Министерство образования Республики Беларусь Учреждение образования

«Могилевский государственный университет имени А.А.Кулешова»

Паркалов Дмитрий Александрович

**МОБИЛЬНОЕ ПРИЛОЖЕНИЕ «РАСПИСАНИЕ ОБЩЕСТВЕННОГО ТРАНСПОРТА ГОРОДА МОГИЛЕВ»**

Дипломная работа по специальности

1-31-03-4

Научный руководитель

доцент, кандидат физико-математических наук, доцент

Сергей Николаевич Батан

Дипломная работа допущена к защите

Заведующий кафедрой математики и информатики

И.В. Марченко

\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2016

Могилев

2016

# Оглавление

[Оглавление 2](#_Toc451541978)

[Введение 3](#_Toc451541979)

[Глава 1 Описание предметной области 5](#_Toc451541980)

[1.1 Классификация мобильных операционных систем 6](#_Toc451541981)

[1.2 Анализ и выбор операционной системы 7](#_Toc451541982)

[1.3 Преимущества использования операционной системы Android 8](#_Toc451541983)

[Глава 2 Выбор и описание базовых технологий для создания мобильного приложения 10](#_Toc451541984)

[2.1 Android Studio – среда разработки 10](#_Toc451541985)

[2.2 Преимущества использования Android Studio 11](#_Toc451541986)

[2.3 JAVA 13](#_Toc451541987)

[2.4 XML 14](#_Toc451541988)

[2.5 Описание Activity в Android-приложении 15](#_Toc451541989)

[2.6 Вывод 16](#_Toc451541990)

[Глава 3 Разработка мобильного приложения 17](#_Toc451541991)

[3.1 Архитектура мобильного приложения 32](#_Toc451541992)

[3.1.1 Программная часть архитектуры мобильного приложения 34](#_Toc451541993)

[3.1.2 Пользовательская часть мобильного приложения 34](#_Toc451541994)

[Заключение 36](#_Toc451541995)

[Список использованной литературы 37](#_Toc451541996)

[Приложение А 38](#_Toc451541997)

# Введение

Начиная с 2011 года мобильные технологии с каждым днем становятся все популярнее и популярнее. По данным Deloitte Canada Research, именно в этом году впервые объем продаж смартфонов и планшетных компьютеров превысил объем продаж персональных компьютеров. Этот факт не является столь удивительным в наше время, ведь мобильные устройства стали настолько мощными и доступными, что немногие могут отказаться от приобретения смартфона или планшета.

Сейчас люди могут себе позволить выполнять на мобильных устройствах все те же задачи, которые они привыкли делать на персональных компьютерах. Вместе с ростом продаж мобильных устройств наблюдается тенденция роста мобильности рабочей силы.

Таким образом данная работа направлена на увеличение продуктивности работы пользователей путем предоставления готового решения, позволяющего уточнять время интересующего их маршрута общественного транспорта нажатием нескольких кнопок.

По мере развития мобильных технологий производители карманных устройств стремятся наделить свои продукты все большим диапазоном функций. Использование языка программирования Java — один из тех подходов, который они для этого применяют.

В смарт-устройствах технологии Java играют роль посредника между операционной системой и аппаратным обеспечением. Одним из важных направлений в написании приложения является язык разметки XML. Он используется в описании различных элементов в проектах под операционную систему Android.

***Цель дипломной работы*** – создать приложение для мобильных устройств с использованием языка программирования Java.

***Задачи дипломной работы:***

1. изучение теоретических основ разработки приложений для средств мобильной связи;
2. выбор платформы и, соответственно, среды программирования, на которой будет реализовано приложение;
3. реализация приложения.

***Объектом исследования*** в данной работе являются приложения для средств мобильной связи.

***Предмет исследования:*** процесс разработки приложений для средств мобильной связи.

В качестве платформы была выбрана Android, разработка велась в среде Android Studio с использованием менеджера для создания эмуляций AVD Manager и утилиты для загрузки обновлений SDK Manager.

Работа состоит из трех глав. В первой главе приводится описание операционных систем мобильных устройств. Во второй главе рассматриваются технологии создания мобильных приложений. Глава 3 посвящена разработке мобильного приложения «Расписание общественного транспорта города Могилева».

Дипломная работа содержит && страниц, && рисунков, 1приложение, & использованных литературных источников.

# Описание предметной области

## Концепция мобильного приложения «Расписание общественного транспорта города Могилева»

За последние несколько лет стала очень актуальна разработка мобильных приложений. Кроме того, сейчас все большую популярность набирают планшетные ПК (персональные компьютеры).

Стремительное развитие продаж мобильных устройств неотъемлемо ведет к увеличению спроса на приложения для них.

Чтобы проще было сориентироваться в городе Могилеве, удобно было бы собрать всю информацию об общественном транспорте и систематизировать её. Мобильное приложение – замечательное решение, чтобы информация всегда и везде была доступна.

Объектом разработки является приложение, которое позволит собрать информацию об общественном транспорте, сгруппировать её таким образом, чтобы пользователь в несколько кликов мог получить достоверную информацию об интересующем его маршруте. Это позволит пользователю быстрее ориентироваться и сэкономит много времени.

Несомненно, с помощью глобальной сети Интернет всегда можно найти интересующий маршрут, просмотреть его расписание. Однако существует некоторое количество проблем: проблемы интернет-покрытия на окраинах города, проблема отображения веб-ресурса на разных экранах мобильных систем и прочие проблемы (например, нехватка средств, отсутствие интернета в определенном устройстве). Учитывая вышеописанные трудности и ограничения, следует задуматься над разработкой мобильного помощника, который без доступа к сети Интернет отображал бы интересующую пользователя информацию о маршрутах общественного транспорта в городе Могилеве.

## Классификация мобильных операционных систем

Существует большое количество различных мобильных операционных систем, среди которых можно выделить основных лидеров: **Android** компании Google**, iOS** компании Apple, и **Windows Phone** компании Microsoft. Это самые популярные в настоящее время мобильные ОС. А лидирующую позицию уже не первый год занимает операционная система Android.

Компания Gartner представила статистику по рынку мобильных операционных систем, в которой доля для Android устройств примерно равна 80% рынка, доля устройств на операционной системе iOS занимает около 13% рынка. Операционная система Windows занимает 3-е место с 5% рынка. Оставшиеся 2% рынка приходятся на другие, менее распространенные операционные системы, такие как Symbian, BadaOs и другие.

Android – операционная система для смартфонов, планшетных компьютеров, электронных книг, цифровых проигрывателей, "умных" наручных часов, игровых приставок, нетбуков, смартбуков, очков Google, телевизоров, систем автоматического управления автомобилем и других устройств. Операционная система основана на ядре Linux и собственной реализации виртуальной машины Java от Google. Изначально разрабатывалась компанией Android Inc., которую в 2005 году купила Google. Впоследствии Google инициировала создание альянса Open Handset Alliance (OHA), который сейчас занимается поддержкой и дальнейшим развитием платформы. Android позволяет создавать Java-приложения, управляющие устройством через разработанные Google библиотеки. Android Native Development Kit позволяет портировать (но не отлаживать) библиотеки и компоненты приложений, написанные на Си и других языках.

iOS – **закрытая операционная система** для [смартфонов](https://ru.wikipedia.org/wiki/%D0%A1%D0%BC%D0%B0%D1%80%D1%82%D1%84%D0%BE%D0%BD), [электронных планшетов](https://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D0%BD%D1%88%D0%B5%D1%82%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80) и носимых проигрывателей, разрабатываемая и выпускаемая американской компанией [Apple](https://ru.wikipedia.org/wiki/Apple). Была выпущена в 2007 году; первоначально – для iPhone и iPod, позже – для таких устройств, как iPad и Apple TV. В отличие от Windows Phone (Microsoft) и Android (Google), выпускается только для устройств, производимых фирмой Apple. В iOS используется [ядро](https://ru.wikipedia.org/wiki/%D0%AF%D0%B4%D1%80%D0%BE_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%BE%D0%B9_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B) [XNU](https://ru.wikipedia.org/wiki/XNU), основанное на [микроядре](https://ru.wikipedia.org/wiki/%D0%9C%D0%B8%D0%BA%D1%80%D0%BE%D1%8F%D0%B4%D1%80%D0%BE) [Mach](https://ru.wikipedia.org/wiki/Mach) и содержащее [программный код](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4), разработанный компанией Apple, а также код из [ОС](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) NeXTSTEP и FreeBSD. Ядро iOS почти идентично ядру настольной операционной системы Apple OS X. Начиная с самой первой версии, iOS работает только на [планшетных компьютерах](https://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D0%BD%D1%88%D0%B5%D1%82%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80) и [смартфонах](https://ru.wikipedia.org/wiki/%D0%A1%D0%BC%D0%B0%D1%80%D1%82%D1%84%D0%BE%D0%BD) с процессорами [архитектуры ARM](https://ru.wikipedia.org/wiki/ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)).

**Windows Phone – закрытая операционная система.** Она является преемником Windows Mobile, хотя и несовместима с ней, с полностью новым интерфейсом и, впервые, с интеграцией сервисов Microsoft: игрового Xbox Live и медиаплеера Zune. В отличие от предшествующей системы, Windows Phone в большей степени ориентирован на рынок потребителей, чем на корпоративную сферу. Новая операционная система Windows 10 для мобильных устройств получила название «Windows 10 Mobile», вместо Windows Phone 10.

## Анализ и выбор операционной системы

На рынке мобильных устройств можно выделить три основных лидера. Операционные системы Windows Phone и iOS имеют закрытый исходный код. Как правило, это означает, что распространяются только бинарные (скомпилированные) версии программы и лицензия подразумевает отсутствие доступа к исходному коду программы, что затрудняет создание модификаций программы. Доступ к исходному коду третьим лицам обычно предоставляется при подписании соглашения о неразглашении.

Учитывая то, что операционная система Android имеет открытый исходный код (Open Source), используется большим процентом населения, то и программный продукт должен быть реализован на операционной системе, которая доступна всем слоям населения. В первую очередь приложение должно участвовать в социальной жизни людей. Так, например, приложение «Расписание общественного транспорта города Могилева» отображает информацию о движении городского транспорта в г. Могилеве.

Операционная система должна соответствовать критериям открытости и доступности конечному пользователю.

В данном случае, данным критериям соответствует лидирующая на рынке мобильных устройств операционная система – Android.

## Преимущества использования операционной системы Android

Исходя из сделанного выбора, стоит поговорить об преимуществах данной операционной системы. Говоря о преимуществах, в первую очередь конечному пользователю хотелось бы видеть перед собой скорость работы. Не смотря на устоявшееся мнение что iOS-девайсы работают быстрее, чем Android, хотелось бы обратить внимание на следующий аспект: Android смартфонов множество, а версии их операционных систем могут быть разными, и все это отражается на общей производительности. В то же время операционная система iOS устанавливается только на конкретные устройства от компании Apple и их количество не превышает и четырех десятков. А учитывая новинки смартфонов 2016 года, нельзя упускать тот факт, что некоторые из устройств под управлением операционной системы Android во много раз превосходят новинки компаний-конкурентов под управлением операционных систем iOS и Windows Phone.

Человек с практически любым уровнем достатка может выбрать себе мобильный телефон с операционной системой Android. iOS же таких возможностей не предоставляет.

Учитывая разнообразность мобильных устройств, стоит затронуть миф об интуитивности мобильной платформы, а в частности в сети, среди пользователей, существует мнение, суть которого сводится к тому, что iOS – более интуитивная операционная система, чем Android. Но стоит внести одну небольшую поправку: большинство таких мнений исходит от людей, которые долгое время использовали устройства под операционной системой iOS, а потом переключились на Android. Из этого следует вывод, что это все дело привычки. iOS же можно назвать более простой операционной системой, так как в ней нет большинства тех возможностей, которые представлены в Android.

Поскольку Android – это открытая операционная система, она очень привлекательна для разработчиков. Одной из сильных сторон этой операционной системы является огромный выбор сторонних приложений. Android предоставляет полную свободу действий. Эта особенность важна далеко не каждому пользователю, но ведь существует большое количество сообществ, благодаря усилиям которых и появилось множество кастомных прошивок и интереснейших приложений. Но не для всех пользователей это имеет значение, они выбирают платформу в соответствие с привычкой. Однако те, кто задается целью оснастить свой смартфон полезными программами, практически всегда добивается успеха: приложения можно купить или скачать бесплатно с помощью встроенного Play Market, а также воспользоваться альтернативными магазинами и установить приложение вручную установочным файлом.

Кто-то утверждает, что подобное сравнение операционных систем вообще проводить не корректно, и попросту выбрать лучшую невозможно. В какой-то мере это так, но учитывая специфику разработки приложения, для упрощения жизни населению.

# Выбор и описание базовых технологий для создания мобильного приложения

Разработка приложения состоит из нескольких больших этапов. В первую очередь появляется основная идея и формируется список целей, которых приложения должно достичь. Цели могут быть как чисто маркетинговые, так и технические. Когда цели становятся более-менее ясны, можно задуматься о конкретном техническом воплощении.

Каждая современная мобильная платформа предоставляет инструментарий для разработчиков (SDK — software development kit), который позволяет получить доступ к большинству сервисам устройства. Разработчики SDK стремятся упростить процесс создания приложений путем решения типовых задач, с которыми сторонние разработчики сталкиваются в повседневности.

Выбор разработчиков SDK для Android в пользу Java вполне оправдан, т.к. этот язык на данный момент является, очевидно, самым популярным. Вместе с тем, он богат возможностями и может с успехом применяться в областях от программирования простых устройств до разработки огромных корпоративных приложений. Богатство возможностей предъявляет разработчику повышенные требования к знаниям языка.

Инструменты для разработки под Андроид доступны на всех популярных ОС (Windows, Linux, Mac). Тестировать приложения можно на любом поддерживаемом устройстве без каких-либо ограничений.

## Android Studio – среда разработки

официальная IDE для разработки под Android, основанная на IntelliJ IDEA от JetBrains. До появления данной среды разработки, разработчики использовали в основном Eclipse. Android Studio доступна для операционных систем Windows, Mac OS X и Linux на[официальном сайте](http://itground.ru/goto/http:/developer.android.com/sdk/installing/studio.html) разработчика. Расширенные возможности разработки и тестирования Android приложений делает Android Studio отличным инструментом для написания программ для различных устройств на ОС Android (телефоны, смартфоны, планшеты, Google TV и др.).

Основные особенности:

* Расширенный редактор макетов: WYSIWYG, способность работать с UI компонентами при помощи Drag-and-Drop, функция предпросмотра макета на нескольких конфигурациях экрана.
* Сборка приложений, основанная на Gradle.
* Различные виды сборок и генерация нескольких .apk файлов
* Статический анализатор кода (Lint), позволяющий находить проблемы производительности, несовместимости версий и другое.
* Шаблоны основных макетов и компонентов Android.
* Поддержка разработки приложений для Android Wear и Android TV.
* Встроенная поддержка Google Cloud Platform, которая включает в себя интеграцию с сервисами Google Cloud Messaging и App Engine.

Android Studio 2.1 поддерживает Android N Preview SDK, а это значит, что разработчики смогут начать работу по созданию приложения для новой программной платформы.

Новая версия Android Studio 2.1 способна работать с обновленным компилятором Jack, а также получила улучшенную поддержку Java 8 и усовершенствованную функцию Instant Run.

## Преимущества использования Android Studio

Очень трудно описать все преимущества среды разработки, однако стоит обратить внимание на главные:

1. Google inc.

Самой огромным плюсом является конечно же то, что разработчиком данной IDE является та же корпорация, которая выпускает платформу. Это означает более удобную интеграцию всех необходимых функций для новых версий Android.

2. Встроенный SDK.

При редактировании старого проекта (для старой версии API), для которого возможно нету многих стилей или любых других элементов, которых нету в проекте, Android Studio определит где и чего не хватает и отобразит окошко с уведомлением. Более того, все будет выбрано в автоматическом режиме и достаточно будет нажать кнопку “установить” - все за считанные секунды будет готово.

3. Удобный конструктор интерфейсов.

Если не заострять внимание на конструкторе, то с первого взгляда он может показаться аналогичным в Eclipse. Однако поработав с проектом, можно обратить внимание на то, что в одно нажатие можно просмотреть отображение нашего экрана на любом устройстве, вплоть до телевизоров и часов. Сами элементы интерфейса отображаются именно так, как они будут выглядеть на конкретной версии ОС, в отличии от других IDE, где виджеты отображаются на всех версиях в виде одной стандартной картинки.

4. Структура проекта.

Структура проекта изменилась в сторону заострения внимания на конкретные проблемы, а в частности работая с Eclipse постоянно мешались под рукой другие проекты, работая с одним проектом приходилось пролистывать и все остальные, с которыми в тот момент не было никакой работы. В AndroidStudio остальные проекты будут отделены отдельным окном. В целом это означает то, что тут для каждого проекта запускается отдельная версия Android Studio и открыть в одном, главном окне программы, файлы из разных проектов не удастся.

5. Удобный дизайн.

Дизайн всей студии выполнен не в особом гугловском стиле, он скорее просто перешёл из IDEA с минимальными изменениями, но все же некоторые улучшения он потерпел. Очень большую часть пространства занимают окна логов, сообщений, свойств элементов и самого проекта, но для каждого окна предусмотрена функция сворачивания в нижнюю часть интерфейса. Более нету необходимости, как это было в Eclipse, закрывать окна по ненадобности, а потом выискивать их в меню Window, окошки будут находится в одном месте на нижней панели в виде интуитивно понятных иконок.

6. Логи.

Хоть в данном случае и присутствует привычный logcat , но все же отслеживание ошибок стало куда удобнее. Помимо привычного лога можно разделить отслеживание для различных процессов, потоков и приложений, что крайне необходимо делать, так как зачастую приложения будут использовать создания новых потоков, а в отдельных случаях еще и взаимодействовать с другими приложениями.

В целом главным достоинством студии стала тесная интеграция с android SDK, что напрямую улучшает не только удобство разработки, но и качество самих приложений, ведь некоторые функции тут строятся не обходными путями, а именно так, как это продумал сам владелец данной ОС.

## JAVA

Язык Java зародился как часть проекта создания передового программного обеспечения (ПО) для различных бытовых приборов. Реализация проекта была начата на языке С++, но вскоре возник ряд проблем, наилучшим средством борьбы с которыми было изменение самого инструмента - языка программирования. Стало очевидным, что необходим платформо-независимый язык программирования, позволяющий создавать программы, которые не приходилось бы компилировать отдельно для каждой архитектуры и можно было бы использовать на различных процессорах под различными операционными системами. Язык Java потребовался для создания интерактивных продуктов для сети Internet. Фактически, большинство архитектурных решений, принятых при создании Java, было продиктовано желанием предоставить синтаксис, сходный с Си и Cи++. В Java используются практически идентичные соглашения для объявления переменных, передачи параметров, операторов и для управления потоком выполнением кода. В Java добавлены все хорошие черты C++.

Три ключевых элемента объединились в технологии языка Java:

- Java предоставляет для широкого использования свои апплеты (applets) — небольшие, надежные, динамичные, не зависящие от платформы активные сетевые приложения, встраиваемые в страницы Web. Апплеты Java могут настраиваться и распространяться потребителям с такой же легкостью, как любые документы HTML.

- Java высвобождает мощь объектно-ориентированной разработки приложений, сочетая простой и знакомый синтаксис с надежной и удобной в работе средой разработки. Это позволяет широкому кругу программистов быстро создавать новые программы и новые апплеты.

- Java предоставляет программисту богатый набор классов объектов для ясного абстрагирования многих системных функций, используемых при работе с окнами, сетью и для ввода-вывода. Ключевая черта этих классов заключается в том, что они обеспечивают создание независимых от используемой платформы абстракций для широкого спектра системных интерфейсов.

## XML

Расширяемый язык разметки (Extensible Markup Language, XML), созданный в 1996 году Консорциумом W3C (World Wide Web Consortium), является подклассом стандартного языка разметки (Standard Generalized Markup Language, SGML). XML был задуман как гибкий и в то же время формальный метаязык для использования в Интернете.

Метаязык (metalanguage) — это язык, предназначенный для описания других языков. Например, можно сказать, что словарь английского языка в совокупности с английской грамматикой образуют метаязык, описывающий английский язык.

Что касается языка XML, то назначение его — описывать языки разметки. В языке разметки (markup language) для структурирования данных используются теги. Язык гипертекстовой разметки (Hypertext Markup Language, HTML), наиболее распространенный на сегодняшний день язык разметки, исходно был написан на SGML, но мог бы быть и в какой-то степени был написан и на XML.

Язык HTML был задуман для решения специфических задач — разметки документов научной и академической направленности. Если вы занимались какой-либо деятельностью по разработке web-страниц, то хорошо знаете, что в настоящее время HTML «трещит по всем швам», так как на протяжении многих лет делаются попытки дополнить его так, чтобы он соответствовал всем запросам web-программистов. Многие называют XML заменой HTML, но это не совсем точное высказывание.

В то время как HTML содержит фиксированный набор тегов, в XML теги вообще отсутствуют. Вместо этого XML позволяет программисту самому создать такой язык разметки, который в точности соответствует требованиям конкретного приложения. В нашего курса мы проиллюстрируем процесс создания приложения, относящегося к области электронной коммерции. Язык разметки, который мы будем использовать, содержит теги, выглядящие осмысленно именно в этой области (такие, например, как <price> и <quantity>) [Price — цена, quantity — количество. — Примеч. перев. ].

В приложениях XML обычно используются следующие типы данных и вспомогательные функции:

* сам файл XML, имеющий строго определенную структуру;
* определение типа документа (Document Type Definition, DTD), где определяется структура файла XML (необязательный элемент);
* таблицы стилей, содержащие информацию о том, как данные должны быть отформатированы при выводе (необязательный элемент);
* процессор XML и различные служебные функции для манипулирования данными и переформатирования данных.

## Описание Activity в Android-приложении

При создании приложения, были созданы следующие Activity компоненты:

About ­ Activity в котором описано приложение, его обновления, новости изменений.

Bus1-…-Bus46­ Activity маршрутов автобусов работающих в городе Могилев.

Dictionary­ Activity содержащие в себе словарь сокращений, которые встречаются в описании некоторых маршрутов.

MainActivity­ Activity главного экрана.

MainBus­ Activity выбора маршрутов автобуса.

MainTrollbus­ Activity выбора маршрутов троллейбуса.

Minibus­ Activity с выбором маршрутов маршрутного транспортного средства.

SplashScreen­ Activity отображающее заставку при запуске приложения.

Taxi­ Activity содержащие в себе контактную информацию об Taxi города.

Trollbus2-…-Trollbus7­ Activity маршрутов троллейбусов работающих в городе Могилев.

## Вывод

Разрабатывая приожение, стоит думать также о фактарах взаимодействия пользователя и информации. К примеру на окраинах большого города могут встречаться проблемы с покрытием в зоне интернет. Для того, что бы обезопасить пользователя от проблемы доступа к информации, стоит задуматься об разработке Offline-справочника маршрутов общественного транспорта. Из преимуществ такого выбора, можно выделить тот фактор, что в любой критической ситуации (нехватка средств для выхода в сеть интернет, практически посаженная батарея или просто плохое покрытие в зоне интернет), пользователь всегда сможет получить информацию об интересующем его маршруте. В разработке подобного приложения стоит учесть широко-распространенную версию Android 4.0 и выше.

# Разработка мобильного приложения

После составления и изучения технического задания был разработан дизайн приложения, который отвечал бы следующим параметрам: ничего лишнего доступно удобно

Следующим этапом была непосредственно программная реализация приложения. Приложение состоит из нескольких частей. Главной частью приложения является Activity.

Все приложения Android имеют жизненный цикл. При запуске пользователем приложения система дает этому приложению высокий приоритет. Каждое приложение запускается в виде отдельного процесса, что позволяет системе давать одним процессам более высокой приоритет, в отличие от других. Благодаря этому, например, при работе с одними приложениями не блокировать входящие звонки. После прекращения работы с приложением, система освобождает все связанные ресурсы и переводит приложение в разряд низкоприоритетного и закрывает его.

Все объекты activity, которые есть в приложении, управляются системой в виде стека activity, который называется back stack. При запуске новой activity она помещается поверх стека и выводится на экран устройства, пока не появится новая activity. Когда текущая activity заканчивает свою работу (например, пользователь уходит из приложения), то она удаляется из стека, и возобновляет работу та activity, которая ранее была второй в стеке.

Каждая activity имеет три состояния:

Активно (Resumed): activity отображается на экране и взаимодействует с пользователем

Приостановлено (paused): activity по-прежнему видна, однако она теряет фокус. На данном этапе в ранг активного переводится другая activity, которая может быть либо невидимой, либо занимать весь экран.

Остановлено (stopped): activity сохраняет свое состояние, однако она уже невидима пользователю, и если другим процессам потребуется память, то она будет завершена.

После перевода activity в приостановленное или остановленное состояние система может удалить activity из памяти, либо уведомив о завершении процесса, либо просто убив соответствующий процесс. При повторном запуске и выводе на экран activity полностью пересоздается и начинает свою работу заново.

Первое что необходимо реализовать – это Activity главного экрана, поскольку именно от главного экрана мы продолжим создание последующих Activity. Внешний вид описывается и верстается в режиме реального времени в соответствующем XML документе. В RelativeLayout дочерние элементы расположены так, что если первый элемент расположен по центру экрана, другие элементы, выровненные относительно первого элемента, будут выровнены относительно центра экрана. При таком расположении, при объявлении разметки в XML-файле, элемент, на который будут ссылаться для позиционирования другие объекты представления, должен быть объявлен раньше, чем другие элементы, которые обращаются к нему по его идентификатору.

*<?***xml version="1.0" encoding="utf-8"***?>*<**RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:paddingBottom="@dimen/activity\_vertical\_margin"  
 android:paddingLeft="@dimen/activity\_horizontal\_margin"  
 android:paddingRight="@dimen/activity\_horizontal\_margin"  
 android:paddingTop="@dimen/activity\_vertical\_margin"  
 app:layout\_behavior="@string/appbar\_scrolling\_view\_behavior"  
 tools:context="com.example.mogtransport.MainActivity"  
 tools:showIn="@layout/activity\_main"  
 android:background="@drawable/logo\_alpha"** >  
</**RelativeLayout>**

**RelativeLayout** содержит Button(обычная кнопка) и несколько ImageButton(представляет собой кнопку с изображением вместо текста).

<**Button  
 style="?android:attr/buttonStyleSmall"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="О программе"  
 android:id="@+id/button3"  
//Задается действие по событию «нажатие на клавишу»   
 android:onClick="about"  
 android:layout\_alignParentBottom="true"  
 android:layout\_alignRight="@+id/imageButton3"  
 android:layout\_alignEnd="@+id/imageButton3"  
 android:layout\_alignParentLeft="true"  
 android:layout\_alignParentStart="true"  
//Цвет фона кнопки  
 android:background="#a15ea6b8"  
 android:alpha="0.8"** />  
  
 <**ImageButton  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/imageButton"  
 android:src="@drawable/bus"  
 android:onClick="busbus"  
 android:layout\_alignParentTop="true"  
 android:layout\_alignParentRight="true"  
 android:layout\_alignParentEnd="true"  
 android:background="#4c7ebc"  
 android:clickable="false"  
 android:alpha="0.8"** />  
  
 <**ImageButton  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/imageButton2"  
 android:src="@drawable/trollebus"  
 android:onClick="trollbustrollbus"  
 android:background="#ffffff"  
 android:layout\_below="@+id/imageButton"  
 android:layout\_alignParentRight="true"  
 android:layout\_alignParentEnd="true"  
 android:alpha="0.8"** />  
  
 <**ImageButton  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/imageButton3"  
 android:src="@drawable/minibus"  
 android:onClick="minibusminibus"  
 android:background="#558fcc"  
 android:layout\_below="@+id/imageButton2"  
 android:layout\_alignParentRight="true"  
 android:layout\_alignParentEnd="true"  
 android:alpha="0.8"** />  
  
 <**ImageButton  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/imageButton4"  
 android:src="@drawable/taxi"  
 android:onClick="taxitaxi"  
 android:background="#558fcc"  
 android:layout\_below="@+id/imageButton3"  
 android:layout\_alignParentRight="true"  
 android:layout\_alignParentEnd="true"  
 android:alpha="0.8"** />  
</**RelativeLayout**>

После описания графической части, нужно создать соответственный JAVA-класс MainActivity, в которым будут описаны наши клавиши:

**package** com.example.mogtransport;  
  
**import** android.content.Intent;  
**import** android.os.Bundle;  
**import** android.support.design.widget.FloatingActionButton;  
**import** android.support.design.widget.Snackbar;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.support.v7.widget.Toolbar;  
**import** android.view.View;  
**import** android.view.Menu;  
**import** android.view.MenuItem;  
**import** android.widget.Toast;  
  
 **public class** MainActivity **extends** AppCompatActivity {  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_main);  
 }  
 @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, 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.action\_settings) {  
 **return true**;  
 }  
 **return super**.onOptionsItemSelected(item);  
 }

//Клавиши с выпадающим Toast сообщением  
 **public void** about(View view) {  
 Toast.*makeText*(getApplicationContext(), **"О программе"**, Toast.***LENGTH\_SHORT***).show();  
 Intent intent = **new** Intent(MainActivity.**this**, about.**class**);  
 startActivity(intent);  
 }  
**public void** busbus(View view) {  
 Toast.*makeText*(getApplicationContext(), **"Расписание автобусов"**, Toast.***LENGTH\_SHORT***).show();  
 Intent intent = **new** Intent(MainActivity.**this**, MainBus.**class**);  
 startActivity(intent);  
 }  
  
 **public void** trollbustrollbus(View view) {  
 Toast.*makeText*(getApplicationContext(), **"Расписание троллейбусов"**, Toast.***LENGTH\_SHORT***).show();  
 Intent intent = **new** Intent(MainActivity.**this**, MainTrollbus.**class**);  
 startActivity(intent);  
 }  
  
 **public void** minibusminibus(View view) {  
 Toast.*makeText*(getApplicationContext(), **"Расписание маршруток"**, Toast.***LENGTH\_SHORT***).show();  
 Intent intent = **new** Intent(MainActivity.**this**, minibus.**class**);  
 startActivity(intent);  
 }  
  
 **public void** taxitaxi(View view) {  
 Toast.*makeText*(getApplicationContext(), **"TAXI"**, Toast.***LENGTH\_SHORT***).show();  
 Intent intent = **new** Intent(MainActivity.**this**, Taxi.**class**);  
 startActivity(intent);  
 }  
 }

При запуске приложения, пользователь видит заставку приложения(Рисунок 1), которая создается в отдельном Activity(SplashScreen) и её графическая часть описывается в соответствующем XML документе.

JAVA:

**package** com.example.mogtransport;  
  
**import** android.content.Intent;  
**import** android.os.Bundle;  
**import** android.support.design.widget.FloatingActionButton;  
**import** android.support.design.widget.Snackbar;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.support.v7.widget.Toolbar;  
**import** android.view.View;  
  
**public class** SplashScreen **extends** AppCompatActivity {  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_splash\_screen***);  
 *//Создаем новый поток:* Thread splash\_time = **new** Thread()  
 {  
 **public void** run()  
 {  
 **try** {  
 *//Целое значение время отображения картинки:* **int** SplashTimer = 0;  
 *//Запускаем цикл длиной в 2 секунды:* **while**(SplashTimer < 2000) {  
 *sleep*(100);  
 SplashTimer = SplashTimer +100;  
 };  
 }  
 **catch** (InterruptedException e) {  
 e.printStackTrace(); }  
 **finally** {  
 *//Закрываем activity:* finish();  
 }startActivity(**new** Intent(SplashScreen.**this**,MainActivity.**class**));  
 }  
 };  
 splash\_time.start();  
 }  
}

XML:

*<?***xml version="1.0" encoding="utf-8"***?>*<**RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:paddingBottom="@dimen/activity\_vertical\_margin"  
 android:paddingLeft="@dimen/activity\_horizontal\_margin"  
 android:paddingRight="@dimen/activity\_horizontal\_margin"  
 android:paddingTop="@dimen/activity\_vertical\_margin"  
 app:layout\_behavior="@string/appbar\_scrolling\_view\_behavior"  
 tools:context="com.example.mogtransport.SplashScreen"  
 tools:showIn="@layout/activity\_splash\_screen"  
 android:background="@drawable/logo"**>  
</**RelativeLayout**>



Заставка приложения



Далее пользователь попадает в главное меню, в котором отображается полноценный список выбора общественного транспорта (Рисунок 2). С помощью нажатий можно перейти на определенный вид транспорта. При нажатии открывается новое Activity.



Главное меню с выбором общественного транспорта

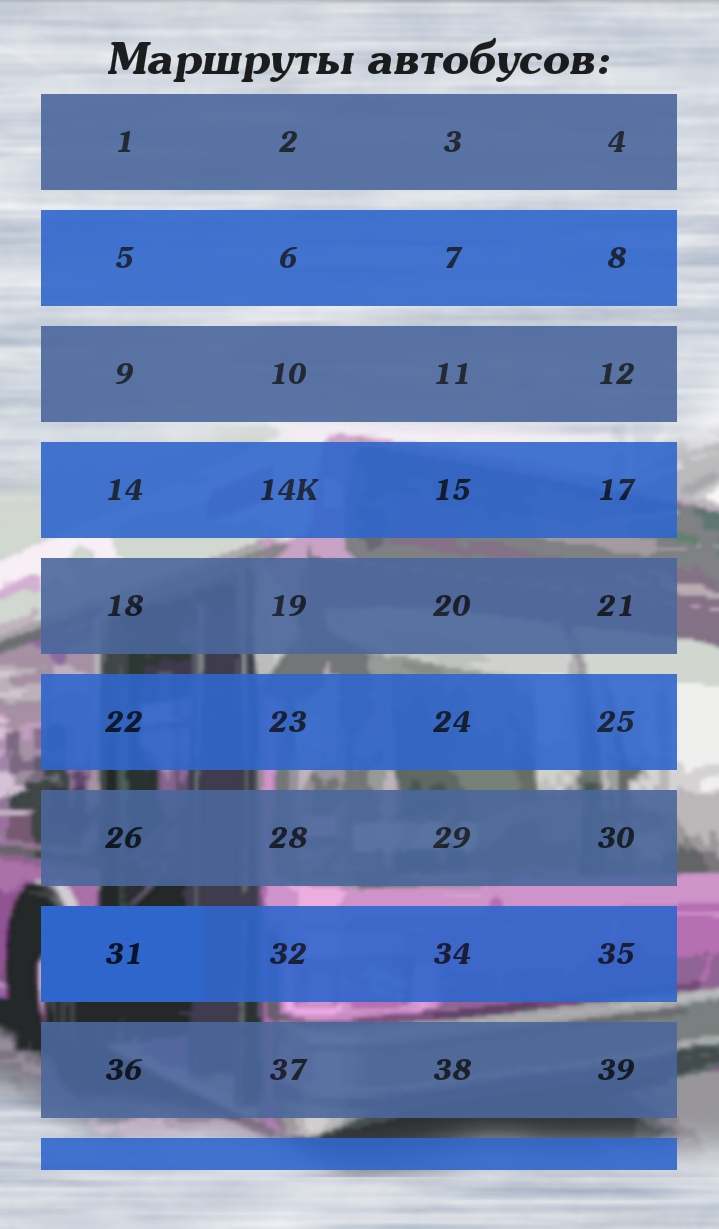


В главном меню перед пользователем стоит выбор между общественным транспортом. Когда пользователь определяется с выбором, нажатием на нужную кнопку он открывает следующее Activity в котором на выбор ему предлагаются различные маршруты определенного вида городского общественного транспорта.

Если пользователь выбирает кнопку, на которой изображен автобус, то ему открывается соответствующий список всех возможных маршрутов (Рисунок 3).



Меню с выбором нужного маршрута

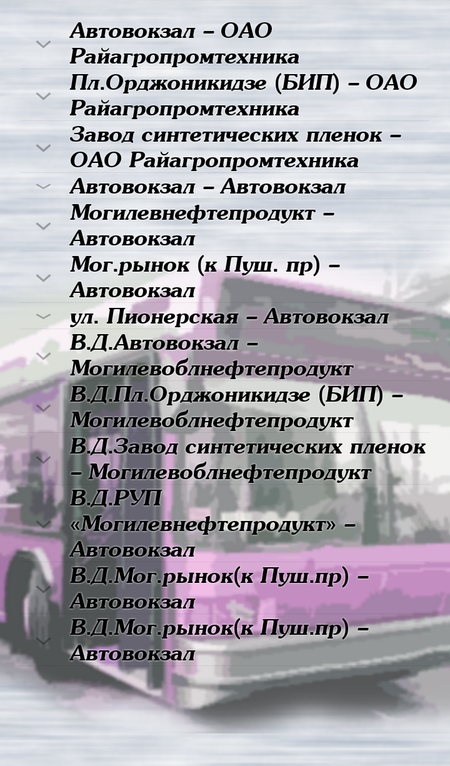


Похожая реализация и у остальных видов общественного транспорта (Троллейбус, маршрутное такси, такси), отличие только в названии кнопок и в Activty на которое переходит пользователь.

Однако, вернувшись к списку маршрутов автобусов, перед пользователем представлен выбор, и если выбрать определенный маршрут, то откроется новое Activity в котором отображается сворачиваемый список (ExpandableList) представленный на рисунке 3.4.



Список остановок одного из маршрутов



Подобная реализация достигается с помощью элемента ExpandList и правильного описания оного. В первую очередь требуется описать элемент ExpandList в соответствующих XML документах. Требуется создать два документа, поскольку в одном будет описан вид самого списка, во втором же ­ вложенные элементы списка.

List\_group.xml

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:orientation="vertical" android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"**>  
  
 <**TextView  
 android:id="@+id/lblListHeader"  
 android:layout\_width="fill\_parent"  
 android:layout\_height="wrap\_content"  
 android:paddingLeft="?android:attr/expandableListPreferredItemPaddingLeft"  
 android:textSize="17dp"  
 android:shadowColor="#fefffe"  
 android:shadowRadius="3.0"  
 android:textColor="#000000"  
 android:textStyle="bold"** />  
</**LinearLayout**>

List\_item.xml

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:orientation="vertical" android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"**>  
  
 <**TextView  
 android:id="@+id/lblListItem"  
 android:layout\_width="fill\_parent"  
 android:layout\_height="wrap\_content"  
 android:textSize="17dip"  
 android:paddingTop="5dp"  
 android:paddingBottom="5dp"  
 android:shadowRadius="3.0"  
 android:textColor="#000000"  
 android:paddingLeft="?android:attr/expandableListPreferredChildPaddingLeft"** />  
</**LinearLayout**>

После описания XML документа, следует заполнить соответствующий JAVA файл:

**package** com.example.mogtransport;  
  
**import** android.os.Bundle;  
**import** android.support.design.widget.FloatingActionButton;  
**import** android.support.design.widget.Snackbar;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.support.v7.widget.Toolbar;  
**import** android.view.View;  
**import** java.util.ArrayList;  
**import** java.util.HashMap;  
**import** java.util.List;  
**import** android.app.Activity;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.ExpandableListView;  
**import** android.widget.ExpandableListView.OnChildClickListener;  
**import** android.widget.ExpandableListView.OnGroupClickListener;  
**import** android.widget.ExpandableListView.OnGroupCollapseListener;  
**import** android.widget.ExpandableListView.OnGroupExpandListener;  
**import** android.widget.Toast;  
  
**public class** Bus3 **extends** AppCompatActivity {  
  
 ExpandableListAdapter **listAdapter**;  
 ExpandableListView **expListView**;  
 List<String> **listDataHeader**;  
 HashMap<String, List<String>> **listDataChild**;  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_bus3***);  
  
 *//Связываемся с нашим ExpandableListView:* **expListView** = (ExpandableListView) findViewById(R.id.***lvExp***);  
 *//Подготавливаем список данных:* prepareListData();  
 **listAdapter** = **new** ExpandableListAdapter(**this**, **listDataHeader**, **listDataChild**);  
  
 *//Настраиваем listAdapter:* **expListView**.setAdapter(**listAdapter**);  
 **expListView**.setOnChildClickListener(**new** OnChildClickListener() {  
 *//Обработчик нажатий по пунктам меню:* @Override  
 **public boolean** onChildClick(ExpandableListView parent, View v,  
 **int** groupPosition, **int** childPosition, **long** id) {  
 Toast.*makeText*(  
 getApplicationContext(),  
 **listDataHeader**.get(groupPosition)  
 + **" : "** + **listDataChild**.get(  
 **listDataHeader**.get(groupPosition)).get(  
 childPosition), Toast.***LENGTH\_SHORT***)  
 .show();  
 **return false**;  
 }  
 });  
 *//Список раскрывается:* **expListView**.setOnGroupExpandListener(**new** OnGroupExpandListener() {  
 @Override  
 **public void** onGroupExpand(**int** groupPosition) {  
 Toast.*makeText*(getApplicationContext(),  
 **listDataHeader**.get(groupPosition) + **" Раскрыт"**,  
 Toast.***LENGTH\_SHORT***).show();  
 }  
 });  
 *//Список сворачивается:* **expListView**.setOnGroupCollapseListener(**new** OnGroupCollapseListener() {  
 @Override  
 **public void** onGroupCollapse(**int** groupPosition) {  
 Toast.*makeText*(getApplicationContext(),  
 **listDataHeader**.get(groupPosition) + **" Свернут"**,  
 Toast.***LENGTH\_SHORT***).show();  
 }  
 });  
 }  
 */\*  
 \* Подготавливаем данные для списка:  
 \*/* **private void** prepareListData() {  
 **listDataHeader** = **new** ArrayList<String>();  
 **listDataChild** = **new** HashMap<String, List<String>>();  
  
 *//Добавляем данные о пунктах списка:* **listDataHeader**.add(**"Автовокзал – ОАО Райагропромтехника"**);  
 **listDataHeader**.add(**"Пл.Орджоникидзе (БИП) – ОАО Райагропромтехника"**);  
 **listDataHeader**.add(**"Завод синтетических пленок – ОАО Райагропромтехника"**);  
 **listDataHeader**.add(**"Автовокзал – Автовокзал"**);  
 **listDataHeader**.add(**"Могилевнефтепродукт – Автовокзал"**);  
 **listDataHeader**.add(**"Мог.рынок (к Пуш. пр) – Автовокзал"**);  
 **listDataHeader**.add(**"ул. Пионерская – Автовокзал"**);  
*//Выходные дни* **listDataHeader**.add(**"В.Д.Автовокзал – Могилевоблнефтепродукт"**);  
 **listDataHeader**.add(**"В.Д.Пл.Орджоникидзе (БИП) – Могилевоблнефтепродукт"**);  
 **listDataHeader**.add(**"В.Д.Завод синтетических пленок – Могилевоблнефтепродукт"**);  
 **listDataHeader**.add(**"В.Д.РУП «Могилевнефтепродукт» – Автовокзал"**);  
 **listDataHeader**.add(**"В.Д.Мог.рынок(к Пуш.пр) – Автовокзал"**);  
 **listDataHeader**.add(**"В.Д.Мог.рынок(к Пуш.пр) – Автовокзал"**);  
  
 List<String> top1 = **new** ArrayList<String>();  
 top1.add(**"05:00---11:00"**);  
 top1.add(**"05.35, 06.09, 06.42, 07.17, 07.53, 08.28, 09.04, 10.16"**);  
 top1.add(**"11:00---17:00"**);  
 top1.add(**"11.27, 12.38, 13.49, 15.00, 15.36, 16.14, 16.58"**);  
 top1.add(**"17:00---20:00"**);  
 top1.add(**"17.26, 18.09, 18.58, 20.03"**);  
  
 List<String> top2 = **new** ArrayList<String>();  
 top2.add(**"05:00---11:00"**);  
 top2.add(**"05.41, 06.15, 06.49, 07.24, 08.00, 08.35, 09.11, 10.23"**);  
 top2.add(**"11:00---18:00"**);  
 top2.add(**"11.34, 12.45, 13.56, 15.07, 15.43, 16.21, 17.05, 17.33"**);  
 top2.add(**"18:00---00:00"**);  
 top2.add(**"18.16, 19.05, 20.10, 21.15, 22.20, 23.27"**);  
  
 List<String> top3 = **new** ArrayList<String>();  
 top3.add(**"05:00---10:00"**);  
 top3.add(**"05.49, 06.23, 06.57, 07.32, 08.08, 08.43, 09.19"**);  
 top3.add(**"10:00---17:00"**);  
 top3.add(**"10.31, 11.42, 12.53, 14.04, 15.15, 15.51, 16.29"**);  
 top3.add(**"17:00---00:00"**);  
 top3.add(**"17.13, 17.41, 18.24, 19.13, 20.18, 21.23, 22.28, 23.35"**);  
  
 List<String> top4 = **new** ArrayList<String>();  
 top4.add(**"06:00---11:00"**);  
 top4.add(**"06.08, 06.41, 07.16, 07.52, 08.28, 09.03, 09.41, 10.51"**);  
 top4.add(**"12:00---19:00"**);  
 top4.add(**"12.02, 13.13, 14.24, 15.35, 16.11, 16.53, 17.33, 17.58, 18.42АП"**);  
  
 List<String> top5 = **new** ArrayList<String>();  
 top5.add(**"19:00---00:00"**);  
 top5.add(**"19.30, 20.33, 21.41, 22.45, 23.48АП"**);  
  
 List<String> top6 = **new** ArrayList<String>();  
 top6.add(**"00:00---01:00"**);  
 top6.add(**"00.08АП"**);  
  
 List<String> top7 = **new** ArrayList<String>();  
 top7.add(**"05:00---12:00"**);  
 top7.add(**"05.28, 06.02, 06.34, 07.08, 07.43, 08.19, 08.55, 09.30, 10.08, 11.18"**);  
 top7.add(**"12:00---20:00"**);  
 top7.add(**"12.29, 13.40, 14.51, 16.02, 16.38, 17.18, 18.00, 18.21, 19.09АП, 19.54"**);  
  
 List<String> vich1 = **new** ArrayList<String>();  
 vich1.add(**"06:00---12:00"**);  
 vich1.add(**"06.48, 07.58, 09.08, 10.41, 11.51"**);  
 vich1.add(**"13:00---21:00"**);  
 vich1.add(**"13.01, 14.11, 15.21, 16.31, 17.41, 18.49, 20.19"**);  
  
 List<String> vich2 = **new** ArrayList<String>();  
 vich2.add(**"06:00---15:00"**);  
 vich2.add(**"06.56, 08.06, 09.16, 10.49, 11.59, 13.09, 14.19"**);  
 vich2.add(**"15:00---00:00"**);  
 vich2.add(**"15.29, 16.39, 17.49, 18.57, 20.27, 21.33, 22.36, 23.39"**);  
  
 List<String> vich3 = **new** ArrayList<String>();  
 vich3.add(**"05:00---15:00"**);  
 vich3.add(**"05.59, 07.03, 08.13, 09.23, 10.56, 12.06, 13.16, 14.26"**);  
 vich3.add(**"15:00---00:00"**);  
 vich3.add(**"15.36, 16.46, 17.56, 19.04, 20.34, 21.40, 22.43, 23.46"**);  
  
 List<String> vich4 = **new** ArrayList<String>();  
 vich4.add(**"06:00---15:00"**);  
 vich4.add(**"06.14, 07.23, 08.33, 09.43, 11.16, 12.26, 13.36, 14.46"**);  
 vich4.add(**"15:00---00:10"**);  
 vich4.add(**"15.56, 17.06, 18.15, 19.23, 20.53, 21.56, 22.59, 00.01АП"**);  
  
 List<String> vich5 = **new** ArrayList<String>();  
 vich5.add(**"00:10---00:30"**);  
 vich5.add(**"00.21АП"**);  
  
 List<String> vich6 = **new** ArrayList<String>();  
 vich6.add(**"06:00---13:00"**);  
 vich6.add(**"06.39, 07.49, 08.59, 10.08, 11.42, 12.52"**);  
 vich6.add(**"14:00---20:00"**);  
 vich6.add(**"14.02, 15.12, 16.22, 17.32, 18.40, 19.46"**);  
  
 **listDataChild**.put(**listDataHeader**.get(0), top1);  
 **listDataChild**.put(**listDataHeader**.get(1), top2);  
 **listDataChild**.put(**listDataHeader**.get(2), top3);  
 **listDataChild**.put(**listDataHeader**.get(3), top4);  
 **listDataChild**.put(**listDataHeader**.get(4), top5);  
 **listDataChild**.put(**listDataHeader**.get(5), top6);  
 **listDataChild**.put(**listDataHeader**.get(6), top7);  
 **listDataChild**.put(**listDataHeader**.get(7), vich1);  
 **listDataChild**.put(**listDataHeader**.get(8), vich2);  
 **listDataChild**.put(**listDataHeader**.get(9), vich3);  
 **listDataChild**.put(**listDataHeader**.get(10), vich4);  
 **listDataChild**.put(**listDataHeader**.get(11), vich5);  
 **listDataChild**.put(**listDataHeader**.get(12), vich6);  
 }  
}

Заполнив Activity для отображения списка остановок, требуется создать соответствующий класс адаптера, в котором мы опишем каким образом будет работать список в приложении. Для этого создадим ещё один JAVA документ:

**package** com.example.mogtransport;  
  
**import** java.util.ArrayList;  
**import** java.util.HashMap;  
**import** java.util.List;  
**import** android.content.Context;  
**import** android.graphics.Typeface;  
**import** android.view.LayoutInflater;  
**import** android.view.View;  
**import** android.view.ViewGroup;  
**import** android.widget.BaseExpandableListAdapter;  
**import** android.widget.Button;  
**import** android.widget.TextView;  
**import** android.widget.Toast;  
  
**public class** ExpandableListAdapter **extends** BaseExpandableListAdapter {  
 **private** Context **\_context**;  
 *//Названия заголовков* **private** List<String> **\_listDataHeader**;  
 *//Данные для элементов подпунктов:* **private** HashMap<String, List<String>> **\_listDataChild**;  
  
 **public** ExpandableListAdapter(Context context, List<String> listDataHeader,  
 HashMap<String, List<String>> listChildData) {  
 **this**.**\_context** = context;  
 **this**.**\_listDataHeader** = listDataHeader;  
 **this**.**\_listDataChild** = listChildData;  
 }  
  
 **public** ExpandableListAdapter(Context applicationContext, ArrayList<ArrayList<String>> groups) {  
  
 }  
 @Override  
 **public** Object getChild(**int** groupPosition, **int** childPosititon) {  
 **return this**.**\_listDataChild**.get(**this**.**\_listDataHeader**.get(groupPosition))  
 .get(childPosititon);  
 }  
 @Override  
 **public long** getChildId(**int** groupPosition, **int** childPosition) {  
 **return** childPosition;  
 }  
 @Override  
 **public** View getChildView(**int** groupPosition, **final int** childPosition,  
 **boolean** isLastChild, View convertView, ViewGroup parent) {  
 **final** String childText = (String) getChild(groupPosition, childPosition);  
 **if** (convertView == **null**) {  
 LayoutInflater infalInflater = (LayoutInflater) **this**.**\_context** .getSystemService(Context.***LAYOUT\_INFLATER\_SERVICE***);  
 convertView = infalInflater.inflate(R.layout.***list\_item***, **null**);  
 }  
  
 TextView txtListChild = (TextView) convertView  
 .findViewById(R.id.***lblListItem***);  
 txtListChild.setText(childText);  
 **return** convertView;  
 }  
 @Override  
 **public int** getChildrenCount(**int** groupPosition) {  
 **return this**.**\_listDataChild**.get(**this**.**\_listDataHeader**.get(groupPosition))  
 .size();  
 }  
 @Override  
 **public** Object getGroup(**int** groupPosition) {  
 **return this**.**\_listDataHeader**.get(groupPosition);  
 }  
 @Override  
 **public int** getGroupCount() {  
 **return this**.**\_listDataHeader**.size();  
 }  
 @Override  
 **public long** getGroupId(**int** groupPosition) {  
 **return** groupPosition;  
 }  
  
 @Override  
 **public** View getGroupView(**int** groupPosition, **boolean** isExpanded,  
 View convertView, ViewGroup parent) {  
 String headerTitle = (String) getGroup(groupPosition);  
 **if** (convertView == **null**) {  
 LayoutInflater infalInflater = (LayoutInflater) **this**.**\_context** .getSystemService(Context.***LAYOUT\_INFLATER\_SERVICE***);  
 convertView = infalInflater.inflate(R.layout.***list\_group***, **null**);  
 }  
 TextView lblListHeader = (TextView) convertView  
 .findViewById(R.id.***lblListHeader***);  
 lblListHeader.setTypeface(**null**, Typeface.***BOLD***);  
 lblListHeader.setText(headerTitle);  
  
 **return** convertView;  
 }  
  
 @Override  
 **public boolean** hasStableIds() {  
 **return false**;  
 }  
  
 @Override  
 **public boolean** isChildSelectable(**int** groupPosition, **int** childPosition) {  
 **return true**;  
 }  
}

После создания класса адаптера в документе, получается вид многоуровневого списка, представленный на рисунке 5:



Пример многоуровневого списка для маршрута 3



Для удобства получения информации список состоит из отображения определенного промежутка времени, в котором отображаются примерные временные остановки общественного транспорта на данной маршруте. Подобным образом реализованы и остальные маршруты общественного транспорта. Учитывая то, что приложение работает без подключения интернет соединения, все изменения в расписание общественного транспорта будут производиться в ручном режиме.

## Архитектура мобильного приложения

Приложения для операционной системы Android разрабатывается на языке JAVA. Инструменты Android SDK компилируют написанный код в файл APK. Результативный APK-файл можно установить на любом устройстве с операционной системой Android. Каждое приложение под данной операционной системой, работает в собственной «песочнице»(изолированной программной среде).

Для запуска компонента приложения системе Android необходимо знать, что данный компонент вообще существует. Для этого она использует файл AndroidManifest.XML, в котором отображаются все компоненты, задействованные в данном приложении.

*<?***xml version="1.0" encoding="utf-8"***?>*<**manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 package="com.example.mogtransport"**>  
/>  
<**uses-permission android:name="android.permission.INTERNET"** />  
 <**uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE"** />  
 <**application  
 android:allowBackup="true"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:supportsRtl="true"  
 android:theme="@style/AppTheme"**>  
 <**activity  
 android:screenOrientation="portrait"  
 android:name=".MainActivity"  
 android:label="@string/app\_name"  
 android:theme="@style/AppTheme.NoActionBar"** />  
 <**activity  
 android:name=".Bus4"  
 android:label="@string/title\_activity\_bus4"  
 android:parentActivityName=".MainActivity"  
 android:theme="@style/AppTheme.NoActionBar"**>  
 <**meta-data  
 android:name="android.support.PARENT\_ACTIVITY"  
 android:value="com.example.mogtransport.MainActivity"** />  
 </**activity**>  
 <**activity  
 android:screenOrientation="portrait"  
 android:name=".about"  
 android:label="@string/title\_activity\_about"  
 android:parentActivityName=".MainActivity"  
 android:theme="@style/AppTheme.NoActionBar"**>  
 <**meta-data  
 android:name="android.support.PARENT\_ACTIVITY"  
 android:value="com.example.mogtransport.MainActivity"** />  
 </**activity**>  
 </**application**>  
</**manifest**>

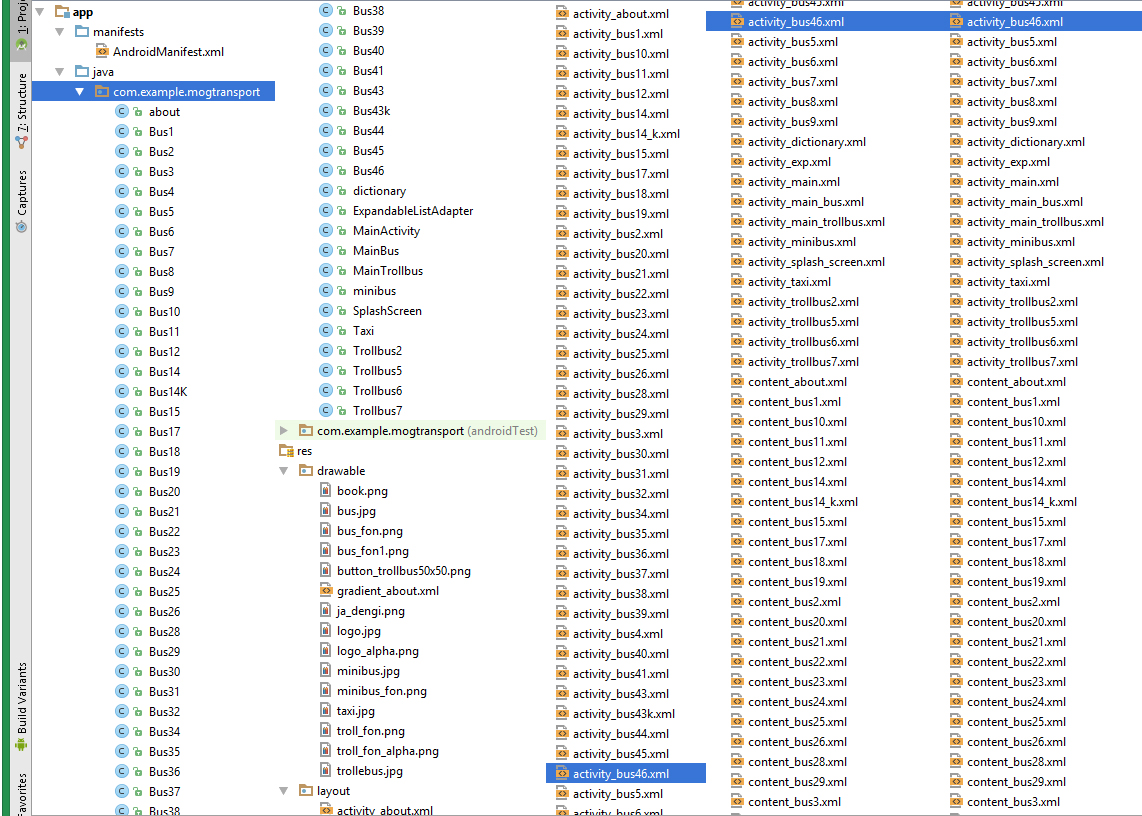
Написание качественного программного обеспечения ­ сложное и многогранное занятие, программа должна не только соответствовать установленным требованиям, но и быть надежной, удобной, гибкой для добавления или изменения функций.

### Программная часть архитектуры мобильного приложения

Структура приложения представлена рисунком 6, поскольку проект в Android Studio занимает большой объем из-за количества Activity, которые потребовались для описания всех существующих маршрутов, то с помощью дополнительных систем работы с графическими файлами, вся структура была собрана в один рисунок 6:



Структура приложения



### Пользовательская часть мобильного приложения

Разрабатывая приложение, в первую очередь стоит думать о том, кто будет использовать приложение. В корпорации Google, инженеры отметили несколько основных принципов при создании графической части приложения:

* **Simple vs Clear**: интерфейс должен быть простым(не нагруженным) и понятным для использования
* **Content vs Chrome**: необходимо использовать максимум экрана, при этом уменьшать его визуальную сложность (использовать ограниченное число кнопок/иконок)
* **Consistent yet engaging**: консистентность реакции пользователя – пользователь должен понимать что он делает/как сделать то, что ему необходимо

При разработке данного приложения учитывались вышеописанные принципы для дальнейшего удобства конечного ользователя. В частности, в главном меню приложения, пользователь видит только информацию, с выбором общественного транспорта. После выбора общественного транспорта, пользователь выбирает нужный ему маршрут и далее, в зависимости от нужд пользователя находит более удобный маршрут.

**Вывод:** При разработке приложения, строго соблюдались все этапы разработки мобильных приложений, было разработано мобильное приложение, которое полностью соответствует техническому заданию и цели данной работы.

# Заключение

В данной дипломной работе было разработано и реализовано приложение для операционной мобильной системы Android. Цели и задачи, которые были поставлены в данной работе, были выполнены – мобильное приложение создано. В процессе разработки был проведен анализ современных мобильных операционных систем. Разработанное приложение обладает всем требуемым функционалом, который полностью соответствует требуемым задачам. Интерфейс разработан основываясь на концепции отмеченной инженерами корпорации Google, он позволяет быстро ориентироваться в приложениях, не требует от пользователя специальных навыков по работе с ним, а также позволяет выводить оперативную информацию без доступа к сети интернет.

Так как данное приложение не является коммерческим проектом, то оно не может охватить все тонкости, но оно может быть усовершенствовано и расширено в рамках, не рассмотренных в данной дипломной работе.

Назначение приложения MogTransport состоит в том, что оно помогает планировать свое передвижение по города более детально, оценивая время определенных маршрутов на той или иной остановке в городе Могилев. После выбора пользователем требуемой группы общественного транспорта, отображается перечень остановок, входящих в данную группу, и после выбора конкретной остановки разворачивается список с временными интервалами, где указано время для данного маршрута.

Таким образом, при выполнении данной дипломной работы были получены навыки по созданию мобильных приложений, выбору операционной системы и выбору SDK для программирования. Также получен навык по анализу и выбору мобильных операционных систем изучены предлагаемые концепции в проектировании внешнего вида приложения под определённую операционную систему.

# Список использованной литературы

# Приложение А