I BUILT THE EXACT SAME APP IN **KOTLIN**, **NATIVESCRIPT** AND **FLUTTER**

WHAT DID I LEARN?

SO WHAT APP IS THIS ?

- “realworld.io" - a project by thinkster (github.com/gothinkster)
- A simplified clone of Medium
- Standardised API
- We can make backend or frontend using any technology as per the API

conduit

A place to share your knowledge.

Global Feed

acblk
February 15, 2019

0

new article

some text

Read more...

xuqi
February 15, 2019

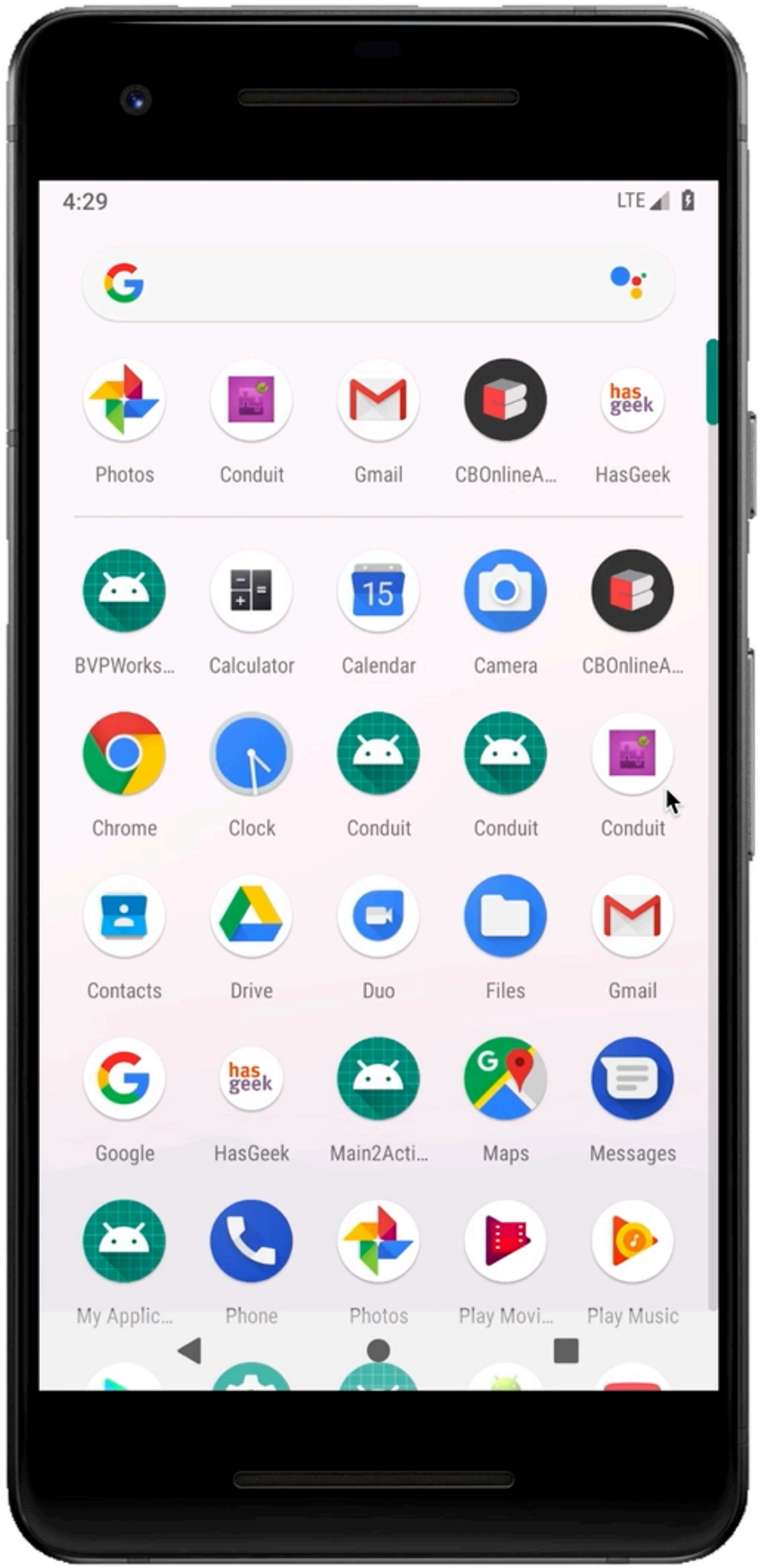
0

dfa

fsdfa

Popular Tags

dragons test training tags as coffee money flowers caramel sugar happiness japan sushi cars cookies animation well baby clean


4:29
LTE
Photos
Conduit
Gmail
CBOnlineA...
HasGeek
BVPWorks...
Calculator
Calendar
Camera
CBOnlineA...
Chrome
Clock
Conduit
Conduit
Conduit
Contacts
Drive
Duo
Files
Gmail
Google
HasGeek
Main2Acti...
Maps
Messages
My Applic...
Phone
Photos
Play Movi...
Play Music

DEVELOPER EXPERIENCE

I WILL TRADE A FEW CPU CYCLES FOR DEVELOPER CYCLES ANY DAY

DHH

(Basecamp & Ruby on Rails founder)

Available IDEs

Available IDEs

Available IDEs

Android Studio

IntelliJ IDEA

Flutter

WebStorm

VS Code

THIS **SHOULD** BE EXTREMELY EASY

GETTING STARTED

- Android Studio installation takes care of all required dependencies
- Create new project in GUI
- Created project works out of the box
- Most problems in starting a project have multiple StackOverflow answers

- Installing Flutter SDK has command-line steps
- **flutter doctor** command to check installation success (not 100% reliable)
- GUI-based steps to create project in Android Studio
- Created project works out of the box

- Installing nativescript-cli has command-line steps
- **tns doctor** command to check installation success (not 100% reliable)
- CLI-based step to init starter template. Multiple starter templates - can be confusing
- Created project may not work out of the box. Circuitous dependency resolution in starter template too

LANGUAGES

MORE SIMILAR THAN YOU THINK

- Differences in implementations, practices, common patterns, internal mechanisms
- On the surface, modern multi-platform languages have very similar syntax and features

Swift

```
print("Hello, world!")
```

Kotlin

```
println("Hello, world!")
```

Dart

```
print('Hello, world!');
```

TypeScript

```
console.log("Hello, world!");
```

VARIABLES

Swift

```swift
var myVariable = 42
myVariable = 50
let myConstant = 42
```

Kotlin

```kotlin
var myVariable = 42
myVariable = 50
val myConstant = 42
```

Dart

```dart
var myVariable = 42;
myVariable = 50;
final myConstant = 42;
```

TypeScript

```typescript
let myVariable = 42;
myVariable = 50;
const myConstant = 42;
```

ARRAYS

Swift

```swift
var shoppingList = ["catfish", "water",
    "tulips", "blue paint"]
shoppingList[1] = "bottle of water"
```

Dart

```dart
final shoppingList = ["catfish", "water",
    "tulips", "blue paint"];
shoppingList[1] = "bottle of water";
```

Kotlin

```kotlin
val shoppingList = arrayOf("catfish", "water",
    "tulips", "blue paint")
shoppingList[1] = "bottle of water"
```

TypeScript

```typescript
let shoppingList = ["catfish", "water",
    "tulips", "blue paint"];
shoppingList[1] = "bottle of water";
```

MAPS

Swift

```swift
var occupations = [
    "Malcolm": "Captain",
    "Kaylee": "Mechanic",
]
occupations["Jayne"] = "Public Relations"
```

Dart

```dart
var occupations = {
    "Malcolm": "Captain",
    "Kaylee": "Mechanic",
};
occupations["Jayne"] = "Public Relations";
```

Kotlin

```kotlin
val occupations = mutableMapOf(
    "Malcolm" to "Captain",
    "Kaylee" to "Mechanic"
)
occupations["Jayne"] = "Public Relations"
```

TypeScript

```typescript
let occupations = {
    "Malcolm": "Captain",
    "Kaylee": "Mechanic",
};
occupations["Jayne"] = "Public Relations";
```

ARRAY - MAP - STREAM

Swift

```swift
let numbers = [20, 19, 7, 12]
numbers.map { 3 * $0 }
```

Kotlin

```kotlin
val numbers = listOf(20, 19, 7, 12)
numbers.map { 3 * it }
```

Dart

```dart
final numbers = [20, 19, 7, 12];
numbers.map((number) => 3 * number);
```

TypeScript

```typescript
let numbers = [20, 19, 7, 12];
numbers.map((it) => 3 * it);
```

CLASSES

Swift

```
class Shape {
    var numberOfSides = 0
    func simpleDescription() -> String {
        return "A shape with \(numberOfSides) sides."
    }
}
```

Kotlin

```
class Shape {
    var numberOfSides = 0
    fun simpleDescription() =
        "A shape with $numberOfSides sides."
}
```

TypeScript

```
class Shape {
    numberOfSides = 0;
    simpleDescription(){
        return `A shape with ${this.numberOfSides} sides.`;
    }
}
```

Dart

```
class Shape {
    var numberOfSides = 0;
    simpleDescription() =>
        "A shape with $numberOfSides sides.";
}
```

INTERFACES

Swift

```
protocol Nameable {
    func name() -> String
}

func f<T: Nameable>(x: T) {
    print("Name is " + x.name())
}
```

Dart

```
abstract class Nameable {
    String name();
}

f(Nameable x) {
    println("Name is " + x.name())
}
```

Kotlin

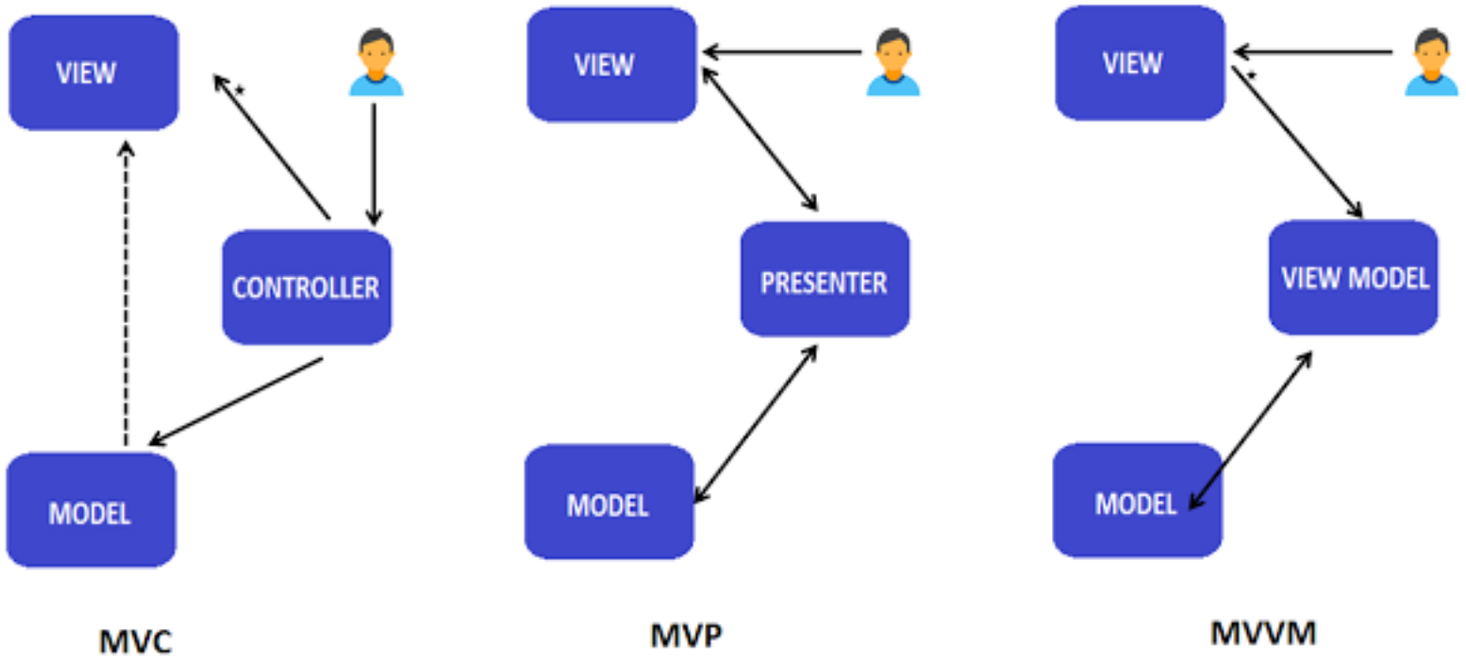
```
interface Nameable {
    fun name(): String
}

fun f<T: Nameable>(x: T) {
    println("Name is " + x.name())
}
```

TypeScript

```
interface Nameable {
    name():string;
};

function f(x: Nameable) {
    console.log("Name is " + x.name());
}
```



IT HELPS TO HAVE WELL-ESTABLISHED

DESIGN PATTERNS

KOTLIN (NATIVE ANDROID)

- All 3 - **MVP, MVC** and **MVVM** well defined and documented
- **MVC**: Conventional (and easy to get started)
- Modern *Android Architecture Components* - focus on **MVVM**
- **ReactiveX** available via *RxJava, RxAndroid* and *RxKotlin*

NATIVESCRIPT (WITH ANGULAR OR VUE)

- Angular - Component based (sort of MVC) but can by **Reactive** is using *RxJS*
- Vue - **MVVM**, can be **Reactive** if using *vuex-rxjs*
- **Flux**-like state management via *vuex* available in Vue

FLUTTER

- No officially documented/endorsed design pattern
- Unofficial MVC and MVVM-like examples available
- **B**usiness **Lo**gic **C**omponent (BLoC) pattern getting popular

GOOD TO HAVE

COMMON TOOLS

DRAG/DROP GUI EDITOR

android	**Yes** - fully featured
NativeScript	**No**
Flutter	**No** - partial support via 3rd party tools

BREAKPOINT DEBUGGING

Yes - fully integrated with Android Studio

No - very limited support, socket connection breaks

Yes - via Android studio but not 100% reliable

HOT RELOAD

Partial - currently opened activity refreshes. Also slow

Yes - Recently released, Webpack HMR based. Very buggy right now

Yes - rock solid

TESTING FRAMEWORKS

Well Supported - JUnit for unit test and Espresso for UI test

??? - Can use Vue/ Angular usual mechanisms, but not well documented

Well Supported - Separate unit and UI test methods not needed

DATA (DE)SERIALIZATION AT RUNTIME

Yes - Gson, Jackson, FasterXML. Also data class generators available

NativeScript

Yes - Not needed for JSON 😄, XML libraries exist

No - Not possible without reflection. Need data classes pre-generated.

2-WAY DATA BINDING

Kind of - Using LiveData and DataBinding

Yes - Vue supports proper 2-way binds with v-model

No - Not in usual sense. There is widget-binding

FULLY FEATURED ORM

Yes - multiple solutions for SQLite and also Realm for object-storage

Yes - TypeORM for SQLite

No - sqflite is *kind of* ORM but lot of hand-written SQL needed.

Not possible because no runtime reflection

INTEGRATING WITH NATIVE WIDGETS

Needed to embed Google Maps or WebViews

android

Ofcourse

NativeScript

Yes - Might need Kotlin/ Swift knowledge to glue together

Flutter

No - Preliminary support recently dropped, doesn't work reliably yet

LET US TAKE A

WALK THROUGH THE CODEBASE

cb.lk/rwappkt

KOTLIN CODEBASE WALKTHROUGH

cb.lk/rwappns

NATIVESCRIPT CODEBASE WALKTHROUGH

cb.lk/rwappfl

FLUTTER CODEBASE WALKTHROUGH

BENCHMARKS

Raw Numbers

App Size (debug)

3.6MB

42MB

App Size (prod)

3.1 MB

5.8 MB

Peak CPU Usage

36%

26%

Peak Memory Usage

127 MB

160 MB

Average Memory Usage

50 MB

85 MB

Lines of Code (excluding autogenerated files)

950

440

Build Time (cold/clean)

32s

50s

Build Time (warm/dirty)

Kandroid

8s

NativeScript

Flutter

8s

Time taken to develop (via wakatime.com stats)

5 hrs

3.5 hrs