Mobile-First Architectures

Alexander Stigsen

QCon 2015

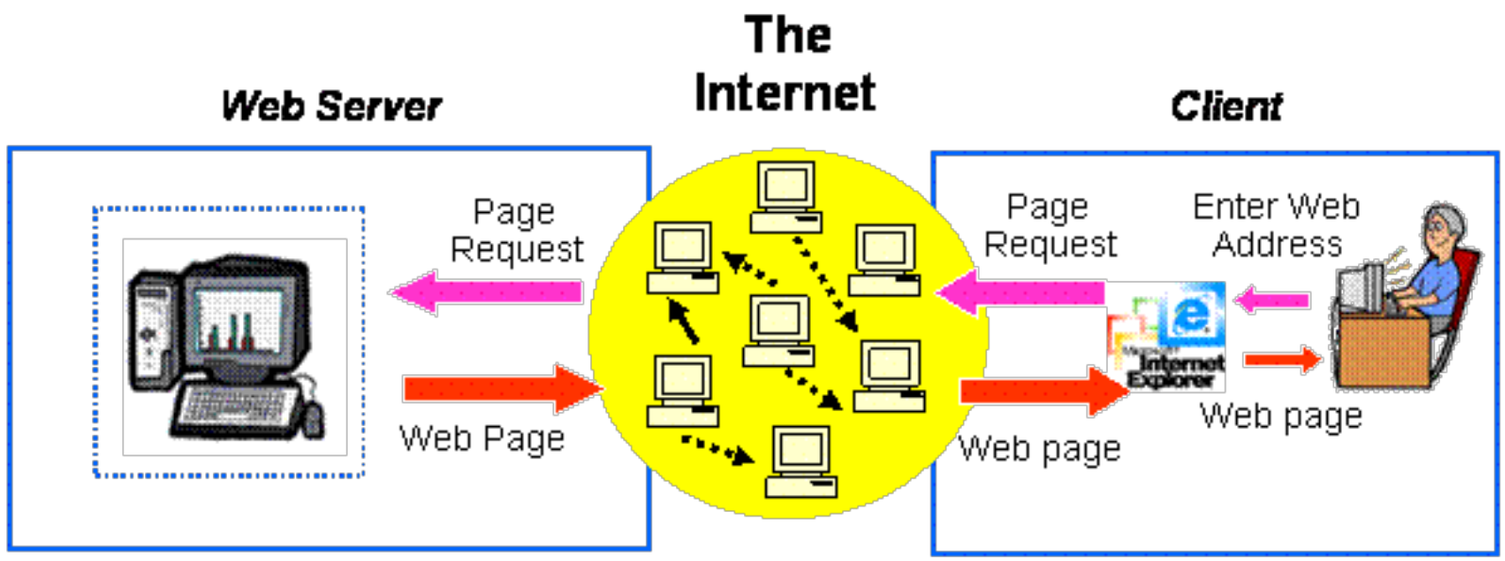
Who am I?

- Former Mobile Engineer from Nokia
- Working on mobile since 1999
- Currently CEO of Realm, building a modern mobile database
- Very biased :-)

Realm

Problem

Mobile Apps are still being build as if they were web apps from the 90s.


Web Server
The Internet
Client
Page Request
Web Page
Page Request
Enter Web Address
Web page
Web page

Mobile apps are still treated as dumb clients

Why?

- This is how (web) apps have been developed in the last decade. So this is what developers are used to.

- Existing API's are designed with this mindset, so the mobile apps has to conform.

- Makes it easier to build both mobile and web versions of the app.

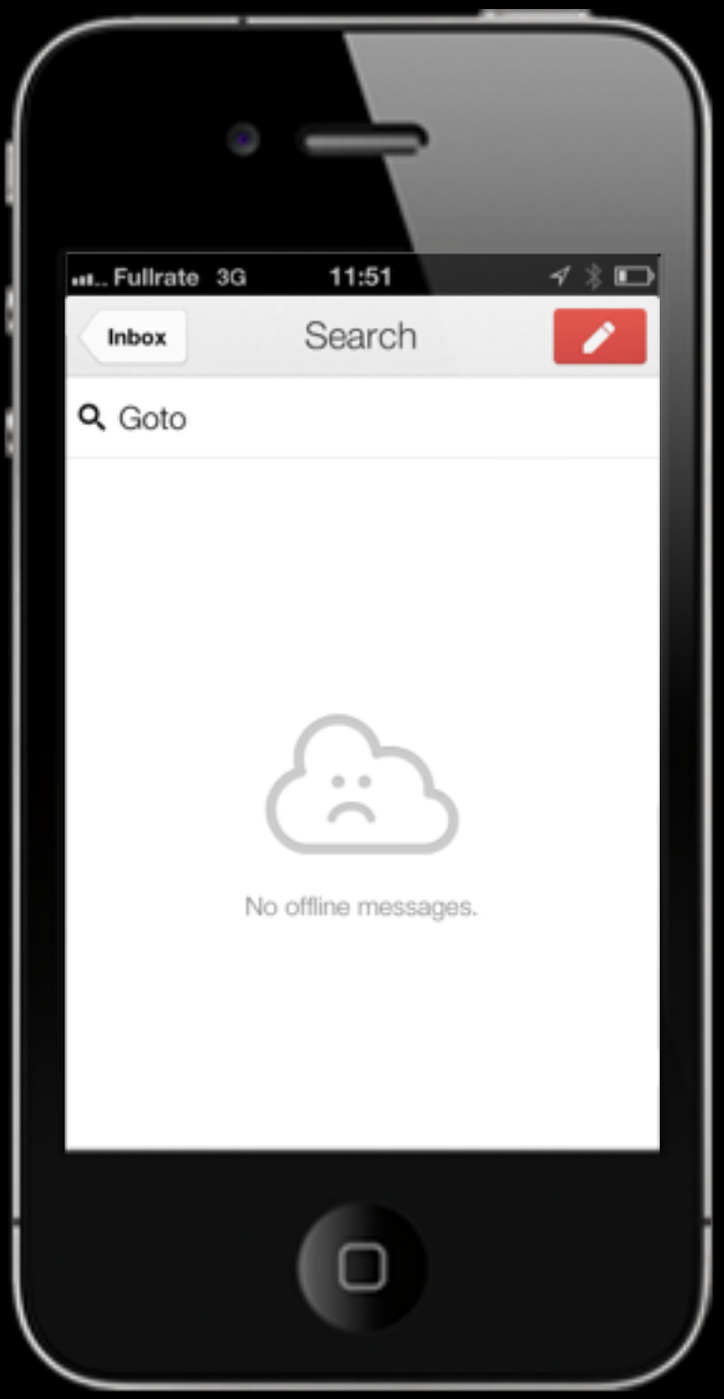
Drawbacks:

- ✗ User Experience
- ✗ Connectivity Tolerance
- ✗ Server Cost

User Experience

- Users have much higher expectations of native apps
- Apps gets more advanced and capable and as such need more complex data models
- No latency or wait times accepted

Connectivity



- Users expect apps to always work
- The failed promise of ubiquitous connectivity
  - Even in SF connectivity is spotty
  - Go to the countryside, still many places with no connectivity
  - Crises or huge events often takes down connectivity
- Developing countries
  - Next big market
  - Low connectivity
  - Expensive data plans

Escalating Server Cost

- #1 Killer of startups
- App may need millions of users before being able to monetize
- Server load scale with number of users
- Amazon is expensive!

Problems with the cache approach?

- Object access
  - *Can* be slow (if they have to get result from server)
  - Access to un-cached objects will not work when offline
- Queries
  - *Will* be slow (as they always have to go over the network)
  - Will not work when offline
- Mobile team is dependent on backend team to implement features (slow iterations)

Solution?

Endpoint Computing

Realm

Modern phone hardware

iPhone 6

1.4 Ghz Dual-core CPU

1 GB memory

16-128 GB storage

Quad-core GPU

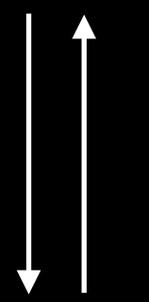
Use it!

- Primary store on phone
- User has all his data on the device
  - Always available
  - Always instant

- Backend is only for coordination, backup and statistics

Server-First

cache

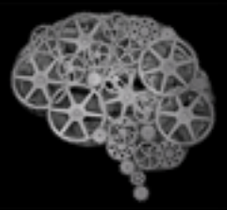


“smarts”

data

Server-First

cache



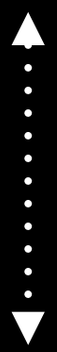
“smarts”

data

Mobile-First

“smarts”

data



backup

Where does this work?

From the data perspective, there are fundamentally three kinds of apps:

- Single user apps
- Collaborative apps
- Browser apps

Single user apps

All the in the app is owned by and only accessible by a single user (but this user may have multiple devices).

Examples: Mint, Expensify, Email apps…

Collaborative apps

The user has access to a subset of a larger dataset, where parts are shared with other users.

Examples: Slack, Asana, Facebook, Twitter…

Browser apps

The user is mostly searching and browsing a larger set of data created by other users.

Examples: YouTube, Tinder, Twitter…

But could probably still benefit from keeping the non-browsing data local

Mobile Driven Design

Change of mindset for developers

Start from device, building the best possible experience for the users

Then let that shape the backend, rather than the other way around

How does this affect your backend API?

Mostly One-way:
Examples: email, facebook timeline, twitter stream

- Probably not much change needed.
- Retrieve list of changes since last update

Two-way sync
Examples: collaboration, games, todo lists…

- Much harder to implement (especially the offline case)
- Need to handle conflicts

Call to action: we need much better synchronization tools

Is this always the right choice?

Primarily for apps that are user-centric (which most are).

Not for apps that just browse other peoples data

Benefits of Endpoint Computing

- Much more responsive apps
- Works even with weak or spotty connectivity
- Huge reduction of server load
- Mobile team can iterate much faster

The Next Frontier

- Live Collaboration
- Internet of Things
- Virtual Reality
- Wearables
- Drones

Q&A

http://realm.io

Realm