**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**МЕХАНИКО-МАТЕМАТИЧЕСКИЙ ФАКУЛЬТЕТ**

**Кафедра веб-технологий и компьютерного моделирования**

**Мобильное приложение по отслеживанию сроков годности продуктов**Курсовая работа

Белявская Елизавета Андреевна

Студентка 2 курса, Веб-программирование и Интернет-технологии

Научный руководитель:  
Старший преподаватель кафедры веб-технологий Вельченко Сергей Александрович

Минск, 2020

# Оглавление

Оглавление 2

Введение 3

Глава 1. Теоретические сведения 4

1.1 Обзор мобильной ОС Android 4

1.2 Мэшапы 5

1.3 Основы программирования на Android 5

1.3.1 Деятельности 6

1.3.2 Намерения 6

1.3.3 Виджеты и видовые окна 8

1.3.4 Асинхронные вызовы 8

1.3.5 Фоновые службы 9

1.4 Архитектура Android приложения 10

1.6 Архитектура операционной системы Android 14

Глава 2. Реализация мобильного приложения 20

2.1. Концепции и использование Ionic Framework 20

2.2. Разработка мобильного приложения на Ionic Framework 21

Заключение 23

Список использованных источников 24

# Введение

Мобильные телефоны давно перестали быть чем-то необычным и великолепно справляются со своей функцией – являются средством коммуникации между людьми. При этом, недавно появившиеся, но уже прочно вошедшие в нашу жизнь смартфоны настолько функциональны, что трудно сказать, чего они не умеют: это и плеер, и фотоаппарат, и возможность использования Интернет-ресурсов, и прочее. По сути, все смартфоны стали небольшой копией компьютера, который постоянно можно иметь при себе. В наше время все больше и больше смартфонов, коммуникаторов, планшетных ПК и других видов устройств, удобных для использования как в повседневной жизни, так и в заграничных поездках в частности, выпускаются на базе ОС Android. Каковы же причины распространения данной операционной системы? Во-первых, Android поддерживает большое количество устройств разных производителей. Во-вторых, Android характеризуется высокой доступностью средств разработки. Средства разработки для платформы Android бесплатны, в то время как разработка, к примеру, под iPhone (от компании Apple) требует немалых начальных финансовых вложений. Кроме всего вышеперечисленного, преимуществом ОС Android является наличие бесплатных библиотек для работы со сторонними ресурсами (Yandex MapKit, Google Map API, др.), в то время как для Windows Phone Mobile такие библиотеки не распространены. В рамках данной курсовой работы будет описано создание приложения для отслеживания сроков годности продуктов на базе ОС Android, ориентированной, в частности, на домашнее использование. Указанные выше преимущества обуславливают массовость и широкое распространение современных устройств на базе Android, оснащенных различными функциями и приложениями, делающими жизнь человека максимально удобной.

# Глава 1. Теоретические сведения

## 1.1 Обзор мобильной ОС Android

Операционная система Android - это открытая платформа, а значит, что она не привязана к одному провайдеру. Это помогает Android завоевать рынок, так как любой производитель и провайдер может создавать и продавать устройства, поддерживающие эту операционную систему. Исходный код Android доступен каждому для изучения или модифицирования. Это позволяет дать широкий простор для создания удобных интерфейсов и приложений для Android-устройств.

ОС Android может использоваться на устройствах с разными размерами экрана и другими техническими характеристиками, поэтому она поставляется с набором инструментов, которые помогают приложению адаптироваться под конкретное устройство. Политика Google более жесткая. Например, если приложение требует наличие фронтальной камеры, то это приложение в Android Market увидят телефоны только с фронтальной камерой.

## 1.2 Мэшапы

Мэшап соединяет две и более службы в одном приложении. Например, можно создать мэшап, использующий камеру и геолокацию для размещения изображений на карте. Такой прием часто можно встретить в социальных сетях.

С помощью встроенных в Android библиотек API, можно легко использовать предоставляемые возможности для создания собственного приложения с необходимыми функциями.

## 1.3 Основы программирования на Android

Android приложение пишется на языке Java, однако при разработке необходимы также документы XML. Язык Java используется здесь не в полнофункциональном варианте, а только в небольшом подмножестве, которое иногда называют виртуальной машиной Davlik. В этом подмножестве не используются те классы Java, которые не могут быть применены или не имеют смысла при разработке приложений на мобильные устройства.

## 1.3.1 Деятельности

Android приложение включает в себя одну или несколько деятельностей (activities). Деятельность можно представить в виде контейнера, содержащего пользовательский интерфейс и код, который его запускает.

## 1.3.2 Намерения

Намерения (intents) составляют систему сообщений на Android. Намерение состоит из действия (action), которое необходимо выполнить (посмотреть, редактировать и др.) и данных. Действие - это то, что должно быть совершено при получении намерения и данных, с которыми необходимо оперировать.

Намерения используются при запуске деятельности и при коммуникации между различными частями Android системы. Приложение может получать или отправлять намерения. При передаче намерения фактически отправляется сообщение системе сделать что-либо, например, запустить новую деятельность в текущем приложении или открыть другое приложение. Если просто отправить намерение, это не означает, что что-то произойдет автоматически. Для этого необходимо зарегистрировать приемник намерений (intent receiver), который получает намерение и указывает Android системе, что нужно сделать: выполнить задачу в новой деятельности или запустить другое приложение. Если же имеется несколько приемников для получения intent, можно создать инструмент, позволяющий пользователю самому выбрать необходимое действие. Одним намерением могут быть найдены несколько приемников, поэтому пользователь лично определяет action, которое нужно выполнить. Например, при долгом нажатии на изображение в галерее, появляется инструмент выбора, который предлагает отправить картинку через e-mail или социальные сети, редактировать или удалить и др.

Если система не может найти подходящее намерение, а инструмент выбора не был создан разработчиком, то приложение потерпит крах и выдаст ошибку исполнения. Поэтому важно следить за тем, чтобы выборы для намерений, не нацеленных на другие деятельности в данном приложении, были созданы.

## 1.3.3 Виджеты и видовые окна

Видовое окно (view) представляет собой базовый элемент управления интерфейса в виде прямоугольной области, где можно рисовать и обрабатывать события. Примерами видовых окон являются: контекстное меню (ContextMenu), меню (Menu), вид (View), поверхность рисования (SurfaceView).

Виджеты (widgets) - это более продвинутые элементы пользовательского интерфейса, например, флажки с переключателями, где можно выбрать одно из нескольких возможных состояний. Виджеты являются теми элементами управления, с которыми взаимодействует пользователь. Виджетами являются: кнопка (Button), выбор даты (DatePicker), галерея (Gallery), флажок (CheckBox) и др.

## 1.3.4 Асинхронные вызовы

Прежде чем перейти к понятию асинхронных вызовов, нужно ясно понимать, что такое поток.

Потоком называется выполнение в одном приложении нескольких операций одновременно. Асинхронным поток называется поток, который выполняется независимо от других потоков, в фоновом режиме.

Асинхронные потоки применяются при выполнении задач, которые занимают много времени, например, загрузка файла из Интернета, воспроизведение аудиозаписи или просмотр видеоклипа, т.е. то, что заставляет пользователя ждать. Если пользователю необходимо подождать решения какой-либо задачи, то лучший выход из ситуации - создать асинхронный поток, чтобы дать возможность пользователю заниматься в это время чем-нибудь другим.

В большинстве случаев специалисты рекомендуют выносить в фоновый поток выполнение тех задач, которые сильно нагружают процессор, или выполняются более 5-6 секунд.

## 1.3.5 Фоновые службы

Служба - это такое приложение, которое выполняется в фоновом режиме и которому не требуется наличие пользовательского интерфейса. Примером службы на компьютере является антивирусная программа, а в ОС Android - аудиоплееры, которые воспроизводят аудиозаписи в фоновом режиме, позволяя пользователю пользоваться с то же время другими приложениями.

## 1.4 Архитектура Android приложения

Android приложение состоит из нескольких деятельностей (Activities), которые занимают весь экран. Приложение же переключается между ними в зависимости от выбранной пользователем задачи.

Деятельности состоят из видовых окон (View). Общую структуру стандартного Android приложения можно представить в виде схемы  
  
 **1.5 Компоненты приложений на Android**

Интересной чертой операционной системы Android является тот факт, что приложение в ней может использовать для своего функционала элементы других приложений, если, конечно, они предоставляют такой доступ. Например, если нашему приложению требуется отобразить прокручиваемый список изображений, а другое приложение уже имеет реализованный подходящий скроллер, открытый для других приложений, то можно просто вызвать его для осуществления этого действия и не разрабатывать свой собственный. При этом код нашего приложения не смешивается с кодом другого и не компонуется с ним. Скорее, он просто запускает некоторый кусок другого файла, когда возникает такая необходимость. Для обеспечения такого принципа работы, система должна быть способна запускать процесс приложения тогда, когда потребуется любая его часть и создавать экземпляры java-объектов именно для этой части. Поэтому приложения Android не имеют единой точки входа, как это принято в большинстве систем. Вместо этого их код представляет собой набор некоторых отдельных целостных сущностей, компонент, из которых система по мере необходимости может создавать экземпляры и использовать их.

Можно выделить четыре типа таких сущностей-компонент. Поговорим о каждой в отдельности.

Activities

Activity представляет собой внешний пользовательский интерфейс для одной операции, которую может совершить пользователь. Если упростить, то это просто один текущий экран как некоторая единица активности, своего рода кадр с одним пользовательским действием. Здесь и далее будет использоваться термин activity без перевода, как некоторое имя собственное. Хотя дословный перевод, «активность» или «действие», хорошо передает общий смысл компонента.

Например, activity может предоставить список пунктов меню, которые может выбрать пользователь или отобразить фотографии с их подписями. Или другой пример - приложение для мгновенного обмена сообщениями может использовать одно activity для того чтобы отобразить лист контактов, другое - чтобы создать сообщение для выбранного контакта, третье - чтобы посмотреть историю сообщений или выполнить настройки и так далее.

Все activity текущего приложения работают вместе и формируют единый пользовательский интерфейс, однако при этом они независимы между собой. Каждое из них реализовано как подкласс базового класса Activity, обеспечивающего создание окна, в котором программист может поместить визуальный интерфейс.

Приложение может состоять из всего одного activity или сразу из нескольких, как упомянутый ранее в качестве примера мессенджер. Какими именно будут activity и сколько их будет, зависит от конкретного приложения и его дизайна. Как правило, одно из activity помечается как первое, это означает, что оно будет предоставлено пользователю при запуске приложения. Одно activity может запускать другое. Таким образом, переход от одного activity к другому осуществляется тогда, когда текущее activity вызывает следующее.

Каждое activity предоставляет окно по умолчанию. Обычно окно создается в полноэкранном виде, но оно также может и не занимать весь экран и находиться поверх других окон. Activity также может задействовать дополнительные окна - например, всплывающее диалоговое для взаимодействия с пользователем в процессе работы activity, или окно для предоставления текущей информации при выборе какой-нибудь важной опции.

Визуальное содержание окна строится с помощью иерархии визуальных компонентов (или представлений) - объектов, производных от базового класса View. Каждый компонент представляет собой просто прямоугольное пространство внутри окна. Родительские компоненты содержат дочерние и организуют их расположение. Иерархию компонент можно представить в виде дерева, а те элементы, которые находятся в самой нижней его части (“листья”) и не имеют дочерних компонент, отрисовывают прямоугольные области и ожидают действий пользователя на этом участке. Таким образом осуществляется интерактивное взаимодействие с пользователем. К примеру, такое представление может отображать на экране маленькую иконку и инициировать какое-нибудь действие, когда пользователь на неё нажмет. В операционной системе Android уже есть набор готовых визуальных компонент, которые доступны для использования разработчиками. Набор включает в себя кнопки, текстовые поля, полосы прокрутки, меню, флажки-переключатели и многое другое.

Для того чтобы поместить в окно такую иерархию, нужно вызвать метод Activity.setContentView(). Параметром метода является экземпляр класса View, лежащий в корне иерархии.

Services(сервисы) представляют из себя компоненты, которые работают в фоновом режиме. Он, как правило, требуется для длительных операций или для обеспечения работы удаленных процессов, но в общем случае это просто режим, который функционирует, когда приложение не в фокусе. Примером такого процесса может стать прослушивание музыки в то время, когда пользователь делает что-то другое или получение данных по сети без блокирования текущей активности. Сервис сам по себе не предоставляет пользовательского интерфейса, то есть с пользователем не взаимодействует, а запускается, управляется и связан с другими компонентами, например, activity. Также может запускаться вместе с системой.

Content providers Данный компонент управляет наборами данных, которые приложения предоставляют другим. Эти данные могут храниться в файловой системе, базах данных SQLite, в сети, или в любом другом постоянном месте, к которому приложение может иметь доступ. Посредством content provider другое приложение может запрашивать данные и, если выставлены соответствующие разрешения, изменять их. Например, система Android содержит content provider, который управляет пользовательской информацией о контактах. Он позволяет любому приложению, обладающему соответствующими правами вызывать составляющие этого компонента для того, чтобы считывать, записывать или изменять информацию о конкретном человеке.

В более общем случае, content provider можно использовать для чтения и записи данных, которые используются приложением и не являются открытыми для других. Например, приложение Note Pad использует такой компонент для сохранения сделанных записей.

Broadcast receivers. Этот компонент отвечает за распространение общесистемных сообщений, отслеживание и реагирование на действия. Многие оповещения идут от системы, например, сообщения о том что заряд батареи мал или экран выключен. Как и сервисы, broadcast receiver не предоставляет пользовательского интерфейса, однако, он способен создавать уведомления в строке состояния, чтобы предупреждать пользователя о том, что произошло какое-то событие. Однако чаще broadcast receiver взаимодействует с другими компонентами для того, чтобы самому выполнять минимальный объем работы. Так, он может инициировать сервисы для выполнения действий, привязанных к какому-то событию.

## 1.6 Архитектура операционной системы Android

Если представить компонентную модель Android в виде некоторой иерархии, то в самом низу, как самая фундаментальная и базовая составляющая, будет располагаться ядро операционной системы.

Часто компонентную модель ещё называют программным стеком. Действительно, это определение тут уместно, потому что речь идет о наборе программных продуктов, которые работают вместе для получения итогового результата. Действия в этой модели выполняются последовательно, и уровни иерархии также последовательно взаимодействуют между собой.

Как известно, Андроид основан на несколько урезанном ядре ОС Linux и поэтому на этом уровне мы можем видеть именно его (версии 2.6.x). Оно обеспечивает функционирование системы и отвечает за безопасность, управление памятью, энергосистемой и процессами, а также предоставляет сетевой стек и модель драйверов. Ядро также действует как уровень абстракции между аппаратным обеспечением и программным стеком.

«Выше» ядра, как программное обеспечение промежуточного слоя, лежит набор библиотек (Libraries), предназначенный для обеспечения важнейшего базового функционала для приложений. То есть именно этот уровень отвечает за предоставление реализованных алгоритмов для вышележащих уровней, поддержку файловых форматов, осуществление кодирования и декодирования информации (в пример можно привести мультимедийные кодеки), отрисовку графики и многое другое. Библиотеки реализованы на C/C++ и скомпилированы под конкретное аппаратное обеспечение устройства, вместе с которым они и поставляются производителем в предустановленном виде.

Перечислим некоторые из них:

Surface Manager - в ОС Android используется композитный менеджер окон, наподобие Compiz (Linux), но более упрощенный. Вместо того чтобы производить отрисовку графики напрямую в буфер дисплея, система посылает поступающие команды отрисовки в закадровый буфер, где они накапливаются вместе с другими, составляя некую композицию, а потом выводятся пользователю на экран. Это позволяет системе создавать интересные бесшовные эффекты, прозрачность окон и плавные переходы.

Media Framework - библиотеки, реализованные на базе PacketVideo OpenCORE. С их помощью система может осуществлять запись и воспроизведение аудио и видео контента, а также вывод статических изображений. Поддерживаются многие популярные форматы, включая MPEG4, H.264, MP3, AAC, AMR, JPG и PNG.

SQLite - легковесная и производительная реляционная СУБД, используемая в Android в качестве основного движка для работы с базами данных, используемыми приложениями для хранения информации.

3D библиотеки -- используются для высокооптимизированной отрисовки 3D-графики, при возможности используют аппаратное ускорение. Их реализации строятся на основе API OpenGL ES 1.0.

o OpenGL ES (OpenGL for Embedded Systems) - подмножество графического программного интерфейса OpenGL, адаптированное для работы на встраиваемых системах.

FreeType - библиотека для работы с битовыми картами, а также для растеризации шрифтов и осуществления операций над ними. Это высококачественный движок для шрифтов и отображения текста.

LibWebCore - библиотеки известного шустрого браузерного движка WebKit, используемого также в десктопных браузерах Google Chrome и Apple Safari.

SGL (Skia Graphics Engine) - открытый движок для работы с 2D-графикой. Графическая библиотека является продуктом Google и часто используется в других их программах. SSL - библиотеки для поддержки одноименного криптографического протокола.

Libc - стандартная библиотека языка C, а именно её BSD реализация, настроенная для работы на устройствах на базе Linux. Носит название Bionic.

На этом же уровне располагается Android Runtime - среда выполнения. Ключевыми её составляющими являются набор библиотек ядра и виртуальная машина Dalvik. Библиотеки обеспечивают большую часть низкоуровневой функциональности, доступной библиотекам ядра языка Java.

Каждое приложение в ОС Android запускается в собственном экземпляре виртуальной машины Dalvik. Таким образом, все работающие процессы изолированы от операционной системы и друг от друга. И вообще, архитектура Android Runtime такова, что работа программ осуществляется строго в рамках окружения виртуальной машины. Благодаря этому осуществляется защита ядра операционной системы от возможного вреда со стороны других её составляющих. Поэтому код с ошибками или вредоносное ПО не смогут испортить Android и устройство на его базе, когда сработают. Такая защитная функция, наряду с выполнением программного кода, является одной из ключевых для надстройки Android Runtime. Уровнем выше располагается Application Framework, иногда называемый уровнем каркаса приложений. Именно через каркасы приложений разработчики получают доступ к API, предоставляемым компонентами системы, лежащими ниже уровнем. Кроме того, благодаря архитектуре фреймворка, любому приложению предоставляются уже реализованные возможности других приложений, к которым разрешено получать доступ.

В базовый набор сервисов и систем, лежащих в основе каждого приложения и являющихся частями фреймворка, входят:

Богатый и расширяемый набор представлений (Views), который может быть использован для создания визуальных компонентов приложений, например, списков, текстовых полей, таблиц, кнопок или даже встроенного web-браузера.

Контент-провайдеры (Content Providers), управляющие данными, которые одни приложения открывают для других, чтобы те могли их использовать для своей работы.

Менеджер ресурсов (Resource Manager), обеспечивающий доступ к ресурсам без функциональности (не несущими кода), например, к строковым данным, графике, файлам и другим.

Менеджер оповещений (Notification Manager), благодаря которому все приложения могут отображать собственные уведомления для пользователя в строке состояния.

Менеджер действий (Activity Manager), который управляет жизненными циклами приложений, сохраняет данные об истории работы с действиями, а также предоставляет систему навигации по ним.

Менеджер местоположения (Location Manager), позволяющие приложениям периодически получать обновленные данные о текущем географическом положении устройства.

Таким образом, благодаря Application Framework, приложения в ОС Android могут получать в своё распоряжение вспомогательный функционал, благодаря чему реализуется принцип многократного использования компонентов приложений и операционной системы. Естественно, в рамках политики безопасности.

o Стоит отметить, просто на понятийном уровне, что фреймворк лишь выполняет код, написанный для него, в отличие от библиотек, которые исполняются сами. Ещё одно отличие заключается в том, что фреймворк содержит в себе большое количество библиотек с разной функциональностью и назначением, в то время как библиотеки объединяют в себе наборы функций, близких по логике.

На вершине программного стека Android лежит уровень приложений (Applications). Сюда относится набор базовых приложений, который предустановлен на ОС Android. Например, в него входят браузер, почтовый клиент, программа для отправки SMS, карты, календарь, менеджер контактов и многие другие. Список интегрированных приложений может меняться в зависимости от модели устройства и версии Android. И помимо этого базового набора к уровню приложений относятся в принципе все приложения под платформу Android, в том числе и установленные пользователем. Считается, что приложения под Android пишутся на языке Java, но нужно отметить, что существует возможность разрабатывать программы и на C/C++ (с помощью Native Development Kit), и на Basic (с помощью Simple) и с использованием других языков. Также можно создавать собственные программы с помощью конструкторов приложений, таких как App Inventor. Словом, возможностей тут много. Ну и ещё раз про архитектуру ОС Android можно узнать из первых рук, посмотрев это видео, в котором разработчики представляют презентацию, посвященную разбору системы.

# Глава 2. Реализация мобильного приложения

## 2.1. Концепции и использование Ionic Framework

Ionic Framework — это фреймворк для создания гибридных мобильных приложений. В его состав входит набор JavaScript и CSS компонент, созданных на основе Angular 2, SASS и Apache Cordova. История этого SDK начинается в 2013 году, когда компания Drifty Co. решила создать собственную инфраструктуру для написания гибридных приложений, которая будет ориентирована на производительность и будет построена с использованием современных веб-стандартов. Релиз Ionic 1 – состоялся в мае 2015 года. В 2016 году была выпущена версия 2. Основное отличия второго релиза от первого – это переход с Angular 1.x на Angular 2.х.  
  
По своей сути, Ionic Framework — это дополнение над очень популярным фреймворком Apache Cordova, но со своим мощным CLI (Command Line Interface) и объемной документацией. Следуя принципам Apache Cordova, приложения на Ionic Framework — это гибридные HTML приложения. Такие приложения на телефоне выполняются в специальной оболочке (UIWebView для iOS и WebView для Android), которая позволяет показывать HTML и выполнять JavaScript. Соответственно при работе в приложении пользователь работает как бы в веб-браузере.

Apache Cordova является своеобразной прослойкой между пользовательским интерфейсом и ресурсами устройства. Какие-либо виджеты пользовательского интерфейса или MV\* (Model-View) фреймворков не входят в его состав. Если необходимо использовать UI-виджеты и/или MV\* фреймворк, то нужно выбрать и включить их в приложение самостоятельно, как ресурсы третьей стороны. Ionic 2 — это один из фреймворков, который предоставляет UI виджеты. MVС модель для него предоставляет Angular 2.  
  
Нативные функции телефона (например, камера, хранилище ключей, GPS координаты) недоступны из веб-браузера. Поэтому для работы с ними используются плагины Apache Cordova. Кроме официальных плагинов есть ряд сторонних открытых плагинов.

## 2.2. Разработка мобильного приложения на Ionic Framework

Для создания приложения на Ionic 2 используются фреймворки Angular 2 и Apache Cordova. Приложение написано на языке TypeScript, который трансформируется в Javascript. Он позиционируется как средство разработки веб-приложений, расширяющее возможности JavaScript, на нем же и реализовано большинство примеров в документации Angular 2. В Angular 2 реализована шаблон проектирования внедрения зависимостей (software design pattern Dependency Injection), который состоит из 3х элементов:

* Injector — предоставляет доступ к API, для внедрения зависимостей
* Provider — описывает как создавать экземпляр зависимости; имеет маркер, указывающий на фабричную функцию, создающую объект
* Dependency — тип, к которому относиться созданный объект

Кроме того, в Angular 2 используется компонентный подход, который позволяет разделить бизнес-логику, работу с фронтом. Кроме того, разделение также используется на уровне элементов проекта: HTML, JavaScript и CSS.  
Отладка приложений в Ionic 2 стандартна и быстра. После запуска приложения на эмуляторе/девайсе возможно отлаживать приложения с использованием Safari веб-инспектор (только для iOS) или Chrome Developer tools. Эти инструменты предоставляют доступ к HTML, CSS и JavaScript. Во время отладки есть возможность быстро редактировать стили, искать узлы в html, выполнять и смотреть результат JavaScript функций, а также сетевых запросов. Кроме запуска на устройстве или эмуляторе, в случае Ionic 2 также есть возможность запустить приложение непосредственно в браузере (ionic serve). А с помощью команды ionic serve --lab упрощается тестирование приложений на разных диагоналях экранов и платформах.

В Ionic 2 пишется HTML5 код и используется SASS. Тут нет ограничений на размерные единицы, используется препроцессор SASS, также можно применить flex разметку.

Дизайн нашего приложения получился достаточно простым и минималистичным.

# Заключение

Ionic Framework – это очень интересный инструмент для гибридных приложений. В комплекте с AngularJS он предоставил нам широкий спектр возможностей для разработки. Скорость работы нашего приложения получилась достаточно высокой, если сравнивать с “родными” языками.

# Список использованных источников

1. Felker, Donn, and Joshua Dobbs. Android App Development for Dummies. 3rd ed. Vol. 357, Indianapolis, Indiana, Wiley Publishing, Inc., 2011.
2. https://www.iphones.ru/iNotes/538609
3. http://www.topobzor.com/obzor-10-oblachnyx-xranilishh
4. AWS Storage Services Overview. A Look at Storage Services Offered by AWS. Vol. 48, Amazon Web Services, Inc., December 2016.
5. http://docs.aws.amazon.com/mobile/sdkforandroid/
6. http://developer.android.com
7. https://habr.com/company/simpleweek/blog/254681/
8. https://habr.com/post/328960/

**Приложение А. Код программы**

**home.html**

<ion-header>

<ion-navbar color="secondary">

<ion-title>

Todo Roles

</ion-title>

<ion-buttons end>

<button ion-button icon-only (click)="addTodo()"><ion-icon name="add"></ion-icon></button>

</ion-buttons>

<ion-buttons start>

<button ion-button icon-only (click)="logout()"><ion-icon name="power"></ion-icon></button>

</ion-buttons>

</ion-navbar>

</ion-header>

<ion-content>

<ion-list no-lines>

<ion-item-sliding \*ngFor="let todo of todos">

<ion-item>

{{todo.title}}

</ion-item>

<ion-item-options>

<button ion-button color="danger" (click)="deleteTodo(todo)">

<ion-icon name="trash"></ion-icon>

Delete

</button>

</ion-item-options>

</ion-item-sliding>

</ion-list>

</ion-content>

**Приложение Б. Код программы**

**login.html**

<ion-header>

<ion-navbar color="secondary">

<ion-title>login</ion-title>

</ion-navbar>

</ion-header>

<ion-content padding>

<ion-row class="login-form">

<ion-col>

<ion-list inset>

<ion-item>

<ion-label><ion-icon name="person"></ion-icon></ion-label>

<ion-input [(ngModel)]="username" placeholder="email" type="text"></ion-input>

</ion-item>

<ion-item>

<ion-label><ion-icon name="lock"></ion-icon></ion-label>

<ion-input [(ngModel)]="password" placeholder="password" type="password"></ion-input>

</ion-item>

</ion-list>

<button ion-button block outline (click)="login()" color="secondary" class="login-button">Login</button>

</ion-col>

</ion-row>

<ion-row>

<ion-col>

<button ion-button block outline color="secondary" (click)="launchSignup()" class="create-account">Create an Account</button>

</ion-col>

</ion-row>

</ion-content>

**Приложение В. Код программы**

**modal-add-post.html**

<ion-header>

<ion-navbar>

<ion-title>Добавить</ion-title>

<ion-buttons end>

<button ion-button (click)="closeModal()">

Закрыть

</button>

</ion-buttons>

</ion-navbar>

</ion-header>

<ion-content>

<ion-list>

<ion-item>

<ion-label>Продукт</ion-label>

<ion-input [(ngModel)]="name" type="text"></ion-input>

</ion-item>

<ion-item>

<ion-label>Дата годности</ion-label>

<ion-datetime [(ngModel)]="date" displayFormat="DD.MM.YYYY " max="2030-12-31" ></ion-datetime>

</ion-item>

</ion-list>

<ion-footer>

<button ion-button (click)="addPost()" color="secondary">Добавить</button>

</ion-footer>

</ion-content>

**Приложение Г. Код программы**

**post-list.html**

<ion-header>

<ion-navbar color="secondary">

<ion-title>Настройки</ion-title>

</ion-navbar>

</ion-header>

<ion-content>

<ion-item>

<ion-label>Уведомления</ion-label>

<ion-toggle color="secondary" [(ngModel)]="toggleNotifications"></ion-toggle>

</ion-item>

<ion-item style="color: red;" (click)="logout()">

Выйти из аккаунта

</ion-item>

<custom-tabs></custom-tabs>

</ion-content>

**Приложение Д. Код программы**

**profile.html**

<ion-header> <ion-navbar color="secondary">

<ion-title>Список продуктов</ion-title>

</ion-navbar>

</ion-header>

<ion-content>

<ion-refresher (ionRefresh)="doRefresh($event)">

<ion-refresher-content></ion-refresher-content>

</ion-refresher>

<ion-list \*ngIf="userPosts">

<ion-item-sliding \*ngFor="let post of userPosts">

<ion-item class="post">

{{post.name}}

<p>{{post.date}}</p>

<h2 item-end right \*ngIf="countOfDaysLeft(post)<0; else elseBlock;" style="color:red;">Просрочено {{countOfDaysLeft(post)\*-1}} сутки(ок)</h2>

<ng-template item-end right #elseBlock><h2 \*ngIf="countOfDaysLeft(post)>0;">Суток осталось: {{countOfDaysLeft(post)}}</h2>

<h2 \*ngIf="countOfDaysLeft(post)==0;" style="color:orange;">Часов осталось: {{countOfHoursLeft(post)}}</h2></ng-template>

</ion-item>

<ion-item-options side="right">

<button ion-button color="danger" (click)="delete(post)">Delete</button>

</ion-item-options>

</ion-item-sliding>

</ion-list>

<ion-fab right bottom>

<button color="danger" ion-fab (click)="addPost()">

<ion-icon name="add"></ion-icon>

</button>

</ion-fab>

</ion-content><custom-tabs class="a"></custom-tabs>

**Приложение Ж. Код программы**

**signup.html**

<ion-header>

<ion-navbar color="secondary">

<ion-title>Create Account</ion-title>

</ion-navbar>

</ion-header>

<ion-content>

<ion-row class="account-form">

<ion-col>

<ion-list inset>

<ion-item \*ngIf="message\_hint">

<p> {{message\_hint}} </p>

</ion-item>

<ion-item>

<ion-label><ion-icon name="mail"></ion-icon></ion-label>

<ion-input [(ngModel)]="username" placeholder="username" type="email"></ion-input>

</ion-item>

<ion-item>

<ion-label><ion-icon name="lock"></ion-icon></ion-label>

<ion-input [(ngModel)]="password" placeholder="Password" type="password"></ion-input>

</ion-item>

</ion-list>

<button ion-button (click)="register()"

class="continue-button"

block outline

color="secondary"

>

Register

</button>

</ion-col></ion-row></ion-content>