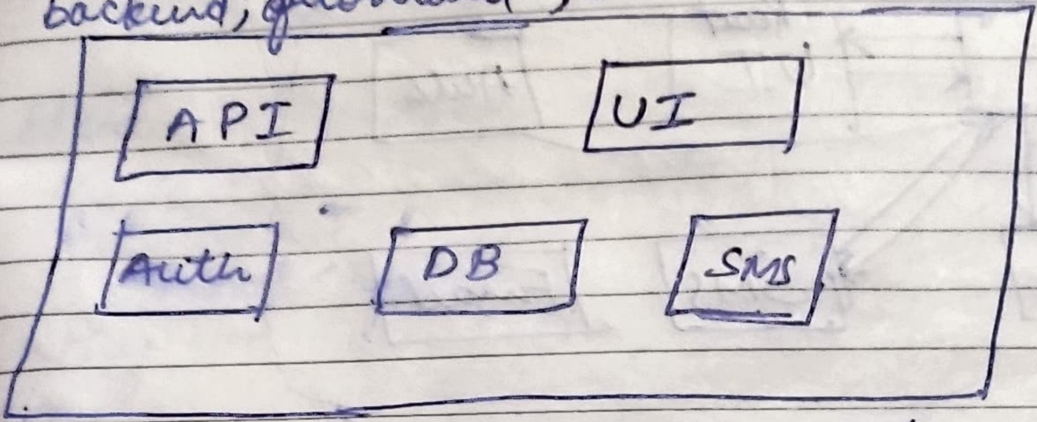
Explore the World → Episode - 06

=) Monolith :- Traditionally app were developed using monolith services

We used to have big project, we have small pieces inside the project, we have backend, frontend, DB link all those inside the project is

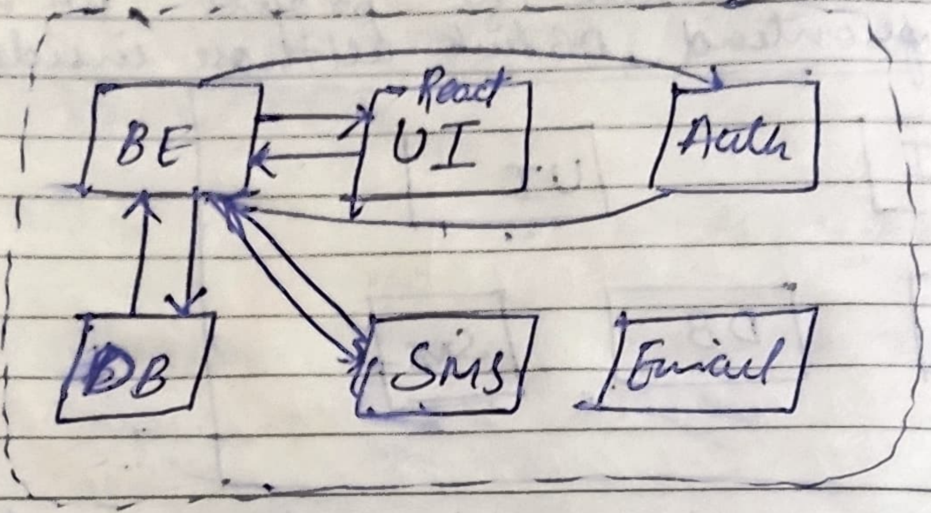


~~People~~ If we have to make small changes (like change in colour of button), we have to deploy the whole project.

=) Microservices → (small, independent & loosely coupled services).

All the different things we need in project like Auth, BE (Backend), UI, DB, SMS, E-mail are all in microservices and they all combine to form the App. unlike monolith which has all the service ~~inside app~~. packaged tightly coupled as a single unit

Separation of concerns and single responsibility principles followed in microservices as small, independent & loosely coupled services are combined to form an app where each & every services has its own job.



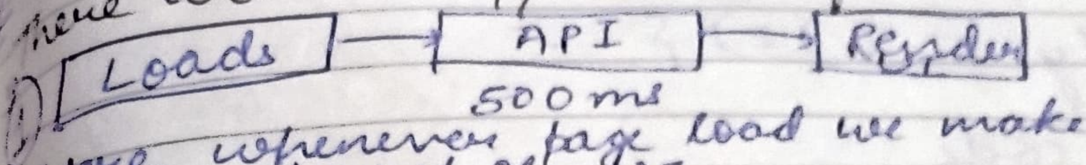
Advantages of microservices

- We can use different ~~services~~ tech stack for different services
- ~~Se~~ Different services can be different port ~~How to connect this services~~? and they can mapped to same domain name.

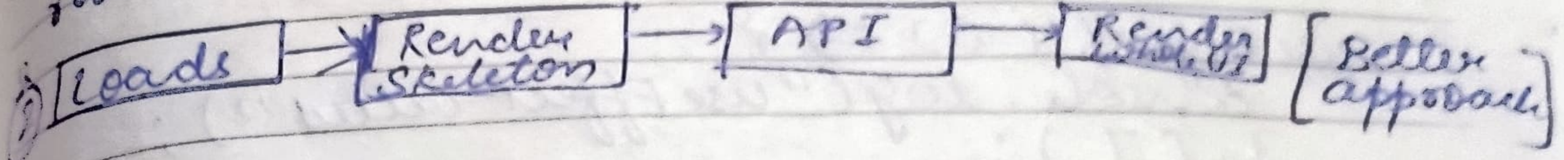
/ → :1234 → UI
/api → :1000 → BE
/sms → :8000 → SMS
→ map to same domain.

How can interact with Backend via UI?

There are 2 approaches for API call & datacom



Here whenever page load we make API call then render the whole UI



Here whenever the page loads we render the skeleton then make API call & then re-render for whole UI

Why 2nd approach is better?
Because we don't make user wait for 500 ms at a page which nothing so showing skeleton is better UX. & ~~As~~ React is very fast to render 2 times

useEffect () hooks

Syntax :- It takes 2 arguments (1 callback function & 2 dependency array)

The callback function is called after the rendering of main function is finished ~~cond~~

Whenever our dependency array is empty.

```
const Body = () => {

//some code

useEffect (() => {
  console.log("useEffect called");
   }, [ ] );

console.log ("Body rendering");
  // some code
~~}~~ console.log ("Body render finish");
}
```

<u>Console.</u>

```
Body rendering
~~useEffect~~
Body render finished
useEffect called.
```

So first our body component render then our useEffect called
so, it is good for using 2nd approach as body will render the API call then Body render with data

Appointments

09 As soon as we get the data we update in the state variable (using useEffect useState function) then react will re-render
10 the page as the ~~how~~state variable data changes.
11

12 We can show the loading spinner before
13 the data comes to not show the blank page
14

But in today's standard, we show the
15 shimmer UI (means the skeleton of page means tells user how our app looks after
16 loading).

How can we make Shimmer UI?

We will create fake cards (or anything we are loading in that page).

SUNDAY

Advantages :

It is psychologically better to show user shimmer UI to make better impressions

09 Why we need state variables? When we use it? Why don't we use normal JS
10 variable?

11 Suppose we have normal JS variable
let btnName = "Login" and we want
12 to change to "Logout" when will click
on a button

13

```
14    < button onClick = { btnName = "Logout" }>
           {btnName}
15     </button>
```

16 But this will only update the variable & will not update the UI but what happen
17 with useState variable?

It will not only update the variable value but also UI.
Suppose

```
const [btnName, setBtnName] = useState ("Logi

<button onClick = { () => btnName = = = "Login"?
             setBtName ("Logout"); setBtnName ("Lo
  } >
   {btnName}
</button>
```

Shimmer doesn't re-render twice cause it was only present during loading phase.

Appointments Now what will happen react will
09 re-render the component with new value and change the UI when we click on
10 button.

~~Suppose we doing this thing button in
11 Header which is inside Body component
So, if we do console.log ("Header
12 render") in Header & console.log("Body
render") in Body we will both
13 will print console 5t because~~ both has same parent (App).

So, state variable update the state &
14 react will track when the state updates & re-render the UI with update
15 State value. ~~React will re-render all the children of App.~~

It happens so fast cause of diffing algo as it
16 is only updating the thing which is different in re-render ~~all~~ though it is re-rendering whole component as ~~header render is printing~~

17 But how come react updating const variable.
Is it bypassing Javascript rule? ~~but body r~~

Actually react is not updating previous variable, at the time of re-render, it is creating a new variable with the new value.

Render 1 → const btnName = "Login"
Render 2 → const btnName = "Logout"

Appointments

09 How to update input box value with a local state variable ?

10

Suppose we have an input like
11 this & a state variable also

```
const [searchText, setSearchText] = useState("")
```

12

```
<input type="text" value={searchText}/>
```

13 So, even if we change the text
14 in the input like we type something
in i/p it will not appear as
15 it bind to searchText which is
empty string so we need to use
16 onChange handler

17

```
<input type = "text"
   value = {searchText}
   onChang = {(e) => {
       setSearchText(e.target.value)
   }}
/>
```

Now the input will updated as we are also updating the searchText.
So, whenever we update the searchText and render the component & it is too fast.

VirtualDOM is object representation of actua

Appointments

09 Whenevee State variable update, react triggers a reconcilliation(Rerender the component cycle)

10

```
useEffect ( ( ) => {
11
     console.Log ( " hello " ); //This runs when
12                              //component mounts
   return ( ) => console.log ("bye"); //This when
13 }, [ ] );                           //component unmounts
```

14 Infinite Scrolling & how to do it is in code?
15 with Explanation!

It uses Intersection Observer class & useRef.

16

17