Building the Fabric iOS App

@Javi

Outline

→ Fabric

→ GraphQL

→ Dependency Injection

→ Error Reporting

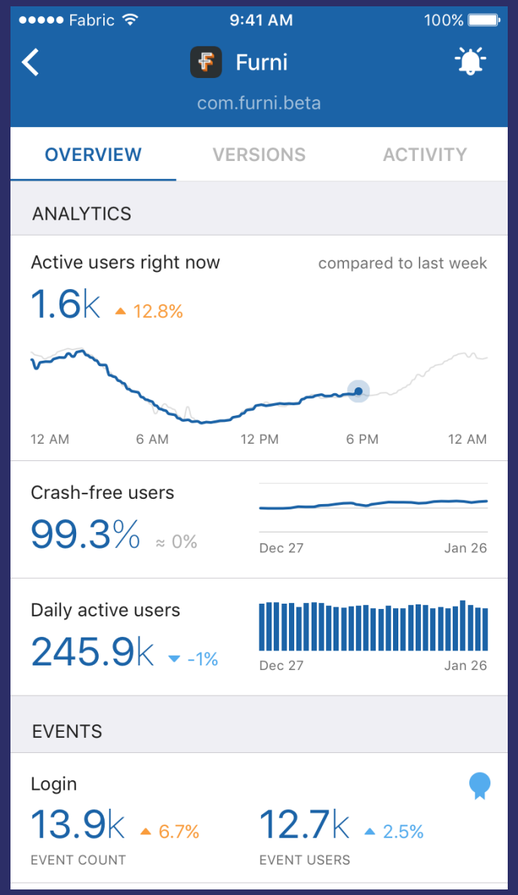
→ Fastlane

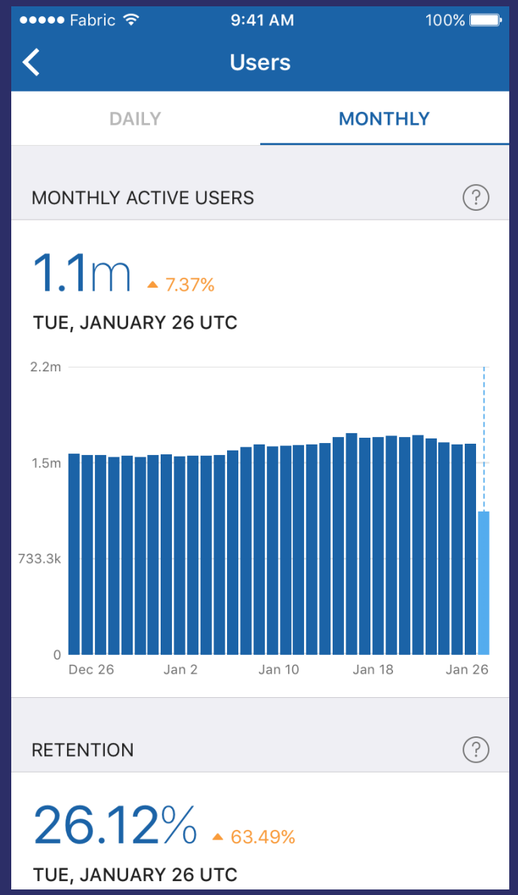
What's Fabric?

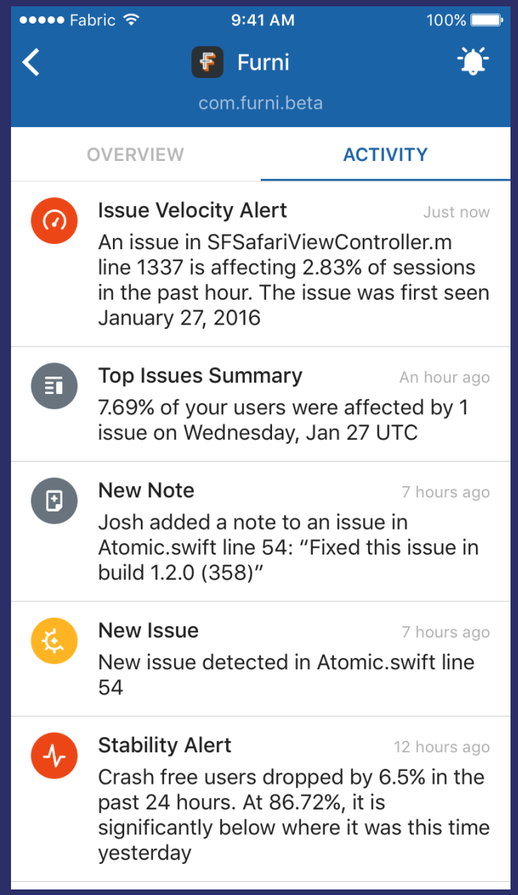
Fabric


fabric.io
Cannonball
io.fabric.samples.cannonball
Luis Cipriani
Cannonball
1
ACTIVE USERS RIGHT NOW
DAILY ACTIVE USERS
2
-33.3%
DAILY NEW USERS
1
MONTHLY ACTIVE USERS
136
-19.0%
CRASH-FREE USERS
50.0 %
SESSIONS
7
75.0%
SHUFFLED WORDS
3
3
DAILY ACTIVE USERS
30
20
10

Fabric


Fabric 9:41 AM 100%
Furni
com.furni.beta
OVERVIEW
VERSIONS
ACTIVITY
ANALYTICS
Active users right now
compared to last week
1.6k ▲ 12.8%
12 AM
6 AM
12 PM
6 PM
12 AM
Crash-free users
99.3% ≈ 0%
Dec 27
Jan 26
Daily active users
245.9k ▼ -1%
Dec 27
Jan 26
EVENTS
Login
13.9k ▲ 6.7%
EVENT COUNT
12.7k ▲ 2.5%
EVENT USERS


Fabric 9:41 AM 100%
Users
DAILY
MONTHLY
MONTHLY ACTIVE USERS
1.1m ▲ 7.37%
TUE, JANUARY 26 UTC
2.2m
1.5m
733.3k
0
Dec 26
Jan 2
Jan 10
Jan 18
Jan 26
RETENTION
26.12% ▲ 63.49%
TUE, JANUARY 26 UTC


Fabric 9:41 AM 100%
Furni
com.furni.beta
OVERVIEW
ACTIVITY
Issue Velocity Alert
Just now
An issue in SFSafariViewController.m line 1337 is affecting 2.83% of sessions in the past hour. The issue was first seen January 27, 2016
Top Issues Summary
An hour ago
7.69% of your users were affected by 1 issue on Wednesday, Jan 27 UTC
New Note
7 hours ago
Josh added a note to an issue in Atomic.swift line 54: “Fixed this issue in build 1.2.0 (358)”
New Issue
7 hours ago
New issue detected in Atomic.swift line 54
Stability Alert
12 hours ago
Crash free users dropped by 6.5% in the past 24 hours. At 86.72%, it is significantly below where it was this time yesterday

GraphQL[1]

[1] http://graphql.org/

GraphQL - Queries

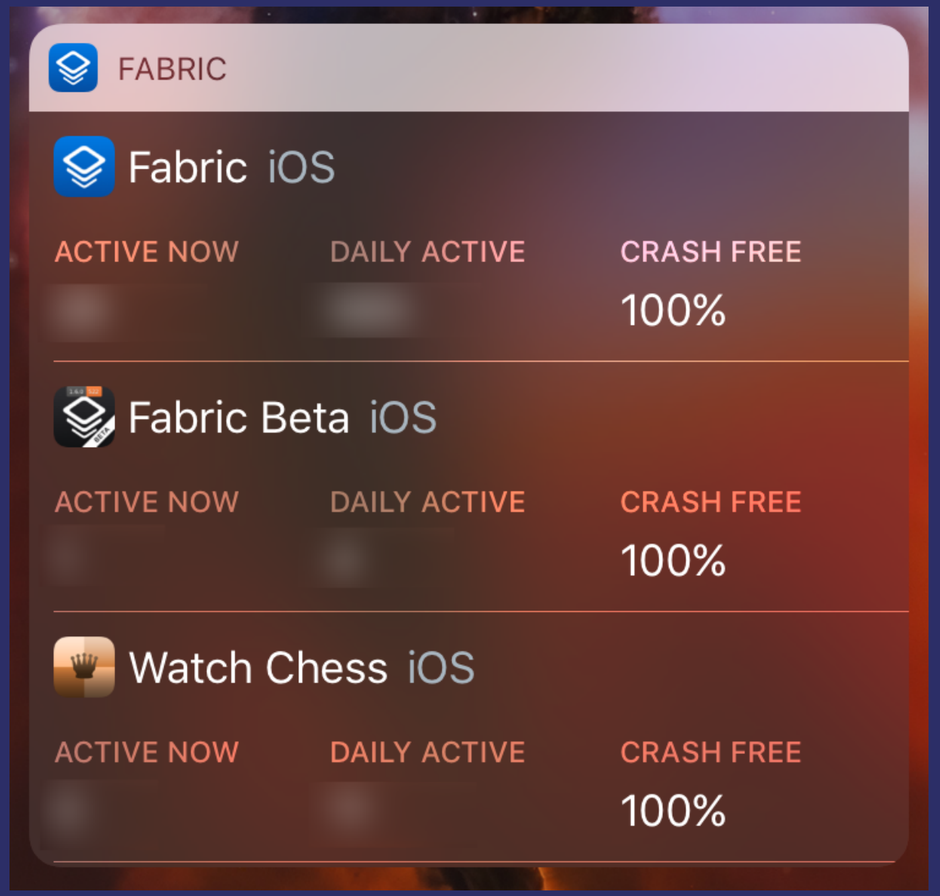
→ `GET /users/<user_id>`

GraphQL - Queries

→ GET /users/<user_id>

→ GET /users/<user_id>/friends

GraphQL - Queries



GraphQL - Queries

```
{
  user(id: 4802170) {
    name
    profilePicture(size: 50)  {
      uri
      width
    }
    friends(first: 5) {
      id
      name
    }
  }
}
```

GraphQL - Queries


```
{
  "data": {
    "user": {
      "id": "4802170",
      "name": "Javi",
      "profilePicture": {
        "uri": "http://bit.ly/1E8lYDq",
        "width": 50
      },
      "friends": [
        {
          "id": "305249",
          "name": "Jony Ive"
        },
        {
          "id": "3108935",
          "name": "Elon Musk"
        }
      ]
    }
  }
}
```

GraphQL - Features

→ Hierarchical, human-readable query language

→ Strongly-typed

→ Version free

GraphQL - GraphiQL

Dependency Injection

Dependency injection

Software design pattern that implements inversion of control for resolving dependencies. An injection is the passing of a dependency to a dependent object that would use it.

— https://en.wikipedia.org/wiki/Dependency_injection

Dependencies

→ View Controller => Data Provider

→ Data Provider => API Client

→ Persistence Manager => Database Client

Dependency Resolution

→ Self-instantiation

→ Global access

→ Injection via setters

→ Injection via constructor

Dependency Resolution - Self-Instantiation

```
class ViewController {
    private let dataProvider: DataProvider

    init() {
        self.dataProvider = DataProvider()
    }
}
```

Dependency Resolution - Global Access

```
class ViewController {
    private let dataProvider: DataProvider

    init() {
        self.dataProvider = DataProvider.shared
    }
}
```

Dependency Resolution - Injection via Setters

```
class ViewController {
    var dataProvider: DataProvider!

    init() {

    }
}
```

Dependency Resolution - Injection via Constructor

```
class ViewController {
    private let dataProvider: DataProvider

    init(dataProvider: DataProvider) {
        self.dataProvider = dataProvider
    }
}
```

Easy vs Simple

Easy - Hard

Simple - Complex

Dependency injection

Implicit vs explicit

Dependency injection - UIStoryboard

```
class UIStoryboard {
    func instantiateViewController(withIdentifier identifier: String) -> UIViewController
}
```

Dependency injection - UIStoryboard

```
let sb = UIStoryboard...
let vc = sb.instantiateViewController(withIdentifier: "ID") as! ApplicationOverviewVC
vc.userSession = ...
vc.applicationID = ...
```

Dependency injection - UIStoryboard

```
final class ApplicationOverviewVC {
    // I hope you like "!"s...
    var userSession: UserSession!
    var applicationID: String!

    func viewDidLoad() {
        super.viewDidLoad()

        self.userSession.foo()
        // Or...
        if let userSession = self.userSession {
            userSession.foo()
        } else { /* Can't do anything useful here... */ }
    }
}
```

Ayaka Nonaka
@ayanonagon

Follow

Optionals are great because we can make things non-optional. Just because we have optionals in Swift shouldn't mean we have ? everywhere.

RETWEETS 24 LIKES 92

12:29 PM - 24 Mar 2016

Dependency injection in the Fabric app

```
final class ApplicationListViewController: UIViewController {
    init(viewModel: ApplicationListViewModel)
}

final class ApplicationListViewModel {
    init(fabricAPI: AuthenticatedFabricAPI)
}

public final class AuthenticatedFabricAPI {
    public init(session: Session)
}

public struct Session {
    let accessToken: String
}
```

DataLoadState

```
final class ApplicationListViewModel {
    var applications: [Application]?
}
```

DataLoadState

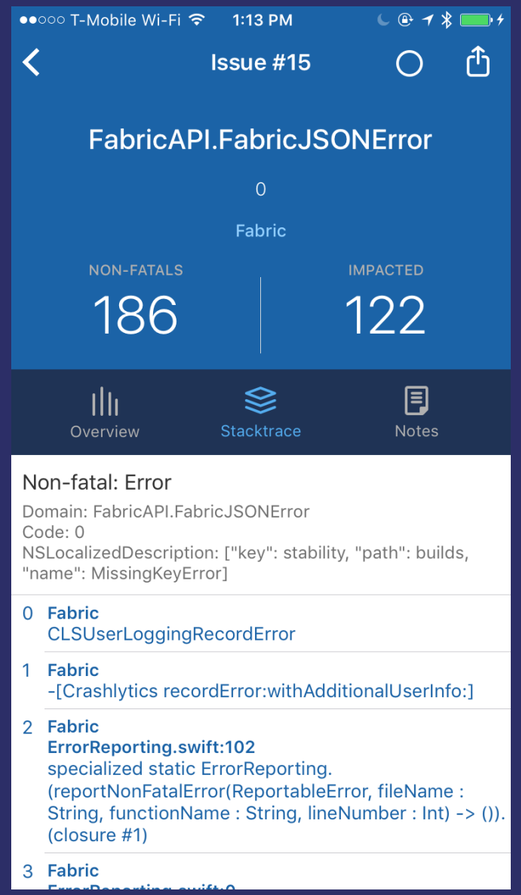
```
enum DataLoadState<T> {
    case loading
    case failed
    case loaded(T)
}

final class ApplicationListViewModel {
    var applications: DataLoadState<[Application]> = .loading
}
```

Error Reporting

Error Reporting

```
class Crashlytics {
    public func record(_ error: NSError, withAdditionalUserInfo: [AnyHashable : Any]?)
}
```

Error Reporting



Error Reporting

```
public protocol ReportableError: Swift.Error {
    var info: [String : Any] { get }
    var underlyingErrors: [ReportableError] { get }
}
```

Error Reporting

```
private enum URLHandlerError: ReportableError {
    case unknownURL(URL)
    case malformedURLMissingParameter

    /// This is useful because the `NSError` conversion will only give us a "code" integer.
    private var name: String {
        switch self {
            case .unknownURL: return "UnknownURL"
            case .malformedURLMissingParameter: return "MalformedURLMissingParameter"
        }
    }

    var info: [String : Any] {
        var info: [String : Any] = ["name": self.name]

        switch self {
            case let .unknownURL(url): info["url"] = url.path ?? ""
            case .malformedURLMissingParameter: ()
        }

        return info
    }
}
```

Error Reporting

```
extension ReportableError {
    var errorCode: Int {
        return (self as Error).code
    }

    var errorDomain: String {
        return (self as Error).domain
    }

    var asNSError: NSError {
        return NSError(
            domain: self.errorDomain,
            code: self.errorCode,
            userInfo: self.info
        )
    }
}
```

Error Reporting

```
var allErrors: [ReportableError] {
    return [
        [self],
        self.underlyingErrors.flatMap { $0.allErrors } // Recur
    ]
    .flatMap { $0 }
}
```

Error Reporting

```
public final class ErrorReporting {
    // We get file and function names for free
    internal func report(
        error: ReportableError,
        filePath: StaticString = #file,
        functionName: StaticString = #function,
        lineNumber: Int = #line
    ) {
        error.allErrors.forEach {
            var userInfo = $0.info
            userInfo["file"] = "\((filePath as NSString).lastPathComponent):\(lineNumber)"
            userInfo["function"] = functionName

            Crashlytics.sharedInstance().recordError($0.asNSError, withAdditionalUserInfo: userInfo)
        }
    }
}
```

fastlane

Fastlane

→ Code signing

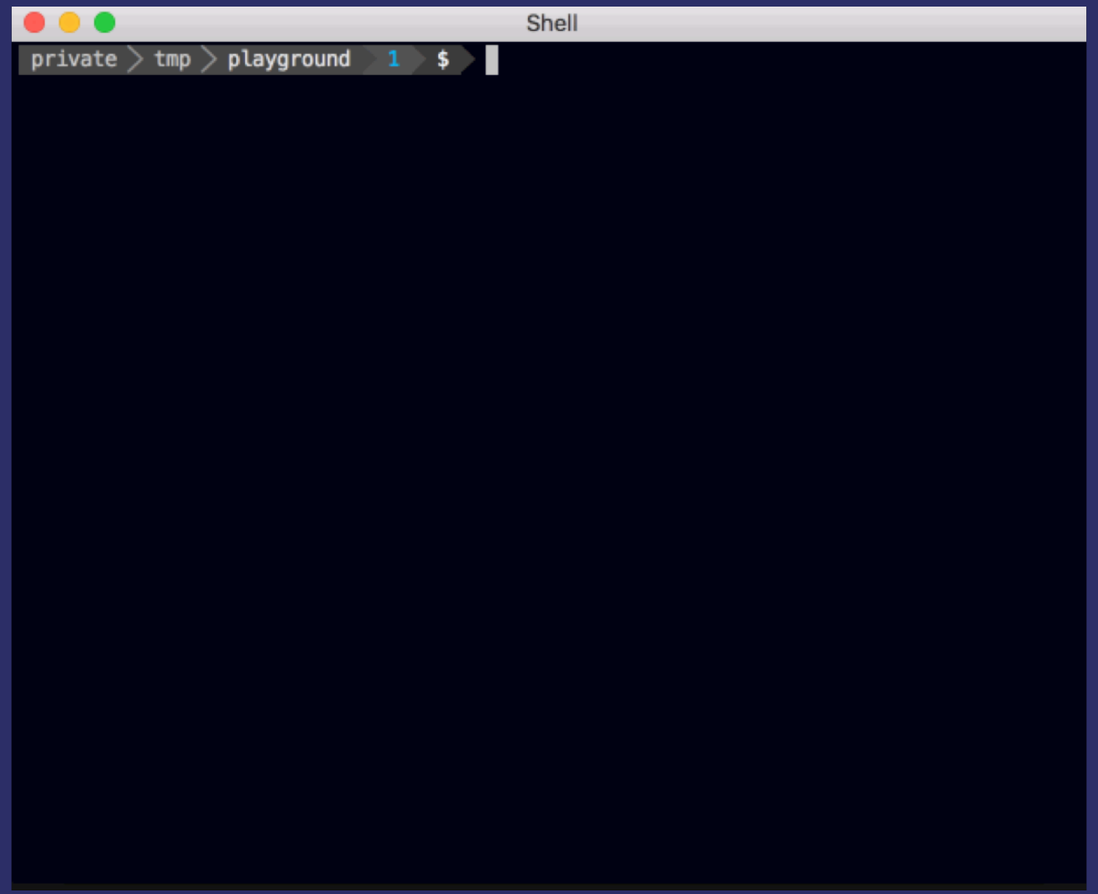
→ Deployment

Fastlane - Code Signing

Fastlane - Code Signing

https://codesigning.guide/

Fastlane - match



Fastlane - Deployment

Fastlane - Deployment

```
lane :appstore do
  snapshot                    # Generate screenshots for the App Store
  match                       # Ensure code signing is set-up
  gym                         # Build your app
  deliver                     # Upload the screenshots and the binary to iTunes
  slack                       # Let your team-mates know the new version is live
end
```

Fastlane - Deployment

```
lane :appstore do
  snapshot                    # Generate screenshots for the App Store
  match                       # Ensure code signing is set-up
  gym                         # Build your app
  deliver                     # Upload the screenshots and the binary to iTunes
  slack                       # Let your team-mates know the new version is live
end
```

```
$ fastlane appstore
```

Fastlane - Deployment

```
lane :appstore do |options|
    ensure_git_branch(branch: "(master|release\\S+)")
    ensure_git_status_clean
    ensure_xcode_version(version: "8.0")
    increment_version_number(version_number: options[:version_number])
    build_number = increment_build_number
    complete_version_number = "#{version_number} (#{build_number})"
    commit_version_bump(message: "Version bump to #{complete_version_number}")
    set_github_release(repository_name: "...", name: complete_version_number)
    ...
end
```

```
$ fastlane appstore version_number:1.6.0
```

Fastlane.tools

Questions?

@Javi / javi@twitter.com

Thanks