09.02.07 ПР-418

**ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ**

**ПП.12.01 Коммерческие аспекты разработки**

**профессионального модуля**

**ПМ.12 Разработка программных решений**

|  |  |  |
| --- | --- | --- |
| **Практиканты** |  | **Степанов А. А.** |
| **Руководители практики** |  | **Смирнова Е. Е.**  **Агафонов И. В.** |

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 3](#_Toc99795633)

[ДНЕВНИК ПРАКТИКИ 5](#_Toc99795634)

[ПОСТАНОВКА ЗАДАЧИ 6](#_Toc99795635)

[РЕАЛИЗАЦИЯ ФУНКЦИЙ ПРИЛОЖЕНИЯ 11](#_Toc99795636)

[ПОДГОТОВКА ПРИЛОЖЕНИЯ К ПУБЛИКАЦИИ 21](#_Toc99795637)

[ЗАКЛЮЧЕНИЕ 22](#_Toc99795638)

[СПИСОК ЛИТЕРАТУРЫ 23](#_Toc99795639)

# ВВЕДЕНИЕ

Главными целями и задачами производственной практики ПП.12.01 «Коммерческие аспекты разработки» профессионального модуля ПМ.12 «Разработка программных решений» специальности 09.02.07 «Информационные системы и программирование» является освоение обучающимися вида профессиональной деятельности «Разработка программных решений» и соответствующих профессиональных и общих компетенций:

|  |  |
| --- | --- |
| ПК 12.2 | Создавать приложения со сложной логикой переходов |
| ОК 1. | Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам |
| ОК 2. | Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности. |
| ОК 3. | Планировать и реализовывать собственное профессиональное и личностное развитие. |
| ОК 4. | Работать в коллективе и команде, эффективно взаимодействовать с коллегами, руководством, клиентами. |
| ОК 5. | Осуществлять устную и письменную коммуникацию на государственном языке с учетом особенностей социального и культурного контекста. |
| ОК 6. | Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе традиционных общечеловеческих ценностей |
| ОК 7. | Содействовать сохранению окружающей среды, ресурсосбережению, эффективно действовать в чрезвычайных ситуациях. |
| ОК 8. | Использовать средства физической культуры для сохранения и укрепления здоровья в процессе профессиональной деятельности и поддержания необходимого уровня физической подготовленности |
| ОК 9. | Использовать информационные технологии в профессиональной деятельности. |
| ОК 10 | Пользоваться профессиональной документацией на государственном и иностранном языке |
| ОК 11 | Планировать предпринимательскую деятельность в профессиональной сфере |

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающиеся в ходе выполнения заданий практики ПП.12.01 «Коммерческие аспект разработки» должны приобрести следующий практический опыт:

* разработки мобильных приложений;
* проведения тестирования и отладки приложения;
* использования инструментальных средств на этапе тестирования программного продукта;
* использования системы контроля версий.

В результате освоения программы учебной практики студент будет уметь:

* использовать технологии для разработки мобильных приложений;
* использовать технологии для работы с различными протоколами обмена данными;
* строить приложения со сложной логикой переходов;
* оформлять документацию на программные средства.

За время практики должны быть выполнены следующие виды работ:

* разработка, тестирование приложений для различных устройств под управлением ОС Android;
* подготовка приложений для публикации;
* публикация приложения в Play Market
* продвижение приложения;
* запуск Play Store из приложения;
* презентация решений.

Производственная практика ПП.12.01 «Коммерческие аспекты разработки» проводится на базе профессиональной мастерской «Разработка мобильных приложений» в БПОУ «Омский авиационный колледж имени Н.Е. Жуковского»

# ДНЕВНИК ПРАКТИКИ

|  |  |  |
| --- | --- | --- |
| ***Дата*** | ***Виды выполненных работ*** | ***Подпись руководителя практики*** |
| 21.03.22 | Создание графического интерфейса пользователя и файлов ресурсов. |  |
| 22.03.22 | Реализация запросов к API для получения данных. |  |
| 23.03.22 | Обработка ошибок при работе с веб-сервисом. |  |
| 24.03.22 | Организация хранения данных в памяти устройства. |  |
| 25.03.22 | Реализация логики работы приложения. |  |
| 26.03.22 | Тестирование работы приложения. |  |
| 28.03.22 | Описание продукта. |  |
| 29.03.22 | Создание файла сертификата. |  |
| 30.03.22 | Получение подписанного apk. |  |
| 31.03.22 | Загрузка подписанного apk и тестирование. |  |
| 01.03.22 | Документирование результатов. |  |
| 02.03.22 | Подготовка и защита портфолио. |  |

# ПОСТАНОВКА ЗАДАЧИ

1. Разработать мобильные приложения для смартфона и для умных часов для управления коллекцией фильмов.
2. Подготовить приложения для публикации (к релизу)

**Необходимый функционал приложения для смартфона:**

1. иконка приложения должна соответствовать макету.
2. экран ***Launch Screen*** должен соответствовать макету. Текст на экране должен быть отдельным элементом. Логотип приложения должен быть расположен по центру экрана
3. экран ***SignUp Screen*** должен соответствовать макету
   * + при нажатии на кнопку "*Зарегистрироваться*" проверяются поля для ввода на пустоту,а также email на корректность (требования к email описаны в документации к API). При некорректном заполнении ошибка отображается с помощью диалогового окна. Так же проверяется равенство пароля и его повтора;
     + при корректном заполнении формы регистрации запрос регистрации отправляется на сервер. При получении ошибки от сервера она отображается с помощью диалогового окна. При успешной регистрации автоматически осуществляется авторизация и переход на ***Main Screen***;
     + при нажатии на кнопку "***У меня уже есть аккаунт***" осуществляется переход на ***SignIn Screen***;
     + при первом запуске приложения после ***Launch Screen*** отображается ***SignUp Screen***. При последующих - ***SignIn Screen;***
4. экран ***SignIn Screen*** должен соответствовать макету:
   * + при нажатии на кнопку "*Войти*" проверяются поля для ввода на пустоту, а также email на корректность (требования к email описаны в документации к API). При некорректном заполнении ошибка отображается с помощью диалогового окна. При корректном заполнении формы на сервер отправляется соответствующий запрос;
     + при нажатии на кнопку "*Регистрация*" осуществляется переход на ***SignUp Screen***;
     + при успешной авторизации осуществляется переход на экран ***MainScreen***. При получении ошибки от сервера она отображается с помощью диалогового окна;
5. экран ***Profile Screen*** должен соответствовать макету:
   * + данные о пользователе запрашиваются с сервера.

при нажатии на кнопку "*Изменить*" реализовано изменение аватара пользователя:

* + - * пользователь выбирает источник фотографии (камера или Галерея), выбор источника реализован с помощью диалогового окна. При выборе *Галереи* открывается *Галерея*. При выборе камеры переход на экран *Камеры* не нужен;
      * пользователь выбирает фотографию;
      * фотография отправляется на сервер;
      * в случае успеха аватар пользователя заменяется на новый, в случае ошибки она отображается с помощью диалогового окна;
    - при нажатии на кнопку "*Выход*" осуществляется переход на экран ***авторизации***;
    - при нажатии на кнопку «*Обсуждения*» осуществляется переход на соответствующий экран;

1. экран ***Chat List Screen*** должен соответствовать макету:
   * + информация запрашивается с сервера (запрос чатов пользователя). Если информация с сервера содержит дубляжи – они удаляются. Если пользователь еще не имеет чатов,

сервер присылает пустой список, от текущего пользователя в чате отправляется сообщение с id = 1 с помощью *Swagger* или *Postman*;

* + - в подзаголовке ячейки отображается последнее сообщение в соответствующем чате + имя его автора. Текст сообщения обрезается до двух строк;
    - реализовано отображение постеров к фильмам. Для получения постеров используется подходящий запрос из API. Если постер для данного фильма нельзя получить из API, генерируется аббревиатура по следующему правилу: если название фильма состоит из одного слова - берутся первые две буквы слова; иначе - первые буквы первого и второго слова;
    - при нажатии на ячейку осуществляется переход на ***Chat Screen*** для выбранного фильма;

1. экран ***Chat Screen*** должен соответствовать макету:
   * + сообщения упорядочены от старых к новым сверху вниз. Для сегодняшних сообщений отображается заголовок "Сегодня";
     + "Облако" сообщения растянуто по содержимому;
     + последовательно идущие сообщения одного автора сгруппированы (расстояния между сообщениями как на макете);
     + реализован блок отправки сообщения как на макете. При вводе сообщения поле для ввода растягивается по вертикали;
     + при нажатии на кнопку "*Отправить*" сообщение отправляется на сервер. При позитивном ответе от сервера отображается сообщение в чате. При возникновении ошибки – отображается ошибка с помощью диалогового окна;
     + поле для ввода валидируется на пустоту. При отсутствии текста сообщения отображается ошибка с помощью диалогового окна;
2. экран ***Collections Screen*** должен соответствовать макету:
   * + при нажатии на иконку в правом верхнем углу осуществляется переход на экран ***Create Collection Screen***;
     + на экране отображаются созданные коллекции. Информация о коллекциях хранится в памяти устройства. Хранится название коллекции и иконка;
     + для удаления коллекции реализован *Swipe-to-delete*, в том числе из памяти устройства;
3. экран ***Create Collection Screen*** должен соответствовать макету:
   * + при открытии экрана в качестве иконки выбирается случайное изображение из коллекции иконок;
     + при нажатии на кнопку "*Выбрать иконку*" осуществляется переход на экран ***Icon Selection***, который реализован в соответствии с макетом;
     + при нажатии на кнопку "*Сохранить*" новая коллекция сохраняется в памяти устройства и экран закрывается;
     + поле для ввода названия коллекции проверяется на пустоту. При отсутствии значения отображается сообщение об ошибке;
4. экран ***Main Screen*** должен соответствовать макету:
   * + на экране отображаются обложки фильмов из подборки «new» (информация о фильмах запрашивается с сервера). Обложки отображаются в виде карусели, они листаются с помощью жеста *swipe*;
     + при пролистывании обложек название фильма в верхней части экрана меняется;
     + при нажатии на обложку происходит переход на ***Chat Screen*** для соответствующего фильма;

**Необходимый функционал приложения для умных часов:**

1. иконка приложения настроена согласно макету.
2. экран ***SignIn Screen*** должен соответствовать макету:

* при нажатии на кнопку "*Войти*" проверяются поля для ввода на пустоту, а также email на корректность (требования к email описаны в документации к API). При некорректном заполнении отображается ошибка с помощью диалогового окна. При корректном заполнении формы на сервер отправляется соответствующий запрос;
* при успешной авторизации осуществляется переход на экран ***Main Screen***. При получении ошибки от сервера она отображается с помощью диалогового окна;

1. экран Main Screen согласно макету.

* при нажатии на иконку осуществляется переход на соответствующий экран. Если экран не описан в Задании - экран отображается с надписью “В разработке”;
* при нажатии на кнопку *Cancel* (для Android - при свайпе вправо) происходит возвратя на ***SignIn Screen***;

1. экран ***Chat List Screen*** должен соответствовать макету:

* данные запрашиваются с сервера;
* при нажатии на кнопку *Cancel* (для Android - при свайпе вправо) происходит возврат на ***Main Screen***;

1. экран ***Movies Screen*** должен соответствовать макету:

* на данном экране отображаются фильмы из подборки «new» в виде списка с вертикальной прокруткой. Данные берутся с сервера;

# РЕАЛИЗАЦИЯ ФУНКЦИЙ ПРИЛОЖЕНИЯ

**Смартфон**

Разработка приложения началась с реализации функции регистрации/авторизации. Листинг процедуры регистрации приведён ниже.

*Листинг 1. Регистрация пользователя*

*//Регистрация пользователя*public void register\_user(){  
 *///Interceptor и Client для вывода логов при работе в сети* HttpLoggingInterceptor interceptor = new HttpLoggingInterceptor();  
 interceptor.setLevel(HttpLoggingInterceptor.Level.*BODY*);  
  
 OkHttpClient.Builder client = new OkHttpClient.Builder()  
 .addInterceptor(interceptor);  
  
 *//Retrofit - библиотека, с помощью которой ведётся работа с сервером* Retrofit retrofit = new Retrofit.Builder()  
 .baseUrl("http://cinema.areas.su")  
 .addConverterFactory(GsonConverterFactory.*create*())  
 .client(client.build())  
 .build();  
  
 API api = retrofit.create(API.class);  
  
 *//RegBody содержит тело запроса* RegBody body = new RegBody();  
 body.email = email;  
 body.firstname = firstname;  
 body.lastname = lastname;  
 body.password = password;  
  
 *//Регистрация пользователя* AsyncTask.*execute*(()->{  
 Call<Void> call;  
 call = api.regUser(body);  
 call.enqueue(new Callback<Void>() {  
 @Override  
 public void onResponse(Call<Void> call, Response<Void> response) {  
 if (response.isSuccessful()){  
 Toast.*makeText*(context, "Registration successful", Toast.*LENGTH\_SHORT*).show();  
 }  
 else {  
 Toast.*makeText*(context, "Error", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
  
 @Override  
 public void onFailure(Call<Void> call, Throwable t) {  
 t.printStackTrace();  
 }  
 });  
 });  
}

*Листинг 2. Авторизация пользователя*

*//Авторизация пользователя*public void log\_user(){  
 LogBody logBody = new LogBody(email, password);  
  
 *///Interceptor и Client для вывода логов при работе в сети* HttpLoggingInterceptor interceptor = new HttpLoggingInterceptor();  
 interceptor.setLevel(HttpLoggingInterceptor.Level.*BODY*);  
  
 OkHttpClient.Builder client = new OkHttpClient.Builder()  
 .addInterceptor(interceptor);  
 *//Retrofit - библиотека, с помощью которой ведётся работа с сервером* Retrofit retrofit = new Retrofit.Builder()  
 .baseUrl("http://cinema.areas.su")  
 .addConverterFactory(GsonConverterFactory.*create*())  
 .client(client.build())  
 .build();  
  
 API api = retrofit.create(API.class);  
  
 *//Авторизация пользователя* Call<LogResponse> call;  
 call = api.logUser(logBody);  
 call.enqueue(new Callback<LogResponse>() {  
 @Override  
 public void onResponse(Call<LogResponse> call, Response<LogResponse> response) {  
 if (response.isSuccessful()){  
 context.startActivity(new Intent(context, MainScreen.class));  
   
 *//Сохранения токена в глобальных настройках приложения* int token = response.body().get\_token();  
 editor.putString(*APP\_PREFERENCES\_TOKEN*, String.*valueOf*(token));  
 editor.apply();  
 Toast.*makeText*(context, "Successful. Your token is: " + token, Toast.*LENGTH\_SHORT*).show();  
 }  
 else{  
 Toast.*makeText*(context, "Error", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
  
 @Override  
 public void onFailure(Call<LogResponse> call, Throwable t) {  
 Toast.*makeText*(context, "Response Error", Toast.*LENGTH\_SHORT*).show();  
 t.printStackTrace();  
 }  
 });  
}

Ниже приведены скриншоты экранов регистрации и авторизации:



Рисунок 1. Экран авторизации

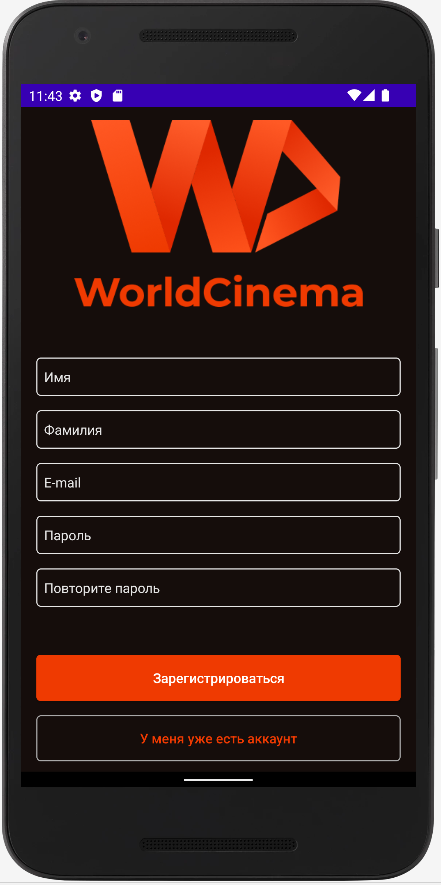


Рисунок 2. Экран регистрации

Далее я приступил к разработке экрана с новыми фильмами. Листинг класса приведён ниже.

*Листинг 3. Класс new\_films\_fragment*

public class new\_films\_fragment extends Fragment {  
  
 private static Context *context*;  
  
 private static final String *ARG\_PARAM1* = "param1";  
 private static final String *ARG\_PARAM2* = "param2";  
  
 private String mParam1;  
 private String mParam2;  
  
 public RecyclerView recycler;  
  
 public FilmAdapter adapter;  
  
 public new\_films\_fragment() {  
  
 }  
  
 *//Создание фрагмента* public static new\_films\_fragment newInstance(Context mcontext) {  
 *context* = mcontext;  
 return new new\_films\_fragment();  
 }  
  
 @Override  
 public void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
  
 if (getArguments() != null) {  
 mParam1 = getArguments().getString(*ARG\_PARAM1*);  
 mParam2 = getArguments().getString(*ARG\_PARAM2*);  
 }  
 }  
  
 *//Получение разметки для фрагмента* @Override  
 public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {  
 return inflater.inflate(R.layout.*fragment\_new\_films\_fragment*, container, false);  
 }  
  
 *//Заполнение RecyclerView фильмами* @Override  
 public void onViewCreated(View view, @Nullable Bundle savedInstanceState){  
 recycler = view.findViewById(R.id.*recycle*);  
  
 getFilms();  
  
 PagerSnapHelper helper = new PagerSnapHelper();  
 LinearLayoutManager manager = new LinearLayoutManager(requireContext(), LinearLayoutManager.*HORIZONTAL*, false);  
 helper.attachToRecyclerView(recycler);  
 recycler.setLayoutManager(manager);  
 }  
  
 *//Парсинг фильмов* public void getFilms(){  
 *///Interceptor и Client для вывода логов при работе в сети* HttpLoggingInterceptor interceptor = new HttpLoggingInterceptor()  
 .setLevel(HttpLoggingInterceptor.Level.*BODY*);  
  
 OkHttpClient.Builder client = new OkHttpClient.Builder()  
 .addInterceptor(interceptor);  
  
 *//Retrofit - библиотека, с помощью которой ведётся работа с сервером* Retrofit retrofit = new Retrofit.Builder()  
 .baseUrl("http://cinema.areas.su")  
 .addConverterFactory(GsonConverterFactory.*create*())  
 .client(client.build())  
 .build();  
  
 API api = retrofit.create(API.class);  
   
 *//Получение списка новых фильмов* Call<ArrayList<FilmsResponse>> call;  
 call = api.getFilms();  
 AsyncTask.*execute*(()->{  
 call.enqueue(new Callback<ArrayList<FilmsResponse>>() {  
 @Override  
 public void onResponse(Call<ArrayList<FilmsResponse>> call, Response<ArrayList<FilmsResponse>> response) {  
 if (response.isSuccessful()){  
 ArrayList<FilmsResponse> films = new ArrayList<>();  
 films = response.body();  
  
 FilmAdapter adapter = new FilmAdapter(films, getContext());  
  
 recycler.setAdapter(adapter);  
 }  
 }  
  
 @Override  
 public void onFailure(Call<ArrayList<FilmsResponse>> call, Throwable t) {  
 t.printStackTrace();  
 }  
 });  
 });  
  
 }  
}

Внешний вид экрана предоставлен на рисунке ниже:

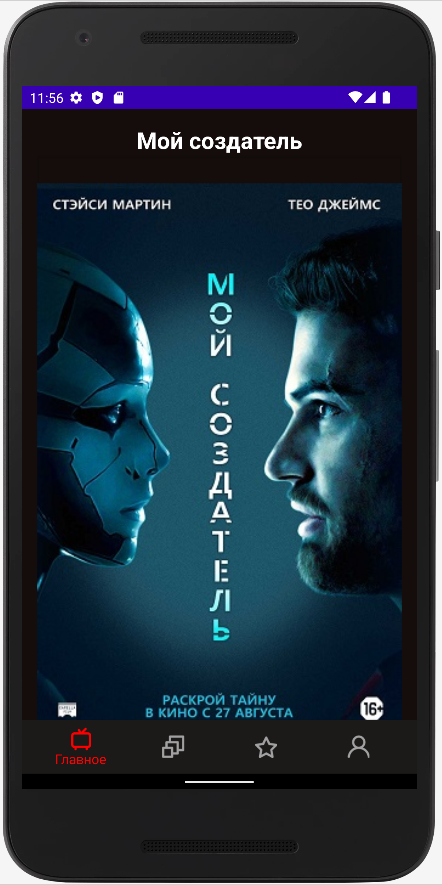


Рисунок 3. Главный экран

**Часы**

В приложении для часов необходимо было разработать систему авторизации, вывод списка новых фильмов и чатов. Разработка была начата с системы авторизации. Её листинг предоставлен ниже.

*Листинг 4. Авторизация пользователя*

public void log\_user(View v){  
 *///Interceptor и Client для вывода логов при работе в сети* HttpLoggingInterceptor interceptor = new HttpLoggingInterceptor()  
 .setLevel(HttpLoggingInterceptor.Level.*BODY*);  
  
 OkHttpClient.Builder client = new OkHttpClient.Builder()  
 .addInterceptor(interceptor);  
  
 *//Retrofit - библиотека, с помощью которой ведётся работа с сервером* Retrofit retrofit = new Retrofit.Builder()  
 .baseUrl("http://cinema.areas.su")  
 .addConverterFactory(GsonConverterFactory.*create*())  
 .client(client.build())  
 .build();  
  
 API api = retrofit.create(API.class);  
  
 *//LoginBody содержит тело запроса* LoginBody lb = new LoginBody(email.getText().toString(), password.getText().toString());  
   
 *//Авторизация пользователя* Call<LoginResponse> call;  
 call = api.log\_user(lb);  
 call.enqueue(new Callback<LoginResponse>() {  
 @Override  
 public void onResponse(Call<LoginResponse> call, Response<LoginResponse> response) {  
 if (response.isSuccessful()){  
 startActivity(new Intent(LogIn.this, MainScreen.class));  
 Toast.*makeText*(LogIn.this, response.body().getToken(), Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
  
 @Override  
 public void onFailure(Call<LoginResponse> call, Throwable t) {  
  
 }  
 });  
}

Внешний вид экрана авторизации предоставлен ниже.

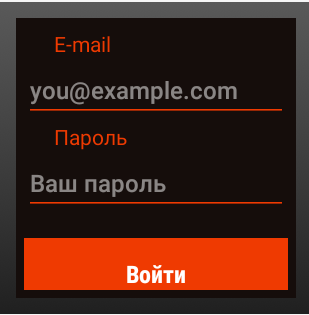


Рисунок 4. Экран авторизации

Листинг экрана со списком новых фильмов предоставлен ниже.

*Листинг 5. Получение списка новых фильмов*

*//Парсинг фильмов*public void getFilms(){  
 *///Interceptor и Client для вывода логов при работе в сети* HttpLoggingInterceptor interceptor = new HttpLoggingInterceptor()  
 .setLevel(HttpLoggingInterceptor.Level.*BODY*);  
  
 OkHttpClient.Builder client = new OkHttpClient.Builder()  
 .addInterceptor(interceptor);  
  
 *//Retrofit - библиотека, с помощью которой ведётся работа с сервером* Retrofit retrofit = new Retrofit.Builder()  
 .baseUrl("http://cinema.areas.su")  
 .addConverterFactory(GsonConverterFactory.*create*())  
 .client(client.build())  
 .build();  
  
 API api = retrofit.create(API.class);  
  
 Call<List<FilmsResponse>> call;  
 call = api.get\_films();  
  
 *//Запрос списка фильмов* AsyncTask.*execute*(()->{  
 call.enqueue(new Callback<List<FilmsResponse>>() {  
 @Override  
 public void onResponse(Call<List<FilmsResponse>> call, Response<List<FilmsResponse>> response) {  
 if (response.isSuccessful()){  
 ListView lv = findViewById(R.id.*films\_lv*);  
  
 films = response.body();  
  
 FilmsAdapter adapter1 = new FilmsAdapter(Films.this, R.layout.*film\_list\_item*, films);  
 Log.*i*("XYI", films.get(0).getName());  
 lv.setAdapter(adapter1);  
 }  
 else{  
 Log.*i*("XYI", String.*valueOf*(response.code()));  
 }  
 }  
  
 @Override  
 public void onFailure(Call<List<FilmsResponse>> call, Throwable t) {  
 t.printStackTrace();  
 }  
 });  
 });  
}

Внешний вид экрана со списком новых фильмов представлен на рисунке ниже.

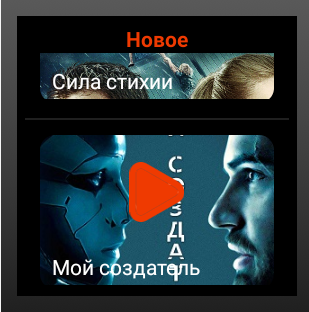


Рисунок 5. Экран "Новое"

# ПОДГОТОВКА ПРИЛОЖЕНИЯ К ПУБЛИКАЦИИ

**Создание .apk**

1. В меню приложения нажать Build
2. Выбрать Generate Signed Bundle or APK
3. Выбрать APK
4. Нажать Next
5. Указать путь до папки с ключами
6. Указать пароли, заполнить сертификат
7. Нажать OK
8. Нажать Next
9. В окне Build Variants выбрать release
10. Нажать Finish

Результат выполнения:

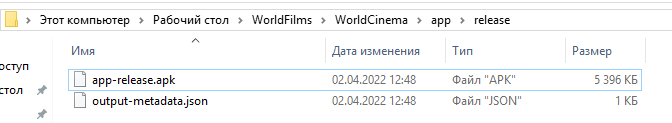


Рисунок 6. Готовый .apk файл

# ЗАКЛЮЧЕНИЕ

В ходе выполнения практики было разработано приложение WorldCinema для смартфонов и смарт-часов. За время практики я приобрёл следующие навыки:

* работа с библиотекой Retrofit;
* работа с фрагментами;
* работа с новыми для меня элементами управления;
* формирование .apk файла;
* разработка приложения для смарт-часов;
* работа с библиотекой Picasso;

Не все цели были достигнуты. Создание коллекций не было реализовано, а окно с обсуждениями готово только наполовину. Задачи, также, выполнены не все

# СПИСОК ЛИТЕРАТУРЫ

1. Barry, Burd Android Application Development All–in–One For Dummies® / Barry Burd. - Москва: **Машиностроение**, 2011. - 816 c.  
2. Биллиг, В. А. Основы объектного программирования на C# (C# 3.0, Visual Studio 2008) / В.А. Биллиг. - М.: Интернет-университет информационных технологий, Бином. Лаборатория знаний, 2010. - 584 c.  
3. Гарнаев, Андрей WEB-программирование на Java и JavaScript / Андрей Гарнаев , Сергей Гарнаев. - М.: БХВ-Петербург, **2012**. - **179** c.  
4. Голощапов, Алексей Google Android. Программирование для мобильных устройств (+ CD-ROM) / Алексей Голощапов. - М.: БХВ-Петербург, 2011. - 438 c.  
5. Голощапов, Алексей Google Android. Программирование для мобильных устройств / Алексей Голощапов. - М.: БХВ-Петербург, 2012. - 448 c.  
6. Давыдов, Станислав IntelliJ IDEA. Профессиональное программирование на Java. Наиболее полное руководство (+ CD-ROM) / Станислав Давыдов , Алексей Ефимов. - М.: БХВ-Петербург, **2011**. - 800 c.  
7. Дэрси, Лорен Android за 24 часа. Программирование приложений под операционную систему Google / Лорен Дэрси , Шейн Кондер. - М.: Рид Групп, 2011. - 464 c.  
8. Майер, Рето Android 2. Программирование приложений для планшетных компьютеров и смартфонов / Рето Майер. - М.: "Издательство "Эксмо", 2011. - 672 c.  
9. Майер, Рето Android 4. Программирование приложений для планшетных компьютеров и смартфонов / Рето Майер. - М.: Эксмо, 2013. - 816 c.  
10. Мартин, К. Соломон Oracle. Программирование на языке Java / Мартин К. Соломон, Нирва Мориссо-Леруа , Джули Басу. - М.: ЛОРИ, 2010. - 512 c.