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 «Каталог товаров техники»

Выполнил студент Руденя Дмитрий 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 – «Каталог товаров техники».

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 «Каталог товаров техники» - ресурса, целью которого является предоставление коллекции товаров, которые могут приобрести люди, заинтересованные в технике.

Целью курсового проекта является проектирование и реализация мобильного приложения «Каталог товаров техники».

# 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 SQLite защищённая с помощью SQLCipher. Авторизация также должна быть реализована с помощью хеширования паролей на основе алгоритма PBKDF2.

Необходимо обеспечить невозможность создания скриншотов некоторых экрана, а также использовать в проекте обфускацию, для оптимизации и защиты кода приложения.

При разработке следует использовать язык программирования Java. В качестве основного паттерна должен использоваться MVP.

1. **Архитекутра проекта**

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].

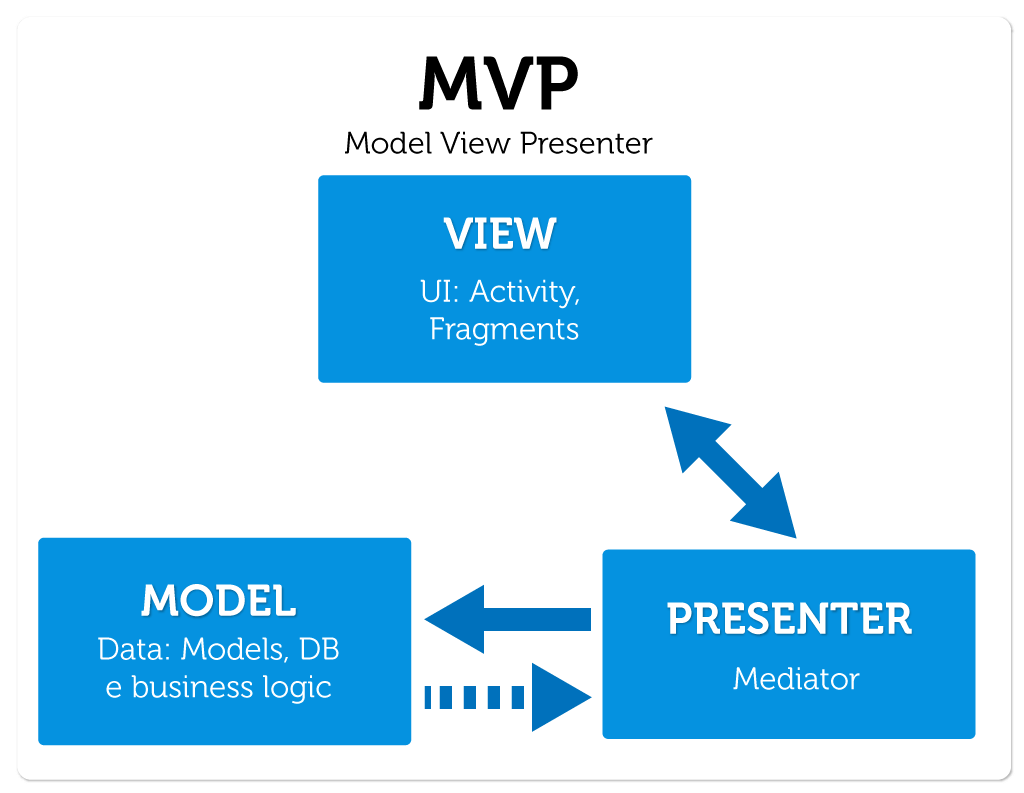


Рис. 2.1 – Схема паттерна MVP

# Схема базы данных

В качестве мобильной СУБД используется SQLite защищённая с помощью SQLCipher(расширение SQLite с открытым исходным кодом, которое обеспечивает прозрачное 256-битное полное шифрование базы данных AES).

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

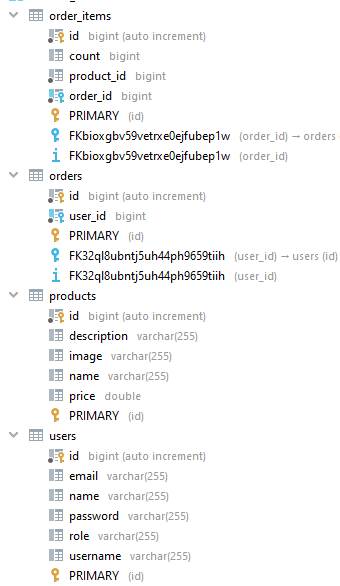


Рисунок 3.1 – структура базы данных

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.

# Программная реализация 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.

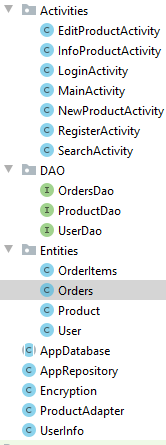


Рисунок 4.2.1 – структура проекта

Подробное описание структуры:

1. Activities содержит реализацию классов, которые отвечают за отображение пользовательского интерфейса
2. DAO содержит сигнатуры методов для работы с БД каждой модели
3. Entites содержит описание моделей, на основе которых строятся таблицы
4. AppDatabase, AppRepository - вспомогательные классы для работы с базой данных
5. Encryption - класс, содержащий реализацию алгоритма шифрования
6. ProductAdapter - класс, содержащий реализацию адаптера, используемого для отображения списка

Dependencies, которые использует мобильное приложение для сборки:



# Руководство пользователя

## 5.1 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 5.1.1). Авторизация происходит с помощью ввода логина и пароля, который хешируется с помощью алгоритма PBKDF2 и сравнивается с хешем, хранимым в БД

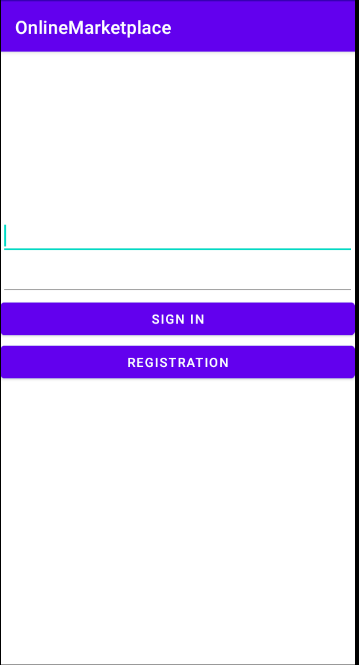


Рисунок 5.1.1 – окно авторизации и регистрации

После ввода данных (имя пользователя и пароль) мы можем видеть все товары, доступные для заказа на данный момент (Рисунок 5.1.2).

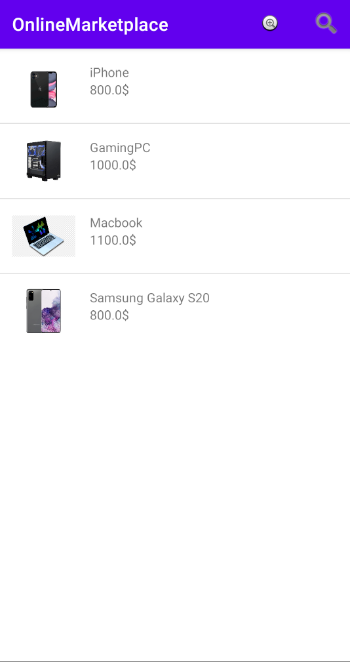
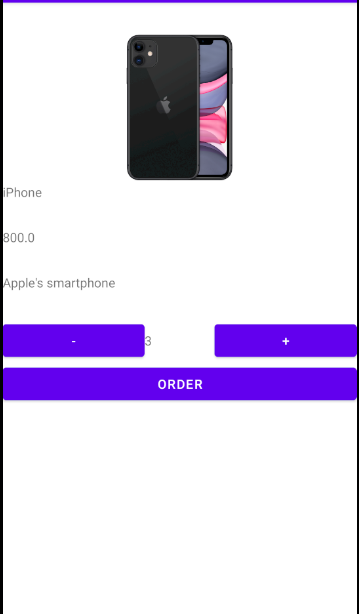


Рисунок 5.1.2 – все записи

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. В данной Activity была реализована невозможность выполнения скриншотов путём установки определённых флагов защиты.  
(Рисунок 5.1.3).



Риснок 5.1.3 – 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 базы данных с соответствующими значениями. (Рисунок 5.1.4)

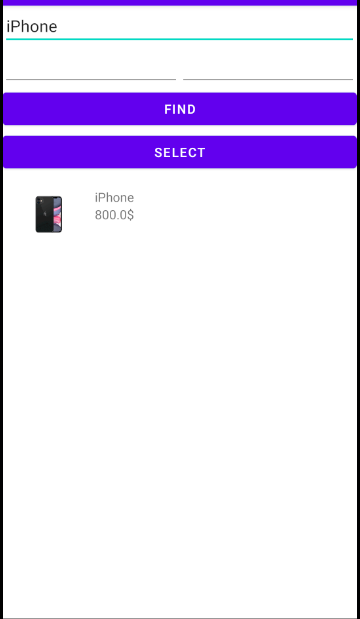


Рисунок 5.1.4 – Поиск и выборка товаров

# Заключение

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

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
* Обфускация проекта

# 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

**AppDatabase.java**

**import** android.content.Context;  
  
import androidx.annotation.NonNull;  
import androidx.room.Database;  
import androidx.room.Room;  
import androidx.room.RoomDatabase;  
import androidx.sqlite.db.SupportSQLiteDatabase;  
  
import com.example.onlinemarketplace.DAO.ProductDao;  
import com.example.onlinemarketplace.DAO.UserDao;  
import com.example.onlinemarketplace.Entities.Product;  
import com.example.onlinemarketplace.Entities.User;  
  
import net.sqlcipher.database.SQLiteDatabase;  
import net.sqlcipher.database.SupportFactory;  
  
import java.security.NoSuchAlgorithmException;  
import java.security.spec.InvalidKeySpecException;  
import java.util.concurrent.ExecutorService;  
import java.util.concurrent.Executors;  
  
@Database(entities = {User.**class**, Product.**class**}, version = 2)  
public abstract class AppDatabase **extends** RoomDatabase {  
  
 **public abstract** UserDao userDao();  
 **public abstract** ProductDao productDao();  
  
 **private static volatile** AppDatabase *INSTANCE*;  
 **private static final int *NUMBER\_OF\_THREADS*** = 4;  
 **static final** ExecutorService ***databaseWriteExecutor*** =  
 Executors.*newFixedThreadPool*(***NUMBER\_OF\_THREADS***);  
  
 **static char**[] *testBytes* = {**'t'**, **'e'**, **'s'**, **'t'**};  
 **static final byte**[] ***passphrase*** = SQLiteDatabase.*getBytes*(*testBytes*);  
 **static final** SupportFactory ***factory*** = **new** SupportFactory(***passphrase***);  
  
 **public static** AppDatabase getDatabase(**final** Context context) {  
 **if** (*INSTANCE* == **null**) {  
 **synchronized** (AppDatabase.**class**) {  
 **if** (*INSTANCE* == **null**) {  
 *INSTANCE* = Room.*databaseBuilder*(context.getApplicationContext(),  
 AppDatabase.**class**, **"app\_database"**)  
 .allowMainThreadQueries().fallbackToDestructiveMigration().addCallback(*sRoomDatabaseCallback*).openHelperFactory(***factory***).build();  
 }  
 }  
 }  
 **return** *INSTANCE*;  
 }  
  
 **private static** RoomDatabase.Callback *sRoomDatabaseCallback* = **new** RoomDatabase.Callback() {  
 @Override  
 **public void** onOpen(@NonNull SupportSQLiteDatabase db) {  
 **super**.onOpen(db);  
  
 *// If you want to keep data through app restarts,*  
 *// comment out the following block*  
***databaseWriteExecutor***.execute(() -> {  
 *// Populate the database in the background.*  
 *// If you want to start with more words, just add them.*  
UserDao dao = *INSTANCE*.userDao();  
  
  
 User user = **null**;  
 **try** {  
 user = **new** User(**"Dmitry"**, **"dimar2211"**, Encryption.*encrypt*(**"123456"**), **"dimar2211@mail.ru"**, **"ADMIN"**);  
 } **catch** (NoSuchAlgorithmException e) {  
 e.printStackTrace();  
 } **catch** (InvalidKeySpecException e) {  
 e.printStackTrace();  
 }  
 dao.insert(user);  
  
 });  
 }  
 };  
}

# Приложение B

**MainActivity.java**

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.view.WindowManager;  
import android.widget.AdapterView;  
import android.widget.EditText;  
import android.widget.LinearLayout;  
import android.widget.ListView;  
  
import com.example.onlinemarketplace.AppRepository;  
import com.example.onlinemarketplace.Entities.Product;  
import com.example.onlinemarketplace.ProductAdapter;  
import com.example.onlinemarketplace.R;  
import com.example.onlinemarketplace.UserInfo;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class MainActivity extends AppCompatActivity {  
  
 private AppRepository repository;  
  
 public List<Product> listProduct;  
 ListView productsList;  
 ProductAdapter productAdapter;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 getWindow().setFlags(WindowManager.LayoutParams.FLAG\_SECURE,  
 WindowManager.LayoutParams.FLAG\_SECURE);  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_main);  
  
 productsList = findViewById(R.id.productList);  
  
 repository = new AppRepository(getApplication());  
 Log.d("UserRole", "onCreate: " + UserInfo.role);  
 Intent intentEdit = new Intent(this, EditProductActivity.class);  
 Intent intentInfo = new Intent(this, InfoProductActivity.class);  
 listProduct = repository.getAllProducts();  
 productAdapter = new ProductAdapter(getApplicationContext(), R.layout.list\_item, listProduct);  
 productsList.setAdapter(productAdapter);  
 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());  
 intentEdit.putExtra("UserRole", UserInfo.role);  
 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);  
 try {  
 if (UserInfo.role.equals("ADMIN")) {  
 menu.getItem(0).setVisible(true);  
 } else {  
 menu.getItem(0).setVisible(false);  
 }  
 } catch (Exception e) {  
 Log.d("MenuOptiona", "onCreateOptionsMenu: " + e.getMessage());  
  
 }  
  
 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);  
 }  
}

**InfoProductActivity.java**

public class InfoProductActivity extends AppCompatActivity {  
  
 ImageView imageView;  
 TextView nameTextView;  
 TextView priceTextView;  
 TextView descriptionTextView;  
 TextView countText;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 getWindow().setFlags(WindowManager.LayoutParams.FLAG\_SECURE,  
 WindowManager.LayoutParams.FLAG\_SECURE);  
 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));  
  
 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));  
 }

**LoginActivity.java**

public class LoginActivity extends AppCompatActivity {  
  
 private AppRepository repository;  
 EditText loginText;  
 EditText passwordText;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 getWindow().setFlags(WindowManager.LayoutParams.FLAG\_SECURE,  
 WindowManager.LayoutParams.FLAG\_SECURE);  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_login);  
  
 repository = new AppRepository(getApplication());  
  
 loginText = findViewById(R.id.loginText);  
 passwordText = findViewById(R.id.passwordText);  
 }  
  
 public void loginClick(View view){  
 Intent intent = new Intent(this, MainActivity.class);  
 try {  
 User user = repository.getUserByLogin(String.valueOf(loginText.getText()));  
 if (Arrays.equals(user.getPassword(), Encryption.encrypt(String.valueOf(passwordText.getText())))) {  
 UserInfo.id = user.getUid();  
 UserInfo.role = user.getRole();  
 UserInfo.username = user.getLogin();  
 startActivity(intent);  
 } else {  
 Log.d("UserSignInCheck", "Wrong password");  
 }  
 } catch (Exception e) {  
 Log.d("Login", e.getMessage());  
 }  
  
  
 }  
  
 public void registrationClick(View view){  
 Intent intent = new Intent(this, RegisterActivity.class);  
 startActivity(intent);  
 }  
}

**Encryption.java**

**import** java.security.NoSuchAlgorithmException;  
import java.security.spec.InvalidKeySpecException;  
import java.security.spec.KeySpec;  
  
import javax.crypto.SecretKeyFactory;  
import javax.crypto.spec.PBEKeySpec;  
  
public class Encryption {  
 **public static byte**[] encrypt(String password) **throws** NoSuchAlgorithmException, InvalidKeySpecException {  
 **byte**[]salt = **new byte**[]{1, 3, 5, 7};  
 KeySpec spec = **new** PBEKeySpec(password.toCharArray(), salt, 65536, 128);  
 SecretKeyFactory factory = SecretKeyFactory.*getInstance*(**"PBKDF2WithHmacSHA1"**);  
 **byte**[]hashPass = factory.generateSecret(spec).getEncoded();  
 **return** hashPass;  
 }  
}