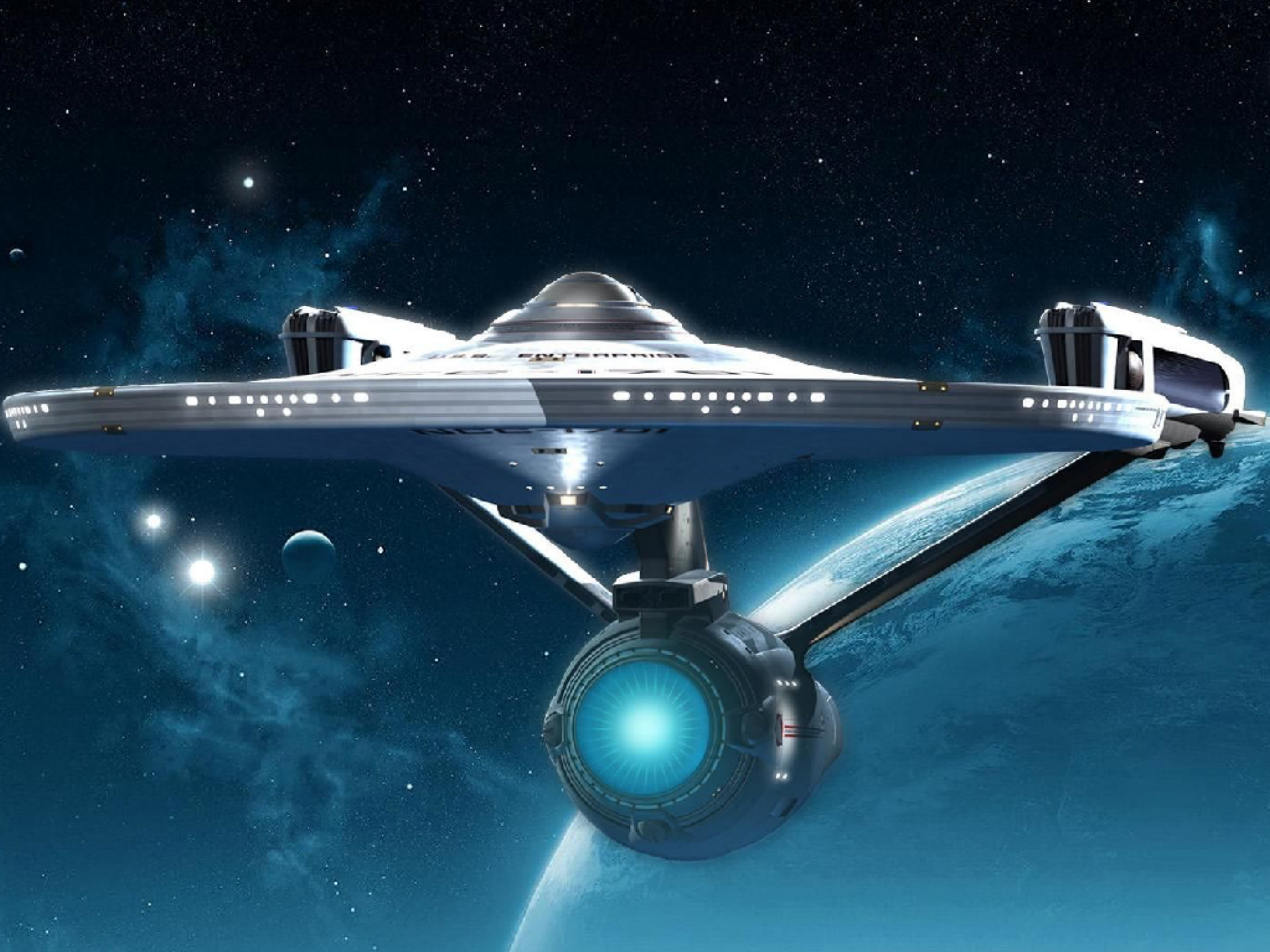
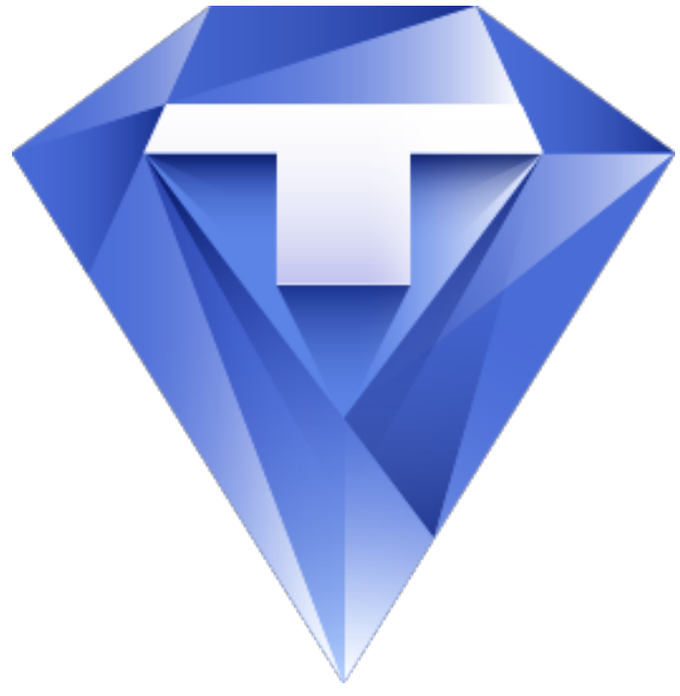


“I hope the field of computer science never loses its sense of fun”

–Structure and interpretation of computer programs

# About me

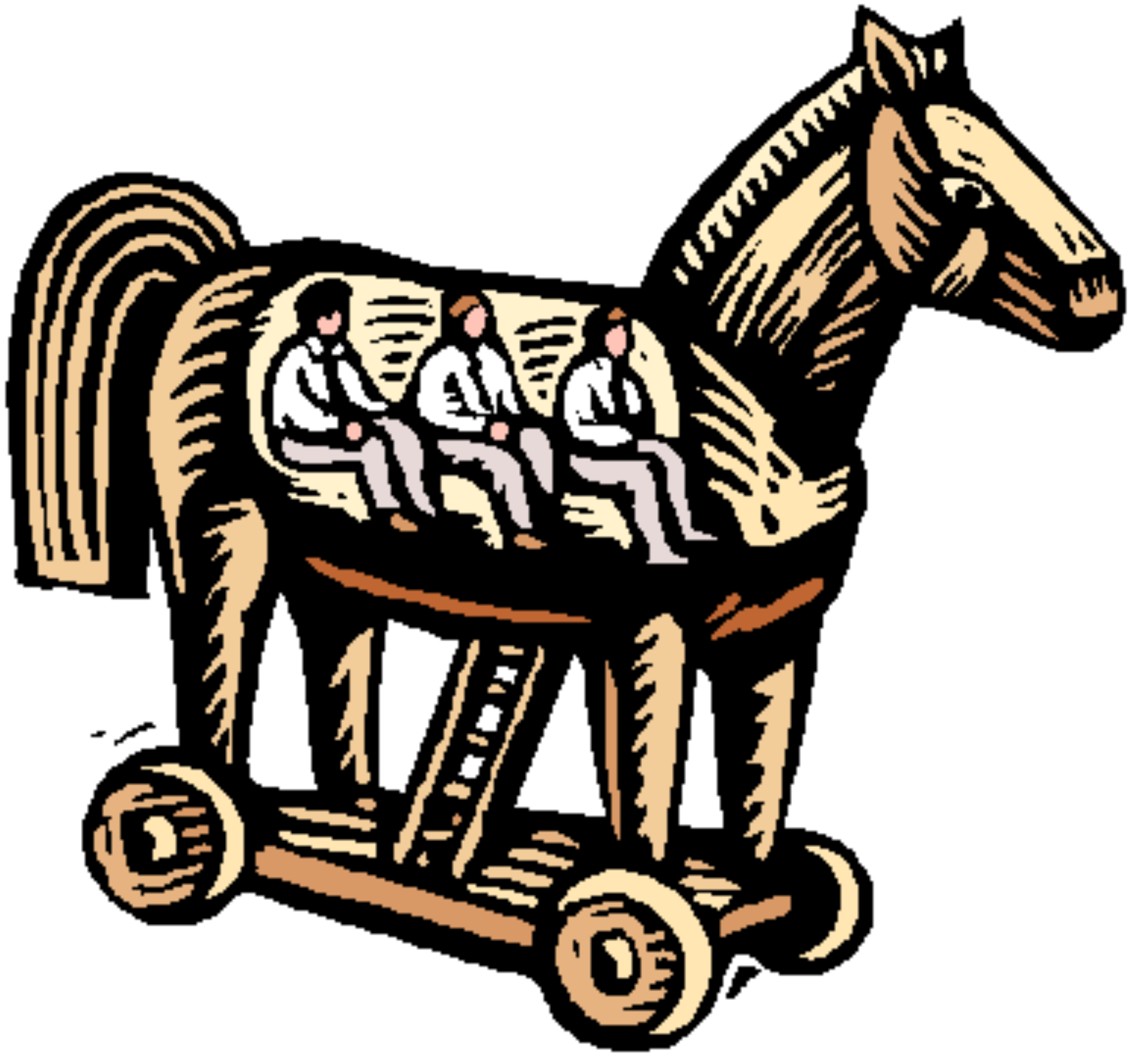
- software engineer / technology trainer
- kung fu instructor in spare time
- ~~clojure~~ data driven evangelist
- @sihingkk



yeah, we're hiring!

# What is this talk about?

- rant on “industry standards” and recent buzz words
- simple, artificial use case - just to make it
- architectural ~~design~~ flavour proposal
- definitely not my own ideas =)




"WARP SPEED AHEAD!"

Test!

# Who is it?

# Who is it?

# Who is it?

# Who is it?

# Who is it?

# Who is it?

# Who is it?

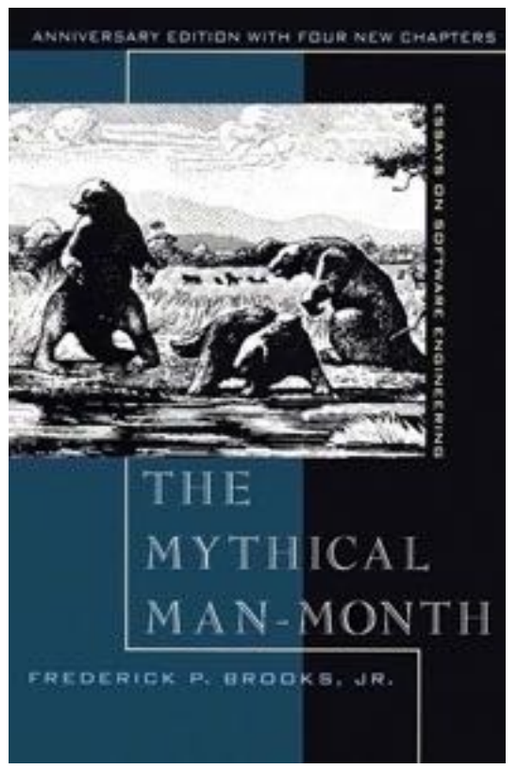

# What is?

- OOP
- FP
- Language paradigm
- DVORAK

# What we doing?

# What we think we are doing?

# What we are really doing?



# What we are really doing

# Meet Rich

- Cost/benefit
- ROI
- Time to market
- Profit

# Structure and Interpretation of Computer Programs

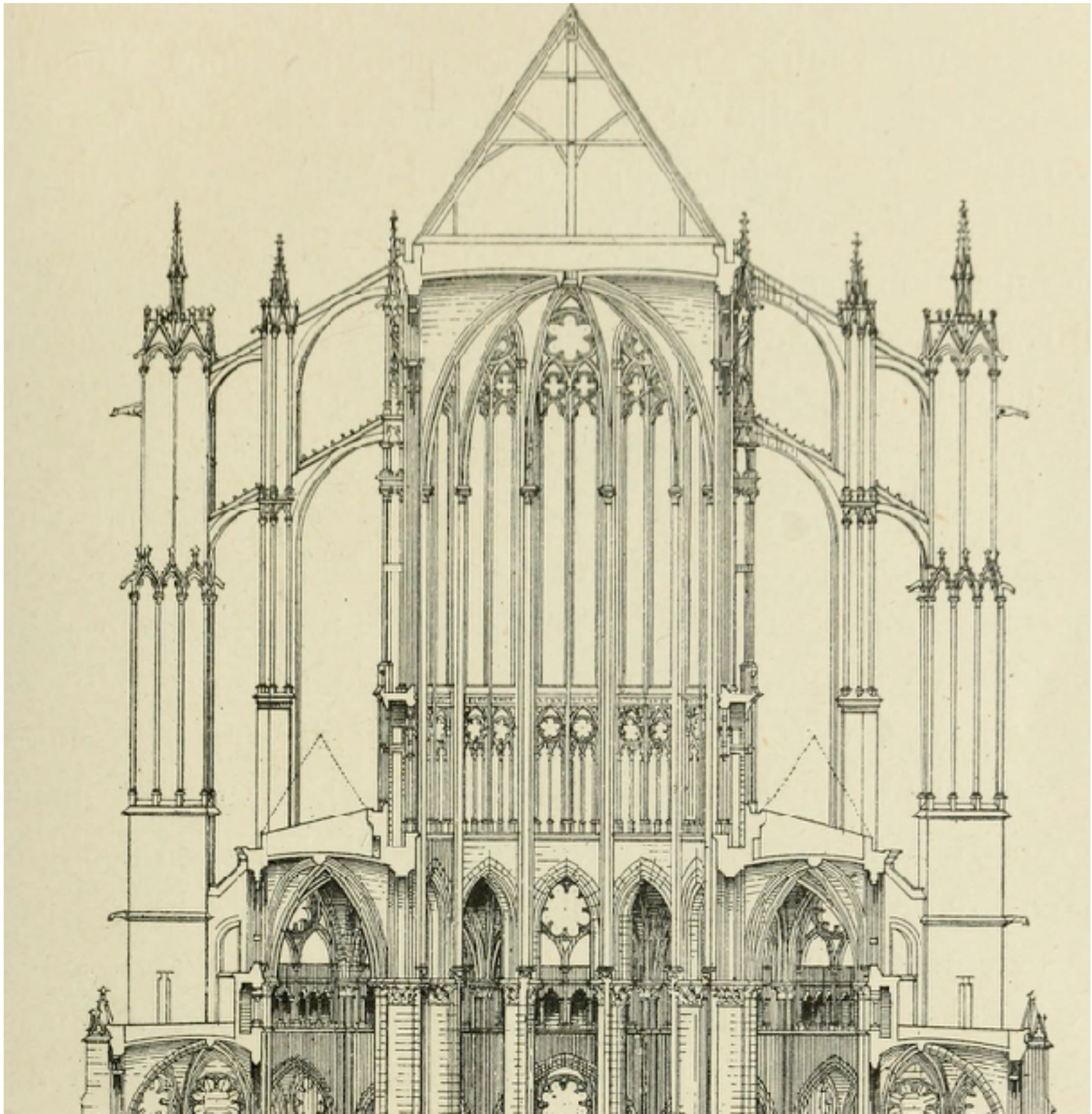
Second Edition


Harold Abelson and
Gerald Jay Sussman
with Julie Sussman

# Architecture

**Architecture** (Latin *architectura*, from the Greek ἀρχιτέκτων *arkhitekton* "architect", from ἀρχι- "chief" and τέκτων "builder") is both the process and the product of planning, designing, and constructing buildings and other physical structures

# Arch-itecture?



# REST

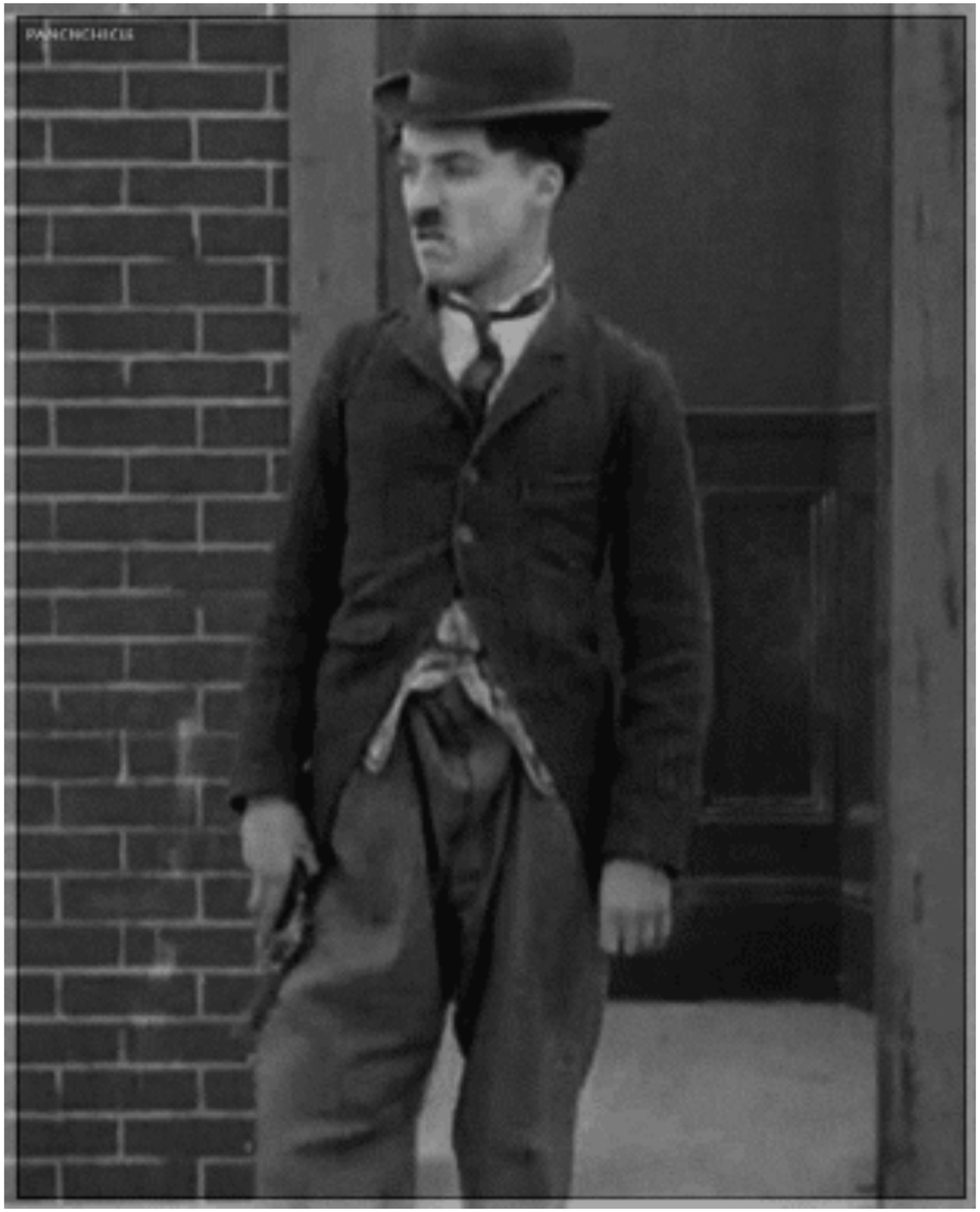
is dead?

- define the problem
- what is REST?
- CRUD-flavoured REST?

*The key abstraction of information in REST is a resource. Any information that can be named can be a resource: a document or image, a temporal service (e.g. "today's weather in Los Angeles"), a collection of other resources, a non-virtual object (e.g. a person), and so on. In other words, any concept that might be the target of an author's hypertext reference must fit within the definition of a resource. A resource is a conceptual mapping to a set of entities, not the entity that corresponds to the mapping at any particular point in time.*

/posts/show/1

/posts/1


PANCHICHECLE

- transfer from account 1 $500 to account 2
- POST /accounts/1/transfer/500.00/to/2

- Wrong!

```
POST /transactions HTTP/1.1
Host: <snip, and all other headers>

from=1&to=2&amount=500.00
```

```
HTTP/1.1 201 OK
Date: Sun, 3 Jul 2011 23:59:59 GMT
Content-Type: application/json
Content-Length: 12345
Location: http://foo.com/transactions/1

{"transaction":{"id":1,"uri":"/transactions/1","type":"transfer"}}
```

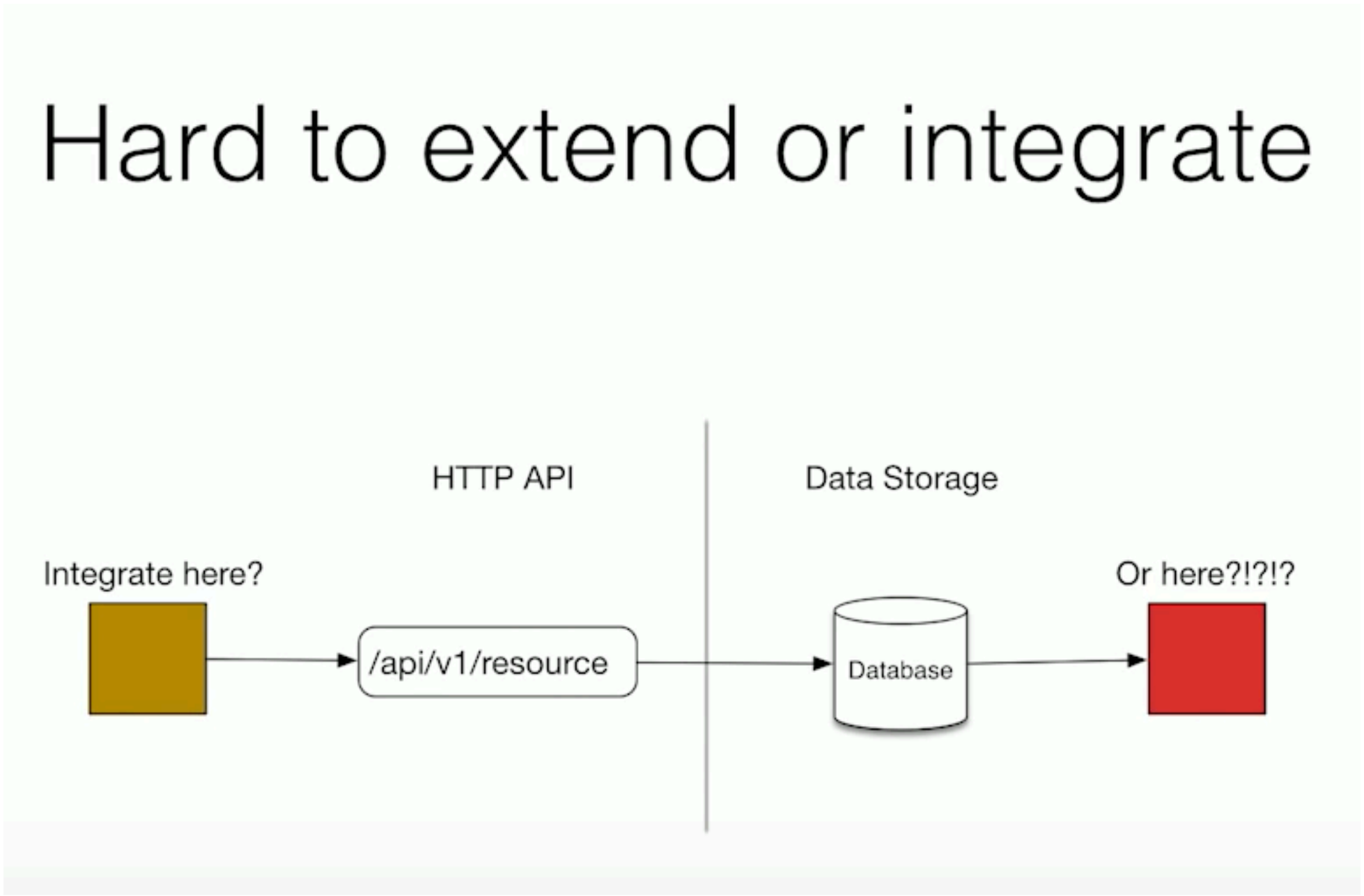
# All the cool kids are doing it

- It’s Agile (TM)
- much easier than SOAP
- easly to get something (poorly specified) intof production
- Microservices! Containers! DevOps! Cloud!

# Operationally rigid

- CRUD flavoured REST issues:
  - Choose one:
    - “pay my credit card bill”
      - POST a new bill payment sub-resource of the credit card resource”
- entire API, with all N endpoints often deploy version change scale together
- Scaling individual resources (single code base) requires re-work
- Scaling read from writes it hard

# Hard to extend or integrate



# Obscures the story

- typical problems with mutable state systems
- **what is the client/customer's actual intent?**
- what happened to bring us to the current state?
- how do we audit that seq of events for security compliance or correctness?


gifbin.com
theeraltras

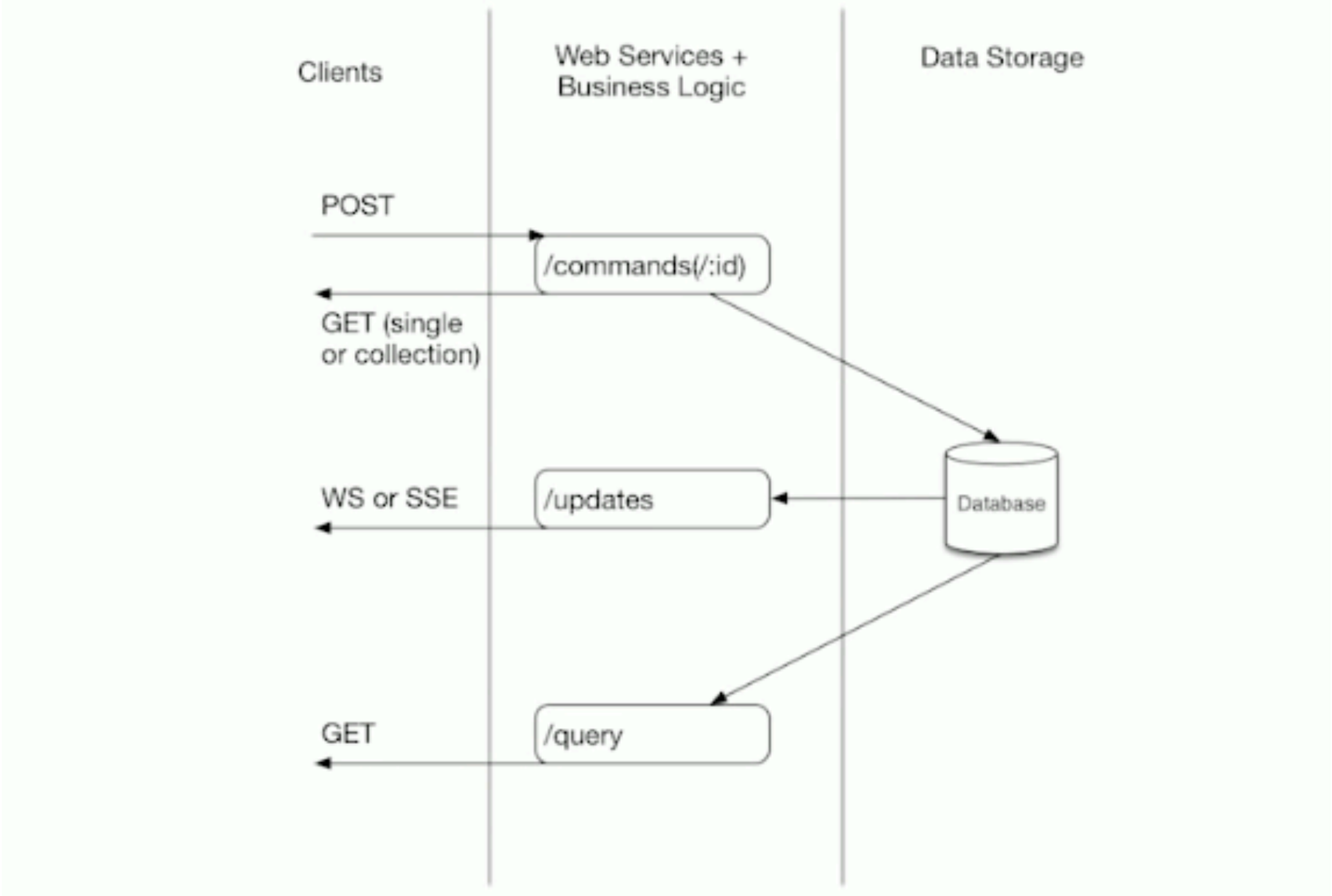
# CQRS as solution

- Command (and) Query Responsibility Segregation
- Separate interface and model for write (command) and read (query) paths in applications
- in REST/HTML terms, perhaps:
  - /commands for writes
  - /query (pull) and /updates (push, via WS/SSE)

# Functional programming!

- Immutability
- values flow though system being transformed
- Isolate and delay side-effects until the edges

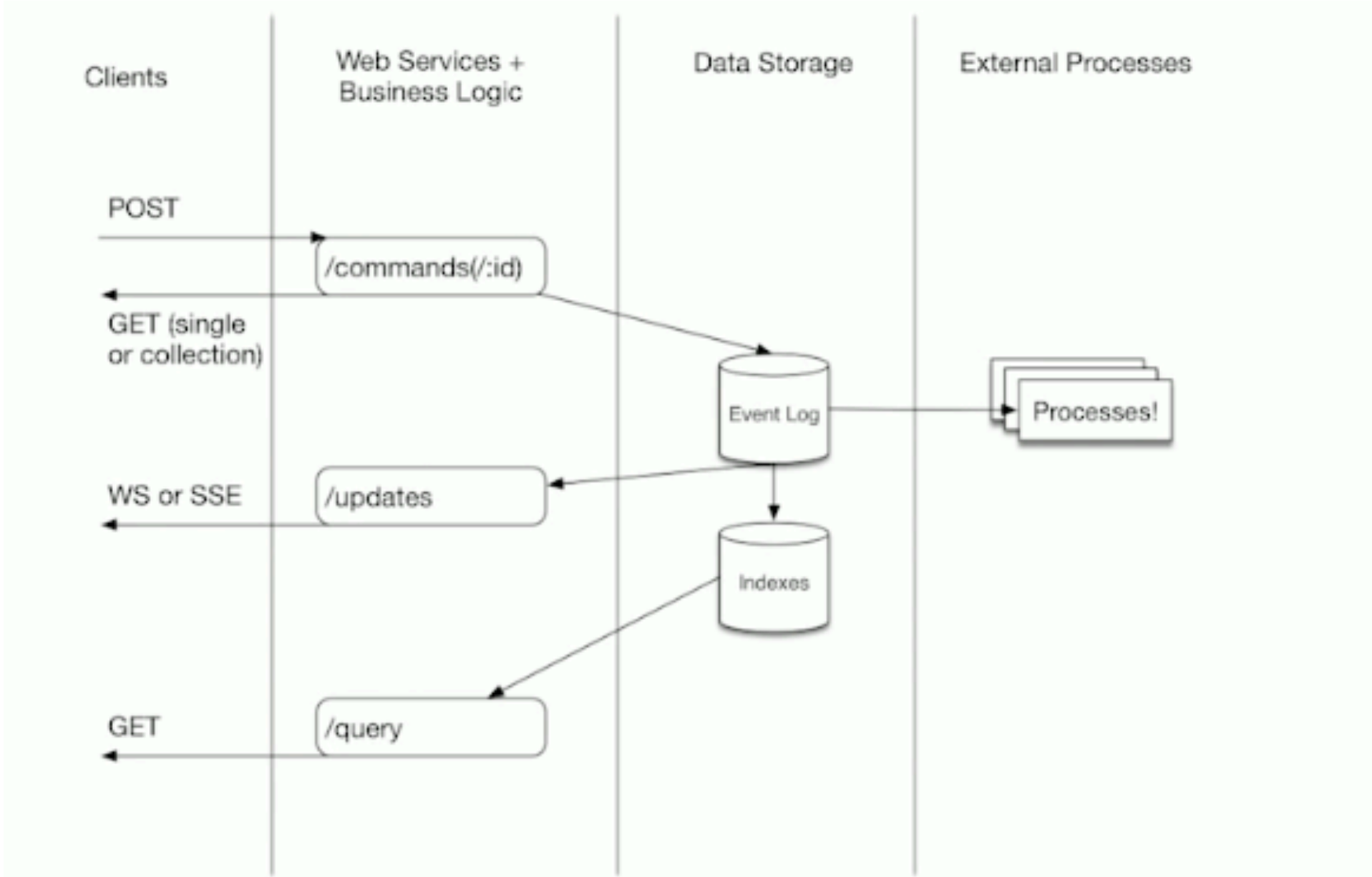
# CQRS via HTTP



# Why might I use CQRS?

- Simplicity and de-complecting
- operational simplicity
- client implementation simplicity and agility (untangled-web!)

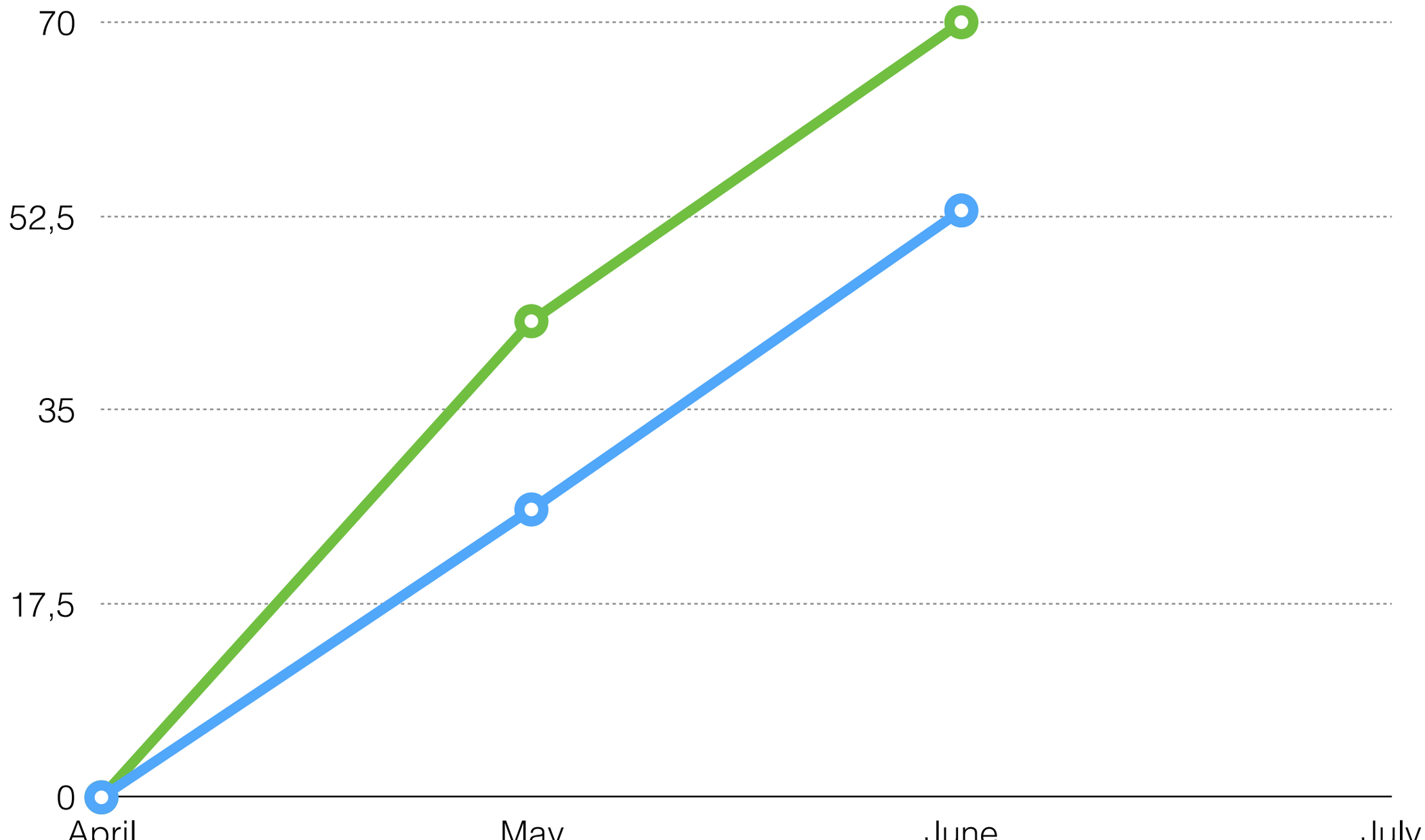
# Perhaps event sourcing?



# Microservices

# What kind of issues it solving?

# Organic architecture patterns



Marketplace

Klient

Oferta

Transakcja

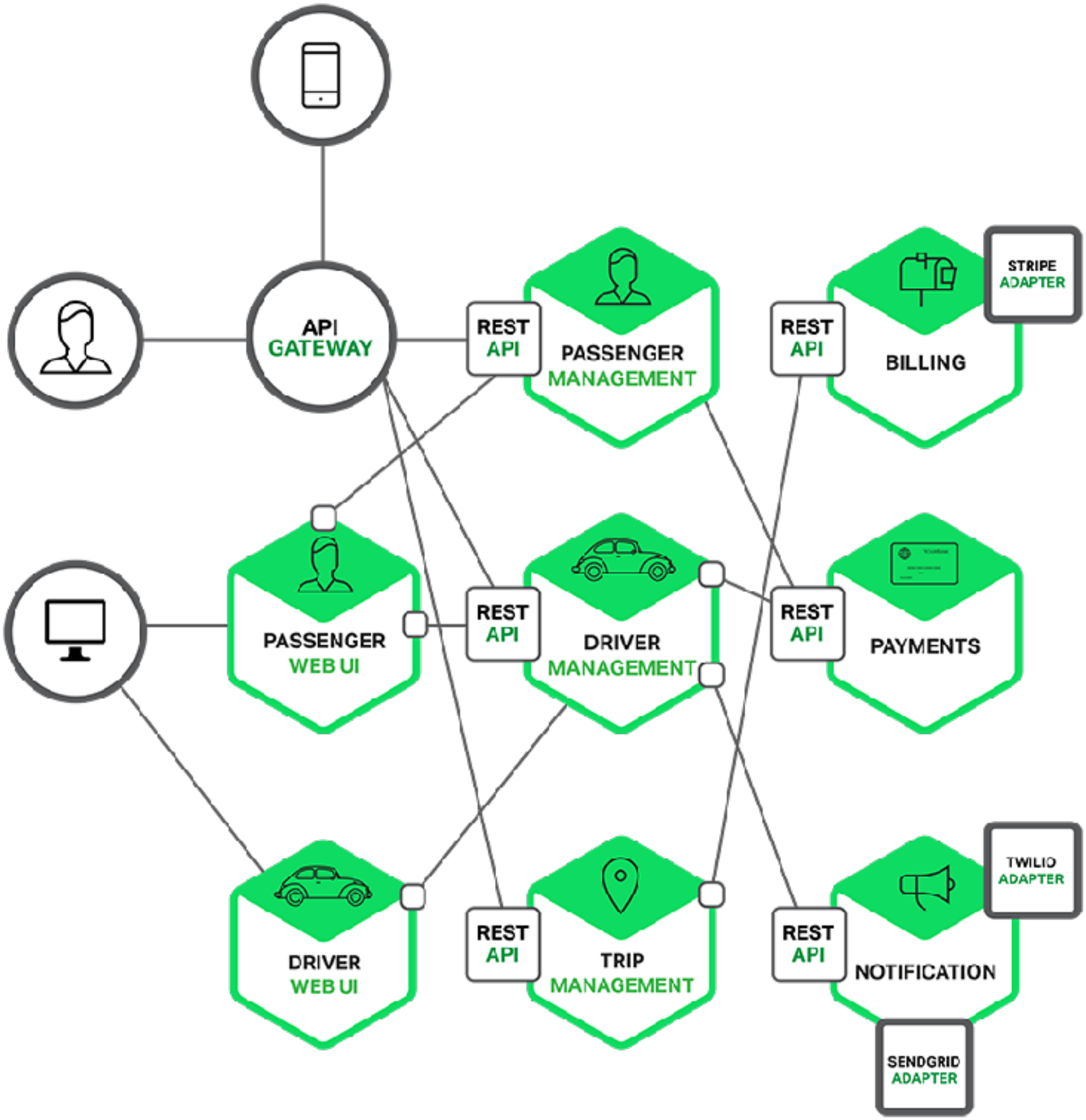
STREAM

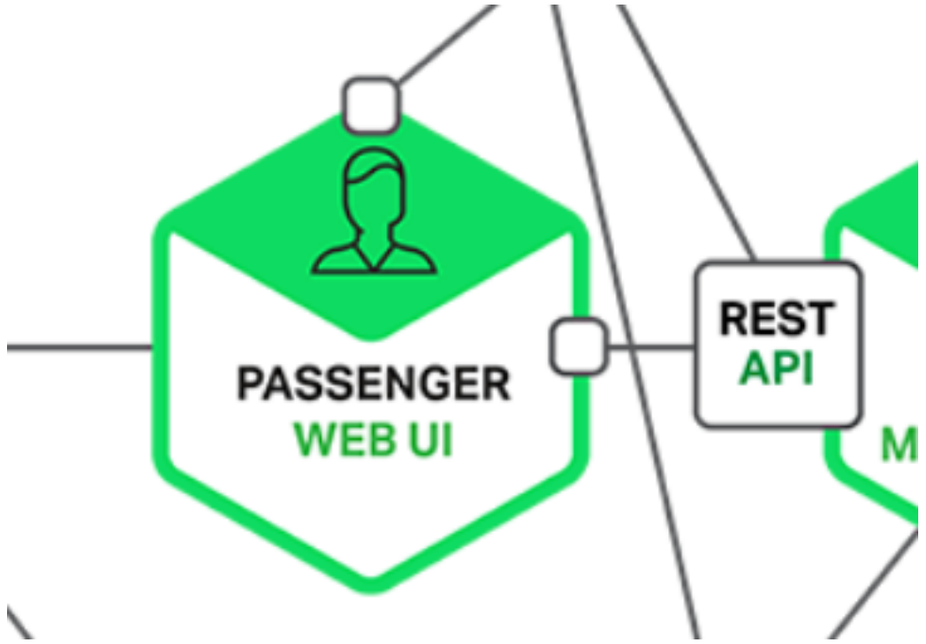
TREE

MESH

# How it look like before?

# Solve the issue?


API
GATEWAY
REST
API
PASSENGER
MANAGEMENT
REST
API
BILLING
STRIPE
ADAPTER
PASSENGER
WEB UI
REST
API
DRIVER
MANAGEMENT
REST
API
PAYMENTS
DRIVER
WEB UI
REST
API
TRIP
MANAGEMENT
REST
API
NOTIFICATION
TWILIO
ADAPTER
SENDGRID
ADAPTER


PASSENGER
WEB UI
REST
API
M

# Monolitic server

JSP

# Light client

# Rich clients


JavaServer™ Faces
JSF

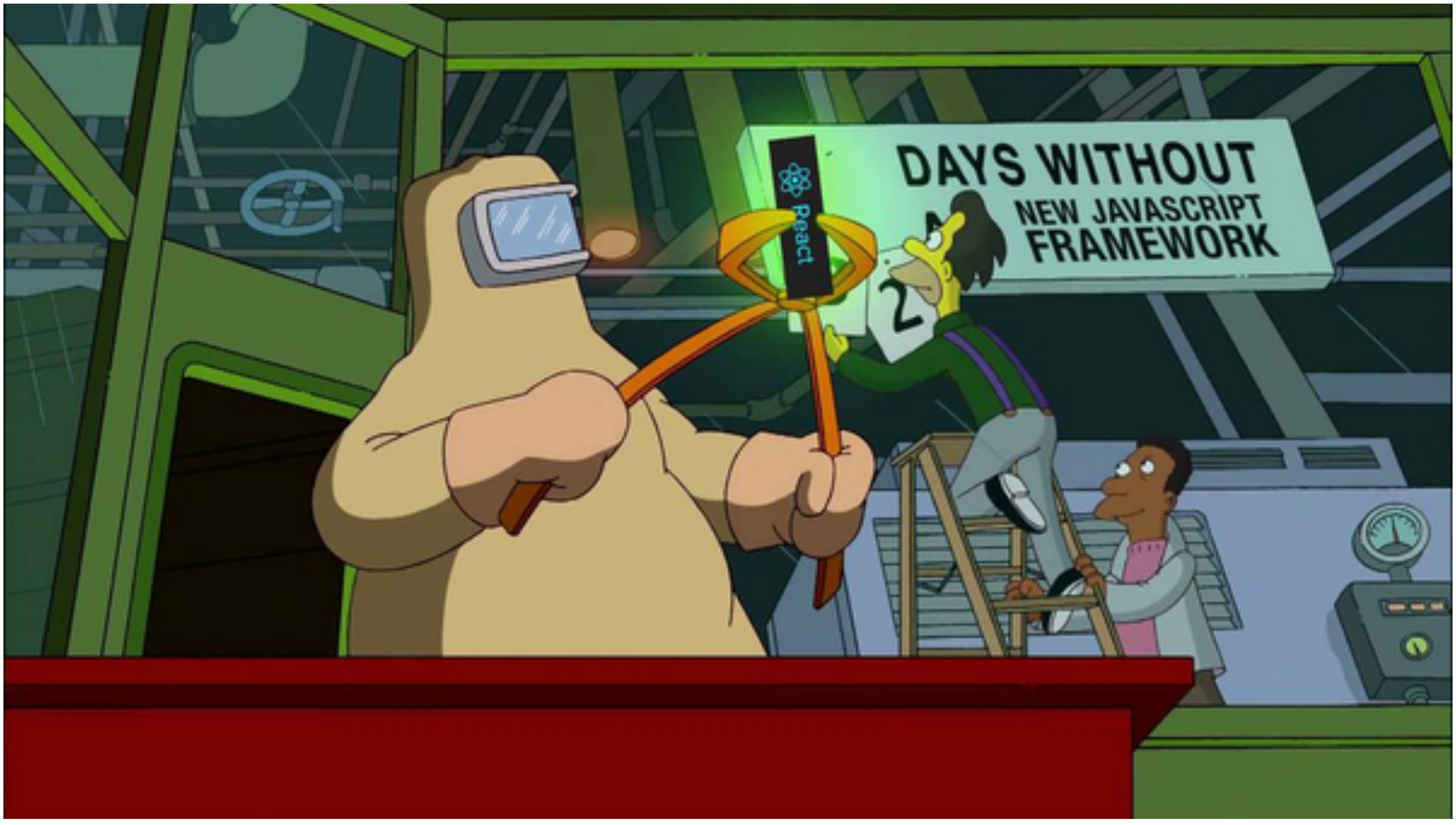

Google
Web Toolkit ®


ANGULARJS
by Google

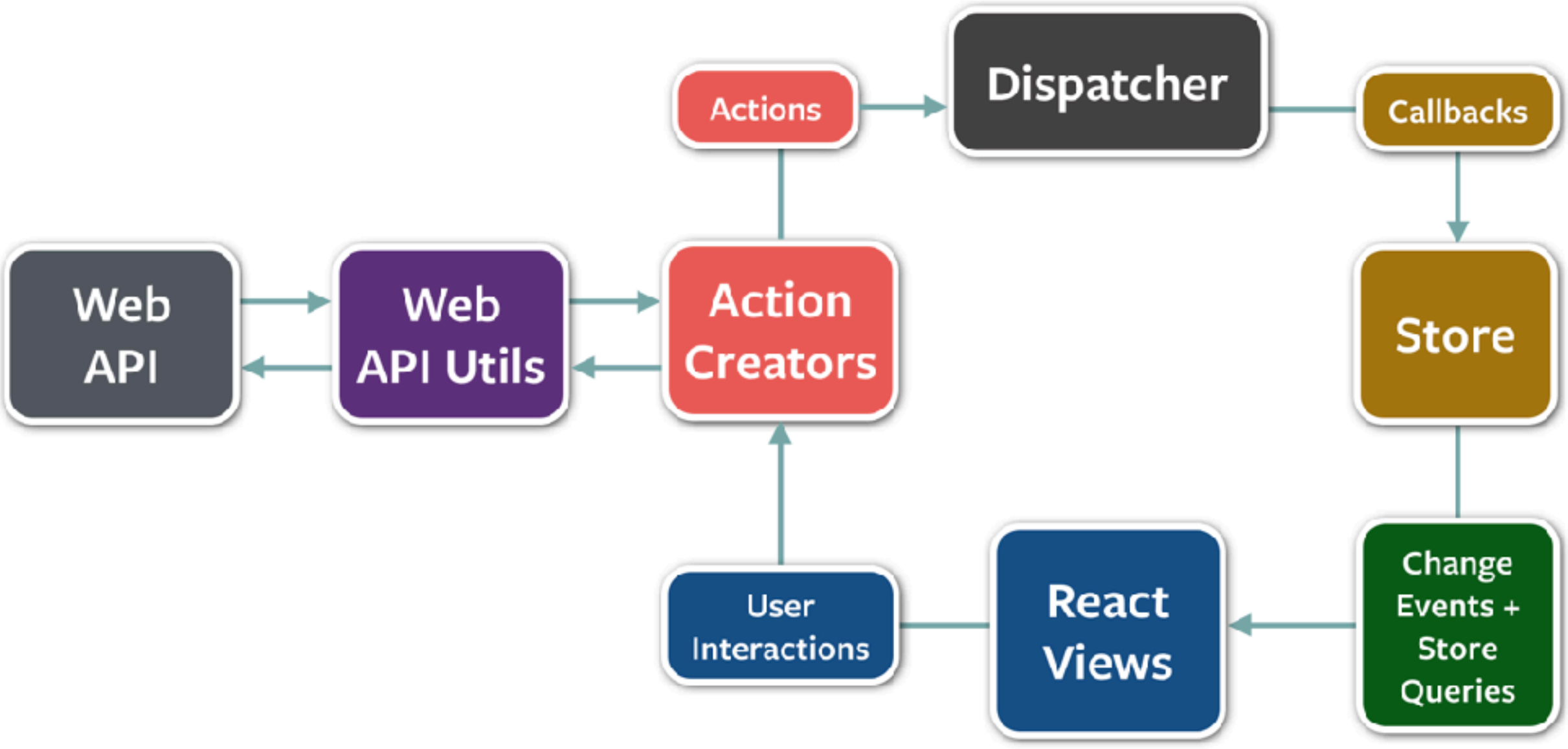

ember

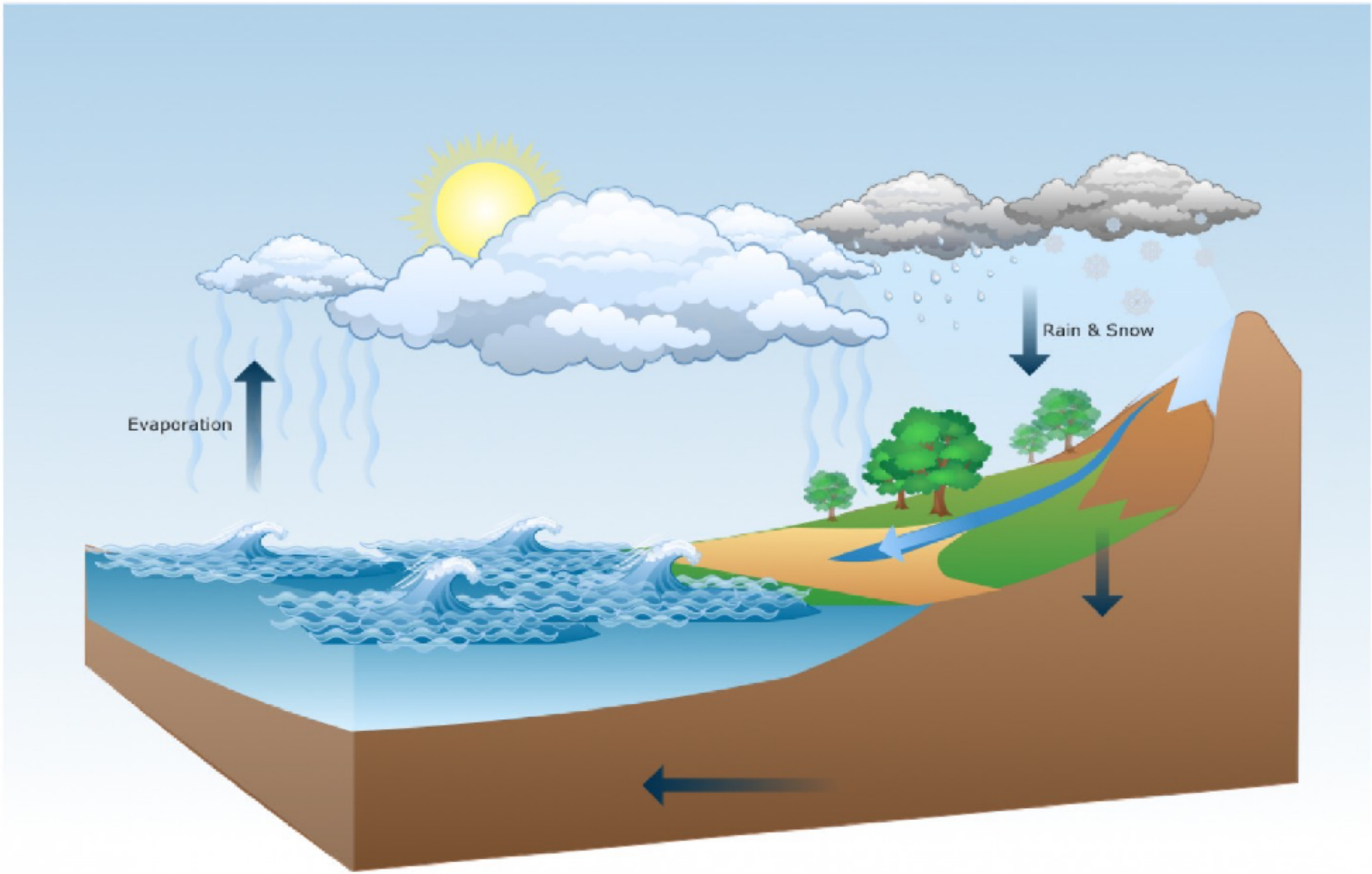

METEOR


React


React
DAYS WITHOUT
NEW JAVASCRIPT
FRAMEWORK
2

# Redux


Actions
Dispatcher
Callbacks
Web API
Web API Utils
Action Creators
Store
User Interactions
React Views
Change Events + Store Queries


Rain & Snow
Evaporation

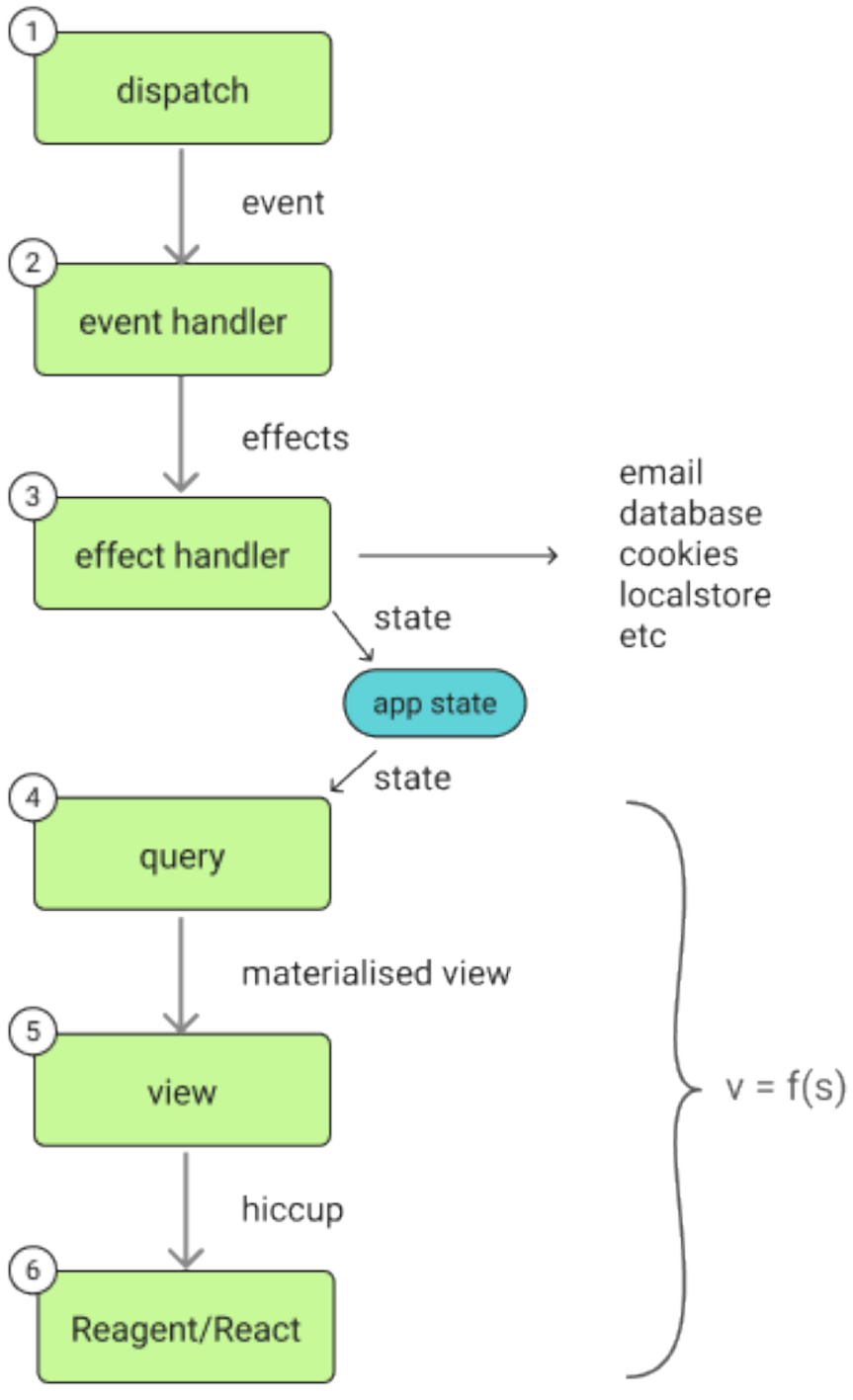
jugosławia

jug**osławia**

jug**gernaut**

jug**owice**

jug**ema**


1
dispatch
event
2
event handler
effects
3
effect handler
email
database
cookies
localstore
etc
state
app state
state
4
query
materialised view
5
view
hiccup
6
Reagent/React
v = f(s)

Items List

Item 0

Item 1

Item 2

Item 3

click

# Rest example friends

/api/person/1
{“name”: “KK”, …}

/api/person/1/mates
{“friends”: [112,123,43]}

vs

/api/friends?mate=1
[{“id”:1, “persons”: [1,121]},
{“id”:1, “persons”: [1,123]}
{“id”:1, “persons”: [1,43]},,]

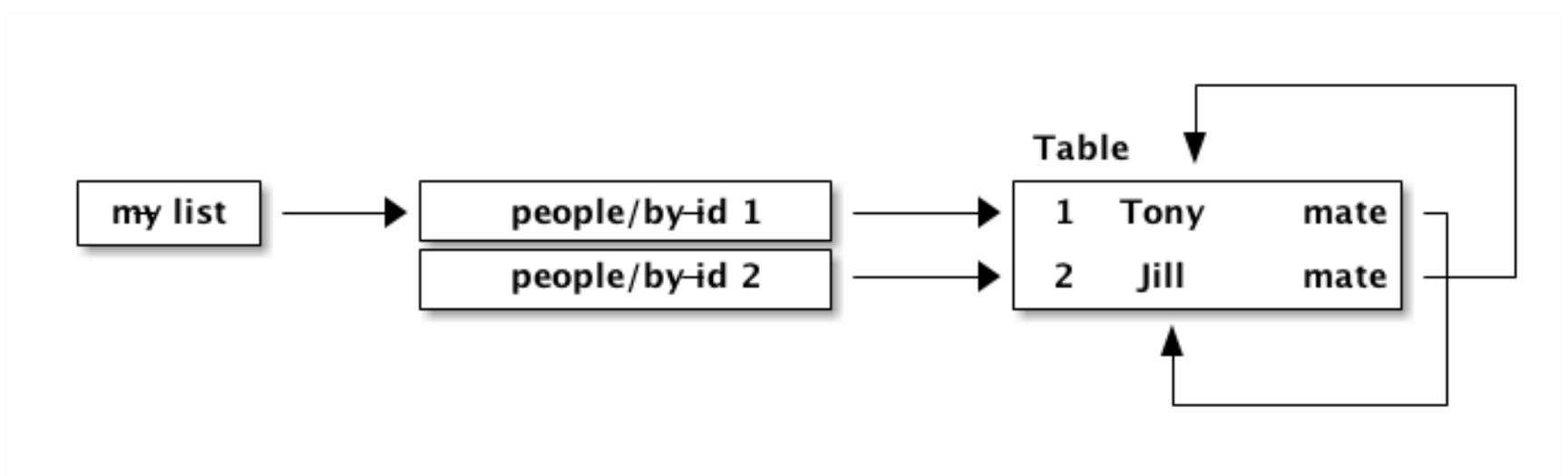
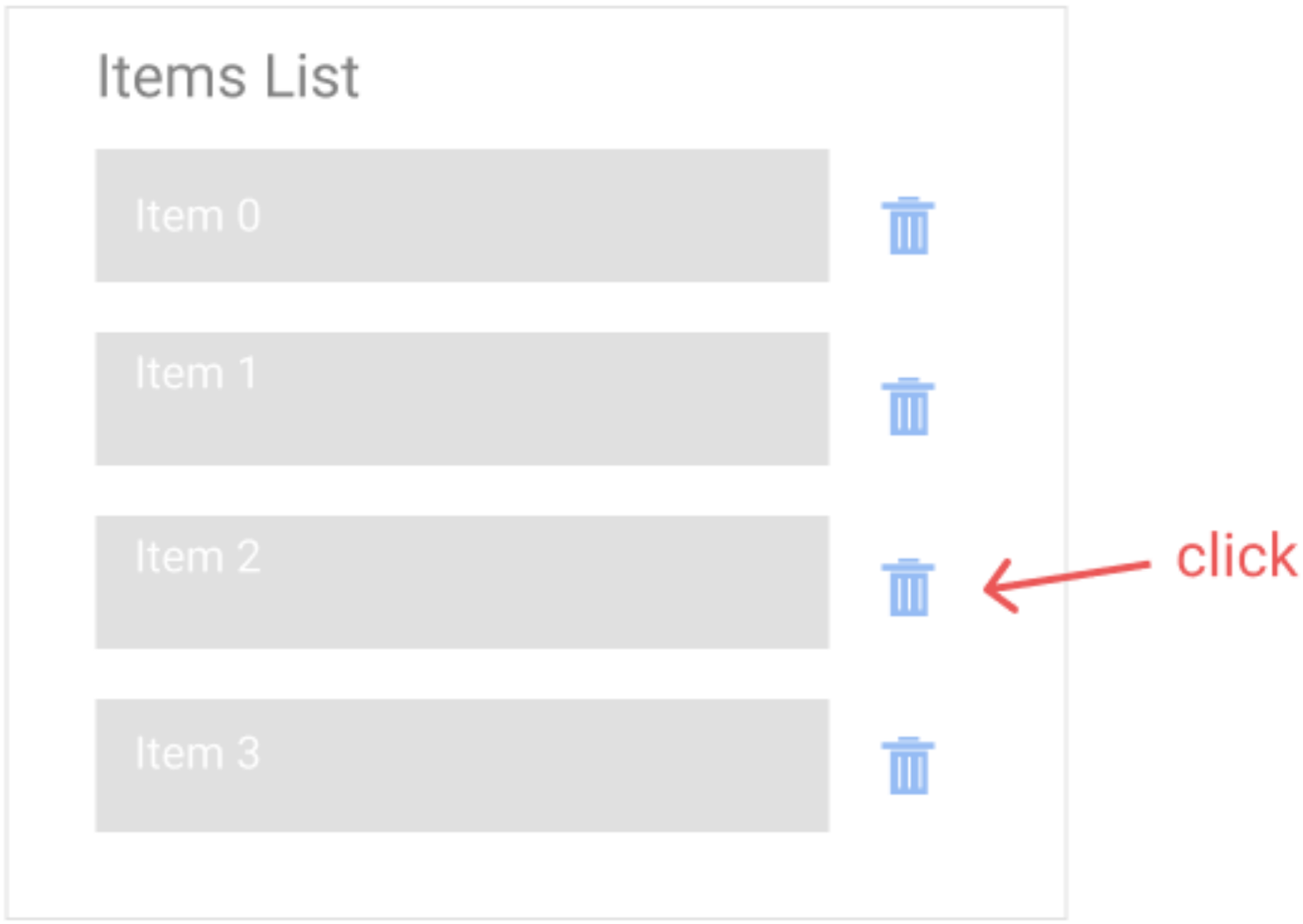
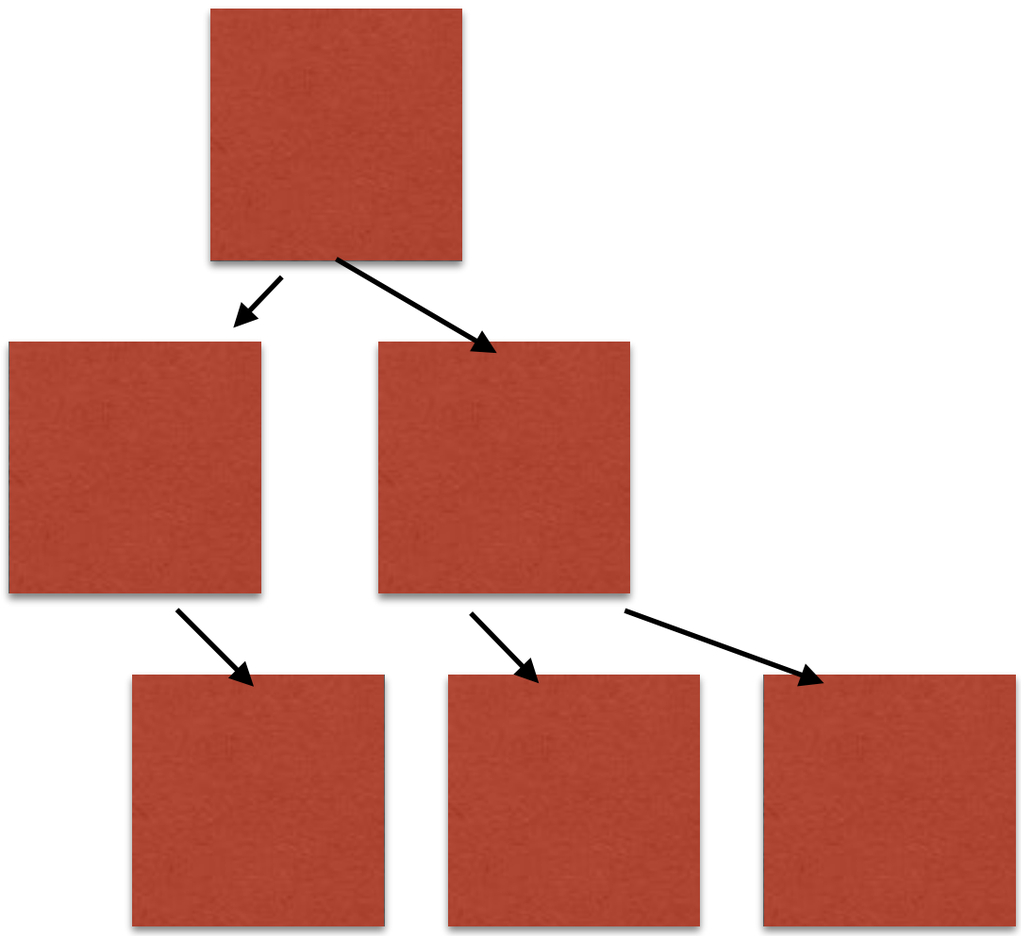


Table
my list
people/by-id 1
people/by-id 2
1 Tony mate
2 Jill mate

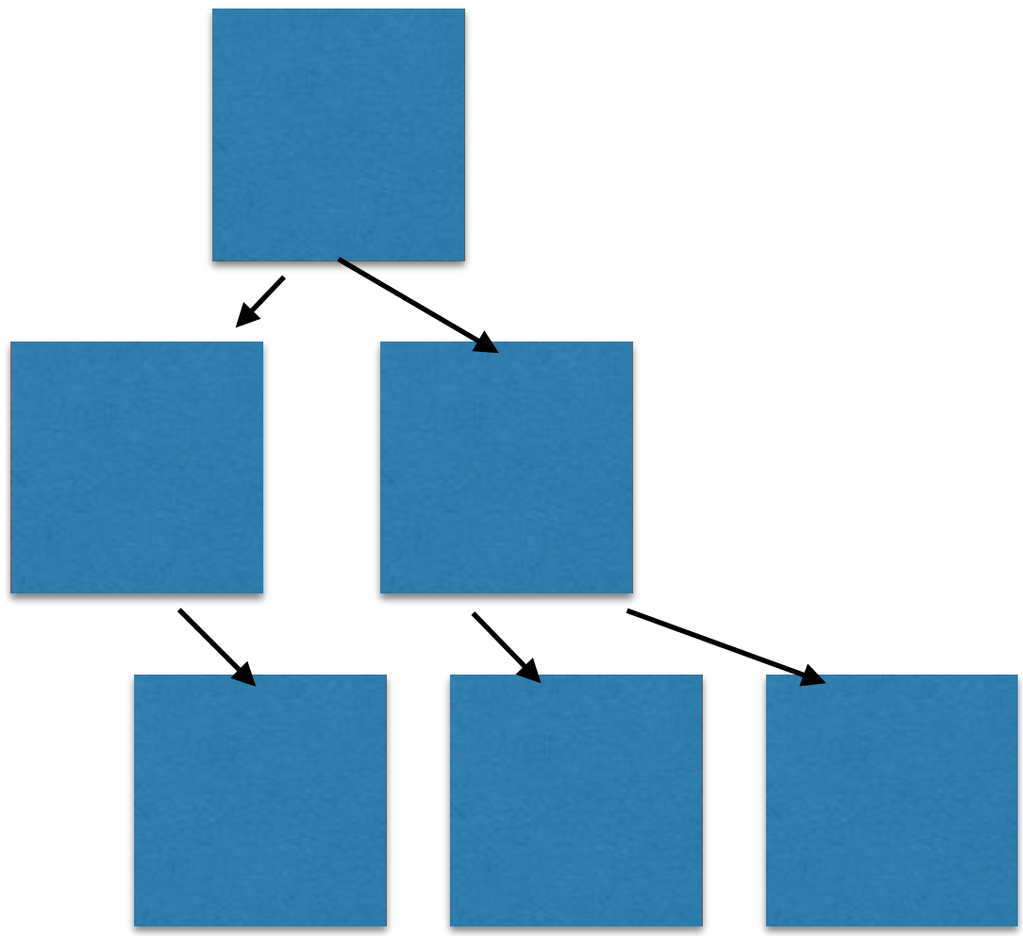
# Co-located queryies



Components



Queries



# How to do it?

[e a v t]

[12 :person/name “KK”]

[1 :name “Bob” 100 true]

[1 :likes “Cats” 100 true]

[1 :likes “Dogs” 111 true]

[1 :likes “Cats” 111 false]

[1 :name “Bob” 100 true]

[2 :name “Marry” 100 true]

[1 :mate 2]

[1 :mate [:name “Marry”]]

```
[:find ?name ?age
 :in $db ?id
 :where
 [?id :actor/name ?name]
 [?id :actor/age ?age]]
```

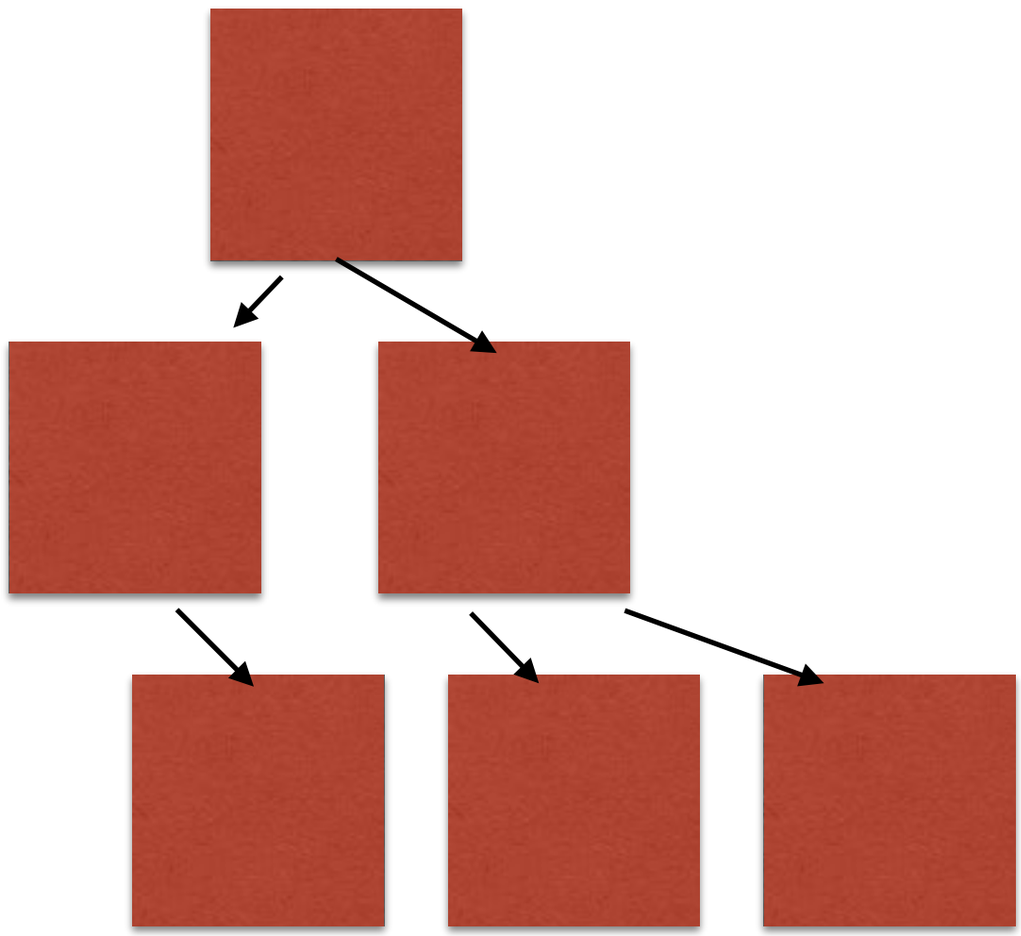
```
[:find ?name ?age
 :in $db ?id
 :where
 [?id :actor/name ?name]
 [?id :actor/age ?age]]
```

```
[:find ?name ?age
:in $db ?id
:where
[?id :actor/name ?name]
[?id :actor/age ?age]]
```

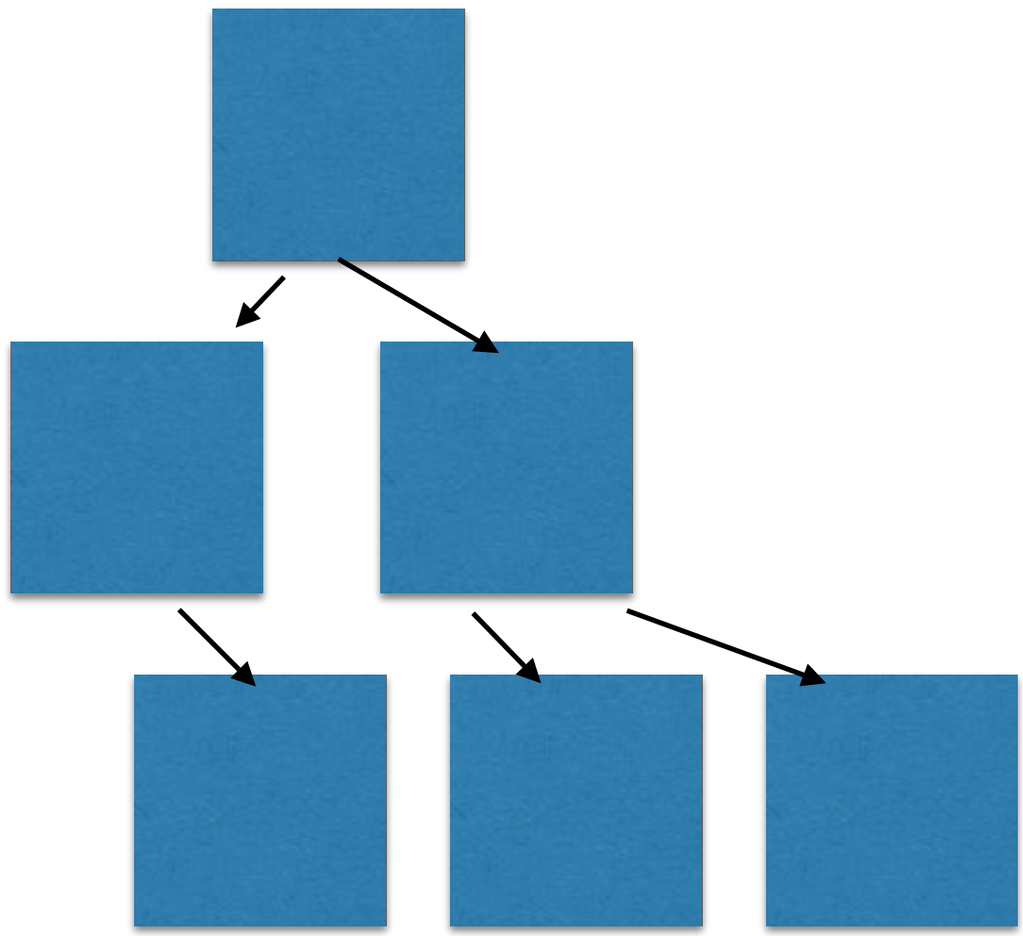
```
[:find ?name ?age
 :in $db ?id
 :where
 [?id :actor/name ?name]
 [?id :actor/age ?age]]
```

[e a v t]

Components



Queries



LearnDatalogToday.org

- Clojure/ClojureScript
- Frontend:
  - react->reagent->om.next->**untangled**
- backend:
  - rest->cqrs->ring->datascript->**datomic**

for spotify: **kafka/samza**

# Questions?

@sihingkk