

1

BACK-END

2

Planning

15/5 Cours Node et Mise en place du back dans le projet

22/5 Cours Evaluation croisée

Développement site et préparation du test croisé

30/5 Finalisation site

5/6 Séance de test croisé : possibilité de faire tester à l'ergothérapeute

6/6 Evaluations

Ecrit QCM Techno de 8h 9H

Présentation du site final

Du 12/6 au 16/6

Terminer le suivi web si nécessaire (intégration des derniers retours)

Mise en place de tests automatiques

Préparation pour le déploiement

Préparation des dernières livraisons

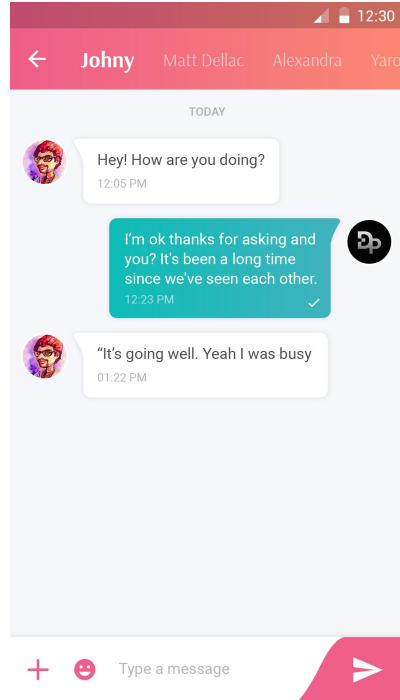
3

SERVEUR ?

4

Serveur

Chat

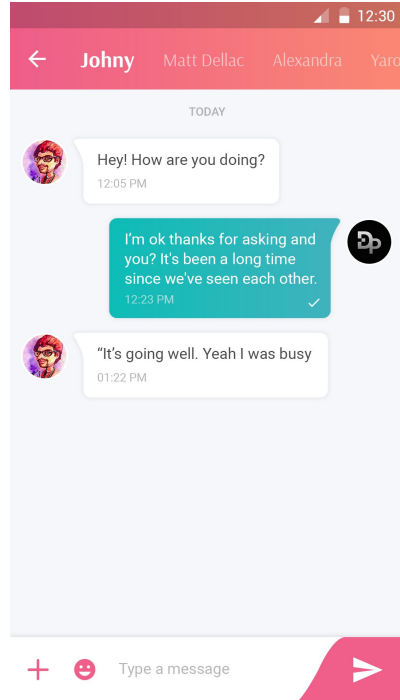


← **Messages ?**

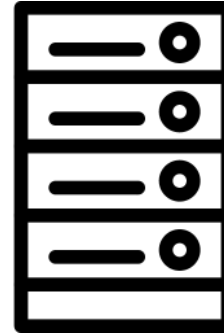
5

Serveur

Client



Server

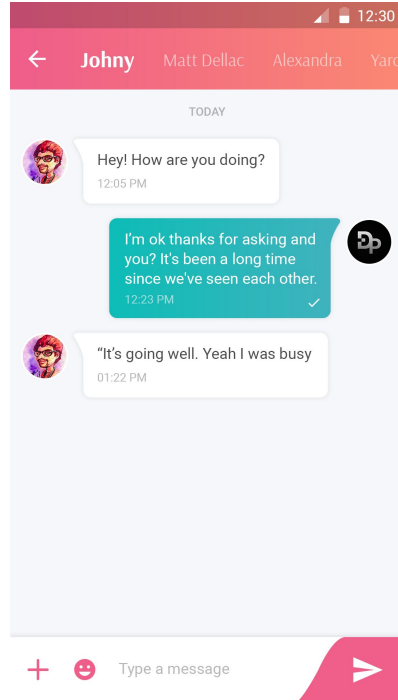


← Messages ?

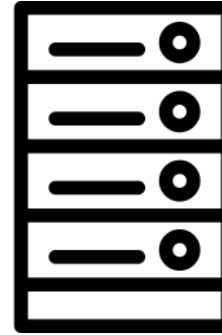
6

Serveur

Client



Server



Database

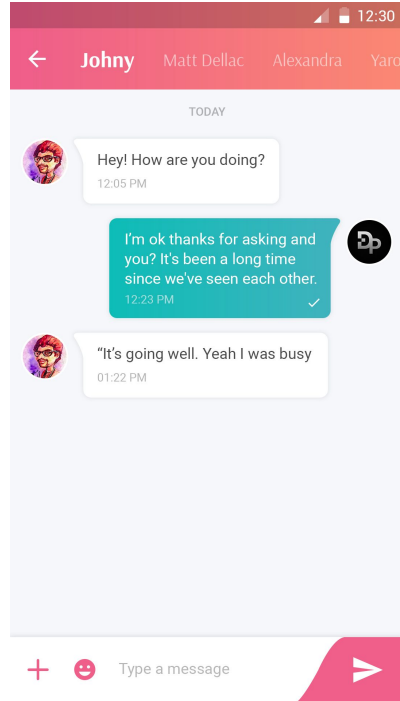


- _ Apply functional logic
- _ Transform the data according to the client needs
- _ Manage authentication
- _ Ensure data consistency
- ...

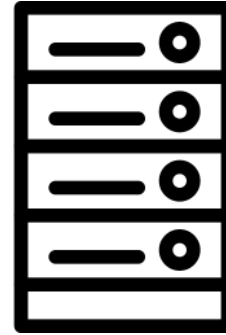
7

Serveur

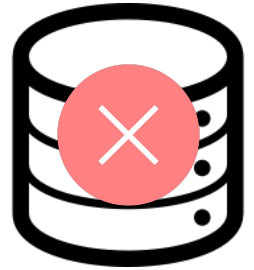
Client



Server



Database



8

Serveur


Client


MyApp

Create a new Quiz

Title

Create

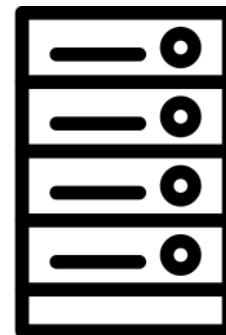
 Les Acteurs

 Les Sports

Select

Select

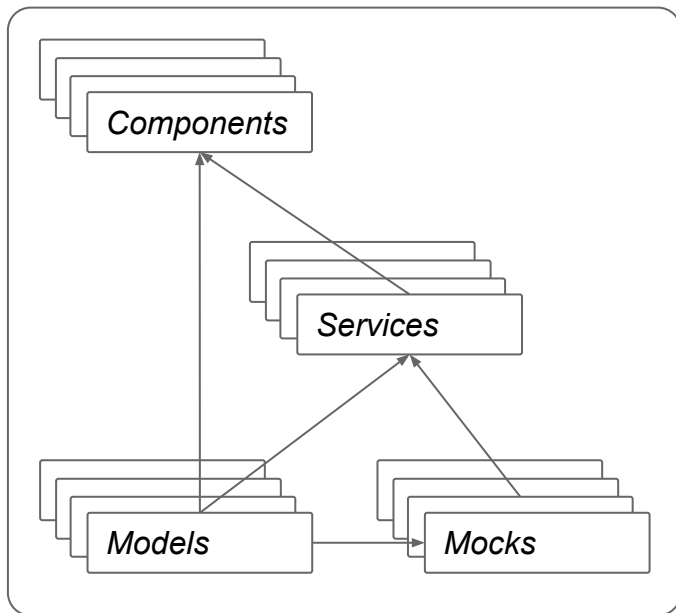
Server



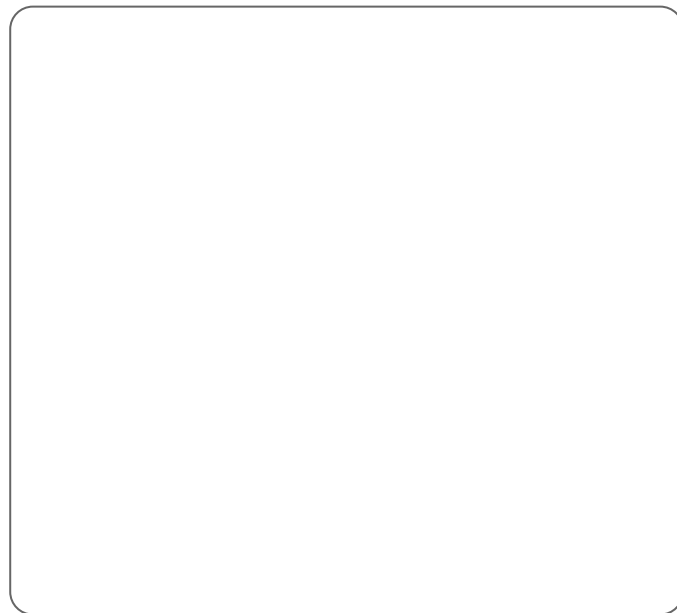
9

Serveur

Client



Server

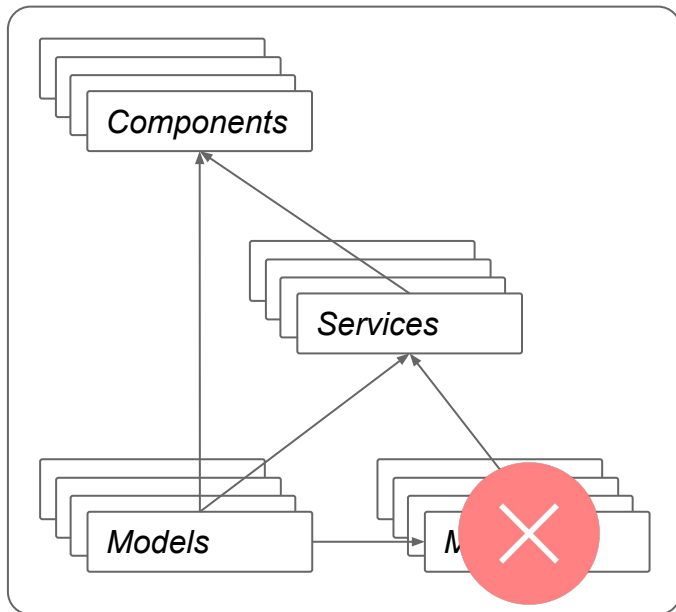


Ce que vous avez aujourd'hui

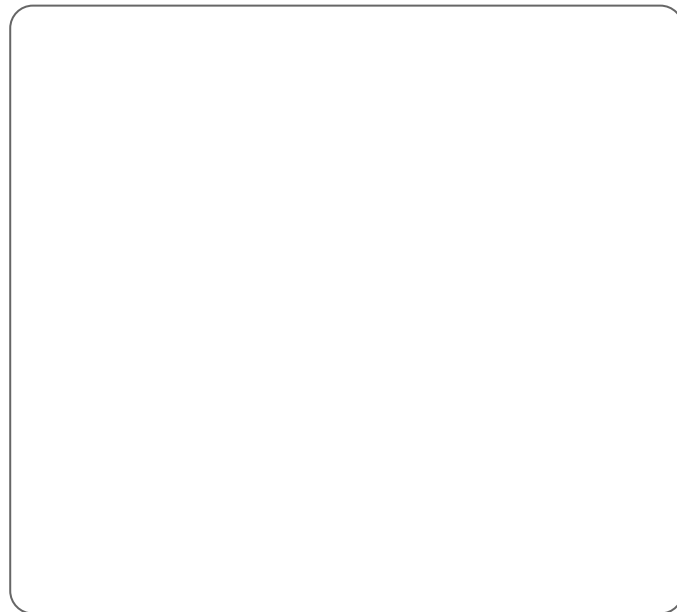
10

Serveur

Client



Server



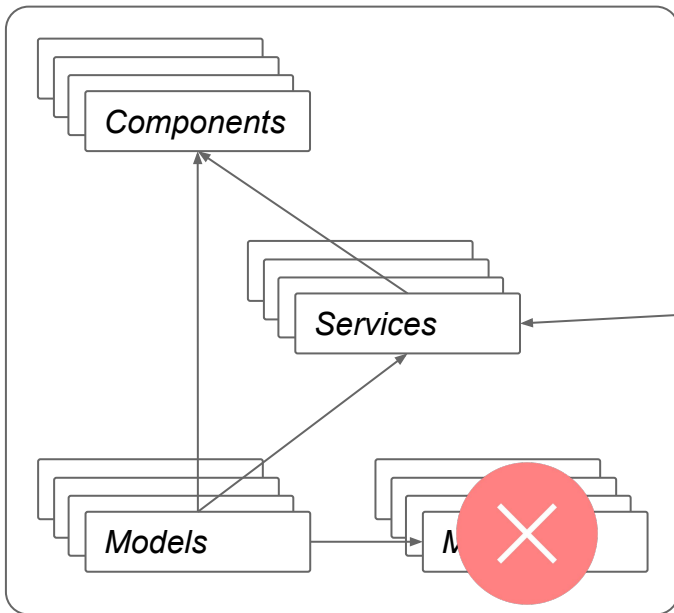
Ce qu'on veut avoir

11

Serveur

Client

Server



Nos besoins :

- Récupérer
- Ajouter
- Modifier
- Supprimer

Nos quizzes, users,
quizGames,
configurations etc...

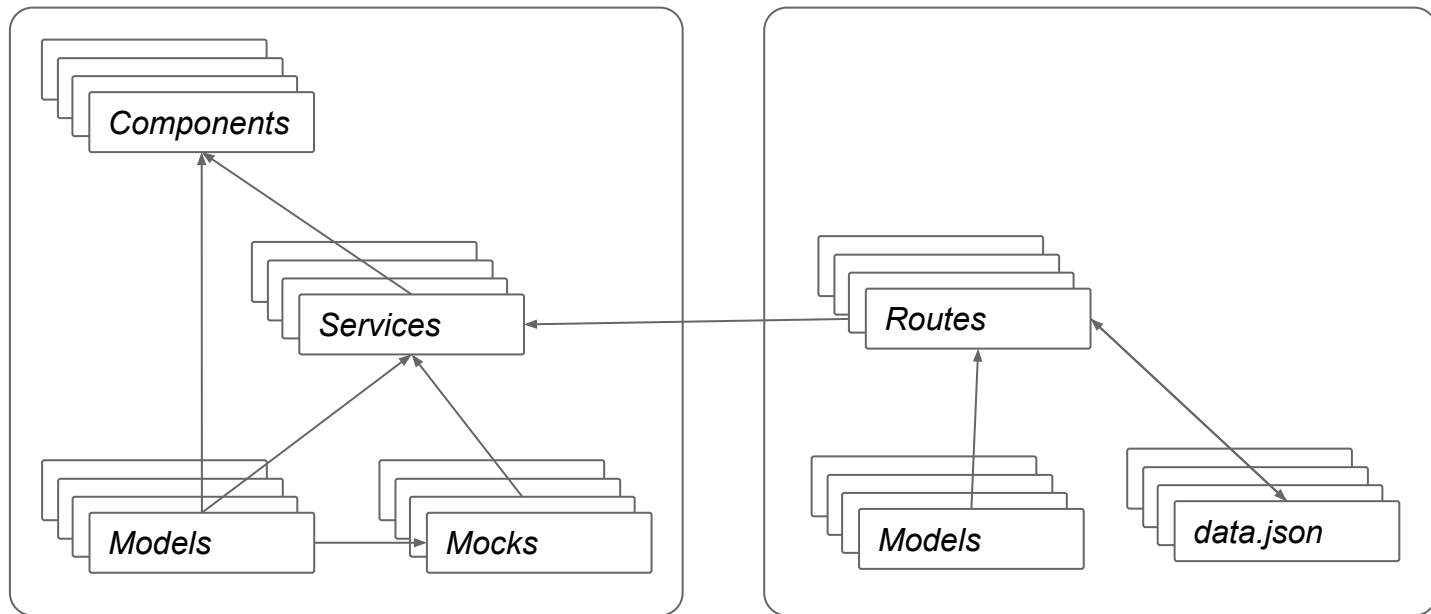
Ce qu'on veut avoir

12

Serveur

Client

Server



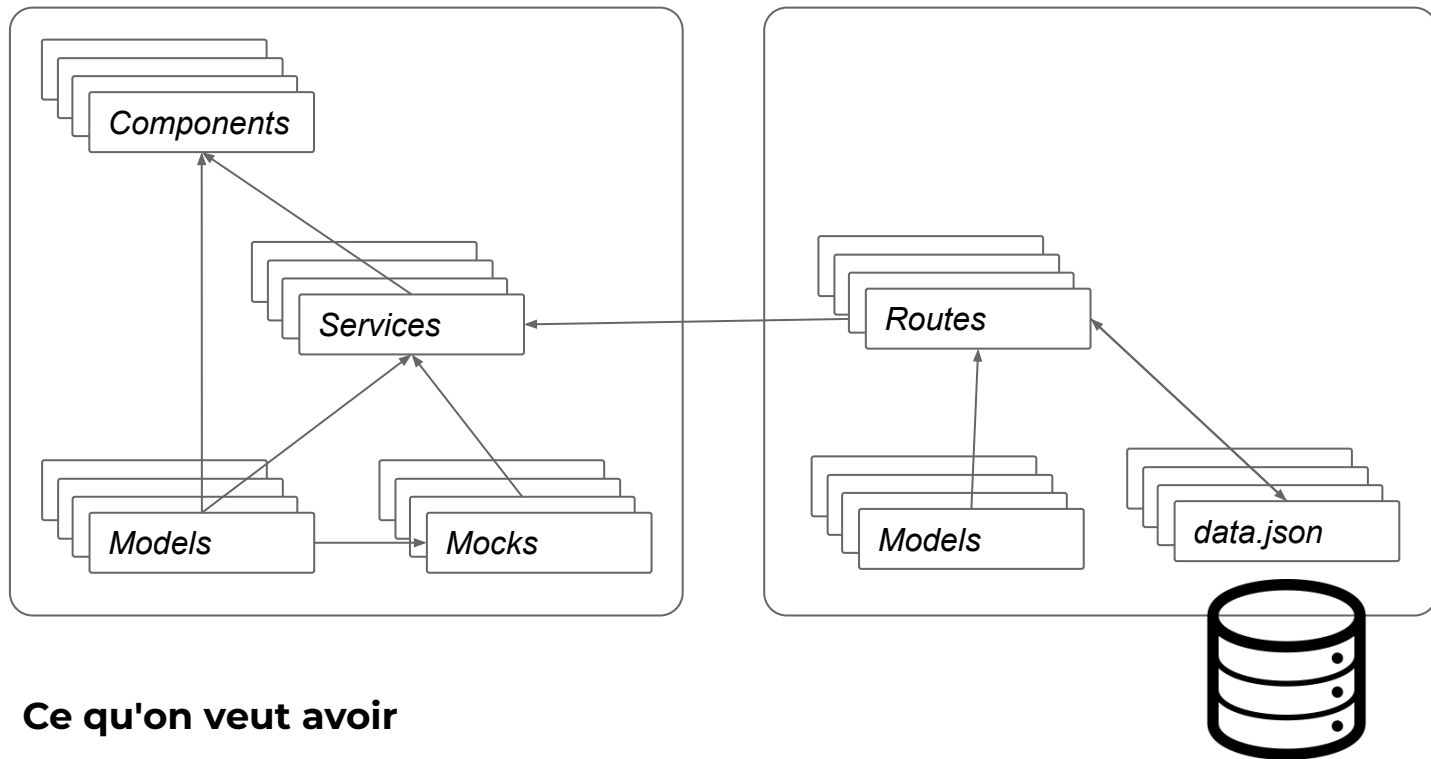
Ce qu'on veut avoir

13

Serveur

Client

Server



14

Définir un modèle

Comment définir un modèle

Exemple avec user.model.js

```
back-end > app > models > JS user.model.js > ...
```

```
1  const Joi = require('joi')
2  const BaseModel = require('../utils/base-model.js')
3
4  module.exports = new BaseModel('User', {
5    |   firstName: Joi.string().required(),
6    |   lastName: Joi.string().required(),
7  |   })
  |
```



Javascript = pas de typage

15

Définir un modèle

Comment définir un modèle

Exemple avec user.model.js

```
back-end > app > models > JS user.model.js > ...
```

```
1  const Joi = require('joi')
2  const BaseModel = require('../utils/base-model.js')
3
4  module.exports = new BaseModel('User', {
5    firstName: Joi.string().required(),
6    lastName: Joi.string().required(),
7  })
```

Attributs

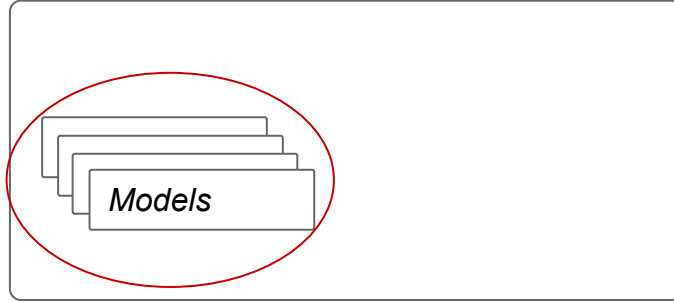
Description des
données

16

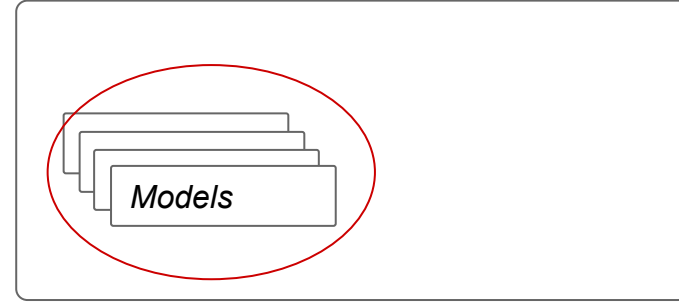
Serveur

model vs model ?

Client



Server



Même but :

- Établir un contrat entre les deux couches
- Définir une structure de données cohérente par rapport à vos besoins
- Réduire les risques de mauvaise communication ou de non-concordance des données

17

Définir un modèle

model vs model?

Exemple avec user.model.js

```
back-end > app > models > JS user.model.js > ...
```

```
1  const Joi = require('joi')
2  const BaseModel = require('../utils/base-model.js')
3
4  module.exports = new BaseModel('User', {
5    firstName: Joi.string().required(),
6    lastName: Joi.string().required(),
7  })
```

Attributs

Description des
données

18

Définir un modèle

Comment définir un modèle

Exemple avec user.model.js

```
back-end > app > models > JS user.model.js > ...
```

```
1  const Joi = require('joi')
2  const BaseModel = require('../utils/base-model.js')
3
4  module.exports = new BaseModel('User', {
5    firstName: Joi.string().required(),
6    lastName: Joi.string().required(),
7  })
```

Chaque model est un
BaseModel

19

Gestion des données

La classe BaseModel

Permet de

- Récupérer : `Model.get()` / `Model.getById(id)`
- Ajouter : `Model.create(payload)`
- Modifier : `Model.update(id, payload)`
- Supprimer : `Model.delete(id)`

Une ou plusieurs données pour notre model

Exemple avec User :

user.model.js

```
route - user/index.js
const { User } = require('../models')
const router = new Router()

router.get('/', (req, res) => {
  try {
    res.status(200).json(User.get())
  } catch (err) {
    manageAllErrors(res, err)
  }
})
```

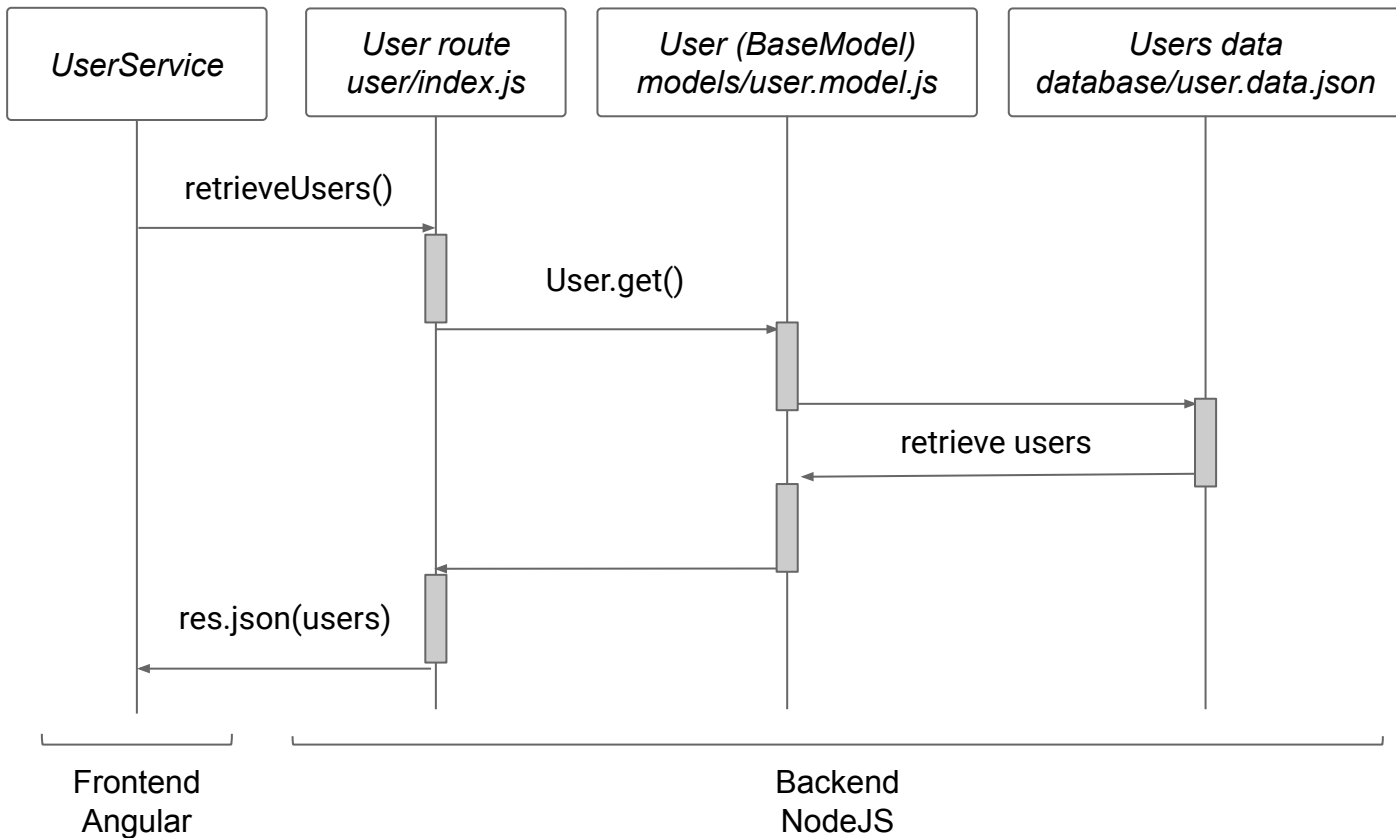
user.data.json

20

Gestion des données

Pour chaque model

Exemple avec user

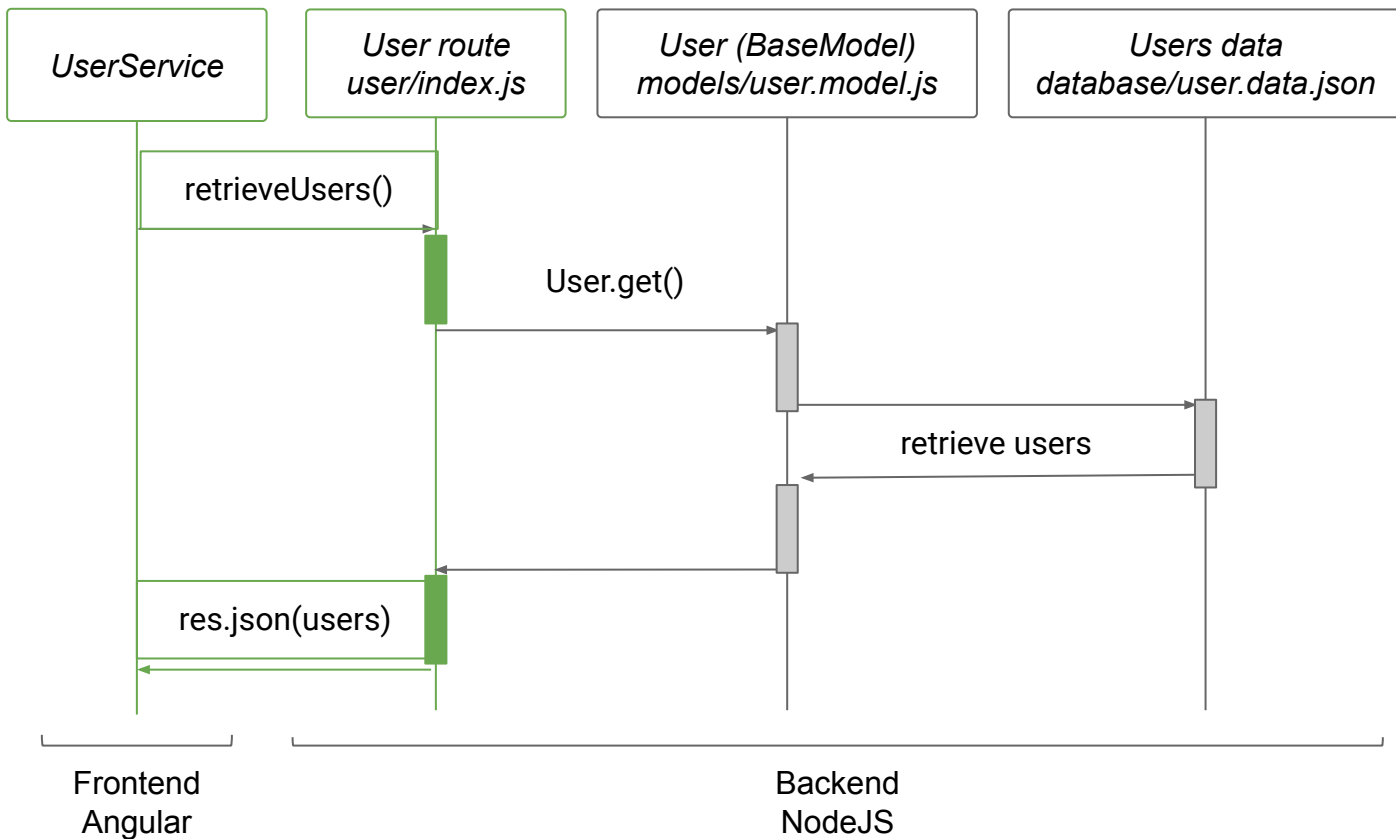


21

Gestion des données

Ce que vous devez faire (en vert)

Exemple avec user

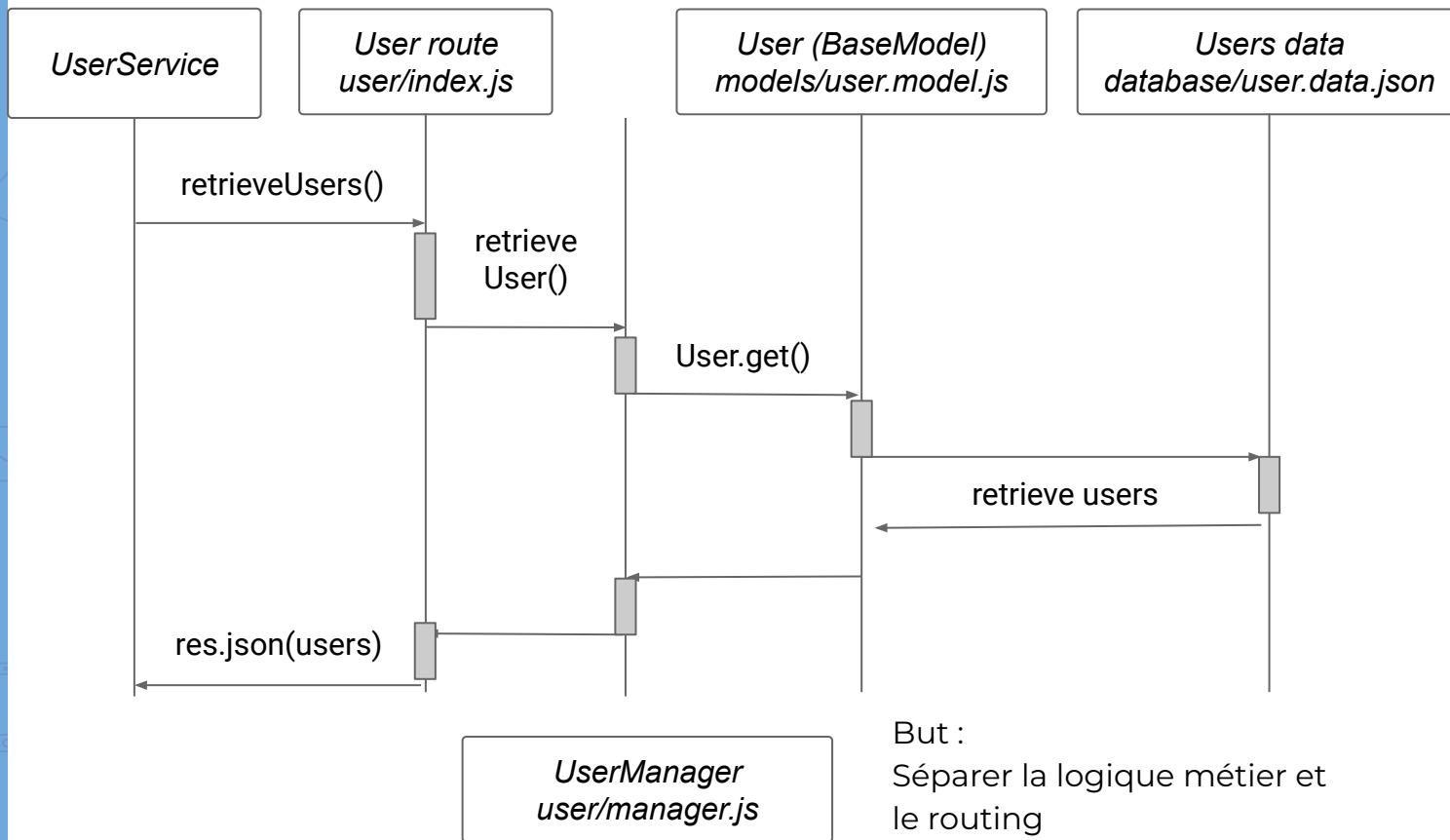


22

Gestion des données

Pour aller plus loin - Ajouter un manager

Exemple avec user



23

Définir un modèle

Comment intégrer côté Angular

Exemple avec UserService : [code](#)

```
export class UserService {  
  private users: User[] = [];  
  public users$: BehaviorSubject<User[]> = new BehaviorSubject([]);  
  private userUrl = serverUrl + '/users';  
  private httpOptions = httpOptionsBase;  
  
  constructor(private http: HttpClient) {  
    this.retrieveUsers();  
  }  
  
  retrieveUsers(): void {  
    this.http.get<User[]>(this.userUrl).subscribe((userList) => {  
      this.users = userList;  
      this.users$.next(this.users);  
    });  
  }  
}
```

24+ API REST

1 URL = 1 Ressource
ex : /books /items/:id

4 Méthodes :
GET POST PUT DELETE

Codes de retour :
404 200 201 400

Lien utile : <http://www.restapitutorial.com/httpstatuscodes.html>

25

TEST, 1, 2

POST /threads

Créer un thread

PUT /threads/:id

Mettre à jour le thread qui a pour id :id

GET /threads/:id/messages

Récupérer la liste des messages du thread :id

GET /users

Récupérer la liste des utilisateurs

DELETE /users/:id/messages/:messageId

Supprimer le message :messageId de la liste des messages de l'utilisateur :id

26

Postman

(Quick Tour + Example)

27



QUESTIONS?