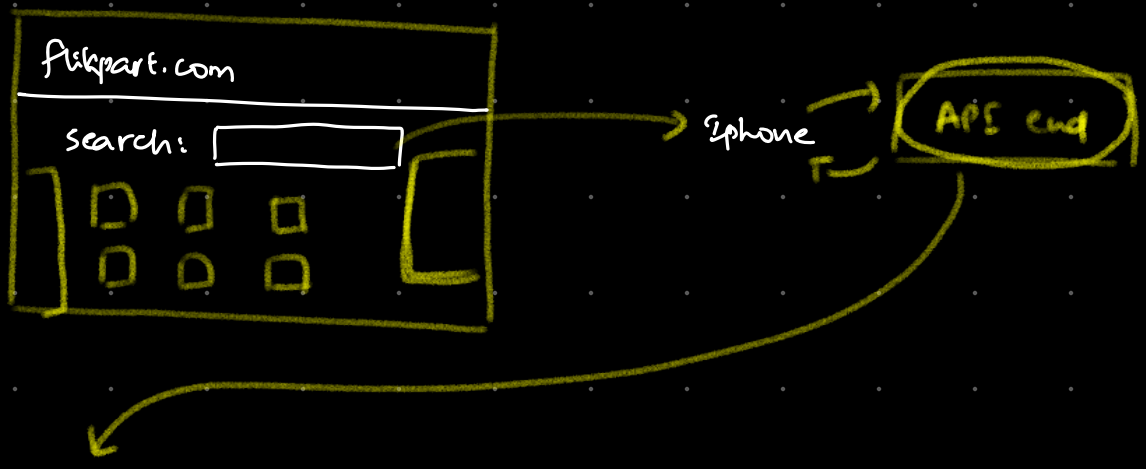
Paging, Sorting, Elastic Search (HLD).

GET vs POST API's.

Searching

How search API's work..

SWE @ fk



url: /search/products/{name} { not a good url }.

/products/
- 1
- 2
- 3
- 4

/users/
- 1
- 2

/movies/
- 1
- 2

POST: /products.

GET: /products/4.

Search query isn't an entity.

↑ arbitrary information.

Query param.

GET: flipkart.com/search/products?q=Iphone ✓

Search often consists of filters, sorting

⇓

```
filters [
  { type: brand,
    values: ["nike", "adidas"],
  } &
  { type: minPrice
    value = 1000
  } &
  { type: maxPrice
    value = 10000
  } &
  { type = fabric
    value = Cotton
  }
]
```

Request body.

According to REST APIs standard, GET cannot have a body.

-> technically its possible to add.

-> restTemplate.

postForObject (url, reqBody, _____)

getForObject (url, _____)

↳ (Request body cannot be passed).

execute (type , ____ , ____)

Query Param

flipkart.com / search ? q = iphone

& filter_brand = adidas OR Nike

& minPrice = 1000 & maxPrice = 10,000

& fabric = cotton.

Cons

⇒ lengthy url

Some browsers will have limit on url length.

in older days :- IE => 2501100 chars } chrome :- => 1M chars

=> 2500 chars.

⇒ Reading all the fields from query and then validating each & logic on all of them won't be very good code.

Search => GET

↳ long urls

↳ becomes complicated to implement filters.

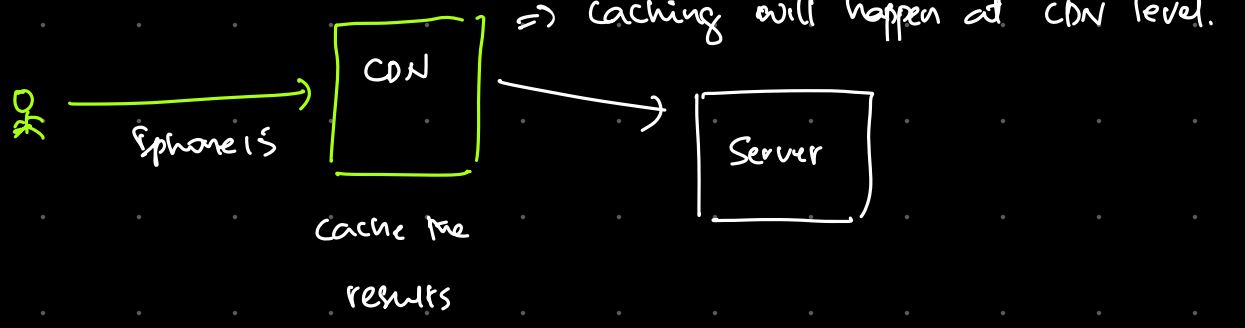
POST

I can pass filters and other info in the body itself.

Pros of GET

1. Shareble URL
2. caching.

CDN's -> Content Delivery Networks.



Search: Iphone

~2-3K products match this search criteria.

response:

```
{
  [
    ==
  ]
  [
    ==
  ]
}
```

-> Too much data
-> User will feel your website is slow
-> fetching a lot of info from DB.

Pagination

1. Instead of sending everything in response, paginate the response
2. Breaking the result set into multiple pieces.

LinkedIn: Only first x posts are sent, when you make another call next set of posts will come.