID", -1)  
 if (clientId != -1) {  
 loadClientData(clientId)  
  
 btnSave.setOnClickListener {  
 updateClientData()  
 }  
 } else {  
 Toast.makeText(this, "Ошибка загрузки данных клиента!", Toast.LENGTH\_LONG).show()  
 finish()  
 }  
 }  
  
 private fun isValidEmail(email: CharSequence?): Boolean {  
 return android.util.Patterns.EMAIL\_ADDRESS.matcher(email).matches()  
 }  
  
 private fun startsWithCapital(name: CharSequence?): Boolean {  
 if (name != null) {  
 return name.matches(Regex("^[A-ZА-Я].\*"))  
 }  
 return false  
 }  
  
 private fun loadClientData(clientId: Int) {  
 CoroutineScope(Dispatchers.IO).launch {  
 val db = AppDatabase.getDatabase(applicationContext)  
 val client = db.clientDao().getClientById(clientId)  
 withContext(Dispatchers.Main) {  
 client?.let {  
 emailEditText.setText(it.email)  
 passwordEditText.setText(it.password)  
 nameEditText.setText(it.name)  
 }  
 }  
 }  
 }  
  
 private fun updateClientData() {  
 val email = emailEditText.text.toString()  
 val password = passwordEditText.text.toString()  
 val name = nameEditText.text.toString()  
  
 if (email.isEmpty() || password.isEmpty() || name.isEmpty()) {  
 Toast.makeText(this, "Пожалуйста, заполните все поля", Toast.LENGTH\_SHORT).show()  
 return  
 }  
  
 CoroutineScope(Dispatchers.IO).launch {  
 val db = AppDatabase.getDatabase(applicationContext)  
 db.clientDao().update(Client(clientId, email, password, "Клиент", name, 0.0))  
 withContext(Dispatchers.Main) {  
 Toast.makeText(applicationContext, "Данные клиента обновлены", Toast.LENGTH\_SHORT).show()  
 finish()  
 }  
 }  
 }  
}

package com.example.zd5\_v8  
  
import androidx.appcompat.app.AppCompatActivity  
import android.os.Bundle  
import android.widget.Button  
import android.widget.EditText  
import android.widget.Toast  
import com.example.zd5\_v8.database.AppDatabase  
import com.example.zd5\_v8.database.Furniture  
import kotlinx.coroutines.CoroutineScope  
import kotlinx.coroutines.Dispatchers  
import kotlinx.coroutines.launch  
import kotlinx.coroutines.withContext  
  
class EditFurnitureActivity : AppCompatActivity() {  
  
 private lateinit var nameEditText: EditText  
 private lateinit var descriptionEditText: EditText  
 private lateinit var costEditText: EditText  
 private lateinit var photoEditText: EditText  
 private var furnitureId: Int = -1  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_edit\_furniture*)  
  
 nameEditText = findViewById(R.id.*etFurnitureName*)  
 descriptionEditText = findViewById(R.id.*etFurnitureDescription*)  
 costEditText = findViewById(R.id.*etFurnitureCost*)  
 photoEditText = findViewById(R.id.*etImage*)  
  
 furnitureId = *intent*.getIntExtra("FURNITURE\_ID", -1)  
 if (furnitureId != -1) {  
 loadFurnitureDetails(furnitureId)  
 }  
  
 val btnSave = findViewById<Button>(R.id.*btnSaveFurniture*)  
 btnSave.setOnClickListener {  
 updateFurnitureDetails()  
 }  
  
 }  
  
 private fun loadFurnitureDetails(furnitureId: Int) {  
 *CoroutineScope*(Dispatchers.IO).*launch* {  
 val db = AppDatabase.getDatabase(*applicationContext*)  
 val furniture = db.furnitureDao().getFurnitureById(furnitureId)  
 withContext(Dispatchers.Main) {  
 furniture?.*let* {  
 nameEditText.setText(**it**.name)  
 descriptionEditText.setText(**it**.description)  
 costEditText.setText(**it**.cost.toString())  
 photoEditText.setText(**it**.photo)  
 }  
 }  
 }  
 }  
  
 private fun updateFurnitureDetails() {  
 val name = nameEditText.*text*.toString()  
 val description = descriptionEditText.*text*.toString()  
 val cost = costEditText.*text*.toString()  
 if (name.*isEmpty*() || description.*isEmpty*() || cost.*isEmpty*()) {  
 Toast.makeText(this, "Заполните все поля", Toast.*LENGTH\_SHORT*).show()  
 } else {  
 if (furnitureId != -1) {  
 *CoroutineScope*(Dispatchers.IO).*launch* {  
 val db = AppDatabase.getDatabase(*applicationContext*)  
 val cost = cost.*toDoubleOrNull*() ?: 0.0  
 val photo = photoEditText.*text*.toString()  
 val furniture = Furniture(  
 furnitureId = furnitureId,  
 typeId = 1,  
 name = name,  
 description = description,  
 cost = cost,  
 photo = photo  
 )  
 db.furnitureDao().update(furniture)  
 withContext(Dispatchers.Main) {  
 finish()  
 }  
 }  
 }  
 }  
 }  
}

package com.example.zd5\_v8  
  
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 com.example.zd5\_v8.database.AppDatabase  
import com.example.zd5\_v8.database.Part  
import com.example.zd5\_v8.database.PartType  
import kotlinx.coroutines.CoroutineScope  
import kotlinx.coroutines.Dispatchers  
import kotlinx.coroutines.launch  
import kotlinx.coroutines.withContext  
  
class EditPartActivity : AppCompatActivity() {  
  
 private lateinit var spinnerPartType: Spinner  
 private lateinit var etPartName: EditText  
 private lateinit var etPartWeight: EditText  
 private lateinit var etPartMaterial: EditText  
 private lateinit var etPartDiameter: EditText  
 private lateinit var etPartPhoto: EditText  
 private lateinit var btnSavePart: Button  
 private var partId: Int = -1  
 private lateinit var partTypes: List<PartType>  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.activity\_edit\_part)  
  
 spinnerPartType = findViewById(R.id.spinnerPartType)  
 etPartName = findViewById(R.id.etPartName)  
 etPartWeight = findViewById(R.id.etPartWeight)  
 etPartMaterial = findViewById(R.id.etPartMaterial)  
 etPartDiameter = findViewById(R.id.etPartDiameter)  
 etPartPhoto = findViewById(R.id.etPartPhoto)  
 btnSavePart = findViewById(R.id.btnSavePart)  
  
 partId = intent.getIntExtra("PART\_ID", -1)  
  
 loadPartTypes()  
 if (partId != -1) {  
 loadPartDetails(partId)  
 }  
  
 btnSavePart.setOnClickListener {  
 updatePartDetails()  
 }  
 }  
  
 private fun loadPartTypes() {  
 CoroutineScope(Dispatchers.IO).launch {  
 val db = AppDatabase.getDatabase(applicationContext)  
 partTypes = db.partTypeDao().getAllPartTypes()  
 withContext(Dispatchers.Main) {  
 val adapter = ArrayAdapter(this@EditPartActivity, android.R.layout.simple\_spinner\_dropdown\_item, partTypes.map { it.partName })  
 spinnerPartType.adapter = adapter  
 }  
 }  
 }  
  
 private fun loadPartDetails(partId: Int) {  
 CoroutineScope(Dispatchers.IO).launch {  
 val db = AppDatabase.getDatabase(applicationContext)  
 val part = db.partDao().getPartById(partId)  
 withContext(Dispatchers.Main) {  
 part?.let {  
 etPartName.setText(it.name)  
 etPartWeight.setText(it.weight.toString())  
 etPartMaterial.setText(it.material)  
 etPartDiameter.setText(it.diameter.toString())  
 etPartPhoto.setText(it.photo)  
 spinnerPartType.setSelection(partTypes.indexOfFirst { type -> type.partTypeId == it.partTypeId })  
 }  
 }  
 }  
 }  
  
 private fun updatePartDetails() {  
 val name = etPartName.text.toString()  
 val weight = etPartWeight.text.toString().toDoubleOrNull() ?: 0.0  
 val material = etPartMaterial.text.toString()  
 val diameter = etPartDiameter.text.toString().toDoubleOrNull() ?: 0.0  
 val photo = etPartPhoto.text.toString()  
  
 if (name.isEmpty() || weight == 0.0 || material.isEmpty() || diameter == 0.0) {  
 Toast.makeText(this, "Заполните все поля", Toast.LENGTH\_SHORT).show()  
 } else {  
 if (partId != -1) {  
 CoroutineScope(Dispatchers.IO).launch {  
 val db = AppDatabase.getDatabase(applicationContext)  
 val selectedPartType = partTypes[spinnerPartType.selectedItemPosition]  
 val updatedPart = Part(  
 partId = partId,  
 partTypeId = selectedPartType.partTypeId,  
 name = name,  
 weight = weight,  
 material = material,  
 diameter = diameter,  
 photo = photo  
 )  
 db.partDao().update(updatedPart)  
 withContext(Dispatchers.Main) {  
 Toast.makeText(this@EditPartActivity, "Деталь обновлена", Toast.LENGTH\_SHORT).show()  
 finish()  
 }  
 }  
 }  
 }  
 }  
}

package com.example.zd5\_v8  
  
import android.os.Bundle  
import android.widget.Button  
import android.widget.EditText  
import android.widget.Toast  
import androidx.appcompat.app.AppCompatActivity  
import androidx.core.widget.doOnTextChanged  
import com.example.zd5\_v8.database.AppDatabase  
import com.example.zd5\_v8.database.Supplier  
import kotlinx.coroutines.CoroutineScope  
import kotlinx.coroutines.Dispatchers  
import kotlinx.coroutines.launch  
import kotlinx.coroutines.withContext  
  
class EditSupplierActivity : AppCompatActivity() {  
 private lateinit var emailEditText: EditText  
 private lateinit var passwordEditText: EditText  
 private lateinit var nameEditText: EditText  
 private lateinit var addressEditText: EditText  
 private lateinit var btnSave: Button  
 private var supplierId: Int = 0  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.activity\_edit\_supplier)  
  
 supplierId = intent.getIntExtra("SUPPLIER\_ID", -1)  
 if (supplierId == -1) {  
 Toast.makeText(this, "Ошибка загрузки данных поставщика!", Toast.LENGTH\_LONG).show()  
 finish()  
 }  
  
 emailEditText = findViewById(R.id.etSupplierEmail)  
 passwordEditText = findViewById(R.id.etSupplierPassword)  
 nameEditText = findViewById(R.id.etSupplierName)  
 addressEditText = findViewById(R.id.etSupplierAddress)  
 btnSave = findViewById(R.id.btnSave)  
  
  
 emailEditText.doOnTextChanged { text, \_, \_, \_ ->  
 if (text != null && !isValidEmail(text.toString())) {  
 emailEditText.error = "Неверный формат email"  
 }  
 }  
  
 passwordEditText.doOnTextChanged { text, \_, \_, \_ ->  
 if (text != null && text.length < 6) {  
 passwordEditText.error = "Пароль должен быть не менее 6 символов"  
 }  
 }  
  
 nameEditText.doOnTextChanged { text, \_, \_, \_ ->  
 if (text != null && !startsWithCapital(text)) {  
 nameEditText.error = "Имя должно начинаться с заглавной буквы"  
 }  
 }  
  
 loadSupplierData(supplierId)  
  
 btnSave.setOnClickListener {  
 updateSupplierData()  
 }  
 }  
  
 private fun isValidEmail(email: CharSequence?): Boolean {  
 return android.util.Patterns.EMAIL\_ADDRESS.matcher(email).matches()  
 }  
  
 private fun startsWithCapital(name: CharSequence?): Boolean {  
 if (name != null) {  
 return name.matches(Regex("^[A-ZА-Я].\*"))  
 }  
 return false  
 }  
  
 override fun onResume() {  
 super.onResume()  
 loadSupplierData(supplierId)  
 }  
  
 private fun loadSupplierData(supplierId: Int) {  
 CoroutineScope(Dispatchers.IO).launch {  
 val db = AppDatabase.getDatabase(applicationContext)  
 val supplier = db.supplierDao().getSupplierById(supplierId)  
 withContext(Dispatchers.Main) {  
 supplier?.let {  
 emailEditText.setText(it.email)  
 passwordEditText.setText(it.password)  
 nameEditText.setText(it.name)  
 addressEditText.setText(it.address)  
 }  
 }  
 }  
 }  
  
 private fun updateSupplierData() {  
 val email = emailEditText.text.toString().trim()  
 val password = passwordEditText.text.toString().trim()  
 val name = nameEditText.text.toString().trim()  
 val address = addressEditText.text.toString().trim()  
  
 if (email.isEmpty() || password.isEmpty() || name.isEmpty() || address.isEmpty()) {  
 Toast.makeText(this, "Пожалуйста заполните все поля", Toast.LENGTH\_SHORT).show()  
 return  
 }  
  
 CoroutineScope(Dispatchers.IO).launch {  
 val db = AppDatabase.getDatabase(applicationContext)  
 val updatedSupplier = Supplier(  
 supplierId = supplierId,  
 email = email,  
 password = password,  
 name = name,  
 address = address  
 )  
 db.supplierDao().update(updatedSupplier)  
 withContext(Dispatchers.Main) {  
 Toast.makeText(applicationContext, "Данные поставщика обновлены", Toast.LENGTH\_SHORT).show()  
 finish()  
 }  
 }  
 }  
  
}

package com.example.zd5\_v8  
  
import android.content.Intent  
import androidx.appcompat.app.AppCompatActivity  
import android.os.Bundle  
import android.view.View  
import android.widget.AdapterView  
import android.widget.ArrayAdapter  
import android.widget.Button  
import android.widget.EditText  
import android.widget.Spinner  
import android.widget.Toast  
import androidx.core.widget.doOnTextChanged  
import com.example.zd5\_v8.database.AppDatabase  
import com.example.zd5\_v8.database.Client  
import com.example.zd5\_v8.database.Supplier  
import com.example.zd5\_v8.database.Worker  
import kotlinx.coroutines.CoroutineScope  
import kotlinx.coroutines.Dispatchers  
import kotlinx.coroutines.launch  
import kotlinx.coroutines.withContext  
  
class LoginActivity : AppCompatActivity() {  
 private lateinit var emailEditText: EditText  
 private lateinit var passwordEditText: EditText  
 private lateinit var roleSpinner: Spinner  
 private lateinit var loginButton: Button  
 private lateinit var registerButton: Button  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_login*)  
  
 emailEditText = findViewById(R.id.*emailEditText*)  
 passwordEditText = findViewById(R.id.*passwordEditText*)  
 roleSpinner = findViewById(R.id.*roleSpinner*)  
 loginButton = findViewById(R.id.*loginButton*)  
 registerButton = findViewById(R.id.*registerButton*)  
  
 val roles = *arrayOf*("Работник", "Клиент", "Поставщик")  
 val adapter = ArrayAdapter(this, android.R.layout.*simple\_spinner\_dropdown\_item*, roles)  
 roleSpinner.*adapter* = adapter  
 roleSpinner.*onItemSelectedListener* = object : AdapterView.*OnItemSelectedListener* {  
 override fun onItemSelected(parent: AdapterView<\*>, view: View?, position: Int, id: Long) {  
 if (view != null) {  
 emailEditText.*text*.clear()  
 passwordEditText.*text*.clear()  
 }  
 }  
 override fun onNothingSelected(parent: AdapterView<\*>) {  
 }  
 }  
  
 emailEditText.*doOnTextChanged* { text, \_, \_, \_ ->  
 if (text != null && !isValidEmail(text)) {  
 emailEditText.*error* = "Неверный формат email"  
 }  
 }  
  
 passwordEditText.*doOnTextChanged* { text, \_, \_, \_ ->  
 if (text != null && text.length < 6) {  
 passwordEditText.*error* = "Пароль должен быть не менее 6 символов"  
 }  
 }  
  
 val worker = Worker(email = "worker@mail.ru", password = "worker", name = "Евгений")  
 *CoroutineScope*(Dispatchers.IO).*launch* {  
 val db = AppDatabase.getDatabase(*applicationContext*)  
 db.workerDao().insert(worker)  
 }  
  
  
 loginButton.setOnClickListener {  
 val email = emailEditText.*text*.toString()  
 val password = passwordEditText.*text*.toString()  
 val role = roleSpinner.*selectedItem*.toString()  
 if (validateLogin(email, password)) {  
 checkCredentials(email, password, role)  
 }else {  
 Toast.makeText(this@LoginActivity, "Неверный ввод. Проверьте данные.", Toast.*LENGTH\_SHORT*).show()  
 }  
 }  
  
 registerButton.setOnClickListener{  
 val intent = Intent(this@LoginActivity, RegisterActivity::class.*java*)  
 startActivity(intent)  
 }  
 }  
 private fun validateLogin(email: String, password: String): Boolean {  
 if (!isValidEmail(email)) {  
 return false  
 }  
  
 if (password.length < 6) {  
 return false  
 }  
  
 return true  
 }  
  
 private fun checkCredentials(email: String, password: String, role: String) {  
 *CoroutineScope*(Dispatchers.IO).*launch* {  
 val db = AppDatabase.getDatabase(*applicationContext*)  
  
 when (role) {  
 "Работник" -> {  
 val worker = db.workerDao().findWorkerByEmail(email)  
 withContext(Dispatchers.Main) {  
 if (worker != null && worker.password == password) {  
 navigateToWorkerActivity(worker)  
 } else {  
 Toast.makeText(this@LoginActivity, "Неверный ввод для Работника. Проверьте данные.", Toast.*LENGTH\_SHORT*).show()  
 }  
 }  
 }  
 "Клиент" -> {  
 val client = db.clientDao().findClientByEmail(email)  
 withContext(Dispatchers.Main) {  
 if (client != null && client.password == password) {  
 navigateToClientActivity(client)  
 } else {  
 Toast.makeText(this@LoginActivity, "Неверный ввод для Клиента. Проверьте данные.", Toast.*LENGTH\_SHORT*).show()  
 }  
 }  
 }  
 "Поставщик" -> {  
 val supplier = db.supplierDao().findSupplierByEmail(email)  
 withContext(Dispatchers.Main) {  
 if (supplier != null && supplier.password == password) {  
 navigateToSupplierActivity(supplier)  
 } else {  
 Toast.makeText(this@LoginActivity, "Неверный ввод для Поставщика. Проверьте данные.", Toast.*LENGTH\_SHORT*).show()  
 }  
 }  
 }  
 else -> {  
 withContext(Dispatchers.Main) {  
 Toast.makeText(this@LoginActivity, "Неизвестная роль: $role", Toast.*LENGTH\_SHORT*).show()  
 }  
 }  
 }  
 }  
 }  
  
 private fun navigateToWorkerActivity(worker: Worker) {  
 val intent = Intent(this@LoginActivity, WorkerActivity::class.*java*)  
 startActivity(intent)  
 }  
  
 private fun navigateToClientActivity(client: Client) {  
 val intent = Intent(this@LoginActivity, ClientActivity::class.*java*)  
 startActivity(intent)  
 }  
  
 private fun navigateToSupplierActivity(supplier: Supplier) {  
 val intent = Intent(this@LoginActivity, SupplierActivity::class.*java*)  
 startActivity(intent)  
 }  
  
 fun isValidEmail(email: *CharSequence*?): Boolean {  
 return android.util.Patterns.*EMAIL\_ADDRESS*.matcher(email).matches()  
 }  
}

package com.example.zd5\_v8  
  
import android.content.Intent  
import android.os.Bundle  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import androidx.appcompat.app.AlertDialog  
import androidx.fragment.app.Fragment  
import androidx.recyclerview.widget.LinearLayoutManager  
import androidx.recyclerview.widget.RecyclerView  
import com.example.zd5\_v8.database.AppDatabase  
import com.example.zd5\_v8.database.Client  
import kotlinx.coroutines.CoroutineScope  
import kotlinx.coroutines.Dispatchers  
import kotlinx.coroutines.launch  
import kotlinx.coroutines.withContext  
  
class ManageClientsFragment : Fragment() {  
  
 private lateinit var recyclerView: RecyclerView  
 private lateinit var clientAdapter: ClientAdapter  
  
 override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?): View? {  
 val view = inflater.inflate(R.layout.fragment\_manage\_clients, container, false)  
  
 recyclerView = view.findViewById(R.id.rvClients)  
 recyclerView.layoutManager = LinearLayoutManager(context)  
 clientAdapter = ClientAdapter(listOf(), onEditClick = { client ->  
 val intent = Intent(context, EditClientActivity::class.java)  
 intent.putExtra("CLIENT\_ID", client.clientId)  
 startActivity(intent)  
 }, onDeleteClick = { client ->  
 showDeleteDialog(client)  
 })  
  
 recyclerView.adapter = clientAdapter  
 loadClientsFromDb()  
  
 return view  
 }  
  
 override fun onResume() {  
 super.onResume()  
 loadClientsFromDb()  
 }  
  
 private fun loadClientsFromDb() {  
 CoroutineScope(Dispatchers.IO).launch {  
 val db = AppDatabase.getDatabase(requireContext())  
 val clients = db.clientDao().getAllClients()  
 withContext(Dispatchers.Main) {  
 clientAdapter.updateData(clients)  
 }  
 }  
 }  
  
 private fun showDeleteDialog(client: Client) {  
 AlertDialog.Builder(requireContext())  
 .setTitle("Подтвердите действие")  
 .setMessage("Вы уверены, что хотите удалить этого клиента?")  
 .setPositiveButton("Удалить") { \_, \_ -> deleteClient(client) }  
 .setNegativeButton("Отмена", null)  
 .show()  
 }  
  
 private fun deleteClient(client: Client) {  
 CoroutineScope(Dispatchers.IO).launch {  
 val db = AppDatabase.getDatabase(requireContext())  
 db.clientDao().delete(client)  
 loadClientsFromDb()  
 }  
 }  
}

package com.example.zd5\_v8  
  
import android.content.Intent  
import android.os.Bundle  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import androidx.appcompat.app.AlertDialog  
import androidx.fragment.app.Fragment  
import androidx.recyclerview.widget.LinearLayoutManager  
import androidx.recyclerview.widget.RecyclerView  
import com.example.zd5\_v8.database.AppDatabase  
import com.example.zd5\_v8.database.Supplier  
import kotlinx.coroutines.CoroutineScope  
import kotlinx.coroutines.Dispatchers  
import kotlinx.coroutines.launch  
import kotlinx.coroutines.withContext  
  
class ManageSuppliersFragment : Fragment() {  
  
 private lateinit var recyclerView: RecyclerView  
 private lateinit var supplierAdapter: SupplierAdapter  
  
 override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?): View? {  
 val view = inflater.inflate(R.layout.fragment\_manage\_suppliers, container, false)  
  
 recyclerView = view.findViewById(R.id.rvSuppliers)  
 recyclerView.layoutManager = LinearLayoutManager(context)  
 supplierAdapter = SupplierAdapter(listOf(), onEditClick = { supplier ->  
 val intent = Intent(context, EditSupplierActivity::class.java)  
 intent.putExtra("SUPPLIER\_ID", supplier.supplierId)  
 startActivity(intent)  
 }, onDeleteClick = { supplier ->  
 showDeleteDialog(supplier)  
 })  
  
 recyclerView.adapter = supplierAdapter  
 loadSuppliersFromDb()  
  
 return view  
 }  
  
 override fun onResume() {  
 super.onResume()  
 loadSuppliersFromDb()  
 }  
  
 private fun loadSuppliersFromDb() {  
 CoroutineScope(Dispatchers.IO).launch {  
 val db = AppDatabase.getDatabase(requireContext())  
 val suppliers = db.supplierDao().getAllSuppliers()  
 withContext(Dispatchers.Main) {  
 supplierAdapter.updateData(suppliers)  
 }  
 }  
 }  
  
 private fun showDeleteDialog(supplier: Supplier) {  
 AlertDialog.Builder(requireContext())  
 .setTitle("Подтвердите действие")  
 .setMessage("Вы уверены, что хотите удалить этого поставщика?")  
 .setPositiveButton("Удалить") { \_, \_ -> deleteSupplier(supplier) }  
 .setNegativeButton("Отмена", null)  
 .show()  
 }  
  
 private fun deleteSupplier(supplier: Supplier) {  
 CoroutineScope(Dispatchers.IO).launch {  
 val db = AppDatabase.getDatabase(requireContext())  
 db.supplierDao().delete(supplier)  
 loadSuppliersFromDb()  
 }  
 }  
}

package com.example.zd5\_v8  
  
import android.content.Intent  
import android.os.Bundle  
import android.view.View  
import android.widget.AdapterView  
import androidx.appcompat.app.AppCompatActivity  
import android.widget.ArrayAdapter  
import android.widget.Button  
import android.widget.EditText  
import android.widget.Spinner  
import android.widget.Toast  
import androidx.core.widget.doOnTextChanged  
import com.example.zd5\_v8.database.AppDatabase  
import com.example.zd5\_v8.database.Client  
import com.example.zd5\_v8.database.Supplier  
import kotlinx.coroutines.\*  
  
class RegisterActivity : AppCompatActivity() {  
 private lateinit var emailEditText: EditText  
 private lateinit var passwordEditText: EditText  
 private lateinit var roleSpinner: Spinner  
 private lateinit var registerButton: Button  
 private lateinit var nameEditText: EditText  
 private lateinit var addressEditText: EditText  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.activity\_register)  
  
 emailEditText = findViewById(R.id.emailEditText)  
 passwordEditText = findViewById(R.id.passwordEditText)  
 roleSpinner = findViewById(R.id.roleSpinner)  
 registerButton = findViewById(R.id.registerButton)  
 nameEditText = findViewById(R.id.nameEditText)  
 addressEditText = findViewById(R.id.addressEditText)  
  
 val roles = arrayOf("Клиент", "Поставщик")  
 val adapter = ArrayAdapter(this, android.R.layout.simple\_spinner\_dropdown\_item, roles)  
 roleSpinner.adapter = adapter  
 roleSpinner.onItemSelectedListener = object : AdapterView.OnItemSelectedListener {  
 override fun onItemSelected(parent: AdapterView<\*>, view: View, position: Int, id: Long) {  
 emailEditText.text.clear()  
 passwordEditText.text.clear()  
 nameEditText.text.clear()  
 addressEditText.text.clear()  
  
 if (roles[position] == "Поставщик") {  
 addressEditText.visibility = View.VISIBLE  
 } else {  
 addressEditText.visibility = View.GONE  
 }  
 }  
  
 override fun onNothingSelected(parent: AdapterView<\*>) {  
  
 }  
 }  
  
 emailEditText.doOnTextChanged { text, \_, \_, \_ ->  
 if (text != null && !isValidEmail(text)) {  
 emailEditText.error = "Неверный формат email"  
 }  
 }  
  
 passwordEditText.doOnTextChanged { text, \_, \_, \_ ->  
 if (text != null && text.length < 6) {  
 passwordEditText.error = "Пароль должен быть не менее 6 символов"  
 }  
 }  
  
 nameEditText.doOnTextChanged { text, \_, \_, \_ ->  
 if (text != null && !startsWithCapital(text)) {  
 nameEditText.error = "Имя должно начинаться с заглавной буквы"  
 }  
 }  
  
 registerButton.setOnClickListener {  
 val email = emailEditText.text.toString()  
 val password = passwordEditText.text.toString()  
 val name = nameEditText.text.toString()  
 val role = roleSpinner.selectedItem.toString()  
 val address = addressEditText.text.toString()  
  
 if (validateRegistration(email, password, name, role, address)) {  
 saveToSharedPreferences(email, password, role)  
 saveToDatabase(email, password, name, role, address)  
  
 Toast.makeText(this@RegisterActivity, "Регистрация успешна", Toast.LENGTH\_SHORT).show()  
  
 val intent = Intent(this@RegisterActivity, LoginActivity::class.java)  
 startActivity(intent)  
 } else {  
 Toast.makeText(this@RegisterActivity, "Ошибка регистрации", Toast.LENGTH\_SHORT).show()  
 }  
 }  
 }  
 private fun startsWithCapital(name: CharSequence?): Boolean {  
 if (name != null) {  
 return name.matches(Regex("^[A-ZА-Я].\*"))  
 }  
 return false  
 }  
  
 fun isValidEmail(email: CharSequence?): Boolean {  
 return android.util.Patterns.EMAIL\_ADDRESS.matcher(email).matches()  
 }  
  
 private fun validateRegistration(email: String, password: String, name: String, role: String, address: String): Boolean {  
 var isValid = false  
  
 runBlocking {  
 val db = AppDatabase.getDatabase(applicationContext)  
 val existingClient = withContext(Dispatchers.IO) { db.clientDao().findClientByEmail(email) }  
 val existingSupplier = withContext(Dispatchers.IO) { db.supplierDao().findSupplierByEmail(email) }  
  
 if (existingClient == null && existingSupplier == null) {  
 isValid = true  
 } else{  
 Toast.makeText(this@RegisterActivity, "Email уже используется", Toast.LENGTH\_SHORT).show()  
 }  
 if (!startsWithCapital(name)) {  
 runOnUiThread { Toast.makeText(this@RegisterActivity, "Имя должно начинаться с заглавной буквы", Toast.LENGTH\_SHORT).show() }  
 isValid = false  
 }  
  
 if (!isValidEmail(email) || password.length < 6 || !startsWithCapital(name) || name.isEmpty() || (role == "Поставщик" && address.isEmpty())) {  
 isValid = false  
 }  
  
 if (!isValidEmail(email)) {  
 isValid = false  
 }  
  
 if (password.length < 6) {  
 isValid = false  
 }  
 }  
  
 return isValid  
 }  
  
 private fun saveToSharedPreferences(email: String, password: String, role: String) {  
 val sharedPreferences = getSharedPreferences("RegistrationPrefs", MODE\_PRIVATE)  
 sharedPreferences.edit().apply {  
 putString("Email", email)  
 putString("Password", password)  
 putString("Role", role)  
 apply()  
 }  
 }  
  
 private fun saveToDatabase(email: String, password: String, name: String, role: String, address: String) {  
 CoroutineScope(Dispatchers.IO).launch {  
 val db = AppDatabase.getDatabase(applicationContext)  
  
 when (role) {  
 "Клиент" -> {  
 val newClient = Client(email = email, password = password, name = name, discount = 0.0)  
 db.clientDao().insert(newClient)  
 }  
 "Поставщик" -> {  
 val newSupplier = Supplier(email = email, password = password, name = name, address = address)  
 db.supplierDao().insert(newSupplier)  
 }  
 }  
 }  
 }  
}

package com.example.zd5\_v8  
  
import android.content.Intent  
import android.os.Bundle  
import android.view.View  
import android.widget.AdapterView  
import android.widget.ArrayAdapter  
import android.widget.Button  
import android.widget.Spinner  
import androidx.appcompat.app.AlertDialog  
import androidx.appcompat.app.AppCompatActivity  
import androidx.recyclerview.widget.LinearLayoutManager  
import androidx.recyclerview.widget.RecyclerView  
import com.example.zd5\_v8.database.AppDatabase  
import com.example.zd5\_v8.database.Part  
import com.example.zd5\_v8.database.PartAdapter  
import com.example.zd5\_v8.database.PartType  
import kotlinx.coroutines.CoroutineScope  
import kotlinx.coroutines.Dispatchers  
import kotlinx.coroutines.launch  
import kotlinx.coroutines.withContext  
  
class SupplierActivity : AppCompatActivity() {  
 private lateinit var recyclerView: RecyclerView  
 private lateinit var partsAdapter: PartAdapter  
 private lateinit var btnAddPart: Button  
 private lateinit var btnAddPartType: Button  
 private lateinit var partTypeSpinner: Spinner  
 private var partTypes: *List*<PartType> = *listOf*()  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_supplier*)  
  
 recyclerView = findViewById(R.id.*rvParts*)  
 recyclerView.*layoutManager* = LinearLayoutManager(this)  
  
 partsAdapter = PartAdapter(  
 *emptyList*(),  
 onItemClick = { part -> onItemClick(part) },  
 onItemLongClick = { part -> showDeleteDialog(part) }  
 )  
 recyclerView.*adapter* = partsAdapter  
  
 partTypeSpinner = findViewById(R.id.*categorySpinner*)  
 loadPartTypes()  
  
 btnAddPart = findViewById(R.id.*btnAddParts*)  
 btnAddPart.setOnClickListener {  
 val intent = Intent(this, AddPartActivity::class.*java*)  
 startActivity(intent)  
 }  
  
 btnAddPartType = findViewById(R.id.*btnAddPartsType*)  
 btnAddPartType.setOnClickListener {  
 val intent = Intent(this, AddPartTypeActivity::class.*java*)  
 startActivity(intent)  
 }  
  
 loadParts()  
 }  
  
 override fun onResume() {  
 super.onResume()  
 loadParts()  
 }  
  
 private fun showDeleteDialog(part: Part) {  
 AlertDialog.Builder(this).*apply* {  
 setTitle("Удаление детали")  
 setMessage("Вы уверены, что хотите удалить эту деталь?")  
 setPositiveButton("Удалить") { \_, \_ -> deletePart(part) }  
 setNegativeButton("Отмена", null)  
 show()  
 }  
 }  
  
 private fun deletePart(part: Part) {  
 *CoroutineScope*(Dispatchers.IO).*launch* {  
 val db = AppDatabase.getDatabase(*applicationContext*)  
 db.partDao().delete(part)  
 loadParts()  
 }  
 }  
  
 private fun loadPartTypes() {  
 *CoroutineScope*(Dispatchers.IO).*launch* {  
 val db = AppDatabase.getDatabase(*applicationContext*)  
 partTypes = db.partTypeDao().getAllPartTypes()  
 val categories = *listOf*("Все категории") + partTypes.*map* { **it**.partName }  
 withContext(Dispatchers.Main) {  
 val adapter = ArrayAdapter(this@SupplierActivity, android.R.layout.*simple\_spinner\_item*, categories)  
 adapter.setDropDownViewResource(android.R.layout.*simple\_spinner\_dropdown\_item*)  
 partTypeSpinner.*adapter* = adapter  
 partTypeSpinner.*onItemSelectedListener* = object : AdapterView.*OnItemSelectedListener* {  
 override fun onItemSelected(parent: AdapterView<\*>, view: View, position: Int, id: Long) {  
 if (position == 0) {  
 loadParts()  
 } else {  
 filterPartsByType(partTypes[position - 1].partTypeId)  
 }  
 }  
 override fun onNothingSelected(parent: AdapterView<\*>) {}  
 }  
 }  
 }  
 }  
  
 private fun filterPartsByType(partTypeId: Int) {  
 *CoroutineScope*(Dispatchers.IO).*launch* {  
 val db = AppDatabase.getDatabase(*applicationContext*)  
 val filteredParts = if (partTypeId == 0) {  
 db.partDao().getAllParts()  
 } else {  
 db.partDao().getPartsByType(partTypeId)  
 }  
 withContext(Dispatchers.Main) {  
 partsAdapter.updateParts(filteredParts)  
 }  
 }  
 }  
  
  
 private fun loadParts() {  
 *CoroutineScope*(Dispatchers.IO).*launch* {  
 val db = AppDatabase.getDatabase(*applicationContext*)  
 val parts = db.partDao().getAllParts()  
 withContext(Dispatchers.Main) {  
 partsAdapter.updateParts(parts)  
 }  
 }  
 }  
 private fun onItemClick(part: Part) {  
 val intent = Intent(this, EditPartActivity::class.*java*)  
 intent.putExtra("PART\_ID", part.partId)  
 startActivity(intent)  
 }  
}

package com.example.zd5\_v8  
  
import android.os.Bundle  
import androidx.appcompat.app.AppCompatActivity  
import androidx.navigation.fragment.NavHostFragment  
import androidx.navigation.ui.setupWithNavController  
import com.google.android.material.bottomnavigation.BottomNavigationView  
  
class WorkerActivity : AppCompatActivity() {  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_worker*)  
  
 val hostFragment = *supportFragmentManager*.findFragmentById(R.id.*nav\_host\_fragment*) as NavHostFragment  
 val navController = hostFragment.navController  
  
 val bottomNav = findViewById<BottomNavigationView>(R.id.*bottom\_navigation*)  
 bottomNav.*setupWithNavController*(navController)  
 }  
}

package com.example.zd5\_v8  
  
import android.content.Intent  
import android.os.Bundle  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import android.widget.Button  
import androidx.appcompat.app.AlertDialog  
import androidx.fragment.app.Fragment  
import androidx.recyclerview.widget.LinearLayoutManager  
import androidx.recyclerview.widget.RecyclerView  
import com.example.zd5\_v8.database.AppDatabase  
import com.example.zd5\_v8.database.Furniture  
import com.example.zd5\_v8.database.FurnitureAdapter  
import kotlinx.coroutines.CoroutineScope  
import kotlinx.coroutines.Dispatchers  
import kotlinx.coroutines.launch  
import kotlinx.coroutines.withContext  
  
class WorkerFragment : Fragment() {  
 private lateinit var recyclerView: RecyclerView  
 private lateinit var furnitureAdapter: FurnitureAdapter  
  
 override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?): View {  
 val view = inflater.inflate(R.layout.fragment\_worker, container, false)  
  
 recyclerView = view.findViewById(R.id.rvFurniture)  
 recyclerView.layoutManager = LinearLayoutManager(context)  
  
 furnitureAdapter = FurnitureAdapter(listOf(), this::onItemClick, this::onItemLongClick)  
 recyclerView.adapter = furnitureAdapter  
  
 loadFurnitureFromDb()  
  
 view.findViewById<Button>(R.id.btnAddFurniture).setOnClickListener {  
 val intent = Intent(requireContext(), AddFurnitureActivity::class.java)  
 startActivity(intent)  
 }  
  
 view.findViewById<Button>(R.id.btnAddFurnitureType).setOnClickListener {  
 val intent = Intent(requireContext(), AddFurnitureTypeActivity::class.java)  
 startActivity(intent)  
 }  
  
 return view  
 }  
  
 override fun onResume() {  
 super.onResume()  
 loadFurnitureFromDb()  
 }  
  
 private fun onItemClick(furniture: Furniture) {  
 val intent = Intent(requireContext(), EditFurnitureActivity::class.java)  
 intent.putExtra("FURNITURE\_ID", furniture.furnitureId)  
 startActivity(intent)  
 }  
  
 private fun onItemLongClick(furniture: Furniture) {  
 AlertDialog.Builder(requireContext()).apply {  
 setTitle("Удаление")  
 setMessage("Вы действительно хотите удалить?")  
 setPositiveButton("Да") { \_, \_ ->  
 deleteFurniture(furniture)  
 }  
 setNegativeButton("Нет", null)  
 show()  
 }  
 }  
  
 private fun deleteFurniture(furniture: Furniture) {  
 CoroutineScope(Dispatchers.IO).launch {  
 val db = AppDatabase.getDatabase(requireContext())  
 db.furnitureDao().delete(furniture)  
 withContext(Dispatchers.Main) {  
  
 loadFurnitureFromDb()  
 }  
 }  
 }  
  
 private fun loadFurnitureFromDb() {  
 CoroutineScope(Dispatchers.IO).launch {  
 val db = AppDatabase.getDatabase(requireContext())  
 val furniture = db.furnitureDao().getAllFurnitures()  
 withContext(Dispatchers.Main) {  
 furnitureAdapter.updateData(furniture)  
 }  
 }  
 }  
}

Код макетов и фрагментов:  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:orientation="vertical"  
 android:padding="16dp"  
 tools:context=".AddFurnitureActivity">  
  
 <!-- EditText для имени -->  
 <EditText  
 android:id="@+id/etFurnitureName"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Название"/>  
  
 <!-- EditText для описания -->  
 <EditText  
 android:id="@+id/etFurnitureDescription"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Описание"/>  
  
 <!-- EditText для цены -->  
 <EditText  
 android:id="@+id/etFurnitureCost"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="numberDecimal"  
 android:hint="Цена"/>  
  
  
 <!-- EditText для фото (или Button для выбора фото) -->  
 <EditText  
 android:id="@+id/etFurniturePhoto"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Фото"/>  
  
 <!-- Spinner для типов мебели -->  
 <Spinner  
 android:id="@+id/spinnerFurnitureType"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"/>  
  
 <!-- Button для сохранения -->  
 <Button  
 android:id="@+id/btnSaveFurniture"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Сохранить"/>  
  
</LinearLayout>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:orientation="vertical"  
 android:padding="16dp"  
 tools:context=".AddFurnitureTypeActivity">  
  
 <EditText  
 android:id="@+id/etFurnitureName"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Название категории" />  
  
 <Button  
 android:id="@+id/btnSaveFurniture"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Сохранить"/>  
  
</LinearLayout>

<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">  
  
 <Spinner  
 android:id="@+id/spinnerPartType"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"/>  
  
 <EditText  
 android:id="@+id/etPartName"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Название"/>  
  
 <EditText  
 android:id="@+id/etPartWeight"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Вес"  
 android:inputType="numberDecimal"/>  
  
 <EditText  
 android:id="@+id/etPartMaterial"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Материал"/>  
  
 <EditText  
 android:id="@+id/etPartDiameter"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Диаметр"  
 android:inputType="numberDecimal"/>  
  
 <EditText  
 android:id="@+id/etPartPhoto"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Фото (URL)"/>  
  
 <Button  
 android:id="@+id/btnSavePart"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Добавить деталь"/>  
  
</LinearLayout>

<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">  
  
 <EditText  
 android:id="@+id/etPartTypeName"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Название типа детали"/>  
  
 <Button  
 android:id="@+id/btnSavePartType"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Сохранить"/>  
</LinearLayout>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".ClientActivity">  
  
 <Spinner  
 android:id="@+id/categorySpinner"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content" />  
  
 <androidx.recyclerview.widget.RecyclerView  
 android:id="@+id/rvFurniture"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_below="@id/categorySpinner" />  
</RelativeLayout>

<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">  
  
 <EditText  
 android:id="@+id/emailEditText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Email" />  
  
 <EditText  
 android:id="@+id/passwordEditText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Пароль" />  
  
 <EditText  
 android:id="@+id/nameEditText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Имя" />  
  
 <Button  
 android:id="@+id/btnSave"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Сохранить" />  
</LinearLayout>

<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">  
  
 <EditText  
 android:id="@+id/etFurnitureName"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Название мебели"/>  
  
 <EditText  
 android:id="@+id/etFurnitureDescription"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Описание"/>  
  
 <EditText  
 android:id="@+id/etFurnitureCost"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Цена"  
 android:inputType="numberDecimal"/>  
  
 <EditText  
 android:id="@+id/etImage"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Изображение"  
 android:inputType="numberDecimal"/>  
  
 <Button  
 android:id="@+id/btnSaveFurniture"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Сохранить"/>  
  
</LinearLayout>

<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">  
  
 <Spinner  
 android:id="@+id/spinnerPartType"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"/>  
  
 <EditText  
 android:id="@+id/etPartName"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Название"/>  
  
 <EditText  
 android:id="@+id/etPartWeight"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Вес"  
 android:inputType="numberDecimal"/>  
  
 <EditText  
 android:id="@+id/etPartMaterial"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Материал"/>  
  
 <EditText  
 android:id="@+id/etPartDiameter"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Диаметр"  
 android:inputType="numberDecimal"/>  
  
 <EditText  
 android:id="@+id/etPartPhoto"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Фото (URL)"/>  
  
 <Button  
 android:id="@+id/btnSavePart"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Добавить деталь"/>  
  
</LinearLayout>

<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">  
  
 <EditText  
 android:id="@+id/etSupplierEmail"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Email" />  
  
 <EditText  
 android:id="@+id/etSupplierPassword"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Пароль" />  
  
 <EditText  
 android:id="@+id/etSupplierName"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Имя" />  
  
 <EditText  
 android:id="@+id/etSupplierAddress"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Адрес" />  
  
 <Button  
 android:id="@+id/btnSave"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Сохранить" />  
</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">  
  
 <EditText  
 android:layout\_marginTop="200dp"  
 android:id="@+id/emailEditText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="@string/email"  
 android:inputType="textEmailAddress"/>  
  
 <EditText  
 android:id="@+id/passwordEditText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="@string/password"  
 android:inputType="textPassword"/>  
  
 <Spinner  
 android:id="@+id/roleSpinner"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"/>  
  
 <Button  
 android:id="@+id/loginButton"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="@string/login"/>  
  
 <Button  
 android:id="@+id/registerButton"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="@string/register"/>  
</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">  
  
 <EditText  
 android:layout\_marginTop="200dp"  
 android:id="@+id/emailEditText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="@string/email"  
 android:inputType="textEmailAddress"/>  
  
 <EditText  
 android:id="@+id/passwordEditText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="@string/password"  
 android:inputType="textPassword"/>  
  
 <EditText  
 android:id="@+id/nameEditText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="@string/name"  
 android:inputType="textPersonName"/>  
  
 <EditText  
 android:id="@+id/addressEditText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="@string/address"  
 android:inputType="textPostalAddress"  
 android:visibility="gone"/>  
  
 <Spinner  
 android:id="@+id/roleSpinner"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"/>  
  
 <Button  
 android:id="@+id/registerButton"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="@string/register"/>  
  
</LinearLayout>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".SupplierActivity">  
  
 <Spinner  
 android:id="@+id/categorySpinner"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content" />  
  
 <androidx.recyclerview.widget.RecyclerView  
 android:id="@+id/rvParts"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_below="@id/categorySpinner"  
 android:layout\_above="@id/btnAddParts"/>  
  
 <Button  
 android:layout\_marginStart="8dp"  
 android:id="@+id/btnAddParts"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentBottom="true"  
 android:text="Добавить материал"/>  
  
 <Button  
 android:layout\_marginStart="10dp"  
 android:id="@+id/btnAddPartsType"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_toEndOf="@id/btnAddParts"  
 android:layout\_alignParentBottom="true"  
 android:text="Добавить тип материала"/>  
  
</RelativeLayout>

<!-- res/layout/activity\_worker.xml -->  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".WorkerActivity">  
  
 <!-- Fragment container to display different fragments -->  
 <androidx.fragment.app.FragmentContainerView  
 android:id="@+id/nav\_host\_fragment"  
 android:name="androidx.navigation.fragment.NavHostFragment"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_above="@id/bottom\_navigation"  
 app:navGraph="@navigation/worker\_navigation"  
 app:defaultNavHost="true" />  
  
 <!-- Bottom Navigation View -->  
 <com.google.android.material.bottomnavigation.BottomNavigationView  
 android:id="@+id/bottom\_navigation"  
 android:layout\_width="match\_parent"  
 android:layout\_height="50dp"  
 android:layout\_alignParentBottom="true"  
 app:menu="@menu/bottom\_nav\_menu" />  
</RelativeLayout>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical">  
  
 <androidx.recyclerview.widget.RecyclerView  
 android:id="@+id/rvClients"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"/>  
  
</LinearLayout>

<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 <androidx.recyclerview.widget.RecyclerView  
 android:id="@+id/rvSuppliers"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"/>  
</FrameLayout>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".WorkerFragment">  
  
 <androidx.recyclerview.widget.RecyclerView  
 android:id="@+id/rvFurniture"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_above="@+id/btnAddFurniture"/>  
  
 <Button  
 android:layout\_marginStart="8dp"  
 android:id="@+id/btnAddFurniture"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentBottom="true"  
 android:text="Добавить мебель"/>  
  
 <Button  
 android:layout\_marginStart="10dp"  
 android:id="@+id/btnAddFurnitureType"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_toEndOf="@id/btnAddFurniture"  
 android:layout\_alignParentBottom="true"  
 android:text="Добавить тип мебели"/>  
  
</RelativeLayout>

<LinearLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 android:padding="16dp">  
  
 <TextView  
 android:id="@+id/tvClientName"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="5"  
 android:text="Имя Клиента" />  
  
 <TextView  
 android:id="@+id/tvClientEmail"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="3"  
 android:text="Email" />  
  
 <Button  
 android:id="@+id/btnEdit"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Редактировать" />  
  
</LinearLayout>

<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">  
  
 <ImageView  
 android:id="@+id/furniture\_photo"  
 android:layout\_width="match\_parent"  
 android:layout\_height="200dp"  
 android:scaleType="centerCrop"/>  
  
 <TextView  
 android:id="@+id/furniture\_name"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:padding="4dp"  
 android:textSize="18sp"/>  
  
 <TextView  
 android:id="@+id/furniture\_description"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:padding="4dp"  
 android:textSize="14sp"/>  
  
 <TextView  
 android:id="@+id/furniture\_cost"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:padding="4dp"  
 android:textSize="16sp"/>  
</LinearLayout>

<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">  
  
 <ImageView  
 android:id="@+id/parts\_photo"  
 android:layout\_width="match\_parent"  
 android:layout\_height="200dp"  
 android:scaleType="centerCrop"/>  
  
 <TextView  
 android:id="@+id/parts\_name"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:padding="4dp"  
 android:textSize="18sp"/>  
  
 <TextView  
 android:id="@+id/parts\_material"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:padding="4dp"  
 android:textSize="14sp"/>  
  
 <TextView  
 android:id="@+id/parts\_diameter"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:padding="4dp"  
 android:textSize="16sp"/>  
  
 <TextView  
 android:id="@+id/parts\_weight"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:padding="4dp"  
 android:textSize="16sp"/>  
</LinearLayout>

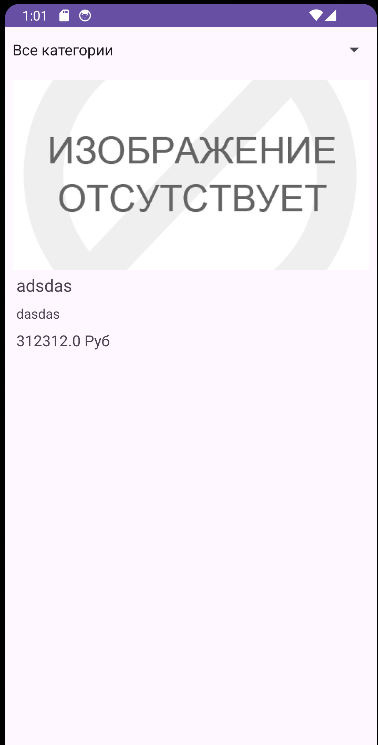
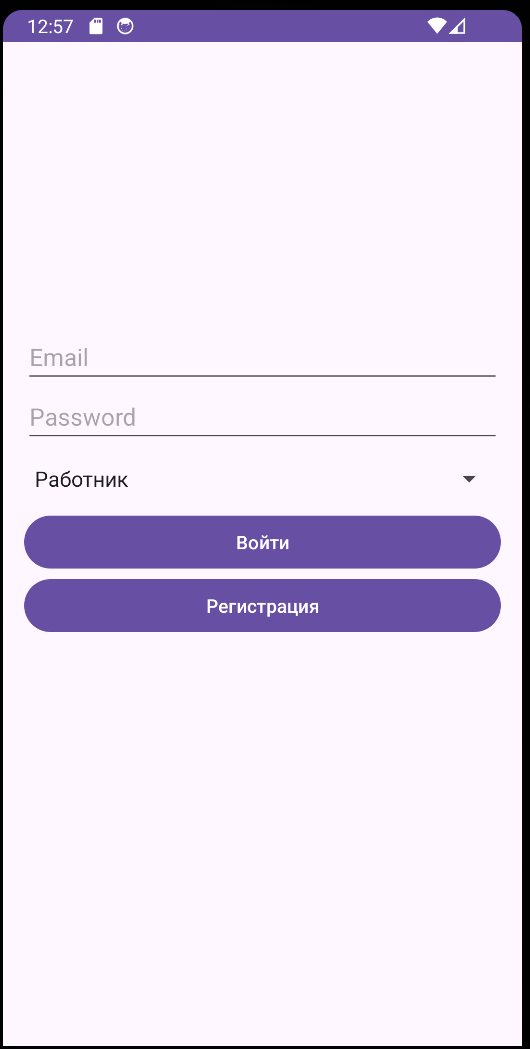
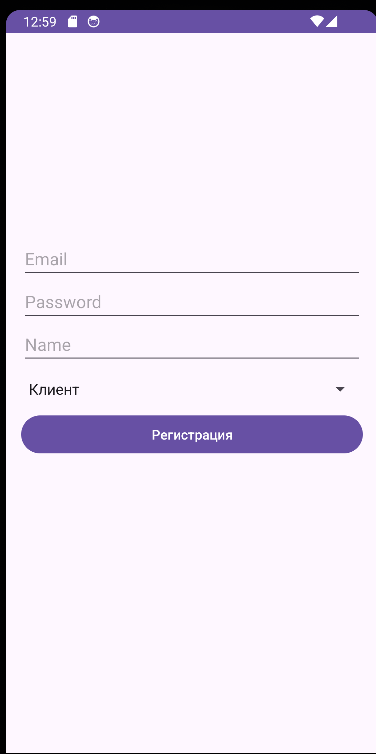
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 android:padding="16dp">  
  
 <TextView  
 android:id="@+id/tvSupplierName"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="3"  
 android:text="Имя Поставщика"  
 android:padding="4dp"/>  
  
 <TextView  
 android:id="@+id/tvSupplierEmail"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="2"  
 android:text="Email"  
 android:padding="4dp"/>  
  
 <Button  
 android:id="@+id/btnEdit"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Редактировать"/>  
</LinearLayout>

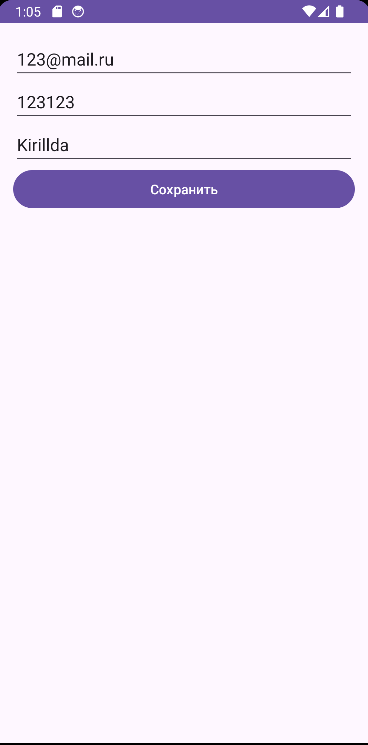
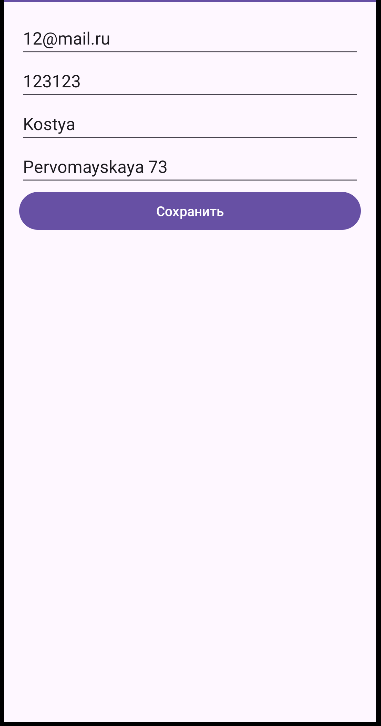
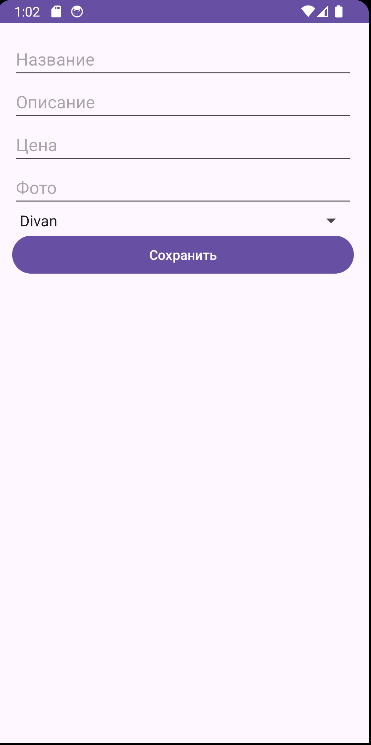
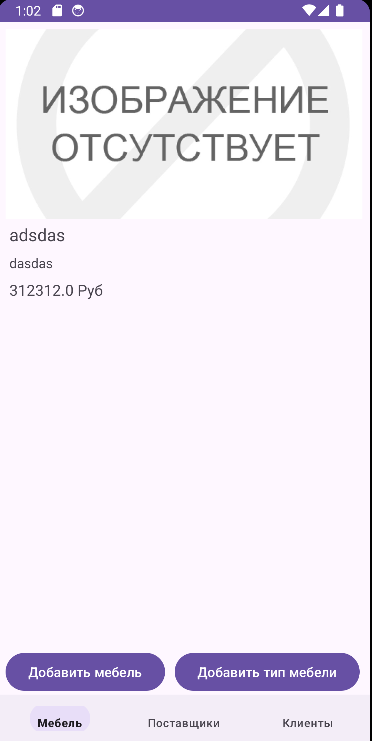
Навигация:

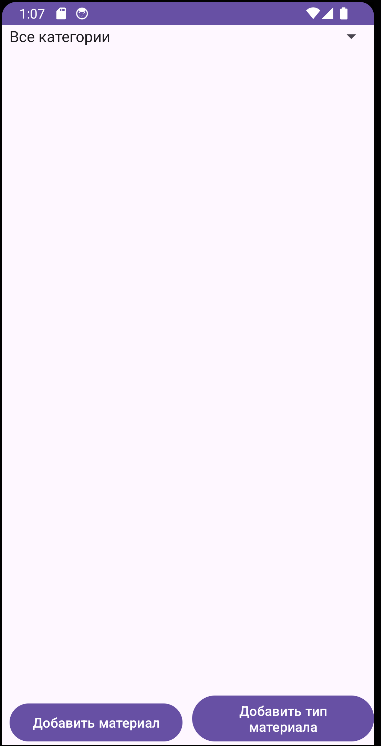
<navigation xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 app:startDestination="@id/workerFragment">  
  
 <!-- Worker Fragment -->  
 <fragment  
 android:id="@+id/workerFragment"  
 android:name="com.example.zd5\_v8.WorkerFragment"  
 android:label="Мебель" >  
 <!-- Action to navigate from Worker to Manage Suppliers -->  
 <action  
 android:id="@+id/action\_workerFragment\_to\_manageSuppliersFragment"  
 app:destination="@id/manageSuppliersFragment" />  
 <!-- Action to navigate from Worker to Manage Clients -->  
 <action  
 android:id="@+id/action\_workerFragment\_to\_manageClientsFragment"  
 app:destination="@id/manageClientsFragment" />  
 </fragment>  
  
 <!-- Manage Suppliers Fragment -->  
 <fragment  
 android:id="@+id/manageSuppliersFragment"  
 android:name="com.example.zd5\_v8.ManageSuppliersFragment"  
 android:label="Поставщики" >  
 <!-- Action to navigate back to Worker Fragment -->  
 <action  
 android:id="@+id/action\_manageSuppliersFragment\_to\_workerFragment"  
 app:destination="@id/workerFragment" />  
 <!-- Action to navigate from Suppliers to Manage Clients -->  
 <action  
 android:id="@+id/action\_manageSuppliersFragment\_to\_manageClientsFragment"  
 app:destination="@id/manageClientsFragment" />  
 </fragment>  
  
 <!-- Manage Clients Fragment -->  
 <fragment  
 android:id="@+id/manageClientsFragment"  
 android:name="com.example.zd5\_v8.ManageClientsFragment"  
 android:label="Клиенты" >  
 <!-- Action to navigate back to Worker Fragment -->  
 <action  
 android:id="@+id/action\_manageClientsFragment\_to\_workerFragment"  
 app:destination="@id/workerFragment" />  
 <!-- Action to navigate from Clients to Manage Suppliers -->  
 <action  
 android:id="@+id/action\_manageClientsFragment\_to\_manageSuppliersFragment"  
 app:destination="@id/manageSuppliersFragment" />  
 </fragment>  
</navigation>

<menu xmlns:android="http://schemas.android.com/apk/res/android">  
 <item  
 android:id="@+id/workerFragment"  
 android:title="Мебель" />  
 <item  
 android:id="@+id/manageSuppliersFragment"  
 android:title="Поставщики" />  
 <item  
 android:id="@+id/manageClientsFragment"  
 android:title="Клиенты" />  
</menu>

**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.