

Back to front

Backend technologies for the frontend developer

Who am I?



Savvas Dalkitsis

(he/him)

Principal Software Engineer
@ASOS

ASOS

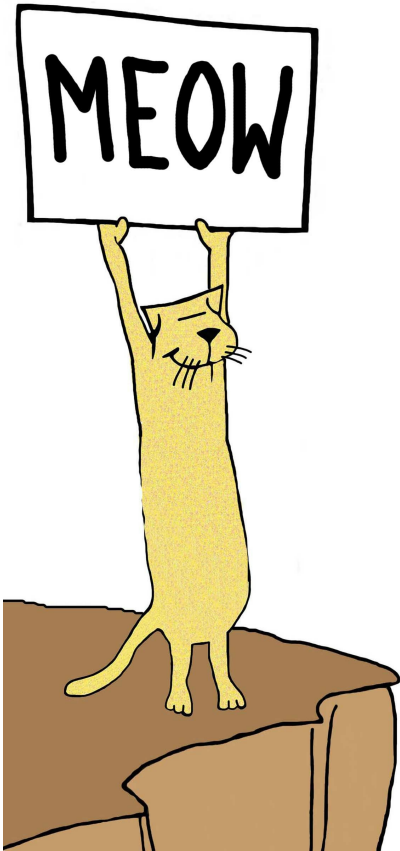


Why am I here?

A word cloud of Android development tools and frameworks. The words are arranged in a circular pattern and include: Studio, gradle, Firebase, Material, Coroutines, Google Play, Room, MVVM, Retrofit, Dagger, MVP, Hilt, RxJava, Jetpack, Compose, and Android.



So what?



Language barriers are real in tech too.

Progressing your career means being able to have meaningful discussions with people outside your domain and comfort zone.

A basic understanding of their language is paramount.

Source: cartoonresource/Adobe Stock

Let's start

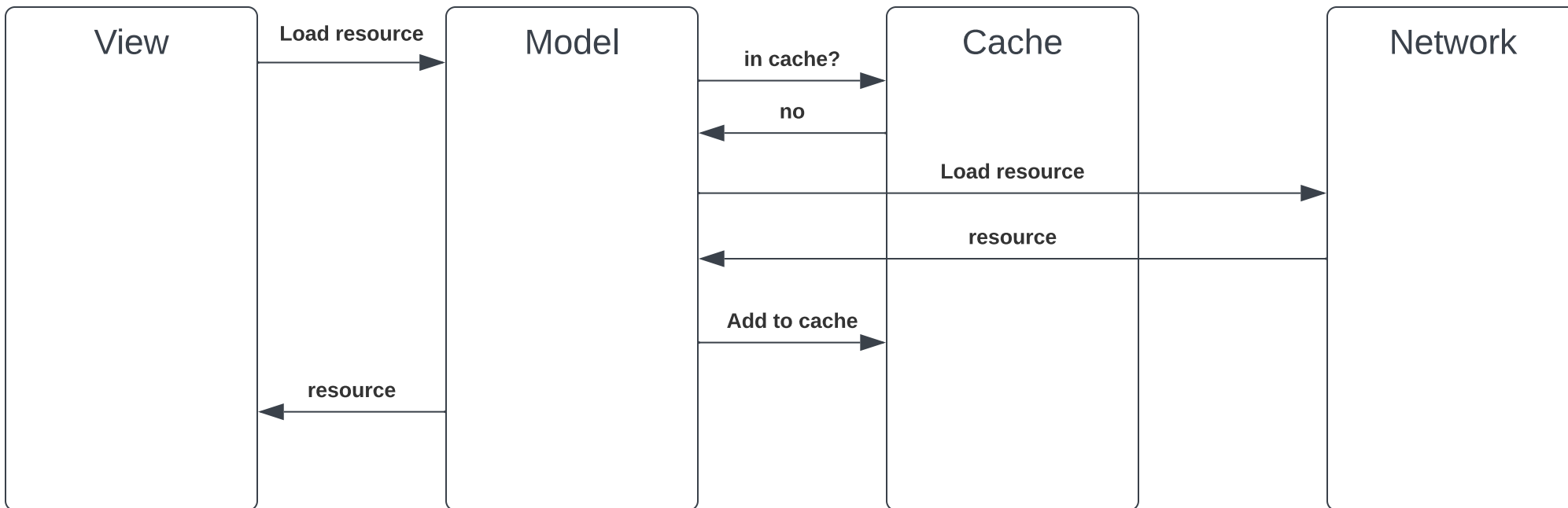
Caching

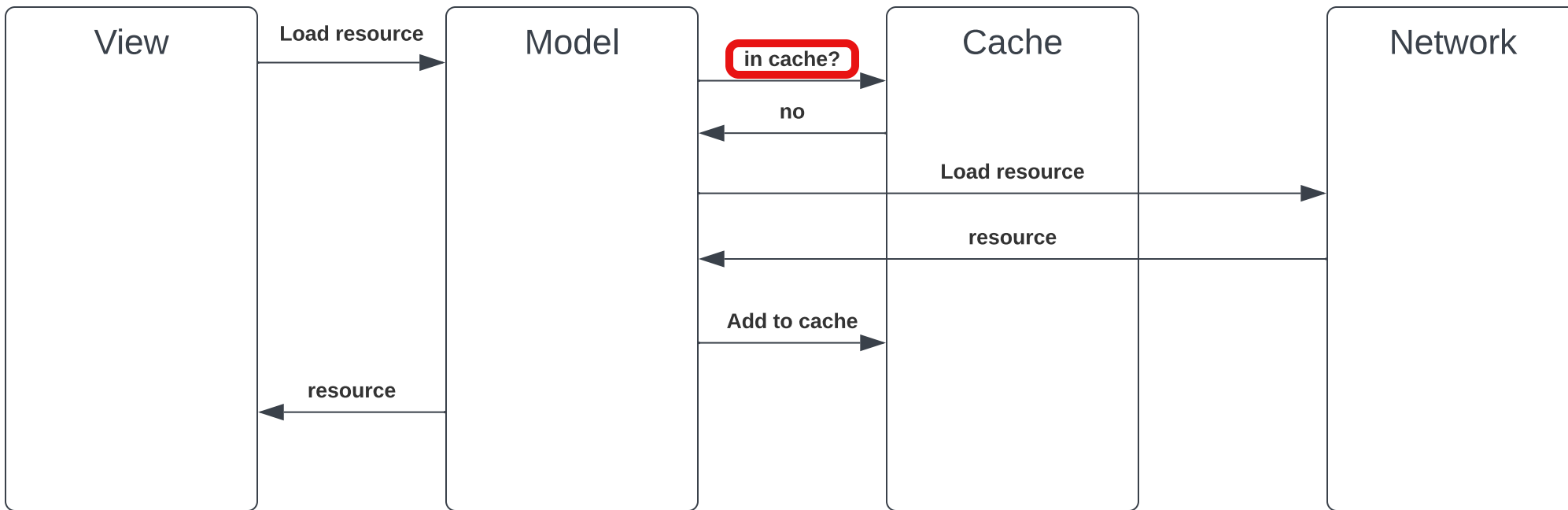
There are only two hard things in Computer Science: cache invalidation and naming things.

| Phil Karlton

Caching

Expiry headers





Caching

Server side controlled expiry

Caching

Server side controlled expiry

Expires header

Example Expires: Wed, 21 Oct 2015 07:28:00 GMT

Caching

Server side controlled expiry

`Cache-Control` header

Example `Cache-Control: max-age=604800`

The following table lists the standard Cache-Control directives:

Request	Response
max-age	max-age
max-stale	-
min-fresh	-
-	s-maxage
no-cache	no-cache
no-store	no-store
no-transform	no-transform
only-if-cached	-

Request	Response
-	must-revalidate
-	proxy-revalidate
-	must-understand
-	private
-	public
-	immutable
-	stale-while-revalidate
stale-if-error	stale-if-error

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Cache-Control>

Caching

ETags

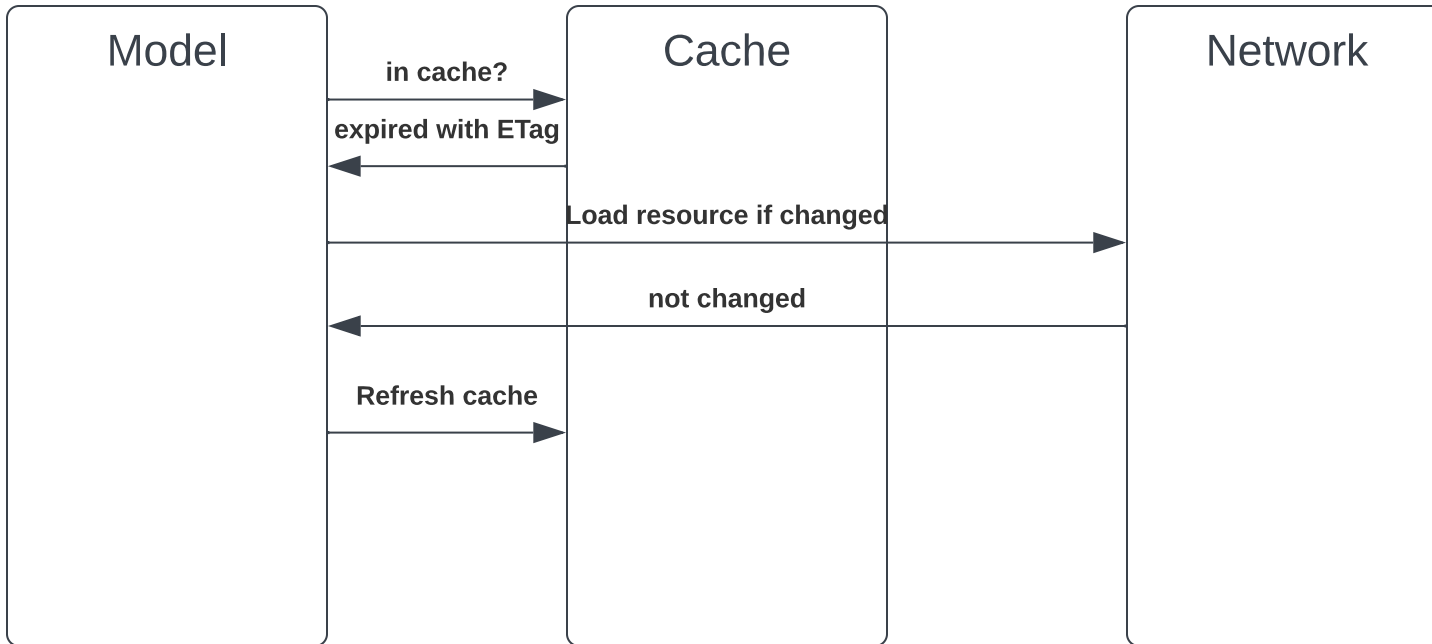
What is an ETag?

The ETag (or entity tag) HTTP response header is an identifier for a specific version of a resource. It lets caches be more efficient and save bandwidth, as a web server does not need to resend a full response if the content was not changed.

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/ETag>

ETag = Unique Resource ID

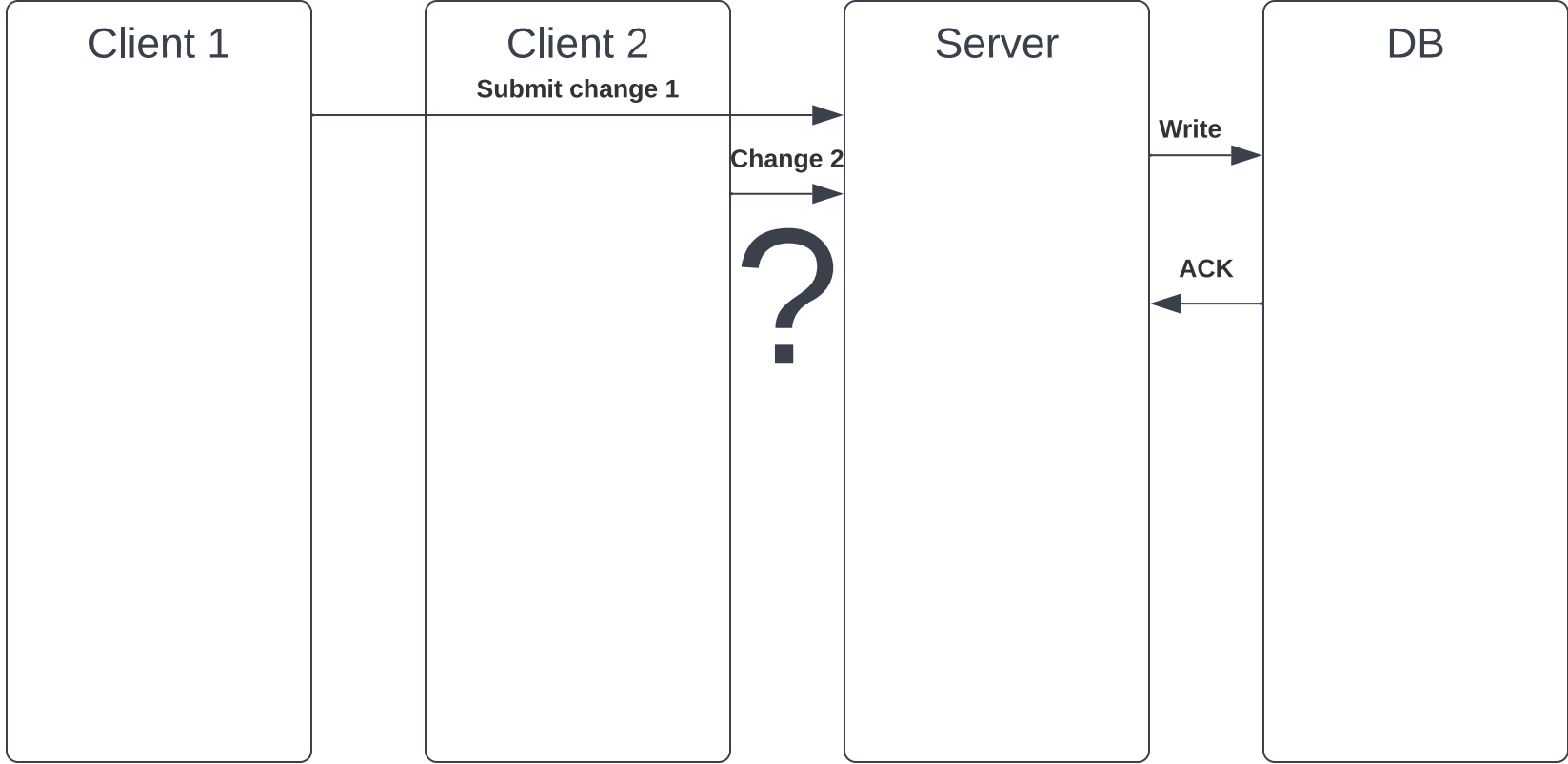
Example ETag: "33a64df551425fcc55e4d42a148795d9f25f89d4"

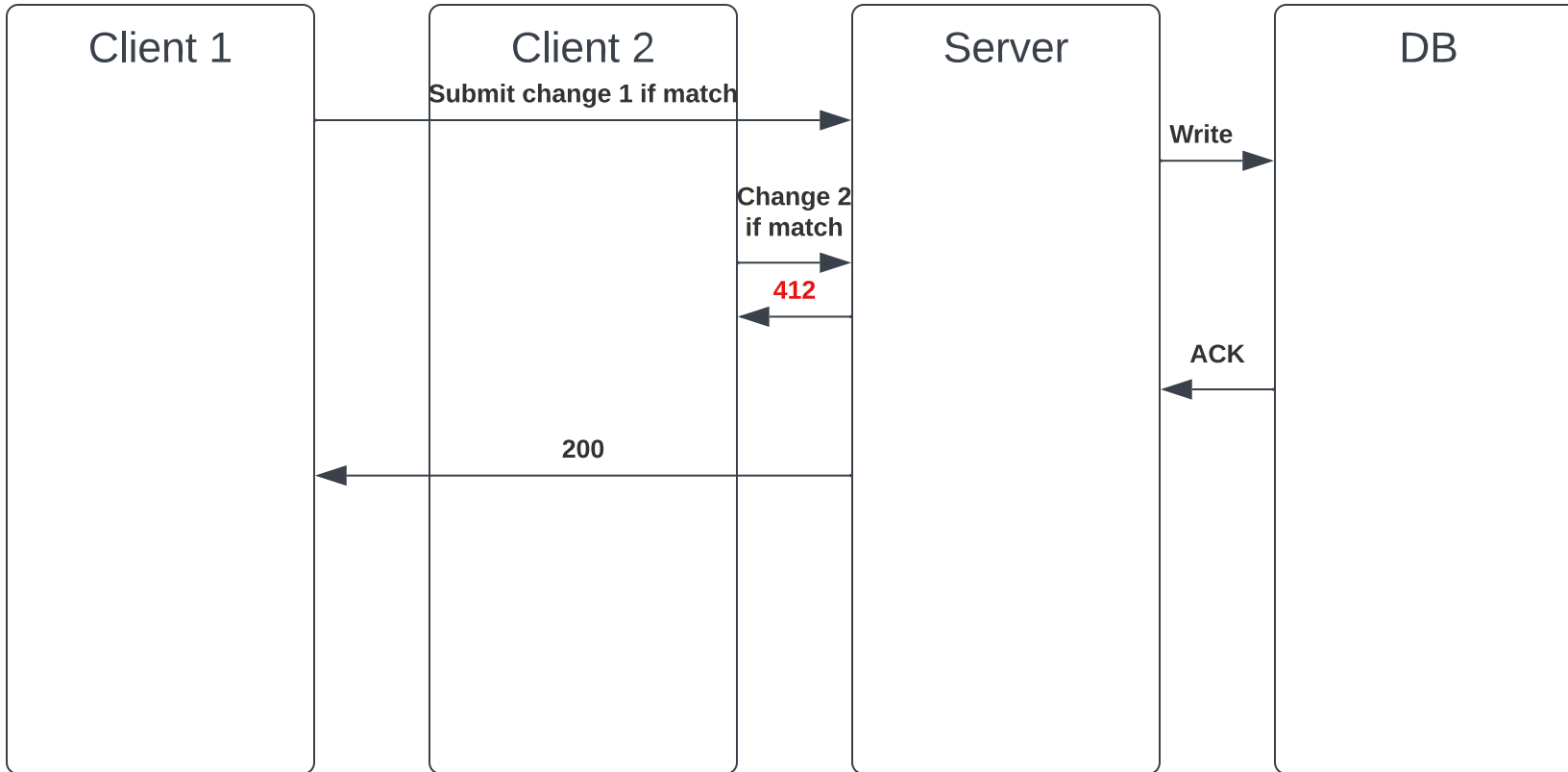


Example `If-None-Match: "33a64df551425fcc55e4d42a148795d9f25f89d4"`

ETag

Mid air conflicts

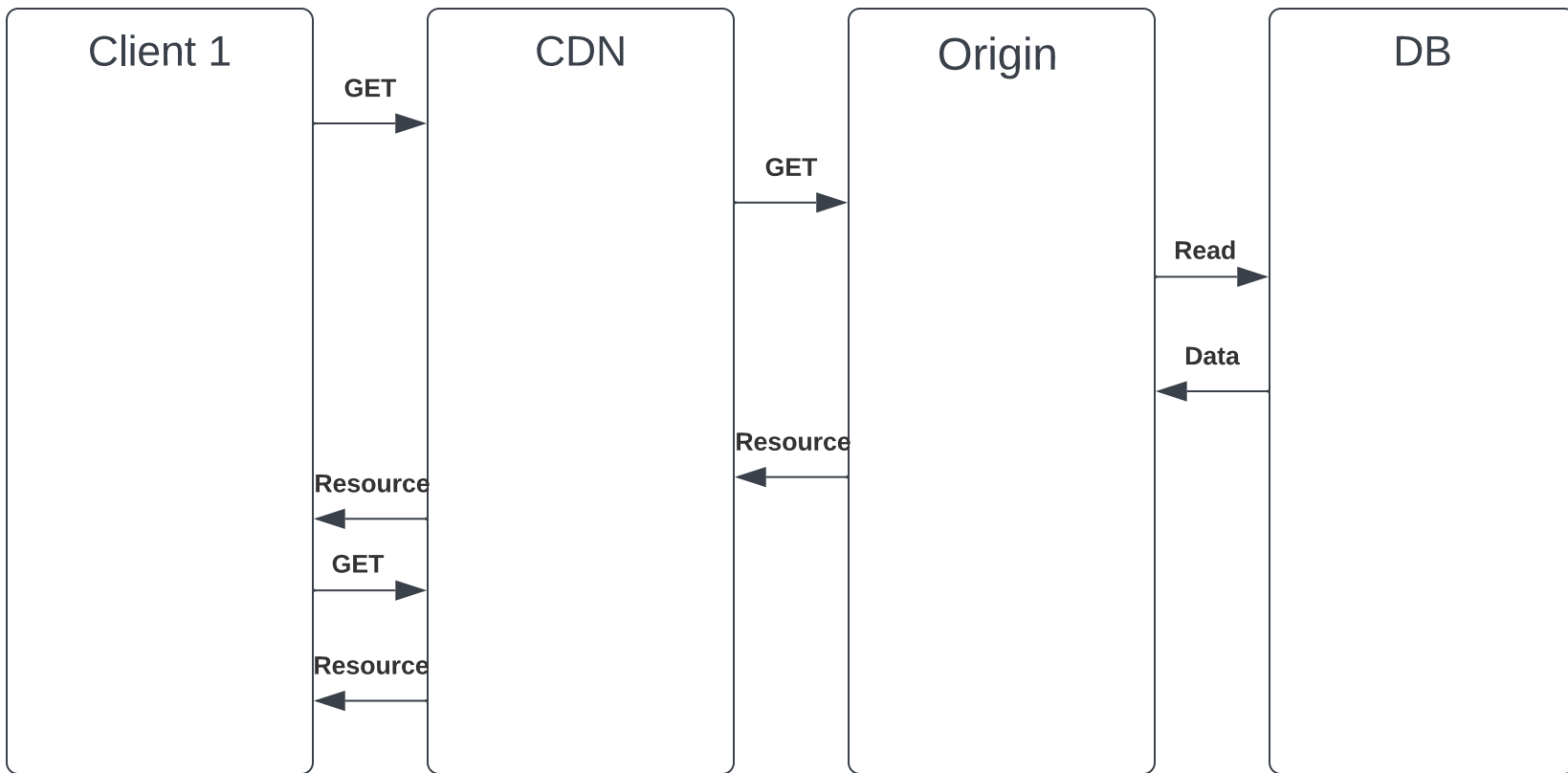




Example: If-Match: "33a64df551425fcc55e4d42a148795d9f25f89d4"

Caching

CD-eh?



Caching

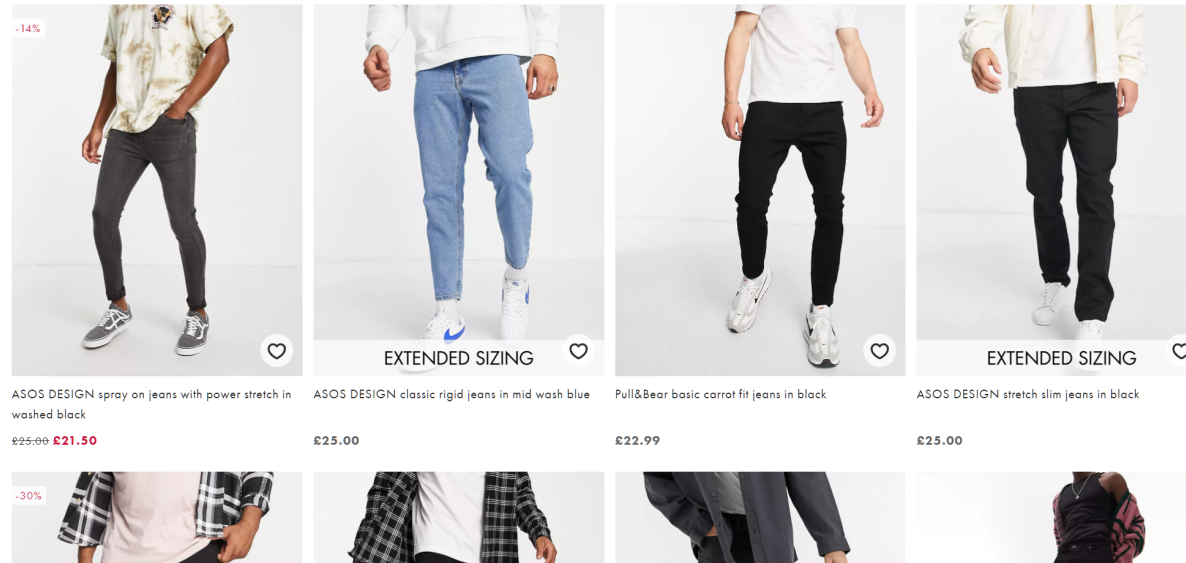
Server side rendered pages

Home > Search results for jeans

Your search results for:
"Jeans"

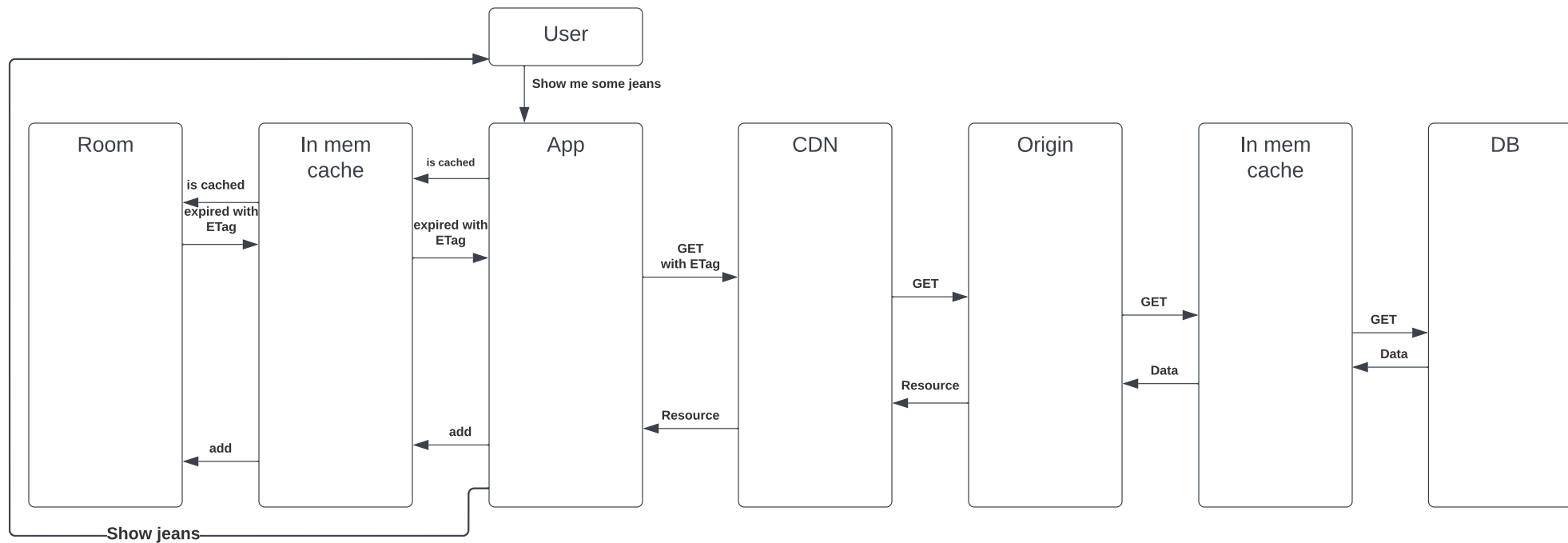
- Sort ▾
- Sale/New Season ▾
- Gender ▾
- Product Type ▾
- Style ▾
- Brand ▾
- Colour ▾
- Body Fit ▾
- Size ▾
- Discount % ▾
- Price Range ▾

9,565 styles found



Caching

Multilayered caching



Web Application/API Design

Web Application/API Design

REST

Give it a REST

REST is a set of architectural constraints, not a protocol or a standard. API developers can implement REST in a variety of ways.

<https://www.redhat.com/en/topics/api/what-is-a-rest-api>

Web Application/API Design

HTTP Verbs

| Oh CRUD

Does this look familiar?

```
https://my.awesome/api/people/1/addAddress
```

```
https://my.awesome/api/people/1/getAddress
```

```
https://my.awesome/api/people/1/changeAddress
```

```
https://my.awesome/api/people/1/deleteAddress
```



`https://my.awesome/api/people/1/` `add` `Address`

`https://my.awesome/api/people/1/` `get` `Address`

`https://my.awesome/api/people/1/` `change` `Address`

`https://my.awesome/api/people/1/` `delete` `Address`

HTTP **verbs**, REST resources **nouns**

POST `https://my.awesome/api/people/1/` `address`

GET `https://my.awesome/api/people/1/` `address`

PUT `https://my.awesome/api/people/1/` `address`

DELETE `https://my.awesome/api/people/1/` `address`

Web Application/API Design

Query parameters

<http://i.already/know?about=those>

TL;DR;

If you ever see this in your api:

<https://something.or.other/login?username=rick&password=pickle>

 from the mountaintops

Web Application/API Design

JWT

| wtj?

Web Application/API Design

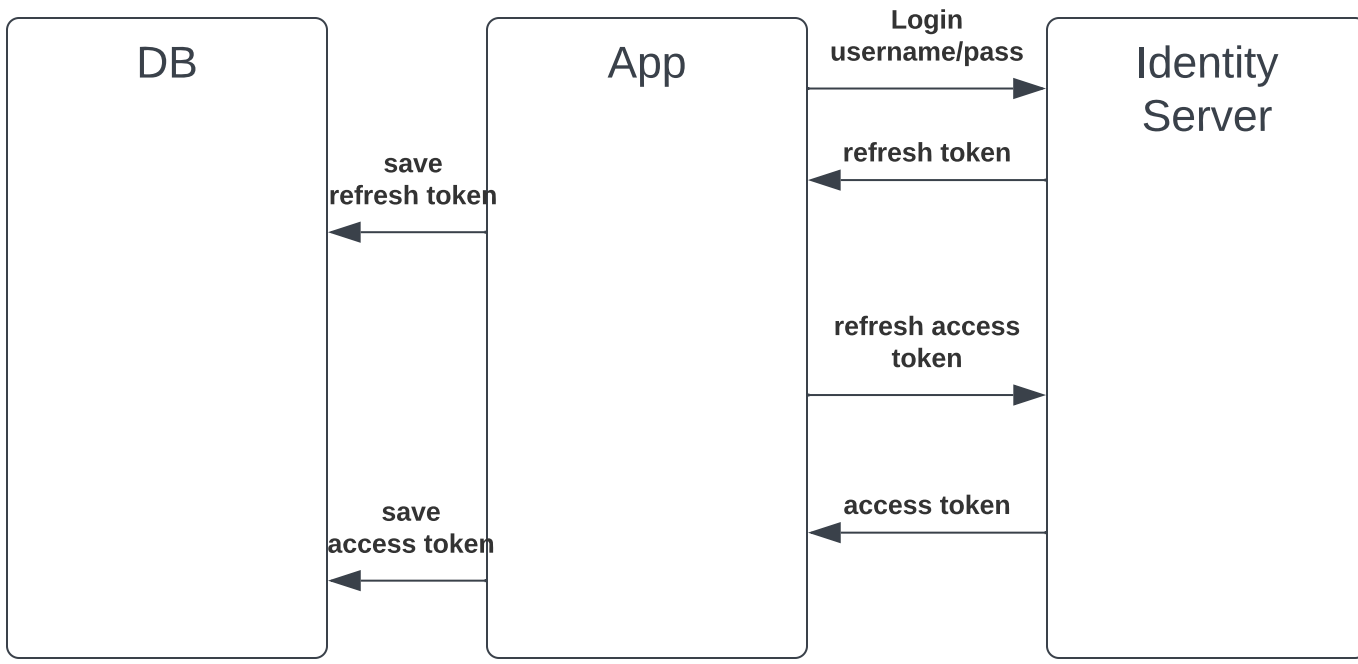
OAuth

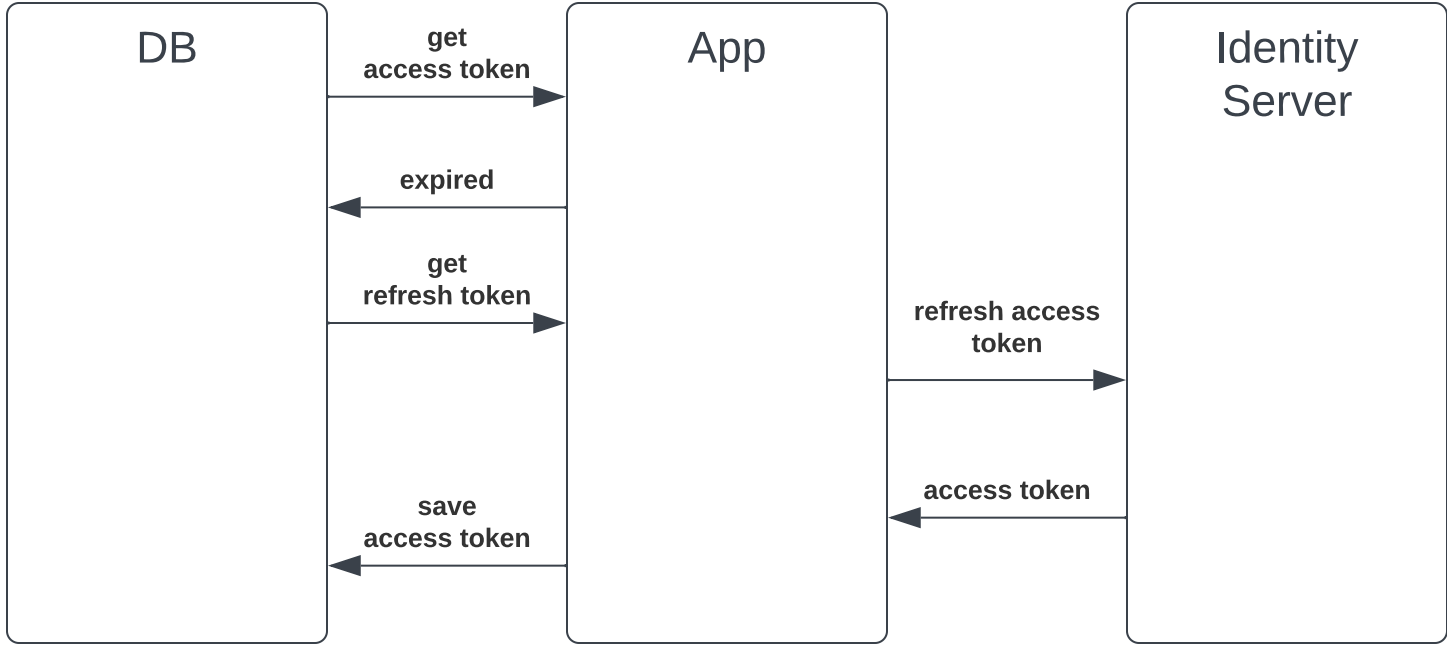
| oh oh ffff...

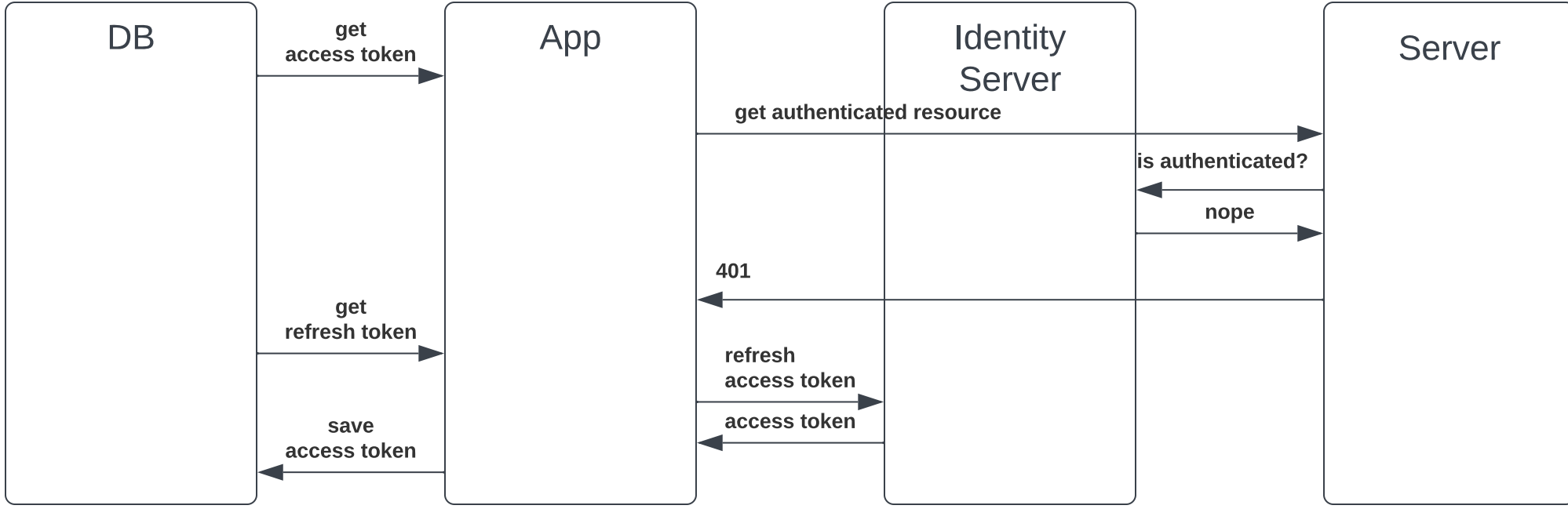
Why OAuth?

Remember this?

<https://something.or.other/login?username=rick&password=pickle>







Web Application/API Design

Idempotency

← Savvas Dalkitsis
3,120 Tweets

Back



Edit profile

Savvas Dalkitsis
@geeky_android

Android geek, TDD nut, ex Shazamer, currently the Principal Software Engineer for apps @ASOS

📍 London 🔗 github.com/savvasdalkitsis
📅 Joined March 2014

124 Following 661 Followers

Tweets Tweets & replies Media Likes

Savvas Dalki... @geeky_and... · Oct 20 ...
I love Droidcon!
🗨️ 1 🔄 ❤️ 📤

Savvas Dalki... @geeky_and... · Oct 20 ...
I love Droidcon!
🗨️ 🔄 ❤️ 📤

 @geeky_android@androiddev.social

ETags 2 - Return of the UUID?

If we follow the REST principles in designing our APIs, we will have automatically idempotent REST APIs for GET, PUT, DELETE, HEAD, OPTIONS, and TRACE methods. Only POST APIs will not be idempotent.

POST is NOT idempotent.

GET, PUT, DELETE, HEAD, OPTIONS and TRACE are idempotent.

<https://restfulapi.net/idempotent-rest-apis/>

Service topology

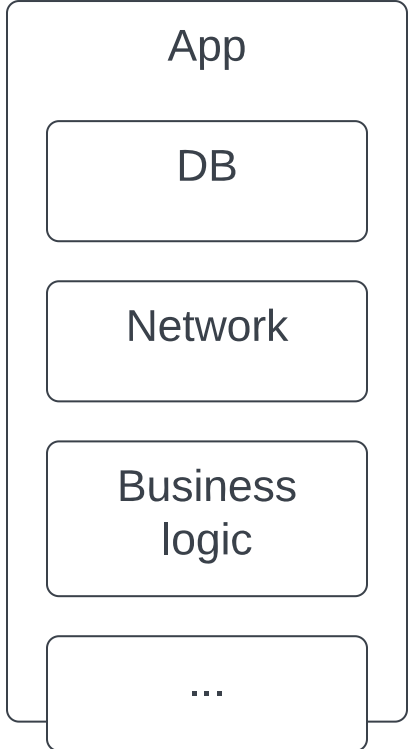


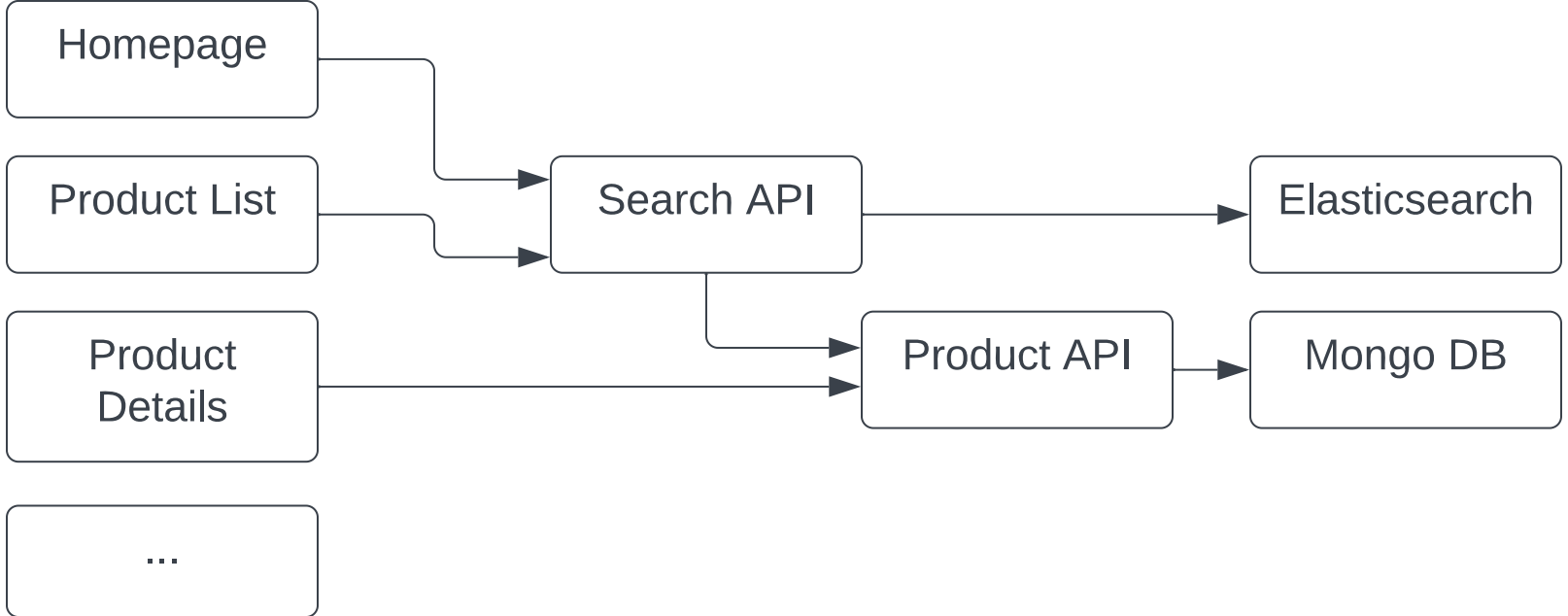
Containers!

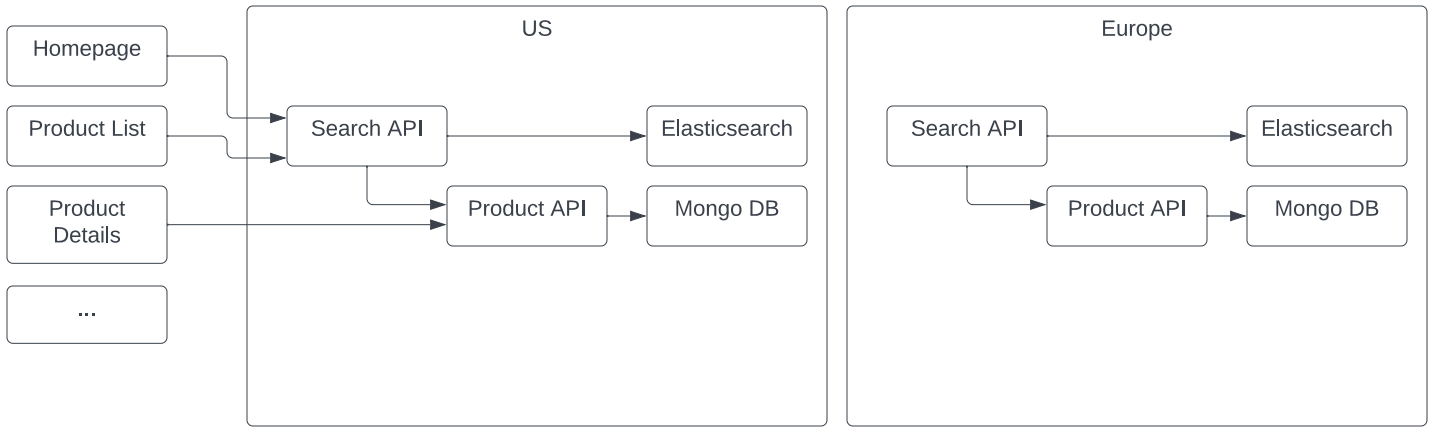


Service topology

Microservices







Service topology

DNS

Service topology

Clusters

Service topology

Traffic Managers / Ingress controllers

`https://my.awesome.service/home`

`https://my.awesome.service/product/123`

`https://my.awesome.service/search?q=jeans`

Load balancers



Image by [Peggy und Marco Lachmann-Anke](#) from [Pixabay](#)

Service topology

Multi-primary replication

Chaos 🐒

Honorable mentions

- Edge workers
- Functions
- Message queues - Event Driven Architecture
- Optimistic Locking
- Eventual Consistency
- NoSQL
- Non blocking sysIO
- Round Robin
- Encryption at rest/In transit - End to End Encryption
- Salting
- Eventual Consistency

Q&A

Slides

