What DBAs Don't Know About Mobile Apps and APIs

Andy Lech

Thank you, Sponsors!

SEMINOLE STATE COLLEGE OF FLORIDA

Microsoft

AMD

COZYROC

ROYAL BLUE ANALYTICS

SQLGrease

O'REILLY

PER SCHOLAS

Join Our Local User Groups:

SQLOrlando

ONE UG Orlando .NET User Group

Topics

- Supporting APIs as a more direct path to your data
- Guaranteeing uniqueness in a stateless API client
- Adding narrower slices of data for just-in-time consumption
- Providing context to database decisions through the API

Part 1

Imagine your house was an app ...

Energy - User

[Web Site or Mobile App User]

- Energy Consumption
  - Just Works
- Energy Production
  - Somebody Else's Problem
- Resiliency
  - Generator (Maybe?)
- **Focus**
  - **Living My Best Life**

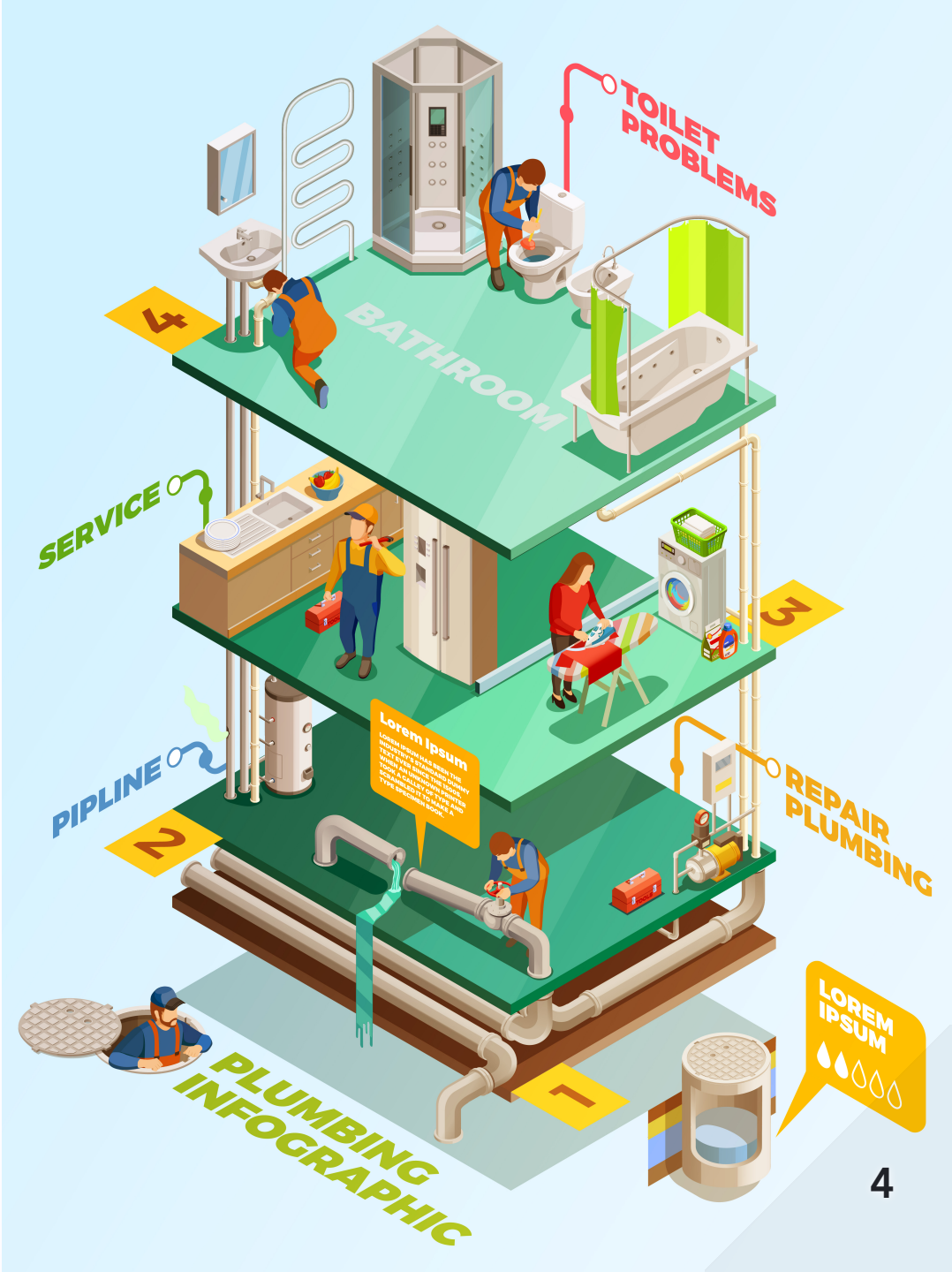
Image by macrovector on Freepik

Energy - Power Grid

[Web Site or Single Page Application Dev]

- User Consumption
  - Always
- Grid Production
  - Our Problem
- Resiliency
  - Our Backups Have Backups (Hopefully)
- **Focus**
  - **Keeping the Lights On (Literally)**

Image by macrovector on Freepik
[Equivalent energy image not available]



Energy - Solar/Wind

[Mobile App Dev]

- Energy Production
  - Local
- Grid Consumption
  - Backup
- Grid Production
  - Their Problem
- Resiliency
  - Battery, Power Grid
- **Focus**
  - **Smartly Using and Storing Energy**

Image by macrovector on Freepik

Andy Lech - What DBAs Don't Know About Mobile Apps and APIs

Part 2

Mobile is just like Web but smaller,

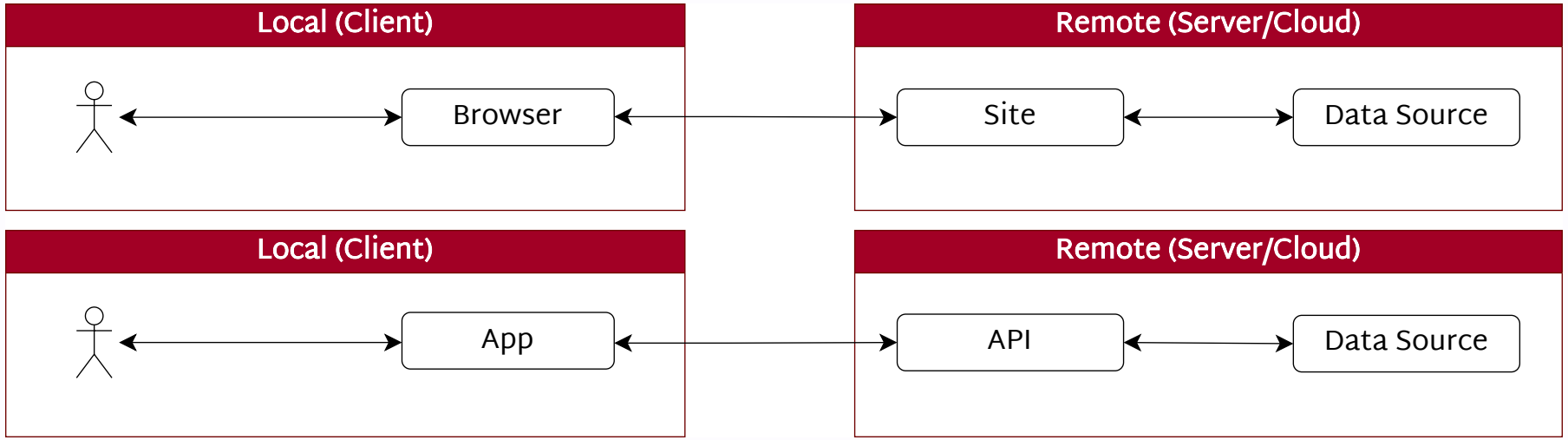
right!?

or

Web devs make bad Mobile devs

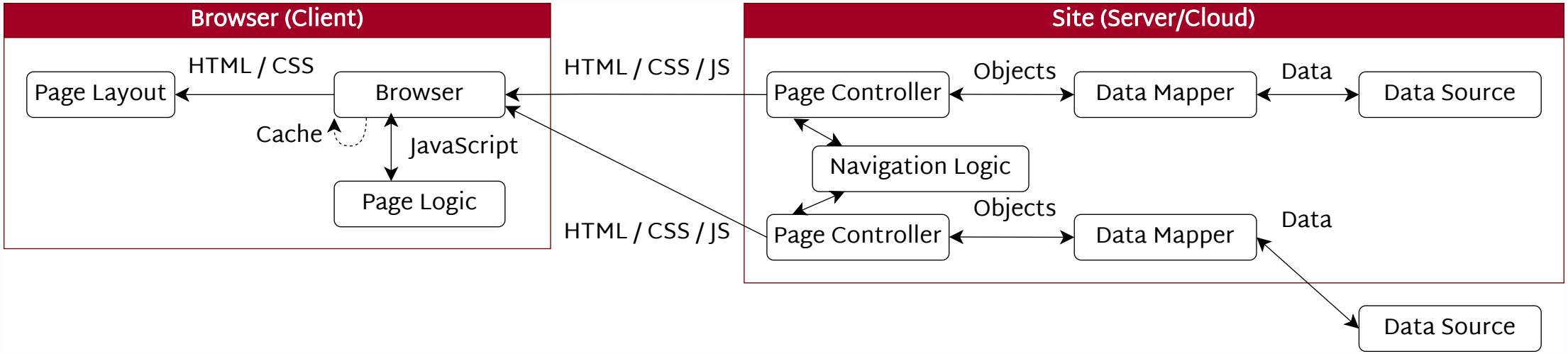
... but they can learn

Web vs Mobile - High Level



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

Web 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
- **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)

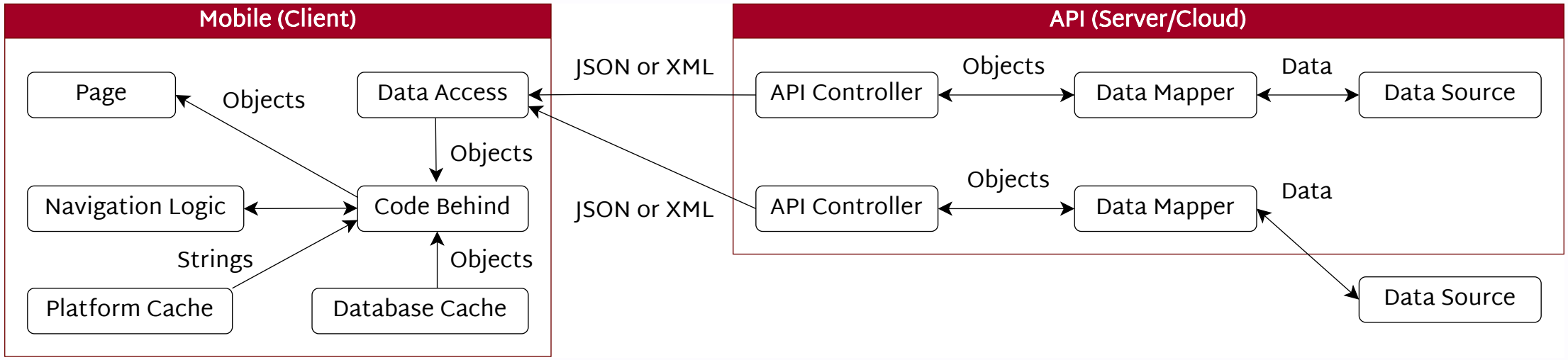
Browser (Client)

- Requests: Browse, Submit
- State Management: Cookies
- Data Mapping: None
- Caching: Local, Session, Cookies
- Resiliency: Browser
- **Dev 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
- **Dev 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
- **Mobile/API devs focus more on reliable, just-in-time data delivery**

Mobile Apps

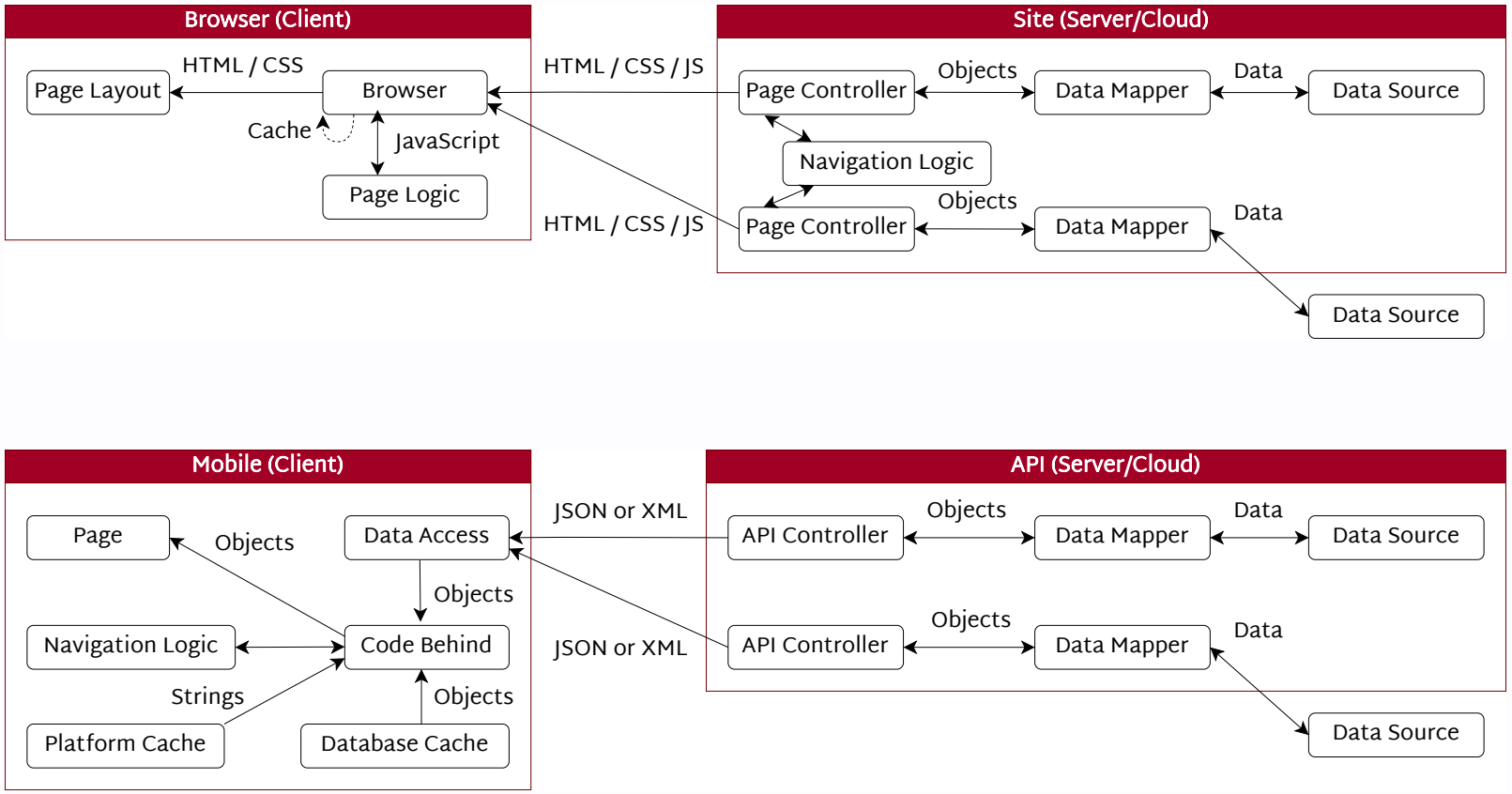
App (Client)

- Requests: API Calls
- State Management: ViewModels, Cache
- Data Mapping: API to App
- Caching: Platform, Local DB
- Resiliency: Network State
- **Dev 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
- **Dev Focus: Data, Status Codes, Messages**
- **Data Goal: Smart Data Pipes (from App)**

Web and Mobile - Two Paths to Data



Part 3

"It's the data, stupid"

[Insert picture of James Carville]

[Insert picture of The War Room whiteboard]

API Design - Operations

- API calls are atomic operations that map closely to CRUD database operations

Create	Read	Update	Delete
Insert	Select	Update	Delete
POST	GET	PATCH PUT	DELETE

- Novice/lazy devs auto-generate API endpoints that map 1:1 to CRUD table operations
- Skilled devs curate API endpoints mapped to consumption patterns and app models
  - Data is often consumed like views of tables targeted to the specific tasks of pages
  - Audit trails don't need to be round-tripped to a consumer that may alter the data

API Design - Responses

- API operations that succeed synchronously have expected data and status codes

Method	Description	Response Status Code
PATCH	Create/Modify the resource with JSON Merge Patch	`200-OK` , `201-Created`
PUT	Create/Replace the *whole* resource	`200-OK` , `201-Created`
POST	Create new resource (ID set by service)	`201-Created` with URL of created resource
POST	Action	`200-OK`
GET	Read (i.e. list) a resource collection	`200-OK`
GET	Read the resource	`200-OK`
DELETE	Remove the resource	`204-No Content` ; avoid `404-Not Found`

Source: Microsoft Azure REST API Guidelines github.com/microsoft/api-guidelines

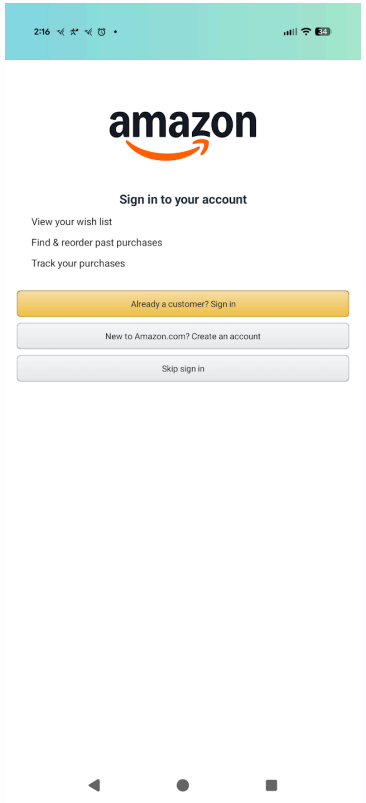
API Design - Stateless State Management

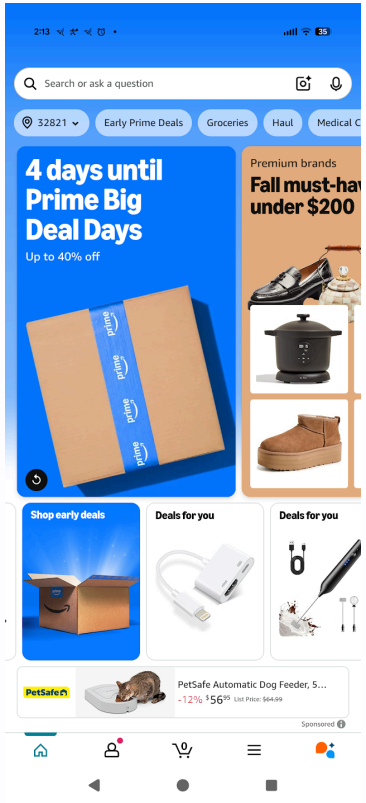
- JSON Web Tokens (JWTs) are commonly used to maintain state between API calls
- API clients get an encoded authentication token which contains the users rights
- API clients returns their tokens on successive calls which are matched to a secret key
- Secret keys are often generated on API server launch which may be tied a DB refresh
- More complex schemas for token generation and API rights and token expiration exist

Source: IETF RFC 7519 - JSON Web Token (JWT) datatracker.ietf.org/doc/html/rfc7519

API Design - Local Cache

- Tokens are often created on login, cached locally, and renewed in the background
- Apps often store the most recent token in fast secure platform-level key:value cache
- If a valid token exists on launch, the app can skip directly to a home page

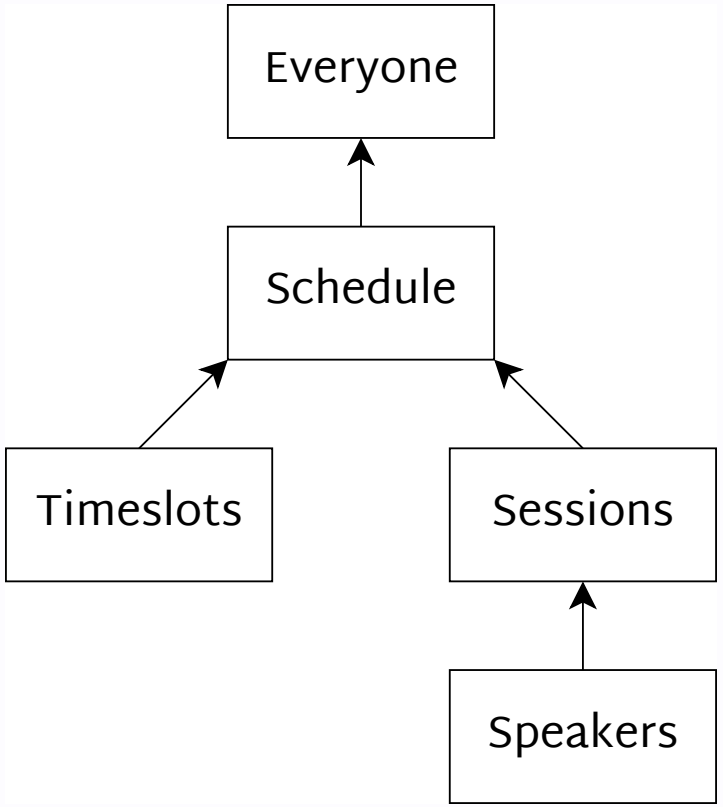




API Design - Conference - Schedule

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 (check for status code `304-Not Modified`)

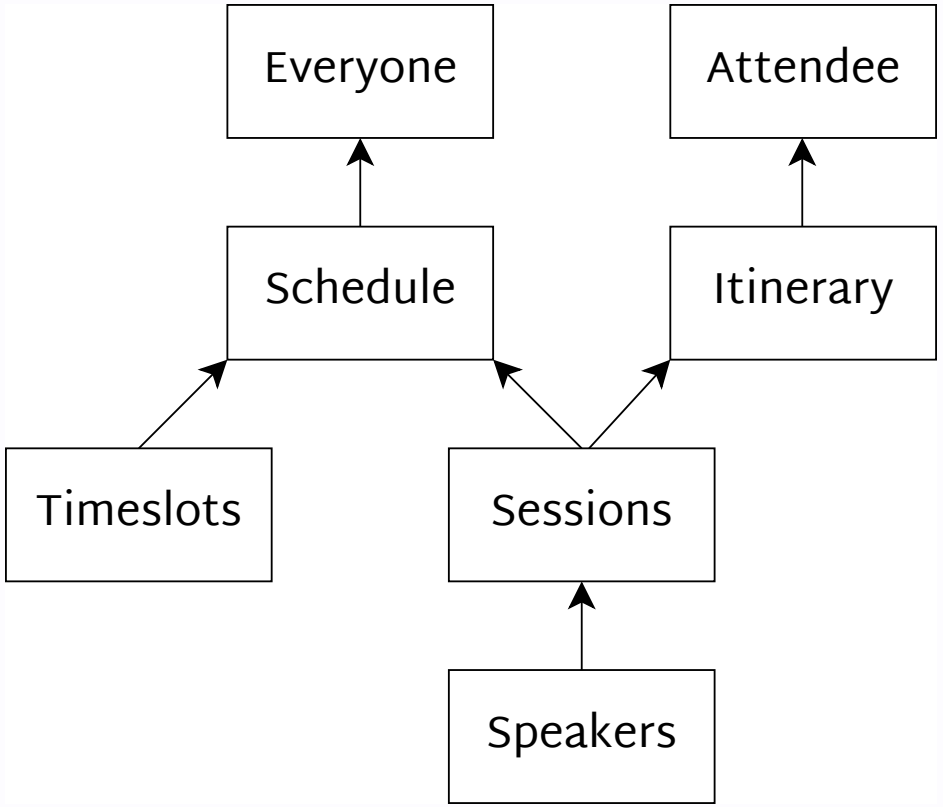
API Design - Conference - Itinerary

Requirements

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

Solution

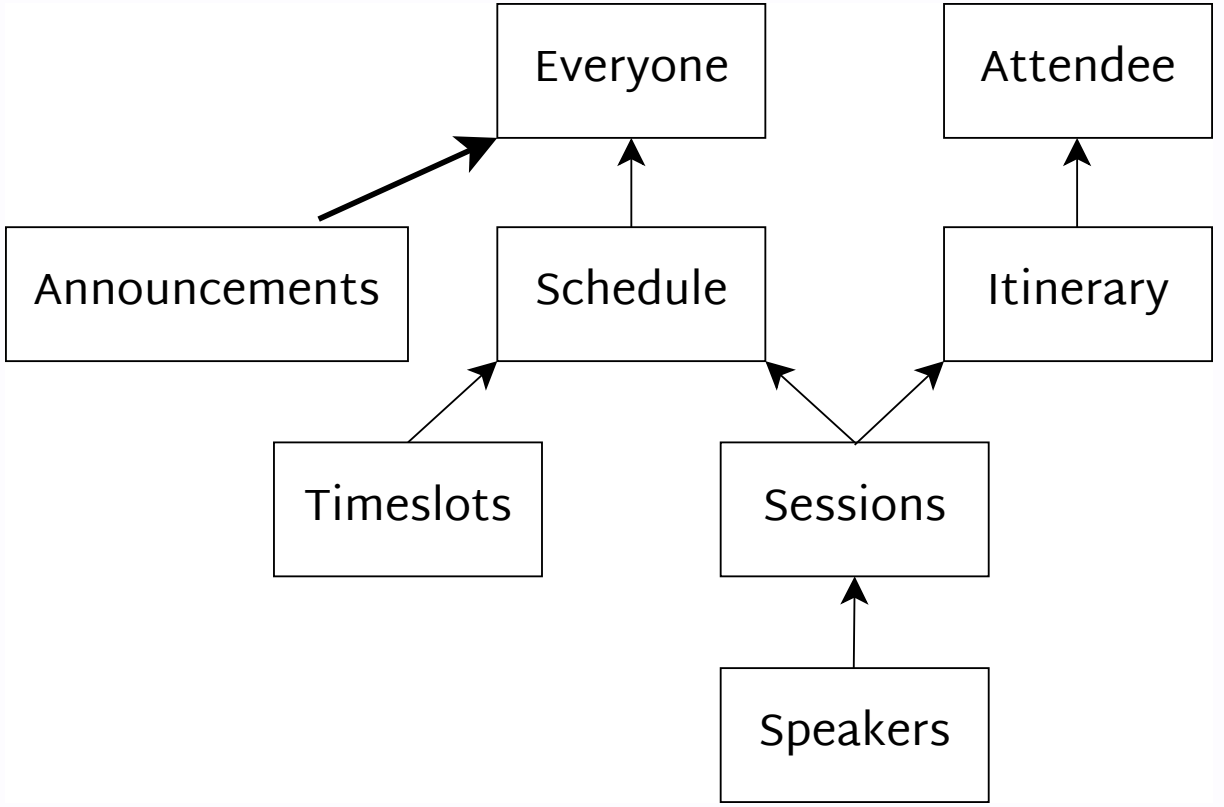
- Send itinerary as updated when network is available, managing sync



API Design - Conference - Announcements

Requirements

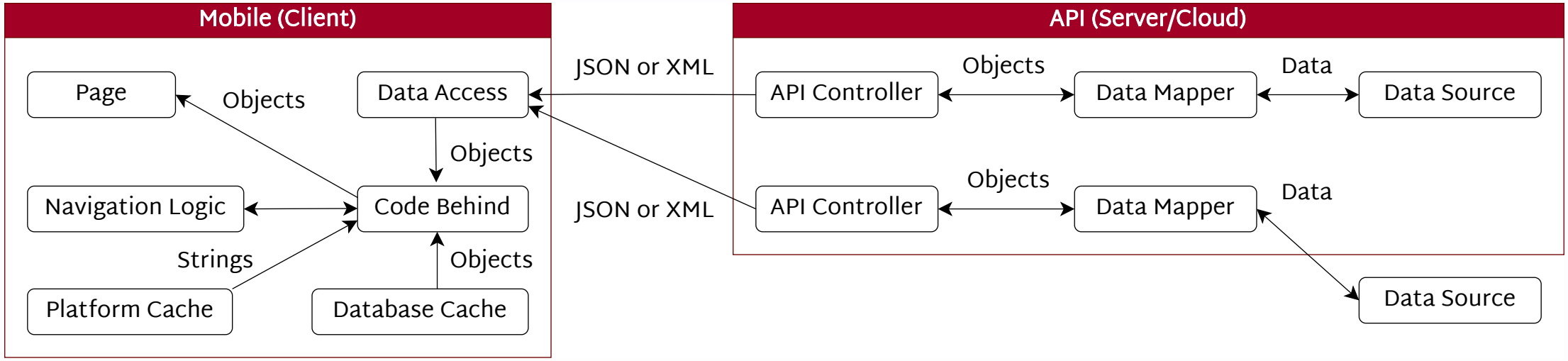
- Everybody gets the same schedule
- Everybody needs to get the same announcements
- Event Wi-Fi is more fickle now that everybody is checking their phones



Solution

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

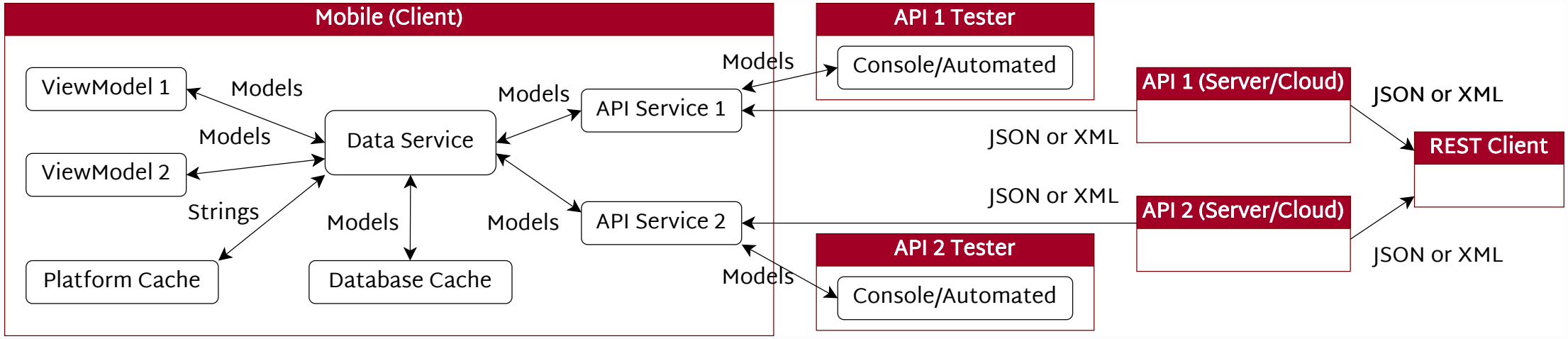
Mobile Architecture - High Level



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

API Design

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

And so much more

Part 4

Explain yourself, dammit

[Insert picture of Lewis Black]

Problem Details

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 return type in ASP.NET
- .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."
    }
  }
}
```

Source: IETF RFC 9457 - Problem Details for HTTP APIs www.rfc-editor.org/rfc/rfc9457.html

Problem Details - Logic Failure Example

```
POST /purchase HTTP/1.1
Host: store.example.com
Content-Type: application/json
Accept: application/json, application/problem+json

{
  "item": 123456,
  "quantity": 2
}
```

```
HTTP/1.1 403 Forbidden
Content-Type: application/problem+json
Content-Language: en

{
 "type": "https://example.com/probs/out-of-credit",
 "title": "You do not have enough credit.",
 "detail": "Your current balance is 30, but that costs 50.",
 "instance": "/account/12345/msgs/abc",
 "balance": 30,
 "accounts": ["/account/12345",
              "/account/67890"]
}
```

Problem Details - Validation Failure Example

```
POST /details HTTP/1.1
Host: account.example.com
Content-Type: application/json
Accept: application/json, application/problem+json

{
  "age": 42.3,
  "profile": {
    "color": "yellow"
  }
}
```

```
HTTP/1.1 422 Unprocessable Content
Content-Type: application/problem+json
Content-Language: en

{
 "type": "https://example.net/validation-error",
 "title": "Your request is not valid.",
 "errors": [
             {
               "detail": "must be a positive integer",
               "pointer": "#/age"
             },
             {
               "detail": "must be 'green', 'red' or 'blue'",
               "pointer": "#/profile/color"
             }
           ]
}
```

Part 5

Questions?

SQLOrlando

Scan the QR code to fill out session evaluations