Taking the REST to go: REST APIs for mobile apps

Joanna Power
AI2
ACT-W 2017

Review of REST API best practices

URIs

- Resource based URI
  - A resource is a noun, e.g. *recipe, appointment, itinerary*
  - Individual resource have IDs
- Resource collections and instances
  - `https://myservice.com/…/<account>/recipes/`
  - `https://myservice.com/…/<account>/recipes/2`

API VERSION

- API version in URI
  - `https://myservice.com/<version>/.../<account>/recipes/`
- Orderable and increase over time
  - Watch out for alphanumeric version strings, e.g. *v1.0, v2.0, v10.1*
  - Personal favorite: *yymm*
  - `https://myservice.com/1709/.../<account>/recipes/`

HTTP METHODS

- POST
  - Create new resource
- GET
  - Access existing resource or resources
- PUT
  - Update existing resource
- DELETE
  - Delete existing resource

HTTP STATUS CODES

- POST
  - 201 Created
- GET
  - 200 OK
- PUT
  - 200 OK
- DELETE
  - 200 OK, 204 No Content

Failed requests

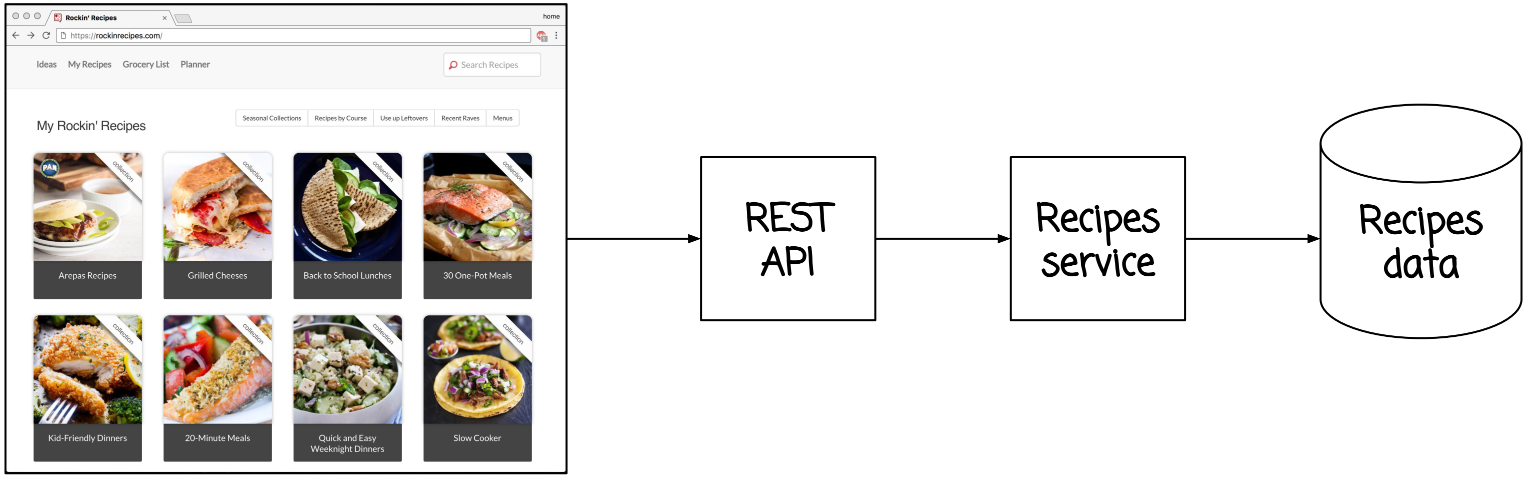
- Non-existent resource
  - 404 Not Found
- Problem with request
  - 400 Bad Request

JSON

- Consistent attribute naming pattern
  - `"start_date" or "startDate"`
  - `"id" or "foo_id"`
- Consistent request and response JSON
- Outer level is always a dict, i.e. wrap lists in a dict
  - `{"items": [{"id": 1, …}, …, {"id": 100, …}]}`

Once upon a time,
there was a cool
web app & REST API.

Rockin' Recipes!



Get recipe

```
GET https://myservice.com/1709/…/recipes/1 -> 200 OK

Response body:

    {"id": 1,
     "name": "Apple Pie",
     "ingredients": ["6 apples", …], …}
```

Create recipe

POST https://myservice.com/1709/…/recipes/ -> 201 Created

Request body:

```
{"name": "Lemon Tart",
 "ingredients": ["3 eggs", …], …}
```

Response body:

```
{"id": 2,
 "name": "Lemon Tart",
 "ingredients": ["3 eggs", …], …}
```

Get recipes

```
GET https://myservice.com/1709/…/recipes/ -> 200 OK

Response body:

    {"items": [

      {"id": 1, "name": "Apple Pie", "ingredients": […], …},

      {"id": 2, "name": "Lemon Tart", "ingredients": […], …}, …]

    }
```

Update recipe

```
PUT https://myservice.com/1709/…/recipes/1 -> 200 OK
```

Request body:

```
    {"name": "Big Apple Pie",
     "ingredients": ["10 apples", …], …}
```

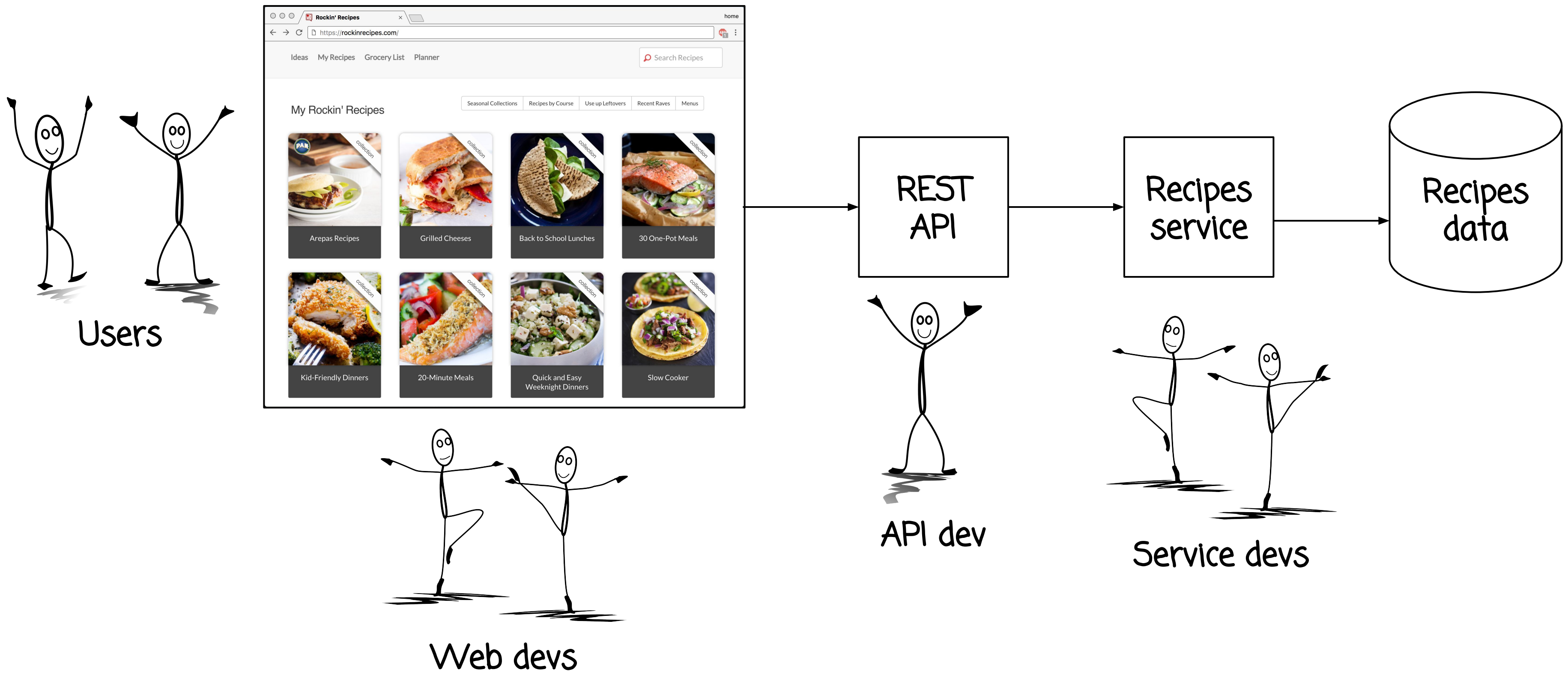
Response body:

```
    {"id": 1,
     "name": "Big Apple Pie",
     "ingredients": ["10 apples", …], …}
```

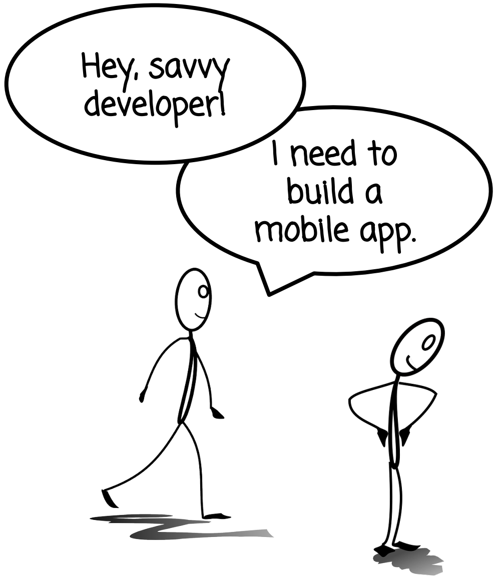
Delete recipe

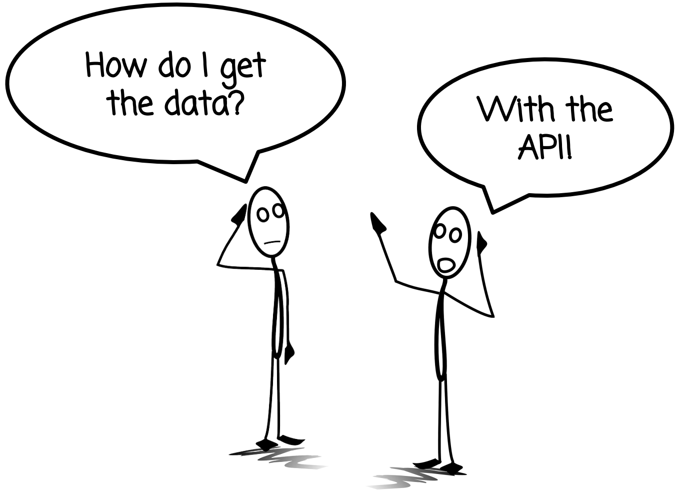
```
DELETE https://myservice.com/1709/…/recipes/2 -> 204 No Content
```

Everyone was happy every day.

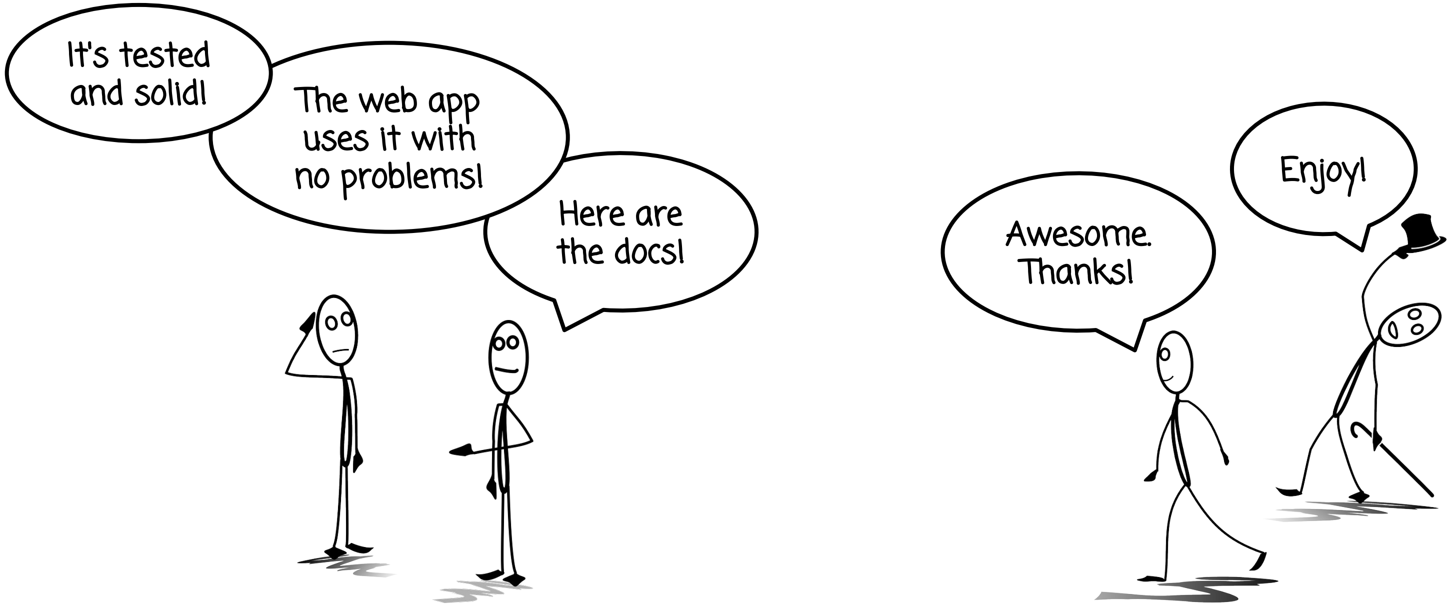


Until one day...

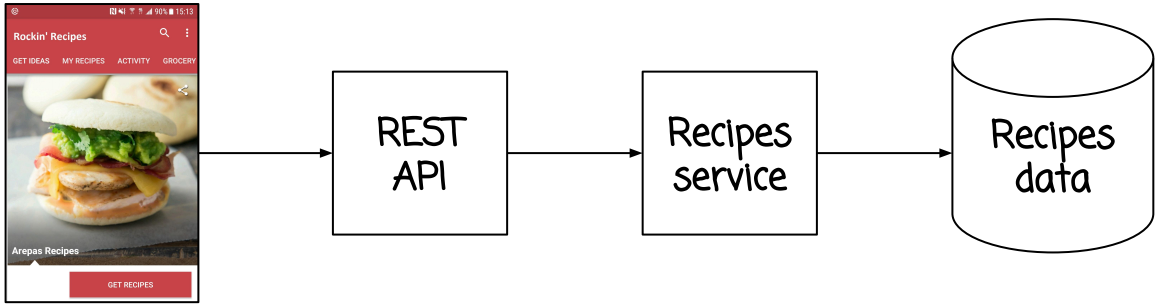




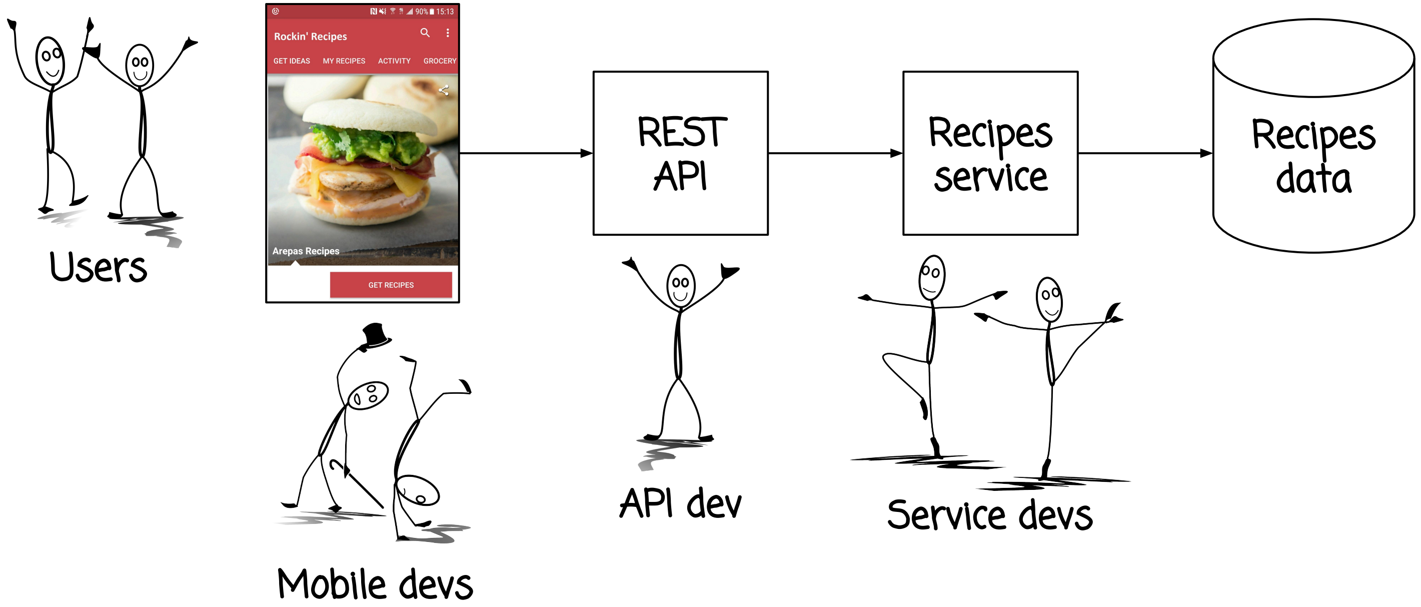
Until one day...



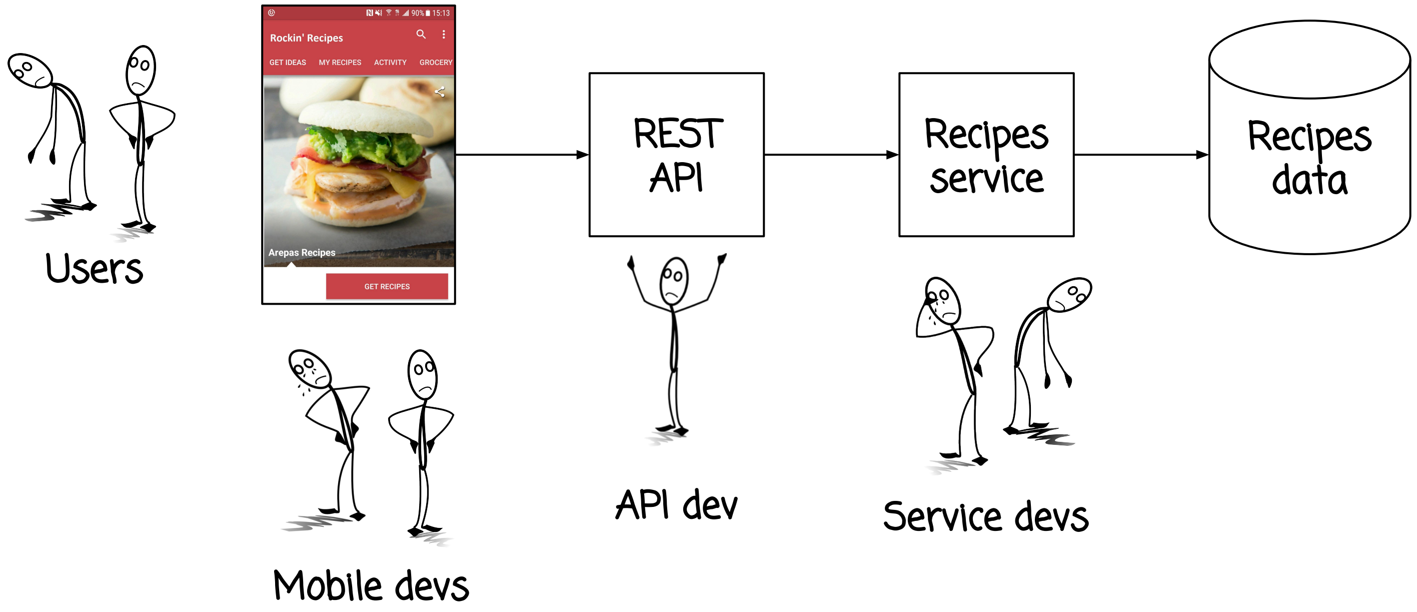
… A mobile app came along.



And it worked!



But it didn't work well. And everyone was sad.



The users were sad.

- Battery life suffered
- Poor performance
- Disappearing data
- Limited offline functionality

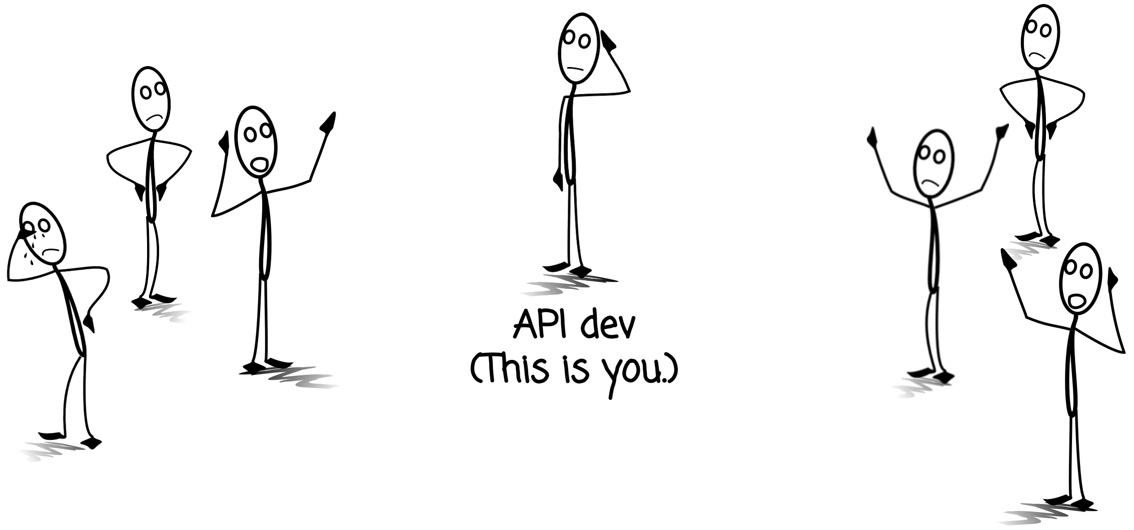
The mobile devs were sad.

- Painful to determine if local data was out of date
- Separate service request for every create, update and delete
- Round-trip for new recipe creation minimized utility of locally cached data
- New API version needed for every resource update

Even the service devs were sad.

- Mobile updates sometimes overwrote data changed by other clients
- Too many versions of the API
- Hard to tell source of problematic API calls
- Impossible to drop support for old API versions

Everyone came to you for help.



REST APIs for Mobile Apps

WHAT'S SO DIFFERENT ABOUT A MOBILE APP?

- Less reliable network connectivity
- Offline scenario increases risk of local data being stale
- Sending and receiving data is relatively power intensive
- No way to change shipped code

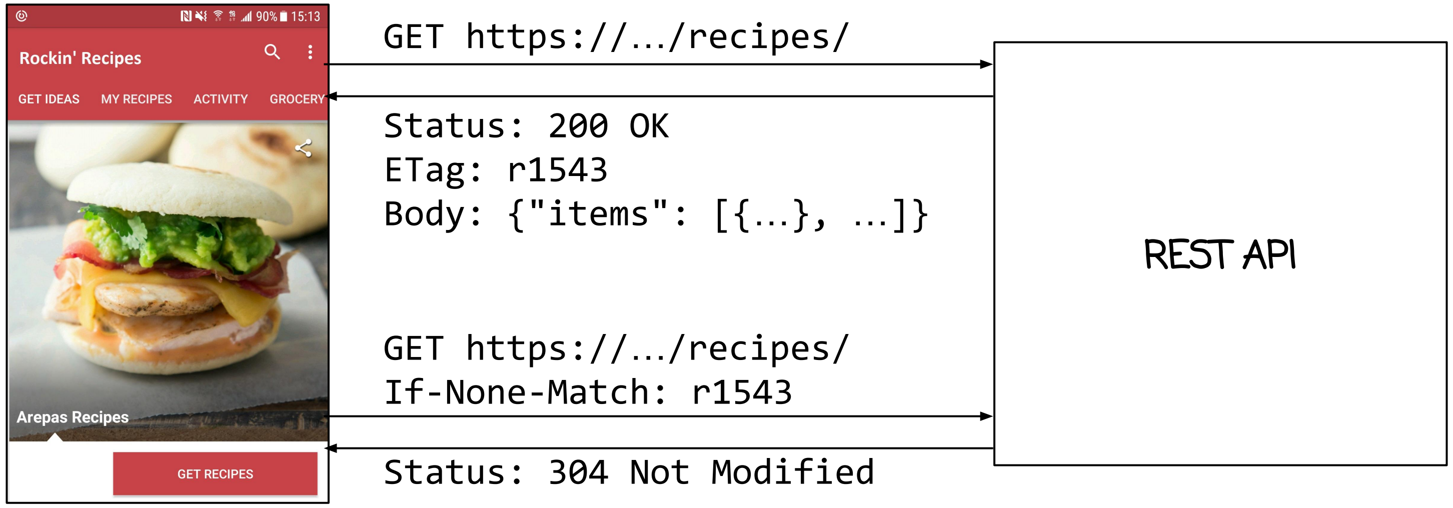
How can I fix it?

- Conditional GET requests
- One request, multiple actions
- Versioned resources
- PUT for resource creation
- Sparse updates
- Client self-identification
- Client status API

Conditional GET requests

- Request for new data if and only if there have been changes
- Implemented using ETags
  - Opaque token generated by service
  - Returned in response header
  - Client includes ETag in next GET request
  - Server responds with `304 Not Modified` if nothing has changed
- Per-request or coarser granularity

Conditional GET requests



```
GET https://…/recipes/

Status: 200 OK
ETag: r1543
Body: {"items": [{…}, …]}

GET https://…/recipes/
If-None-Match: r1543

Status: 304 Not Modified
```

One request, multiple actions

```
POST https://myservice.com/1709/…/recipes/ -> 202 Accepted

Request body:

    [{"action": "create",
      "body": {"name": "Pumpkin Bread", "ingredients": ["2 C flour", …], …}},
     {"action": "update",
      "body": {"id": 1, "name": "Biggest Apple Pie", "ingredients": […], …},
     {"action": "update",
      "body": {"id": 2, "name": "Tangy Lemon Tart", "ingredients": […], …}},
     {"action": "delete", "body": {"id": 2}}]
```

One request, multiple actions

Response body:

```
    [{"status": 201,
      "body": {"id": 3, "name": "Pumpkin Bread", "ingredients": […], …}},
     {"status": 200,
      "body": {"id": 1, "name": "Biggest Apple Pie", "ingredients": […], …}},
     {"status": 404},
     {"status": 204}]
```

Versioned resources

```
POST https://myservice.com/1709/…/recipes/ -> 201 Created
Request body:
    {"name": "Caramel Cake",
     "ingredients": ["1 C butter", …], …}
Response body:
    {"id": 4,
     "name": "Caramel Cake",
     "ingredients": ["1 C butter", …],
     "version": 1, …}
```

Versioned resources

```
PUT https://myservice.com/1709/…/recipes/4 -> 200 OK
Request body:
    {"name": "Caramel Cake",
     "ingredients": ["1 C unsalted butter", …],
     "version": 1}
Response body:
    {"id": 4,
     "name": "Caramel Cake",
     "ingredients": ["1 C unsalted butter", …],
     "version": 2, …}
```

Versioned resources

```
PUT https://myservice.com/1709/…/recipes/4 -> 409 Conflict
Request body:
    {"name": "Grandma's Caramel Cake",
     "ingredients": ["1 C unsalted butter", …],
     "version": 2, …}
Response body:
    {"id": 4,
     "name": "Favorite Caramel Cake",
     "ingredients": ["1 C unsalted butter", …],
     "version": 3, …}
```

PUT FOR RESOURCE CREATION

```
PUT https://myservice.com/1709/…/recipes/590a6… -> 201 Created
Request body:
    {"name": "Bread Pudding",
     "ingredients": ["2 C milk", …], …}
Response body:
    {"id": "590a658f-6e61-492d-b9ee-f3e1fbfce549",
     "name": "Bread Pudding",
     "ingredients": ["2 C milk", …],
     "version": 1, …}
```

Sparse updates - The problem

```
GET https://myservice.com/1709/…/recipes/590a6… -> 200 OK
Response body:
    {"id": "590a658f-6e61-492d-b9ee-f3e1fbfce549",
     "name": "Bread Pudding",
     "ingredients": ["2 C milk", …],
     "rating": 4.5,
     "version": 1, …}
```

Sparse updates - the problem

```
PUT https://myservice.com/1709/…/recipes/590a6… -> 200 OK

Request body:
    {"name": "Bread Pudding",
     "ingredients": ["2 C whole milk", …],
     "version": 1, …}

Response body:
    {"id": "590a658f-6e61-492d-b9ee-f3e1fbfce549",
     "name": "Bread Pudding",
     "ingredients": ["2 C whole milk", …],
     "rating": null, …}
```

Sparse updates - the fix

```
PUT https://myservice.com/1709/…/recipes/590a6… -> 200 OK
Request body:
    {"ingredients": ["2 C whole milk", …],
     "version: 1}
Response body:
    {"id": "590a658f-6e61-492d-b9ee-f3e1fbfce549",
    "name": "Bread Pudding",
    "ingredients": ["2 C whole milk", …],
    "rating": 4.5,
    "version": 2, …}
```

Client self-identification

- Include device model, os version, app build number, and release tag
  - `e.g. SM-G930T|7.0|9593f1c|1709`
- Custom request header
  - `x-rr-client-id: SM-G930T|7.0|9593f1c|1709`

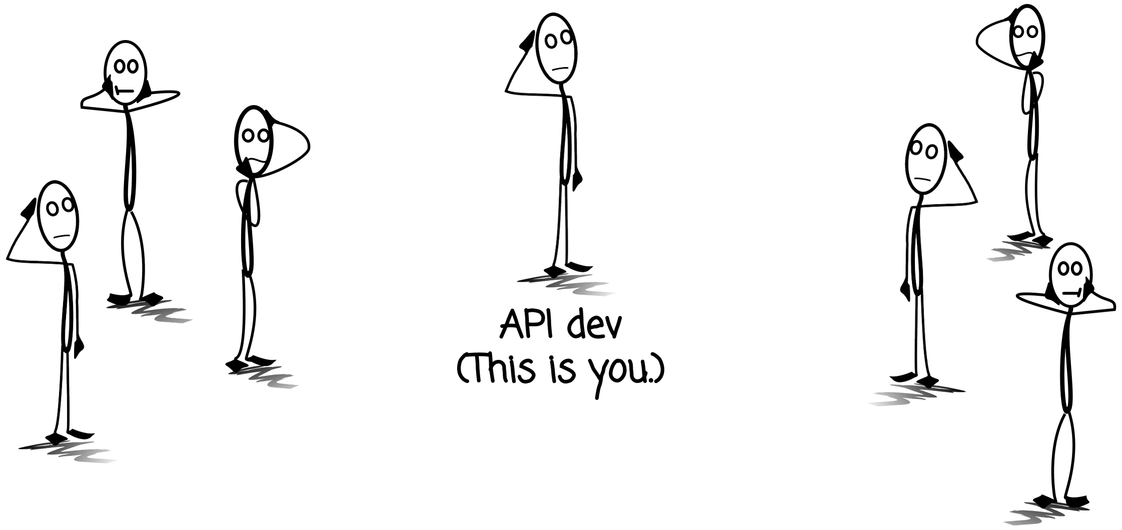
Client status API

```
GET https://myservice.com/1709/…/recipes/client-info -> 200 OK

Response body:

    {"status": ["supported", "unsupported", "must upgrade", "eol"],
      "msg": …}
```

So... did it work?



User complaints

- Battery life suffered
  - Conditional GET requests
  - One request, multiple actions
  - Sparse updates
- Poor performance
  - Conditional GET requests
  - One request, multiple actions
  - PUT for resource creation

User complaints

- Disappearing data
  - Versioned resources
  - Sparse updates
- Limited offline functionality
  - PUT for resource creation

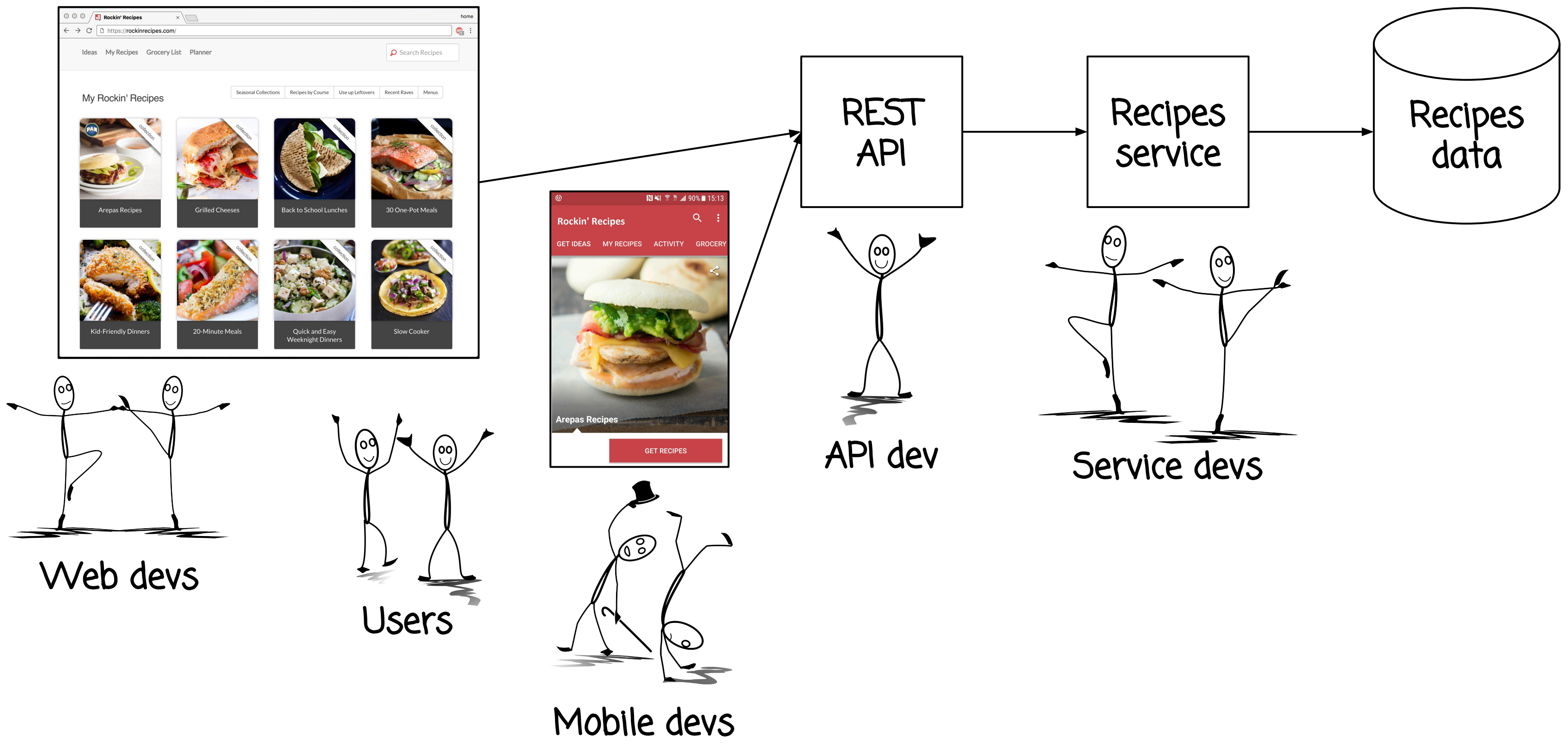
Mobile dev complaints

- Painful to determine if local data was out of date
  - Conditional GET requests
- Separate service request for every create, update and delete
  - One request, multiple actions
- Round-trip for new recipe creation minimized utility of locally cached data
  - PUT for resource creation
- New API version needed for every resource update
  - Sparse updates

Service dev complaints

- Mobile updates sometimes overwrote data changed by other clients
    - Versioned resources, sparse updates
- Too many versions of the API
    - Sparse updates
- Hard to tell source of problematic API calls
    - Client self-identification
- Impossible to drop support for old API versions
    - Client status API

And they all lived happily ever after.


Rockin' Recipes
home
https://rockinrecipes.com/
Ideas
My Recipes
Grocery List
Planner
Search Recipes
My Rockin' Recipes
Seasonal Collections
Recipes by Course
Use up Leftovers
Recent Raves
Menus
Arepas Recipes
Grilled Cheeses
Back to School Lunches
30 One-Pot Meals
Kid-Friendly Dinners
20-Minute Meals
Quick and Easy Weeknight Dinners
Slow Cooker
Rockin' Recipes
GET IDEAS
MY RECIPES
ACTIVITY
GROCERY
Arepas Recipes
GET RECIPES
REST API
Recipes service
Recipes data
Web devs
Users
Mobile devs
API dev
Service devs