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 «iCatalog»

Выполнил студент Корольчук Владислав Александрович

(Ф.И.О.)

Руководитель проекта асс. 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 асс. Северинчик Н.А.

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

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

Нормоконтроллер асс. Северинчик Н.А.

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

Курсовой проект защищен с оценкой

Минск 2020

Реферат

Курсовой проект содержит 23 страницы пояснительной записки, 10 рисунков, 4 источника литературы, 1 приложение.

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ “КАТАЛОГ ТОВАРОВ”.

Целью данного курсового проекта является создание мобильного приложения “Каталог товаров”.

Пояснительная записка состоит из введения, пяти разделов и заключения.

Во введении представлена общая информация о текущем состоянии в сфере рассматриваемой проблемы.

В первом раздели рассмотрена постановка задачи. Определены основные программные средства, фреймворки, а также архитектурный стиль разработки.

Во втором разделе рассмотрена архитектура приложения.

В третьем разделе спроектирована схема базы данных приложения.

В четвертом разделе представлена файловая структура приложения и используемые зависимости.

В пятом разделе представлено руководство пользователя.

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

Содержание

[ВВЕДЕНИЕ 5](#_Toc499688344)

[1 ПОСТАНОВКА ЗАДАЧИ 6](#_Toc499688345)

[2 АРХИТЕКТУРА ПРИЛОЖЕНИЯ 7](#_Toc499688346)

[3 СХЕМА БАЗЫ ДАННЫХ 8](#_Toc499688347)

[4 РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ 9](#_Toc499688348)

[5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 10](#_Toc499688349)

[ЗАКЛЮЧЕНИЕ 14](#_Toc499688350)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 15](#_Toc499688351)

[ПРИЛОЖЕНИЕ 16](#_Toc499688352)

# Введение

Время – это то, чего нам всегда не хватает. Требование экономии времени приводит к тому, что мы автоматически отдаём приоритет быстрым решениям. Одним из таких решений являются интернет-магазины. Однако современный человек мобилен, он то и дело срывается с места. Таким образом мы подходим к необходимости переноса интернет-торговли в мобильные устройства.

Целью курсового проекта является разработка мобильного приложения «Каталог товаров».

В разделе «Постановка задачи» описаны основные цели и задачи, которые были поставлены на этапе разработки данного курсового проекта. Кроме того, данный раздел описывает технологии, с помощью которых эти цели и задачи будут достигнуты.

Раздел «Архитектура приложения» дает представление о конечном продукте, полученном в результате выполнения курсовой работы. Этот раздел описывает принцип связи и работы распределенного приложения.

«Схема базы данных» – это раздел который предоставляет описание структуры мобильной локальной базы.

В разделе «Программная реализация приложения» описываются основные технологии, которые были использованы в данной работе. Позволяет понять взаимодействие сервера и мобильного устройства.

Последний раздел «Руководство пользователя» описывает взаимодействия пользователя с мобильным приложением. Дает краткие инструкции по эксплуатации ПО.

# 1 Постановка задачи

Необходимо разработать приложение для мобильных устройств, которые работают под управлением операционной системы Android. Приложение должно представлять из себя каталог товаров.

В качестве базы данных должна выступать Firebase. Авторизация также должна быть реализована через Firebase.

Логическая схема вариантов работы приложения представлена на рисунке 1.1.

Рис. 1.1 – Жизненный цикл мобильного приложения

При разработке приложения следует использовать язык программирования Java. В качестве основного паттерна должен использоваться MVP. Для удобства реализации данного паттерна необходимо использовать библиотеку Moxy.

# 2 Архитектура приложения

MVP — шаблон проектирования пользовательского интерфейса, который был разработан для облегчения автоматического модульного тестирования и улучшения разделения ответственности в презентационной логике (отделения логики от отображения):

* Модель (англ. Model) — хранит в себе всю бизнес-логику, при необходимости получает данные из хранилища.
* Вид(англ. View) — реализует отображение данных (из Модели), обращается к Presenter за обновлениями.
* Представитель (англ. Presenter) — реализует взаимодействие между моделью и представлением.

Обычно экземпляр Представления создаёт экземпляр Presenterʼа, передавая ему ссылку на себя. При этом Presenter работает с Представлением в абстрактном виде, через его интерфейс. Когда вызывается событие Представления, оно вызывает конкретный метод Presenterʼа, не имеющего ни параметров, ни возвращаемого значения. Presenter получает необходимые для работы метода данные о состоянии пользовательского интерфейса через интерфейс Представления и через него же передаёт в Представление данные из Модели и другие результаты своей работы [1].

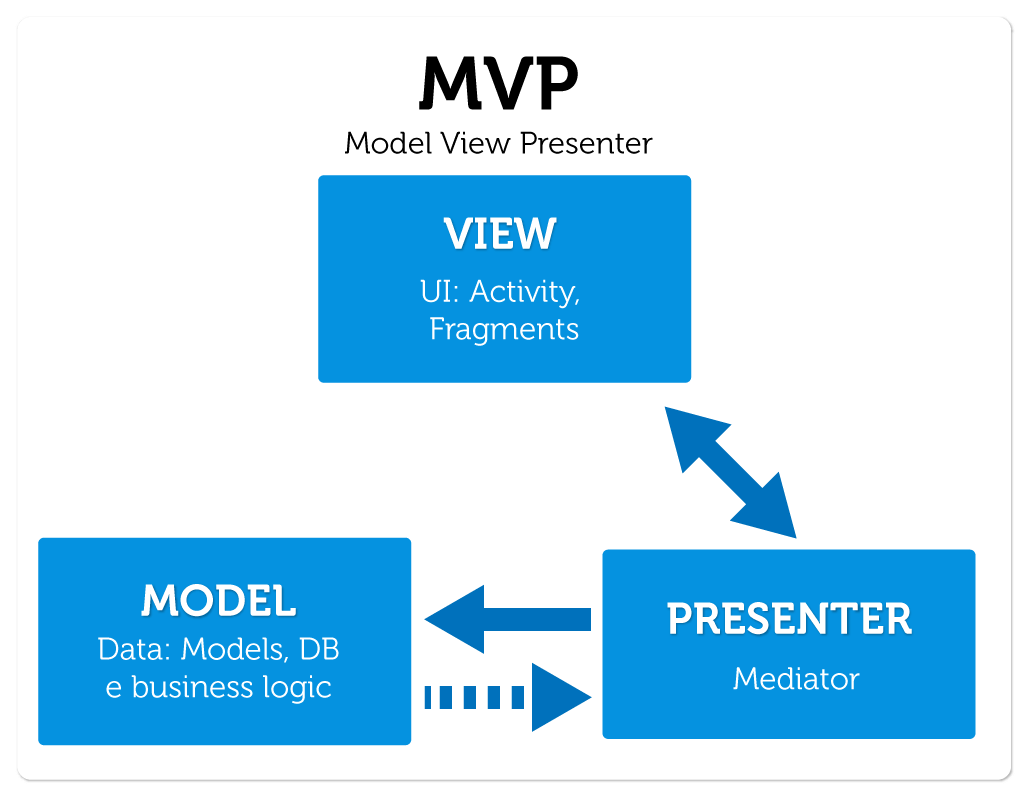


Рис. 2.1 – Схема паттерна MVP

# 3 Схема базы данных

В качестве мобильной СУБД используется Firebase. Firebase предоставляет облачную NoSQL БД для real-time приложений как сервис. Данный сервис предоставляет API для разработчиков, позволяющий синхронизировать данные приложения между клиентами и хранить их в облаке FireBase. Компания также учла возможность интеграции с Android, iOS, JavaScript, Java, Objective-C и Node.js приложениями [2].

Структура Firebase напоминает дерево, где ветки – это узлы, а листья – данные. Ключи – это уникальный номер узла и полей данных. Поле данных это строка с ключом и самими данными.

В данном приложении два основных узла. В узле “items” храниться информация о товарах. В узле “users” – содержимое корзины пользователей, а также флаг “isCartEmpty”.



Рис. 3.1 – Структура базы данных приложения

# 4 Реализация приложения

Файловая структура приложения имеет вид:

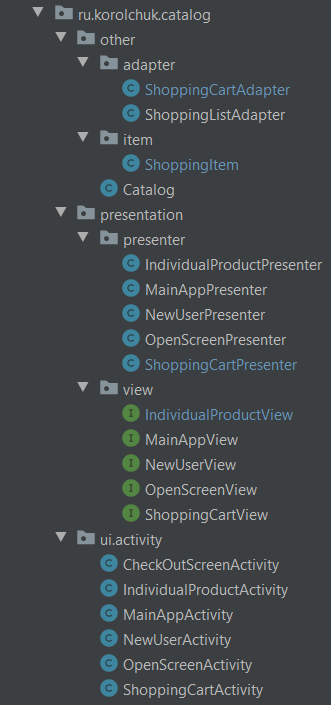


Рис. 4.1 – Структура приложения Catalog

Подробное описание структуры:

* Adapter содержит реализацию классов-адаптеров, которые используются для отображения списков
* Item содержит пользовательские модели
* Presentation содержит реализацию классов, которые являются частью паттерна MVP
* UI содержит реализацию классов, которые отвечают за отображение пользовательского интерфейса [3].

Dependencies, которые использует мобильное приложение для сборки:

# 

Рис. 4.2 – Список зависимостей приложения

# 5 Руководство пользователя

Первым шагом для пользователя является авторизация. Пользователь должен ввести e-mail и пароль.

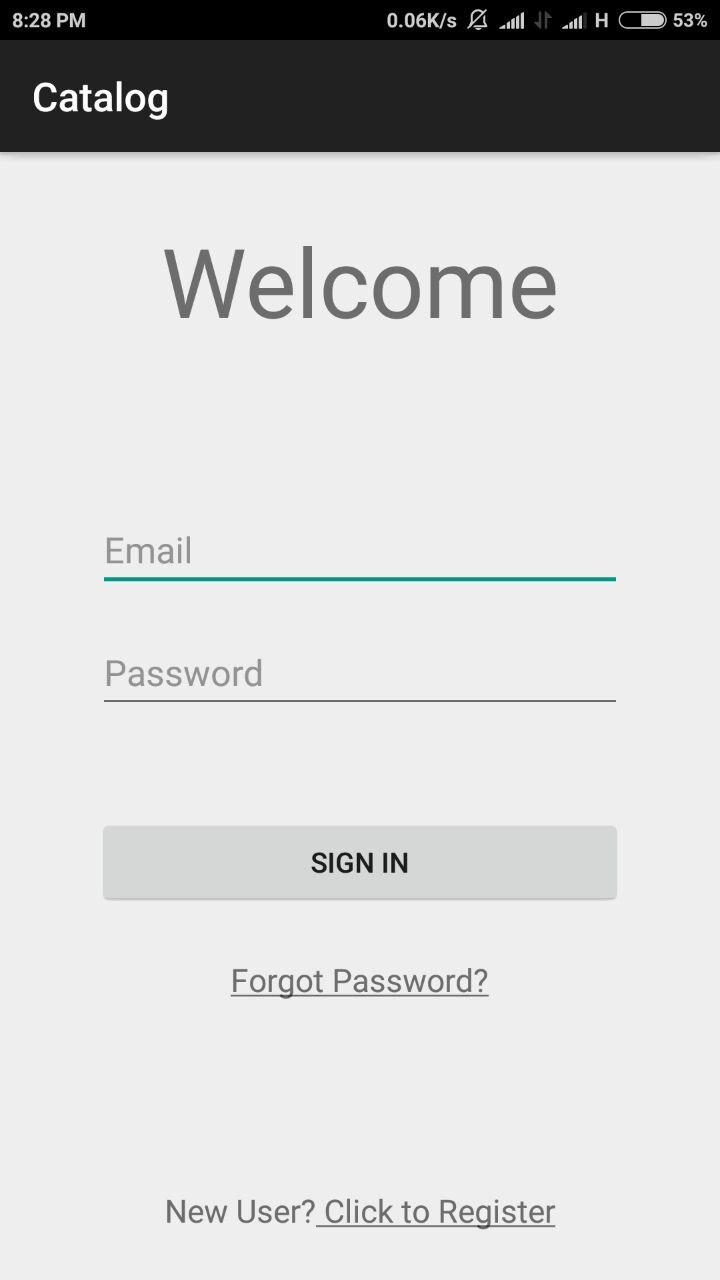


Рис. 5.1 – Окно авторизации и разрешения доступа

Далее нажать кнопку “SIGN IN”. После чего, пользователь будет перенаправлен в каталог.

Если же пользователь не имеет логина и пароля, ему необходимо зарегистрироваться. Для этого нужно нажать “Click to Register”.

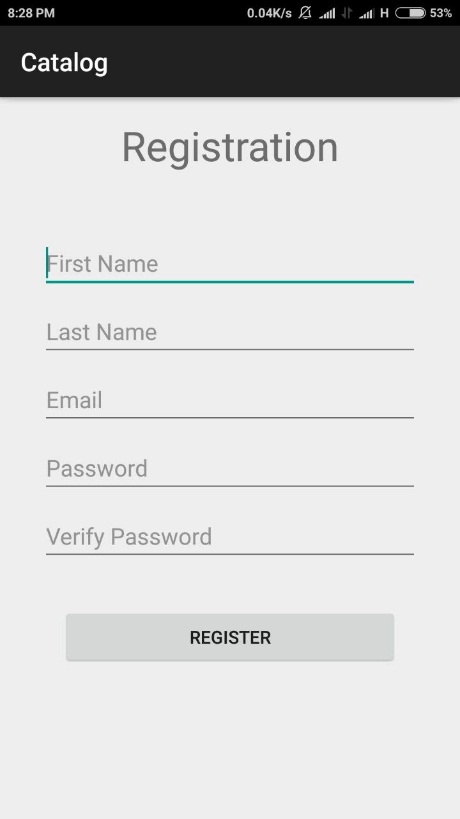


Рис. 5.2 – Окно регистрации нового пользователя

Пользователь должен заполнить все поля, после чего нажать “Register”. После подтверждения электронной почты он сможет воспользоваться своим аккаунтом.

Успешно авторизовавшись пользователь увидит список товаров.

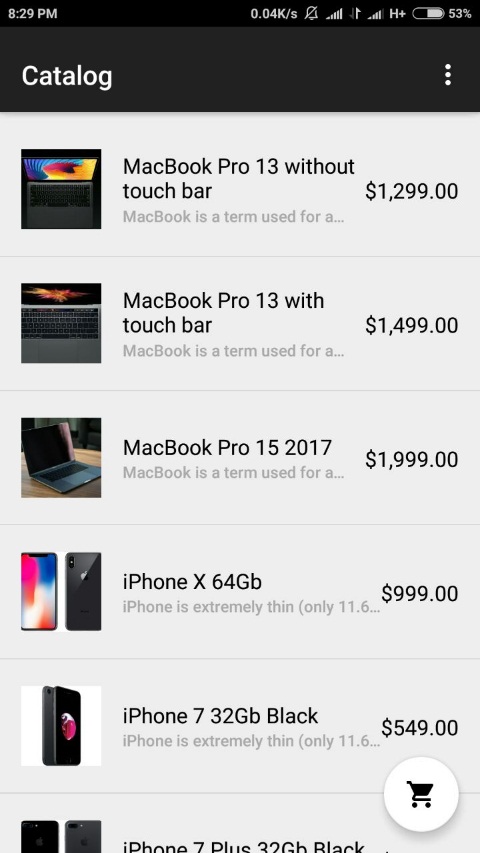


Рис. 5.3 – Главная страница. Список товаров

Нажав на элемент списка, пользователь попадает на страницу товара. Здесь находиться фотография, стоимость и описание товара. Товар можно добавить в корзину, выбрав необходимое количество. Далее можно перейти в корзину либо продолжить покупки.

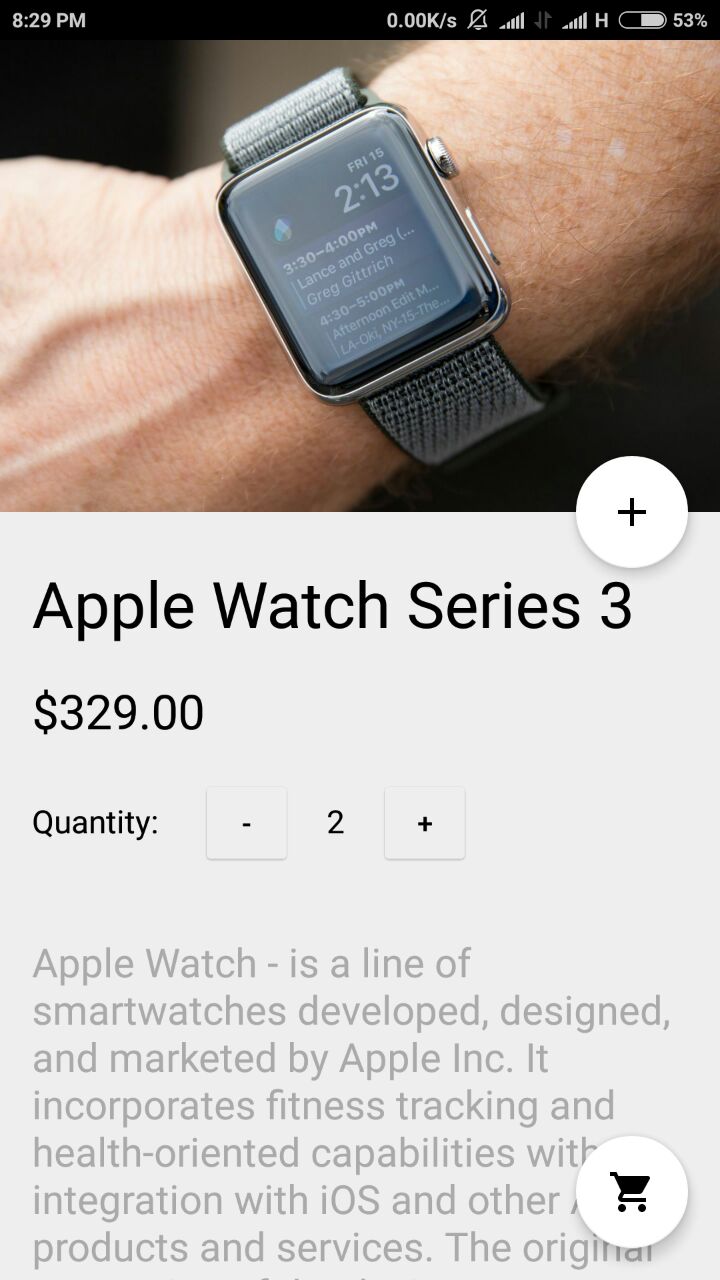


Рис. 5.4 – Страница товара

Далее можно перейти в корзину либо продолжить покупки.

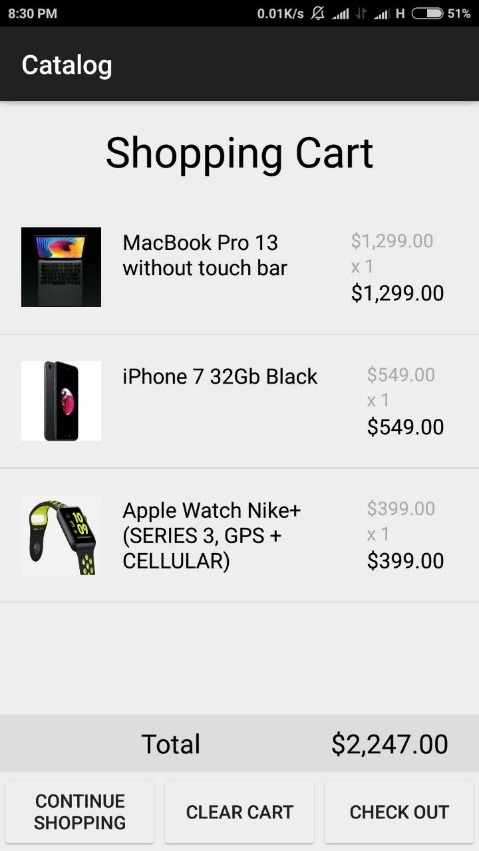


Рис. 5.5 – Корзина

# Заключение

Целью данной курсовой работы является разработка мобильного приложения “iCatalog”, которое позволяет совершать покупки в сети интернет при помощи своего мобильного устройства.

Для взаимодействия с серверной частью приложения используется набор сервисов Firebase, таких как Simple Login и Realtime Database.

При создании приложения были применены актуальные на текущий момент времени средства разработки, которые позволили реализовать программу.

# Список используемой литературы

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

(обязательное)

Приложение А

Листинги мобильного приложения

***MainAppActivity.java:***

age ru.korolchuk.catalog.ui.activity;  
  
import android.content.Intent;  
import android.os.Bundle;  
import android.os.Handler;  
import android.support.design.widget.FloatingActionButton;  
import android.support.design.widget.Snackbar;  
import android.support.v7.widget.Toolbar;  
import android.view.Menu;  
import android.view.MenuItem;  
import android.view.View;  
import android.view.WindowManager;  
import android.widget.AdapterView;  
import android.widget.ListView;  
import android.widget.ProgressBar;  
  
import com.arellomobile.mvp.MvpAppCompatActivity;  
import com.arellomobile.mvp.presenter.InjectPresenter;  
  
import java.util.ArrayList;  
  
import ru.korolchuk.catalog.R;  
import ru.korolchuk.catalog.other.item.ShoppingItem;  
import ru.korolchuk.catalog.other.adapter.ShoppingListAdapter;  
import ru.korolchuk.catalog.presentation.presenter.MainAppPresenter;  
import ru.korolchuk.catalog.presentation.view.MainAppView;  
  
public class MainAppActivity extends MvpAppCompatActivity implements MainAppView {  
  
 public final String TAG = MainAppActivity.class.getSimpleName();  
  
 @InjectPresenter  
 MainAppPresenter presenter;  
  
 ListView shoppingItemView;  
 ProgressBar progressBar;  
 Toolbar toolbar;  
 FloatingActionButton shoppingCart;  
  
 ShoppingListAdapter adapter;  
  
 private Boolean exit = false;  
  
 private void init() {  
 progressBar = findViewById(R.id.*mainPageProgressBar*);  
 shoppingItemView = findViewById(R.id.*shoppingList*);  
 shoppingCart = findViewById(R.id.*cartMainPage*);  
  
 toolbar = findViewById(R.id.*toolbar*);  
 toolbar.setTitle(R.string.*app\_name*);  
 setSupportActionBar(toolbar);  
 }  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 getWindow().setFlags(WindowManager.LayoutParams.*FLAG\_SECURE*,  
 WindowManager.LayoutParams.*FLAG\_SECURE*);  
 setContentView(R.layout.*activity\_main\_app\_page*);  
 init();  
  
 shoppingCart.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 startActivity(new Intent(getApplicationContext(), ShoppingCartActivity.class));  
 }  
 });  
  
 shoppingItemView.setOnItemClickListener(new AdapterView.OnItemClickListener() {  
 @Override  
 public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {  
 Intent productIntent = new Intent(MainAppActivity.this, IndividualProductActivity.class);  
 productIntent.putExtra("product", presenter.getShopingItem(i));  
 startActivity(productIntent);  
 }  
 });  
 }  
  
 @Override  
 public boolean onCreateOptionsMenu(Menu menu) {  
 // Inflate the menu; this adds items to the action bar if it is present.  
 getMenuInflater().inflate(R.menu.*menu\_main\_app\_page*, menu);  
 return true;  
 }  
  
 @Override  
 public boolean onOptionsItemSelected(MenuItem item) {  
 // Handle action bar item clicks here. The action bar will  
 // automatically handle clicks on the Home/Up button, so long  
 // as you specify a parent activity in AndroidManifest.xml.  
 int id = item.getItemId();  
  
 //noinspection SimplifiableIfStatement  
 if (id == R.id.*logoutItem*) {  
 presenter.signOut();  
 startActivity(new Intent(getApplicationContext(), OpenScreenActivity.class));  
 finish();  
 }  
  
 return super.onOptionsItemSelected(item);  
 }  
  
 // For exiting the application  
 @Override  
 public void onBackPressed() {  
 if (exit) {  
 finish();  
 } else {  
 showSnack("Press back again to exit");  
 exit = true;  
 new Handler().postDelayed(new Runnable() {  
 @Override  
 public void run() {  
 exit = false;  
 }  
 }, 2000);  
 }  
 }  
  
 @Override  
 public void setProgressBarVisibility(int visibility) {  
 progressBar.setVisibility(visibility);  
 }  
  
 @Override  
 public void setShoppingItems(ArrayList<ShoppingItem> shoppingItems) {  
 adapter = new ShoppingListAdapter(getApplicationContext(), shoppingItems);  
 setProgressBarVisibility(View.*GONE*);  
 shoppingItemView.setAdapter(adapter);  
 }  
  
 @Override  
 public void showSnack(String text) {  
 Snackbar.*make*(  
 findViewById(R.id.*main\_content*),  
 text,  
 Snackbar.*LENGTH\_SHORT*).show();  
 }  
}

***MainAppPresenter.java:***

package ru.korolchuk.catalog.presentation.presenter;  
  
import android.util.Log;  
  
import com.arellomobile.mvp.InjectViewState;  
import com.arellomobile.mvp.MvpPresenter;  
import com.google.firebase.auth.FirebaseAuth;  
import com.google.firebase.database.DataSnapshot;  
import com.google.firebase.database.DatabaseError;  
import com.google.firebase.database.DatabaseReference;  
import com.google.firebase.database.FirebaseDatabase;  
import com.google.firebase.database.ValueEventListener;  
  
import java.util.ArrayList;  
  
import ru.korolchuk.catalog.other.item.ShoppingItem;  
import ru.korolchuk.catalog.presentation.view.MainAppView;  
  
@InjectViewState  
public class MainAppPresenter extends MvpPresenter<MainAppView> {  
  
 FirebaseDatabase database = FirebaseDatabase.*getInstance*();  
 DatabaseReference myRef = database.getReference("items");  
 private Boolean exit = false;  
 private ArrayList<ShoppingItem> shoppingItems;  
  
 private final String TAG = MainAppPresenter.class.getSimpleName();  
 public MainAppPresenter() {  
 myRef.addValueEventListener(new ValueEventListener() {  
 // This listener is only for database with reference of key "items"  
 @Override  
 public void onDataChange(DataSnapshot dataSnapshot) {  
 // This method is called once with the initial value and again  
 // whenever data at this location is updated.  
  
 // Now the Shopping List gets updated whenever the data changes in the server  
 shoppingItems = *getAllItems*(dataSnapshot);  
 getViewState().setShoppingItems(shoppingItems);  
 }  
  
 @Override  
 public void onCancelled(DatabaseError error) {  
 // Failed to read value  
 Log.*w*(TAG, "Failed to read value.", error.toException());  
 }  
 });  
 }  
  
 public static ArrayList<ShoppingItem> getAllItems(DataSnapshot dataSnapshot){  
  
 ArrayList<ShoppingItem> items = new ArrayList<ShoppingItem>();  
  
 for (DataSnapshot item : dataSnapshot.getChildren()) {  
  
 items.add(new ShoppingItem(  
 Integer.*valueOf*(item.child("productID").getValue().toString()),  
 item.child("name").getValue().toString(),  
 item.child("type").getValue().toString(),  
 item.child("description").getValue().toString(),  
 Integer.*valueOf*(item.child("price").getValue().toString()),  
 Integer.*valueOf*(item.child("quantity").getValue().toString())  
 ));  
 }  
 return items;  
 }  
  
 public void signOut() {  
 FirebaseAuth.*getInstance*().signOut();  
 }  
  
 public ShoppingItem getShopingItem(int i){  
 return shoppingItems.get(i);  
 }  
}

***ShoppingItem.java***

package ru.korolchuk.catalog.other.item;  
  
import android.content.Context;  
  
import java.io.Serializable;  
import java.text.NumberFormat;  
  
public class ShoppingItem implements Serializable{  
  
 private String name, type, description;  
 private int price, quantity, productID;  
  
 public ShoppingItem(int productId, String name, String type, String description, int price, int quantity){  
 this.productID = productId;  
 this.name = name;  
 this.type = type;  
 this.description = description;  
 this.price = price;  
 this.quantity = quantity;  
 }  
  
 public void setQuantity(int quantity){  
 this.quantity = quantity;  
 }  
  
 public int getProductID() {  
 return productID;  
 }  
  
 public String getTitle() {  
 return name;  
 }  
  
 public String getType() {  
 return type;  
 }  
  
 public String getDescription() {  
 return description;  
 }  
  
 public int getQuantity() {  
 return quantity;  
 }  
  
  
 public String getPrice() {  
 return NumberFormat.*getCurrencyInstance*().format(price);  
 }  
}

***ShoppingCartAdapter.java***

package ru.korolchuk.catalog.other.adapter;  
  
import android.content.Context;  
import android.support.annotation.NonNull;  
import android.support.annotation.Nullable;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.ArrayAdapter;  
import android.widget.ImageView;  
import android.widget.TextView;  
import com.bumptech.glide.Glide;  
import java.text.NumberFormat;  
import java.text.ParseException;  
import java.util.ArrayList;  
import java.util.List;  
  
import ru.korolchuk.catalog.R;  
import ru.korolchuk.catalog.other.item.ShoppingItem;  
  
public class ShoppingCartAdapter extends ArrayAdapter {  
 Context context;  
 ArrayList<ShoppingItem> items;  
  
 public ShoppingCartAdapter(Context context, List<ShoppingItem> items){  
 super(context, 0, items);  
 this.context = context;  
 this.items = (ArrayList<ShoppingItem>) items;  
 }  
  
 @NonNull  
 @Override  
 public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {  
  
 View listItemView = convertView;  
 if (listItemView == null) {  
 listItemView = LayoutInflater.*from*(getContext()).inflate(  
 R.layout.*cart\_item*, parent, false  
 );  
 }  
  
 ShoppingItem currentItem = (ShoppingItem) getItem(position);  
 ImageView img = (ImageView) listItemView.findViewById(R.id.*cartItemIcon*);  
 Glide.*with*(context).load(*getLinkImage*(currentItem)).into(img);  
  
 ((TextView) listItemView.findViewById(R.id.*cartItemName*))  
 .setText(currentItem.getTitle());  
  
 String x = "x " + String.*valueOf*(currentItem.getQuantity());  
 ((TextView) listItemView.findViewById(R.id.*cartItemQuantity*))  
 .setText(x);  
  
 int itemPrice = 0;  
 try{  
 itemPrice = Integer.*valueOf*(NumberFormat.*getCurrencyInstance*()  
 .parse(String.*valueOf*(currentItem.getPrice()))  
 .toString());  
 System.*out*.println(itemPrice);  
 System.*out*.println(itemPrice);  
 System.*out*.println(itemPrice);  
 System.*out*.println(itemPrice);  
 System.*out*.println(itemPrice);  
  
 } catch (ParseException e){  
 e.printStackTrace();  
 }  
 ((TextView) listItemView.findViewById(R.id.*cartItemPrice*))  
 .setText(NumberFormat.*getCurrencyInstance*().format(itemPrice));  
  
 ((TextView) listItemView.findViewById(R.id.*cartItemTotal*))  
 .setText(NumberFormat.*getCurrencyInstance*().format(itemPrice \* currentItem.getQuantity()));  
  
  
 // No idea how to implement remove individual item from cart  
 // Appreciated if anyone can fix it.  
// ImageView removeFromCart = (ImageView) listItemView.findViewById(R.id.removeFromCart);  
// removeFromCart.setImageResource(R.drawable.ic\_clear\_black\_24dp);  
// removeFromCart.setOnClickListener(new View.OnClickListener() {  
// @Override  
// public void onClick(View view) {  
// items.remove()  
// }  
// });  
  
 return listItemView;  
 }  
  
 public static String getLinkImage(ShoppingItem item)  
 {  
 String[] urls = new String[] {  
 "https://firebasestorage.googleapis.com/v0/b/test-b0bb9.appspot.com/o/0.jpg?alt=media&token=698cb551-6863-4964-982a-5490be43bc1d",  
 "https://firebasestorage.googleapis.com/v0/b/test-b0bb9.appspot.com/o/1.jpg?alt=media&token=c5e4669e-05c1-43b7-8763-ad07e51ee7b4",  
 "https://firebasestorage.googleapis.com/v0/b/test-b0bb9.appspot.com/o/2.jpg?alt=media&token=4313c61f-5b88-4729-af5c-c73bded00950",  
 "https://firebasestorage.googleapis.com/v0/b/test-b0bb9.appspot.com/o/3.jpg?alt=media&token=4bbbb0f9-ea55-40e8-b5e6-91ac186589c8",  
 "https://firebasestorage.googleapis.com/v0/b/test-b0bb9.appspot.com/o/4.jpg?alt=media&token=bc7e2f48-c9ae-4f90-871c-5f2313708591",  
 "https://firebasestorage.googleapis.com/v0/b/test-b0bb9.appspot.com/o/5.jpg?alt=media&token=9a6950ab-d8a7-4b1b-9c5e-a4e5acfa5d7d",  
 "https://firebasestorage.googleapis.com/v0/b/test-b0bb9.appspot.com/o/6.jpg?alt=media&token=57b76824-b3c5-47da-8ef0-9f1108af8783",  
 "https://firebasestorage.googleapis.com/v0/b/test-b0bb9.appspot.com/o/7.jpg?alt=media&token=751384be-2565-4ff8-bb99-933d59b58c37",  
 "https://firebasestorage.googleapis.com/v0/b/test-b0bb9.appspot.com/o/8.jpg?alt=media&token=49456e63-d354-4435-985a-fb43bf47dc52",  
 "https://firebasestorage.googleapis.com/v0/b/test-b0bb9.appspot.com/o/9.jpg?alt=media&token=2e59a06c-d47a-43dd-8d6b-9d0955c641ff",  
 };  
 return urls[item.getProductID()];  
 }  
 public static String getLinkImage(int item)  
 {  
 String[] urls = new String[] {  
 "https://firebasestorage.googleapis.com/v0/b/test-b0bb9.appspot.com/o/0.jpg?alt=media&token=698cb551-6863-4964-982a-5490be43bc1d",  
 "https://firebasestorage.googleapis.com/v0/b/test-b0bb9.appspot.com/o/1.jpg?alt=media&token=c5e4669e-05c1-43b7-8763-ad07e51ee7b4",  
 "https://firebasestorage.googleapis.com/v0/b/test-b0bb9.appspot.com/o/2.jpg?alt=media&token=4313c61f-5b88-4729-af5c-c73bded00950",  
 "https://firebasestorage.googleapis.com/v0/b/test-b0bb9.appspot.com/o/3.jpg?alt=media&token=4bbbb0f9-ea55-40e8-b5e6-91ac186589c8",  
 "https://firebasestorage.googleapis.com/v0/b/test-b0bb9.appspot.com/o/4.jpg?alt=media&token=bc7e2f48-c9ae-4f90-871c-5f2313708591",  
 "https://firebasestorage.googleapis.com/v0/b/test-b0bb9.appspot.com/o/5.jpg?alt=media&token=9a6950ab-d8a7-4b1b-9c5e-a4e5acfa5d7d",  
 "https://firebasestorage.googleapis.com/v0/b/test-b0bb9.appspot.com/o/6.jpg?alt=media&token=57b76824-b3c5-47da-8ef0-9f1108af8783",  
 "https://firebasestorage.googleapis.com/v0/b/test-b0bb9.appspot.com/o/7.jpg?alt=media&token=751384be-2565-4ff8-bb99-933d59b58c37",  
 "https://firebasestorage.googleapis.com/v0/b/test-b0bb9.appspot.com/o/8.jpg?alt=media&token=49456e63-d354-4435-985a-fb43bf47dc52",  
 "https://firebasestorage.googleapis.com/v0/b/test-b0bb9.appspot.com/o/9.jpg?alt=media&token=2e59a06c-d47a-43dd-8d6b-9d0955c641ff",  
 };  
 return urls[item];  
 }  
}

***ShoppingListAdapter.java***

package ru.korolchuk.catalog.other.adapter;  
  
import android.content.Context;  
import android.support.annotation.NonNull;  
import android.support.annotation.Nullable;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.ArrayAdapter;  
import android.widget.ImageView;  
import android.widget.TextView;  
  
import com.bumptech.glide.Glide;  
  
import java.util.List;  
  
import ru.korolchuk.catalog.R;  
import ru.korolchuk.catalog.other.item.ShoppingItem;  
  
public class ShoppingListAdapter extends ArrayAdapter<ShoppingItem> {  
  
 Context context;  
  
 public ShoppingListAdapter(Context context, List<ShoppingItem> items){  
 super(context, 0, items);  
 this.context = context;  
 }  
  
 @NonNull  
 @Override  
 public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {  
  
 View listItemView = convertView;  
 if (listItemView == null) {  
 listItemView = LayoutInflater.*from*(getContext()).inflate(  
 R.layout.*list\_item*, parent, false  
 );  
 }  
  
 ShoppingItem currentItem = getItem(position);  
  
 ImageView img = (ImageView) listItemView.findViewById(R.id.*itemIcon*);  
 Glide.*with*(context).load(ShoppingCartAdapter.*getLinkImage*(currentItem)).into(img);  
  
 TextView name = (TextView) listItemView.findViewById(R.id.*itemName*);  
 name.setText(currentItem.getTitle());  
  
 TextView description = (TextView) listItemView.findViewById(R.id.*itemDescription*);  
 description.setText(currentItem.getDescription());  
  
 TextView cost = (TextView) listItemView.findViewById(R.id.*itemPrice*);  
 cost.setText(currentItem.getPrice());  
  
 return listItemView;  
 }  
}