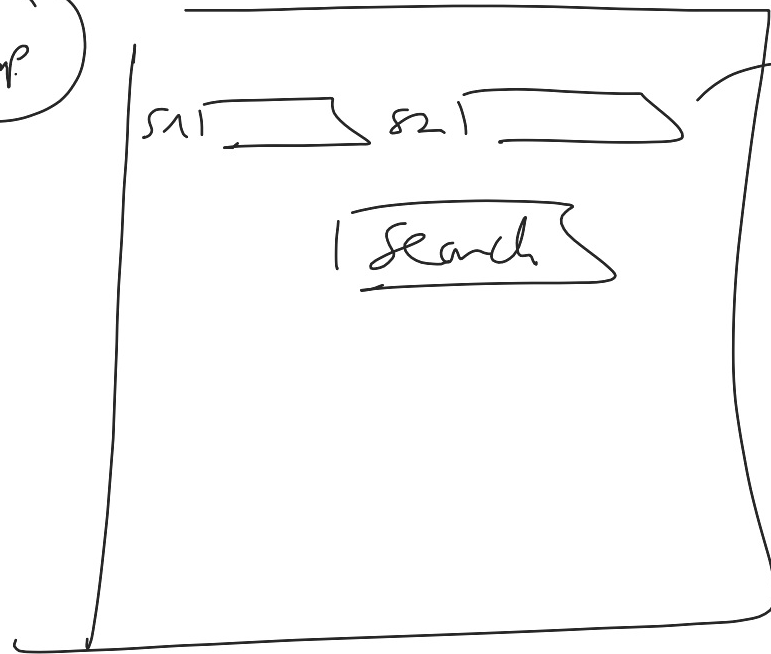


weather. temp



undefined

app.get('---/<sup>undefined</sup>s1/s2')

@Input: décorateur qui permet  
d'échanges / de passer des données d' -

Compt Parent à un Compt Child.

result compt

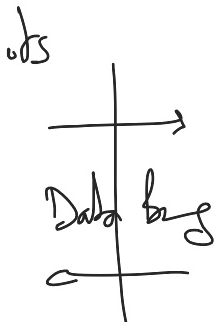
@Input()

matches compt

obs

mlf

@Input()  
resultInput;  
↓  
param



mlf

@Output()



<app-result  
[resultInput] := -->  
<app-result>

matches.tr

0,0  
A,B  
1,3  
C,D  
2,4  
E,F

T: any = [  
un tableau d'objets  
récupérés de la DB  
]

Il faut rendre Result un  
CompT paramétré

@input() ~~X~~: any;

Déclarat  
d-param paramètre du CompT  
Result

matches.html

<div ~~ngFor~~="let elt of T" > appel

<app-result [x]=elt >

1 A 0/0 B

1 C 1/3 D

1 E 2/1 F

</app-result>

</div>

T.length fois

{x.klee} result.html

<div> <p> ~~juu~~ </p>  
<span> ~~x~~ = ~~x~~ </span>  
{x.klee} {x.klee}

<p> ~~juu~~

juu 4/1 nm

</div>

function A ( ) {

}

func B(a, b) {  
    rel      a+b;

}

A ( );

A ( );

A ( );

A ( );

A ( );

B(0, 3); → 3

B(7, 12); → 19

B(4, 8); → 12

Parent

— hhml

$\langle \text{app} - C2 \quad [x] = \text{valeur} \rangle$   
param

$\langle / \text{app} - C2 \rangle$

avec un  
affichage  
dynamique.

---

Child | C2: Déclarer un param  $\bar{x}$  travers  
le décorateur @Input

( x )

affiche("hello");

hello

f

affiche (title = "lg" ) {  
alert ( title );

}

banner

<app-banner [title] = "lg">

@Input() title! : string;

{{ title }}

• ts

• html

attribut

matches . vs

matches : any = [ { one : 4, two : 0 },  
elt  
elt  
elt ]

Parent

matches h.t.p  
<div x:for="let elt of matches">  
<app-result [resultInput] = elt>  
</app-result>  
</div>

delete  
delete  
delete  
delete

Décorateur

result . vs

param du comp - Result .

@Input() resultInput ; any = { } ;

deleteR(id: any) {  
// envoyer -> Req de Suppression  
// envoyer -> Req pr récupérer  
les objets de la DB après suppression  
}

Child

result . h.t.p

{ { resultInput . one } -  
{ { resultInput . two } }

(click) = "delete R"  
(resultInput . -id -)

attribut

matches . vs

```
matches: any = [ { one: 4, two: 0 },  
                  { }  
                  ]  
                  { }  
                  { }
```

Parent

matches h1

```
<div langFor="let elt of matches">  
  <app-result [resultInput] = elt >  
  </app-result>  
</div>
```

delete  
delete  
delete  
delete

Décorateur

@Input()

result . vs  
param decorator Result.

```
resultInput; any = { };
```

```
deleteId(id: any) {
```

// envoyer -> Req de Suppression

// envoyer -> Req pr récupérer  
les objets de la DB après suppression  
Envoyer le tableau vers matches

Child

result.h1

```
{ { resultInput: one } -  
  { { resultInput: two } }
```

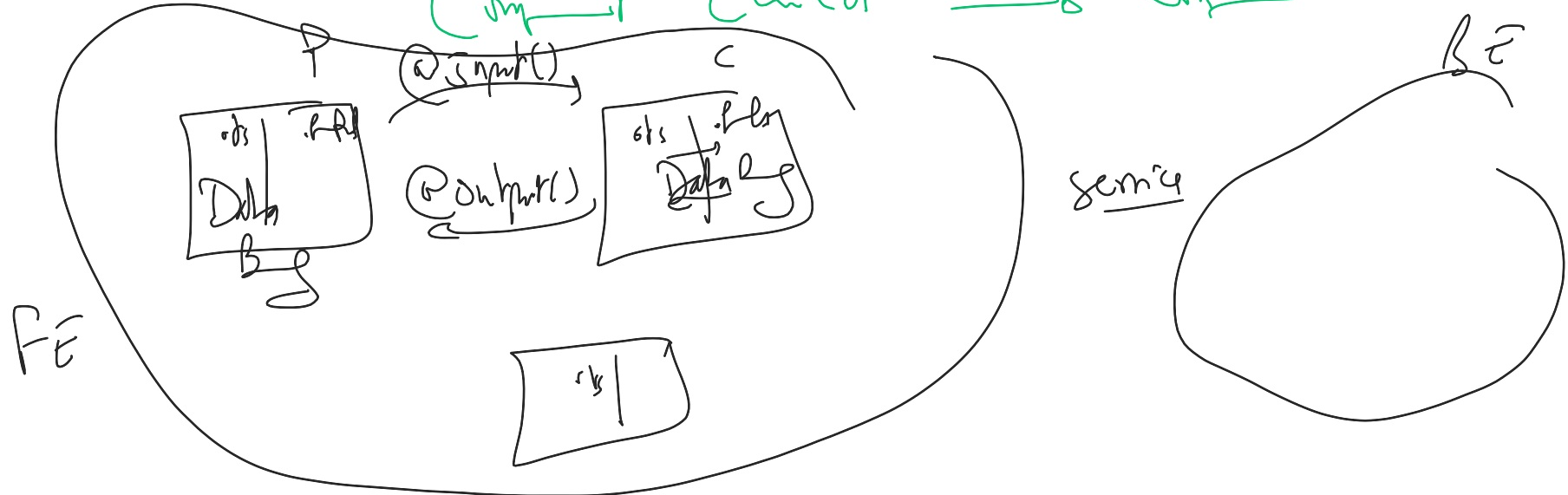
(click) = "deleteId"  
(resultInput = id)

Delete



@Output() : décorateur qui permet  
d'envoyer / passer des données d' -

Compt Child → Compt Parent.



if parent declares @Output() as Child.

Socket.io:  
Node

Brave

Real Time Notificat<sup>o</sup>  
Chat

<u>http://localhost:4200</u> <u>User 1</u>	<u>http://localhost:4200</u> <u>User 2</u>
---	---

Chrome

Real Time

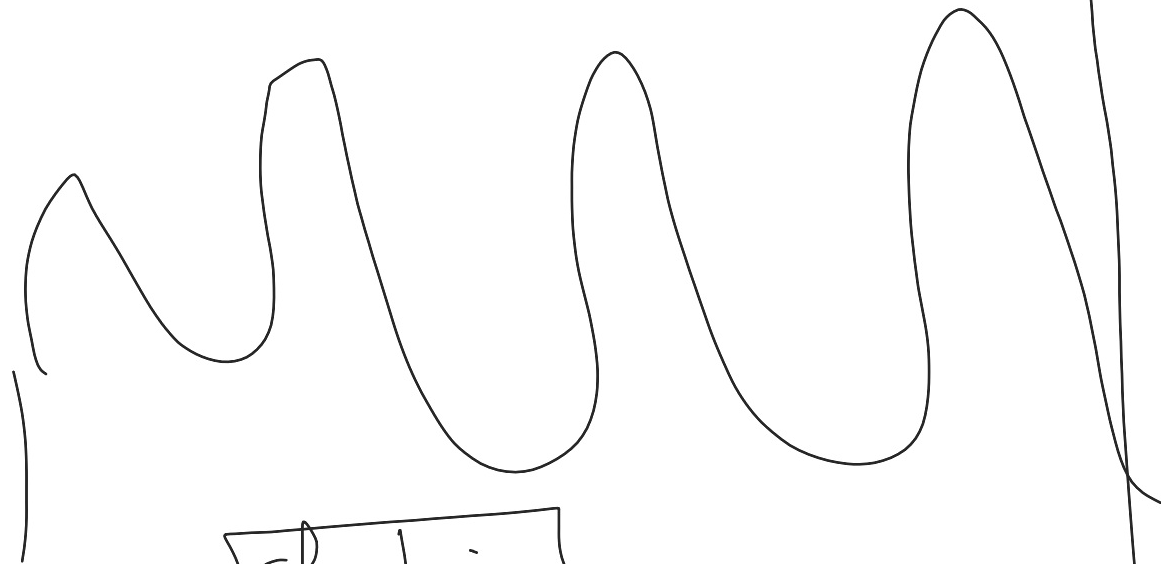


chart.js

Info



API

untable an  
objekt. empfängt die Resultat

```
update(T) {  
  this.matches = T;  
}
```

matches.js

Binding

<div syntax="let dt of matches">

<app-result (matchOutput) = update(event)

untable an  
objekt

result.b

result.l

matchOutput: // send / emit event  
an table an  
objekt

Admin :

Players-table

Afficher que les  
joueurs ayant un  
age  $\geq 30$

Player.find { age : 30 }

chercher les joueurs ayant  
un age == 30

Player.find ( { age : {  $\geq$  30 } } ) where

attr du Modèle

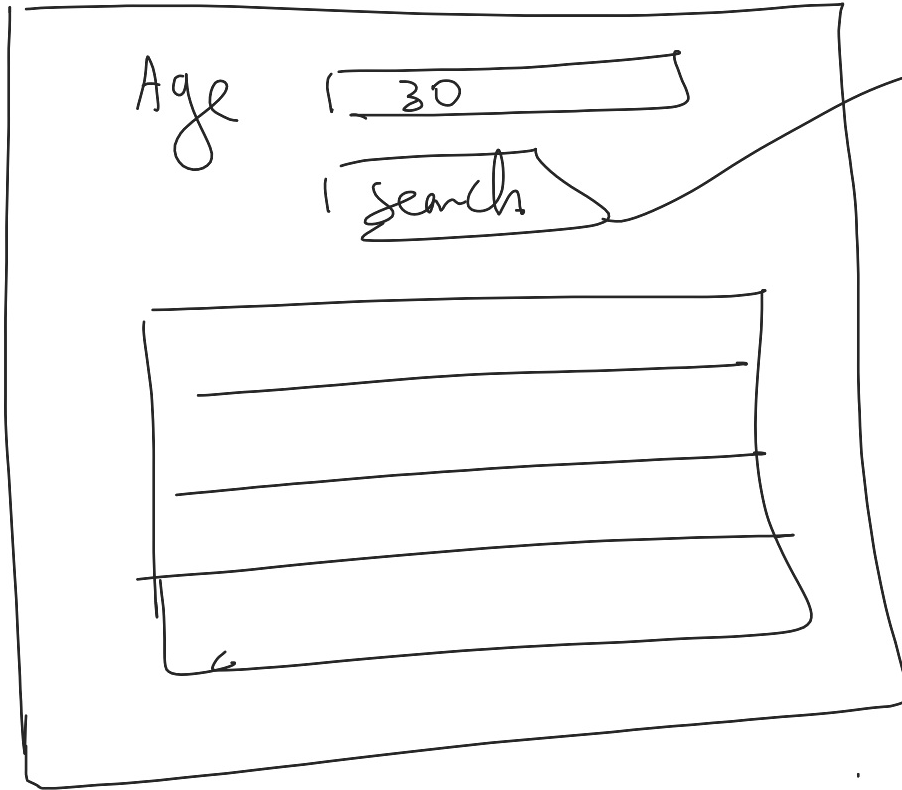
attribut  
du modèle

param du  
find

players

$> : \text{\$gt}$  |  $< : \text{\$lt}$  |  $\leq : \text{\$lte}$

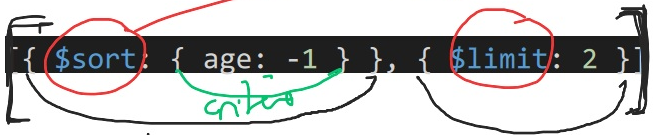
ngOnInit()  
get all players  
↓ req  
↓ req  
↓ BL



(click) = 'search' "  
↓  
search() {  
 // req 2  
 ↓ BL  
}

Retourner la liste des players ds  
un ordre Décroissant par age.

Tri par  
age ds  
un ordre ↘

aggregate( [  ] )

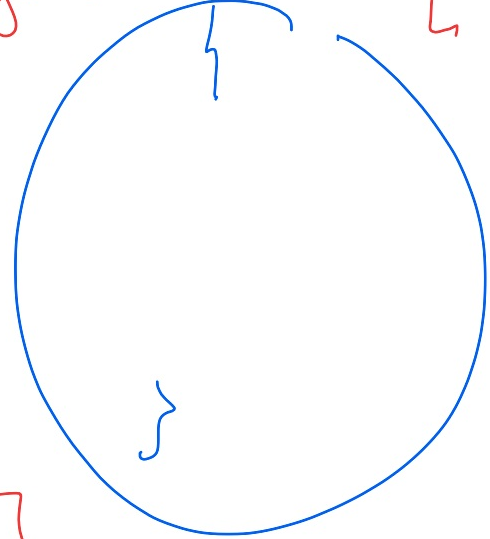
0 1

```

{
  "_id": "68b831c9dad5200b503a2788",
  "name": "Ines",
  "age": 28,
  "position": "DEF",
  "playerHeight": 175,
  "playerWeight": 92,
  "imc": 0.003004081632653061,
  "nbr": 76,
  "tld": [
    {
      "_id": "68b5806ae9f166307c879b05",
      "playerIds": [
        "68b58221e9f166307c879b14",
        "68b831c9dad5200b503a2788"
      ],
      "name": "RMD",
      "owner": "SALAH",
      "foundation": 1920,
      "__v": 2
    }
  ],
  "__v": 0
}

```

player.tId → [



]

player.tId [0] . name

player.tId [0] → { }



Conception : ( Au moins 2 jours )

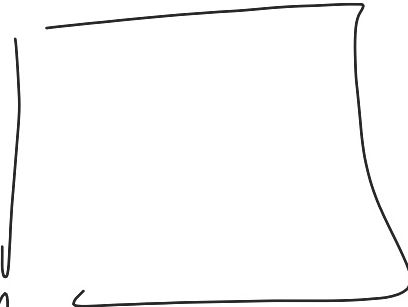
Module Course



Module Classe

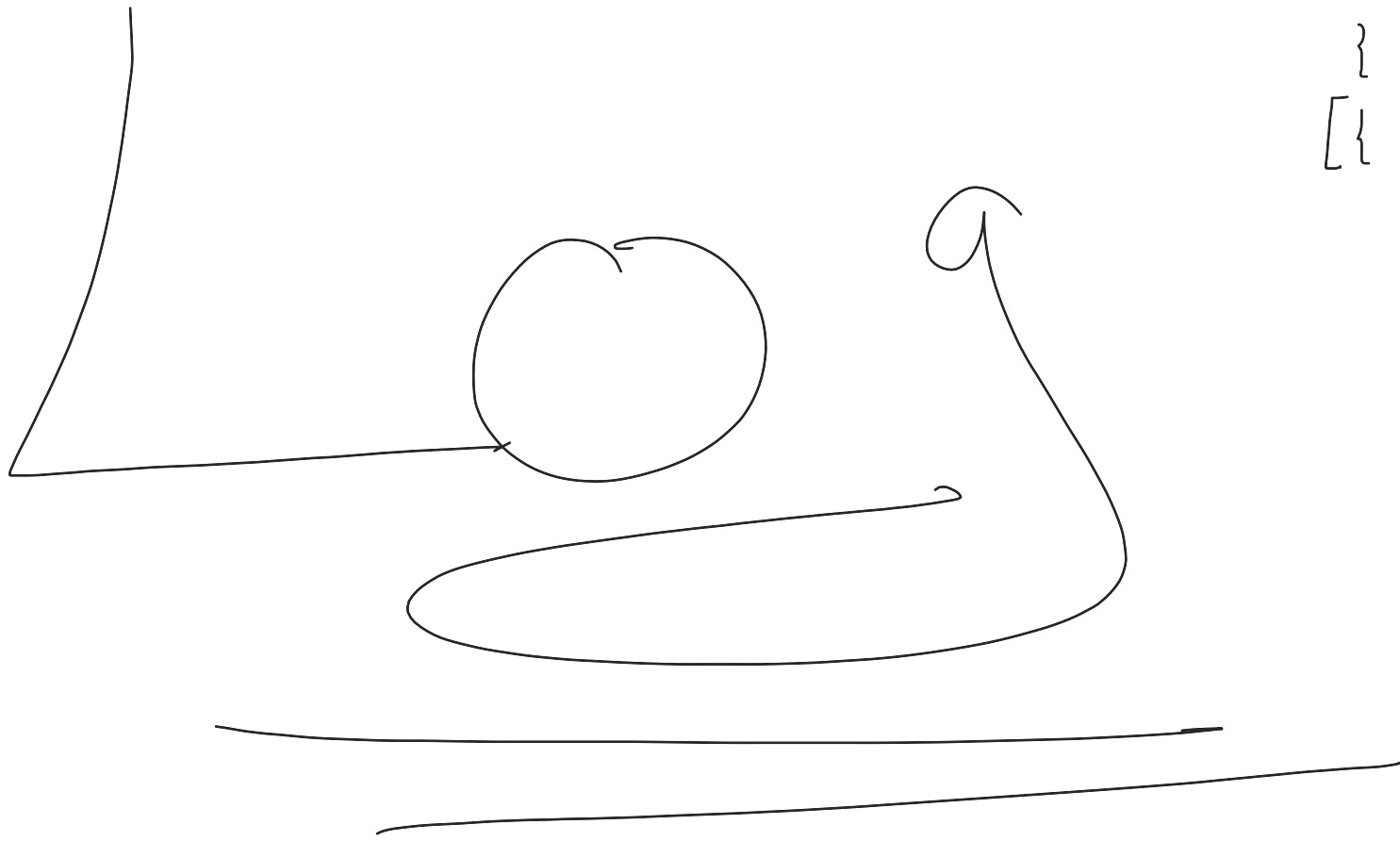
name  
studentsId: T  
teacherId  
courseId: ]

Module Uses



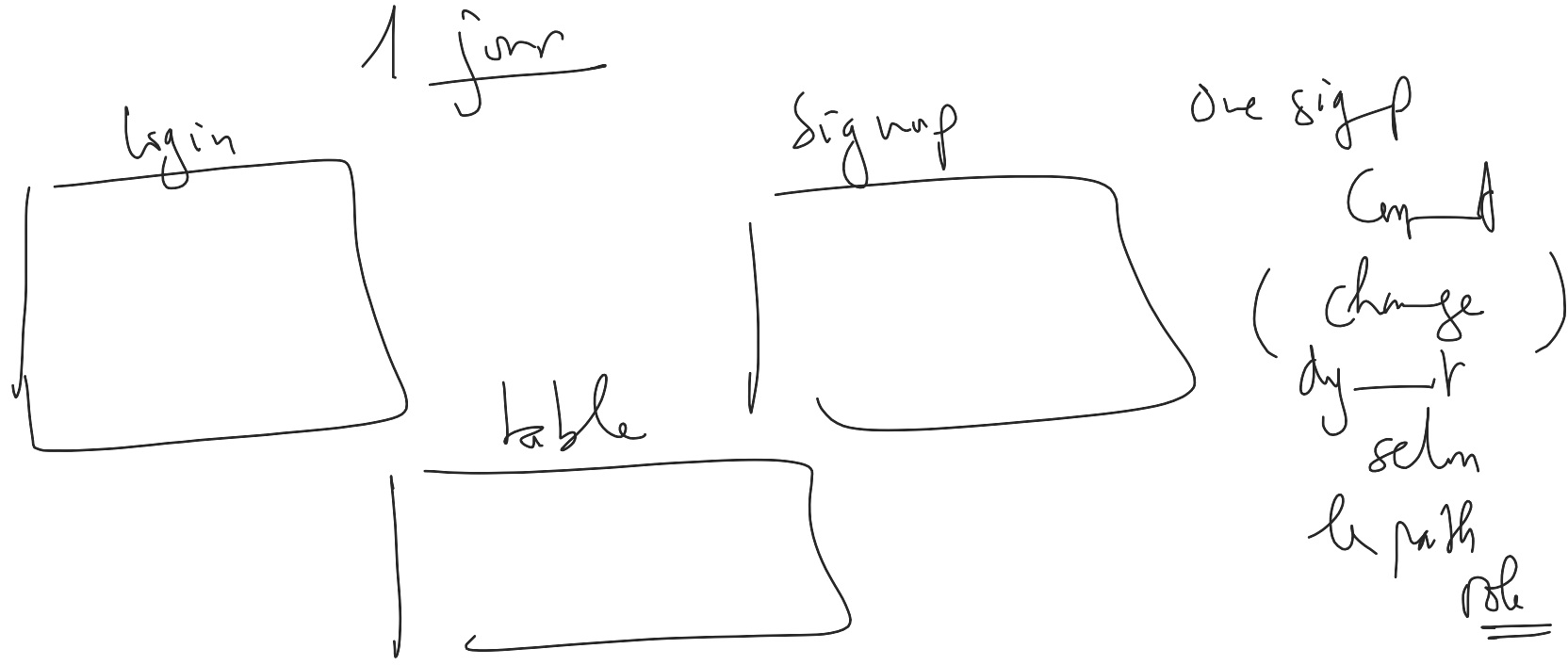
Module: Evaluation

note  
eval  
courseId  
studentsId

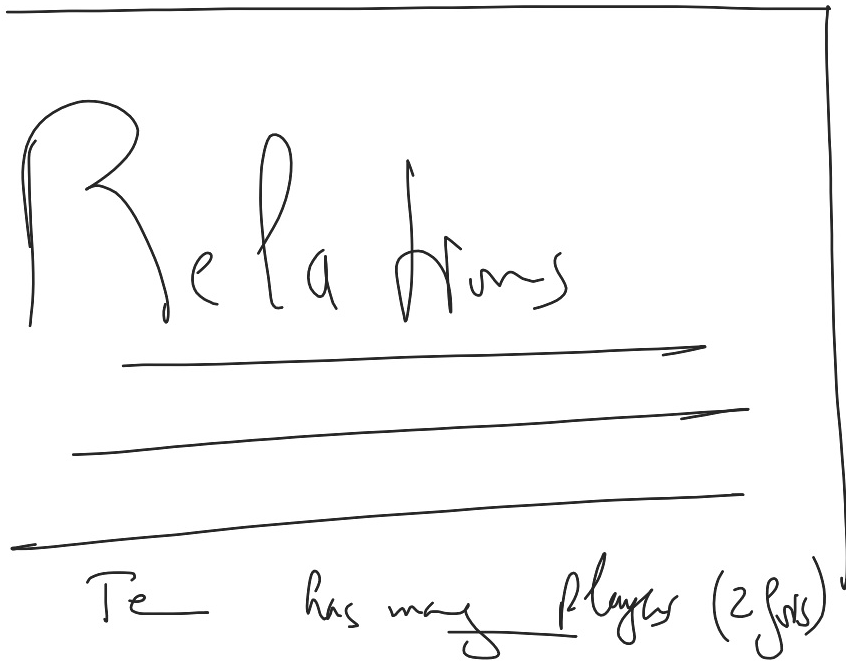


$\left\{ \begin{array}{l} \{ \\ \} \end{array} \right\}$

Pas de perte de temps ds le template :



Un seul modèle: User



on déclare  
fs les  
attributs  
possibles

role:

userInfo

userInfo

80%

↓

50%