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 мобильного приложения «Ordering food»

Выполнил студент Голубев Александр Владимирович

(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 21 страницы, 7 изображений, 4 источника литературы, 2 приложения.

Основными целями курсового проекта являются: разработка мобильного приложения – «Ordering food».

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. Архитектура проекта 9](#_Toc515317849)

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

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

[4.1 Реализация сервера 17](#_Toc515317852)

[4.2 Реализация мобильного приложения 17](#_Toc515317853)

[4.2.1 REST-клиент 18](#_Toc515317854)

[5. Руководство пользователя 21](#_Toc515317855)

[5.1 Android приложение 21](#_Toc515317856)

[Заключение 27](#_Toc515317857)

[Список использованных источников 28](#_Toc515317858)

[Приложение А 29](#_Toc515317859)

[Приложение B 29](#_Toc515317860)

# Введение

Мобильное приложение - программное обеспечение, предназначенное для работы на смартфонах, планшетах и других мобильных устройствах, разработанное для конкретной платформы (iOS, Android, Windows Phone и т. д.). Многие мобильные приложения предустановлены на самом устройстве или могут быть загружены на него из онлайновых магазинов приложений, таких как App Store, Google Play, и других, бесплатно или за плату.

Первоначально мобильные приложения использовались для быстрой проверки электронной почты, но их высокий спрос привел к расширению их назначений и в других областях, таких как игры для мобильных телефонов и GPS, общение, просмотр видео и пользование интернетом.

Темой курсового проекта является разработка мобильного приложения «Ordering food» - ресурса, целью которого является предоставление коллекции товаров, которые могут приобрести люди, заинтересованные в технике.

Целью курсового проекта является проектирование и реализация мобильного приложения «Ordering food».

# Постановка задачи.

Необходимо разработать приложение для мобильных устройств, которые работают под управлением операционной системы Android. Приложение должно представлять из себя каталог еды с возможность добавления в корзину с последующем редактированием.

Приложение должно иметь 2 роли, пользователь, который может создать заказ, на основе предложенного меню и администратор, который имеет возможность обработать заказ, создавать новые элементы меню и редактировать уже созданные.

В качестве базы данных должна выступать SQLite. Авторизация должна быть реализована с помощью хеширования паролей на основе алгоритма SHA-512. Также необходимо обеспечить невозможность создания скриншотов некоторых экрана.

1. **Архитекутра проекта**

Мой проект состоит из:

* Java-классов, являющихся подклассами основных классов из Android SDK ([View](http://developer.android.com/reference/android/view/View.html), [Activity](http://developer.android.com/reference/android/app/Activity.html),  [Intent](http://developer.android.com/reference/android/content/Intent.html)) и Java-классов, у которых нет родителей в Android SDK.
* Манифеста приложения
* Ресурсов наподобие строк, изображений и т.п.
* Файлов

Манифест Android представляет собой XML файл и выполняет несколько функций :

* Определяет имя Java-пакета приложения. Имя пакета представляет собой уникальный идентификатор для приложения.
* Описывает компоненты приложения — активити, сервисы, броадкаст-ресиверы и контент-провайдеры. Определяет имена классов, реализующие каждый из компонентов и оглашает их возможности (например, какие Intent-сообщения они могут обрабатывать). Эти объявления позволяют системе Android знать, какие компоненты и при каких условиях могут быть запущены.
* Предопределяет процессы, которые будут содержать компоненты приложения.
* Объявляет разрешения, которые приложение должно иметь для доступа к защищённым частям API и взаимодействия с другими приложениями.

Каждое современное GUI приложение в той или иной форме использует ресурсы. Android-приложения — не исключение. Они используют следующие типы ресурсов:

* Изображения
* Слои GUI (XML файлы)
* Объявления меню (XML файлы)
* Текстовые строки

# Схема базы данных

В качестве мобильной СУБД используется SQLite. Пароли пользователей хранятся в виде хэша, созданного с помощью SHA-512

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

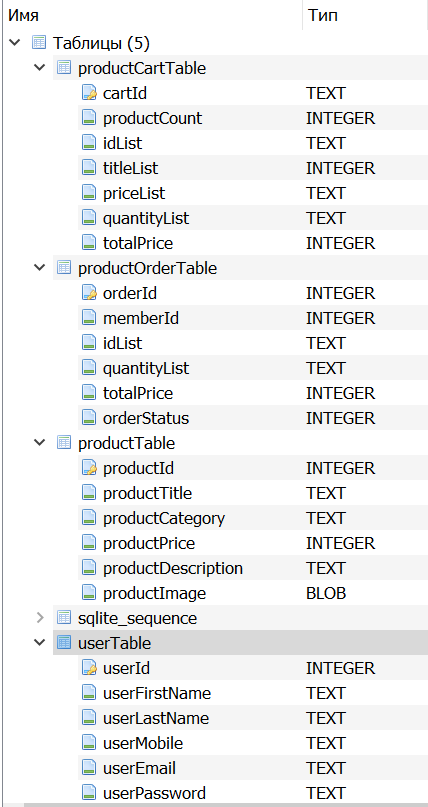


Рисунок 3.1 – структура базы данных

В таблице userTable хранятся данные о каждом зарегистрированном пользователе. Таблица productTable содержит себе все продукты, добавленные администратором. В таблице productOrderTable храниться вся информация о заказанной продукции. Таблица productCartTable хранит в себе данные, которые пользователь занёс в корзину.

# Программная реализация приложения

Мобильное приложение реализовано для операционной системы Android 8.0 и выше. Данное приложение было разработано в специальной среде разработки –Android Studio.

Графическая структура классов представлена на рисунке 4.2.1.

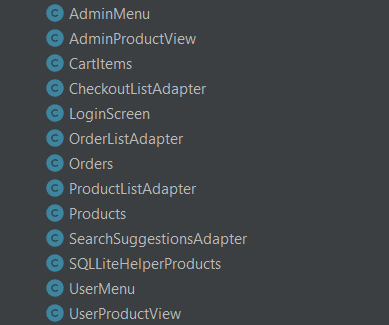


Рисунок 4.2.1 – структура классов

Подробное описание структуры:

1. AdminMenu, UserMenu, LoginScreen, AdminProductView, UserProductView содержат реализация основных Activity моего приложения.
2. SQLLiteHelperProducts содержит в себе всю реализация необходимой для работы с БД.
3. Orders, Products, CartItems являются моделями на таблицы БД с дополнительными методами.
4. CheckoutListAdapter, OrderListAdapter, ProductListAdapter, SearchSuggestionsAdapter - вспомогательные классы необходимые для отображения списков обьектов на Activity.

В каждой из Activities добавлен следующие флаги   
необходимые для защиты нашего приложения, которые не позволяют пользователю делать снимок экрана Рисунок 4.2.2



Рисунок 4.2.2 – установка флагов

Так же в приложение происходит работа с анимациями и жестами Рисунок 4.2.3



Рисунок 4.2.3 – переход к меню

Для использования жестов мы должны реализовать интерфейс GestureDetector.OnGestureListener, для обработчика свайпа вверх отвечает переопределённый метод onFling Рисунок 4.2.4

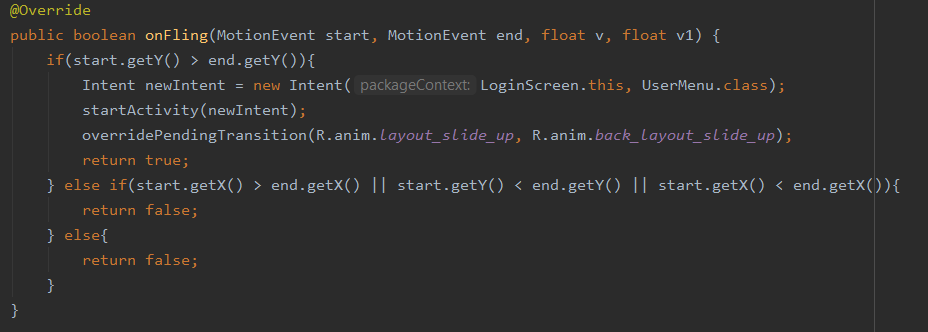


Рисунок 4.2.4 – обработчик свайпа вверх

Для анимации кнопки перехода к меню я использовал векторную анимацию рисунок 4.2.5 В которой указал 2 движущееся стрелки и их маршрут.

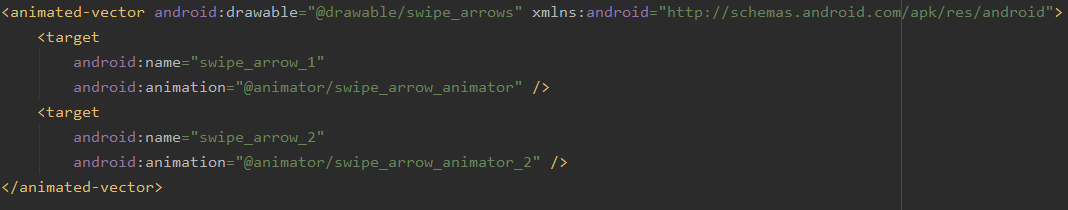


Рисунок 4.2.5 – реализация векторной анимации

# Руководство пользователя

## 5.1 Android приложение

Для начала необходимо установить приложение на мобильный устройство с API не меньше 21 (Android 4.1.1). Запустим наше приложение. Сразу после запуска мы увидим окно регистрации (Рисунок 5.1.1). Для авторизации необходимо нажать на кнопку “Авторизоваться”, после чего пользователю будет предоставлено диалоговое окно (Рисунок 5.1.2), авторизация происходит с помощью ввода логина и пароля, который хешируется с помощью алгоритма SHA-512 и сравнивается с хешем, хранимым в БД

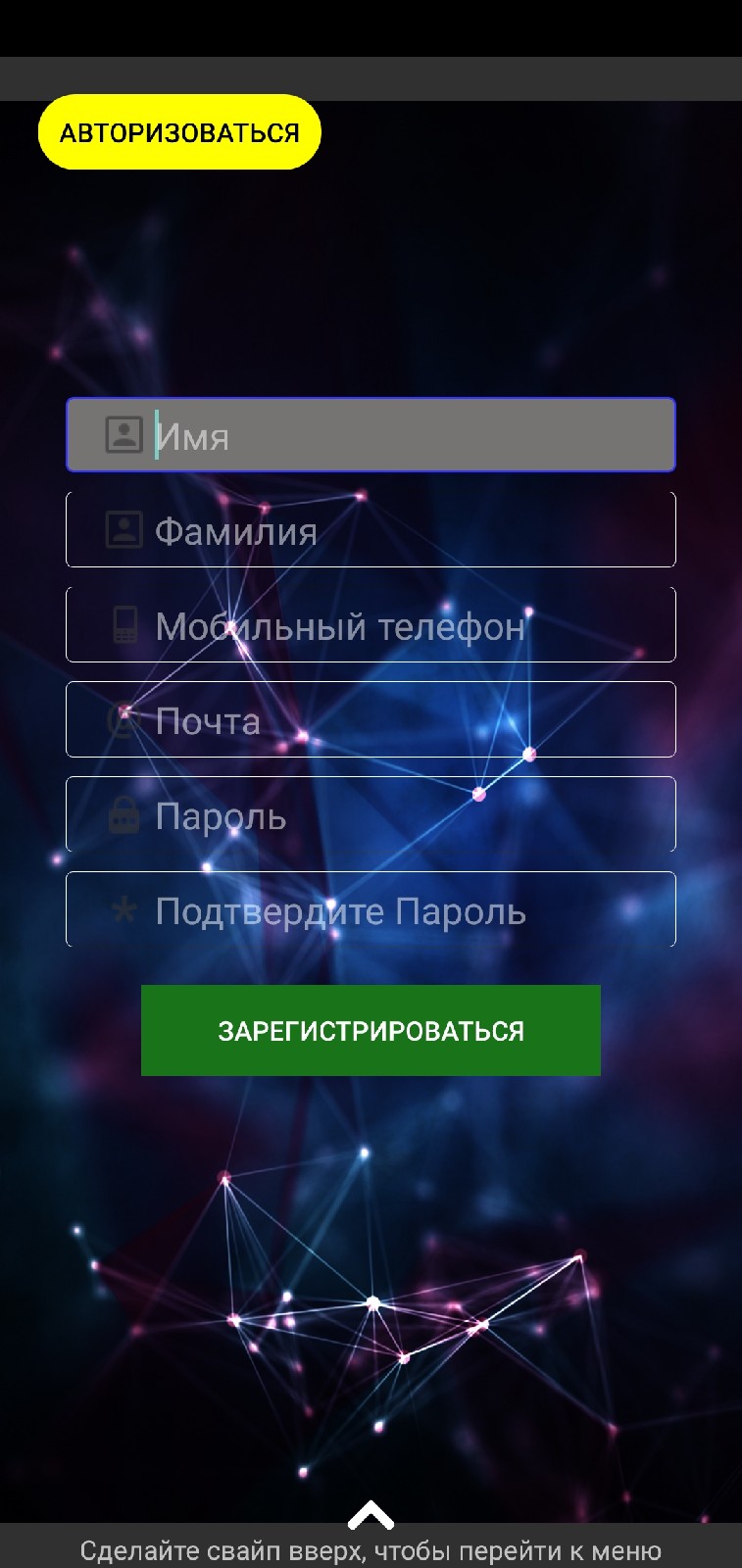


Рисунок 5.1.1 – окно авторизации и регистрации

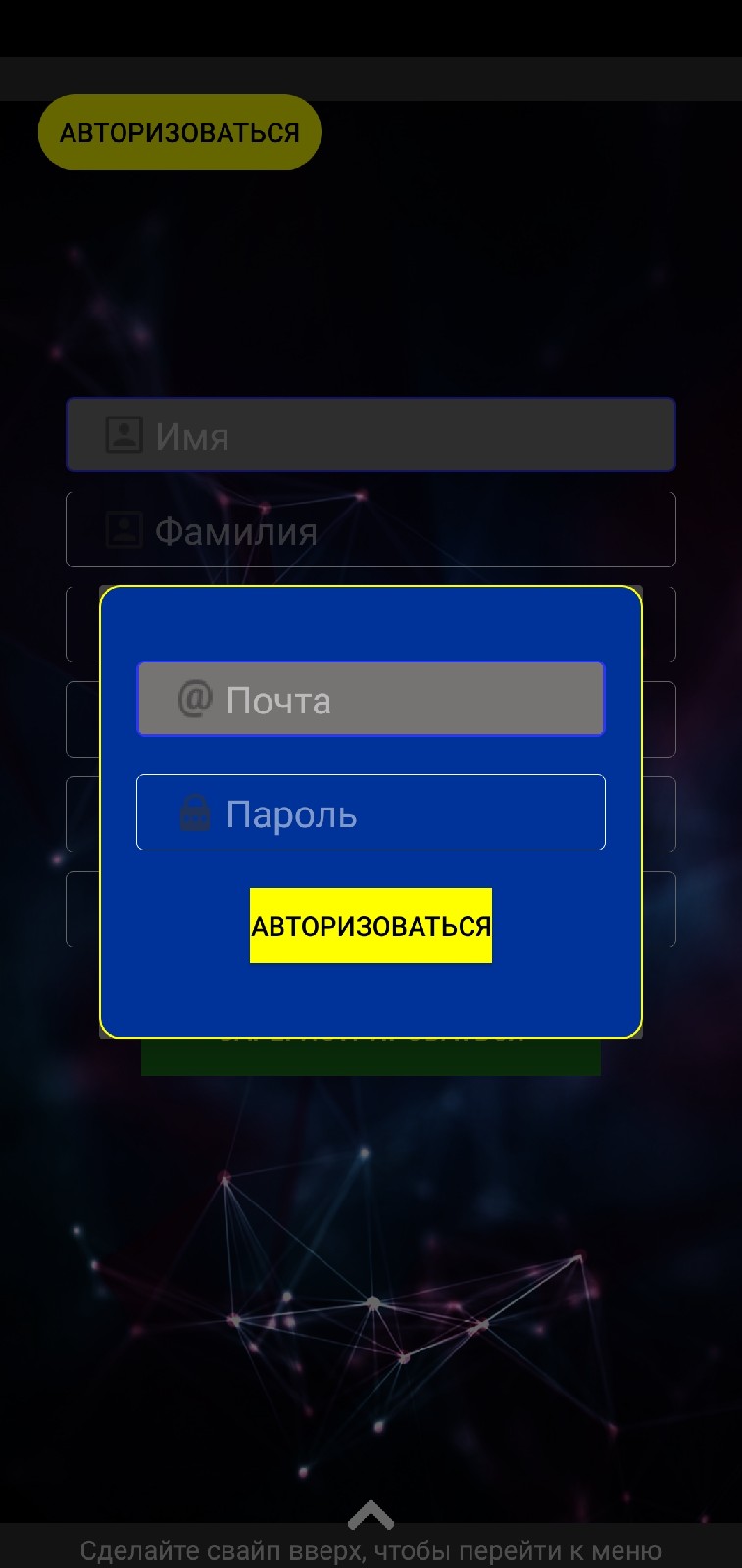


Рисунок 5.1.2 – диалоговое окно авторизации

После ввода данных (имя пользователя и пароль) мы можем видеть все еду, доступную для заказа на данный момент, жёлтая кнопка справа внизу является сортировкой по цене (Рисунок 5.1.3).

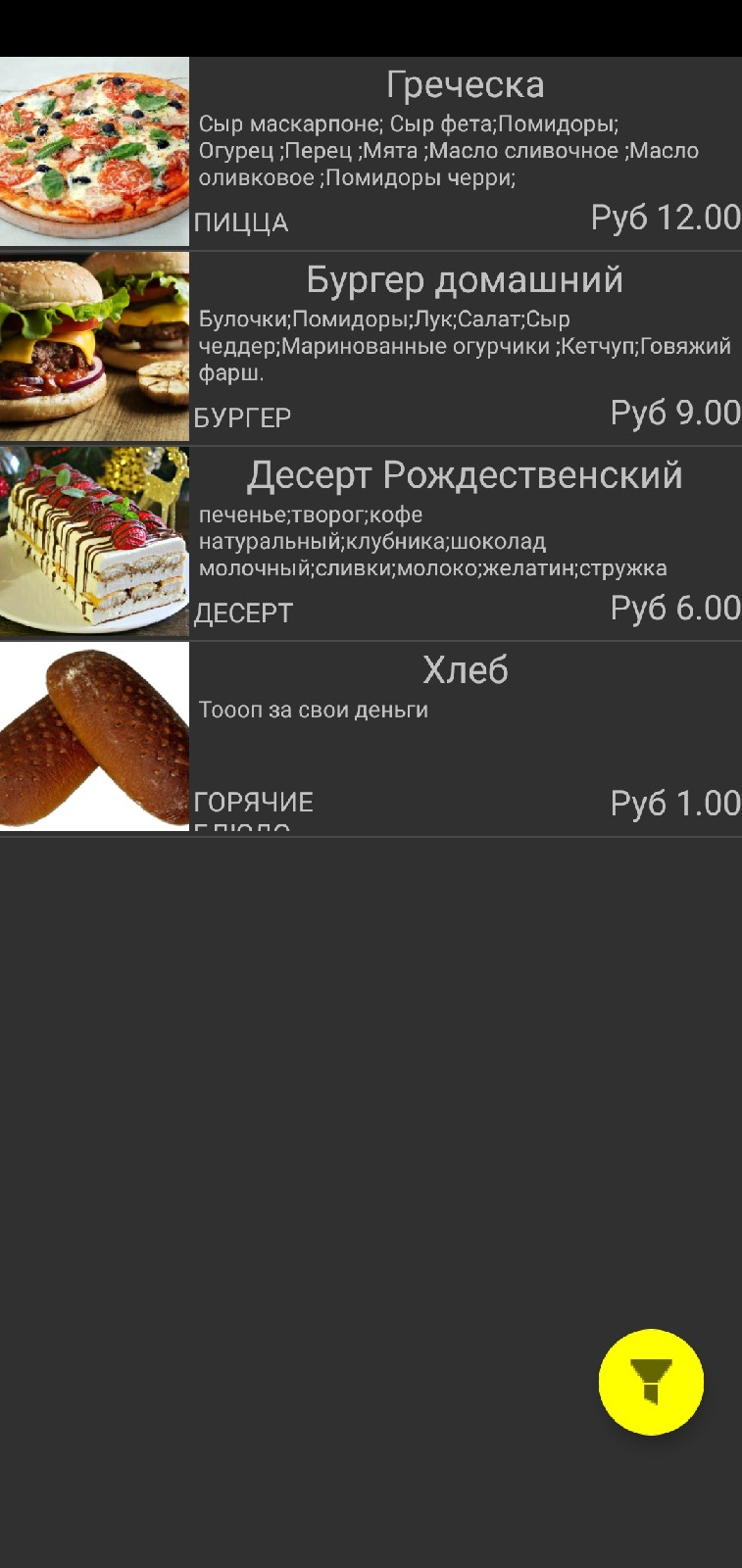
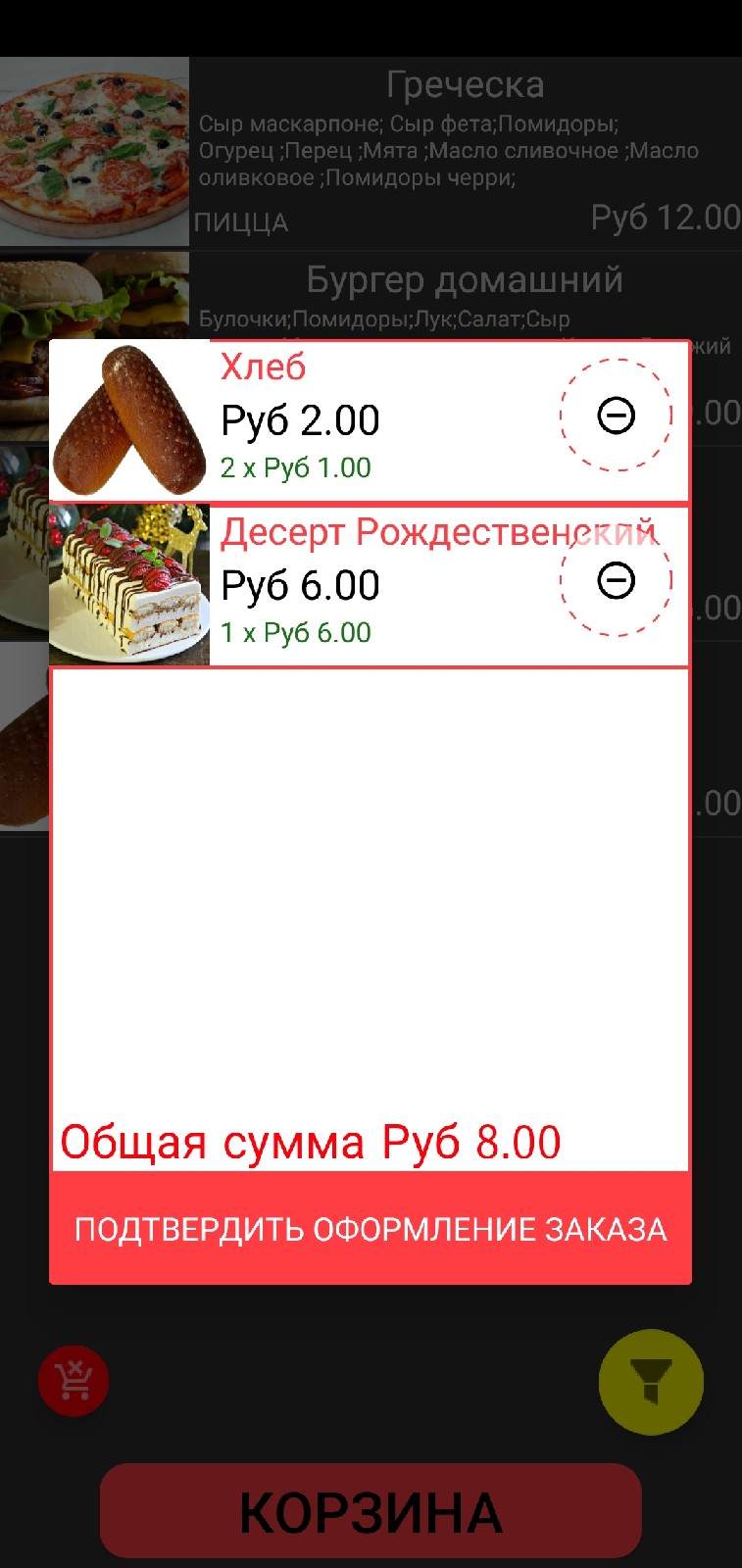


Рисунок 5.1.3 – меню доступной еды

После того, как пользователь попал на страницу с едой, он начинает добавлять её в карзину, которая выглядит следующим образом, так же пользователь может в 1 клик полностью её очистить, нажав на изображение в левом нижнем углу экрана, которое не видно пользователю, пока у него пустая корзина(Рисунок 5.1.4).



Риснок 5.1.4 – корзина с заказом пользователя

После авторизации администратор попадает на меню, схожее с меню пользователя, это видно по заднему плану, на рисунке можно увидеть интерфейс администратора при добавлении новой еды в меню, так же администратор имеет поле поиска продуктов по названию (Рисунок 5.1.5)

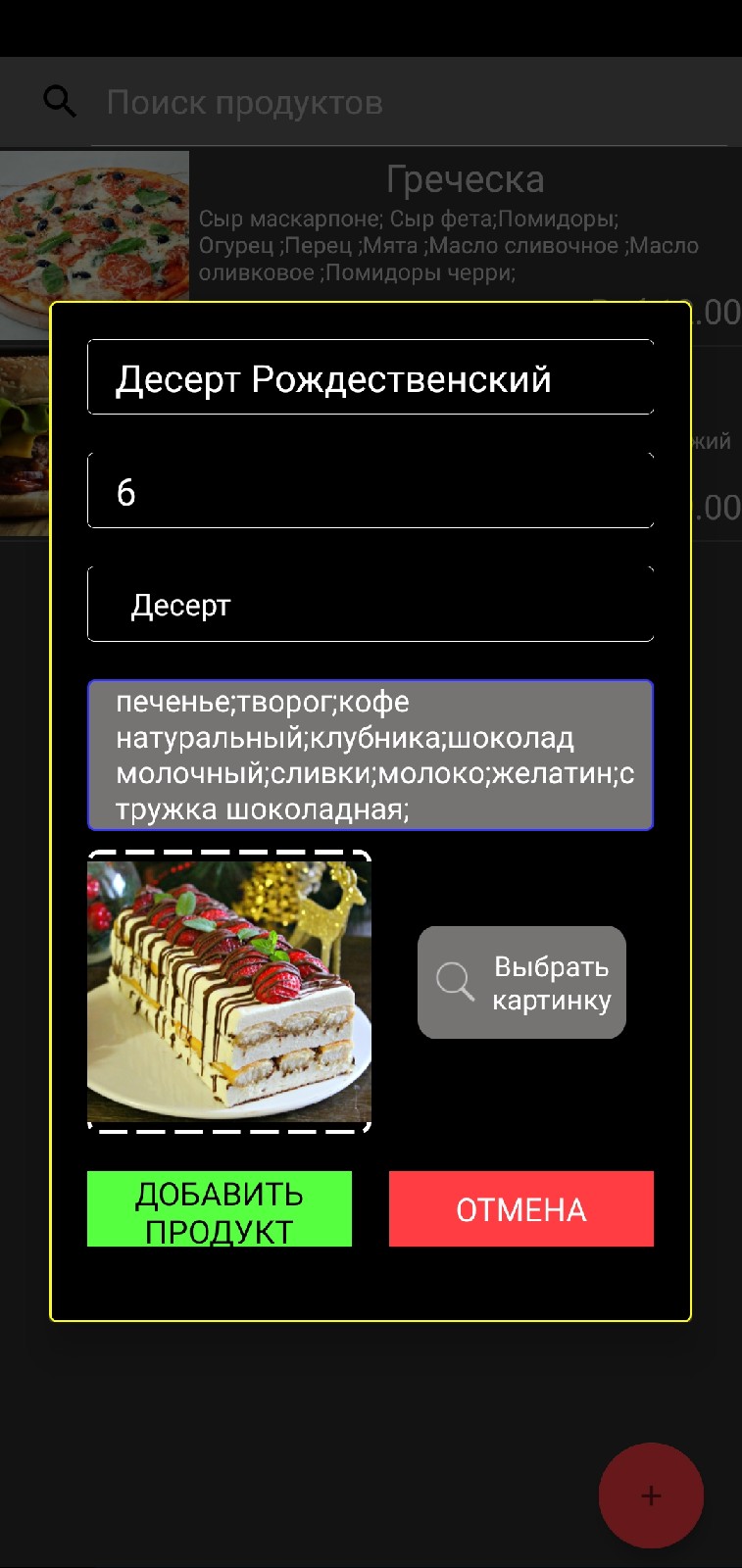


Рисунок 5.1.5 – Добавление новой еды

# Заключение

Решая поставленную задачу, я пришел к удовлетворительному результату. Таким образом, была достигнута цель, и было создано мобильное приложение «Ordering food».

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:

* Защищённое хранилище данных
* Авторизация на основе хэширования
* Невозможность создания скриншотов определённых экранов

# Список использованных источников

1. Хабрахабр [Электронный ресурс]. – Moxy – реализация MVP под Android с щепоткой магии – TM, 2006-2017. – Режим доступа: <https://habrahabr.ru/post/276189/> – Дата доступа: 01.11.2017.

2. Google Developers [Электронный ресурс]. – Интеграция Firebase в Android – Google. – Режим доступа: <https://codelabs.developers.google.com/codelabs/firebase-android-ru/index.html?index=..%2F..%2Flang-ru> – Дата доступа: 10.11.2017.

3. Fernandocejas [Электронный ресурс]. – Architecting Android… The clean way? – Fernando Cejas, 2012 - 2017. – Режим доступа: <https://fernandocejas.com/2014/09/03/architecting-android-the-clean-way/> – Дата доступа: 15.11.2017.

4. Horton J. Android Programming for Beginners/ Horton J. – Packt Publishing, 2015. – 698 с.

# Приложение А

**UserMenu.java**

**package com.cmbpizza.razor.golubev;**

**import android.app.ActivityOptions;**

**import android.app.Dialog;**

**import android.app.SearchManager;**

**import android.content.Intent;**

**import android.database.Cursor;**

**import android.database.DataSetObserver;**

**import android.os.Bundle;**

**import android.app.Activity;**

**import android.provider.BaseColumns;**

**import android.support.design.widget.FloatingActionButton;**

**import android.util.Log;**

**import android.util.Pair;**

**import android.util.TypedValue;**

**import android.view.View;**

**import android.view.ViewStub;**

**import android.view.WindowManager;**

**import android.widget.AbsListView;**

**import android.widget.AdapterView;**

**import android.widget.Button;**

**import android.widget.ListView;**

**import android.widget.RelativeLayout;**

**import android.widget.SearchView;**

**import android.widget.TextView;**

**import android.widget.Toast;**

**import java.util.ArrayList;**

**import java.util.Collections;**

**import java.util.Random;**

**public class UserMenu extends Activity implements View.OnClickListener, SearchView.OnSuggestionListener, SearchView.OnCloseListener, AdapterView.OnItemClickListener {**

**private static final String[] COLUMNS = {**

**BaseColumns.\_ID,**

**SearchManager.SUGGEST\_COLUMN\_TEXT\_1,**

**SearchManager.SUGGEST\_COLUMN\_ICON\_1**

**};**

**public String memberId;**

**public static boolean isUserLoggedIn = false, sort = true;**

**private static SQLLiteHelperProducts sqLiteHelper;**

**public int cartCount = 0;**

**public String randomID;**

**public Dialog checkoutDialog;**

**FloatingActionButton FilterFab, FilterCheapFab, FilterExpensiveFab, ClearCartFab;**

**TextView CartNetTotal;**

**Button CheckoutButton, ConfirmCheckoutButton;**

**SearchView UserSearchView;**

**ViewStub ListViewStub;**

**ListView ProductListView, CheckoutItemsListView;**

**ArrayList<CartItems> CartItemsList;**

**ArrayList<Products> ProductList;**

**ArrayList<Integer> ProductIDList;**

**CheckoutListAdapter CheckoutItemsListAdapter;**

**ProductListAdapter ListAdapter;**

**private SearchSuggestionsAdapter mSuggestionsAdapter;**

**@Override**

**protected void onCreate(Bundle savedInstanceState) {**

**getWindow().setFlags(WindowManager.LayoutParams.FLAG\_SECURE,**

**WindowManager.LayoutParams.FLAG\_SECURE);**

**super.onCreate(savedInstanceState);**

**setContentView(R.layout.activity\_user\_menu);**

**initializeComponents();**

**initializeViewModes();**

**initializeListeners();**

**checkLoginStatus();**

**checkView();**

**generateRandomID();**

**sqlLiteDB();**

**getAllData();**

**getCartCounter();**

**checkCartCount();**

**clearCart();**

**}**

**@Override**

**protected void onStart() {**

**super.onStart();**

**getCartCounter();**

**checkLoginStatus();**

**checkCartCount();**

**getAllData();**

**}**

**private void getCartCounter() {**

**cartCount = sqLiteHelper.getCartCount(randomID);**

**}**

**private void checkLoginStatus() {**

**if(getIntent().getExtras() == null){**

**isUserLoggedIn = false;**

**} else {**

**isUserLoggedIn = true;**

**memberId = getIntent().getExtras().get("userId").toString();**

**}**

**}**

**private void initializeComponents() {**

**ListViewStub = findViewById(R.id.userListViewStub);**

**checkoutDialog = new Dialog(UserMenu.this);**

**checkoutDialog.setContentView(R.layout.user\_checkout\_dialog);**

**CartNetTotal = checkoutDialog.findViewById(R.id.txtNetTotal);**

**CheckoutItemsListView = checkoutDialog.findViewById(R.id.userCheckoutList);**

**CartItemsList = new ArrayList<>();**

**CheckoutItemsListAdapter = new CheckoutListAdapter(checkoutDialog.getContext(), R.layout.user\_checkout\_product\_list\_view, CartItemsList);**

**CheckoutItemsListView.setAdapter(CheckoutItemsListAdapter);**

**ConfirmCheckoutButton = checkoutDialog.findViewById(R.id.btnConfirmCheckout);**

**FilterFab = findViewById(R.id.filter\_fab);**

**FilterCheapFab = findViewById(R.id.filter\_cheap\_fab);**

**FilterExpensiveFab = findViewById(R.id.filter\_expensive\_fab);**

**ClearCartFab = findViewById(R.id.clear\_cart\_fab);**

**CheckoutButton = findViewById(R.id.userCheckoutButton);**

**ProductList = new ArrayList<>();**

**ProductIDList = new ArrayList<>();**

**ListAdapter = new ProductListAdapter(UserMenu.this, R.layout.user\_product\_items\_list\_view, ProductList);**

**}**

**private void initializeListeners() {**

**FilterFab.setOnClickListener(this);**

**FilterCheapFab.setOnClickListener(this);**

**FilterExpensiveFab.setOnClickListener(this);**

**ProductListView.setOnItemClickListener(this);**

**CheckoutButton.setOnClickListener(this);**

**ConfirmCheckoutButton.setOnClickListener(this);**

**ClearCartFab.setOnClickListener(this);**

**ProductListView.setOnScrollListener(new AbsListView.OnScrollListener() {**

**private int currentVisibleItemCount;**

**private int currentScrollState;**

**private int currentFirstVisibleItem;**

**private int totalItem;**

**@Override**

**public void onScrollStateChanged(AbsListView absListView, int scrollState) {**

**this.currentScrollState = scrollState;**

**if(currentScrollState == SCROLL\_STATE\_FLING){**

**UserSearchView.setAlpha(0.0f);**

**} else if(currentScrollState == SCROLL\_STATE\_TOUCH\_SCROLL){**

**UserSearchView.setAlpha(0.5f);**

**} else if(currentScrollState == SCROLL\_STATE\_IDLE){**

**UserSearchView.setAlpha(1);**

**}**

**}**

**@Override**

**public void onScroll(AbsListView absListView, int firstVisibleItem, int visibleItemCount, int totalItemCount) {**

**this.currentFirstVisibleItem = firstVisibleItem;**

**this.currentVisibleItemCount = visibleItemCount;**

**this.totalItem = totalItemCount;**

**}**

**});**

**CheckoutItemsListAdapter.registerDataSetObserver(new DataSetObserver() {**

**@Override**

**public void onChanged() {**

**super.onChanged();**

**getCheckoutNetTotal();**

**getCartCounter();**

**checkCartCount();**

**}**

**});**

**}**

**private void initializeViewModes() {**

**ProductListView = (ListView) ListViewStub.inflate();**

**}**

**@Override**

**public void onClick(View view) {**

**switch (view.getId()) {**

**case R.id.filter\_fab:**

**sortProductList();**

**break;**

**case R.id.clear\_cart\_fab:**

**clearCart();**

**checkCartCount();**

**break;**

**case R.id.userCheckoutButton:**

**checkoutDialog();**

**break;**

**case R.id.btnConfirmCheckout:**

**createOrder();**

**break;**

**}**

**}**

**@Override**

**public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {**

**TextView ListProductId = view.findViewById(R.id.productId);**

**Intent newIntent = new Intent(UserMenu.this, UserProductView.class);**

**Pair<View, String> pairImage = Pair.create(view.findViewById(R.id.productImage), "productImage");**

**Pair<View, String> pairTitle = Pair.create(view.findViewById(R.id.productTitle), "productTitle");**

**Pair<View, String> pairDescription = Pair.create(view.findViewById(R.id.productDescription), "productDescription");**

**Pair<View, String> pairCategory = Pair.create(view.findViewById(R.id.productCategory), "productCategory");**

**Pair<View, String> pairPrice = Pair.create(view.findViewById(R.id.productPrice), "productPrice");**

**ActivityOptions activityOptions = ActivityOptions.makeSceneTransitionAnimation(**

**UserMenu.this,**

**pairImage,**

**pairTitle,**

**pairDescription,**

**pairCategory,**

**pairPrice);**

**newIntent.putExtra("userId", memberId);**

**newIntent.putExtra("productId", ListProductId.getText().toString());**

**newIntent.putExtra("cartId", randomID);**

**startActivity(newIntent, activityOptions.toBundle());**

**}**

**private void checkView() {**

**ListViewStub.setVisibility(View.VISIBLE);**

**ProductListView.setAdapter(ListAdapter);**

**}**

**private void sqlLiteDB() {**

**sqLiteHelper = new SQLLiteHelperProducts(UserMenu.this, "ProductDB", null, 1);**

**sqLiteHelper.dataQuery("CREATE TABLE IF NOT EXISTS productTable (productId INTEGER PRIMARY KEY AUTOINCREMENT, productTitle VARCHAR, productCategory VARCHAR, productPrice INTEGER, productDescription VARCHAR, productImage BLOB)");**

**sqLiteHelper.dataQuery("CREATE TABLE IF NOT EXISTS productCartTable (id INTEGER PRIMARY KEY AUTOINCREMENT, cartId VARCHAR, productId INTEGER, productQuantity INTEGER)");**

**sqLiteHelper.dataQuery("CREATE TABLE IF NOT EXISTS productOrderTable (orderId INTEGER PRIMARY KEY, memberId INTEGER, idList VARCHAR, quantityList VARCHAR, totalPrice INTEGER, orderStatus INTEGER)");**

**}**

**private void clearCart() {**

**sqLiteHelper.dataQuery("DELETE FROM productCartTable");**

**sqLiteHelper.dataQuery("DELETE FROM sqlite\_sequence WHERE name='productCartTable'");**

**cartCount = 0;**

**Log.d("sql", "cart cleared");**

**}**

**private void getAllData() {**

**ProductIDList.clear();**

**ProductList.clear();**

**for (Products product : sqLiteHelper.getAllData()) {**

**ProductIDList.add(product.getProductId());**

**ProductList.add(new Products(product.getProductId(), product.getProductTitle(), product.getProductCategory(), product.getProductPrice(), product.getProductDescription(), product.getProductImage()));**

**}**

**ListAdapter.notifyDataSetChanged();**

**}**

**@Override**

**public boolean onClose() {**

**getAllData();**

**return true;**

**}**

**@Override**

**public boolean onSuggestionSelect(int i) {**

**return false;**

**}**

**@Override**

**public boolean onSuggestionClick(int i) {**

**Cursor c = (Cursor) mSuggestionsAdapter.getItem(i);**

**String query = c.getString(c.getColumnIndex(SearchManager.SUGGEST\_COLUMN\_TEXT\_1));**

**UserSearchView.setQuery(query, true);**

**return true;**

**}**

**private void checkCartCount() {**

**RelativeLayout relativeLayoutButton, relativeLayoutBadge;**

**relativeLayoutButton = findViewById(R.id.layoutRelativeCheckoutButton);**

**relativeLayoutBadge = findViewById(R.id.layoutRelativeButtonBadge);**

**RelativeLayout.LayoutParams layoutParams = new RelativeLayout.LayoutParams(RelativeLayout.LayoutParams.MATCH\_PARENT, RelativeLayout.LayoutParams.WRAP\_CONTENT);**

**if (cartCount > 0) {**

**ClearCartFab.setVisibility(View.VISIBLE);**

**ClearCartFab.setClickable(true);**

**layoutParams.setMargins(layoutParams.getMarginStart(), 0, layoutParams.getMarginEnd(), 0);**

**layoutParams.addRule(RelativeLayout.ALIGN\_PARENT\_BOTTOM, R.layout.activity\_user\_menu);**

**relativeLayoutButton.setLayoutParams(layoutParams);**

**RelativeLayout.LayoutParams layoutParamsBadge = new RelativeLayout.LayoutParams(RelativeLayout.LayoutParams.WRAP\_CONTENT, RelativeLayout.LayoutParams.WRAP\_CONTENT);**

**layoutParamsBadge.setMargins(0, 0, 0, 0);**

**layoutParamsBadge.addRule(RelativeLayout.ALIGN\_START, R.id.layoutLinearCheckoutButton);**

**layoutParamsBadge.addRule(RelativeLayout.ALIGN\_TOP, R.id.layoutLinearCheckoutButton);**

**relativeLayoutBadge.setLayoutParams(layoutParamsBadge);//the layout params is used to set the margin and the alignment**

**relativeLayoutBadge.setVisibility(View.VISIBLE);**

**} else {**

**ClearCartFab.setVisibility(View.GONE);**

**ClearCartFab.setClickable(false);**

**int value = (int) TypedValue.applyDimension(TypedValue.COMPLEX\_UNIT\_DIP, -60.0f, getResources().getDisplayMetrics());**

**layoutParams.setMargins(layoutParams.getMarginStart(), 0, layoutParams.getMarginEnd(), value);**

**layoutParams.addRule(RelativeLayout.ALIGN\_PARENT\_BOTTOM, R.layout.activity\_user\_menu);**

**relativeLayoutButton.setLayoutParams(layoutParams);**

**}**

**}**

**private void generateRandomID() {**

**String randomNumbers = null;**

**StringBuilder stringBuilder = new StringBuilder();**

**for (int i = 0; i <= 10; i++) {**

**Random random = new Random();**

**stringBuilder.append(random.nextInt(9));**

**randomNumbers = stringBuilder.toString();**

**}**

**randomID = randomNumbers;**

**}**

**@Override**

**public void onBackPressed() {**

**sqLiteHelper.dataQuery("DELETE FROM productCartTable");**

**cartCount = 0;**

**finish();**

**}**

**private void checkoutDialog() {**

**getCheckoutDetails();**

**checkoutDialog.setCancelable(true);**

**checkoutDialog.setCanceledOnTouchOutside(true);**

**checkoutDialog.show();**

**}**

**private void getCheckoutDetails() {**

**CartItemsList.clear();**

**int productNetTotalPrice = 0;**

**for (int i = 1; i <= sqLiteHelper.getCartCount(randomID); i++) {**

**String cartId = randomID;**

**int productId = sqLiteHelper.getCartProductId(i);**

**int productQuantity = sqLiteHelper.getCartProductQuantity(i);**

**int productPrice = sqLiteHelper.getProductPrice(productId);**

**productNetTotalPrice += (productPrice \* productQuantity);**

**CartItemsList.add(new CartItems(cartId, productId, productQuantity));**

**}**

**CartNetTotal.setText(getResources().getString(R.string.txtNetTotalPricePrefix, productNetTotalPrice));**

**CheckoutItemsListAdapter.notifyDataSetChanged();**

**}**

**private void getCheckoutNetTotal(){**

**int productNetTotalPrice = 0;**

**for (int i = 1; i <= sqLiteHelper.getCartCount(randomID); i++) {**

**int productId = sqLiteHelper.getCartProductId(i);**

**int productQuantity = sqLiteHelper.getCartProductQuantity(i);**

**int productPrice = sqLiteHelper.getProductPrice(productId);**

**productNetTotalPrice += (productPrice \* productQuantity);**

**}**

**CartNetTotal.setText(getResources().getString(R.string.txtNetTotalPricePrefix, productNetTotalPrice));**

**}**

**private void createOrder(){**

**if(cartCount >= 1){**

**int arrayLength = CartItemsList.size();**

**String[] productIdList = new String[arrayLength];**

**String[] productQuantityList = new String[arrayLength];**

**int productNetTotalPrice = 0;**

**for(int i = 0; i < arrayLength; i++){**

**int productId = CartItemsList.get(i).getProductId();**

**int productQuantity = CartItemsList.get(i).getProductQuantity();**

**int productPrice = sqLiteHelper.getProductPrice(productId);**

**productIdList[i] = String.valueOf(productId);**

**productQuantityList[i] = String.valueOf(productQuantity);**

**productNetTotalPrice += (productPrice \* productQuantity);**

**}**

**sqLiteHelper.createOrder(randomID, memberId, productIdList, productQuantityList, productNetTotalPrice, 0);**

**Toast.makeText(this, "Заказ номер: " + randomID + " отправлено на обработку!", Toast.LENGTH\_SHORT).show();**

**} else {**

**Toast.makeText(this, "Корзина пуста", Toast.LENGTH\_SHORT).show();**

**}**

**checkoutDialog.cancel();**

**}**

**private void sortProductList(){**

**Collections.sort(ProductList);**

**if(sort){**

**Collections.reverse(ProductList);**

**sort = !sort;**

**}else{**

**sort = !sort;**

**}**

**ListAdapter.notifyDataSetChanged();**

**}**

**}**