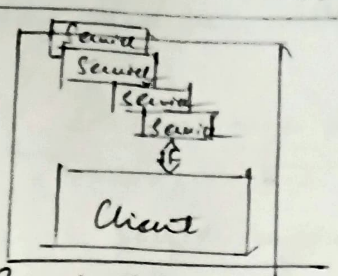
Appconfig.java

Introduction To Web Services

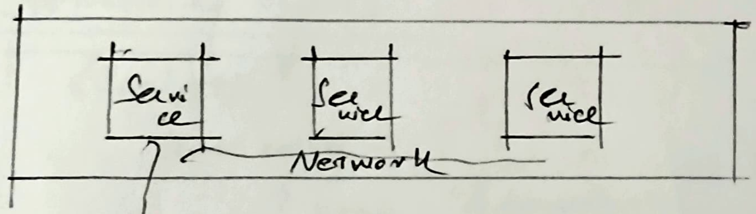


Standard Application

1) Monolithic Application
When everything is in one place all services (frontend, backend etc) is clubbed together at one place.
It is kind of ; Not Scalable.

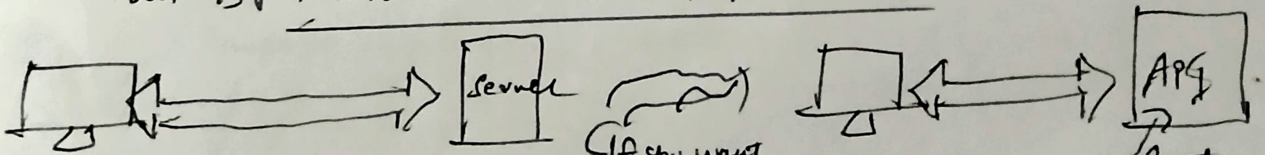
1) Scalable
1) Create different type of services → These services available on N/w or independent as well. When we want to use it the client can consume these services. This Architecture → Web Services

1) diff → System
→ H/w



Independent

1) One of the service not available, your entire application will not go down because only one part is not working. The rest of app should work. This is Micro Service Based Architecture,



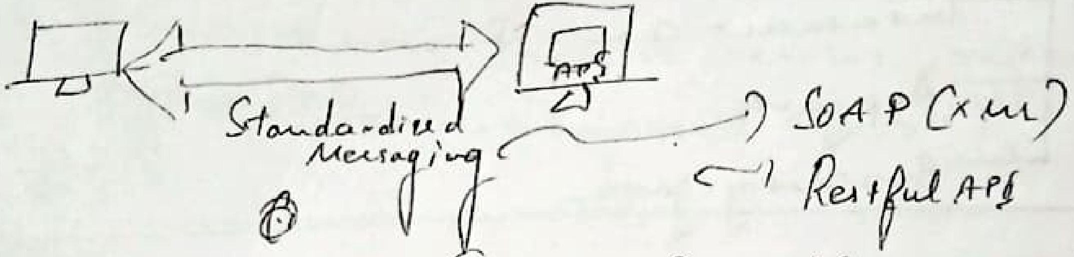
Create web services

If you want to serve diff type of systems

Create API

↳ If you expose APIs → APIs consumed by → front end devices

↳ Frontend can utilize → data → based on req.



↳ When it comes to rest → It is HTTP Protocol / Response codes.

↳ HTTP Methods
- Get
- Post
- Put
- Delete

↳ HTTP Status Code
- 200 — ok
- 404 ~ Not Found
- 500 ~ Internal Server error

foll by pattern

↳ 1xx	Informational
↳ 2xx	Successful
↳ 3xx	Redirection
↳ 4xx	Client error
↳ 5xx	Server error

↳ Link Based URIs

↳ Travel.com/cities

↳ Noun (Pluratform)

↳ Allcities

↳ Filter Based URI

↳ Travel.com/cities/startswith=m

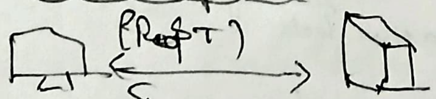
↳ Travel.com/cities/offset=25&limit=50

↳ URL Relatioships

↳ Travel.com/countries/India/cities

↳ Travel.com/countries/India/cities/{id}

↳ Rest response



Information

↳ REST ⟶ Represential State Transfer

RST

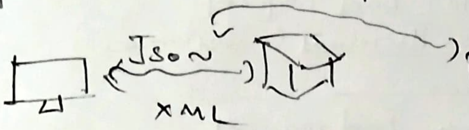
↝ Introduced /defined → 2000
→ Roy Fielding ↝ among → principal authors of HTTP Specification

↝ Design ↝ Based on HTTP 1.0

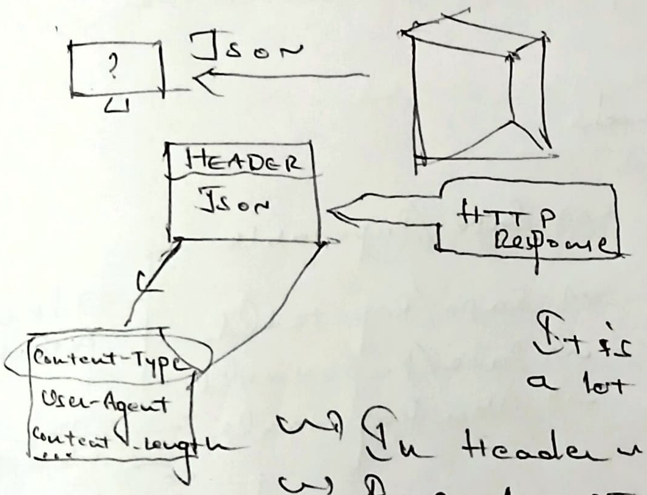
Information
name: John
Age: 33
gender: Male
↓
Representations

↝ JSON
```
{"Name": "John",
"Age": "33",
"gender": "Male"
}
```

XML
```
<user>
<Name> John</Name>
<Age> 33 </Age>
<gender> male</gender>
</user>
```

↝ 

↝ How both client /server ↝ communicate with JSON?



↝ First understand → HTTP Response
↝ Apart from the Data itself we also send Header
↝ Header
It is important because header can store a lot of info.
↝ In Header ↝ couple of things
↝ Prefered → JSON.
↝ Tokens ↝ Authentication/Authorization takes place
↝ It can be transfered using headers as well
↝ Status code ↝ Communication purpose.

↝ Status codes

$\textcircled{5}$xx (Server Error)
↳ Type of code

4xx ↝ 403 → Forbidden
404 → Not found
5xx ↝ 500 → Internal Server Error
503 ↝ Service Unavailable

1xx → 100 → Continue
102 → Processing
2xx ↝ 200 → OK
201 → Created
3xx ↝ 301 → Moved Permanently
304 → Not Modified.

⤳ Operation	Status	Method	Code
READ	Succesful	GET	200 OK
	Not Found		404 Not Found
	Failure		500 Internal Server Error
DELETE	Succesful	DELETE	200 OK /204 No content
	Not Found		404 Not Found
	Failure		500 Internal Server Error
Create	Succesful	POST	201 Created
	Data error		400 Bad Request
	Failure		500 Internal Server Error
Update	Succesful	PUT	200
	Data error		404 Bad request
	Not Found		404 Not Found
	Failure		500 Internal Server Error

⤳ Idempotence of HTTP Methods

Get	⤳ Read	⤳ READ	⤳ Safe / Repetable
Delete	⤳ Delete	⤳ Write	⤳ Safe/Repetable
Put	⤳ Update	⤳ Write	⤳ Safe/Repetable
Post	⤳ Create	⤳ Write	⤳ Non Repetable

Safe/Repetable (Get, Delete, Put) → Idempotent

Non Repetable → Non Idempotent

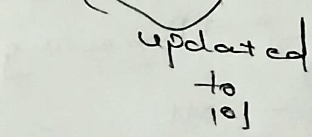
∵ → ID is unique ↓ Car with Id 125 → Already deleted.

/cars/125 ← Delete

☐ ⤳ Delete → Resource ⤳ from Server, If we → once again → won't be any change

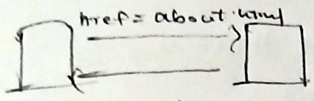
If we ⤳ Rerun a app / Rerun API

/cars/125 ← update



/cars ← Create → It will create same resource (duplicate)

- HATeoas → Hypermedia as the Engine of Application State
- Hypermedia → extension of Hypertext → Text that contains links to other blocks of text.
- response → contain
  ↓
  link → other Block of Media → including → Text, Audio, Image → Video, Still/Moving graphics



- SoAP → Service Specification required (SS)
  - WSDL
- REST → SS is optional
  - Collection → Arrays of links
  - Hypermedia → Derive app..

```
{"id":1,
 "name":"John",
 "links":[
        {"ref": "self",
         "href": "/users/1"
        },
        {
         "rel": "employee",
         "href": "/users/1/employee"
        },
        {
         "rel": "contact",
         "href": "/users/1/contact"
        },
        {
         "rel": "projects",
         "href": "/employees/{empId}/projects"
        }],
}
```

→ Richardson Maturity Modal

Level 0 : → Not a Restful API → Not use (URI — HTTP Methods — HATEOAS)

Level I : → Multiple URI and single Verb → use (Single URI → Single HTTP Method → SOAP Based Payloads → Plain old XML (POX) based)

→ Eg: http://Showroom/manage

→ URIs based on resource

→ Verbs → HTTP Methods (→ Put → Post → get → delete)

→ Single action performed

Level 2 : → Multiple URI & multiple verbs

Level 3 : URIs , HATEOAS , HTTP