Flutter

Опыт в продакшн-разработке

О себе

- Меня зовут **Артем Зайцев**
- Flutter TeamLead at Surf
- Co-ведущий Flutter Dev Подкаст
- Любитель всего нового


MARS
ДОДО ПИЦЦА
Лабиринт
Яндекс
KFC
Delivery Club
UniCredit
go lama
БАНК ЗЕНИТ
Forbes
О'КЕЙ
ЛитРес:
один клик до книг
Газпромнефть
Моё дело
Интернет-бухгалтерия
Промсвязьбанк
lamoda

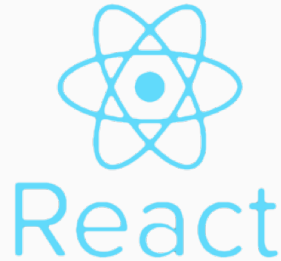
Мультиплатформенный UI-фреймворк от Google.

Нативно и православно

iOS SDK и Android SDK самым оптимальным образом взаимодействуют с платформой, потому что код приложения компилируется в нативный код платформы - этим всё сказано. Но вот какое-либо переиспользование кодовой базы между платформами становится невозможно.

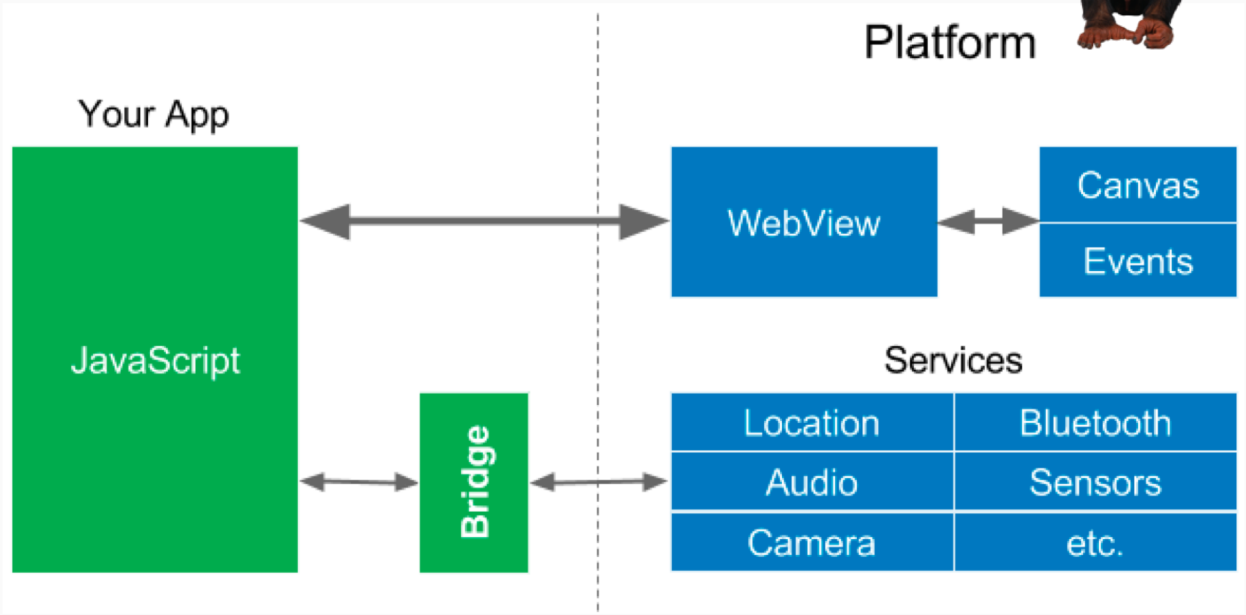
Кросс-платформа? Тссс...

“Кросс-платформа” - это изощрённое ругательство в кругах мобильной разработки.



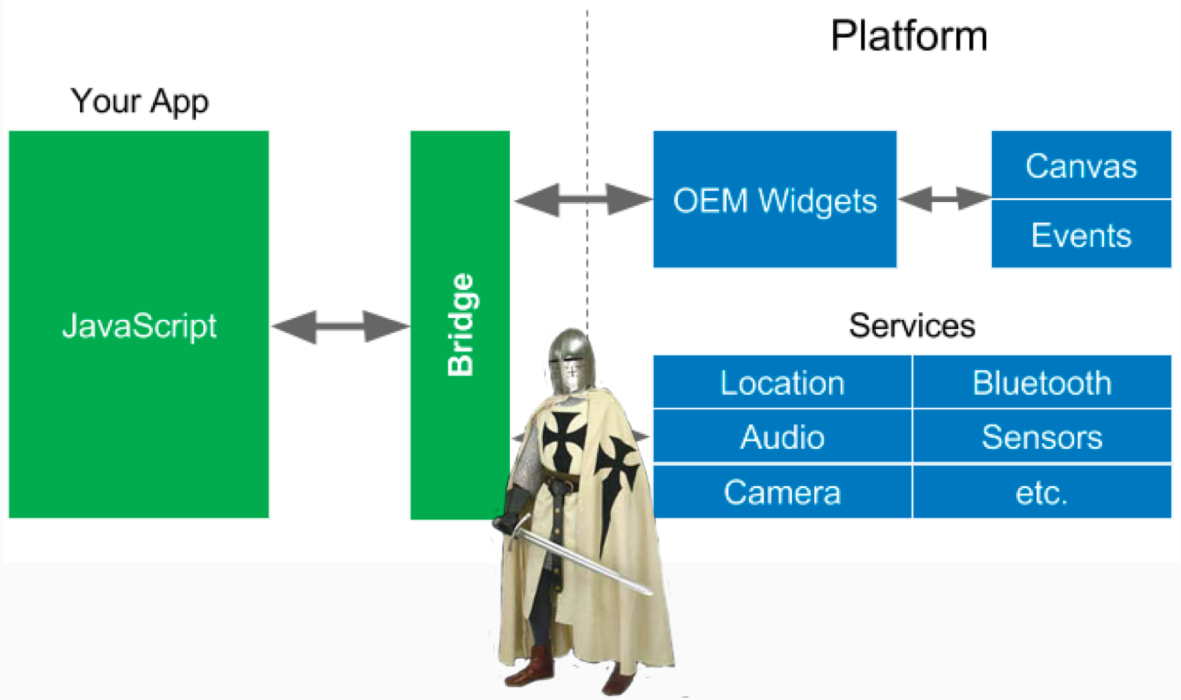
Неудачные решения

Titanium, Cordova, Ionic, PhoneGap и т.д. - веб-приложения, маскирующиеся под нативные. Чистый JavaScript, вся отрисовка UI происходит на WebView, для обращения к платформенным APIs необходимо ходить по JSBridge.


Platform
Your App
JavaScript
Bridge
WebView
Canvas
Events
Services
Location
Bluetooth
Audio
Sensors
Camera
etc.

React/Xamarin

- **ReactNative** – на смену WebView приходят нативные виджеты, но **JSBridge** не только не отходит в сторону, теперь он применяется при рендеринге каждого фрейма. Непозволительная роскошь. *Всё тот же чистый JavaScript*.
- **Xamarin** – тот же зверь, только писать надо на C#, а вся работа с платформой только через свои **обёртки**, которые не успевают обновляются вслед за обновлениями операционной системы.

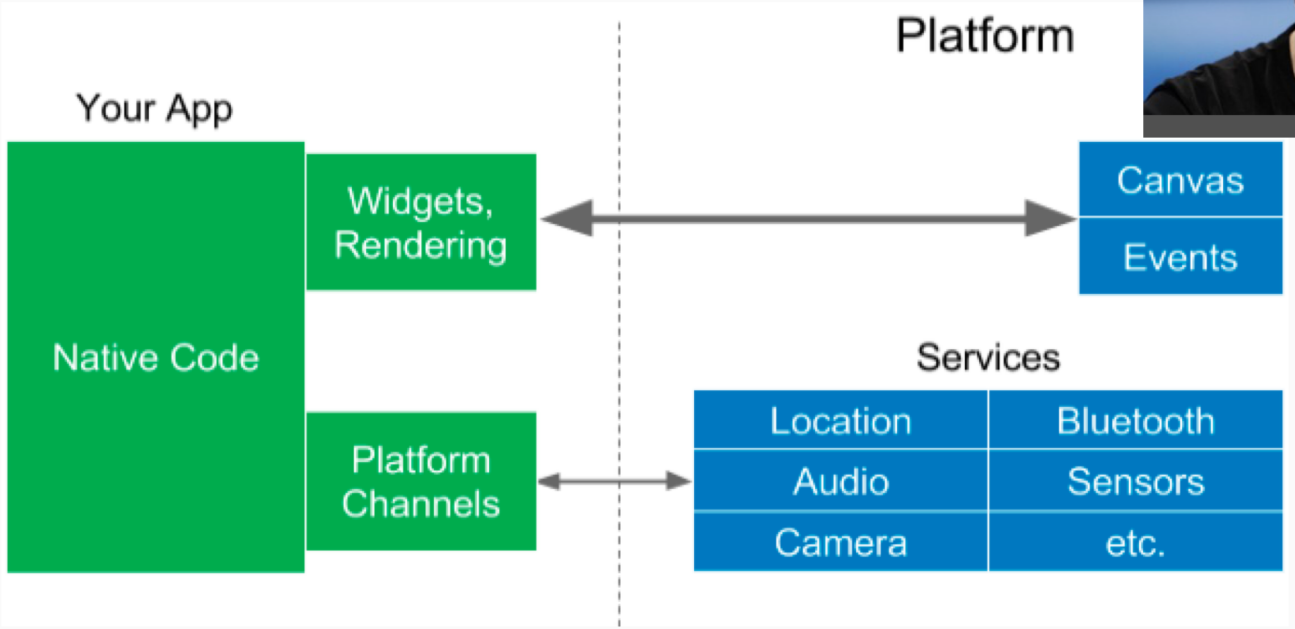

Platform
Your App
JavaScript
Bridge
OEM Widgets
Canvas
Events
Services
Location
Bluetooth
Audio
Sensors
Camera
etc.

Flutter

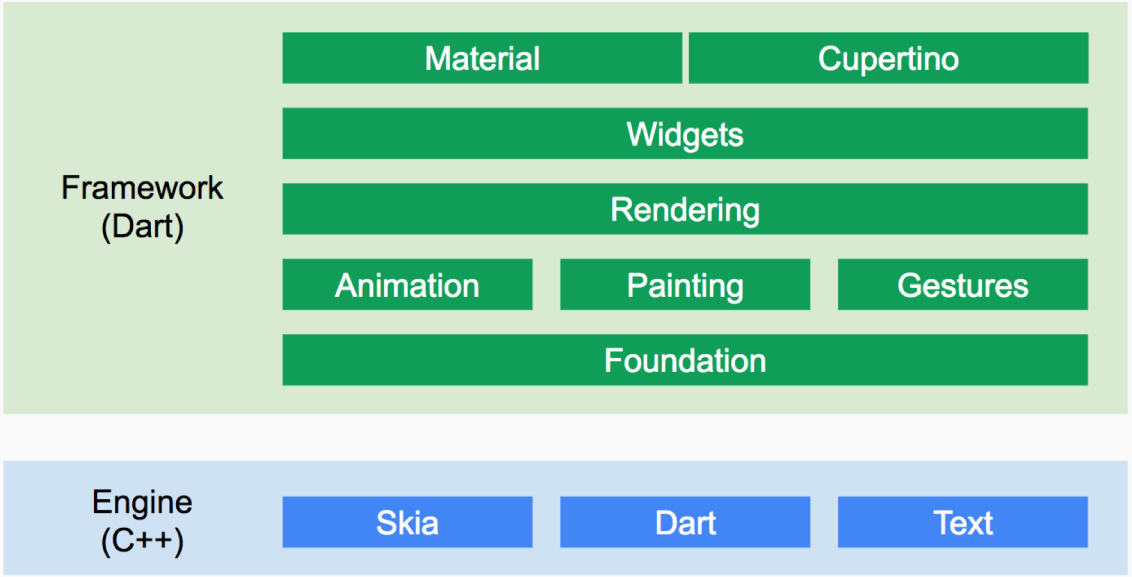
Flutter

- Команда разработки **Flutter** подумала и решила: можно же взять Dart и платформозависимые либы и скомпилировать их в **нативный** код.
- К тому же, можно не использовать платформенные виджеты, а использовать движок(**Skia**), который будет рендерить все сам внутри приложения.


not bad
meme-arsenal.ru


Platform
Your App
Native Code
Widgets, Rendering
Platform Channels
Canvas
Events
Services
Location
Bluetooth
Audio
Sensors
Camera
etc.

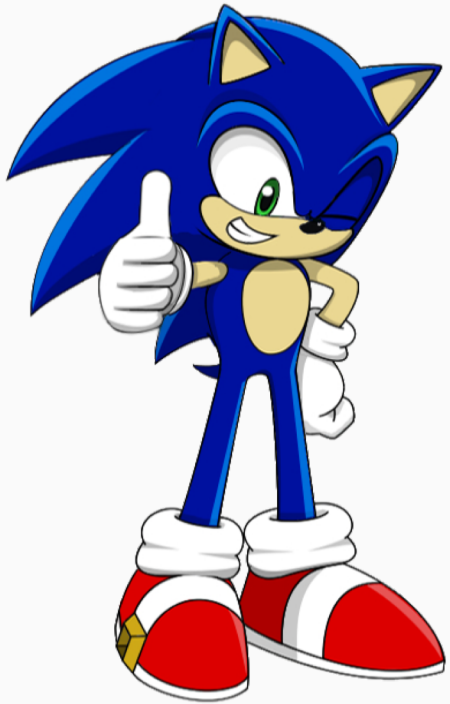
Flutter под капотом



Позволяет делать красивые приложения быстро.

Очень быстро.

Flutter



Преимущества Flutter

- **60**-120 fps
- Отличный **туллинг** для быстрой разработки
- **Активное** коммьюнити
- привычный **ООП** язык
- **Низкий** порог вхождения

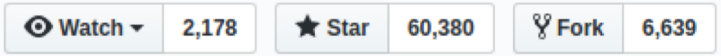
Dart

- Разработан в Google
- Уже используется в веб-разработке
- Имеет очень быстрый GC

Dart

Немного цифр

- Более 60k звезд на github
- Более сотни активных pull-request
- Более 13k готовых и активно развивающихся библиотек



Влияние Flutter

на наши процессы

Основные роли в разработке приложения

Посмотрим на основные роли:

- Заказчик
- Менеджер проекта
- Дизайнер
- Тестеровщик
- Разработчики(2 лида + 2 их команды)

Хочу круто, дешево и быстро

Для заказчика часто важно наличие фич, сроки и цена.

Нативная разработка:

- 1 платформа - x
- 2 платформы - 2x
- Flutter - x(но продаем за 1,5x)

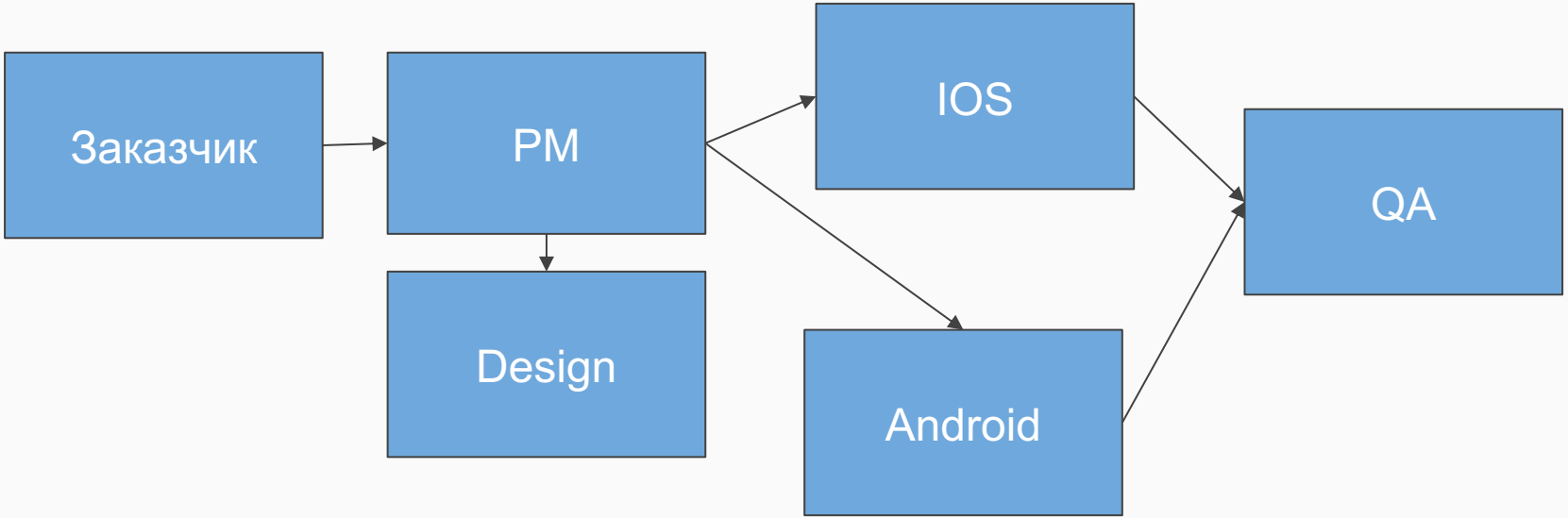
Как мне за ними следить?!

Менеджер проекта - человека который должен синхронизировать работу остальных команд, следить за сроками, заниматься бумажной работой и тд...

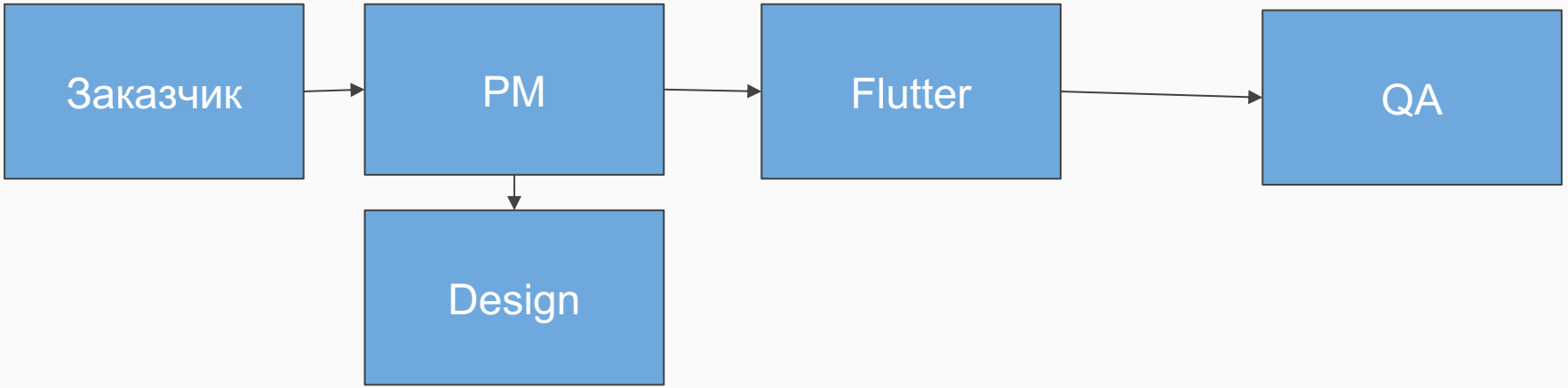
В случае 2-х платформ у него команда 2х

В случае Flutter - на 50% меньше.

Взаимодействие при нативной разработке



Взаимодействие при Flutter разработке



А что по дизайну?

Две ситуации:

- приложение имеет один дизайн для обеих платформ
- приложение имеет полностью разный дизайн

С этой точки зрения Flutter не привносит никаких нововведений.

Тестеровщик(QA)

К тестеру Владу приходят два приложения: под Android и под iOS.

Влад не знает, что используется Flutter: тестирует оба, заводит баги под оба.

Влад знает об используемой технологии: баги по логике - тестирует один раз, верстку проверяет везде. Так же внимание уделяется и платформозависимым фичам.

Замечание

Но не все привычные инструменты продолжают работать:

- **Charles** - сниффер запросов, не ловит их
- **UIAutomator** - видит экран, но не правильно определяет сами элементы.
- ...

Команда(-ы) разработки

Some month ago...

- **Два лида** - размытие ответственности за взаимодействие с тех командой заказчика
- Два лида - **больше** коммуникации, больше обсуждений, дольше процесс принятия решения, если проблема охватывает обе платформы.
- Одинаковые баги правятся **разными** людьми на **разных** языках
- Логика может несколько **отличаться**.

… and now

- Лид Flutter команды - действительно **лид технической команды**
- Взаимодействие **упрощается** до неузнаваемости
- Процессы становятся **прозрачнее**
- Время принятия решений - **меньше**

Порхай, как бабочка

Выход из зоны комфорта

Будучи *закоренелыми* “нативщиками” мы:

- **плевались**, когда слышали слово “кроссплатформа”
- **привыкли**, что есть проверенные библиотеки и инструменты
- **знали,** что лучше использовать и когда
- **опирались** на лучшие практики от авторитетов разработки

Дух авантюризма

Когда ты становишься **Flutter-разработчиком**, ты ощущаешь себя первопроходцем. Почти *Колумбом*, который ступает на неизведанные земли.

Тебе:

- **страшно**, ведь фреймворк может выстрелить в самый неподходящий момент
- **неуютно**, ведь проверенных практик просто нет

Flutter-разработчик

Когда ты **уже** Flutter-разработчик:

- ты **наслаждаешься** тем, как все оказывается просто
- ты при этом пишешь **много всего заново**
- ты **плюешься** иногда от языка , на котором пишешь...

Но ты все равно “**порхаешь**”.

Наш опыт

Немного сухих цифр

- экран приложения на Android в среднем делался за **1 рабочий день**
- такой же экран с использованием Flutter выходит **примерно также**…
- ...но при этом всем мы *параллельно создавали свою архитектуру*.

Декларативный подход - это очень круто

Писать экраны и в принципе UI в декларативном стиле - лучшая практика на данный момент. Да, это очень непривычно, немного меняется само сознание.

```
// Imperative style
b.setColor(red)
b.clearChildren()
ViewC c3 = new ViewC(...)
b.add(c3)
```

```
// Declarative style
return ViewB(
  color: red,
  child: ViewC(...),
)
```

Верстка в коде - это даже проще

Первый раз, когда вместо привычного xml(XAML, дизайнера интерфейсов), необходимо написать кнопку на том же Dart , что и весь основной код - это выглядит слегка странно.

После, ты понимаешь, что из-за единого языка - код становится более читаемым. Нет необходимости напрягаться и переключать мозг с восприятия одной кодировки на другую.

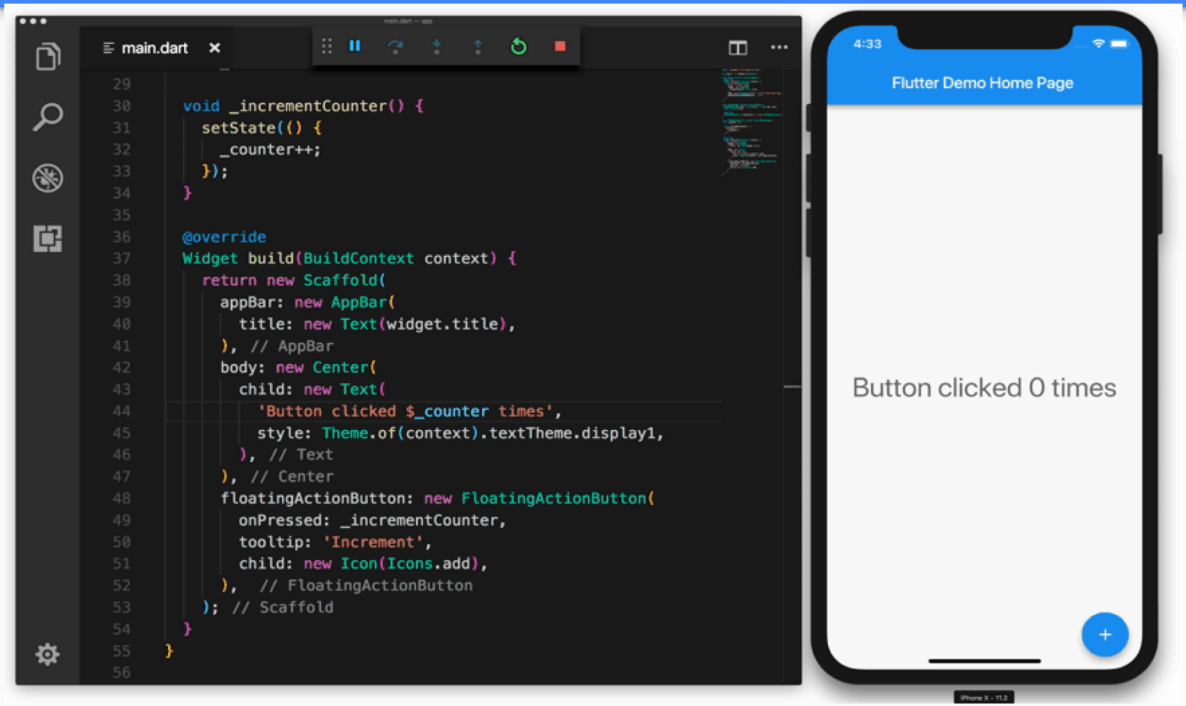
Dart - ну… такое

На данный момент мне, как для человека, который перешел с Kotlin не хватает:

- сахара
- extension методов
- data классов
- и много чего другого.

Но при этом он все равно выполняет необходимые функции и круто развивается.

Hot Reload - как вообще жить без него?!



Мы думали, что без помощи “нативщиков” не обойдется

За полгода разработки приложений, мы прибегли к помощи IOs разработки, только, чтобы он нам рассказал как работать с сертификатами и подписью.

Все остальное было покрыто плагинами. Пока.

Но при этом, необходимо понимать следущее...

Flutter - это не замена “**нативки**”

Это крайне важно понимать.

Не в сказку попали

Выстрелить в ногу может сам фреймврок

- баг во фреймворке - жди исправления
- внезапное обновление - может что-то поломаться, если не готов.

Никому не доверяй

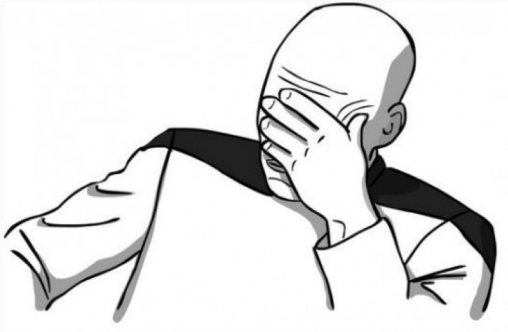
Количество библиотек на Flutter уже начинает зашкаливать. При этом действительно качественные найти крайне сложно.

Да и даже вроде бы качественным и вроде от проверенных людей - не всегда стоит доверять.

Нет нормальной системы сборки

На данный момент Flutter имеет команды для сборки с небольшим набором флагов.

При этом некоторая часть функционала либо слегка забагована, либо может попросту не работать, либо не покрывает необходимые кейсы.



Как мы искали CI/CD

Jenkins Pipelines

У нас были готовые пайплайны для сборок IOS и Android приложений.

Стояла задача с помощью них собирать Flutter.

Погружение в shell

Мы составили набор билд скриптов для сборок. Мы использовали возможность указания main-файла для разделения тестовых и релизных окружений.

build.sh -qa

Почему два apk?

Был такой момент: Flutter не умел прокидывать ndk библиотеку с приложением для определенных архитектур процессоров, если не указать это явно.

Из-за этого приходилось два раза запускать билд скрипт и немного **костылить**. Для пользователей Google Play это все было совершенно незаметно.

В одном из обновлений это поправили, добавили одну команду для сразу двух apk

Но беда пришла откуда не ждали

Мы используем **Beta by Fabric** для доставки артефактов.

Выяснилось, что на Android плагин gradle этой библиотеки имеет весьма неприятный баг. И не умеет в **split-apk**.

Разделение на .dev и .prod

Но самое больное - это **flavors**.

В продакшн разработке необходимо разграничивать версии приложения для разработчиков и версию для пользователей. Самый простой вариант: версия для разработчиков имеет постфикс .dev в id.

Кроме этого, flavors предоставляют возможность иметь разные файлы ресурсов, конфигов и тд для разных flavor.

Так что, пора?

Да

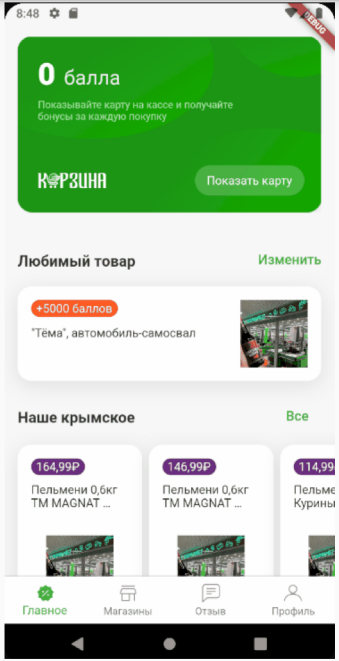
Мы доверяем Flutter

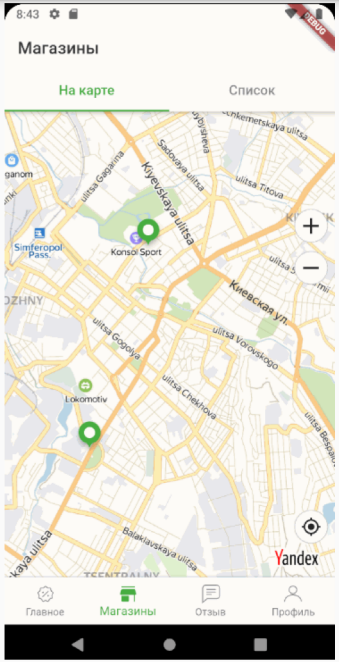
Главное не боятся современных и новых решений.

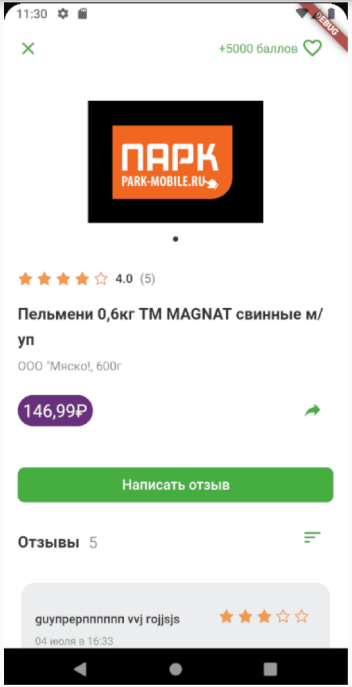
Когда-то давно сами нативные SDK были мало кому известны и все их боялись. Но при том люди, которые тогда не побоялись являются авторитетами в индустрии. Мы знаем их библиотеки и доверяем их мнению.

Сейчас время становится такими людьми во Flutter. Быть может именно Вы через десять лет будете богом этой платформы.

Пример проекта на Flutter







Q&A