# ПРИЛОЖЕНИЕ Д. ТЕКСТ ПРОГРАММЫ

АННОТАЦИЯ

В данном программном документе приведён текст программы для мобильного приложения «Болотный Ясень».

В разделе «Текст программы» указано наименование приложения, область применения приложения, модули приложения в виде таблицы с указанием описания и размера каждого модуля, код программы.

1. ТЕКСТ ПРОГРАММЫ

1.1. Наименование программы

Полное наименование: «Разработка мобильного приложения для магазина гитар».

1.2. Область применения программы

Программа «Болотный Ясень» предназначена для личного использования с целью ведения записей о событиях дня, отслеживания своего настроения и друзей, организации повседневных задач. Основными пользователями являются люди, стремящиеся к структурированному ведению заметок, анализу эмоционального состояния и повышению продуктивности. Программа ориентирована на широкую аудиторию, включая студентов, профессионалов, людей, интересующихся саморазвитием и улучшением эмоционального благополучия.

1.3. Модули

Таблица 1. Модули

| **№** | **Название файла** | **Описание модуля** | **Размер** | **Кол-во строк** |
| --- | --- | --- | --- | --- |
| 1 | App.java | Инициализация приложения | 1 Кб | 58 |
| 2 | ProductsFragment.java | Отображение каталога товаров | 15 Кб | 320 |
| 3 | CartFragment.java | Функционал корзины | 12 Кб | 280 |
| 4 | OrderActivity.java | Оформление заказа | 10 Кб | 245 |
| 5 | ProductDetailActivity.java | Детальная информация о товаре | 8 Кб | 195 |
| 6 | AdminPanelFragment.java | Панель администратора | 14 Кб | 310 |
| 7 | EditProductDialog.java | Диалог редактирования товара | 6 Кб | 150 |
| 8 | FileUtils.java | Утилита для работы с файлами | 8 Кб | 180 |
| 9 | OrderHistoryFragment.java | История заказов | 9 Кб | 220 |
| 10 | ProfileFragment.java | Профиль пользователя | 7 Кб | 175 |
| 11 | ProductAdapter.java | Адаптер для списка товаров | 5 Кб | 125 |
| 12 | OrderAdapter.java | Адаптер для списка заказов | 5 Кб | 120 |
| 13 | CartAdapter.java | Адаптер корзины | 4 Кб | 110 |
| 14 | Product.java | Модель данных товара | 2 Кб | 45 |
| 15 | Order.java | Модель данных заказа | 2 Кб | 50 |
| 16 | User.java | Модель данных пользователя | 2 Кб | 55 |
| 17 | PaymentManager.java | Логика оплаты | 6 Кб | 145 |
| 18 | DatabaseHelper.java | Работа с базой данных | 12 Кб | 285 |
| 19 | LoginActivity.java | Авторизация | 5 Кб | 120 |
| 20 | MainActivity.java | Главная активность, навигация | 6 Кб | 140 |
| 21 | activity\_main.xml | Главный экран | 3 Кб | 65 |
| 22 | fragment\_products.xml | Каталог товаров | 4 Кб | 90 |
| 23 | fragment\_cart.xml | Корзина | 4 Кб | 85 |
| 24 | fragment\_profile.xml | Профиль пользователя | 3 Кб | 70 |
| 25 | fragment\_admin\_panel.xml | Админ-панель | 5 Кб | 100 |
| 26 | dialog\_edit\_product.xml | Диалог редактирования товара | 2 Кб | 40 |
| 27 | activity\_order.xml | Оформление заказа | 3 Кб | 75 |
| 28 | fragment\_order\_history.xml | История заказов | 3 Кб | 80 |
| 29 | activity\_login.xml | Авторизация | 2 Кб | 60 |
| 30 | activity\_product\_detail.xml | Детальный просмотр товара | 3 Кб | 70 |
| 31 | ic\_cart.xml | Иконка корзины | 1 Кб | 15 |
| 32 | ic\_profile.xml | Иконка профиля | 1 Кб | 16 |
| 33 | ic\_products.xml | Иконка товаров | 1 Кб | 16 |
| 34 | ic\_admin.xml | Иконка админ-панели | 1 Кб | 18 |
| 35 | background\_item.xml | Фон элемента | 2 Кб | 22 |
| 36 | colors.xml | Цветовая палитра | 1 Кб | 20 |
| 37 | strings.xml | Локализованные строки | 2 Кб | 60 |
| 38 | styles.xml | Оформление элементов UI | 2 Кб | 55 |
| 39 | ProductViewModel.java | Логика отображения товаров | 4 Кб | 100 |
| 40 | CartViewModel.java | Логика корзины | 3 Кб | 90 |
| 41 | UserViewModel.java | Логика профиля | 3 Кб | 85 |
| 42 | menu\_main.xml | Меню навигации | 1 Кб | 25 |
| 43 | ic\_launcher.png | Иконка приложения (mipmap) | 15 Кб | — |
| 44 | logo.png | Логотип в ресурсах | 12 Кб | — |
| 45 | terms.txt | Текстовое соглашение (assets) | 5 Кб | 200 |
| 46 | AndroidManifest.xml | Манифест приложения | 3 Кб | 85 |

1.4. Код программы

**~1) AddProductActivity.java;~**

package com.example.bolotnyiysen;

import android.content.Intent;

import android.net.Uri;

import android.os.Bundle;

import android.provider.MediaStore;

import android.view.MenuItem;

import android.view.View;

import android.widget.ImageView;

import android.widget.Toast;

import androidx.annotation.Nullable;

import androidx.appcompat.app.AppCompatActivity;

import androidx.appcompat.widget.Toolbar;

import com.google.android.material.button.MaterialButton;

import com.google.android.material.textfield.TextInputEditText;

public class AddProductActivity extends AppCompatActivity {

private static final int PICK\_IMAGE\_REQUEST = 1;

private ImageView productImageView;

private TextInputEditText productNameInput;

private TextInputEditText productPriceInput;

private TextInputEditText productCodeInput;

private TextInputEditText manufacturerInput;

private TextInputEditText seriesInput;

private TextInputEditText typeInput;

private TextInputEditText conditionInput;

private TextInputEditText bodyShapeInput;

private TextInputEditText orientationInput;

private TextInputEditText stringsCountInput;

private TextInputEditText fretsCountInput;

private TextInputEditText scaleLengthInput;

private MaterialButton addProductButton;

private String photoPath = "";

private AppDatabase db;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_add\_product);

// Инициализация базы данных

db = AppDatabase.getInstance(this);

// Получение email пользователя из Intent

String userEmail = getIntent().getStringExtra("user\_email");

if (userEmail == null) {

Toast.makeText(this, "Ошибка: неверный email", Toast.LENGTH\_SHORT).show();

finish();

return;

}

// Проверяем, является ли пользователь администратором

checkAdminStatus(userEmail);

// Настройка Toolbar

Toolbar toolbar = findViewById(R.id.toolbar);

setSupportActionBar(toolbar);

if (getSupportActionBar() != null) {

getSupportActionBar().setDisplayHomeAsUpEnabled(true);

getSupportActionBar().setTitle("Добавление товара");

}

// Инициализация views

productImageView = findViewById(R.id.productImageView);

productNameInput = findViewById(R.id.productNameInput);

productPriceInput = findViewById(R.id.productPriceInput);

productCodeInput = findViewById(R.id.productCodeInput);

manufacturerInput = findViewById(R.id.manufacturerInput);

seriesInput = findViewById(R.id.seriesInput);

typeInput = findViewById(R.id.typeInput);

conditionInput = findViewById(R.id.conditionInput);

bodyShapeInput = findViewById(R.id.bodyShapeInput);

orientationInput = findViewById(R.id.orientationInput);

stringsCountInput = findViewById(R.id.stringsCountInput);

fretsCountInput = findViewById(R.id.fretsCountInput);

scaleLengthInput = findViewById(R.id.scaleLengthInput);

addProductButton = findViewById(R.id.addProductButton);

// Обработчик нажатия на изображение для выбора фото

productImageView.setOnClickListener(v -> {

Intent intent = new Intent(Intent.ACTION\_PICK, MediaStore.Images.Media.EXTERNAL\_CONTENT\_URI);

startActivityForResult(intent, PICK\_IMAGE\_REQUEST);

});

// Обработчик нажатия на кнопку добавления товара

addProductButton.setOnClickListener(v -> addProduct());

}

@Override

protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {

super.onActivityResult(requestCode, resultCode, data);

if (requestCode == PICK\_IMAGE\_REQUEST && resultCode == RESULT\_OK && data != null) {

Uri selectedImageUri = data.getData();

if (selectedImageUri != null) {

productImageView.setImageURI(selectedImageUri);

photoPath = selectedImageUri.toString();

}

}

}

private void addProduct() {

// Получение данных из полей ввода

String name = productNameInput.getText().toString().trim();

String priceStr = productPriceInput.getText().toString().trim();

String code = productCodeInput.getText().toString().trim();

String manufacturer = manufacturerInput.getText().toString().trim();

String series = seriesInput.getText().toString().trim();

String type = typeInput.getText().toString().trim();

String condition = conditionInput.getText().toString().trim();

String bodyShape = bodyShapeInput.getText().toString().trim();

String orientation = orientationInput.getText().toString().trim();

String stringsCountStr = stringsCountInput.getText().toString().trim();

String fretsCountStr = fretsCountInput.getText().toString().trim();

String scaleLengthStr = scaleLengthInput.getText().toString().trim();

// Проверка заполнения обязательных полей

if (name.isEmpty() || priceStr.isEmpty() || code.isEmpty() || manufacturer.isEmpty() ||

series.isEmpty() || type.isEmpty() || condition.isEmpty() || bodyShape.isEmpty() ||

orientation.isEmpty() || stringsCountStr.isEmpty() || fretsCountStr.isEmpty() ||

scaleLengthStr.isEmpty()) {

Toast.makeText(this, "Пожалуйста, заполните все поля", Toast.LENGTH\_SHORT).show();

return;

}

try {

// Преобразование строковых значений в числовые

double price = Double.parseDouble(priceStr);

int stringsCount = Integer.parseInt(stringsCountStr);

int fretsCount = Integer.parseInt(fretsCountStr);

double scaleLength = Double.parseDouble(scaleLengthStr);

// Создание объекта Product

Product product = new Product(name, price, R.drawable.ic\_add\_photo, code, manufacturer,

series, type, condition, bodyShape, orientation,

stringsCount, fretsCount, scaleLength, photoPath);

// Добавление товара в базу данных в фоновом потоке

new Thread(() -> {

db.productDao().insert(product);

runOnUiThread(() -> {

Toast.makeText(AddProductActivity.this, "Товар успешно добавлен", Toast.LENGTH\_SHORT).show();

finish();

});

}).start();

} catch (NumberFormatException e) {

Toast.makeText(this, "Пожалуйста, введите корректные числовые значения", Toast.LENGTH\_SHORT).show();

}

}

private void checkAdminStatus(String userEmail) {

new Thread(() -> {

User currentUser = db.userDao().getUserByEmail(userEmail);

boolean isAdmin = currentUser != null && currentUser.isAdmin();

runOnUiThread(() -> {

if (!isAdmin) {

// Если пользователь не администратор, закрываем активность

Toast.makeText(this, "Доступ запрещен. Только администратор может добавлять товары.", Toast.LENGTH\_LONG).show();

finish();

}

});

}).start();

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

if (item.getItemId() == android.R.id.home) {

onBackPressed();

return true;

}

return super.onOptionsItemSelected(item);

}

}

~2) Purchase.java;~

package com.example.bolotnyiysen;

import androidx.room.Entity;

import androidx.room.PrimaryKey;

@Entity(tableName = "purchases")

public class Purchase {

@PrimaryKey(autoGenerate = true)

private int id;

private String userEmail;

private String productName;

private String paymentType;

private String date;

private double price;

private String status;

public Purchase(String userEmail, String productName, String paymentType, String date, double price) {

this.userEmail = userEmail;

this.productName = productName;

this.paymentType = paymentType;

this.date = date;

this.price = price;

this.status = "Подтвержден"; // Изменено с "completed" на "Подтвержден"

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getUserEmail() {

return userEmail;

}

public String getProductName() {

return productName;

}

public String getPaymentType() {

return paymentType;

}

public String getDate() {

return date;

}

public double getPrice() {

return price;

}

public String getStatus() {

return status;

}

public void setStatus(String status) {

this.status = status;

}

}

~3) Product.java;~

package com.example.bolotnyiysen;

import androidx.room.Entity;

import androidx.room.PrimaryKey;

import java.io.Serializable;

@Entity(tableName = "products")

public class Product implements Serializable {

@PrimaryKey(autoGenerate = true)

private int id;

private String name;

private double price;

private int imageResourceId;

private String code;

private String manufacturer;

private String series;

private String type;

private String condition;

private String bodyShape;

private String orientation;

private int stringsCount;

private int fretsCount;

private double scaleLength;

private int quantity;

private String photoPath;

private boolean isHeader;

public Product(String name, double price, int imageResourceId, String code, String manufacturer,

String series, String type, String condition, String bodyShape, String orientation,

int stringsCount, int fretsCount, double scaleLength, String photoPath) {

this.name = name;

this.price = price;

this.imageResourceId = imageResourceId;

this.code = code;

this.manufacturer = manufacturer;

this.series = series;

this.type = type;

this.condition = condition;

this.bodyShape = bodyShape;

this.orientation = orientation;

this.stringsCount = stringsCount;

this.fretsCount = fretsCount;

this.scaleLength = scaleLength;

this.quantity = 1;

this.photoPath = photoPath;

this.isHeader = false;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public double getPrice() {

return price;

}

public void setPrice(double price) {

this.price = price;

}

public int getImageResourceId() {

return imageResourceId;

}

public void setImageResourceId(int imageResourceId) {

this.imageResourceId = imageResourceId;

}

public String getCode() {

return code;

}

public void setCode(String code) {

this.code = code;

}

public String getManufacturer() {

return manufacturer;

}

public void setManufacturer(String manufacturer) {

this.manufacturer = manufacturer;

}

public String getSeries() {

return series;

}

public void setSeries(String series) {

this.series = series;

}

public String getType() {

return type;

}

public void setType(String type) {

this.type = type;

}

public String getCondition() {

return condition;

}

public void setCondition(String condition) {

this.condition = condition;

}

public String getBodyShape() {

return bodyShape;

}

public void setBodyShape(String bodyShape) {

this.bodyShape = bodyShape;

}

public String getOrientation() {

return orientation;

}

public void setOrientation(String orientation) {

this.orientation = orientation;

}

public int getStringsCount() {

return stringsCount;

}

public void setStringsCount(int stringsCount) {

this.stringsCount = stringsCount;

}

public int getFretsCount() {

return fretsCount;

}

public void setFretsCount(int fretsCount) {

this.fretsCount = fretsCount;

}

public double getScaleLength() {

return scaleLength;

}

public void setScaleLength(double scaleLength) {

this.scaleLength = scaleLength;

}

public int getQuantity() {

return quantity;

}

public void incrementQuantity() {

quantity++;

}

public void decrementQuantity() {

if (quantity > 1) {

quantity--;

}

}

public void setQuantity(int quantity) {

if (quantity > 0) {

this.quantity = quantity;

}

}

public String getPhotoPath() {

return photoPath;

}

public void setPhotoPath(String photoPath) {

this.photoPath = photoPath;

}

public boolean isHeader() {

return isHeader;

}

public void setIsHeader(boolean header) {

isHeader = header;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

Product product = (Product) o;

return code.equals(product.code);

}

@Override

public int hashCode() {

return code.hashCode();

}

}

~4) UserPurchaseCount.java;~

package com.example.bolotnyiysen;

import androidx.room.ColumnInfo;

public class UserPurchaseCount {

@ColumnInfo(name = "userEmail")

private String userEmail;

@ColumnInfo(name = "count")

private int count;

public UserPurchaseCount(String userEmail, int count) {

this.userEmail = userEmail;

this.count = count;

}

public String getUserEmail() {

return userEmail;

}

public int getCount() {

return count;

}

}

~5) Receipt.java;~

package com.example.bolotnyiysen;

import java.io.Serializable;

import java.util.Date;

import java.util.List;

public class Receipt implements Serializable {

private String receiptNumber;

private Date date;

private String customerEmail;

private List<Product> products;

private double totalAmount;

private String storeName = "Болотный Усень";

public Receipt(String receiptNumber, String customerEmail, List<Product> products, double totalAmount) {

this.receiptNumber = receiptNumber;

this.date = new Date();

this.customerEmail = customerEmail;

this.products = products;

this.totalAmount = totalAmount;

}

public String getReceiptNumber() {

return receiptNumber;

}

public Date getDate() {

return date;

}

public String getCustomerEmail() {

return customerEmail;

}

public List<Product> getProducts() {

return products;

}

public double getTotalAmount() {

return totalAmount;

}

public String getStoreName() {

return storeName;

}

}

~6) Transaction.java;~

package com.example.bolotnyiysen;

import androidx.room.Entity;

import androidx.room.PrimaryKey;

@Entity(tableName = "transactions")

public class Transaction {

@PrimaryKey(autoGenerate = true)

private int id;

private String userEmail;

private String date;

private String items;

private double total;

public Transaction(String userEmail, String date, String items, double total) {

this.userEmail = userEmail;

this.date = date;

this.items = items;

this.total = total;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getUserEmail() {

return userEmail;

}

public String getDate() {

return date;

}

public String getItems() {

return items;

}

public double getTotal() {

return total;

}

}

~7) AddProductActivity.java;~

package com.example.bolotnyiysen;

import android.content.Intent;

import android.net.Uri;

import android.os.Bundle;

import android.provider.MediaStore;

import android.view.MenuItem;

import android.view.View;

import android.widget.ImageView;

import android.widget.Toast;

import androidx.annotation.Nullable;

import androidx.appcompat.app.AppCompatActivity;

import androidx.appcompat.widget.Toolbar;

import com.google.android.material.button.MaterialButton;

import com.google.android.material.textfield.TextInputEditText;

public class AddProductActivity extends AppCompatActivity {

private static final int PICK\_IMAGE\_REQUEST = 1;

private ImageView productImageView;

private TextInputEditText productNameInput;

private TextInputEditText productPriceInput;

private TextInputEditText productCodeInput;

private TextInputEditText manufacturerInput;

private TextInputEditText seriesInput;

private TextInputEditText typeInput;

private TextInputEditText conditionInput;

private TextInputEditText bodyShapeInput;

private TextInputEditText orientationInput;

private TextInputEditText stringsCountInput;

private TextInputEditText fretsCountInput;

private TextInputEditText scaleLengthInput;

private MaterialButton addProductButton;

private String photoPath = "";

private AppDatabase db;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_add\_product);

// Инициализация базы данных

db = AppDatabase.getInstance(this);

// Получение email пользователя из Intent

String userEmail = getIntent().getStringExtra("user\_email");

if (userEmail == null) {

Toast.makeText(this, "Ошибка: неверный email", Toast.LENGTH\_SHORT).show();

finish();

return;

}

// Проверяем, является ли пользователь администратором

checkAdminStatus(userEmail);

// Настройка Toolbar

Toolbar toolbar = findViewById(R.id.toolbar);

setSupportActionBar(toolbar);

if (getSupportActionBar() != null) {

getSupportActionBar().setDisplayHomeAsUpEnabled(true);

getSupportActionBar().setTitle("Добавление товара");

}

// Инициализация views

productImageView = findViewById(R.id.productImageView);

productNameInput = findViewById(R.id.productNameInput);

productPriceInput = findViewById(R.id.productPriceInput);

productCodeInput = findViewById(R.id.productCodeInput);

manufacturerInput = findViewById(R.id.manufacturerInput);

seriesInput = findViewById(R.id.seriesInput);

typeInput = findViewById(R.id.typeInput);

conditionInput = findViewById(R.id.conditionInput);

bodyShapeInput = findViewById(R.id.bodyShapeInput);

orientationInput = findViewById(R.id.orientationInput);

stringsCountInput = findViewById(R.id.stringsCountInput);

fretsCountInput = findViewById(R.id.fretsCountInput);

scaleLengthInput = findViewById(R.id.scaleLengthInput);

addProductButton = findViewById(R.id.addProductButton);

// Обработчик нажатия на изображение для выбора фото

productImageView.setOnClickListener(v -> {

Intent intent = new Intent(Intent.ACTION\_PICK, MediaStore.Images.Media.EXTERNAL\_CONTENT\_URI);

startActivityForResult(intent, PICK\_IMAGE\_REQUEST);

});

// Обработчик нажатия на кнопку добавления товара

addProductButton.setOnClickListener(v -> addProduct());

}

@Override

protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {

super.onActivityResult(requestCode, resultCode, data);

if (requestCode == PICK\_IMAGE\_REQUEST && resultCode == RESULT\_OK && data != null) {

Uri selectedImageUri = data.getData();

if (selectedImageUri != null) {

productImageView.setImageURI(selectedImageUri);

photoPath = selectedImageUri.toString();

}

}

}

private void addProduct() {

// Получение данных из полей ввода

String name = productNameInput.getText().toString().trim();

String priceStr = productPriceInput.getText().toString().trim();

String code = productCodeInput.getText().toString().trim();

String manufacturer = manufacturerInput.getText().toString().trim();

String series = seriesInput.getText().toString().trim();

String type = typeInput.getText().toString().trim();

String condition = conditionInput.getText().toString().trim();

String bodyShape = bodyShapeInput.getText().toString().trim();

String orientation = orientationInput.getText().toString().trim();

String stringsCountStr = stringsCountInput.getText().toString().trim();

String fretsCountStr = fretsCountInput.getText().toString().trim();

String scaleLengthStr = scaleLengthInput.getText().toString().trim();

// Проверка заполнения обязательных полей

if (name.isEmpty() || priceStr.isEmpty() || code.isEmpty() || manufacturer.isEmpty() ||

series.isEmpty() || type.isEmpty() || condition.isEmpty() || bodyShape.isEmpty() ||

orientation.isEmpty() || stringsCountStr.isEmpty() || fretsCountStr.isEmpty() ||

scaleLengthStr.isEmpty()) {

Toast.makeText(this, "Пожалуйста, заполните все поля", Toast.LENGTH\_SHORT).show();

return;

}

try {

// Преобразование строковых значений в числовые

double price = Double.parseDouble(priceStr);

int stringsCount = Integer.parseInt(stringsCountStr);

int fretsCount = Integer.parseInt(fretsCountStr);

double scaleLength = Double.parseDouble(scaleLengthStr);

// Создание объекта Product

Product product = new Product(name, price, R.drawable.ic\_add\_photo, code, manufacturer,

series, type, condition, bodyShape, orientation,

stringsCount, fretsCount, scaleLength, photoPath);

// Добавление товара в базу данных в фоновом потоке

new Thread(() -> {

db.productDao().insert(product);

runOnUiThread(() -> {

Toast.makeText(AddProductActivity.this, "Товар успешно добавлен", Toast.LENGTH\_SHORT).show();

finish();

});

}).start();

} catch (NumberFormatException e) {

Toast.makeText(this, "Пожалуйста, введите корректные числовые значения", Toast.LENGTH\_SHORT).show();

}

}

private void checkAdminStatus(String userEmail) {

new Thread(() -> {

User currentUser = db.userDao().getUserByEmail(userEmail);

boolean isAdmin = currentUser != null && currentUser.isAdmin();

runOnUiThread(() -> {

if (!isAdmin) {

// Если пользователь не администратор, закрываем активность

Toast.makeText(this, "Доступ запрещен. Только администратор может добавлять товары.", Toast.LENGTH\_LONG).show();

finish();

}

});

}).start();

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

if (item.getItemId() == android.R.id.home) {

onBackPressed();

return true;

}

return super.onOptionsItemSelected(item);

}

}

~8) CartActivity.java;~

package com.example.bolotnyiysen;

import android.Manifest;

import android.content.Intent;

import android.content.pm.PackageManager;

import android.os.Build;

import android.os.Bundle;

import android.text.Editable;

import android.text.InputFilter;

import android.text.Spanned;

import android.text.TextWatcher;

import android.view.MenuItem;

import android.widget.Button;

import android.widget.LinearLayout;

import android.widget.RadioButton;

import android.widget.RadioGroup;

import android.widget.TextView;

import android.widget.Toast;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity;

import androidx.core.app.ActivityCompat;

import androidx.core.content.ContextCompat;

import androidx.recyclerview.widget.LinearLayoutManager;

import androidx.recyclerview.widget.RecyclerView;

import com.google.android.material.bottomnavigation.BottomNavigationView;

import com.google.android.material.textfield.TextInputEditText;

import java.io.File;

import java.util.List;

import java.util.regex.Pattern;

public class CartActivity extends AppCompatActivity {

private static final int STORAGE\_PERMISSION\_CODE = 1001;

private RecyclerView recyclerView;

private CartAdapter adapter;

private TextView totalPriceTextView;

private BottomNavigationView bottomNavigationView;

private Button checkoutButton;

private RadioGroup paymentMethodGroup;

private String selectedPaymentMethod;

private LinearLayout cardDetailsLayout;

private TextInputEditText cardNumberInput;

private TextInputEditText cardExpiryInput;

private TextInputEditText cardCvvInput;

private TextInputEditText cardHolderInput;

private String userEmail;

private CartManager cartManager;

private static final Pattern CARD\_NUMBER\_PATTERN = Pattern.compile("^[0-9]{16}$");

private static final Pattern CARD\_EXPIRY\_PATTERN = Pattern.compile("^(0[1-9]|1[0-2])/([0-9]{2})$");

private static final Pattern CARD\_CVV\_PATTERN = Pattern.compile("^[0-9]{3}$");

private static final Pattern CARD\_HOLDER\_PATTERN = Pattern.compile("^[A-Z\\s]+$");

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_cart);

// Получаем email из Intent

userEmail = getIntent().getStringExtra("user\_email");

if (userEmail == null) {

Toast.makeText(this, "Ошибка: неверный email", Toast.LENGTH\_SHORT).show();

finish();

return;

}

// Инициализация CartManager

cartManager = CartManager.getInstance(this);

cartManager.setCurrentUser(userEmail);

recyclerView = findViewById(R.id.cartRecyclerView);

totalPriceTextView = findViewById(R.id.totalPriceTextView);

bottomNavigationView = findViewById(R.id.bottomNavigationView);

checkoutButton = findViewById(R.id.checkoutButton);

paymentMethodGroup = findViewById(R.id.paymentMethodGroup);

cardDetailsLayout = findViewById(R.id.cardDetailsLayout);

cardNumberInput = findViewById(R.id.cardNumberInput);

cardExpiryInput = findViewById(R.id.cardExpiryInput);

cardCvvInput = findViewById(R.id.cardCvvInput);

cardHolderInput = findViewById(R.id.cardHolderInput);

recyclerView.setLayoutManager(new LinearLayoutManager(this));

List<Product> cartProducts = cartManager.getCartProducts();

adapter = new CartAdapter(cartProducts, this::updateTotal);

recyclerView.setAdapter(adapter);

// Calculate initial total

updateTotal(cartManager.getTotalPrice());

setupBottomNavigation();

setupPaymentMethods();

setupCardInputListeners();

setupOrderButton();

// Устанавливаем выбранный пункт меню

bottomNavigationView.setSelectedItemId(R.id.navigation\_cart);

// Проверяем разрешения для сохранения файлов

if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.M) {

if (ContextCompat.checkSelfPermission(this, Manifest.permission.WRITE\_EXTERNAL\_STORAGE)

!= PackageManager.PERMISSION\_GRANTED) {

ActivityCompat.requestPermissions(this,

new String[]{Manifest.permission.WRITE\_EXTERNAL\_STORAGE},

STORAGE\_PERMISSION\_CODE);

}

}

}

@Override

protected void onResume() {

super.onResume();

// Обновляем данные при возвращении на экран

if (userEmail != null) {

cartManager.setCurrentUser(userEmail);

adapter.updateCartProducts(cartManager.getCartProducts());

updateTotal(cartManager.getTotalPrice());

}

}

@Override

public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,

@NonNull int[] grantResults) {

super.onRequestPermissionsResult(requestCode, permissions, grantResults);

if (requestCode == STORAGE\_PERMISSION\_CODE) {

if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION\_GRANTED) {

Toast.makeText(this, "Разрешение на сохранение файлов получено", Toast.LENGTH\_SHORT).show();

} else {

Toast.makeText(this, "Для сохранения чеков необходимо разрешение на запись файлов",

Toast.LENGTH\_LONG).show();

}

}

}

private void setupCardInputListeners() {

// Ограничение ввода только цифр для номера карты

cardNumberInput.setFilters(new InputFilter[] {

new InputFilter() {

@Override

public CharSequence filter(CharSequence source, int start, int end, Spanned dest, int dstart, int dend) {

for (int i = start; i < end; i++) {

if (!Character.isDigit(source.charAt(i))) {

return "";

}

}

return null;

}

},

new InputFilter.LengthFilter(16)

});

// Форматирование срока действия (MM/YY)

cardExpiryInput.addTextChangedListener(new TextWatcher() {

private boolean isFormatting = false;

@Override

public void beforeTextChanged(CharSequence s, int start, int count, int after) {}

@Override

public void onTextChanged(CharSequence s, int start, int before, int count) {}

@Override

public void afterTextChanged(Editable s) {

if (isFormatting) {

return;

}

isFormatting = true;

String input = s.toString().replaceAll("/", "");

if (input.length() >= 2) {

String formatted = input.substring(0, 2) + "/" + input.substring(2);

if (!formatted.equals(s.toString())) {

cardExpiryInput.setText(formatted);

}

}

isFormatting = false;

}

});

// Ограничение ввода только заглавных букв для имени владельца

cardHolderInput.addTextChangedListener(new TextWatcher() {

@Override

public void beforeTextChanged(CharSequence s, int start, int count, int after) {}

@Override

public void onTextChanged(CharSequence s, int start, int before, int count) {}

@Override

public void afterTextChanged(Editable s) {

String input = s.toString().toUpperCase();

if (!input.equals(s.toString())) {

cardHolderInput.setText(input);

}

}

});

}

private boolean validateCardData() {

String cardNumber = cardNumberInput.getText().toString().replaceAll("\\s", "");

String cardExpiry = cardExpiryInput.getText().toString();

String cardCvv = cardCvvInput.getText().toString();

String cardHolder = cardHolderInput.getText().toString().trim();

// Проверка номера карты

if (!CARD\_NUMBER\_PATTERN.matcher(cardNumber).matches()) {

cardNumberInput.setError("Введите корректный номер карты (16 цифр)");

return false;

}

// Проверка срока действия

if (!CARD\_EXPIRY\_PATTERN.matcher(cardExpiry).matches()) {

cardExpiryInput.setError("Введите срок действия в формате MM/YY");

return false;

}

// Проверка месяца

int month = Integer.parseInt(cardExpiry.substring(0, 2));

if (month < 1 || month > 12) {

cardExpiryInput.setError("Некорректный месяц");

return false;

}

// Проверка CVV

if (!CARD\_CVV\_PATTERN.matcher(cardCvv).matches()) {

cardCvvInput.setError("Введите корректный CVV (3 цифры)");

return false;

}

// Проверка имени владельца

if (!CARD\_HOLDER\_PATTERN.matcher(cardHolder).matches()) {

cardHolderInput.setError("Введите имя латинскими буквами");

return false;

}

return true;

}

private void updateTotal(double total) {

totalPriceTextView.setText(String.format("Итого: %.2f ₽", total));

}

private void setupBottomNavigation() {

bottomNavigationView.setOnNavigationItemSelectedListener(new BottomNavigationView.OnNavigationItemSelectedListener() {

@Override

public boolean onNavigationItemSelected(@NonNull MenuItem item) {

int itemId = item.getItemId();

if (itemId == R.id.navigation\_home) {

Intent intent = new Intent(CartActivity.this, MainActivity.class);

intent.putExtra("user\_email", userEmail);

startActivity(intent);

finish();

return true;

} else if (itemId == R.id.navigation\_catalog) {

Intent intent = new Intent(CartActivity.this, CatalogActivity.class);

intent.putExtra("user\_email", userEmail);

startActivity(intent);

finish();

return true;

} else if (itemId == R.id.navigation\_favorites) {

Intent intent = new Intent(CartActivity.this, FavoritesActivity.class);

intent.putExtra("user\_email", userEmail);

startActivity(intent);

finish();

return true;

} else if (itemId == R.id.navigation\_cart) {

// Уже в корзине

return true;

} else if (itemId == R.id.navigation\_profile) {

Intent intent = new Intent(CartActivity.this, ProfileActivity.class);

intent.putExtra("user\_email", userEmail);

startActivity(intent);

finish();

return true;

}

return false;

}

});

}

private void setupPaymentMethods() {

paymentMethodGroup.setOnCheckedChangeListener((group, checkedId) -> {

RadioButton selectedRadioButton = findViewById(checkedId);

if (selectedRadioButton != null) {

selectedPaymentMethod = selectedRadioButton.getText().toString();

// Показываем или скрываем поля для карты

if (checkedId == R.id.cardPayment) {

cardDetailsLayout.setVisibility(LinearLayout.VISIBLE);

} else {

cardDetailsLayout.setVisibility(LinearLayout.GONE);

}

}

});

}

private void setupOrderButton() {

checkoutButton.setOnClickListener(v -> {

// Проверяем, есть ли товары в корзине

if (cartManager.isEmpty()) {

Toast.makeText(this, "Корзина пуста. Добавьте товары для оформления заказа", Toast.LENGTH\_SHORT).show();

return;

}

// Проверяем, выбран ли способ оплаты

if (selectedPaymentMethod == null) {

Toast.makeText(this, "Выберите способ оплаты", Toast.LENGTH\_SHORT).show();

return;

}

// Проверяем данные карты, если выбран этот способ оплаты

if (selectedPaymentMethod.equals("Банковской картой")) {

if (!validateCardData()) {

return;

}

}

// Создаем чек

ReceiptManager receiptManager = ReceiptManager.getInstance(this);

String receiptNumber = receiptManager.generateReceiptNumber();

List<Product> cartProducts = cartManager.getCartProducts();

double totalAmount = cartManager.getTotalPrice();

Receipt receipt = new Receipt(receiptNumber, userEmail, cartProducts, totalAmount);

String receiptPath = receiptManager.generateReceipt(receipt);

if (receiptPath != null) {

// Сохраняем заказ в базу данных

AppDatabase appDatabase = AppDatabase.getInstance(this);

new Thread(() -> {

try {

// Получаем текущую дату

String currentDate = new java.text.SimpleDateFormat("dd.MM.yyyy HH:mm", java.util.Locale.getDefault())

.format(new java.util.Date());

// Сохраняем каждый товар как отдельную покупку

for (Product product : cartProducts) {

Purchase purchase = new Purchase(userEmail, product.getName(), selectedPaymentMethod, currentDate, product.getPrice() \* product.getQuantity());

appDatabase.purchaseDao().insert(purchase);

}

// Очищаем корзину

cartManager.clearCart();

// Показываем сообщение об успешном оформлении заказа

runOnUiThread(() -> {

// Создаем диалог с информацией о сохраненном чеке

android.app.AlertDialog.Builder builder = new android.app.AlertDialog.Builder(this);

builder.setTitle("Заказ успешно оформлен!");

builder.setMessage("Чек сохранен в папке Documents/BolotnyiYsen\_Receipts\n\nПуть: " + receiptPath);

builder.setPositiveButton("Открыть чек", (dialog, which) -> {

// Открываем PDF файл

Intent intent = new Intent(Intent.ACTION\_VIEW);

File file = new File(receiptPath);

android.net.Uri uri;

if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.N) {

uri = androidx.core.content.FileProvider.getUriForFile(this,

getApplicationContext().getPackageName() + ".provider", file);

intent.addFlags(Intent.FLAG\_GRANT\_READ\_URI\_PERMISSION);

} else {

uri = android.net.Uri.fromFile(file);

}

intent.setDataAndType(uri, "application/pdf");

intent.addFlags(Intent.FLAG\_ACTIVITY\_NO\_HISTORY);

startActivity(intent);

});

builder.setNegativeButton("Закрыть", (dialog, which) -> {

// Возвращаемся на главный экран

Intent intent = new Intent(this, MainActivity.class);

intent.putExtra("user\_email", userEmail);

startActivity(intent);

finish();

});

builder.setCancelable(false);

builder.show();

});

} catch (Exception e) {

runOnUiThread(() -> {

Toast.makeText(this, "Ошибка при сохранении заказа", Toast.LENGTH\_SHORT).show();

});

}

}).start();

} else {

Toast.makeText(this, "Ошибка при создании чека", Toast.LENGTH\_SHORT).show();

}

});

}

}

~9) CartAdapter.java;~

package com.example.bolotnyiysen;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.ImageButton;

import android.widget.ImageView;

import android.widget.TextView;

import androidx.annotation.NonNull;

import androidx.recyclerview.widget.RecyclerView;

import com.bumptech.glide.Glide;

import java.util.List;

public class CartAdapter extends RecyclerView.Adapter<CartAdapter.CartViewHolder> {

private List<Product> cartProducts;

private OnTotalUpdateListener totalUpdateListener;

public interface OnTotalUpdateListener {

void onTotalUpdate(double newTotal);

}

public CartAdapter(List<Product> cartProducts, OnTotalUpdateListener listener) {

this.cartProducts = cartProducts;

this.totalUpdateListener = listener;

}

public void updateCartProducts(List<Product> newProducts) {

this.cartProducts = newProducts;

notifyDataSetChanged();

}

@NonNull

@Override

public CartViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {

View view = LayoutInflater.from(parent.getContext())

.inflate(R.layout.item\_cart, parent, false);

return new CartViewHolder(view);

}

@Override

public void onBindViewHolder(@NonNull CartViewHolder holder, int position) {

Product product = cartProducts.get(position);

holder.bind(product);

}

@Override

public int getItemCount() {

return cartProducts.size();

}

private void updateTotal() {

double total = 0;

for (Product product : cartProducts) {

total += product.getPrice() \* product.getQuantity();

}

if (totalUpdateListener != null) {

totalUpdateListener.onTotalUpdate(total);

}

}

class CartViewHolder extends RecyclerView.ViewHolder {

private ImageView productImage;

private TextView productName;

private TextView productPrice;

private TextView quantityText;

private ImageButton addButton;

private ImageButton removeButton;

private ImageButton deleteButton;

private Product currentProduct;

CartViewHolder(@NonNull View itemView) {

super(itemView);

productImage = itemView.findViewById(R.id.productImage);

productName = itemView.findViewById(R.id.productName);

productPrice = itemView.findViewById(R.id.productPrice);

quantityText = itemView.findViewById(R.id.quantityText);

addButton = itemView.findViewById(R.id.addButton);

removeButton = itemView.findViewById(R.id.removeButton);

deleteButton = itemView.findViewById(R.id.deleteButton);

setupButtons();

}

private void setupButtons() {

addButton.setOnClickListener(v -> {

if (currentProduct != null) {

CartManager.getInstance(itemView.getContext()).addToCart(currentProduct);

updateQuantity();

updateTotal();

}

});

removeButton.setOnClickListener(v -> {

if (currentProduct != null) {

CartManager.getInstance(itemView.getContext()).removeFromCart(currentProduct);

updateQuantity();

updateTotal();

}

});

deleteButton.setOnClickListener(v -> {

if (currentProduct != null) {

CartManager.getInstance(itemView.getContext()).removeAllFromCart(currentProduct);

int position = getAdapterPosition();

if (position != RecyclerView.NO\_POSITION) {

cartProducts.remove(position);

notifyItemRemoved(position);

updateTotal();

}

}

});

}

void bind(Product product) {

currentProduct = product;

productName.setText(product.getName());

productPrice.setText(String.format("%.2f ₽", product.getPrice() \* product.getQuantity()));

quantityText.setText(String.valueOf(product.getQuantity()));

Glide.with(itemView.getContext())

.load(product.getImageResourceId())

.into(productImage);

}

private void updateQuantity() {

int quantity = CartManager.getInstance(itemView.getContext()).getProductCount(currentProduct);

if (quantity > 0) {

quantityText.setText(String.valueOf(quantity));

productPrice.setText(String.format("%.2f ₽", currentProduct.getPrice() \* quantity));

} else {

int position = getAdapterPosition();

if (position != RecyclerView.NO\_POSITION) {

cartProducts.remove(position);

notifyItemRemoved(position);

}

}

}

}

}

~10) CartManager.java;~

package com.example.bolotnyiysen;

import android.content.Context;

import android.content.SharedPreferences;

import com.google.gson.Gson;

import com.google.gson.reflect.TypeToken;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

public class CartManager {

private static CartManager instance;

private final Map<String, Map<Product, Integer>> userCarts;

private final SharedPreferences sharedPreferences;

private final Gson gson;

private String currentUserEmail;

private CartManager(Context context) {

userCarts = new HashMap<>();

sharedPreferences = context.getSharedPreferences("cart\_preferences", Context.MODE\_PRIVATE);

gson = new Gson();

}

public static CartManager getInstance(Context context) {

if (instance == null) {

instance = new CartManager(context);

}

return instance;

}

public void setCurrentUser(String email) {

this.currentUserEmail = email;

if (!userCarts.containsKey(email)) {

loadUserCart(email);

}

}

private void loadUserCart(String email) {

String cartJson = sharedPreferences.getString("cart\_" + email, null);

if (cartJson != null && !cartJson.isEmpty()) {

try {

TypeToken<Map<Product, Integer>> typeToken = new TypeToken<Map<Product, Integer>>() {};

Map<Product, Integer> cart = gson.fromJson(cartJson, typeToken.getType());

if (cart != null) {

userCarts.put(email, cart);

} else {

userCarts.put(email, new HashMap<>());

}

} catch (Exception e) {

// If there's an error parsing the JSON, create a new empty cart

userCarts.put(email, new HashMap<>());

// Clear the invalid data

sharedPreferences.edit().remove("cart\_" + email).apply();

}

} else {

userCarts.put(email, new HashMap<>());

}

}

private void saveUserCart(String email) {

Map<Product, Integer> cart = userCarts.get(email);

if (cart != null) {

String cartJson = gson.toJson(cart);

sharedPreferences.edit().putString("cart\_" + email, cartJson).apply();

}

}

public void addToCart(Product product) {

if (currentUserEmail == null) return;

Map<Product, Integer> cart = userCarts.get(currentUserEmail);

if (cart == null) {

cart = new HashMap<>();

userCarts.put(currentUserEmail, cart);

}

if (cart.containsKey(product)) {

cart.put(product, cart.get(product) + 1);

} else {

cart.put(product, 1);

}

saveUserCart(currentUserEmail);

}

public void removeFromCart(Product product) {

if (currentUserEmail == null) return;

Map<Product, Integer> cart = userCarts.get(currentUserEmail);

if (cart != null && cart.containsKey(product)) {

int count = cart.get(product);

if (count > 1) {

cart.put(product, count - 1);

} else {

cart.remove(product);

}

saveUserCart(currentUserEmail);

}

}

public void removeAllFromCart(Product product) {

if (currentUserEmail == null) return;

Map<Product, Integer> cart = userCarts.get(currentUserEmail);

if (cart != null) {

cart.remove(product);

saveUserCart(currentUserEmail);

}

}

public List<Product> getCartProducts() {

if (currentUserEmail == null) return new ArrayList<>();

List<Product> products = new ArrayList<>();

Map<Product, Integer> cart = userCarts.get(currentUserEmail);

if (cart != null) {

for (Map.Entry<Product, Integer> entry : cart.entrySet()) {

Product product = entry.getKey();

int quantity = entry.getValue();

product.setQuantity(quantity);

products.add(product);

}

}

return products;

}

public int getProductCount(Product product) {

if (currentUserEmail == null) return 0;

Map<Product, Integer> cart = userCarts.get(currentUserEmail);

return cart != null ? cart.getOrDefault(product, 0) : 0;

}

public void clearCart() {

if (currentUserEmail == null) return;

Map<Product, Integer> cart = userCarts.get(currentUserEmail);

if (cart != null) {

cart.clear();

saveUserCart(currentUserEmail);

}

}

public double getTotalPrice() {

if (currentUserEmail == null) return 0;

double total = 0;

Map<Product, Integer> cart = userCarts.get(currentUserEmail);

if (cart != null) {

for (Map.Entry<Product, Integer> entry : cart.entrySet()) {

total += entry.getKey().getPrice() \* entry.getValue();

}

}

return total;

}

public boolean isEmpty() {

if (currentUserEmail == null) return true;

Map<Product, Integer> cart = userCarts.get(currentUserEmail);

return cart == null || cart.isEmpty();

}

}

~11) CatalogActivity.java;~

package com.example.bolotnyiysen;

import android.content.Intent;

import android.content.SharedPreferences;

import android.os.Bundle;

import android.view.MenuItem;

import android.view.View;

import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import androidx.recyclerview.widget.LinearLayoutManager;

import androidx.recyclerview.widget.RecyclerView;

import com.google.android.material.bottomnavigation.BottomNavigationView;

import com.google.android.material.floatingactionbutton.FloatingActionButton;

import com.google.android.material.tabs.TabLayout;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

public class CatalogActivity extends AppCompatActivity {

private BottomNavigationView bottomNavigationView;

private TabLayout tabLayout;

private String currentCategoryType = "type"; // По умолчанию сортировка по типу

private RecyclerView recyclerView;

private ProductAdapter adapter;

private List<Product> productList;

private String userEmail;

private CartManager cartManager;

private FloatingActionButton addProductFab;

private AppDatabase db;

private boolean isAdmin = false;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_catalog);

// Инициализация базы данных

db = AppDatabase.getInstance(this);

// Получение email пользователя из Intent

userEmail = getIntent().getStringExtra("user\_email");

if (userEmail == null) {

Toast.makeText(this, "Ошибка: неверный email", Toast.LENGTH\_SHORT).show();

finish();

return;

}

// Инициализация CartManager

cartManager = CartManager.getInstance(this);

cartManager.setCurrentUser(userEmail);

// Инициализация views

bottomNavigationView = findViewById(R.id.bottomNavigationView);

tabLayout = findViewById(R.id.tabLayout);

recyclerView = findViewById(R.id.recyclerView);

addProductFab = findViewById(R.id.addProductFab);

// Настройка RecyclerView

recyclerView.setLayoutManager(new LinearLayoutManager(this));

productList = new ArrayList<>();

adapter = new ProductAdapter(productList, this);

recyclerView.setAdapter(adapter);

// Добавляем обработчик клика на товар

adapter.setOnProductClickListener(product -> {

if (!product.isHeader()) {

Intent intent = new Intent(this, ProductDetailActivity.class);

intent.putExtra("product", product);

ArrayList<Integer> imageResources = new ArrayList<>();

ArrayList<String> imagePaths = new ArrayList<>();

if (product.getPhotoPath() != null && !product.getPhotoPath().isEmpty()) {

// Если есть путь к фото, добавляем его в список путей

imagePaths.add(product.getPhotoPath());

imageResources.add(0); // Placeholder для сохранения индекса

} else {

// Если нет пути к фото, используем стандартное изображение из ресурсов

imageResources.add(product.getImageResourceId());

imagePaths.add(null);

}

intent.putIntegerArrayListExtra("imageResources", imageResources);

intent.putStringArrayListExtra("imagePaths", imagePaths);

startActivity(intent);

}

});

// Добавляем обработчик удаления товара

adapter.setOnProductDeleteListener(product -> {

if (!isAdmin) {

Toast.makeText(this, "У вас нет прав для удаления товаров", Toast.LENGTH\_SHORT).show();

return;

}

// Показываем диалог подтверждения

new androidx.appcompat.app.AlertDialog.Builder(this)

.setTitle("Удаление товара")

.setMessage("Вы уверены, что хотите удалить этот товар?")

.setPositiveButton("Да", (dialog, which) -> {

new Thread(() -> {

// Удаляем товар из базы данных

db.productDao().delete(product);

runOnUiThread(() -> {

// Удаляем товар из списка

int position = productList.indexOf(product);

if (position != -1) {

productList.remove(position);

adapter.notifyItemRemoved(position);

Toast.makeText(this, "Товар успешно удален", Toast.LENGTH\_SHORT).show();

// Обновляем список товаров

loadProducts();

}

});

}).start();

})

.setNegativeButton("Нет", null)

.show();

});

// Настройка навигации

setupBottomNavigation();

setupTabs();

// Проверяем, является ли пользователь администратором

checkAdminStatus();

// Загрузка товаров из базы данных и статических товаров

loadProducts();

// Устанавливаем выбранный пункт меню

bottomNavigationView.setSelectedItemId(R.id.navigation\_catalog);

}

private void setupTabs() {

tabLayout.addTab(tabLayout.newTab().setText("По типу"));

tabLayout.addTab(tabLayout.newTab().setText("По бренду"));

tabLayout.addTab(tabLayout.newTab().setText("По состоянию"));

tabLayout.addOnTabSelectedListener(new TabLayout.OnTabSelectedListener() {

@Override

public void onTabSelected(TabLayout.Tab tab) {

switch (tab.getPosition()) {

case 0:

currentCategoryType = "type";

break;

case 1:

currentCategoryType = "brand";

break;

case 2:

currentCategoryType = "condition";

break;

}

filterProductsByCategory();

}

@Override

public void onTabUnselected(TabLayout.Tab tab) {}

@Override

public void onTabReselected(TabLayout.Tab tab) {}

});

}

private void filterProductsByCategory() {

// Если нет выбранной категории, показываем все товары

if (currentCategoryType == null) {

adapter.notifyDataSetChanged();

return;

}

// Создаем Map для группировки товаров

Map<String, List<Product>> groupedProducts = new HashMap<>();

// Группируем товары по выбранной категории

for (Product product : productList) {

String category;

switch (currentCategoryType) {

case "brand":

category = product.getManufacturer();

break;

case "condition":

category = product.getCondition();

break;

default:

category = product.getType();

break;

}

if (!groupedProducts.containsKey(category)) {

groupedProducts.put(category, new ArrayList<>());

}

groupedProducts.get(category).add(product);

}

// Создаем новый список для отображения

List<Product> filteredProducts = new ArrayList<>();

// Добавляем заголовки категорий и товары

for (Map.Entry<String, List<Product>> entry : groupedProducts.entrySet()) {

// Добавляем заголовок категории

Product headerProduct = new Product(

"=== " + entry.getKey() + " ===",

0,

0,

"",

"",

"",

"",

"",

"",

"",

0,

0,

0,

""

);

headerProduct.setIsHeader(true);

filteredProducts.add(headerProduct);

// Добавляем товары категории

filteredProducts.addAll(entry.getValue());

}

// Обновляем адаптер с отфильтрованными товарами

adapter = new ProductAdapter(filteredProducts, this);

recyclerView.setAdapter(adapter);

// Устанавливаем статус администратора для нового адаптера

adapter.setAdmin(isAdmin);

// Добавляем обработчик клика на товар

adapter.setOnProductClickListener(product -> {

if (!product.isHeader()) {

Intent intent = new Intent(this, ProductDetailActivity.class);

intent.putExtra("product", product);

ArrayList<Integer> imageResources = new ArrayList<>();

ArrayList<String> imagePaths = new ArrayList<>();

if (product.getPhotoPath() != null && !product.getPhotoPath().isEmpty()) {

// Если есть путь к фото, добавляем его в список путей

imagePaths.add(product.getPhotoPath());

imageResources.add(0); // Placeholder для сохранения индекса

} else {

// Если нет пути к фото, используем стандартное изображение из ресурсов

imageResources.add(product.getImageResourceId());

imagePaths.add(null);

}

intent.putIntegerArrayListExtra("imageResources", imageResources);

intent.putStringArrayListExtra("imagePaths", imagePaths);

startActivity(intent);

}

});

// Добавляем обработчик удаления товара для нового адаптера

adapter.setOnProductDeleteListener(product -> {

if (!isAdmin) {

Toast.makeText(this, "У вас нет прав для удаления товаров", Toast.LENGTH\_SHORT).show();

return;

}

// Показываем диалог подтверждения

new androidx.appcompat.app.AlertDialog.Builder(this)

.setTitle("Удаление товара")

.setMessage("Вы уверены, что хотите удалить этот товар?")

.setPositiveButton("Да", (dialog, which) -> {

new Thread(() -> {

// Удаляем товар из базы данных

db.productDao().delete(product);

runOnUiThread(() -> {

// Удаляем товар из списка

int position = productList.indexOf(product);

if (position != -1) {

productList.remove(position);

adapter.notifyItemRemoved(position);

Toast.makeText(this, "Товар успешно удален", Toast.LENGTH\_SHORT).show();

// Обновляем список товаров

loadProducts();

}

});

}).start();

})

.setNegativeButton("Нет", null)

.show();

});

}

private List<Product> getStaticProducts() {

List<Product> products = new ArrayList<>();

products.add(new Product("Электрогитара Magneto US-1300 RC/CAR Sonnet Classic Stratocaster HSS Candy Red", 28000, R.drawable.img1, "GTR001", "Magneto", "Sonnet Classic", "Электрогитара", "Новая", "Strat", "Правая", 6, 22, 25.5, ""));

products.add(new Product("Электрогитара Magneto US-1300 RC/MKPW Sonnet Classic Stratocaster HSS Metallic Pearl White", 28000, R.drawable.img2, "GTR002", "Magneto", "Sonnet Classic", "Электрогитара", "Б/У", "Strat", "Правая", 6, 22, 24.75, ""));

products.add(new Product("Электрогитара Gibson Les Paul Traditional Pro LPCGEBCHI H-H Black 2012 USA W/Case", 90000, R.drawable.img3, "GTR003", "Gibson", "Les Paul", "Электрогитара", "Новая", "Les Paul", "Правая", 6, 24, 25.5, ""));

products.add(new Product("Электрогитара Fender Telecaster Custom 1972 Vintage Reissue HS Black w/gigbag Japan 1995", 60000, R.drawable.img4, "GTR004", "Fender", "Telecaster", "Электрогитара", "Новая", "Telecaster", "Правая", 6, 24, 25.0, ""));

products.add(new Product("Электроакустическая гитара Cort Gold-A8 Grand Auditorium Natural Glossy 2023", 20000, R.drawable.img5, "GTR005", "Cort", "Gold", "Электроакустика", "Среднее", "Grand Auditorium", "Правая", 6, 22, 24.75, ""));

products.add(new Product("Акустическая гитара Fender F-1050 Auditorium Natural w/case USA 1960s", 45000, R.drawable.img6, "GTR006", "Fender", "RG", "Акустическая гитара", "Новая", "Auditorium", "Правая", 6, 24, 25.5, ""));

products.add(new Product("Акустическая гитара Yamaha L10 Natural Gloss Japan 1980 Case", 35000, R.drawable.img7, "GTR007", "Yamaha", "L10", "Акустическая гитара", "Новая", "Dreadnought", "Правая", 6, 24, 24.75, ""));

products.add(new Product("Электрогитара B.C. Rich Mockingbird Supreme H-H Pearl White USA 1984", 55000, R.drawable.img8, "GTR008", "B.C. Rich", "SE Custom", "Электрогитара", "Плохое", "Mockingbird", "Правая", 6, 24, 25.0, ""));

products.add(new Product("Электрогитара Kononykheen Breed Thirty Nine Red HS W/Gigbag Russia 2023", 48000, R.drawable.img9, "GTR009", "Fender", "Pro Series", "Электрогитара", "Превосходное", "Telecaster", "Правая", 6, 24, 25.5, ""));

products.add(new Product("Электрогитара Fender Japan Art Canvas Esquire S Katsushika Hokusai The Great Wave w/case Japan 2024", 42000, R.drawable.img10, "GTR010", "Fender", "C-1", "Электрогитара", "Очень хорошее", "Telecaster", "Правая", 6, 24, 25.5, ""));

return products;

}

private void loadProducts() {

new Thread(() -> {

// Получаем товары из базы данных

List<Product> dbProducts = db.productDao().getAllProducts();

// Получаем статические товары

List<Product> staticProducts = getStaticProducts();

// Объединяем списки товаров

List<Product> allProducts = new ArrayList<>();

allProducts.addAll(staticProducts);

allProducts.addAll(dbProducts);

runOnUiThread(() -> {

productList.clear();

productList.addAll(allProducts);

adapter.notifyDataSetChanged();

// Применяем фильтрацию по категории, если она выбрана

if (currentCategoryType != null) {

filterProductsByCategory();

}

});

}).start();

}

@Override

protected void onResume() {

super.onResume();

// Обновляем список товаров при возвращении на экран

loadProducts();

}

private void setupBottomNavigation() {

bottomNavigationView.setOnNavigationItemSelectedListener(this::onNavigationItemSelected);

}

private boolean onNavigationItemSelected(MenuItem item) {

int itemId = item.getItemId();

Intent intent;

if (itemId == R.id.navigation\_home) {

intent = new Intent(CatalogActivity.this, MainActivity.class);

} else if (itemId == R.id.navigation\_catalog) {

// Уже в каталоге

return true;

} else if (itemId == R.id.navigation\_favorites) {

intent = new Intent(CatalogActivity.this, FavoritesActivity.class);

} else if (itemId == R.id.navigation\_cart) {

intent = new Intent(CatalogActivity.this, CartActivity.class);

} else if (itemId == R.id.navigation\_profile) {

intent = new Intent(CatalogActivity.this, ProfileActivity.class);

} else {

return false;

}

intent.putExtra("user\_email", userEmail);

startActivity(intent);

finish();

return true;

}

private void checkAdminStatus() {

new Thread(() -> {

User currentUser = db.userDao().getUserByEmail(userEmail);

isAdmin = currentUser != null && currentUser.isAdmin();

runOnUiThread(() -> {

if (isAdmin) {

// Если пользователь администратор, показываем кнопку добавления товара

addProductFab.setVisibility(View.VISIBLE);

setupAddProductButton();

} else {

// Если пользователь не администратор, скрываем кнопку

addProductFab.setVisibility(View.GONE);

}

// Обновляем статус администратора в адаптере

adapter.setAdmin(isAdmin);

});

}).start();

}

private void setupAddProductButton() {

addProductFab.setOnClickListener(v -> {

Intent intent = new Intent(this, AddProductActivity.class);

// Получаем email текущего пользователя из SharedPreferences

SharedPreferences prefs = getSharedPreferences("user\_prefs", MODE\_PRIVATE);

String userEmail = prefs.getString("user\_email", "");

intent.putExtra("user\_email", userEmail);

startActivity(intent);

});

}

}

~12) EditProfileActivity.java;~

package com.example.bolotnyiysen;

import android.content.Intent;

import android.content.SharedPreferences;

import android.os.Bundle;

import android.text.Editable;

import android.text.TextWatcher;

import android.view.MenuItem;

import android.widget.ArrayAdapter;

import android.widget.AutoCompleteTextView;

import android.widget.Toast;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity;

import androidx.appcompat.widget.Toolbar;

import com.google.android.material.bottomnavigation.BottomNavigationView;

import com.google.android.material.button.MaterialButton;

import com.google.android.material.textfield.TextInputEditText;

import com.google.android.material.textfield.TextInputLayout;

import com.google.gson.Gson;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

import java.util.regex.Pattern;

public class EditProfileActivity extends AppCompatActivity {

private BottomNavigationView bottomNavigationView;

private TextInputEditText fioEditText;

private TextInputEditText phoneEditText;

private TextInputEditText cityEditText;

private AutoCompleteTextView genderAutoComplete;

private MaterialButton saveButton;

private SharedPreferences sharedPreferences;

private static final String PREFS\_NAME = "UserProfile";

private String userEmail;

private AppDatabase appDatabase;

private UserDao userDao;

private ExecutorService executorService;

// Паттерны для валидации

private static final Pattern FIO\_PATTERN = Pattern.compile("^[А-Яа-яЁё\\s-]{3,}$");

private static final Pattern PHONE\_PATTERN = Pattern.compile("^\\+7\\d{10}$");

private static final Pattern CITY\_PATTERN = Pattern.compile("^[А-Яа-яЁё\\s-]{2,}$");

private static final Pattern EMAIL\_PATTERN = Pattern.compile("^[A-Za-z0-9+\_.-]+@(.+)$");

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_edit\_profile);

// Получаем email из Intent

userEmail = getIntent().getStringExtra("user\_email");

if (userEmail == null || !EMAIL\_PATTERN.matcher(userEmail).matches()) {

Toast.makeText(this, "Ошибка: неверный email", Toast.LENGTH\_SHORT).show();

finish();

return;

}

// Инициализация базы данных

appDatabase = AppDatabase.getInstance(this);

userDao = appDatabase.userDao();

executorService = Executors.newSingleThreadExecutor();

// Инициализация тулбара

Toolbar toolbar = findViewById(R.id.toolbar);

setSupportActionBar(toolbar);

getSupportActionBar().setDisplayHomeAsUpEnabled(true);

getSupportActionBar().setTitle("Редактирование профиля");

// Инициализация SharedPreferences

sharedPreferences = getSharedPreferences(PREFS\_NAME, MODE\_PRIVATE);

// Инициализация полей ввода

fioEditText = findViewById(R.id.nameEditText);

phoneEditText = findViewById(R.id.phoneEditText);

cityEditText = findViewById(R.id.cityEditText);

genderAutoComplete = findViewById(R.id.genderAutoComplete);

saveButton = findViewById(R.id.saveButton);

if (saveButton == null) {

Toast.makeText(this, "Ошибка: кнопка сохранения не найдена", Toast.LENGTH\_SHORT).show();

return;

}

// Настройка выпадающего списка для пола

String[] genders = {"Мужской", "Женский"};

ArrayAdapter<String> adapter = new ArrayAdapter<>(this, R.layout.dropdown\_item, genders);

genderAutoComplete.setAdapter(adapter);

genderAutoComplete.setOnItemClickListener((parent, view, position, id) -> {

String selectedGender = (String) parent.getItemAtPosition(position);

genderAutoComplete.setText(selectedGender, false);

});

// Добавляем маски и валидацию для полей

setupFieldValidation();

// Загрузка существующих данных профиля

loadProfile();

// Обработчик нажатия на кнопку сохранения

saveButton.setOnClickListener(v -> saveProfile());

// Инициализация нижней навигации

bottomNavigationView = findViewById(R.id.bottomNavigationView);

if (bottomNavigationView != null) {

setupBottomNavigation();

// Устанавливаем выбранный пункт меню

bottomNavigationView.setSelectedItemId(R.id.navigation\_profile);

}

}

private void setupFieldValidation() {

// Валидация ФИО

fioEditText.addTextChangedListener(new TextWatcher() {

@Override

public void beforeTextChanged(CharSequence s, int start, int count, int after) {}

@Override

public void onTextChanged(CharSequence s, int start, int before, int count) {}

@Override

public void afterTextChanged(Editable s) {

String fio = s.toString().trim();

TextInputLayout layout = (TextInputLayout) fioEditText.getParent();

if (!FIO\_PATTERN.matcher(fio).matches()) {

layout.setError("Введите корректное ФИО");

} else {

layout.setError(null);

}

}

});

// Валидация телефона

phoneEditText.addTextChangedListener(new TextWatcher() {

@Override

public void beforeTextChanged(CharSequence s, int start, int count, int after) {}

@Override

public void onTextChanged(CharSequence s, int start, int before, int count) {

String phone = s.toString().replaceAll("[^0-9+]", "");

if (!phone.startsWith("+7") && !phone.isEmpty()) {

phone = "+7" + phone;

}

if (!phone.equals(s.toString())) {

phoneEditText.setText(phone);

phoneEditText.setSelection(phone.length());

}

}

@Override

public void afterTextChanged(Editable s) {

String phone = s.toString();

TextInputLayout layout = (TextInputLayout) phoneEditText.getParent();

if (!PHONE\_PATTERN.matcher(phone).matches()) {

layout.setError("Введите корректный номер телефона");

} else {

layout.setError(null);

}

}

});

// Валидация города

cityEditText.addTextChangedListener(new TextWatcher() {

@Override

public void beforeTextChanged(CharSequence s, int start, int count, int after) {}

@Override

public void onTextChanged(CharSequence s, int start, int before, int count) {}

@Override

public void afterTextChanged(Editable s) {

String city = s.toString().trim();

TextInputLayout layout = (TextInputLayout) cityEditText.getParent();

if (!CITY\_PATTERN.matcher(city).matches()) {

layout.setError("Введите корректное название города");

} else {

layout.setError(null);

}

}

});

}

private void loadProfile() {

executorService.execute(() -> {

User user = userDao.getUserByEmail(userEmail);

if (user != null) {

runOnUiThread(() -> {

fioEditText.setText(user.getFio());

phoneEditText.setText(user.getPhone());

cityEditText.setText(user.getCity());

genderAutoComplete.setText(user.getGender());

});

} else {

// Если пользователь не найден в базе, пробуем загрузить из SharedPreferences

String profileJson = sharedPreferences.getString(userEmail, null);

if (profileJson != null) {

Gson gson = new Gson();

UserProfile profile = gson.fromJson(profileJson, UserProfile.class);

runOnUiThread(() -> {

fioEditText.setText(profile.getFio());

phoneEditText.setText(profile.getPhone());

cityEditText.setText(profile.getCity());

genderAutoComplete.setText(profile.getGender());

});

}

}

});

}

private void saveProfile() {

String fio = fioEditText.getText().toString().trim();

String phone = phoneEditText.getText().toString().trim();

String city = cityEditText.getText().toString().trim();

String gender = genderAutoComplete.getText().toString().trim();

// Проверка валидации

if (!FIO\_PATTERN.matcher(fio).matches()) {

Toast.makeText(this, "Пожалуйста, введите корректное ФИО", Toast.LENGTH\_SHORT).show();

return;

}

if (!PHONE\_PATTERN.matcher(phone).matches()) {

Toast.makeText(this, "Пожалуйста, введите корректный номер телефона", Toast.LENGTH\_SHORT).show();

return;

}

if (!CITY\_PATTERN.matcher(city).matches()) {

Toast.makeText(this, "Пожалуйста, введите корректное название города", Toast.LENGTH\_SHORT).show();

return;

}

if (gender.isEmpty()) {

Toast.makeText(this, "Пожалуйста, выберите пол", Toast.LENGTH\_SHORT).show();

return;

}

executorService.execute(() -> {

try {

User user = userDao.getUserByEmail(userEmail);

if (user != null) {

user.setFio(fio);

user.setPhone(phone);

user.setCity(city);

user.setGender(gender);

userDao.update(user);

// Также сохраняем в SharedPreferences для совместимости

UserProfile profile = new UserProfile(fio, phone, city, gender, userEmail);

Gson gson = new Gson();

String profileJson = gson.toJson(profile);

sharedPreferences.edit().putString(userEmail, profileJson).apply();

runOnUiThread(() -> {

Toast.makeText(this, "Профиль успешно обновлен", Toast.LENGTH\_SHORT).show();

setResult(RESULT\_OK);

finish();

});

} else {

runOnUiThread(() -> {

Toast.makeText(this, "Ошибка: пользователь не найден", Toast.LENGTH\_SHORT).show();

});

}

} catch (Exception e) {

runOnUiThread(() -> {

Toast.makeText(this, "Ошибка при сохранении: " + e.getMessage(), Toast.LENGTH\_LONG).show();

});

}

});

}

private void setupBottomNavigation() {

if (bottomNavigationView != null) {

bottomNavigationView.setOnItemSelectedListener(item -> {

int itemId = item.getItemId();

Intent intent;

if (itemId == R.id.navigation\_home) {

intent = new Intent(EditProfileActivity.this, MainActivity.class);

} else if (itemId == R.id.navigation\_catalog) {

intent = new Intent(EditProfileActivity.this, CatalogActivity.class);

} else if (itemId == R.id.navigation\_favorites) {

intent = new Intent(EditProfileActivity.this, FavoritesActivity.class);

} else if (itemId == R.id.navigation\_cart) {

intent = new Intent(EditProfileActivity.this, CartActivity.class);

} else if (itemId == R.id.navigation\_profile) {

intent = new Intent(EditProfileActivity.this, ProfileActivity.class);

} else {

return false;

}

intent.putExtra("user\_email", userEmail);

startActivity(intent);

finish();

return true;

});

}

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

if (item.getItemId() == android.R.id.home) {

onBackPressed();

return true;

}

return super.onOptionsItemSelected(item);

}

@Override

protected void onDestroy() {

super.onDestroy();

executorService.shutdown();

}

}

~13) EditProfileNewActivity.java;~

package com.example.bolotnyiysen;

import android.content.Intent;

import android.content.SharedPreferences;

import android.os.Bundle;

import android.text.Editable;

import android.text.TextWatcher;

import android.view.MenuItem;

import android.widget.ArrayAdapter;

import android.widget.AutoCompleteTextView;

import android.widget.Toast;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity;

import androidx.appcompat.widget.Toolbar;

import com.google.android.material.bottomnavigation.BottomNavigationView;

import com.google.android.material.button.MaterialButton;

import com.google.android.material.textfield.TextInputEditText;

import com.google.android.material.textfield.TextInputLayout;

import com.google.gson.Gson;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

import java.util.regex.Pattern;

public class EditProfileNewActivity extends AppCompatActivity {

private BottomNavigationView bottomNavigationView;

private TextInputEditText fioEditText;

private TextInputEditText phoneEditText;

private TextInputEditText cityEditText;

private AutoCompleteTextView genderAutoComplete;

private MaterialButton saveButton;

private SharedPreferences sharedPreferences;

private static final String PREFS\_NAME = "UserProfile";

private String userEmail;

private AppDatabase appDatabase;

private UserDao userDao;

private ExecutorService executorService;

// Паттерны для валидации

private static final Pattern FIO\_PATTERN = Pattern.compile("^[А-Яа-яЁё\\s-]{3,}$");

private static final Pattern PHONE\_PATTERN = Pattern.compile("^\\+7\\d{10}$");

private static final Pattern CITY\_PATTERN = Pattern.compile("^[А-Яа-яЁё\\s-]{2,}$");

private static final Pattern EMAIL\_PATTERN = Pattern.compile("^[A-Za-z0-9+\_.-]+@(.+)$");

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_edit\_profile\_new);

// Получаем email из Intent

userEmail = getIntent().getStringExtra("user\_email");

if (userEmail == null || !EMAIL\_PATTERN.matcher(userEmail).matches()) {

Toast.makeText(this, "Ошибка: неверный email", Toast.LENGTH\_SHORT).show();

finish();

return;

}

// Инициализация базы данных

appDatabase = AppDatabase.getInstance(this);

userDao = appDatabase.userDao();

executorService = Executors.newSingleThreadExecutor();

// Инициализация тулбара

Toolbar toolbar = findViewById(R.id.toolbar);

setSupportActionBar(toolbar);

getSupportActionBar().setDisplayHomeAsUpEnabled(true);

getSupportActionBar().setTitle("Редактирование профиля");

// Инициализация SharedPreferences

sharedPreferences = getSharedPreferences(PREFS\_NAME, MODE\_PRIVATE);

// Инициализация полей ввода

fioEditText = findViewById(R.id.fioEditText);

phoneEditText = findViewById(R.id.phoneEditText);

cityEditText = findViewById(R.id.cityEditText);

genderAutoComplete = findViewById(R.id.genderAutoComplete);

saveButton = findViewById(R.id.saveButton);

// Настройка выпадающего списка для пола

String[] genders = {"Мужской", "Женский"};

ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.simple\_dropdown\_item\_1line, genders);

genderAutoComplete.setAdapter(adapter);

genderAutoComplete.setThreshold(0); // Показывать список сразу при фокусе

// Добавляем маски и валидацию для полей

setupFieldValidation();

// Загрузка существующих данных профиля

loadProfile();

// Обработчик нажатия на кнопку сохранения

saveButton.setOnClickListener(v -> saveProfile());

bottomNavigationView = findViewById(R.id.bottomNavigationView);

setupBottomNavigation();

// Устанавливаем выбранный пункт меню

bottomNavigationView.setSelectedItemId(R.id.navigation\_profile);

}

private void setupFieldValidation() {

// Валидация ФИО

fioEditText.addTextChangedListener(new TextWatcher() {

@Override

public void beforeTextChanged(CharSequence s, int start, int count, int after) {}

@Override

public void onTextChanged(CharSequence s, int start, int before, int count) {}

@Override

public void afterTextChanged(Editable s) {

String fio = s.toString().trim();

TextInputLayout layout = findViewById(R.id.fioLayout);

if (!FIO\_PATTERN.matcher(fio).matches()) {

layout.setError("Введите корректное ФИО");

} else {

layout.setError(null);

}

}

});

// Валидация телефона

phoneEditText.addTextChangedListener(new TextWatcher() {

@Override

public void beforeTextChanged(CharSequence s, int start, int count, int after) {}

@Override

public void onTextChanged(CharSequence s, int start, int before, int count) {

String phone = s.toString().replaceAll("[^0-9+]", "");

if (!phone.startsWith("+7") && !phone.isEmpty()) {

phone = "+7" + phone;

}

if (!phone.equals(s.toString())) {

phoneEditText.setText(phone);

phoneEditText.setSelection(phone.length());

}

}

@Override

public void afterTextChanged(Editable s) {

String phone = s.toString();

TextInputLayout layout = findViewById(R.id.phoneLayout);

if (!PHONE\_PATTERN.matcher(phone).matches()) {

layout.setError("Введите корректный номер телефона");

} else {

layout.setError(null);

}

}

});

// Валидация города

cityEditText.addTextChangedListener(new TextWatcher() {

@Override

public void beforeTextChanged(CharSequence s, int start, int count, int after) {}

@Override

public void onTextChanged(CharSequence s, int start, int before, int count) {}

@Override

public void afterTextChanged(Editable s) {

String city = s.toString().trim();

TextInputLayout layout = findViewById(R.id.cityLayout);

if (!CITY\_PATTERN.matcher(city).matches()) {

layout.setError("Введите корректное название города");

} else {

layout.setError(null);

}

}

});

}

private void loadProfile() {

executorService.execute(() -> {

User user = userDao.getUserByEmail(userEmail);

if (user != null) {

runOnUiThread(() -> {

fioEditText.setText(user.getFio());

phoneEditText.setText(user.getPhone());

cityEditText.setText(user.getCity());

genderAutoComplete.setText(user.getGender());

});

} else {

// Если пользователь не найден в базе, пробуем загрузить из SharedPreferences

String profileJson = sharedPreferences.getString(userEmail, null);

if (profileJson != null) {

Gson gson = new Gson();

UserProfile profile = gson.fromJson(profileJson, UserProfile.class);

runOnUiThread(() -> {

fioEditText.setText(profile.getFio());

phoneEditText.setText(profile.getPhone());

cityEditText.setText(profile.getCity());

genderAutoComplete.setText(profile.getGender());

});

}

}

});

}

private void saveProfile() {

String fio = fioEditText.getText().toString().trim();

String phone = phoneEditText.getText().toString().trim();

String city = cityEditText.getText().toString().trim();

String gender = genderAutoComplete.getText().toString().trim();

// Проверка валидации

if (!FIO\_PATTERN.matcher(fio).matches()) {

Toast.makeText(this, "Пожалуйста, введите корректное ФИО", Toast.LENGTH\_SHORT).show();

return;

}

if (!PHONE\_PATTERN.matcher(phone).matches()) {

Toast.makeText(this, "Пожалуйста, введите корректный номер телефона", Toast.LENGTH\_SHORT).show();

return;

}

if (!CITY\_PATTERN.matcher(city).matches()) {

Toast.makeText(this, "Пожалуйста, введите корректное название города", Toast.LENGTH\_SHORT).show();

return;

}

if (gender.isEmpty()) {

Toast.makeText(this, "Пожалуйста, выберите пол", Toast.LENGTH\_SHORT).show();

return;

}

executorService.execute(() -> {

try {

User user = userDao.getUserByEmail(userEmail);

if (user != null) {

user.setFio(fio);

user.setPhone(phone);

user.setCity(city);

user.setGender(gender);

userDao.update(user);

// Также сохраняем в SharedPreferences для совместимости

UserProfile profile = new UserProfile(fio, phone, city, gender, userEmail);

Gson gson = new Gson();

String profileJson = gson.toJson(profile);

sharedPreferences.edit().putString(userEmail, profileJson).apply();

runOnUiThread(() -> {

Toast.makeText(this, "Профиль успешно обновлен", Toast.LENGTH\_SHORT).show();

setResult(RESULT\_OK);

finish();

});

} else {

runOnUiThread(() -> {

Toast.makeText(this, "Ошибка: пользователь не найден", Toast.LENGTH\_SHORT).show();

});

}

} catch (Exception e) {

runOnUiThread(() -> {

Toast.makeText(this, "Ошибка при сохранении: " + e.getMessage(), Toast.LENGTH\_LONG).show();

});

}

});

}

private void setupBottomNavigation() {

bottomNavigationView.setOnNavigationItemSelectedListener(item -> {

int itemId = item.getItemId();

Intent intent;

if (itemId == R.id.navigation\_home) {

intent = new Intent(EditProfileNewActivity.this, MainActivity.class);

} else if (itemId == R.id.navigation\_catalog) {

intent = new Intent(EditProfileNewActivity.this, CatalogActivity.class);

} else if (itemId == R.id.navigation\_favorites) {

intent = new Intent(EditProfileNewActivity.this, FavoritesActivity.class);

} else if (itemId == R.id.navigation\_cart) {

intent = new Intent(EditProfileNewActivity.this, CartActivity.class);

} else if (itemId == R.id.navigation\_profile) {

intent = new Intent(EditProfileNewActivity.this, ProfileActivity.class);

} else {

return false;

}

intent.putExtra("user\_email", userEmail);

startActivity(intent);

finish();

return true;

});

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

if (item.getItemId() == android.R.id.home) {

onBackPressed();

return true;

}

return super.onOptionsItemSelected(item);

}

@Override

protected void onDestroy() {

super.onDestroy();

executorService.shutdown();

}

}

~14) FavoritesActivity.java;~

package com.example.bolotnyiysen;

import android.content.Intent;

import android.os.Bundle;

import android.view.MenuItem;

import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import androidx.recyclerview.widget.LinearLayoutManager;

import androidx.recyclerview.widget.RecyclerView;

import com.google.android.material.bottomnavigation.BottomNavigationView;

import java.util.List;

public class FavoritesActivity extends AppCompatActivity {

private RecyclerView recyclerView;

private ProductAdapter adapter;

private String userEmail;

private CartManager cartManager;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_favorites);

// Получаем email из Intent

userEmail = getIntent().getStringExtra("user\_email");

if (userEmail == null) {

Toast.makeText(this, "Ошибка: неверный email", Toast.LENGTH\_SHORT).show();

finish();

return;

}

// Инициализация CartManager

cartManager = CartManager.getInstance(this);

cartManager.setCurrentUser(userEmail);

// Настройка RecyclerView

recyclerView = findViewById(R.id.favoritesRecyclerView);

recyclerView.setLayoutManager(new LinearLayoutManager(this));

adapter = new ProductAdapter(FavoritesManager.getInstance().getFavoriteItems(), this);

adapter.setFavoritesView(true);

recyclerView.setAdapter(adapter);

setupBottomNavigation();

}

private void setupBottomNavigation() {

BottomNavigationView bottomNavigationView = findViewById(R.id.bottomNavigationView);

bottomNavigationView.setOnNavigationItemSelectedListener(this::onNavigationItemSelected);

}

private boolean onNavigationItemSelected(MenuItem item) {

int itemId = item.getItemId();

Intent intent;

if (itemId == R.id.navigation\_home) {

intent = new Intent(FavoritesActivity.this, MainActivity.class);

} else if (itemId == R.id.navigation\_catalog) {

intent = new Intent(FavoritesActivity.this, CatalogActivity.class);

} else if (itemId == R.id.navigation\_favorites) {

// Уже в избранном

return true;

} else if (itemId == R.id.navigation\_cart) {

intent = new Intent(FavoritesActivity.this, CartActivity.class);

} else if (itemId == R.id.navigation\_profile) {

intent = new Intent(FavoritesActivity.this, ProfileActivity.class);

} else {

return false;

}

intent.putExtra("user\_email", userEmail);

startActivity(intent);

finish();

return true;

}

}

~15) FavoritesManager.java;~

package com.example.bolotnyiysen;

import java.util.ArrayList;

import java.util.List;

public class FavoritesManager {

private static FavoritesManager instance;

private List<Product> favoriteItems;

private FavoritesManager() {

favoriteItems = new ArrayList<>();

}

public static FavoritesManager getInstance() {

if (instance == null) {

instance = new FavoritesManager();

}

return instance;

}

public void addToFavorites(Product product) {

if (!isFavorite(product)) {

favoriteItems.add(product);

}

}

public void removeFromFavorites(Product product) {

favoriteItems.removeIf(p -> p.getId() == product.getId());

}

public List<Product> getFavoriteItems() {

return new ArrayList<>(favoriteItems);

}

public boolean isFavorite(Product product) {

return favoriteItems.stream().anyMatch(p -> p.getId() == product.getId());

}

}

~16) ForgotPasswordActivity.java;~

package com.example.bolotnyiysen;

import android.content.Intent;

import android.content.SharedPreferences;

import android.os.Bundle;

import android.text.Editable;

import android.text.TextWatcher;

import android.widget.Button;

import android.widget.EditText;

import android.widget.TextView;

import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import com.google.android.material.textfield.TextInputLayout;

import com.google.gson.Gson;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

import java.util.regex.Pattern;

public class ForgotPasswordActivity extends AppCompatActivity {

private EditText emailEditText;

private EditText codeWordEditText;

private EditText newPasswordEditText;

private Button verifyButton;

private TextView backToLoginTextView;

private UserDao userDao;

private SharedPreferences sharedPreferences;

private static final String PREFS\_NAME = "UserProfile";

private static final Pattern PASSWORD\_PATTERN = Pattern.compile("^(?=.\*[0-9])(?=.\*[a-z])(?=.\*[A-Z])(?=.\*[@#$%^&+=])(?=\\S+$).{8,}$");

private ExecutorService executorService;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_forgot\_password);

// Инициализация базы данных и ExecutorService

userDao = AppDatabase.getInstance(this).userDao();

sharedPreferences = getSharedPreferences(PREFS\_NAME, MODE\_PRIVATE);

executorService = Executors.newSingleThreadExecutor();

// Инициализация полей ввода

emailEditText = findViewById(R.id.emailEditText);

codeWordEditText = findViewById(R.id.codeWordEditText);

newPasswordEditText = findViewById(R.id.newPasswordEditText);

verifyButton = findViewById(R.id.verifyButton);

backToLoginTextView = findViewById(R.id.backToLoginTextView);

// Добавляем валидацию для нового пароля

newPasswordEditText.addTextChangedListener(new TextWatcher() {

@Override

public void beforeTextChanged(CharSequence s, int start, int count, int after) {}

@Override

public void onTextChanged(CharSequence s, int start, int before, int count) {}

@Override

public void afterTextChanged(Editable s) {

String password = s.toString();

TextInputLayout layout = findViewById(R.id.newPasswordLayout);

if (!PASSWORD\_PATTERN.matcher(password).matches()) {

layout.setError("Пароль должен содержать минимум 8 символов, включая цифры, буквы верхнего и нижнего регистра и специальные символы");

} else {

layout.setError(null);

}

}

});

// Обработчик нажатия на кнопку сброса пароля

verifyButton.setOnClickListener(v -> {

String email = emailEditText.getText().toString().trim();

String codeWord = codeWordEditText.getText().toString().trim();

String newPassword = newPasswordEditText.getText().toString().trim();

if (email.isEmpty() || codeWord.isEmpty() || newPassword.isEmpty()) {

Toast.makeText(ForgotPasswordActivity.this, "Пожалуйста, заполните все поля", Toast.LENGTH\_SHORT).show();

return;

}

if (!PASSWORD\_PATTERN.matcher(newPassword).matches()) {

Toast.makeText(ForgotPasswordActivity.this, "Пароль не соответствует требованиям", Toast.LENGTH\_SHORT).show();

return;

}

// Выполняем операции с базой данных в фоновом потоке

executorService.execute(() -> {

User user = userDao.getUserByEmailAndCodeWord(email, codeWord);

if (user != null) {

int result = userDao.updatePasswordWithCodeWord(email, codeWord, newPassword);

runOnUiThread(() -> {

if (result > 0) {

// Обновляем пароль в SharedPreferences

SharedPreferences.Editor editor = sharedPreferences.edit();

editor.putString("user\_password", newPassword);

editor.apply();

Toast.makeText(ForgotPasswordActivity.this, "Пароль успешно изменен", Toast.LENGTH\_SHORT).show();

Intent intent = new Intent(ForgotPasswordActivity.this, LoginActivity.class);

startActivity(intent);

finish();

} else {

Toast.makeText(ForgotPasswordActivity.this, "Ошибка при изменении пароля", Toast.LENGTH\_SHORT).show();

}

});

} else {

runOnUiThread(() -> {

Toast.makeText(ForgotPasswordActivity.this, "Неверный email или кодовое слово", Toast.LENGTH\_SHORT).show();

});

}

});

});

// Обработчик нажатия на текст возврата к входу

backToLoginTextView.setOnClickListener(v -> {

startActivity(new Intent(ForgotPasswordActivity.this, LoginActivity.class));

finish();

});

}

@Override

protected void onDestroy() {

super.onDestroy();

executorService.shutdown();

}

}

~17) ImagePagerAdapter.java;~

package com.example.bolotnyiysen;

import android.content.Context;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.ImageView;

import androidx.annotation.NonNull;

import androidx.recyclerview.widget.RecyclerView;

import com.bumptech.glide.Glide;

import java.util.List;

public class ImagePagerAdapter extends RecyclerView.Adapter<ImagePagerAdapter.ImageViewHolder> {

private Context context;

private List<Integer> imageResources;

private List<String> imagePaths;

public ImagePagerAdapter(Context context, List<Integer> imageResources) {

this.context = context;

this.imageResources = imageResources;

}

public ImagePagerAdapter(Context context, List<Integer> imageResources, List<String> imagePaths) {

this.context = context;

this.imageResources = imageResources;

this.imagePaths = imagePaths;

}

@NonNull

@Override

public ImageViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {

View view = LayoutInflater.from(context).inflate(R.layout.item\_image, parent, false);

return new ImageViewHolder(view);

}

@Override

public void onBindViewHolder(@NonNull ImageViewHolder holder, int position) {

if (imagePaths != null && position < imagePaths.size() && imagePaths.get(position) != null) {

// Если есть путь к файлу, загружаем изображение из файла

Glide.with(context)

.load(imagePaths.get(position))

.into(holder.imageView);

} else if (imageResources != null && position < imageResources.size()) {

// Иначе загружаем изображение из ресурсов

Glide.with(context)

.load(imageResources.get(position))

.into(holder.imageView);

}

}

@Override

public int getItemCount() {

if (imagePaths != null && !imagePaths.isEmpty()) {

return imagePaths.size();

}

return imageResources != null ? imageResources.size() : 0;

}

static class ImageViewHolder extends RecyclerView.ViewHolder {

ImageView imageView;

ImageViewHolder(@NonNull View itemView) {

super(itemView);

imageView = itemView.findViewById(R.id.imageView);

}

}

}

~18) LoginActivity.java;~

package com.example.bolotnyiysen;

import android.content.Intent;

import android.content.SharedPreferences;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

public class LoginActivity extends AppCompatActivity {

private EditText emailEditText;

private EditText passwordEditText;

private Button loginButton;

private Button registerButton;

private Button forgotPasswordButton;

private Button adminLoginButton;

private AppDatabase appDatabase;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_login);

// Инициализация базы данных

appDatabase = AppDatabase.getInstance(this);

// Инициализация полей ввода

emailEditText = findViewById(R.id.emailEditText);

passwordEditText = findViewById(R.id.passwordEditText);

loginButton = findViewById(R.id.loginButton);

registerButton = findViewById(R.id.registerButton);

forgotPasswordButton = findViewById(R.id.forgotPasswordButton);

adminLoginButton = findViewById(R.id.adminLoginButton);

// Обработчик нажатия на кнопку входа

loginButton.setOnClickListener(v -> loginUser());

// Обработчик нажатия на кнопку входа администратора

adminLoginButton.setOnClickListener(v -> {

emailEditText.setText("admin@example.com");

passwordEditText.setText("admin123");

Toast.makeText(this, "Введите данные администратора", Toast.LENGTH\_SHORT).show();

});

// Обработчик нажатия на кнопку регистрации

registerButton.setOnClickListener(v -> {

Intent intent = new Intent(LoginActivity.this, RegisterActivity.class);

startActivity(intent);

});

// Обработчик нажатия на кнопку восстановления пароля

forgotPasswordButton.setOnClickListener(v -> {

Intent intent = new Intent(LoginActivity.this, ForgotPasswordActivity.class);

startActivity(intent);

});

}

private void loginUser() {

String email = emailEditText.getText().toString().trim();

String password = passwordEditText.getText().toString().trim();

if (email.isEmpty() || password.isEmpty()) {

Toast.makeText(this, "Пожалуйста, заполните все поля", Toast.LENGTH\_SHORT).show();

return;

}

new Thread(() -> {

// Проверяем существование администратора

User admin = appDatabase.userDao().getAdmin();

if (admin == null) {

runOnUiThread(() -> {

Toast.makeText(this, "Администратор не найден в базе данных", Toast.LENGTH\_LONG).show();

});

return;

}

User user = appDatabase.userDao().getUserByEmailAndPassword(email, password);

runOnUiThread(() -> {

if (user != null) {

// Сохраняем email пользователя

SharedPreferences preferences = getSharedPreferences("user\_prefs", MODE\_PRIVATE);

preferences.edit().putString("user\_email", email).apply();

if (user.isAdmin()) {

Toast.makeText(this, "Добро пожаловать, администратор!", Toast.LENGTH\_SHORT).show();

Intent intent = new Intent(LoginActivity.this, MetricsActivity.class);

startActivity(intent);

} else {

Toast.makeText(this, "Вход выполнен успешно", Toast.LENGTH\_SHORT).show();

Intent intent = new Intent(LoginActivity.this, MainActivity.class);

intent.putExtra("user\_email", email);

startActivity(intent);

}

finish();

} else {

Toast.makeText(this, "Неверный email или пароль", Toast.LENGTH\_SHORT).show();

}

});

}).start();

}

}

~19) MainActivity.java;~

package com.example.bolotnyiysen;

import android.content.Intent;

import android.os.Bundle;

import android.text.Editable;

import android.text.TextWatcher;

import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import androidx.recyclerview.widget.LinearLayoutManager;

import androidx.recyclerview.widget.RecyclerView;

import androidx.viewpager2.widget.ViewPager2;

import com.google.android.material.bottomnavigation.BottomNavigationView;

import com.google.android.material.textfield.TextInputLayout;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

public class MainActivity extends AppCompatActivity {

private RecyclerView recyclerView;

private ProductAdapter adapter;

private List<Product> productList;

private List<Product> filteredList;

private String userEmail;

private CartManager cartManager;

private ViewPager2 bannerViewPager;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

// Получаем email из Intent

userEmail = getIntent().getStringExtra("user\_email");

if (userEmail == null) {

Toast.makeText(this, "Ошибка: неверный email", Toast.LENGTH\_SHORT).show();

finish();

return;

}

// Инициализация CartManager

cartManager = CartManager.getInstance(this);

cartManager.setCurrentUser(userEmail);

// Инициализация баннеров

setupBanners();

// Инициализация списка товаров

productList = new ArrayList<>();

filteredList = new ArrayList<>();

initializeProducts();

// Настройка RecyclerView

recyclerView = findViewById(R.id.recyclerView);

recyclerView.setLayoutManager(new LinearLayoutManager(this));

adapter = new ProductAdapter(filteredList, this);

recyclerView.setAdapter(adapter);

// Настройка поиска

TextInputLayout searchLayout = findViewById(R.id.searchLayout);

searchLayout.getEditText().addTextChangedListener(new TextWatcher() {

@Override

public void beforeTextChanged(CharSequence s, int start, int count, int after) {}

@Override

public void onTextChanged(CharSequence s, int start, int before, int count) {

filterProducts(s.toString());

}

@Override

public void afterTextChanged(Editable s) {}

});

setupBottomNavigation();

}

private void setupBanners() {

bannerViewPager = findViewById(R.id.bannerViewPager);

List<Integer> bannerImages = Arrays.asList(

R.drawable.banner\_fender,

R.drawable.banner\_gibson,

R.drawable.banner\_ibanez

);

BannerAdapter bannerAdapter = new BannerAdapter(bannerImages);

bannerViewPager.setAdapter(bannerAdapter);

}

private void initializeProducts() {

productList.add(new Product("Электрогитара Magneto US-1300 RC/CAR Sonnet Classic Stratocaster HSS Candy Red",

28000, R.drawable.img1, "GTR001", "Magneto", "Sonnet Classic", "Электрогитара", "Новая", "Strat", "Правая", 6, 22, 25.5, ""));

productList.add(new Product("Электрогитара Magneto US-1300 RC/MKPW Sonnet Classic Stratocaster HSS Metallic Pearl White",

28000, R.drawable.img2, "GTR002", "Magneto", "Sonnet Classic", "Электрогитара", "Б/У", "Strat", "Правая", 6, 22, 24.75, ""));

productList.add(new Product("Электрогитара Gibson Les Paul Traditional Pro LPCGEBCHI H-H Black 2012 USA W/Case",

90000, R.drawable.img3, "GTR003", "Gibson", "Les Paul", "Электрогитара", "Новая", "Les Paul", "Правая", 6, 24, 25.5, ""));

productList.add(new Product("Электрогитара Fender Telecaster Custom 1972 Vintage Reissue HS Black w/gigbag Japan 1995",

60000, R.drawable.img4, "GTR004", "Fender", "Telecaster", "Электрогитара", "Новая", "Telecaster", "Правая", 6, 24, 25.0, ""));

productList.add(new Product("Электроакустическая гитара Cort Gold-A8 Grand Auditorium Natural Glossy 2023",

20000, R.drawable.img5, "GTR005", "Cort", "Gold", "Электроакустика", "Среднее", "Grand Auditorium", "Правая", 6, 22, 24.75, ""));

productList.add(new Product("Акустическая гитара Fender F-1050 Auditorium Natural w/case USA 1960s",

45000, R.drawable.img6, "GTR006", "Fender", "RG", "Акустическая гитара", "Новая", "Auditorium", "Правая", 6, 24, 25.5, ""));

productList.add(new Product("Акустическая гитара Yamaha L10 Natural Gloss Japan 1980 Case",

35000, R.drawable.img7, "GTR007", "Yamaha", "L10", "Акустическая гитара", "Новая", "Dreadnought", "Правая", 6, 24, 24.75, ""));

productList.add(new Product("Электрогитара B.C. Rich Mockingbird Supreme H-H Pearl White USA 1984",

55000, R.drawable.img8, "GTR008", "B.C. Rich", "SE Custom", "Электрогитара", "Плохое", "Mockingbird", "Правая", 6, 24, 25.0, ""));

productList.add(new Product("Электрогитара Kononykheen Breed Thirty Nine Red HS W/Gigbag Russia 2023",

48000, R.drawable.img9, "GTR009", "Fender", "Pro Series", "Электрогитара", "Превосходное", "Telecaster", "Правая", 6, 24, 25.5, ""));

productList.add(new Product("Электрогитара Fender Japan Art Canvas Esquire S Katsushika Hokusai The Great Wave w/case Japan 2024",

42000, R.drawable.img10, "GTR010", "Fender", "C-1", "Электрогитара", "Очень хорошее", "Telecaster", "Правая", 6, 24, 25.5, ""));

filteredList.addAll(productList);

}

private void filterProducts(String query) {

filteredList.clear();

if (query.isEmpty()) {

filteredList.addAll(productList);

} else {

String lowerQuery = query.toLowerCase();

for (Product product : productList) {

if (product.getName().toLowerCase().contains(lowerQuery)) {

filteredList.add(product);

}

}

}

adapter.notifyDataSetChanged();

}

private void setupBottomNavigation() {

BottomNavigationView bottomNavigationView = findViewById(R.id.bottomNavigationView);

bottomNavigationView.setOnNavigationItemSelectedListener(item -> {

int itemId = item.getItemId();

Intent intent;

if (itemId == R.id.navigation\_home) {

// Уже на главной

return true;

} else if (itemId == R.id.navigation\_catalog) {

intent = new Intent(MainActivity.this, CatalogActivity.class);

} else if (itemId == R.id.navigation\_favorites) {

intent = new Intent(MainActivity.this, FavoritesActivity.class);

} else if (itemId == R.id.navigation\_cart) {

intent = new Intent(MainActivity.this, CartActivity.class);

} else if (itemId == R.id.navigation\_profile) {

intent = new Intent(MainActivity.this, ProfileActivity.class);

} else {

return false;

}

intent.putExtra("user\_email", userEmail);

startActivity(intent);

finish();

return true;

});

}

}

~20) MetricsActivity.java;~

package com.example.bolotnyiysen;

import android.content.ActivityNotFoundException;

import android.content.Intent;

import android.content.SharedPreferences;

import android.graphics.Color;

import android.net.Uri;

import android.os.Bundle;

import android.util.Log;

import android.view.View;

import android.widget.Button;

import android.widget.TextView;

import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import androidx.appcompat.widget.Toolbar;

import androidx.core.content.FileProvider;

import androidx.recyclerview.widget.LinearLayoutManager;

import androidx.recyclerview.widget.RecyclerView;

import com.github.mikephil.charting.charts.PieChart;

import com.github.mikephil.charting.data.PieData;

import com.github.mikephil.charting.data.PieDataSet;

import com.github.mikephil.charting.data.PieEntry;

import com.github.mikephil.charting.formatter.PercentFormatter;

import com.github.mikephil.charting.utils.ColorTemplate;

import com.google.android.material.bottomnavigation.BottomNavigationView;

import com.google.android.material.card.MaterialCardView;

import com.google.android.material.floatingactionbutton.FloatingActionButton;

import java.io.File;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

public class MetricsActivity extends AppCompatActivity {

private AppDatabase appDatabase;

private TextView totalUsersTextView;

private TextView totalPurchasesTextView;

private TextView totalRevenueTextView;

private BottomNavigationView bottomNavigationView;

private RecyclerView usersRecyclerView;

private RecyclerView productsRecyclerView;

private UserAdapter userAdapter;

private ProductAdapter productAdapter;

private View metricsContainer;

private MaterialCardView revenueChartCard;

private PieChart revenueChart;

private FloatingActionButton addProductFab;

private FloatingActionButton exportRevenueFab;

private String userEmail;

private RevenueReportGenerator reportGenerator;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_metrics);

// Инициализация базы данных

appDatabase = AppDatabase.getInstance(this);

// Получение email пользователя из SharedPreferences

SharedPreferences prefs = getSharedPreferences("user\_prefs", MODE\_PRIVATE);

userEmail = prefs.getString("user\_email", "");

// Настройка тулбара

Toolbar toolbar = findViewById(R.id.toolbar);

setSupportActionBar(toolbar);

getSupportActionBar().setDisplayHomeAsUpEnabled(true);

getSupportActionBar().setTitle("Метрики");

// Инициализация компонентов

totalUsersTextView = findViewById(R.id.totalUsersTextView);

totalPurchasesTextView = findViewById(R.id.totalPurchasesTextView);

totalRevenueTextView = findViewById(R.id.totalRevenueTextView);

metricsContainer = findViewById(R.id.metricsContainer);

usersRecyclerView = findViewById(R.id.usersRecyclerView);

productsRecyclerView = findViewById(R.id.productsRecyclerView);

bottomNavigationView = findViewById(R.id.bottom\_navigation);

revenueChartCard = findViewById(R.id.revenueChartCard);

revenueChart = findViewById(R.id.revenueChart);

addProductFab = findViewById(R.id.addProductFab);

exportRevenueFab = findViewById(R.id.exportRevenueFab);

reportGenerator = new RevenueReportGenerator(this);

// Настройка RecyclerView для списка пользователей

usersRecyclerView.setLayoutManager(new LinearLayoutManager(this));

userAdapter = new UserAdapter(new ArrayList<>(), this::deleteUser);

usersRecyclerView.setAdapter(userAdapter);

// Настройка RecyclerView для списка товаров

productsRecyclerView.setLayoutManager(new LinearLayoutManager(this));

productAdapter = new ProductAdapter(new ArrayList<>(), this);

productAdapter.setAdmin(true);

productAdapter.setOnProductDeleteListener(this::deleteProduct);

productsRecyclerView.setAdapter(productAdapter);

// Настройка кнопки выхода

Button logoutButton = findViewById(R.id.logoutButton);

if (logoutButton != null) {

logoutButton.setOnClickListener(v -> {

Intent intent = new Intent(MetricsActivity.this, LoginActivity.class);

intent.setFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK | Intent.FLAG\_ACTIVITY\_CLEAR\_TASK);

startActivity(intent);

finish();

});

}

// Настройка кнопки добавления товара

addProductFab.setOnClickListener(v -> {

Intent intent = new Intent(this, AddProductActivity.class);

intent.putExtra("user\_email", userEmail);

startActivity(intent);

});

// Настройка нижней навигации

setupBottomNavigation();

// Загрузка начальных данных

loadUsersMetrics();

setupExportButton();

}

private void setupBottomNavigation() {

bottomNavigationView.setOnItemSelectedListener(item -> {

int itemId = item.getItemId();

if (itemId == R.id.navigation\_users) {

showUsersList();

return true;

} else if (itemId == R.id.navigation\_orders) {

showOrdersMetrics();

return true;

} else if (itemId == R.id.navigation\_revenue) {

showRevenueMetrics();

return true;

} else if (itemId == R.id.navigation\_products) {

showProductsList();

return true;

}

return false;

});

}

private void showUsersList() {

// Скрываем все метрики

metricsContainer.setVisibility(View.GONE);

revenueChartCard.setVisibility(View.GONE);

productsRecyclerView.setVisibility(View.GONE);

addProductFab.setVisibility(View.GONE);

// Показываем список пользователей

usersRecyclerView.setVisibility(View.VISIBLE);

// Скрываем все текстовые представления

totalUsersTextView.setVisibility(View.GONE);

totalPurchasesTextView.setVisibility(View.GONE);

totalRevenueTextView.setVisibility(View.GONE);

// Скрываем элементы, связанные с заказами

TextView totalOrdersTextView = findViewById(R.id.totalOrdersTextView);

TextView userOrdersTextView = findViewById(R.id.userOrdersTextView);

if (totalOrdersTextView != null) totalOrdersTextView.setVisibility(View.GONE);

if (userOrdersTextView != null) userOrdersTextView.setVisibility(View.GONE);

getSupportActionBar().setTitle("Пользователи");

loadUsers();

}

private void showOrdersMetrics() {

metricsContainer.setVisibility(View.VISIBLE);

usersRecyclerView.setVisibility(View.GONE);

productsRecyclerView.setVisibility(View.GONE);

revenueChartCard.setVisibility(View.GONE);

addProductFab.setVisibility(View.GONE);

totalUsersTextView.setVisibility(View.GONE);

totalPurchasesTextView.setVisibility(View.GONE);

totalRevenueTextView.setVisibility(View.GONE);

// Показываем только элементы, связанные с заказами

TextView totalOrdersTextView = findViewById(R.id.totalOrdersTextView);

TextView userOrdersTextView = findViewById(R.id.userOrdersTextView);

if (totalOrdersTextView != null) totalOrdersTextView.setVisibility(View.VISIBLE);

if (userOrdersTextView != null) userOrdersTextView.setVisibility(View.VISIBLE);

getSupportActionBar().setTitle("Заказы");

loadOrdersMetrics();

}

private void showRevenueMetrics() {

metricsContainer.setVisibility(View.VISIBLE);

usersRecyclerView.setVisibility(View.GONE);

productsRecyclerView.setVisibility(View.GONE);

revenueChartCard.setVisibility(View.VISIBLE);

addProductFab.setVisibility(View.GONE);

totalUsersTextView.setVisibility(View.GONE);

totalPurchasesTextView.setVisibility(View.GONE);

totalRevenueTextView.setVisibility(View.VISIBLE);

// Скрываем элементы, связанные с заказами

TextView totalOrdersTextView = findViewById(R.id.totalOrdersTextView);

TextView userOrdersTextView = findViewById(R.id.userOrdersTextView);

if (totalOrdersTextView != null) totalOrdersTextView.setVisibility(View.GONE);

if (userOrdersTextView != null) userOrdersTextView.setVisibility(View.GONE);

getSupportActionBar().setTitle("Выручка");

loadRevenueMetrics();

}

private void showProductsList() {

// Скрываем все метрики

metricsContainer.setVisibility(View.GONE);

revenueChartCard.setVisibility(View.GONE);

usersRecyclerView.setVisibility(View.GONE);

// Показываем список товаров и кнопку добавления

productsRecyclerView.setVisibility(View.VISIBLE);

addProductFab.setVisibility(View.VISIBLE);

// Скрываем все текстовые представления

totalUsersTextView.setVisibility(View.GONE);

totalPurchasesTextView.setVisibility(View.GONE);

totalRevenueTextView.setVisibility(View.GONE);

// Скрываем элементы, связанные с заказами

TextView totalOrdersTextView = findViewById(R.id.totalOrdersTextView);

TextView userOrdersTextView = findViewById(R.id.userOrdersTextView);

if (totalOrdersTextView != null) totalOrdersTextView.setVisibility(View.GONE);

if (userOrdersTextView != null) userOrdersTextView.setVisibility(View.GONE);

getSupportActionBar().setTitle("Товары");

loadProducts();

}

private void loadProducts() {

new Thread(() -> {

try {

List<Product> products = appDatabase.productDao().getAllProducts();

runOnUiThread(() -> {

productAdapter.updateProducts(products);

});

} catch (Exception e) {

Log.e("MetricsActivity", "Error loading products: " + e.getMessage(), e);

runOnUiThread(() -> {

Toast.makeText(this, "Ошибка при загрузке товаров", Toast.LENGTH\_SHORT).show();

});

}

}).start();

}

private void loadUsersMetrics() {

new Thread(() -> {

try {

int totalUsers = appDatabase.userDao().getTotalUsers();

List<User> users = appDatabase.userDao().getAllUsers();

runOnUiThread(() -> {

totalUsersTextView.setText("Всего пользователей: " + totalUsers);

totalPurchasesTextView.setVisibility(View.GONE);

totalRevenueTextView.setVisibility(View.GONE);

getSupportActionBar().setTitle("Пользователи");

userAdapter.updateUsers(users);

});

} catch (Exception e) {

Log.e("MetricsActivity", "Error loading users metrics: " + e.getMessage());

runOnUiThread(() -> {

Toast.makeText(this, "Ошибка загрузки данных о пользователях", Toast.LENGTH\_SHORT).show();

});

}

}).start();

}

private void loadUsers() {

new Thread(() -> {

try {

List<User> users = appDatabase.userDao().getAllUsers();

runOnUiThread(() -> {

userAdapter.updateUsers(users);

});

} catch (Exception e) {

Log.e("MetricsActivity", "Error loading users: " + e.getMessage(), e);

runOnUiThread(() -> {

Toast.makeText(this, "Ошибка при загрузке пользователей", Toast.LENGTH\_SHORT).show();

});

}

}).start();

}

private void loadOrdersMetrics() {

new Thread(() -> {

try {

int totalOrders = appDatabase.purchaseDao().getConfirmedPurchasesCount();

List<Purchase> purchases = appDatabase.purchaseDao().getAllPurchases();

// Group purchases by user email

Map<String, Integer> userOrderCounts = new HashMap<>();

for (Purchase purchase : purchases) {

if ("Подтвержден".equals(purchase.getStatus())) {

String userEmail = purchase.getUserEmail();

userOrderCounts.put(userEmail, userOrderCounts.getOrDefault(userEmail, 0) + 1);

}

}

runOnUiThread(() -> {

TextView totalOrdersTextView = findViewById(R.id.totalOrdersTextView);

TextView userOrdersTextView = findViewById(R.id.userOrdersTextView);

totalOrdersTextView.setText("Всего подтвержденных заказов: " + totalOrders);

StringBuilder userStats = new StringBuilder("Заказы по пользователям:\n");

for (Map.Entry<String, Integer> entry : userOrderCounts.entrySet()) {

userStats.append(entry.getKey()).append(": ").append(entry.getValue()).append(" заказов\n");

}

userOrdersTextView.setText(userStats.toString());

});

} catch (Exception e) {

Log.e("MetricsActivity", "Error loading order metrics", e);

runOnUiThread(() -> {

Toast.makeText(MetricsActivity.this,

"Ошибка при загрузке статистики заказов",

Toast.LENGTH\_SHORT).show();

});

}

}).start();

}

private void loadRevenueMetrics() {

new Thread(() -> {

try {

double totalRevenue = appDatabase.purchaseDao().getTotalRevenue();

List<Purchase> purchases = appDatabase.purchaseDao().getAllPurchases();

runOnUiThread(() -> {

totalRevenueTextView.setText(String.format("Общая выручка: %.2f ₽", totalRevenue));

updateRevenueChart(purchases);

});

} catch (Exception e) {

Log.e("MetricsActivity", "Error loading revenue metrics: " + e.getMessage());

runOnUiThread(() -> {

Toast.makeText(this, "Ошибка загрузки данных о выручке", Toast.LENGTH\_SHORT).show();

});

}

}).start();

}

private void updateRevenueChart(List<Purchase> purchases) {

// Группируем покупки по продуктам

Map<String, Double> productRevenue = new HashMap<>();

for (Purchase purchase : purchases) {

if ("Подтвержден".equals(purchase.getStatus())) {

String productName = purchase.getProductName();

double price = purchase.getPrice();

productRevenue.put(productName, productRevenue.getOrDefault(productName, 0.0) + price);

}

}

// Создаем данные для диаграммы

List<PieEntry> entries = new ArrayList<>();

for (Map.Entry<String, Double> entry : productRevenue.entrySet()) {

entries.add(new PieEntry(entry.getValue().floatValue(), entry.getKey()));

}

// Настраиваем внешний вид диаграммы

PieDataSet dataSet = new PieDataSet(entries, "Выручка по продуктам");

dataSet.setColors(ColorTemplate.MATERIAL\_COLORS);

dataSet.setValueTextSize(12f);

dataSet.setValueTextColor(Color.WHITE);

PieData data = new PieData(dataSet);

data.setValueFormatter(new PercentFormatter(revenueChart));

// Настраиваем диаграмму

revenueChart.setData(data);

revenueChart.setDescription(null);

revenueChart.setHoleRadius(40f);

revenueChart.setTransparentCircleRadius(45f);

revenueChart.setDrawHoleEnabled(true);

revenueChart.setRotationEnabled(true);

revenueChart.setHighlightPerTapEnabled(true);

revenueChart.animateY(1000);

revenueChart.invalidate();

}

private void deleteUser(User user) {

new Thread(() -> {

try {

// Удаляем пользователя из базы данных

appDatabase.userDao().deleteUser(user);

// Обновляем UI в главном потоке

runOnUiThread(() -> {

// Обновляем список пользователей

loadUsers();

// Показываем сообщение об успешном удалении

Toast.makeText(this, "Пользователь успешно удален", Toast.LENGTH\_SHORT).show();

});

} catch (Exception e) {

Log.e("MetricsActivity", "Error deleting user: " + e.getMessage(), e);

runOnUiThread(() -> {

Toast.makeText(this, "Ошибка при удалении пользователя", Toast.LENGTH\_SHORT).show();

});

}

}).start();

}

private void deleteProduct(Product product) {

new Thread(() -> {

try {

appDatabase.productDao().delete(product);

runOnUiThread(() -> {

Toast.makeText(this, "Товар успешно удален", Toast.LENGTH\_SHORT).show();

loadProducts(); // Перезагружаем список товаров

});

} catch (Exception e) {

Log.e("MetricsActivity", "Error deleting product: " + e.getMessage(), e);

runOnUiThread(() -> {

Toast.makeText(this, "Ошибка при удалении товара", Toast.LENGTH\_SHORT).show();

});

}

}).start();

}

private void setupExportButton() {

exportRevenueFab.setOnClickListener(v -> {

new Thread(() -> {

try {

List<Purchase> purchases = appDatabase.purchaseDao().getAllPurchases();

String reportPath = reportGenerator.generateRevenueReport(purchases);

runOnUiThread(() -> {

if (reportPath != null) {

// Открываем PDF файл

File reportFile = new File(reportPath);

Uri reportUri = FileProvider.getUriForFile(

MetricsActivity.this,

getApplicationContext().getPackageName() + ".provider",

reportFile

);

Intent intent = new Intent(Intent.ACTION\_VIEW);

intent.setDataAndType(reportUri, "application/pdf");

intent.setFlags(Intent.FLAG\_GRANT\_READ\_URI\_PERMISSION);

try {

startActivity(intent);

} catch (ActivityNotFoundException e) {

Toast.makeText(MetricsActivity.this,

"Установите приложение для просмотра PDF файлов",

Toast.LENGTH\_LONG).show();

}

} else {

Toast.makeText(MetricsActivity.this,

"Ошибка при создании отчета",

Toast.LENGTH\_LONG).show();

}

});

} catch (Exception e) {

e.printStackTrace();

runOnUiThread(() -> {

Toast.makeText(MetricsActivity.this,

"Ошибка при создании отчета: " + e.getMessage(),

Toast.LENGTH\_LONG).show();

});

}

}).start();

});

}

@Override

protected void onResume() {

super.onResume();

// Определяем, какой раздел активен, и загружаем соответствующие данные

int selectedItemId = bottomNavigationView.getSelectedItemId();

if (selectedItemId == R.id.navigation\_users) {

loadUsersMetrics();

} else if (selectedItemId == R.id.navigation\_orders) {

loadOrdersMetrics();

} else if (selectedItemId == R.id.navigation\_revenue) {

loadRevenueMetrics();

} else if (selectedItemId == R.id.navigation\_products) {

showProductsList();

}

}

}

~21) Order.java;~

package com.example.bolotnyiysen;

public class Order {

private String orderNumber;

private String date;

private String status;

private double totalAmount;

public Order(String orderNumber, String date, String status, double totalAmount) {

this.orderNumber = orderNumber;

this.date = date;

this.status = status;

this.totalAmount = totalAmount;

}

public String getOrderNumber() {

return orderNumber;

}

public String getDate() {

return date;

}

public String getStatus() {

return status;

}

public double getTotalAmount() {

return totalAmount;

}

}

~22) OrderConfirmationActivity.java;~

package com.example.bolotnyiysen;

import android.os.Bundle;

import android.util.Log;

import android.widget.Button;

import android.widget.RadioButton;

import android.widget.RadioGroup;

import android.widget.TextView;

import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import androidx.recyclerview.widget.LinearLayoutManager;

import androidx.recyclerview.widget.RecyclerView;

import java.util.List;

public class OrderConfirmationActivity extends AppCompatActivity {

private RecyclerView recyclerView;

private CartAdapter adapter;

private TextView totalPriceTextView;

private RadioGroup paymentMethodGroup;

private Button confirmOrderButton;

private CartManager cartManager;

private String userEmail;

private String paymentMethod;

private String productsList;

private double totalPrice;

private AppDatabase appDatabase;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_order\_confirmation);

// Инициализация базы данных

appDatabase = AppDatabase.getInstance(this);

// Получаем данные из Intent

userEmail = getIntent().getStringExtra("user\_email");

paymentMethod = getIntent().getStringExtra("payment\_method");

productsList = getIntent().getStringExtra("products");

totalPrice = getIntent().getDoubleExtra("total\_price", 0.0);

if (userEmail == null || paymentMethod == null || productsList == null) {

Toast.makeText(this, "Ошибка: неверные данные заказа", Toast.LENGTH\_SHORT).show();

finish();

return;

}

// Инициализация CartManager

cartManager = CartManager.getInstance(this);

cartManager.setCurrentUser(userEmail);

recyclerView = findViewById(R.id.cartRecyclerView);

totalPriceTextView = findViewById(R.id.totalPriceTextView);

paymentMethodGroup = findViewById(R.id.paymentMethodGroup);

confirmOrderButton = findViewById(R.id.confirmOrderButton);

// Устанавливаем способ оплаты

int radioId = -1;

if (paymentMethod.equals("Наличными при получении")) {

radioId = R.id.radioCash;

} else if (paymentMethod.equals("Банковской картой")) {

radioId = R.id.radioCard;

} else if (paymentMethod.equals("Онлайн-оплата")) {

radioId = R.id.radioOnline;

}

if (radioId != -1) {

paymentMethodGroup.check(radioId);

}

// Отображаем список товаров

TextView productsTextView = findViewById(R.id.productsTextView);

productsTextView.setText(productsList);

// Устанавливаем общую сумму

totalPriceTextView.setText(String.format("Итого: %.2f ₽", totalPrice));

confirmOrderButton.setOnClickListener(v -> {

// Сохраняем заказ в базу данных

saveOrderToDatabase();

// Очищаем корзину

cartManager.clearCart();

// Показываем уведомление о подтверждении заказа

String message = String.format("Ваш заказ подтвержден!\nСпособ оплаты: %s\nСумма заказа: %.2f ₽",

paymentMethod, totalPrice);

Toast.makeText(this, message, Toast.LENGTH\_LONG).show();

// Закрываем активность

finish();

});

}

private void saveOrderToDatabase() {

new Thread(() -> {

try {

// Получаем текущую дату

String currentDate = new java.text.SimpleDateFormat("dd.MM.yyyy HH:mm", java.util.Locale.getDefault())

.format(new java.util.Date());

Log.d("OrderConfirmation", "Starting to save order to database");

Log.d("OrderConfirmation", "User email: " + userEmail);

Log.d("OrderConfirmation", "Payment method: " + paymentMethod);

Log.d("OrderConfirmation", "Products list: " + productsList);

// Разбиваем строку с товарами на отдельные товары

String[] products = productsList.split("\n");

Log.d("OrderConfirmation", "Number of products to save: " + products.length);

// Сохраняем каждый товар как отдельную покупку

for (String product : products) {

if (!product.trim().isEmpty()) {

try {

// Извлекаем название товара и цену из строки

String productName = product.substring(0, product.lastIndexOf(" ("));

String priceStr = product.substring(product.lastIndexOf("(") + 1, product.lastIndexOf(" ₽)"));

double price = Double.parseDouble(priceStr);

Log.d("OrderConfirmation", "Saving product: " + productName + ", Price: " + price);

// Создаем и сохраняем покупку

Purchase purchase = new Purchase(userEmail, productName, paymentMethod, currentDate, price);

appDatabase.purchaseDao().insert(purchase);

Log.d("OrderConfirmation", "Successfully saved purchase for product: " + productName);

} catch (Exception e) {

Log.e("OrderConfirmation", "Error saving product: " + product, e);

}

}

}

// Проверяем, что заказы сохранились

int savedPurchases = appDatabase.purchaseDao().getTotalPurchases();

Log.d("OrderConfirmation", "Total purchases in database after saving: " + savedPurchases);

// Показываем сообщение об успехе в UI потоке

runOnUiThread(() -> {

Toast.makeText(this, "Заказ успешно сохранен", Toast.LENGTH\_SHORT).show();

});

} catch (Exception e) {

Log.e("OrderConfirmation", "Error in saveOrderToDatabase: " + e.getMessage(), e);

runOnUiThread(() -> {

Toast.makeText(this, "Ошибка при сохранении заказа", Toast.LENGTH\_SHORT).show();

});

}

}).start();

}

}

~23) OrdersActivity.java;~

package com.example.bolotnyiysen;

import android.content.Intent;

import android.os.Bundle;

import android.view.MenuItem;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity;

import androidx.appcompat.widget.Toolbar;

import androidx.recyclerview.widget.LinearLayoutManager;

import androidx.recyclerview.widget.RecyclerView;

import com.google.android.material.bottomnavigation.BottomNavigationView;

import java.util.ArrayList;

import java.util.List;

public class OrdersActivity extends AppCompatActivity {

private BottomNavigationView bottomNavigationView;

private RecyclerView ordersRecyclerView;

private OrdersAdapter ordersAdapter;

private String userEmail;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_orders);

// Инициализация тулбара

Toolbar toolbar = findViewById(R.id.toolbar);

setSupportActionBar(toolbar);

getSupportActionBar().setDisplayHomeAsUpEnabled(true);

getSupportActionBar().setTitle("Мои заказы");

// Инициализация RecyclerView

ordersRecyclerView = findViewById(R.id.ordersRecyclerView);

ordersRecyclerView.setLayoutManager(new LinearLayoutManager(this));

// Создание тестовых данных (в реальном приложении данные будут загружаться из базы данных)

List<Order> orders = new ArrayList<>();

orders.add(new Order("Заказ #12345", "15.03.2024", "Доставлен", 2500));

orders.add(new Order("Заказ #12346", "16.03.2024", "В обработке", 1800));

orders.add(new Order("Заказ #12347", "17.03.2024", "Отменен", 3200));

// Установка адаптера

ordersAdapter = new OrdersAdapter(orders);

ordersRecyclerView.setAdapter(ordersAdapter);

bottomNavigationView = findViewById(R.id.bottomNavigationView);

setupBottomNavigation();

// Устанавливаем выбранный пункт меню

bottomNavigationView.setSelectedItemId(R.id.navigation\_profile);

// Получаем userEmail

Intent intent = getIntent();

if (intent != null) {

userEmail = intent.getStringExtra("user\_email");

}

}

private void setupBottomNavigation() {

bottomNavigationView.setOnNavigationItemSelectedListener(new BottomNavigationView.OnNavigationItemSelectedListener() {

@Override

public boolean onNavigationItemSelected(@NonNull MenuItem item) {

return onNavigationItemSelected(item);

}

});

}

private boolean onNavigationItemSelected(MenuItem item) {

int itemId = item.getItemId();

Intent intent;

if (itemId == R.id.navigation\_home) {

intent = new Intent(OrdersActivity.this, MainActivity.class);

} else if (itemId == R.id.navigation\_catalog) {

intent = new Intent(OrdersActivity.this, CatalogActivity.class);

} else if (itemId == R.id.navigation\_favorites) {

intent = new Intent(OrdersActivity.this, FavoritesActivity.class);

} else if (itemId == R.id.navigation\_cart) {

intent = new Intent(OrdersActivity.this, CartActivity.class);

} else if (itemId == R.id.navigation\_profile) {

intent = new Intent(OrdersActivity.this, ProfileActivity.class);

} else {

return false;

}

intent.putExtra("user\_email", userEmail);

startActivity(intent);

finish();

return true;

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

if (item.getItemId() == android.R.id.home) {

onBackPressed();

return true;

}

return super.onOptionsItemSelected(item);

}

}

~24) OrdersAdapter.java;~

package com.example.bolotnyiysen;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.TextView;

import androidx.annotation.NonNull;

import androidx.recyclerview.widget.RecyclerView;

import java.util.List;

public class OrdersAdapter extends RecyclerView.Adapter<OrdersAdapter.OrderViewHolder> {

private List<Order> orders;

public OrdersAdapter(List<Order> orders) {

this.orders = orders;

}

@NonNull

@Override

public OrderViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {

View view = LayoutInflater.from(parent.getContext())

.inflate(R.layout.item\_order, parent, false);

return new OrderViewHolder(view);

}

@Override

public void onBindViewHolder(@NonNull OrderViewHolder holder, int position) {

Order order = orders.get(position);

holder.orderNumberText.setText(order.getOrderNumber());

holder.dateText.setText(order.getDate());

holder.statusText.setText(order.getStatus());

holder.totalAmountText.setText(String.format("%,.2f ₽", order.getTotalAmount()));

}

@Override

public int getItemCount() {

return orders.size();

}

static class OrderViewHolder extends RecyclerView.ViewHolder {

TextView orderNumberText;

TextView dateText;

TextView statusText;

TextView totalAmountText;

OrderViewHolder(View itemView) {

super(itemView);

orderNumberText = itemView.findViewById(R.id.orderNumberText);

dateText = itemView.findViewById(R.id.dateText);

statusText = itemView.findViewById(R.id.statusText);

totalAmountText = itemView.findViewById(R.id.totalAmountText);

}

}

}

~25) ProductAdapter.java;~

package com.example.bolotnyiysen;

import android.content.Context;

import android.content.Intent;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.ImageButton;

import android.widget.ImageView;

import android.widget.TextView;

import android.widget.Toast;

import androidx.annotation.NonNull;

import androidx.recyclerview.widget.RecyclerView;

import com.bumptech.glide.Glide;

import java.util.ArrayList;

import java.util.List;

public class ProductAdapter extends RecyclerView.Adapter<ProductAdapter.ProductViewHolder> {

private List<Product> products = new ArrayList<>();

private Context context;

private boolean isCartView;

private boolean isFavoritesView;

private CartManager cartManager;

private OnProductClickListener listener;

private OnProductDeleteListener deleteListener;

private boolean isAdmin = false;

private ImageButton editButton;

private ImageButton deleteButton;

public interface OnProductClickListener {

void onProductClick(Product product);

}

public interface OnProductDeleteListener {

void onProductDelete(Product product);

}

public void setOnProductClickListener(OnProductClickListener listener) {

this.listener = listener;

}

public void setOnProductDeleteListener(OnProductDeleteListener listener) {

this.deleteListener = listener;

}

public ProductAdapter(List<Product> products, Context context) {

this.products = products;

this.context = context;

this.isCartView = false;

this.isFavoritesView = false;

this.cartManager = CartManager.getInstance(context);

}

public void setCartView(boolean isCartView) {

this.isCartView = isCartView;

}

public void setFavoritesView(boolean isFavoritesView) {

this.isFavoritesView = isFavoritesView;

}

public void updateProducts(List<Product> newProducts) {

this.products.clear();

this.products.addAll(newProducts);

notifyDataSetChanged();

}

public void setAdmin(boolean isAdmin) {

this.isAdmin = isAdmin;

notifyDataSetChanged();

}

@NonNull

@Override

public ProductViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {

View itemView = LayoutInflater.from(parent.getContext())

.inflate(R.layout.item\_product, parent, false);

return new ProductViewHolder(itemView);

}

@Override

public void onBindViewHolder(@NonNull ProductViewHolder holder, int position) {

Product product = products.get(position);

if (product.isHeader()) {

holder.productNameTextView.setText(product.getName());

holder.productCodeTextView.setVisibility(View.GONE);

holder.productPriceTextView.setVisibility(View.GONE);

holder.productImageView.setVisibility(View.GONE);

holder.addToCartButton.setVisibility(View.GONE);

holder.addToFavoritesButton.setVisibility(View.GONE);

holder.deleteProductButton.setVisibility(View.GONE);

holder.editButton.setVisibility(View.GONE);

} else {

holder.productNameTextView.setText(product.getName());

holder.productCodeTextView.setText("Код: " + product.getCode());

holder.productPriceTextView.setText(String.format("%.2f ₽", product.getPrice()));

if (product.getPhotoPath() != null && !product.getPhotoPath().isEmpty()) {

Glide.with(context)

.load(product.getPhotoPath())

.into(holder.productImageView);

} else {

holder.productImageView.setImageResource(product.getImageResourceId());

}

holder.addToCartButton.setVisibility(isCartView ? View.GONE : View.VISIBLE);

holder.addToFavoritesButton.setVisibility(isFavoritesView ? View.GONE : View.VISIBLE);

holder.deleteProductButton.setVisibility(isAdmin ? View.VISIBLE : View.GONE);

holder.editButton.setVisibility(isAdmin ? View.VISIBLE : View.GONE);

FavoritesManager favoritesManager = FavoritesManager.getInstance();

holder.addToFavoritesButton.setImageResource(

favoritesManager.isFavorite(product) ? R.drawable.ic\_favorite : R.drawable.ic\_favorite\_border

);

holder.deleteProductButton.setOnClickListener(v -> {

if (deleteListener != null) {

deleteListener.onProductDelete(product);

}

});

holder.addToCartButton.setOnClickListener(v -> {

cartManager.addToCart(product);

Toast.makeText(context, "Товар добавлен в корзину", Toast.LENGTH\_SHORT).show();

});

holder.addToFavoritesButton.setOnClickListener(v -> {

if (favoritesManager.isFavorite(product)) {

favoritesManager.removeFromFavorites(product);

holder.addToFavoritesButton.setImageResource(R.drawable.ic\_favorite\_border);

Toast.makeText(context, "Товар удален из избранного", Toast.LENGTH\_SHORT).show();

} else {

favoritesManager.addToFavorites(product);

holder.addToFavoritesButton.setImageResource(R.drawable.ic\_favorite);

Toast.makeText(context, "Товар добавлен в избранное", Toast.LENGTH\_SHORT).show();

}

});

holder.editButton.setOnClickListener(v -> {

Intent intent = new Intent(v.getContext(), EditProductActivity.class);

intent.putExtra("product", product);

v.getContext().startActivity(intent);

});

}

holder.itemView.setOnClickListener(v -> {

if (listener != null && !product.isHeader()) {

Intent intent = new Intent(v.getContext(), ProductDetailActivity.class);

intent.putExtra("product", product);

v.getContext().startActivity(intent);

}

});

}

@Override

public int getItemCount() {

return products.size();

}

class ProductViewHolder extends RecyclerView.ViewHolder {

ImageView productImageView;

TextView productNameTextView;

TextView productCodeTextView;

TextView productPriceTextView;

ImageButton addToCartButton;

ImageButton addToFavoritesButton;

ImageButton deleteProductButton;

ImageButton editButton;

ProductViewHolder(View itemView) {

super(itemView);

productImageView = itemView.findViewById(R.id.productImageView);

productNameTextView = itemView.findViewById(R.id.productNameTextView);

productCodeTextView = itemView.findViewById(R.id.productCodeTextView);

productPriceTextView = itemView.findViewById(R.id.productPriceTextView);

addToCartButton = itemView.findViewById(R.id.addToCartButton);

addToFavoritesButton = itemView.findViewById(R.id.addToFavoritesButton);

deleteProductButton = itemView.findViewById(R.id.deleteProductButton);

editButton = itemView.findViewById(R.id.editButton);

}

}

}

~26) ProductDao.java;~

package com.example.bolotnyiysen;

import androidx.room.Dao;

import androidx.room.Delete;

import androidx.room.Insert;

import androidx.room.Query;

import androidx.room.Update;

import java.util.List;

@Dao

public interface ProductDao {

@Insert

void insert(Product product);

@Update

void update(Product product);

@Delete

void delete(Product product);

@Query("SELECT \* FROM products")

List<Product> getAllProducts();

@Query("SELECT \* FROM products WHERE id = :id")

Product getProductById(int id);

@Query("SELECT \* FROM products WHERE code = :code")

Product getProductByCode(String code);

}

~27) ProductDetailActivity.java;~

package com.example.bolotnyiysen;

import android.content.Intent;

import android.os.Bundle;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;

import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

import androidx.appcompat.widget.Toolbar;

import androidx.viewpager2.widget.ViewPager2;

import com.bumptech.glide.Glide;

import com.google.android.material.tabs.TabLayout;

import com.google.android.material.tabs.TabLayoutMediator;

import java.util.ArrayList;

import java.util.List;

public class ProductDetailActivity extends AppCompatActivity {

private ViewPager2 imageViewPager;

private TabLayout imageTabLayout;

private TextView productNameTextView;

private TextView productCodeTextView;

private TextView manufacturerTextView;

private TextView seriesTextView;

private TextView typeTextView;

private TextView conditionTextView;

private TextView bodyShapeTextView;

private TextView orientationTextView;

private TextView stringsCountTextView;

private TextView fretsCountTextView;

private TextView scaleLengthTextView;

private TextView productPriceTextView;

private Product product;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_product\_detail);

// Инициализация тулбара

Toolbar toolbar = findViewById(R.id.toolbar);

setSupportActionBar(toolbar);

getSupportActionBar().setDisplayHomeAsUpEnabled(true);

// Получение данных о товаре

product = (Product) getIntent().getSerializableExtra("product");

List<Integer> imageResources = getIntent().getIntegerArrayListExtra("imageResources");

List<String> imagePaths = getIntent().getStringArrayListExtra("imagePaths");

// Инициализация ViewPager2 и TabLayout

imageViewPager = findViewById(R.id.imageViewPager);

imageTabLayout = findViewById(R.id.imageTabLayout);

// Настройка адаптера для ViewPager2

if ((imageResources != null && !imageResources.isEmpty()) ||

(imagePaths != null && !imagePaths.isEmpty())) {

ImagePagerAdapter imagePagerAdapter = new ImagePagerAdapter(this, imageResources, imagePaths);

imageViewPager.setAdapter(imagePagerAdapter);

// Связывание ViewPager2 с TabLayout

new TabLayoutMediator(imageTabLayout, imageViewPager,

(tab, position) -> {}).attach();

// Показываем TabLayout только если есть больше одного изображения

int totalImages = Math.max(

imageResources != null ? imageResources.size() : 0,

imagePaths != null ? imagePaths.size() : 0

);

imageTabLayout.setVisibility(totalImages > 1 ? View.VISIBLE : View.GONE);

}

// Инициализация TextView

productNameTextView = findViewById(R.id.productNameTextView);

productCodeTextView = findViewById(R.id.productCodeTextView);

manufacturerTextView = findViewById(R.id.manufacturerTextView);

seriesTextView = findViewById(R.id.seriesTextView);

typeTextView = findViewById(R.id.typeTextView);

conditionTextView = findViewById(R.id.conditionTextView);

bodyShapeTextView = findViewById(R.id.bodyShapeTextView);

orientationTextView = findViewById(R.id.orientationTextView);

stringsCountTextView = findViewById(R.id.stringsCountTextView);

fretsCountTextView = findViewById(R.id.fretsCountTextView);

scaleLengthTextView = findViewById(R.id.scaleLengthTextView);

productPriceTextView = findViewById(R.id.productPriceTextView);

// Заполнение данных о товаре

if (product != null) {

productNameTextView.setText(product.getName());

productCodeTextView.setText("Код товара: " + product.getCode());

manufacturerTextView.setText("Производитель: " + product.getManufacturer());

seriesTextView.setText("Серия: " + product.getSeries());

typeTextView.setText("Тип: " + product.getType());

conditionTextView.setText("Состояние: " + product.getCondition());

bodyShapeTextView.setText("Форма корпуса: " + product.getBodyShape());

orientationTextView.setText("Ориентация: " + product.getOrientation());

stringsCountTextView.setText("Количество струн: " + product.getStringsCount());

fretsCountTextView.setText("Количество ладов: " + product.getFretsCount());

scaleLengthTextView.setText("Мензура: " + product.getScaleLength() + " дюймов");

productPriceTextView.setText(String.format("%.2f ₽", product.getPrice()));

}

}

@Override

public boolean onSupportNavigateUp() {

onBackPressed();

return true;

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

getMenuInflater().inflate(R.menu.menu\_product\_detail, menu);

return true;

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

if (item.getItemId() == R.id.action\_edit) {

Intent intent = new Intent(this, EditProductActivity.class);

intent.putExtra("product", product);

startActivity(intent);

return true;

}

return super.onOptionsItemSelected(item);

}

}

~28) ProfileActivity.java;~

package com.example.bolotnyiysen;

import android.content.Intent;

import android.content.SharedPreferences;

import android.os.Bundle;

import android.text.Editable;

import android.text.TextWatcher;

import android.view.MenuItem;

import android.widget.ArrayAdapter;

import android.widget.AutoCompleteTextView;

import android.widget.Button;

import android.widget.TextView;

import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import androidx.appcompat.widget.Toolbar;

import androidx.recyclerview.widget.LinearLayoutManager;

import androidx.recyclerview.widget.RecyclerView;

import com.google.android.material.bottomnavigation.BottomNavigationView;

import com.google.android.material.textfield.TextInputEditText;

import com.google.android.material.textfield.TextInputLayout;

import com.google.gson.Gson;

import java.util.ArrayList;

import java.util.List;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

import java.util.regex.Pattern;

public class ProfileActivity extends AppCompatActivity {

private TextView nameTextView;

private TextView emailTextView;

private TextView phoneTextView;

private TextView cityTextView;

private TextView genderTextView;

private RecyclerView transactionHistoryRecyclerView;

private PurchaseAdapter purchaseAdapter;

private AppDatabase appDatabase;

private BottomNavigationView bottomNavigationView;

private SharedPreferences sharedPreferences;

private static final String PREFS\_NAME = "UserProfile";

private String userEmail;

private ExecutorService executorService;

private Button logoutButton;

private Button saveProfileButton;

// Поля для редактирования

private TextInputEditText editNameText;

private TextInputEditText editPhoneText;

private TextInputEditText editCityText;

private AutoCompleteTextView editGenderText;

// Паттерны для валидации

private static final Pattern FIO\_PATTERN = Pattern.compile("^[А-Яа-яЁё\\s-]{3,}$");

private static final Pattern PHONE\_PATTERN = Pattern.compile("^\\+7\\d{10}$");

private static final Pattern CITY\_PATTERN = Pattern.compile("^[А-Яа-яЁё\\s-]{2,}$");

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_profile);

// Получаем email из Intent

userEmail = getIntent().getStringExtra("user\_email");

if (userEmail == null) {

Toast.makeText(this, "Ошибка: email не найден", Toast.LENGTH\_SHORT).show();

finish();

return;

}

// Инициализация базы данных

appDatabase = AppDatabase.getInstance(this);

executorService = Executors.newSingleThreadExecutor();

// Инициализация компонентов

nameTextView = findViewById(R.id.nameTextView);

emailTextView = findViewById(R.id.emailTextView);

phoneTextView = findViewById(R.id.phoneTextView);

cityTextView = findViewById(R.id.cityTextView);

genderTextView = findViewById(R.id.genderTextView);

transactionHistoryRecyclerView = findViewById(R.id.transactionHistoryRecyclerView);

bottomNavigationView = findViewById(R.id.bottom\_navigation);

logoutButton = findViewById(R.id.logoutButton);

saveProfileButton = findViewById(R.id.saveProfileButton);

// Инициализация полей редактирования

editNameText = findViewById(R.id.editNameText);

editPhoneText = findViewById(R.id.editPhoneText);

editCityText = findViewById(R.id.editCityText);

editGenderText = findViewById(R.id.editGenderText);

// Настройка тулбара

Toolbar toolbar = findViewById(R.id.toolbar);

setSupportActionBar(toolbar);

getSupportActionBar().setDisplayHomeAsUpEnabled(true);

getSupportActionBar().setTitle("Профиль");

// Настройка RecyclerView для истории покупок

transactionHistoryRecyclerView.setLayoutManager(new LinearLayoutManager(this));

purchaseAdapter = new PurchaseAdapter(new ArrayList<>());

transactionHistoryRecyclerView.setAdapter(purchaseAdapter);

// Инициализация SharedPreferences

sharedPreferences = getSharedPreferences(PREFS\_NAME, MODE\_PRIVATE);

// Настройка выпадающего списка для пола

String[] genders = {"Мужской", "Женский"};

ArrayAdapter<String> adapter = new ArrayAdapter<>(this, R.layout.dropdown\_item, genders);

editGenderText.setAdapter(adapter);

editGenderText.setOnItemClickListener((parent, view, position, id) -> {

String selectedGender = (String) parent.getItemAtPosition(position);

editGenderText.setText(selectedGender, false);

});

// Устанавливаем значение только если оно уже есть в базе данных

String currentGender = editGenderText.getText().toString();

if (!currentGender.isEmpty()) {

editGenderText.setText(currentGender, false);

}

// Загрузка данных пользователя

loadUserData(userEmail);

loadPurchaseHistory(userEmail);

// Настройка валидации полей

setupFieldValidation();

// Настройка кнопки сохранения

saveProfileButton.setOnClickListener(v -> saveProfile());

// Настройка кнопки выхода

logoutButton.setOnClickListener(v -> {

// Очищаем данные пользователя

SharedPreferences.Editor editor = sharedPreferences.edit();

editor.remove(userEmail);

editor.apply();

// Переходим на экран входа

Intent intent = new Intent(ProfileActivity.this, LoginActivity.class);

intent.setFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK | Intent.FLAG\_ACTIVITY\_CLEAR\_TASK);

startActivity(intent);

finish();

});

// Настройка нижней навигации

setupBottomNavigation();

}

private void setupFieldValidation() {

// Валидация имени

editNameText.addTextChangedListener(new TextWatcher() {

@Override

public void beforeTextChanged(CharSequence s, int start, int count, int after) {}

@Override

public void onTextChanged(CharSequence s, int start, int before, int count) {}

@Override

public void afterTextChanged(Editable s) {

String name = s.toString().trim();

TextInputLayout layout = findViewById(R.id.nameLayout);

if (!FIO\_PATTERN.matcher(name).matches()) {

layout.setError("Введите корректное имя");

} else {

layout.setError(null);

}

}

});

// Валидация телефона

editPhoneText.addTextChangedListener(new TextWatcher() {

@Override

public void beforeTextChanged(CharSequence s, int start, int count, int after) {}

@Override

public void onTextChanged(CharSequence s, int start, int before, int count) {

String phone = s.toString().replaceAll("[^0-9+]", "");

if (!phone.startsWith("+7") && !phone.isEmpty()) {

phone = "+7" + phone;

}

if (!phone.equals(s.toString())) {

editPhoneText.setText(phone);

editPhoneText.setSelection(phone.length());

}

}

@Override

public void afterTextChanged(Editable s) {

String phone = s.toString();

TextInputLayout layout = findViewById(R.id.phoneLayout);

if (!PHONE\_PATTERN.matcher(phone).matches()) {

layout.setError("Введите корректный номер телефона");

} else {

layout.setError(null);

}

}

});

// Валидация города

editCityText.addTextChangedListener(new TextWatcher() {

@Override

public void beforeTextChanged(CharSequence s, int start, int count, int after) {}

@Override

public void onTextChanged(CharSequence s, int start, int before, int count) {}

@Override

public void afterTextChanged(Editable s) {

String city = s.toString().trim();

TextInputLayout layout = findViewById(R.id.cityLayout);

if (!CITY\_PATTERN.matcher(city).matches()) {

layout.setError("Введите корректное название города");

} else {

layout.setError(null);

}

}

});

}

private void loadUserData(String userEmail) {

executorService.execute(() -> {

User user = appDatabase.userDao().getUserByEmail(userEmail);

if (user != null) {

runOnUiThread(() -> {

// Заполняем текстовые поля

nameTextView.setText(user.getFio());

emailTextView.setText("Email: " + user.getEmail());

phoneTextView.setText("Телефон: " + user.getPhone());

cityTextView.setText("Город: " + user.getCity());

genderTextView.setText("Пол: " + user.getGender());

// Заполняем поля редактирования

editNameText.setText(user.getFio());

editPhoneText.setText(user.getPhone());

editCityText.setText(user.getCity());

editGenderText.setText(user.getGender());

});

} else {

// Если пользователь не найден в базе, пробуем загрузить из SharedPreferences

String profileJson = sharedPreferences.getString(userEmail, null);

if (profileJson != null) {

Gson gson = new Gson();

UserProfile profile = gson.fromJson(profileJson, UserProfile.class);

runOnUiThread(() -> {

// Заполняем текстовые поля

nameTextView.setText(profile.getFio());

emailTextView.setText("Email: " + profile.getEmail());

phoneTextView.setText("Телефон: " + profile.getPhone());

cityTextView.setText("Город: " + profile.getCity());

genderTextView.setText("Пол: " + profile.getGender());

// Заполняем поля редактирования

editNameText.setText(profile.getFio());

editPhoneText.setText(profile.getPhone());

editCityText.setText(profile.getCity());

editGenderText.setText(profile.getGender());

});

}

}

});

}

private void saveProfile() {

String name = editNameText.getText().toString().trim();

String phone = editPhoneText.getText().toString().trim();

String city = editCityText.getText().toString().trim();

String gender = editGenderText.getText().toString().trim();

// Проверка валидации

if (!FIO\_PATTERN.matcher(name).matches()) {

Toast.makeText(this, "Пожалуйста, введите корректное имя", Toast.LENGTH\_SHORT).show();

return;

}

if (!PHONE\_PATTERN.matcher(phone).matches()) {

Toast.makeText(this, "Пожалуйста, введите корректный номер телефона", Toast.LENGTH\_SHORT).show();

return;

}

if (!CITY\_PATTERN.matcher(city).matches()) {

Toast.makeText(this, "Пожалуйста, введите корректное название города", Toast.LENGTH\_SHORT).show();

return;

}

if (gender.isEmpty()) {

Toast.makeText(this, "Пожалуйста, выберите пол", Toast.LENGTH\_SHORT).show();

return;

}

executorService.execute(() -> {

try {

User user = appDatabase.userDao().getUserByEmail(userEmail);

if (user != null) {

user.setFio(name);

user.setPhone(phone);

user.setCity(city);

user.setGender(gender);

appDatabase.userDao().update(user);

// Также сохраняем в SharedPreferences для совместимости

UserProfile profile = new UserProfile(name, phone, city, gender, userEmail);

Gson gson = new Gson();

String profileJson = gson.toJson(profile);

sharedPreferences.edit().putString(userEmail, profileJson).apply();

runOnUiThread(() -> {

Toast.makeText(this, "Профиль успешно обновлен", Toast.LENGTH\_SHORT).show();

// Обновляем отображение данных

loadUserData(userEmail);

});

} else {

runOnUiThread(() -> {

Toast.makeText(this, "Ошибка: пользователь не найден", Toast.LENGTH\_SHORT).show();

});

}

} catch (Exception e) {

runOnUiThread(() -> {

Toast.makeText(this, "Ошибка при сохранении: " + e.getMessage(), Toast.LENGTH\_LONG).show();

});

}

});

}

private void loadPurchaseHistory(String userEmail) {

executorService.execute(() -> {

List<Purchase> purchases = appDatabase.purchaseDao().getPurchasesByUserEmail(userEmail);

runOnUiThread(() -> {

purchaseAdapter.updatePurchases(purchases);

});

});

}

private void setupBottomNavigation() {

bottomNavigationView.setOnItemSelectedListener(item -> {

int itemId = item.getItemId();

Intent intent;

if (itemId == R.id.navigation\_home) {

intent = new Intent(ProfileActivity.this, MainActivity.class);

} else if (itemId == R.id.navigation\_catalog) {

intent = new Intent(ProfileActivity.this, CatalogActivity.class);

} else if (itemId == R.id.navigation\_favorites) {

intent = new Intent(ProfileActivity.this, FavoritesActivity.class);

} else if (itemId == R.id.navigation\_cart) {

intent = new Intent(ProfileActivity.this, CartActivity.class);

} else if (itemId == R.id.navigation\_profile) {

return true;

} else {

return false;

}

intent.putExtra("user\_email", userEmail);

startActivity(intent);

finish();

return true;

});

bottomNavigationView.setSelectedItemId(R.id.navigation\_profile);

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

if (item.getItemId() == android.R.id.home) {

onBackPressed();

return true;

}

return super.onOptionsItemSelected(item);

}

@Override

protected void onDestroy() {

super.onDestroy();

executorService.shutdown();

}

}

~29) PurchaseAdapter.java;~

package com.example.bolotnyiysen;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.TextView;

import androidx.annotation.NonNull;

import androidx.recyclerview.widget.RecyclerView;

import java.util.ArrayList;

import java.util.List;

public class PurchaseAdapter extends RecyclerView.Adapter<PurchaseAdapter.PurchaseViewHolder> {

private List<Purchase> purchases;

public PurchaseAdapter(List<Purchase> purchases) {

this.purchases = purchases != null ? purchases : new ArrayList<>();

}

@NonNull

@Override

public PurchaseViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {

View view = LayoutInflater.from(parent.getContext())

.inflate(R.layout.item\_purchase, parent, false);

return new PurchaseViewHolder(view);

}

@Override

public void onBindViewHolder(@NonNull PurchaseViewHolder holder, int position) {

Purchase purchase = purchases.get(position);

holder.productNameTextView.setText(purchase.getProductName());

holder.paymentTypeTextView.setText("Способ оплаты: " + purchase.getPaymentType());

holder.dateTextView.setText("Дата: " + purchase.getDate());

holder.priceTextView.setText(String.format("Цена: %.2f ₽", purchase.getPrice()));

}

@Override

public int getItemCount() {

return purchases.size();

}

public void updatePurchases(List<Purchase> newPurchases) {

this.purchases = newPurchases;

notifyDataSetChanged();

}

static class PurchaseViewHolder extends RecyclerView.ViewHolder {

TextView productNameTextView;

TextView paymentTypeTextView;

TextView dateTextView;

TextView priceTextView;

PurchaseViewHolder(View itemView) {

super(itemView);

productNameTextView = itemView.findViewById(R.id.productNameTextView);

paymentTypeTextView = itemView.findViewById(R.id.paymentTypeTextView);

dateTextView = itemView.findViewById(R.id.dateTextView);

priceTextView = itemView.findViewById(R.id.priceTextView);

}

}

}

~30) PurchaseDao.java;~

package com.example.bolotnyiysen;

import androidx.room.Dao;

import androidx.room.Insert;

import androidx.room.Query;

import java.util.List;

@Dao

public interface PurchaseDao {

@Insert

void insert(Purchase purchase);

@Query("SELECT \* FROM purchases WHERE userEmail = :userEmail ORDER BY date DESC")

List<Purchase> getPurchasesByUserEmail(String userEmail);

@Query("SELECT \* FROM purchases WHERE userEmail = :userEmail AND status = :status ORDER BY date DESC")

List<Purchase> getPurchasesByUserEmailAndStatus(String userEmail, String status);

@Query("SELECT COUNT(\*) FROM purchases")

int getTotalPurchases();

@Query("SELECT SUM(price) FROM purchases WHERE status = 'Подтвержден'")

double getTotalRevenue();

@Query("SELECT COUNT(\*) FROM purchases WHERE status = 'Подтвержден'")

int getConfirmedPurchasesCount();

@Query("SELECT \* FROM purchases")

List<Purchase> getAllPurchases();

@Query("SELECT userEmail, COUNT(\*) as count FROM purchases WHERE status = 'Подтвержден' GROUP BY userEmail")

List<UserPurchaseCount> getPurchaseCountsByUser();

}

~31) ReceiptManager.java;~

package com.example.bolotnyiysen;

import android.content.Context;

import android.os.Build;

import android.os.Environment;

import android.util.Log;

import com.itextpdf.kernel.pdf.PdfDocument;

import com.itextpdf.kernel.pdf.PdfWriter;

import com.itextpdf.layout.Document;

import com.itextpdf.layout.element.Paragraph;

import com.itextpdf.layout.element.Table;

import com.itextpdf.layout.properties.TextAlignment;

import com.itextpdf.layout.properties.UnitValue;

import java.io.File;

import java.io.FileOutputStream;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.Locale;

import java.util.UUID;

public class ReceiptManager {

private static ReceiptManager instance;

private Context context;

private static final String RECEIPTS\_FOLDER = "BolotnyiYsen\_Receipts";

private ReceiptManager(Context context) {

this.context = context.getApplicationContext();

}

public static ReceiptManager getInstance(Context context) {

if (instance == null) {

instance = new ReceiptManager(context);

}

return instance;

}

public String generateReceipt(Receipt receipt) {

try {

// Создаем папку для чеков в публичной директории Documents

File receiptsDir;

if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.Q) {

// Для Android 10 и выше используем публичную директорию Documents

receiptsDir = new File(Environment.getExternalStoragePublicDirectory(

Environment.DIRECTORY\_DOCUMENTS), RECEIPTS\_FOLDER);

} else {

// Для более старых версий также используем публичную директорию Documents

receiptsDir = new File(Environment.getExternalStoragePublicDirectory(

Environment.DIRECTORY\_DOCUMENTS), RECEIPTS\_FOLDER);

}

if (!receiptsDir.exists()) {

boolean created = receiptsDir.mkdirs();

if (!created) {

Log.e("ReceiptManager", "Failed to create receipts directory");

return null;

}

}

// Генерируем имя файла с датой и номером чека

SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd\_HH-mm", Locale.getDefault());

String dateStr = dateFormat.format(new Date());

String fileName = "Чек\_" + dateStr + "\_" + receipt.getReceiptNumber() + ".pdf";

File receiptFile = new File(receiptsDir, fileName);

// Создаем PDF документ

PdfWriter writer = new PdfWriter(new FileOutputStream(receiptFile));

PdfDocument pdf = new PdfDocument(writer);

Document document = new Document(pdf);

// Добавляем заголовок

Paragraph header = new Paragraph(receipt.getStoreName())

.setTextAlignment(TextAlignment.CENTER)

.setFontSize(20);

document.add(header);

// Добавляем информацию о чеке

document.add(new Paragraph("Чек №: " + receipt.getReceiptNumber()));

document.add(new Paragraph("Дата: " + formatDate(receipt.getDate())));

document.add(new Paragraph("Email: " + receipt.getCustomerEmail()));

document.add(new Paragraph("\n"));

// Создаем таблицу товаров

Table table = new Table(UnitValue.createPercentArray(4)).useAllAvailableWidth();

table.addHeaderCell("Товар");

table.addHeaderCell("Кол-во");

table.addHeaderCell("Цена");

table.addHeaderCell("Сумма");

// Добавляем товары в таблицу

for (Product product : receipt.getProducts()) {

table.addCell(product.getName());

table.addCell(String.valueOf(product.getQuantity()));

table.addCell(String.format("%.2f ₽", product.getPrice()));

table.addCell(String.format("%.2f ₽", product.getPrice() \* product.getQuantity()));

}

document.add(table);

// Добавляем итоговую сумму

document.add(new Paragraph("\n"));

document.add(new Paragraph(String.format("Итого: %.2f ₽", receipt.getTotalAmount()))

.setTextAlignment(TextAlignment.RIGHT)

.setBold());

// Закрываем документ

document.close();

return receiptFile.getAbsolutePath();

} catch (Exception e) {

Log.e("ReceiptManager", "Error generating receipt", e);

return null;

}

}

private String formatDate(Date date) {

SimpleDateFormat sdf = new SimpleDateFormat("dd.MM.yyyy HH:mm:ss", Locale.getDefault());

return sdf.format(date);

}

public String generateReceiptNumber() {

return UUID.randomUUID().toString().substring(0, 8).toUpperCase();

}

}

~32) RegisterActivity.java;~

package com.example.bolotnyiysen;

import android.content.Intent;

import android.content.SharedPreferences;

import android.os.Bundle;

import android.text.Editable;

import android.text.TextWatcher;

import android.widget.Button;

import android.widget.EditText;

import android.widget.TextView;

import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import com.google.android.material.textfield.TextInputLayout;

import com.google.gson.Gson;

import java.util.regex.Pattern;

public class RegisterActivity extends AppCompatActivity {

private EditText emailEditText;

private EditText passwordEditText;

private EditText confirmPasswordEditText;

private EditText fioEditText;

private EditText phoneEditText;

private EditText cityEditText;

private EditText codeWordEditText;

private Button registerButton;

private TextView loginTextView;

private UserDao userDao;

private SharedPreferences sharedPreferences;

private static final String PREFS\_NAME = "UserProfile";

// Паттерны для валидации

private static final Pattern EMAIL\_PATTERN = Pattern.compile("^[A-Za-z0-9+\_.-]+@(.+)$");

private static final Pattern PASSWORD\_PATTERN = Pattern.compile("^(?=.\*[0-9])(?=.\*[a-z])(?=.\*[A-Z])(?=.\*[@#$%^&+=])(?=\\S+$).{8,}$");

private static final Pattern FIO\_PATTERN = Pattern.compile("^[А-Яа-яЁё\\s-]{3,}$");

private static final Pattern PHONE\_PATTERN = Pattern.compile("^\\+7\\d{10}$");

private static final Pattern CITY\_PATTERN = Pattern.compile("^[А-Яа-яЁё\\s-]{2,}$");

private static final Pattern CODE\_WORD\_PATTERN = Pattern.compile("^[А-Яа-яЁёA-Za-z0-9\\s-]{3,}$");

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_register);

// Initialize database

userDao = AppDatabase.getInstance(this).userDao();

// Initialize views

emailEditText = findViewById(R.id.emailEditText);

passwordEditText = findViewById(R.id.passwordEditText);

confirmPasswordEditText = findViewById(R.id.confirmPasswordEditText);

fioEditText = findViewById(R.id.fioEditText);

phoneEditText = findViewById(R.id.phoneEditText);

cityEditText = findViewById(R.id.cityEditText);

codeWordEditText = findViewById(R.id.codeWordEditText);

registerButton = findViewById(R.id.registerButton);

loginTextView = findViewById(R.id.loginTextView);

// Initialize SharedPreferences

sharedPreferences = getSharedPreferences(PREFS\_NAME, MODE\_PRIVATE);

// Add validation and masks to fields

setupFieldValidation();

// Set click listeners

registerButton.setOnClickListener(v -> registerUser());

loginTextView.setOnClickListener(v -> {

startActivity(new Intent(RegisterActivity.this, LoginActivity.class));

finish();

});

}

private void setupFieldValidation() {

// Validation for email

emailEditText.addTextChangedListener(new TextWatcher() {

@Override

public void beforeTextChanged(CharSequence s, int start, int count, int after) {}

@Override

public void onTextChanged(CharSequence s, int start, int before, int count) {}

@Override

public void afterTextChanged(Editable s) {

String email = s.toString().trim();

TextInputLayout layout = findViewById(R.id.emailLayout);

if (!EMAIL\_PATTERN.matcher(email).matches()) {

layout.setError("Введите корректный email");

} else {

layout.setError(null);

}

}

});

// Validation for password

passwordEditText.addTextChangedListener(new TextWatcher() {

@Override

public void beforeTextChanged(CharSequence s, int start, int count, int after) {}

@Override

public void onTextChanged(CharSequence s, int start, int before, int count) {}

@Override

public void afterTextChanged(Editable s) {

String password = s.toString();

TextInputLayout layout = findViewById(R.id.passwordLayout);

if (!PASSWORD\_PATTERN.matcher(password).matches()) {

layout.setError("Пароль должен содержать минимум 8 символов, включая цифры, буквы верхнего и нижнего регистра и специальные символы");

} else {

layout.setError(null);

}

}

});

// Validation for FIO

fioEditText.addTextChangedListener(new TextWatcher() {

@Override

public void beforeTextChanged(CharSequence s, int start, int count, int after) {}

@Override

public void onTextChanged(CharSequence s, int start, int before, int count) {}

@Override

public void afterTextChanged(Editable s) {

String fio = s.toString().trim();

TextInputLayout layout = findViewById(R.id.fioLayout);

if (!FIO\_PATTERN.matcher(fio).matches()) {

layout.setError("Введите корректное ФИО");

} else {

layout.setError(null);

}

}

});

// Validation for phone

phoneEditText.addTextChangedListener(new TextWatcher() {

@Override

public void beforeTextChanged(CharSequence s, int start, int count, int after) {}

@Override

public void onTextChanged(CharSequence s, int start, int before, int count) {

String phone = s.toString().replaceAll("[^0-9+]", "");

if (!phone.startsWith("+7") && !phone.isEmpty()) {

phone = "+7" + phone;

}

if (!phone.equals(s.toString())) {

phoneEditText.setText(phone);

phoneEditText.setSelection(phone.length());

}

}

@Override

public void afterTextChanged(Editable s) {

String phone = s.toString();

TextInputLayout layout = findViewById(R.id.phoneLayout);

if (!PHONE\_PATTERN.matcher(phone).matches()) {

layout.setError("Введите корректный номер телефона");

} else {

layout.setError(null);

}

}

});

// Validation for city

cityEditText.addTextChangedListener(new TextWatcher() {

@Override

public void beforeTextChanged(CharSequence s, int start, int count, int after) {}

@Override

public void onTextChanged(CharSequence s, int start, int before, int count) {}

@Override

public void afterTextChanged(Editable s) {

String city = s.toString().trim();

TextInputLayout layout = findViewById(R.id.cityLayout);

if (!CITY\_PATTERN.matcher(city).matches()) {

layout.setError("Введите корректное название города");

} else {

layout.setError(null);

}

}

});

// Validation for code word

codeWordEditText.addTextChangedListener(new TextWatcher() {

@Override

public void beforeTextChanged(CharSequence s, int start, int count, int after) {}

@Override

public void onTextChanged(CharSequence s, int start, int before, int count) {}

@Override

public void afterTextChanged(Editable s) {

String codeWord = s.toString().trim();

TextInputLayout layout = findViewById(R.id.codeWordLayout);

if (!CODE\_WORD\_PATTERN.matcher(codeWord).matches()) {

layout.setError("Кодовое слово должно содержать минимум 3 символа");

} else {

layout.setError(null);

}

}

});

}

private void registerUser() {

String email = emailEditText.getText().toString().trim();

String password = passwordEditText.getText().toString();

String confirmPassword = confirmPasswordEditText.getText().toString();

String fio = fioEditText.getText().toString().trim();

String phone = phoneEditText.getText().toString().trim();

String city = cityEditText.getText().toString().trim();

String codeWord = codeWordEditText.getText().toString().trim();

// Check for empty fields

if (email.isEmpty() || password.isEmpty() || confirmPassword.isEmpty() ||

fio.isEmpty() || phone.isEmpty() || city.isEmpty() || codeWord.isEmpty()) {

Toast.makeText(this, "Пожалуйста, заполните все поля", Toast.LENGTH\_SHORT).show();

return;

}

// Check validation

if (!EMAIL\_PATTERN.matcher(email).matches()) {

Toast.makeText(this, "Пожалуйста, введите корректный email", Toast.LENGTH\_SHORT).show();

return;

}

if (!PASSWORD\_PATTERN.matcher(password).matches()) {

Toast.makeText(this, "Пароль не соответствует требованиям", Toast.LENGTH\_SHORT).show();

return;

}

if (!password.equals(confirmPassword)) {

Toast.makeText(this, "Пароли не совпадают", Toast.LENGTH\_SHORT).show();

return;

}

if (!FIO\_PATTERN.matcher(fio).matches()) {

Toast.makeText(this, "Пожалуйста, введите корректное ФИО", Toast.LENGTH\_SHORT).show();

return;

}

if (!PHONE\_PATTERN.matcher(phone).matches()) {

Toast.makeText(this, "Пожалуйста, введите корректный номер телефона", Toast.LENGTH\_SHORT).show();

return;

}

if (!CITY\_PATTERN.matcher(city).matches()) {

Toast.makeText(this, "Пожалуйста, введите корректное название города", Toast.LENGTH\_SHORT).show();

return;

}

if (!CODE\_WORD\_PATTERN.matcher(codeWord).matches()) {

Toast.makeText(this, "Пожалуйста, введите корректное кодовое слово", Toast.LENGTH\_SHORT).show();

return;

}

// Check if user already exists

new Thread(() -> {

User existingUser = userDao.getUserByEmail(email);

if (existingUser != null) {

runOnUiThread(() -> Toast.makeText(this, "Пользователь с таким email уже существует", Toast.LENGTH\_SHORT).show());

return;

}

// Create new user

User user = new User();

user.setEmail(email);

user.setPassword(password);

user.setFio(fio);

user.setPhone(phone);

user.setCity(city);

user.setGender(""); // Пустое значение, пользователь должен выбрать пол

user.setCodeWord(codeWord);

// Save user to database

userDao.insert(user);

// Create profile object

UserProfile profile = new UserProfile(fio, phone, city, "", email);

// Save profile to SharedPreferences

SharedPreferences.Editor editor = sharedPreferences.edit();

Gson gson = new Gson();

String profileJson = gson.toJson(profile);

editor.putString(email, profileJson);

editor.apply();

// Save authorization data

editor.putString("user\_email", email);

editor.putString("user\_password", password);

editor.apply();

runOnUiThread(() -> {

Toast.makeText(this, "Регистрация успешна", Toast.LENGTH\_SHORT).show();

// Go to main screen

Intent intent = new Intent(RegisterActivity.this, MainActivity.class);

intent.putExtra("user\_email", email);

startActivity(intent);

finish();

});

}).start();

}

}

~33) RevenueReportGenerator.java;~

package com.example.bolotnyiysen;

import android.content.Context;

import android.graphics.Bitmap;

import android.graphics.Canvas;

import android.graphics.Color;

import android.graphics.Paint;

import android.graphics.RectF;

import android.os.Build;

import android.os.Environment;

import android.util.Log;

import com.itextpdf.kernel.pdf.PdfDocument;

import com.itextpdf.kernel.pdf.PdfWriter;

import com.itextpdf.layout.Document;

import com.itextpdf.layout.element.Image;

import com.itextpdf.layout.element.Paragraph;

import com.itextpdf.layout.element.Table;

import com.itextpdf.layout.properties.TextAlignment;

import com.itextpdf.layout.properties.UnitValue;

import com.itextpdf.io.image.ImageData;

import com.itextpdf.io.image.ImageDataFactory;

import java.io.ByteArrayOutputStream;

import java.io.File;

import java.io.FileOutputStream;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.List;

import java.util.Locale;

public class RevenueReportGenerator {

private static final String REPORTS\_FOLDER = "BolotnyiYsen\_Reports";

private Context context;

public RevenueReportGenerator(Context context) {

this.context = context;

}

public String generateRevenueReport(List<Purchase> purchases) {

try {

// Создаем папку для отчетов

File reportsDir;

if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.Q) {

reportsDir = new File(Environment.getExternalStoragePublicDirectory(

Environment.DIRECTORY\_DOCUMENTS), REPORTS\_FOLDER);

} else {

reportsDir = new File(Environment.getExternalStoragePublicDirectory(

Environment.DIRECTORY\_DOCUMENTS), REPORTS\_FOLDER);

}

if (!reportsDir.exists()) {

boolean created = reportsDir.mkdirs();

if (!created) {

Log.e("RevenueReport", "Failed to create reports directory");

return null;

}

}

// Генерируем имя файла с датой

SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd\_HH-mm", Locale.getDefault());

String dateStr = dateFormat.format(new Date());

String fileName = "Отчет\_выручки\_" + dateStr + ".pdf";

File reportFile = new File(reportsDir, fileName);

// Создаем PDF документ

PdfWriter writer = new PdfWriter(new FileOutputStream(reportFile));

PdfDocument pdf = new PdfDocument(writer);

Document document = new Document(pdf);

// Добавляем заголовок

Paragraph header = new Paragraph("Отчет о выручке")

.setTextAlignment(TextAlignment.CENTER)

.setFontSize(20);

document.add(header);

// Добавляем дату генерации отчета

document.add(new Paragraph("Дата генерации: " + formatDate(new Date())));

document.add(new Paragraph("\n"));

// Создаем таблицу с данными

Table table = new Table(UnitValue.createPercentArray(4)).useAllAvailableWidth();

table.addHeaderCell("Email");

table.addHeaderCell("Товар");

table.addHeaderCell("Способ оплаты");

table.addHeaderCell("Сумма");

double totalRevenue = 0;

for (Purchase purchase : purchases) {

table.addCell(purchase.getUserEmail());

table.addCell(purchase.getProductName());

table.addCell(purchase.getPaymentType());

table.addCell(String.format("%.2f ₽", purchase.getPrice()));

totalRevenue += purchase.getPrice();

}

document.add(table);

// Добавляем итоговую сумму

document.add(new Paragraph("\n"));

document.add(new Paragraph(String.format("Общая выручка: %.2f ₽", totalRevenue))

.setTextAlignment(TextAlignment.RIGHT)

.setBold());

// Создаем и добавляем круговую диаграмму

Bitmap chartBitmap = createPieChart(purchases);

ByteArrayOutputStream stream = new ByteArrayOutputStream();

chartBitmap.compress(Bitmap.CompressFormat.PNG, 100, stream);

byte[] byteArray = stream.toByteArray();

ImageData imageData = ImageDataFactory.create(byteArray);

Image chartImage = new Image(imageData);

chartImage.setWidth(UnitValue.createPercentValue(100));

document.add(chartImage);

// Закрываем документ

document.close();

return reportFile.getAbsolutePath();

} catch (Exception e) {

Log.e("RevenueReport", "Error generating revenue report", e);

return null;

}

}

private Bitmap createPieChart(List<Purchase> purchases) {

// Создаем битмап для диаграммы

int width = 800;

int height = 600;

Bitmap bitmap = Bitmap.createBitmap(width, height, Bitmap.Config.ARGB\_8888);

Canvas canvas = new Canvas(bitmap);

Paint paint = new Paint();

// Вычисляем общую сумму

double total = 0;

for (Purchase purchase : purchases) {

total += purchase.getPrice();

}

// Рисуем круговую диаграмму

float startAngle = 0;

int[] colors = {

Color.rgb(255, 99, 132),

Color.rgb(54, 162, 235),

Color.rgb(255, 206, 86),

Color.rgb(75, 192, 192),

Color.rgb(153, 102, 255),

Color.rgb(255, 159, 64)

};

RectF rect = new RectF(50, 50, width - 50, height - 50);

int colorIndex = 0;

for (Purchase purchase : purchases) {

float sweepAngle = (float) (360 \* (purchase.getPrice() / total));

paint.setColor(colors[colorIndex % colors.length]);

canvas.drawArc(rect, startAngle, sweepAngle, true, paint);

startAngle += sweepAngle;

colorIndex++;

}

// Добавляем легенду

paint.setTextSize(30);

float legendY = 50;

colorIndex = 0;

for (Purchase purchase : purchases) {

paint.setColor(colors[colorIndex % colors.length]);

canvas.drawRect(50, legendY, 70, legendY + 20, paint);

paint.setColor(Color.BLACK);

canvas.drawText(

String.format("%s: %.2f ₽ (%.1f%%)",

purchase.getProductName(),

purchase.getPrice(),

(purchase.getPrice() / total) \* 100),

80,

legendY + 15,

paint

);

legendY += 30;

colorIndex++;

}

return bitmap;

}

private String formatDate(Date date) {

SimpleDateFormat sdf = new SimpleDateFormat("dd.MM.yyyy HH:mm:ss", Locale.getDefault());

return sdf.format(date);

}

}

~34) TransactionAdapter.java;~

package com.example.bolotnyiysen;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.TextView;

import androidx.annotation.NonNull;

import androidx.recyclerview.widget.RecyclerView;

import java.util.List;

public class TransactionAdapter extends RecyclerView.Adapter<TransactionAdapter.TransactionViewHolder> {

private List<Transaction> transactions;

public TransactionAdapter(List<Transaction> transactions) {

this.transactions = transactions;

}

@NonNull

@Override

public TransactionViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {

View view = LayoutInflater.from(parent.getContext())

.inflate(R.layout.item\_transaction, parent, false);

return new TransactionViewHolder(view);

}

@Override

public void onBindViewHolder(@NonNull TransactionViewHolder holder, int position) {

Transaction transaction = transactions.get(position);

holder.dateTextView.setText(transaction.getDate());

holder.itemsTextView.setText(transaction.getItems());

holder.totalTextView.setText(String.format("Итого: %.2f ₽", transaction.getTotal()));

}

@Override

public int getItemCount() {

return transactions.size();

}

public void updateTransactions(List<Transaction> newTransactions) {

this.transactions = newTransactions;

notifyDataSetChanged();

}

static class TransactionViewHolder extends RecyclerView.ViewHolder {

TextView dateTextView;

TextView itemsTextView;

TextView totalTextView;

TransactionViewHolder(View itemView) {

super(itemView);

dateTextView = itemView.findViewById(R.id.dateTextView);

itemsTextView = itemView.findViewById(R.id.itemsTextView);

totalTextView = itemView.findViewById(R.id.totalTextView);

}

}

}

~35) TransactionDao.java;~

package com.example.bolotnyiysen;

import androidx.room.Dao;

import androidx.room.Insert;

import androidx.room.Query;

import java.util.List;

@Dao

public interface TransactionDao {

@Insert

void insert(Transaction transaction);

@Query("SELECT \* FROM transactions WHERE userEmail = :userEmail ORDER BY date DESC")

List<Transaction> getTransactionsByUserEmail(String userEmail);

}

~36) TransactionHistoryActivity.java;~

package com.example.bolotnyiysen;

import android.content.Intent;

import android.os.Bundle;

import android.view.MenuItem;

import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import androidx.appcompat.widget.Toolbar;

import androidx.recyclerview.widget.LinearLayoutManager;

import androidx.recyclerview.widget.RecyclerView;

import com.google.android.material.bottomnavigation.BottomNavigationView;

import java.util.List;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

public class TransactionHistoryActivity extends AppCompatActivity {

private String userEmail;

private RecyclerView recyclerView;

private PurchaseAdapter purchaseAdapter;

private AppDatabase appDatabase;

private ExecutorService executorService;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_transaction\_history);

// Инициализация

userEmail = getIntent().getStringExtra("user\_email");

if (userEmail == null) {

Toast.makeText(this, "Ошибка: email пользователя не найден", Toast.LENGTH\_SHORT).show();

finish();

return;

}

// Инициализация базы данных

appDatabase = AppDatabase.getInstance(this);

executorService = Executors.newSingleThreadExecutor();

// Настройка Toolbar

Toolbar toolbar = findViewById(R.id.toolbar);

setSupportActionBar(toolbar);

if (getSupportActionBar() != null) {

getSupportActionBar().setTitle("История покупок");

getSupportActionBar().setDisplayHomeAsUpEnabled(true);

}

// Настройка RecyclerView

recyclerView = findViewById(R.id.recyclerView);

recyclerView.setLayoutManager(new LinearLayoutManager(this));

purchaseAdapter = new PurchaseAdapter(null);

recyclerView.setAdapter(purchaseAdapter);

// Загрузка истории покупок

loadPurchaseHistory();

// Настройка нижней навигации

setupBottomNavigation();

}

private void loadPurchaseHistory() {

executorService.execute(() -> {

try {

List<Purchase> purchases = appDatabase.purchaseDao().getPurchasesByUserEmail(userEmail);

runOnUiThread(() -> {

if (purchases != null && !purchases.isEmpty()) {

purchaseAdapter.updatePurchases(purchases);

} else {

Toast.makeText(this, "У вас пока нет покупок", Toast.LENGTH\_SHORT).show();

}

});

} catch (Exception e) {

runOnUiThread(() -> {

Toast.makeText(this, "Ошибка при загрузке истории покупок", Toast.LENGTH\_SHORT).show();

});

}

});

}

private void setupBottomNavigation() {

BottomNavigationView bottomNavigationView = findViewById(R.id.bottomNavigationView);

bottomNavigationView.setOnNavigationItemSelectedListener(item -> {

int itemId = item.getItemId();

Intent intent;

if (itemId == R.id.navigation\_home) {

intent = new Intent(TransactionHistoryActivity.this, MainActivity.class);

} else if (itemId == R.id.navigation\_catalog) {

intent = new Intent(TransactionHistoryActivity.this, CatalogActivity.class);

} else if (itemId == R.id.navigation\_favorites) {

intent = new Intent(TransactionHistoryActivity.this, FavoritesActivity.class);

} else if (itemId == R.id.navigation\_cart) {

intent = new Intent(TransactionHistoryActivity.this, CartActivity.class);

} else if (itemId == R.id.navigation\_profile) {

intent = new Intent(TransactionHistoryActivity.this, ProfileActivity.class);

} else {

return false;

}

intent.putExtra("user\_email", userEmail);

startActivity(intent);

finish();

return true;

});

bottomNavigationView.setSelectedItemId(R.id.navigation\_profile);

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

if (item.getItemId() == android.R.id.home) {

onBackPressed();

return true;

}

return super.onOptionsItemSelected(item);

}

@Override

protected void onDestroy() {

super.onDestroy();

if (executorService != null) {

executorService.shutdown();

}

}

}

~37) User.java;~

package com.example.bolotnyiysen;

import androidx.room.Entity;

import androidx.room.PrimaryKey;

import androidx.room.Ignore;

@Entity(tableName = "users")

public class User {

@PrimaryKey(autoGenerate = true)

private int id;

private String email;

private String password;

private String fio;

private String phone;

private String city;

private String gender;

private String codeWord;

private boolean isAdmin;

@Ignore

public User() {

}

public User(String email, String password, String fio, String phone, String city, String gender, String codeWord, boolean isAdmin) {

this.email = email;

this.password = password;

this.fio = fio;

this.phone = phone;

this.city = city;

this.gender = gender;

this.codeWord = codeWord;

this.isAdmin = isAdmin;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getEmail() {

return email;

}

public String getPassword() {

return password;

}

public String getFio() {

return fio;

}

public String getPhone() {

return phone;

}

public String getCity() {

return city;

}

public String getGender() {

return gender;

}

public String getCodeWord() {

return codeWord;

}

public boolean isAdmin() {

return isAdmin;

}

public void setEmail(String email) {

this.email = email;

}

public void setPassword(String password) {

this.password = password;

}

public void setFio(String fio) {

this.fio = fio;

}

public void setPhone(String phone) {

this.phone = phone;

}

public void setCity(String city) {

this.city = city;

}

public void setGender(String gender) {

this.gender = gender;

}

public void setCodeWord(String codeWord) {

this.codeWord = codeWord;

}

public void setAdmin(boolean admin) {

isAdmin = admin;

}

}

~38) UserAdapter.java;~

package com.example.bolotnyiysen;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.ImageButton;

import android.widget.TextView;

import androidx.annotation.NonNull;

import androidx.recyclerview.widget.RecyclerView;

import java.util.List;

public class UserAdapter extends RecyclerView.Adapter<UserAdapter.UserViewHolder> {

private List<User> users;

private OnUserDeleteListener deleteListener;

public interface OnUserDeleteListener {

void onUserDelete(User user);

}

public UserAdapter(List<User> users, OnUserDeleteListener deleteListener) {

this.users = users;

this.deleteListener = deleteListener;

}

@NonNull

@Override

public UserViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {

View view = LayoutInflater.from(parent.getContext())

.inflate(R.layout.item\_user, parent, false);

return new UserViewHolder(view);

}

@Override

public void onBindViewHolder(@NonNull UserViewHolder holder, int position) {

User user = users.get(position);

holder.userNameTextView.setText(user.getFio());

holder.userEmailTextView.setText("Email: " + user.getEmail());

holder.userPhoneTextView.setText("Телефон: " + user.getPhone());

holder.userCityTextView.setText("Город: " + user.getCity());

holder.userGenderTextView.setText("Пол: " + user.getGender());

holder.deleteButton.setOnClickListener(v -> {

if (deleteListener != null) {

deleteListener.onUserDelete(user);

}

});

}

@Override

public int getItemCount() {

return users.size();

}

public void updateUsers(List<User> newUsers) {

this.users = newUsers;

notifyDataSetChanged();

}

static class UserViewHolder extends RecyclerView.ViewHolder {

TextView userNameTextView;

TextView userEmailTextView;

TextView userPhoneTextView;

TextView userCityTextView;

TextView userGenderTextView;

ImageButton deleteButton;

UserViewHolder(View itemView) {

super(itemView);

userNameTextView = itemView.findViewById(R.id.userNameTextView);

userEmailTextView = itemView.findViewById(R.id.userEmailTextView);

userPhoneTextView = itemView.findViewById(R.id.userPhoneTextView);

userCityTextView = itemView.findViewById(R.id.userCityTextView);

userGenderTextView = itemView.findViewById(R.id.userGenderTextView);

deleteButton = itemView.findViewById(R.id.deleteUserButton);

}

}

}

~39) UserDao.java;~

package com.example.bolotnyiysen;

import androidx.room.Dao;

import androidx.room.Delete;

import androidx.room.Insert;

import androidx.room.Query;

import androidx.room.Update;

import java.util.List;

@Dao

public interface UserDao {

@Insert

void insert(User user);

@Update

void update(User user);

@Query("SELECT \* FROM users WHERE email = :email")

User getUserByEmail(String email);

@Query("SELECT \* FROM users WHERE email = :email AND password = :password")

User getUserByEmailAndPassword(String email, String password);

@Query("SELECT \* FROM users WHERE email = :email AND codeWord = :codeWord")

User getUserByEmailAndCodeWord(String email, String codeWord);

@Query("UPDATE users SET password = :newPassword WHERE email = :email AND codeWord = :codeWord")

int updatePasswordWithCodeWord(String email, String codeWord, String newPassword);

@Query("SELECT COUNT(\*) FROM users")

int getTotalUsers();

@Query("SELECT \* FROM users WHERE email = 'admin@example.com'")

User getAdmin();

@Query("SELECT \* FROM users ORDER BY FIO ASC")

List<User> getAllUsers();

@Delete

void deleteUser(User user);

}

~40) UserProfile.java;~

package com.example.bolotnyiysen;

import java.io.Serializable;

public class UserProfile implements Serializable {

private String fio;

private String phone;

private String city;

private String gender;

private String email;

public UserProfile() {

// Пустой конструктор для SharedPreferences

}

public UserProfile(String fio, String phone, String city, String gender, String email) {

this.fio = fio;

this.phone = phone;

this.city = city;

this.gender = gender;

this.email = email;

}

public String getFio() {

return fio;

}

public String getPhone() {

return phone;

}

public String getCity() {

return city;

}

public String getGender() {

return gender;

}

public String getEmail() {

return email;

}

public void setFio(String fio) {

this.fio = fio;

}

public void setPhone(String phone) {

this.phone = phone;

}

public void setCity(String city) {

this.city = city;

}

public void setGender(String gender) {

this.gender = gender;

}

public void setEmail(String email) {

this.email = email;

}

}

~41) AppDatabase.java;~

package com.example.bolotnyiysen;

import android.content.Context;

import androidx.room.Database;

import androidx.room.Room;

import androidx.room.RoomDatabase;

import androidx.room.migration.Migration;

import androidx.sqlite.db.SupportSQLiteDatabase;

import androidx.annotation.NonNull;

@Database(entities = {User.class, Purchase.class, Product.class}, version = 7, exportSchema = false)

public abstract class AppDatabase extends RoomDatabase {

private static AppDatabase instance;

public abstract UserDao userDao();

public abstract PurchaseDao purchaseDao();

public abstract ProductDao productDao();

private static final Migration MIGRATION\_6\_7 = new Migration(6, 7) {

@Override

public void migrate(SupportSQLiteDatabase database) {

// Add isHeader column to products table

database.execSQL("ALTER TABLE products ADD COLUMN isHeader INTEGER NOT NULL DEFAULT 0");

}

};

private static final Migration MIGRATION\_5\_6 = new Migration(5, 6) {

@Override

public void migrate(SupportSQLiteDatabase database) {

// Add photoPath column to products table

database.execSQL("ALTER TABLE products ADD COLUMN photoPath TEXT");

}

};

private static final Migration MIGRATION\_3\_4 = new Migration(3, 4) {

@Override

public void migrate(SupportSQLiteDatabase database) {

// Добавляем колонку isAdmin в таблицу users

database.execSQL("ALTER TABLE users ADD COLUMN isAdmin INTEGER NOT NULL DEFAULT 0");

}

};

private static final Migration MIGRATION\_2\_3 = new Migration(2, 3) {

@Override

public void migrate(SupportSQLiteDatabase database) {

// Создаем новую таблицу products с правильной схемой

database.execSQL("CREATE TABLE IF NOT EXISTS products\_new (" +

"id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, " +

"name TEXT, " +

"price REAL NOT NULL, " +

"imageResourceId INTEGER NOT NULL, " +

"code TEXT, " +

"manufacturer TEXT, " +

"series TEXT, " +

"type TEXT, " +

"condition TEXT, " +

"bodyShape TEXT, " +

"orientation TEXT, " +

"stringsCount INTEGER NOT NULL, " +

"fretsCount INTEGER NOT NULL, " +

"scaleLength REAL NOT NULL, " +

"quantity INTEGER NOT NULL DEFAULT 1)");

// Проверяем существование старой таблицы products

android.database.Cursor cursor = database.query("SELECT name FROM sqlite\_master WHERE type='table' AND name='products'");

boolean tableExists = cursor.moveToFirst();

cursor.close();

if (tableExists) {

// Копируем данные из старой таблицы в новую

database.execSQL("INSERT INTO products\_new (id, name, price, imageResourceId, code, manufacturer, " +

"series, type, condition, bodyShape, orientation, stringsCount, fretsCount, scaleLength, quantity) " +

"SELECT id, name, price, imageResourceId, code, manufacturer, series, type, condition, " +

"bodyShape, orientation, stringsCount, fretsCount, scaleLength, quantity FROM products");

// Удаляем старую таблицу

database.execSQL("DROP TABLE products");

}

// Переименовываем новую таблицу

database.execSQL("ALTER TABLE products\_new RENAME TO products");

// Создаем новую таблицу purchases с обновленной схемой

database.execSQL("CREATE TABLE IF NOT EXISTS purchases\_new (" +

"id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, " +

"userEmail TEXT, " +

"productName TEXT, " +

"paymentType TEXT, " +

"date TEXT, " +

"price REAL NOT NULL, " +

"status TEXT)");

// Проверяем существование старой таблицы purchases

cursor = database.query("SELECT name FROM sqlite\_master WHERE type='table' AND name='purchases'");

tableExists = cursor.moveToFirst();

cursor.close();

if (tableExists) {

// Копируем данные из старой таблицы в новую

database.execSQL("INSERT INTO purchases\_new (id, userEmail, productName, paymentType, date, price, status) " +

"SELECT id, userEmail, productName, paymentType, date, 0.0, status FROM purchases");

// Удаляем старую таблицу

database.execSQL("DROP TABLE purchases");

}

// Переименовываем новую таблицу

database.execSQL("ALTER TABLE purchases\_new RENAME TO purchases");

}

};

private static final Migration MIGRATION\_4\_5 = new Migration(4, 5) {

@Override

public void migrate(SupportSQLiteDatabase database) {

// Удаляем таблицу промокодов

database.execSQL("DROP TABLE IF EXISTS promo\_codes");

}

};

public static synchronized AppDatabase getInstance(Context context) {

if (instance == null) {

instance = Room.databaseBuilder(context.getApplicationContext(),

AppDatabase.class, "app\_database")

.addMigrations(MIGRATION\_2\_3, MIGRATION\_3\_4, MIGRATION\_4\_5, MIGRATION\_5\_6, MIGRATION\_6\_7)

.addCallback(new RoomDatabase.Callback() {

@Override

public void onCreate(@NonNull SupportSQLiteDatabase db) {

super.onCreate(db);

createAdminUser();

}

@Override

public void onOpen(@NonNull SupportSQLiteDatabase db) {

super.onOpen(db);

// Проверяем существование администратора при каждом открытии базы данных

new Thread(() -> {

User admin = instance.userDao().getAdmin();

if (admin == null) {

createAdminUser();

}

}).start();

}

})

.fallbackToDestructiveMigration()

.build();

}

return instance;

}

private static void createAdminUser() {

new Thread(() -> {

User admin = new User(

"admin@example.com",

"admin123",

"Администратор",

"+79001234567",

"Москва",

"Мужской",

"admin",

true

);

instance.userDao().insert(admin);

}).start();

}

}

~42) CartItem.java;~

package com.example.bolotnyiysen;

public class CartItem {

private Product product;

private int quantity;

public CartItem(Product product, int quantity) {

this.product = product;

this.quantity = quantity;

}

public Product getProduct() {

return product;

}

public void setProduct(Product product) {

this.product = product;

}

public int getQuantity() {

return quantity;

}

public void setQuantity(int quantity) {

this.quantity = quantity;

}

public double getTotalPrice() {

return product.getPrice() \* quantity;

}

}

~43) EditProductActivity.java;~

package com.example.bolotnyiysen;

import android.content.Intent;

import android.net.Uri;

import android.os.Bundle;

import android.provider.MediaStore;

import android.widget.ImageView;

import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import androidx.appcompat.widget.Toolbar;

import com.google.android.material.button.MaterialButton;

import com.google.android.material.textfield.TextInputEditText;

import java.io.File;

import java.io.FileOutputStream;

import java.io.InputStream;

import java.io.OutputStream;

public class EditProductActivity extends AppCompatActivity {

private static final int PICK\_IMAGE\_REQUEST = 1;

private ImageView productImageView;

private TextInputEditText productNameInput;

private TextInputEditText productPriceInput;

private TextInputEditText productCodeInput;

private TextInputEditText manufacturerInput;

private TextInputEditText seriesInput;

private TextInputEditText typeInput;

private TextInputEditText conditionInput;

private TextInputEditText bodyShapeInput;

private TextInputEditText orientationInput;

private TextInputEditText stringsCountInput;

private TextInputEditText fretsCountInput;

private TextInputEditText scaleLengthInput;

private MaterialButton saveProductButton;

private String currentPhotoPath;

private Product product;

private AppDatabase appDatabase;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_edit\_product);

// Инициализация базы данных

appDatabase = AppDatabase.getInstance(this);

// Получение данных о товаре из Intent

product = (Product) getIntent().getSerializableExtra("product");

if (product == null) {

Toast.makeText(this, "Ошибка: товар не найден", Toast.LENGTH\_SHORT).show();

finish();

return;

}

// Настройка Toolbar

Toolbar toolbar = findViewById(R.id.toolbar);

setSupportActionBar(toolbar);

if (getSupportActionBar() != null) {

getSupportActionBar().setTitle("Редактирование товара");

getSupportActionBar().setDisplayHomeAsUpEnabled(true);

}

// Инициализация views

initializeViews();

// Заполнение полей данными товара

fillProductData();

// Настройка обработчиков событий

setupEventHandlers();

}

private void initializeViews() {

productImageView = findViewById(R.id.productImageView);

productNameInput = findViewById(R.id.productNameInput);

productPriceInput = findViewById(R.id.productPriceInput);

productCodeInput = findViewById(R.id.productCodeInput);

manufacturerInput = findViewById(R.id.manufacturerInput);

seriesInput = findViewById(R.id.seriesInput);

typeInput = findViewById(R.id.typeInput);

conditionInput = findViewById(R.id.conditionInput);

bodyShapeInput = findViewById(R.id.bodyShapeInput);

orientationInput = findViewById(R.id.orientationInput);

stringsCountInput = findViewById(R.id.stringsCountInput);

fretsCountInput = findViewById(R.id.fretsCountInput);

scaleLengthInput = findViewById(R.id.scaleLengthInput);

saveProductButton = findViewById(R.id.saveProductButton);

}

private void fillProductData() {

productNameInput.setText(product.getName());

productPriceInput.setText(String.valueOf(product.getPrice()));

productCodeInput.setText(product.getCode());

manufacturerInput.setText(product.getManufacturer());

seriesInput.setText(product.getSeries());

typeInput.setText(product.getType());

conditionInput.setText(product.getCondition());

bodyShapeInput.setText(product.getBodyShape());

orientationInput.setText(product.getOrientation());

stringsCountInput.setText(String.valueOf(product.getStringsCount()));

fretsCountInput.setText(String.valueOf(product.getFretsCount()));

scaleLengthInput.setText(String.valueOf(product.getScaleLength()));

// Загрузка изображения товара

if (product.getPhotoPath() != null) {

File photoFile = new File(product.getPhotoPath());

if (photoFile.exists()) {

productImageView.setImageURI(Uri.fromFile(photoFile));

currentPhotoPath = product.getPhotoPath();

}

}

}

private void setupEventHandlers() {

productImageView.setOnClickListener(v -> {

Intent intent = new Intent(Intent.ACTION\_PICK, MediaStore.Images.Media.EXTERNAL\_CONTENT\_URI);

startActivityForResult(intent, PICK\_IMAGE\_REQUEST);

});

saveProductButton.setOnClickListener(v -> saveProduct());

}

@Override

protected void onActivityResult(int requestCode, int resultCode, Intent data) {

super.onActivityResult(requestCode, resultCode, data);

if (requestCode == PICK\_IMAGE\_REQUEST && resultCode == RESULT\_OK && data != null) {

try {

Uri selectedImage = data.getData();

productImageView.setImageURI(selectedImage);

// Копируем изображение во внутреннее хранилище

String fileName = "product\_" + System.currentTimeMillis() + ".jpg";

File destinationFile = new File(getFilesDir(), fileName);

InputStream inputStream = getContentResolver().openInputStream(selectedImage);

OutputStream outputStream = new FileOutputStream(destinationFile);

byte[] buffer = new byte[1024];

int length;

while ((length = inputStream.read(buffer)) > 0) {

outputStream.write(buffer, 0, length);

}

inputStream.close();

outputStream.close();

currentPhotoPath = destinationFile.getAbsolutePath();

} catch (Exception e) {

e.printStackTrace();

Toast.makeText(this, "Ошибка при загрузке изображения", Toast.LENGTH\_SHORT).show();

}

}

}

private void saveProduct() {

// Получение данных из полей ввода

String name = productNameInput.getText().toString().trim();

String priceStr = productPriceInput.getText().toString().trim();

String code = productCodeInput.getText().toString().trim();

String manufacturer = manufacturerInput.getText().toString().trim();

String series = seriesInput.getText().toString().trim();

String type = typeInput.getText().toString().trim();

String condition = conditionInput.getText().toString().trim();

String bodyShape = bodyShapeInput.getText().toString().trim();

String orientation = orientationInput.getText().toString().trim();

String stringsCountStr = stringsCountInput.getText().toString().trim();

String fretsCountStr = fretsCountInput.getText().toString().trim();

String scaleLengthStr = scaleLengthInput.getText().toString().trim();

// Проверка обязательных полей

if (name.isEmpty() || priceStr.isEmpty() || code.isEmpty()) {

Toast.makeText(this, "Заполните все обязательные поля", Toast.LENGTH\_SHORT).show();

return;

}

try {

// Парсинг числовых значений

double price = Double.parseDouble(priceStr);

int stringsCount = stringsCountStr.isEmpty() ? 0 : Integer.parseInt(stringsCountStr);

int fretsCount = fretsCountStr.isEmpty() ? 0 : Integer.parseInt(fretsCountStr);

double scaleLength = scaleLengthStr.isEmpty() ? 0 : Double.parseDouble(scaleLengthStr);

// Обновление данных товара

product.setName(name);

product.setPrice(price);

product.setCode(code);

product.setManufacturer(manufacturer);

product.setSeries(series);

product.setType(type);

product.setCondition(condition);

product.setBodyShape(bodyShape);

product.setOrientation(orientation);

product.setStringsCount(stringsCount);

product.setFretsCount(fretsCount);

product.setScaleLength(scaleLength);

if (currentPhotoPath != null) {

product.setPhotoPath(currentPhotoPath);

}

// Сохранение в базу данных

new Thread(() -> {

try {

appDatabase.productDao().update(product);

runOnUiThread(() -> {

Toast.makeText(EditProductActivity.this,

"Товар успешно обновлен",

Toast.LENGTH\_SHORT).show();

finish();

});

} catch (Exception e) {

e.printStackTrace();

runOnUiThread(() -> {

Toast.makeText(EditProductActivity.this,

"Ошибка при сохранении: " + e.getMessage(),

Toast.LENGTH\_LONG).show();

});

}

}).start();

} catch (NumberFormatException e) {

Toast.makeText(this, "Проверьте правильность ввода числовых значений",

Toast.LENGTH\_SHORT).show();

}

}

@Override

public boolean onSupportNavigateUp() {

onBackPressed();

return true;

}

}

~44) BannerAdapter.java;~

package com.example.bolotnyiysen;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.ImageView;

import androidx.annotation.NonNull;

import androidx.recyclerview.widget.RecyclerView;

import java.util.List;

public class BannerAdapter extends RecyclerView.Adapter<BannerAdapter.BannerViewHolder> {

private List<Integer> bannerImages;

public BannerAdapter(List<Integer> bannerImages) {

this.bannerImages = bannerImages;

}

@NonNull

@Override

public BannerViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {

View view = LayoutInflater.from(parent.getContext())

.inflate(R.layout.banner\_item, parent, false);

return new BannerViewHolder(view);

}

@Override

public void onBindViewHolder(@NonNull BannerViewHolder holder, int position) {

holder.bannerImage.setImageResource(bannerImages.get(position));

}

@Override

public int getItemCount() {

return bannerImages.size();

}

static class BannerViewHolder extends RecyclerView.ViewHolder {

ImageView bannerImage;

BannerViewHolder(@NonNull View itemView) {

super(itemView);

bannerImage = itemView.findViewById(R.id.bannerImage);

}

}

}