The Secret to Mobile

- API Design
- App Architecture
- Data Handling

Andy Lech

Topics

- Infrastructure differences between Web and Mobile
- Architecting your Mobile app for better results
- Making better data decisions for your Mobile app
- Replacing exceptions with extensible error models

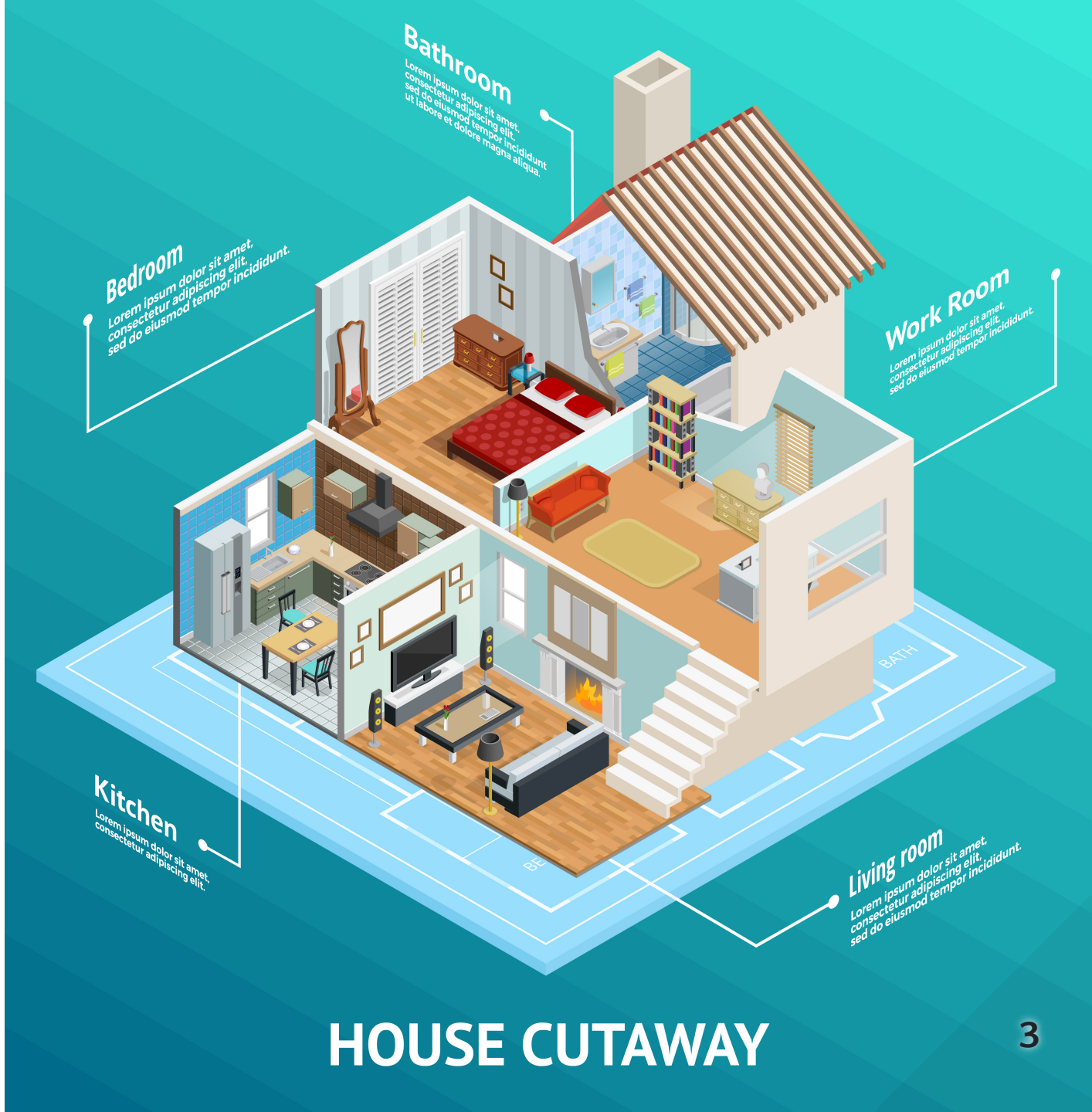
Part 1

Imagine your house was an app ...

User Perspective

- Energy Consumption: Just Works
- Energy Production: Somebody Else's Problem
- Resiliency: Generator (Maybe?)
- Focus: Amenities, Location, My Stuff is Here

Image by macrovector on Freepik



Energy Perspective (Power Grid)

- Energy Consumption: Always
- Energy Production: Our Problem
- Resiliency: Our Backups Have Backups (Hopefully)
- Focus: Keeping the Lights On (Literally)

Image by macrovector on Freepik

Energy Perspective (Solar)

- Power Grid Consumption: Backup
- Power Grid Production: Their Problem
- Resiliency: Battery, Power Grid
- Focus: Smartly Converting and Storing What I Get From The Sun

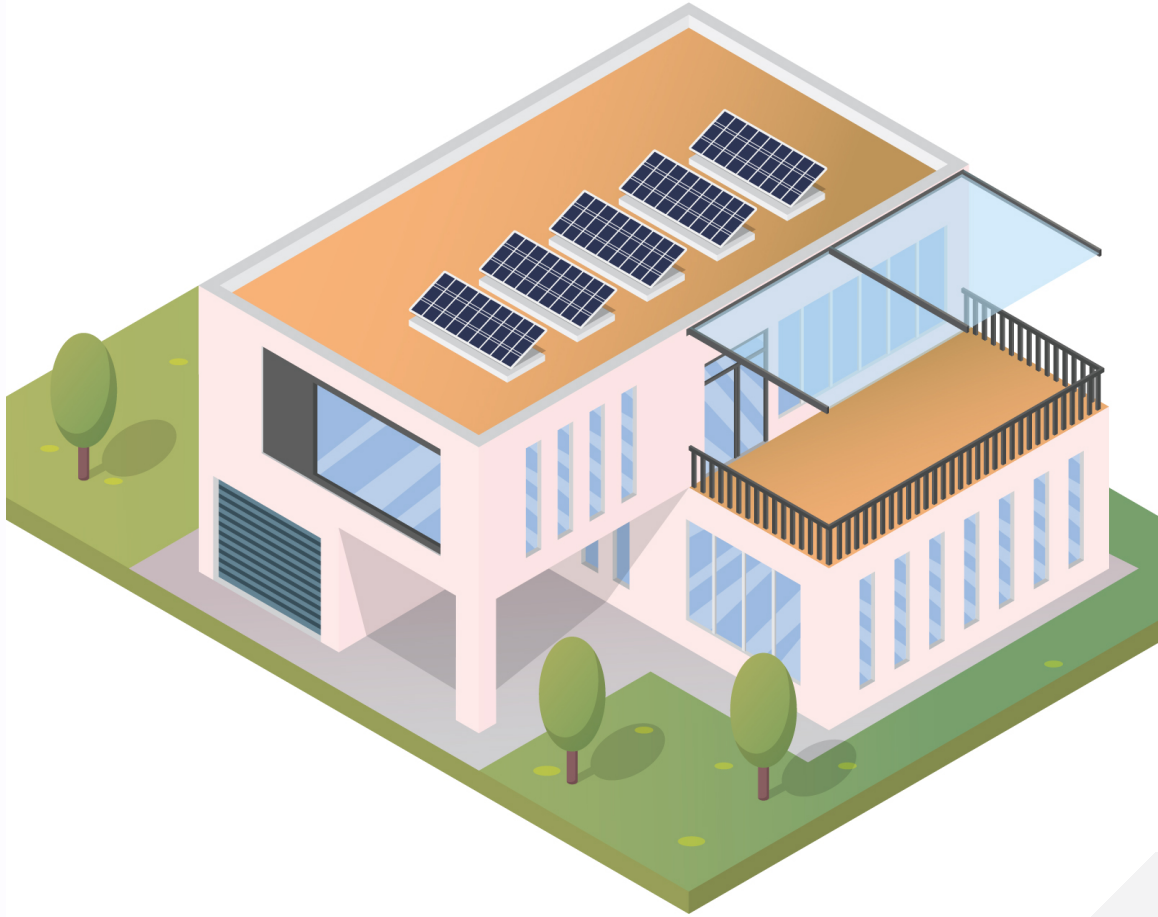


Image by macrovector on Freepik

Part 2

Mobile is just like Web but smaller

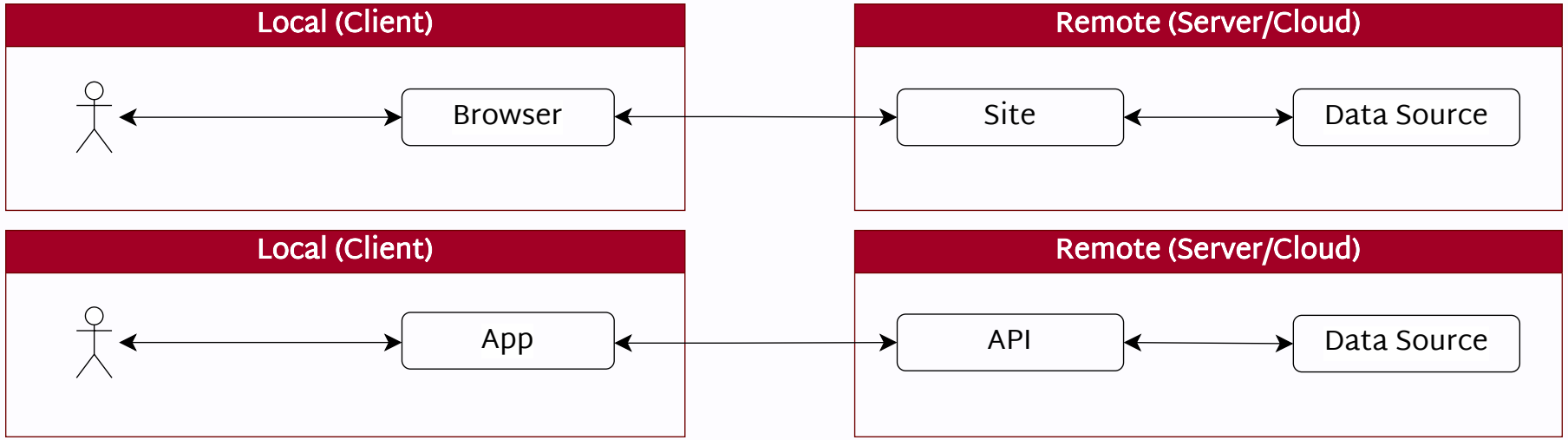
Right!?

or

Web devs make bad Mobile devs

... but they can learn

Web vs Mobile - High Level

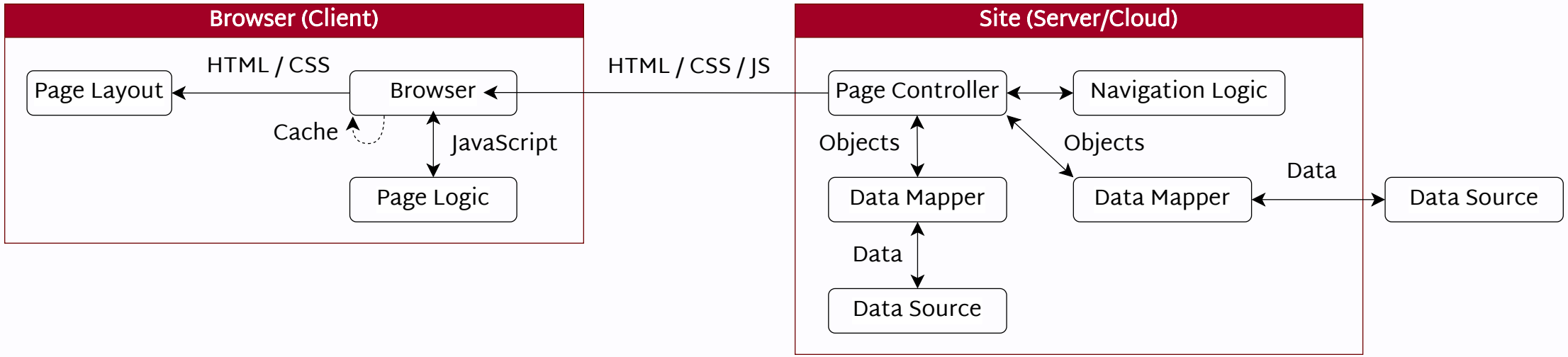


Same thing, right? Problem solved! Crisis averted!
Thank you and good night!

Web vs Mobile - Tech Stacks (.NET)

- Languages: C#, LINQ, F#, VB.NET, etc.
- Frameworks: .NET, Entity Framework, etc.
- Data: SQL Server, Azure, etc.
- IDEs: Visual Studio, Visual Studio Code, Rider, etc.
- Tools: ReSharper, (IDE) Extensions, etc.

Web Sites (Traditional)



- Server/cloud stack is the focus here, turning data into pages and layouts
- Complex layouts are generally built on top of templating frameworks
- Result: Web devs expect fat data pipes in the server/cloud stack to get large data payloads (object graphs) to choose what to filter/display in page layouts

Web Sites (Traditional)

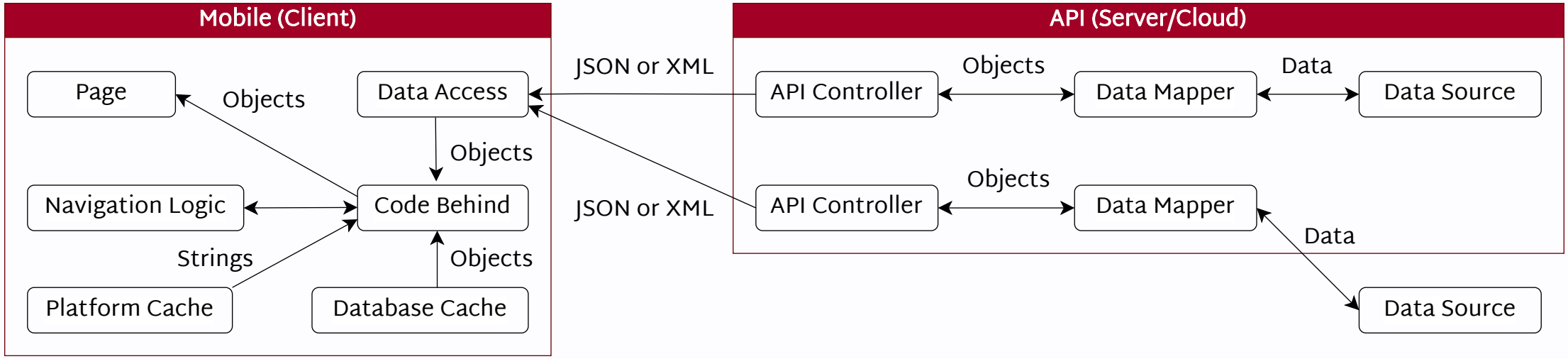
Browser (Client)

- Requests: Browse, Submit
- State Management: Cookies
- Data Mapping: None
- Caching: Local, Session, Cookies
- Resiliency: Browser
- Development Focus: Interactivity, Data Updates
- Data Goal: Fat Data Pipes (to Site)

Site (Server/Cloud)

- Responses: Page Layouts (Whole)
- State Management: Server/Cloud
- Data Mapping: Source to Site
- Caching: Server/Cloud
- Resiliency: Server/Cloud, Services
- Development Focus: Layouts, State, Services
- Data Goal: Fat Data Pipes (inside Site)

Mobile Apps



- App handles interactivity, page layouts, local caching, and API calls
- API stack delivers only data or status codes in response to app requests
- App pages tend to be focused on single tasks that call the API selectively
- Result: mobile/API devs focus more on reliable, just-in-time data delivery

Mobile Apps

App (Client)

- Requests: API Calls
- State Management: ViewModels, Cache
- Data Mapping: API to App
- Caching: Platform, Local DB
- Resiliency: Network State
- Development Focus: Interactivity, Data Updates, Layouts, State, API Calls
- Data Goal: Smart Data Pipes (to API)

API (Server/Cloud)

- Responses: Data (JSON/XML)
- State Management: Auth Tokens
- Data Mapping: Source to API
- Caching: Server/Cloud
- Resiliency: Server/Cloud, Services
- Development Focus: Data, Status Codes and Messages
- Data Goal: Smart Data Pipes (from App)

Part 3

"It's the data, stupid"

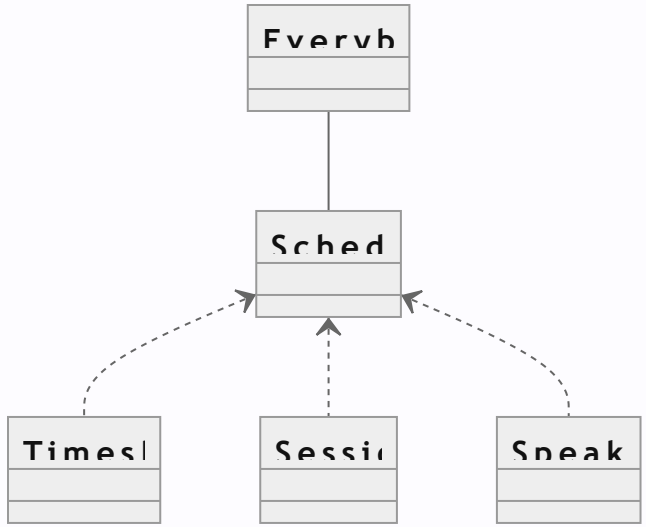
Insert picture of James Carville

Insert picture of The War Room whiteboard

API Design - Conference - Schedule

Requirements

- Everybody gets the same schedule
- The schedule changes infrequently
- When the schedule does change, it often involves multiple sessions
- Event Wi-Fi is notoriously fickle



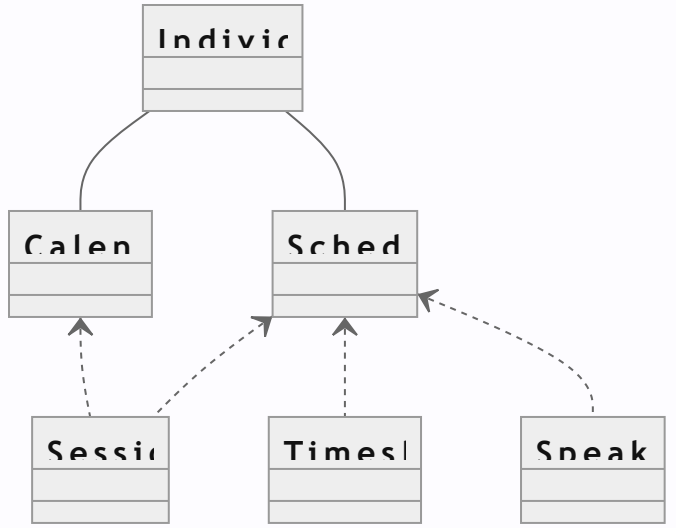
Solution

- Send whole schedule w/o redundant data but time-gated and cached

API Design - Conference - Itinerary

Requirements

- Everybody gets the same schedule
- Attendee adds sessions to itinerary
- Attendee shares their itinerary
- Event Wi-Fi is still fickle



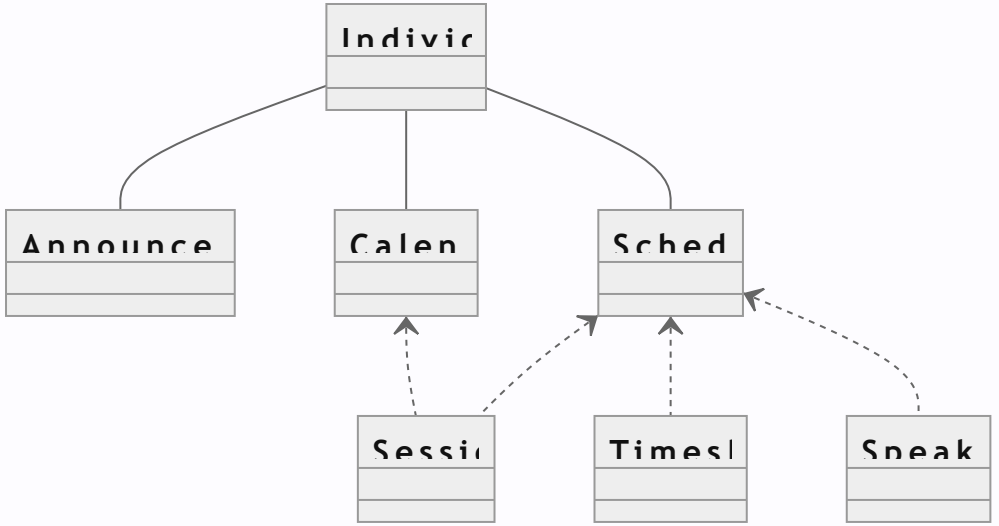
Solution

- Send itinerary as updated when network is available, managing sync

API Design - Conference - Announcements

Requirements

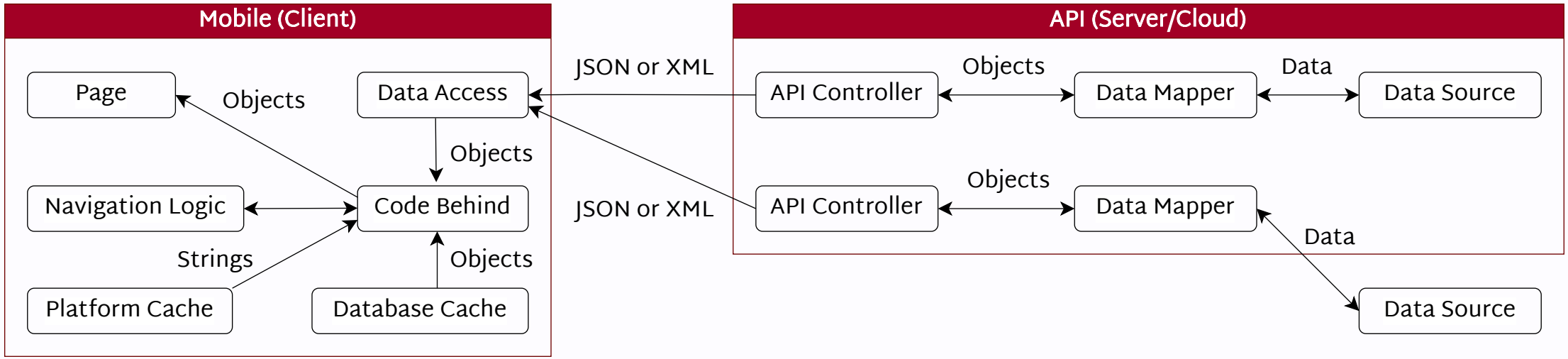
- Everybody gets the same schedule
- Everybody needs to get the same announcements
- Event Wi-Fi is more fickle now that everybody is checking their phones



Solution

- Use a highly-available service
- Poll API or wait for real-time data
- Or hope Slack, Twitter, etc. are up

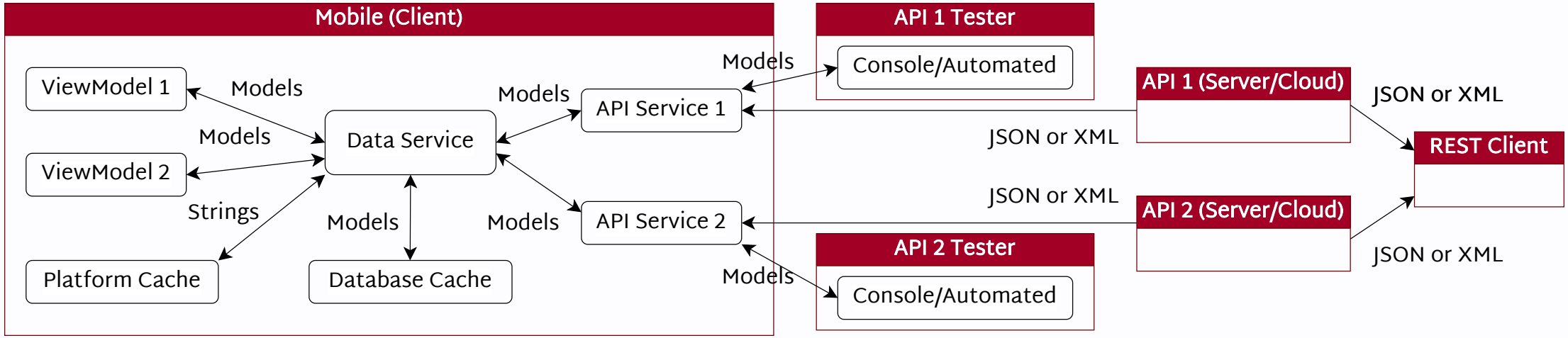
Mobile Architecture - High Level



Mobile Architecture - Real-world

- ViewModels - Moving navigation out of the View allows ViewModels to be tested individually with frameworks like ReactiveUI (RxUI)
- Api Services - Individually testable to return app Models and not just Data Transfer Objects (DTOs)
- Data Service - Decides whether to call the local cache DB, the platform cache, an API endpoint, or a service based on the requested data and the app lifecycle

Mobile Architecture - Real-world



API Design + App Design - Other Things

App Design

- Centralized analytics
- Centralized logging
- Auto-generated API wrappers (Refit)
- Centralized API retry (Polly)
- Platform customizations (Xamarin.Essentials)
- Platform dependency-injection

- And so much more

API Design

- Authorization tokens
- Refresh tokens
- eTags
- Idempotency
- HTTP status codes
- ProblemDetails
- ValidationProblemDetails

Part 4

Explain yourself, dammit

Picture of Lewis Black

ProblemDetails

Pros

- Extensible JSON object
- Provides context to status code
- Can report errors or warnings
- Supported in .NET without ASP.NET
- Returned by Refit's ApiException

Cons

- Buggy in ASP.NET (still)
- .NET implementations follow JSON

```
# NOTE: '\' line wrapping per RFC 8792
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "title": "An RFC 7807 problem object",
  "type": "object",
  "properties": {
    "type": {
      "type": "string",
      "format": "uri-reference",
      "description": "A URI reference that identifies the \
problem type."
    },
    "title": {
      "type": "string",
      "description": "A short, human-readable summary of the \
problem type."
    },
    "status": {
      "type": "integer",
      "description": "The HTTP status code \
generated by the origin server for this occurrence of the problem.",
      "minimum": 100,
      "maximum": 599
    },
    "detail": {
      "type": "string",
      "description": "A human-readable explanation specific to \
this occurrence of the problem."
    },
    "instance": {
      "type": "string",
      "format": "uri-reference",
      "description": "A URI reference that identifies the \
specific occurrence of the problem. It may or may not yield \
further information if dereferenced."
    }
  }
}
```

Demo - ProblemReports (PdHelpers)

Part 5

Questions?