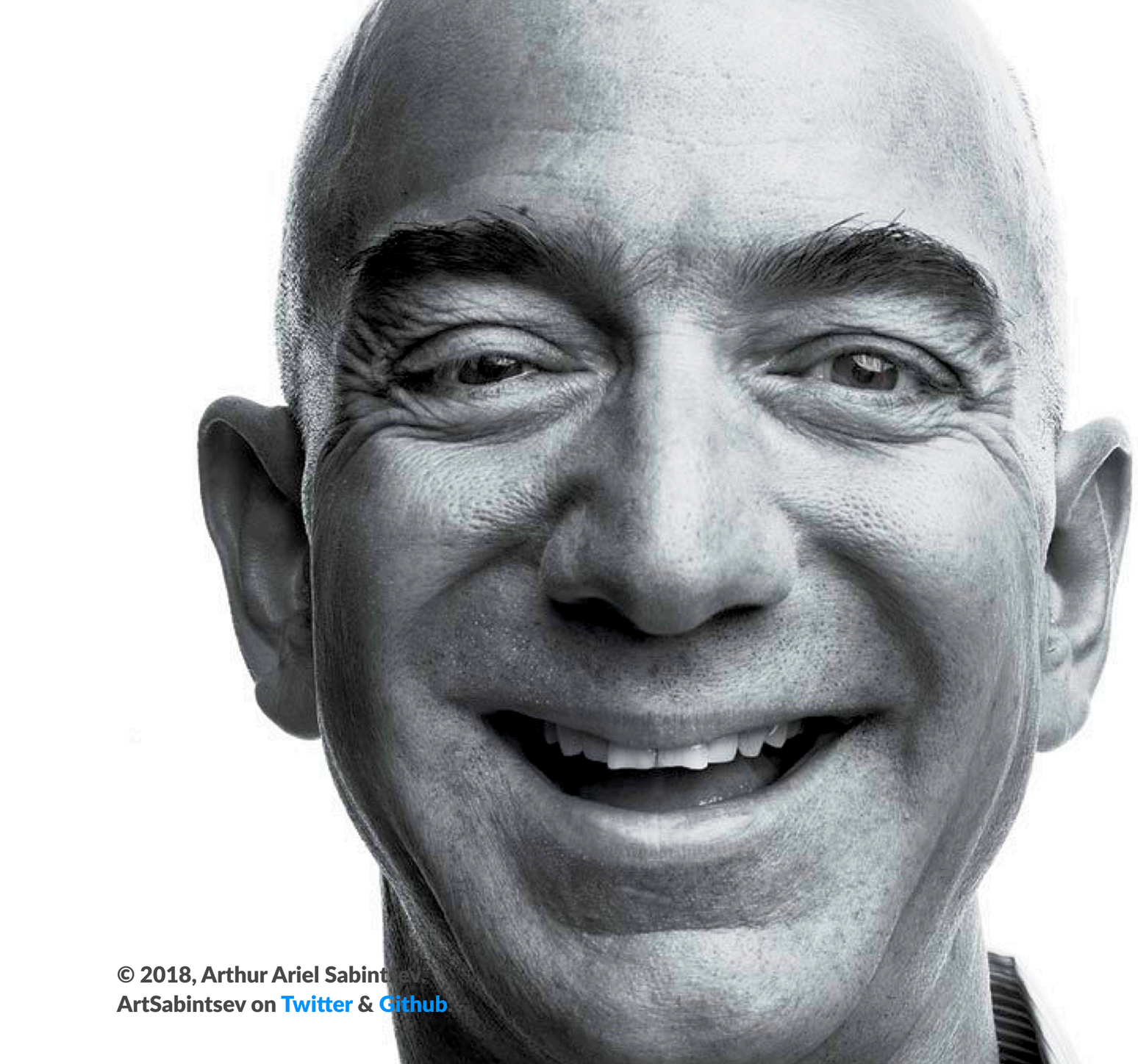
Scaling iOS Architecture

Arthur Sabintsev
Lead iOS Developer
The Washington Post

ArtSabintsev on Twitter & Github.



Jeff Bezos

CEO, Amazon
Founder, Blue Origin
Owner, The Washington Post

ArtSabintsev on Twitter & Github

Apps (pre-Bezos Acquisition)

Apps (post-Bezos Acquisition)

ArtSabintsev on Twitter & Github.

THE GLOBE AND MAIL

DAILY NEWS

The Salt Lake Tribune

Chicago Tribune

ADVANCE LOCAL

NZME. NEW ZEALAND MEDIA AND ENTERTAINMENT

Los Angeles Times

Le Parisien

infobae

The Boston Globe

THE BALTIMORE SUN
baltimoresun.com

The Philadelphia Inquirer

The Washington Post is now building and maintaining native mobile apps for other newspaper publishers.

Project Unicorn [1] [2]

(aka: The White Label App, Arc Mobile, Arc Native)

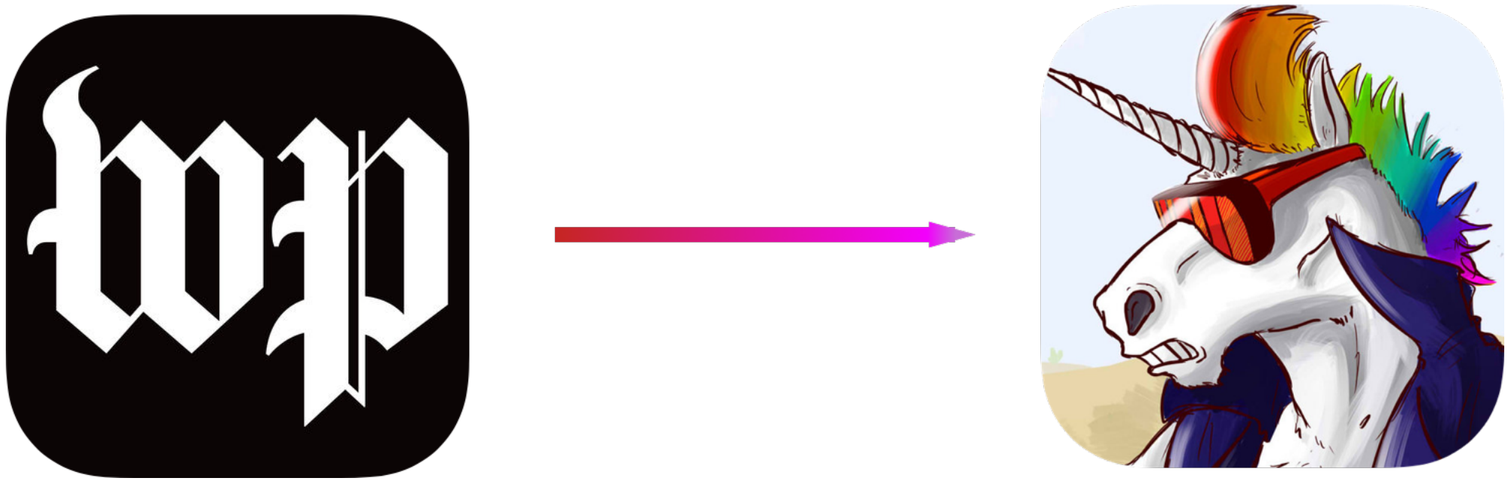
[1] Image Credit: Unicorn King by rubymight

[2] Unofficial (Internal) Logo

ArtSabintsev on Twitter & Github.

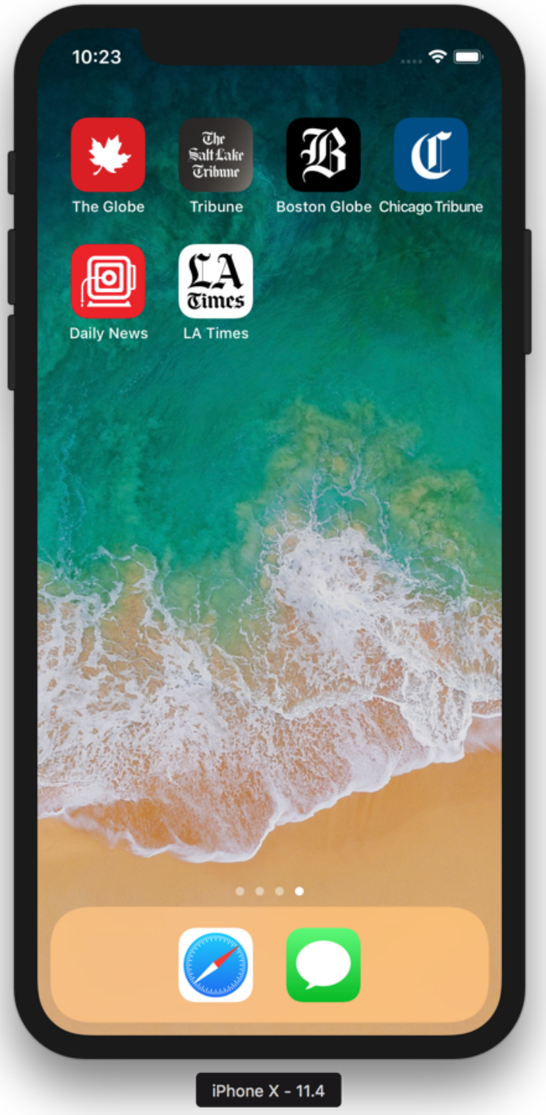
What is the Unicorn / Arc Mobile app?

Unicorn was pitched to be a clone of The Washington Post Classic app.



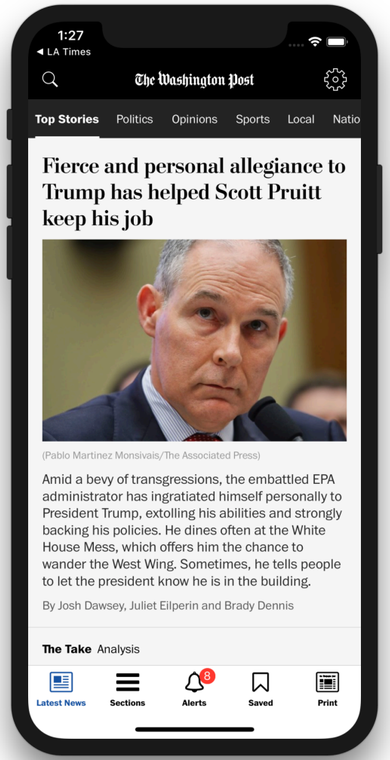
Unicorn

Powered

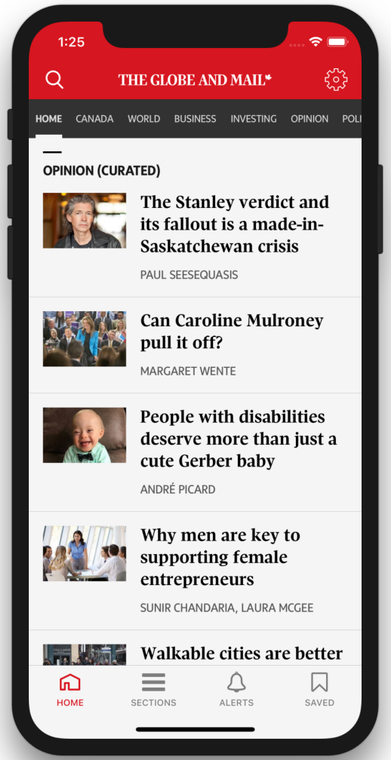


ArtSabintsev on Twitter & Github.

Unicorn-Powered Apps

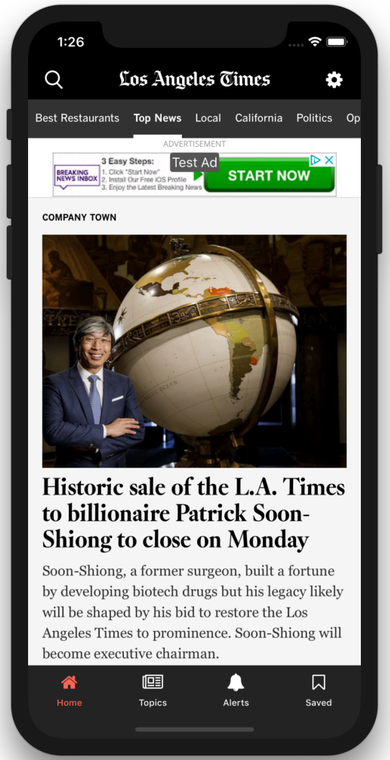


iPhone X - 11.4



iPhone X - 11.4

iPhone X - 11.4

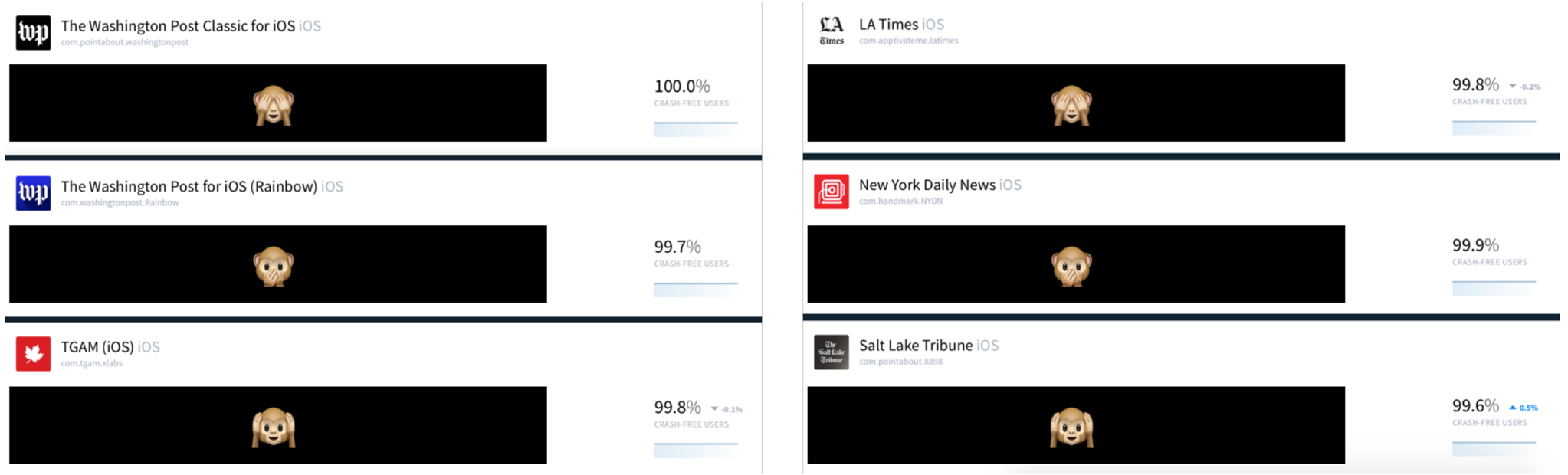


iPhone X - 11.4

We were not prepared!

Image: Qichao Wang on ArtStation

Though we've done a decent job



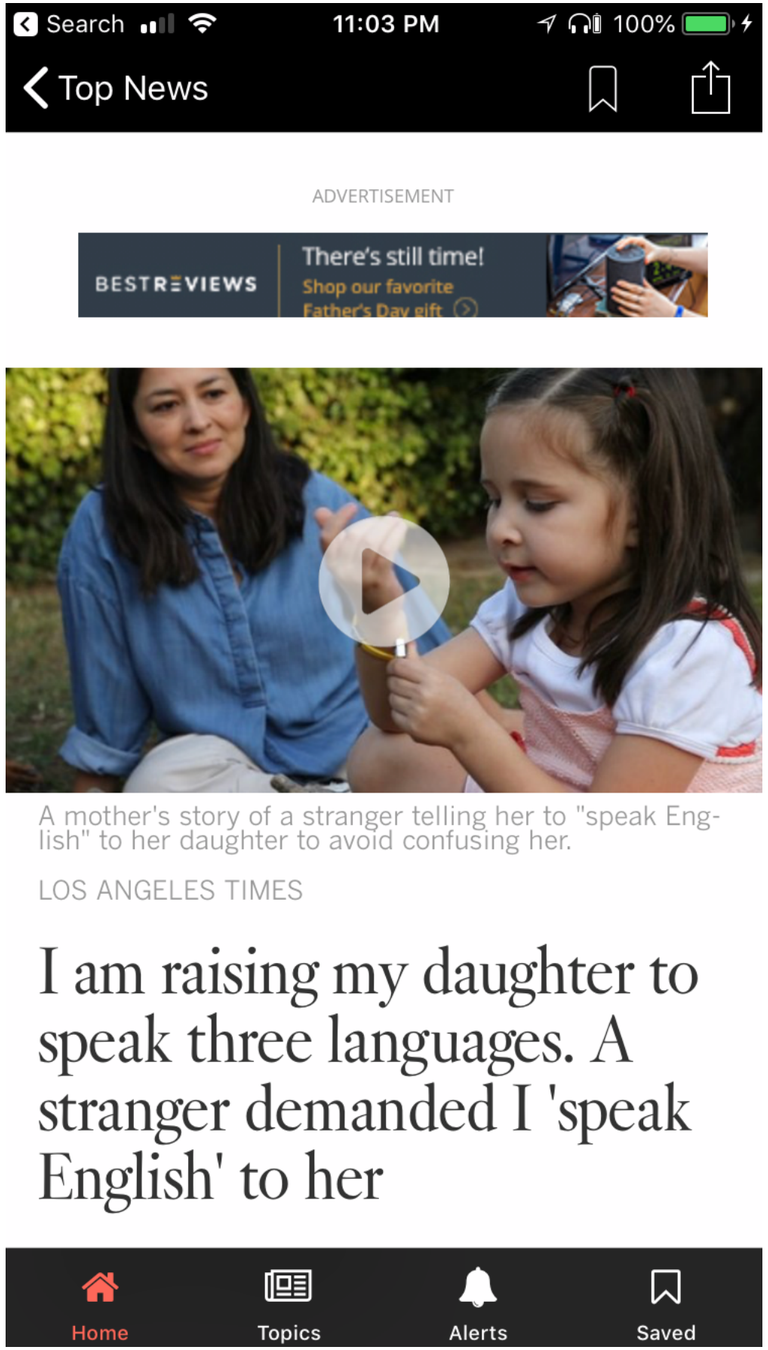
ArtSabintsev on Twitter & Github.

Structure of a News App

- Section Fronts
- Articles
- Section List
- Search
- Saved Articles
- Segmented Push Notifications

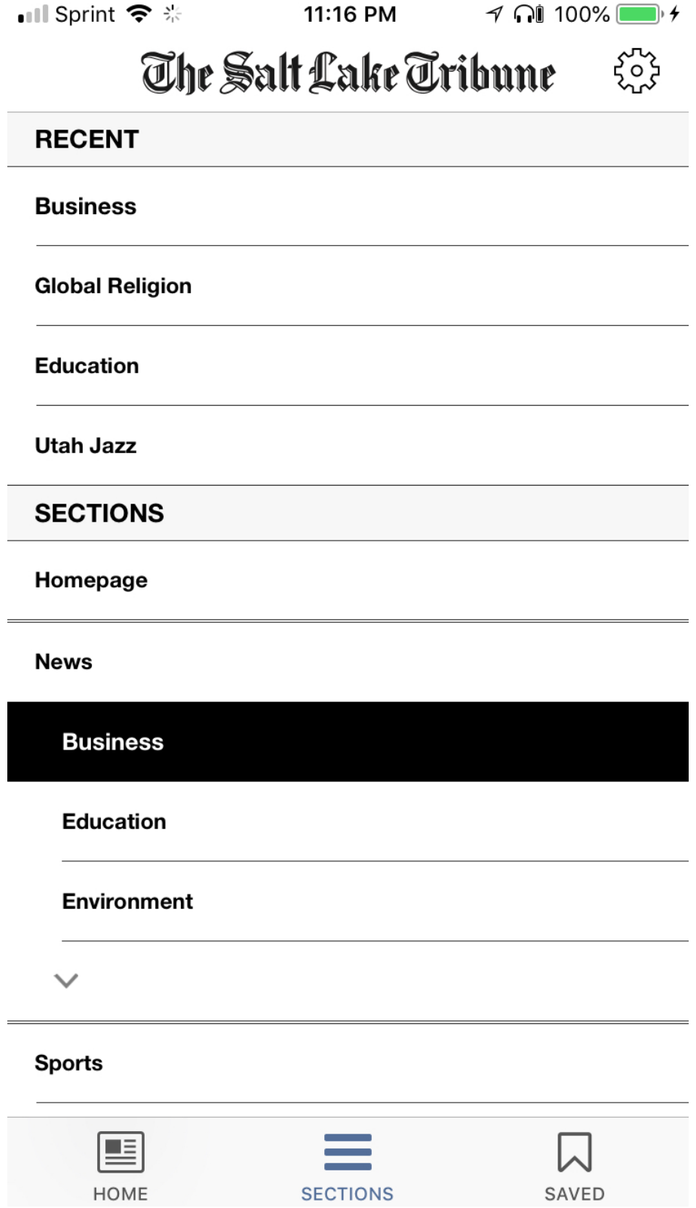
Structure of a News App

- Section Fronts
- Articles
- Section List
- Search
- Saved Articles
- Segmented Push Notifications



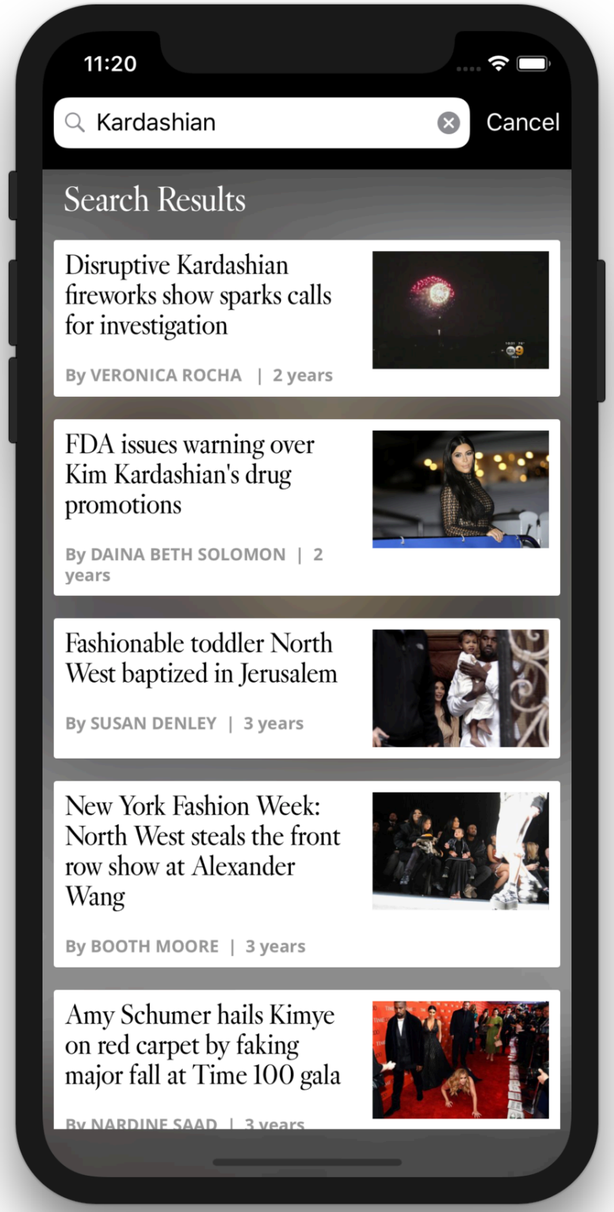
Structure of a News App

- Section Fronts
- Articles
- Section List
- Search
- Saved Articles
- Segmented Push Notifications



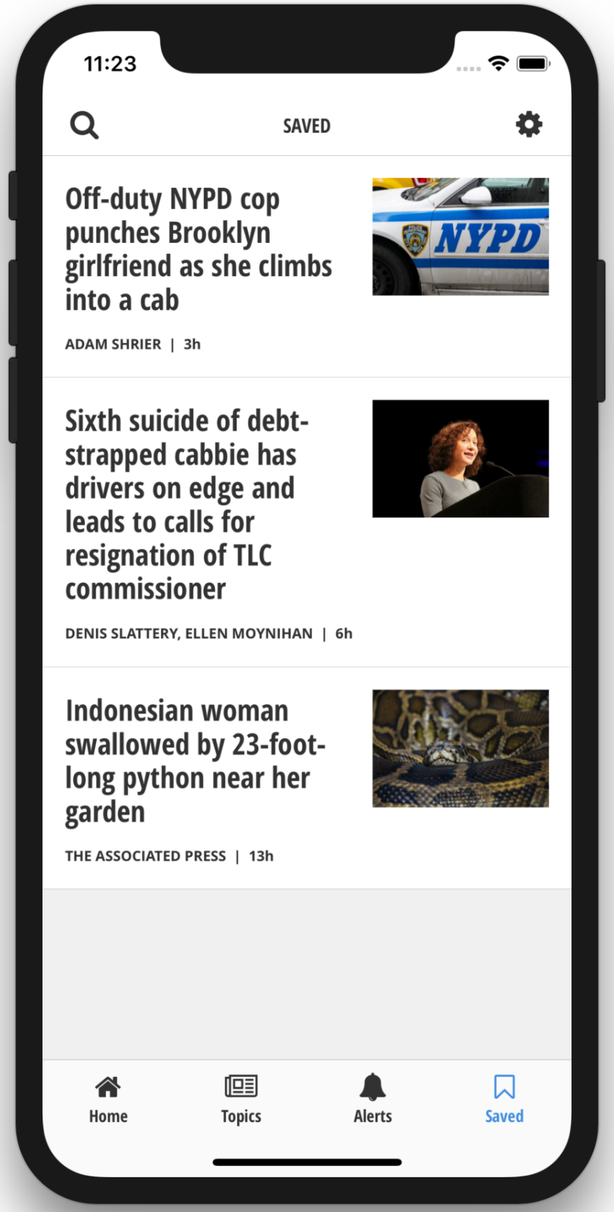
Structure of a News App

- Section Fronts
- Articles
- Section List
- Search
- Saved Articles
- Segmented Push Notifications



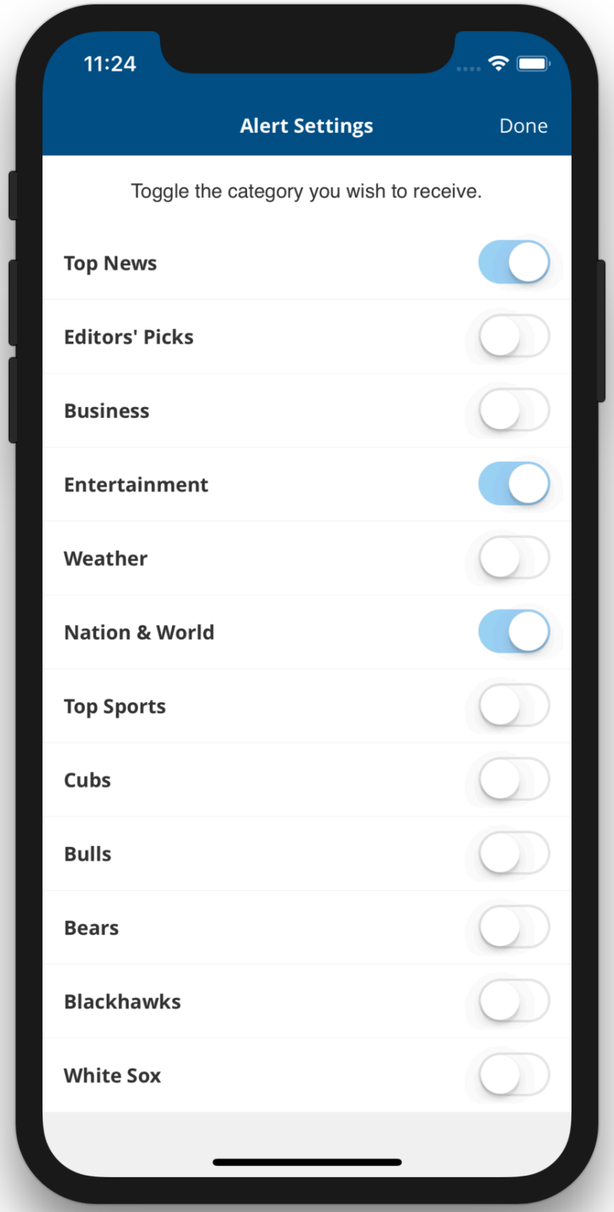
Structure of a News App

- Section Fronts
- Articles
- Section List
- Search
- Saved Articles
- Segmented Push Notifications



Structure of a News App

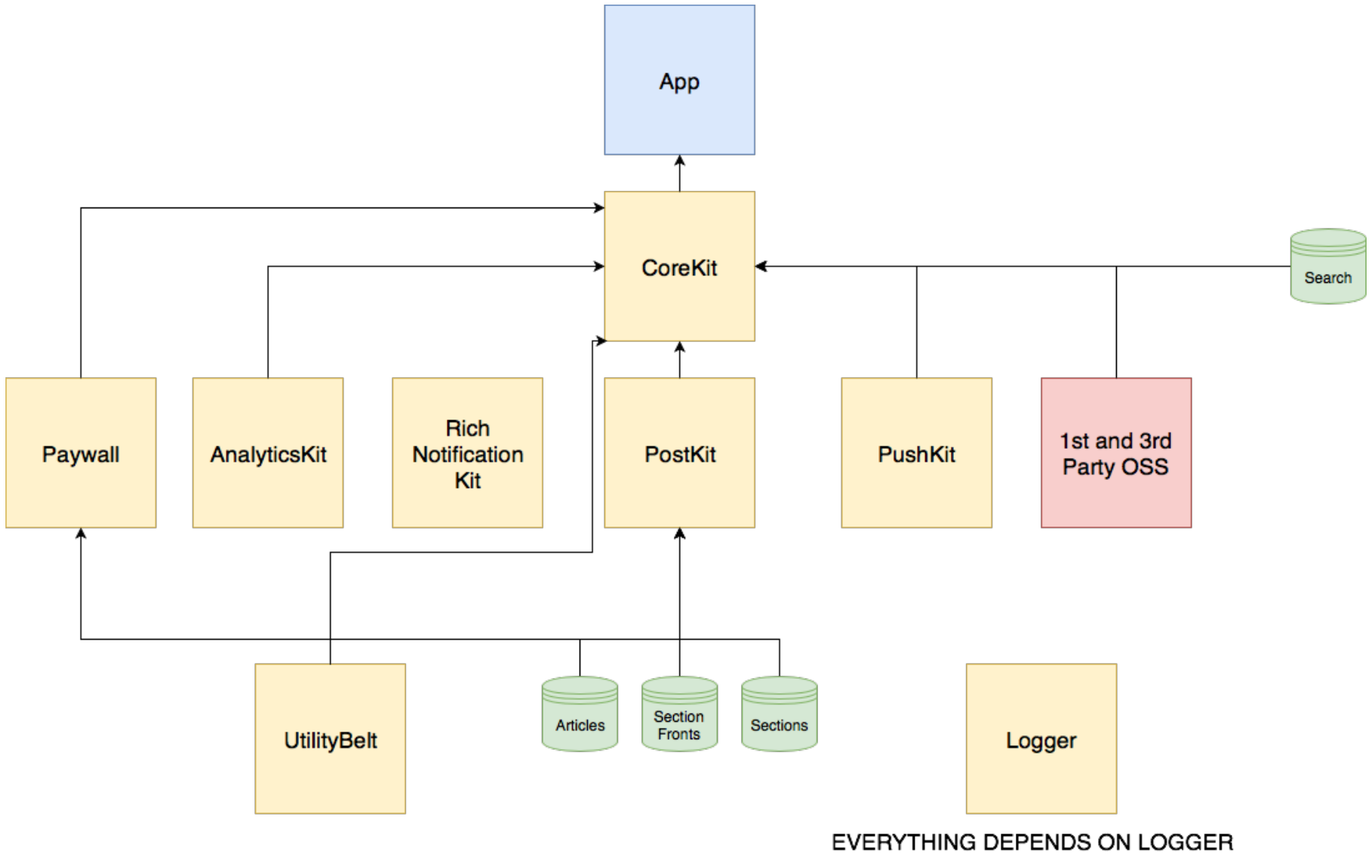
- Section Fronts
- Articles
- Section List
- Search
- Saved Articles
- Segmented Push Notifications



App Customization Options

- Ad Provider(s)
- Analytics Provider(s)
- Paywall Provider
- Push Notification Provider
- Theming (Colors, Fonts, Images, Paragraph Styles)
- Traffic Provider
- Weather Provider

High-Level Architecture of a Unicorn App



Partner.swift

```
struct Partner {
  static let shared = Partner()
  struct Colors {...}
  struct Fonts {...}
}

extension Partner: AnalyticsManaging {...}
extension Partner: AdsManging {...}
extension Partner: PartnerManaging {...}
extension Partner: PaywallManaging {...}
extension Partner: PushManaging {...}
extension Partner: Themeable {...}
/// etc.
```

Partner+PartnerManaging.swift

```
extension Partner: PartnerManaging {

  // Feature Toggles
  var isOfflineModeEnabled: Bool { get }
  var isPaywallEnabled: Bool { get }
  var isSearchEnabled: Bool { get }

  // Legal Toggles
  var requiresTrackingConsent: Bool { get } // GDPR

  // Cost-Saving Toggles
  var searchOnUserInput: Bool { get } // e.g., Typeahead

  /// and many more (seriously, tons of knobs to tweak)!
}
```

Partner Setup

```
// PartnerAppDelegate.partner
override var partner: Any {
    return Partner.shared
}

// CoreAppDelegate().setupPartner()
open func setupPartner() {
    PartnerManager.currentPartner = partner as? PartnerManaging
                                  ?? PartnerManager.currentPartner

      // etc.
}

// PartnerManager.swift
public struct PartnerManager {
    public static var currentPartner: PartnerManaging = DefaultPartner()
}

// PartnerManaging.swift
public struct DefaultPartner: PartnerManaging {
    // Default PartnerManaging Implementation
}
```

Protocol Oriented Theming

- AppThemeable
- ArticleThemeable
- ImageThemable
- NavigationThemeable
- PaywallThemeable
- PushThemeable

Problems with Existing Solution

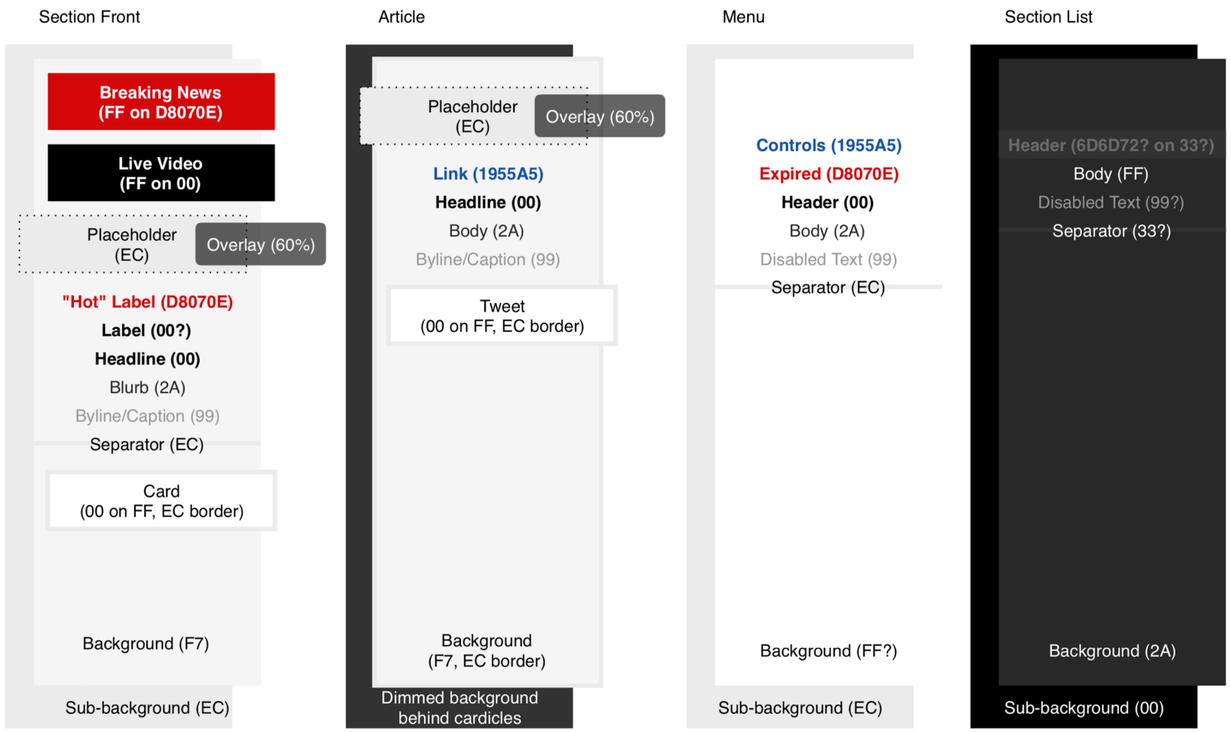
- Current implementation does not scale as there are hundreds of knobs.
- Default protocol implementations make it hard to discover:
  - Renamed variables
  - New variables
  - Deprecated methods (Swift Forums Post)

Future of Theming

`Codable`-backed JSON files

- **Style.json**
  - A list of *all* the *tweakable* elements, broken up in a declarative way by view, VC, and element type.
- **SuperStyle.json**
  - A superset of *tweakable* elements that are customized and *trickle down* to all the values defined in `Style.json`.

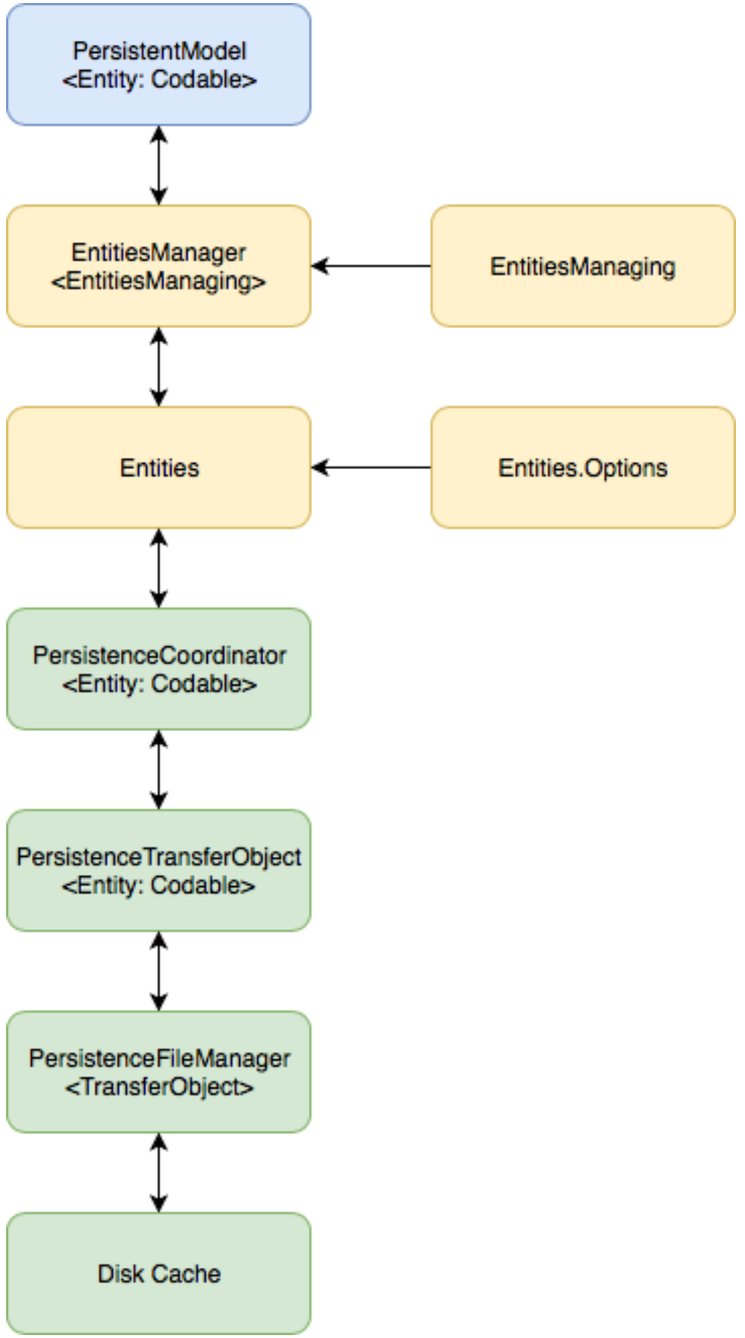
Image: Andrew Schoenfeld, Classic App Maintainer



Entities

A generic front-end that performs CRUD operations on Codable-conforming models.

- Persists configuration files
- Persists user data
  - Saved Articles
  - Push Notifications
- Scales to store any codable model

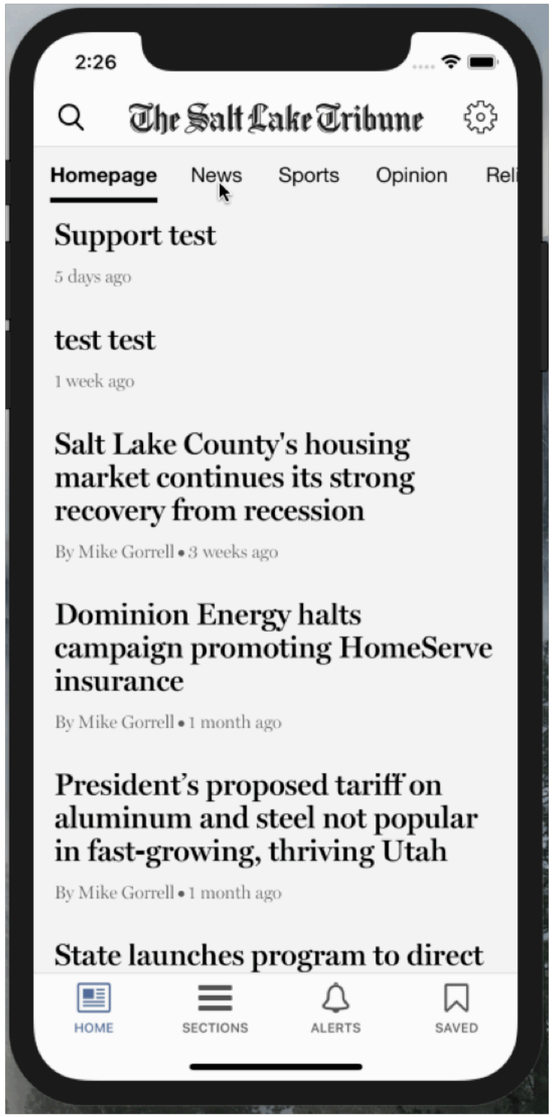


Entities: Abstract Example

```
public var main: MainConfig? { /// Example: Main Configuration File
    get {
        return getValue(MainConfig.self, for: .main)
    } set {
        setValue(for: .main, toNewValue: newValue)
    }
}

public func getValue<T: Codable>(_ type: T.Type = T.self, for options: Options,
                                 environmentOverride environment: Environment = EnvironmentManager.shared.current) -> T? {
    let key = coordinatorsKey(for: options, environment: environment)
    let coordinator = getCoordinator(for: key) as? PersistenceCoordinator<T> ??
        PersistenceCoordinator<T>(for: options, environmentOverride: environment)
    setCoordinator(coordinator, forKey: key)
    return coordinator.read()
}

public func setValue<T: Codable>(for options: Options, toNewValue newValue: T?,
                                 environmentOverride environment: Environment = EnvironmentManager.shared.current) {
    let key = coordinatorsKey(for: options, environment: environment)
    let coordinator = getCoordinator(for: key) as? PersistenceCoordinator<T> ??
        PersistenceCoordinator<T>(for: options, environmentOverride: environment)
    coordinator.update(with: newValue)

    setCoordinator(newValue != nil ? coordinator : nil, forKey: key)
}
```

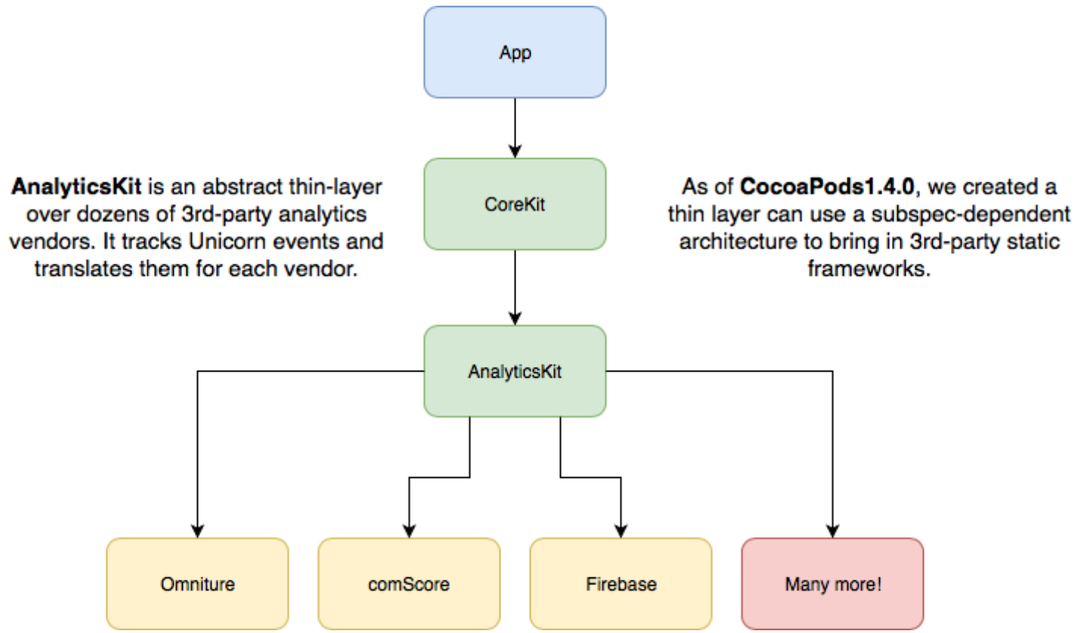
Environment Switching

- Built on top of **Entities**, environment switching allows partners to test their app against production, staging, and development environments.
- Useful for testing:
  - Section front layouts
  - Paywall API changes
- In the past we had to deploy new TestFlight builds whenever they wanted to test against a different environment.



CocoaPods (AnalyticsKit)

```
s.static_framework = true
s.swift_version = '4.1'

s.ios.deployment_target = 9.0
s.tvos.deployment_target = 9.0

s.default_subspec = 'Core'
s.subspec 'Core' do |sub|
  sub.source_files = 'Sources/Core/**/*'
end

s.subspec 'ComScore' do |sub|
  sub.dependency 'AnalyticsKit/Core'
  sub.ios.dependency 'ComScore'
  sub.source_files = 'Sources/ComScore'
end

s.subspec 'Firebase' do |sub|
  sub.dependency 'AnalyticsKit/Core'
  sub.ios.dependency 'FirebaseAnalytics', '>= 4'
  sub.ios.source_files = 'Sources/Firebase'
end

s.subspec 'Omniture' do |sub|
  sub.dependency 'AnalyticsKit/Core'
  sub.ios.dependency 'AdobeMobileSDK', '>= 4.15.0'
  sub.tvos.dependency 'AdobeMobileSDK/TVOS', '>= 4.8.1'
  sub.source_files = 'Sources/Omniture'
end

// Any many many more!
```



AnalyticsKit is an abstract thin-layer over dozens of 3rd-party analytics vendors. It tracks Unicorn events and translates them for each vendor.

As of **CocoaPods1.4.0**, we created a thin layer can use a subspec-dependent architecture to bring in 3rd-party static frameworks.

BuddyBuild

- Built for *mobile* developers with both CI and CD and ***just works!***™
- Deploys to ~~iTunes~~ App Store Connect automatically with custom rules.
- No need to maintain *fastlane* scripts or `yaml` files across dozens of repos.
- Only CI that did not fail during our tests (seriously - we spent a month)
  - We tested `circle`, `travis`, `bitrise`
  - We tested Xcode Bots on a Mac Mini (in 2015-2017) 😆 😂 🤣

We ❤️ BuddyBuild!

> *For all 20 iOS applications and libraries we have under active development, we're able to focus solely on improving our codebase and iterating on our projects, and we trust buddybuild to handle the rest.*

Arthur Sabintsev, Lead iOS Engineer at the Washington Post 

Thoughts on Open Source

I ❤️ contributing to Open Source.

I rarely use other people's Open Source tools.

Why?

Open Source Tools We Use [3]

3rd Party	3rd Party	1st Party
CocoaLumberjack	RNCryptor	Failable
Eureka	SVProgressHUD	Freedom
Mockingjay	XCDYouTubeKit	Siren

[3] Not listed here are base required SDKs (Ad SDKs, Analytics SDKs, Crash Monitoring, etc.)

What's Next?

- Arc Mobile SDK Offering
- Automating Upstream PRs
- I18n and L10n
- Spin-up Apps with a push of a button via a web admin portal
- Theming Engine Rewrite
- Unified Configuration Files

Architecture

- Native Development (Swift+Objc & Kotlin+Java)
- Model-View-Controller (MVC)
- Builders, View-Models, etc
- POP (Protocol-Oriented-Programming)

Process

- Git Strategies
  - GitFlow
  - All code comes in via a detailed Pull Request
  - Pull Request Templates
  - Pair Programming for Code Reviews
- Rituals
  - Daily Scrum
  - Daily Check-in
  - Weekly Kanban

Team Structure

Big proponents of remote work!

- 1 Lead (Baltimore)
- 3 Seniors (DC, Denver, Tallahassee)
- 2 Juniors/Mid (Cincinnati, Manhattan)
- 1-2 interns in DC every summer

WaPo @ iOSDevCampDC 2018

Mac, Dustin, Josip, Arthur, Erik

ArtSabintsev on Twitter & Github.

Thank You!

https://github.com/ArtSabintsev/iOSDevCampDC-2018

ArtSabintsev on Twitter & Github.