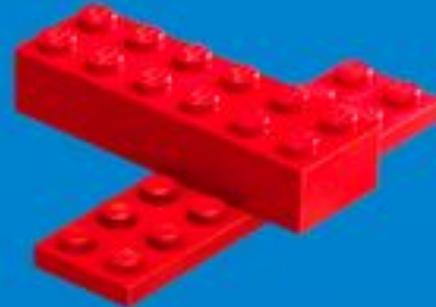
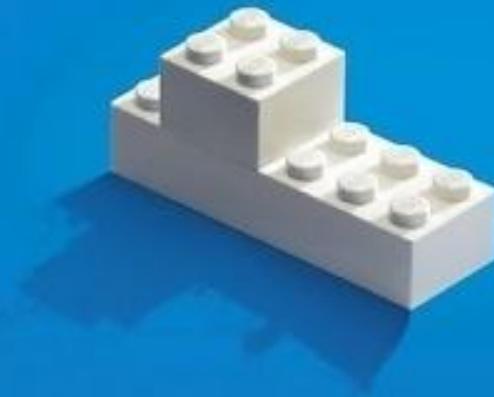


A quest for simplicity



From depths of IS to heights of API

Arnaud Lauret
@apihandyman

AXA Banque

One does not simply start a quest without a goal

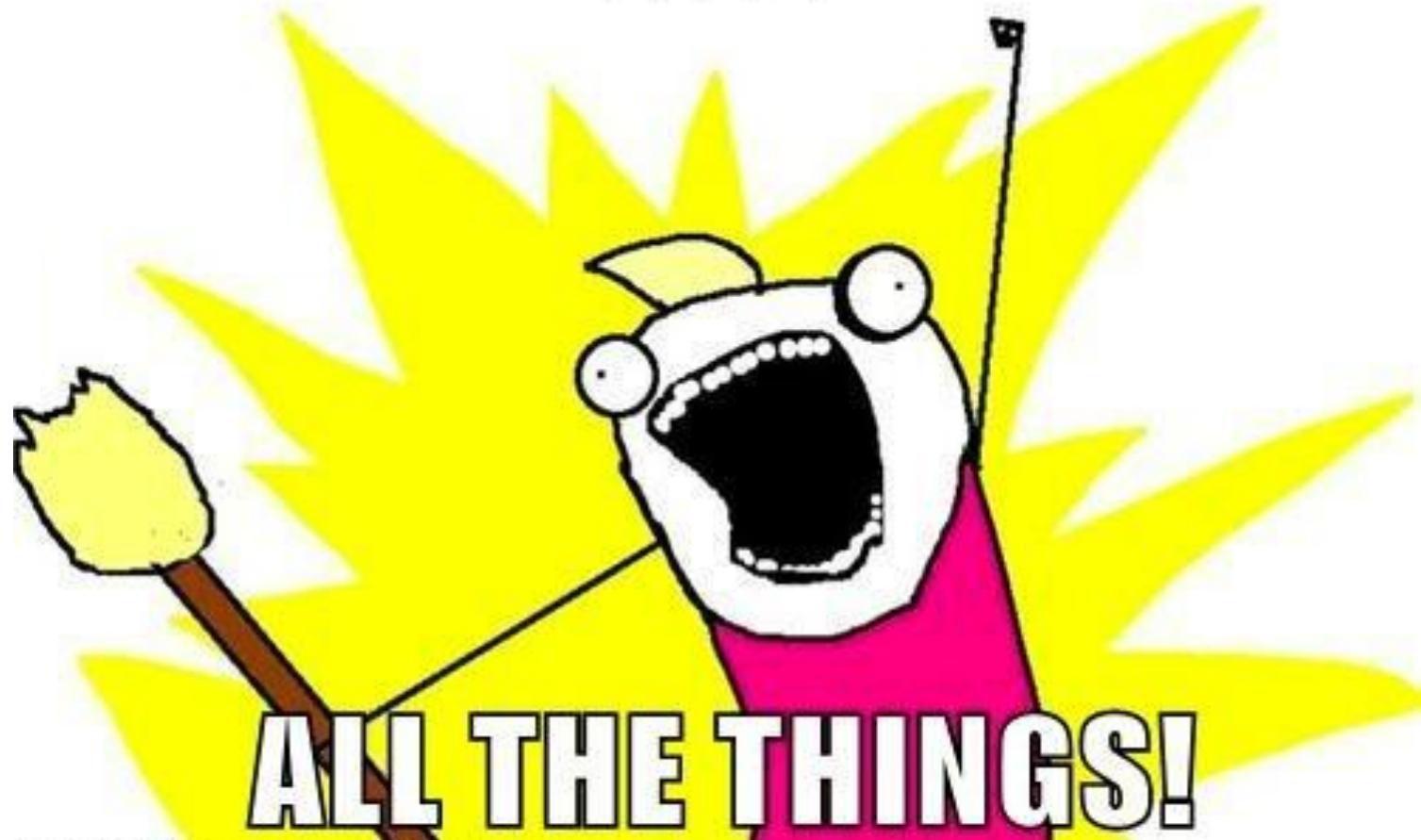
•••

The events depicted in this talk are fictitious. Any similarity to any information system living or dead is merely coincidental.



What is your quest?

API



ALL THE THINGS!

A knight in full chainmail armor stands in a misty, medieval setting, facing a large wooden cage. Inside the cage, a bearded man with long, wild hair and a fur-trimmed coat looks out with a weary expression. The background is filled with smoke and the faint glow of fire.

What is your favorite word in API?

A scene from the movie Monty Python's Life of Brian. In the foreground, two men are looking upwards with expressions of surprise or awe. The man on the left has long hair and is wearing a brown vest over a light-colored tunic. The man on the right is wearing a dark cap and a patterned tunic. They are surrounded by other people in period clothing, some wearing hats. The background is dark and textured.

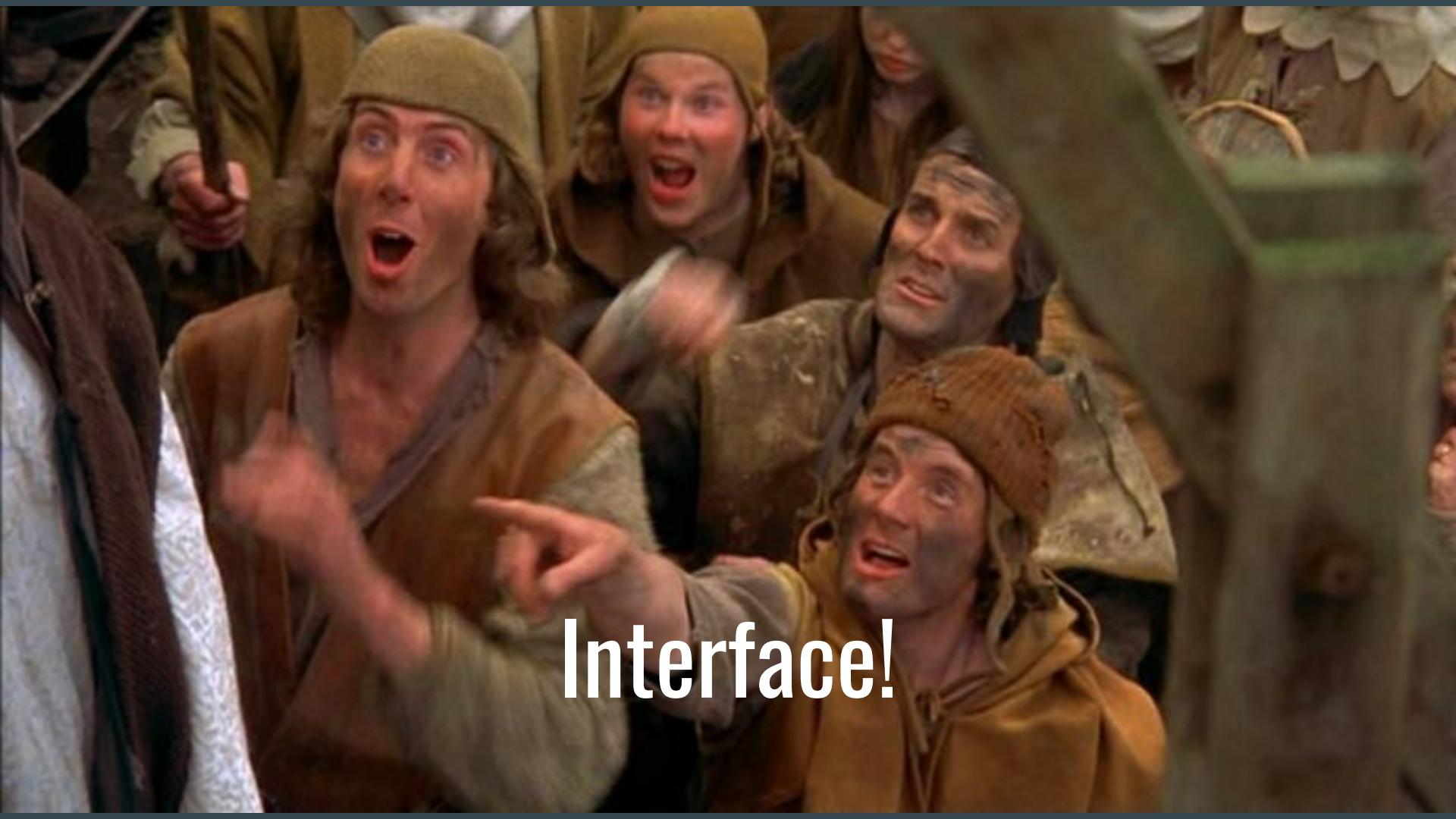
Application?

A group of soldiers in a dark, cramped space, looking up with concern.

Programming?

A scene from a movie or TV show set in a medieval or fantasy world. In the foreground, a man with long brown hair and a beard, wearing a brown tunic, looks off-camera with a surprised expression. Behind him, another man in a brown tunic and a grey cap is pointing his finger towards the same direction. In the background, a woman with long brown hair and a man in a green tunic are also looking in the same direction. The setting appears to be a crowded indoor space with wooden beams and decorations.

Interface?

A group of people in medieval-style clothing, including tunics and hats, are looking up with expressions of surprise or awe. One man in the foreground points upwards with his right hand.

Interface!

The place at which **independent** and often

unrelated systems meet and

interact with each other



POST  https://getpocket.com/v3/send    



Body

Cookies

Headers (17)

Tests (0/0)

Status

200 OK

Time

6689 ms

 Pretty Raw Preview JSON

```
1  {
2    "action_results": [
3      {
4        "item_id": "806114722",
5        "normal_url": "http://apihandyman.io",
6        "resolved_id": "806114722",
7        "extended_item_id": "806114722",
8        "resolved_url": "http://apihandyman.io",
9        "domain_id": "22272648",
10       "origin_domain_id": "22272648",
11       "response_code": "200",
12       "mime_type": "text/html",
13       "content_length": "4147"
```

What have they
in common?

Us



People

(Even APIs)

API = UI

for people building programs

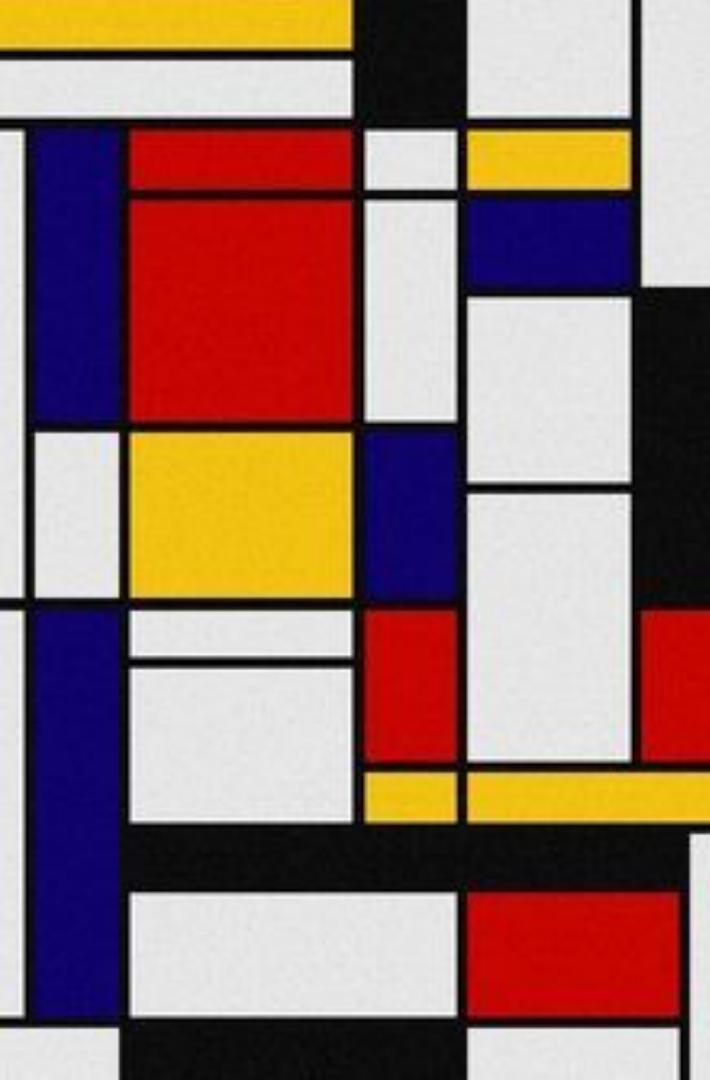
What's happening behind an interface?

The microwave oven parable



Abstraction

A picture is worth a thousand words



What do we seek when using an interface
(especially an API) ?

Simplicity



Easy to use



Error 1543

Easy to understand

Missing
email

**THE
ULTIMATE
SOUND OF MOOG.**

The sound of the Minimoog in a polyphonic programmable instrument. The Memorymoog, from Moog. 3 oscillators per voice through the patented Moog filter, 75 programs with 10 program chains, programmable foot pedals, return-to-zero or unconditional contours and extensive voice modulation options give the Memorymoog more sound and musical expressiveness than any instrument of its kind. Only Moog engineering could create this instrument; only Moog manufacturing could produce it at such an affordable price. The Memorymoog. From Moog.

The people who started it all. memorymoog

Moog Music Inc., 2500 Walden Avenue, Buffalo, NY 14225
Moog Music, Waalhaven Z.Z. 48, Rotterdam 3088 H.J., Holland - The Netherlands

bouoh 贝优汇

让宝宝尽情的展现自己的音乐才华

大嘴猫钢琴

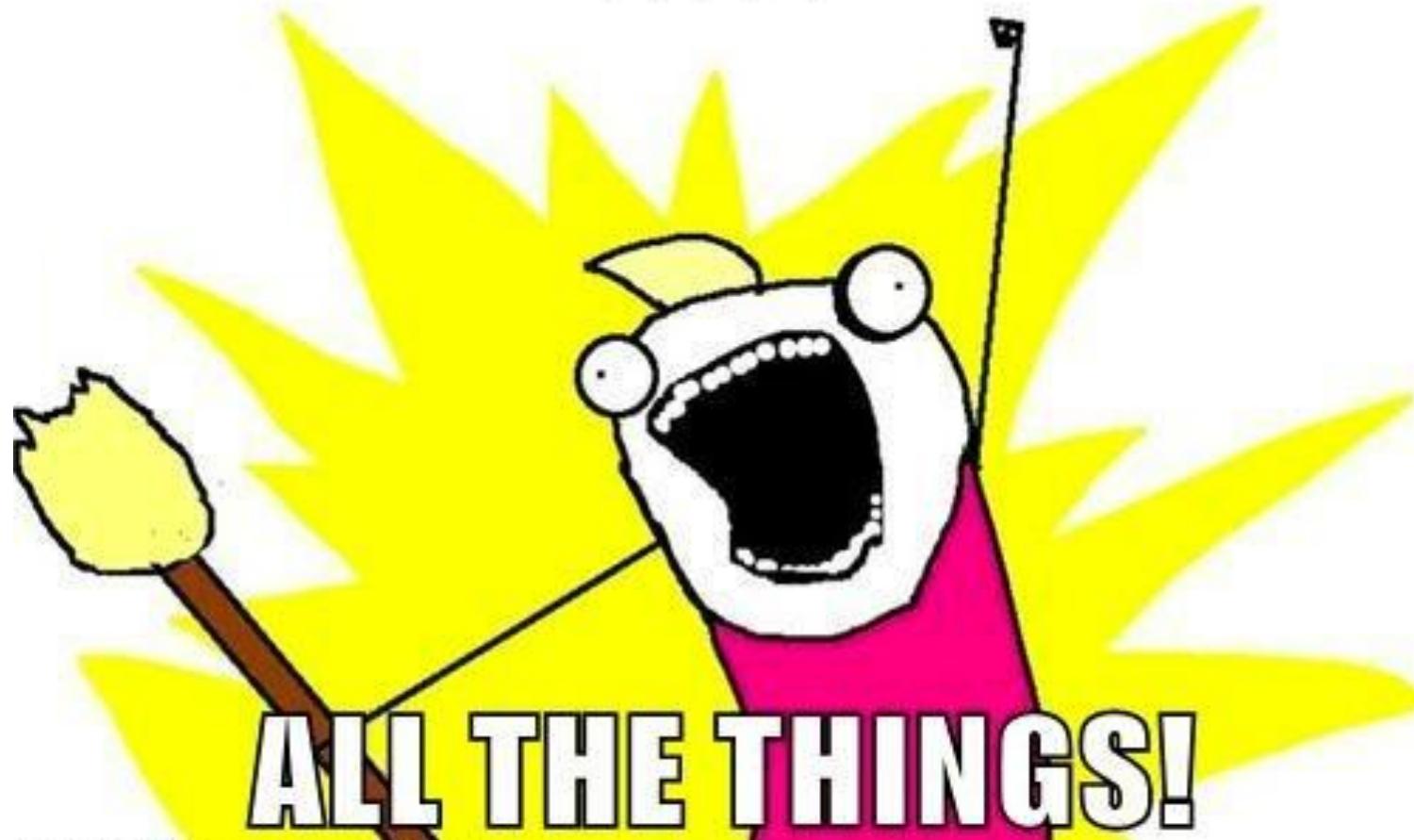
Adapted to the targeted audience

A young girl with a blue headband is singing into a purple microphone. She is holding a large, colorful piano keyboard shaped like a cat's mouth. The piano has white keys and black keys, and features a yellow body with orange stripes, a large eye, and a wide mouth. Musical notes are floating around the piano. The background is light blue.

A knight in full chainmail armor stands on the left, facing a bearded man with long, wild hair who is trapped in a wooden cage. The knight's armor is dark with a white shoulder guard and red accents on the belt. The man in the cage has a long, shaggy beard and is wearing a dark fur-trimmed coat. They are outdoors in a misty, overcast environment.

So, what is our quest?

~~API~~ Simplify



Core Banking System

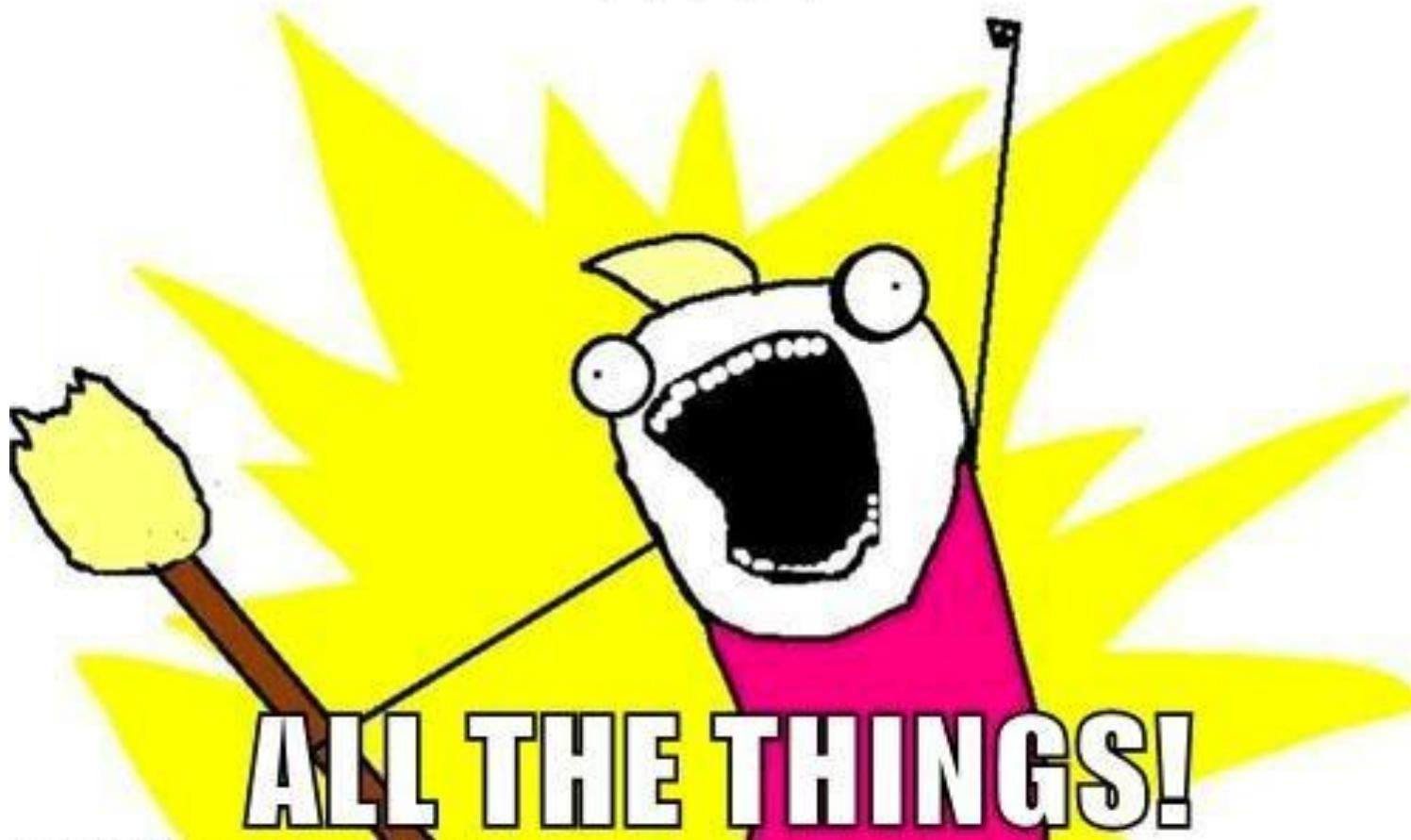
• • •

A long time ago...



This is an interface

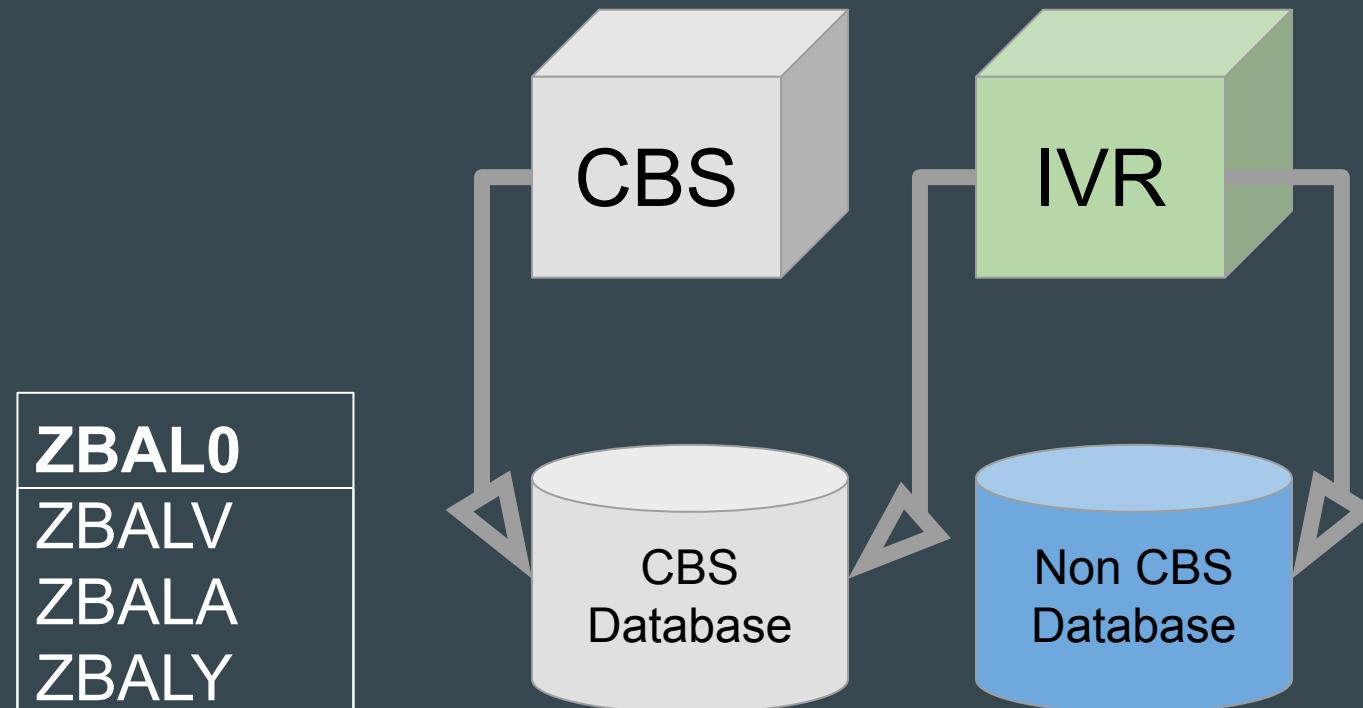
~~API~~ Access



IVR



Interactive Voice Response



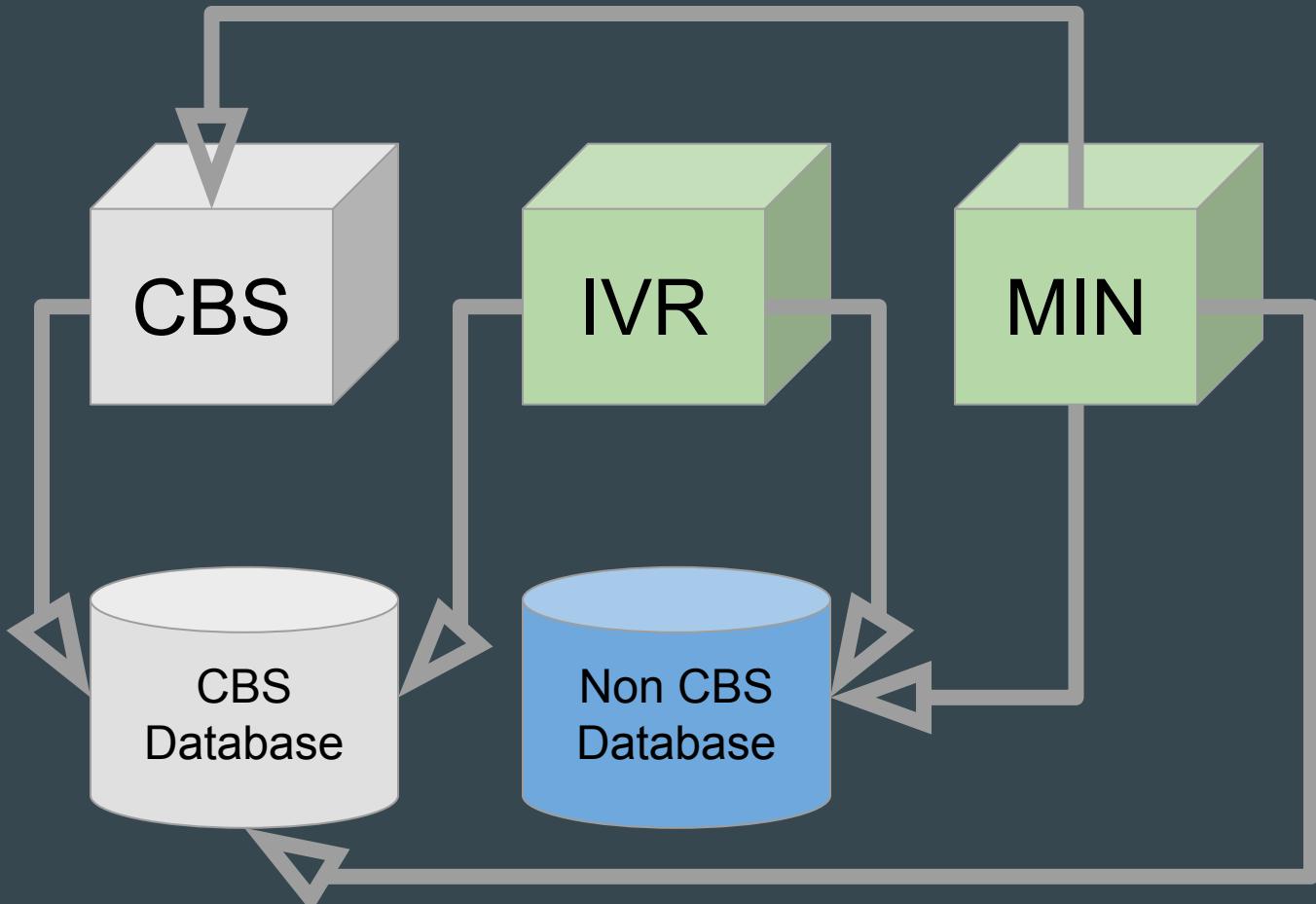
FWW



Minitel

MQ Message:

F1202167754151912540123777463779FRF0300020160917



Is interfacing with CBS simple?

- Is it easy to use?
- Is it easy to understand?
- Is the abstraction adapted to the audience?

INTE PROT MEMR INP MI OUT HLTA STACK WO INT

STATUS

WAIT HLDA A15 A14 A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0

SENSE SW.

So, it's not simple

STOP
RIN

SINGLE
STEP

EXAMINE
EXAMINE
NEXT

DEPOSIT
DEPOSIT
NEXT

RESET
CLR

PROTECT
UNPROTECT

AUX

AUX



ALTAIR 8800 COMPUTER

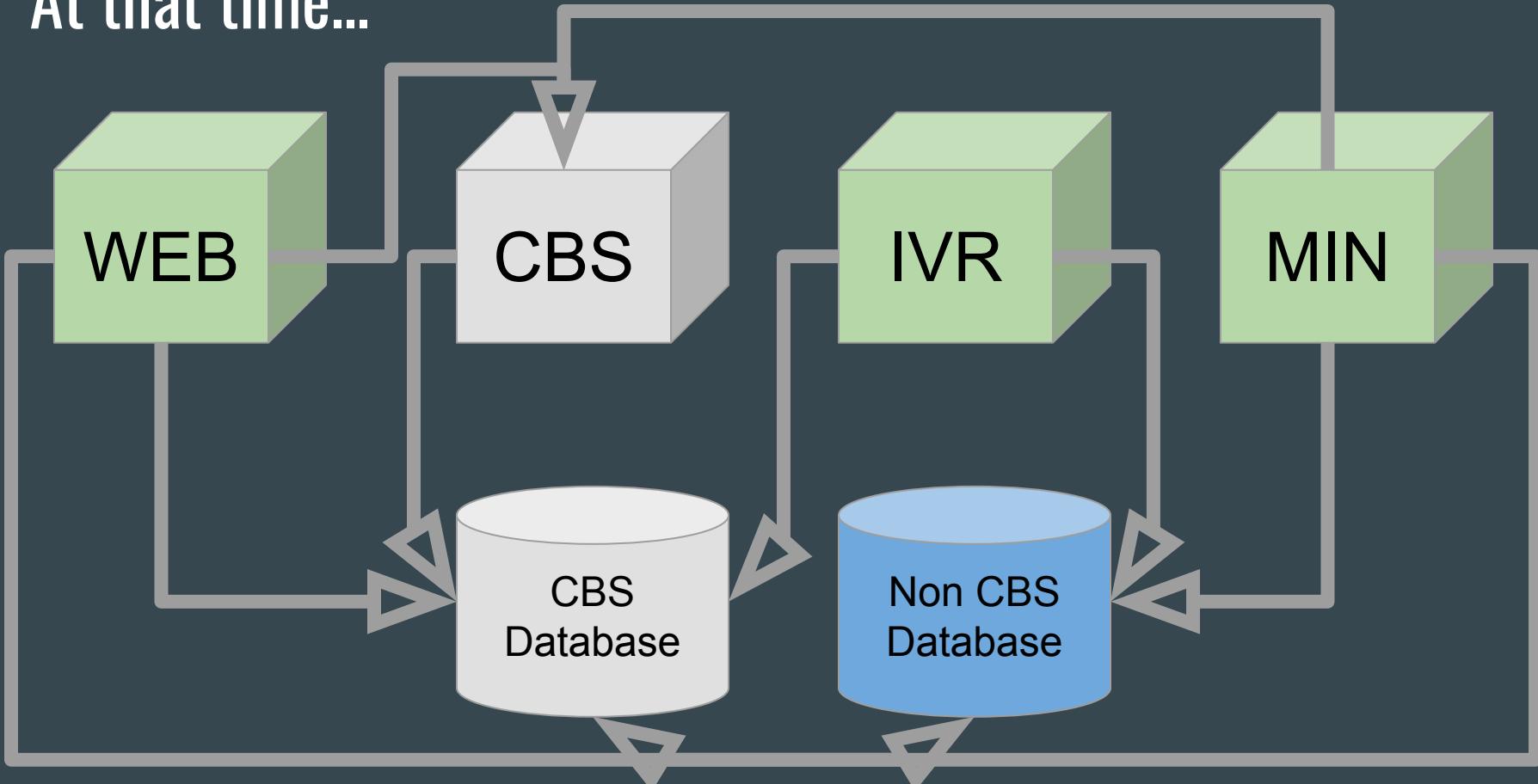
Web services

• • •

A few years later...

This is a dinosaur

At that time...



SOAP



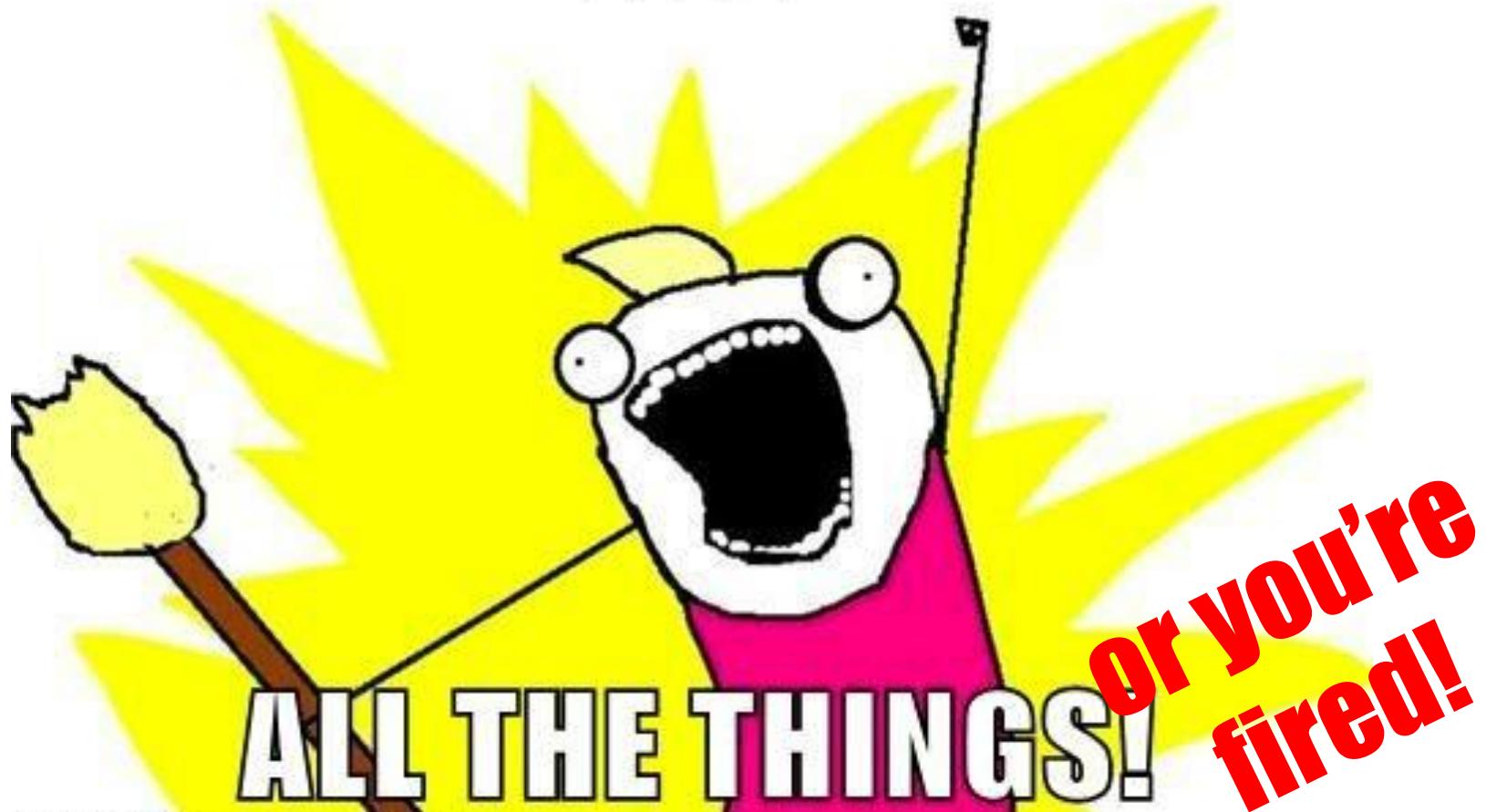
SOA Principles

- Service
- Loose coupling
- Reusability

SOAP Protocol

- Use HTTP as a transport protocol
- XML based
- Input message contains the action to trigger and the data

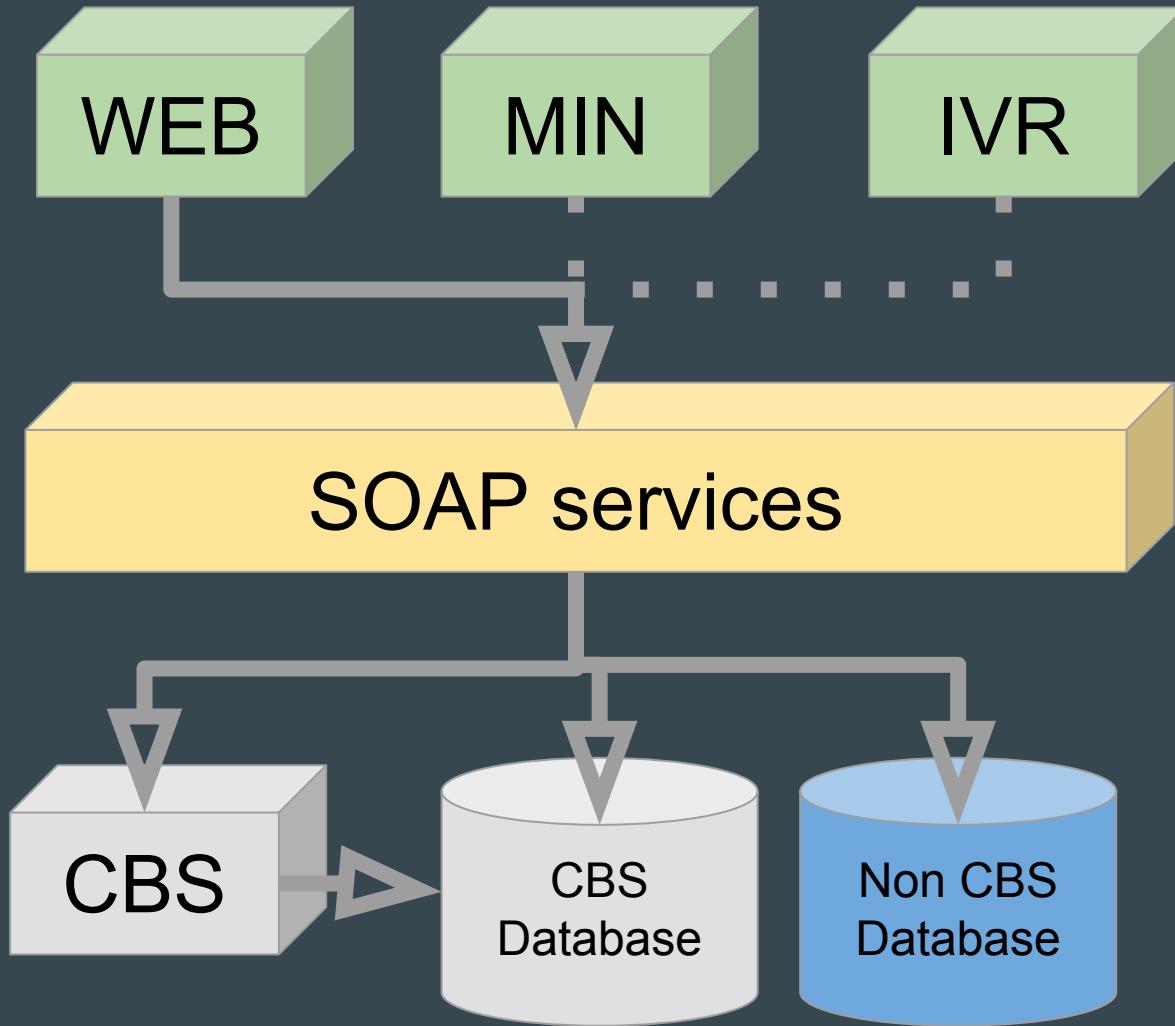
~~API~~ Service



ALL THE THINGS!

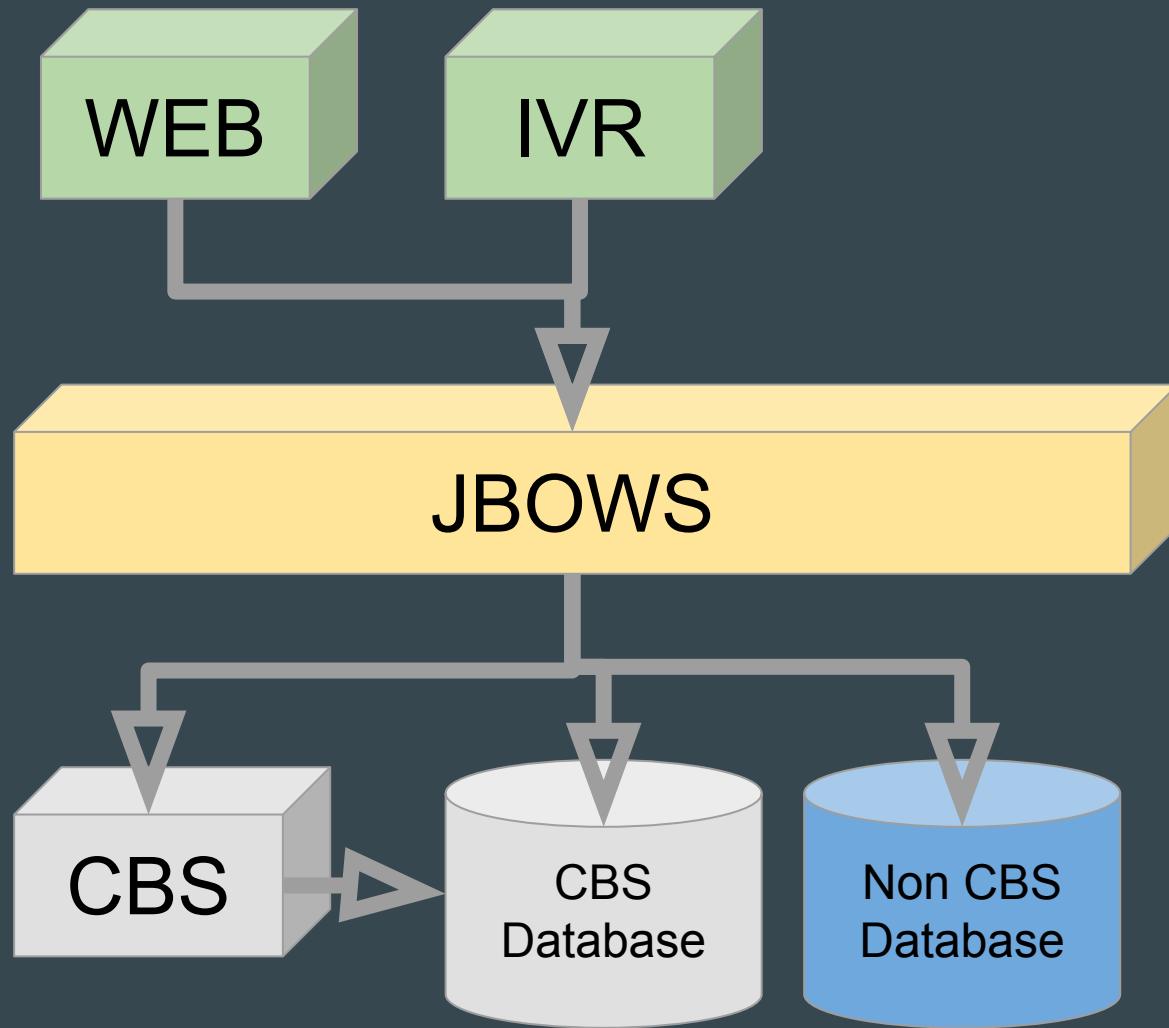
or you're
fired!

SOA(P)



Several years later

SOA(P)



Is interfacing with SOA simple?

- Is it easy to use?
- Is it easy to understand?
- Is the abstraction adapted to the audience?



Better but not awesome...

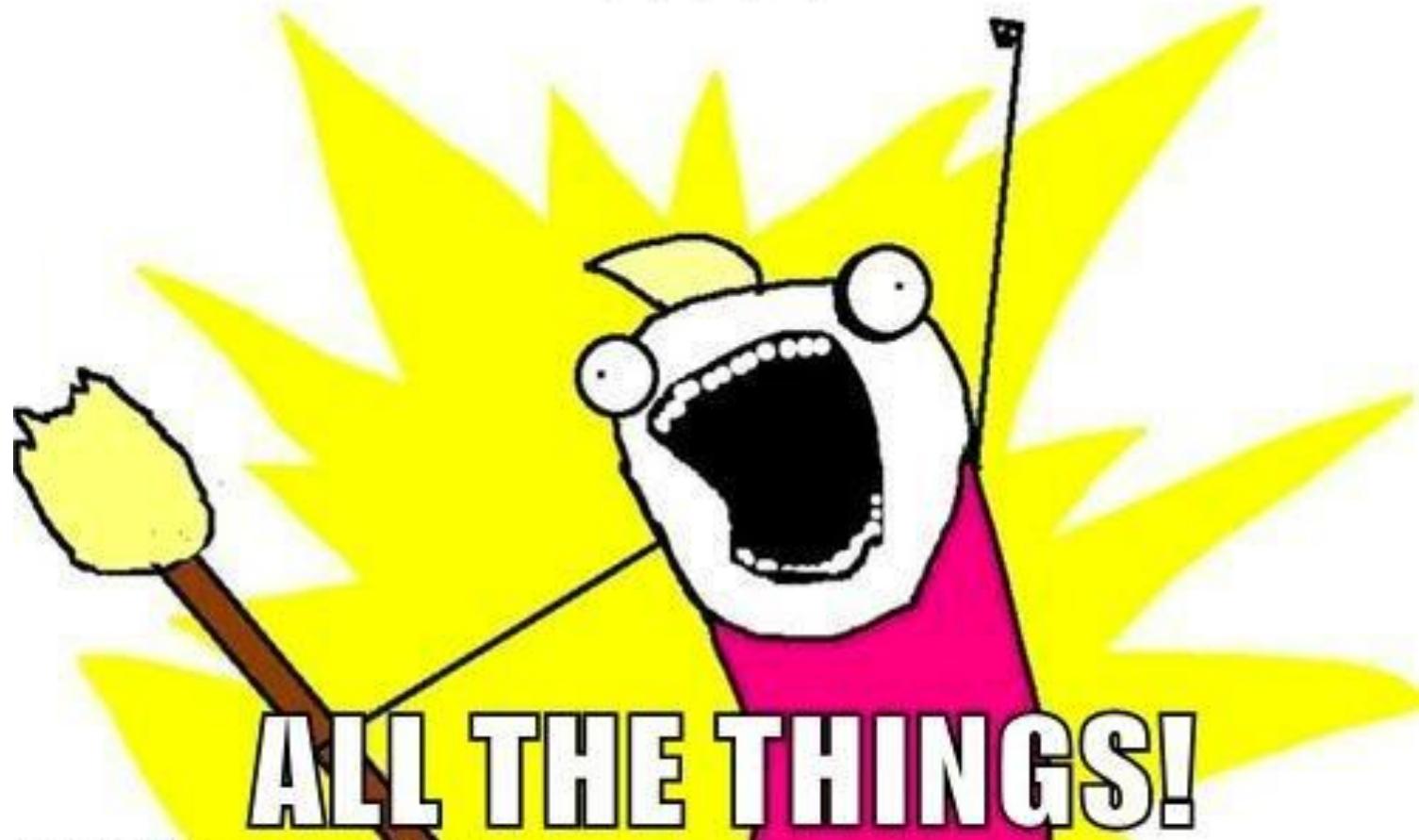
APIs

• • •

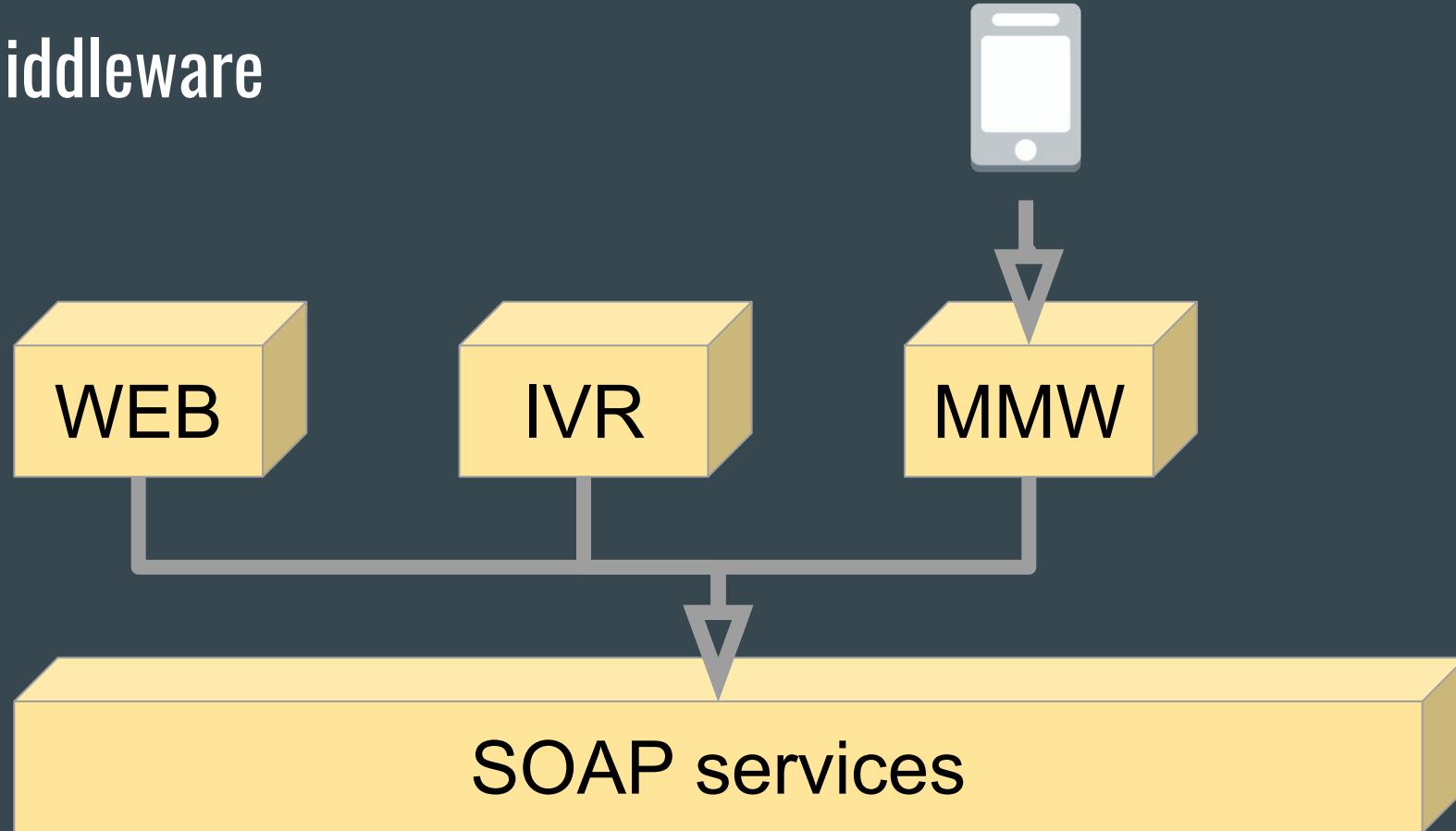
A few years ago ...



~~API~~ Mobile



Middleware



ROAST



ROAST API recipe

1. Take a SOAP/XML web service name add a / before it
2. Choose randomly an HTTP method between GET, PUT, POST, PATCH or DELETE, put it before the /
3. Transform input/output data from XML to JSON
4. If the method is GET or DELETE, put all parameters in query variables
5. And be sure to always return HTTP status 200

A man in a dark suit and white shirt is sitting at a desk, covering his face with his right hand. He appears to be in a state of distress or despair. The background is a studio set with a blue and red color scheme.

*I GIVE
UP*

The mobile team discovering
`GET /cancelTrfr?ztr1={id}`

Several years later

API



ALL THE THINGS!

Seriously

ARREST API



Design First

Use resource instead of actions

A list of wire transfers

/transfers

A wire transfers

/transfers/{transferId}

Use relevant HTTP method

Create a transfer

POST /transfers

Delete a pending wire transfer

DELETE /transfers/{transferId}

Update a customer email

PATCH /customers/me

Update a customer phone number

PATCH /customers/me

Use relevant HTTP status

403 Not enough money

503 No transfer between 1 am and 2 am

Provide hypermedia controls

**IF YOU HIT THIS SIGN,
YOU WILL HIT THAT BRIDGE**



GET /accounts/C1

```
{  
  "id": "C1",  
  "balance": <how much money I have>,  
  "actions": <hypermedia controls>  
 [  
   { "name": "transfer",  
     "method": "POST",  
     "href": "https://bank.com/transfers"}  
 ]  
}
```

Different ways to fill the set of actions

#1
Takeshi's Castle
Knock Knock

403 Forbidden

{

“code”: 1012,
“message”: “Insufficient
balance.”

}



503 Service unavailable

{

“code”: 1214,

“message”: “No
transfer between 1am
and 2am.”

}



#2
This is bowling.
There are rules.

```
GET /accounts/C1
{
  "id": "C1",
  "balance": -200,
  "actions": []
}
```



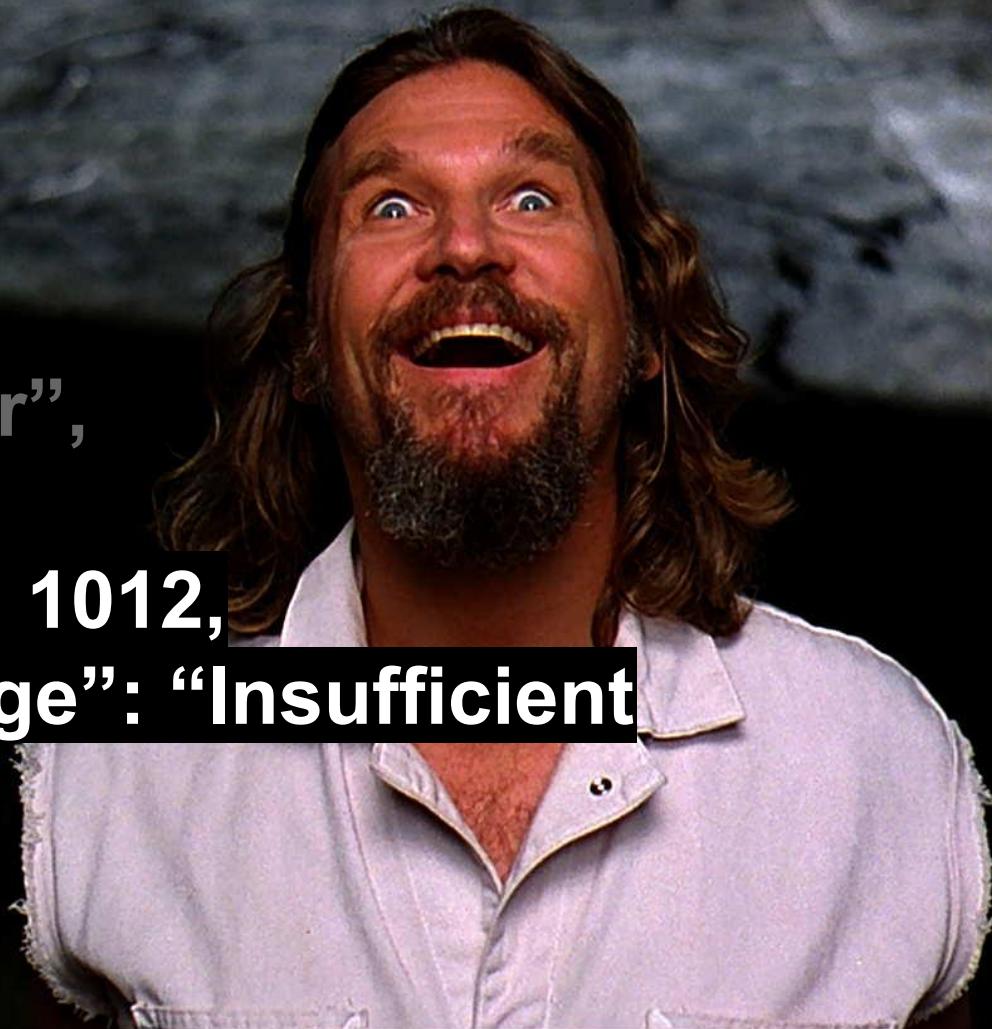
```
GET /accounts/C1
{
  "id": "C1",
  "balance": 20000,
  "actions": []
}
```



#3
The Dude abides.

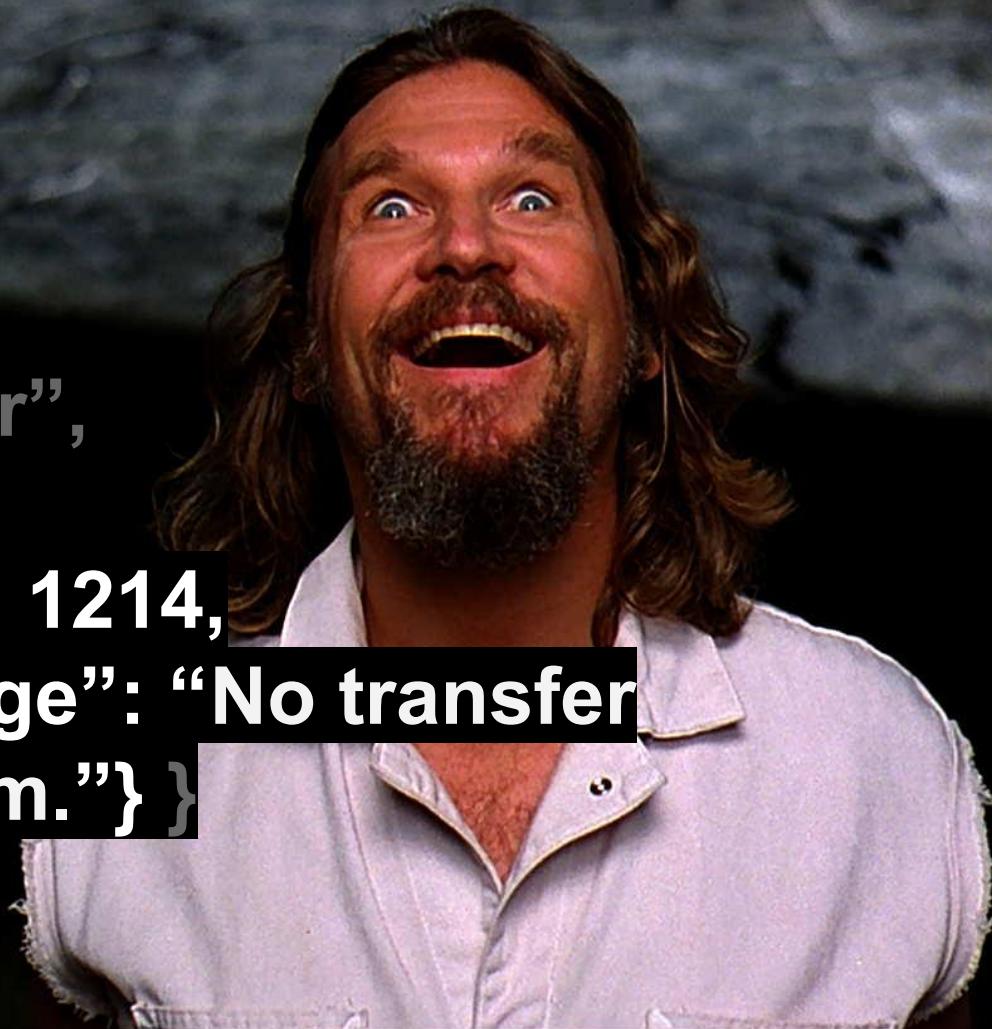
```
GET /accounts/C1
```

```
{ "id": "C1",
  "balance": -200,
  "actions": [
    { "name": "transfer",
      "status": 403,
      "error": { "code": 1012,
                 "message": "Insufficient
                           balance."} }
  ]
}
```



GET /accounts/C1

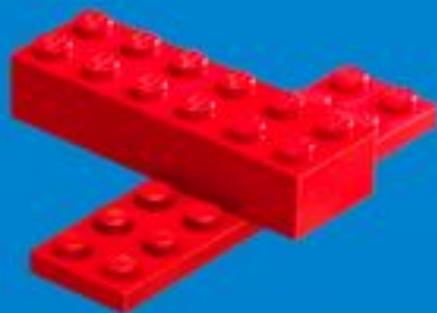
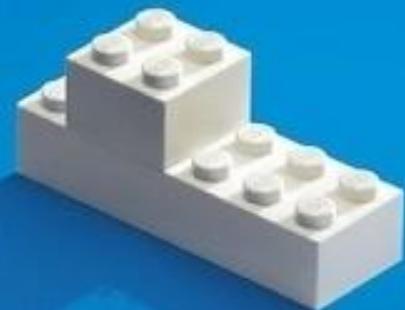
```
{ "id": "C1",
  "balance": 20000,
  "actions": [
    { "name": "transfer",
      "status": 503,
      "error": { "code": 1214,
                 "message": "No transfer
between 1am and 2am."} }
  ]
}
```



Is it really so simple to design an API?

Is interfacing with a RESTful API simple?

- Is it easy to use?
- Is it easy to understand?
- Is the abstraction adapted to the audience?



Simple as a lego brick

The end?

• • •

To the heights of API and beyond...