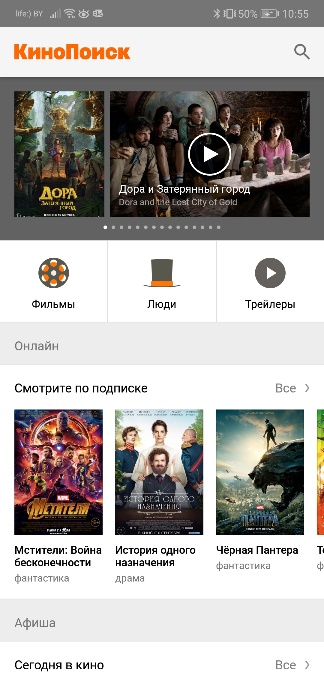
# ОБЗОР ЛИТЕРАТУРЫ

## 1.1 Обзор существующих аналогов.

На этапе проектирования системы были тщательно изучены существующие аналоги. Одним из наиболее похожих примеров является мобильное приложение «Кинопоиск» (рисунок 1.1):



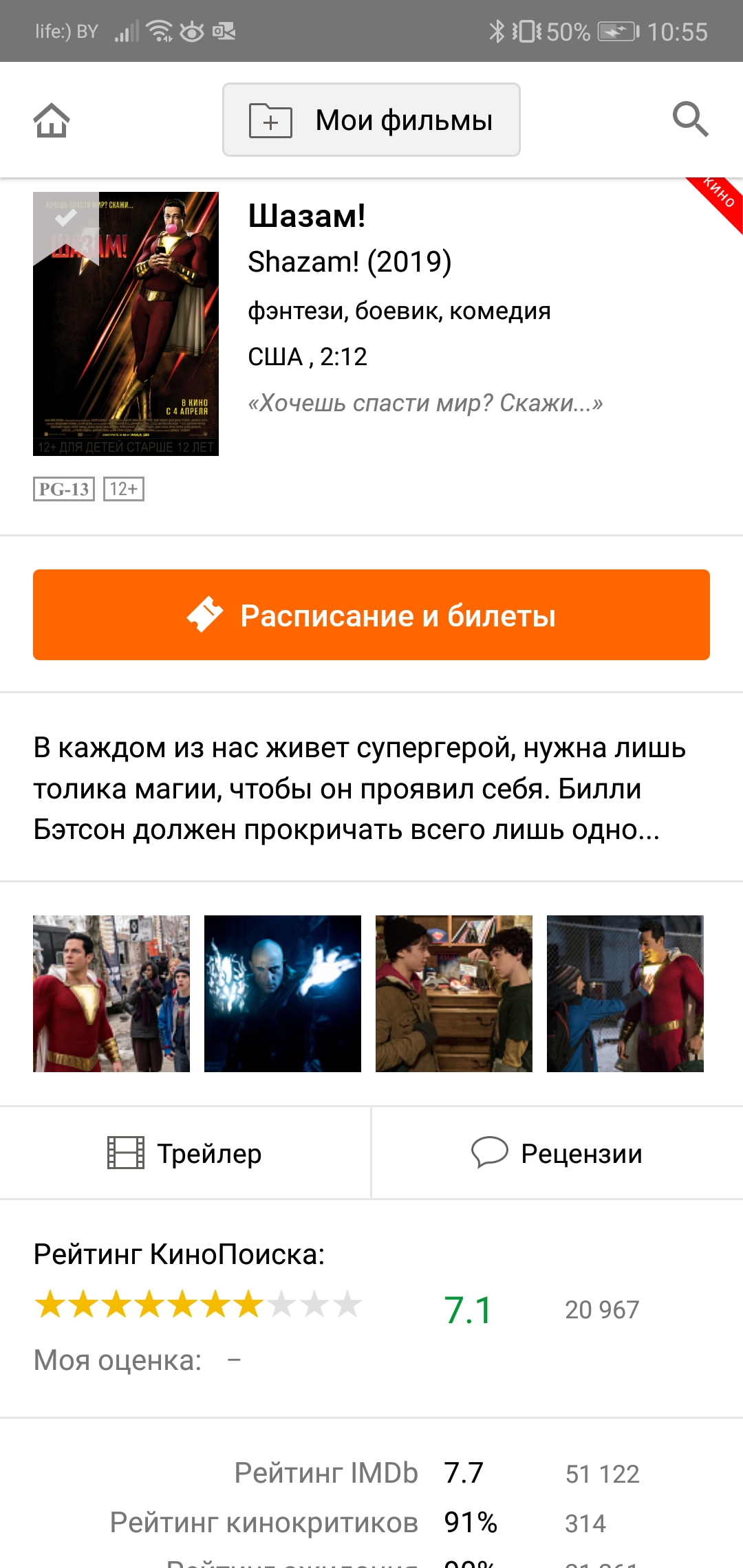
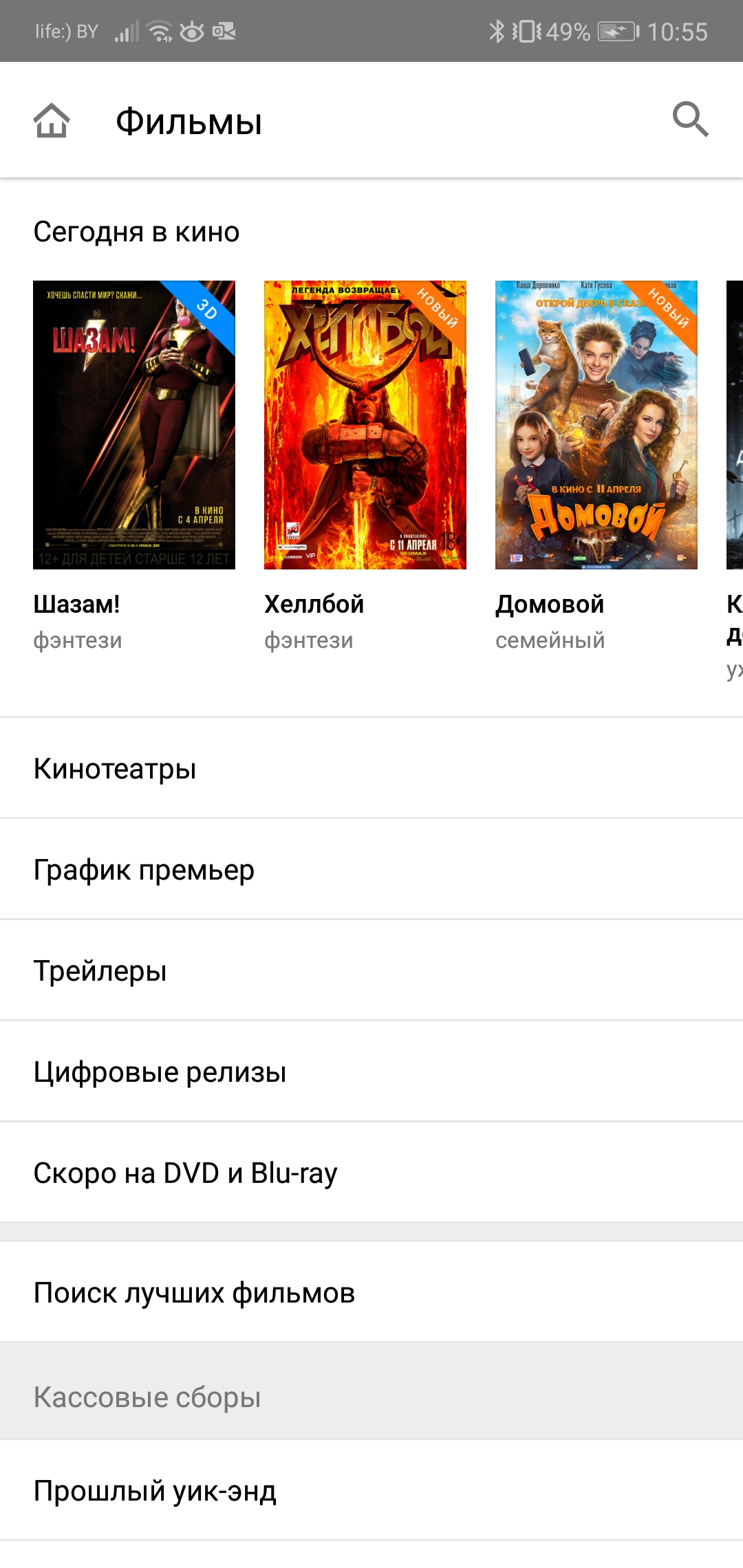


Рисунок 1.1 – Мобильное приложение «Кинопоиск»

Данное мобильное приложение обладает большим функционалом и позволяет производить поиск фильмов по различным категориям. Плюсом приложения можно выделить возможность просмотреть трейлер к фильму или сериалу. Личный профиль зрителя также присутствует. Присутствует и возможность отмечать фильмы как просмотренные, а также ставить им оценки.

Однако приложение не позволяет формировать личные предпочтения и получать рекомендации к просмотру, можно лишь просмотреть список похожих фильмов, выбрав его в библиотеке. Пользовательский интерфейс слишком загружен информацией, которую невозможно скрыть из отображения. Приложение в основном ориентировано на русскоязычную аудиторию. Также существует и веб-версия этого ресурса, практически аналогичная по своему содержанию и функционалу. Стоит также отметить, что в веб-версии этого приложения присутствует возможность просмотреть медиаконтент целиком, а не только трейлер к нему.

Еще одно похожее приложение - «IMDb Movies & TV» (рисунок 1.2). Это мобильное приложение очень похоже на «Кинопоиск». Приложение ориентировано на охват большей целевой аудитории, поддерживая множество мировых языков, однако в нём нет возможности установки русскоязычного интерфейса. Также в данном приложении присутствует возможность получать личные рекомендации к просмотру. Существует и веб-версия приложения, аналогичная по своему функционалу.

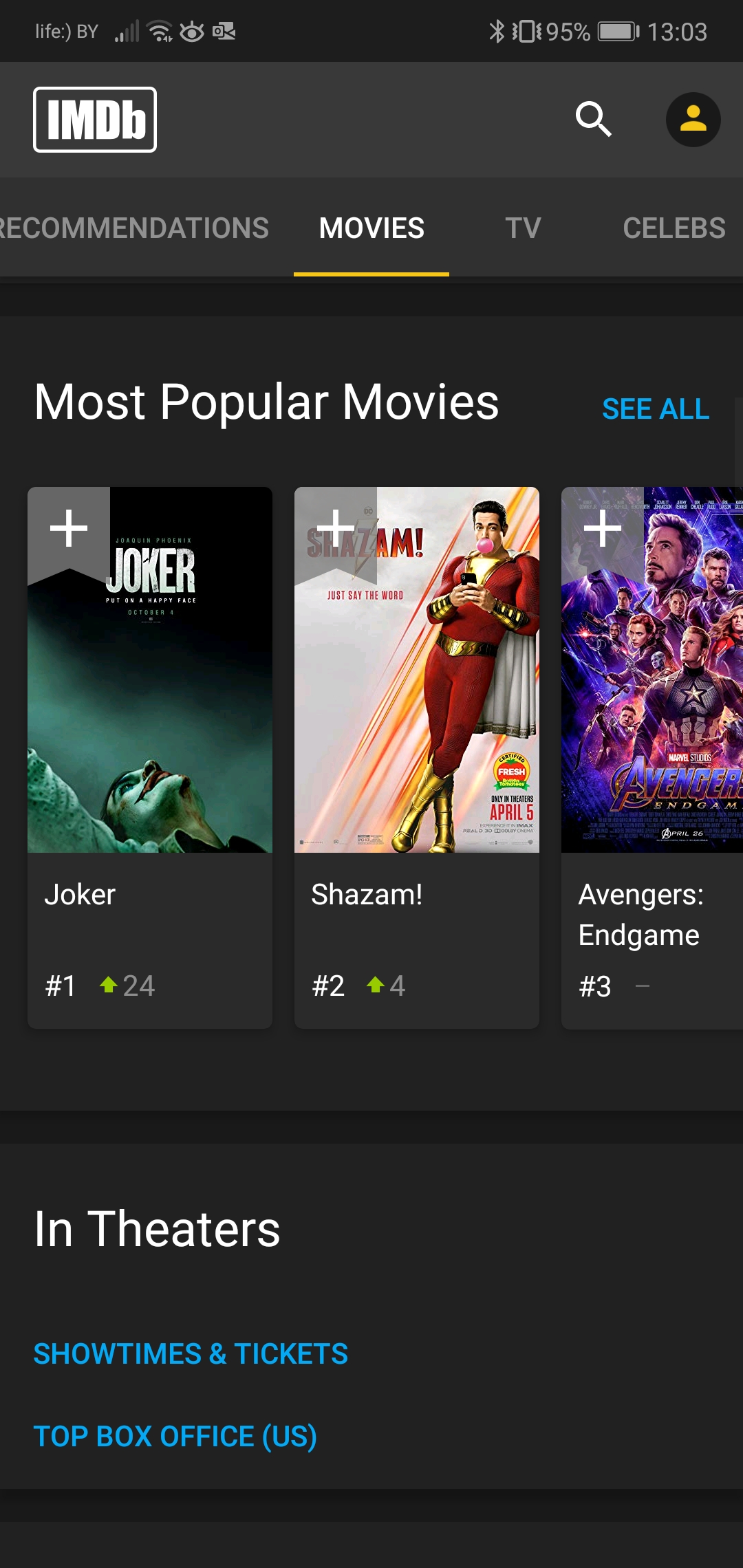
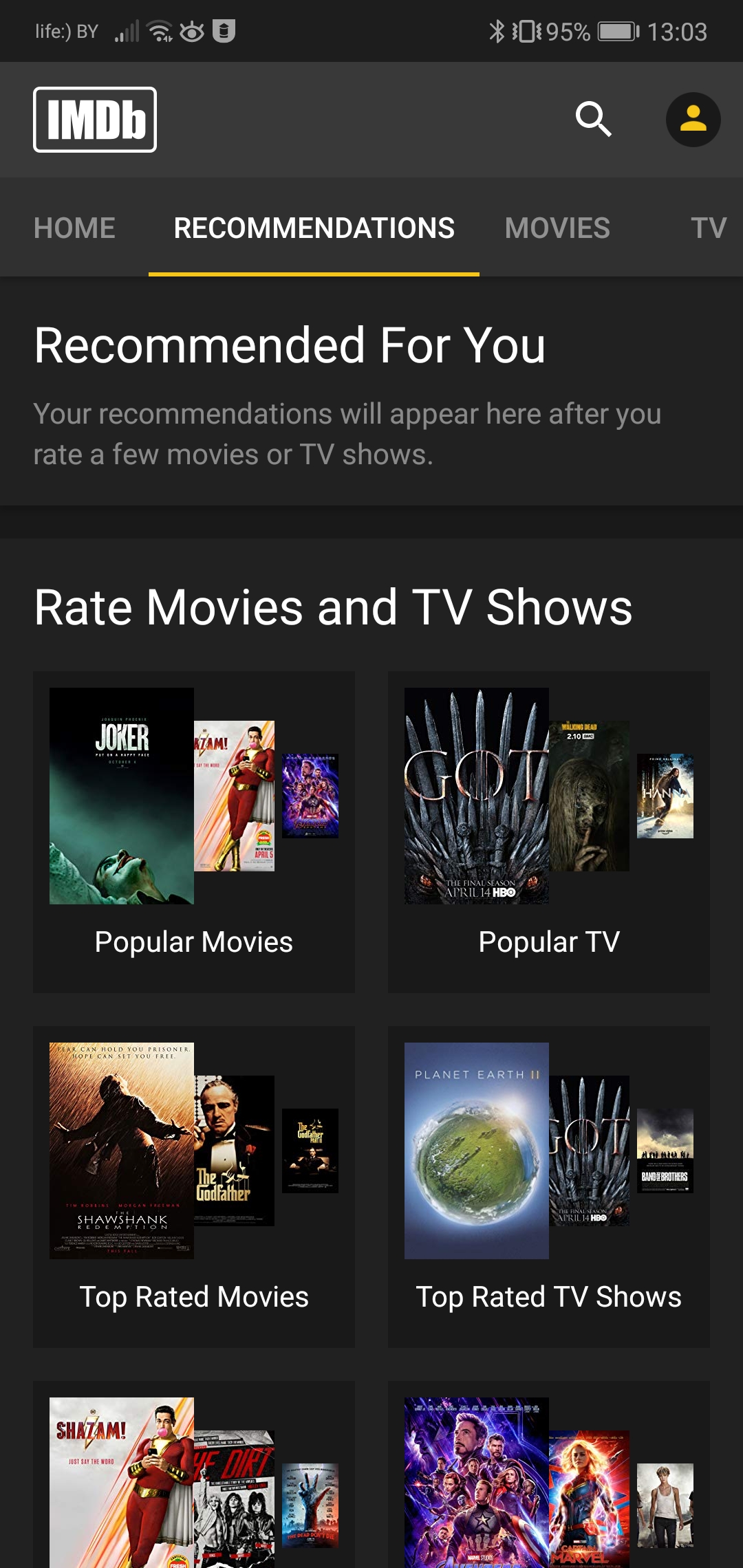


Рисунок 1.2 – Мобильное приложение «IMDb Movies & TV»

«MyShows» - в отличие от сервисов, которые описаны ранее, «MyShows» посвящён сугубо сериалам (рисунок 1.3).

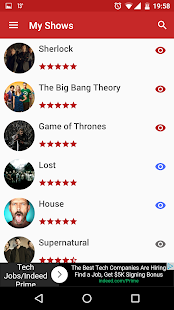
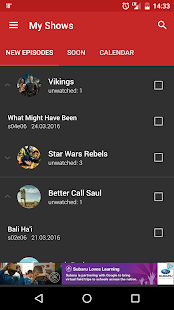
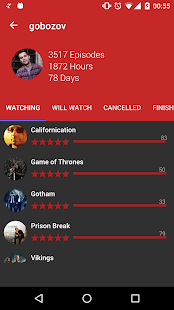


Рисунок 1.3 – Мобильное приложение «MyShows»

Это приложение не источник информации о фильмах и другом контенте, а простой и удобный трекер, который позволяет отслеживать зрительский прогресс. Из информации в приложении можно найти только описания сериалов и названия серий на русском языке. Есть также рейтинг всех шоу на основе популярности среди участников MyShows. Каждый пользователь может отмечать, сколько эпизодов каких сериалах он уже посмотрел, а также телепроекты на будущее. В результате приложение показывает количество оставшихся и просмотренных эпизодов в каждом шоу вместе с потраченным на просмотр временем. Функционала, позволяющего получать список рекомендаций к просмотру в данном сервисе нет.

## 1.2 Аналитический обзор.

Мобильное приложение – программное обеспечение, предназначенное для работы на смартфонах, планшетах и других мобильных устройствах. Многие мобильные приложения предустановлены на самом устройстве или могут быть загружены на него из онлайн-магазинов приложений, таких как App Store, Google Play и другие.

Для разработки мобильных приложений под операционную систему Android используется язык программирования Java, однако он не является единственным возможным вариантом. Тем не менее это традиционный выбор для таких приложений.

Java является одним из самых популярных языков программирования. Язык имеет большую историю - первый релиз Java состоялся ещё в 1995 году - поэтому не удивительно, что существует огромное количество учебников и специализированных сайтов, посвящённых Java-разработке. В качестве ярких примеров можно привести наиболее распространённые [1] и [2].

Java - объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems (в последующем приобретённой компанией Oracle). Приложения Java обычно транслируется в специальный байт-код, поэтому они могут работать на любой виртуальной Java-машине вне зависимости от компьютерной архитектуры. Bиртуальная машина Java (Java Virtual Machine, далее JVM)) - основная часть исполняющей системы Java, так называемой Java Runtime Environment (JRE). Виртуальная машина Java интерпретирует байт-код, предварительно созданный из исходного текста программы компилятором.

Одним из главных свойств Java, обеспечившим ему огромную популярность, является переносимость. Философия Java провозглашает принцип «Write once - run everywhere», указывающий на то, что, в теории, Java-код может быть исполнен на любой JVM, работающей на любом устройстве. В реальной жизни, конечно, данный принцип работает с большими оговорками, однако это не мешает успешной работоспособности JVM более чем на трёх миллиардах устройств. Скорее всего, благодаря заманчивой перспективе переносимости, именно Java был выбран в качестве языка программирования для операционной системы (ОС) Android. Философия платформы включает в себя варианты использования ОС на любом устройстве с любой микропроцессорной архитектурой, будь то ARM, x86 или любая другая

Одно из главных преимуществ платформы Android — ее открытость. Операционная система Android построена на основе открытого исходного кода и находится в свободном распространении. Это позволяет разработчикам получить доступ к исходному коду Android и понять, каким образом реализованы свойства и функции приложений.

Любое Android-приложение, содержит в себе хотя бы один из так называемых App Components - компонентов приложения. Существуют четыре вида основных компонентов приложения: Activity, Service, Content Provider и Broadcast Receiver.

В связи с большой популярностью мобильных разработок, существует большое количество источников информации по Android-разработке. Основным источником, в первую очередь, является официальная документация [1]. Помимо технической документации комплекта средств разработки (Software Development Kit, SDK), официальный сайт содержит множество примеров, руководства по реализации стандартных компонентов, пошаговые рекомендации по разработке приложений с нуля, а также требования к дизайну.

Одной из наиболее популярных книг является [4]. Данная книга представляет собой практический курс по написанию программного обеспечения на базе второй версии Android SDK. Несомненно, книга сильно устарела на данный момент, однако всё ещё является хорошим структурированным руководством по изучению платформы. Поскольку множество приложений создается с условием совместимости с более старыми версиями ОС Android, многие примеры из данной книги не потеряли актуальности.

Активити представляет собой один экран с пользовательским интерфейсом. Все пользовательские активити являются классами, унаследованными от стандартного класса Activity. Каждая активити обладает определённым жизненным циклом (рисунок 1.1), который является хорошей демонстрацией специфики мобильных приложений. Поскольку ресурсы мобильных устройств весьма ограничены, достоверно можно сказать лишь то, что в памяти существуют лишь те объекты, которые так или иначе относятся к отображаемому на экране содержимому. Как только активити скрывается с экрана, например, если пользователь сворачивает приложение кнопкой «Домой», активити переходит в состояние «остановлено» (stopped). Начиная с этого момента, ОС может уничтожить все объекты в оперативной памяти, относящиеся к данной активити, включая саму активити. Исходя из этого очень важно вовремя сохранять состояние приложения. Второй особенностью активити является то, что каждый раз при смене ориентации устройства с ландшафтной на портретную или наоборот, активити уничтожается и создаётся заново. Сложно сказать, чем продиктовано подобное решение, скорее всего это сделано для того, чтобы обеспечить поддержку различных вариантов пользовательского интерфейса для различной ориентации устройства. Данная особенность порождает вторую важную проблему - проблему сохранения состояния при изменении конфигурации (например, при повороте экрана). Подробнее о работе с активностью можно прочитать в специализированном разделе документации [3].

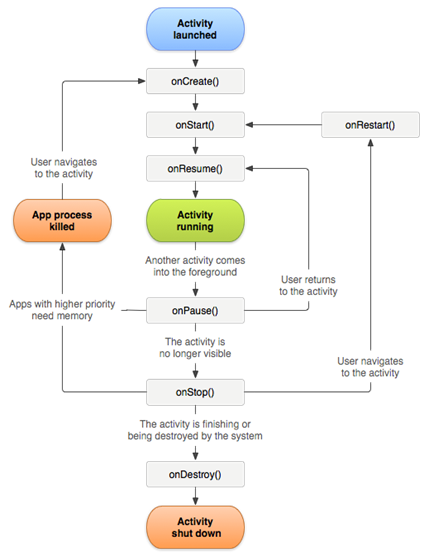


Рисунок 1.1 – Жизненный цикл активити.

Сервис - это компонент, который работает в фоне и используется для длительных и ресурсоёмких операций. Отличительной особенностью сервиса является возможность запуска в привилегированном режиме (foreground mode), благодаря которому ОС не завершает приложение в фоне максимально долгое время. Несмотря на такую возможность, разработчикам крайне не рекомендуется злоупотреблять ею и наоборот, завершать сервисы как можно раньше. Привилегированный режим обычно используется для записи или воспроизведения потокового содержимого, а также для скачивания или загрузки файлов. Помимо упоминавшихся ранее источников, хорошим руководством по работе с сервисами является [5].

Ещё одним часто используемым компонентом является Broadcast Receiver (в русскоязычной литературе встречается перевод «широковещательный приемник»). Broadcast receiver - это компонент, служащий для обработки широковещательных сообщений. ОС Android использует специальные объекты для сообщения между компонентами - интенты (intent). Иногда в литературе встречается прямой перевод - «намерения». Интент - это объект класса Intent представляющий собой некоторое сообщение. С каждым интентом может ассоциироваться некий код ошибки, действие, URI (Uniform Resource Identifier), а также произвольный набор пересылаемых объектов. Объекты, так называемые extras, хранятся в ассоциативном контейнере с уникальными строковыми ключами. Поддерживаются все примитивные типы, строки, а также объекты, реализующие интерфейсы Serializable или Parcelable. Интенты могут быть узконаправленными, например, для запуска сервиса или активити, или же широковещательными, обычно используемые в качестве нотификации о каком-либо событии. Интенты широко используются в ОС Android: благодаря широковещательным интентам, с помощью широковещательных приемников можно подписываться на различные системные события, от срабатывания будильника и до перехода по ссылке в браузере.

Последним нерассмотренным компонентом является поставщик контента - content provider. Это, пожалуй, наименее часто используемый компонент, служащий для предоставления доступа к некоторому хранилищу данных. Существуют системные поставщики контента для доступа к телефонным контактам, получения информации об установленных приложениях и многого другого. Доступ осуществляется с помощью URI по аналогии с доступом к веб-страницам. Низкая популярность данного компонента связана с его специфичностью: далеко не каждое приложение имеет необходимость в предоставлении доступа к своим данным.

Широковещательные приёмники и поставщики контента хорошо описаны во многих источниках, например, на популярном русскоязычном ресурсе, посвящённом изучению программирования для ОС Android [6].

Помимо изучения работы с основными компонентами, каждый разработчик должен ознакомится с основными шаблонами проектирования ОС Android. Наиболее часто используемыми стандартными шаблонами проектирования являются наблюдатель (observer) и слушатель (listener). Также широко используются такие принципы как обмен сообщений и позднее связывание. Помимо этого, Android диктует свои правила при реализации некоторых стандартных процессов, тем самым заставляя разработчикам следовать определённым специфическим шаблонам проектирования. Ярким примеров этого могут служить загрузчики (loaders). Загрузчики являются потомками класса Loader и служат для асинхронной загрузки данных. Главным преимуществом загрузчиков является то, что их жизненный цикл не зависит от других компонентов. Для доступа к загрузчикам используется специальный компонент Loader Manager, который позволяется работать с загрузчиками любым классам, реализующим интерфейс Loader Manager Loader Callbacks. Несмотря на некоторые встроенные во фреймворк реализации, разработчикам довольно часто приходится реализовывать свои специфические загрузчики, например, для работы с локальной базой данных или получения массивов информации по сети. Хорошую статью о практической реализации загрузчиков можно найти по адресу [7]. Там же можно отыскать и другие полезные статьи по специфическим шаблонам проектирования.

Несмотря на довольно динамичное развитие SDK и внедрение новых шаблонов проектирования, в последнее время издавалось не много литературы с упором на актуальные версии средств разработки. В качестве одного из примеров, можно привести [8]. Книга во многом схожа с [2], однако базируется на современной версии платформы и отдельно затрагивает тему обеспечения совместимости с использованием стандартных компонентов.

На данный момент весьма перспективно выглядит IDE от Google - Android Studio. Она позиционируется как специализированное средство для Android-разработки от компании разработчика самой ОС. Android Studio основывается на InelliJ IDEA и обладает всеми её достоинствами. В добавок к этому Android Studio содержит специализированные средства разработки и анализа кода, специфичные для Android-разработки. Пожалуй, единственным недостатком Android Studio является то, что она официально всё ещё находится в стадии бета-тестирования. При разработке данного проекта использовалась именно эта IDE. Подробнее ознакомится с данным продуктом можно с помощью [9]. Там же можно найти подробные описания процесса миграции проектов на новою IDE с других популярных решений.

Отличительной особенностью Android Studio является использование постепенно набирающей популярность системы сборки Gradle. Её описание можно найти на официальном сайте [10] или же на ресурсе [11], применительно конкретно к Android Studio. К особенностям Gradle, выгодно отличающей её от других систем сборки, стоит отнести сплав концепции «build by convention», используемой во всех популярных аналогах, например, Maven, с использование скриптов на языке Groovy. Подобный подход позволяет легко создавать многомодульные сборки, что весьма актуально для Android-проектов, использующих пользовательские библиотеки. Ещё одной возможностью Gradle, активно используемой в Android Studio является возможность создавать основанные на Gradle проекты, при этом фактически всё, что необходимо - это пара стандартных скриптов для построения и скачиваемый из общедоступного репозитория модуль для их обработки. Подобная лаконичная структура весьма эффектно смотрится при использовании системы контроля версий: фактически все файлы проекта являются функционально необходимыми и не захламляют репозиторий.

Стоит отметить, что все выше перечисленные среды разработки либо являются полностью бесплатными, либо имеют бесплатные версии.

В заключении следует упомянуть некоторые источники, специфичные для данного дипломного проекта. В проекте активно используется взаимодействие с базой данных (БД). На ОС Android в качестве БД использует SQLite. SQLite является простой, легковесной реляционной базой данных, используемой на всех популярных мобильных платформах (iOS, Windows Phone). Помимо официальной документации [3], описывающей шаблон проектирования для работы с БД, полезным является официальная документация БД [12]. SQLite обладает определёнными особенностями, ознакомление с которыми необходимо даже при наличии опыта работы с другими реляционными БД. SQLite содержит всего лишь четыре типа данных и не поддерживает хранимые процедуры. Вдобавок к этим ограничениям, добавляется и то, что входящие в Android SDK средства для работы БД не способствуют созданию даже самой простой структуры БД с зависимостями между таблицами, всячески подталкивая к хранению данных в нескольких больших, несвязанных таблицах.

Следует отметить, что отличительной чертой Android-разработки, с точки зрения обзора литературы, является большое число интернет-ресурсов. Это обусловлено, во-первых, популярностью платформы, а, во-вторых, особенностями мобильной разработки. Несмотря на наличие официальной документации и нескольких основных книг по разработке, на практике существует большое число нюансов, которые обычно не рассматриваются в официальных источниках.