

Министерство образования и молодежной политики Свердловской области

ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

Отчёт по учебной практике

УП 01.01

Выполнил: Лазаренко Александр Алексеевич

Группа: ПР-24

Преподаватель: Мирошниченко Г.В.

2023

Содержание

# [1.](#_Toc118960339) **[Библиотека Retorifit и Элемент WebView](#_Toc118960339)**

[3](#_Toc118960339)

[1.1 Описание задачи 3](#_Toc118960340)

[1.2 Структура проекта 3](#_Toc118960341)

[1.3 Описание разработанных функций 4](#_Toc118960342)

[1.4 Алгоритм решения 5](#_Toc118960343)

[1.5 Используемые библиотеки 5](#_Toc118960344)

[1.6 Тестовые случаи 5](#_Toc118960345)

[1.7 Используемые инструменты 5](#_Toc118960346)

[1.8 Описание пользовательского интерфейса 6](#_Toc118960347)

[1.9 Приложение (pr screen экранов) 6](#_Toc118960348)

# 1. **Библиотека Retorifit и Элемент WebView**

## Описание задачи

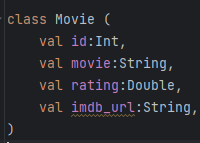
Создать проект на языке kotlin в android studio, который будет содержать и выводить на экран список фильмов содержащий их название и рейтинг, а при нажатии на фильм открывал страницу фильма на сайте.

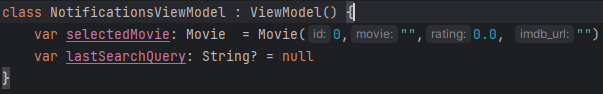
## Структура проекта

1. class MovieAdapter(private val viewModel: NotificationsViewModel) : RecyclerView.Adapter<MovieAdapter.ViewHolder>() {  
     
    private var items: List<Movie> = *emptyList*()  
    private var onClickListener: OnClickListener? = null  
    var movieList: List<Movie> = *listOf*()  
    var filteredMovieList: List<Movie> = *listOf*()  
    companion object Factory {  
    }  
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {  
    return ViewHolder(  
    MovieItemBinding.inflate(  
    LayoutInflater.from(parent.*context*), parent, false  
    )  
    )  
    }  
    fun setOnClickListener(listener: OnClickListener?) {  
    this.onClickListener = listener  
    }  
    override fun onBindViewHolder(holder: ViewHolder, position: Int) {  
    val item = items.*sortedByDescending* **{ it**.rating **}**[position]  
     
    holder.bind(item)  
    holder.itemView.setOnClickListener **{** onClickListener?.onClick(position, item)  
    **}** }  
    override fun getItemCount(): Int {  
    return items.size  
    }  
    fun refreshMovies(newItems: List<Movie>) {  
    this.items = newItems  
    movieList = newItems  
    filteredMovieList = newItems  
    notifyDataSetChanged()  
    }  
    fun refreshMoviess(newItems: List<Movie>) {  
    movieList = newItems  
    filter(viewModel.lastSearchQuery ?: "")  
    }  
    interface OnClickListener {  
    fun onClick(position: Int, model: Movie)  
    }  
    class ViewHolder(private val binding: MovieItemBinding) : RecyclerView.ViewHolder(binding.*root*) {  
    fun bind(movie: Movie) {  
    binding.MovieName.*text* = movie.movie  
    binding.ratingMovie.*text* = movie.rating.toString()  
    }  
    }  
    fun filter(query: String) {  
    filteredMovieList = if (query.*isEmpty*()) {  
    movieList  
    } else {  
    movieList.*filter* **{ it**.movie.*contains*(query, ignoreCase = true) **}** }  
    items = filteredMovieList  
    notifyDataSetChanged()  
    }  
   }

class MovieFragment : Fragment() {  
 private lateinit var binding: FragmentMovieBinding  
 private lateinit var context: Context  
 private lateinit var adapter: MovieAdapter  
 private lateinit var viewModel: NotificationsViewModel  
 override fun onCreateView(  
 inflater: LayoutInflater,  
 container: ViewGroup?,  
 savedInstanceState: Bundle?  
 ): View {  
 binding = FragmentMovieBinding.inflate(inflater, container, false)  
 context = requireContext()  
  
 viewModel = ViewModelProvider(requireActivity())[NotificationsViewModel::class.*java*]  
 adapter = MovieAdapter(viewModel)  
  
 binding.rvMovie.*layoutManager* = LinearLayoutManager(context)  
 binding.rvMovie.*adapter* = adapter  
 binding.rvMovie.setPadding(0, 0, 0, 0)  
  
 viewModel.lastSearchQuery?.*let* **{** adapter.filter(**it**)  
 binding.searchBadge.setQuery(**it**, false)  
 **}** val retrofit = Retrofit.Builder()  
 .baseUrl("https://dummyapi.online/api/")  
 .addConverterFactory(GsonConverterFactory.create())  
 .build()  
 val movieApi = retrofit.create(IMovie::class.*java*)  
 val call = movieApi.getMovies()  
 call.enqueue(object : Callback<List<Movie>> {  
 override fun onResponse(  
 call: Call<List<Movie>>,  
 response: Response<List<Movie>>  
 ) {  
 if (response.*isSuccessful*) {  
 Log.d("d", "dsa")  
 val movies = response.body()  
 if (movies != null) {  
 adapter.refreshMoviess(movies)  
 }  
 } else {  
 Toast.makeText(context, "Ошибка загрузки данных", Toast.*LENGTH\_SHORT*).show()  
 }  
 }  
 override fun onFailure(call: Call<List<Movie>>, t: Throwable) {  
 Log.e("Error", t.message.*toString*())  
 Toast.makeText(context, "Ошибка сети", Toast.*LENGTH\_SHORT*).show()  
 }  
 })  
 adapter.setOnClickListener(object : MovieAdapter.OnClickListener{  
 override fun onClick(position: Int, model: Movie) {  
 viewModel.selectedMovie = model  
 *findNavController*().navigate(R.id.*action\_navigation\_notifications\_to\_navigation\_movieweb*)  
 }  
 })  
 binding.searchBadge.setOnQueryTextListener(object : androidx.appcompat.widget.SearchView.OnQueryTextListener {  
 override fun onQueryTextSubmit(query: String?): Boolean {  
 query?.*let* **{** adapter.filter(**it**)  
 viewModel.lastSearchQuery = **it  
 }** return false  
 }  
 override fun onQueryTextChange(newText: String?): Boolean {  
 newText?.*let* **{** adapter.filter(**it**)  
 viewModel.lastSearchQuery = **it  
 }** return false  
 }  
 })  
 return binding.*root* }  
}

## 

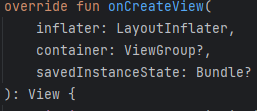




В данном проекте задействовано 4 класса и интерфейс:

* MovieFragment – фрагмент отвечает за отображение списка фильмов, их фильтрацию по поисковому запросу и обработку навигации при выборе фильма из списка
* MovieAdapter – класс использует шаблон проектирования ViewHolder для оптимизации работы RecyclerView, что позволяет более эффективно управлять списком фильмов и обеспечивает плавную прокрутку.
* IMovie – интерфейс сгенерирует и выполнит HTTP-запрос к указанному URL(movies).
* Movie – класс содержащий данные элемента которые содержатся в основном списке фильмов.
* NotificationViewModel – используется для сохранения состояния выбранного фильма и последнего поискового запроса при пересоздании активности или фрагмента, что позволяет пользовательскому интерфейсу быстро восстанавливаться после изменений конфигурации.

## 1.3 Описание разработанных функций



Инициализация функции onCreate.



Присваиваем экземпляр binding, который содержит все представления, определенные в макете фрагмента



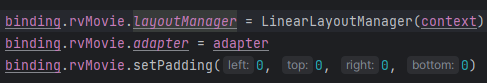
Связываем контекст с фрагментом.



## Присваиваем viewModel.



Присваиваем к adapter созданную переменную viewModel.



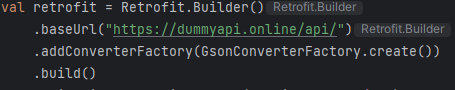
Устанавливаем LayoutManager для RecyclerView

Присваиваем адаптер RecyclerView

Устанавливаем отступы для RecyclerView

## 

Если последний запрос не null, используем его для фильтрации и установки SearchView.



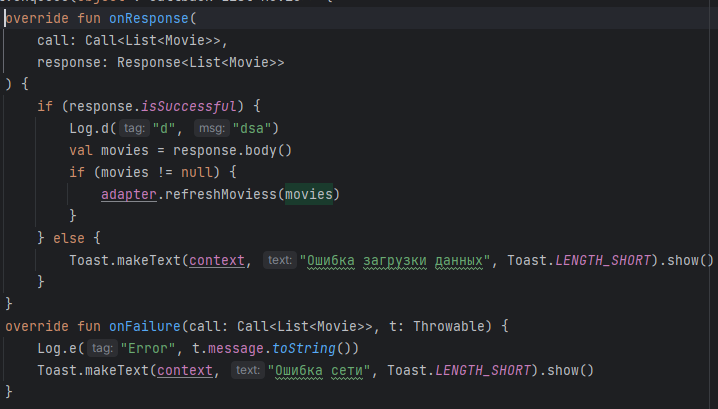
Настраиваем retrofit для сетевых запросов введя URL и конвертер.



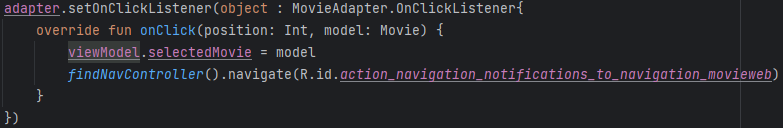
Создаём API интерфейс для запросов серверу.



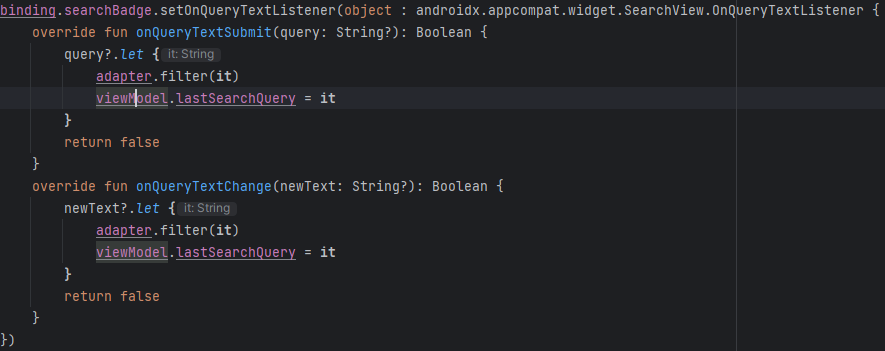
Получаем список фильмов.



При получении ответа от сервера проверяем правильно ли переданы данные и обновляем адаптер, если данные переданы неверно, то выводи сообщение о ошибке загрузки данных, если же сервер не отвечает выводим ошибку сети.

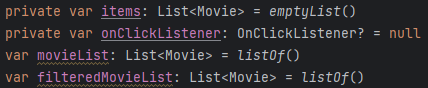


Обрабатываем нажатие на элемент списка и открываем web вкладку фильма.



Производим поиск по введённой в строку символам. Если совпадения найдены, то сортируем список по совпадающим символам и рейтингу.

Определение класса MovieAdapter, который наследуется от RecyclerView.Adapter с параметром типа MovieAdapter.ViewHolder. viewModel передается в конструктор и используется в адаптере.

Объявление приватной переменной items, которая хранит список фильмов. Изначально она инициализируется пустым списком.

Объявление приватной переменной onClickListener, которая может хранить объект слушателя кликов. Изначально она null.

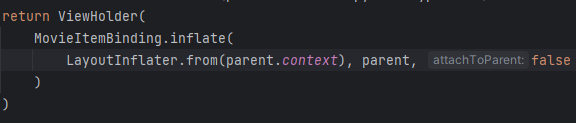
Объявление переменной movieList, которая хранит полный список фильмов.

Объявление переменной filteredMovieList, которая хранит отфильтрованный список фильмов.



Объявление companion object, который может содержать статические методы или переменные. В данном случае он пуст.

Переопределение метода onCreateViewHolder, который вызывается RecyclerView для создания нового ViewHolder.

Создание нового ViewHolder, раздувая макет элемента списка фильмов.



Метод для установки слушателя кликов на элементы списка.



Присваивание переданного слушателя переменной onClickListener.

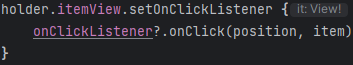
Переопределение метода onBindViewHolder, который связывает данные с ViewHolder.



Получение элемента списка фильмов, отсортированного по убыванию рейтинга, для текущей позиции.



Вызов метода bind у ViewHolder для привязки данных фильма к представлению.



Установка обработчика клика на элемент списка, который вызывает метод onClick у onClickListener.



Переопределение метода getItemCount, который возвращает количество элементов в списке.



Возвращение размера списка items.



Метод для обновления списка фильмов с учетом фильтрации по поисковому запросу.



Обновление полного списка фильмов.



Применение фильтра к списку фильмов.



Определение интерфейса OnClickListener с методом onClick.



Метод onClick, который должен быть реализован для обработки кликов по элементам списка.

Определение класса ViewHolder, который содержит привязку к макету элемента списка фильмов.

Метод bind, который связывает данные фильма с представлением.

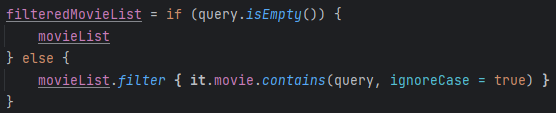


Присваивание названия фильма текстовому полю в макете.

Присваивание рейтинга фильма текстовому полю в макете.



Метод для фильтрации списка фильмов по поисковому запросу.

Присваивание отфильтрованного списка фильмов переменной filteredMovieList в зависимости от поискового запроса.

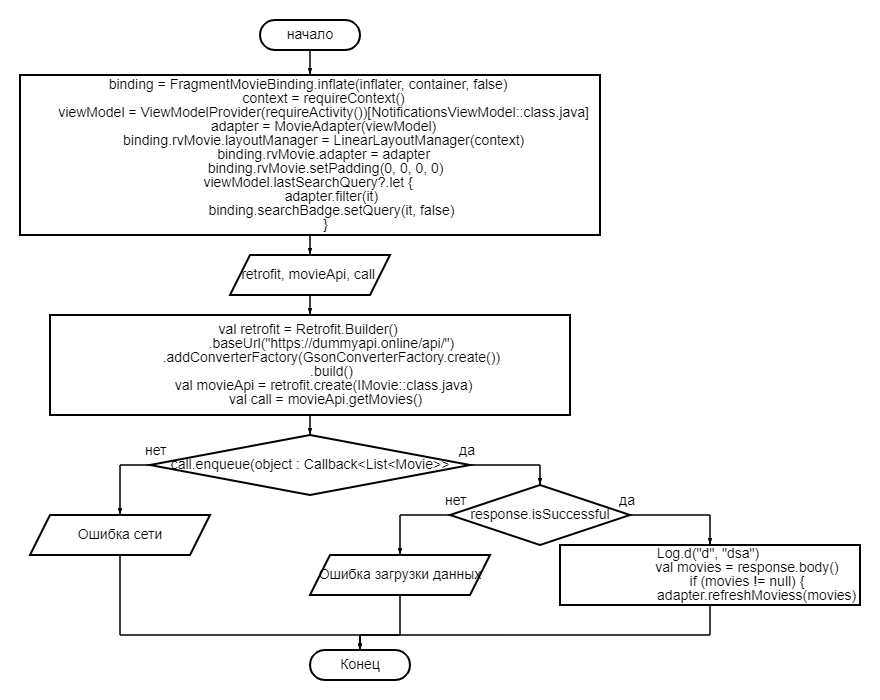


Обновление списка items отфильтрованным списком фильмов.



Уведомление RecyclerView о том, что данные изменились, и необходимо перерисовать список.

## 1.4 Алгоритм решения



## 1.5 Используемые библиотеки

*implementation*("androidx.core:core-ktx:1.13.1")  
*implementation*("androidx.fragment:fragment:1.8.0")  
*implementation*("androidx.lifecycle:lifecycle-runtime-ktx:2.8.1")  
*implementation* ("com.squareup.retrofit2:retrofit:2.9.0")  
*implementation* ("com.squareup.retrofit2:converter-gson:2.9.0")  
*implementation* ("org.jetbrains.kotlinx:kotlinx-coroutines-android:1.7.1")  
*implementation*("com.squareup.okhttp3:logging-interceptor:4.12.0")  
*implementation* ("com.auth0.android:jwtdecode:2.0.2")  
*implementation*("androidx.core:core-ktx:1.13.1")  
*implementation*("androidx.appcompat:appcompat:1.7.0")  
*implementation*("com.google.android.material:material:1.12.0")  
*implementation*("androidx.constraintlayout:constraintlayout:2.1.4")  
*implementation*("androidx.lifecycle:lifecycle-livedata-ktx:2.8.2")  
*implementation*("androidx.lifecycle:lifecycle-viewmodel-ktx:2.8.2")  
*implementation*("androidx.navigation:navigation-fragment-ktx:2.7.7")  
*implementation*("androidx.navigation:navigation-ui-ktx:2.7.7")  
*testImplementation*("junit:junit:4.13.2")  
*androidTestImplementation*("androidx.test.ext:junit:1.2.0")  
*androidTestImplementation*("androidx.test.espresso:espresso-core:3.6.0")  
*implementation*("androidx.core:core-ktx:1.13.1")  
*implementation*("androidx.fragment:fragment:1.8.0")  
*implementation*("androidx.lifecycle:lifecycle-runtime-ktx:2.8.1")  
*implementation*("androidx.activity:activity-compose:1.9.0")  
*implementation*(platform("androidx.compose:compose-bom:2023.08.00"))  
*implementation*("androidx.compose.ui:ui")  
*implementation*("androidx.compose.ui:ui-graphics")  
*implementation*("androidx.compose.ui:ui-tooling-preview")  
*implementation*("androidx.compose.material3:material3")  
*implementation*("androidx.appcompat:appcompat:1.7.0")  
*implementation*("com.google.android.material:material:1.12.0")  
*implementation*("androidx.constraintlayout:constraintlayout:2.1.4")  
*testImplementation*("junit:junit:4.13.2")  
*androidTestImplementation*("androidx.test.ext:junit:1.1.5")  
*androidTestImplementation*("androidx.test.espresso:espresso-core:3.5.1")  
*androidTestImplementation*(platform("androidx.compose:compose-bom:2023.08.00"))  
*androidTestImplementation*("androidx.compose.ui:ui-test-junit4")  
*debugImplementation*("androidx.compose.ui:ui-tooling")  
*debugImplementation*("androidx.compose.ui:ui-test-manifest")  
*implementation*("androidx.room:room-runtime:$roomVersion")  
*annotationProcessor*("androidx.room:room-compiler:$roomVersion")  
*kapt*("androidx.room:room-compiler:$roomVersion")  
*implementation*("androidx.room:room-ktx:$roomVersion")  
*implementation*("androidx.room:room-rxjava2:$roomVersion")  
*implementation*("androidx.room:room-rxjava3:$roomVersion")  
*implementation*("androidx.room:room-guava:$roomVersion")  
*testImplementation*("androidx.room:room-testing:$roomVersion")  
*implementation*("androidx.room:room-paging:$roomVersion")

## 1.6 Тестовые случаи

## 

Проверяем получены ли данные с сервера и если получены, то получены ли исправно.

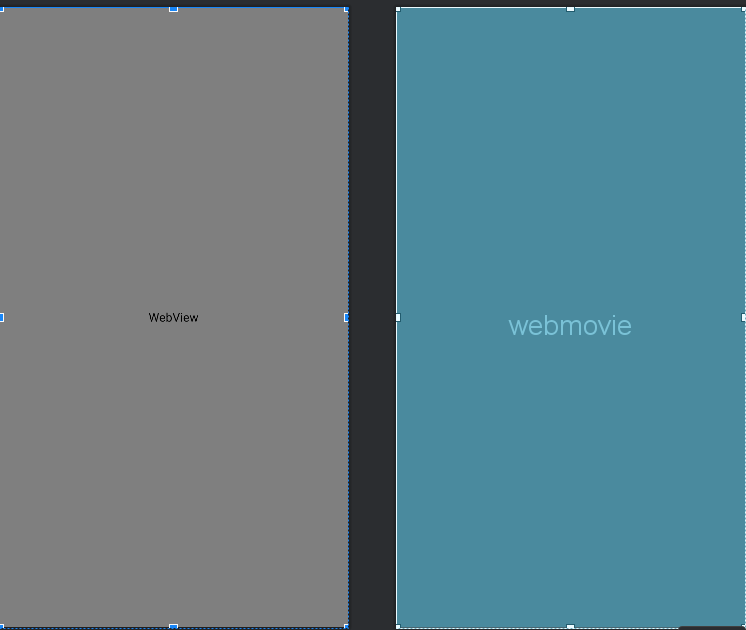
## 1.7 Используемые инструменты

Kotlin, Android Studio Hedgehog 2023.1.1 patch 2

## 1.8 Описание пользовательского интерфейса

## 

Лист фильмов



Страница с фильмом.

## 1.9 Приложение (pr screen экранов)