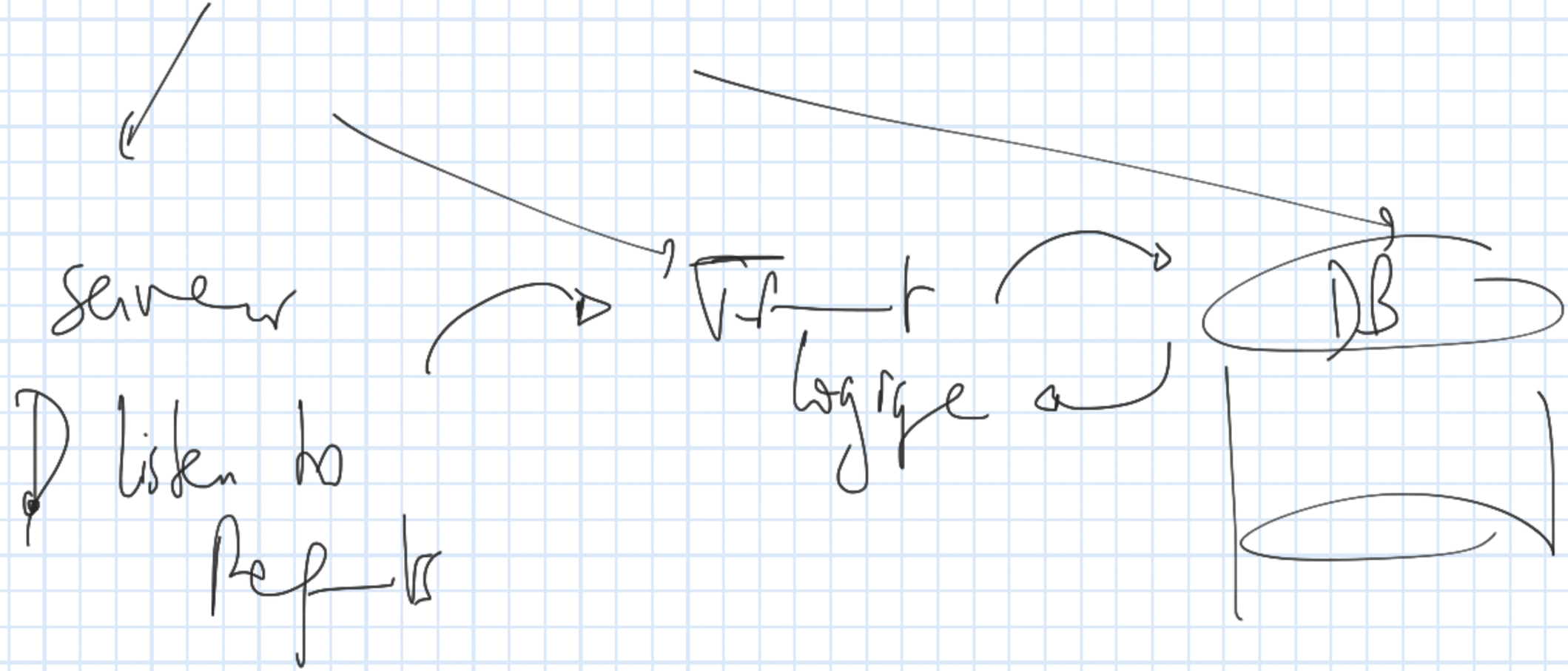
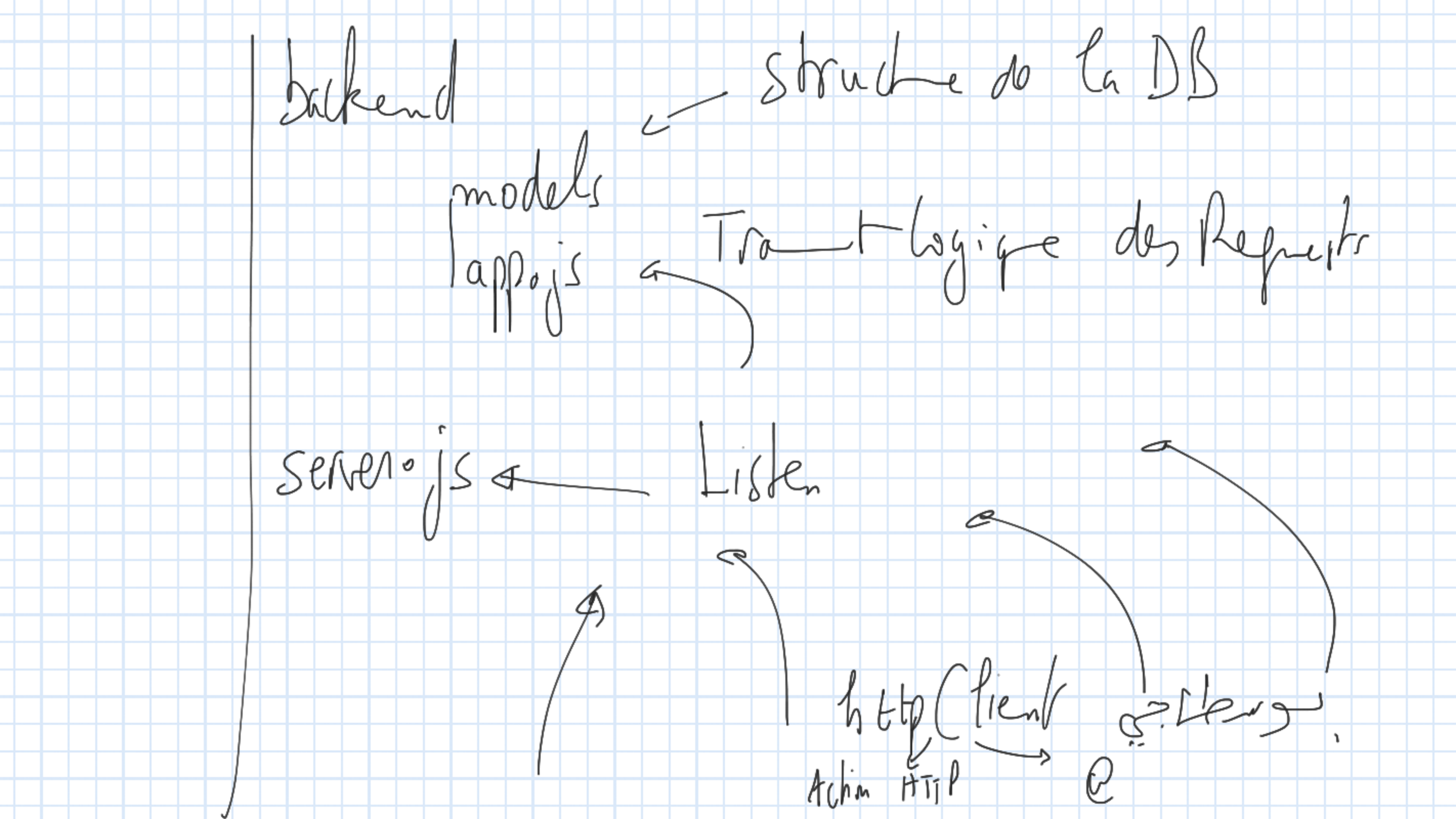
BE : la partie invisible au client




backend
structure de la DB
models
app.js
Traitement logique des Requêtes
server.js
Listen
http
Client
Achim
HTTP

require(" N_Module ")

require(" path ")

$ ng s -o

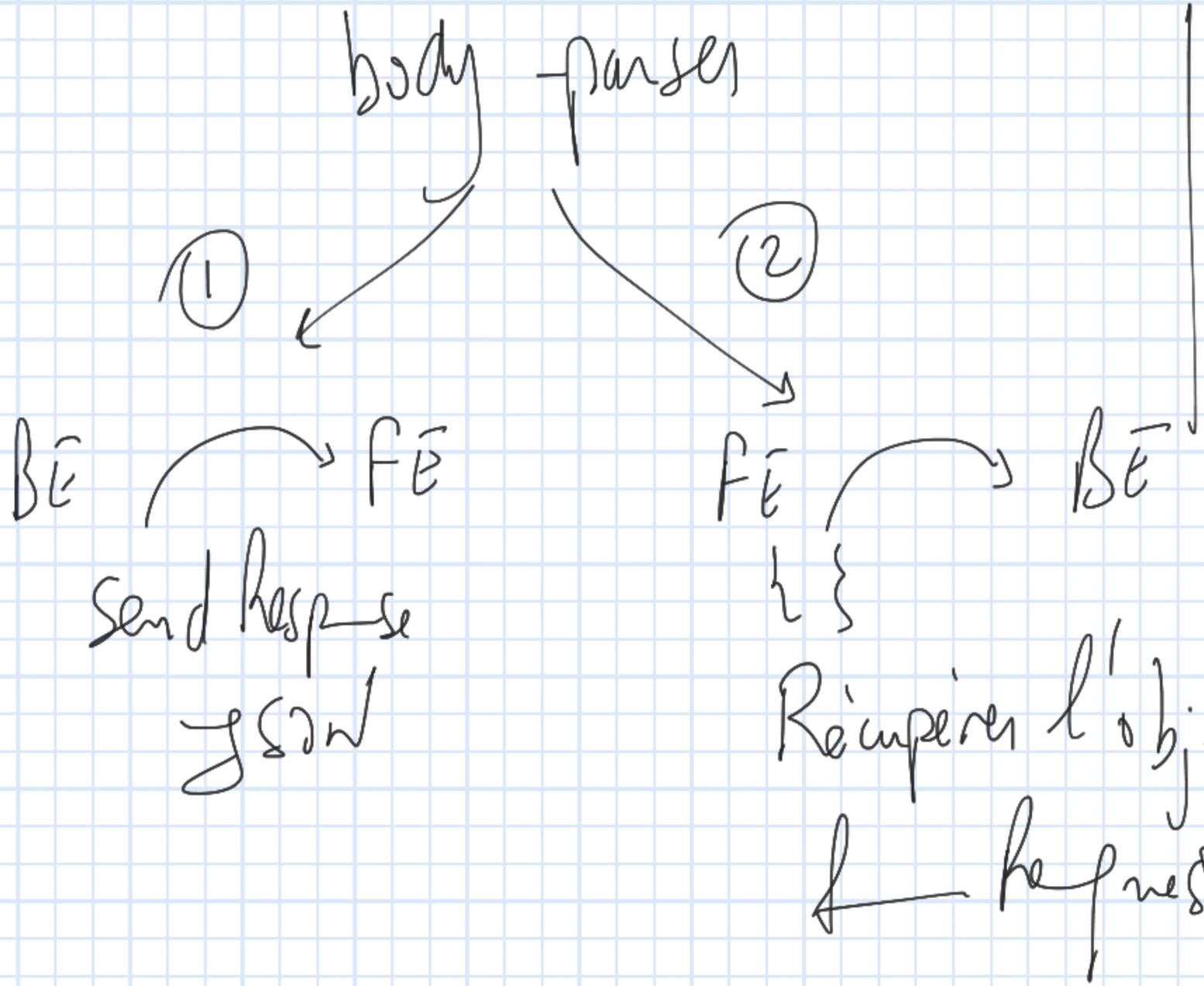
(1)

$ node server.js

→ pas d'auto reload

$ npm i nodemon

(2) $ nodemon server.js



- use : pour la configuration de l'app.

User Service

email pwd

login(user) {
return httpClient.post("http://...:3000/user",)
}

fN lN email pwd

signup(user) {
return httpClient.post("http://...:3000/user",)
}

Player Service

Dans le même projet

⚠ Strictement interdit d'envoyer 2 Req avec la même @ et la même méthode/Action HTTP

React

Angular

Server BE

http://...3000

/sall2

Rep Login

Post {}

http://l...:3000/login

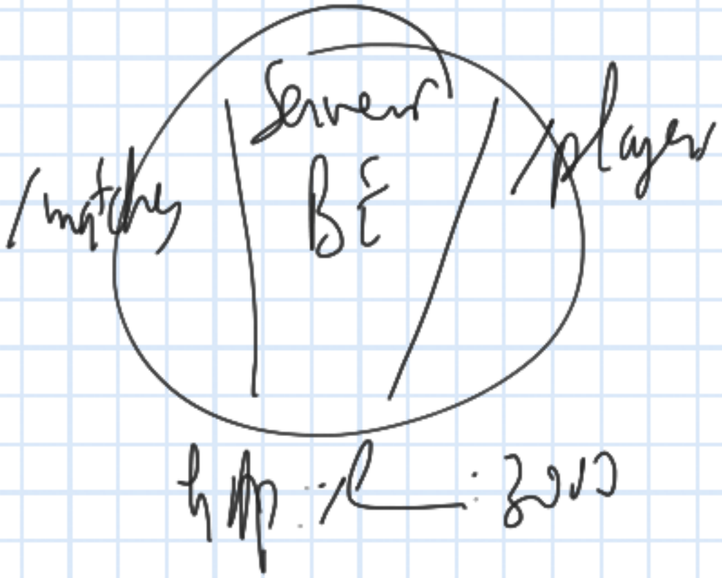
Rep Signup

Post {}

http://l...:3000/signup

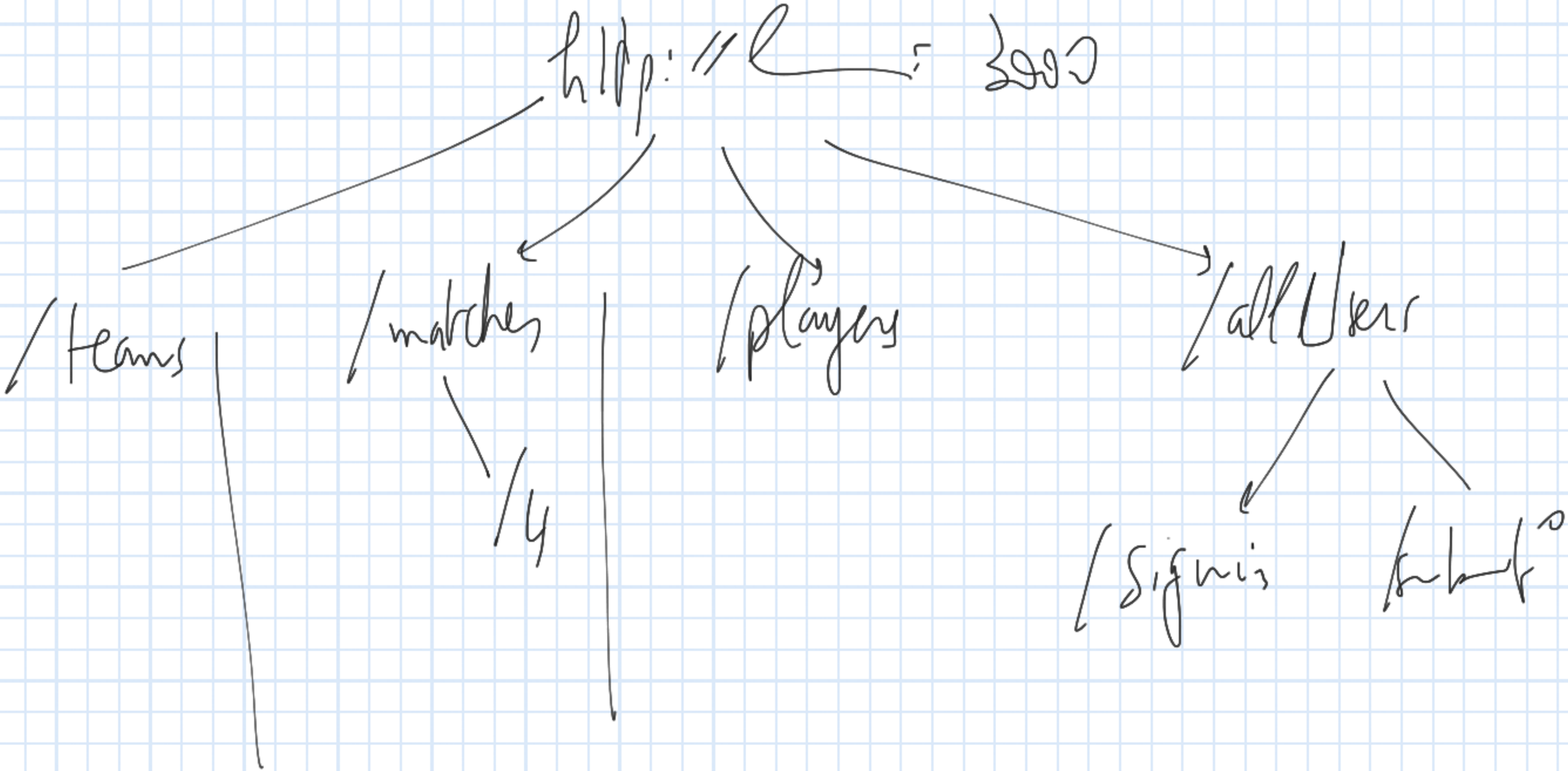
get (http://l___:3000) Get All Matches

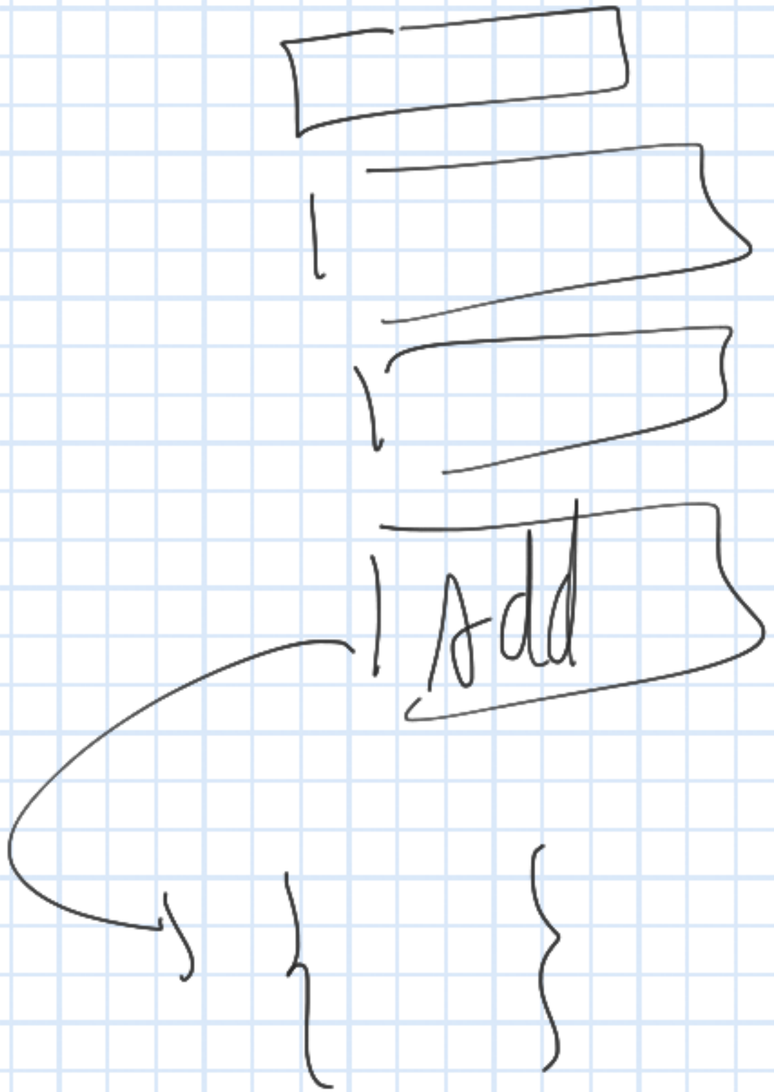
get (http://l___:3000) Get All Players



App Match
/matches

App Players
/players


http://localhost:3000
/teams
/matches
/:id
/players
/allUser
/signin



Service

App Service

Compter

Service

Envoyer des Requêtes

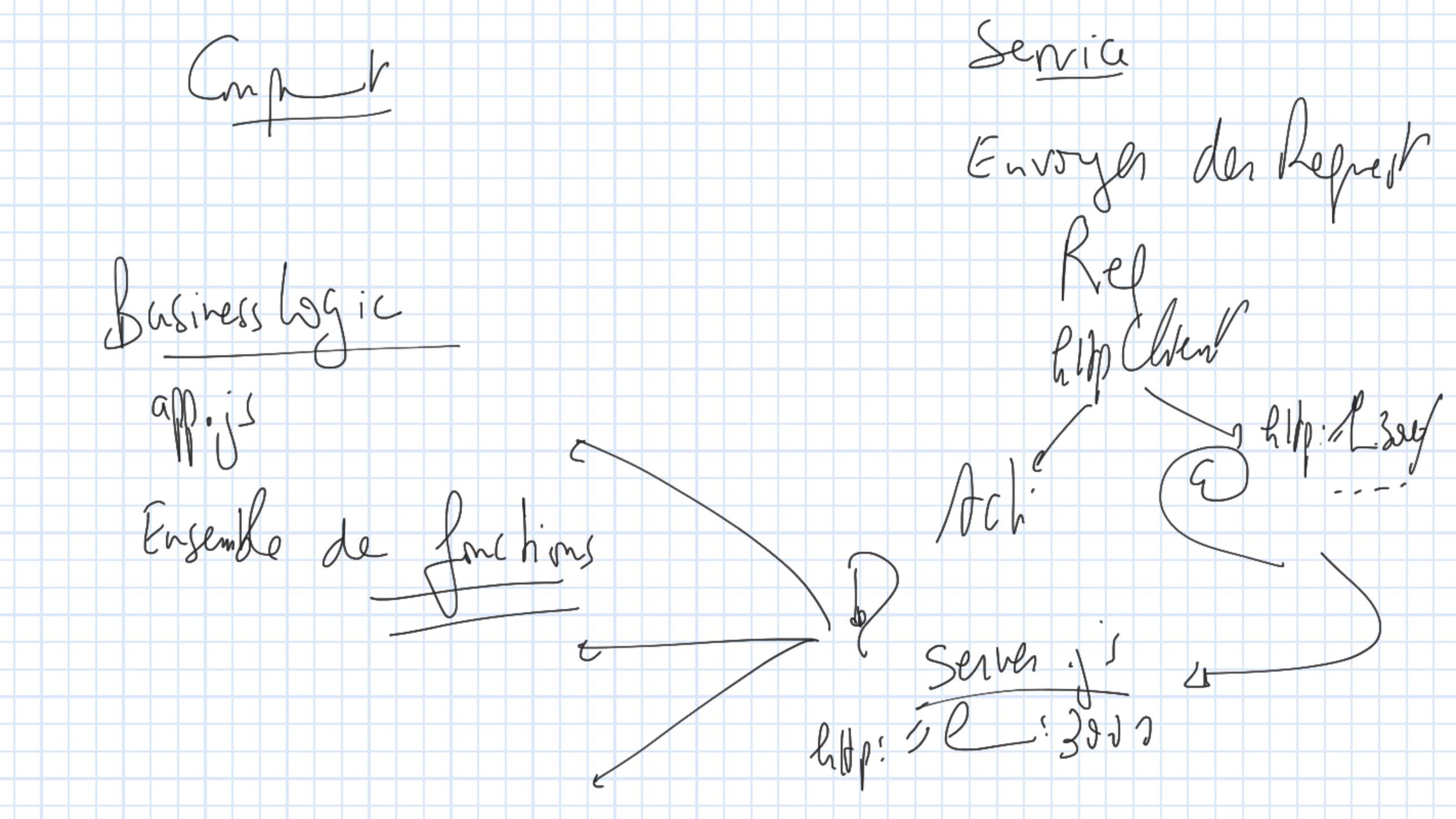
Req

http Client

Business logic

app.js

Ensemble de fonctions



Pour Traiter un Req

Request

Response

```
app.MethodeHTTP("/path", (req, res) => {
```

app Express

Post
Put
Delete
Get

Path du Request

Traitement

```
});
```

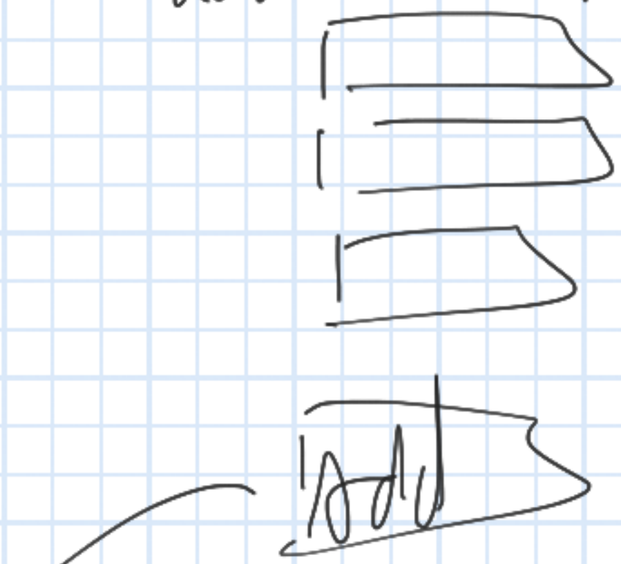
http://localhost:3000/matches/4 variable
8
9
2

app.get("/matches/:id"

variable paramètre

Faire Apl d'un Service ds un Component

add-match. ftn



add-match ors

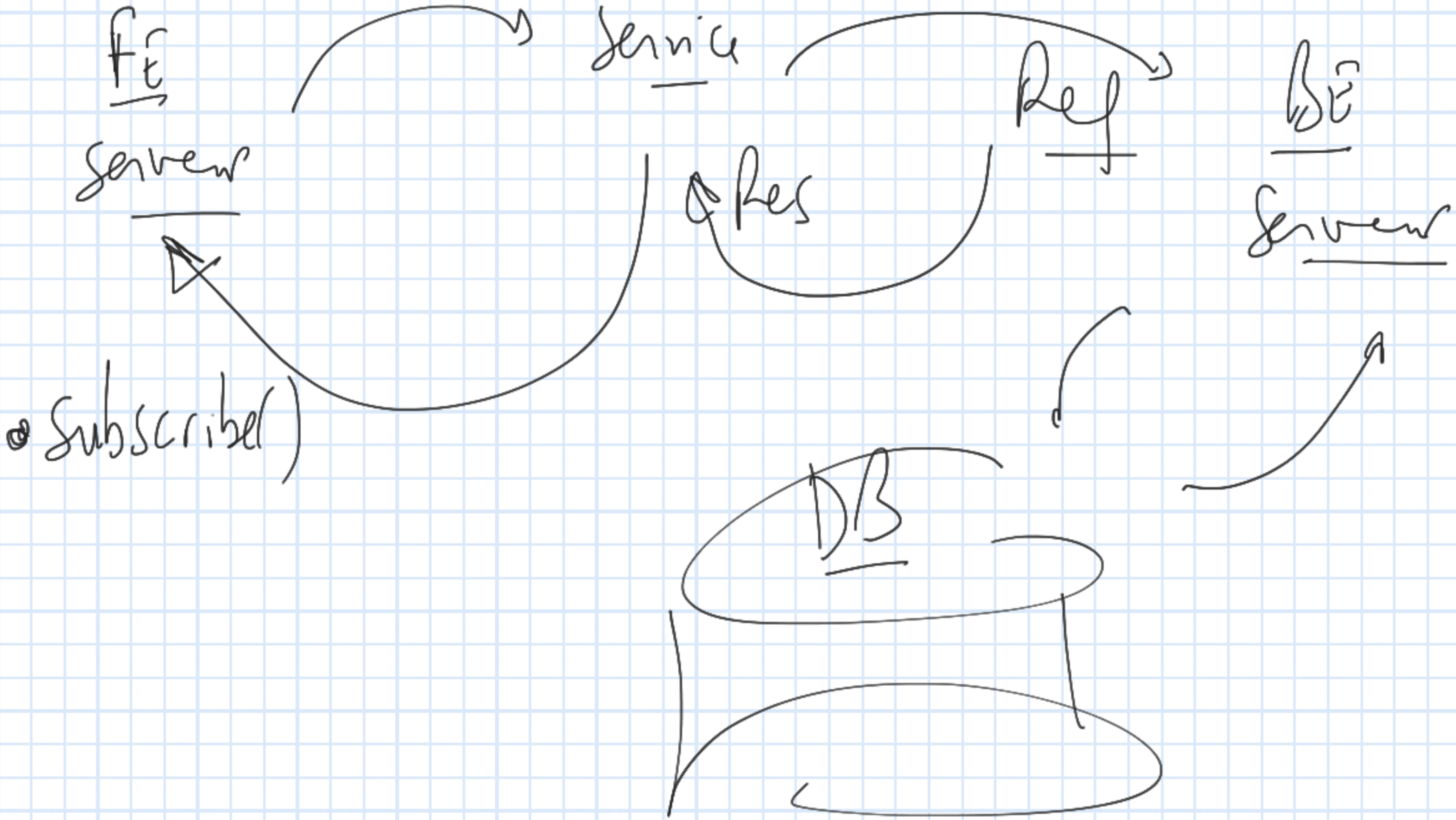
addMatch ( ) {

(click) = " addMatch() "

1) Injecter le service dans le constructor du component

2) Faire Apl à la méthode du service

3) .subscribe()
   → Récupérer la réponse du service


FE
Server
Service
Req
Res
BE
Server
Subscribe()
DB