МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Информационные системы и технологии

Специальность 1-98 01 03 «Программное обеспечение информационной

безопасности мобильных систем»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

Разработка программного средства «ShoeStore»

Выполнил студент Петрович Антон Геннадьевич

(Ф.И.О.)

Руководитель проекта ст. преподаватель Северинчик Н.А.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Заведующий кафедрой д.к.т.н., Смелов В.В.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Консультанты ст. преподаватель Северинчик Н.А.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Нормоконтроллер ст. преподаватель Северинчик Н.А.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Минск 2020

**Реферат**

Пояснительная записка курсового проекта содержит 27 страницы, 7 изображений, 4 источника литературы,1 приложение.

Основными целями курсового проекта являются: разработка программного средства – «ShoeStore».

Пояснительная записка состоит из введения, пяти разделов, заключения.

Во введении представлена общая информация, дающая представление о предстоящей работе, определены цели.

В первом разделе представлена постановка задачи.

Во втором разделе рассматривается разработка архитектуры проекта.

В третьем разделе рассматривается разработка функциональной модели приложения.

В четвертом разделе представлено руководство пользователя.

В заключении представлены итоги курсового проектирования и задачи, которые были решены в ходе разработки программного средства.

Содержание

[Введение 6](#_Toc515317847)

[1. Постановка задачи. 7](#_Toc515317848)

[2. Схема базы данных 10](#_Toc515317850)

[3. Программная реализация приложения 17](#_Toc515317851)

[3.1 Реализация обфускации приложения 17](#_Toc515317852)

[3.2 Реализация защиты пароля приложения 17](#_Toc515317853)

[3.3. Реализация защита от снимка экрана приложения 18](#_Toc515317854)

[4. Руководство пользователя 21](#_Toc515317855)

[4.1 Android приложение 21](#_Toc515317856)

[Заключение 27](#_Toc515317857)

[Список использованных источников 28](#_Toc515317858)

[Приложение А 29](#_Toc515317859)

[Приложение B 29](#_Toc515317860)

# Введение

Защита мобильного приложения (особенно для Android) – один из наиважнейших этапов его разработки. Без принятия должных мер в Сети сможет быстро появиться его взломанная копия, в которой, например, будет отключена реклама и/или верификация через социальные сервисы. Но главная опасность – пират сумеет произвести декомпиляцию приложения чтобы узнать его изнутри и воссоздать аналогичное. Существует целый ряд мер, направленных на защиту программы. То, какие именно нужно использовать, зависит от специфики проекта.

При разработке мобильного приложения следует учитывать, что данные, которыми оперирует это приложение, могут представлять определенный интерес для третьих лиц. Степень ценности этих данных варьируется в широких пределах, тем не менее, даже наиболее простая приватная информация, например, пароль входа в приложение, требует проработки ее защиты. Особенно это важно в свете распространения мобильных приложений на все сферы электронных услуг, включая финансовые, банковские операции, хранение и передачу личных данных и так далее.

Темой курсового проекта является разработка программного средства «ShoeStore» - мобильного приложения, целью которого является, предоставление коллекции товаров, которые могут приобрести люди, заинтересованные в обуви.

Целью курсового проекта является проектирование и реализация программного средства «ShoeStore».

# Постановка задачи.

Конечным продуктом будет выступать мобильное приложение на устройстве с операционной системой Android 6.0 и выше.

SQLCipher применяется во множестве коммерческих разработок и продуктов с открытым исходным кодом. Это расширение для SQLite с открытым исходным кодом, которое поддерживает прозрачное 256-битное AES-шифрование файлов баз данных.В общем, SQLCipher – одна из самых популярных систем шифрования баз данных для мобильных, встраиваемых и настольных платформ. SQLite является самым распространенным средством для хранения данных на iPhone со стабильным API и хорошей документацией. Очевидно, что не хотелось бы мешать код приложения с собственно шифрованием, а делать это как можно прозрачней. Платформа Android имеет встроенный инструментарий для управления базой данных sqlite3. SQLite - это замечательный продукт, который успел завоевать признание во всем мире и получить множество наград. Я сам в свое время узнал о существовании SQLite от своего коллеги по работе, когда он сказал, что браузер Firefox использует SQLite для своих настроек. Интересно отметить, что iOS (iPhone, iPod, iPad) также используют базы данных SQLite. А недавно и Microsoft присоединилась к данному решению и телефоны Windows Phone 8 также работают с SQLite.

SQLite - это проект с открытыми исходными кодами, поддерживающий стандартные возможности обычной SQL: синтаксис, транзакции и др. Занимает очень мало места - около 250 кб. Домашняя страница SQLite: http://www.sqlite.org.

Такую функциональность предоставляют SQLCipher и OpenSSL. Оба проекта замечательны своим открытым исходным кодом, а также поддержкой Mac/Windows/Linux. При использовании этой библиотеки доступ к БД осуществляется с помощью стандартного SQLite API. Все, связанное с шифрованием, происходит за сценой, разработчику не требуется ни о чем заботиться.

Proguard – наверное самый часто используемый инструмент для обфускации кода при разработке под Андроид. Главная задача ProGuard — поменять имена объектов, классов, методов, тем самым затрудняя анализ кода для реверс-инженера. Помимо этого, он еще и оптимизирует код, удаляя неиспользуемые в программе ресурсы. На шаге Shrinking step ProGuard начиная от точек входа в программу рекурсивно определяет, какие классы и члены классов (переменные, методы, константы) используются. Все другие классы или члены классов будут удалены из приложения. Далее на шаге Optimization step ProGuard может классы и методы, которые не являются точками входа, сделать private, final, static, удалить неиспользуемые параметры, “встроить” (inline) методы, и т.д. При шаге Obfuscation step ProGuard переименовывает классы и члены классов, которые не являются точками входа. Точки входа сохраняют свое оригинальное название. Это затрудняет декомпиляцию и исследование работы приложения (reverse engineering).

PBKDF2 (англ. Password-Based Key Derivation Function) — стандарт формирования ключа на основе пароля. Является частью PKCS #5 v2.0 (RFC 2898). Заменил PBKDF1, который ограничивал длину порождаемого ключа 160 битами. PBKDF2 использует псевдослучайную функцию для получения ключей. Длина генерируемого ключа не ограничивается (хотя эффективная мощность пространства ключей может быть ограничена особенностями применяемой псевдослучайной функции). Использование PBKDF2 рекомендовано для новых программ и продуктов. В качестве псевдослучайной может быть выбрана криптографическая хеш-функция, шифр, HMAC. Используется как первая и последняя стадия в адаптивной криптографической функции формирования ключа на основе пароля scrypt. Данная функция была специально разработана для приложений, где вычисление PBKDF2 оказывается слишком быстрым.

# 2.Схема базы данных

В ходе написания приложения была разработана удаленная база данных «ShoeStore»схема которой представлена на рисунке 3.1.

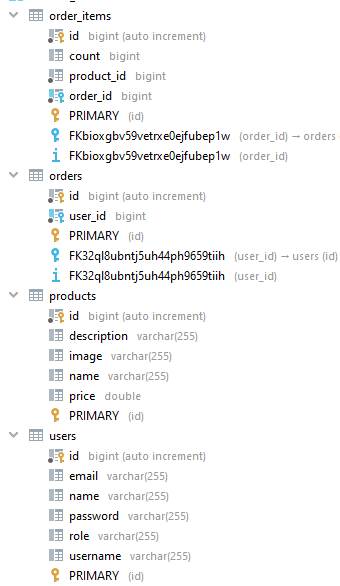


Рисунок 3.1 – структура базы данных

В таблице order\_items хранятся данные о каждом заказанном товаре. Таблица orders содержит себе все заказы конкретного пользователя. В таблице products храниться вся информация о доступных товарах в каталоге. Таблица users хранит в себе данные, которые относятся к пользователю.

# 3.Программная реализация приложения

## 3.1 Реализация мобильного приложения

Мобильное приложение реализовано для операционной системы Android 6.0 и выше. Данное приложение было разработано в специальной среде разработки –Android Studio.

Графическая структура классов представлена на рисунке 4.2.1.

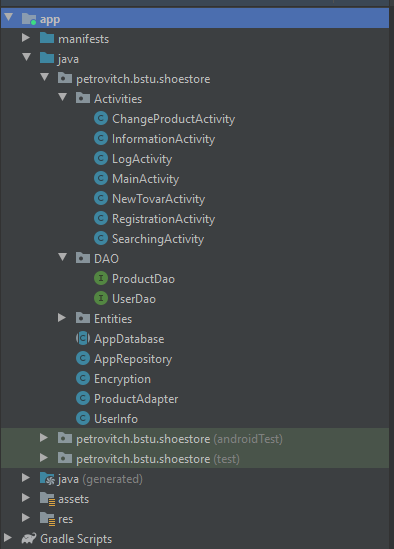


Рисунок 4.2.1 – структура проекта

Entites содержит в себе модель данных для работы мобильного клиента и реализует:

1. Order – модель заказа;
2. Product – модель товара;
3. User – модель пользователя;
4. Клиент управления Snackbar на любой из активностей;

Также проект содержит два основных класса активности:

1. MainActivity – основной класс для работы с приложением. Здесь выполняются основные функции: получение актуальных курсовых работ, загрузка приложений к этим работам, добавление и удаление избранных работ, а также детализация по определенной курсовой работе;
2. LogActivity – класс, реализующий методы для регистрации и авторизации пользователя.

### **3.2 Реализация обфускации приложения**

Обфуска́ция (от лат. obfuscare — затенять, затемнять; и англ. obfuscate — делать неочевидным, запутанным, сбивать с толку) или запутывание кода — приведение исходного текста или исполняемого кода программы к виду, сохраняющему её функциональность, но затрудняющему анализ, понимание алгоритмов работы и модификацию при декомпиляции.

«Запутывание» кода может осуществляться на уровне алгоритма, исходного текста или ассемблерного текста. Для создания запутанного ассемблерного текста могут использоваться специализированные компиляторы, использующие неочевидные или недокументированные возможности среды исполнения программы. Существуют также специальные программы, производящие обфускацию, называемые обфускаторами.

Proguard – наверное самый часто используемый инструмент для обфускации кода при разработке под Андроид. Главная задача Proguard — поменять имена объектов, классов, методов, тем самым затрудняя анализ кода для реверс-инженера. Помимо этого, он еще и оптимизирует код, удаляя неиспользуемые в программе ресурсы.

На рисунке 3.1.1 представлен пример использования инструмента Proguard.

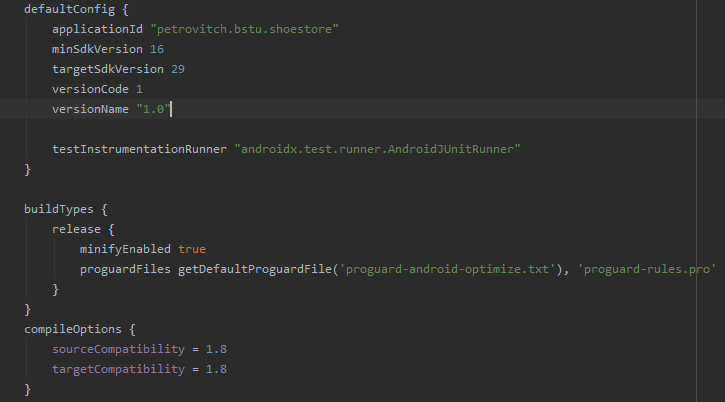


Рисунок 3.1.1 – реализация proguard

### **3.3 Реализация защиты пароля приложения**

Для реализации защиты мобильного приложения авторизацией применяется стандарт формирования ключа на основе пароля PBKDF2. Основная задача PBKDF2 использует псевдослучайную функцию для получения ключей. Длина генерируемого ключа не ограничивается (хотя эффективная мощность пространства ключей может быть ограничена особенностями применяемой псевдослучайной функции). На рисунке 3.1.1 представлен пример использования стандарта формирования ключа на основе пароля PBKDF2.

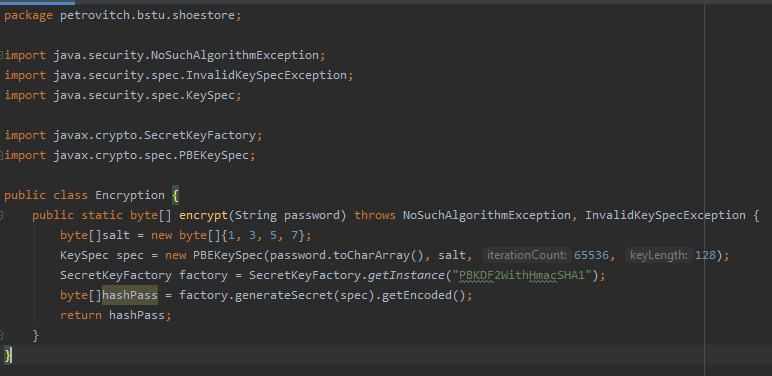


Рисунок 3.1.1 – class Encryption

### **3.4 Реализация защиты приложения от снимка экрана**

Для реализации защиты мобильного приложения авторизацией применяется работа с параметрами безопасности.

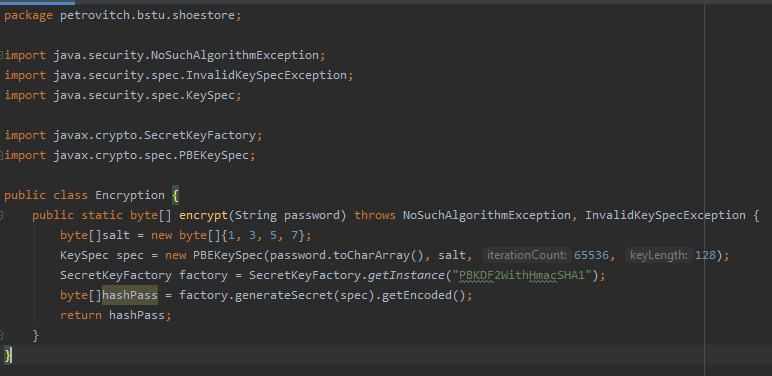


Рисунок 3.1.1 – class Encryption

# 4.Руководство пользователя

## 4.1 Android приложение

Для начала необходимо установить приложение на мобильный устройство с API не меньше 23 (Android 6.0). Запустим наше приложение. Сразу после запуска мы увидим окно авторизации и регистрации (Рисунок 4.1).

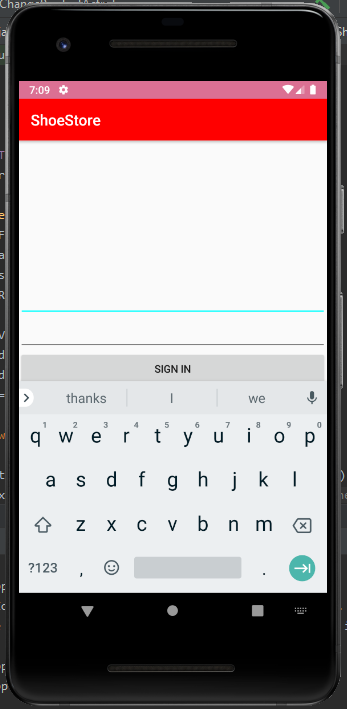


Рисунок 4.1. – окно авторизации и регистрации

После ввода данных (имя пользователя и пароль) мы можем видеть все товары, доступные для заказа на данный момент (Рисунок 4.2).

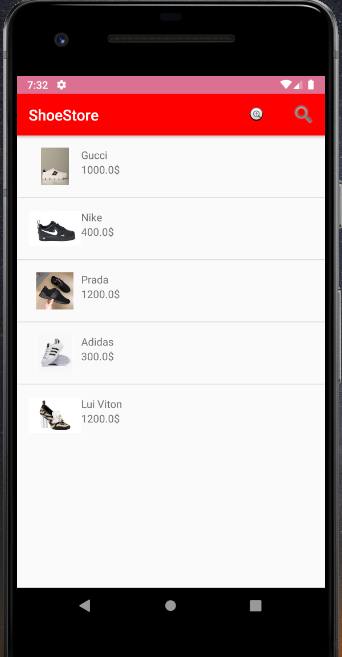
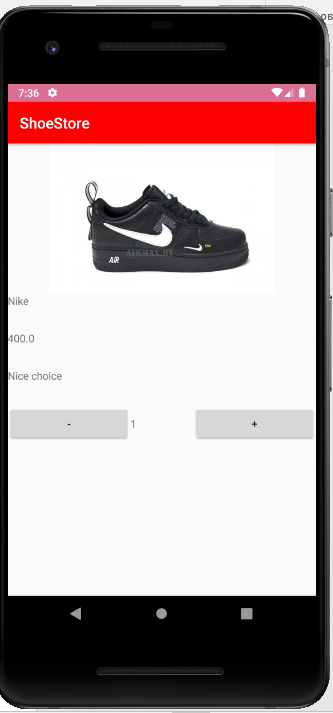


Рисунок 4.2 – все записи

Для переключения получения информации о конкретном товаре и возможности его заказа используются обработчик нажатия класса ListView, в котором данные о выбраном товаре передаются на следующую Activity с помощью Intent  
(Рисунок 4.3).



Риснок 4.3 – окно информации о товаре

Для поиска товара по названию и сортировки товаров по цене была реализована отдельная Activity с соответствующими полями и кнопками, в обработчиках которых получается информация от сервера с соответствующими значениями. (Рисунок 4.4)

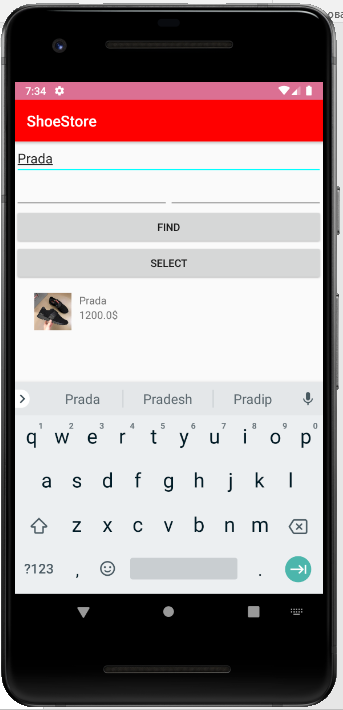


Рисунок 4.4 – Поиск и выборка товаров

# Заключение

Решая поставленную задачу, я пришел к удовлетворительному результату. Таким образом, была достигнута цель, и был создан проект «Shoestore-магазин спортивной обуви», который включает в себя android–клиент.

Программная система имеет простой, интуитивно понятный пользователю интерфейс.

В соответствии с полученным результатом работы программы можно сделать вывод, что разработанный продукт работает корректно, а требования технического задания выполнены в полном объеме.

В результате приложение обладает следующими особенностями:

* реализован качественный алгоритм для защиты приложения при авторизации;
* реализована обфускация кода для проекта;
* реализовано около 7 экранов;
* реализовано хранилище данных в локальной бд;
* реализована защита от снимков экрана;

# Список использованных источников

1. Spring - https://spring.io/;

2. Retrofit - https://square.github.io/retrofit/;

3. SQL Server Management Studio - https://docs.microsoft.com/en-us/sql/ssms;

4. Android documentation - https://developer.android.com/docs.

# Приложение А

package com.example.onlinemarketplace.services;

import retrofit2.Retrofit;

import retrofit2.converter.gson.GsonConverterFactory;

public class RetrofitService {

public static Retrofit retrofit;

public static UserService userService;

public static ProductService productService;

public static OrderService orderService;

public static void init(){

if (retrofit == null) {

retrofit = new Retrofit.Builder()

.baseUrl("https://4088047b3fc6.ngrok.io/")

.addConverterFactory(GsonConverterFactory.create())

.build();

userService = retrofit.create(UserService.class);

productService = retrofit.create(ProductService.class);

orderService = retrofit.create(OrderService.class);

}

}

}

# Приложение B

package com.example.onlinemarketplace;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;

import android.os.Bundle;

import android.util.Log;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;

import android.widget.AdapterView;

import android.widget.EditText;

import android.widget.LinearLayout;

import android.widget.ListView;

import java.util.ArrayList;

import java.util.List;

import retrofit2.Call;

import retrofit2.Callback;

import retrofit2.Response;

import com.example.onlinemarketplace.Entities.Product;

import com.example.onlinemarketplace.Entities.User;

import com.example.onlinemarketplace.dto.ProductInfoDto;

import com.example.onlinemarketplace.services.ProductService;

import com.example.onlinemarketplace.services.RetrofitService;

public class MainActivity extends AppCompatActivity {

final int DIALOG = 1;

public static String searchText;

EditText searchTextView;

public List<User> allUsers;

public ProductService productService;

LinearLayout view;

public List<Product> listProduct;

ListView productsList;

ProductAdapter productAdapter;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

productsList = findViewById(R.id.productList);

productService = RetrofitService.productService;

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

Intent intentInfo = new Intent(this, InfoProductActivity.class);

listProduct = new ArrayList<>();

getAllProducts();

Log.d("LifecycleCheck", "onCreate: ");

productsList.setOnItemClickListener(new AdapterView.OnItemClickListener() {

@Override

public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

Product product = listProduct.get((int)id);

if (UserInfo.role.equals("ADMIN")){

intentEdit.putExtra("productId", product.getPid());

intentEdit.putExtra("productName", product.getProductName());

intentEdit.putExtra("productPrice", product.getPrice());

intentEdit.putExtra("productImagePath", product.getImagePath());

intentEdit.putExtra("productDescription", product.getDescription());

startActivity(intentEdit);

} else if (UserInfo.role.equals("USER")){

intentEdit.putExtra("productId", product.getPid());

intentInfo.putExtra("productName", product.getProductName());

intentInfo.putExtra("productPrice", product.getPrice());

intentInfo.putExtra("productImagePath", product.getImagePath());

intentInfo.putExtra("productDescription", product.getDescription());

startActivity(intentInfo);

}

}

});

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

getMenuInflater().inflate(R.menu.main\_menu, menu);

if (UserInfo.role.equals("ADMIN")){

menu.getItem(0).setVisible(true);

}

else {

menu.getItem(0).setVisible(false);

}

return true;

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

switch(item.getItemId()){

case R.id.new\_product:

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

startActivity(intent);

return true;

case R.id.search:

Intent intent1 = new Intent(this, SearchActivity.class);

startActivity(intent1);

return true;

}

return super.onOptionsItemSelected(item);

}

public void getAllProducts(){

Call<List<ProductInfoDto>> call = productService.getProducts(UserInfo.getAuthToken());

call.enqueue(new Callback<List<ProductInfoDto>>() {

@Override

public void onResponse(Call<List<ProductInfoDto>> call, Response<List<ProductInfoDto>> response) {

List<ProductInfoDto> productInfoDtos = response.body();

for(ProductInfoDto elem: productInfoDtos){

Product product = elem.toProduct();

listProduct.add(product);

}

productAdapter = new ProductAdapter(getApplicationContext(), R.layout.list\_item, listProduct);

productsList.setAdapter(productAdapter);

}

@Override

public void onFailure(Call<List<ProductInfoDto>> call, Throwable t) {

Log.d("CheckProducts", String.valueOf(t.getMessage()));

}

});

}

public void okClicked(){

}

# Приложение C

package com.example.onlinemarketplace;

import androidx.appcompat.app.AppCompatActivity;

import android.graphics.drawable.Drawable;

import android.os.Bundle;

import android.util.Log;

import android.view.View;

import android.widget.ImageView;

import android.widget.TextView;

import android.widget.Toast;

import java.io.IOException;

import java.io.InputStream;

import java.util.ArrayList;

import java.util.List;

import java.util.Objects;

import retrofit2.Call;

import retrofit2.Callback;

import retrofit2.Response;

import com.example.onlinemarketplace.dto.CreateOrderDto;

import com.example.onlinemarketplace.dto.CreateOrderItemDto;

import com.example.onlinemarketplace.services.OrderService;

import com.example.onlinemarketplace.services.ProductService;

import com.example.onlinemarketplace.services.RetrofitService;

public class InfoProductActivity extends AppCompatActivity {

ImageView imageView;

TextView nameTextView;

TextView priceTextView;

TextView descriptionTextView;

TextView countText;

OrderService orderSerice;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

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

imageView = findViewById(R.id.imageView);

nameTextView = findViewById(R.id.nameTextView);

priceTextView = findViewById(R.id.priceTextView);

descriptionTextView = findViewById(R.id.descriptionTextView);

countText = findViewById(R.id.countText);

countText.setText(String.valueOf(0));

orderSerice = RetrofitService.orderService;

nameTextView.setText(getIntent().getStringExtra("productName"));

priceTextView.setText(String.valueOf(getIntent().getDoubleExtra("productPrice", 1000)));

descriptionTextView.setText(getIntent().getStringExtra("productDescription"));

InputStream inputStream = null;

try{

inputStream = getApplicationContext().getAssets().open(Objects.requireNonNull(getIntent().getStringExtra("productImagePath")));

Drawable d = Drawable.createFromStream(inputStream, null);

imageView.setImageDrawable(d);

}

catch (IOException e){

Log.d("ImageCheck", "getView: " + e.getMessage());

}

finally {

try{

if(inputStream!=null)

inputStream.close();

}

catch (IOException ex){

ex.printStackTrace();

}

}

}

public void minusClick(View view){

if(Integer.parseInt(countText.getText().toString()) > 0){

countText.setText(String.valueOf(Integer.parseInt(countText.getText().toString()) - 1));

}

}

public void plusClick(View view){

countText.setText(String.valueOf(Integer.parseInt(countText.getText().toString()) + 1));

}

public void orderClick(View view){

CreateOrderItemDto createOrderItemDto = new CreateOrderItemDto(getIntent().getLongExtra("productId", 1), Long.parseLong(countText.getText().toString()));

List<CreateOrderItemDto> createOrderItemsList = new ArrayList<>();

createOrderItemsList.add(createOrderItemDto);

CreateOrderDto createOrderDto = new CreateOrderDto(createOrderItemsList);

Toast.makeText(this, "Order accepted", Toast.LENGTH\_SHORT).show();

Call<Void> call = orderSerice.createOrder(UserInfo.getAuthToken(), createOrderDto);

call.enqueue(new Callback<Void>() {

@Override

public void onResponse(Call<Void> call, Response<Void> response) {

Log.d("OrderCheck", "onResponse: " + response.message());

}

@Override

public void onFailure(Call<Void> call, Throwable t) {

}

});

}

}