

1 лабораторная работа

Тема: Ранние мобильные устройства и мобильные приложения. Современные мобильные ОТ.

Цель работы: ознакомиться с ранними и современными мобильными устройствами.

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.

Начало мобильных телефонов

В апреле 1973 года, именно 3-го числа этого месяца, Мартин Купер из Motorola впервые позвонил доктору Джоэлу Энгелю 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).



Что представляет собой современный телефон? Нет, это не «чемоданчик» с «крутилкой» и не «кирпич» с большими кнопками, которые обладают функцией звонка. Современный мобильный телефон – это устройство, в котором сочетаются не только габариты и дизайн, но и технические характеристики, и операционная система.

О лидерах рынка мобильных ОС и пойдет речь в данном материале. В продолжении [предыдущей статьи](#) расскажем про операционные системы iOS, Android и Windows Phone.

Операционная система iOS

iOS

iOS – одна из самых передовых операционных систем на рынке смартфонов и планшетов. Устанавливается на iPhone, iPod и iPad. Имеет множество функций, многозадачная, не требовательна к ресурсам. Но данная система платная, устройства компании Apple стоят не дёшево.

Также данная система имеет закрытый исходный код, а значит, пользователь не может самостоятельно вносить изменения в систему, а также практически все приложения для неё платны, ну если вы добросовестный пользователь и не ставите Jailbreak, чтобы не потерять гарантию. Хотя и последнее не гарантирует вам свободу действий.



Устройства Apple, интерфейс iOS.

Первая версия мобильной операционной системы была построена на том же ядре Unix, что и Mac OS X. Но с первых минут презентации стало ясно, что отличия будут колоссальными.



iPhone OS 1 — 29 июня 2007.

Каким бы инновационным ни был iPhone на момент презентации, его функциональность крайне ограничена. Мало какие знакомые сейчас функции были реализованы в первой версии операционки:

- Основной интерфейс;

- Мультитач жесты;
- iPod;
- Safari;
- Карты;
- Синхронизация с iTunes.

В обновлениях появились:

- Веб–приложения на домашнем экране;
- Изменение расположения иконок;
- Клавиатура с поддержкой мультитач;
- iTunes Music Store.



iPhone OS 2 — 11 июля 2008.

Во второй версии операционки Стив Джобс признал идею использовать только веб–приложения провальной. Появление App Store стало одним из самых важных изменений за время существования мобильной операционки;

- App Store;
- Mobile Me;
- Нативные приложения от сторонних разработчиков;
- Поддержка Exchange;
- Поиск в приложении Контакты.

В обновлениях появились:

- Плейлисты iTunes Genius;
- Google Street View;
- Скачивание подкастов.

Главное событие iOS – это запуск App Store и возможность выкладывать туда сторонние приложения для смартфона. Такие магазины приложений были уже не редкостью, но именно такого шага ждали фанаты Apple, потому что потенциал у iOS был огромный.

У основных конкурентов, такие магазины были или слабо интегрированы в само устройства, либо их не было совсем. Именно поэтому это обновление опять сделала iOS лидером рынка мобильных ОС. Теперь, доступ к магазину контента, был у человека в кармане.



Магазин приложений App Store.

С точки зрения бизнеса – это идеальная схема, облегчающая процесс продажи музыки и приложений. Прямо через телефон теперь стало возможно скачать и купить песню или игру, оплачивая это со своей карты.

iOS SDK. Со второй версии операционной системы появилось SDK (Software Development Kit) – это комплекс средств для разработки приложений. Теперь любой разработчик программ или игр, сможет писать свои приложения для iPhone и iPod. Это самое ожидаемое нововведение в iOS 2, которое ждали со времён презентации первой версии в 2007 году.

Сама операционная система остаётся ещё закрытой, приложения можно устанавливать только с App Store, много функций айфона нельзя привязать ещё к компьютеру.



iPhone OS 3 — 17 июня 2009.

В третью версию операционной системы было добавлено много функций, которых так сильно не хватало пользователям, например поиск по смартфону и копирование/вставка. Кроме того, в версии 3.2 появилась поддержка планшетов.

Возможности:

- Вырезать, копировать, вставить;
- Голосовое управление;
- MMC;
- Съёмка видео;
- Автофокус;

- Поиск Spotlight;
- Push уведомления;
- Ландшафтная ориентация клавиатуры;
- Find my phone;
- Компас;
- Возможность войти в аккаунт YouTube;
- Bluetooth A2DP;
- Покупки внутри приложений;
- Родительский контроль;
- Прямые покупки в iTunes;
- HTML5 в Safari.

В обновлениях появились:

- Genius;
- Загрузка рингтонов;
- Поддержка голосового управления по Bluetooth;
- Поддержка разрешения экрана iPad;
- Приложения для iPad;
- Поддержка Bluetooth клавиатур;
- iBooks.

Spotlight – удобный поиск. Был реализован поиск файлов, приложений, документов внутри системы. Основная идея – это создание быстрого поиска, для этого был добавлен ещё один рабочий стол, на котором находилась форма для ввода текста. Теперь iOS догнал по этому вопросу основных конкурентов (BlackBerry, PalmOS, webOS и Windows Mobile), у которых это уже было хорошо реализовано.

Также, в третьей версии была добавлена поддержка MMS, что на середину 2009 года уже было не актуально, потому что это дополнение уже теряло популярности из-за того что стал более доступен мобильный интернет.

iOS 3.0 была намного удачней второй версии, потому что разработчики учли много пожеланий простых пользователей и постарались максимально убрать все недочёты.



iOS 4 — 21 июня 2010.

Летом 2010 iPhone OS была переименована в хорошо всем знакомую iOS. Наконец-то появилась многозадачность, пусть урезанная, но это стало очередным скачком в развитии:

- Многозадачность;
- Папки на домашнем экране;
- FaceTime;
- Группировка сообщений в Mail;
- iAd;
- Установка любых картинок в качестве заставки;
- Возможность дарить приложения;
- Поиск в приложении Сообщения.

В обновлениях появились:

- Game Center;
- iTunes Ping;
- Многозадачность на iPad;
- Папки на iPad;
- AirPlay;
- AirPrint;
- iPhone в качестве точки доступа;
- iTunes Home Sharing;
- AirPlay для сторонних приложений.

FaceTime. Четвёртая версия iOS появилась в iPhone 4, поэтому так сильно была изменена фронтальная камера. Теперь кроме видео связи, пользователь может создавать видео конференции. Проблема в том, что конференция может быть только среди устройств Apple (iPad, MacBook).

Пользовательские папки. После того, как прошло три года с выхода первой версии, наконец-то была добавлена возможность создания папок на рабочем столе, теперь ярлыки можно объединять в папки и благодаря чему получалось очень компактное меню. Это функция была первой среди сенсорных смартфонов. В Android добавили такую возможность только в версии 4.0 Ice Cream Sandwich.

После выпуска четвёртой версии, многие пользователи заметили, что iPhone 4 неправильно отображает информацию, показывающую уровень приёма сигнала мобильной связи.

Многие пользователи стали проверять и ранние версии айфона и оказалось, что они тоже не правильно отображают уровень сигнала. Дело в том, что деления уровня сигнала начинают уменьшаться, только когда телефон находится в зоне очень слабого сигнала и то на пару делений, хотя должно

уходить все 4 или 5. Вроде бы ничего особенного в этой неточности нету, но на западе разыгрался целый скандал и Apple пришлось быстро исправлять проблему с датчиком сигнала. В июле 2010 года вышла версия 4.0.1 (и 3.2.1 iOS для iPad), она и исправляла проблему с делениями сигнала.

iOS 4.1 появилась в сентябре, в ней расширились игровые возможности аппаратов, добавились Game Center. Добавили возможность съёмки фотографий в качестве HDR и была добавлена возможность загрузки видеороликов на YouTube.

Следующие обновление iOS 4.2.1 привнесло папки и мультизадачность в iPad, добавили установку этой версии на все мобильные устройства (правда часть функций не работало на старых устройствах — iPhone 3G и iPod Touch 2G). Apple внедрила технологию для воспроизведения потокового видео AirPlay, оно работала между всеми девайсами Apple с Apple TV.



iOS 5 — 12 октября 2011.

Пятая версия мобильной операционки смогла похвастаться виртуальным ассистентом Siri, фирменным сервисом сообщений и облаком:

- Siri;
- Центр уведомлений;
- iMessage;
- iCloud;
- Синхронизация с iTunes по Wi-Fi;
- Включение новых устройств без подключения к компьютеру;
- Киоск;
- Напоминания;
- Интеграция с Twitter;
- Пробки и альтернативные маршруты на картах;
- Поддержка RTF в почте;
- Базовое редактирование снимков;
- Функция Reader в Safari.

В обновлениях появились:

- Измененное приложение Камера для iPad;

- Увеличение лимита скачивания приложений по 3G до 50 МБ;
- Удаление фотографий из Фото потока.

Эта версия операционной системы появилась с презентацией iPhone 4S в октябре 2011 года. Новых функций и возможностей добавилось очень много, но самые главные коснулись аппаратной части.

Технология Siri. Эта функций появилась в iOS 5.0 и была доступна только для iPhone 4S, она предоставляла возможность голосового контроля над телефоном, можно даже задавать вопросы, телефон будет пытаться отвечать на них. Только одна проблема – если вы не знаете английский язык, то у вас ничего не получится.

Погода и другая информация. В этом направлении iOS отставала от Android и в пятой версии разработчики решили эту проблему, добавив вывод на рабочий экран сведения о погоде, местонахождении, температуре воздуха.

iMessage. Apple решили позаимствовать интересную технологию у BlackBerry, которая позволяет обмениваться короткими сообщениями между устройствами с одной операционной системой. При установке iOS 5 на iPod Touch или iPad, эта бесплатная услуга будет также работать.

iCloud. Один из самых интересных нововведений в новой версии прошивки. Apple старается расширить функционал, хранения данных на удалённых серверах. iCloud и выполняет эту роль, он заменил старую версию MobileMe. Документы, фотографии и другие файлы теперь можно скопировать на хранение в облачное хранилище данных. Данные могут синхронизироваться между всеми Apple устройствами одного владельца.



iOS 6 — 11 июня 2012.

Самым значительным изменением в актуальной версии операционной системы стали собственные карты Apple, кроме того программисты немного «прокачали» Siri и добавили Passbook в качестве утешения всем, кто ждал NFC.

Возможности новой ОС:

- Фирменные карты Apple;
- Новые возможности Siri;
- Интеграция Facebook;

- Passbook;
- Синхронизация вкладок через iCloud;
- Улучшения приложения Mail;
- Звонки FaceTime посредством мобильного интернета;
- Общий Фотопоток;
- Функция «Не беспокоить»;
- Новые опции ответа на звонок;
- Поиск по всем полям в приложении Контакты;
- Полноэкранный режим в браузере в ландшафтной ориентации;
- Новый интерфейс виртуальных магазинов;
- Приложение Часы для iPad;
- Удалено приложение YouTube;
- Панорамная съемка.

Обновления iOS 6 не добавили существенных изменений и в основном были направлены на исправление ошибок и повышение стабильности.



iOS 7 — июнь 2013.

В июне 2013 года на WWDC представлена новая версия мобильной операционной системы. Джони Айв стал главой подразделения Human Interface и после обновления приложения подкасты можно оценить грядущие изменения интерфейса.

Apple называет операционную систему «самым большим изменением iOS после выхода оригинального iPhone». Наиболее заметным изменением является кардинально переработанный интерфейс. В рекламном ролике, показанном во время основной презентации, Джонатан Айв описал обновление как «внесение порядка в запутанность», подчеркивая следующие особенности: изысканные шрифты, новые иконки, прозрачность и слои. Дизайн iOS 7 и OS X Mavericks (представленной в тот же день) содержат значительно меньше элементов скевоморфизма (таких как зеленый стол в Game Center, деревянная текстура в приложениях Киоск и iBooks и кожи в Календаре).

«Установив iOS 7, получаешь новый телефон, но хорошо знакомый в использовании» Крейг Федериги.

Кроме того, произошли существенные изменения функциональности, которых было маловато в шестой версии прошивки.

Во время презентации новой операционной системы старший вице-президент Apple Крейг Федериги подчеркнул десять основных изменений:

- **Пункт управления.** Подобно Центру уведомлений, Пункт управления (Control Center) доступен с помощью жеста вверх в нижней части экрана и обеспечивает доступ к таким параметрам как авиа режим, яркость, управление мультимедиа, AirPlay, AirDrop и быстрый доступ к некоторым приложениям: фонарик, компас, калькулятор и камера.

- **Многозадачность.** iOS 7 основана на вытесняющей многозадачности, введенной в iOS 4. Поддерживается фоновое обновление приложений.

- **Safari.** В Safari стало доступно поле «умного» поиска, впервые представленное в Safari 6 для OS X Mountain Lion. Другие изменения включают в себя бесконечное число вкладок, родительский контроль, улучшенные обмен ссылками в Twitter и Список для чтения.

- **AirDrop.** С помощью функции AirDrop (впервые представлена в OS X Lion), доступной для iPhone 5, iPad (4-го поколения), iPad mini, и iPod touch 5-го поколения, можно обмениваться файлами (фотографиями, видео, карточками контактов и т.д.) с другими пользователями iOS-устройств.

- **Камера.** Новый интерфейс камеры позволяет переходить между четырьмя различными режимами: видео, фото, квадратные и панорамные фото. Также появилась возможность наложить на снимки один из девяти различных фильтров.

- **Фотографии.** Приложение «Фото» в iOS 7 использует информацию каждой фотографии, чтобы сортировать их по дате, месту и году съёмки, а также поддерживает загрузку видео в фотопоток (представлен в iOS 5).

- **Siri.** Интерфейс Siri также подвергся редизайну. Для каждого представленного языка появилась опция выбора между мужским и женским голосом. Улучшена интеграция с Твиттером, Википедией и поисковой системой Bing. Появилась возможность управлять системными настройками с помощью голосовых команд.

- **iOS в Машине.** «iOS в Машине» использует Siri, интегрированную в отдельные модели автомобилей для управления навигацией,

телефоном, музыкой и iMessage на экране автомобиля. Выход ожидается в 2014 году.

- App Store. Новые функции: поиск по возрастной группе, «Рядом со мной», а также реализовано автоматическое обновление приложений.

- Музыка и iTunes Radio. Наряду с изменениями интерфейса, в приложение Музыка теперь встроен iTunes Radio (сервис потокового воспроизведения музыки, наподобие Spotify). В США служба стала доступна после выхода бета-версии iOS 7. Запуск в Великобритании и Европе ожидается позже.

Другие изменения включают в себя FaceTime Audio, многостраничные папки, черный список контактов, а также новые возможности Find My iPhone, препятствующие использованию потерянного или украденного iPhone или iPad посторонними.

Операционная система Android



Android – операционная система для смартфонов и планшетов, а в последнее время и для фотоаппаратов, телевизоров, mini-pc и прочих гаджетов, разработанная компанией Google на базе ядра Unix, как бесплатная операционная система с открытым исходным кодом. Эта система занимает лидирующую позицию среди ОС для телефонов и является основным конкурентом iOS. Для неё, как и для iOS, разрабатывается множество приложений и игр, на данной платформе выпускаются как бюджетные телефоны от 1500 рублей, так и довольно продвинутые и современные аппараты. Лидером по производству смартфонов на этой платформе является компания Samsung, также на Android телефоны выпускают LG, HTC, Sony, Lenovo и многие другие компании.



Интерфейс современного Android.

Android 1.0 – начало.



Первая версия мобильной операционки появилась в далеком 2008 году, и поначалу работала исключительно в руках разработчиков. В некоммерческой версии операционной системы были реализованы:

- Android Market;
- Поддержка камеры;
- Приложения Gmail и синхронизация;
- Синхронизация контактов с приложением People;
- Синхронизация календаря с приложением Calendar;
- Google Maps, Sync, Search и Talk;
- IM, СМС, MMC;
- Медиаплеер;
- Уведомления;
- Обои рабочего стола;
- Голосовой набор;
- Плеер YouTube;
- Wi-Fi и Bluetooth.



Android 1.0.

Android 1.5 — Cupcake.



30 апреля 2009 года была официально представлена первая потребительская версия «зеленого робота». В Android 1.5 появилось много изменений, которые сделали инженерный прототип хоть немного похожим на операционку для пользователей:

- Поддержка виджетов;
- Загрузка видео на YouTube;
- Загрузка фотографий в Picasa;
- Автоматическая смена ориентации экрана;
- Анимированные переходы между рабочими столами;
- Копирование и вставка в браузере;
- Запись и просмотр видео;
- Поддержка Bluetooth A2DP, AVRCP;
- Клавиатура с предикативным вводом.



Android 1.5.

Android 1.6 — Donut.



15 сентября 2009 вышло в свет обновление операционной системы. Android нуждался в доработках стабильности, производительности и безопасности. Это стало отличным поводом расширить функциональность:

- Быстрый поиск;
- Галерея, камера;
- Поддержка CDMA, EVDO, 802.1x, VPN;
- Индикатор аккумулятора;
- Преобразование речи в текст;
- Поддержка движка многоязычного произношения;
- Облегченный поиск приложений и скриншоты в Android Market;
- Поддержка разрешения WVGA;
- Новые API и инструмент для работы с жестами.



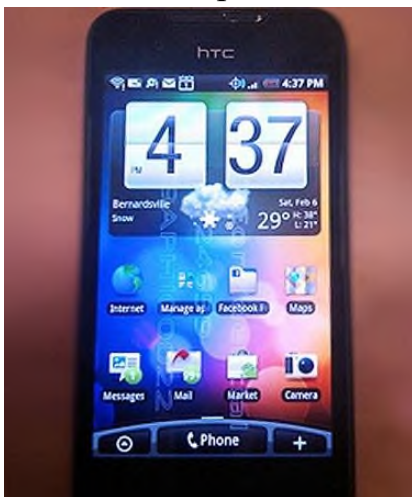
Android 1.6.

Android 2.1 — Eclair.



26 октября 2009 Google представил вторую версию мобильной операционной системы. В очередной раз было добавлено много необходимых функций, но операционка все еще была далека от совершенства:

- Поддержка нескольких аккаунтов;
- Поддержка Exchange и синхронизации почты с контактами;
- Bluetooth 2.1;
- Живые обои;
- Поиск по сохраненным СМС и ММС;
- Новые возможности камеры;
- Пошаговая навигация в картах Google;
- Поддержка различных размеров экрана и разрешений;
- Новые API;
- Поддержка HTML 5 в браузере.



Android 2.1.

Android 2.2 — Froyo.



20 мая 2010 появилось обновление операционной системы с многочисленными изменениями и нововведениями:

- Улучшенная поддержка Exchange;
- Движок Chrome V8 JS в браузере;
- Функция точки доступа;
- Отключение передачи данных по сотовой сети;
- Android to Cloud Device Messaging;
- Поддержка Flash;
- Поддержка высокого разрешения экрана;

- Поддержка многоязычной клавиатуры;
- Поддержка док-станций Bluetooth;
- Набор номера и передача контактов по Bluetooth;
- Фреймворки автомобильного и ночного режима.



Android 2.2.

Android 2.3 — Gingerbread.



6 декабря 2010 «зеленый робот» в очередной раз поумнел:

- Поддержка NFC;
- Поддержка новых размеров и разрешений экранов;
- Новый менеджер загрузок;
- Поддержка WebM/VP8, AAC и открытые API для нативного аудио;
- Клавиатура с мультитач;
- Улучшенная функция копирования и вставки;
- Видеочат в Google Talk;
- Поддержка SIP-телефонии;
- Новые аудиоэффекты.



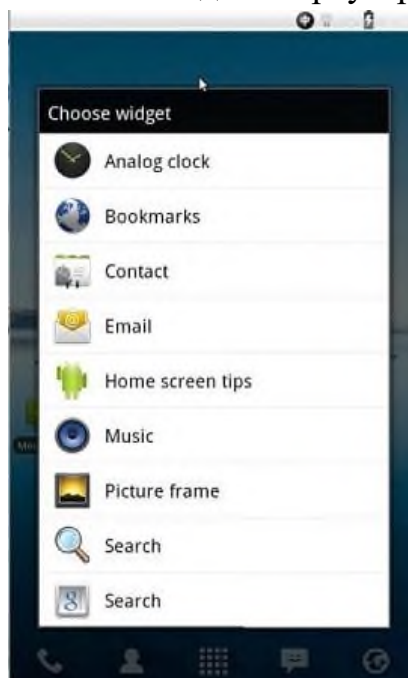
Android 2.3.

Android 3.0 — Honeycomb.



22 февраля 2011 появилась третья версия Android со значительными визуальными и функциональными изменениями:

- Новый интерфейс «holo»;
- Системная строка меню;
- Действия со строкой меню;
- Упрощенная многозадачность;
- Настраиваемый домашний экран;
- Новая клавиатура;
- Быстрый доступ к камере;
- Аппаратное ускорение;
- Шифрование пользовательской информации;
- Вкладки в браузере вместо окон.



Android 3.0.

Android 4.0 — Ice Cream Sandwich.



19 октября 2011 состоялся дебют «мороженки». Начиная с четвертой версии операционки Android стал все больше напоминать полноценную операционную систему. Теперь он достаточно стабильный, удобный и симпатичный, чтобы все больше людей могло сделать выбор в пользу устройств с операционкой Google:

- В панели многозадачности появились недавно запущенные приложения;
- Android Beam для NFC;
- Доступ к приложениям с экрана блокировки;
- Разблокировка экрана при помощи лица;
- Google Chrome;
- Предупреждения об использовании мобильного трафика;
- Остановка приложений в фоне;
- Встроенный фоторедактор;
- Изменение размера виджетов;
- Wi-Fi Direct;
- Единая платформа для смартфонов и планшетов;
- Скрытие уведомлений свайпом;
- Android Market переименован в Google Play.



Android 4.0.

Android 4.1 — Jelly Bean.



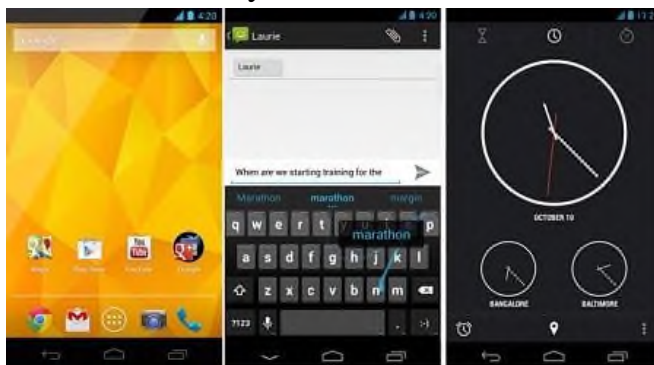
9 июля 2012 поисковый гигант представил новую версию мобильной операционной системы, самым главным нововведением которой стало то, что Android наконец-то перестал тормозить за счет внедрения технологии Project Butter. Интерфейс стал быстрым и плавным, этого вполне было бы достаточно, чтобы обрадовать пользователей. Впрочем, и на этот раз программисты подготовили много изменений:

- Google Now;
- Google Cloud Messaging;
- Vsync timing;
- Тройная буферизация;
- Многоязычная поддержка;
- Расширяемые уведомления;
- Передача данных через Android Beam;
- Диктовка офлайн;
- Многоканальное USB аудио;
- Действия с уведомлениями;
- Виджеты на экране блокировки;
- Поддержка профилей пользователей.



Android 4.1.

Android 4.2 Jelly Bean.



В данной версии произошли следующие изменения:

- Одним из основных нововведений в Android 4.2 стала поддержка профилей, теперь на одном устройстве может использоваться несколько

учетных записей, каждая со своими настройками, программами и личной информацией;

- Клавиатура теперь поддерживает жесты для ввода по типу популярного Swype. Для ввода слов теперь можно водить пальцем по клавиатуре выбирая необходимые буквы, а встроенный словарь будет стараться предоставлять нужные варианты. В целом, по заявлению Google, в Android 4.2 словари стали более точными и полными;

- В приложении «Камера» появился новый режим съемки под название Photo Sphere, он позволяет создавать панорамы 360 градусов и публиковать их в Google+ или же в Google Maps, создавая таким образом свою версию Street View;

- появилась поддержка беспроводной передачи видео и игр на совместимые телевизоры по технологии Miracast;

- В панели оповещений теперь есть доступ к меню быстрых настроек;

- Появился новый режим ожидания Daydream, когда устройство подключено к док-станции или находится в режиме сна, оно может отображать на дисплее разную полезную информацию, новости, погоду или же просто фотографии из альбома;

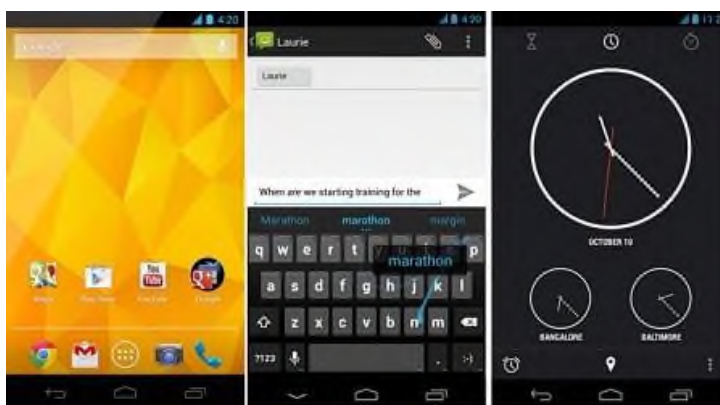
- Google Now также получило небольшое улучшение, теперь оно может сканировать почту Gmail на предмет нахождения различного релевантного контента, такого как билеты на самолет или приглашения на встречу, и создавать карточки с напоминанием;

- Появилась возможность размещать виджеты на экране блокировки;

- Обновлён интерфейс камеры;

- Обновление ядра Linux до ветки 3.4;

- Удален классический планшетный интерфейс, используемый в 3.0 — 4.0.4.



Android 4.2.

Android 4.3 Jelly Bean.



Изменения:

- Keep и Hangouts теперь будет включен в прошивку и Google Apps по умолчанию;
- Ошибка под названием MasterKey исправлена;
- Smart или Bluetooth 4.0 LowEnergy включен в новую прошивку, что повышает энергоэффективность устройства при работе по данному протоколу;
- В области уведомлений теперь показаны все работающие приложения, даже в фоновом режиме. Отключить отображение значка в панели можно в настройках;
- В настройках можно установить постоянную работу Wi-Fi для улучшенного гео-позиционирования;
- AVRCP 1.3, доступный совместно с Bluetooth Smart, позволяет при помощи Bluetooth объединять два устройства и использовать их как управляющие друг другом;
- Доступна новая камера и галерея;
- Play версии 4.2.3;
- Система теперь поддерживает OpenGL ES 3.0;
- API был повышен до 18 версии.



Android 4.3.

Android 4.4 KitKat.



Русское название обновления: «KitKat». 31 октября 2013 компания Google представила следующую версию Android 4.4. Изменения коснулись интерфейса. Также, Android оптимизировали для работы на бюджетных смартфонах с оперативной памятью 512 Мбайт.

Одновременно с новой версией Android Google также представил новый смартфон Google Nexus 5.

Нововведения версии:

- Постоянно активный голосовой помощник. Прикосаться к экрану не нужно, достаточно лишь произнести «Ok Google», затем можно дать команду или продиктовать, что нужно найти;
- Отображение обложек и кнопок управления на экране блокировки при воспроизведении музыки или показе фильмов через Chromecast;
- Кнопки навигации и панель уведомлений автоматически скрываются;
- Более быстрое переключение между задачами и оптимизированное распределение памяти;
- Приоритетность в телефонной книге. Контакты, с которыми пользователь общается чаще всего, показаны вверху списка. Прямо в приложении «Контакты» можно искать людей, места на карте и организации;
- Умный определитель номера. Если номер не найден в «Контактах», телефон попытается определить по данным Google Maps, из какой организации звонят;
- Центр общения. Приложение Hangouts позволяет вести переписку в чате, совершать и принимать видеозвонки, отправлять и получать SMS и MMS-сообщения и делиться GIF-анимацией;
- Японские смайлики. В стандартную клавиатуру добавлены красочные миниатюрные картинки Emoji;

- Поддержка облачных принтеров. Фотографии, документы и веб-страницы можно распечатывать на принтерах, подключенных к сервису Google Cloud Print, и на других принтерах, которые поддерживают печать через мобильные приложения;

- Быстрое сохранение файлов в облако. Некоторые приложения (например, обновленный QuickOffice) поддерживают моментальную отправку файлов в Диск Google;

- Поддержка Message Access Profile в автомобилях, оснащенных модулями Bluetooth;

- Поддержка Chromecast;

- Запуск веб-приложений через Chrome;

- Отображение субтитров к фильмам в стандартном видеоплеере;

- Встроенный сервис «Удаленное управление Android»;

- Обновленный дизайн загрузчика файлов с сортировкой и настройкой способа отображения скачанных файлов: списком или миниатюрами;

- Переключение между установленными лаунчерами через системные настройки;

- Обновленное приложение электронной почты с папками, фотографиями аккаунтов и улучшенной навигацией;

- Поддержка приложений, использующих инфракрасный порт;

- Доступ к настройкам определения местоположения через значок в «Быстрых настройках»;

- Настройка способа определения местоположения: точное или с меньшим расходом батареи. В системных настройках можно посмотреть, какие приложения пытаются определить местоположение;

- Запуск приложений в песочнице Security-Enhanced Linux;

- Поддержка шагомеров;

- Осуществление NFC-платежей через Google Wallet и другие платежные системы. Статистика хранится в облаке или памяти устройства;

- Экспериментальный предкомпилятор Android Runtime (ART);

- Режим, который автоматически скрывает лишние данные на экране устройства во время игры, чтения или просмотра видео.



Android 4.4.

Операционная система Windows Phone



Данная система разработана компанией Microsoft для смены устаревшей мобильной операционной системы Windows Mobile, которая плохо подходила для сенсорных устройств. Операционная система спроектирована с нуля, поэтому у неё не очень много доступных приложений. Некоторым интерфейс данной операционной системы нравится, некоторым нет, как говорится, "О вкусах не спорят".



Интерфейс современного Windows Phone 8 (8.1).

История появления.



Windows SE.

В 1996 году появилась первая версия Windows CE, которая была урезанной версией Windows 95. Работы над мобильной и настольной операционкой велись совместно вплоть до 2000 года. Тогда их пути разошлись и мобильная операционка стала по-настоящему мобильным продуктом. В июне 2000 года была представлена третья версия Windows CE с кодовым названием «Cedar». Она легла в основу операционных систем Pocket PC 2000 и Handheld PC 2000, чуть позже были выпущены Pocket PC 2002 и Smartphone 2002.

В 2003 году Pocket PC отправилась на свалку истории, и была представлена Windows Mobile 2003.



Windows Mobile 2003 — 23 июня 2003.

В 2003 году появилась первая мобильная операционная система из семейства Windows Mobile. До этого момента в мобильной операционке программисты реализовали основной набор софта для работы вне офиса, развлекательные приложения, например Windows Media Player 8 и модули беспроводных соединений.

- L2TP/IPsec VPN;
- Улучшенная поддержка Bluetooth;
- Приложение для просмотра картинок с функцией обрезания и пересылки файлов;
- Игра Jawbreaker;
- Опция отправки СМС в ответ на звонок в Phone Edition.



Windows Mobile 2003 SE — 24 марта 2004.

Обновление мобильной операционки было направлено на расширение модельного ряда устройств за счет поддержки новых разрешений экрана.

- Смена ориентации экрана;
- Разрешение экрана 640x480, 240x240, 480x480;
- Wi-Fi Protected Access;
- Просмотр страниц в одну колонку в Pocket Internet Explorer.



Windows Mobile 5 — 9–12 мая 2005.

На Developers Conference 2005 в Лас Вегасе была представлена новая версия мобильной Windows, в которой появилось огромное количество изменений и операционка избавилась от своего Pocket PC-прошлого в названиях приложений.

- Поддержка .NET Compact Framework 1.0 SP3;
- Среда для программ, основанных на .NET;
- Office Mobile;
- Windows Media Player 10 Mobile;
- Photo Caller ID;
- Улучшенная поддержка Bluetooth;
- ActiveSync 4.0;
- Интерфейс работы с GPS;
- Поддержка Persistent Storage;
- Microsoft Exchange Server;
- DirectPush.



Windows Mobile 6 — 12 февраля 2007.

Обновленная операционка была представлена в Барселоне на выставке 3GSM World Congress 2007, а в ее основу легла Windows CE 5.2. Windows Mobile 6 получила много новых функций, стала намного более производительной. Под управлением этой операционки появился HTC Touch:

- Первый смартфон, ориентированный на управление пальцем, а не пером;
- Поддержка IP-телефонии;
- Marketplace;
- Интеграция Windows Live;
- Шифрование карты памяти;
- Обновленный интерфейс;
- Windows Update;
- Exchange 2007;
- Microsoft Office OneNote;
- Поддержка HTML в Outlook Mobile;
- Новые разрешения экранов 320x320 и 800x480.



Windows Mobile 6.1 — 1 апреля 2008.

На выставке CTIA Wireless 2008 было представлено небольшое обновление операционной системы — Windows Mobile 6.1. В нем не было ничего особенного, но именно с устройств под управлением этой операционки началось распространение не только среди бизнес-сегмента, но и обычных пользователей.

Под управлением WM 6.1 вышел Samsung WiTu, получивший нестандартный на тот момент широкоформатный экран и ставший первым смартфоном Samsung с оболочкой TouchWiz:

- Масштаб и обзор страницы в Internet Explorer;
- Шифрование файлов на устройстве;
- Mobile Device Manager.



Windows Mobile 6.5 — 6 октября 2009 года.

В октябре 2009 состоялась долгожданная премьера Windows Mobile 6.5, которая получила существенное обновление функциональности. Одним из главных нововведений стала оптимизация интерфейса для управления пальцами. Под управлением WM 6.5 появился легендарный HTC HD2 — первый и единственный с поддержкой мультитач и до сих пор актуальный. Производителю удалось создать устройство, на котором энтузиастам удалось запустить практически все мобильные операционные системы, например, Android, Windows Phone 7, Ubuntu, MeeGo и даже Windows RT.

- Интерфейс для управления пальцами;
- Новый домашний экран Titanium;
- Новое меню;
- Internet Explorer Mobile 6;
- A-GPS.



Windows Mobile 6.5.3.

Долгая история Windows Mobile подошла к концу вместе с обновлением до версии WM 6.5.3. В ней было не много изменений и большинство из них

были косметические, которые завершили переход от управления смартфонами стилусом к интерфейсу, ориентированным пальцем:

- Крупная иконка Пуск;
- Увеличенный нижний и уменьшенный верхний бар;
- Новые меню стандартных программ.



Windows Phone 7 — 15 февраля 2010.

В Барселоне на Mobile World Congress 2010 Стив Балмер анонсировал совершенно новую мобильную операционную систему — Windows Phone 7. Официально новую разработку представили 11 октября а уже 21 октября в продаже появились первые смартфоны. На самом деле бренд «Windows Phone» начали продвигать еще в Windows Mobile 6.5, но в 2010 году он начал работать в полную силу. Новая операционка полностью лишилась совместимости с предыдущими версиями, получила кардинальное обновление интерфейса, полноценный магазин приложений и многое другое.

- Пользовательский интерфейс;
- Магазин приложений;
- Закрытая файловая система;
- Internet Explorer Mobile;
- Microsoft Office Mobile;
- Многозадачность;
- Синхронизация Zune.



Windows Phone 7.5 — февраль 2011.

На Mobile World Congress 2011 Стив Балмер представил первое крупное обновление Windows Phone. Программисты сделали более 500 улучшений и

дополнений, которые положительно сказались на стабильности, быстродействии и удобстве операционки:

- Internet Explorer Mobile 9;
- Поддержка в браузере HTML5, Canvas, аппаратного ускорения;
- Поддержка фронтальной камеры;
- Поддержка 19 новых языков;
- Многозадачность в сторонних приложениях;
- Windows Live SkyDrive;
- Интеграция Twitter в People Hub.



Windows Phone 7.5 Tango — февраль 2012.

На Mobile World Congress 2012 мобильная операционка получила небольшое обновление, в котором уменьшились требования к «железу» для захвата рынков развивающихся стран недорогими смартфонами, например, Lumia 610:

- Поддержка процессоров с частотой 800 МГц;
- Поддержка устройств с 256 МБ оперативной памяти;
- Исправление ошибок.



Windows Phone 7.8 — конец 2012.

В конце 2012 года было объявлено об обновлении смартфонов до версии 7.8, которая получила часть функций новой, не совместимой со старыми устройствами, Windows Phone 8:

- Обои Bing на экране блокировки;
- Виртуальная кнопка спуска затвора в приложении Камера;

- Новые типы плиток;
- Новые цветовые схемы.



Windows Phone 8 — 29 октября 2012.

Windows Phone 8 — второе поколение Windows Phone. Операционка получила новую архитектуру Windows NT, что лишило владельцев старых смартфонов надежды на обновление и запуска приложений, созданных для 7.8:

- Живые плитки;
- Новый экран блокировки;
- Data Sense;
- Kid's Corner;
- Интеграция с SkyDrive;
- Internet Explorer 10;
- NFC и Wallet;
- Скриншоты.



Windows Phone 8.1 — 17 октября 2013.

В 2013 году вышла новая версия операционной системы Windows Phone 8.1, которая направлена на слияние настольной, планшетной и смартфонной «восьмерок». Началось полноценное объединение версий, приложения будут поставляться в качестве универсальных установщиков для всех устройств. Кроме того:

- Появился «Центр уведомлений»;
- Улучшена работа Data Sense;
- Поддержка USB On-The-Go
- Поддержка радио.

Заключение

Время подвести итоги и выделить отличительные черты каждой из рассмотренных операционных систем:

- iOS – закрытая система. Нет поддержки MicroSD-карт, зато есть магазин App Store. К плюсам можно отнести скорость работы (система способна работать даже на не самом мощном «железе») и множество приложений, в том числе написанных только под эту ОС. Устройства Apple изначально отличались дизайном, а их стоимость выше конкурентов.

- Android – открытая система. Очень много как устройств, на которые она установлена, так и производителей смартфонов (Samsung, HTC, Huawei, Oppo, LG, Lenovo, Meizu, Sony, Zopo, Jiayu, THL, Alcatel, Fly и прочие). Система с каждым обновлением становится все удобнее, в последних версиях появилась «плавность» работы, а по количеству приложений она лишь немного уступает iOS. В России стоимость аппарата на Android начинается с планки 1500 рублей. Это главный конкурент системы iOS.

- Windows Phone – закрытая система. В ее достоинства можно записать большое количество приложений, интересный интерфейс в виде «плиток», разнообразный модельный ряд, представленный смартфонами HTC, Nokia, LG, Samsung, Dell и прочими.

Каждая система хороша по-своему, но то, какую систему или устройство выбрать, решать уже конечному пользователю.

Активными игроками на рынке смартфонов были и являются Sony (Sony Ericsson), Nokia, HTC, BlackBerry, Motorola. С недавних пор на пятки им начали наступать Apple, Samsung, LG и Meizu. Можно отметить и очень активное развитие китайских смартфонов, хотя некоторые из них отличились не самыми лучшими программным обеспечением и поддержкой. Чтобы устройство работало стабильно, иногда приходится искать альтернативные прошивки (Android), чтобы дольше работала батарея, чтобы работал Wi-Fi, звонки, камера. Поэтому такие качества операционной системы, как стабильность, удобство, плавность, поддержка, многообразие приложений являются обязательными условиями для пользователей современных гаджетов.

До недавних пор лидерами были Symbian и BlackBerry. Теперь же в лидирующую тройку вошли такие игроки как iOS Apple, Android Google и Windows Phone Microsoft.

Лабораторная работа №2

Эмуляторы, установка любого эмулятора, установка и запуск на нем программы.

Цель работы: Знакомство с эмуляторами, установка и запуск программы.

Далеко не все знают о том, что существуют специальные программы-эмуляторы, при помощи которых можно запускать и полноценно работать с приложениями для Android на ПК с операционной системой Windows. Они могут быть полезны тем людям, которые не обладают смартфоном или же используют портативное устройство на базе прочих "операционок" (iOS, Windows Phone).

При установке практически всех современных Android-эмуляторов на ваш компьютер будет установлен ряд драйверов, которые необходимы для их корректной работы. Обязательно подтвердите установку каждого, если этого "попросит" установщик. Для того, чтобы запустить необходимое приложение будет достаточно скачать APK-файл и открыть его в эмуляторе. Некоторые программы даже предоставляют доступ к Play Market, тем самым давая возможность скачивать игры и программы "напрямую".

Из прочих особенностей, которые присущи практически всем современным эмуляторам Android, можно выделить возможность настройки соотношения сторон "рабочего" окна, ассоциацию файлов APK с программой, поддержку одновременной работы сразу с несколькими играми и приложениями, а также выделение определенного количества "ресурсов" компьютера для работы эмулятора. Сразу отметим, что некоторые приложения, запущенные в подобных программах, могут "фризить" или работать медленнее из-за проблем, не связанных с недостатком мощности вашего ПК.

Рано или поздно наступает момент, когда пользователь понимает, что его мобильный гаджет уже не соответствует определенным требованиям, и не «потянет» только что вышедшую новую игрушку. Но руки-то чешутся испытать ее как можно скорее! В таких случаях можно воспользоваться эмуляторами Android, развернув нужную версию операционки прямо на рабочем столе персонального компьютера.

Или, к примеру, смартфон находится в ремонте, а терять прогресс в играх и общение в мессенджерах категорически не хочется. Здесь также можно прибегнуть к помощи виртуального гаджета. Да и честно говоря, смотреть в большой монитор все же приятней, чем напрягать глаза в маленький экран.

И совсем не стоит забывать о категории блоггеров, несущих в массы разумное доброе, вечное. А именно прохождения мобильных игр, и выкладывающих ролики с ними на свои каналы YouTube. Для них эмулятор — профессиональный инструмент захвата потокового видео.

Если с назначением программ-эмуляторов более-менее понятно, то вопрос: «Какую установить?» многих может поставить в тупик. Далее перейдем непосредственно к кандидатам, способным заменить собой мобильный гаджет.

MEmu Play

Версия программы предоставляется совершенно бесплатно, скачать дистрибутив можно с [сайта разработчика](#). Эмулятор порадует пользователей своей полной русификацией, причем не только переведенными пунктами меню, но и глубокой проработкой пользовательского интерфейса внутри самой оболочки.

Для запуска эмулятора потребуется как минимум 1 ГБ оперативной памяти, видеокарта, поддерживающая OpenGL 2.0 и процессор, понимающий технологии виртуализации.

После установки MEmu пользователь получает полноценную версию мобильного устройства, дополненную различными удобными «фишками»:

- возможностью ввода сообщений на кириллице с внешней клавиатуры;
- общим доступом к локальным папкам на компьютере;

- назначением клавиш физической клавиатуры для управления функциональными кнопками и областями виртуального экрана;
- эмуляцией модуля GPS;
- установкой приложений (*.apk) одним кликом мышки.

BlueStacks

Обновленная, четвертая версия эмулятора скачивается с [официального сайта](#) разработчика. По сравнению с предшественниками, стала проще, понятнее, и что самое важное - шустрее.

После установки и начальной настройки получаем полноценную копию своего гаджета на рабочем столе. В программе работает Google Play магазин, из которого доступно к установке огромное количество игр и приложений.

Интересной особенностью эмулятора является своя собственная игровая валюта, которая начисляется за скачивание и запуск приложений из магазина. Монетизировать ее не получится, а вот потратить в том же магазине на приобретение дополнительных «плюшек» — всегда пожалуйста.

Важно понимать, BlueStacks — довольно требователен к ресурсам ПК. Чтобы получить сносную графику (до 30 fps), рабочая станция должна иметь на борту 4-6 ГБ ОЗУ и как минимум двухъядерный процессор с тактовой частотой 1,8-2,5 ГГц и поддержкой виртуализации.

Именно по этой причине на слабом компьютере можно получить «подлагивание» эмулятора и его беспричинное прекращение работы.

NOX Player

Еще один эмулятор Android для запуска приложений и игр на стационарном компьютере. Получить бесплатную версию программы можно по [адресу](#). Девиз разработчика: «Чем проще, тем лучше», поэтому интерфейс программы прост и интуитивно понятен, но в то же время наделен достаточно богатым функционалом. В арсенале эмулятора:

- настраиваемые клавиши управления;
- возможность создания скриншотов и записи потокового видео;
- виртуальный GPS;
- регулировка громкости;
- наличие Root-прав для виртуального девайса.

К сильным сторонам эмулятора относятся: стабильная и довольно шустрая работа, невысокие системные требования. По заявлению разработчика, минимум, что надо иметь в системном блоке: процессор с поддержкой виртуализации, 512 МБ ОЗУ и 1 ГБ свободного места на жестком диске. Этого вполне достаточно для запуска мессенджеров и простых 2D-приложений. Для запуска «тяжелых» игрушек потребуется значительно больше вычислительных ресурсов.

Remix OS Player

Уже из названия видно, что, скачав данный эмулятор, пользователь получает полноценную операционную систему. Получить бесплатную версию эмулятора можно с [сайта производителя](#). Внешне интерфейс программы очень схож с NOX Player, и в этом нет ничего удивительного, ведь NOX разрабатывался как облегченная и упрощенная версия Remix OS.

Инсталляционный пакет на сайте разработчика имеет внушительные 780 МБ. Приложение предъявляет низкие системные требования к железу (ОЗУ от 1 ГБ и двухъядерный процессор с технологией виртуализации), но в то же время наделено хорошим быстродействием. В виртуальной среде существует возможность запуска большинства «тяжелых» игр с достойными показателями fps.

Существенным недостатком эмулятора является его категорическое нежелание работать с процессорами AMD. Чтобы насладиться богатым функционалом, системой должен управлять центральный процессор производства Intel.

Genymotion

Единственный софт из представленной подборки, предназначенный для профессионального использования. Если у предшественников во главу угла ставятся игровые возможности, то в случае с Genymotion, главенствуют механизмы тестирования приложений. Простая, но надежная система с высокой производительностью — хорошее тому подспорье.

Эмулятор работает в связке со средой разработки приложений Android Studio, и в последующем не требует потери времени на компиляцию исполняемого файла и его установку на устройство. Разработчик сразу может видеть в окне программы результат проделанных изменений.

Единственное из представленной подборки приложение, которое для получения инсталляционного файла, требует прохождения несложной процедуры регистрации на [официальном сайте](#) разработчика. Использование эмулятора платное. Для частника он обойдется в \$136 в год, для организаций от \$412 в год за каждую лицензию.

Еще одну сложность при первичной настройке приложения вызывает процесс интеграции Genymotion в Android Studio. Для этого необходимо дополнительно скачать плагин и "внедрить" его в среду разработчика. На выходе получаем мощный пакет для оперативного тестирования созданных под Android приложений.

Установка Андроид на ПК

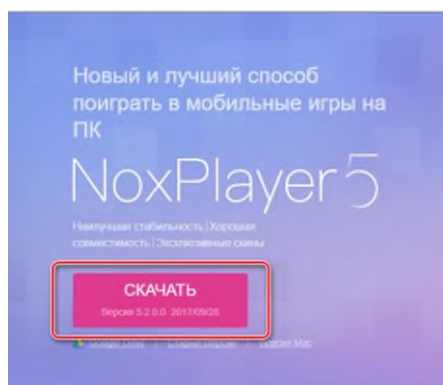
Операционная система Android в настоящее время развилась настолько, что многие пользователи смартфонов или планшетов не могут воспользоваться ею по максимуму из-за недостаточно производительной «начинки» своего устройства. Поэтому, чтобы поиграть в требовательные игры или воспользоваться некоторыми нужными программами, созданными под Андроид, были разработаны эмуляторы данной ОС. С их помощью вы с персонального компьютера или ноутбука сможете зайти в учетную запись [Play Market](#), скачать любое приложение или игру и пользоваться всеми их возможностями.

Устанавливаем Андроид на компьютер

Рассмотрим погружение в виртуальный мир Android с компьютера на примере эмулятора Nox App Player. Программа бесплатна и не имеет никакой навязчивой всплывающей рекламы. Работает на Андроиде версии 4.4.2, позволяя открывать множество игр, будь то большой симулятор, требовательный шутер или любое другое приложение.

Шаг 1: Скачивание

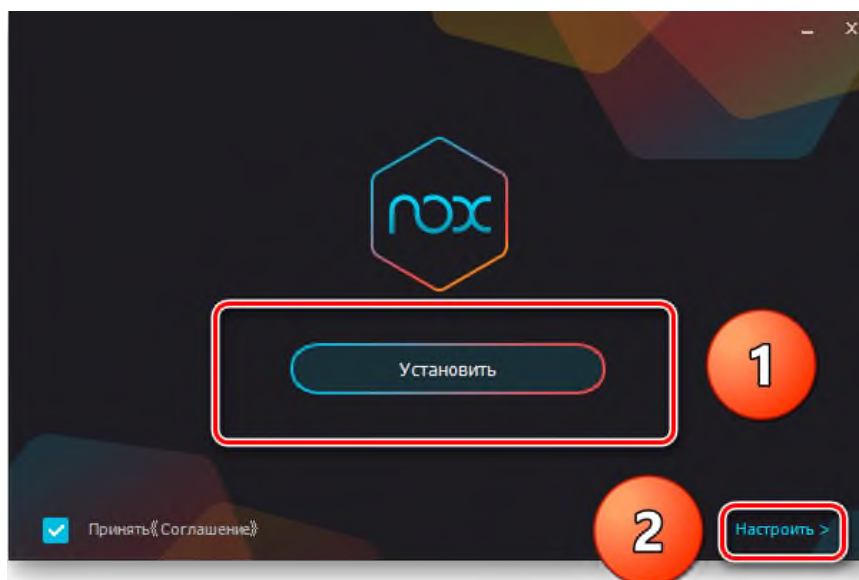
1. Переходим на официальный сайт разработчика по ссылке выше.
2. Для того чтобы установить эмулятор Nox App Player, нажимаем на кнопку **«СКАЧАТЬ»**.



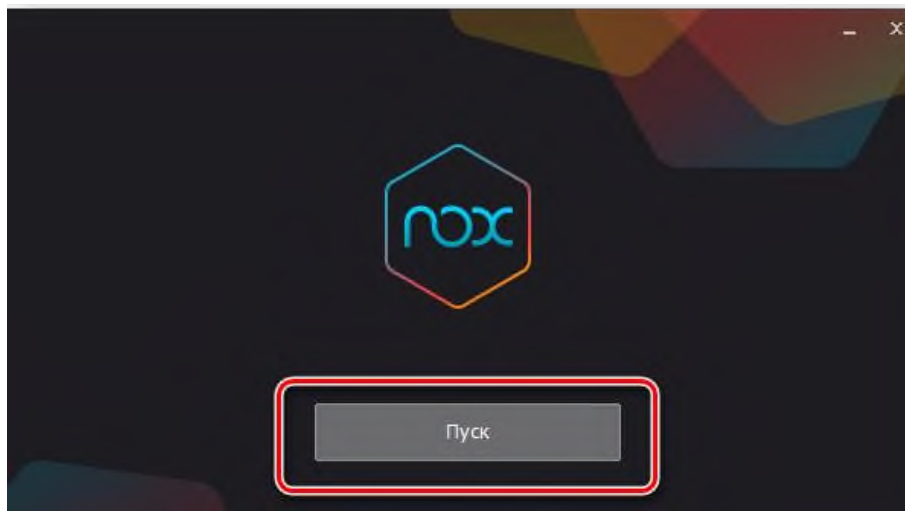
3. Далее начнется автоматическая загрузка, по завершении которой необходимо будет перейти в папку **«Загрузки»** и нажать на установочный файл скачанной программы.

Шаг 2: Установка и запуск программы

1. Для продолжения установки необходимо в открывшемся окне нажать на кнопку **«Установить»**. Выберите дополнительные параметры инсталляции, нажав на кнопку **«Настроить»**, если вам это необходимо. Не снимайте галочку с пункта **«Принять «Соглашение»»**, иначе вы не сможете продолжить.



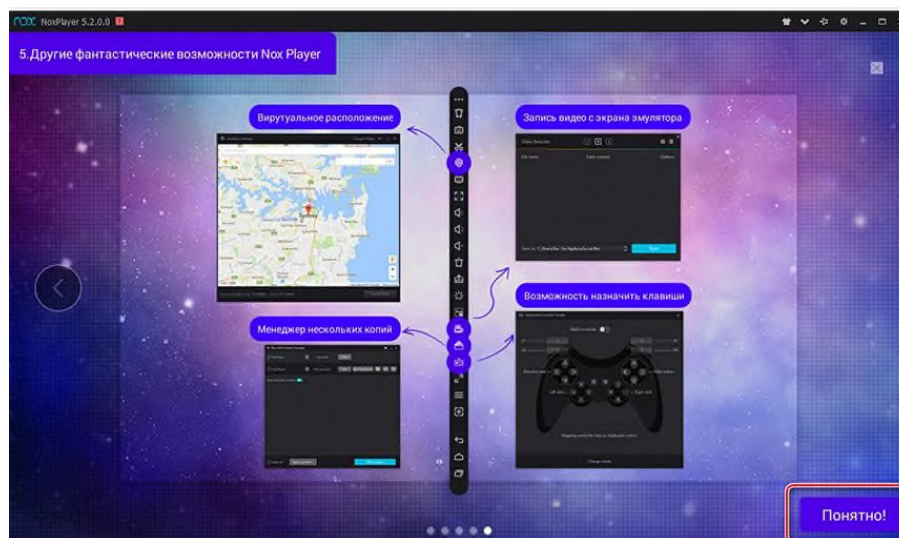
2. После того как эмулятор будет установлен на компьютер, вы увидите на экране окно запуска, где необходимо будет нажать на кнопку **«Пуск»**.



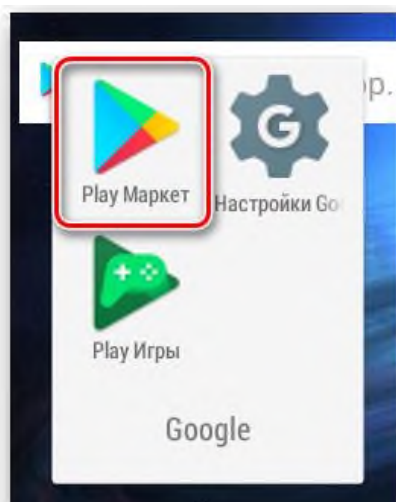
3. Ознакомьтесь с небольшой инструкцией по работе в программе, нажимая на кнопки в виде стрелочек.



4. Далее нажмите на кнопку «*Понятно*» в нижнем правом углу.

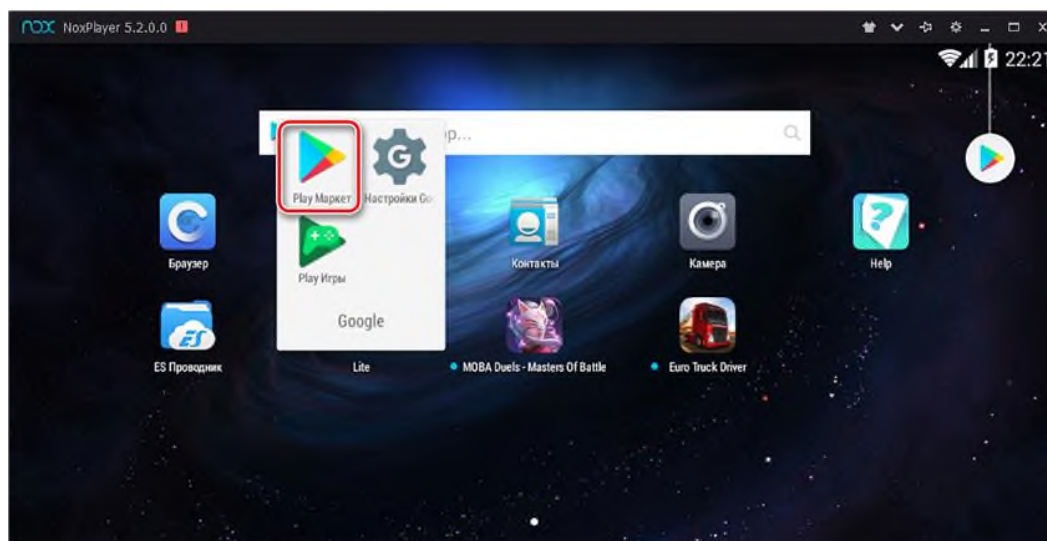


Все, на этом этапе установка эмулятора Nox App Player завершена. Для полноценной работы программы вам необходимо будет зайти в свой аккаунт Play Market — нажмите на иконку приложения в папке Google, введите логин и пароль от вашей учетной записи.

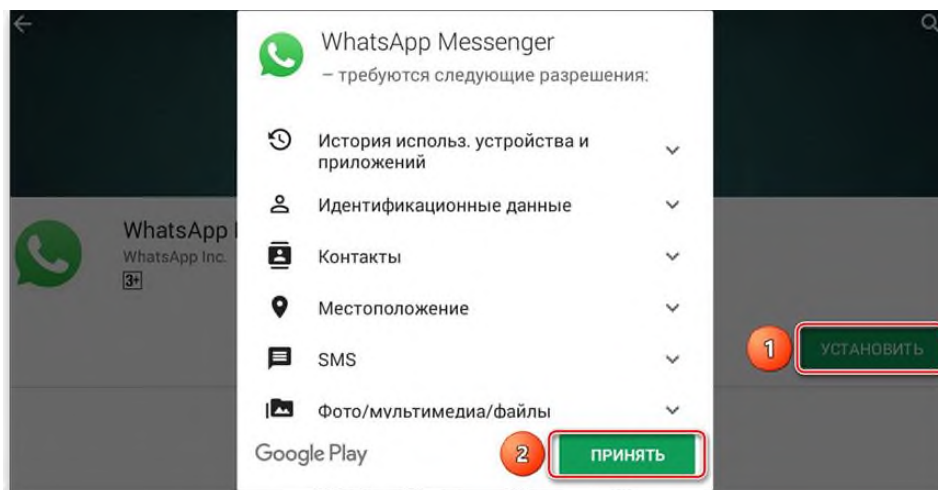


Шаг 3: Загрузка и установка приложений

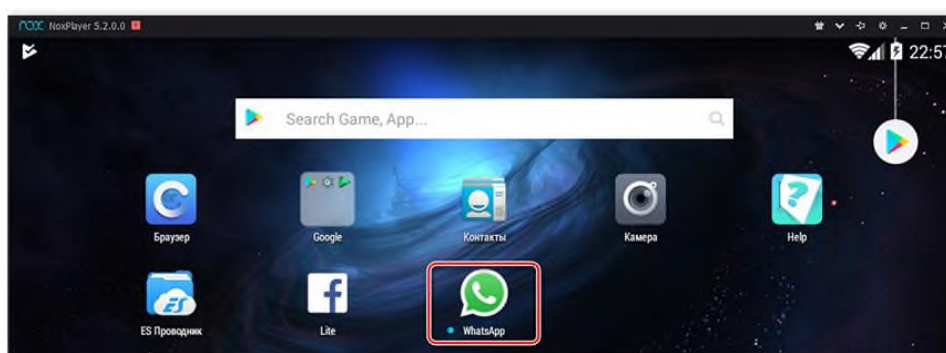
Nox Player может похвастаться полной совместимостью с операционными системами Mac OS и Windows, начиная от XP и до крайней «Десятки». А встроенный Play Market позволит вам прокачивать показатели в играх под своей учетной записью Google.



Для того чтобы установить необходимое приложение, вам нужно ввести его название в строку поиска в приложении Play Market, выбрать его, нажать кнопки «**Установить**» и «**Принять**». На изображении ниже данная процедура показана на примере популярного мессенджера [WhatsApp](#).



После установки иконка приложения появится на рабочем столе эмулятора. Вам остается зайти в него и пользоваться по назначению.



Теперь вы сможете открыть все игры и приложения, доступные для смартфонов, на вашем ПК в режиме полного экрана. Если у вас имеется веб камера и микрофон, то они самостоятельно подстроятся под приложения, где присутствует возможность общения по аудио или видеоканалу.

В эмулятор, помимо контента из Play Market, можно загружать игры и приложения прямо с компьютера. Для этого вам необходимо скачать файл приложения в формате **APK** и просто перетащить его на рабочий стол Nox App Player. После этого сразу начнется установка, по окончании которой вы увидите значок этого приложения на главном экране. Таким образом, как и на смартфоне, вы можете устанавливать приложения двумя способами.

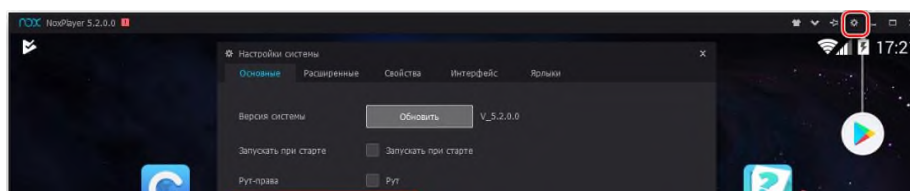
Шаг 4: Применение различных настроек

Эмулятор имеет большое количество настроек, которые расположены на правой стороне окна Плеера. Для удобства использования клавиатуры, мышки или контроллера в играх вы найдете эмуляцию нажатий и конфигурацию контроллера. Не обошлось и без возможности записи игрового процесса и скриншота окна.

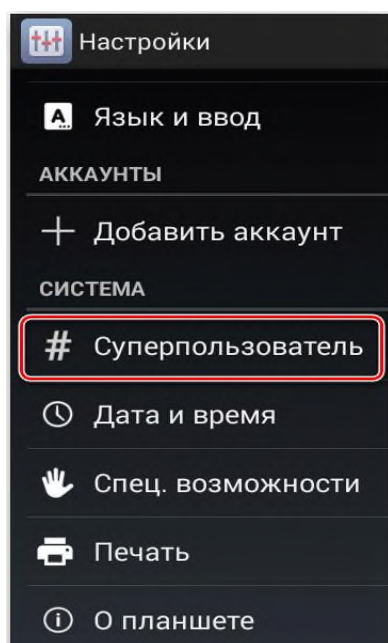
В некоторых играх требуется встряхивать ваше устройство – об этом тоже не забыли и добавили такую функцию в панель настроек. Еще в плеере предусмотрен поворот экрана, что очень удобно в некоторых играх или приложениях. Наличие режима «**Мультиплеер**» позволит вам пользоваться возможностями Плеера в несколько окон. Для активации каждой из этих функций достаточно нажать на соответствующую кнопку в панели настроек эмулятора Nox App Player.



Для тех, кто хочет попробовать в среде эмулированного Android Root-права, Nox App Player может дать и такую возможность. Для активации режима «Суперпользователь» нужно перейти в настройки Плеера в правом верхнем углу и поставить галочку напротив соответствующей позиции.



После активации этой функции вы можете в настройках Android испытать все возможности Root.



Таким образом вы сможете полноценно пользоваться оболочкой Android на своем компьютере. В интернете существует немало эмуляторов, которые имеют схожие параметры и функции, так что просто выбирайте подходящий и смело ставьте его на свою систему. Но не стоит забывать о возможностях вашего ПК. Если вы имеете старенький компьютер, предназначенный для офисных задач, то поиграть в требовательные игры будет сложно.

Лабораторная работа №3.

Платформы для поиска мобильных приложений. Play Market, интернет магазины Apple Store.

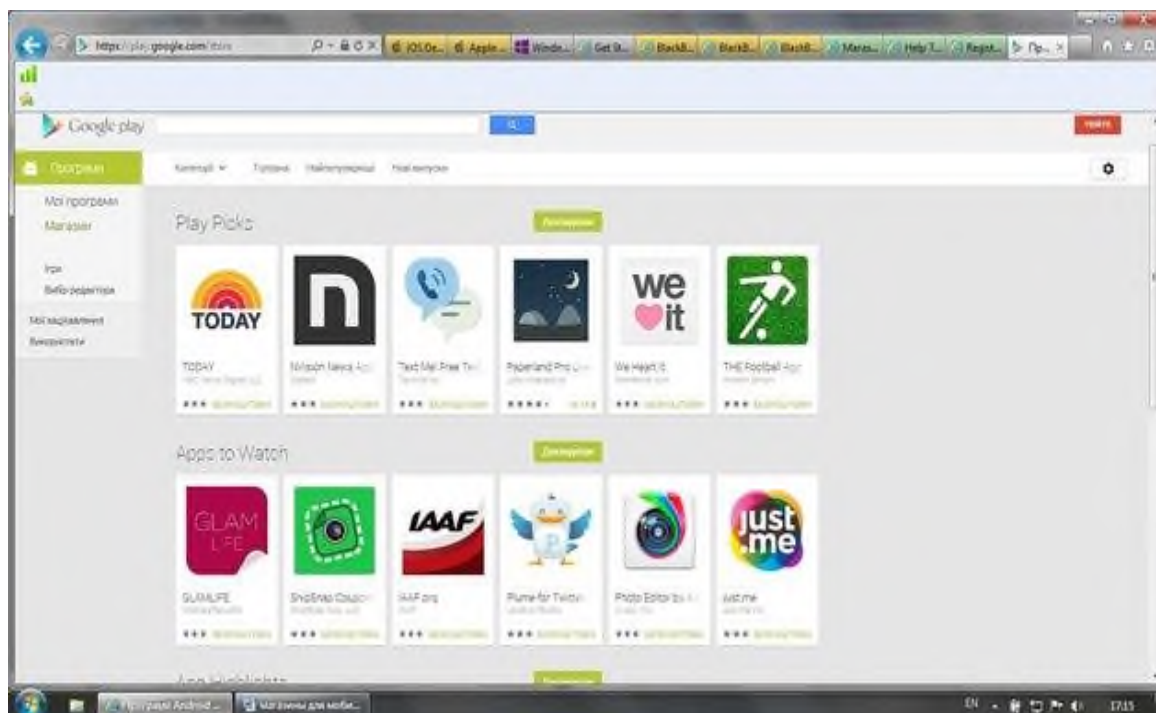
Цель работы: Ознакомление с интернет магазинами и выбор платформы.

Бум смартфонов и планшетов делает из приложений для них золотую жилу. Заработать можно как на продаже собственно приложений, так и на рекламе в них. Продаются мобильные приложения на нескольких торговых площадках, основные – App Store и Google Play.

Первым в этой нише был запущен магазин [Apple App Store](#), онлайн-сервис по продаже различного программного обеспечения для владельцев плееров iPod, телефонов iPhone и планшетных компьютеров iPad, выпускаемых компанией Apple на платформе iOS. Фактически он начался со специализированного отдела iTunes Store, «торговавшего» контентом для iPod. С 2008 года App Store остается лидером продаж, по итогам 2012 г. [приложений](#) в нем было продано на 3,45 млрд. евро.



Второй по величине всемирно известный магазин мобильных приложений – [Google Play](#), предлагающий программное обеспечение для планшетов и смартфонов, разработанных на платформе Android. Конкурента «яблочному» App Store Google открыл в марте 2012 г., объединив каталог приложений Android Market (работал с 2008 г.), музыкальный магазин Google Music, магазин электронных книг Google eBookstore, сервисы видеопроката через интернет и облачного хранения и синхронизации данных.



Самая востребованная категория программ в App Store и Google Play – игры (40% загрузок и 75% дохода). Ещё 15% поступают в основном от музыкальных и социальных приложений.

По данным на апрель 2013 г., App Store продолжал оставаться более доходной площадкой, замыкая на себе около 74% мировой выручки от торговли мобильными приложениями. Однако Google Play сумел обойти старожилу по количеству скачиваний – в нем совершается 51% всех загрузок, не в последнюю очередь благодаря тому, что здесь предлагается больше бесплатных и условно бесплатных программ.

До последнего времени «эппловская» платформа iOS была для разработчиков приложений предпочтительной – она наиболее популярна, ее пользователи, замкнутые на свою систему, готовы платить за контент, поэтому App Store, и удерживает лидерство по доходам.

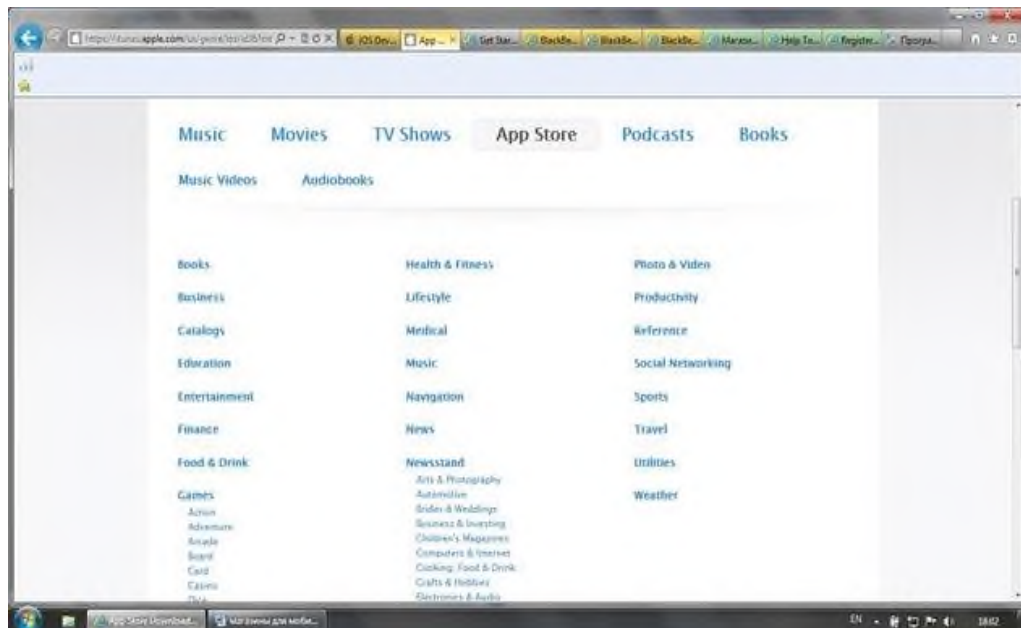
Однако сейчас рынок все больше наполняется устройствами на платформе Android. По итогам второго квартала 2012 г. 71,2% проданных планшетов базировались на iOS, а уже во втором квартале 2013 г. эта цифра упала до 42,7%, тогда как планшеты на базе Android достигли уровня 53%. Растет и количество покупок Android-смартфонов, особенно в развивающихся странах.

Таким образом, разработчикам мобильных приложений, выбирающим между App Store и Google Play, приходится взвешивать преимущества, с одной стороны, крупной выручки, с другой – темпов роста и количества продаж. Впрочем, это не единственные нюансы.

Выделяемся, как можем

Сейчас в магазине App Store насчитывается более 900 тыс. приложений, объединенных в разделы «Бизнес», «Образование», «Развлечения», «Финансы», «Игры», «Здоровье и фитнес», «Музыка», «Новости», «Фото и видео», «Социальные сети», «Путешествия», «Утилиты» и многое другое. Однако, по подсчетам специалистов, более

половины из них не могут похвастать особым количеством скачиваний. Как отмечают эксперты, данная проблема характерна и для Google Play, где размещено более 700 тыс. приложений (по состоянию на конец 2012 г.), также разбитых на категории. Причем здесь вопрос невостребованности большого процента программ возник даже раньше, чем на «яблочном» ресурсе.



Очевидно, такая ситуация – во многом следствие перенасыщенности ресурсов однотипными программами. Впрочем, самобытный и действительно полезный продукт обязательно найдет своего покупателя, ведь магазины предоставляют ряд возможностей для продвижения. Так, например, на Google Play в рамках каждой достаточно узко определенной категории приложения ранжируются в зависимости от таких показателей, как просмотры, загрузки, рейтинг по отзывам пользователей, выбор редактора ресурса и проч. Есть также топ-чарты, коллекции и очень детализированный поиск, а разработчики при загрузке приложения могут его географически таргетировать.



Программисту на заметку

Организационно-технические аспекты работы программистов с App Store и Google Play во многом подобны. Для начала необходимо присоединиться к сообществу разработчиков, причем **Apple** запросит за это 99 долл./год, а **Google** – 25 долл. разово. Затем вы сможете приобрести (или, у Google, – получить бесплатно) программный инструментарий (SDK), при помощи которого будете разрабатывать мобильные игры и приложения. Для App Store это программа XCode, доступная лишь для компьютеров с операционной системой Mac OS X и языков программирования Objective-C, C, C++ или javascript. Пакет для разработки приложений от Google Play можно установить в любой «операционке» – Mac, Windows или Linux. Язык программирования – преимущественно Java, однако без ограничений используется и C/C++.

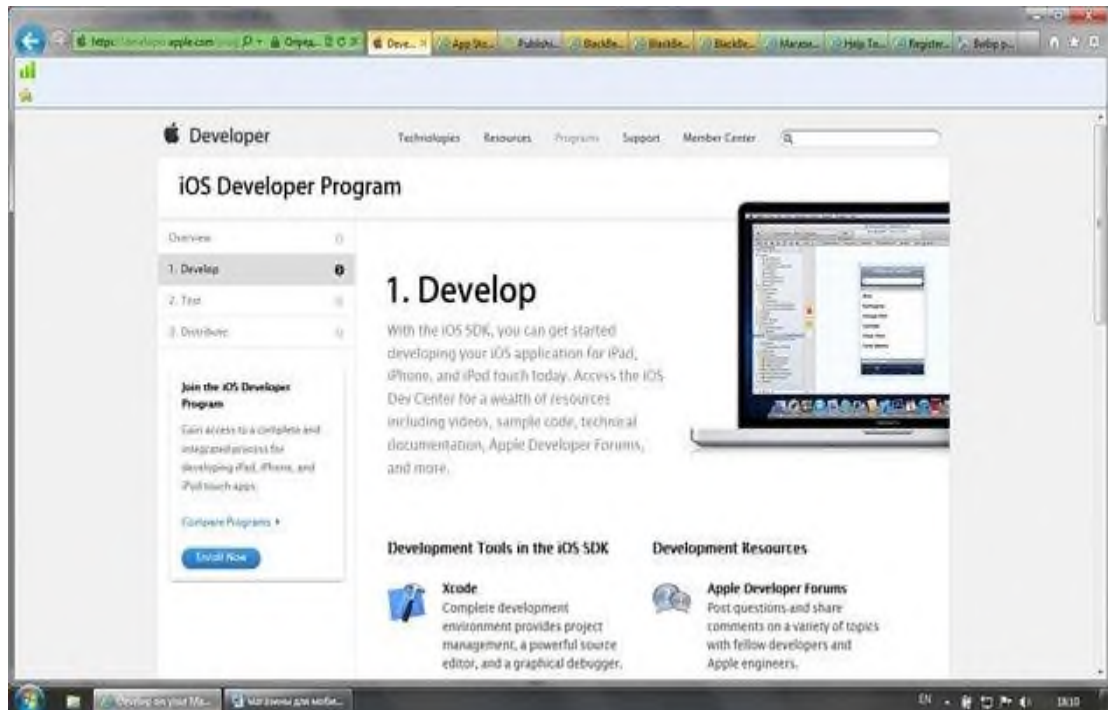
Количество бесплатных приложений в Google Play превышает 70%, тогда как в App Store этот показатель не дотягивает и до половины. Кроме меньшей финансовой выгоды, есть еще один неприятный момент в работе с Google Play. Производители электронных устройств часто не успевают вовремя адаптировать новые платформы Android под свои новинки. Поэтому разработанные вами под новые версии операционной системы приложения будут доступны пользователям с опозданием – когда они смогут установить эти самые адаптированные ОС.



Впрочем, есть у Google Play и ряд технических преимуществ. Так, например, для синхронизации приложений на нескольких устройствах Apple предлагает сервис синхронизации iCloud, тогда как все приложения, установленные вами через Google Play, сразу же привязаны к единой учетной записи Google, и стоит вам лишь подключиться к интернету со своим планшетом или смартфоном, как на нем автоматически устанавливается загруженная программа. Кроме того, к Google Play можно подключиться с любого устройства с выходом в интернет, тогда как с App Store можно взаимодействовать лишь через медиа-плеер iTunes, установленный на Mac или, в крайнем случае, Windows.

Google Play и App Store сертифицируют программные продукты и предоставляют разработчикам подробнейшие инструкции по их созданию и размещению. Тем не менее,

стоит помнить о ряде ограничений, с которыми может столкнуться программист, работая с «яблочным» ресурсом. Так, например, на нем не допускается размещение бета-версий (в отличие от Google Play), использование сторонних (кроме схемы Apple) форматов оплаты, превышение лимита времени загрузки приложения, упоминание в приложении других платформ и товарных знаков, неправильное использование файлов или данных (ограничения по хранению типов данных или подключению к приложению через, например, соцсети), даже неоднотипность используемых в приложении иконок.



Лабораторная работа №4

Принципы разработки мобильных приложений, изучения требований и формирования технической задачи, выбора платформы.

Цель работы: Ознакомление с разработками мобильных приложений и задач, выбор платформы.

Год от года потребность в мобильных приложениях возрастает, это легко объясняется - сейчас каждый в несколько кликов заказывает билеты в кино, продукты, одежду. Если вы раздумываете над [созданием мобильного приложения](#), то решайтесь — это удобно и вам, и вашим клиентам, это быстро, а самое актуальное — это идеальный инструмент для бесконтактных продаж.

В работе описаны главные принципы разработки мобильного приложения и обязательные шаги, которые проходят при создании приложений — от идеи до публикации.

Этапы создания мобильного приложения

1. Идея
2. Техническое задание
3. Бюджет
4. Организация команды
5. Разработка
6. Тестирование
7. Публикация
8. Доработка и поддержка

Идея

Для начала определите цели вашего бизнеса, изучите ваших клиентов и как вы с ними коммуницируете, изучите конкурентов — это нужно, чтобы правильно себя позиционировать. Изучение может включать в себя интервью с руководителями и клиентами, фокус-группы и экспертную оценку.

С такой подготовкой вы сможете собрать все требования и составить из них простые для понимания модели:

Такая подготовка поможет собрать все требования и упаковать их в понятные визуальные модели: схемы бизнес-процессов, диаграммы связей, движения пользователей, чтобы определить базу для разработки и переходить к прототипу.

Техническое задание

Неправильное описание может загубить даже самую перспективную идею, потому ему стоит уделить особое внимание.

Что должно содержать ТЗ:

- Цель вашего проекта.
- Пользовательские сценарии и карта действий пользователя — представления, какие задачи решает ваш сервис, и как люди будут это выполнять при его использовании.
- Функционал, который необходим и обязателен.
- Технические требования к интерфейсу, производительности, пользовательским ролям, безопасности.
- Реализация функциональности: UX и UI дизайн.
- Выделение этапов разработки.
- Время, которое вы закладываете на разработку.

Бюджет

Чем подробнее вы опишите требование к интерфейсу, тем легче дизайнеру и разработчику вас понять и сделать все ровно так, как вы задумали. Подробное ТЗ - залог получения качественного задуманного проекта с минимальным числом правок.

Организация команды

Для реализации любых проектов выделяют команду разработки, состоящую из специалистов с высшим профильным образованием, все разработчики находятся в офисе, что упрощает коммуникацию с клиентами. При необходимости привлекают к работе внешних специалистов, это позволяет качественно реализовать нам совершенно любой проект.

Разработка

Главная и, пожалуй, самая трудозатратная часть реализации. Разработка включает в себя создание архитектуры и написание кода, согласно ТЗ. При разработке не используется конструктор мобильных приложений, пишется код под каждый проект и по завершении передается заказчику. Над созданием приложения работают frontend backend разработчики. Команда разработки включена в концепцию и все процессы проекта, что позволяет в вопросах создания приложения предлагать свои решения, удовлетворяющие запросам заказчика.

В ходе разработке разработчиками создается продуманный интерфейс, отвечающий стандартам отрасли и логикам платформ.

Разработчиками со стороны Back-end создаются сервер, чтобы хранить и обмениваться данными. Программисты выбирают язык написания кода, согласовывают с заказчиком, выбирают хостинг для сервера и API. После этого, выстраивается система управления БД. Наши специалисты точно выберут все параметры, чтобы приложение работало быстрее.

Тестирование

Есть компании, выделяющие отдельным этапом тестирования и проверяющие приложение один раз - только перед публикацией.

На наш взгляд, тестирование - неотъемлемая часть окончания каждого этапа разработки и проводится после готовности каждой части функционала. Правильнее заложить больше часов на обнаружение багов перед релизом, что собирать плохие отзывы после публикации в магазине приложений. Каждый функционал приложения тестируется нами максимальное число раз.

Публикация

Важно до запуска тщательно ознакомиться с актуальными правилами Google Play Store и Apple App Store и на их основании подготовить контент для публикации. После приложение будет проверено: на соответствие информации и публикацию пропускают в магазин, в таком случае приложение станет доступно для скачивания за несколько дней.

При публикации приложения впервые могут возникать сложности и вопросы.

Доработка и техподдержка

Когда приложение уже опубликовано и активно используется, вы увидите какие страницы приводят пользователя к целевым, а какие требует усовершенствования и доработки. Отнеситесь внимательно к изучению данных: они покажут какие функции лишние, а какие следует развивать.

5 лабораторная работа

Тема: Разработать дизайн мобильного приложения в соответствии с техническим заданием.

Цель работы: Ознакомление с разработкой дизайна и техническим заданием мобильных приложений.

Сейчас поговорим о том, как строится работа над созданием мобильного приложения: что включает в себя этап аналитики и что должно быть в техническом задании.

Этапы создания мобильного приложения

Мы в студии обычно строим работу так:

- аналитика;
- техническое задание;
- проектирование и дизайн;
- разработка;
- тестирование и стабилизация;
- публикация в сторах;
- поддержка и развитие.

Каждый проект — особенный. Для одного можно объединить несколько этапов в один, чтобы реализовать задуманное быстрее и дешевле. Для другого целесообразно пройти все этапы. Мы поможем выбрать оптимальный путь.

Этап 1. Аналитика

Каждое приложение начинается с идеи. Вы рассказываете нам, какие задачи должен решать будущий сервис, и мы приступаем к сбору аналитики. Глубокий срез рынка, анализ уже существующих решений, изучение конкурентов и моделей поведения покупателей... На каждом этапе анализа мы помним о конечном пользователе и продумываем жизненный цикл клиента.

Это помогает нам вместе понять, как люди будут использовать новое приложение — и сделать его максимально удобным, понятным и полезным. Такой сервис принесет пользу и вашему бизнесу.

Этап 2. Техническое задание

Мы составляем подробное описание функциональности и дизайна будущего приложения. Определяем персонажи пользователей, описываем пользовательские истории (User Story), составляем карту путешествия пользователей (Customer Journey Map) и формируем технические требования к сервису. То есть фиксируем, каким должно быть приложение, что оно должно уметь и как это будет работать.

Благодаря такому техническому заданию (ТЗ) наша команда дизайнеров и разработчиков четко понимает, какой сервис хочет получить заказчик, и поэтапно реализует первоначальную идею.

Что в результате:

- перечень функций, которые должны быть в приложении;
- требования к интерфейсу, ролям пользователя, безопасности, производительности и другие нефункциональные требования;
- описание того, как будут реализованы все эти требования;
- смета проекта.

Что такое пользовательские истории

Пользовательские истории (User Story) пошагово описывают, как пользователь ведет себя в приложении: проходит авторизацию, просматривает каталог, оформляет заказ, совершает покупку. Такая история описывает задачу пользователя, которую он решает с помощью и приложения, и его конечную выгоду. В результате мы получаем список требований, который позволяет определить функциональность будущего приложения и сделать его максимально удобным для пользователя.

Что такое карта путешествий пользователя

Карта путешествия пользователя (Customer Journey Map) позволяет наглядно представить, как разные персонажи будут пользоваться приложением в каждой из пользовательских историй. На такой карте виден весь путь пользователя — перемещение между экранами и клики на кнопки.

Составление карты помогает понять, как технически реализовать все функции приложения.

Чек-лист: что должно быть в ТЗ

У каждой студии разработки свой подход к составлению этого документа. Мы считаем, что для успешной реализации проекта в нем должно быть отражено следующее.

1. Общие сведения:

- цель создания сервиса;
- совместимость с платформами: это будет приложение для iOS, Android или других платформ;
- масштабируемость: умеет ли приложение быстро адаптироваться к внезапным изменениям и пиковым нагрузкам, например к росту числа пользователей или объема передачи данных;
- отказоустойчивость: должно ли приложение продолжить свою работу, если откажет один или несколько его компонентов.

2. Функциональные требования к приложению:

- роли пользователей: какие уровни доступа должны быть у разных пользователей, например у гостя и авторизованного пользователя;
- форматы данных: как будет реализован обмен данными в приложении;
- интеграция: должно ли приложение поддерживать совместную работу с другими сервисами, например с платежными системами и почтовыми серверами;
- интерфейсы доступа: как приложение будет обмениваться данными с внешними сервисами;
- дополнительные функции: должно ли приложение уметь что-то еще, например работать с файлами или библиотеками шифрования;
- конфигурация и администрирование: с помощью каких элементов администратор будет управлять приложением;
- состав системы: из чего состоит мобильное приложение, то есть экраны, пуш-уведомления, система аутентификации и т.д.

3. Нефункциональные требования к приложению:

- безопасность: требования к безопасности приложения;
- логирование: нужно ли системе формировать и сохранять отчеты об ошибках, которые возникли при работе приложения, и для каких типов событий это надо делать;

- производительность: требования к работе приложения, например к скорости работы базы данных;
- требования к аппаратному обеспечению сервера: перечень технических характеристик.

4. Реализация функциональности приложения:

- экран загрузки;
- регистрация и авторизация;
- основной экран;
- меню;
- поиск;
- ...
- уведомления.

6 лабораторная работа

Тема: Настройте Android Studio и запрограммируйте ее во фреймворке Flutter.

Цель работы: Ознакомление с фреймворком Flutter и настройка Android Studio

Flutter - это новый кроссплатформенный UI framework, который включает в себе набор инструментов: движок для 2D рендеринга, готовый набор UI компонентов и набор инструментов для разработки. Он позволяет писать кроссплатформенные приложения с общей кодовой базой.

Flutter состоит из трёх компонентов:

- Flutter engine.
- Foundation library.
- Design-specific widgets.

Flutter engine - написан на C++, предоставляет низкоуровневую поддержку рендеринга, использует для этого графическую библиотеку [Skia](#). Skia - это 2D графическая библиотека, с открытым исходным кодом, которая предоставляет кроссплатформенное API. Она служит графическим движком для Google Chrome, Chrome OS, Android, Mozilla Firefox, Firefox OS и т.д. На официальном сайте есть примеры, которые запускаются сразу из браузера.

Foundation library - включает в себя базовые классы и функции, которые используются для создания приложения с использованием Flutter. Данная библиотека написана уже на языке Dart.

Design-specific widgets - это набор уже готовых базовых графических компонентов для построения пользовательского интерфейса. Данный набор состоит из двух крупных наборов компонентов:

- Material Design widgets - графические компоненты реализующие спецификацию material design.
- Cupertino widgets - набор реализующий Apple iOS дизайн.

Виджет представляет собой строительный блок для построения пользовательского интерфейса, который по сути является согласованной унифицированной моделью.

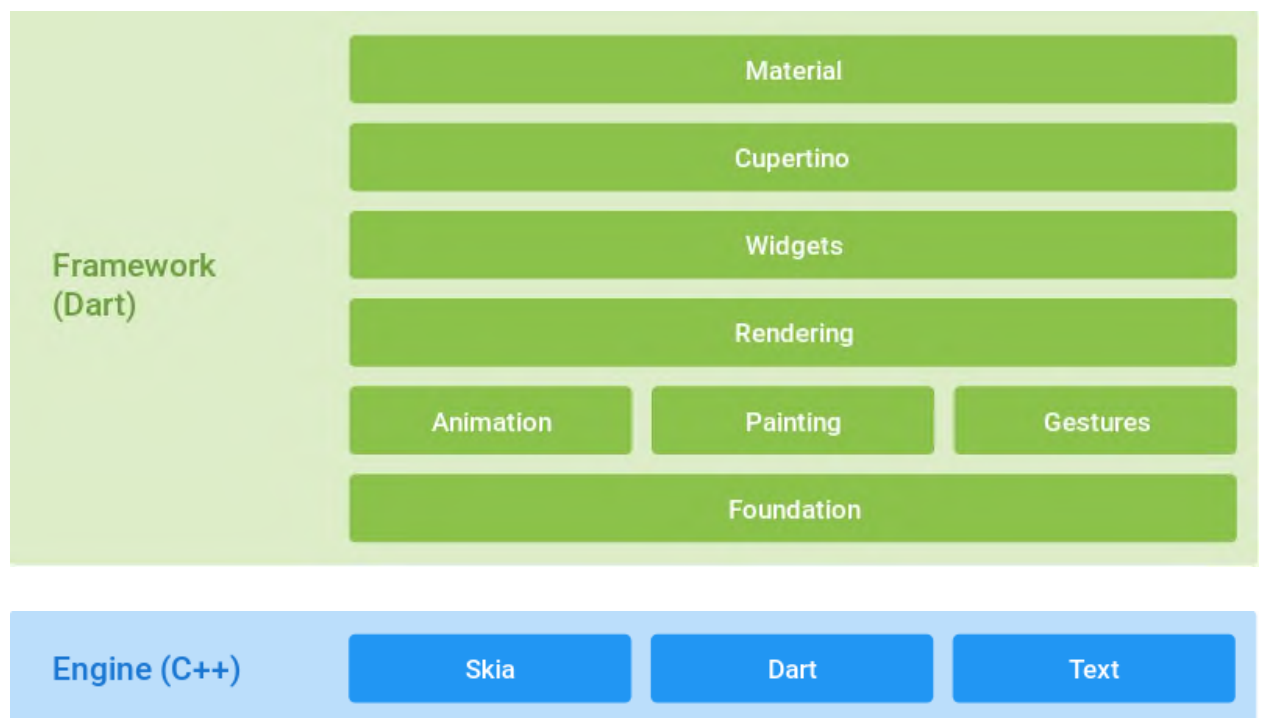
С помощью виджета можно определить:

- структурные элементы, например кнопки или меню
- стилистические элементы, например шрифты или цветовую схему
- разметку макета, например отступы.

Виджеты выстраиваются в иерархию с помощью композиции. Каждый виджет находится внутри другого виджета и наследует его свойства. Характерной особенностью является то что корневой виджет выполняет роль application.

Так же виджет позволяет получать и обрабатывать события и заменять виджеты в иерархии динамически.

Иерархия слоёв



Установка Flutter

Что бы попробовать Flutter в деле необходимо клонировать репозиторий с GitHub из ветки beta с помощью команды:

```
1 $ git clone -b beta https://github.com/flutter/flutter.git
```

Далее нужно добавить переменные окружения PATH. Для этого в конец файла ~/.bashrc нужно добавить следующие строки:

```
1 export PATH=<путь к каталогу>/flutter/bin:$PATH
```



```
2 export ANDROID_HOME=/<путь к каталогу>/android-sdk-linux
3 export PATH=${PATH}:${ANDROID_HOME}/tools:${ANDROID_HOME}/platform-tools
```

Где в первой строке необходимо указать путь к каталогу с исполняемым файлом **flutter**, который находится в каталоге **/bin**. Во второй строке нужно добавить путь к Android SDK.

Следующая команда проверит и при необходимости закачает все зависимости, которые нужны для Flutter:

```
1 $ flutter doctor
```

После того как все необходимое будут закачено, в терминале будет выведена следующая информация:

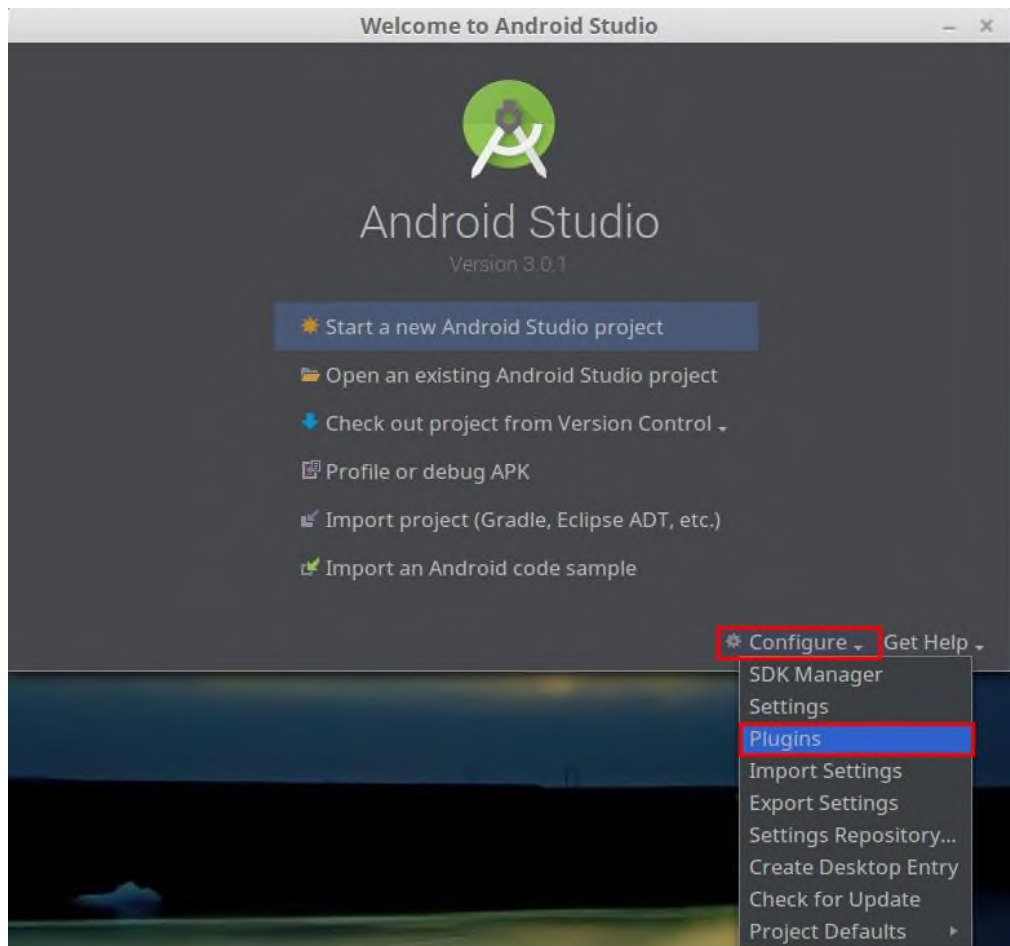
```
1 $ flutter doctor
2 Doctor summary (to see all details, run flutter doctor -v):
3 [✓] Flutter (Channel beta, v0.1.5, on Linux, locale en_US.UTF-8)
4 [✓] Android toolchain - develop for Android devices (Android SDK 27.0.3)
5 [✓] Android Studio (version 3.0)
6 [✓] Connected devices (1 available)
7 • No issues found!
```

С помощью команды **flutter devices** можно увидеть список подключённых устройств. В моём случае это эмулятор:

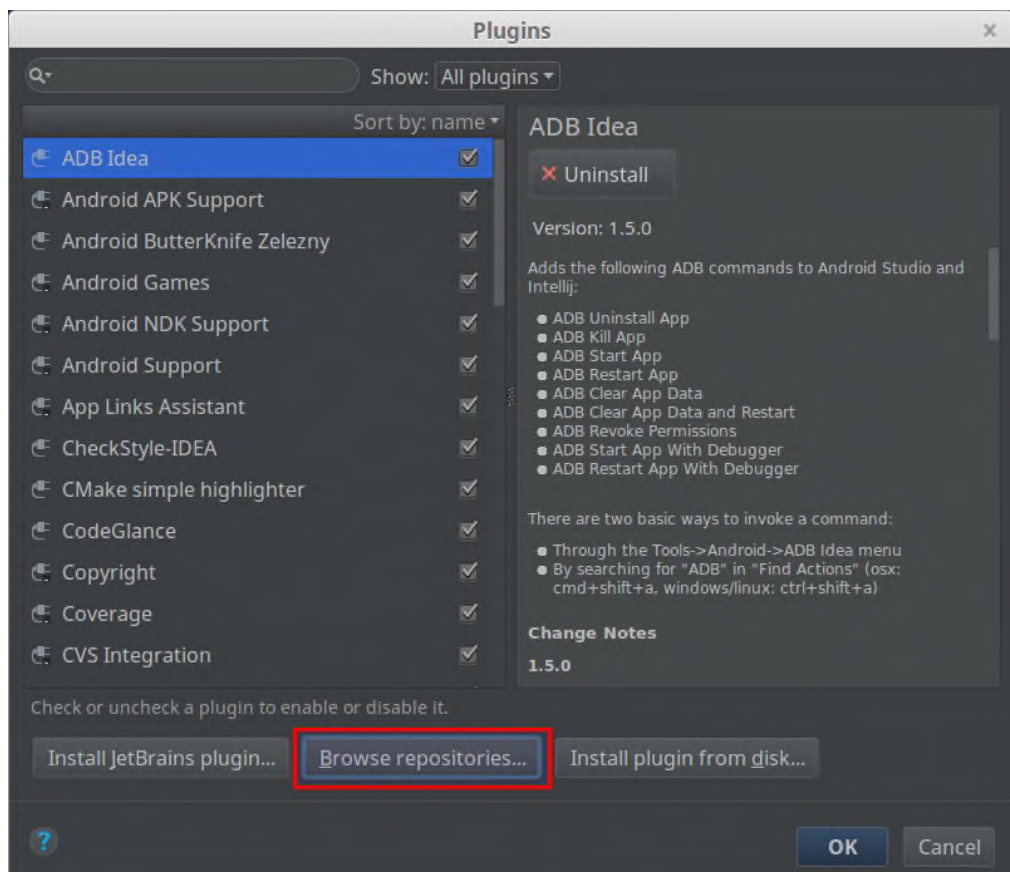
```
1 $ flutter devices
2 1 connected device:
3 Android SDK built for x86 • emulator-5554 • android-x86 • Android 8.1.0 (API 27) (emulator)
```

Установка плагинов в Android studio (Flutter и Dart)

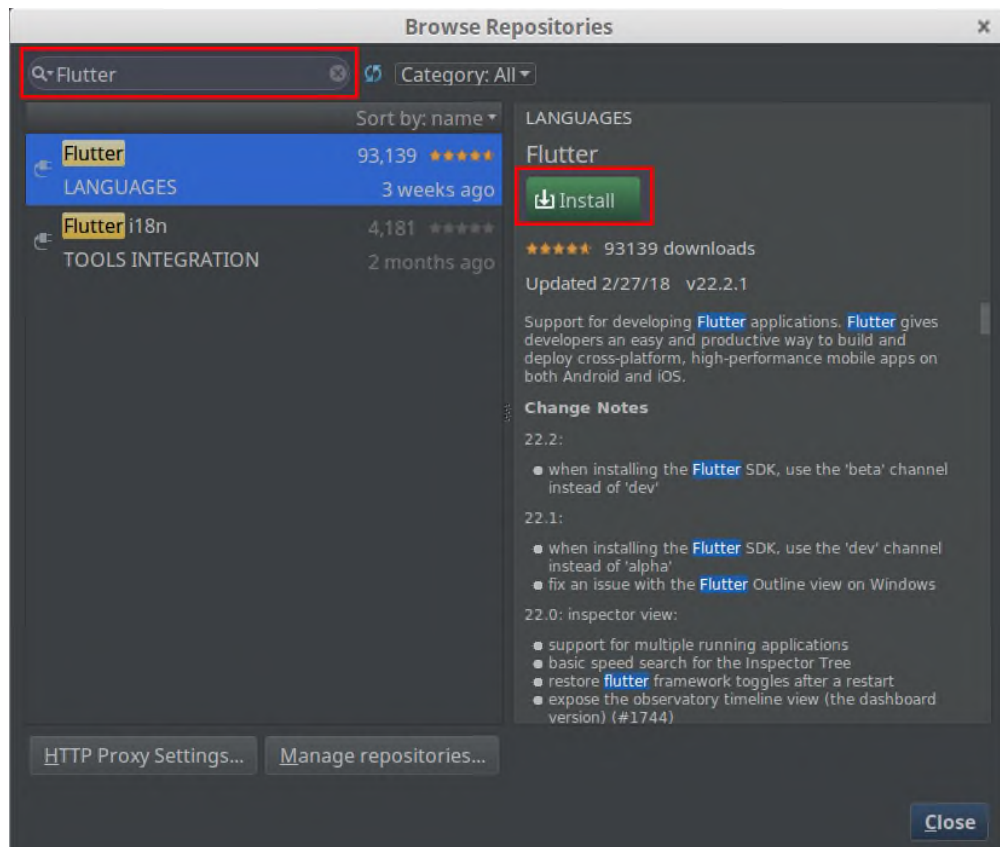
Для поддержки Flutter в Android Studio необходимо установить плагин **Flutter** из репозитория. Для этого на экране приветствия нужно выбрать пункт меню **Configure | Plugins**.



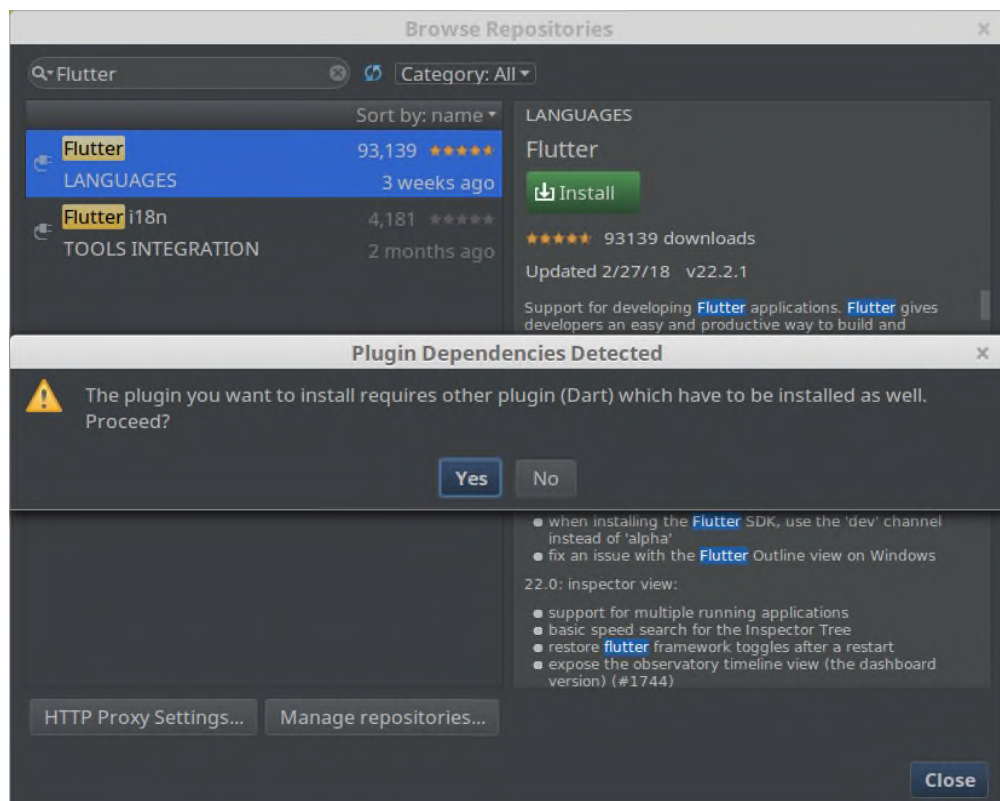
В окне **Plugins** нужно выбрать кнопку **Browse repositories**.



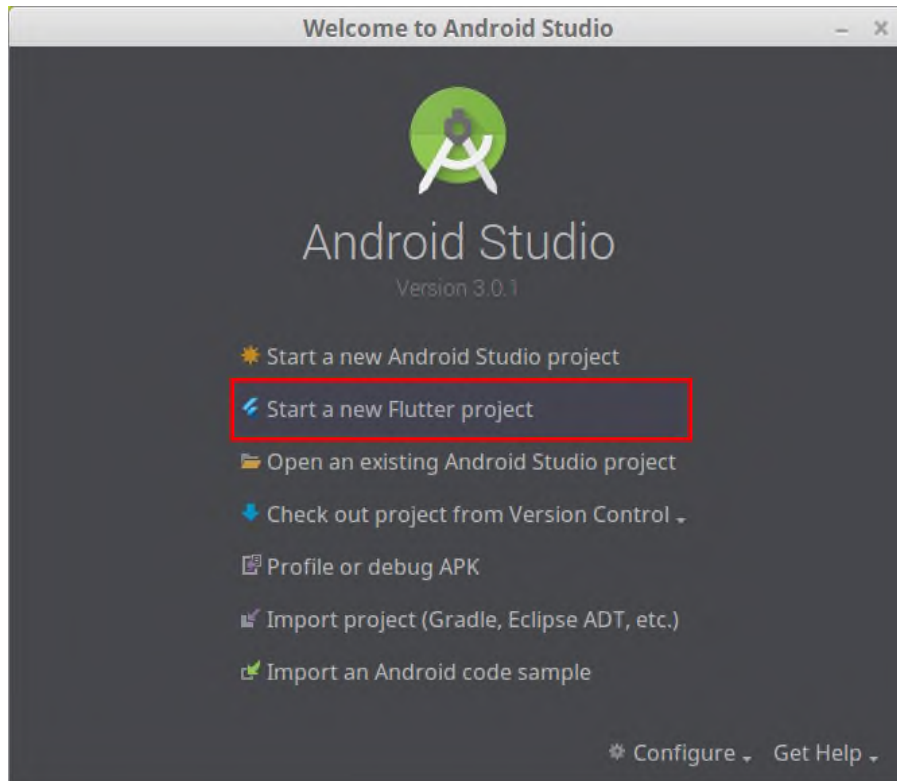
В строке поиска вбиваем название плагина **Flutter** и выбираем кнопку **Install**.



Если ранее не был установлен плагин **Dart**, то появиться диалоговое окно, где предложат это сделать. Соглашаемся с установкой.

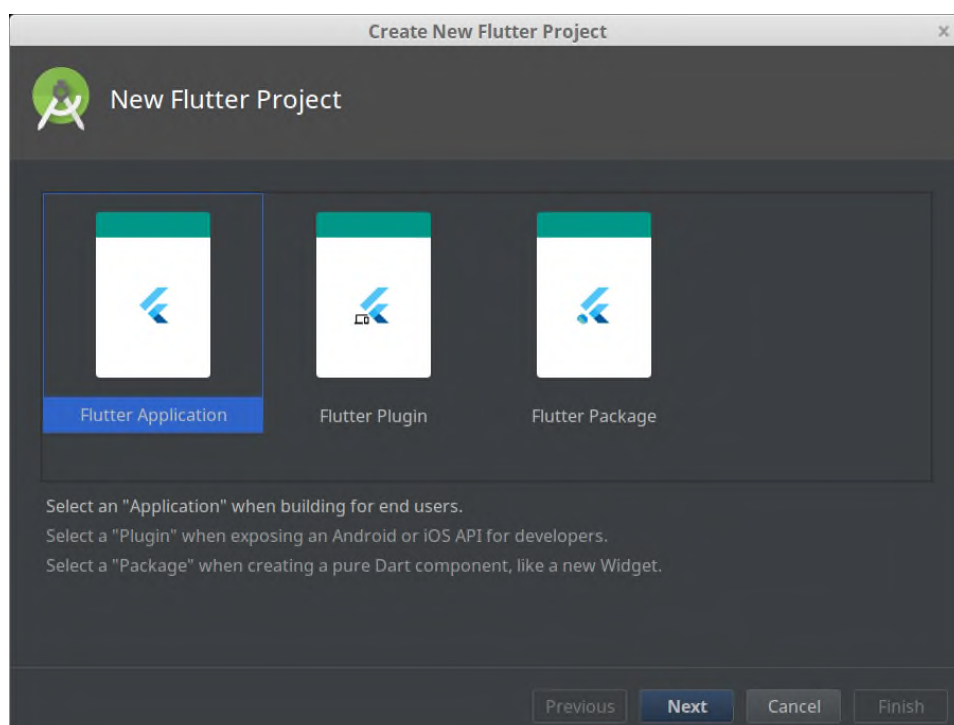


После установки плагина необходимо перезапустить Android Studio. После этого в основном меню появится новый пункт **Start a new Flutter project**.

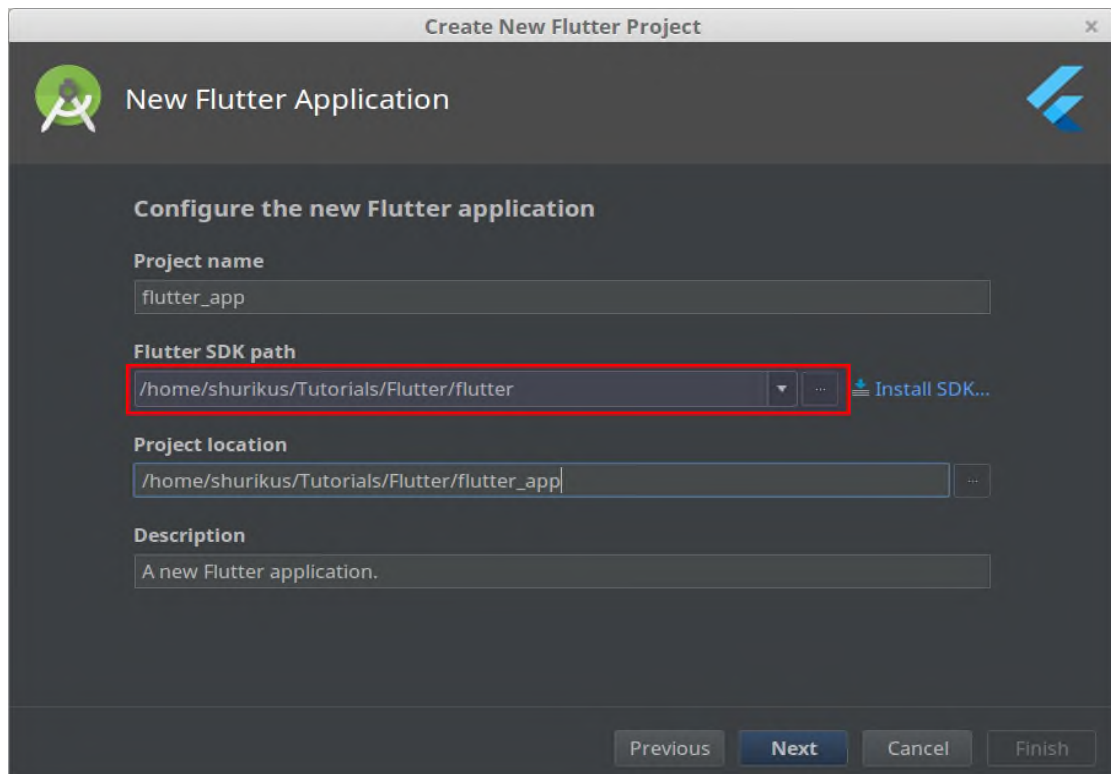


Создание нового проекта

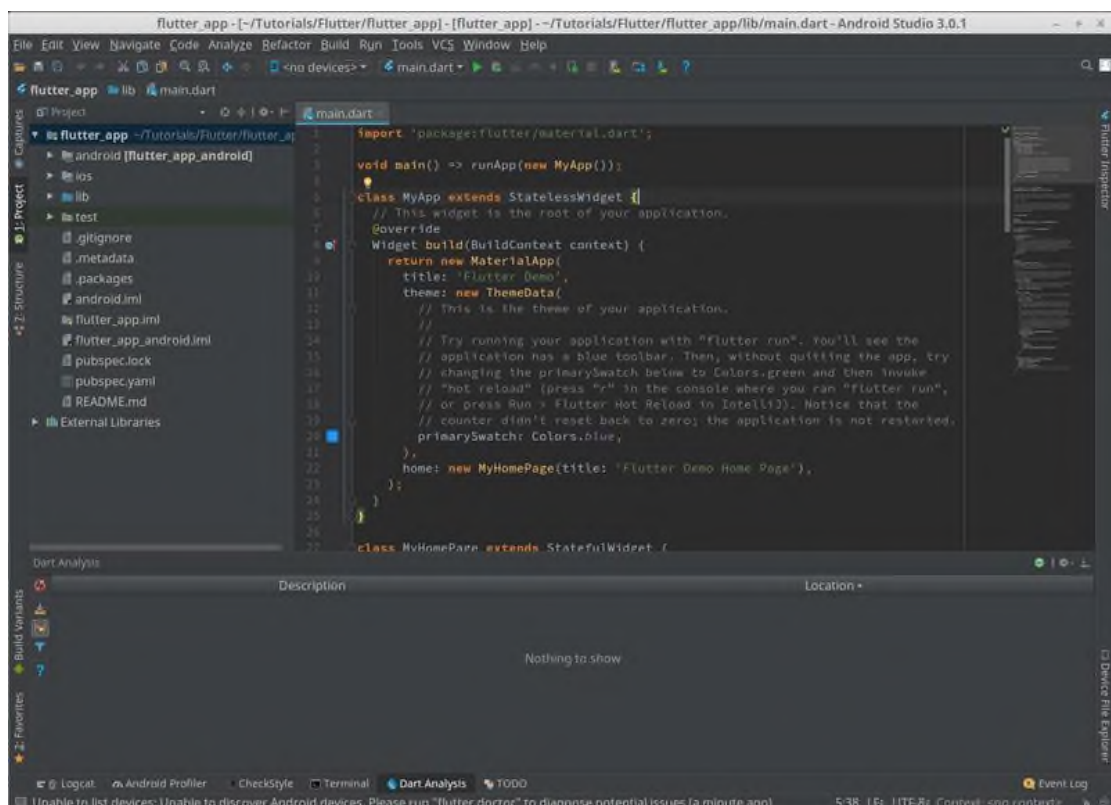
После установки плагинов в основном меню Android Studio выбираем пункт **Start a new Flutter project**. После выбора пункта запусится мастер создания проекта:



Выбираем пункт **Flutter Application**. Далее нажимаем на кнопку **Next** и переходим к следующему шагу:



Здесь необходимо указать путь к **Flutter SDK**, который был закачен ранее. Нажимаем на кнопку **Next**. На следующем завершающем шаге нажимаем кнопку **Finish**. После этого сгенерируется Flutter проект.



Запустим эмулятор, и произведём запуск приложения. Нужно перейти в корневой каталог проекта и выполнить команду:

```
1 $ flutter run
```

После компиляции приложение запустится на устройстве:



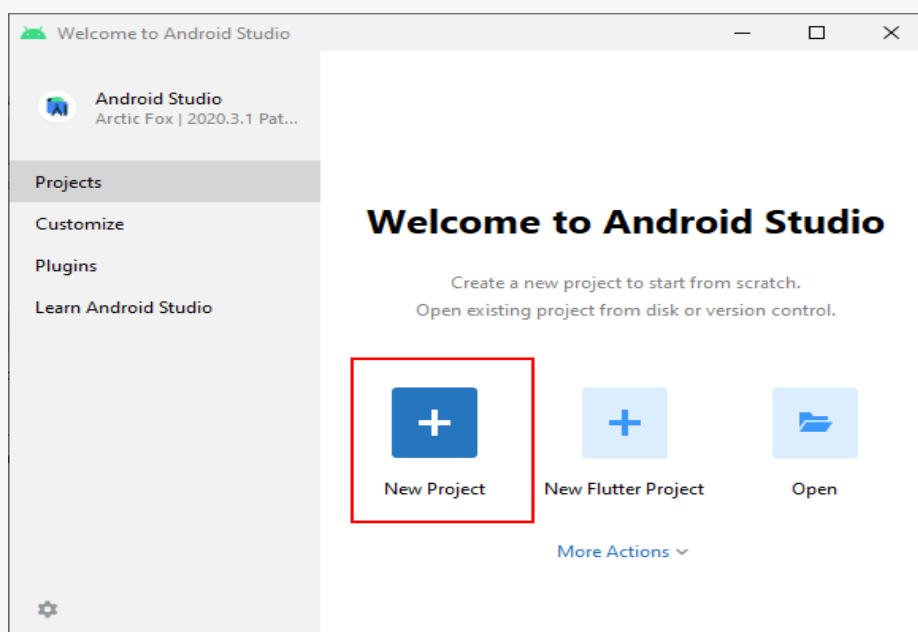
В следующей статье создадим свой собственный экран из готовых компонентов Flutter.

7 лабораторная работа

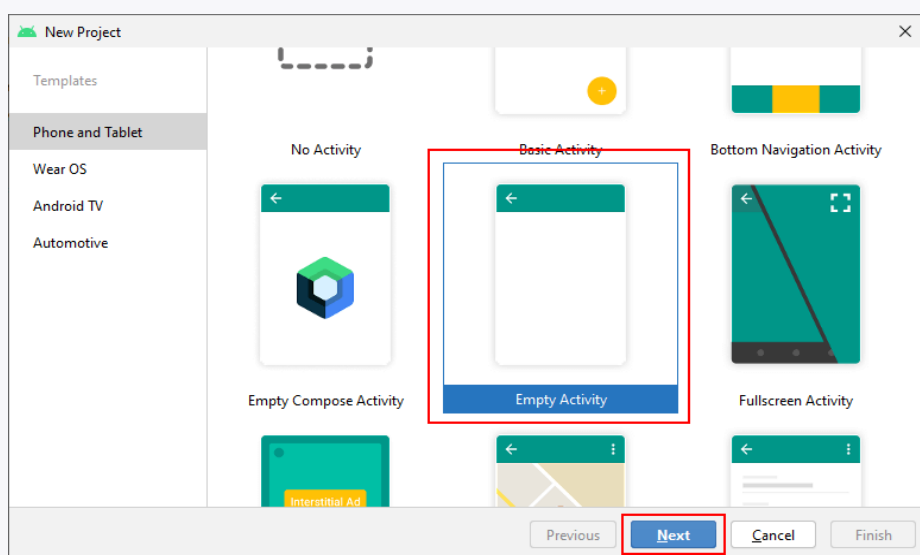
Тема: Напишите первое приложение в Android Studio и протестируйте его на эмуляторе.

Цель работы: Умение писать первое приложение на Android Studio и компиляция на AVD.

Теперь создадим первое приложение в среде Android Studio для операционной системы Android. Откроем Android Studio и на начальном экране выберем пункт **New Project**:



При создании проекта Android Studio вначале предложит нам выбрать шаблон проекта:



Android Studio предоставляет ряд шаблонов для различных ситуаций. Выберем в этом списке шаблон **Empty Activity**, который предоставляет самый простейший функционал, необходимый для начала, и нажмем на кнопку **Next**.

После этого отобразится окно настроек нового проекта:

New Project

Empty Activity

Creates a new empty activity

Name: HelloApp

Package name: com.example.helloapp

Save location: C:\Users\Eugene\AndroidStudioProjects\Java\HelloApp

Language: Java

Minimum SDK: API 21: Android 5.0 (Lollipop)

i Your app will run on approximately **94,1%** of devices.
[Help me choose](#)

☐ Use legacy android.support libraries *?*
Using legacy android.support libraries will prevent you from using the latest Play Services and Jetpack libraries

Выберем Java

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

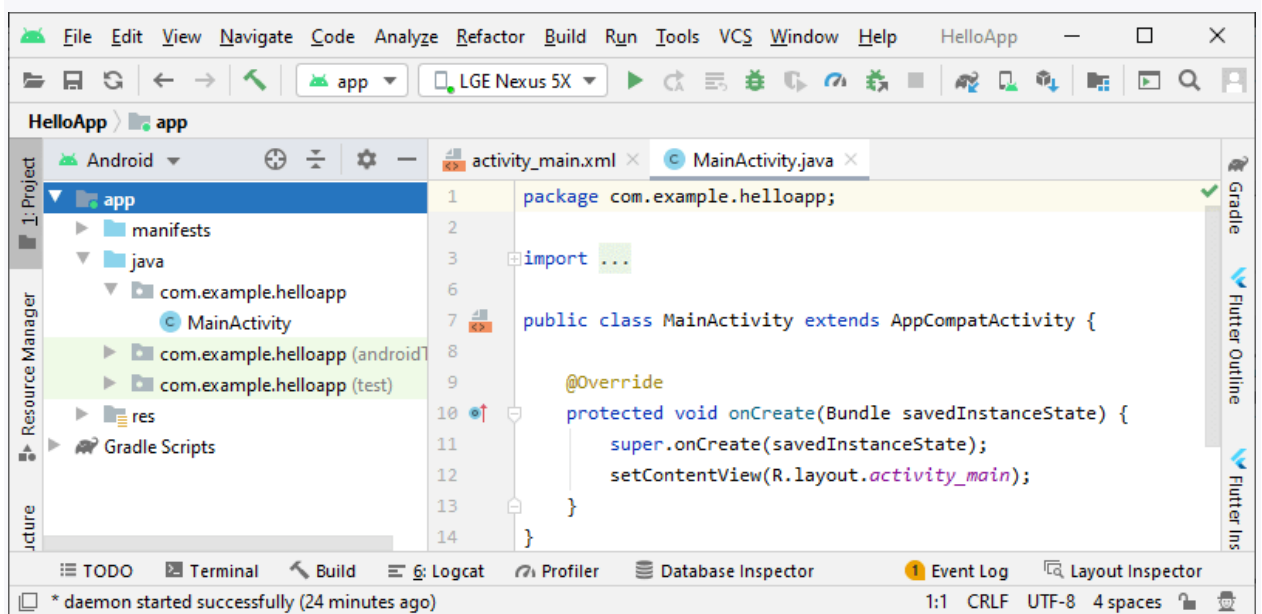
В окне создания нового проекта мы можем установить его начальные настройки:

- В поле **Name** вводится название приложения. Укажем в качестве имени название **HelloApp**
- В поле **Package Name** указывается имя пакета, где будет размещаться главный класс приложения. В данном случае для тестовых проектов это значение не играет большого значения, поэтому установим **com.example.helloapp**.
- В поле **Save Location** устанавливается расположение файлов проекта на жестком диске. Можно оставить значение по умолчанию.

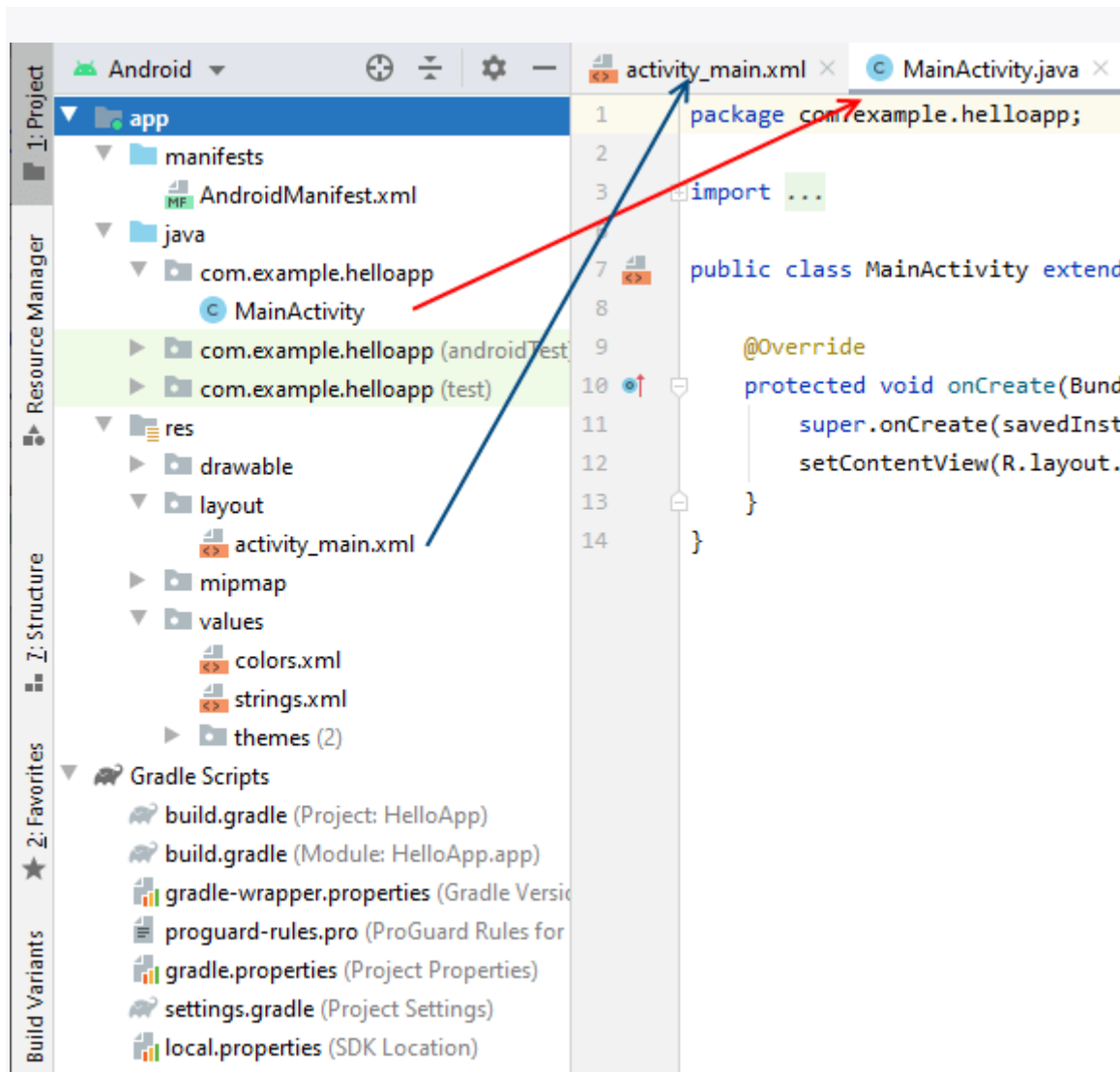
- В поле **Language** в качестве языка программирования укажем **Java** (будьте внимательны, так как по умолчанию в этом поле стоит Kotlin)
- В поле **Minimum SDK** указывается самая минимальная поддерживаемая версия SDK. Оставим значение по умолчанию - **API 21: Android 5.0 (Lollipop)**, которая означает, что наше приложение можно будет запустить начиная с Android 5.0, а это 94% устройств. На более старых устройствах запустить будет нельзя.

Стоит учитывать, что чем выше версия SDK, тем меньше диапазон поддерживаемых устройств.

Далее нажмем на кнопку Finish, и Android Studio создаст новый проект:



Вначале вкратце рассмотрим структуру проекта, что он уже имеет по умолчанию



Проект Android может состоять из различных модулей. По умолчанию, когда мы создаем проект, создается один модуль - **app**. Модуль имеет три подпапки

- **manifests**: хранит файл манифеста **AndroidManifest.xml**, который описывает конфигурацию приложения и определяет каждый из компонентов данного приложения.
- **java**: хранит файлы кода на языке java, которые структурированы по отдельным пакетам. Так, в папке **com.example.helloapp** (название которого было указано на этапе создания проекта) имеется по умолчанию файл **MainActivity.java** с кодом на языке Java, который представляет класс **MainActivity**, запускаемый по умолчанию при старте приложения
- **res**: содержит используемые в приложении ресурсы. Все ресурсы разбиты на подпапки.

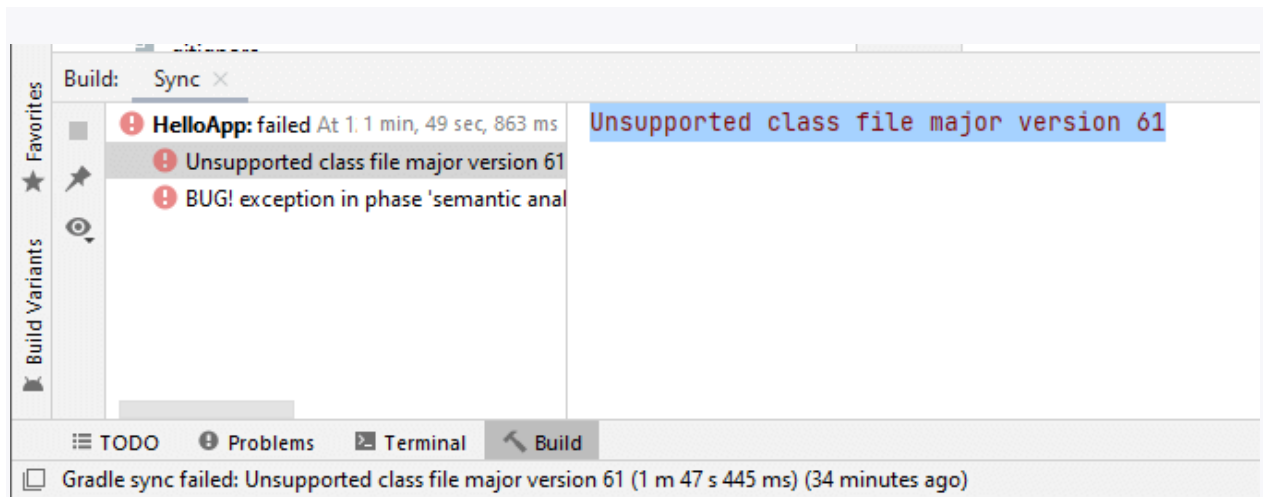
- папка **drawable** предназначена для хранения изображений, используемых в приложении
- папка **layout** предназначена для хранения файлов, определяющих графический интерфейс. По умолчанию здесь есть файл **activity_main.xml**, который определяет интерфейс для класса MainActivity в виде xml
- папки **mipmap** содержат файлы изображений, которые предназначены для создания иконки приложения при различных разрешениях экрана.
- папка **values** хранит различные xml-файлы, содержащие коллекции ресурсов - различных данных, которые применяются в приложении. По умолчанию здесь есть два файла и одна папка:
 - файл **colors.xml** хранит описание цветов, используемых в приложении
 - файл **strings.xml** содержит строковые ресурсы, используемые в приложении
 - папки **themes** хранит две темы приложения - для светлую (дневную) и темную (ночную)

Отдельный элемент **Gradle Scripts** содержит ряд скриптов, которые используются при построении приложения.

Во всей этой структуре следует выделить файл MainActivity.java, который открыт в Android Studio и который содержит логику приложения и собственно с него начинается выполнение приложения. И также выделим файл **activity_main.xml**, который определяет графический интерфейс - по сути то, что увидит пользователь на своем смартфоне после загрузки приложения.

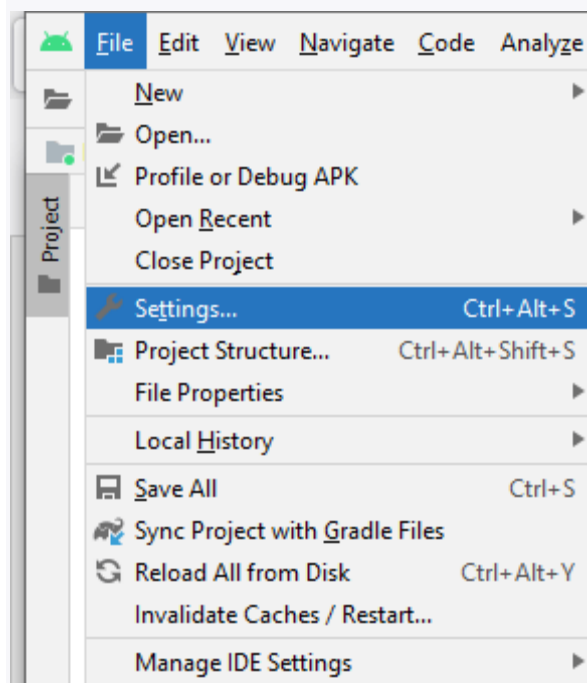
Возможные проблемы

Для создания приложения используется Java. А для построения приложения применяется инфраструктура Gradle. Однако текущая используемая версия Gradle может быть несовместима с выбранной по умолчанию версией JDK. И в этом случае Android Studio может отображать ошибки, например, ошибку **Unsupported class file major version 61**:



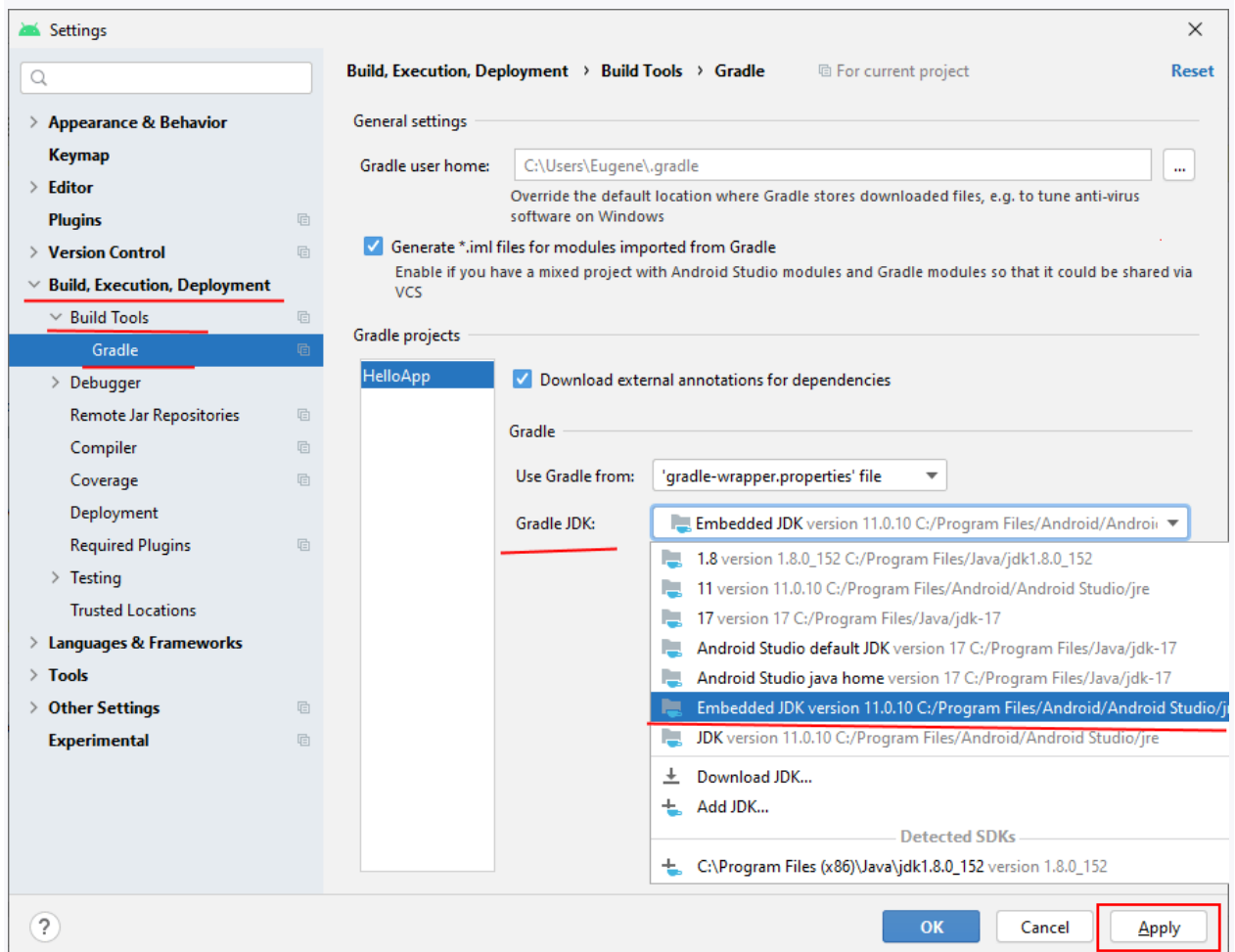
Эта ошибка говорит о том, что версия JDK 17 несовместима с текущей версией Gradle. И надо использовать меньшую версию.

Для решения этой проблемы перейдем в студии к меню **File ->Settings** (на MacOS это пункт **Android Studio -> Preferences**)



Затем в открывшемся окне настроек перейдем к пункту меню **Build, Execution, Deployment -> Build Tools -> Gradle** и далее найдем поле **Gradle**

JDK, где изменим версию JDK. Она должна иметь версию 11 и выше. Как правило, вместе с Android Studio устанавливается и поддерживаемая версия JDK - на данный момент это JDK 11. И ее можно выбрать в списке JDK:



Наиболее оптимальный пункт для выбора версий JDK, которая идет вместе с Android Studio, называется **Embedded JDK version....** Как видно на скриншоте, это версия 11, но при последующих обновлениях Android Studio эта версия может измениться.

После сделанных изменений сначала нажмем на кнопку **Apply**, а затем на кнопку **ОК**. И повторим запуск проекта.

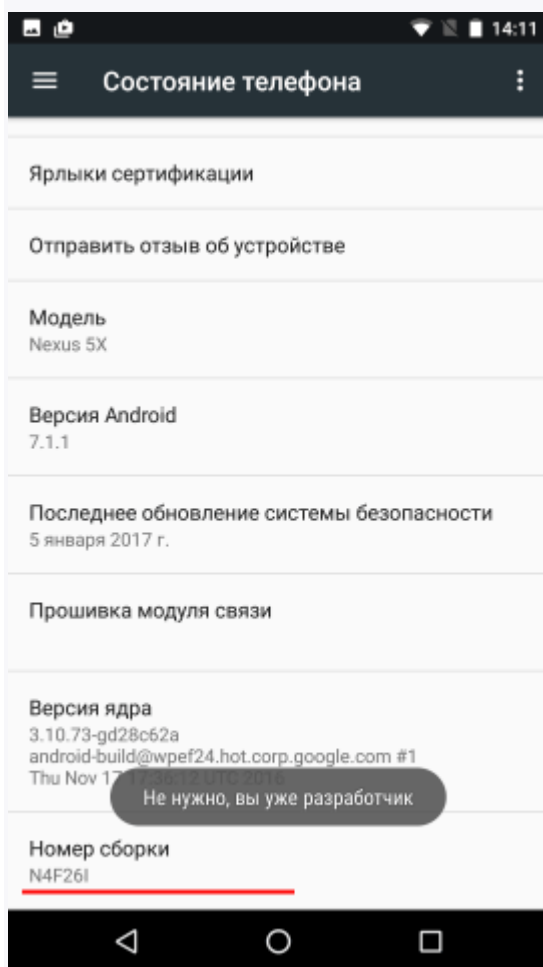
Запуск проекта

Созданный выше проект уже содержит некоторый примитивный функционал. Правда, этот функционал почти ничего не делает, только выводит на экран строку "Hello world!". Тем не менее это уже фактически приложение, которое мы можем запустить.

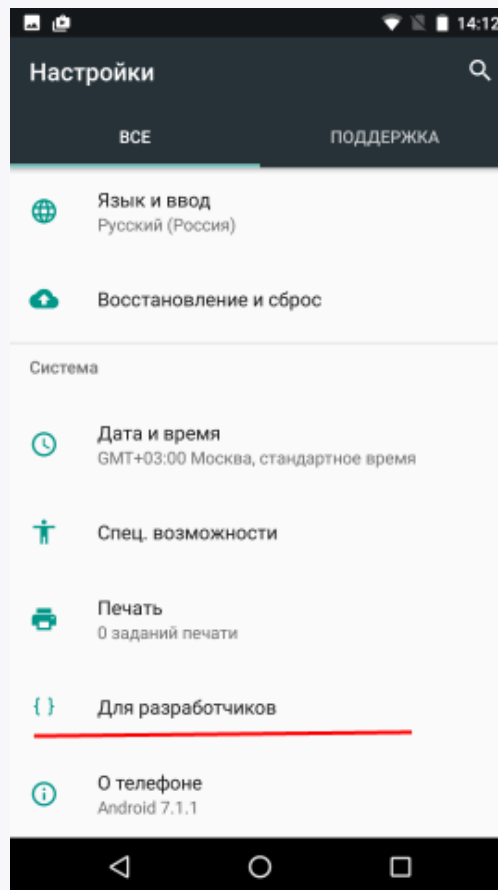
Для запуска и тестирования приложения мы можем использовать эмуляторы или реальные устройства. Но в идеале лучше тестировать на реальных устройствах. К тому же эмуляторы требуют больших аппаратных ресурсов, и не каждый компьютер может потянуть требования эмуляторов. А для использования мобильного устройства для тестирования может потребоваться разве что установить необходимый драйвер.

Режим разработчика на телефоне

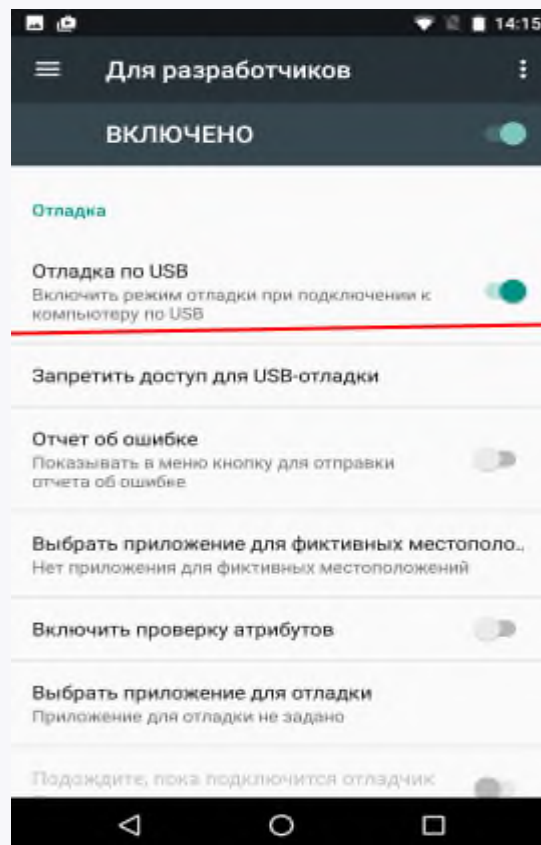
По умолчанию опции разработчика на смартфонах скрыты. Чтобы сделать их доступными, надо зайти в **Settings > About phone** (Настройки > О телефоне) (в Android 8 это в **Settings > System > About phone** (Настройки > Система > О телефоне)) и семь раз нажать **Build Number** (Номер сборки).



Теперь необходимо включить отладку по USB. Для этого перейдем в **Settings > System > Advanced > Developer options** или **Настройки > Система > Дополнительно > Для разработчиков** (в Android 8 это в **Settings > System > Developer options** или **Настройки > Система > Для разработчиков**).

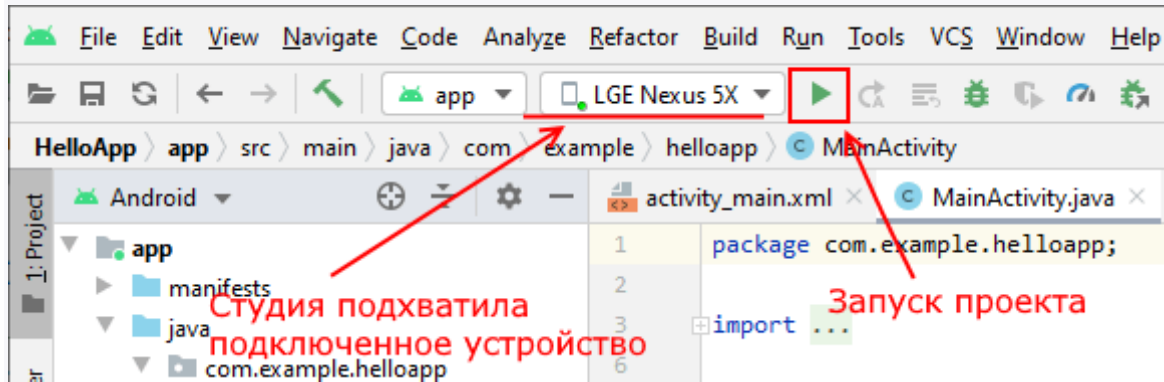


И включим возможность отладки по USB:



Запуск приложения

Подключим устройство с ОС Android (если мы тестируем на реальном устройстве) и запустим проект, нажав на зеленую стрелочку на панели инструментов.



Выберем устройство и нажмем на кнопку ОК. И после запуска мы увидим наше приложение на экране устройства:



Hello World!



8 лабораторная работа

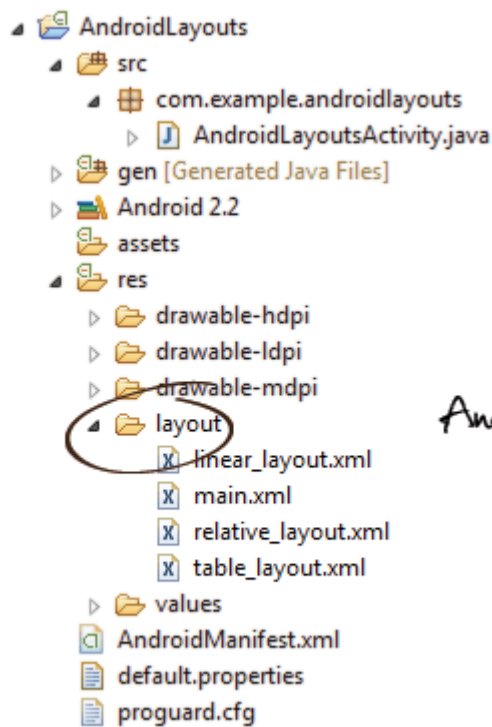
Тема: Работа с ключевыми компонентами в Android Studio: Layout, Table, ListView, Grid, List и др.

Цель работы: Ознакомление о макетах представления в мобильном приложении Android.

Шесть различных макетов:

1. Linear Layout
2. Relative Layout
3. Table Layout
4. Grid View
5. Tab Layout
6. List View

Android позволяет создавать макеты представлений с помощью простого XML-файла (мы также можем создать макет с помощью кода Java). Все макеты должны быть помещены в папку /res/layout.



Android Layout Directory

Хорошо, теперь давайте начнем с макетов представлений.

1. Линейная компоновка

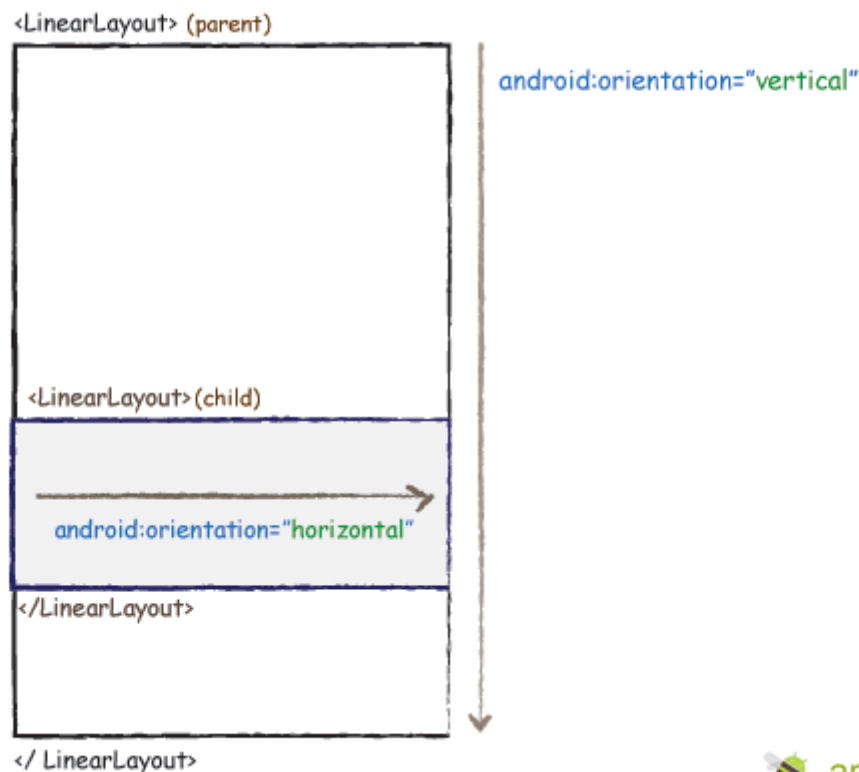
В линейном макете, как следует из названия, все элементы отображаются линейно (ниже приведен пример линейного макета), либо по горизонтали, либо по вертикали, и это поведение задается в android: ориентация, которая является атрибутом узла Линейный макет.

Пример фрагмента вертикального макета

```
<LinearLayout android:orientation="vertical"> .... </LinearLayout>
```

Example of Horizontal layout snippet

```
<LinearLayout android:orientation="horizontal"> .... </LinearLayout>
```



Теперь, когда мы знаем два типа линейных макетов, вот шаги, которые вам нужно выполнить для их создания.

1. Создайте новый проект File -> New -> Android Project
2. В проводнике пакетов щелкните правой кнопкой мыши папку res/layout, создайте новый XML-файл Android и назовите его по своему усмотрению. Я называю его «linear_layout.xml».
res/layout -> Щелкните правой кнопкой мыши -> Создать -> XML-файл Android
3. Теперь откройте только что созданный XML-файл (в моем случае «linear_layout.xml») и введите следующий код.

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Parent linear layout with vertical orientation -->
<LinearLayout
```

```

xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="match_parent"
android:layout_height="match_parent">

<TextView android:layout_width="fill_parent" android:layout_height="wrap_content"
    android:text="Email:" android:padding="5dip"/>

<EditText android:layout_width="fill_parent" android:layout_height="wrap_content"
    android:layout_marginBottom="10dip"/>

<Button android:layout_width="fill_parent" android:layout_height="wrap_content"
    android:text="Login"/>

<!-- Child linear layout with horizontal orientation -->
<LinearLayout android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal" android:background="#2a2a2a"
    android:layout_marginTop="25dip">

<TextView android:layout_width="fill_parent" android:layout_height="wrap_content"
    android:text="Home" android:padding="15dip" android:layout_weight="1"
    android:gravity="center"/>

<TextView android:layout_width="fill_parent" android:layout_height="wrap_content"
    android:text="About" android:padding="15dip" android:layout_weight="1"
    android:gravity="center"/>

</LinearLayout>

</LinearLayout>

```

4. Чтобы установить это только что созданное представление в качестве начального представления вашего приложения, откройте файл MainActivity.java. Вы увидите следующую строку внутри функции onCreate setContentView(R.layout.main). Измените R.layout.main на R.layout.yourlinearviewname. В моем случае это R.layout.linear_layout

```

package com.example.androidlayouts;
import android.app.Activity;
import android.os.Bundle;

```

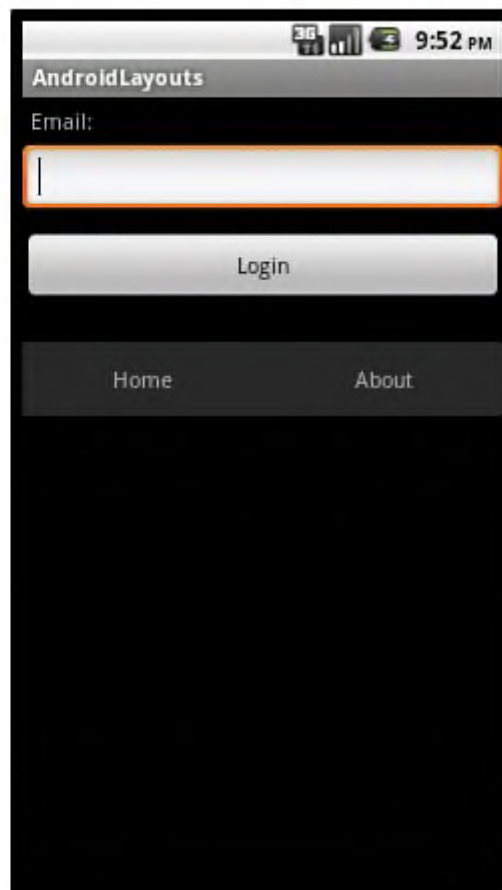
```
public class AndroidLayoutsActivity extends Activity {  
  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.linear_layout);  
    }  
}
```

155 / 5 000

Результаты перевода

5. Чтобы запустить приложение, щелкните проект правой кнопкой мыши -> Запустить от имени -> 1. Приложение Android. Вы должны увидеть только что созданный линейный макет в эмуляторе.

Linear Layout Output



2. Относительный макет

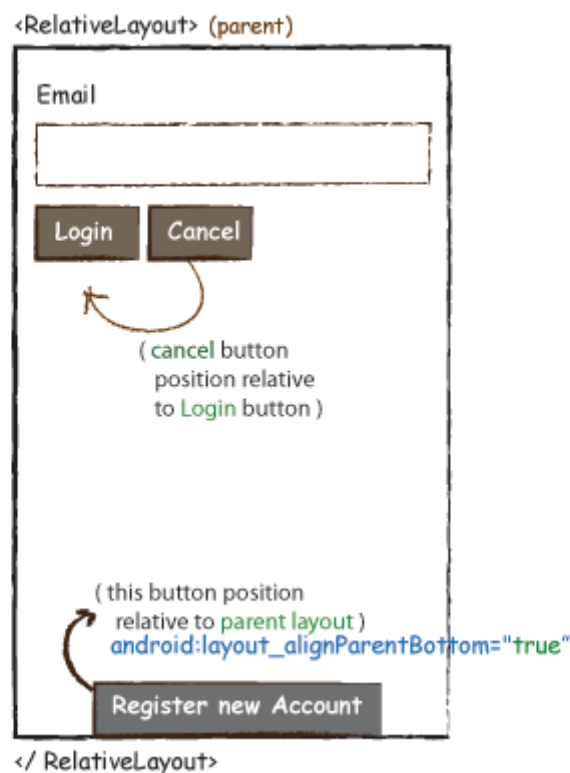
В относительной компоновке каждый элемент упорядочивает себя относительно других элементов или родительского элемента.

В качестве примера рассмотрим макет, определенный ниже. Кнопка «Отмена» расположена относительно, справа от кнопки «Войти» параллельно. Вот фрагмент кода, который обеспечивает упомянутое выравнивание (кнопка «Право на вход» параллельно)

Фрагмент кода примера

```
<Button android:id="@+id/btnLogin" ..></Button>
```

```
<Button android:layout_toRightOf="@id/btnLogin"  
    android:layout_alignTop="@id/btnLogin" ..></Button>
```



Вот шаги для создания относительного макета

1. Создайте новый проект File -> New -> Android Project
 2. В проводнике пакетов щелкните правой кнопкой мыши папку res/layout, создайте новый XML-файл Android и назовите его по своему усмотрению. Я называю его «relative_layout.xml».
- res/layout -> Щелкните правой кнопкой мыши -> Создать -> XML-файл Android

3. Теперь откройте только что созданный XML-файл (в моем случае «relative_layout.xml») и введите следующий код.

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

    <TextView android:id="@+id/label" android:layout_width="fill_parent"
        android:layout_height="wrap_content" android:text="Email" />

    <EditText android:id="@+id/inputEmail" android:layout_width="fill_parent"
        android:layout_height="wrap_content" android:layout_below="@id/label" />

    <Button android:id="@+id/btnLogin" android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:layout_below="@id/inputEmail"
        android:layout_alignParentLeft="true" android:layout_marginRight="10px"
        android:text="Login" />

    <Button android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:layout_toRightOf="@id/btnLogin"
        android:layout_alignTop="@id/btnLogin" android:text="Cancel" />

    <Button android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:layout_alignParentBottom="true" android:text="Register new Account"
        android:layout_centerHorizontal="true"/>
</RelativeLayout>
```

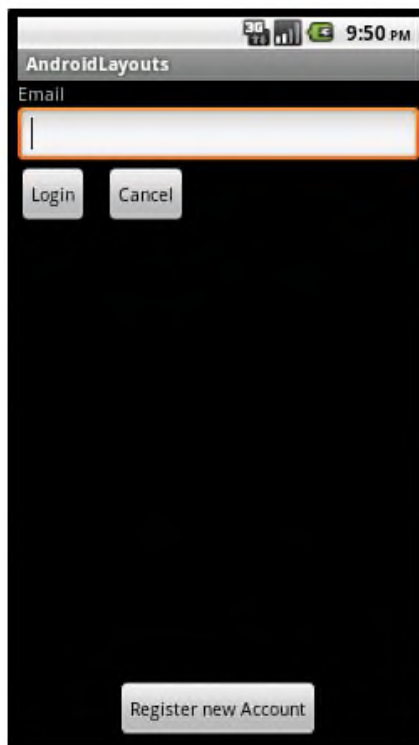
4. Так же, как и раньше, откройте файл MainActivity.java и установите макет в свой только что созданный относительный файл макета. В моем случае это R.layout.relative_layout

```
setContentView(R.layout.relative_layout);
```

5. Чтобы запустить приложение, щелкните проект правой кнопкой мыши -> Запустить от имени -> 1. Приложение Android. Вы должны увидеть только

что созданный относительный макет в эмуляторе.

Relative Layout Output



3. Макет таблицы

Макеты таблиц в Android работают так же, как макеты таблиц HTML. Вы можете разделить макеты на строки и столбцы. Его очень легко понять. Изображение ниже должно дать вам представление.

<TableLayout>

Row 1		
Row 2 column 1	Row 2 column 2	Row 2 column 3
Row 3 column 1		Row 3 column 2

</ TableLayout>



1. Создайте новый проект **File -> New -> Android Project**
2. В проводнике пакетов щелкните правой кнопкой мыши папку **res/layout**, создайте новый XML-файл **Android** и назовите его по своему усмотрению. Я называю его «**table_layout.xml**».
res/layout -> Щелкните правой кнопкой мыши -> Создать -> XML-файл Android
3. Теперь откройте только что созданный XML-файл (в моем случае «**table_layout.xml**») и введите следующий код.

```
<TableLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:shrinkColumns="*" android:stretchColumns="*" android:background="#ffffff"
    <!-- Row 1 with single column -->
    <TableRow
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:gravity="center_horizontal">
        <TextView
            android:layout_width="match_parent" android:layout_height="wrap_content"
            android:textSize="18dp" android:text="Row 1" android:layout_span="3"
            android:padding="18dip" android:background="#b0b0b0"
            android:textColor="#000"/>
        </TableRow>

    <!-- Row 2 with 3 columns -->
    <TableRow
        android:id="@+id/tableRow1"
        android:layout_height="wrap_content"
        android:layout_width="match_parent">
        <TextView
            android:id="@+id/TextView04" android:text="Row 2 column 1"
            android:layout_weight="1" android:background="#dcdcdc"
            android:textColor="#000000"
            android:padding="20dip" android:gravity="center"/>
        <TextView
            android:id="@+id/TextView04" android:text="Row 2 column 2"
            android:layout_weight="1" android:background="#d3d3d3"
            android:textColor="#000000"
```

```

        android:padding="20dip" android:gravity="center"/>
    <TextView
        android:id="@+id/TextView04" android:text="Row 2 column 3"
        android:layout_weight="1" android:background="#cac9c9"
        android:textColor="#000000"
        android:padding="20dip" android:gravity="center"/>
</TableRow>

<!-- Row 3 with 2 columns -->
<TableRow
    android:layout_height="wrap_content"
    android:layout_width="fill_parent"
    android:gravity="center_horizontal">
    <TextView
        android:id="@+id/TextView04" android:text="Row 3 column 1"
        android:layout_weight="1" android:background="#b0b0b0"
        android:textColor="#000000"
        android:padding="20dip" android:gravity="center"/>

    <TextView
        android:id="@+id/TextView04" android:text="Row 3 column 2"
        android:layout_weight="1" android:background="#a09f9f"
        android:textColor="#000000"
        android:padding="20dip" android:gravity="center"/>
</TableRow>

</TableLayout>

```

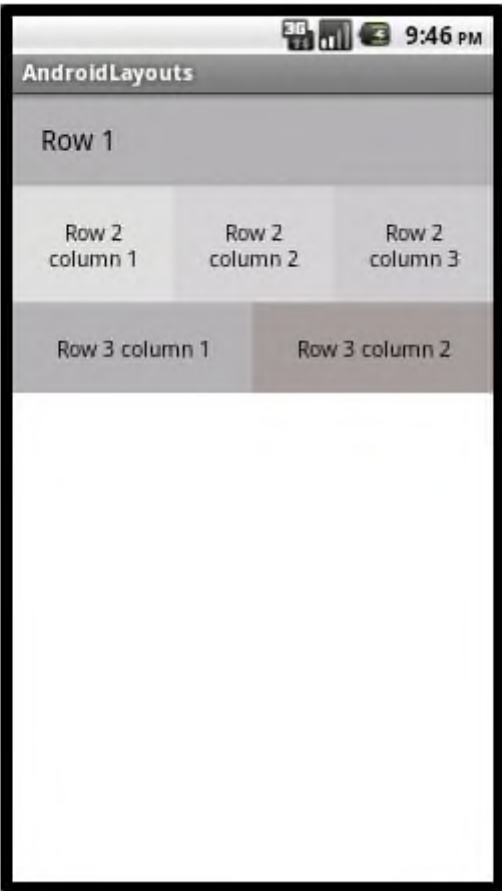
4. Так же, как и раньше, откройте файл MainActivity.java и установите макет для только что созданного файла макета таблицы. В моем случае это R.layout.table_layout

```
setContentView(R.layout.table_layout);
```

5. Чтобы запустить приложение, щелкните проект правой кнопкой мыши -> Запустить от имени -> 1. Приложение Android. Вы должны увидеть только что

созданный макет таблицы в эмуляторе.

Table Layout Output



Row 1		
Row 2 column 1	Row 2 column 2	Row 2 column 3
Row 3 column 1	Row 3 column 2	

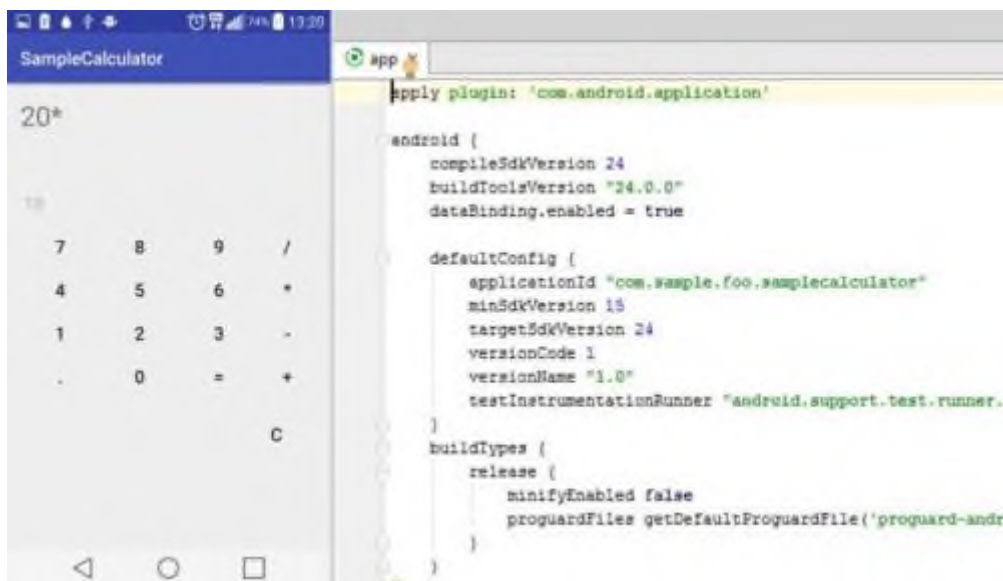


9 лабораторная работа

Тема: Создайте приложение-калькулятор и установите его на свой мобильный телефон.

Цель работы: Создание приложение- калькулятор и его компиляция.

В этом руководстве мы расскажем, как создать калькулятор на Java для Android. Если вы новичок в программировании и никогда раньше не создавали приложения, ознакомьтесь с нашим предыдущим [руководством по написанию первого приложения для Android](#):

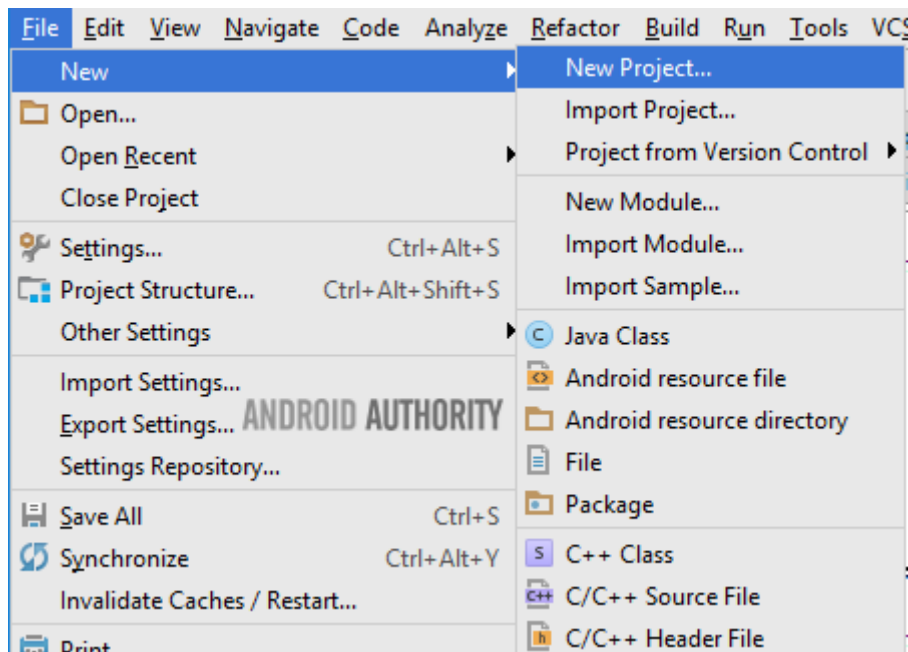


Предполагается, что у вас есть хотя бы минимальный базовый опыт создания **Android – приложений**.

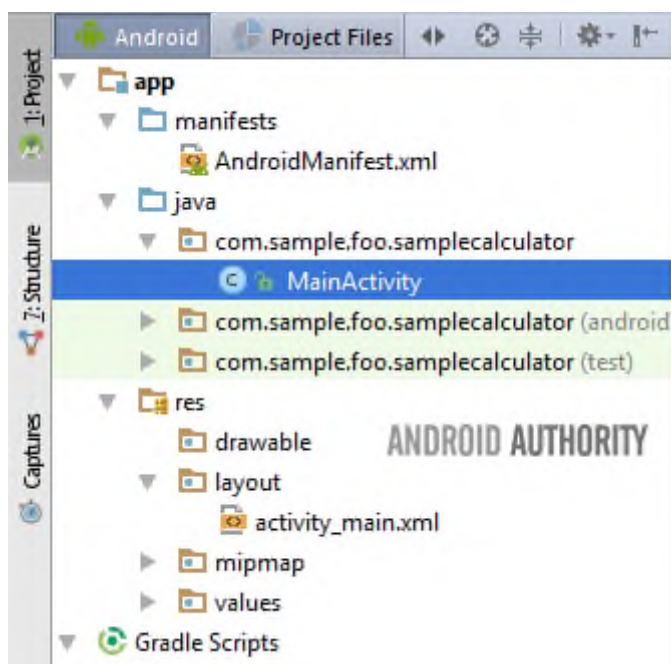
Полный исходный код калькулятора, описанного ниже, доступен для использования и изменения на [github](#).

Создание проекта

Первое, что нужно сделать - это создать в **Android Studio** новый проект: **Start a new Android Studio project** или **File - New - New Project**:



Для этого руководства мы выбрали в панели «*Add an Activity to Mobile*» опцию «*EmptyActivity*», для «*MainActivity*» мы оставили имя по умолчанию — «*Activity*». На этом этапе структура должна выглядеть, как показано на рисунке ниже. У вас есть **MainActivity** внутри пакета проекта и файл **activity_main.xml** в папке **layout**:



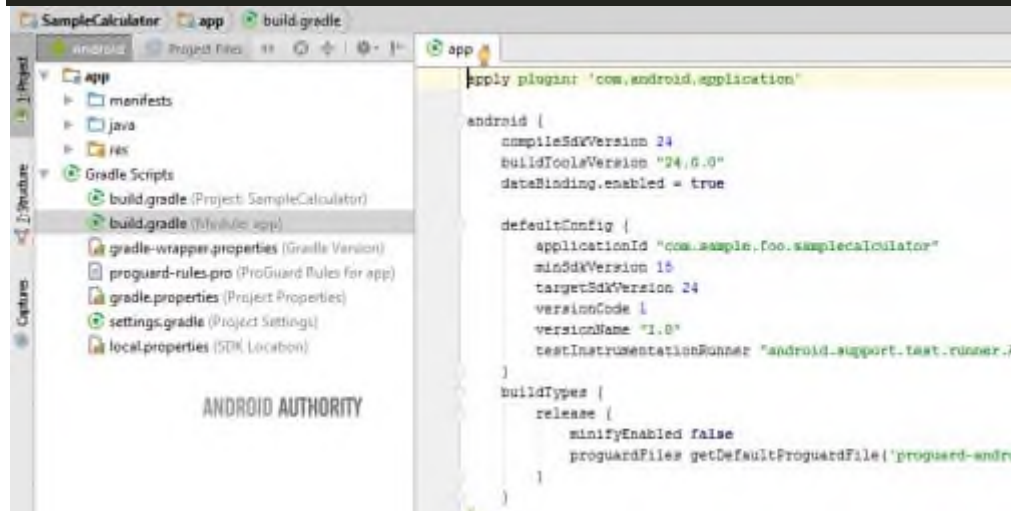
Включение привязки данных в проекте

Перед тем, как создать приложение для Android с нуля, нужно уяснить, что использование привязки данных помогает напрямую обращаться к виджетам (**Buttons**, **EditText** и **TextView**), а не находить их с помощью

методов **findViewById()**. Чтобы включить привязку данных, добавить следующую строку кода в файл **build.gradle**.

JAVA

```
android {  
    ...  
    dataBinding.enabled = true  
    ...  
}
```



Разработка макета калькулятора

Для включения привязки данных в файле **activity_main.xml** требуется еще одно изменение. Оберните сгенерированный корневой тег (**RelativeLayout**) в **layout**, таким образом сделав его новым корневым тегом.

XML

```
<?xml version="1.0" encoding="utf-8"?>  
<layout>  
    <RelativeLayout>  
        ...  
    </RelativeLayout>  
</layout>
```

Как научиться создавать приложения для Android? Читайте наше руководство дальше.

Тег layout - это предупреждает систему построения приложения, что этот файл макета будет использовать привязку данных. Затем система генерирует для этого файла макета класс **Binding**. Поскольку целевой **XML-файл** называется **activity_main.xml**, система построения приложения создаст класс **ActivityMainBinding**, который можно использовать в приложении, как

и любой другой класс **Java**. Имя класса составляется из имени файла макета, в котором каждое слово через подчеркивание будет начинаться с заглавной буквы, а сами подчеркивания убираются, и к имени добавляется слово «*Binding*».

Теперь перейдите к файлу **MainActivity.java**. Создайте закрытый экземпляр **ActivityMainBinding** внутри вашего класса, а в методе **onCreate()** удалите строку **setContentView ()** и вместо нее добавьте **DataBindingUtil.setContentView()**, как показано ниже.

JAVA

```
public class MainActivity extends AppCompatActivity {  
    private ActivityMainBinding binding;  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        binding = DataBindingUtil.setContentView(this, R.layout.activity_main);  
    }  
}
```

Общие принципы создания виджетов макета

В приложении калькулятора есть четыре основных элемента:

RelativeLayout - определяет, как другие элементы будут укладываться или отображаться на экране. **RelativeLayout** используется для позиционирования дочерних элементов по отношению друг к другу или к самим себе.

TextView - элемент используется для отображения текста. Пользователи не должны взаимодействовать с этим элементом. С помощью **TextView** отображается результат вычислений.

EditText - похож на элемент **TextView**, с той лишь разницей, что пользователи могут взаимодействовать с ним и редактировать текст. Но поскольку калькулятор допускает только фиксированный набор вводимых данных, мы устанавливаем для него статус «не редактируемый». Когда пользователь нажимает на цифры, мы выводим их в **EditText**.

Button - реагирует на клики пользователя. При создании простого приложения для Андроид мы используем кнопки для цифр и операторов действий в калькуляторе.

Создание макета калькулятора

Многопоточность в Java – руководство с примерами



Код макета калькулятора объемный. Это связано с тем, что мы должны явно определять и тщательно позиционировать каждую из кнопок интерфейса. Ниже представлен фрагмент сокращенной версии файла макета **activity_main**:

```
<layout>
  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
```

```
android:layout_height="match_parent"
android:paddingBottom="@dimen/activity_vertical_margin"
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
tools:context="com.sample.foo.samplecalculator.MainActivity">
```

```
<TextView
    android:id="@+id/infoTextView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="30dp"
    android:textSize="30sp" />
```

```
<EditText
    android:id="@+id/editText"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/infoTextView"
    android:enabled="false"
    android:gravity="bottom"
    android:lines="2"
    android:maxLines="2"
    android:textSize="20sp" />
```

```
<Button
    android:id="@+id/buttonSeven"
    style="@style/Widget.AppCompat.Button.Borderless"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/editText"
    android:text="7"
    android:textSize="20sp" />
```

```
<Button
    android:id="@+id/buttonEight"
    style="@style/Widget.AppCompat.Button.Borderless"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/editText"
    android:layout_toRightOf="@id/buttonSeven"
    android:text="8"
    android:textSize="20sp" />
```

```
<Button
```



```
android:id="@+id/buttonNine"
style="@style/Widget.AppCompat.Button.Borderless"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@id/editText"
android:layout_toRightOf="@id/buttonEight"
android:text="9"
android:textSize="20sp" />
```

...

...

```
<Button
    android:id="@+id/buttonDot"
    style="@style/Widget.AppCompat.Button.Borderless"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/buttonOne"
    android:text="."
    android:textSize="20sp" />
```

```
<Button
    android:id="@+id/buttonZero"
    style="@style/Widget.AppCompat.Button.Borderless"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignRight="@id/buttonEight"
    android:layout_below="@id/buttonTwo"
    android:text="0"
    android:textSize="20sp" />
```

```
<Button
    android:id="@+id/buttonEqual"
    style="@style/Widget.AppCompat.Button.Borderless"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignRight="@id/buttonNine"
    android:layout_below="@id/buttonThree"
    android:text="="
    android:textSize="20sp" />
```

```
<Button
    android:id="@+id/buttonDivide"
    style="@style/Widget.AppCompat.Button.Borderless"
```

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignTop="@id/buttonNine"
android:layout_toRightOf="@id/buttonNine"
android:text="/"
android:textSize="20sp" />
```

```
<Button
    android:id="@+id/buttonMultiply"
    style="@style/Widget.AppCompat.Button.Borderless"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignTop="@id/buttonSix"
    android:layout_toRightOf="@id/buttonSix"
    android:text="*"
    android:textSize="20sp" />
```

```
<Button
    android:id="@+id/buttonSubtract"
    style="@style/Widget.AppCompat.Button.Borderless"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignTop="@id/buttonThree"
    android:layout_toRightOf="@id/buttonThree"
    android:text="-"
    android:textSize="20sp" />
```

```
<Button
    android:id="@+id/buttonAdd"
    style="@style/Widget.AppCompat.Button.Borderless"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignTop="@id/buttonEqual"
    android:layout_toRightOf="@id/buttonEqual"
    android:text="+"
    android:textSize="20sp" />
```

```
<Button
    android:id="@+id/buttonClear"
    style="@style/Widget.AppCompat.Button.Borderless"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignRight="@id/buttonAdd"
    android:layout_below="@id/buttonAdd"
    android:layout_marginTop="@dimen/activity_vertical_margin"
```

```
        android:text="C"  
        android:textSize="20sp" />  
    </RelativeLayout>  
</layout>
```

Язык программирования Java - руководство для начинающих

Внутренние компоненты калькулятора

Перед тем, как создать приложение на телефон **Android**, отметим, что **valueOne** и **valueTwo** содержат цифры, которые будут использоваться. Обе переменные имеют тип **double**, поэтому могут содержать числа с десятичными знаками и без них. Мы устанавливаем для **valueOne** специальное значение **NaN** (*не число*) - подробнее это будет пояснено ниже.

```
private double valueOne = Double.NaN;  
private double valueTwo;
```

Этот простой калькулятор сможет выполнять только операции сложения, вычитания, умножения и деления. Поэтому мы определяем четыре статических символа для представления этих операций и переменную **CURRENT_ACTION**, содержащую следующую операцию, которую мы намереваемся выполнить.

```
private static final char ADDITION = '+';  
private static final char SUBTRACTION = '-';  
private static final char MULTIPLICATION = '*';  
private static final char DIVISION = '/';  
private char CURRENT_ACTION;
```

Затем мы используем класс **DecimalFormat** для форматирования результата. Конструктор десятичного формата позволяет отображать до десяти знаков после запятой.

```
decimalFormat = new DecimalFormat("#.#####");
```

Обработка нажатий на цифры

В нашем создаваемом простом приложении для **Андроид** всякий раз, когда пользователь нажимает на цифру или точку, нам нужно добавить эту цифру в **editText**. Пример кода ниже иллюстрирует, как это делается для цифры ноль (0).

```
binding.buttonZero.setOnClickListener(new View.OnClickListener() {
```

```
@Override
public void onClick(View view) {
    binding.editText.setText(binding.editText.getText() + "0");
}
});
```

Обработка кликов по кнопкам операторов



Обработка нажатия кнопок операторов (*действий*) выполняется по-другому. Сначала нужно выполнить все ожидающие в очереди вычисления. Поэтому мы определяем метод **computeCalculation**. В **computeCalculation**, если **valueOne** является допустимым числом, мы считываем **valueTwo** из **editText** и выполняем текущие операции в очереди.

Если же **valueOne** является NaN, для **valueOne** присваивается цифра в **editText**.

JAVA

```
private void computeCalculation() {
    if(!Double.isNaN(valueOne)) {
        valueTwo = Double.parseDouble(binding.editText.getText().toString());
        binding.editText.setText(null);
        if(CURRENT_ACTION == ADDITION)
            valueOne = this.valueOne + valueTwo;
        else if(CURRENT_ACTION == SUBTRACTION)
            valueOne = this.valueOne - valueTwo;
        else if(CURRENT_ACTION == MULTIPLICATION)
            valueOne = this.valueOne * valueTwo;
        else if(CURRENT_ACTION == DIVISION)
            valueOne = this.valueOne / valueTwo;
    }
    else {
        try {
            valueOne = Double.parseDouble(binding.editText.getText().toString());
        }
        catch (Exception e){ }
    }
}
```

Продолжаем создавать копию приложения на **Андроид**. Для каждого оператора мы сначала вызываем **computeCalculation()**, а затем устанавливаем для выбранного оператора **CURRENT_ACTION**. Для оператора равно (=) мы вызываем **computeCalculation()**, а затем очищаем содержимое **valueOne** и **CURRENT_ACTION**.

JAVA

```
binding.buttonAdd.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        computeCalculation();
        CURRENT_ACTION = ADDITION;
        binding.infoTextView.setText(decimalFormat.format(valueOne) + "+");
        binding.editText.setText(null);
    }
});
binding.buttonSubtract.setOnClickListener(new View.OnClickListener() {
    @Override
```

```

        public void onClick(View view) {
            computeCalculation();
            CURRENT_ACTION = SUBTRACTION;
            binding.infoTextView.setText(decimalFormat.format(valueOne) + "-");
            binding.editText.setText(null);
        }
    });
    binding.buttonMultiply.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            computeCalculation();
            CURRENT_ACTION = MULTIPLICATION;
            binding.infoTextView.setText(decimalFormat.format(valueOne) + "*");
            binding.editText.setText(null);
        }
    });
    binding.buttonDivide.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            computeCalculation();
            CURRENT_ACTION = DIVISION;
            binding.infoTextView.setText(decimalFormat.format(valueOne) + "/");
            binding.editText.setText(null);
        }
    });
    binding.buttonEqual.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            computeCalculation();

            binding.infoTextView.setText(binding.infoTextView.getText().toString() +
                decimalFormat.format(valueTwo) + " = " +
                decimalFormat.format(valueOne));
            valueOne = Double.NaN;
            CURRENT_ACTION = '0';
        }
    });

```

Мы завершили создание простого калькулятора. Теперь вы сможете **создать приложение для Android** сами.

10 лабораторная работа

Тема: Flutter: Работа с базами данных в Android Studio с использованием исходных команд SQL SELECT, INSERT, UPDATE, DELETE

Цель работы: Работа с базой данных и использование исходных команд.

SQLite это наиболее популярный способ для хранения данных на мобильных устройствах. В этой статье мы будем использовать пакет sqflite для использования SQLite. Sqflite — одна из наиболее часто используемых и актуальных библиотек для подключения SQLite базы данных в Flutter.

1. Добавление зависимостей

В нашем проекте открываем файл **pubspec.yaml**. Под зависимостями добавляем последнюю версию sqflite и path_provider.

```
dependencies:  
  flutter:  
    sdk: flutter  
  sqflite: any  
  path_provider: any
```

2. Создадим DB Client

Теперь создадим новый файл Database.dart. В нем создадим синглтон.

Почему нам нужен синглтон: мы используем этот паттерн для уверенности в том что у нас есть только одна сущность класса и для обеспечения глобальной точки входа в него.

1. Создадим приватный конструктор, который может использоваться только внутри этого класса

```
class DBProvider {  
  DBProvider._();
```

```
static final DBProvider db = DBProvider._();  
}
```

2. Настроим базу данных

Следующим шагом будет создания объекта базы данных и предоставим геттер, где мы создадим объект базы данных, если он еще не был создан (ленивая инициализация)

```
static Database _database;  
Future<Database> get database async {  
  if (_database != null)  
    return _database;  
  
  // if _database is null we instantiate it  
  _database = await initDB();  
  return _database;  
}
```

Если нет объекта, присвоенного базе данных, то мы вызовем функцию `initDB` для создания базы данных. В этой функции мы получим путь для сохранения базы данных и создания желаемых таблиц

```
initDB() async {  
  Directory documentsDirectory = await getApplicationDocumentsDirectory();  
  String path = join(documentsDirectory.path, "TestDB.db");  
  return await openDatabase(path, version: 1, onOpen: (db) {  
  }, onCreate: (Database db, int version) async {  
    await db.execute("CREATE TABLE Client ("  
      "id INTEGER PRIMARY KEY,"  
      "first_name TEXT,"  
      "last_name TEXT,"  
      "blocked BIT"  
    ")");  
  });  
}
```

```
});  
}
```

3. Создадим класс модели

Данные внутри базы данных будут конвертироваться в Dart Maps. Нам необходимо создать классы моделей с toMap и fromMap методами.

Для создания классов моделей, я собираюсь использовать этот [сайт](#)

Наша

модель:

```
/// ClientModel.dart  
import 'dart:convert';  
  
Client clientFromJson(String str) {  
  final jsonData = json.decode(str);  
  return Client.fromJson(jsonData);  
}  
  
String clientToJson(Client data) {  
  final dyn = data.toJson();  
  return json.encode(dyn);  
}  
  
class Client {  
  int id;  
  String firstName;  
  String lastName;  
  bool blocked;  
  
  Client({  
    this.id,  
    this.firstName,  
    this.lastName,
```

```

        this.blocked,
    });

    factory Client.fromJson(Map<String, dynamic> json) => new Client(
        id: json["id"],
        firstName: json["first_name"],
        lastName: json["last_name"],
        blocked: json["blocked"],
    );

    Map<String, dynamic> toJson() => {
        "id": id,
        "first_name": firstName,
        "last_name": lastName,
        "blocked": blocked,
    };
}

```

4. CRUD operations

Create

Используя

rawInsert:

```

newClient(Client newClient) async {
    final db = await database;
    var res = await db.rawInsert(
        "INSERT Into Client (id,first_name)"
        " VALUES (${newClient.id},${newClient.firstName})");
    return res;
}

```

Используя

insert:

```

newClient(Client newClient) async {
    final db = await database;
    var res = await db.insert("Client", newClient.toMap());
    return res;
}

```

Другой пример с использованием большого ID в качестве нового ID

```

newClient(Client newClient) async {
    final db = await database;
    //get the biggest id in the table
    var table = await db.rawQuery("SELECT MAX(id)+1 as id FROM Client");
    int id = table.first["id"];
    //insert to the table using the new id
    var raw = await db.rawQuery(
        "INSERT Into Client (id,first_name,last_name,blocked)"
        " VALUES (?, ?, ?, ?)",
        [id, newClient.firstName, newClient.lastName, newClient.blocked]);
    return raw;
}

```

Read

Get Client by id

```

getClient(int id) async {
    final db = await database;
    var res = await db.query("Client", where: "id = ?", whereArgs: [id]);
    return res.isNotEmpty ? Client.fromMap(res.first) : Null ;
}

```

Get all Clients with a condition

```

getAllClients() async {
    final db = await database;
    var res = await db.query("Client");
    List<Client> list =
        res.isNotEmpty ? res.map((c) => Client.fromMap(c)).toList() : [];
    return list;
}

```

Получить только заблокированных клиентов

```

getBlockedClients() async {
    final db = await database;
    var res = await db.rawQuery("SELECT * FROM Client WHERE blocked=1");
    List<Client> list =
        res.isNotEmpty ? res.toList().map((c) => Client.fromMap(c)) : null;
    return list;
}

```

Update

Update an existing Client

```

updateClient(Client newClient) async {
    final db = await database;
    var res = await db.update("Client", newClient.toMap(),
        where: "id = ?", whereArgs: [newClient.id]);
    return res;
}

```

Блокировка/разблокировка клиента

```
blockOrUnblock(Client client) async {  
    final db = await database;  
    Client blocked = Client(  
        id: client.id,  
        firstName: client.firstName,  
        lastName: client.lastName,  
        blocked: !client.blocked);  
    var res = await db.update("Client", blocked.toMap(),  
        where: "id = ?", whereArgs: [client.id]);  
    return res;  
}
```

Delete

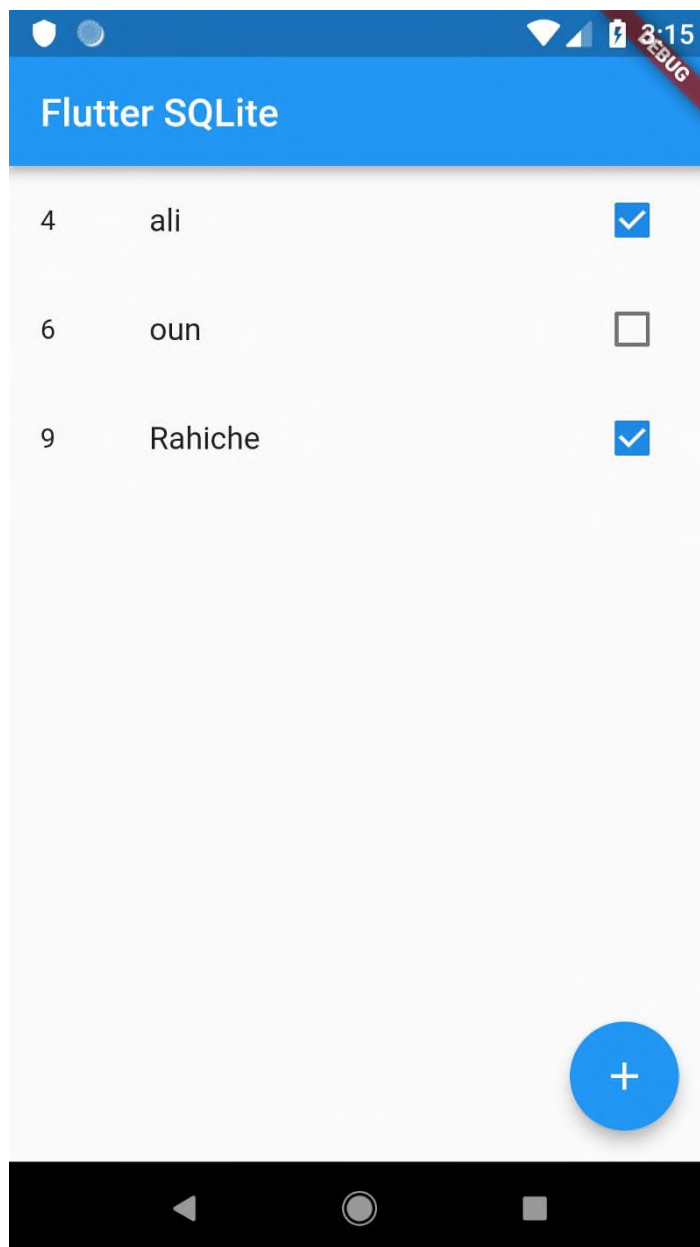
Delete one Client

```
deleteClient(int id) async {  
    final db = await database;  
    db.delete("Client", where: "id = ?", whereArgs: [id]);  
}
```

Delete All Clients

```
deleteAll() async {  
    final db = await database;  
    db.rawDelete("Delete * from Client");  
}
```

Demo



Для нашего демо мы создадим простое приложение, отображающее нашу базу данных.

Для начала сверстаем экран

```
Widget build(BuildContext context) {  
  return Scaffold(  
    appBar: AppBar(title: Text("Flutter SQLite")),  
    body: FutureBuilder<List<Client>>(  
      future: DBProvider.db.getAllClients(),
```

```

builder: (BuildContext context, AsyncSnapshot<List<Client>> snapshot) {
  if (snapshot.hasData) {
    return ListView.builder(
      itemCount: snapshot.data.length,
      itemBuilder: (BuildContext context, int index) {
        Client item = snapshot.data[index];
        return ListTile(
          title: Text(item.lastName),
          leading: Text(item.id.toString()),
          trailing: Checkbox(
            onChanged: (bool value) {
              DBProvider.db.blockClient(item);
              setState(() {});
            },
            value: item.blocked,
          ),
        );
      },
    );
  } else {
    return Center(child: CircularProgressIndicator());
  }
},
floatingActionButton: FloatingActionButton(
  child: Icon(Icons.add),
  onPressed: () async {
    Client rnd = testClients[Math.Random().nextInt(testClients.length)];
    await DBProvider.db.newClient(rnd);
    setState(() {});
  },
),
);
}

```

Заметки:

1. FutureBuilder используется для получения данных из бд

2. FAB для инициализации тестовых клиентов

```
List<Client> testClients = [  
    Client(firstName: "Raouf", lastName: "Rahiche", blocked: false),  
    Client(firstName: "Zaki", lastName: "oun", blocked: true),  
    Client(firstName: "oussama", lastName: "ali", blocked: false),  
];
```

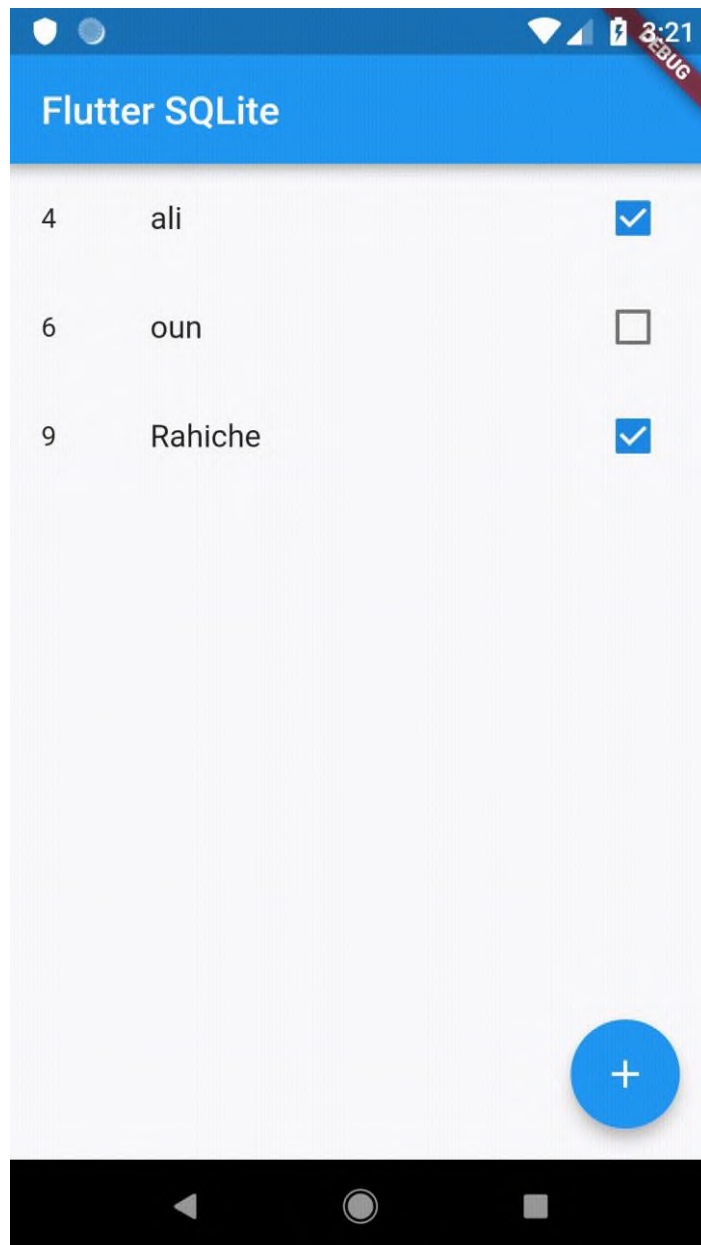
3. CircularProgressIndicator показывается, когда нет данных

4. Когда пользователь кликает по чекбоксам клиент блокируется/разблокируется

Теперь очень просто добавлять новые фичи, например если мы хотим удалить клиента, в момент когда он свайпнут, просто оберните ListTile в Dismissible widget

Вот так:

```
return Dismissible(  
    key: UniqueKey(),  
    background: Container(color: Colors.red),  
    onDismissed: (direction) {  
        DBProvider.db.deleteClient(item.id);  
    },  
    child: ListTile(...),  
);
```



Рефакторинг для использования BLoC паттерна

Мы сделали многое в этой статье, но в приложения в реальном мире, инициализация состояний в UI слое не очень хорошая идея. Отделим логику от UI.

Существует множество паттернов в Flutter, но мы будем использовать BLoC так как он наиболее гибкий для настройки.

```

class ClientsBloc {
  ClientsBloc() {
    getClients();
  }
  final _clientController = StreamController<List<Client>>.broadcast();
  get clients => _clientController.stream;

  dispose() {
    _clientController.close();
  }

  getClients() async {
    _clientController.sink.add(await DBProvider.db.getAllClients());
  }
}

```

Заметки:

Notes:

1. `getClients` получает данные из БД (`Client table`) асинхронно. Мы будем использовать этот метод всегда, когда нам будет необходимо обновить таблицу, следовательно стоит поместить его в тело конструктора.
2. Мы создали `StreamController.broadcast`, для того чтобы слушать широковещательные события более одного раза. В нашем примере это не имеет особо значения, поскольку мы слушаем их только один раз, но неплохо было бы реализовать это на будущее.
3. Не забываем закрывать потоки. Таким образом мы предотвратим мемори лики. В нашем примере мы закрываем их используя `dispose method` в `StatefulWidget`

Теперь посмотрим на код

```

blockUnblock(Client client) {
  DBProvider.db.blockOrUnblock(client);
}

```

```

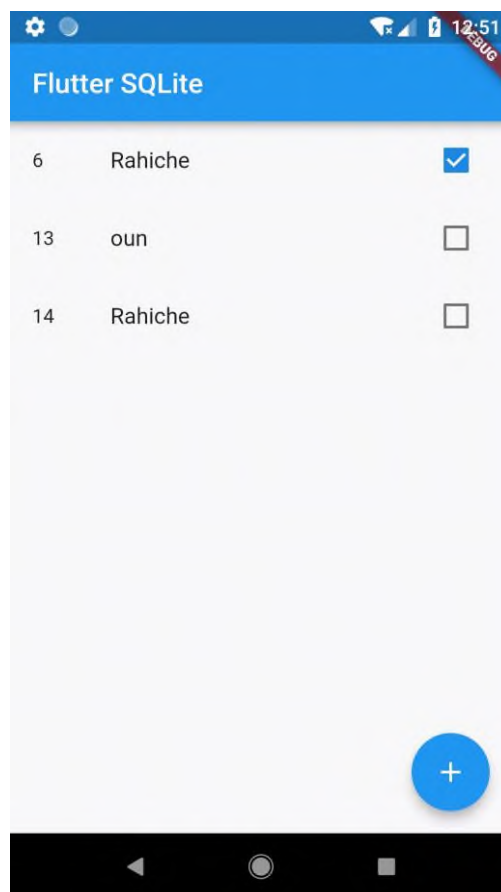
    getClients();
  }

  delete(int id) {
    DBProvider.db.deleteClient(id);
    getClients();
  }

  add(Client client) {
    DBProvider.db.newClient(client);
    getClients();
  }

```

И наконец финальный результат



Исходники можно посмотреть здесь — [Github](#)

Создадим BLoC

11 лабораторная работа

Тема: Работа с API на примере мобильного приложения с запущенным Telegram-ботом. Управляйте каналом или группой Telegram с помощью мобильного приложения.

Цель работы: Ознакомление с созданием telegram бота и управление каналом и группой.

Для любого бизнеса важно установить канал общения с клиентами. Но сделать это не так просто. Психология людей такова, что они не хотят захламлять память своего смартфона новым фирменным приложением из того места, которое они посетили. Совсем другое дело — чат-бот. Ненавязчивый и дружелюбный.

А кроме того — со всеми необходимыми фишками: рекламными акциями, скидками и быстрым заказом. И уведомления в одном единственном удобном мессенджере. Почему бы его не реализовать в своем бизнесе? Тем более, что это не так сложно, как вы думаете.

Сегодня мы поговорим о ботах и их API (telegram api) на базе популярного мессенджера Telegram.

Содержание

Telegram Bot API и Telegram API BotFather: быстрый StartПростой эхо-бот
aiogram — асинхронная библиотека

Создаем эхо-бота

Оформление сообщений

Учим бот-модерации

Делаем бота админом

Telegram Bot API и Telegram API

Все началось с того, что Николай Дуров совместно с командой программистов создал криптографический протокол. Его движок задействовал комбинацию симметричного шифрования

AES, протокол Диффи-Хеллмана для обмена ключами шифрования между клиентами и ряд хеш-функций. На основе этого протокола был построен MTProto — механизм, позволяющий пользователям сегодня использовать Telegram-мессенджеры.

На данный момент есть два основных инструмента API, с помощью которых можно задействовать сервисы Telegram — Telegram Bot API и Telegram API.

Первый служит для разработки чат-ботов, второй позволяет делать полностью кастомные Telegram-клиенты. Разработчикам также доступна открытая библиотека TDLib (Telegram Database Library), с помощью которой

можно создавать свою версию мессенджера с уникальными опциями (как например, Telegram X, построенный именно на TDLib).

Telegram Bot API является надстройкой над Telegram API, поэтому пользоваться Bot API можно без знаний о механизме используемого протокола MTProto.

Для его работы задействован промежуточный сервер с HTTPS-интерфейсом, который шифрует трафик и обеспечивает связь с Telegram API. Bot API позволяет легко создавать программы, которые используют интерфейс Telegram для выполнения кода на локальном сервере. Пользователи могут взаимодействовать с ботами, отправляя им сообщения, команды и встроенные запросы.

Принцип работы любого бота заключается в том, что он перманентно направляет запросы на сервер и регулярно получает обновления. Получать их можно двумя способами. Во-первых, можно использовать вебхуки, когда сервер делает обратный вызов на указанный URL. А во-вторых, можно просто «забрасывать» запросами Telegram, получая постоянные ответы.

Обратите внимание — получать уведомления о новых сообщениях в боте и других событиях вы можете всего один раз. Поэтому, если данные чата представляются вам очень важными, то придется самостоятельно сохранять список чатов и историю старых сообщений. Если вы случайно сотрете/потеряете эту информацию, то вы ее больше никак не восстановите.

Сервер

Bot API имеет возможность настройки порта и локального IP-адреса для вебхука, а также поддерживает до 100000 одновременных подключений — более чем достаточно для большинства задач.

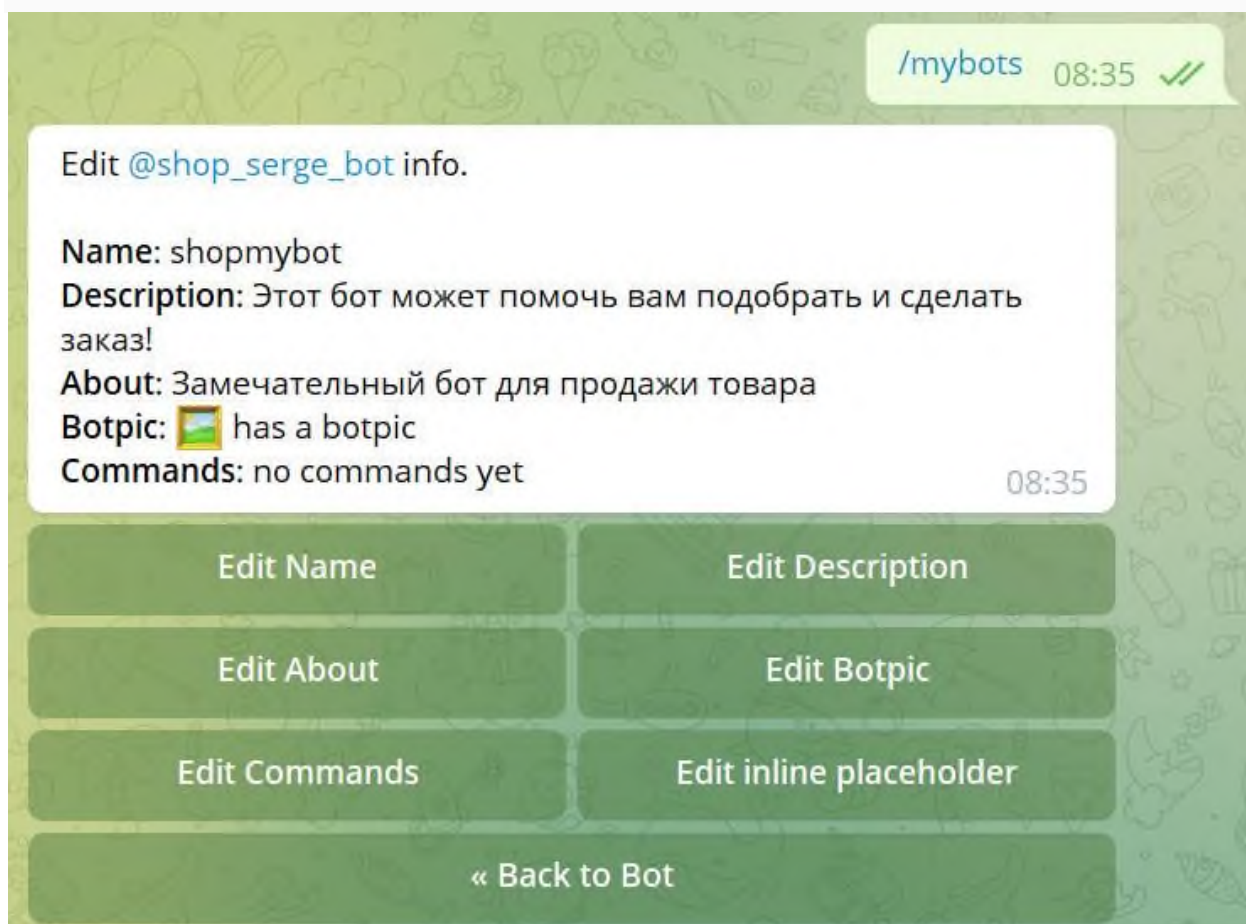
BotFather: быстрый Start Чтобы приступить к созданию собственного бота, необходимо получить токен для авторизации и подключения через API. Делается это при помощи служебного бота. Введите в поиске Telegram его имя — BotFather. Далее следует выбрать команду newbot и дать имя боту.

Затем BotFather спросит вас имя, которое обязательно должно заканчиваться на bot, например, shop_serge_bot. Далее для бота будет сгенерирован уникальный токен, который будет выглядеть примерно так — 2093336709:AAGiH64Ec1R8r222sM9IywvlIGFkb7wFqyo.

Всего можно генерировать не более 20 ботов на одного пользователя. Управление ботами также происходит через меню команд служебного бота BotFather. Например, если вам потребуется настроить какой-то из ваших ботов, вы должны перейти по командам **mybots** и затем нажать на кнопку Edit Bot. Здесь вы сможете настроить имя — Edit Name

и указать описание — Edit Description.

Настраивая description для бота вы задаете текст, который будет показываться пользователям под сообщением «Что может делать этот бот?» когда они его будут подключать. В профиле бота также будут показаны сведения, которые вы задали в поле «About» (команда Edit About). В меню команд имеется возможность установить аватар — BotPic.



В меню настроек BotSettings включается режим встроенных запросов (по умолчанию эта опция отключена). Когда встроенные запросы активированы, пользователи могут вызвать вашего бота, просто введя имя пользователя в поле для ввода текста в любом чате, группе или канале.

Если Telegram использовать в коммерческих целях, чат-бот можно вооружить средствами для приема платежей. Стоит обратить внимание, что сам Telegram не занимается проведением транзакций, он лишь дает возможность подключить услуги длинного списка провайдеров.

В их числе такие платежные системы, как Stripe, YooMoney, Сбербанк, PayMaster, PSB, Tranzzo, Payme, CLICK, LiqPay, Portmone, Paymega, ECOMMPAY и др. Разумеется, чтобы использовать эти платежные системы, нужно быть юридическим лицом.

aiogram — асинхронная библиотека

Реализовывать свой проект удобнее при помощи библиотек, таких как aiogram— это, пожалуй, лучшая стабильная асинхронная библиотека для Python (<https://pypi.org/project/aiogram/>).

Относительно асинхронности нужно сказать несколько слов. Python — язык однопоточный, и любые команды в нем выполняются только после того, как завершилось выполнение предыдущей команды. Однако в случае с ботом программе необходимо отвечать на множество сообщений и делать это очень быстро. Можно, конечно, плодить множество потоков выполнений, но такой вариант чаще всего неприемлем, и альтернативой выступает асинхронная реализация задачи.

Когда в потоке выполнения команды возникает какая-то пауза, например, работает функция и у нее при этом есть какой-то таймаут ожидания, мы получаем участок времени, который не занят решением алгоритма. Асинхронность в aiogram позволяет заполнить эти промежутки, ускоряя работу нашего бота и быстро отвечая на сообщения. Устанавливается эта библиотека следующей командой:

```
pip install -U aiogram.
```

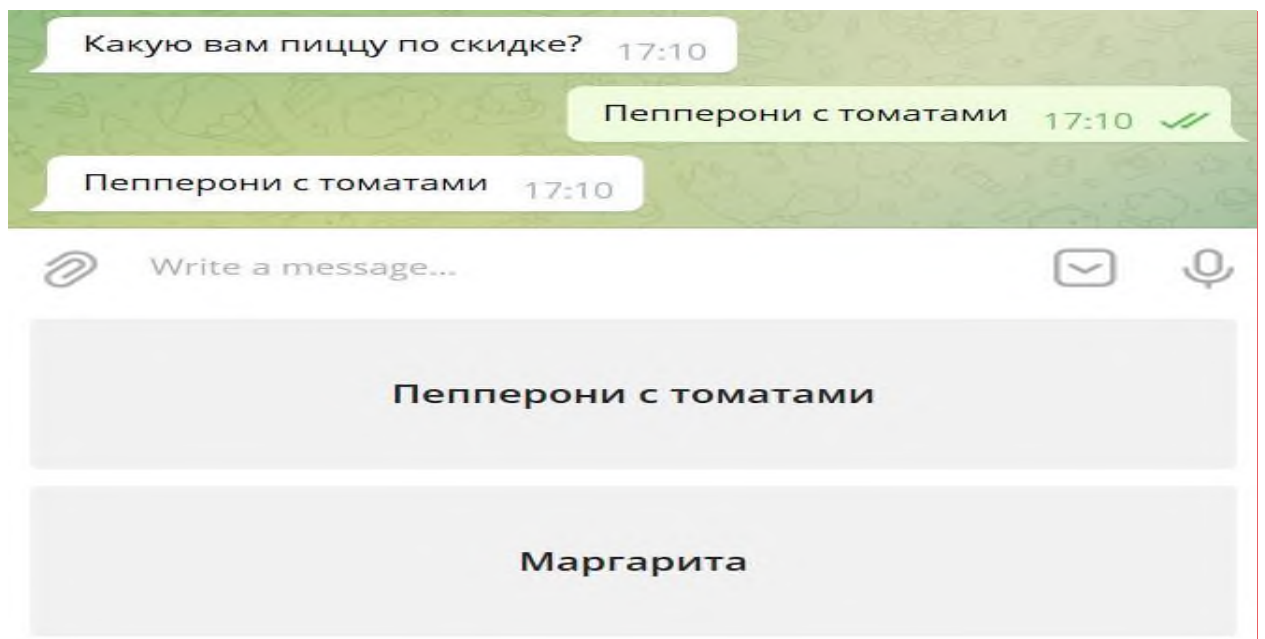
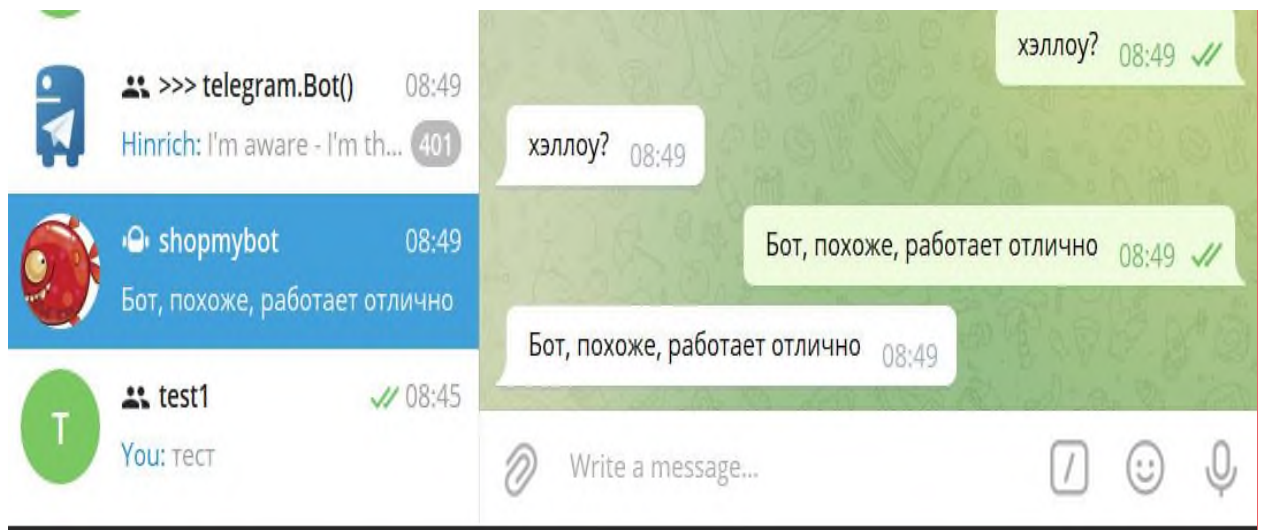
Создаем эхо-бот

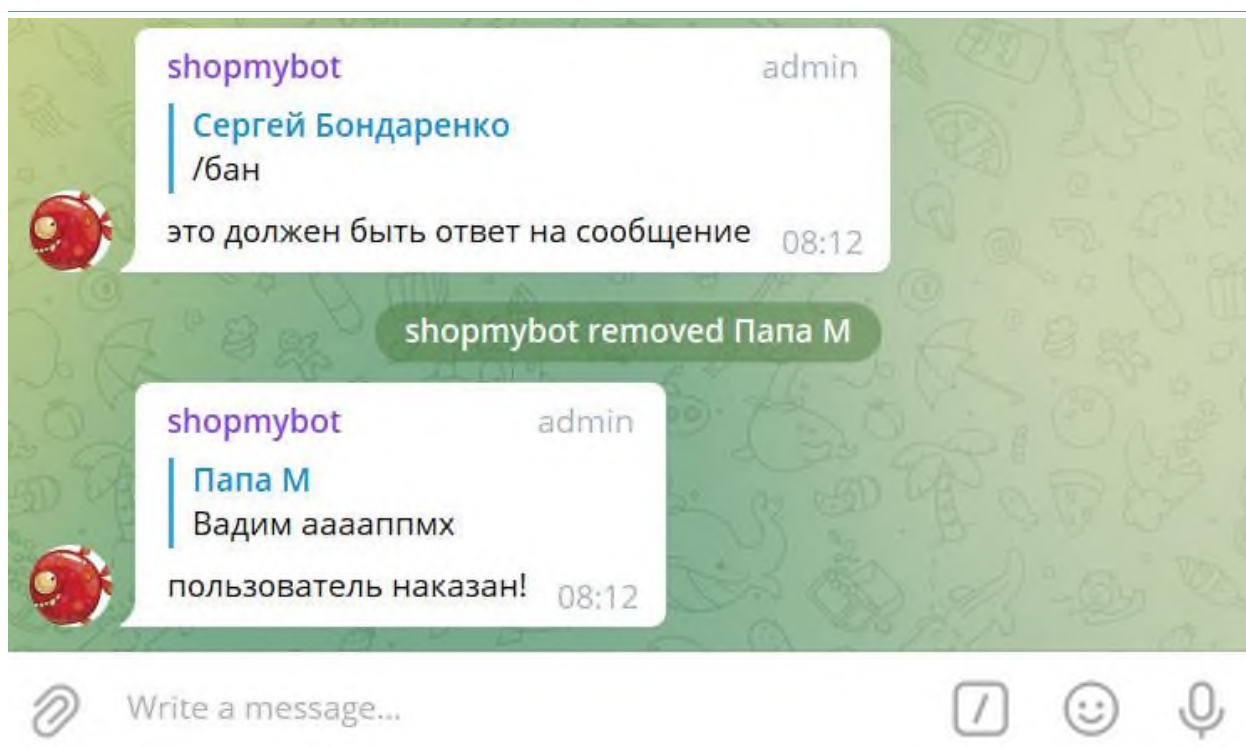
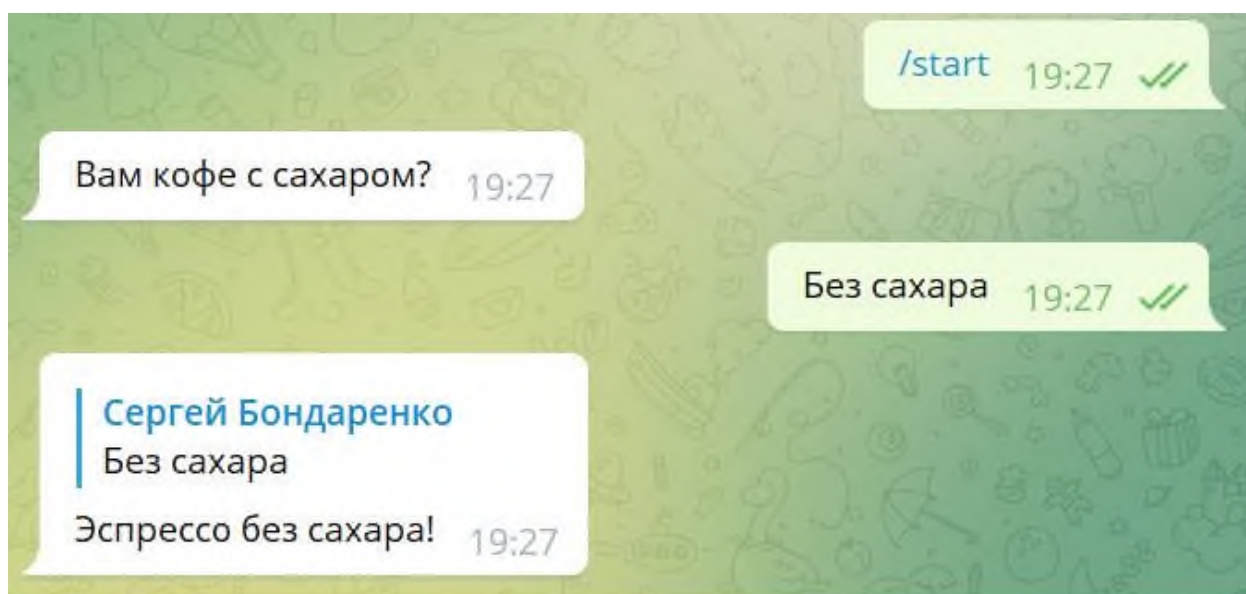
Перед тем как начать разбираться с Telegram-ботами, попробуйте создать простенький эхо-бот, который будет возвращать все поступающие сообщения. Это поможет понять механизм работы и разобраться с его функциями.

Создаем конфигурационный файл config.py и указываем в нем значение, сгенерированное ботом BotFather. Приступаем к написанию кода самого чат-бота.

Для начала импортируем конфигурацию и систему логирования. Затем подключаем все необходимые модули из aiogram. Указываем уровень логирования, а затем инициализируем бота, создав две переменные —

Bot и Dispatcher





12 лабораторная работа

Тема: Flutter: Создание мессенджера в Android Studio

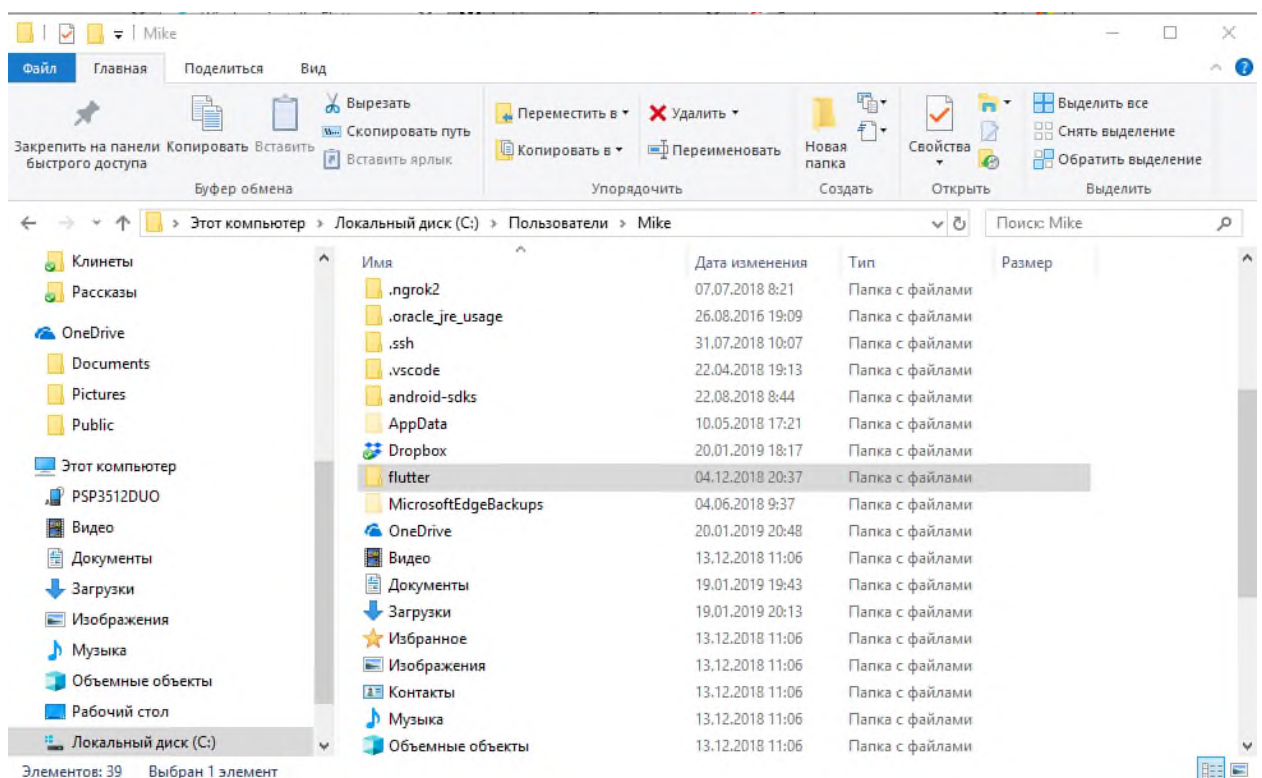
Цель работы: Создание простого мессенджера

Теперь за дело. Сейчас мы сделаем следующее

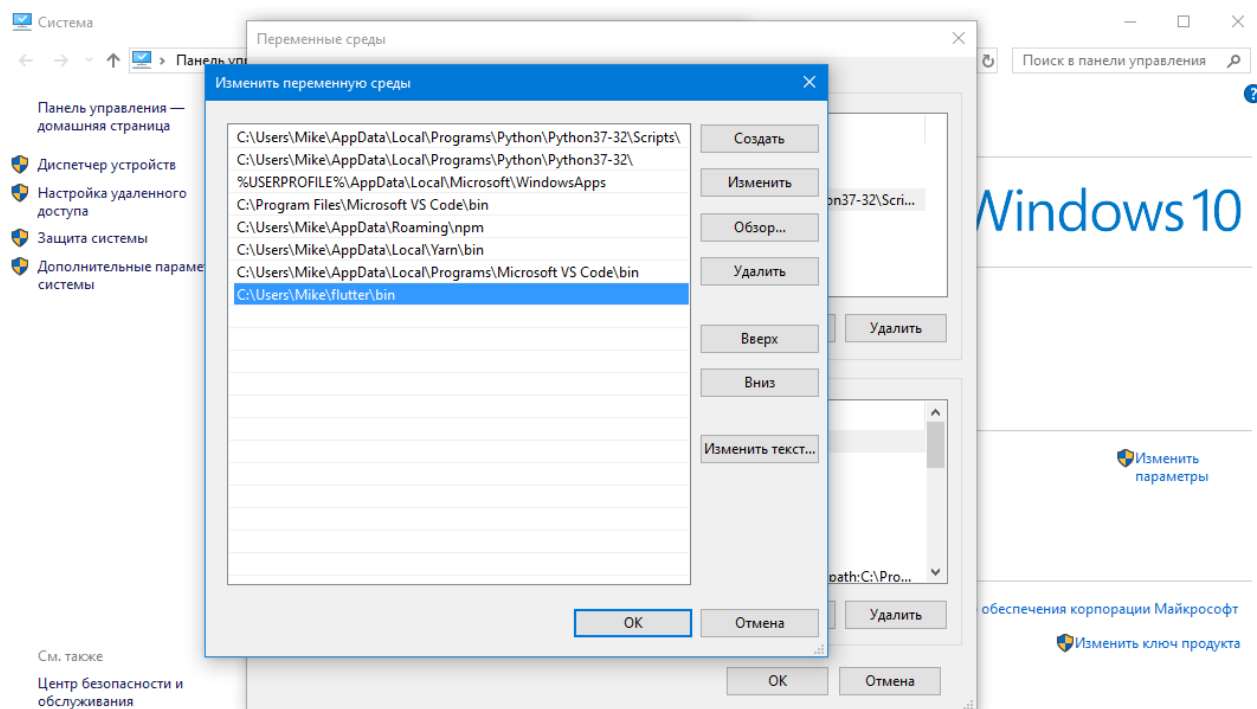
1. Установим Flutter SDK, и создадим проект из командной строки Windows.
2. Установим, JDK, Android Studio, плагины для Flutter и Dart, и создадим проект в Android Studio.
3. Установим VS Code, расширения для Flutter и Dart, и создадим проект в VS Code.

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 системы -> Переменные среды):



Можно создавать проект из командной строки Windows:

```
flutter create my_app
```

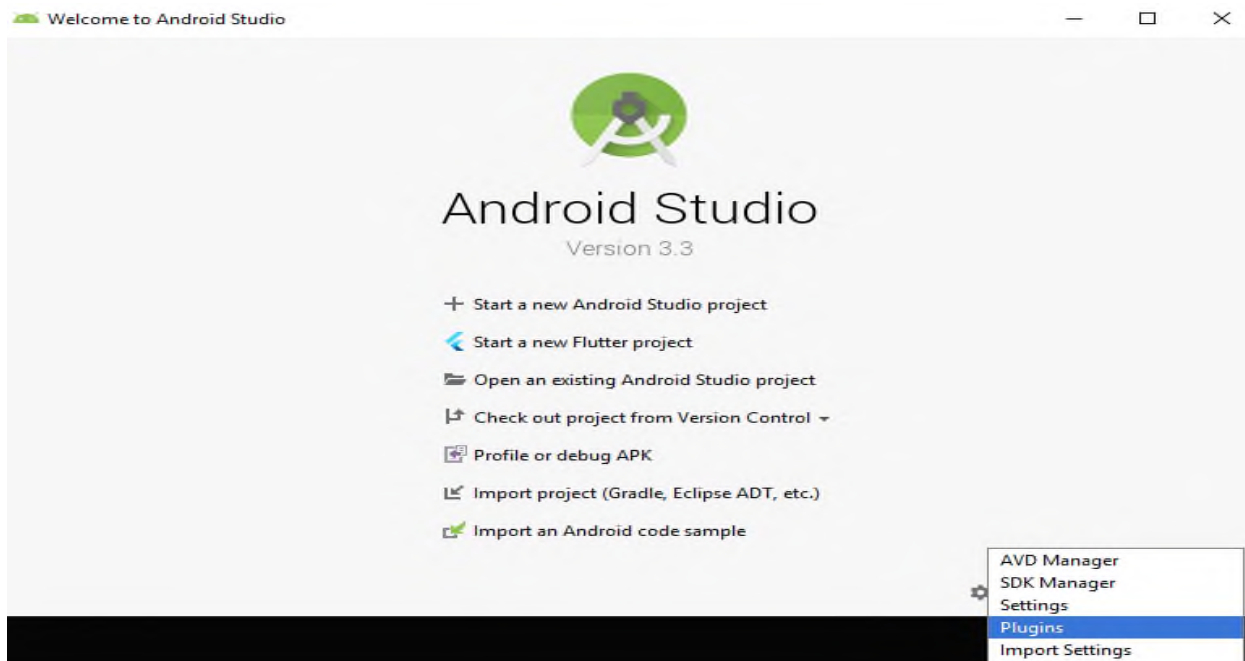
```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.17134.556]
(c) Корпорация Майкрософт (Microsoft Corporation), 2018. Все права защищены.

C:\Users\Mike>cd C:\Users\Mike\Documents\Flutter\projects
C:\Users\Mike\Documents\Flutter\projects>flutter create my_app
```

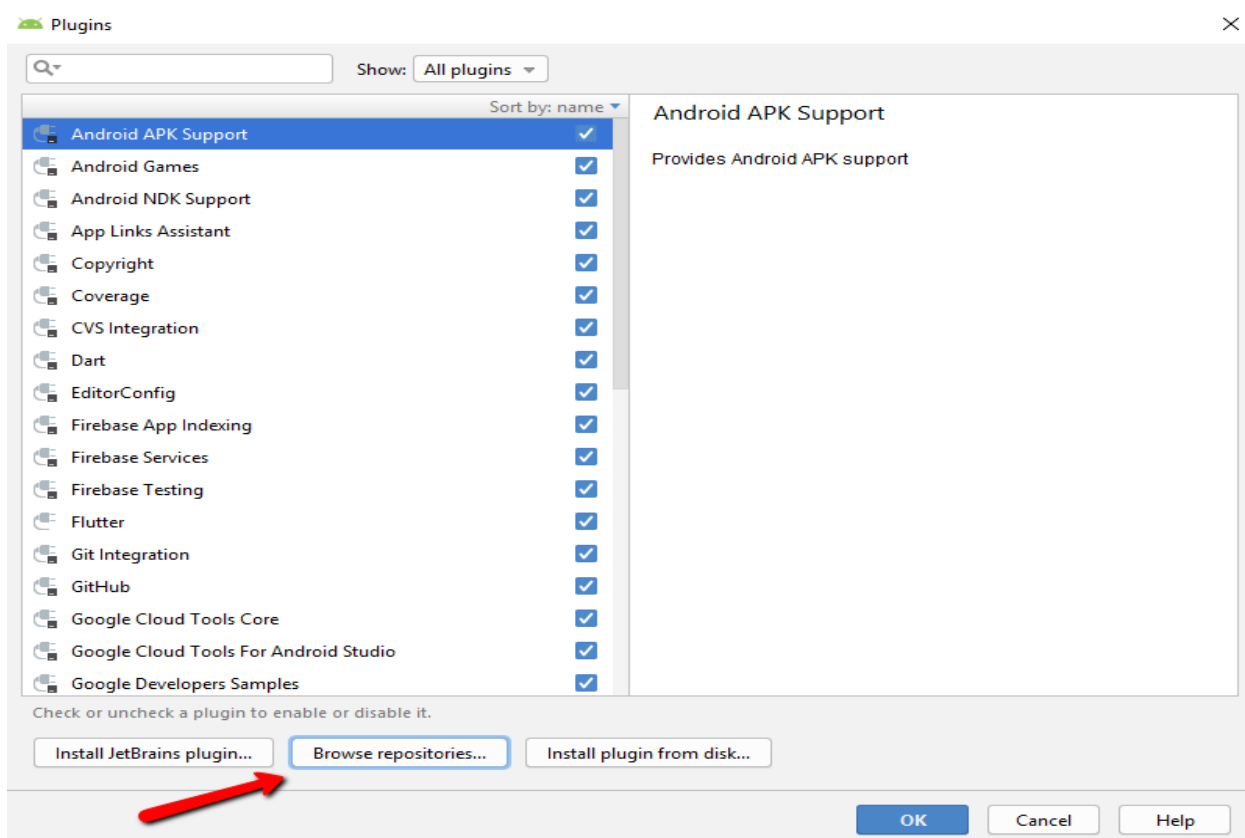
Готово! Файлы проекта можно редактировать любым текстовым редактором, хоть в блокноте. Но это хорошо разве что для мелких правок. Поэтому мы...

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

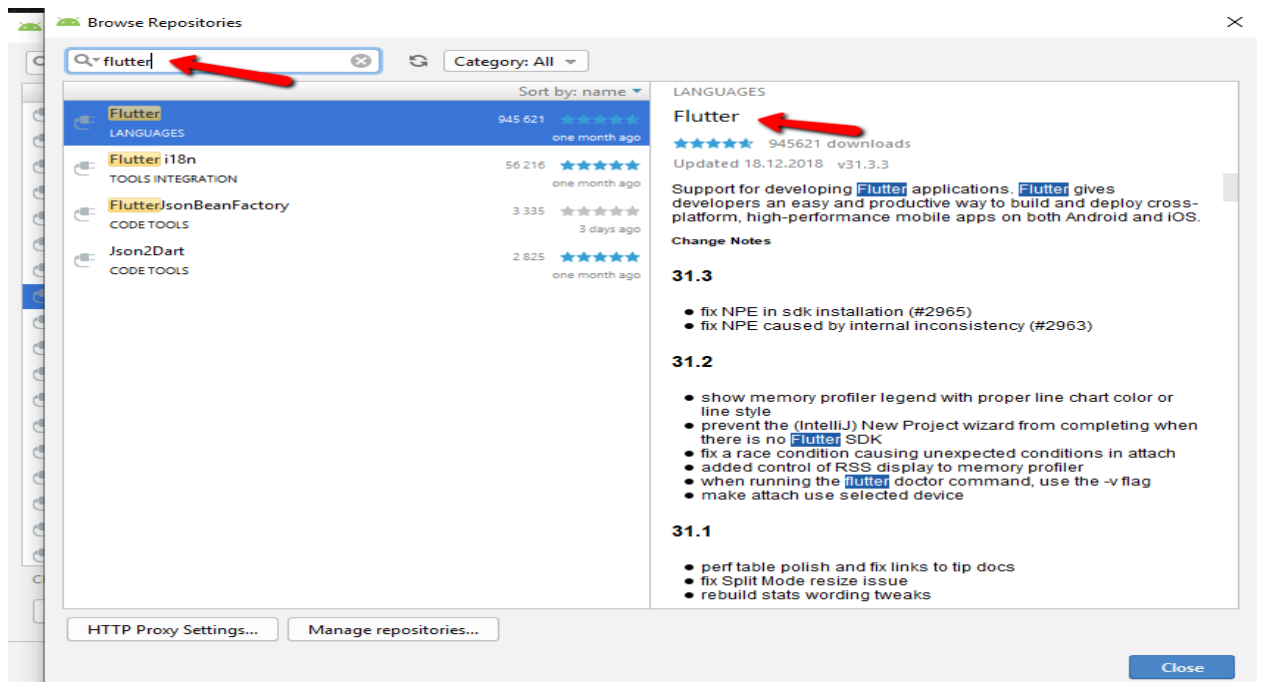
Скачиваем последнюю версию Java SE Development Kit 8 для своей операционной системы (потребуется для Android SDK), устанавливаем на свой компьютер, следуя за мастером установки, и создаём системную переменную среды `JAVA_HOME` с указанием пути к JDK, например: `C:\Program Files\Java\jdk1.8.0_201`.



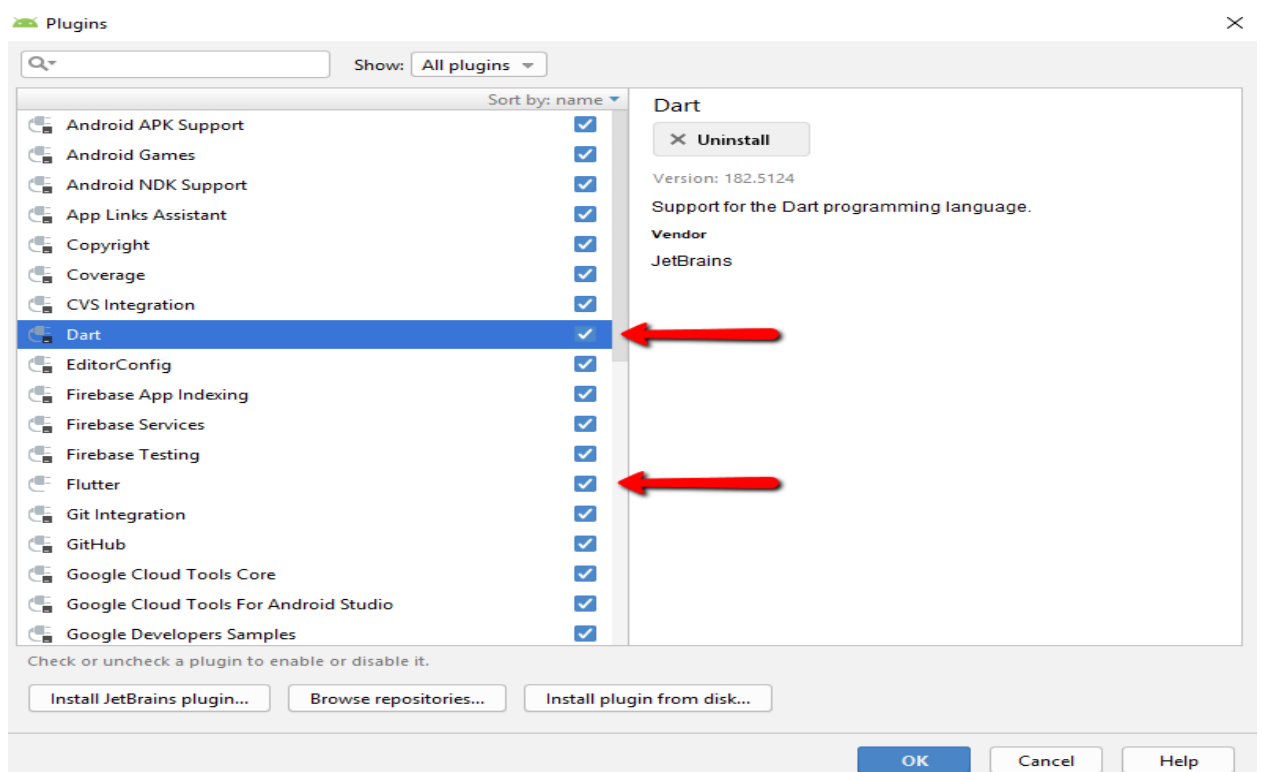
В открывшемся окне внизу нажимаем кнопку *Browse repositories...*



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:

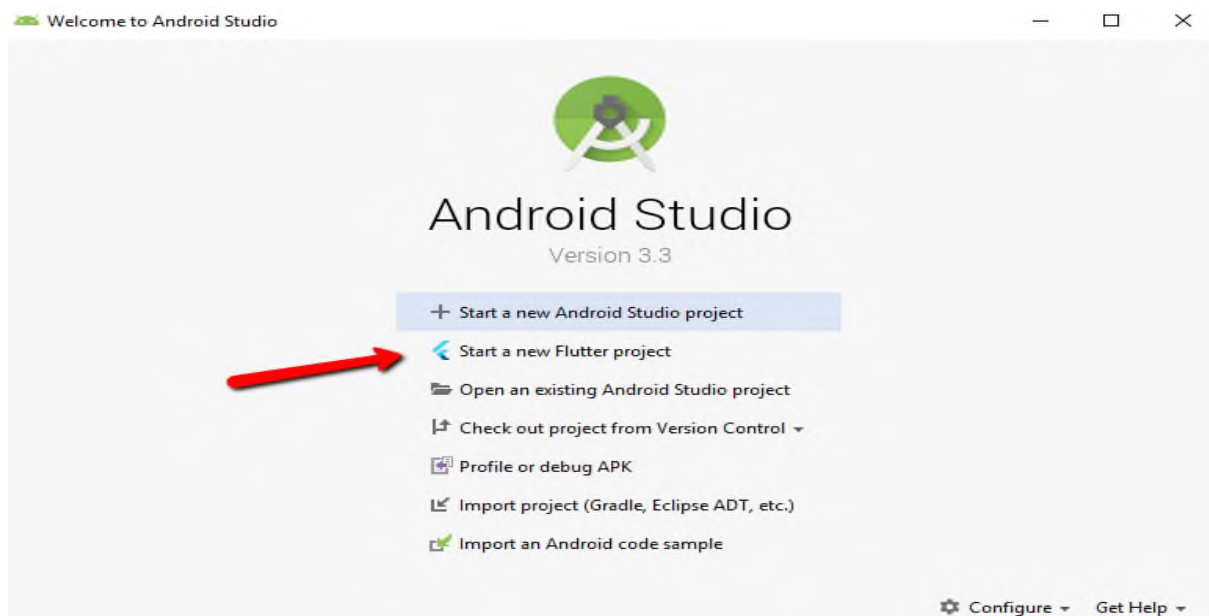
flutter doctor

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

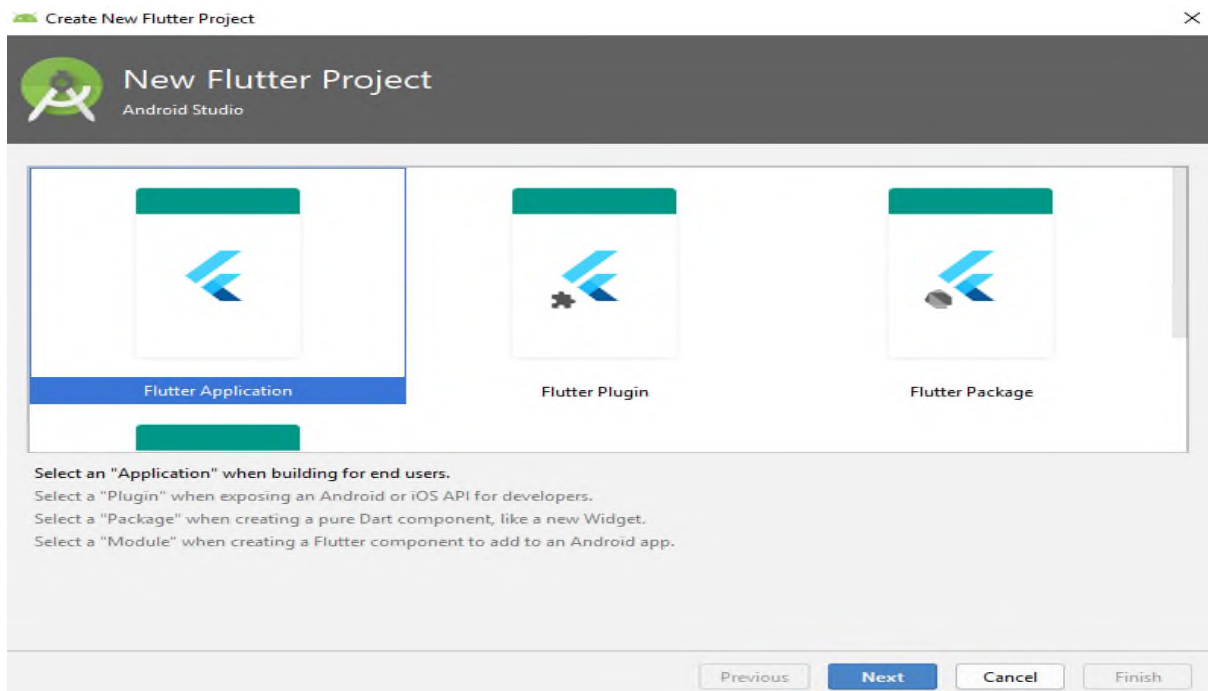
```
Doctor summary (to see all details, run flutter doctor -v):  
[✓] Flutter (Channel stable, v1.0.0, on Microsoft Windows [Version 10.0.17763.253], locale ru-RU)  
[✓] Android toolchain - develop for Android devices (Android SDK 28.0.3)  
[✓] Android Studio (version 3.3)
```

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).

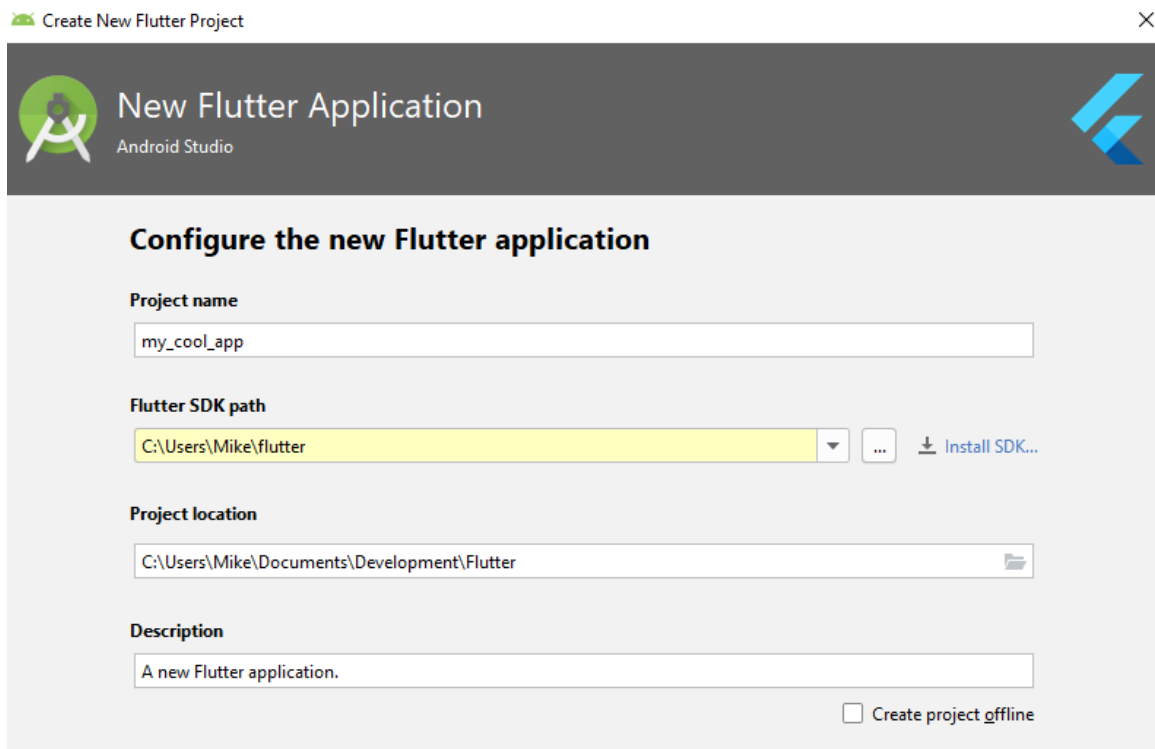
Вот мы и готовы создать Flutter проект в Android Studio. После установки плагинов Flutter и Dart в начальном экране Android Studio должна появиться опция *Start a new Flutter project*. Выбираем её:



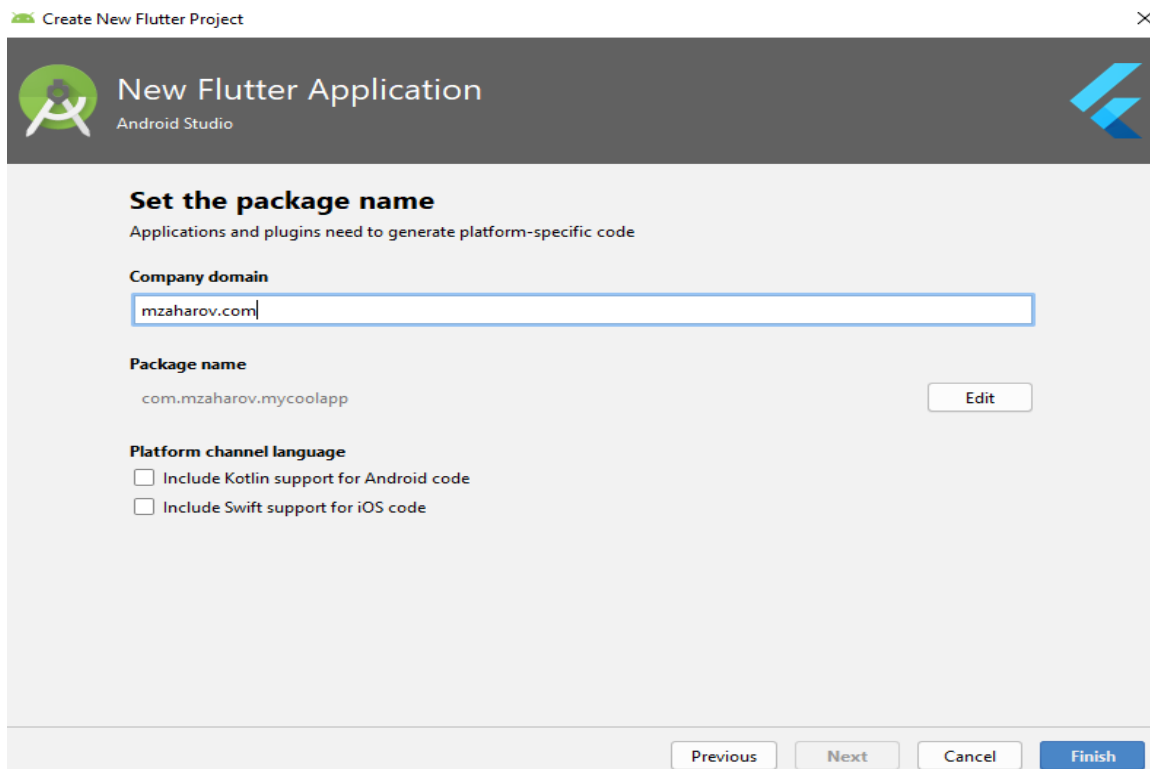
Далее соглашаемся с выбранной по умолчанию опцией *Flutter Application* и нажимаем кнопку *Next*:



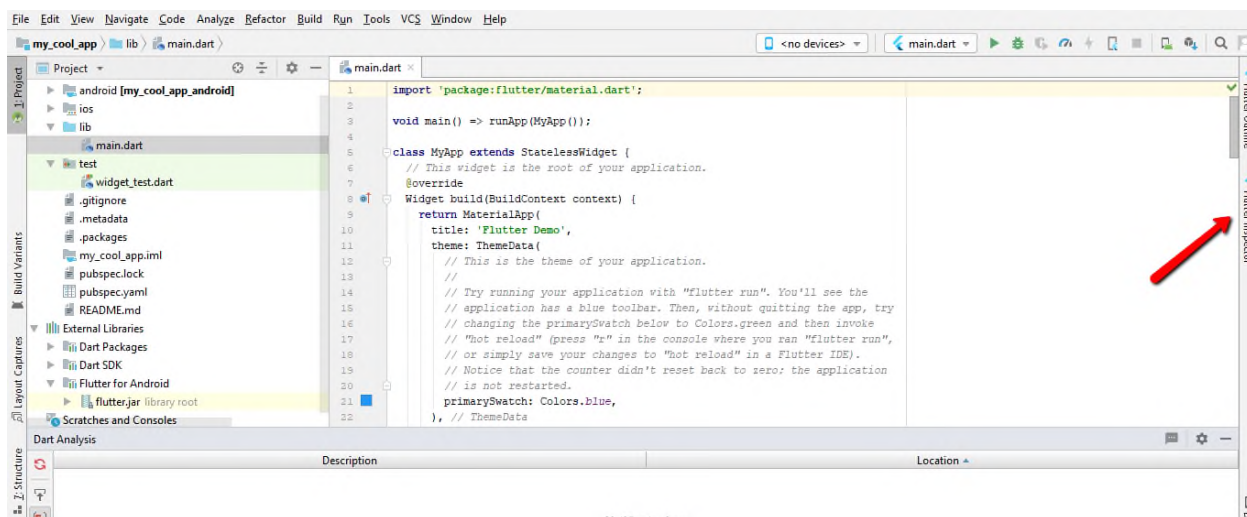
Указываем название проекта, путь к папке Flutter SDK, путь к папке проекта, даём краткое описание проекта (опционально), и вновь нажимаем кнопку *Next*:



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*.



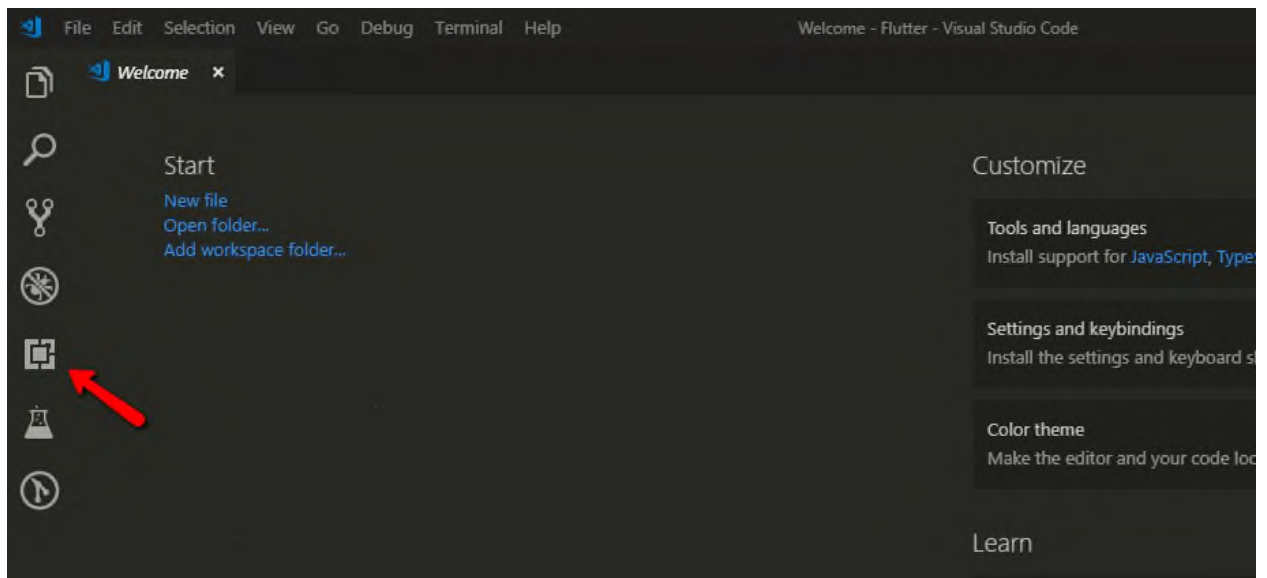
В зависимости от производительности компьютера, ждём несколько минут пока проект будет создан... Готово! Он должен выглядеть примерно так:



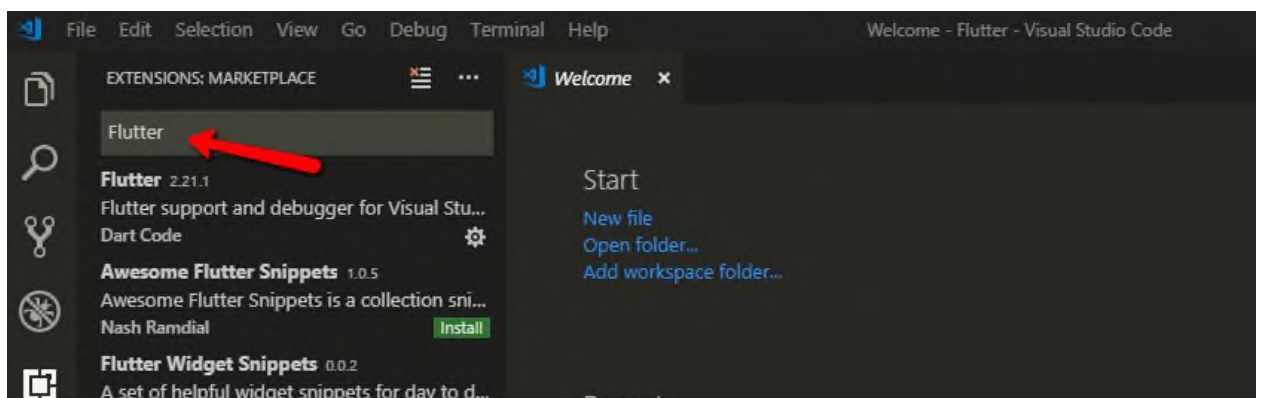
Обратите внимание на стрелку, указывающую на вкладку *Flutter Inspector*. В этом инспекторе имеется функционал, позволяющий делать ряд очень полезных во время разработки вещей, в т.ч. просмотр приложения на девайсе Android в режиме представления на iOS!

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

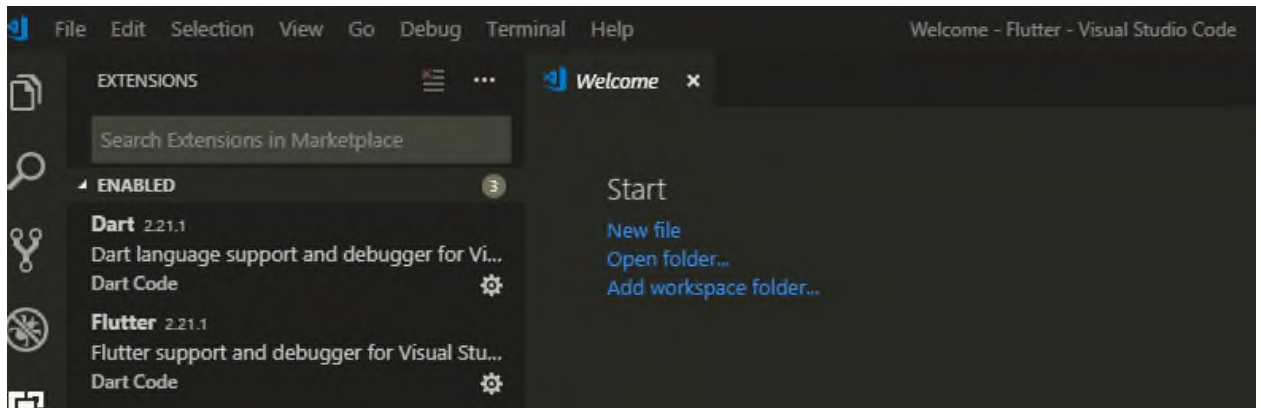
Скачиваем последнюю версию Visual Studio Code для своей операционной системы, устанавливаем на свой компьютер, следуя за мастером установки, и запускаем VS Code. Затем на боковой панели нажимаем на кнопку *Extensions* (показана стрелкой) или на клавиатуре — *Ctrl+Shift+X*:



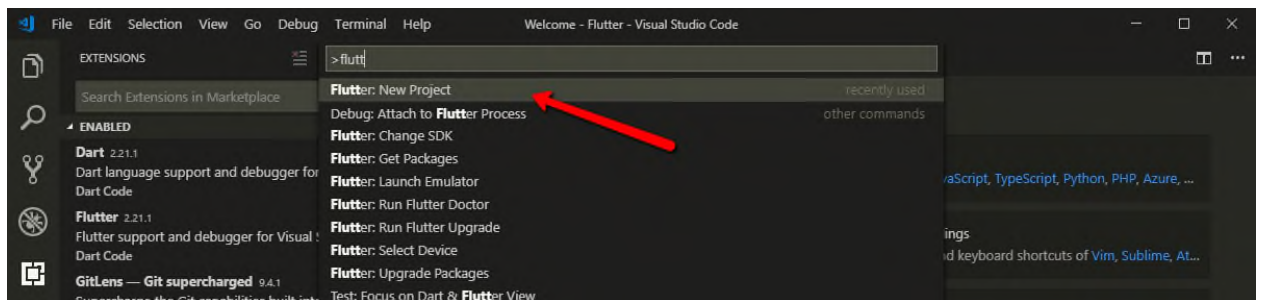
С помощью поиска ищем расширение Flutter.



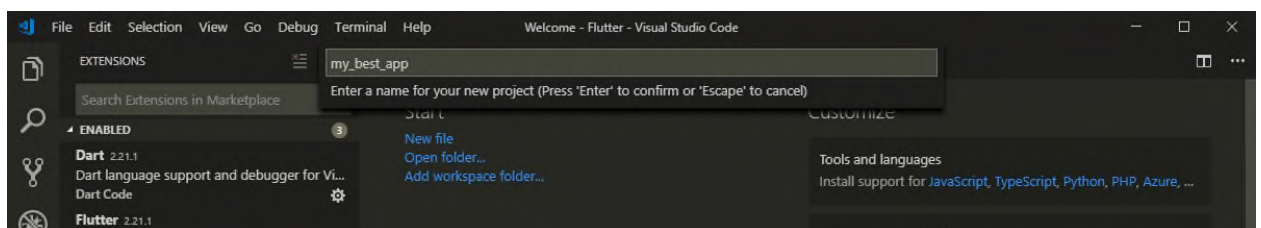
VS Code, как и в случае с Android Studio, предложит установить необходимое дополнительное расширение Dart. Устанавливаем и его. В итоге должны иметь два (или более) активированных расширения:



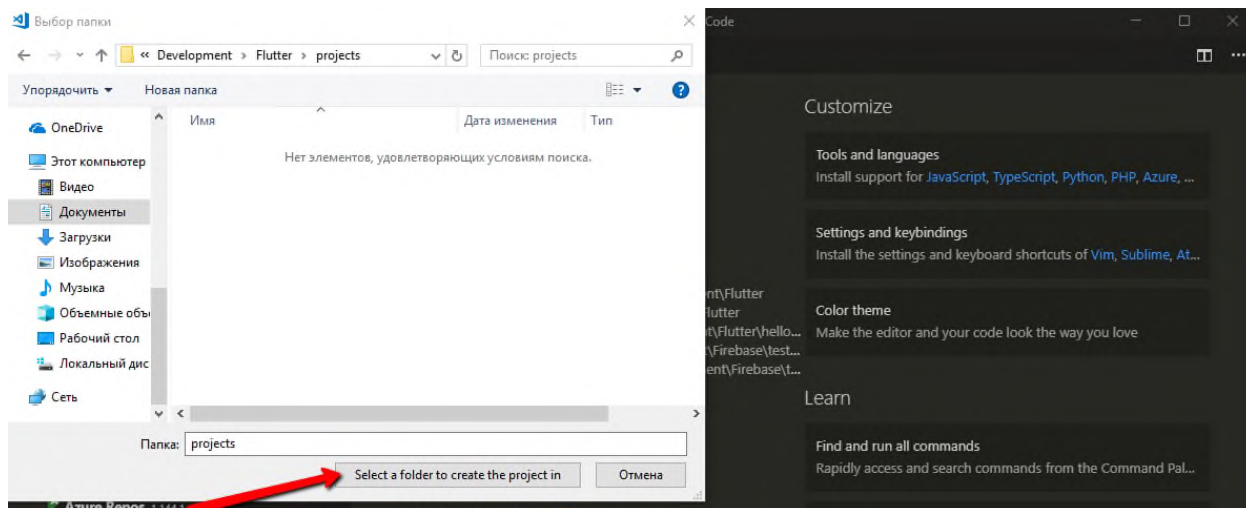
А теперь создаём Flutter проект. Нажимаем на значок шестерёнки в левом нижнем углу, и выбираем *Command Palette...* (или на клавиатуре — *Ctrl+Shift+P*). В командной строке Command Palette начинаем печатать *flutter*, и из появившегося списка выбираем *Flutter: New Project*:



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*:

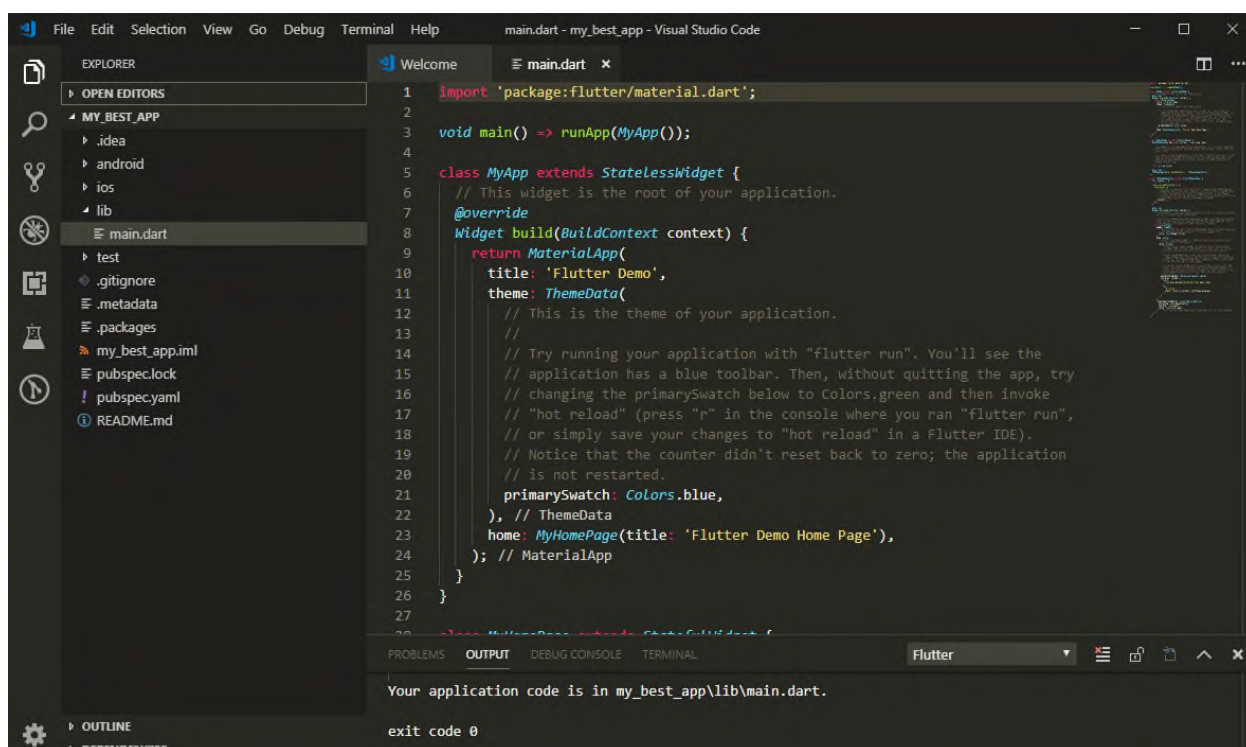


Минута

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.

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 лабораторная работа

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

Цель работы: Изучение размещения готовых приложений в интернет магазины.

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 обfuscацию 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 обfuscации 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 обусфикация 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 лабораторная работа

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

Цель работы: изучение настройки безопасности мобильных приложений

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 приватных 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". Требуется 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.