

1 лекция.

Тема: Введение. История разработки мобильных приложений. Ранние мобильные устройства и мобильные приложения. Современные мобильные ОТ

План:

- 1. Определение мобильного приложения**
- 2. История разработки мобильных приложений**
- 3. Начало мобильных телефонов**

1. Определение мобильного приложения

Футуристы и исследователи считают, что цифровые технологии, в том числе мобильные приложения, будут стремительно развиваться, исходя из прошлых тенденций.

Если вы проведете аналогию между приложениями, которые мы используем сегодня и которые мы использовали несколько лет назад, разница будет поразительной.

В начале 2000-х приложения демонстрировали только контент, который было возможно, вместо того, чтобы показывать контент, который нужен людям. Сегодня правила другие.

Мы видим интуитивный контент, который никогда не отклоняется от интересов и ожиданий пользователя. Технические новинки, среди которых гироскопы и данные о местоположении, постоянно улучшаются и заботятся об опыте пользователя.

Данные для определения демографических и других характеристик, изучение предпочтений и особенностей поведения человека по ту сторону экрана — миллионы приложений каждый день собирают информацию, чтобы лучше соответствовать вашим ожиданиям.

Все это происходит автоматически, без вмешательства человека. Вспомните, как Amazon удивляет вас продуктами, которые вы уже просматривали, а затем выводит рекомендации, основанные на предыдущих поисках.

Удовлетворенность пользователя и пользовательский опыт являются двумя ключевыми факторами, вокруг которых сегодня вращается **разработка мобильных приложений**.

Очень скоро компьютеры будут собирать все, что вы искали в соответствии с вашими предпочтениями, в приложение, отображая данные на экране смартфона.

Кроме того, мобильные приложения становятся все более разнообразными, позволяя технически подкованным пользователям находить оптимальный вариант для себя.

Люди ищут приложение, которое было бы цельным и безграничным. Мы используем по несколько разных мобильных приложений каждый день, и каждое из них значительно облегчает жизнь. С появлением Интернета вещей они переходят на качественно новый уровень, где мир сосредоточен на кончике пальца.

Мобильное приложение — это специальное программное обеспечение, которое разрабатывается для смартфонов, планшетов и других мобильных устройств.

Приложения для мобильных устройств — это настолько просто, что мы вряд ли подберем лучшее определение. Как правило, эти программы предназначены для iOS и Android.

Приложения облегчили нам жизнь, и мы достигли точки, когда мы не можем представить работу и досуг без них. Давайте же посмотрим, как все начиналось.

2. История разработки мобильных приложений

Если мы вернемся к первоисточкам проектирования и разработки мобильных приложений, то обнаружим, что первыми приложениями были календари, калькуляторы и даже игры, разрабатываемые в среде Java.

Что интересно, первый в мире смартфон был выпущен IBM в 1993 году.

Он имел такие функции, как книга контактов, календарь, мировое время и калькулятор.

Несколько лет спустя, в 2002 году, был выпущен следующий смартфон BlackBerry.

Это было одним из главных достижений в области разработанного мобильных приложений, которое сделало бессмертным имя скромной компании BlackBerry Limited, также известной как Research in Motion Limited (RIM).

Их работа привела к интеграции концепции, известной как wireless email.

Интересные факты о первых мобильных телефонах

Первые модели Motorola DynaTAC, поступившие в продажу, получили прозвище «кирпич» из-за их громоздких размеров и веса. Эти первые мобильные телефоны были более 22,5 см без учета антенны в высоту, а весили около 1,2 килограмма.

Ранние мобильные телефоны стоили почти 4000 долларов, что составляет около 10000 долларов в сегодняшних долларах с учетом инфляции. Несмотря на недостатки, технология нравилась богатым деловым людям, которые рассматривали ее как альтернативу пейджеру.

Пользователи были вынуждены заряжать свои мобильные телефоны не менее 10 часов, так как в течение дня батарея разряжалась. Пользователи могли совершать звонки в течение 30 минут в день, после чего мобила требовала зарядки.

Проблемы с сетью и радиусом действия были еще одним препятствием, поскольку они иногда не позволяли звонить абонентам даже в ближайших окрестностях.

3. Начало мобильных телефонов

В апреле 1973 года, именно 3-го числа этого месяца, Мартин Купер из Motorola впервые позвонил доктору Джоэлу Ангелю из Bell Labs по мобильному телефону.

Всего через пару десятилетий ученые активно разрабатывали операционные системы и мобильные приложения для этих устройств. Отдел исследований и разработок IBM Simon представил первое мобильное приложение для смартфонов в 1993 году.

У портативных компьютеров, или КПК, стояла первая операционная система EPOC, разработанная PSION. Выпущенное на рынок в начале 90-х, это было первое из узнаваемых приложений. 16-битная система EPOC могла запускать дневники и базы данных, электронные таблицы и текстовые процессоры.

Но будущие модели были способны работать с 32-битной ОС и были интегрированы с 2 МБ ОЗУ, что позволяло пользователям добавлять дополнительные приложения через свои пакеты программного обеспечения. Это был значительный шаг вперед.

Затем настала целая эпоха Palm OS.

Разработанные Palm Inc. в 1996 году, эти системы были в основном предназначены для персональных цифровых помощников и известны как Garnet OS. Здесь вам и графический интерфейс пользователя с сенсорным экраном, солидный набор базовых приложений, сторонние приложения на языке C / C ++ и многое другое.

Позже были представлены браузеры протокола беспроводных приложений (WAP).



Язык разметки для беспроводных устройств

Разработанный WAP Forum, язык разметки для беспроводных устройств был специально создан для систем, которые зависели от XML и работали по протоколам WAP.

Он был простой и способный работать на низкой пропускной способности мобильных сетей 90-х, преодолевая существующие ограничения HTML, языка разметки гипертекста, слишком требовательного к вычислительной мощности.

Затем вышел Java ME, J2ME или JME — он был впервые представлен как JSR 68. Позже его заменили на персонализированный Java, который стал любимым для многих. Неудивительно, почему этот язык программирования все еще существует. Ему придавали различные вариации и формы для использования в мобильных устройствах.

Кроме того, CLDC мог работать на устройствах с объемом памяти от 160 КБ до 512 КБ и поставлялся с библиотеками Java, которые способны работать на виртуальных машинах.

Операционная система Symbian стала следующим шагом. Разработанная компанией Symbian Ltd, совместным предприятием Ericsson, Motorola, Nokia и PSION, это была усовершенствованная версия PSION EPOC OS.

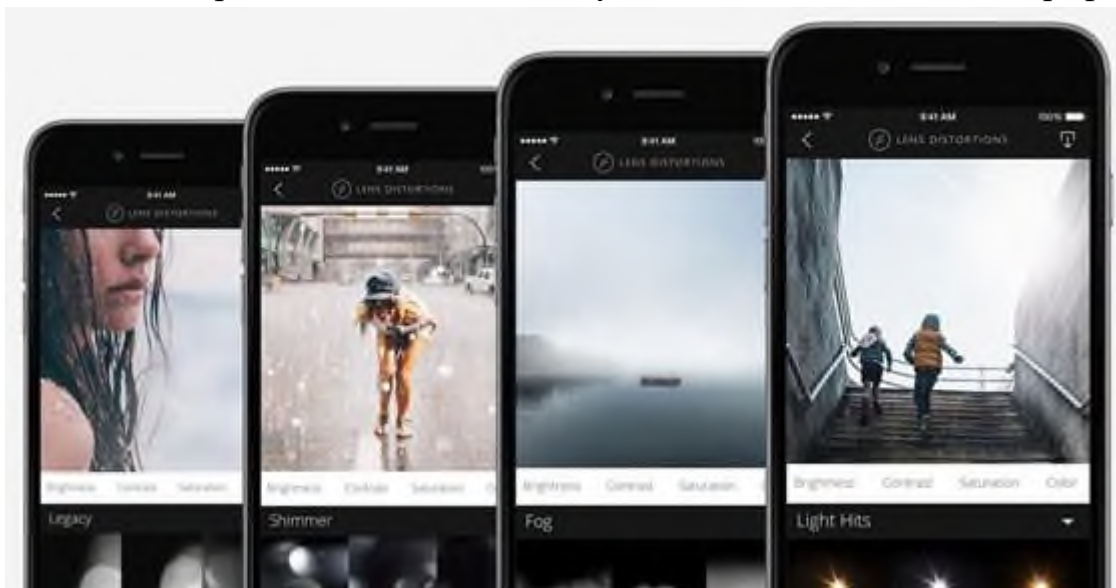
До 2008 года это интересное предприятие имело вездесущую операционную систему, способную поддерживать около 250 миллионов устройств во всем мире.

Nokia продолжила работу над импровизацией Symbian, и эта программная платформа под названием S60 была реализована на различных телефонах Nokia, Samsung и LG.

Позже современные смартфоны, которые мы используем сегодня, существенно эволюционировали, сделав жизнь людей приятней и проще. Изменения особенно коснулись доступности и удобства приложений. Подумайте, какой была бы жизнь без миллионов приложений вокруг нас, включая социальные сети, банковские услуги, здоровье и фитнес, игры, путешествия и отдых, покупки, новости и так далее.

Сегодня существуют игровые магазины, которые добавляют около 20 000 приложений каждый месяц. Вам не кажется это фантастической цифрой? Количество загрузок приложений для iPhone составляет не менее 30 миллиардов, за которыми следует загрузка приложений для Android с показателем в 15 миллиардов.

Теперь мы даже не можем представить жизнь без этих плодов эволюции мобильных приложений. Да, жизнь существовала задолго до смартфонов!



Список приложений, которые нужно попробовать

Познакомимся с некоторыми приложениями, которые стоит попробовать:

Abstract 4K

Хотите простой способ получать отличные качественные 4K обои для смартфона? Это Abstract. Создатель знаменитых обоев Хэмпс Олссон создал это приложение для всех эстетов мобильного мира, предоставив доступ к более чем 300 обоям.

Curator

Хотели бы вы организовать свою библиотеку фотографий? Тогда Curator — лучший выбор для вас. Приложение поможет пометать фото и выполнять поиск по своим тегам, облегчив доступ к многочисленным снимкам, которыми забито ваше устройство.

Curator может интеллектуально пометать фотографии на основании композиции изображения, и эта функция работает действительно эффективно.

Lens Distortion

Это мобильное приложение помогает добавить высококачественные эффекты к вашим изображениям. Некоторые из эффектов включают туман, дождь или снег, естественный солнечный свет, и все можно увидеть одновременно в виде галереи.

Удобный функционал облегчит выбор фотографии по своим предпочтениям. В бесплатном приложении вы найдете пять фильтров, но если хочется больше, всегда можно подписаться на платную версию программы *Lens Distortion Unlimited*.

Dinggo

Когда вы подписываетесь на слишком большое количество потоковых сервисов, вы попросту не можете определиться, что следует смотреть. Значит пора установить *Dinggo*.

Это идеальное решение. Программа может подбирать потоковые сервисы по предпочтениям, выбирая жанр из нескольких телешоу или фильмов, которые могут вас заинтересовать.

Dinggo позволяет просматривать широкий спектр опций, делая свой мобильный кинозал одновременно разнообразным и отвечающим личным предпочтениям, не за цикливаясь на рекомендациях нескольких сервисов.

2 лекция.

Тема: Языки и технологии создания мобильных приложений: программирование под iOS и Android, эмуляторы.

План:

1. Обзор существующих технологий разработки мобильных приложений

Сейчас у всех есть сотовый телефон, смартфон или КПК. В каждом из них есть свои мобильные приложения. Они позволяют расширить возможности Вашего мобильного устройства. Мобильные приложения позволяют общаться с друзьями, выходить в интернет, смотреть погоду и многое другое. Существует множество технологий, которые используются для создания мобильных приложений и которые часто обновляются, а иногда и появляются новые. Но многие из них не дают возможность разрабатывать приложения для различных ОС, и это является существенным недостатком данных технологий.

В последнее время многие разработчики программного обеспечения (ПО) задумались о том, каким образом можно усовершенствовать технологии так, чтобы приложения были не только мощными, но и кросс-платформенными.

1. Обзор существующих технологий разработки мобильных приложений

В настоящее время существует хороший выбор языков программирования для разработки мобильных приложений. Это связано с тем, что для различных мобильных устройств приходится использовать различные языки программирования. Обычно это связано с тем, что мобильные устройства имеют различные ОС.

Ниже будут рассмотрены такие технологии как Java, Qt (основанный на библиотеках C++), Windows Phone SDK (написание на языке XAML), iPhone SDK (основной язык - Objective-C), Android SDK (основной язык - Java) и Symbian (основной язык - C++).

1. Java 2 Micro Edition (J2ME). В первую очередь J2ME это набор спецификаций и технологий, предназначенных для различных типов портативных устройств. Существуют два основных направления: Connected Device Configuration (CDC) и Connected Limited Device Configuration (CLDC).

Направление определяет тип конфигурации центральных библиотек Java, а также параметров виртуальной машины Java (в которой будут исполняться приложения). Логично предположить, что устройства CDC будут более «развитыми», в качестве примера можно привести коммуникаторы.

К устройствам CLDC относятся обычные мобильные телефоны, аппаратно обладающие более скромными возможностями (ресурсами). Специальные режимы позволяют определять функциональность конфигураций для различных типов устройств. Режим Mobile Information Device Profile (MIDP) предназначен для CLDC портативных устройств с возможностью коммуницировать. Режим MIDP определяет функциональность - работу пользовательского интерфейса, сохранение настроек, работу в сети и модель приложения. CLDC и MIDP закладывают основу реализации J2ME [1].

Java-код интерпретируется непосредственно самим устройством при помощи так называемой Java Virtual Machine. Этот механизм делает возможным свободное распространение Java-приложений, так как они работают на всех устройствах с аналогичной Java-платформой [2].

Программирование Java-приложений и на сегодняшний день занимает большую часть, так как большинство мобильных устройств (в основном мобильные телефоны) в мире имеют уже предустановленную Java-машину.

2. Qt. Среда разработки Qt была приобретена Nokia в 2008 г. у норвежской Trolltech за 150 миллионов долларов. Qt в основном используется в качестве кросс-платформенной среды, которая позволяет использовать написанные с ее помощью приложения на различных устройствах и операционных системах, в том числе Windows, Mac OS X, Linux, Symbian, Android и других [3]. Начиная с версии Qt 4.0 появилась возможность программировать для мобильных устройств. С растущей пользовательской базой Qt, растёт потребность во встроенных, мобильных приложениях и UI-разработчиках.

Qt является одной из самых удачных библиотек для C++. Отладка приложений, разработанных для мобильных устройств, происходит с помощью эмулятора, который содержится в среде разработки. Таким образом, мы можем писать сложные приложения для мобильных устройств с использованием библиотек C++ и поддержкой кроссплатформенности.

В настоящее время последняя версия - Qt 5 бета. Финальный релиз планируется на 2012 год. Для работы Qt на мобильных устройствах необходима установка соответствующего Фреймворка.

3. Windows Phone SDK. На момент написания этой статьи, последняя версия инструментария доступна в версии Windows Phone SDK 7.1 Release Candidate в лицензии «Go Live» с возможностью разрабатывать свои приложения и публиковать их в Windows Phone Marketplace. Windows Phone SDK 7.1 Release Candidate содержит следующие компоненты [4]:

- Windows Phone SDK 7.1;
- Windows Phone Emulator;
- Windows Phone SDK 7.1 Assemblies;
- Silverlight 4 SDK and DRT;
- Windows Phone SDK 7.1 Extensions for XNA Game Studio 4.0;
- Expression Blend SDK for Windows Phone 7;
- Expression Blend SDK for Windows Phone OS 7.1;
- WCF Data Services Client for Windows Phone;
- Microsoft Advertising SDK for Windows Phone.

Код разрабатываемого приложения описывается на языке XAML. На самом деле - это просто XML файлы с языком разметки XAML.

Платформа Windows Phone не просто очередная платформа для мобильных устройств. Она содержит в себе не только технологическую составляющую, но и полностью проработанную концепцию дизайна интерфейса и взаимодействия с пользователем под названием Metro-дизайн или стиль Metro.

Вся разработка под Windows Phone ведется в среде Visual Studio. Среда является очень удобной для разработки и отладки приложений. Для мобильных приложений под Windows Phone отладка происходит с помощью эмулятора Windows Phone с помощью среды разработки Windows Phone.

4. iPhone SDK. Разработка под iPhone под операционную систему iOS возможна только под Mac OS X. Но в Интернете можно найти статьи, как можно программировать и на Macintosh и даже на VM. Стоит заметить, что Apple предоставляет инструменты бесплатно, платить придется за подписку разработчика [5].

Для написания программ под iPhone предлагается использовать Objective-C. При этом есть возможность писать так же и на C и на C++ (для этого необходимо изменять расширения файлов с .m на .mm). Правда при этом полностью уйти от Obj-C не удастся, почти весь API рассчитан именно на Obj-C, исключения составляют например OpenGL (хотя для его инициализации придется использовать несколько строк кода на Obj-C), так же полностью доступны стандартные библиотеки C/C++ (так, например, с файловой системой можно работать как средствами SDK на Obj-C, так и используя стандартную библиотеку C для ввода/вывода (fopen(), fgetc(), etc)).

Отладка приложения происходит с помощью среды XCode и эмулятора iPhone установленного в ней.

5. Android SDK. Для разработки под Android можно использовать среду Eclipse с установленным плагином ADT. Разработка ведется на языке программирования Java. Есть возможность отладки с использованием эмулятора встроенного в ADT или непосредственно на мобильном устройстве с ОС Android.

Существует различные версии SDK, которые используются для написания кода для различных версий Android. В настоящее время большое распространения получили версии 2.2 и 2.3. Поддерживается почти полная обратная совместимость версий.

Кроме разработки на языке Java поддерживается возможность более низкоуровневая разработка с использованием Android NDK (Native Development Kit) на языке C/C++.

6. Symbian и C++. Для написания приложений под Symbian можно использовать язык программирования C++. В основном данный подход используется для Symbian OS v6.1, 7.0, 7.0s и 8.0 [6].

Разработка для Symbian OS (если говорить о C++) обычно ведется на ПК. Среда разработки - привычная многим программистам Visual Studio, это также могут быть IDE Metrowerks CodeWarrior Development Studio, Borland C++BuilderX Mobile Edition, Carbide.C++ (относительно новая IDE, созданная компанией Nokia на базе Eclipse), снабженная дополнительными инструментальными пакетами (SDK). Разработчику доступны практически все привычные возможности в отношении как создания ПО, так и отладки (трассировка, просмотр переменных, стека вызовов, структур классов и др.).

Отлаживаемая программа запускается в эмуляторе Symbian OS. Отметим, что эту подсистему правильнее было бы назвать симулятором, поскольку имитируются не аппаратные средства, а лишь программное окружение (соответствующие API операционной системы, реализованные поверх API Win32). При этом программные модули, которые загружаются в эмулятор, представляют собой исполнимые файлы для архитектуры x86 (не ARM, на базе которой построены смартфоны), соответствующее ПО для целевой платформы формируется после итоговой компиляции. Это предполагает определенную специфику (скажем, ранее была довольно распространена ситуация, когда программа, нормально функционировавшая в среде эмулятора, отказывалась работать на реальном устройстве), но сегодня эмулятор обеспечивает достаточно высокую степень сходства и проблемы возникают лишь при создании программ, нестандартно использующих API [7].

Выводы по результатам обзора. С появлением новых технологий, ранее использовавшиеся уходят в историю. Сейчас разработчиков, которые используют такие технологии как Symbian с использованием C++ и/или J2ME, становятся все меньше и лидирующую позицию занимают технологии, использующие различные SDK (Windows Phone SDK, iPhone SDK, Android SDK). Но недостаток всех существующих SDK в том, что разрабатываются нативные приложения, т.е. приложения, функционирующие под управлением только одной ОС.

Платформа PhoneGap

Разработчикам мобильных приложений приходится непросто, когда они разрабатывают код для каждой ОС, разбираясь с различными SDK, компиляторами и эмуляторами.

PhoneGap - это платформа с открытым исходным кодом от компании Nitobi (в настоящее время купленная Adobe), которая позволяет разрабатывать приложения для нескольких мобильных платформ, используя стандартные Веб-технологии [11]. Создатели кросс-платформенного фреймворка PhoneGap постарались упростить задачу: позиционируя себя как единственный open source мобильный фреймворк с поддержкой шести мобильных платформ [8].

В настоящее время поддерживаются такие операционные системы как: Android, iOS, Symbian, Windows Phone, Bada, WebOS. Каждая из перечисленных ОС имеет полную или ограниченную функциональность (рис. 1), [9].

Написание приложения ведется на JavaScript с использованием HTML и CSS для разметки. Вы пишете мобильное приложение как обычный сайт или Веб-сервис.

Платформа PhoneGap расширяет API браузера и добавляет следующие возможности: доступ к акселометру, доступ к камере (пока только фото), доступ к компасу, доступ к списку контактов, запись и прослушивание аудио файлов, предоставляет доступ к файловой системе, позволяет работать с разными HTML5 хранилищами localStorage, Web SQL и т.п., а также позволяет безболезненно обращаться к любому кросс-доменному адресу [12].

Для того чтобы использовать PhoneGap необходимо скачать с официального сайта последнюю версию PhoneGap (на момент написания статьи она была 1.3.0) и указать в среде разработки расположение библиотеки. Так же необходимо подключить JavaScript файл, который расположен в папке вместе с библиотекой под нужную платформу.

С помощью PhoneGap Build - онлайн-конвертер из HTML 5. Нужно загрузить приложение на html/js/css - и на выходе получаете готовый бинарный файл для Apple App Store, Android Marketplace, Palm, Symbian или BlackBerry.

У приложений, разработанных под PhoneGap, есть ряд достоинств и недостатков.

Начнем с *достоинств*:

1. Очень просто реализовать совместную разработку приложения.
2. Написание кода ведется на HTML, Java Script, CSS с возможностью использования сторонних библиотек.
3. Поддержка кросс-платформенности (в настоящее время 6 платформ);
4. Отладка приложений с помощью браузера.
5. Возможность конвертации приложения под все необходимые платформы с помощью PhoneGap.

Недостатки разработки под PhoneGap:

1. Приложения не поддерживают многопоточность.
2. Проблемная реализация длинных списков (более 1000).
3. Не все приложения могут быть оформлены как Web-приложения.
4. Обращение к аппаратным частям мобильного устройства ведется по разному, что может вызывать неожиданный эффект.
5. Сложность настройки программной среды для написания приложений под PhoneGap.

Ведя разработку с использованием платформы PhoneGap, мы получим мощное кросс-платформенное приложение, которое будет удовлетворять многих пользователей. Простота синтаксиса и отладки уменьшает время по изучению данной технологии по сравнению с такими языками, как Qt или Java. Следовательно, если приложение не является вычислительным и ресурсоемким, то разработка с помощью платформы PhoneGap является оптимальной и в сумме займет меньше времени, по сравнению с разработкой кросс-платформенного приложения под различные ОС.

3 лекция.

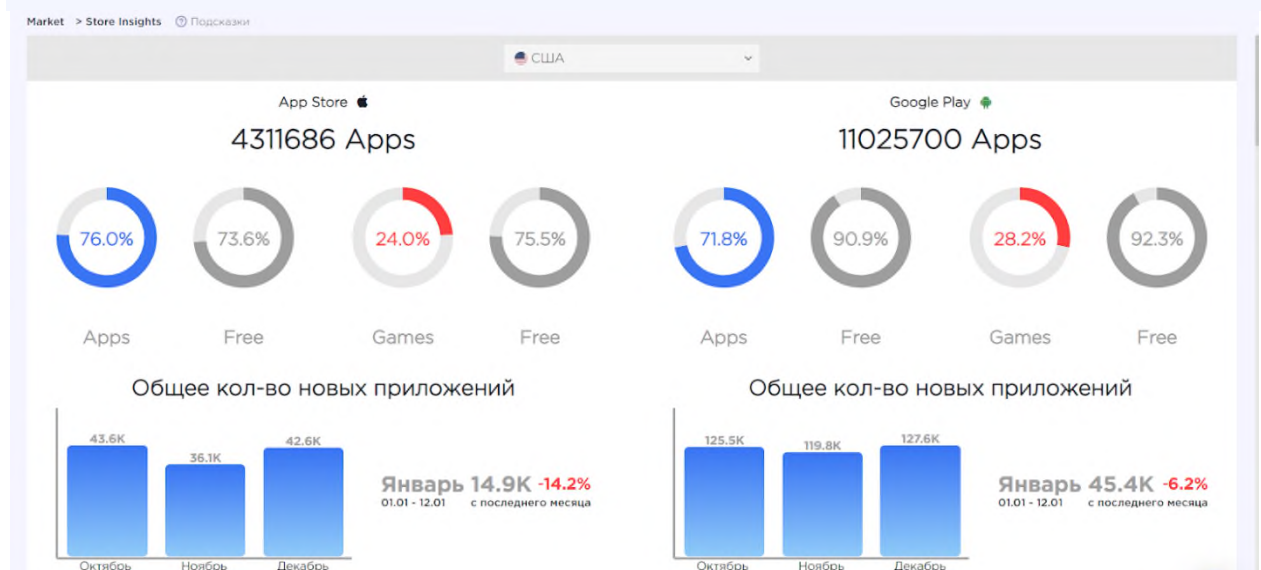
Тема: Платформы для мобильных приложений. Интернет-магазины Play Market, Apple Store.

План:

1. История магазинов приложений.
2. **АЛЬТЕРНАТИВНЫЕ МАГАЗИНЫ ANDROID**
3. **ЛУЧШИЕ АЛЬТЕРНАТИВЫ APPLE APP STORE**
4. **УНИВЕРСАЛЬНЫЕ ПЛОЩАДКИ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ ДЛЯ IOS, ANDROID И НЕ ТОЛЬКО**

1. История магазинов приложений.

Эволюция магазинов приложений, Google объявил об открытии онлайн-площадки приложений на базе Android - Android Market, 22 октября 2008 года, лишь немного отстав от App Store, который празднует свой день рождения начиная с 10 июля 2008 года. Пройдя через определенные этапы становления и развития (смена названий, новые правила и порядки внутри маркета) эти два гиганта теперь практически монополизировали рынок и закончили 2021 год с показателями:



И это только на рынке США. Показатели и статистику по каждому маркету и стране вы легко можете посмотреть во вкладке [Store Insights](#) от **ASOMobile**.

Всеобщая популярность этих двух маркетов определила интересы пользователей. Мы посещаем Google Play Market с целью найти действительно нужное приложение, зная что оно безопасно, наши покупки проходят через надежную систему безопасности и разработчики также, в какой-то мере, верифицированы маркетом. App Store представляет аналогичный надежный

источник с самым широким выбором игр и приложений, с крутыми редакторскими подборками, новейшими трендами и клиентоориентированным сервисом.

С точки зрения пользователей - все довольно очевидно, но давайте посмотрим на маркет со стороны разработчика. Для продвижения и популяризации своего приложения, стоит рассматривать не только то, что очевидно, лежит на поверхности. Ведь это только на первый взгляд, App Store и Google Play - “единственная возможность”, а это совсем не так и давайте посмотрим почему.

Зачем выбирать альтернативные маркеты для разработчиков?

1. **Низкая конкуренция.** На малоизвестных площадках количество публикуемых приложений в разы меньше, но будем справедливы - пользователей также меньше.
2. **Различные системы распределения дохода.** Это было бы неправдой, если бы мы сказали, что каждый альтернативный магазин приложений может предложить столь же прибыльную модель распределения доходов, как Google Play Store или App Store. Но сторонние магазины приложений платят за загрузки и часто предлагают более выгодные предложения, чем два популярных магазина приложений.
3. **Фичеринг.** Альтернативы магазинов могут предложить приложениям желанное место в списке рекомендуемых приложений, которые не попадают в десятку лучших чартов, например в Google Play.
4. **Продвижение и реклама со стороны стор.** Альтернативные маркеты могут лучше продвигать ваше приложение, показывая его как приложение дня или предлагая скидки на рекламу разработчикам.
5. **География.** Если мы говорим про рынок Китая, то локальные сторы это важная доля рынка. Ведь тот же Google Play Market там не работает.

А что же про пользователей? Зачем выбирать альтернативные маркеты?

1. **Возможность выбора.** Конечно хорошо, когда есть проверенный маркет с безопасными и разнообразными вариантами игр и приложений. Но согласитесь, иметь право выбора - это всегда приятно.
2. **Бесплатно и уникально.** Существует много других альтернативных магазинов приложений, у которых может не быть такой огромной коллекции приложений, но у них есть уникальные функции, на которые стоит обратить внимание. Они даже позволяют загружать различные приложения, которые традиционные сторы не предлагает. Не только это,

иногда сторонние магазины предлагают приложения бесплатно, а на традиционных площадках они находятся в категории платных.

3. **Локальные приложения.** Если вас, как пользователя интересует продукт, который в большей степени ориентирован на локальный рынок - альтернативные маркеты отлично справятся с данной задачей.

2. АЛЬТЕРНАТИВНЫЕ МАГАЗИНЫ ANDROID

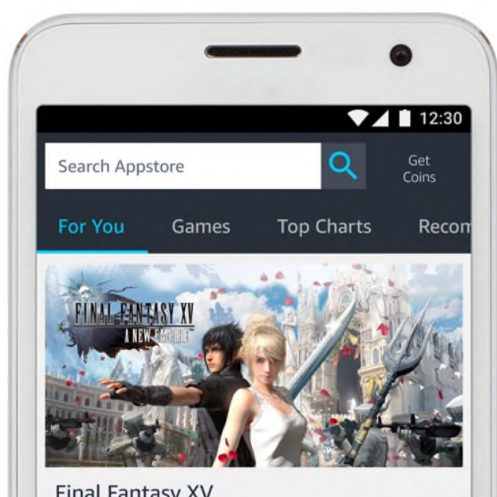
Google следует подходу открытого рынка приложений, владельцы приложений Android могут использовать эту политику в своих интересах, отправляя свое приложение в следующие лучшие альтернативы магазинов приложений Android.

Ну а для пользователя, не стоит забывать, что установка приложений из любого источника, кроме Google Play, заблокирована по умолчанию. Первое, что вам нужно сделать, это включить установку приложения из непроверенных источников, после наслаждайтесь возможностью устанавливать приложения (или APK) за пределами Google Play Market.

Amazon App Store

Для многих станет сюрпризом, что у этого гиганта электронной коммерции также есть магазин приложений, который является отличной альтернативой Google Play Store.

Выпущенный 22 марта 2011 года магазин приложений [Amazon App Store](#) предлагает множество бесплатных и платных приложений, а также эксклюзивные игровые приложения с простым и понятным интерфейсом.



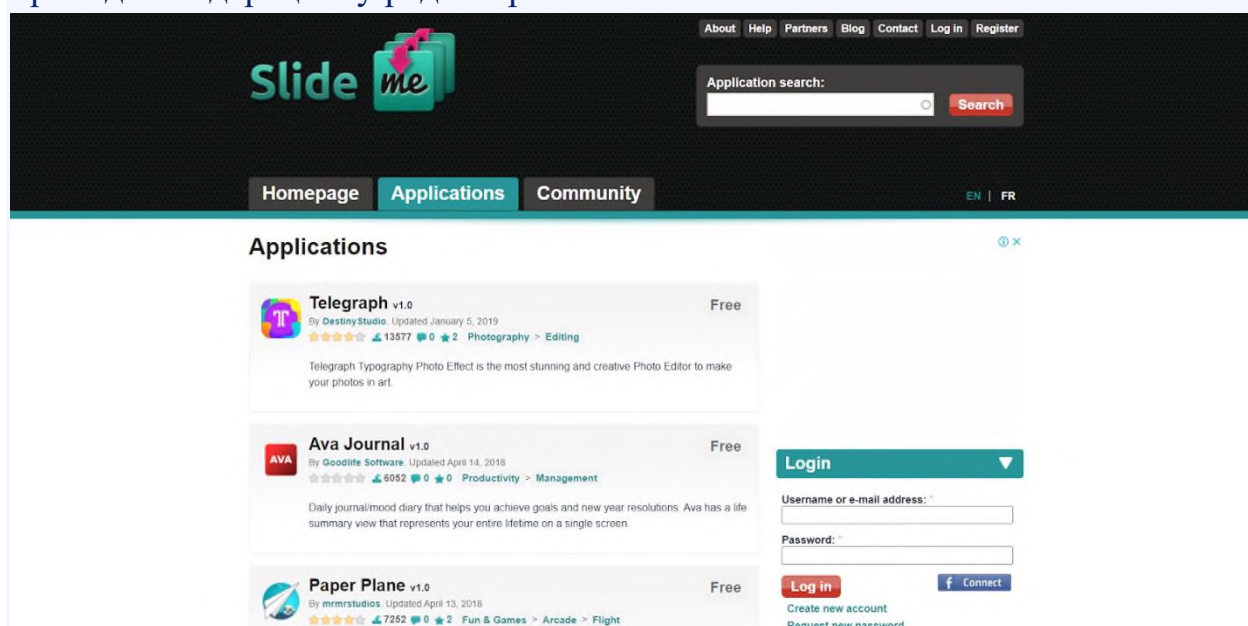
Welcome to the Amazon Appstore

Discover top apps and games and get recommendations based on the titles you love.

SlideMe - американский альтернативный магазин мобильных приложений для Android, впервые стартовал в 2008 году, как торговая площадка, ориентированная на различные типы предпочтений аудитории. Он быстро прославился, как вторая лучшая альтернатива магазина приложений

Google Play Market.

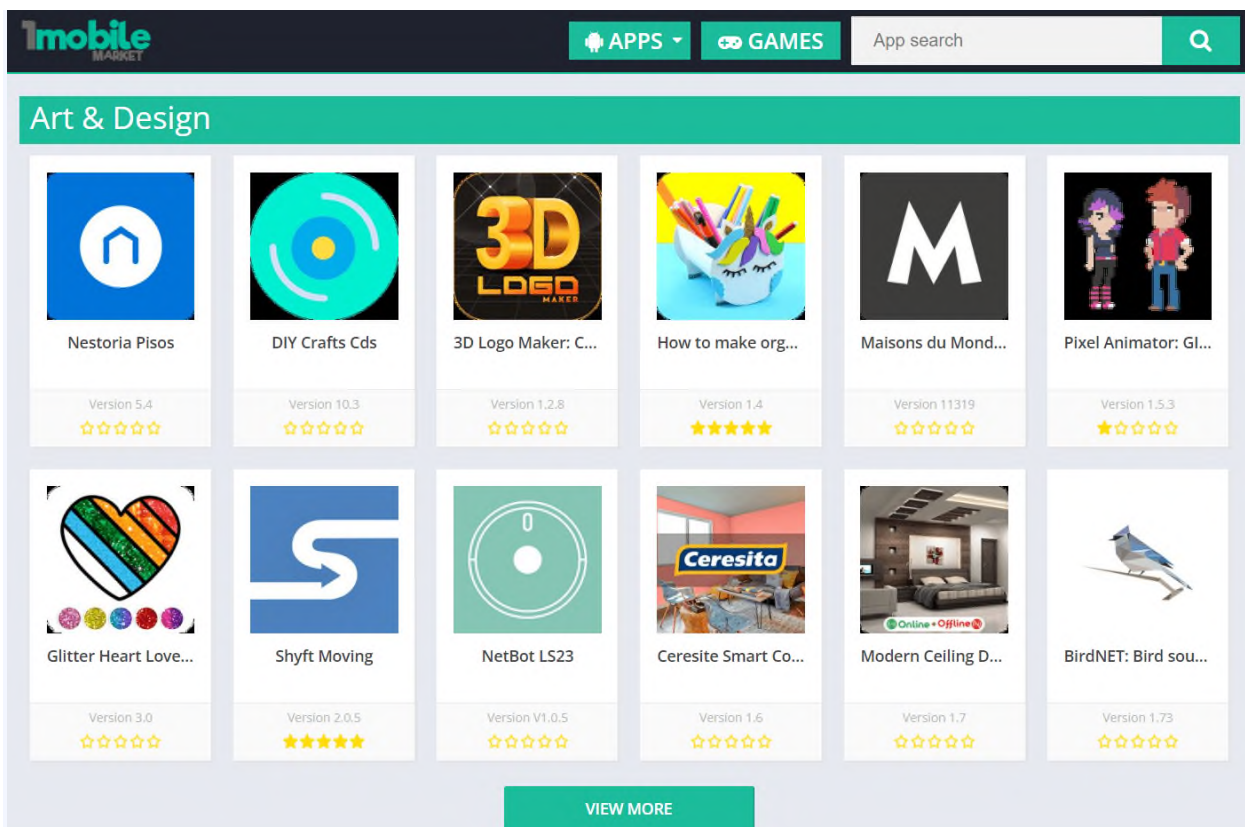
В приоритете данной платформы находится безопасность приложений для пользователей - все приложения сканируются на наличие вирусов и проходят модерацию у редакторов.



1Mobile

Помимо бесплатного размещения множества мобильных игр, веб-приложений и даже видео, **1Mobile** предлагает очень простой поиск необходимых приложений. Для разработчиков - это отличная возможность тестировать бета-версии и бесплатные версии платных приложений в этой альтернативе Play Store.

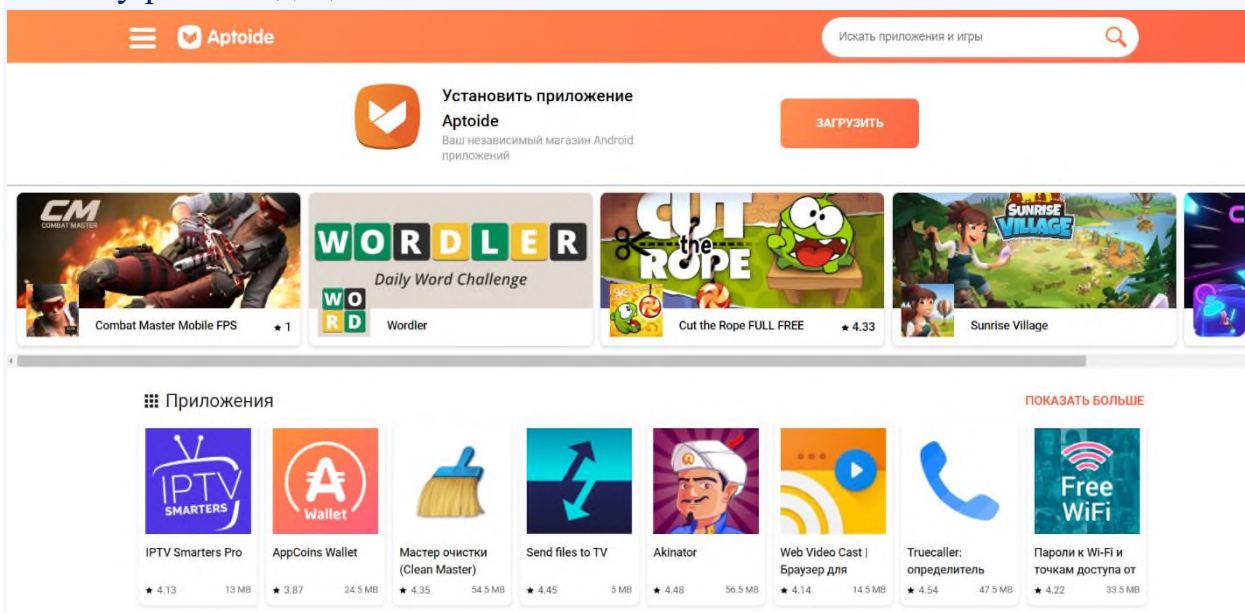
Кроме безопасности приложений для пользователей, для разработчиков также есть бонусы - своя встроенная система продвижения приложений, что советами и рекомендациями помогает настроить все самостоятельно. Эта площадка содержит подборки ТОП приложений, как местных, так и глобальных.



Aptoide

Это одна из первых альтернатив магазинам приложений для Android, в которой размещаются приложения с рейтингом 4 и 5 звезд. Будучи независимой платформой с открытым исходным кодом, [Aptoide](#), альтернатива Play Store, позволяет своим пользователям находить различные приложения для Android с такими категориями, как «Лучшие загрузки», рекомендованные Aptoide, служебные приложения с отдельными категориями.

Это упрощает для конечных пользователей поиск приложений по категориям и предлагает владельцам приложений более настраиваемую систему рекомендаций.



AppBrain

Платформа основана в 2009 году в сотрудничестве с компанией AppTornado из Цюриха. AppBrain является одной из самых надежных альтернатив магазинам приложений на базе Android. Этот магазин приложений использует очень прозрачную модель продвижения приложений, предлагая цену за установку (CPI), немотивированный пользовательский трафик и алгоритмы обнаружения мошенничества.

Это больше напоминает инструмент для монетизации и продвижения, чем на стандартную площадку приложений. Вы можете получить необходимые рекомендации, статистику и функцию удаленной установки.

Android Apps > Trivia > 3in1 Quiz : Logo Quiz - Flag Quiz - Capital Quiz



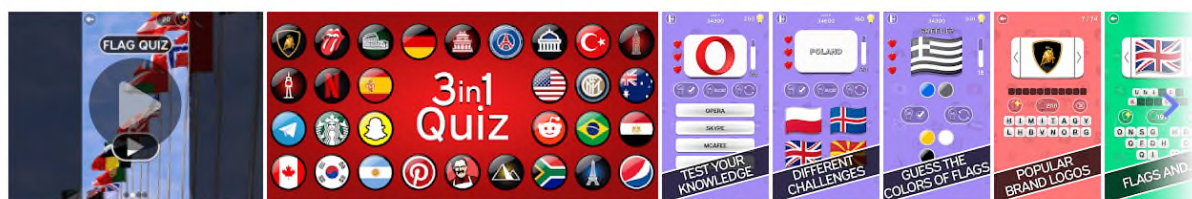
3in1 Quiz : Logo Quiz - Flag Quiz - Capital Quiz

Flag Quiz, Capital Quiz, Logo Quiz in one.

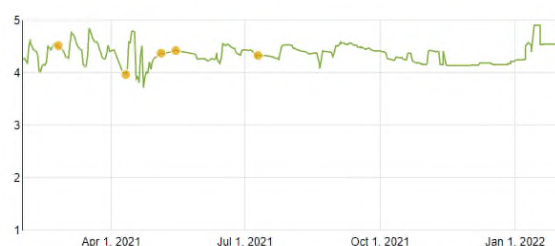
by VnS



1+ Downloads	PREMIUM Est. downloads	PREMIUM Recent d/loads	4.55 Rating 4,127	Highly ranked Ranking	21 Libraries	5.0+ Android version	7/10/21 Last updated	2020 April App age	44.1 MB App size	Everyone Content rating	FREE + In-App Price
-----------------	---------------------------	---------------------------	-------------------------	-----------------------------	-----------------	----------------------------	-------------------------	--------------------------	---------------------	----------------------------	---------------------------



Google Play Rating history and histogram



Changelog

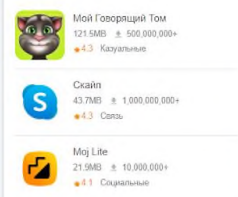
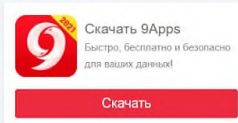
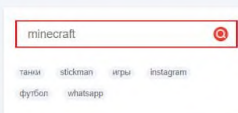
Dec 11, 2021	INSTALLS	1,000,000+ installs
Jul 10, 2021	UPDATE	Version 2.2.0
May 15, 2021	UPDATE	Version 2.0.8
May 5, 2021	UPDATE	Version 2.0.7
Apr 19, 2021	INSTALLS	500,000+ installs
Apr 11, 2021	UPDATE	Version 2.0.5
Feb 24, 2021	UPDATE	Version 2.0.4

9Apps

Еще один отличный пример альтернативных площадок приложений для Android устройств - 9Apps.

Магазин позволяет загружать бесплатные приложения и игры для Android. Принцип аналогичный, как и Play Market - лучшие рекомендации приложений, широкий диапазон категорий, мультиязычность и удобный поиск.

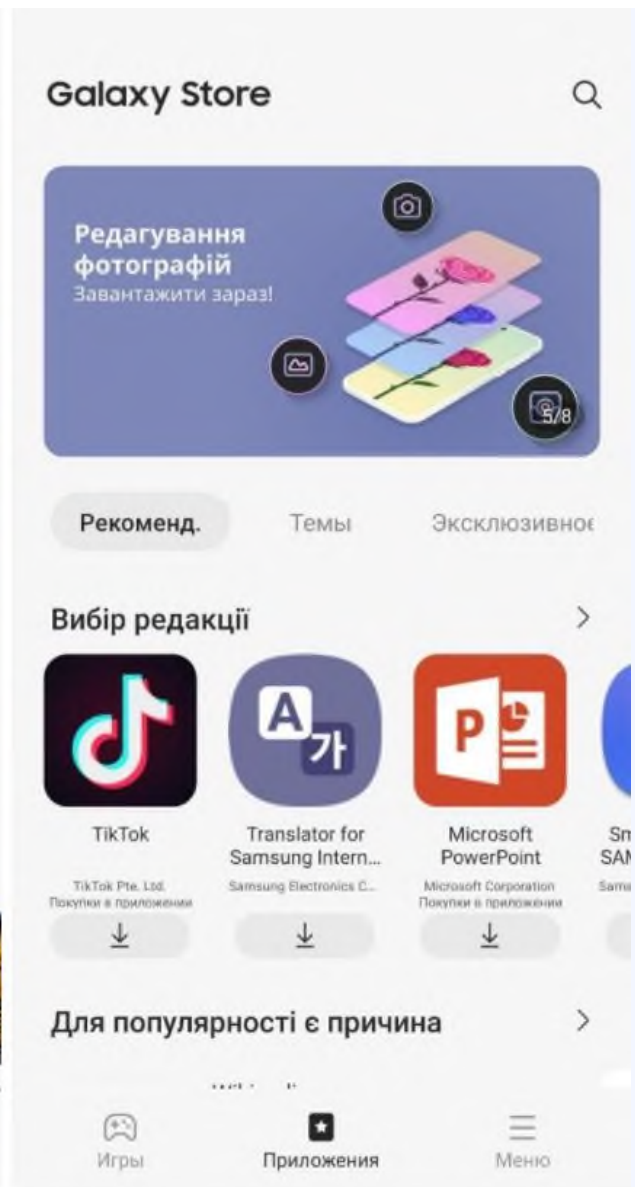
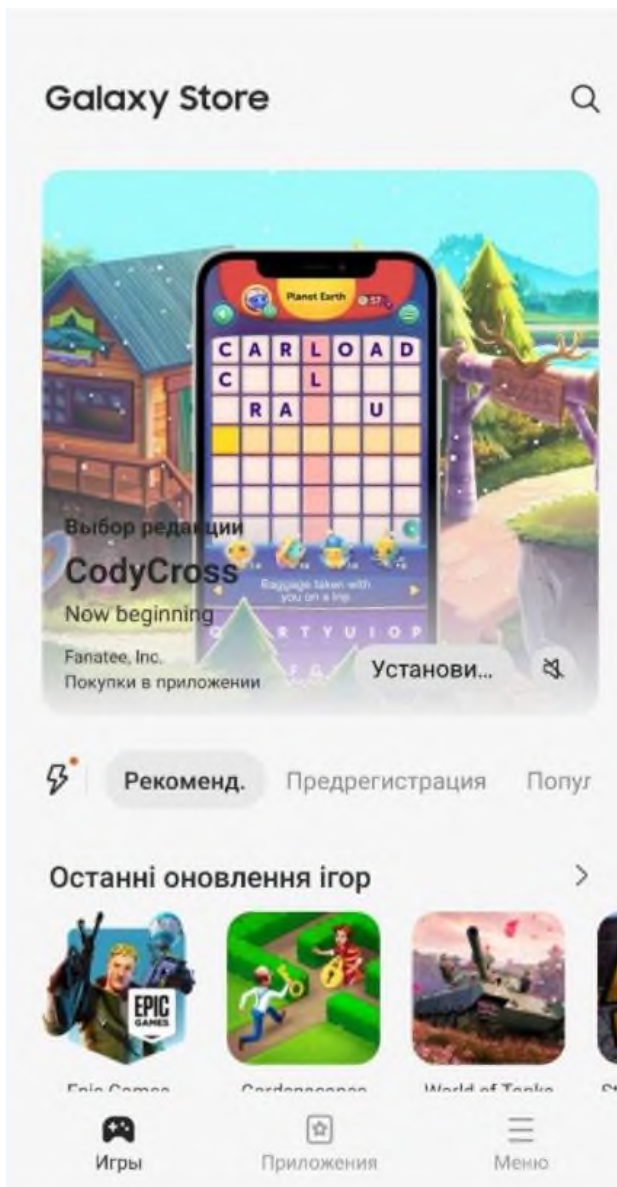
Можно установить 9Apps с [официального сайта](#).



Samsung Galaxy Apps

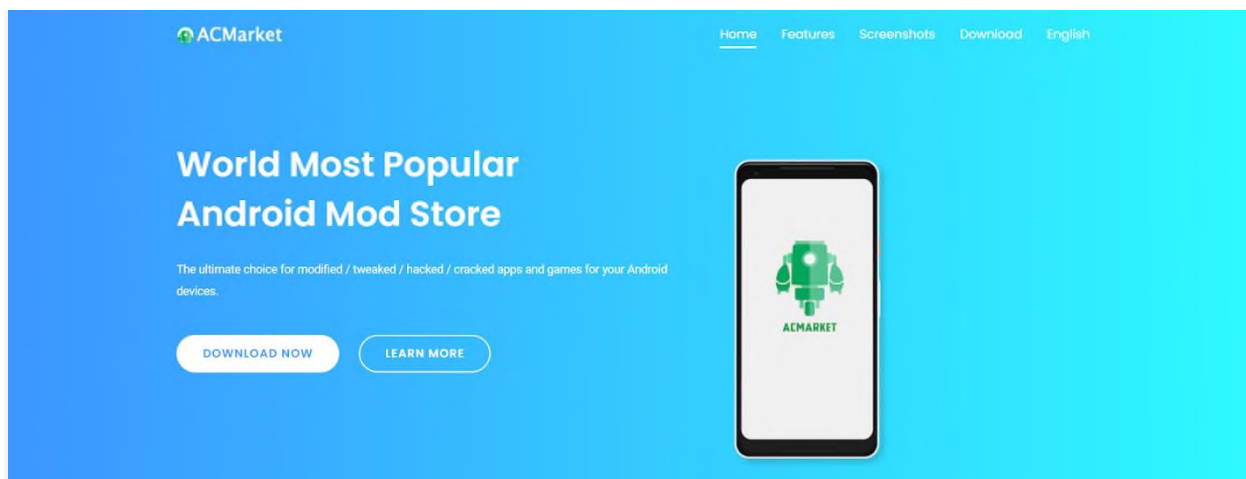
Компания-производитель смартфонов Samsung не смогла обойти вниманием такую нишу, как магазин приложений. Он ориентирован на владельцев одноименных устройств. По сравнению с другими магазинами приложений, в Samsung Galaxy Apps относительно небольшое количество приложений, но это может быть положительным моментом для того, чтобы приложения выделялись.

Плюс, это является предустановленным приложением во всех девайсах, что облегчает использование. Даже при наличии Google Play Market на телефоне, устройство будет предлагать обновления текущих приложений (такие как Темы, Галерея, Редактор фото) именно из Samsung Galaxy Store.



ACMarket

ACMarket был разработан специально для взломанных или модифицированных приложений. Это один из крупнейших неофициальных магазинов с более чем 15 тысячами приложений. Поскольку этот магазин приложений постоянно развивается и вносит улучшения, он часто предоставляет новые обновления.



Это далеко не полный список альтернативных источников для установки приложений на базе Android. Ниже мы подготовили для вас перечень кроссплатформенных магазинов, на которые тоже можно обратить внимание.

3. ЛУЧШИЕ АЛЬТЕРНАТИВЫ APPLE APP STORE

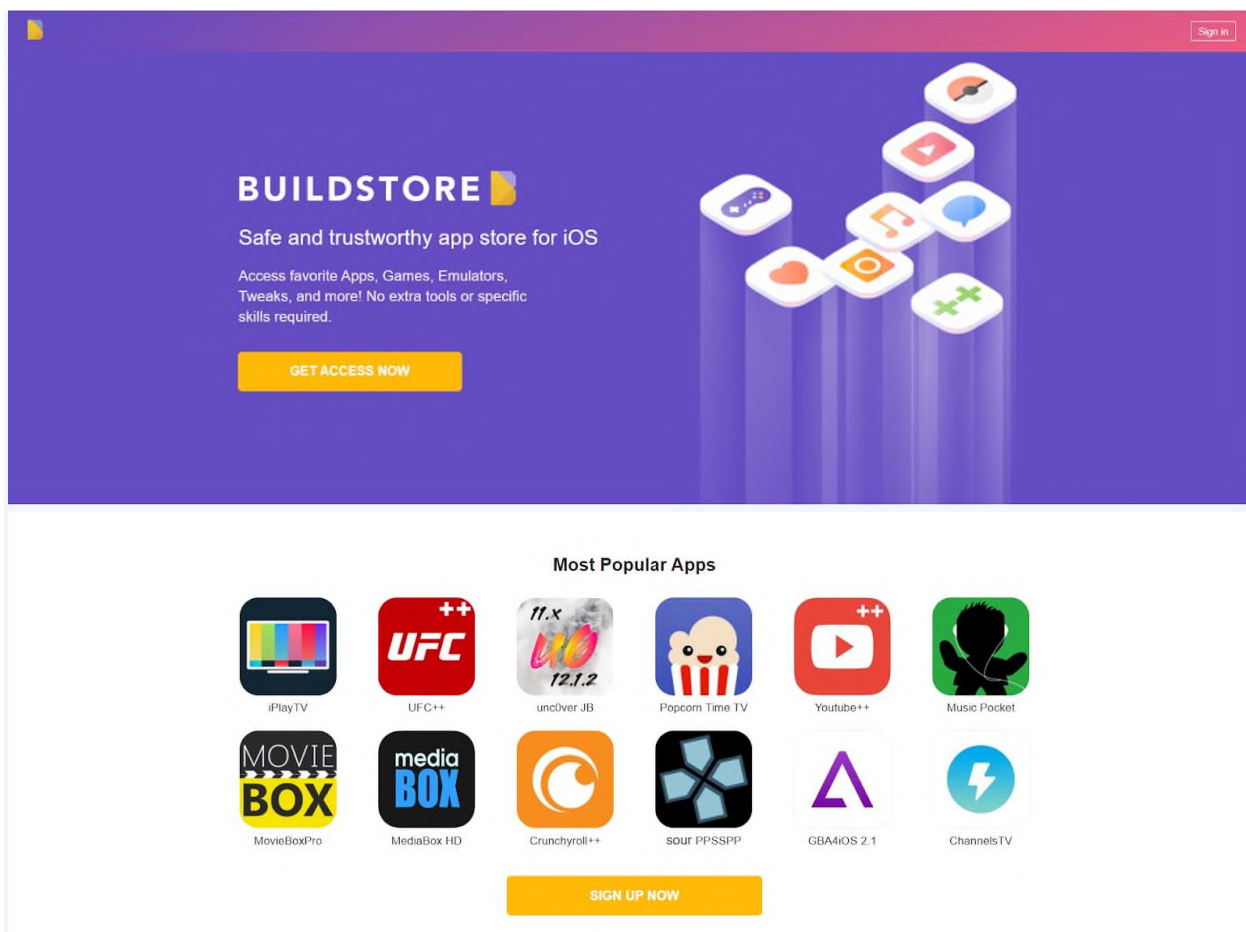
Монополия App store для владельцев iPhone кажется безоговорочной, но на самом деле вариантов больше, чем кажется на первый взгляд. Ведь альтернативные сторы для iOS существуют и могут принести пользу вашему приложению.

Так как на iOS, в отличие от Android, запрещена загрузка и запуск сторонних приложений в обход App Store, работа альтернативных магазинов возможна только на устройствах с джейлбрейком (взломанной операционной системой). Это конечно сужает аудиторию до тех, кто может выполнить эту непростую процедуру.

Мы подготовили для вас список альтернативных площадок, где можно опубликовать свое приложения или игру на базе iOS и установить ее без джейлбрейка (или же с ним).

Builds.io

[Builds.io](#) - это веб-сайт, который позволяет вам устанавливать приложения iOS на ваше устройство прямо из браузера (Safari).



Площадка была основана в 2013 году и представляет собой веб-сайт для размещения приложений, с менее строгими правилами чем официальный магазин. Контентная политика Builds.io очень лояльна к инди-разработчикам, но в тоже время следит за безопасностью предлагаемых апп.

Cydia

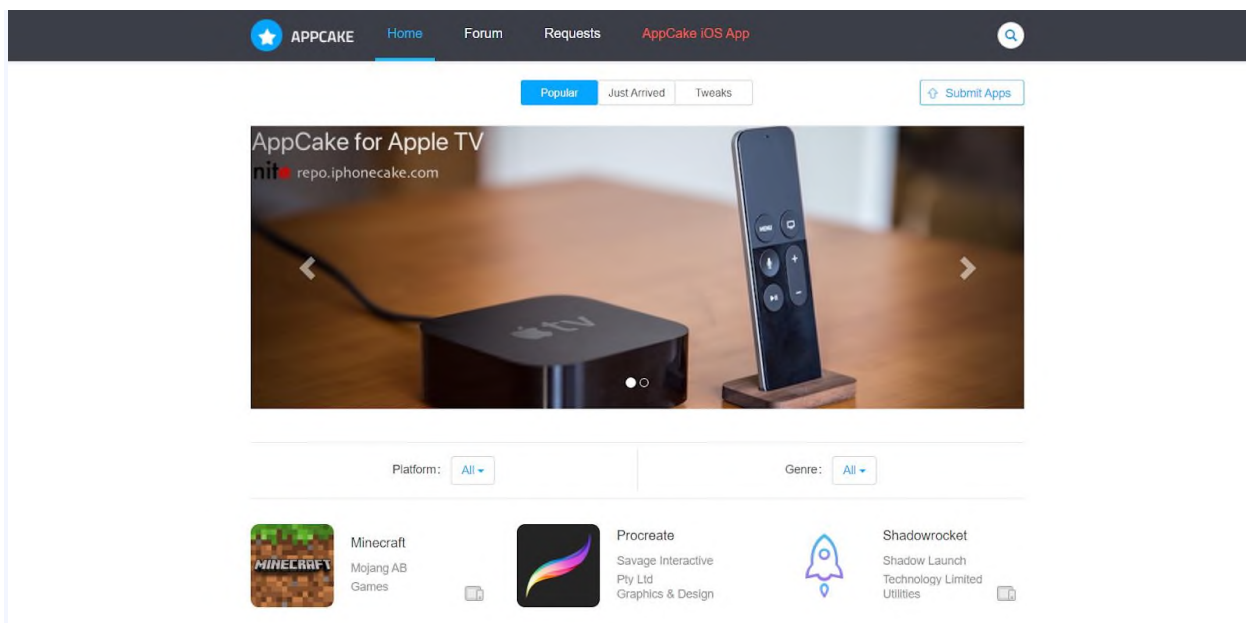
Cydia - это возможность установки сторонних приложения на ваш iPhone, правда начало работы необходимо начать с джейлбрейка вашего устройства. Магазин приложений Cydia запустился еще в 2008 году — это был первый магазин приложений для iPhone, который предлагал приложения еще за несколько месяцев до того, как у Apple появился собственный App Store. С тех пор, Cydia служит репозиторием приложений для iPhone и iPad с джейлбрейком, упрощая установку неавторизованного программного обеспечения на совместимые устройства.

В конце 2020 года Cydia присоединяется к растущему числу разработчиков, обвиняющих Apple в анти конкурентном поведении.



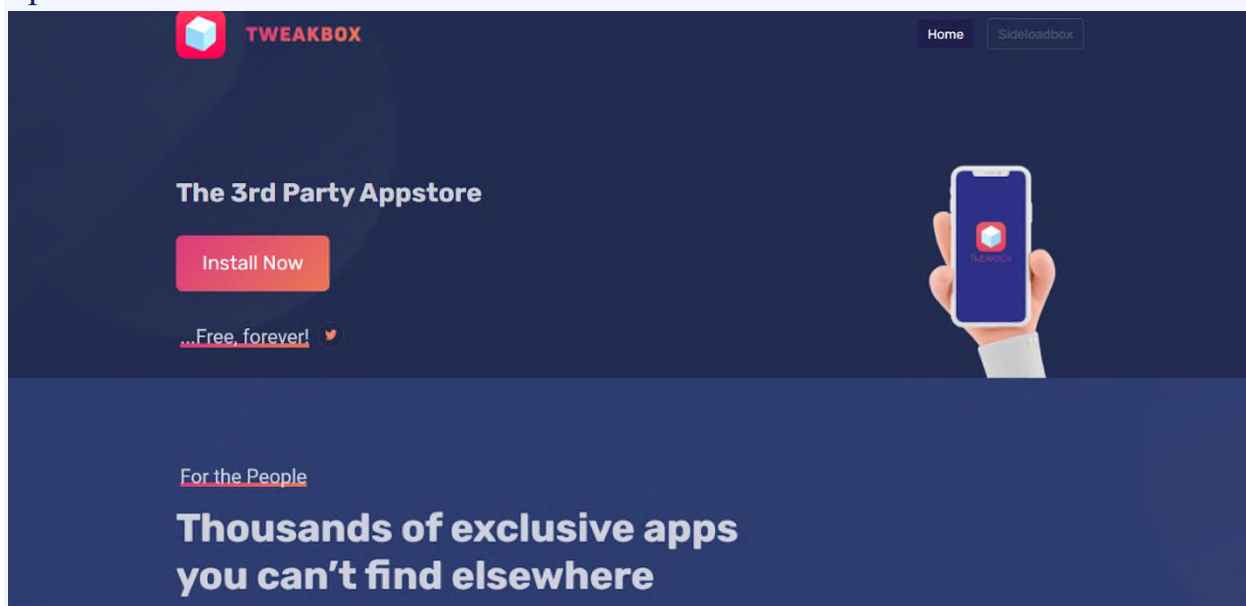
AppCake

[AppCake](#) — еще одна популярная альтернатива App Store. Вы можете легко загрузить любые неподписанные файлы IPA на свое iOS-устройство. AppCake — отличное место для тех, у кого есть джейлбрейк, так как это достойная альтернатива App Store и имеет пользовательский интерфейс, похожий на App Store. Его можно установить на устройства с iOS 9 по iOS 13. Кроме того, AppCake предоставляет приложения и игры, которые можно запускать на устройстве и без взлома.



TweakBox

TweakBox - магазин предлагает модифицированные версии приложений в обход App Store. В нём есть Facebook, Snapchat, YouTube, Instagram и прочие продукты с дополненной или измененной функциональностью. Все приложения абсолютно бесплатны.

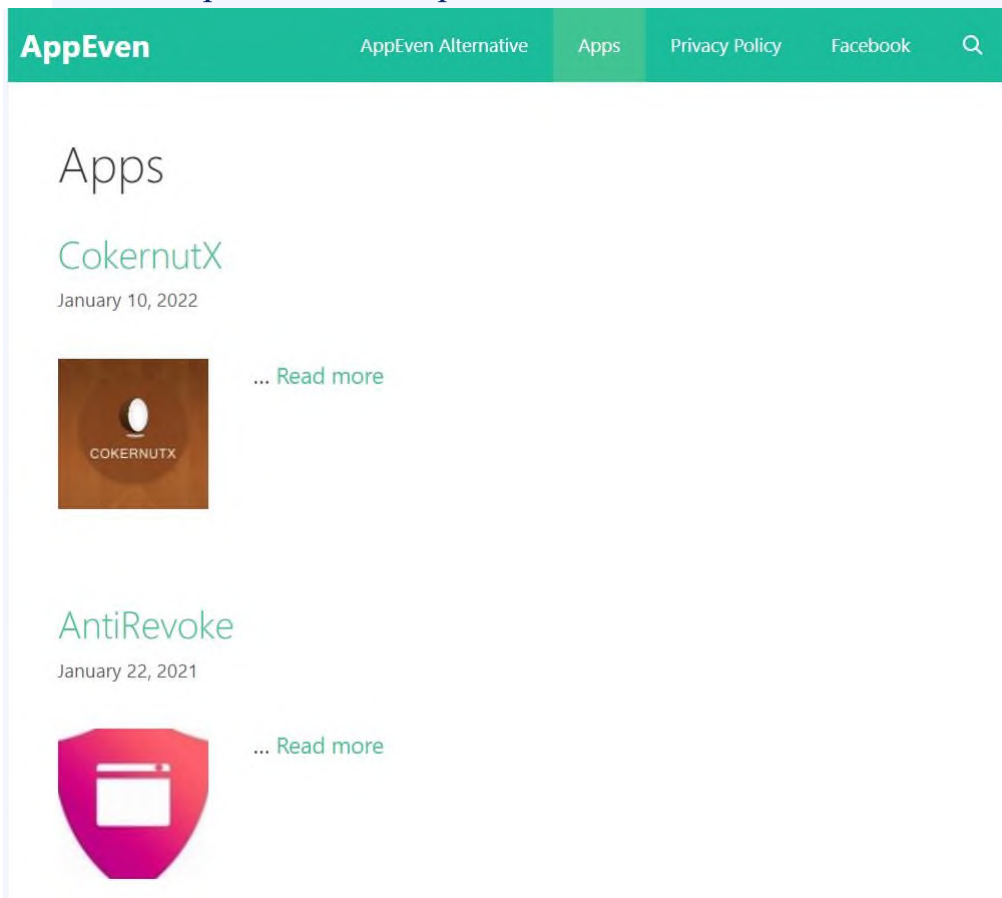


AppEven

Альтернатива для сторонних приложений на базе iOS. **AppEven** - магазин приложений, который помогает пользователям загружать измененные приложения и модифицированные игры, недоступные в App Store. Основным плюсом этой площадки является то, что для ее работы нет необходимости джейлбрейка на iPhone, кроме этого:

- Не требуется Apple ID

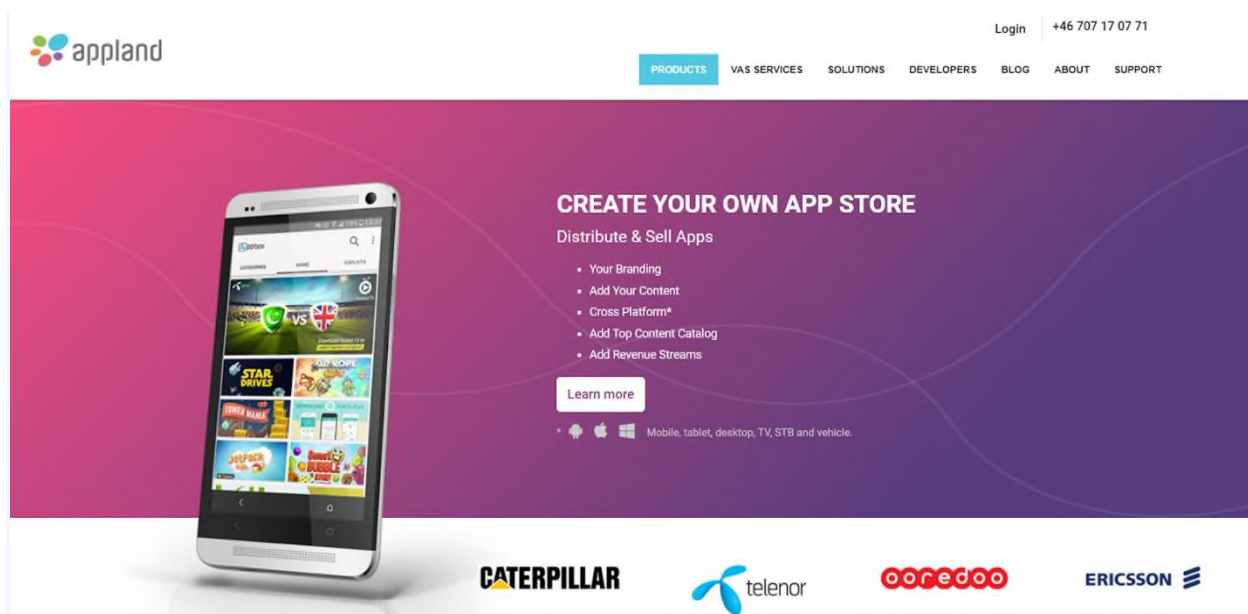
- Возможность установить платные приложения бесплатно.
- Просто использовать.
- Легко устанавливать.
- Поддерживает все версии iOS.



4. УНИВЕРСАЛЬНЫЕ ПЛОЩАДКИ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ ДЛЯ IOS, ANDROID И НЕ ТОЛЬКО

Appland

Appland — это альтернатива магазину приложений для iOS. Компания была основана в 2011 году с основной целью - создать платформу магазина приложений для любого устройства.



Как видите, это универсальная площадка для приложений для устройств на базе Android, iOS, Windows и прочие.

Разработчикам предлагается широкий выбор планов по монетизации приложений, готовых материалов и контента, статистику и аналитику поведения пользователей, обучение и многое другое.



TutuApp

Предлагает три вида доступа:

- iOS VIP
- iOS Free
- Android Free


Платформа **TutuApp** для iOS позволяет загружать неограниченное количество настроек, приложений ++, модов и взломанных игр без джейлбрейка.

TutuApp может похвастаться более чем 300 миллионами пользователей и поддержкой нескольких языков. Альтернатива магазина приложений iOS и Android включает в себя согласованный SDK, простой процесс отправки приложения и поддержку различных способов оплаты.


TutuApp VIP


TutuApp Store

TutuApp is the top un-official app store offering 3rd-party apps and games on both iPhone and Android devices.




Press the download button below to get TutuApp on your phone.

Download for iPhone


Download for Android

TutuApp Features:




AppValley

[AppValley](#) присоединяется к длинному и растущему списку сторонних площадок для установки приложений. Благодаря множеству приложений для iOS, модифицированных и измененных приложений и игр, настроек Cydia и многому другому, AppValley поддерживает как Android, так и iOS. Его можно использовать совершенно бесплатно. AppValley предоставляет своим пользователям один из самых больших наборов бесплатного контента, а также модифицированный и неофициальный контент для устройств iOS, таких как iPhone и iPad.


AppValley


AppValley Store

Приложение AppValley предлагает самый большой выбор неофициальных приложений и настроек для iOS.




Вы можете скачать приложения AppValley без предварительного джейлбрейка вашего телефона.

Скачать для iPhone

Скачать для Android

Особенности приложения AppValley:



AppValley — один из лучших установщиков iOS-приложений, предлагающий большой выбор контента для любых возрастов и предпочтений.

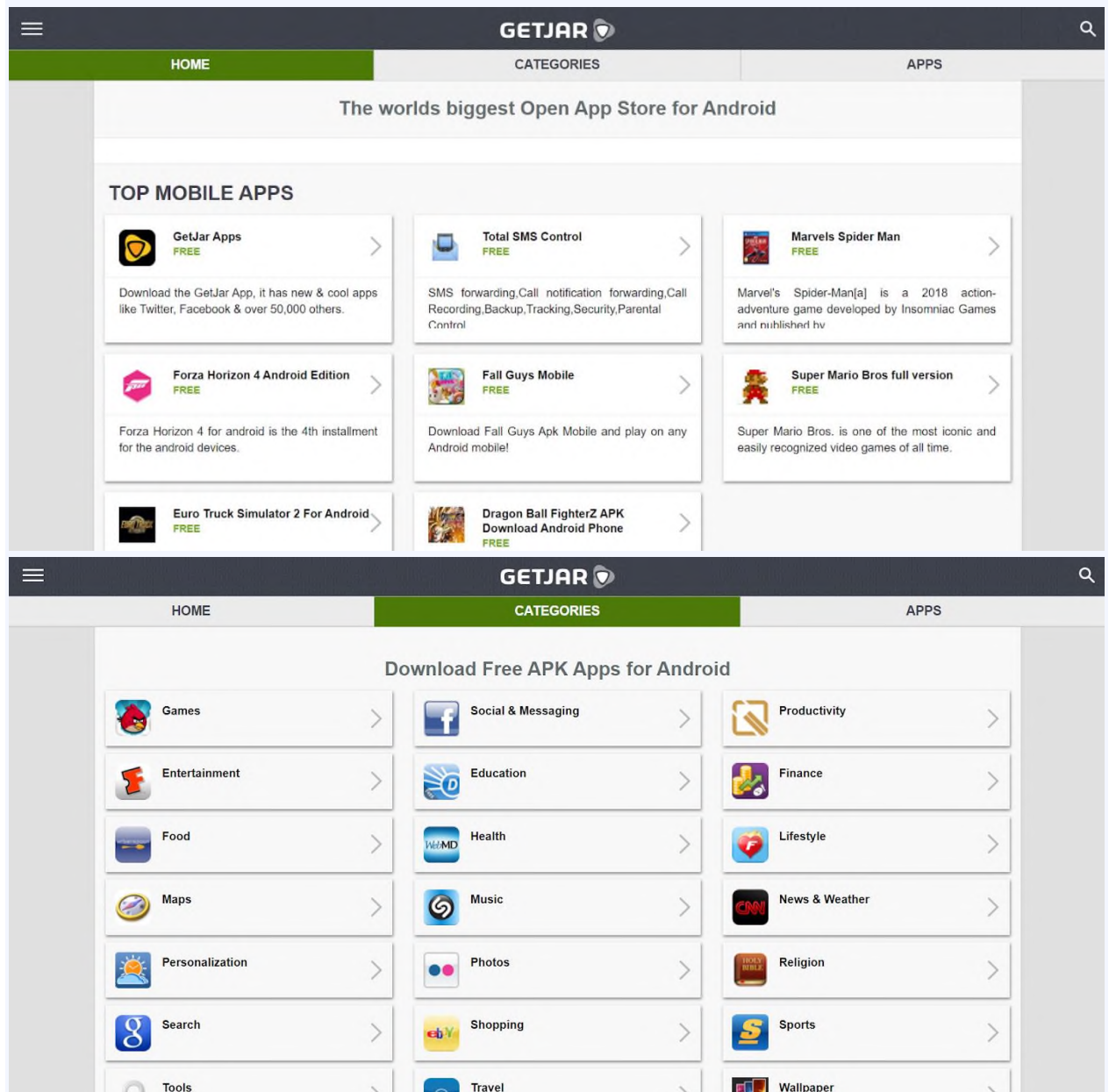
- **Приложения iOS** – Приложения и настройки для iPhone и iPad.
- **Эксклюзивные приложения** – Приложения и настройки Cydia, включая экранные рекордеры, игровые эмуляторы, файловые менеджеры и многое другое.
- **Неофициальные приложения** – Приложения и игры, которых вы не найдете в официальном магазине Apple appstore.

GetJar

[GetJar](#) — независимый магазин приложений для мобильных телефонов, основанный в Литве в 2004 году.

GetJar была запущена разработчиками для разработчиков в 2004 году, как платформа для бета-тестирования приложений. В последствии, она стала первым и крупнейшим бесплатным открытым рынком приложений в начале 2005 года. Таким образом, можно сказать, что GetJar является пионером в распространении мобильных приложений.

GetJar поддерживает самый широкий спектр различных платформ (Android, Java, Symbian, Blackberry, iOS) и многих производителей - Samsung, HTC, Huawei, Nexus.

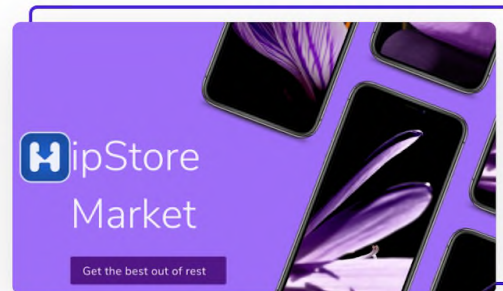


HipStore

HipStore впервые был запущен пару лет назад и с тех пор занимает лидирующие позиции на рынке альтернатив App Store. HipStore предлагает широкие коллекции приложений для iOS. Особенность - бесплатная загрузка приложений в данный магазин, наличие внутренней валюты (gems) для покупок, отслеживание взаимодействия пользователей и приложений.

Download HipStore iOS and Apk

Discover how you can download the premium apps and games using Hipstore. Hipstore is now available for iOS, Android and Pc. You can download HipStore by using the methods below.

[GET TO HIPSTORE](#)

В последствии развития этой платформы появилась возможность устанавливать приложения и для Android.

Opera Mobile Store

Кроссплатформенный магазин мобильных приложений Vemobi, который напрямую связан с Яндексом, поэтому имеет самое широкое представление на российском рынке. Именно поэтому представить свое приложение такому большому сегменту пользователей, мечта любого разработчика. Но **Opera Mobile Store** берет 30% от любых продаж, поэтому стоит подумать - широкая аудитория, возможности для установок и комиссия магазина с другой стороны.

Еще один из списка мультиплатформенный магазин приложений, особенностью которого является работа с аффилированными лицами для привлечения трафика с помощью CPI и рекламных методов. Каждое мобильное приложение, упомянутое в этой платформе содержит такую информацию, как рейтинги, скриншоты, описания приложений и обзоры. Данный магазин содержит в себе как платные, так и бесплатные приложения.



Конечно этот список был бы неполным, без огромного и особенного рынка приложений - рынка Китая. Huawei, Xiaomi, Baidu и многие другие площадки приложений мы рассмотрим подробнее в **следующей статье нашего блога**, не пропустите.

ТАК ЗАЧЕМ ИСПОЛЬЗОВАТЬ АЛЬТЕРНАТИВНЫЕ МАГАЗИНЫ ПРИЛОЖЕНИЙ?

Конечно есть свои нюансы и подводные камни, но большинство представленных площадок не преследуют цель - составить конкуренцию Google Play или App Store. Скорее всего, можно сказать что они следуют разными подходам (как для пользователя, так и для разработчика) и ориентированы на различные ниши.

Признаем честно, альтернативные магазины приложений предлагают оплату за загрузки и часто могут оперировать более выгодными предложениями, чем App Store и Google Play.

Альтернативы - реально и выгодно? Что ж, на этот вопрос нет единственно правильного ответа, и мы рекомендуем учитывать все за и против - не избегать возможностей, но помнить про безопасность.

Чего стоит опасаться - программного обеспечения, которое может навредить вашему устройству. Протоколы безопасности очень сильно разнятся от одной площадки к другой и стоит обращать внимание на пользовательский опыт. Некоторые магазины могут быть местом размещения нелегальных версий программ, игр или приложений.

4 лекция.

Тема: Принципы разработки мобильного приложения, изучение требований и формирование технического задания, выбор платформы

План:

- 1. Основные стадии разработки мобильного приложения**
- 2. Техническое задание в процессе создания мобильного приложения**
- 3. Разработка**

Каждый год популярность мобильных приложений только растет — с помощью нескольких нажатий пальцем мы заказываем продукты и одежду, покупаем авиабилеты и записываемся на прием к врачу. Если вы уже хотите [разработать приложение](#), сейчас самое время для его создания — удобный, быстрый и бесконтактный сервис в следующие несколько лет будет как никогда кстати.

В статье рассказываем про основные этапы и принципы разработки приложений — от аналитики и тестирования до выхода на рынок.



1. Основные стадии разработки мобильного приложения

Любой проект важно начинать с детального планирования, изучения собственного бизнеса, аудитории и конкурентов. Чем качественнее будут исследования, тем меньше проблем и доработок будет в дальнейшем.

Аналитика

Во время исследования определите цели бизнеса, изучите аудиторию и каналы коммуникации, проанализируйте конкурентов — это поможет определиться с правильным позиционированием. Аналитика обычно включает в себя интервью с руководителями и клиентами, фокус-группы и экспертную оценку.

Такая подготовка поможет собрать все требования и упаковать их в понятные визуальные модели: схемы бизнес-процессов, майнд мэп, пути пользователей, чтобы определить основы для разработки и перейти к прототипу.

Варианты монетизации

Приложение — еще один способ увеличить прибыль, так что еще на берегу продумайте схему монетизации, чтобы учесть ее при создании интерфейса.

Определиться с монетизацией помогут наводящие вопросы:

- Какую проблему решает сервис?
- За какие возможности люди готовы будут заплатить?
- Сколько у вас есть времени для монетизации? Можно ли подождать, чтобы набрать базу клиентов?

Какие способы монетизации бывают:

- **Реклама.** Аудитория бесплатно использует продукт, но взамен вы показываете им объявления от рекламодателей.
- **Платные функции.** Помимо основных функций приложения, реализованы дополнительные возможности, которые доступны только после оплаты. Например, доступ к новой локации в игре или скрыть анкету и рекламу в дейтинг-сервисе.
- **Платное скачивание.** В этом случае оплата поступает при первом контакте, но для начала создается высокий спрос: ваше предложение должно быть уникальным и полезным. Такой вариант часто используют создатели корпоративных платформ и программ для обработки фото и видео.

- **Покупки.** Люди платят за реальные и виртуальные товары — одежду и технику для себя или игровых персонажей, валюту внутри игры или другие возможности.
- **Подписка.** После знакомства с бесплатным контентом возможна подписка на дополнительный, чтобы получать больше информации. Такой подход используют платформы с подкастами, медиа и узкопрофильными материалами.
- **Вознаграждение за действия.** Человек получает бонусы за определенную активность: просмотр видео, следование правилам или прохождение опроса. В итоге выигрывают все стороны: рекламодатель рассказывает о своем бренде, разработчики получают премию, пользователи — вознаграждения.

Продвижение

Без маркетинговой стратегии даже самый перспективный проект канет в пучину небытия. Позаботьтесь о продвижении в начале, чтобы к моменту реализации приложение уже было в виш-листе покупателей.

Рекламироваться можно с помощью таргетированной рекламы, нативных материалов в СМИ и блогах, партнерских программ и виральных техник. Распишите портрет своей аудитории, изучите способы продвижения конкурентов, составьте собственную стратегию и меняйте ее в зависимости от обстоятельств.

2. Техническое задание в процессе создания мобильного приложения

После аналитики и проработки стратегии развития наступает процесс разработки [приложения для Android и iOS](#), на первоначальном этапе которого изучается техническая документация и готовится техническое задание.

Обычно в нем прописываются:

- Цели проекта.
- Пользовательские истории и карта путешествия человека — описывают, какие задачи будут решать люди с помощью сервиса, и как они будут это делать.
- Обязательные функции.

- Технические требования к интерфейсу, производительности, роли пользователей, безопасности.
- Реализация функциональности: UX и UI дизайн.
- Этапы разработки.
- Время, необходимое, для всех работ.
- Бюджет.

Описание требований к интерфейсу помогает дизайнерам и разработчикам понять, что именно хочет клиент и как это можно выполнить. Чем подробнее будет ТЗ, тем выше шанс получить то, что действительно нужно и избежать бесконечных правок.

Чаще всего студии разработки помогают с подготовкой ТЗ. Например, в AppCraft мы всегда проверяем ТЗ на соответствие требованиям платформ и разрабатываем его с нуля, если у вас не хватает на него времени или возникли какие-то сложности.

Организация команды

Обычно в состав выделенной проектной команды входят: тестировщик, UX/UI дизайнер, мобильные разработчики, — количество зависит от масштаба проекта — и проектный менеджер, который организывает работу команды.

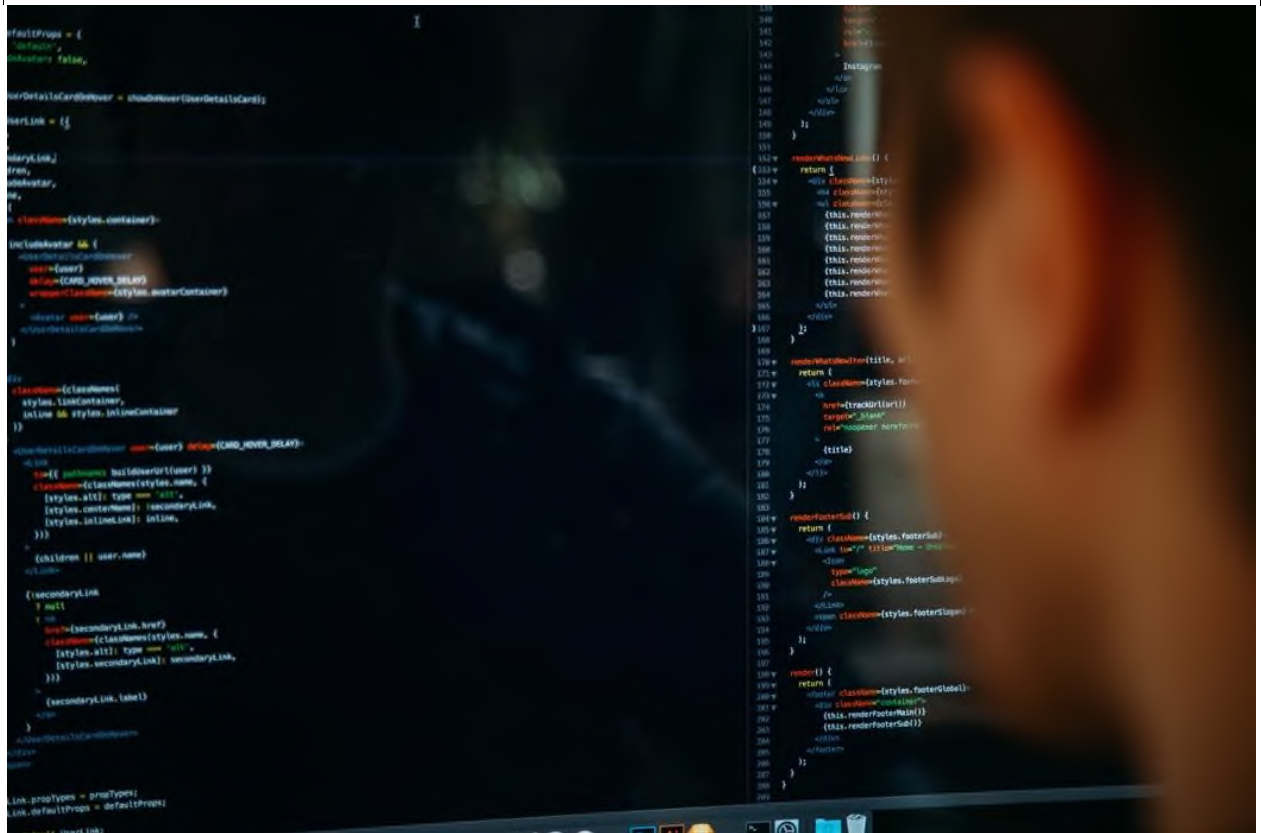
Мы в AppCraft никогда не привлекаем для работы внешних специалистов, потому что выбираем работать с проверенными временем людьми и плотно общаться внутри команды. У такого подхода есть большой плюс — каждый сотрудник сфокусирован на конечном продукте и заинтересован сделать свою работу качественно.

Создание дизайна и прототипа

На этом этапе UX/UI дизайнер выстраивает логику взаимодействия между страницами экранов регистрации и авторизации, заполнения данных, личного кабинета, корзины, оплаты покупки и отслеживания заказа. Разрабатывает внешний вид будущего сервиса в соответствии с техзаданием и фирменным стилем: подбирает цветовое решение, шрифты, отрисовывает иконки, кнопки, пуш-уведомления, слайдеры и т.д.

После согласования дизайна, дизайнер готовит прототип (если это не было сделано на этапе подготовки ТЗ) — в нем воспроизводится базовая логика, структура и функционал.

Обычно прототип создается в виде экранов на каждом этапе пользовательского пути. Это еще не готовый продукт, но он помогает выстроить фундамент и протестировать его функциональность, чтобы уже на начальном этапе исправить ошибки и улучшить пользовательский опыт.



3. Разработка

Одна из трудозатратных стадий включает написание кода, проработку архитектуры и делится на Back-end и Front-end разработку. Мобильные разработчики должны знать концепцию проекта, его уникальность и включаться во все процессы, чтобы оценить жизнеспособность идеи и реализовать желания заказчика.

На этом этапе Front-end программисты разрабатывают продуманный и протестированный клиентский интерфейс и логику платформы.

Back-end разработчики создают сервер для хранения и обмена информации. Специалисты выбирают язык программирования для написания кода и хостинг для сервера и API, выстраивают систему управления базой данных. Чем лучше выбраны параметры, тем быстрее будет работать приложение.

Разработка может быть реализована несколькими способами:

- **Нативная.** Разрабатывается отдельное приложение для каждой мобильной платформы. Этот способ самый дорогой, но надежный: вы получите полную поддержку от сторов, а интерфейс будет работать быстро и выглядеть максимально органично.
- **Кроссплатформенная.** Разработчики используют универсальный код под все платформы, но операционная система все равно запускает его как нативное. Самый оптимальный вариант в плане «цена-качество».

Подробнее о плюсах и минусах нативной и кроссплатформенной разработки писали в [этой статье](#).

Тестирование

Некоторые компании выделяют тестирование в отдельный этап и досконально проверяют приложение только перед релизом.

Мы думаем, что тестирование приложения нужно проводить на каждом этапе разработки — по готовности каждой части функционала. Лучше потратить больше времени на исправление багов до релиза, чем каждый час получать негативные отзывы на странице после публикации в сторе. Поэтому каждую страницу тестируем настолько часто, насколько это возможно.

Публикация

Перед запуском важно внимательно изучить правила Google Play Store и Apple App Store и подготовить скриншоты страниц, маркетинговый план и описание. После загрузки сторы проверяют всю информацию, актуальность проекта и дают заключение: будут они публиковать приложение или нет. Если все прошло удачно, его можно будет скачивать через несколько дней.

С публикацией могут возникать трудности, поэтому действительно важно ознакомиться со всеми правилами магазинов. В AppCraft проектные менеджеры не оставляют клиентов со всем этим наедине: помогают с публикацией приложения и консультируют по всем вопросам, связанным с регистрацией аккаунтов в магазинах, требованиями к материалам и их форматам.

Доработка и техподдержка

После запуска вы сможете анализировать, какие разделы самые популярные, а какие не очень, сколько человек завершили целевые действия, а какие

страницы стоит доработать. Внимательно изучайте и обрабатывайте все входящие данные: они помогут дорабатывать приложение и убирать ненужные функции. Процесс аналитики практически бесконечен, так что вам понадобится техническая поддержка, которая будет фиксировать и оперативно решать текущие проблемы, оптимизировать приложение и дорабатывать его.

В Appcraft гарантийная поддержка кода — 12 месяцев. Мы полностью передаем заказчику права на приложение, но продолжаем мониторить системную аналитику и оперативно устраняем неполадки в приложении, если они вдруг возникают.

Описанные этапы — классический вариант процесса разработки, но мы всегда обсуждаем этот процесс отдельно с каждым новым клиентом. Потому что для нас важно синхронизироваться с заказчиком и сделать так, чтобы процесс разработки был удобным и понятным.

В AppCraft мы занимаемся всеми этапами разработки от аналитики (базовой первичной аналитики или глубоких исследований) до релиза и обеспечиваем оперативную техподдержку. За 10 лет мы создали несколько собственных проектов и больше 200 мобильных приложений для клиентов — мессенджеры, корпоративные решения, банковские системы, e-commerce и соцсети.

5 лекция.

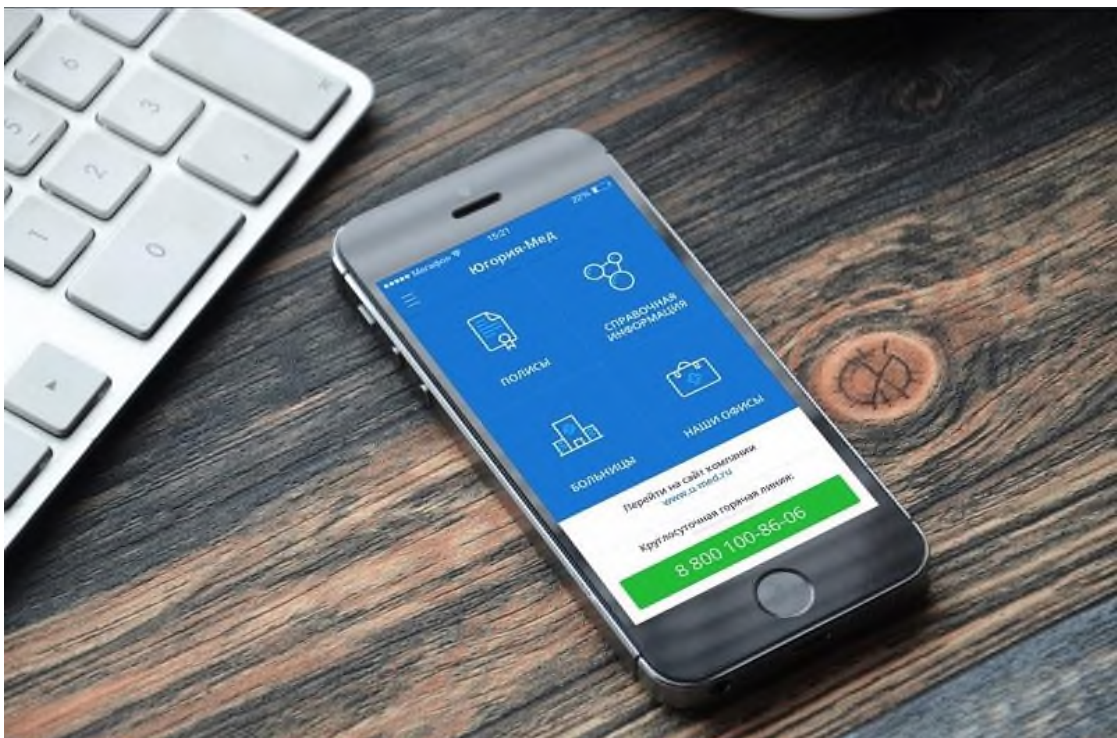
Тема: Разработка дизайна мобильных приложений, UI, UX дизайн

План:.

- 1. Особенности дизайна мобильных приложений и его отличия от десктопных ресурсов.**
- 2. Этапы разработки mobile design**
- 3. Создание User Flow Diagram: пример блок-схемы**

1. Особенности дизайна мобильных приложений и его отличия от десктопных ресурсов

Проектируя интерфейс приложения для мобильных устройств, нужно знать об особенностях дизайна, и дело здесь не только в разрешении экрана. То, что хорошо смотрится с ноутбука или стационарного (десктопного) компьютера, может совершенно не подходить для мобильного устройства, и наоборот.



О чем же следует помнить при разработке интерфейса мобильного приложения?

1. **Размер экрана.** Очевидно, что экран телефона меньше экрана компьютера или ноутбука. На дисплее мобильного устройства может поместиться

гораздо меньше элементов, поэтому при разработке дизайна важно продумать, что это будет, а также упростить навигацию для пользователя.

2. Кликабельность. Редко кто сегодня использует стилусы. Поэтому все кликабельные элементы должны быть такого размера, чтобы пользователь мог легко нажать на каждый из них пальцем. Если для этого ему придется увеличить страницу или неоднократно пытаться попасть по кнопке, вряд ли он останется доволен приложением.
3. Трафик и производительность. Все знают так называемые тяжелые сайты, которые очень неудобно смотреть с телефона — страницы долго грузятся, возникают ошибки. Все, что проектируется для мобильного устройства, должно быть легким — и приложения в том числе. Во-первых, тяжелые файлы много весят, и это может оказаться дорого для аудитории. Во-вторых, все та же скорость работы, которая очень важна. Также стоит следить, чтобы количество обращений к API не снижало общую производительность сервера.
4. Ориентация страницы. Большую часть времени (около 94%) пользователь держит устройство вертикально. И тем не менее, важно продумать, как будет выглядеть интерфейс при повороте телефона в горизонтальную позицию — это не должно повлиять на удобство.
5. Управление жестами. То есть можно отключить стандартные кнопки типа «вперед», «назад» и т. д., и все эти команды будут выполняться с помощью определенных жестов. Это отличительная особенность всех современных мобильных устройств, и ее нужно использовать при разработке мобильного приложения.
6. Камера и микрофон. Ими оснащены все гаджеты. Чаще всего их используют для обеспечения безопасности устройства: помимо стандартного ввода пароля, можно сделать распознавание лица или голоса. Стоит подумать и о других вариантах применительно к конкретному мобильному приложению.

2. Этапы разработки mobile design

Существуют два основных этапа работы над дизайном:

1. UX, или User experience — это разработка алгоритма, понимание того, как пользователь будет взаимодействовать с приложением. Иными словами, это некий каркас, архитектура ресурса.
2. UI, или User Interface Design. UI дизайн определяет внешний вид, удобство и эстетику интерфейса.

Если говорить простым языком, то UX дизайн отвечает за то, как система будет работать, а UI — за то, как все это будет выглядеть. Оба этапа неразрывно связаны между собой, и очень часто всю работу выполняет один человек, особенно в небольших дизайнерских студиях.

В зависимости от особенностей ресурса может потребоваться выполнение каких-то отдельных специфических этапов, но основные шаги при разработке интерфейса любого мобильного приложения всегда одинаковы. Именно поэтапная работа позволяет сэкономить время и ресурсы, а также избежать неожиданных замечаний от клиента.

Создание концепции

Это первый этап разработки после того, как идея самого приложения уже есть. Так как интерфейс разрабатывается под определенную аудиторию, ее нужно изучить и ответить на главный вопрос — чего ждут от нового продукта? Для этого нужно провести исследования. Они бывают:

- качественные (интервью, фокус-группы, экспертное мнение) — отвечают на вопросы «как» и «почему»;
- количественные (анкетирование, опрос, телефонное интервью) — отвечают на вопросы «сколько» и «как часто».

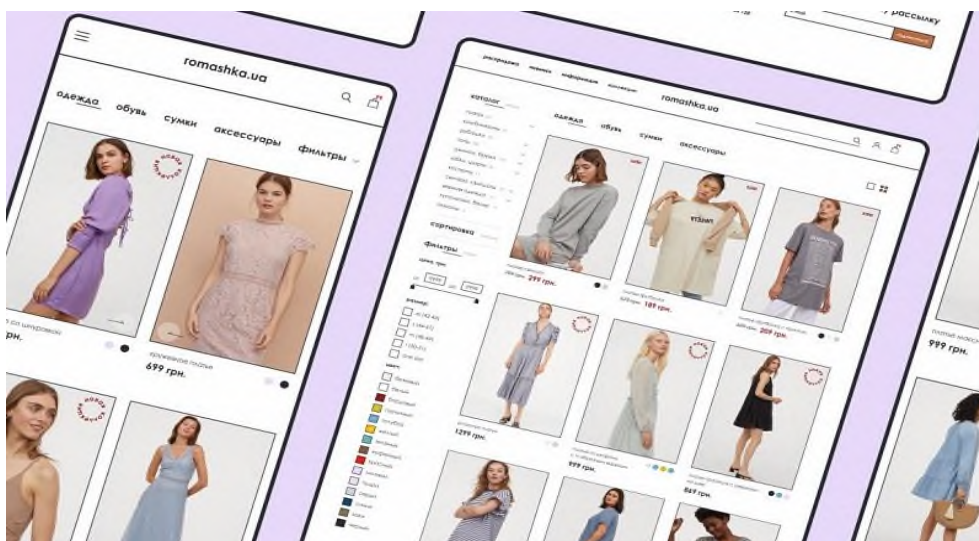
Все эти исследования помогут составить портрет потенциального потребителя. И уже опираясь на это, можно заняться разработкой функционала приложения.

Важно! В случаях, когда разработка мобильного приложения начинается с нуля, первоначально проводят качественные исследования, чтобы выявить потребности аудитории, а потом количественные. Если же приложению нужна просто доработка, то действовать нужно наоборот.

Далее нужно изучить конкурентов, а также отзывы потребителей об их продукте — это поможет избежать ошибок в разработке. После этого можно перейти к составлению User story map — функционалу будущего прототипа. Дизайнер на основе полученных данных ставит цели, которые преследует пользователь в приложении, а затем указывает действия для их достижения. На этом этапе решается вопрос о том, как будет выглядеть интерфейс приложения: что будет на основном экране, размер кнопок и т. д.



К примеру, если речь идет о разработке приложения для вызова такси, уместно будет разместить минимум иконок и хорошую карту. Пользователь должен просто открыть приложение, нажать пару кнопок и заказать машину — лишняя навигация в прямом смысле будет лишней.



А вот интерфейс приложения интернет-магазина совсем иной. Он, наоборот, может пестрить иконками, которые будут меньше по размеру, но при этом удобны в плане кликабельности.

Мозговой штурм

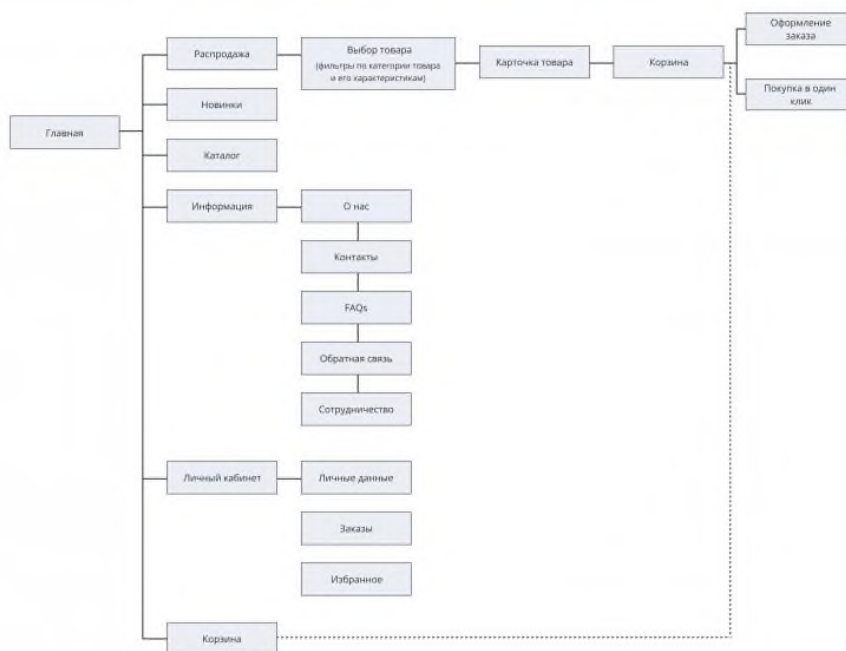
После разработки концепции стоит приступить к методике мозгового штурма. Для этого все члены команды должны выдвигать свои идеи. Если дизайнер один, то он делает это самостоятельно. Главное правило — предложений должно быть много, пусть даже самых необычных и невыполнимых. Как показывает практика, именно из них в итоге и выходит что-то стоящее.



Именно мозговой штурм, или брейнсторминг дает жизнь самому интерфейсу, превращая идею в реальность. Можно работать на бумаге или в специальных программах, например, Balsamiq Mockups, Sketch, Photoshop и InVision — дело вкуса и привычки.

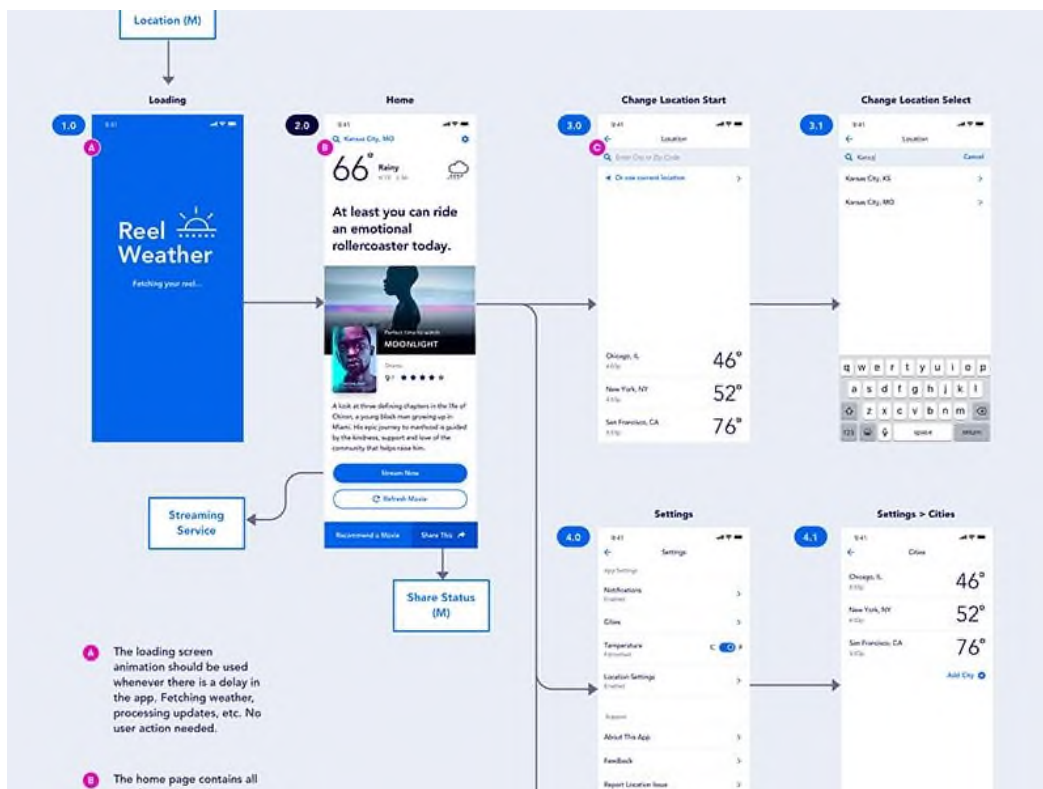
3. Создание User Flow Diagram: пример блок-схемы

Разработка интерфейса мобильного приложения — это хорошо, но без представления, как им будут пользоваться, дальше продвинуться не получится. И тут на помощь приходит создание User Flow Diagram.



Это визуальное представление действий пользователя, которые он совершает при взаимодействии с мобильным приложением. Выполняется в виде блок-схемы. User Flow Diagram иллюстрирует всю логику и возможные варианты использования.

Согласование структуры интерфейса и переходов

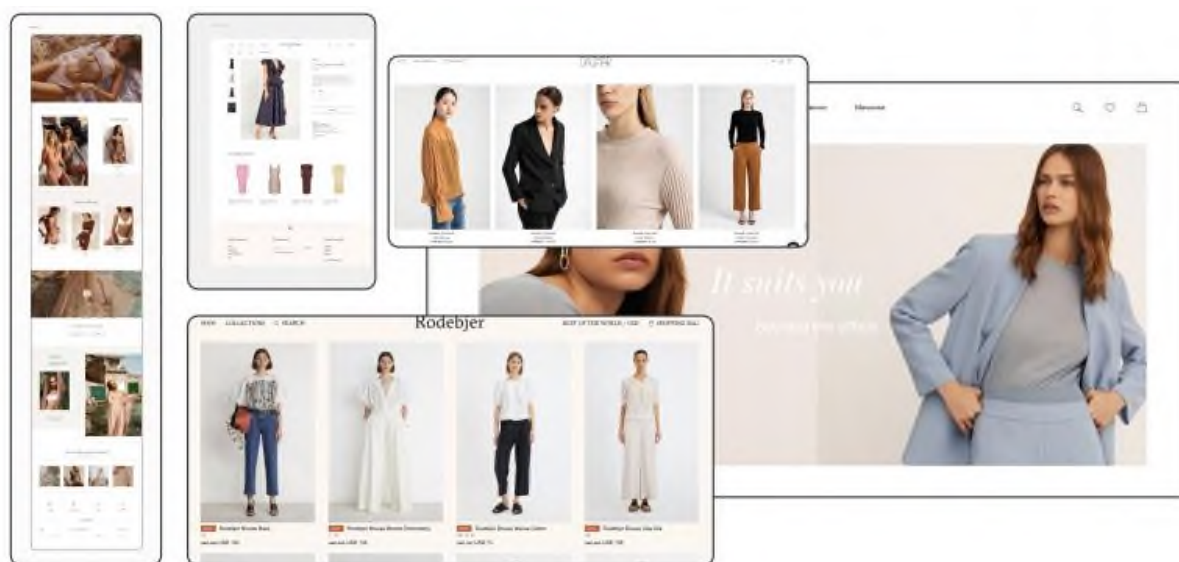


После создания структуры интерфейса и диаграммы переходов нужно согласовать это с заказчиком. Не стоит отправлять верстку на утверждение, продолжая работать дальше: если клиенту что-то не понравится, проще всего внести исправления именно сейчас.

Если продолжить разрабатывать интерфейс, а затем что-то поменять в структуре, может оказаться, что половину работы придется делать заново.

Определение внешнего вида интерфейса и утверждение стиля

Если предыдущий этап успешно пройден, можно переходить к визуальной части — выбору стиля. Лучший вариант — попросить у заказчика примеры ресурсов, интерфейс которых его устраивает, и отталкиваться от них.

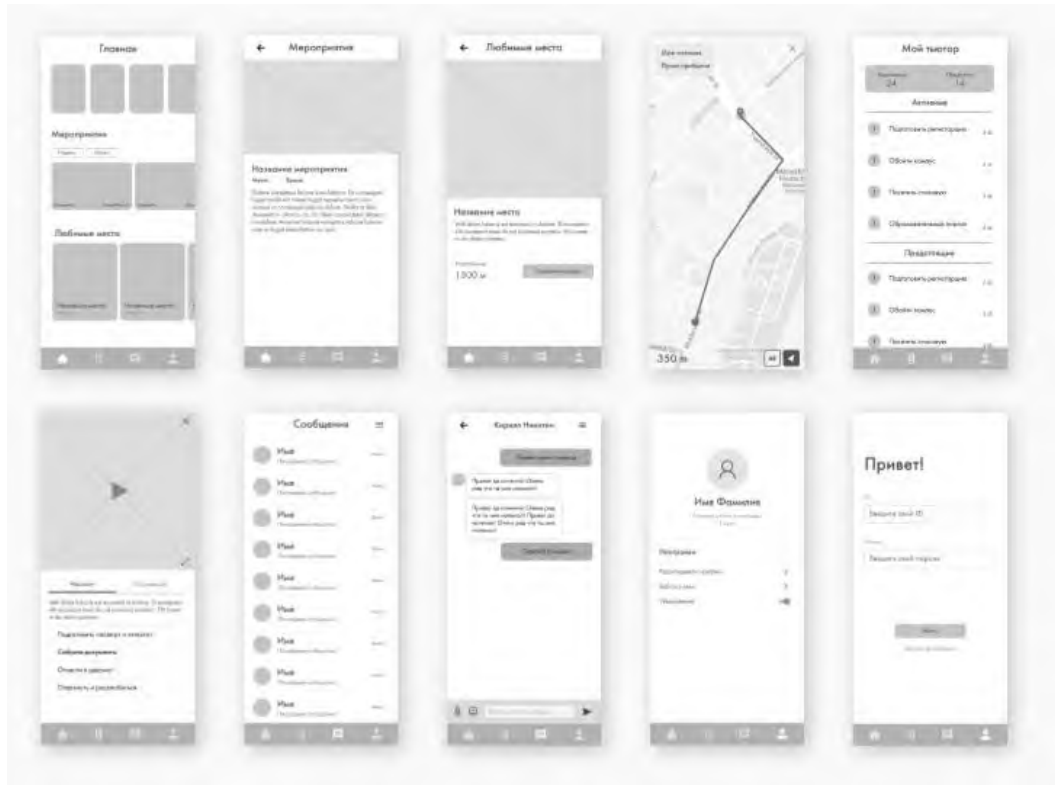


Хороший web-дизайнер должен разбираться в актуальных трендах оформления, а также понимать, что именно подойдет по тематике именно этому клиенту. На основе этих знаний, пожеланий и требований заказчика нужно разработать дизайн. На этом же этапе определяется масштабирование и общее время, которое уйдет, чтобы выполнить соответствующий дизайн.

После этого нужно рассказать заказчику о своем видении, а также объяснить, почему были приняты такие решения для разработки. Целью этого этапа является принятие того интерфейса, который устроит обе стороны.

Демонстрация проекта: макеты, прототипирование и другие варианты

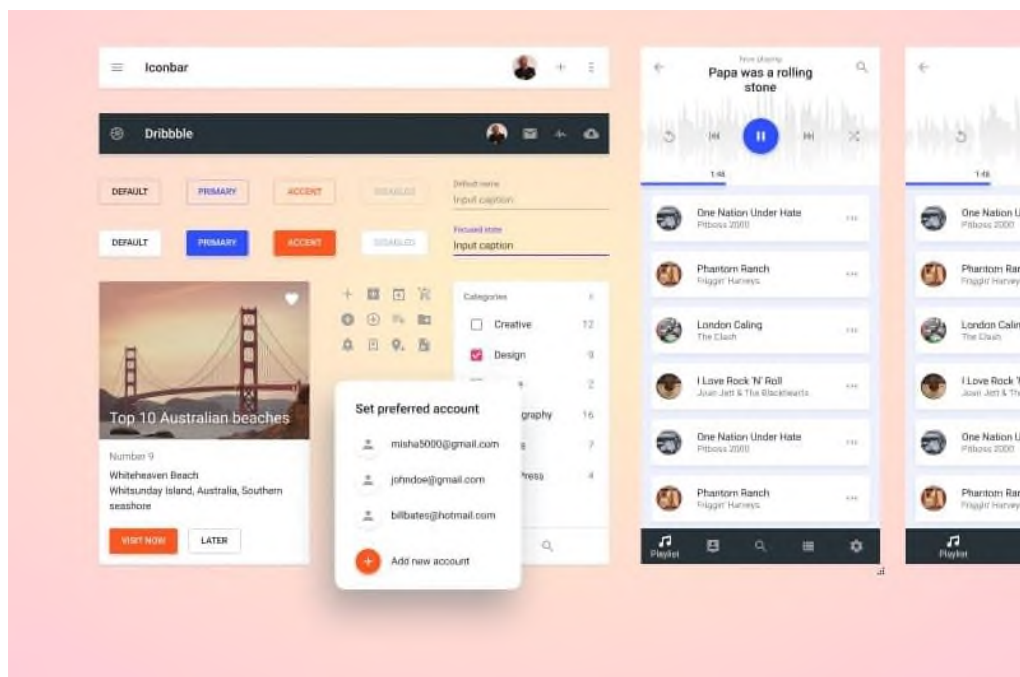
После утверждения общего стиля мобильного приложения нужно показать клиенту полную версию дизайна. Есть разные способы демонстрации продукта, поэтому можно смело выбирать наиболее удобный для себя вариант прототипирования.



- Wireframe. Это самый простой инструмент прототипирования интерфейсов. Довольно низкоуровневая демонстрация готового ресурса позволяет наглядно показать, как пользователь будет взаимодействовать с приложением, а также его структуру. По ходу работы можно добавлять необходимые комментарии.

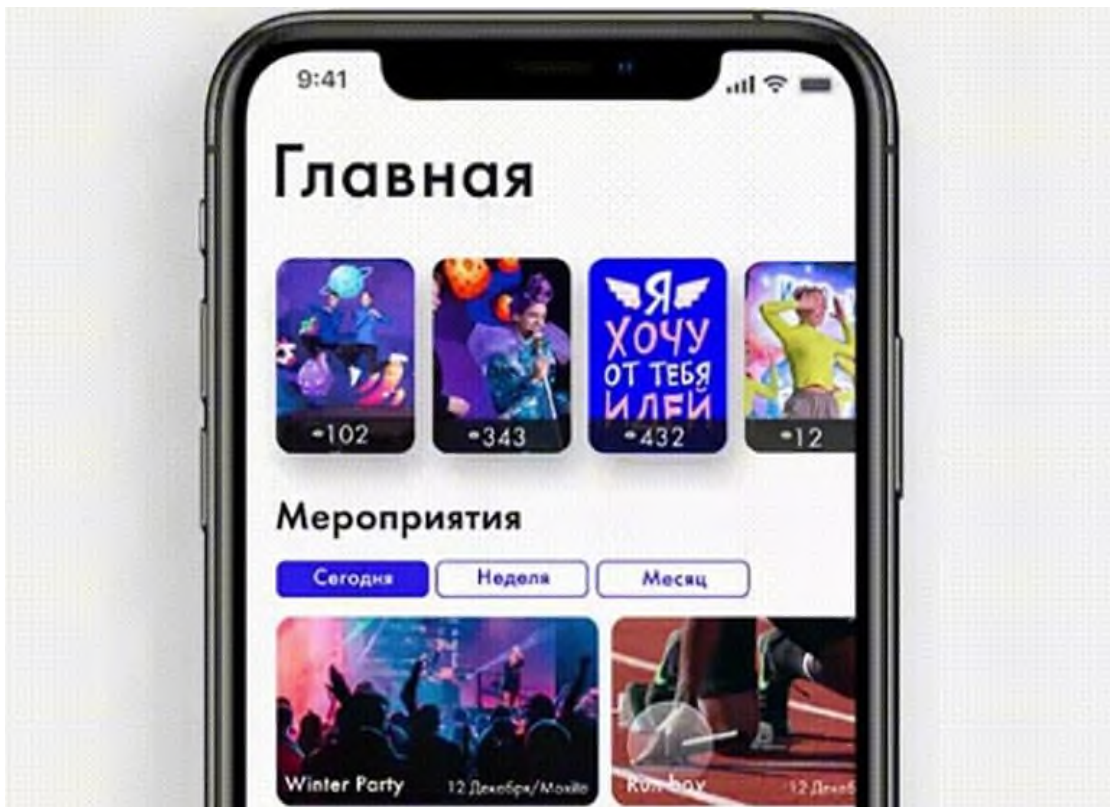


- Макет. Это более реальная демонстрация готового приложения. Картинки здесь статичны, но заказчику сразу понятно, как и что будет выглядеть. Можно создать презентацию. Ранее макеты часто рисовали с помощью Adobe Photoshop, сейчас ему на смену пришло приложение Sketch.



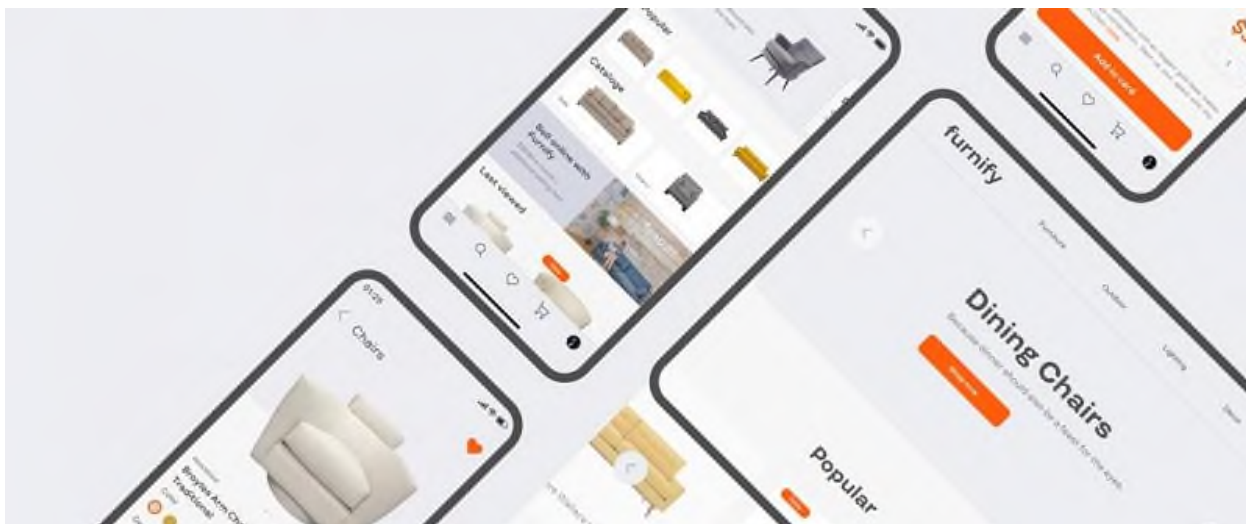
- Кликабельный прототип. Это наиболее близкий к реальности прототип мобильного приложения. Здесь можно нажимать кнопки, переходить по

ветке меню и делать почти все из того, что будет доступно реальному пользователю. Один из самых наглядных вариантов демонстрации разработки приложения, на создание которого уходит довольно много времени и ресурсов. Для работы часто используют InVision, нередко ее дополняет Craft.



- Анимированные flow. Это самый наглядный и в то же время самый сложный в исполнении вариант демонстрации процесса работы приложения. В отличие от кликабельного прототипа, здесь еще нужно записать видео взаимодействия с ресурсом. Важно понимать, что сложная анимация существенно затянуть весь процесс, что нежелательно для всех сторон. К тому же соответствующими навыками владеют далеко не все дизайны.

Утверждение дизайна приложения



После демонстрации необходимо внести финальные правки по дизайну. Заказчик уже представляет, как будет выглядеть мобильное приложение. Нередко оказывается, что тот, кто соглашался со всеми предложениями ранее, сейчас заявит, что итоговый результат он представлял иначе. И тогда велика вероятность, что дизайнеру придется очень много переделать. Но в большинстве случаев, при качественном взаимодействии дизайнера и клиента на предыдущих этапах, правки оказываются минимальны.

IOS против Android — две глобальные операционные mobile системы

На сегодняшний день 98% рынка поделили между собой IOS от Apple и Android от Google, поэтому все популярные мобильные приложения разрабатываются под одну из этих операционных систем. Для того чтобы улучшить пользовательский опыт, каждый из гигантов IT-индустрии выпустил рекомендации по дизайну, разработке и проектированию приложений:

- Google Material Design для Android;
- Human Interface Guidelines для IOS.

Их можно найти в свободном доступе в интернете. Периодически рекомендации изменяются, дополняются, поэтому стоит время от времени перечитывать их заново — это поможет быть в курсе происходящих изменений и актуальных трендов.



Слепо следовать рекомендациям не нужно. Достаточно понимать общие принципы разработки и особенностей веб-дизайна. Более того, можно даже попробовать применить какие-то моменты из рекомендаций для Android к iOS, и наоборот.

Важно! Если у дизайнера телефон Iphone, но в данный момент идет работа над приложением для андроида, стоит хотя бы на время обзавестись устройством именно на андроиде, чтобы лучше понять его интерфейс. И наоборот.

О том, как создавать приложения для разных операционных систем, мы писали в статьях [«Дизайн приложений для IOS»](#) и [«Дизайн приложений для android»](#).

Советы начинающему дизайнеру мобильных приложений

Если вы только начинаете работать над проектированием мобильных приложений, стоит заранее изучить все аспекты разработки и послушать советы бывалых дизайнеров.

Работа с заказчиком

Хороший дизайнер всегда открыт к диалогу и слушает пожелания заказчика на всех этапах разработки. Даже если идея клиента кажется не самой удачной, стоит дослушать его до конца и предложить свои варианты с учетом запросов заказчика. Мнение и представление о mobile design у клиента и дизайнера могут отличаться, и очень важно донести свои предложения аргументировано, а также понять, что именно в итоге хочет получить клиент. Стоит брать во внимание, что на сам проект влияет сразу несколько факторов:

- сроки выполнения заказа;

- бюджет, который заказчик закладывает в проектирование;
- цель дизайна — одно дело, когда нужна просто презентация для инвесторов, другое — полноценное мобильное приложение;
- технологии выполнения..

Взаимодействие с разработчиками

Сделать красивый графический дизайн и в готовом виде отдать его разработчикам — большая ошибка. Велик риск, что многое придется переделывать, но уже в другие, сжатые сроки. Хороший дизайнер с самого начала активно взаимодействует с разработчиками мобильного приложения, обсуждая свои решения и предложения.

В идеале дизайнер должен разбираться не только в своей работе, но и в вопросах разработки, а также четко представлять, насколько реализуемы его дизайнерские решения на практике. Это значит, что придется работать в команде и изучать соответствующую литературу. Если на проекте еще нет разработчиков, можно контактировать с разработчиками из других проектов на той же платформе.

Процесс проектирования мобильного приложения

6 лекция.

Тема: Технологии кроссплатформенного программирования. Дротик => Технология флаттера. Установите и настройте Android Studio.

План:

1. Платформа Flutter. Установка и настройка Android Studio.

1. Платформа Flutter.

Flutter - это платформа для разработки мобильных приложений, запущенная и открытая в Google, ориентированная на кроссплатформенность, высокую точность и производительность.

Фактически, кроссплатформенная структура приложений больше не является новой технологией. Самые распространенные из них: webApp, React Native, Weex и т. Д.

Разработка набора кода может выполняться на нескольких платформах, таких как Android и iOS, разве это не экономит много трудовых и материальных ресурсов? Выглядит это так, но сегодня приложения, разработанные на кроссплатформенных фреймворках, все еще несколько неудовлетворительны. Основные недостатки:

- Например, в пользовательском интерфейсе: веб-приложение должно полагаться на веб-просмотр, а рендеринг пользовательского интерфейса h5 далек от собственного пользовательского интерфейса; React Native необходимо преобразовать в собственные элементы управления через промежуточное программное обеспечение, но процесс промежуточного преобразования по-прежнему имеет определенную потерю производительности ; Weex похож на React Native и также требует некоторого уровня преобразования. По оценкам, эти проблемы неразличимы при рисовании простых страниц, но неудобства при рисовании с частым взаимодействием и анимацией.
- Что касается взаимодействия с собственным api: все три похожи, и всем требуется промежуточное программное обеспечение для взаимодействия с собственным api. В этом общем случае могут возникнуть проблемы совместимости.

Самый молодой Флаттер резюмировал вышеупомянутый опыт и уроки:

- В пользовательском интерфейсе: **Flutter не нужно преобразовывать в собственные элементы управления, он напрямую отрисовывает пользовательский интерфейс через собственный движок Skia.** (Skia - это движок обработки графики Google, Android использует движок Skia)

- И по коду: **Используйте язык Dart для компиляции собственного кода платформы через AOT, чтобы Flutter мог напрямую взаимодействовать с платформой без необходимости использования промежуточного программного обеспечения.**

Производительность - это только основа, действительно важна экология. В этом отношении нынешний Flutter еще немного незрел. Но потенциал все еще довольно велик из-за платформы Google. Кроме того, система Google Fuchsia амбициозна (страница Fuchsia написана на Flutter), поэтому нет ничего плохого в простом понимании и изучении Flutter.

2. Android Studio настраивает Flutter

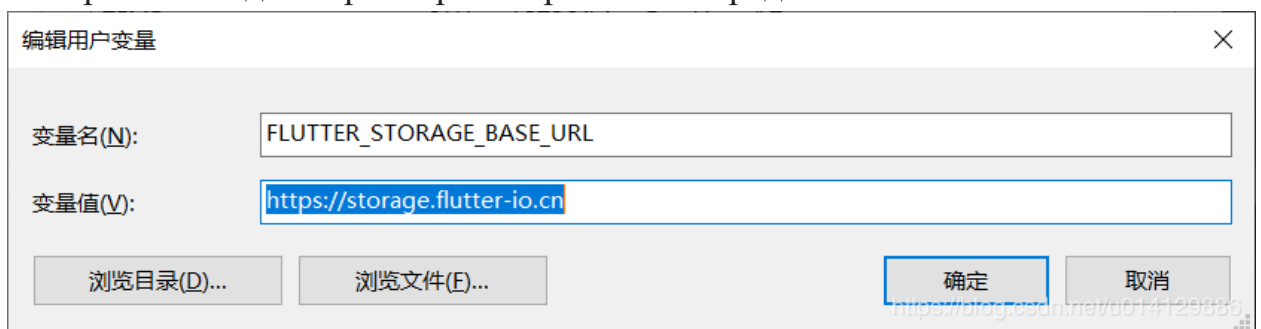
Готов к работе:

1. Android Studio 3.0 и выше (я использую 3.4)
2. git

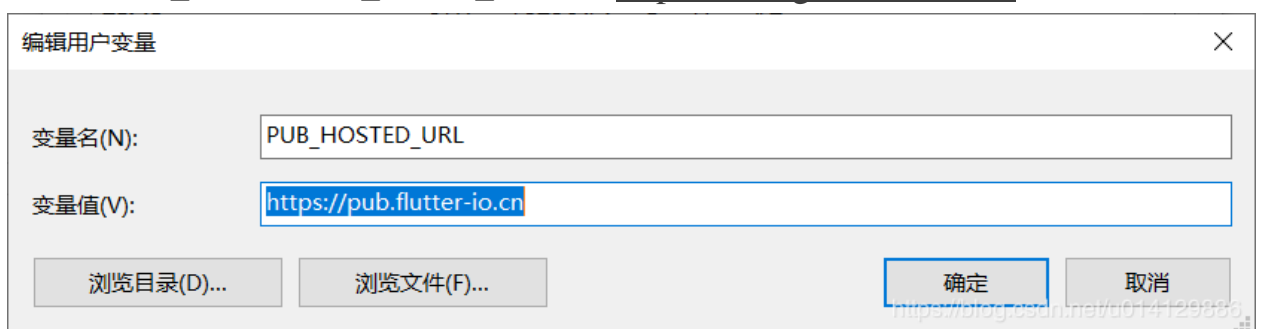
Нечего сказать, у всех есть.

2.1 Сначала настройте зеркало, иначе загрузка будет медленной.

Настройте эти два параметра в переменных среды:



FLUTTER_STORAGE_BASE_URL : <https://storage.flutter-io.cn>



PUB_HOSTED_URL : <https://pub.flutter-io.cn>

2.2 Загрузите через командную строку git, куда вы хотите скачать

Щелкните правой кнопкой мыши «Git Bash Here» в каталоге, который вы хотите загрузить, и введите «git clone <https://github.com/flutter/flutter.git>».

电脑 > 本地磁盘 (C:) > DevelopTools > Android

名称	修改日期	类型	大小
AndroidStudio	2019/4/19 11:51	文件夹	
flutter	2019/5/6 10:10	文件夹	
SDK	2019/5/4 17:28	文件夹	



<https://blog.csdn.net/u014129885>

```
MINGW64:/c/DevelopTools/Android

CTS@DESKTOP-1T4C0V7 MINGW64 /c/DevelopTools/Android
$

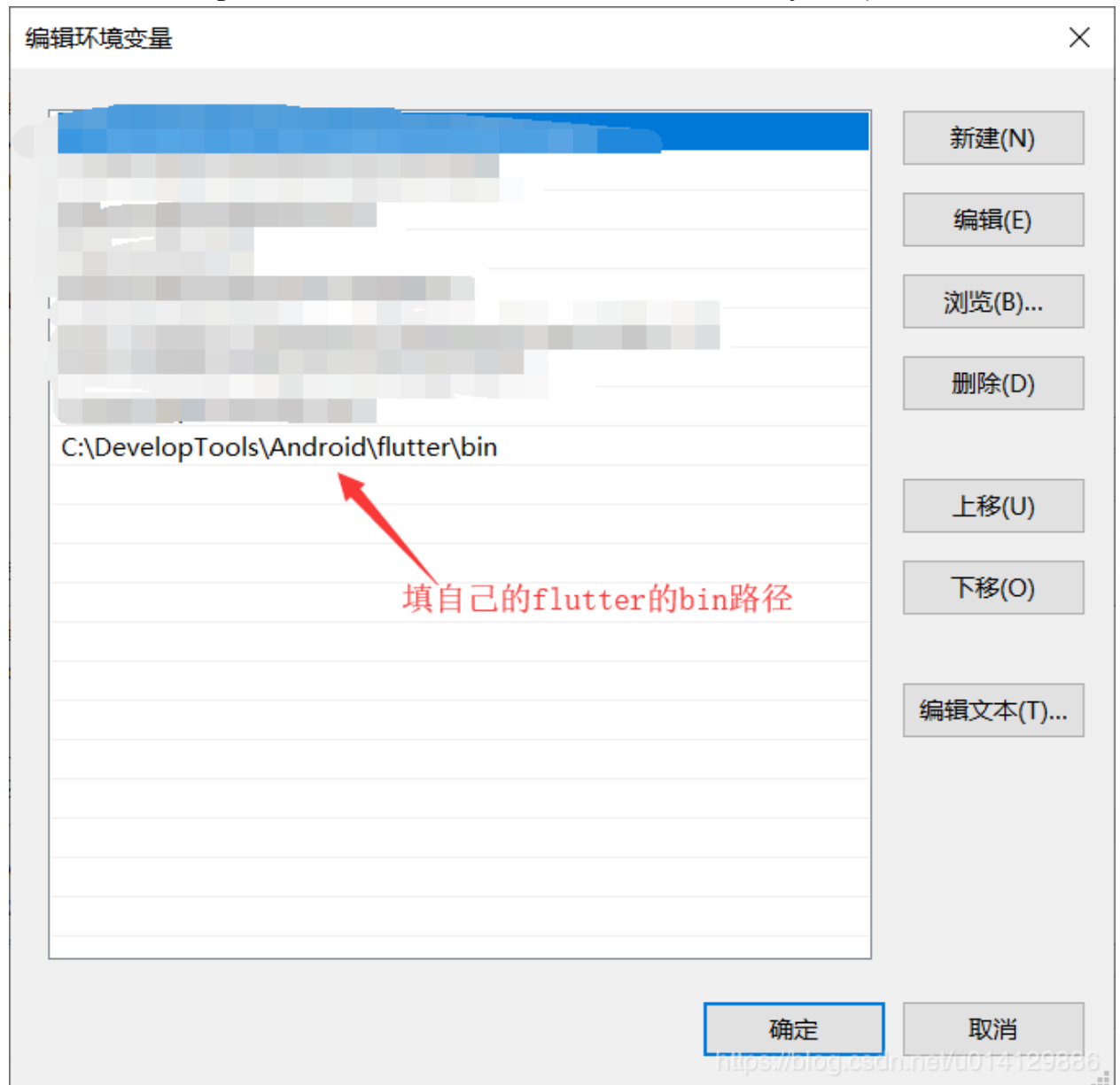
CTS@DESKTOP-1T4C0V7 MINGW64 /c/DevelopTools/Android
$ git clone https://github.com/flutter/flutter.git
```

<https://blog.csdn.net/u014129885>

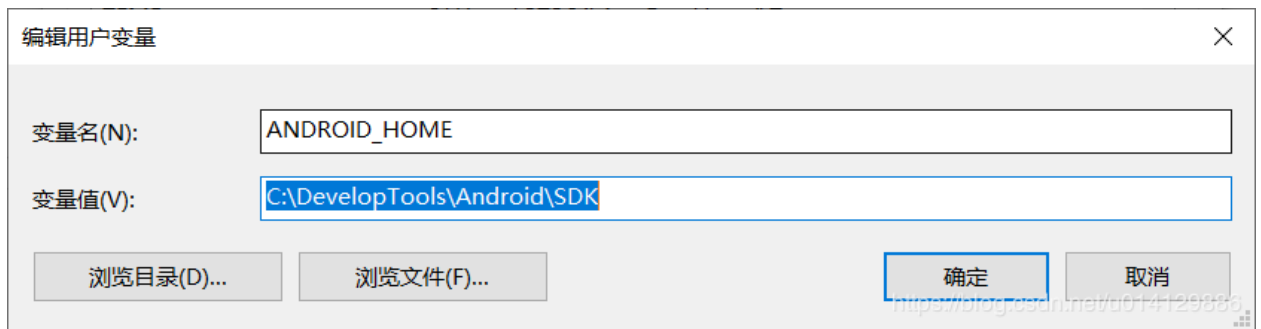
Затем подождите, пока он загрузится ~~

2.3 Добавить flutter и Android SDK в переменные среды

1. Добавьте «C: \ DevelopTools \ Android \ flutter \ bin» (ваш путь «flutter \ bin») к пути переменной среды. На картинке ниже показан снимок экрана win10, просто нажмите «Создать» справа и введите значение. Для win7 добавьте это в "; C: \ DevelopTools \ Android \ flutter \ bin" (";" Не забудьте)



2. Также добавьте Android SDK в переменную среды, как показано ниже.



ANDROID_HOME: C: \ DevelopTools \ Android \ SDK (собственный путь к SDK)

После настройки этих двух переменных среды **Начать сначала**.

2.4 Откройте командную строку и введите flutter doctor

Тогда придется долго ждать ~~

```
C:\>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel master, v1.5.9-pre.150, on Microsoft Windows [Version 10.0.17763.475], locale zh-CN)

[!] Android toolchain - develop for Android devices (Android SDK version 28.0.3)
    ! Some Android licenses not accepted. To resolve this, run: flutter doctor --android-licenses
[✓] Android Studio (version 3.4)
[✓] Connected device (1 available)

! Doctor found issues in 1 category.

C:\>flutter doctor --android-licenses
Warning: File C:\...\.android\repositories
.cfg could not be loaded.
5 of 6 SDK package licenses not accepted. 100% Computing updates...
Review licenses that have not been accepted (y/N)? y
```

Это должно быть предложено после завершения установки. Существуют некоторые соглашения, которые необходимо согласовать. Введите «flutter doctor --android-licenses» в соответствии с приведенным выше запросом. Будет ряд различных соглашений. Просто нажмите «Y» до конца. (Если вам интересно, можете взглянуть на соглашение. Конечно, это бесполезно. В любом случае, вам все равно придется согласиться в конце ~~)

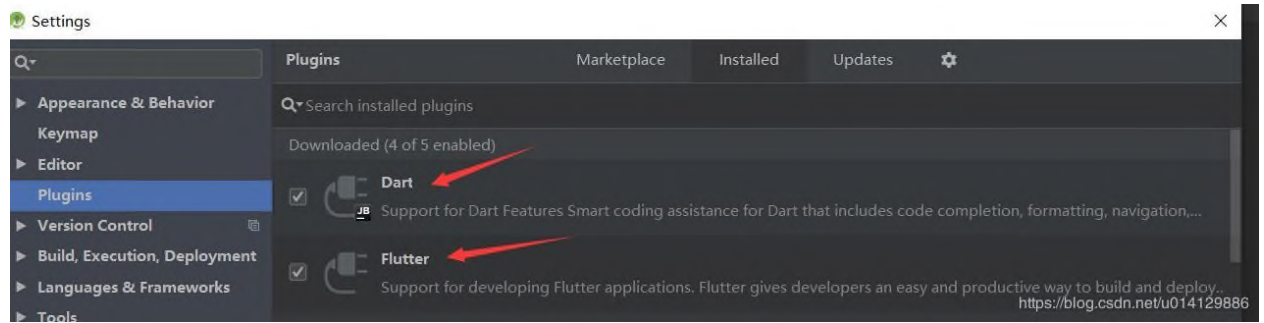
При появлении следующего запроса подключите телефон к компьютеру или откройте виртуальную машину.

```
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel master, v1.5.9-pre.150, on Microsoft Windows [Version 10.0.17763.475],
[✓] Android toolchain - develop for Android devices (Android SDK version 28.0.3)
[✓] Android Studio (version 3.4)
[!] Connected device
    ! No devices available

! Doctor found issues in 1 category.
```

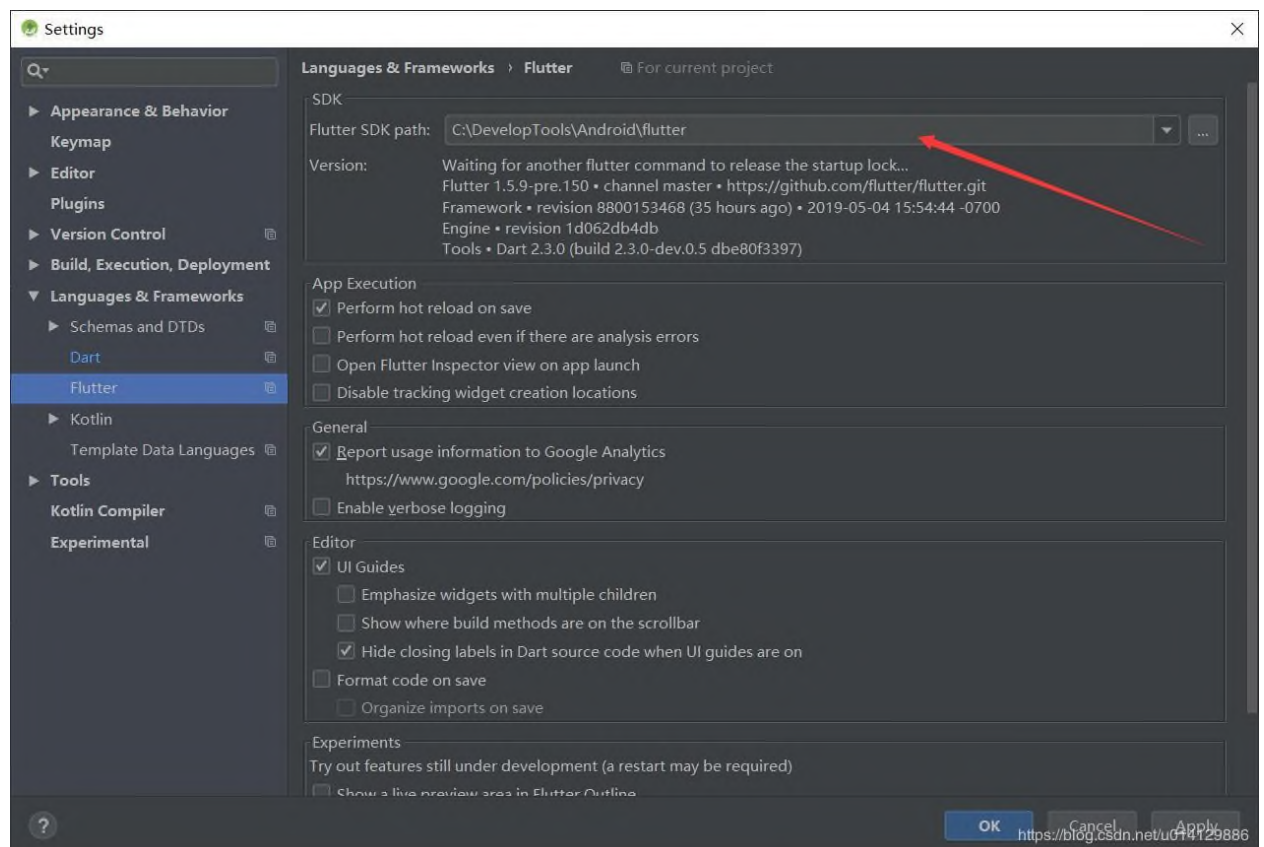
2.5 Скачайте плагины flutter и dart

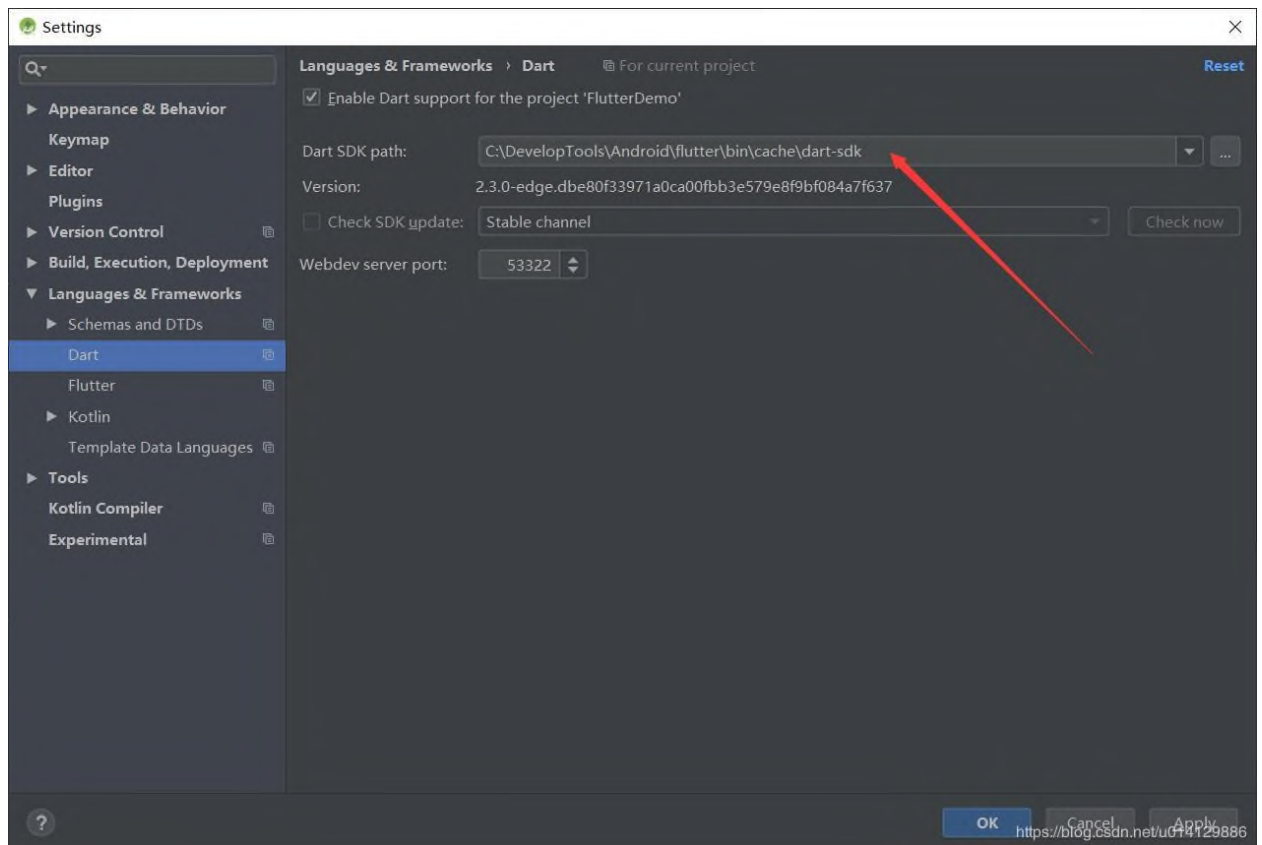
Откройте файл - настройка - плагины, затем найдите следующие два плагина и установите их. Перезагрузите после установки.



2.6 Настроить траекторию флаттера и дротика

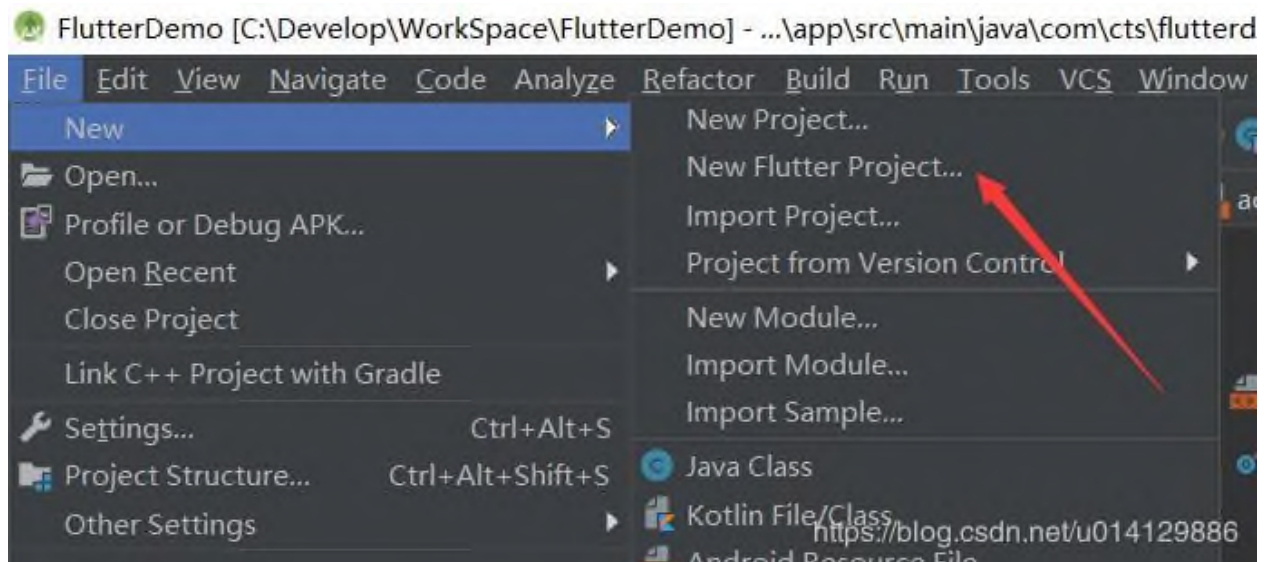
Найдите в настройках «Языки - Flutter», а затем заполните путь установки Flutter в месте, показанном на рисунке ниже. После того, как я его заполнил, Дарт автоматически заполнил его здесь, в противном случае - вручную. Путь Dart показан ниже:



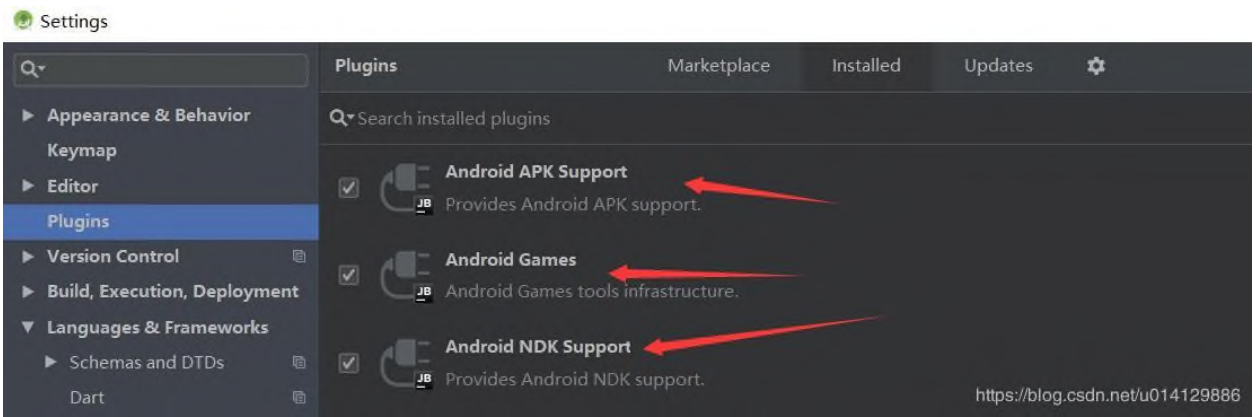


2.7 Завершен, новый проект

Теперь вы можете создать проект Flutter. В файле - Новое - Новый проект Flutter



Интерлюдия: Если все вышеперечисленные шаги настроены правильно, но в «Новый» нет опции «Новый проект Flutter». Откройте файл - настройка - плагины и отметьте, чтобы использовать следующие плагины:



Тогда посмотри на это еще раз ~~~
The End...

Интеллектуальная рекомендация



Принцип и алгоритм прямой сортировки java

Первый: основная идея означает, что указанная позиция сортировки сравнивается с другими элементами массива, и элементы обмениваются, когда выполняются условия. Обратите внимание на разницу между пузырь...



01 linux

ключ vmware <http://www.hack520.com/293.html> Ключ vmwarePro15 YG5H2-ANZ0H-M8ERY-TXZZZ-YKRV8 1. Структура каталогов Linux 1.

Переключите каталог команды ...



Инкапсуляция образа Docker (в качестве примера возьмем ssh и apache)

Инкапсулируйте образ apache Тест: пакет ssh Тест:...

Berkeley DB база данных с открытым исходным кодом

Файловая база данных с открытым исходным кодом, между реляционной базой данных и базой данных в памяти, хранящаяся в парах ключ-значение. Ниже приведен пример, скопированный онлайн, а затем запустить е...

Формат данных JSON часто используется в веб-разработке. Часто возникает проблема с преобразованием класса в формат данных JSON. Особенно наглядным примером является представление внешних данных в форм...

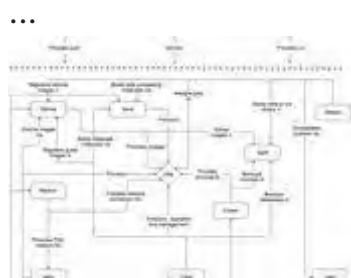
Вам также может понравиться

Предисловие Увидеть TJ Великий Бог звезд adom-to-image И всегда было любопытно html Как повернуть image Так что просто посмотрите исходный код и посмотрите, как он реализован: на самом деле, всего кода мень...



Application.yml Файл конфигурации ниже Много Baidu сказал, что это разрешение, но разрешения не проблем. Все не проблема, но это ошибка. Затем я нашел другое объяснение, 15672 в Port = 15672 - это пор...

Существует двусторонний связанный список, его значение не повторяется, удаляет ключевые слова к узлам ключей



Анализ архитектуры OpenStack

1. Общая структура На следующем рисунке показана взаимосвязь между OpenStack Services. Nova: управление жизненным циклом виртуальных машин Neutron: предоставляет услуги сетевого подключения для других...

7 лекция.

Тема: Flutter: интерфейс Android Studio, написание первой программы и ее использование в эмуляторе, работа с компонентами.

План:

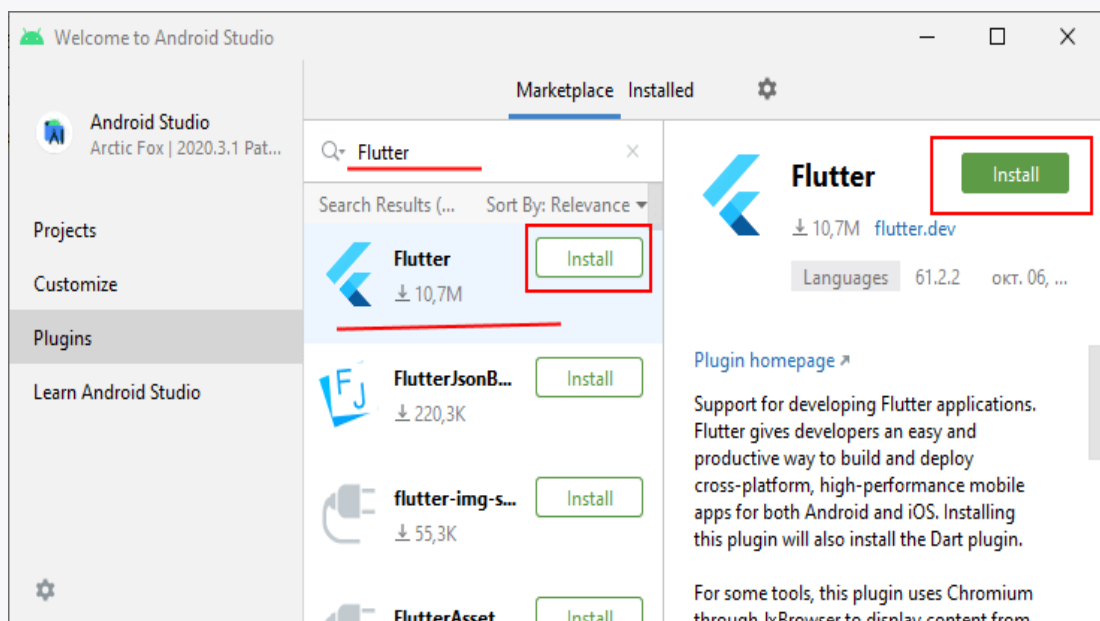
1. Разработка под Flutter.
2. Установка плагина в Android Studio.
3. Компиляция проекта.

1. Разработка под Flutter.

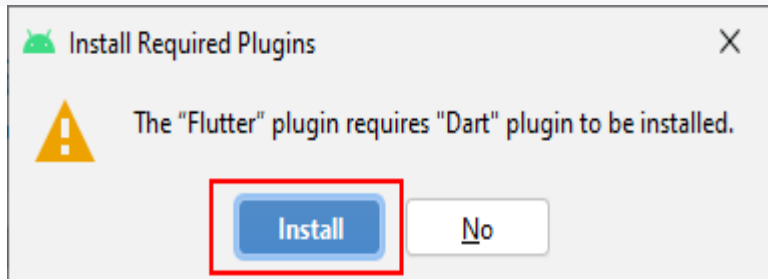
Для разработки под Flutter нередко выбирается такая среда разработки как **Android Studio**. Хотя мы можем набирать код и в простейшем текстовом редакторе и компилировать его в консоли, но среда разработки существенно позволяет упростить процесс написания и построения приложения. Причем Android Studio позволяет создать приложения на Flutter не только собственно под Android, но и под другие поддерживаемые платформы.

Для работы с Android Studio ее естественно вначале надо установить. Инсталлятор можно загрузить по ссылке <https://developer.android.com/studio>.

По умолчанию Android Studio не поддерживает Flutter, поэтому нам надо установить соответствующий плагин. Для этого в Android Studio на стартовом экране выберем пункт **Plugins** (либо в открытой студии перейдем в меню **File** -> **Settings** и далее в открывшемся окне также выберем пункт **Plugins**). И в панели плагинов найдем плагин **Flutter**:



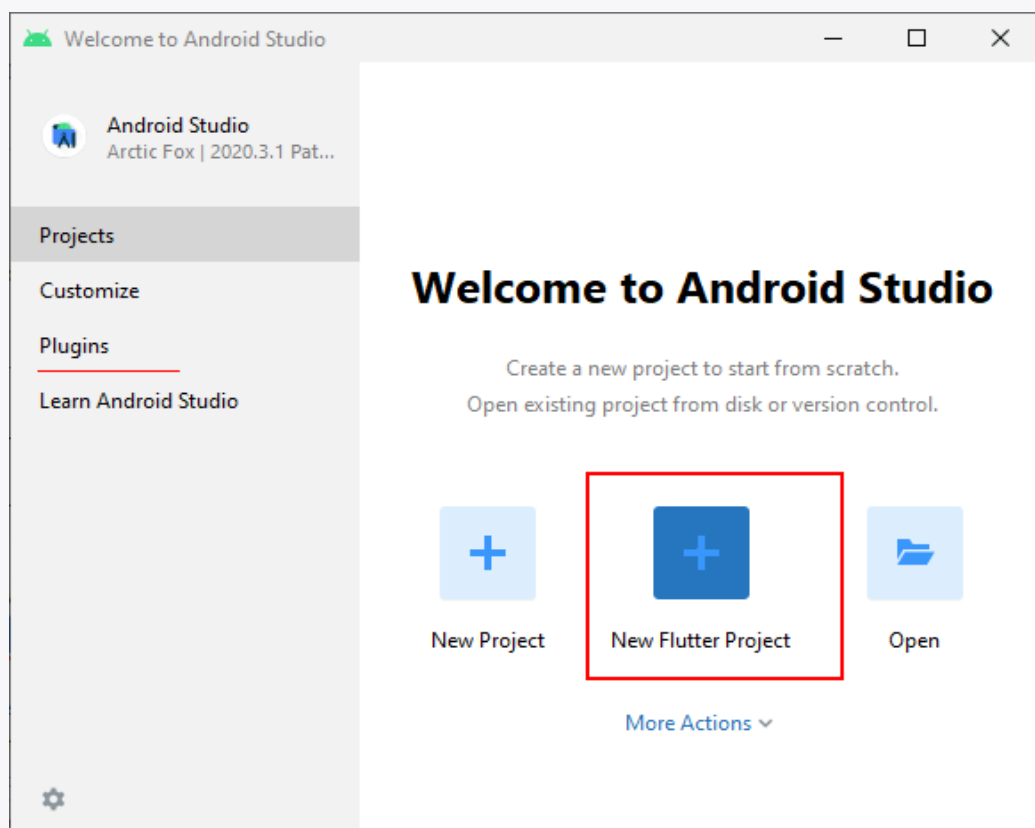
Для упрощения поиска нужного плагина мы можем ввести в поисковую строку слово "Flutter", и первый результат будет как раз тем, который надо установить. При установке плагина также отобразится окно с предложением установить плагин для Dart. Также нажмем на ОК для его установки:



После установки плагина необходимо будет перезагрузить Android Studio.

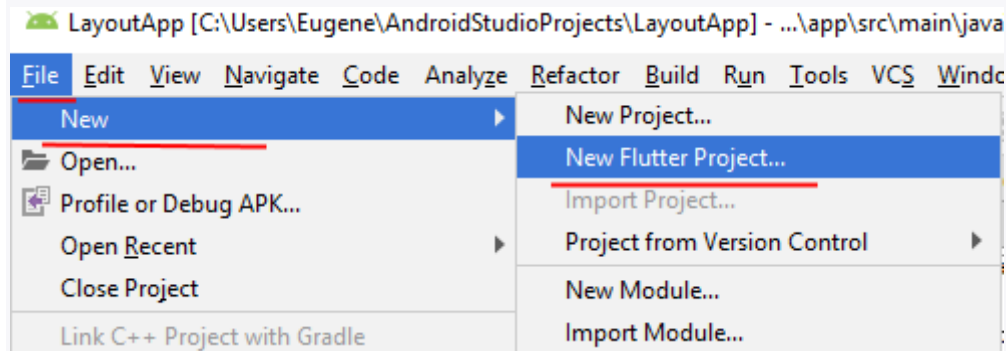
2. Установка плагина в Android Studio.

После перезагрузки на стартовом экране в Android Studio мы можем увидеть кнопку **New Flutter Project**:

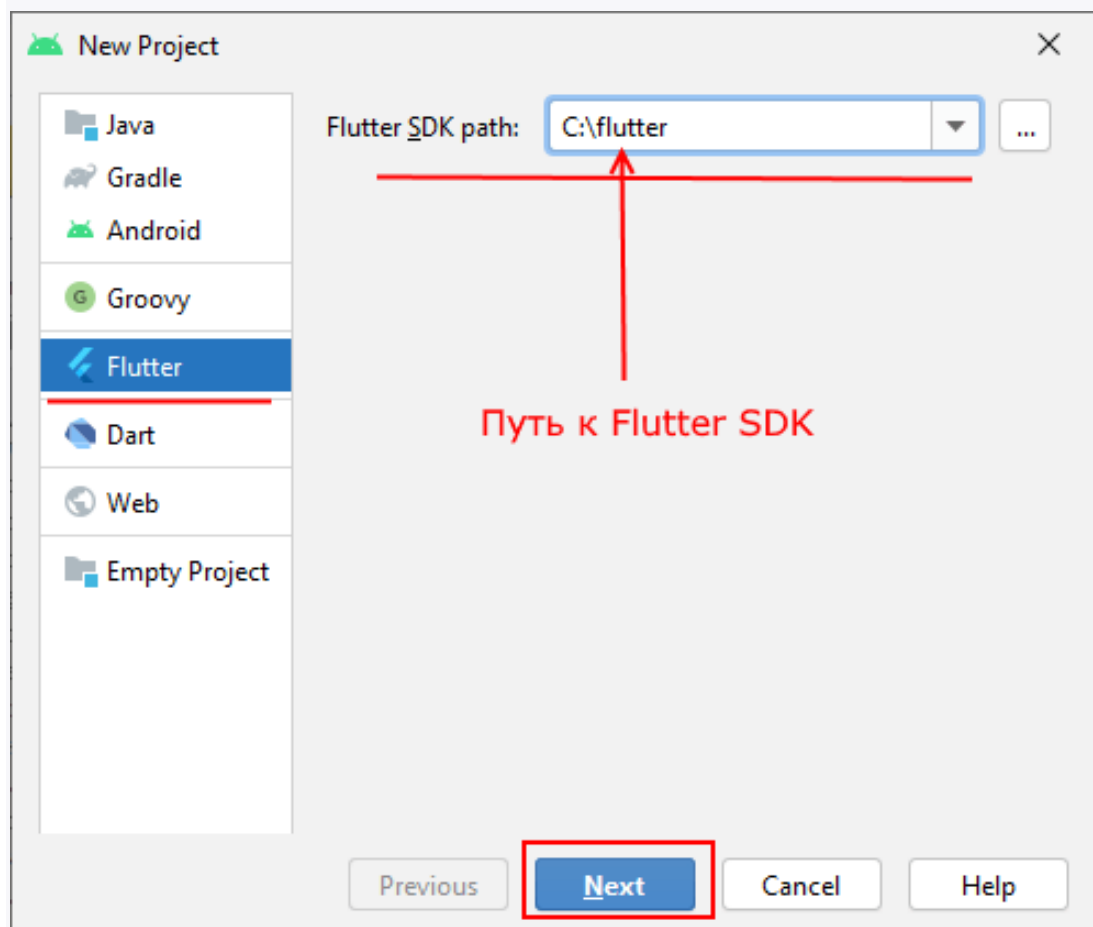


Нажмем на эту кнопку для создания проекта под Flutter.

В качестве альтернативы для создания проекта в студии можно перейти в меню к пункту **File -> New -> New Flutter Project**:



Далее нам откроется окно создания нового проекта. В левой части выберем пункт **Flutter**, а в центре в поле **Flutter SDK path** укажем путь к Flutter SDK:



На следующем окне укажем ряд настроек проекта:

The screenshot shows the 'New Project' dialog box in Android Studio. The fields are filled with the following values:

- Project name:** hello_app
- Project location:** C:\Users\Eugene\AndroidStudioProjects\Flutter\hello_app
- Description:** A new Flutter project.
- Project type:** Application
- Organization:** com.metanit
- Android language:** Kotlin (selected)
- iOS language:** Swift (selected)
- Platforms:** Android, iOS, and Web are checked.

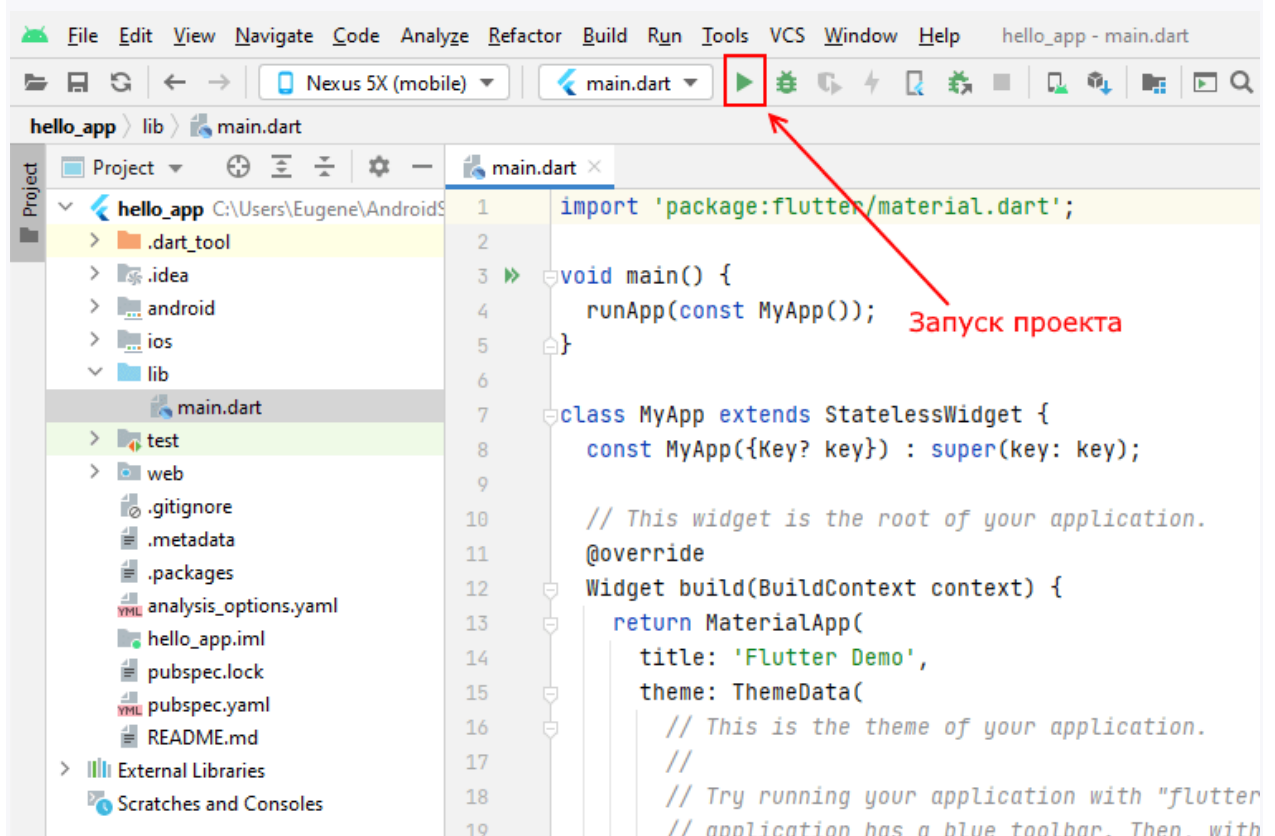
At the bottom, there is a 'More Settings' link and four buttons: 'Previous', 'Finish' (highlighted with a red box), 'Cancel', and 'Help'.

- В поле **Project name** дадим проекту какое-либо имя. Так, в моем случае он называется **hello_app**.
- В поле **Project location** можно изменить расположение проекта, если предложенное расположение по умолчанию не устраивает.
- В поле **Description** можно указать описание проекта
- В поле **Project type** указывается тип проекта. По умолчанию он имеет значение **Application** (то есть проект предназначен для создания приложения). Оставим это значение по умолчанию.
- В поле **Organization** можно задать название для пакета приложения. Можно оставить по умолчанию, а можно и изменить. Например, в моем случае это com.metanit.

- В поле **Android language** указывается язык для Android. Можно оставить значение по умолчанию - Kotlin.
- В поле **iOS language** указывается язык для платформы iOS. Можно оставить значение по умолчанию - Swift.
- В поле **Organization** можно задать название для пакета приложения. Можно оставить по умолчанию, а можно и изменить. Например, в моем случае это com.metanit.
- В поле **Platforms** можно указать платформы, под которые будет создаваться проект. По умолчанию отмечены пункты Android и iOS, но можно выбрать и другие доступные платформы. Так, как видно выше на скриншоте, я также выбрал пункт "Web" для создания проекта под веб.

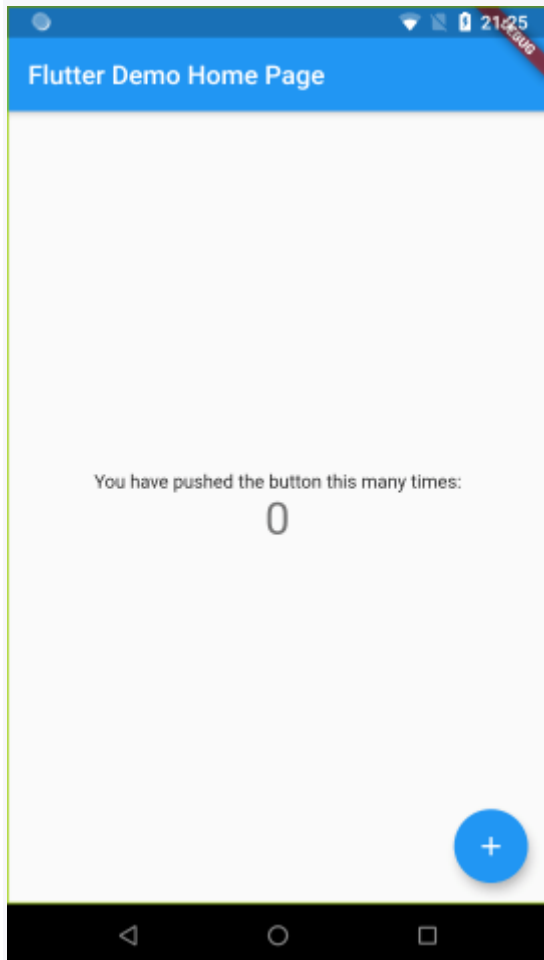
3. Компиляция проекта.

И затем после установки всех настроек нажмем на кнопку "Finish" для непосредственного создания проекта. Сразу после создания Android Studio откроет созданный проект.

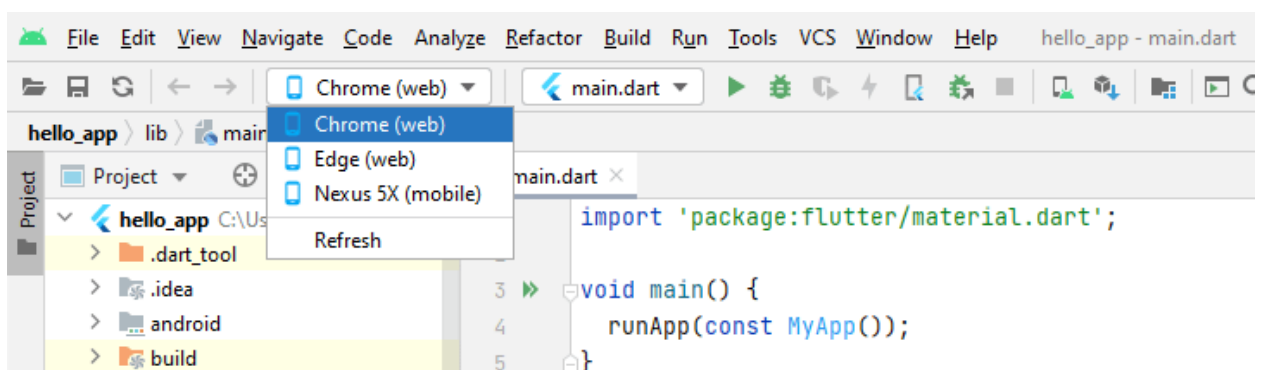


Созданный проект будет иметь ту же самую структуру, что был создан в прошлой теме в консоли с помощью команды flutter create. В центре студии будет открыт файл **main.dart**, который содержит собственно код приложения.

Подключим к компьютеру устройство Android (или воспользуемся эмуляторами) и в панели Android Studio нажмем на зеленую стрелочку для запуска приложения.



Подобным образом в Android Studio можно запускать проект и под другие "устройства", например, под web. Для этого лишь нужно выбрать соответствующее устройство в панели инструментов:



8 лекция.

Flutter: Работа с ключевыми компонентами в Android Studio: Layout, Table, ListView, Grid, List и т. д.

План:

1. Компоненты **ListView**.
2. Виды методов
3. Динамическое заполнение списка

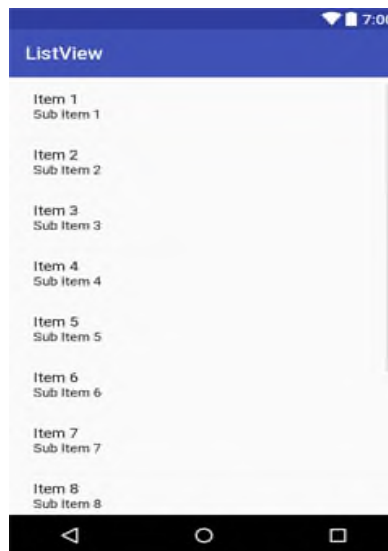
1. Компоненты **ListView**.

В ранних версиях Android компонент **ListView** был одним из самым популярных элементов интерфейса. Но теперь его время ушло, недаром на панели инструментов студии он находится в разделе **Legacy** (устаревший код). **ListView** представляет собой прокручиваемый список элементов. Очень популярен на мобильных устройства из-за своего удобства. Даже кот способен пользоваться этим элементом, проводя лапкой по экрану вашего телефона.



Компонент **ListView** более сложен в применении по сравнению с **TextView** и другим простыми элементами. Работа со списком состоит из двух частей. Сначала мы добавляем на форму сам **ListView**, а затем заполняем его элементами списка.

Рассмотрим для начала самый простой пример. Поместите на форму компонент **ListView** и присвойте идентификатор. Вы увидите, что список будет содержать несколько элементов **Item** и **Sub Item**.



Однако, если посмотрим XML-код, то там ничего не увидим.

```
<ListView
    android:id="@+id/listView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >
</ListView>
```

Переходим в класс активности и пишем следующий код:

```
// код пишется в методе onCreate()

// получаем экземпляр элемента ListView
ListView listView = findViewById(R.id.listView);

// определяем строковый массив
final String[] catNames = new String[] {
    "Рыжик", "Барсик", "Мурзик", "Мурка", "Васька",
    "Томасина", "Кристина", "Пушок", "Дымка", "Кузя",
    "Китти", "Масяня", "Симба"
};

// используем адаптер данных
ArrayAdapter<String> adapter = new ArrayAdapter<>(this,
    android.R.layout.simple_list_item_1, catNames);

listView.setAdapter(adapter);
Вот и всё. Давайте разберёмся с кодом.
```

Адаптеры - заполнение списка данными

Компоненту **ListView** требуются данные для наполнения. Источником наполнения могут быть массивы, базы данных. Чтобы связать данные со списком, используется так называемый **адаптер**.

Адаптер для стандартного списка обычно создается при помощи конструкции *new ArrayAdapter(Context context, int textViewResourceId, String[] objects)*.

- **context** - текущий контекст
- **textViewResourceId** - идентификатор ресурса с разметкой для каждой строки. Можно использовать системную разметку с идентификатором **android.R.layout.simple_list_item_1** или создать собственную разметку
- **objects** - массив строк

2. Виды методов

Метод **setAdapter(ListAdapter)** связывает подготовленный список с адаптером.

Переходим к java-коду. Сначала мы получаем экземпляр элемента **ListView** в методе **onCreate()**. Далее мы определяем массив типа **String**. И, наконец, используем адаптер данных, чтобы сопоставить данные с шаблоном разметки. Выбор адаптера зависит от типа используемых данных. В нашем случае мы использовали класс **ArrayAdapter**.

Отступление

Если вы будете брать строки из ресурсов, то код будет таким:

```
final String[] catNames = {  
    getResources().getString(R.string.name1),  
    getResources().getString(R.string.name2),  
    getResources().getString(R.string.name3),  
    getResources().getString(R.string.name4),  
    getResources().getString(R.string.name5),  
};
```

А будет еще лучше, если вы воспользуетесь специально предназначенным для этого случая типом ресурса **<string-array>**. В файле **res/values/strings.xml** добавьте следующее:

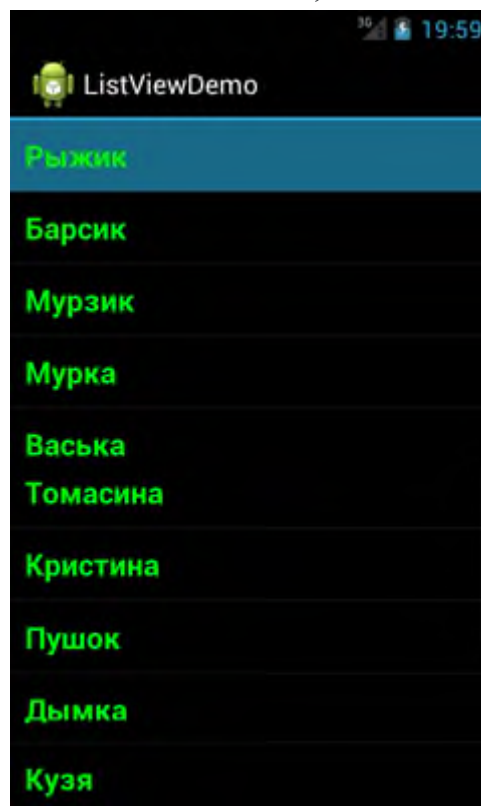
```
<string-array name="cat_names">  
    <item>Рыжик</item>
```

```
<item>Барсик</item>
<item>Мурзик</item>
<item>Мурка</item>
<item>Васька</item>
<item>Томасина</item>
<item>Кристина</item>
<item>Пушок</item>
<item>Дымка</item>
<item>Кузя</item>
<item>Китти</item>
<item>Масяня</item>
<item>Симба</item>
</string-array>
```

И тогда в коде используйте для объявления массива строк:

```
String[] catNames = getResources().getStringArray(R.array.cat_names);
```

Запустив проект, вы увидите работающий пример прокручиваемого списка. Правда, созданный список пока не реагирует на нажатия. Но при нажатии выбранный элемент выделяется цветным прямоугольником (в версии Android 2.3 был оранжевый, а в Android 4.0 - синий, потом был серый цвет и т.д.).



Собственная разметка

В примере мы используем готовую системную разметку `android.R.layout.simple_list_item_1`, в которой настроены цвета,

фон, высота пунктов и другие параметры. Но нет никаких препятствий самому создать собственную разметку под своё приложение.

Но для начала неплохо бы взглянуть на содержание системной разметки. Студия позволяет увидеть исходный код, достаточно в коде поставить курсор на **simple_list_item_1** и нажать на комбинацию клавиш **Ctrl+B**. Наш *simple_list_item_1* выглядит так (в одной из версий):

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<!-- Copyright (C) 2006 The Android Open Source Project
```

```
    Licensed under the Apache License, Version 2.0 (the "License");  
    you may not use this file except in compliance with the License.  
    You may obtain a copy of the License at
```

```
        http://www.apache.org/licenses/LICENSE-2.0
```

```
    Unless required by applicable law or agreed to in writing, software  
    distributed under the License is distributed on an "AS IS" BASIS,  
    WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express  
    or implied.
```

```
    See the License for the specific language governing permissions and  
    limitations under the License.
```

```
-->
```

```
<TextView xmlns:android="http://schemas.android.com/apk/res/android"  
    android:id="@android:id/text1"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:textAppearance="?android:attr/textAppearanceListItemSmall"  
    android:gravity="center_vertical"  
    android:paddingStart="?android:attr/listPreferredItemPaddingStart"  
    android:paddingEnd="?android:attr/listPreferredItemPaddingEnd"  
    android:minHeight="?android:attr/listPreferredItemHeightSmall" />
```

Мы видим, что в качестве разметки используется **TextView** с набором атрибутов.

Если говорить о системных разметках, то имеется несколько вариантов. Вкратце ознакомимся с ними.

`android.R.layout.simple_list_item_1`

Состоит из одного **TextView** (см. выше)

```
android.resource.id.text1
```

```
android.R.layout.simple_list_item_2
```

Состоит из двух **TextView** - один побольше сверху и второй поменьше под ним.

```
android.resource.id.text1
```

```
android.resource.id.text2
```

```
android.R.layout.simple_list_item_checked
```

Справа от **CheckedTextView** будет находиться флажок

```
android.resource.id.text1 ☒
```

```
android.R.layout.activity_list_item
```

Слева от **TextView** находится значок **ImageView** с идентификатором **android.resource.id.Icon**.

```
android.resource.id.text1
```

Создадим свой шаблон для отдельного пункта списка. Для этого в папке **res/layout/** создадим новый файл **list_item.xml**:

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="10dp"
    android:textColor="#00FF00"
    android:textSize="20sp"
    android:textStyle="bold" >
```

```
</TextView>
```

В некоторых случаях желательно установить атрибут **android:background="?android:attr/activatedBackgroundIndicator"** у родительского элемента, чтобы элементы списка реагировали на нажатие изменением цвета. Можно задать и [собственное поведение](#).

Вы можете настраивать все атрибуты у **TextView**, кроме свойства **Text**, так как текст будет автоматически заполняться элементом **ListView** программным путём. Ну, а дальше просто меняете в коде системную разметку на свою:

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
R.layout.list_item, catNames);
```


При создании собственного элемента списка, состоящего из **TextView** можете использовать специальный стиль для минимального размера текста.

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/list_item"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center_vertical"
    android:minHeight="?android:attr/listPreferredItemHeight" />
```

3. Динамическое заполнение списка

Рассмотрим пример динамического заполнения списка, когда список изначально пуст и пользователь сам добавляет новые элементы. Разместим на экране текстовое поле, в котором пользователь будет вводить известные ему имена котов. Когда пользователь будет нажимать на клавишу Enter на клавиатуре, то введенное имя кота будет попадать в список.

@Override

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    // получаем экземпляр элемента ListView
    ListView listView = (ListView) findViewById(R.id.listView);
    final EditText editText = (EditText) findViewById(R.id.editText);

    // Создаём пустой массив для хранения имен котов
    final ArrayList<String> catNames = new ArrayList<>();

    // Создаём адаптер ArrayAdapter, чтобы привязать массив к ListView
    final ArrayAdapter<String> adapter;
    adapter = new ArrayAdapter<>(this,
        android.R.layout.simple_list_item_1, catNames);
    // Привяжем массив через адаптер к ListView
    listView.setAdapter(adapter);

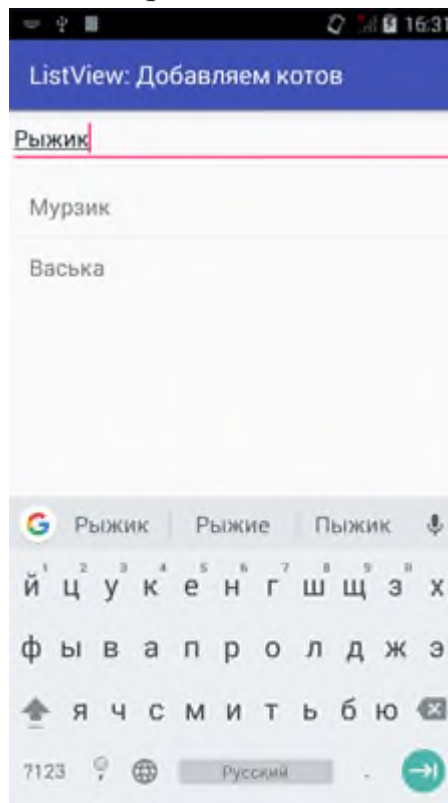
    // Прослушиваем нажатия клавиш
    editText.setOnKeyListener(new View.OnKeyListener() {
```

```

public boolean onKey(View v, int keyCode, KeyEvent event) {
    if (event.getAction() == KeyEvent.ACTION_DOWN)
        if (keyCode == KeyEvent.KEYCODE_ENTER) {
            catNames.add(0, editText.getText().toString());
            adapter.notifyDataSetChanged();
            editText.setText("");
            return true;
        }
    return false;
}
});
}

```

При нажатии на Enter мы получаем текст из текстового поля и заносим его в массив. А также оповещаем адаптер об изменении, чтобы список автоматически обновил своё содержание.



У нас получился каркас для чата, когда пользователь вводит текст и он попадает в список. Далее надо получить текст от другого пользователя и также добавить в список. К слову сказать, слово **chat** с французского означает "кошка". Но это уже совсем другая история.

Прослушивание событий элемента ListView

Нам нужно реагировать на определенные события, генерируемые элементом **ListView**, в частности, нас интересует событие, которое возникает, когда пользователь нажимает на один из пунктов списка.

В этом нам поможет метод **setOnItemClickListener** элемента **ListView** и метод **OnItemClick()** интерфейса **AdapterView.OnItemClickListener**.

```
listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {  
    @Override  
    public void onItemClick(AdapterView<?> parent, View itemClicked, int  
position,  
                                long id) {  
        Toast.makeText(getApplicationContext(), ((TextView)  
itemClicked).getText(),  
                                Toast.LENGTH_SHORT).show();  
    }  
});
```

Теперь при нажатии на любой элемент списка мы получим всплывающее сообщение, содержащее текст выбранного пункта. Естественно, мы можем не только выводить сообщения, но и запускать новые активности и т.п.

```
listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {  
    @Override  
    public void onItemClick(AdapterView<?> parent, View itemClicked, int  
position,  
                                long id) {  
        TextView textView = (TextView) itemClicked;  
        String strText = textView.getText().toString(); // получаем текст  
нажатого элемента  
  
        if(strText.equalsIgnoreCase(getResources().getString(R.string.name1))) {  
            // Запускаем активность, связанную с определенным именем  
кота  
            startActivity(new Intent(this, BarsikActivity.class));  
        }  
    }  
});
```

В метод **onItemClick()** передаётся вся необходимая информация, необходимая для определения нажатого пункта в списке. В приведенном выше примере использовался простой способ - приводим выбранный элемент к объекту **TextView**, так как известно, что в нашем случае все пункты являются

элементами **TextView** (Для дополнительной проверки можете использовать оператор **instanceOf**). Мы извлекаем текст из выбранного пункта и сравниваем его со своей строкой.

Также можно проверять атрибут **id** для определения нажатия пункта списка.

Программное нажатие на элемент списка

Вдруг вам захочется программно нажать на элемент списка. Мы задали код, который будет выполняться при нажатии, в предыдущем примере. Теперь добавим кнопку и напишем код для щелчка.

```
public void onClick(View view) {  
    int activePosition = 0; // первый элемент списка  
    listView.performItemClick(listView.getAdapter().  
        getView(activePosition, null, null), activePosition, listView.getAdapter().  
        getItemId(activePosition));  
}
```

Код громоздкий, но работоспособный.

ListView не реагирует на нажатия

В некоторых случаях нажатия на пунктах меню не срабатывают. Ниже приводятся несколько возможных причин.

Элемент списка содержит **CheckBox**, который также имеет свой слушатель нажатий. Попробуйте удалить фокус у него:

```
android:focusable="false"
```

```
android:focusableInTouchMode="false"
```

Попробуйте переместить **OnItemClickListener** перед установкой адаптера. Иногда помогает.

Элемент списка содержит **ImageButton**. Установите фокус в **false**:

```
ImageButton                imageButton                =                (ImageButton)  
convertView.findViewById(R.id.imageButton);  
imageButton.setFocusable(false);
```

Элемент списка содержит **TextView**. Если вы используете атрибут **android:inputType="textMultiLine"**, то замените его на **android:minLines/android:maxLines**.

Элемент списка содержит **TextView**, содержащий ссылку на веб-страницу или электронный адрес. Удалите атрибут **android:autoLink**.

Настраиваем внешний вид ListView

У **ListView** есть несколько полезных атрибутов, позволяющих сделать список более привлекательным. Например, у него есть атрибут **divider**, который

отвечает за внешний вид разделителя, а также атрибут **dividerHeight**, отвечающий за высоту разделителя. Мы можем установить какой-нибудь цвет или даже картинку для разделителя. Например, создадим для разделителя цветовой ресурс с красным цветом, а также ресурс размера для его высоты:

```
<color name="reddivider">#FF0000</color>
```

```
<dimen name="twodp">2dp</dimen>
```

Далее присвоим созданный ресурс атрибуту **divider**, а также зададим его высоту в атрибуте **dividerHeight** у нашего элемента ListView:

```
<ListView
```

```
    android:id="@+id/listView1"
```

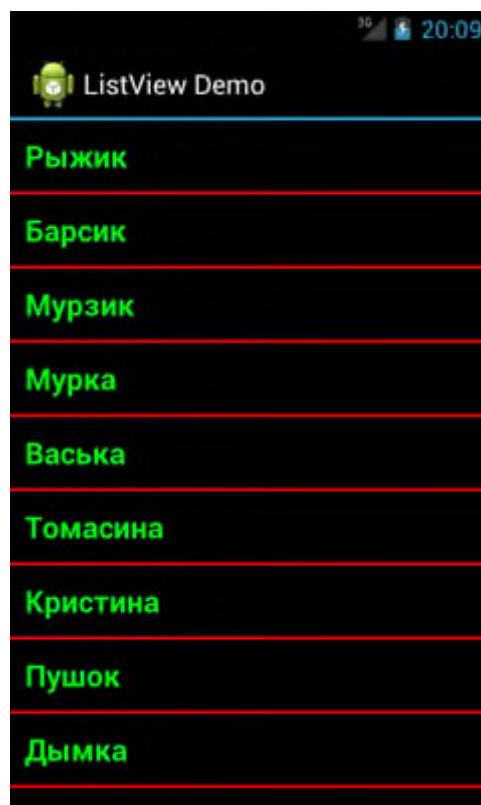
```
    android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content"
```

```
    android:divider="@color/reddivider"
```

```
    android:dividerHeight="@dimen/twodp" >
```

```
</ListView>
```



Если вас не устраивает стандартный разделитель, что можете нарисовать какую-нибудь волнистую черту, сохранить ее в PNG-файле и использовать как drawable-ресурс. Прodelайте это самостоятельно.

Можно работать с данными атрибутами программно:

```
ColorDrawable divcolor = new ColorDrawable(Color.DKGRAY);
```

```
listView.setDivider(divcolor);
```

```
listView.setDividerHeight(2);
```

Если хотите убрать разделители, то используйте прозрачный цвет.

```
listView.setDivider(getResources().getDrawable(android.R.color.transparent));
```

Заметил, что порядок вызова двух методов важен, если установку высоты вызвать перед установкой цвета разделителя, то метод затирает цвет и результат будет такой же, как с прозрачным цветом.

Обратите внимание, что по умолчанию разделитель не выводится перед первым и последним элементом списка. Если вы хотите изменить эти настройки, то используйте свойства **Footer dividers enabled** (атрибут **footerDividersEnabled**) и **Header dividers enabled** (атрибут **headerDividersEnabled**):

...

```
android:footerDividersEnabled="true"
```

```
android:headerDividersEnabled="true"
```

...

Пользовательский селектор

Мы уже видели, что по умолчанию выбранный элемент списка выделяется при помощи цветной полоски. Данный селектор также можно настроить через атрибут **android:listSelector**. Создайте какую-нибудь текстуру для селектора и привяжите его через ресурс. Вот образец текстурированного ореола желтого цвета для селектора.



Нужно подготовить сначала файл **res/drawable/selector.xml**:

```
<selector xmlns:android="http://schemas.android.com/apk/res/android">
```



```
<item android:state_pressed="true" android:drawable="@drawable/bgground"/>
</selector>
```

Если вам нужно сразу подсветить нужный элемент списка при запуске программы, то используйте связку двух методов:

```
listView.requestFocusFromTouch();
```

```
listView.setSelection(4); // выбираем 5 пункт списка
```

Множественный выбор

ListView позволяет выбрать не только один пункт, но и несколько. В этом случае нужно установить свойство **Choice Mode** в значение **multipleChoice**, что соответствует атрибуту **android:choiceMode="multipleChoice"**.

Также множественный выбор можно установить программно при помощи метода **setChoiceMode(ListView.CHOICE_MODE_MULTIPLE)**.

Теперь, если создать массив строк, например список продуктов для кошачьего завтрака, то получим следующий результат.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
```

```
<TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Завтрак для кота" />
```

```
<ListView
    android:id="@+id/listView1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:choiceMode="multipleChoice" >
</ListView>
```

```
</LinearLayout>
```

```
public class MultiChoiceListViewActivity extends Activity {
    ListView choiceList;
```

```

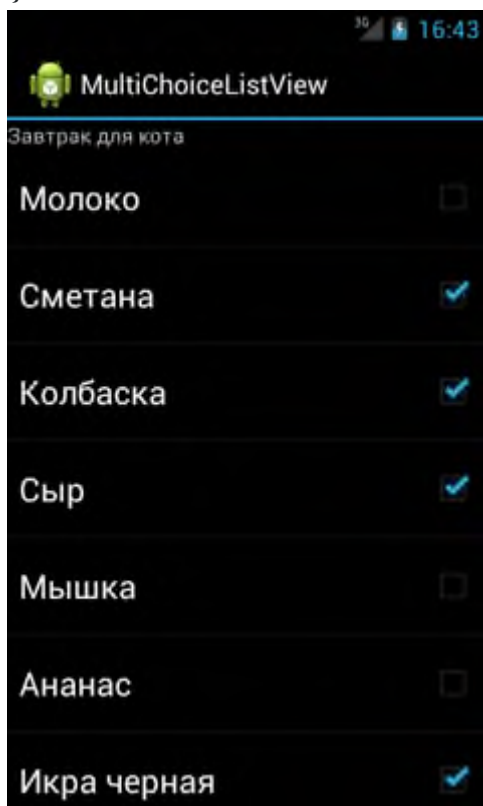
TextView selection;
String[] foods = { "Молоко", "Сметана", "Колбаска", "Сыр", "Мышка",
                  "Ананас", "Икра черная", "Икра кабачковая", "Яйцо" };

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    selection = (TextView) findViewById(R.id.textView1);
    choiceList = (ListView) findViewById(R.id.listView1);
    ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,

    android.R.layout.simple_list_item_multiple_choice, foods);
    //
    choiceList.setChoiceMode(ListView.CHOICE_MODE_MULTIPLE);
    choiceList.setAdapter(adapter);
}
}

```

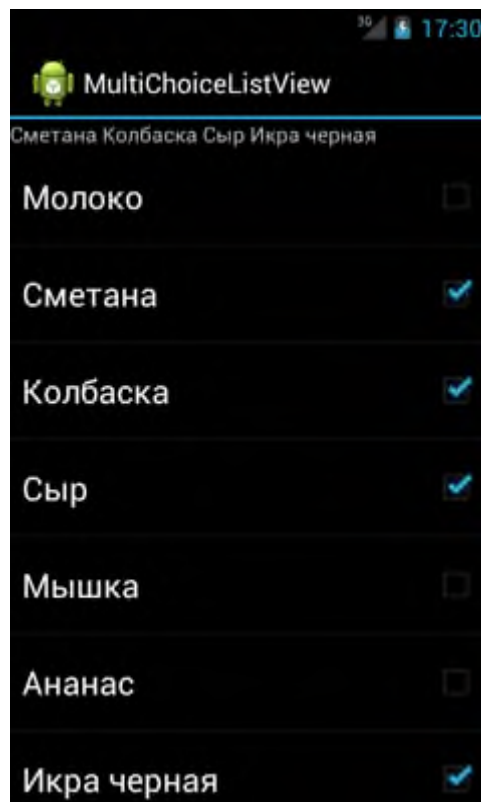


Осталось только программно получить отмеченные пользователем элементы списка. Вот мой список продуктов, который я хочу предложить коту. Надеюсь, ему понравится мой выбор. Выбранные элементы будем помещать в **TextView**:

```

choiceList.setOnItemClickListener(new OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View v,
        int position, long id) {
        // Очистим TextView
        selection.setText("");
        // получим булев массив для каждой позиции списка
        // Объект SparseBooleanArray содержит массив значений, к
        // которым можно получить доступ
        // через valueAt(index) и keyAt(index)
        SparseBooleanArray chosen = ((ListView)
parent).getCheckedItemPositions();
        for (int i = 0; i < chosen.size(); i++) {
            // если пользователь выбрал пункт списка,
            // то выводим его в TextView.
            if (chosen.valueAt(i)) {
                selection.append(foods[chosen.keyAt(i)] + " ");
            }
        }
    }
});

```



Если нужно получить отдельно список выбранных и невыбранных элементов списка, то можно написать следующее:

```

choiceList.setOnItemClickListener(new OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View v,
        int position, long id) {
        // Очистим TextView перед вставкой нового контента.
        selection.setText("");

        int cntChoice = choiceList.getCount();

        String checked = "";

        String unchecked = "";
        SparseBooleanArray sparseBooleanArray = choiceList
            .getCheckedItemPositions();

        for (int i = 0; i < cntChoice; i++) {

            if (sparseBooleanArray.get(i) == true) {
                checked
choiceList.getItemAtPosition(i).toString()
                + "\n";
                // выводим список выбранных элементов
                //selection.setText(checked);
            } else if (sparseBooleanArray.get(i) == false) {
                unchecked
choiceList.getItemAtPosition(i).toString()
                + "\n";
                // выводим список невыбранных элементов
                selection.setText(unchecked);
            }
        }
    }
});

```

Переменная *checked* будет содержать список выбранных элементов, а переменная *unchecked* - список невыбранных элементов.

Следует отметить, что в примерах использовался старый метод **getCheckedItemPositions()**, доступный с Android 1. В Android 2.2 появился новый метод **getCheckedItemIds()**. Учтите, что с новым методом

можно получить массив только выбранных элементов, хотя в большинстве случаев этого достаточно. Но данный метод требует своих заморочек и в данном моём примере он не заработал.

Подсветка нажатий

На данный момент используется следующая техника подсвечивания элементов списка при нажатии. Здесь учитывается версия Android (до и после API 21).

res/values/colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="grey">#cccccc</color>
    <color name="light_blue">#ff64c2f4</color>
</resources>
```

res/drawable/item_selector.xml

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">

    <!-- Палец нажимает на пункт, но пункт ещё не активирован -->
    <item android:state_pressed="true"
        android:drawable="@color/light_blue" />

    <!-- Пункт активирован. В режиме SINGLE_CHOICE_MODE пункт
    подсвечивается -->
    <item android:state_activated="true"
        android:drawable="@color/light_blue" />

    <!-- По умолчанию -->
    <item android:drawable="@android:color/transparent" />
</selector>
```

res/drawable-v21/item_selector.xml

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">

    <!-- Палец нажимает на пункт, но пункт ещё не активирован -->
    <item android:state_pressed="true">
        <ripple android:color="@color/grey" />
    </item>
</selector>
```

```
</item>
```

<!-- Пункт активирован. В режиме SINGLE_CHOICE_MODE пункт подсвечивается -->

```
<item android:state_activated="true"
      android:drawable="@color/light_blue" />
```

<!-- По умолчанию -->

```
<item android:drawable="@android:color/transparent" />
```

```
</selector>
```

Разница заключается в том, что в версии 21 рекомендуется использовать серый цвет с применением **ripple**.

Созданные ресурсы следует применить для фона элемента списка (**list_item.xml**): **android:background="@drawable/item_selector"**.

Для проверки установим режим **singleChoice** для активации выбранного элемента списка.

```
<ListView
```

```
...
```

```
      android:choiceMode="singleChoice" />
```

Обычно режим активации выбранного элемента списка применяют для двухпанельной разметки, а в телефонах такой режим не используют. В таких случаях удобнее создать специальный стиль для списка.

Создадим стиль для планшетов.

res/values-sw600dp/styles.xml

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<resources>
```

```
    <style name="MyListStyle">
```

```
        <item name="android:choiceMode">singleChoice</item>
```

```
    </style>
```

```
</resources>
```

А в обычном **styles.xml** оставим заглушку.

```
<style name="MyListStyle">
```

```
</style>
```


Теперь применим стиль к списку и нужное поведение с активацией будет применяться только на планшетах.

```
<ListView
    android:id="@+id/listView"
    style="@style/MyListStyle"
    ... />
```

При повороте выбранный пункт списка может оказаться за пределами экрана. С помощью метода **smoothScrollToPosition()** мы можем автоматически прокрутить список к нужному месту. Код показан в [продвинутых приёмах](#).

Кнопка под списком

Если вы хотите разместить кнопку под списком, которая бы не зависела от количества элементов в **ListView**, то воспользуйтесь весом (`layout_weight`).

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <ListView
        android:id="@+id/listView1"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1" >
    </ListView>

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="0"
        android:text="Button" />
```

```
</LinearLayout>
```

```
ListView listView = (ListView) findViewById(R.id.listView);
int n = 0; // прокручиваем до начала
listView.smoothScrollToPosition(n);
```

9 лекция.

Тема: Flutter: создайте простое приложение-калькулятор в Android Studio, протестируйте его на эмуляторе и установите на свой мобильный телефон.

План:

1. Основные шаги создания приложения.

Шаг 1

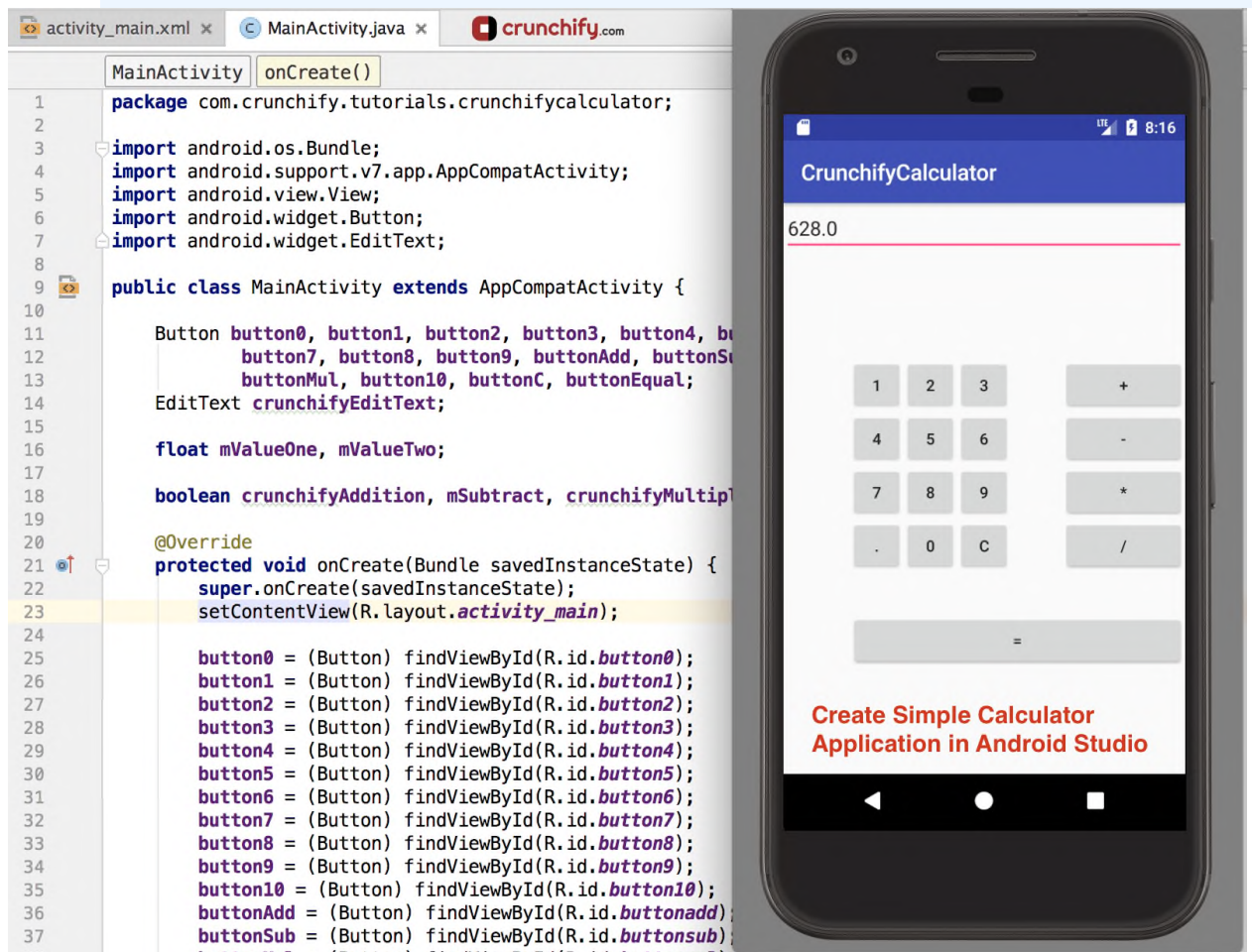
Шаг 2

Шаг 3

Шаг 4

Шаг 5

1. Основные шаги создания приложения.



В моей предыдущей статье я написал подробные инструкции о том, как создать простое приложение для Android. В этом конкретном приложении я также объяснил понятия кнопки Android и основные понятия Android.

Все остальные мои статьи вы можете найти в разделе Android.

В этой статье мы создадим `calculator android app` , Это простой калькулятор с ограниченными функциональными возможностями.

Прежде чем мы продолжим, было бы неплохо пройти полный курс HelloWorld Tutorial. Вот снова ссылка: Мое первое приложение для Android HelloWorld

- Как создать простое приложение-калькулятор — полное руководство
- Создание простого калькулятора с помощью Android Studio
- Разработка Android: создание базового калькулятора
- Создать простое приложение для Android
- Как создать приложение калькулятор для Android

Давайте начнем с нашего приложения для калькулятора Android:

Шаг 1

- Откройте вашу Android Studio
- Нажмите «Начать новый проект Android Studio».
- Дайте название вашей заявки `CrunchifyCalculator` и оставьте другие поля пустыми, как есть, затем нажмите NEXT.

Configure your new project

Application name:

Company domain:

Package name: [Edit](#)

☐ Include C++ support

Project location:

Шаг 2

- Выберите минимальный SDK API 15: Android 4.0.3(IceCreamSandwich) , Я выбрал API 15 (IceCreamSandwich),

потому что он охватывает почти 94% устройств и обладает практически всеми функциями. Если вы хотите покрыть 100% устройств, вы можете выбрать API 8: Android 2.2 (Froyo).

☒ Phone and Tablet

Minimum SDK

Lower API levels target more devices, but have fewer features available.
By targeting API 15 and later, your app will run on approximately **100.0%** of the devices that are active on the Google Play Store.
[Help me choose](#)

Шаг 3

- Выберите `Empty Activity` и нажмите ДАЛЕЕ.
- Оставьте название деятельности `MainActivity` как есть и оставь все как есть. Нажмите Готово.

Creates a new empty activity

Activity Name:

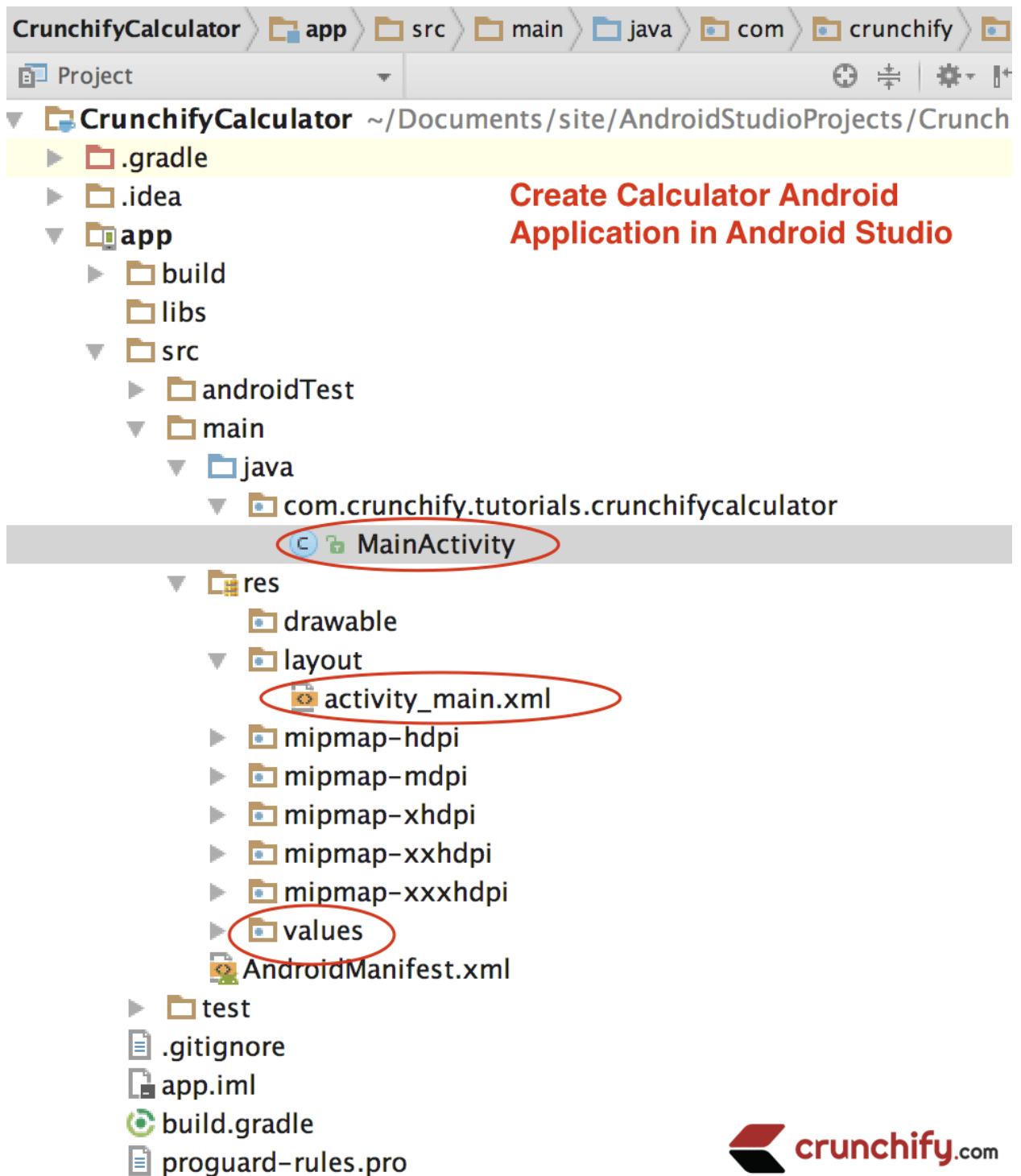
☒ Generate Layout File

Layout Name:

☒ Backwards Compatibility (AppCompat)

Шаг 4

- После нажатия кнопки «Готово» на создание действия и файлов уходит около 2 минут.
- Вот окончательная структура проекта для вашего приложения.



Шаг 5

- Теперь мы должны добавить наш Java-код в наш файл MainActivity.java.
- Так открою тебя MainActivity.java файл с левой стороны IDE (приложение -> java -> com.crunchify.tutorials.crunchifycalculator -> MainActivity.java)

Вы можете найти объяснение выделенной строки под кодом.

MainActivity.java

```

1  пакет ком . crunchify. учебники . хрустящий калькулятор ;
2  импорт андроид . ос . Расслоение ;
3  импорт андроид . поддержка. v7 . приложение AppCompatActivity ;
4  импорт андроид . вид. Вид ;
5  импорт андроид . виджет. Кнопка ;
6  импорт андроид . виджет. EditText ;
7  общественности учебный класс Основная
8  деятельность продолжается AppCompatActivity {
9      Кнопка button0 , button1 , button2 , button3 , button4 , button5 , button6 ,
10         button7 , кнопка8 , button9 , кнопка
11  Добавить , buttonSub , buttonDivision ,
12         buttonMul , кнопка10 , кнопка C , buttonEqual ;
13     EditText crunchifyEditText ;
14     поплавок mValueOne , mValueTwo ;
15     логический crunchifyAddition , mSubtract , crunchifyMultiplication ,
16     crunchifyDivision ;
17     @ Override
18     защищенный недействительным onCreate (Пачка savedInstanceState) {
19         супер. onCreate ( savedInstanceState ) ;
20         setContentView (.. Р компоновка activity_main);
21         button0 знак равно ( Кнопка ) findViewById (.. R ID button0);
22         button1 знак равно ( Кнопка ) findViewById (.. R ID Button1);
23         button2 знак равно ( Кнопка ) findViewById (.. R ID button2);
24         Button3 знак равно ( Кнопка ) findViewById (.. R ID Button3);
25         Button4 знак равно ( Кнопка ) findViewById (.. R ID Button4);
26         button5 знак равно ( Кнопка ) findViewById (.. R ID button5);
27         Button6 знак равно ( Кнопка ) findViewById (.. R ID Button6);
28         Button7 знак равно ( Кнопка ) findViewById (.. R ID Button7);
29         Button8 знак равно ( Кнопка ) findViewById (.. R ID Button8);
30         Button9 знак равно ( Кнопка ) findViewById (.. R ID Button9);
31         Button10 знак равно ( Кнопка ) findViewById (.. R ID Button10);
32         buttonAdd знак равно ( Кнопка ) findViewById (R ID buttonadd.);
33         buttonSub знак равно ( Кнопка ) findViewById (.. R ID buttonsub);
34         buttonMul знак равно ( Кнопка ) findViewById (.. R ID buttonmul);
35         buttonDivision знак равно ( Кнопка ) findViewById (. R ID buttondiv.);
36         buttonC знак равно ( Кнопка ) findViewById (. R ID buttonC.);
37         buttonEqual знак равно ( Кнопка ) findViewById (.. R ID buttoneql);
38         crunchifyEditText знак равно ( EditText ) findViewById (.. R ID edt1);
39         кнопка1 . setOnClickListener ( new Промотр. OnClickListener ( ) {

```



```
40         @ Override
41         общественности недействительным OnClickListener ( Просмотр v ) {
42             crunchifyEditText . setText ( crunchifyEditText . getText ( ) + «1» ) ;
43         }
44     } ) ;
45     кнопка2 . setOnClickListener ( new Просмотр. OnClickListener ( ) {
46         @ Override
47         общественности недействительным OnClickListener ( Просмотр v ) {
48             crunchifyEditText . setText ( crunchifyEditText . getText ( ) + 2 ) ;
49         }
50     } ) ;
51     кнопка3 . setOnClickListener ( new Просмотр. OnClickListener ( ) {
52         @ Override
53         общественности недействительным OnClickListener ( Просмотр v ) {
54             crunchifyEditText . setText ( crunchifyEditText . getText ( ) + «3» ) ;
55         }
56     } ) ;
57     кнопка4 . setOnClickListener ( new Просмотр. OnClickListener ( ) {
58         @ Override
59         общественности недействительным OnClickListener ( Просмотр v ) {
60             crunchifyEditText . setText ( crunchifyEditText . getText ( ) + 4 ) ;
61         }
62     } ) ;
63     кнопка5 . setOnClickListener ( new Просмотр. OnClickListener ( ) {
64         @ Override
65         общественности недействительным OnClickListener ( Просмотр v ) {
66             crunchifyEditText . setText ( crunchifyEditText . getText ( ) + 5 ) ;
67         }
68     } ) ;
69     кнопка6 . setOnClickListener ( new Просмотр. OnClickListener ( ) {
70         @ Override
71         общественности недействительным OnClickListener ( Просмотр v ) {
72             crunchifyEditText . setText ( crunchifyEditText . getText ( ) + «6» ) ;
73         }
74     } ) ;
75     кнопка7 . setOnClickListener ( new Просмотр. OnClickListener ( ) {
76         @ Override
77         общественности недействительным OnClickListener ( Просмотр v ) {
78             crunchifyEditText . setText ( crunchifyEditText . getText ( ) + 7 ) ;
```

```

79         }
80     } );
81     кнопка8 . setOnClickListener ( new Просмотр. OnClickListener () {
82         @ Override
83         общественности недействительным onClick ( Просмотр v ) {
84             crunchifyEditText . setText ( crunchifyEditText . getText ( ) + «8» );
85         }
86     } );
87     кнопка9 . setOnClickListener ( new Просмотр. OnClickListener () {
88         @ Override
89         общественности недействительным onClick ( Просмотр v ) {
90             crunchifyEditText . setText ( crunchifyEditText . getText ( ) + 9 );
91         }
92     } );
93     кнопка0 . setOnClickListener ( new Просмотр. OnClickListener () {
94         @ Override
95         общественности недействительным onClick ( Просмотр v ) {
96             crunchifyEditText . setText ( crunchifyEditText . getText ( ) + «0» );
97         }
98     } );
99     Кнопка Добавить . setOnClickListener ( new Просмотр.
100 OnClickListener () {
101         @ Override
102         общественности недействительным onClick ( Просмотр v ) {
103             если ( crunchifyEditText == ноль ) {
104                 crunchifyEditText . setText ( );
105             } еще {
106                 mValueOne знак
107 равно Поплавок. parseFloat ( crunchifyEditText . getText ( ) + );
108                 crunchifyAddition знак равно правда ;
109                 crunchifyEditText . setText ( null );
110             }
111         }
112     } );
113     ButtonSub . setOnClickListener ( new Просмотр. OnClickListener () {
114         @ Override
115         общественности недействительным onClick ( Просмотр v ) {
116             mValueOne знак
117 равно Поплавок. parseFloat ( crunchifyEditText . getText ( ) + );

```

```

118         mSubtract знак равно правда ;
119         crunchifyEditText . setText ( null ) ;
120     }
121 } ) ;
122 ButtonMul . setOnClickListener ( new Просмотр. OnClickListener ( ) {
123     @ Override
124     общественности недействительным onClick ( Просмотр v ) {
125         mValueOne знак
126 равно Поплавок. parseFloat ( crunchifyEditText . getText ( ) + ) ;
127         crunchifyMultiplication знак равно правда ;
128         crunchifyEditText . setText ( null ) ;
129     }
130 } ) ;
131 ButtonDivision . setOnClickListener ( new Просмотр. OnClickListener ( ) {
132     @ Override
133     общественности недействительным onClick ( Просмотр v ) {
134         mValueOne знак
135 равно Поплавок. parseFloat ( crunchifyEditText . getText ( ) + ) ;
136         crunchifyDivision знак равно правда ;
137         crunchifyEditText . setText ( null ) ;
138     }
139 } ) ;
140 КнопкаEqual . setOnClickListener ( new Просмотр. OnClickListener ( ) {
141     @ Override
142     общественности недействительным onClick ( Просмотр v ) {
143         mValueTwo знак
144 равно Поплавок. parseFloat ( crunchifyEditText . getText ( ) + ) ;
145         если ( crunchifyAddition == правда ) {
146             crunchifyEditText . setText ( mValueOne + mValueTwo + ) ;
147             crunchifyAddition знак равно ложь ;
148         }
149         если ( mSubtract == правда ) {
150             crunchifyEditText . setText ( mValueOne — mValueTwo + ) ;
151             mSubtract знак равно ложь ;
152         }
153         если ( crunchifyMultiplication == правда ) {
154             crunchifyEditText . setText ( mValueOne * mValueTwo + ) ;
155             crunchifyMultiplication знак равно ложь ;
156         }

```

```

157         если ( crunchifyDivision == правда ) {
158             crunchifyEditText . setText ( mValueOne / mValueTwo + ) ;
159             crunchifyDivision знак равно ложь ;
160         }
161     }
162 } ) ;
163 buttonC. setOnClickListener ( new Просмотр. OnClickListener ( ) {
164     @ Override
165     общественности недействительным onClick ( Просмотр v ) {
166         crunchifyEditText . setText ( ) ;
167     }
168 } ) ;
169 кнопка10 . setOnClickListener ( new Просмотр. OnClickListener ( ) {
170     @ Override
171     общественности недействительным onClick ( Просмотр v ) {
172         crunchifyEditText . setText ( crunchifyEditText . getText ( ) + ) ;
173     }
174 } ) ;
175 }
176 }

```

Здесь у нас есть 1 EditText. Он определяет тип контента.

Давайте разберемся в коде немного больше.

- Строка 11 — 14: Здесь мы создали ссылку на кнопки и EditText.
- Строка 16: здесь мы создали две переменные типа float для значений 1 и 2.
- Строка 21: мы переопределяем метод onCreate (), который является методом класса Activity .
- Строка 45 — 50: мы устанавливаем onClickListener на Button1. Если мы нажмем кнопку1, EditText будет отображаться.
- Мы реализовали одинаковую логику для каждой кнопки.
- Строка 115 — 127: Здесь мы установили прослушиватель щелчков на кнопке «Добавить».
- Здесь мы ставим условие так: если EditText равен Null, тогда мы устанавливаем EditText как пустое значение. В противном случае мы добавляем два значения, которые нажимаются до нажатия кнопки добавления и после нажатия кнопки добавления.

- Мы также установили `crunchifyAddition` Логическое значение `True`. Это верно означает, что кнопка добавления нажата, и она будет использоваться, когда пользователь нажимает кнопку «=».
- Мы реализуем ту же логику для других кнопок, таких как `buttonSub`, `ButtonMul`, `buttonDivision`.
- Строка 156 — 183: здесь мы устанавливаем `clickListener` на кнопку «=». Здесь мы ставим условие, как будто пользователь нажимает кнопку Добавить `crunchifyAddition` Значение установлено в `True` для прослушивания щелчка кнопки «Добавить».
- В соответствии с этим будет выполнено соответствующее действие, соответствующее нажатию кнопки.

```

1  если ( crunchifyAddition == правда ) {
2      crunchifyEditText . setText ( mValueOne + mValueTwo + ) ;
3      crunchifyAddition знак равно ложь ;
4  }
```

если кнопка «Добавить» нажата до кнопки «=», то действие «Добавить» будет выполнено, как показано выше.

- После выполненного действия мы устанавливаем `crunchifyAddition` значение `false`, чтобы мы могли снова выполнить действие `Add`.

Ниже приведен файл макета, с помощью которого можно создать внешний интерфейс для калькулятора:

Activity_Main.xml

```

1  xml version = 1.0 encoding = utf-8 ?>
2  < RelativeLayout xmlns : android = http://schemas.android.com/apk/res/android
3      xmlns : tools = http://schemas.android.com/tools
4      Android : ID = @ + ID / родственник1
5      android : layout_width = match_parent
6      android : layout_height = match_parent
7      tools : context = .MainActivity >
8      < EditText
9          android : id = @ + id / edt1
10         android : layout_width = match_parent
11         android : layout_height = wrap_content / >
12     < Кнопка
13         android : id = @ + id / button1
14         style = ? android: attr / buttonStyleSmall
15         android : layout_width = wrap_content
```

```

16      android : layout_height = wrap_content
17      android : layout_alignEnd = @ + id / button4
18      android : layout_alignRight = @ + id / button4
19      android : layout_below = @ + id / edt1
20      android : layout_marginTop = 94dp
21      android : text = 1 / >
22  < Кнопка
23      android : id = @ + id / button2
24      style = ? android: attr / buttonStyleSmall
25      android : layout_width = wrap_content
26      android : layout_height = wrap_content
27      android : layout_alignTop = @ + id / button1
28      android : layout_toLeftOf = @ + id / button3
29      android : layout_toStartOf = @ + id / button3
30      android : text = 2 / >
31  < Кнопка
32      android : id = @ + id / button3
33      style = ? android: attr / buttonStyleSmall
34      android : layout_width = wrap_content
35      android : layout_height = wrap_content
36      android : layout_alignTop = @ + id / button2
37      android : layout_centerHor horizontal = true
38      android : text = 3 / >
39  < Кнопка
40      android : id = @ + id / button4
41      style = ? android: attr / buttonStyleSmall
42      android : layout_width = wrap_content
43      android : layout_height = wrap_content
44      android : layout_below = @ + id / button1
45      android : layout_toLeftOf = @ + id / button2
46      android : text = 4 / >
47  < Кнопка
48      android : id = @ + id / button5
49      style = ? android: attr / buttonStyleSmall
50      android : layout_width = wrap_content
51      android : layout_height = wrap_content
52      android : layout_alignBottom = @ + id / button4
53      android : layout_alignLeft = @ + id / button2
54      android : layout_alignStart = @ + id / button2

```



```

55     android : text = 5 / >
56 < Кнопка
57     android : id = @ + id / button6
58     style = ? android: attr / buttonStyleSmall
59     android : layout_width = wrap_content
60     android : layout_height = wrap_content
61     android : layout_alignLeft = @ + id / button3
62     android : layout_alignStart = @ + id / button3
63     android : layout_below = @ + id / button3
64     android : text = 6 / >
65 < Кнопка
66     android : id = @ + id / button7
67     style = ? android: attr / buttonStyleSmall
68     android : layout_width = wrap_content
69     android : layout_height = wrap_content
70     android : layout_below = @ + id / button4
71     android : layout_toLeftOf = @ + id / button2
72     android : text = 7 / >
73 < Кнопка
74     android : id = @ + id / button8
75     style = ? android: attr / buttonStyleSmall
76     android : layout_width = wrap_content
77     android : layout_height = wrap_content
78     android : layout_alignLeft = @ + id / button5
79     android : layout_alignStart = @ + id / button5
80     android : layout_below = @ + id / button5
81     android : text = 8 / >
82 < Кнопка
83     android : id = @ + id / button9
84     style = ? android: attr / buttonStyleSmall
85     android : layout_width = wrap_content
86     android : layout_height = wrap_content
87     android : layout_alignLeft = @ + id / button6
88     android : layout_alignStart = @ + id / button6
89     android : layout_below = @ + id / button6
90     android : text = 9 / >
91 < Кнопка
92     android : id = @ + id / buttonadd
93     style = ? android: attr / buttonStyleSmall

```

```

94     android : layout_width = wrap_content
95     android : layout_height = wrap_content
96     android : layout_alignEnd = @ + id / edt1
97     android : layout_alignRight = @ + id / edt1
98     android : layout_alignTop = @ + id / button3
99     android : layout_marginLeft = 46dp
100    android : layout_marginStart = 46dp
101    android : layout_toRightOf = @ + id / button3
102    android : text = + / >
103    < Кнопка
104        android : id = @ + id / buttonsub
105        style = ? android: attr / buttonStyleSmall
106        android : layout_width = wrap_content
107        android : layout_height = wrap_content
108        android : layout_alignEnd = @ + id / buttonadd
109        android : layout_alignLeft = @ + id / buttonadd
110        android : layout_alignRight = @ + id / buttonadd
111        android : layout_alignStart = @ + id / buttonadd
112        android : layout_below = @ + id / buttonadd
113        android : text = — / >
114    < Кнопка
115        android : id = @ + id / buttonmul
116        style = ? android: attr / buttonStyleSmall
117        android : layout_width = wrap_content
118        android : layout_height = wrap_content
119        android : layout_alignLeft = @ + id / buttonsub
120        android : layout_alignParentEnd = true
121        android : layout_alignParentRight = true
122        android : layout_alignStart = @ + id / buttonsub
123        android : layout_below = @ + id / buttonsub
124        android : text = * / >
125    < Кнопка
126        android : id = @ + id / button10
127        style = ? android: attr / buttonStyleSmall
128        android : layout_width = wrap_content
129        android : layout_height = wrap_content
130        android : layout_below = @ + id / button7
131        android : layout_toLeftOf = @ + id / button2
132        android : text = . / >

```

```
133 < Кнопка
134     android : id = @ + id / button0
135     style = ? android: attr / buttonStyleSmall
136     android : layout_width = wrap_content
137     android : layout_height = wrap_content
138     android : layout_alignLeft = @ + id / button8
139     android : layout_alignStart = @ + id / button8
140     android : layout_below = @ + id / button8
141     android : text = / >
142 < Кнопка
143     android : id = @ + id / buttonC
144     style = ? android: attr / buttonStyleSmall
145     android : layout_width = wrap_content
146     android : layout_height = wrap_content
147     android : layout_alignLeft = @ + id / button9
148     android : layout_alignStart = @ + id / button9
149     android : layout_below = @ + id / button9
150     android : text = C / >
151 < Кнопка
152     android : id = @ + id / buttondiv
153     style = ? android: attr / buttonStyleSmall
154     android : layout_width = wrap_content
155     android : layout_height = wrap_content
156     android : layout_alignEnd = @ + id / buttonmul
157     android : layout_alignLeft = @ + id / buttonmul
158     android : layout_alignRight = @ + id / buttonmul
159     android : layout_alignStart = @ + id / buttonmul
160     android : layout_below = @ + id / buttonmul
161     android : text = / / >
162 < Кнопка
163     android : id = @ + id / buttoneql
164     android : layout_width = wrap_content
165     android : layout_height = wrap_content
166     android : layout_alignEnd = @ + id / buttondiv
167     android : layout_alignLeft = @ + id / button10
168     android : layout_alignRight = @ + id / buttondiv
169     android : layout_alignStart = @ + id / button10
170     android : layout_below = @ + id / button0
171     android : layout_marginTop = 37dp
```

```
172         android : text == / >
173     < / RelativeLayout >
```



Теперь все должно работать нормально, и мы готовы запустить наше приложение-калькулятор для Android. Для запуска нашего приложения я использовал свой мобильный, вы можете использовать эмулятор или ваше устройство.

Запуск нашего калькулятора для Android

- Нажмите на диспетчер устройств Android. После выбора вашего пользовательского устройства в **Android device manager** окно, нажмите **START**,
- Нажмите на кнопку «Выполнить».
- Выберите Ваше устройство или эмулятор и нажмите ОК.
- Теперь вы можете увидеть калькулятор для Android, работающий как этот скриншот .

10 лекция.

Тема: Flutter: Работа с базами данных в Android Studio, работа с SQLite, MySQL.

План:

1. Конфигурация проекта
2. Создание простой модели
3. Класс базы данных
4. Основной файл приложения

Вместо случайного смешивания логики базы данных в приложении основные методы обработки базы данных помещены в `lib/services/db.dart` для удобства и простоты обслуживания:

```
db.dart

import 'dart:async';
import 'package:flutter_sqlite_demo/models/model.dart';
import 'package:sqflite/sqflite.dart';

abstract class DB {

  static Database _db;

  static int get _version => 1;

  static Future<void> init() async {

    if (_db != null) { return; }

    try {
```

```

    String _path = await getDatabasesPath() + 'example';
    _db = await openDatabase(_path, version: _version, onCreate: onCreate);
  }
  catch(ex) {
    print(ex);
  }
}

static void onCreate(Database db, int version) async =>
  await db.execute('CREATE TABLE todo_items (id INTEGER PRIMARY
KEY NOT NULL, task STRING, complete BOOLEAN)');

static Future<list<map<string, dynamic="">>> query(String table) async =>
_db.query(table);

static Future<int> insert(String table, Model model) async =>
  await _db.insert(table, model.toMap());

static Future<int> update(String table, Model model) async =>
  await _db.update(table, model.toMap(), where: 'id = ?', whereArgs:
[model.id]);

static Future<int> delete(String table, Model model) async =>
  await _db.delete(table, where: 'id = ?', whereArgs: [model.id]);
}
</int></int></int></list<map<string,></void>

```

Этот **abstract** класс, поскольку он не предназначен для создания экземпляра, и требуется только одна его копия в памяти. Внутренне он содержит ссылку на базу данных SQLite в свойстве **_db**. Номер версии базы данных жестко

запрограммирован (1), но в более сложных приложениях версию базы данных можно использовать для переноса схем базы данных вверх или вниз по версии, чтобы обеспечить развертывание новых функций без необходимости стирать базу данных и начинать с нуля.

Экземпляр базы данных SQLite создается в методе `init` с использованием имени базы данных для этого проекта `example`. Если база данных `example` еще не существует, автоматически вызывается `onCreate`. Здесь размещаются запросы на создание структуры таблицы. В этом случае у нас есть таблица `todo_items` с первичным ключом `id`, а также поля, соответствующие свойствам в приведенном выше классе `TodoItem`.

Метод `query` наряду с `insert`, `update` и `delete` определены для выполнения стандартных операций CRUD в базе данных. Они предоставляют простые абстракции и позволяют содержать логику базы данных в этом классе, что может быть чрезвычайно полезно при рефакторинге или выполнении другого обслуживания приложения вместо того, чтобы, например, выполнять поиск и замену строк в нескольких файлах или исправлять странные ошибки, которые появляются при создании простые изменения

На этой лекции мы рассмотрим, как использовать SQLite во Flutter с пакетом `sqflite` для локального хранения данных приложения. SQLite существует с 2000 года и является популярным выбором для встраивания баз данных в локальные приложения. Для примера проекта мы создадим очень простое приложение «TODO», которое может создавать, обновлять и удалять элементы TODO из базового интерфейса.

Если у вас еще нет Flutter, вы можете получить копию со страницы установки:

Install - Flutter

Select the operating system on which you are installing Flutter:{{site.alert.note}}
Are you on Chrome OS? If so, see the official [Chrome OS Flutter installation docs!](/docs/get-started/install/chromeos){{site.alert.end}}
<https://flutter.dev/docs/get-started/install>

Исходный код, используемый в этой статье, доступен на GitHub:

GitHub - kenreilly/flutter-sqlite-demo: Example project demonstrating how to use Flutter with SQLite

Example project demonstrating how to use Flutter with SQLite - kenreilly/flutter-sqlite-demo

1. Конфигурация проекта

Чтобы использовать SQLite в приложении Flutter, первым шагом является включение пакета **sqflite** в pubspec.yaml проекта, например, так:

```
pubspec.yaml
name: flutter_sqlite_demo
description: Example project demonstrating how to use Flutter with SQLite
version: 1.0.0+1

environment:
  sdk: ">=2.1.0 <3.0.0"

dependencies:
  flutter:
    sdk: flutter
  sqflite: ^1.2.0
  path_provider: ^1.6.0
  cupertino_icons: ^0.1.2
```

```
dev_dependencies:  
  flutter_test:  
    sdk: flutter  
  
flutter:  
  uses-material-design: true
```

Здесь мы указали **sqflite** версию **1.2.0** или выше и **path_provider** на **1.6.0** или выше. Помимо этого, проект остается простым, чтобы с ним было легко работать и понимать.

2. Создание простой модели

Для хранения данных простой класс модели данных предоставит необходимые методы для преобразования между дружественным к SQLite форматом данных и объектом, который можно использовать в приложении. Во-первых, абстрактный класс **Model** будет служить базовым классом для моделей данных. Этот файл находится в `lib/models/model.dart`:

```
model.dart  
  
abstract class Model {  
  
  int id;  
  
  static fromMap() {}  
  toMap() {}  
}
```

Класс **Model** очень прост и создан для удобства, чтобы определить свойства / методы, которые можно ожидать от моделей данных, например **id**, как показано выше. Это позволяет создавать одну или несколько конкретных моделей данных, которые будут соответствовать этому базовому шаблону

проектирования. Для этого приложения класс модели элементов TODO создается в lib/models/todo-item.dart:

```
todo-item.dart

import 'package:flutter_sqlite_demo/models/model.dart';

class TodoItem extends Model {

  static String table = 'todo_items';

  int id;
  String task;
  bool complete;

  TodoItem({ this.id, this.task, this.complete });

  Map<string, dynamic> toMap() {

    Map<string, dynamic> map = {
      'task': task,
      'complete': complete
    };

    if (id != null) { map['id'] = id; }
    return map;
  }

  static TodoItem fromMap(Map<string, dynamic> map) {

    return TodoItem(
```

```

        id: map['id'],
        task: map['task'],
        complete: map['complete'] == 1
    );
}
}
</string,></string,></string,>

```

Класс `TodoItem` содержит свойства `task` и `complete` а также имеет простой конструктор для создания нового элемента `TODO`. Для преобразования между экземплярами объектов `TodoItem` и `Map`, используемых базой данных, были определены методы `toMap` и `fromMap`.

3. Класс базы данных

Вместо случайного смешивания логики базы данных в приложении основные методы обработки базы данных помещены в `lib/services/db.dart` для удобства и простоты обслуживания:

```

db.dart
import 'dart:async';
import 'package:flutter_sqlite_demo/models/model.dart';
import 'package:sqflite/sqflite.dart';

abstract class DB {

    static Database _db;

    static int get _version => 1;

    static Future<void> init() async {

```

```

    if (_db != null) { return; }

    try {
        String _path = await getDatabasesPath() + 'example';
        _db = await openDatabase(_path, version: _version, onCreate: onCreate);
    }
    catch(ex) {
        print(ex);
    }
}

static void onCreate(Database db, int version) async =>
    await db.execute('CREATE TABLE todo_items (id INTEGER PRIMARY
KEY NOT NULL, task STRING, complete BOOLEAN)');

static Future<list<map<string, dynamic="">>> query(String table) async =>
_db.query(table);

static Future<int> insert(String table, Model model) async =>
    await _db.insert(table, model.toMap());

static Future<int> update(String table, Model model) async =>
    await _db.update(table, model.toMap(), where: 'id = ?', whereArgs:
[model.id]);

static Future<int> delete(String table, Model model) async =>
    await _db.delete(table, where: 'id = ?', whereArgs: [model.id]);
}
</int></int></int></list<map<string,></void>

```


Этот **abstract** класс, поскольку он не предназначен для создания экземпляра, и требуется только одна его копия в памяти. Внутренне он содержит ссылку на базу данных SQLite в свойстве **_db**. Номер версии базы данных жестко запрограммирован (1), но в более сложных приложениях версию базы данных можно использовать для переноса схем базы данных вверх или вниз по версии, чтобы обеспечить развертывание новых функций без необходимости стирать базу данных и начинать с нуля.

Экземпляр базы данных SQLite создается в методе **init** с использованием имени базы данных для этого проекта **example**. Если база данных **example** еще не существует, автоматически вызывается **onCreate**. Здесь размещаются запросы на создание структуры таблицы. В этом случае у нас есть таблица **todo_items** с первичным ключом **id**, а также поля, соответствующие свойствам в приведенном выше классе **TodoItem**.

Метод **query** наряду с **insert**, **update** и **delete** определены для выполнения стандартных операций CRUD в базе данных. Они предоставляют простые абстракции и позволяют содержать логику базы данных в этом классе, что может быть чрезвычайно полезно при рефакторинге или выполнении другого обслуживания приложения вместо того, чтобы, например, выполнять поиск и замену строк в нескольких файлах или исправлять странные ошибки, которые появляются при создании простых изменениях.

4. Основной файл приложения

И последнее, но не менее важное: у нас есть основная логика приложения и UX в **lib/main.dart**:

```
lib/main.dart
import 'package:flutter/material.dart';
import 'package:flutter_sqlite_demo/models/todo-item.dart';
import 'package:flutter_sqlite_demo/services/db.dart';
```

```
void main() async {

  WidgetsFlutterBinding.ensureInitialized();

  await DB.init();
  runApp(MyApp());
}

class MyApp extends StatelessWidget {

  @override
  Widget build(BuildContext context) {

    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData( primarySwatch: Colors.indigo ),
      home: MyHomePage(title: 'Flutter SQLite Demo App'),
    );
  }
}

class MyHomePage extends StatefulWidget {

  MyHomePage({ Key key, this.title }) : super(key: key);

  final String title;

  @override
  _MyHomePageState createState() => _MyHomePageState();
}
```

```

}

class _MyHomePageState extends State<myhomepage> {

  String _task;

  List<todoitem> _tasks = [];

  TextStyle _style = TextStyle(color: Colors.white, fontSize: 24);

  List<widget> get _items => _tasks.map((item) => format(item)).toList();

  Widget format(TodoItem item) {

    return Dismissible(
      key: Key(item.id.toString()),
      child: Padding(
        padding: EdgeInsets.fromLTRB(12, 6, 12, 4),
        child: FlatButton(
          child: Row(
            mainAxisAlignment: MainAxisAlignment.spaceBetween,
            children: <widget>[
              Text(item.task, style: _style),
              Icon(item.complete == true ? Icons.radio_button_checked :
Icons.radio_button_unchecked, color: Colors.white)
            ]
          ),
          onPressed: () => _toggle(item),
        )
      ),
    );
  }
}

```

```
        onDismissed: (DismissDirection direction) => _delete(item),
    );
}

void _toggle(TodoItem item) async {

    item.complete = !item.complete;
    dynamic result = await DB.update(TodoItem.table, item);
    print(result);
    refresh();
}

void _delete(TodoItem item) async {

    DB.delete(TodoItem.table, item);
    refresh();
}

void _save() async {

    Navigator.of(context).pop();
    TodoItem item = TodoItem(
        task: _task,
        complete: false
    );

    await DB.insert(TodoItem.table, item);
    setState(() => _task = " ");
    refresh();
}
```

```

void _create(BuildContext context) {

  showDialog(
    context: context,
    builder: (BuildContext context) {
      return AlertDialog(
        title: Text("Create New Task"),
        actions: <widget>[
          FlatButton(
            child: Text('Cancel'),
            onPressed: () => Navigator.of(context).pop()
          ),
          FlatButton(
            child: Text('Save'),
            onPressed: () => _save()
          )
        ],
        content: TextField(
          autofocus: true,
          decoration: InputDecoration(labelText: 'Task Name', hintText: 'e.g.
pick up bread'),
          onChanged: (value) { _task = value; },
        ),
      );
    }
  );
}

```

@override

```
void initState() {
```

```
  refresh();
```

```
  super.initState();
```

```
}
```

```
void refresh() async {
```

```
  List<map<string, dynamic>>> _results = await DB.query(TodoItem.table);
```

```
  _tasks = _results.map((item) => TodoItem.fromMap(item)).toList();
```

```
  setState(() { });
```

```
}
```

```
@override
```

```
Widget build(BuildContext context) {
```

```
  return Scaffold(
```

```
    backgroundColor: Colors.black,
```

```
    appBar: AppBar( title: Text(widget.title) ),
```

```
    body: Center(
```

```
      child: ListView( children: _items )
```

```
    ),
```

```
    floatingActionButton: FloatingActionButton(
```

```
      onPressed: () { _create(context); },
```

```
      tooltip: 'New TODO',
```

```
      child: Icon(Icons.library_add),
```

```
    )
```

```
  );
```

```
}
```

```
}
```

```
</map<string,></widget></widget></widget></todoitem></myhomepage>
```

Этот файл является стандартным для любого приложения Flutter и определяет базовый внешний вид приложения и его взаимодействия. Во время инициализации строка `WidgetsFlutterBinding.ensureInitialized()` обеспечит правильную инициализацию приложения Flutter при инициализации базы данных с помощью `await DB.init()`.

Когда приложение запускается и виджет `MyHomePage` отображается, делается вызов списка задач с помощью `refresh()` из таблицы `todo_items` и сопоставление его с одним из объектов `TodoItem`. Они будут предоставляться в рамках основного `ListView` в приложении через аксессор `_items`, который принимает `List` из объектов `TodoItem` и форматирует его в списке виджетов, содержащего текст TODO элемента и индикатор, показывающий, была ли завершена задача.

Задачи могут быть добавлены путем нажатия плавающей кнопки с действием и ввода имени задачи. Когда нажата `Save`, создается элемент списка с помощью `DB.insert`. Задача добавляется в базу данных с помощью щелчка по задаче в списке, который переключает логическое значение `complete` и сохраняя измененный объект обратно в базу данных с `DB.update`. Свайп по горизонтали на задаче удалит элемент TODO, предоставив его методу `DB.delete`. Всякий раз, когда в список вносятся изменения, обращение к `refresh()` и последующему `setState()` гарантирует, что список обновляется должным образом.

11 лекция.

Тема: Flutter: научитесь работать с API на примере мобильного приложения с запущенным ботом Telegram в Android Studio

План:

1. Разработка прототипа
2. Генерация SSL сертификата
3. Автоматическое создание бота

1. Разработка прототипа

Изучение темы создания Telegram ботов по [официальной документации](#) и по примерам. В основном все примеры были написаны на Python. Поэтому не долго думая, модно искать способы запуска Python сервера на Android. Но оценив время на изучение Python и не найдя ничего подходящего для запуска сервера, занялся поиском альтернатив и наткнулся на несколько библиотек на Java для написания Telegram ботов. В итоге остановился на проекте от Pengrad: [java-telegram-bot-api](#).

Данная библиотека позволяла, на тот момент, инициализировать бота и получать-отправлять сообщения, что мне было и нужно. Добавив библиотеку в свой проект, я реализовал простой сервис, который запускал в фоновом потоке цикл по получению сообщений из Telegram и их обработке. Предварительно необходимо было зарегистрировать нового бота через родительский бот @Botfather и получить его токен. Подробнее о создании бота [по ссылке](#).

Для того, чтобы сервис не убивался системой, когда устройство находится с выключенным экраном, при запуске сервиса, устанавливался WakeLock.

Приведу в пример функцию, позволяющую получать последние сообщения и отправлять их на обработку:

```
private void getUpdates(final TelegramBot bot)
```

Позже, в целях безопасности, я добавил возможность привязки бота к разрешенным Telegram-аккаунтам и возможность запрета выполнения

определенных команд для заданных пользователей.

Добавив несколько команд для бота, такие как: отправка, чтение СМС, просмотр пропущенных звонков, информация о батарее, определение местоположения и др., я опубликовал приложение в Google Play, создал темы на нескольких форумах, стал ждать комментарии и отзывы.

В основном отзывы были хорошие, но вскрылась проблема большого расхода батареи, что, как вы могли догадаться, было связано с WakeLock и постоянной активностью сервиса. Немного погуглив, решил периодически запускать сервис через AlarmManager, затем после получения сообщений и ответа на них сервис останавливать.

Это немного помогло, но появилась другая проблема, AlarmManager некорректно работал на некоторых китайских устройствах. И поэтому бот иногда не просыпался после нескольких часов, проведенных в состоянии сна. Изучая официальную документацию, я читал о том, что Long Polling это не единственная возможность получения сообщений, сообщения еще можно было получать используя Webhook.

Получение сообщений через Webhook

Я зарегистрировался на [Digital Ocean](#), создал VPS на Ubuntu, затем реализовал простейший http сервер на Java, использующий [Spark Framework](#). На сервер можно делать запросы 2 типов: push (отправка пуш-уведомления через webhook) и ping.

Пуш-нотификации отправлялись с помощью Google Firebase.

2. Генерация SSL сертификата

Протестировав отправки пуш-уведомлений, я стал разбираться с тем, как настроить и запустить сервер с HTTPS, так как это одно из требований при получении сообщений из Telegram через webhook.

Бесплатный сертификат можно сгенерировать с помощью сервиса [letsencrypt.org](#), но одним из ограничений является то, что указываемый хост при генерации сертификата не может быть ip адресом. Регистрировать

доменное имя я пока не хотел, тем более официальная документация Telegram Bot API [разрешает](#) использование самоподписанных сертификатов, поэтому я стал разбираться, как создать свой сертификат.

После нескольких часов, проведенных в попытках и поисках, получился скрипт, позволяющий сгенерировать нужный сертификат.

`create_cert.sh`

После запуска скрипта, на выходе получаем два файла: `keystore.jks` — используется на сервере, `public_cert.pem` — используется при установке webhook в Android приложении.

Для того, чтобы запустить HTTPS на Spark Framework достаточно добавить 2 строки, одну указывающую порт (разрешенные порты для webhook: 443, 80, 88, 8443), другую, указывающую сгенерированный сертификат и пароль к нему:

```
port(8443);
secure("keystore.jks", "password", null, null);
```

Чтобы установить webhook для бота, необходимо добавить в андроид-приложение следующие строки:

```
SetWebhook setWebHook = new SetWebhook().url(WEBHOOK_URL + "/" +
pushToken + "/" + secret).certificate(getCert(context));
BaseResponse res = bot.execute(setWebHook);
```

При регистрации webhook, в качестве URL указывается адрес webhook, затем передается пуш-токен, необходимый для отправки пуш-уведомлений и секретный ключ, генерируемый на устройстве, который я добавил для дополнительной проверки входящих уведомлений.

Функция чтения публичного сертификата из RAW ресурса:

```
private static byte[] getCert(Context context) throws IOException {  
    return  
    IOUtils.toByteArray(context.getResources().openRawResource(R.raw.public_cert)  
    );  
}
```

После модификации сервиса по обработке сообщений в Android приложении, бот стал расходовать батарею намного меньше, но и добавилась зависимость работы приложения от сервера пуш-нотификаций, что было необходимо для стабильной работы приложения.

3. Автоматическое создание бота

После обновления механизма получения сообщений, осталась еще одна проблема, которая не позволяла пользоваться приложением некоторому проценту пользователей из-за сложности создания бота через BotFather. Поэтому я решил автоматизировать этот процесс.

В этом мне помогла библиотека [tdlib](#) от создателей Telegram. К сожалению, я нашел очень мало примеров использования этой библиотеки, но разобравшись в API, оказалось, что не так все сложно. В итоге удалось реализовать авторизацию в Telegram по номеру телефона, добавление @Botfather в список контактов и отправку и получение сообщений заданному контакту, а в конкретном случае, боту @Botfather.

Добавление новых возможностей

После решения первостепенных проблем с автономностью, я занялся добавлением новых команд.

В итоге были добавлены такие команды как: фото, запись видео, диктофон, скриншот экрана, управление плеером, запуск избранных приложений и т.д. Для удобного запуска команд, добавил Telegram-клавиатуру и разбил команды по категориям.

По просьбам пользователей, я также добавил возможность вызова команд Tasker и отправки сообщений из Tasker в Telegram.

12 лекция.

Тема: Flutter: Полноценное мобильное приложение в Android Studio: создаем простой мессенджер

План:

1. Создание простого мессенджера

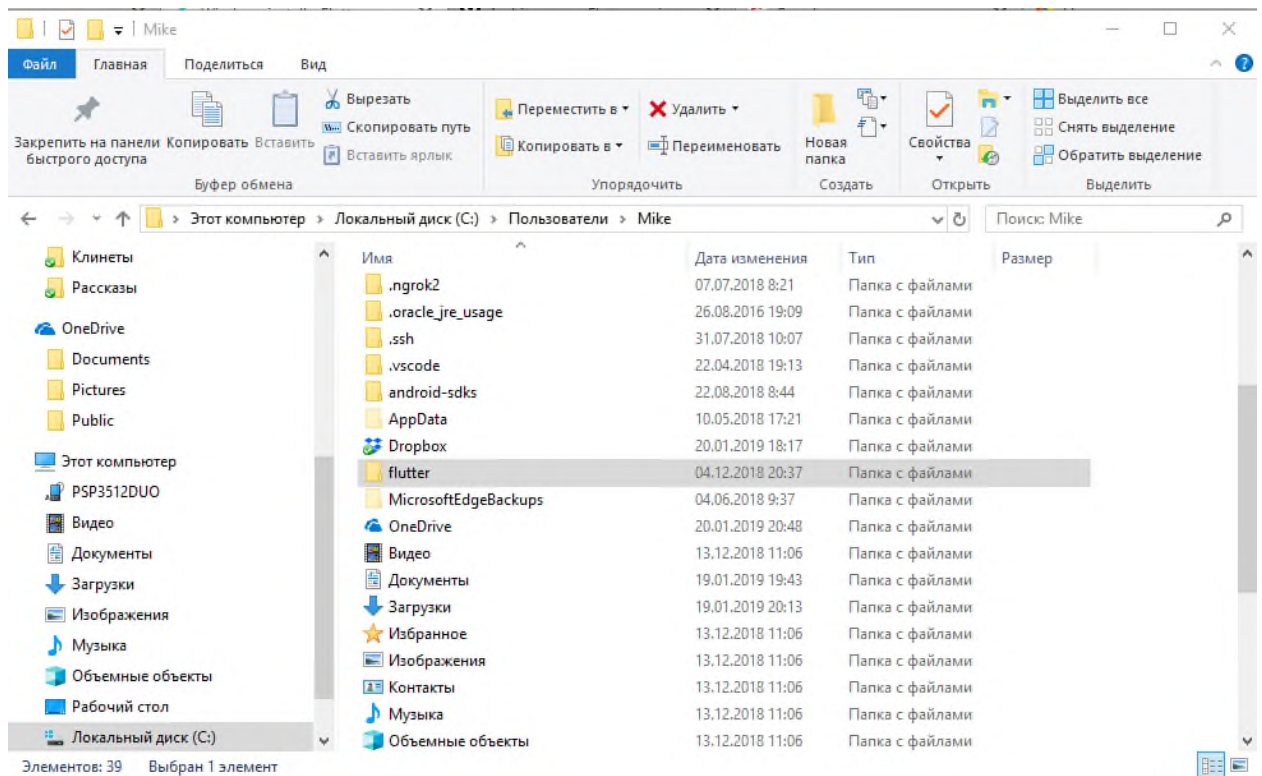
Поэтому, если вы изучите **Dart** и **Flutter**, вы сможете писать нативные приложения для двух самых популярных мобильных операционных систем, т.е. быть одновременно Android и iOS разработчиком. И это, вероятно, ещё не предел, поскольку было объявлено, что Google ведёт работу по расширению Flutter на предмет возможности создания с его помощью приложений для Windows, Mac и Web (проект Flutter для Web носит название Hummingbird — Колибри). В итоге может получиться так, что зная Dart и Flutter вы сможете писать всё очень многое. Поэтому многие IT эксперты называли 2018 год —
годом Flutter.

Теперь за дело. Сейчас мы сделаем следующее

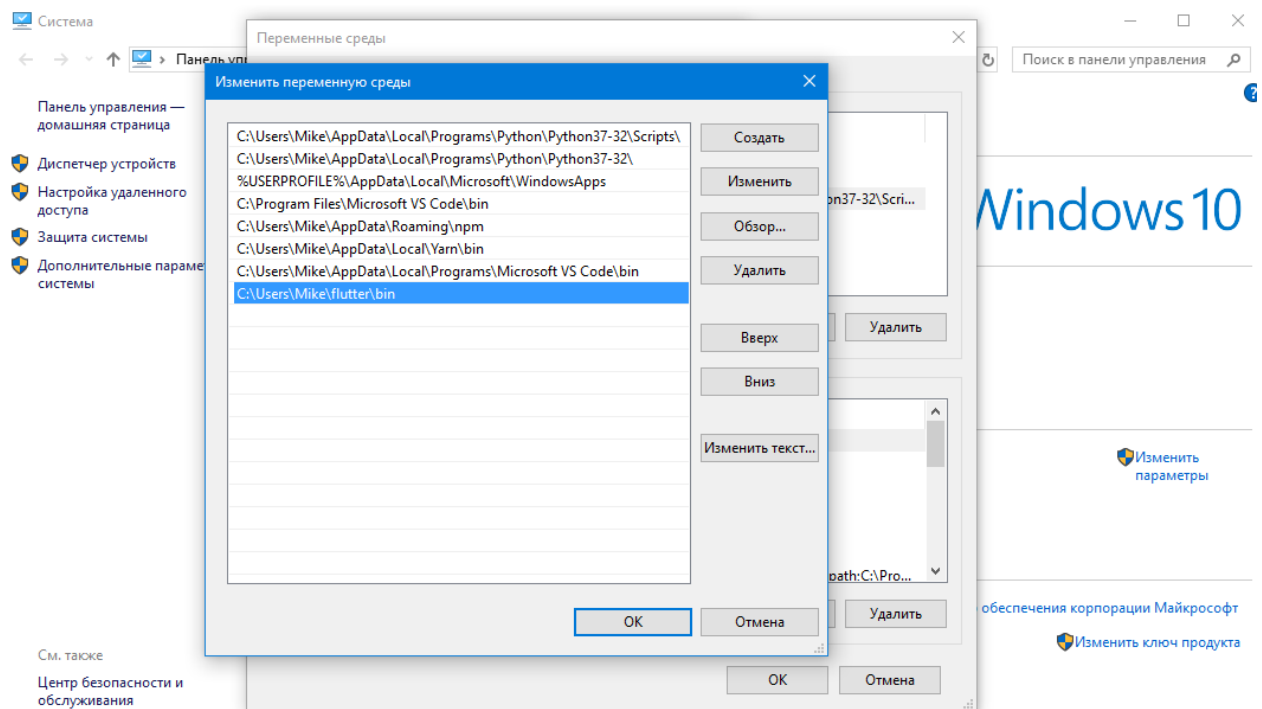
1. Установим Flutter SDK, и создадим проект из командной строки Windows.
2. Установим, JDK, Android Studio, плагины для Flutter и Dart, и создадим проект в Android Studio.
3. Установим VS Code, расширения для Flutter и Dart, и создадим проект в VS Code.

Устанавливаем Flutter и создаём проект из командной строки

Переходим на [страницу установки Flutter](#), выбираем свою операционную систему — Windows, Mac или Linux (здесь будет описано для Windows 10, как наиболее популярной ОС), и скачиваем zip файл, содержащий Flutter SDK. Затем распаковываем zip, например, в папку текущего пользователя, как показано на скриншоте:

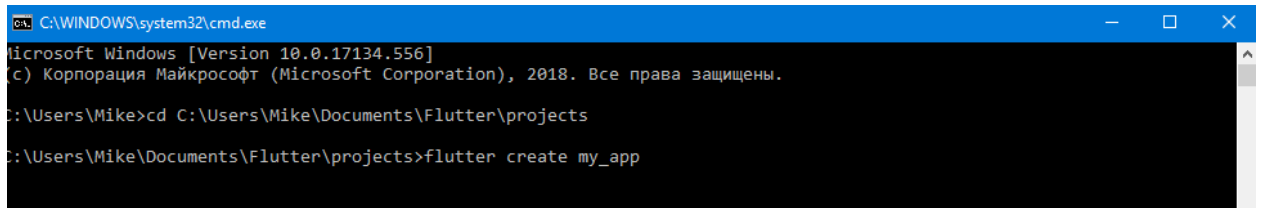


Сейчас пропишем путь к `flutter\bin` в переменную `Path` среды пользователя Windows (Этот компьютер -> Свойства -> Дополнительные параметры среды):



Можно создавать проект из командной строки Windows:

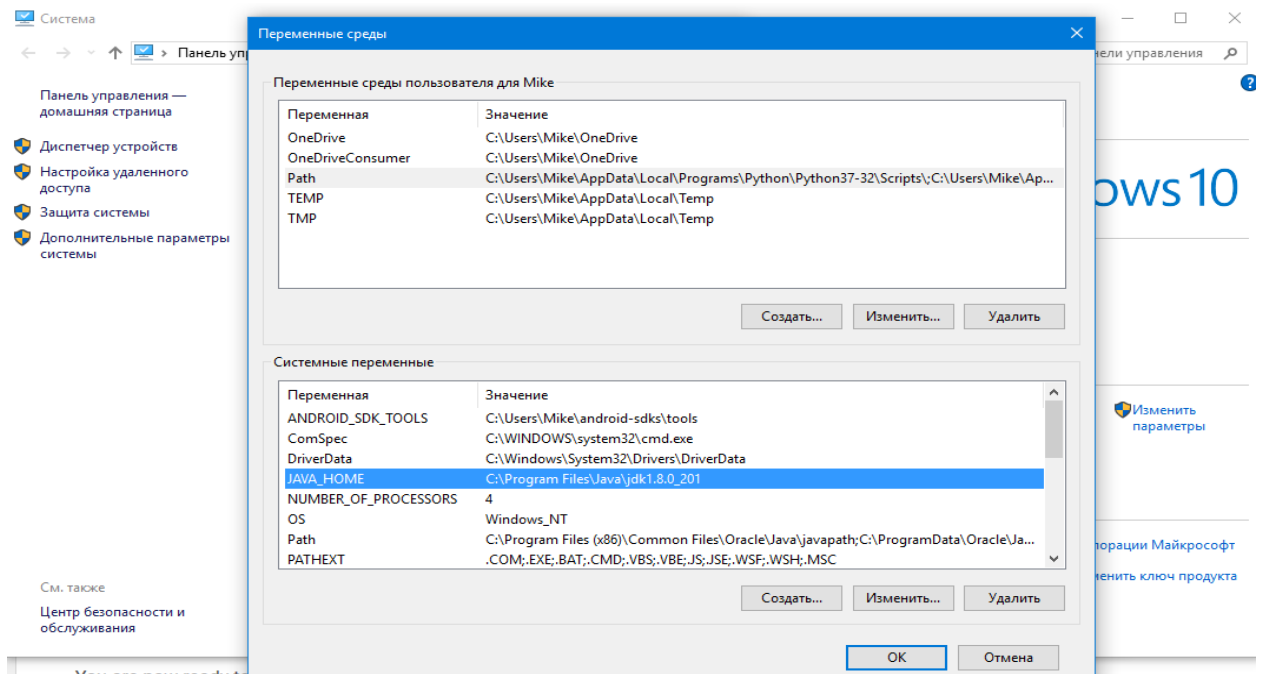
```
flutter create my_app
```



Готово! Файлы проекта можно редактировать любым текстовым редактором, хоть в блокноте. Но это хорошо разве что для мелких правок. Поэтому мы...

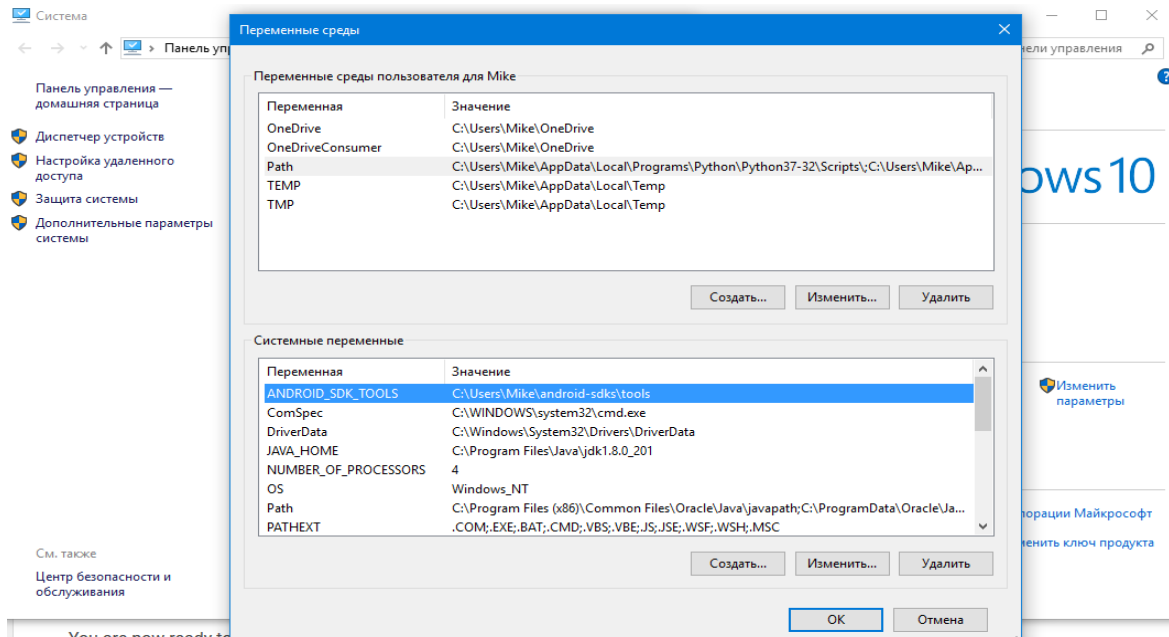
Устанавливаем JDK, Android Studio (вместе с Android SDK) и необходимые плагины

Скачиваем последнюю версию [Java SE Development Kit 8](#) для своей операционной системы (потребуется для Android SDK), устанавливаем на свой компьютер, следуя за мастером установки, и создаём системную переменную среды `JAVA_HOME` с указанием пути к JDK, например: `C:\Program Files\Java\jdk1.8.0_201`.

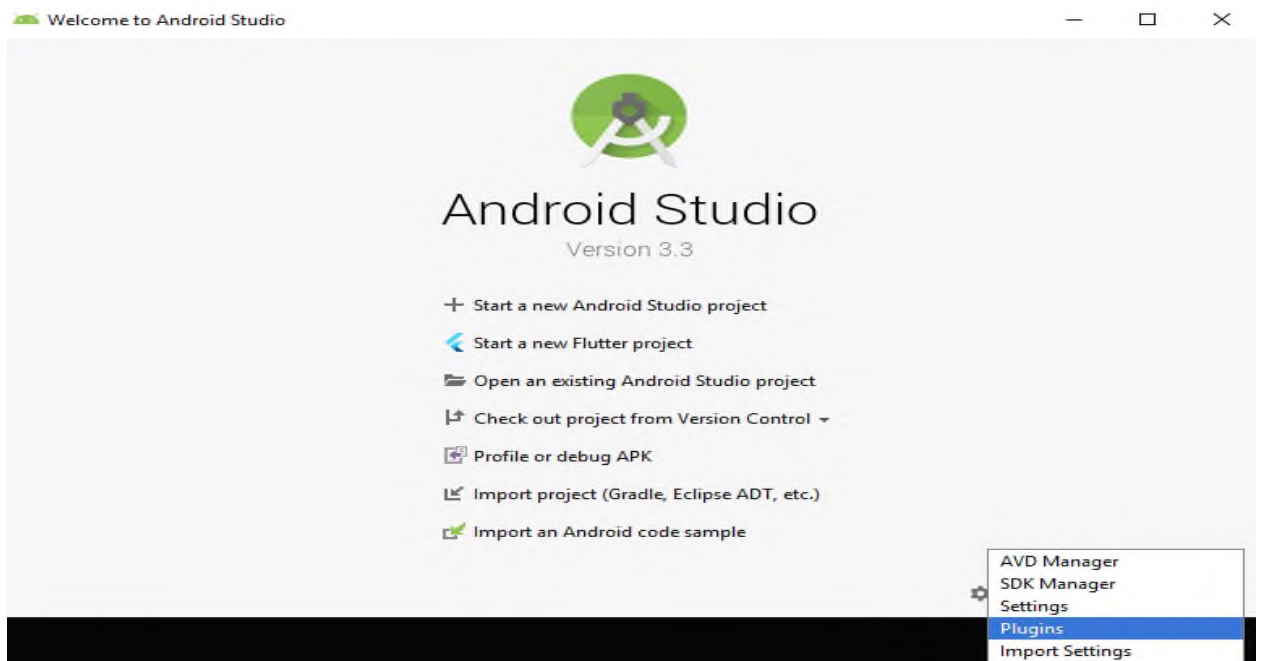


Теперь скачиваем [Android Studio](#). Запускаем процесс установки, следуя за мастером установки, и обращаем внимание на путь, куда будет установлен Android SDK. Создаём системную переменную

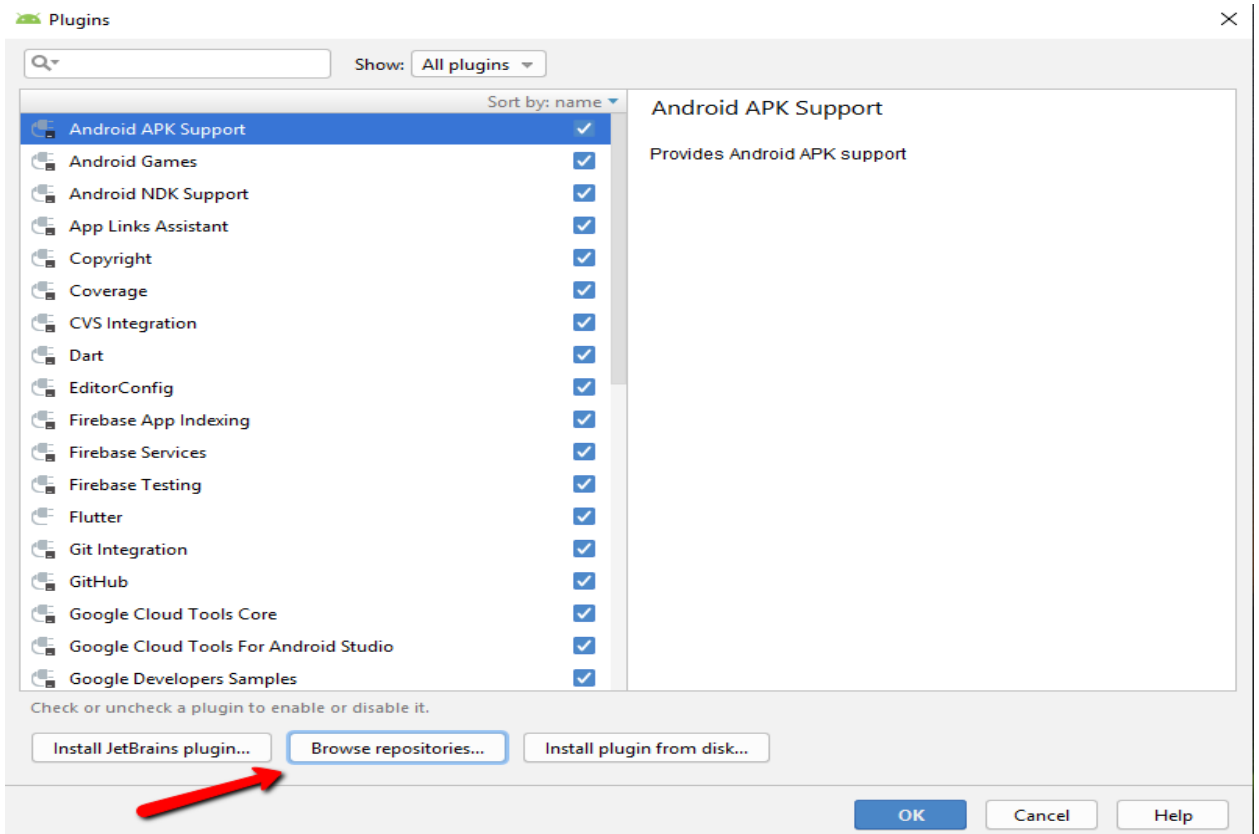
среды *ANDROID_SDK_TOOLS* с указанием пути к папке *\tools* в Android SDK, примерно так:



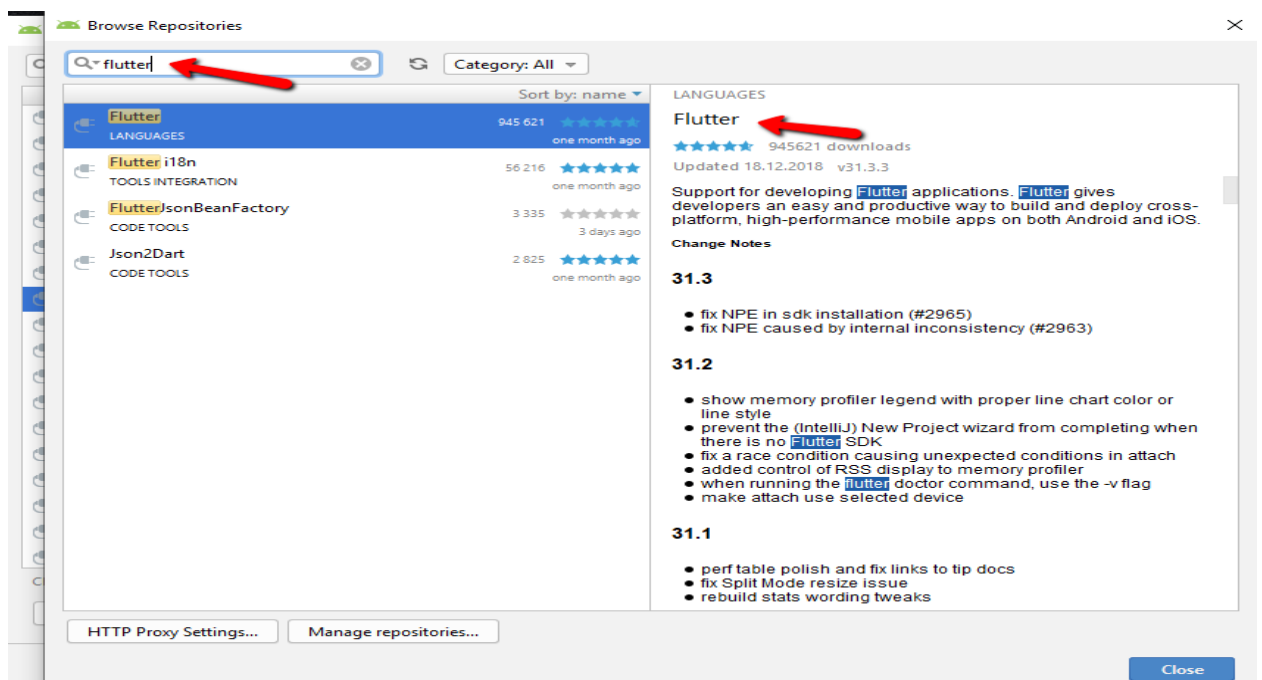
Когда всё готово — запускаем Android Studio, и устанавливаем плагины Flutter и Dart. Для этого в начальном экране Android Studio справа внизу жмём на значок шестерёнки и выбираем Plugins:



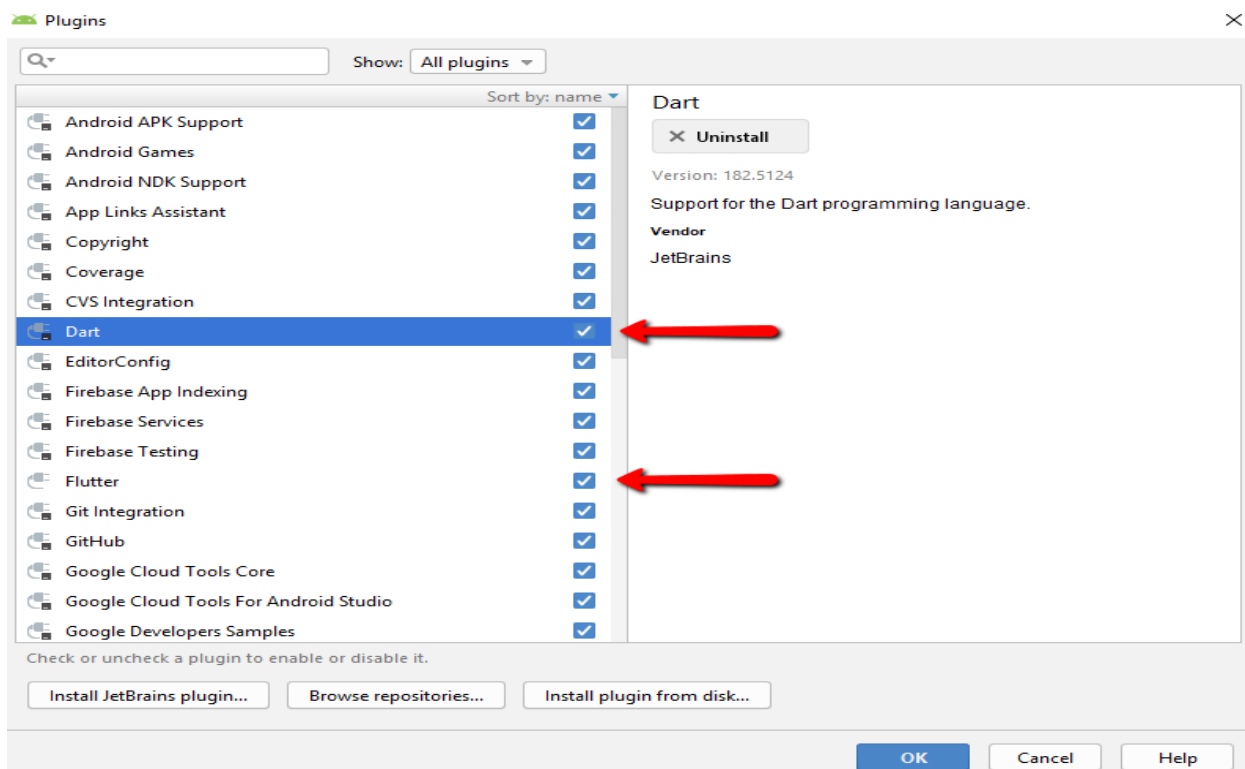
В открывшемся окне внизу нажимаем кнопку *Browse repositories...*



В поисковую строку вводим *flutter*, выбираем и устанавливаем (у меня уже установлен, поэтому не видно соответствующей кнопки):



Android Studio предложит также установить плагин Dart от которого зависит работа плагина Flutter. Соглашаемся. В итоге у вас должно быть установлено как минимум два плагина:



Перезапускаем Android Studio, и теперь давайте убедимся, что всё идёт хорошо. Для этого в командной строке выполним команду:

```
flutter doctor
```

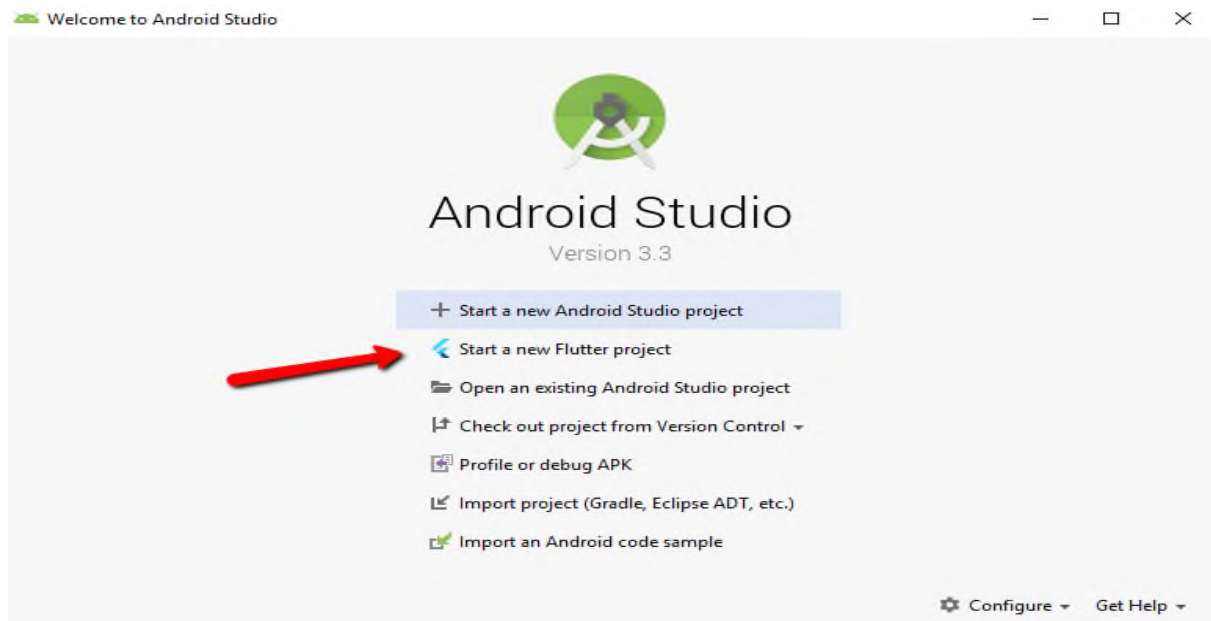
Сканирование займёт десяток секунд, и затем вы можете увидеть примерно такой результат:

```
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, v1.0.0, on Microsoft Windows [Version 10.0.17763.253], locale ru-RU)
[✓] Android toolchain - develop for Android devices (Android SDK 28.0.3)
[✓] Android Studio (version 3.3)
```

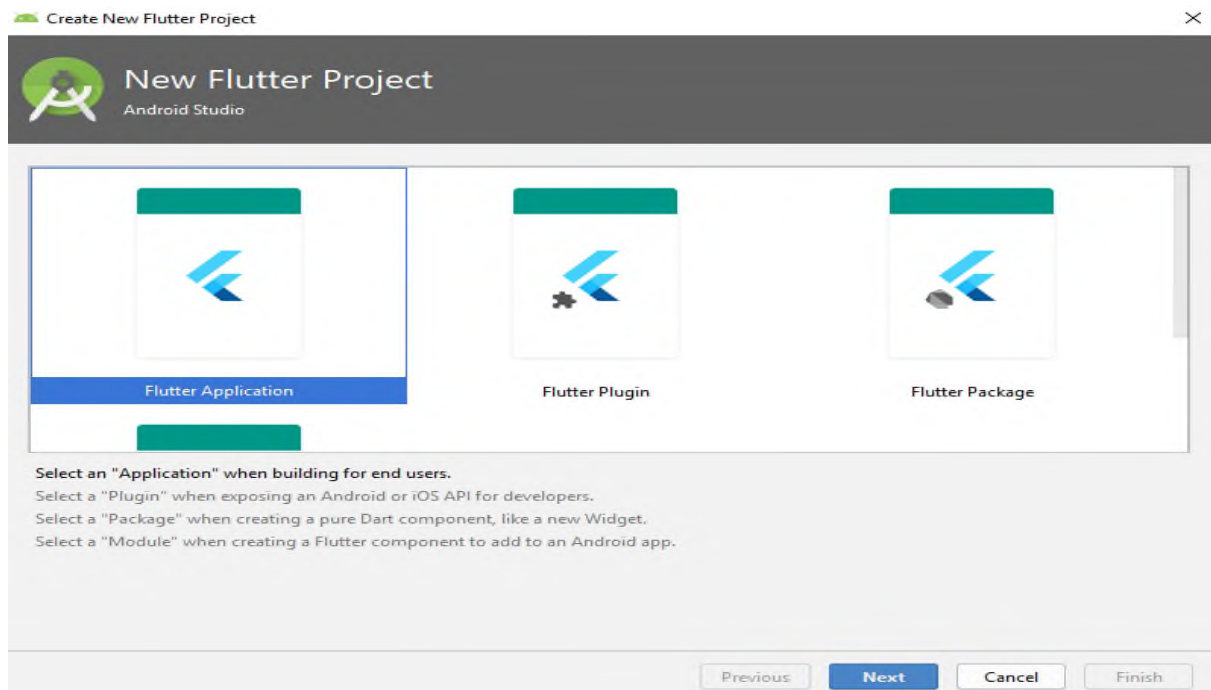
А возможно будет пункт, отмеченный красным крестиком, с пояснением (на английском), что вами ещё не приняты какие-то лицензии (licences), касающиеся Android SDK, и предложение их принять (Y/n). Примите их, напечатав в командной строке Y. Возможно это придётся сделать несколько раз (если имеется несколько лицензий).

Вот мы и готовы создать Flutter проект в Android Studio. После установки плагинов Flutter и Dart в начальном экране Android Studio должна появиться

опция *Start a new Flutter project*. Выбираем её:



Далее соглашаемся с выбранной по умолчанию опцией *Flutter Application* и нажимаем кнопку *Next*:



Указываем название проекта, путь к папке Flutter SDK, путь к папке проекта, даём краткое описание проекта (опционально), и вновь нажимаем кнопку *Next*:

Create New Flutter Project

New Flutter Application
Android Studio

Configure the new Flutter application

Project name
my_cool_app

Flutter SDK path
C:\Users\Mike\flutter [Install SDK...](#)

Project location
C:\Users\Mike\Documents\Development\Flutter

Description
A new Flutter application.

☐ Create project offline

Наконец, указываем доменное имя (которое в реверсивном порядке будет использовано как ID Android приложения), а также опционально — поддержку языков Kotlin и Swift (если не указать — по умолчанию будут поддерживаться только Java и Objective-C). Нажимаем кнопку *Finish*.

Create New Flutter Project

New Flutter Application
Android Studio

Set the package name

Applications and plugins need to generate platform-specific code

Company domain
mzaharov.com

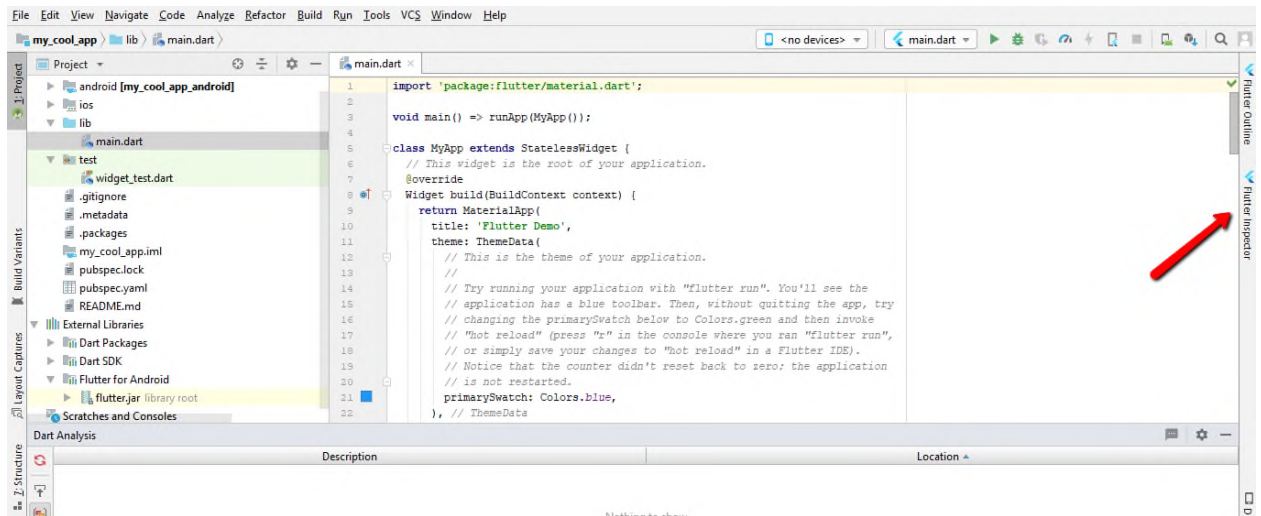
Package name
com.mzaharov.mycoolapp [Edit](#)

Platform channel language

- ☐ Include Kotlin support for Android code
- ☐ Include Swift support for iOS code

[Previous](#) [Next](#) [Cancel](#) [Finish](#)

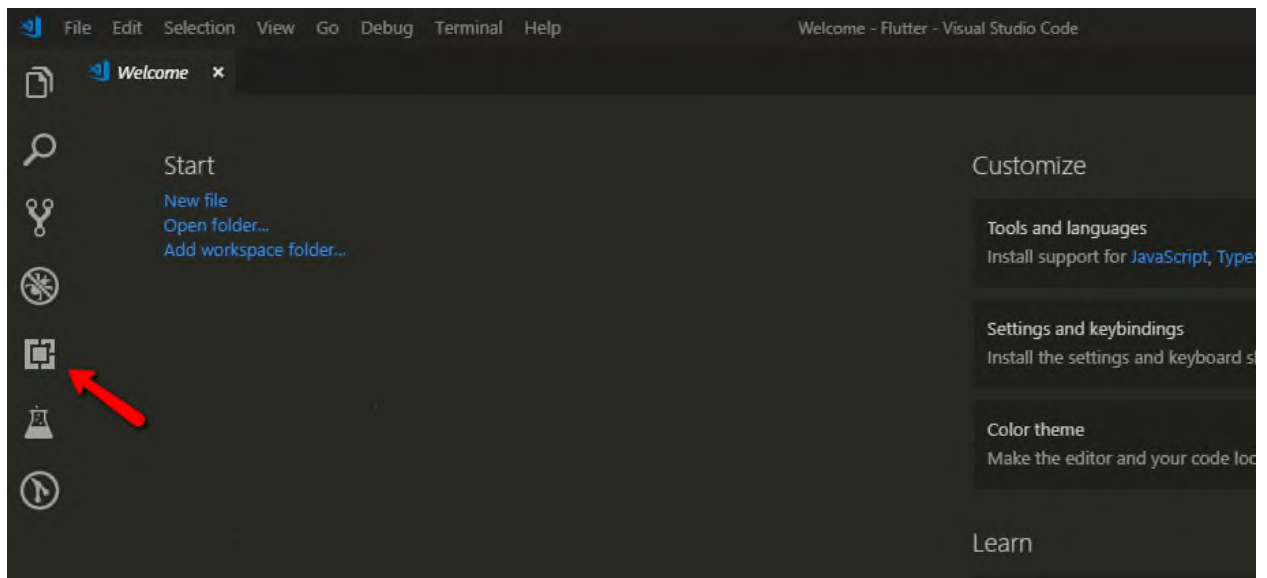
В зависимости от производительности компьютера, ждём несколько минут пока проект будет создан... Готово! Он должен выглядеть примерно так:



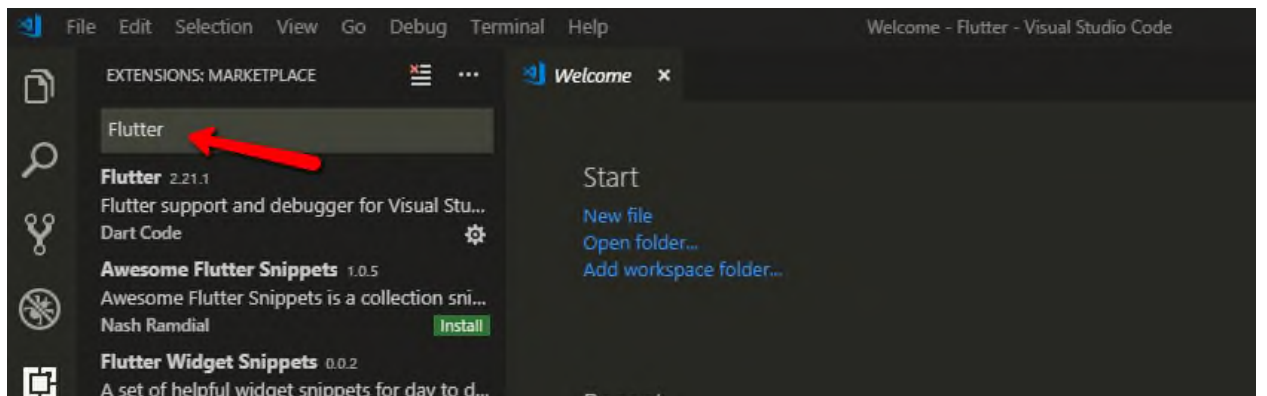
Обратите внимание на стрелку, указывающую на вкладку *Flutter Inspector*. В этом инспекторе имеется функционал, позволяющий делать ряд очень полезных во время разработки вещей, в т.ч. просмотр приложения на девайсе Android в режиме представления на iOS!

И наконец, устанавливаем VS Code, расширения, и создаём третий Flutter проект

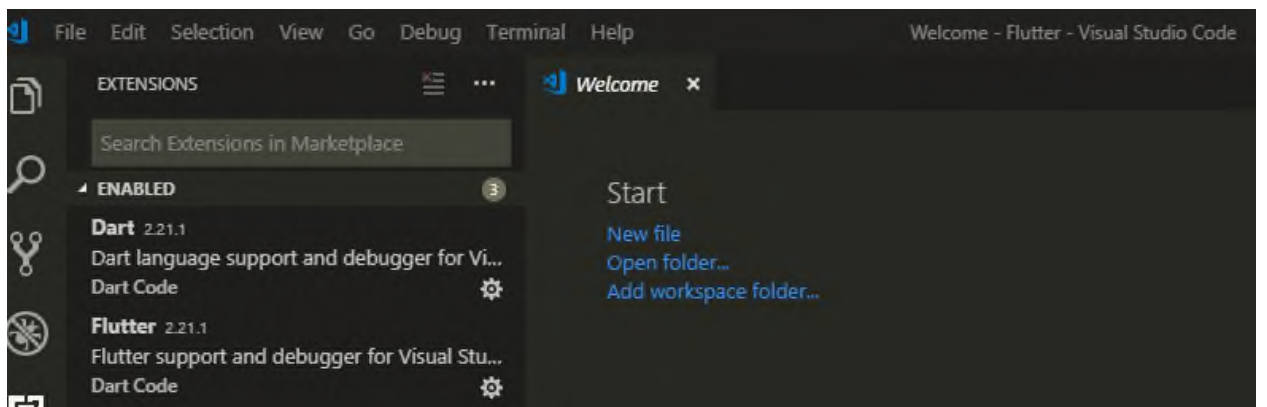
Скачиваем последнюю версию Visual Studio Code для своей операционной системы, устанавливаем на свой компьютер, следуя за мастером установки, и запускаем VS Code. Затем на боковой панели нажимаем на кнопку *Extensions* (показана стрелкой) или на клавиатуре — *Ctrl+Shift+X*:



С помощью поиска ищем расширение Flutter.

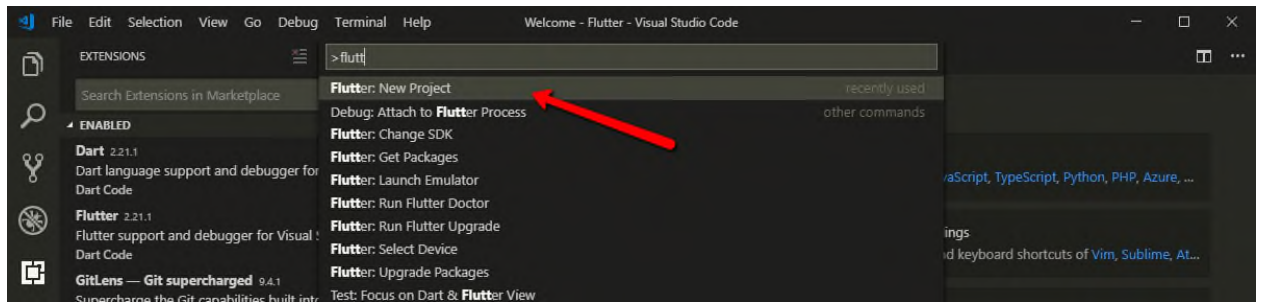


VS Code, как и в случае с Android Studio, предложит установить необходимое дополнительное расширение Dart. Устанавливаем и его. В итоге должны иметь два (или более) активированных расширения:

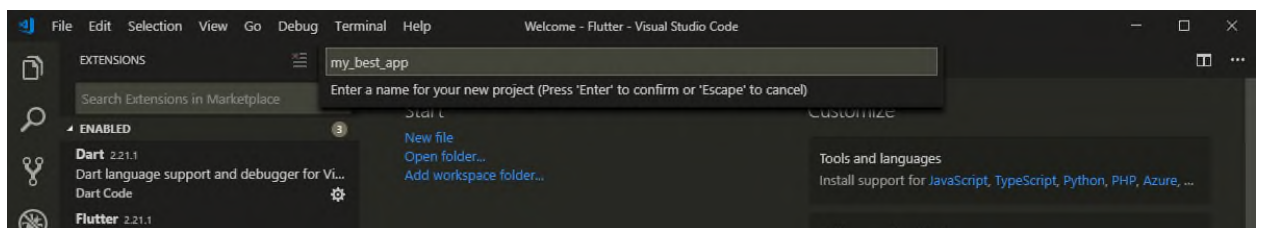


А теперь создаём Flutter проект. Нажимаем на значок шестерёнки в левом нижнем углу, и выбираем *Command Palette...* (или на клавиатуре

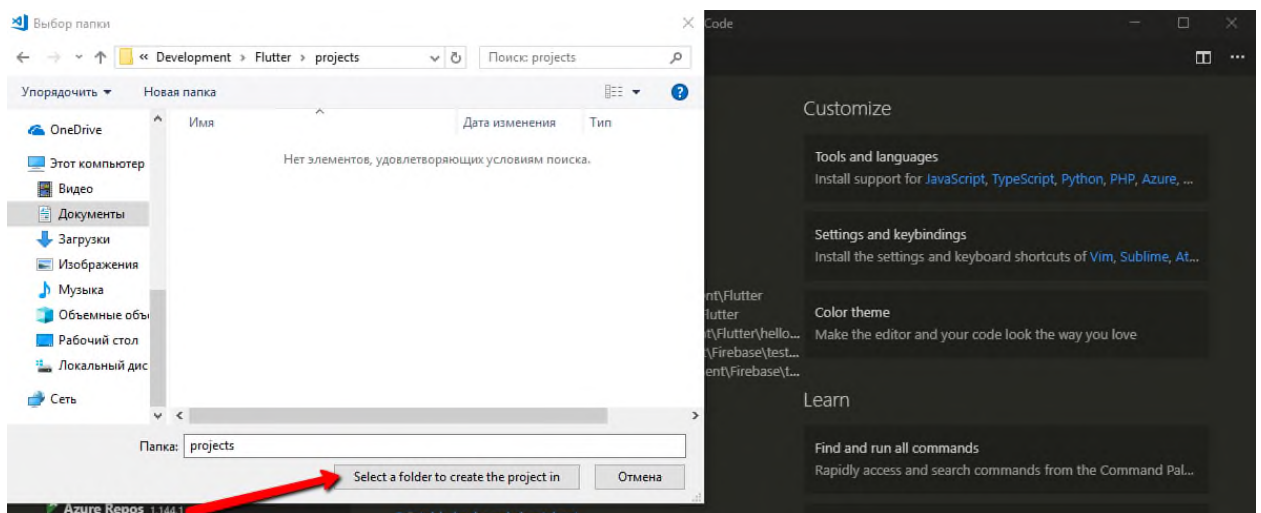
— *Ctrl+Shift+P*). В командной строке Command Palette начинаем печатать *flutter*, и из появившегося списка выбираем *Flutter: New Project*:



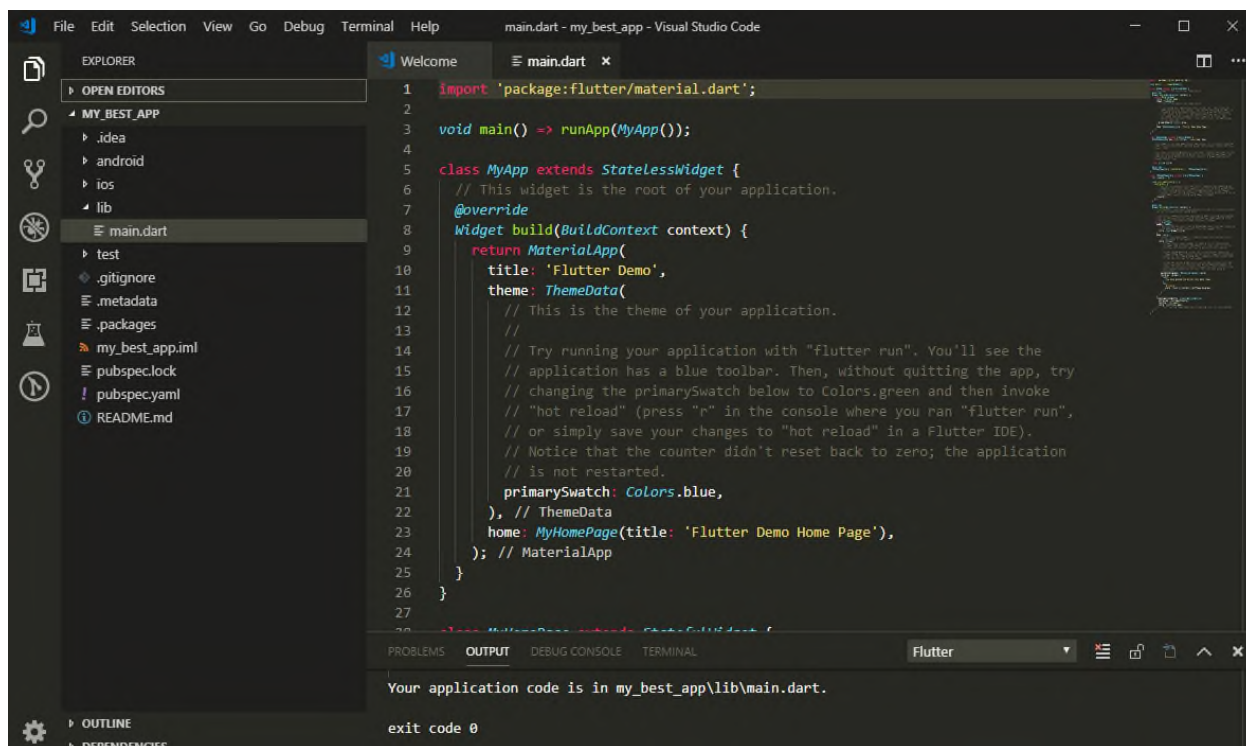
Даём проекту название и нажимаем клавишу *Enter*:



Появится диалоговое окно, предлагающее выбрать папку, в которой необходимо создать Flutter проект. Выбираем и нажимаем кнопку с длинным названием *Select a folder to create the project in*:



Минута ожидания... И, готово!



Мы установили необходимое программное обеспечение (всё бесплатное!), и создали проект Flutter тремя различными способами: из командной строки, с помощью Android Studio, с помощью VS Code.

Каким способом создавать проект и в какой среде разработки (IDE) лучше с ним работать — каждый решает сам. Например, я предпочитаю быстро создать проект из командной строки, затем открыть его в VS Code и большую часть времени над проектом работать именно в нём, т.к. VS Code довольно быстрый.

Но когда надо хорошо протестировать внешний вид и производительность приложения — открываю проект в Android Studio, чтобы использовать имеющийся пока только там *Flutter Inspector*, обладающий, как я уже говорил, рядом очень полезных опций.

13 лекция

Тема: Преобразуйте приложение Android, написанное на Flutter, в iOS и разместите приложение в Play Market и Apple Store.

План:

- 1. Иконка приложения**
- 2. Подпись приложения**
- 3. Связь хранилища ключей с приложением**
- 4. Активация Proguard**

В привычном цикле разработки Flutter приложения мы запускаем `flutter run` или используем опции `Run` или `Debug` в IDE. По умолчанию Flutter создает версию приложения для отладки.

И вот версия для публикации готова к размещению, например в Google Play Store. Перед тем как опубликовать приложение необходимо несколько завершающих штрихов:

- Добавить иконку приложения (launchpad icon)
- Подписать приложение (signing)
- Активировать Proguard
- Проверить манифест приложения (AndroidManifest.xml)
- Проверить конфигурацию сборки (build.gradle)
- Создать версию приложения для публикации (--release)
- Опубликовать в Google Play Store
- Еще раз посмотреть Android release FAQ

1. Иконка приложения

Когда новое Flutter приложение создается, туда добавляется иконка по умолчанию (логотип Flutter). Для настройки иконки можно воспользоваться пакетом [flutter_launcher_icons](#). Альтернативный вариант, сделать всё самостоятельно, выполнив следующие шаги:

- Изучить, или хотя бы просмотреть рекомендации и правила [Material Design Product icons](#).
- В директории `<app dir>/android/app/src/main/res/`, поместить файлы иконки в директории соответствующие [соглашению](#). Стандартные `mipmap`-директории демонстрируют правильное именование.
- В файле `AndroidManifest.xml` в теге `application` обновить атрибут `android:icon` для соответствия иконкам (из предыдущего шага), (для например - `<application android:icon="@mipmap/ic_launcher" ...`
- Для проверки, что иконка заменилась, запустить приложение и проверить "иконку запуска"

2. Подпись приложения

Для публикации приложения в Play Store необходимо подписать приложение цифровой подписью.

Создание хранилища ключей

На Mac/Linux

```
# хранилище можно сохранить в файл ~/key.kjs
# или в ~/.keystore - это имя/расположение по умолчанию
keytool -genkey -v -keystore ~/.keystore \
    -keyalg RSA \
    -keysize 2048 \
    -validity 10000 \
    -alias key
```

сохраняйте файл от публичного доступа и конечно не публикуйте его в репозитории проекта

формат файла начиная с версии JAVA 9 по -умолчанию PKCS12. Если вы создавали файл ранее, то keytool предложит конвертировать файл в формат PKCS12.

3. Связь хранилища ключей с приложением

Создать файл `<app dir>/android/key.properties`, который содержит ссылку на хранилище

```
storePassword=<пароль>
```

```
keyPassword=<пароль>  
keyAlias=key  
storeFile=<путь к файлу ключа например /home/user/.keystore>
```

и здесь тоже, хранить файл `key.properties` в секрете и не публиковать его в репозиторий проекта

Конфигурация подписи в gradle

в файле `<app dir>/android/app/build.gradle`

1. заменить следующее

```
android {
```

на информацию о файле конфигурации

```
def keystoreProperties = new Properties()  
def keystorePropertiesFile = rootProject.file('key.properties')  
if (keystorePropertiesFile.exists()) {  
    keystoreProperties.load(new FileInputStream(keystorePropertiesFile))  
}  
  
android {
```

2. Заменить конфигурацию сборки релиза

```
buildTypes {  
    release {  
        // TODO: Add your own signing config for the release build.  
        // Signing with debug keys for now,  
        // so `flutter run --release` works  
        signingConfig signingConfig.debug  
    }  
}
```

на следующую информацию, содержащую параметры подписи

```
signingConfigs {  
    release {  
        keyAlias keystoreProperties['keyAlias']  
        keyPassword keystoreProperties['keyPassword']  
        storeFile file(keystoreProperties['storeFile'])  
        storePassword keystoreProperties['storePassword']  
    }  
}  
  
buildTypes {
```

```
        release {  
            signingConfig signingConfigs.release  
        }  
    }  
}
```

Теперь сборка релиза будет подписываться автоматически.

4. Активация Proguard

По умолчанию Flutter не проводит обусфикацию и минификацию кода Android. Если есть намерение использовать сторонние библиотеки Java, Kotlin или Android, то имеет смысл снизить размер APK и защитить код от "reverse engineering".

Шаг 1 - конфигурация Proguard

Создать файл `/android/app/proguard-rules.pro` и добавить правила

```
## Flutter wrapper  
-keep class io.flutter.app.** { *; }  
-keep class io.flutter.plugin.** { *; }  
-keep class io.flutter.util.** { *; }  
-keep class io.flutter.view.** { *; }  
-keep class io.flutter.** { *; }  
-keep class io.flutter.plugins.** { *; }  
-dontwarn io.flutter.embedding.**
```

Эти правила относятся только ко Flutter, для других библиотек нужны свои собственные правила, ну например для Firebase.

Шаг 2 - активация обусфикации и/или минификации

Откроем `/android/app/build.gradle`, найдём определение `buildTypes`. Внутри конфигурации `release`, добавим `minifyEnabled` и `useProguard`, а также укажем файл с конфигурацией Proguard.

```
android {  
    ...  
  
    buildTypes {  
        release {  
            signingConfig signingConfigs.release
```

```

minifyEnabled true
useProguard true

proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-
rules.pro'
}
}
}

```

конечно минификация и обусффикация увеличат время компиляции файла

Проверим манифест приложения

Проверим манифест приложения, `AndroidManifest.xml`, размещенный в `<app dir>/android/app/src/main` и убедимся, что установленные значения корректны:

- `application`: отредактируйте `android:label` в теге `application` чтобы установить реальное, правильное имя приложения.
- `uses-permissions`: Добавьте разрешение `android.permissions.INTERNET` если приложению требуется доступ в Интернет. Стандартный шаблон приложения не включает этот тег, но позволяет устанавливать соединение в процессе разработки между приложением и инструментами Flutter.

```

<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="org.diera.app.digital_era">

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />

    <application ...>
    </application>

```

Проверим конфигурацию сборки

Проверить файл конфигурации Gradle, `build.gradle`, расположенный в `<app dir>/android/app` и убедиться, что установленные значения верны:

- `defaultConfig`
- `applicationId` - указан уникальный идентификатор приложения

- `versionCode` и `versionName`: указать внутреннюю версию приложения (число), и строковую версию приложения (для показа пользователям)
- `minSdkVersion` и `targetSdkVersion`: указать минимальный уровень API, и уровень API для которого было разработано приложение.

`versionCode` - это положительное число, которое используется в качестве внутренней версии приложения. Это число используется только для определения является ли одна версия более новой чем другая. Чем больше число, тем новее версия. Пользователи не видят эти версии. **Максимально допустимое число - 2100000000.**

`versionName` - строка, используемая в качестве номера версии, показываемой пользователям. Единственная цель `versionName` - отображение пользователю.

Посмотреть распределение версий можно на [странице статистики версий](#).
Список [API Levels](#)

Сборка релиза

Есть два возможных варианта публикации в Play Store

- App bundle (рекомендуемый)
- APK

Сборка app bundle

Если была произведена настройка подписи, то пакет будет подписан автоматически. Перейти в директорию приложения и выполнить
сейчас (2019) это рекомендованный способ публикации приложений

```
flutter build appbundle --release
```

версии сборки и приложения определяются на основе файла `pubspec.yaml`. `version: 1.0.0+1` - версия приложения это первые три цифры разделенных точками, а версия сборки это число после знака `+`. Это значения можно переопределить при выполнении команды используя опции `--build-name` (`versionName`) и `--build-number` (`versionCode`).

Тестирование app bundle

Есть несколько путей чтобы протестировать app bundle, здесь представлены два

Оффлайн, используя bundle tools

1. Если еще не имеет `bundletools`, то получаем [это здесь](#)
2. [Генерируем набор APK](#) из пакета приложения
3. [Устанавливаем APK](#) на присоединенные устройства

Онлайн, используя Google Play

1. Загрузить пакет в GooglePlay и протестировать. Можно использовать alpha или beta-релизы перед реальной публикацией приложения.
2. [Процесс загрузки пакета приложения](#)

Сборка APK

Несмотря на то, что сборка пакета приложения считается приоритетной перед сборкой APK. Могут быть случаи когда пакеты поддерживаются механизмом распространения. Тогда надо будет выпустить APK для каждой цели ABI (Application Binary Interface).

Если цифровая подпись уже настроена в приложении, все APK будут подписаны:

```
flutter build apk --release --split-per-abi
```

на выходе получим два APK

- `<app dir>/build/app/outputs/apk/release/app-armeabi-v7a-release.apk`
- `<app dir>/build/app/outputs/apk/release/app-arm64-v8a-release.apk`

Если убрать опцию `--split-per-abi`, то получим один "жирный" файл содержащий весь код для всех ABI целей. И пользователю также придется загружать код который не совместим с его платформой.

Для установки APK на устройство, подключим его через USB и выполним команду `flutter install`

14 лекция.

Тема: Настройки безопасности, разрешения, шифрование данных при разработке и использовании мобильных приложений.

План:

- 1. Защита мобильного приложения**
- 2. Защита мобильного приложения**
- 3. Дополнительные рекомендации**

При разработке мобильного приложения следует учитывать, что данные, которыми оперирует это приложение, могут представлять определенный интерес для третьих лиц. Степень ценности этих данных варьируется в широких пределах, тем не менее, даже наиболее простая приватная информация, например, пароль входа в приложение, требует проработки ее защиты. Особенно это важно в свете распространения мобильных приложений на все сферы электронных услуг, включая финансовые, банковские операции, хранение и передачу личных данных и так далее. Всем интересующимся — добро пожаловать под кат.

Все нижесказанное — исключительно мой опыт, безусловно, данные могут быть неточными, поэтому буду благодарен за любые поправки и дополнения к статье. Я не нашел исчерпывающих статей в сети на подобную тематику, которые бы собирали всю нужную (хотя бы базовую) информацию в одном месте, поэтому решил обобщить свой опыт в этой области на текущий момент времени.

1. Защита мобильного приложения

Основные виды атак на мобильное приложение:

- Декомпиляция файла приложения (.ipa-файлы для Apple iOS и .apk-файлы для Google Android) и разбор локально сохраненных данных. Защита этого, наиболее важного в настоящее время, уровня целиком лежит на плечах мобильного разработчика.
- Перехват данных, передаваемых по сети (MITM-атаки). Большинство мобильных приложений являются клиент-серверными, следовательно, постоянно передают и принимают большие объемы информации. И хотя современная мобильная и веб-разработка активно завершают переход на HTTPS-протокол общения, тем не менее, не стоит полагаться на единственный рубеж защиты в виде защищенного канала связи.

- Рутование устройства и атака на приложение и применяемые в нем алгоритмы через внешние отладочные инструменты.

Перечень основных уязвимостей приложений

Рассмотрим уязвимости общего характера, без привязки к конкретной платформе. Здесь и далее используется аббревиатура КВД — критически важные данные пользователей. К КВД относятся любые данные, которые не должны быть доступны третьей стороне, это касается как персональных данных пользователя (дата рождения, адрес проживания, личная переписка), так и его частных данных (пароли, данные кредитных карт, номера банковских счетов, номера заказов и так далее).

Перечень основных уязвимостей следующий:

- Использование незащищенных локальных хранилищ.
- **Опасность:** Очень высокая.
- **Комментарий:** Встречается повсеместно, выражается в хранении КВД в незащищенных или слабо защищенных локальных хранилищах, специфических для конкретной платформы. Вскрытие третьей стороной — элементарное, и, как правило, не требуется наличие специальных навыков у атакующего.
- **Защита:** Хранить КВД можно только в защищенных хранилищах платформы.
- Хранение КВД в коде.
- **Опасность:** Высокая.
- **Комментарий:** Уязвимость касается хранения КВД внутри кода (в статических константных строках, в ресурсах приложения и т.п.). Яркие примеры: хранение соли для пароля (password salt) в константе или макросе, которая применяется по всему коду для шифрования паролей; хранение приватного ключа для асимметричных алгоритмов; хранение паролей и логинов для серверных узлов или баз данных. Легко вскрывается третьей стороной при наличии базовых навыков декомпиляции.
- **Защита:** Не хранить никакие КВД в коде или ресурсах приложения.

- Применение алгоритмов с хранением приватного ключа.
 - **Опасность:** Высокая.
 - **Комментарий:** Уязвимость актуальна в случае, если приватная информация алгоритма (приватный ключ) вынужденно сохраняется в коде или ресурсах мобильного приложения (чаще всего так и бывает). Легко вскрывается методом декомпиляции.
 - **Защита:** В мобильной разработке желательно применять только современные симметричные алгоритмы с генерируемым случайным одноразовым ключом, обладающие высокой стойкостью с взлому методом грубой силы, либо выводить асимметричный приватный ключ за пределы приложения, либо персонализировать этот ключ (как пример — приватным ключом может выступать пользовательский код входа, сохраненный в зашифрованном виде в защищенном хранилище операционной системы).
- Использование асимметричного алгоритма с приватным ключом, известным серверу.
 - **Опасность:** Зависит от степени защищенности сервера.
 - **Комментарий:** Уязвимость носит двойной характер. Хранение приватного ключа допускает возможность расшифровки пользовательских данных на стороне сервера. Во-первых, это некорректно с точки зрения безопасности (если сервер будет взломан — атакующий также получит доступ к приватным данным пользователей), а во-вторых, это нарушает приватность персональных данных. Пользователь всегда должен быть уверен, что его персональная информация не известна никому, кроме него самого (только если он явно не дал разрешение на ее публикацию). Часто приложения позиционируют себя как защищенные, но на деле таковыми не являются, так как содержат внутри себя средства для расшифровки персональной информации.
 - **Защита:** Без явной необходимости и явного разрешения пользователя (чаще всего через лицензионное соглашение) ни приложение, ни сервер не должны иметь никакой возможности расшифровать приватные данные пользователя. Простейший пример — пароль пользователя должен уходить на сервер уже в

виде хеша, и проверяться должен хеш, а не исходный пароль (серверу абсолютно незачем знать пользовательский пароль; если же пользователь его забыл — для такой ситуации существует давно отлаженный механизм восстановления пароля, в том числе с двухфакторной авторизацией клиента для повышенной безопасности процедуры восстановления).

- Использование самописных алгоритмов шифрования и защиты.
 - **Опасность:** Средняя.
 - **Комментарий:** Это прямое нарушение принципа Керкгоффса. Выражается в попытке разработчика изобрести "свой личный, не известный никому, а поэтому супер-защищенный алгоритм шифрования". Любое отклонение от существующих, многократно проверенных и изученных, математически доказанных алгоритмов шифрования в 99% случаев оборачивается быстрым взломом подобной "защиты". Требует наличия средне-высоких навыков у атакующего.
 - **Защита:** Следует подбирать подходящий алгоритм только из отлаженных и актуальных общеизвестных криптографических алгоритмов.
- Передача КВД во внешнюю среду в открытом виде.
 - **Опасность:** Средняя.
 - **Комментарий:** Выражается в передаче КВД без применения шифрования по любому доступному каналу связи с внешней средой, будь то передача данных стороннему приложению или передача в сеть. Может быть вскрыто опосредованно путем вскрытия не приложения, а его хранилища, или целевого приложения. Взлом требователен к наличию навыков у атакующего, при условии, что хранилище является защищенным.
 - **Защита:** Любые КВД перед выходом за пределы приложения должны быть зашифрованы. Локальные хранилища платформы *не являются* областью приложения, они тоже должны получать на вход только зашифрованные данные.
- Игнорирование факта наличия рутованных или зараженных устройств.

- **Опасность:** Средняя.
 - **Комментарий:** Рутованные устройства — это девайсы, где выполнена модификация для получения прав суперпользователя на любые операции, изначально запрещенные производителем операционной системы. Выполняется пользователем на своем устройстве самостоятельно, и не обязательно добровольно (клиент может быть не в курсе, что устройство взломано). Установка приложения на рутованный девайс нивелирует все штатные средства защиты операционной системы.
 - **Защита:** Если это технически возможно для платформы — то желательно запрещать работу приложения, если удалось понять, что запуск производится на рутованном устройстве, или хотя бы предупреждать об этом пользователя (спасибо за дополнение [DjPhoenix](#)).
- Хранение КВД в защищенных хранилищах, но в открытом виде.
 - **Опасность:** Средняя.
 - **Комментарий:** Разработчики зачастую склонны сохранять КВД в защищенные системные хранилища без дополнительной защиты, поскольку системные механизмы хорошо сопротивляются взлому. Однако уровень их стойкости падает до минимума в случае, если устройство рутованное.
 - **Защита:** КВД не должны использоваться в приложении без дополнительного шифрования. Как только надобность в "открытых" КВД отпала — они немедленно должны быть либо зашифрованы, либо уничтожены.
 - Перевод части функционала во встроенные веб-движки.
 - **Опасность:** Средняя.
 - **Комментарий:** Чаще всего выглядит как передача КВД во встроенный браузер, где загружается внешняя веб-страница, выполняющая свою часть функционала. Уровень защиты в этом случае резко снижается, особенно для рутованных устройств.

- **Защита:** Не использовать встроенный браузер и встроенный веб-движок в операциях с КВД. На крайний случай — шифровать КВД перед передачей.
- Реверсивная инженерия алгоритмов, представляющих интеллектуальную ценность.
- **Опасность:** Низкая, зависит от ценности алгоритма.
- **Комментарий:** Если при разработке приложения внутри компании используются некие собственные алгоритмы, которые могут представлять высокую ценность для потенциальных конкурентов или взломщиков, то эти алгоритмы должны быть защищены от постороннего доступа.
- **Защита:** Автоматическая или ручная обфускация кода.

Специфика разработки мобильных приложений

Есть несколько общих для всех мобильных платформ моментов, которые следует соблюдать при разработке.

Защита пользовательским кодом

- Если приложение защищено пользовательским паролем (PIN-кодом, сканом отпечатка пальца, графическим паролем и т.д.), то при уходе приложения в фон ("сворачивании") оно должно немедленно отображать окно ввода этого защитного кода, перекрывая собой весь экран приложения. Это исключает возможность для злоумышленника получить приватную информацию в случае кражи устройства, пока приложение все еще запущено и находится в спящем режиме.
- Любой пользовательский код должен иметь ограниченное количество попыток ввода (например, 5 раз), затем, в случае неудачи, приложение должно автоматически разлогиниваться (или и вовсе блокироваться, зависит от конкретного приложения).
- В настоящее время при использовании цифровых кодов строго рекомендуется использовать ограничение на длину кода в минимум 6 цифр (больше можно, меньше — нельзя).

2. Функционирование клиент-серверного приложения

- Для клиент-серверных приложений очень полезно применять сессионный механизм с ограниченным временем жизни сессии. Это позволит избежать "простаивания" приложения в незащищенном режиме, если пользователь просто забыл закрыть его и оставил устройство в свободном доступе. Следует учитывать, что срок действия сессии и ее идентификатор относятся к КВД, со всеми вытекающими отсюда последствиями. Одним из удачных примеров реализации подобного механизма является получение абсолютного значения времени с сервера после прохождения процедуры авторизации пользователя (дата и время должны показывать, когда именно сессия станет неактивной). Дату и время окончания действия сессии не следует генерировать на устройстве, это снижает безопасность и гибкость приложения.
- Клиент-серверное приложение не должно осуществлять изменение КВД в локальном режиме. Любое действие, требующее изменения КВД, должно проходить синхронизацию с сервером. Исключение из этого правила составляет только пользовательский код входа, задаваемый лично пользователем и сохраненный в защищенном локальном хранилище.

Работа с датами

- При оперировании важными для работы приложения датами, вроде времени уничтожения сессии, не следует опираться на относительное время. То есть, передаваемые с сервера данные не должны содержать дату в виде "плюс N секунд/часов/дней от текущего момента". В силу наличия потенциально высоких задержек в передаче данных по сети от мобильного приложения к серверу и обратно, подобный способ синхронизации будет обладать слишком большой погрешностью. Кроме того, атакующий (или просто недобросовестный пользователь) может попросту сменить локальный пояс на устройстве, нарушив таким образом логику работы ограничительных механизмов приложения. Всегда нужно передавать только абсолютное значение времени.
- Абсолютные значения следует передавать с применением универсальных способов обмена подобной информацией, без привязки к часовому поясу конкретного пользовательского устройства. Чаще всего, оптимальным вариантом является поведение приложения, при котором данные отображаются пользователю в его локальном часовом

поясе, но их хранение и передача осуществляется в формате, не привязанном к тайм-зоне. Подходящими форматами для дат и времени являются либо универсальный UNIX timestamp, сохраненный в переменной 64-битного целого знакового типа (UNIX timestamp — это количество секунд, прошедшее с 1 января 1970 года), либо, на крайний случай, строка в полном формате ISO-8601 с нулевой тайм-зоной. Предпочтителен именно UNIX timestamp, он позволяет избежать потенциальных ошибок и проблем с конвертацией строк в дату и обратно на разных мобильных платформах.

3. Дополнительные рекомендации

- Приложение не должно отображать приватную пользовательскую информацию большими, яркими, хорошо читаемыми шрифтами, без явной на то необходимости и без отдельного запроса пользователя, чтобы исключить возможность чтения этих данных издали с экрана устройства.
- Не стоит слепо доверять библиотекам с открытым исходным кодом, которые предлагают некую защиту приватным данным пользователей. Исключение составляют библиотеки, проверенные временем и используемые в крупных проектах корпораций (например, встроенное шифрование в открытом движке базы данных Realm). Штатных механизмов защиты операционной системы и общедоступных проверенных криптографических алгоритмов в подавляющем большинстве случаев будет более, чем достаточно.
- Абсолютно недопустимо использовать криптографические библиотеки с закрытым исходным кодом (даже если они платные). В таких решениях вы никак не сможете проверить, насколько эффективна данная библиотека, а также насколько "честная" у нее защита (нет ли там механизма backdoor, или не отсылаются ли "защищенные" данные какой-то третьей стороне).
- В релизных сборках приложений должно быть отключено логгирование данных в системную консоль и незащищенные файлы. Специфические логи для разработчиков могут присутствовать, но желательно в зашифрованном виде, во избежание доступа третьих лиц к закрытой служебной информации, которая может присутствовать в логах.

