API(CRUD) - DEMO-SEQUELIZE

Dossier : C:\**WORK**\NODEJS\demo-seq

Table des matières

[1 Projet CRUD avec EJS layout et SEQUELIZE 3](#_Toc157272525)

[1.1 Initialiser le projet 3](#_Toc157272526)

[1.2 Fichiers du projet CRUD app.js / ejs 3](#_Toc157272527)

[1.3 MYSQL WORKBENCH : mysql3 4](#_Toc157272528)

[1.4 Fichier base de données user 6](#_Toc157272529)

[1.5 Fichier app.js 7](#_Toc157272530)

[1.6 Mise en place du layout ejs dans : views 9](#_Toc157272531)

[1.7 Mise en place du homepage : première PAGE du layout 11](#_Toc157272532)

[1.8 La 12](#_Toc157272533)

[1.9 La 12](#_Toc157272534)

[1.10 La 12](#_Toc157272535)

[1.11 La 12](#_Toc157272536)

[1.12 La 12](#_Toc157272537)

[1.13 La 12](#_Toc157272538)

[1.14 La 12](#_Toc157272539)

[1.15 La 12](#_Toc157272540)

[1.16 La 12](#_Toc157272541)

[1.17 La 12](#_Toc157272542)

[1.18 La 12](#_Toc157272543)

[1.19 La 12](#_Toc157272544)

[1.20 La 12](#_Toc157272545)

[1.21 La 12](#_Toc157272546)

[1.22 La 12](#_Toc157272547)

[1.23 La 12](#_Toc157272548)

[1.24 La 12](#_Toc157272549)

[1.25 La 12](#_Toc157272550)

# Projet CRUD avec EJS layout et SEQUELIZE

## Initialiser le projet

<https://www.youtube.com/watch?v=PAm_QcN6Ffs&t=6343s>

git : <https://github.com/RaddyTheBrand/Nodejs-User-Management-Express-EJS-MongoDB>

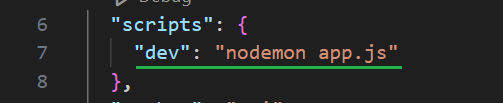
flash : <https://raddy.dev/blog/user-management-system-nodejs-express-mongodb-ejs-crud/>

**CREER** le dossier DEMO-SEQ

**CREER** le fichier app.js (par copier coller)

**INITIALISER** le projet : npm init entry : app.js (=> ça va créer le fichier : package.json)

**MODIFIER** le package.json (pour mettre en place le nodemon : npm run dev)



**INSTALLER** le projet : projet avec ejs layout et un accès base de données **SEQUELIZE** + joi + mysql2

npm install express dotenv ejs

npm install mysql2 sequelize joi

npm install express-session connect-flash express-fileupload cookie-parser

npm install express-ejs-layouts express-fileupload method-override

npm install nodemon --save-dev

-

**COPIER** les dossiers suivant :

* Dossier : public (on le garde tel quel)
* Dossier : views (on le garde comme principe et il doit obligatoirement se nommer : **views** pour ejs layout)

**CREER** le dossier: serveur, avec dedans les dossiers suivant :

* + Dossier : config (on garde le fichier .env)
  + Dossier : routes (on va y placer nos routes CRUD – même principe)
  + Dossier : controleurs (on va y placer les méthodes de chaque routes en get, post, …)
  + Dossier : models (on y place le fichier database.js (paramètres de connexions) & les fichiers Class sequilize)

**ARCHITECTURE** du projet NodeJS EJS views layout (résumé)

[app.js]------------[routes] ---------------- [controleurs] ---------------- [models] ---------------- [views]

Démarre le serveur CRUD Méthodes Class PAGES

listen

Et

Met en place

Le routage

## Fichiers du projet CRUD app.js / ejs

On commence par garder le fichier app.js tel quel, on change juste, pour que sa puisse compiler :

* Le nom des routes : user-routes
* Dans la config, vérifier : /serveur/config/.env (port & serveur nom)
* L’accès/démarrage de la base de données : **Db** (prendre un modèle ici sur sequelize mysql2)
* Dans user-routes.js mettre user-contoleurs & userControleur pour chaque route du CRUD
* Dans user-controleurs.js mettre chaque méthode du CRUD comme ci-dessous (méthode vide pour compiler) :
* Dans les routes on peut garder la structure du CRUD tel quel
  + soumettreAddFormulaire
  + addUser
  + homepage
  + getOne
  + soumettreUpdateFormulaire
  + updateOne
  + deleteOne
  + rechercheUsers
  + about()
* Dans le contrôleur, on créé chaque méthode à vide pour compiler
  + soumettreAddFormulaire ()
  + addUser()
  + homepage()
  + getOne()
  + soumettreUpdateFormulaire()
  + updateOne()
  + deleteOne()
  + rechercheUsers()
  + about()

* + Mettre à jour la class User – models/users.js
* Dans models/database.js mettre les paramètres de la base
* Dans models/user.js mettre à jour les champs de la table **user**

NE PAS COMPILER…

Il faut créer/mettre en place la base de données… [**crud**]

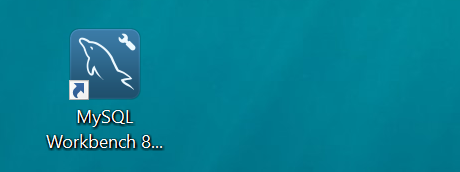
## MYSQL WORKBENCH : mysql3

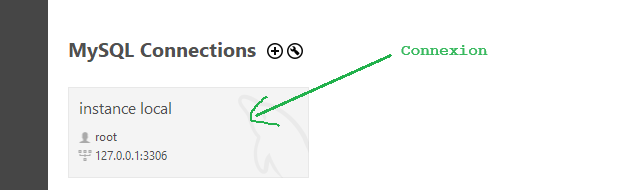
On commence par créer la base de données **crud**

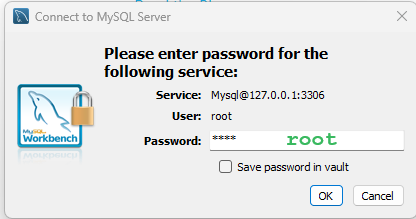
<https://www.youtube.com/watch?v=C_rjHMiGXlc&list=PL_fWONAepR_Bk2SyRkb_WtDxFarD51mr1&index=4>

vidéo à 3’20’’ ---- pour la connexion à WORKBENCH

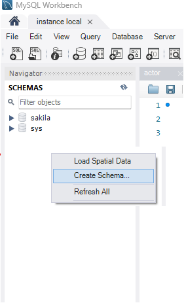
Ma documentation dans le fichier: API(MySQL-Workbench).docx



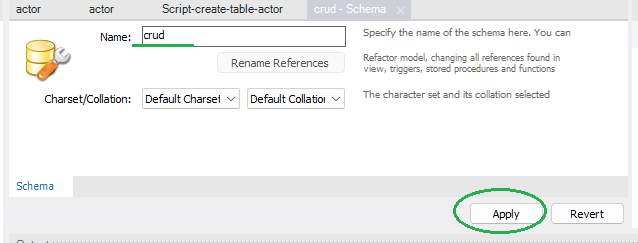


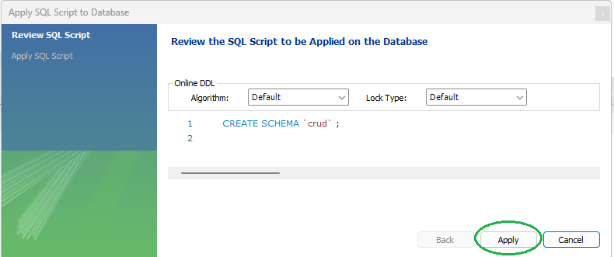


Faire un click droit + Create Schema…



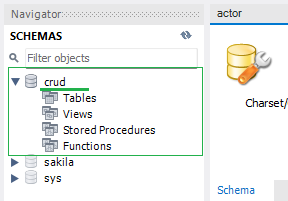
Nom : crud





Appliquer

Résultat : la base de données est créée :



Ne pas compiler il faut Mettre en place le modèle de données dans models/user.js

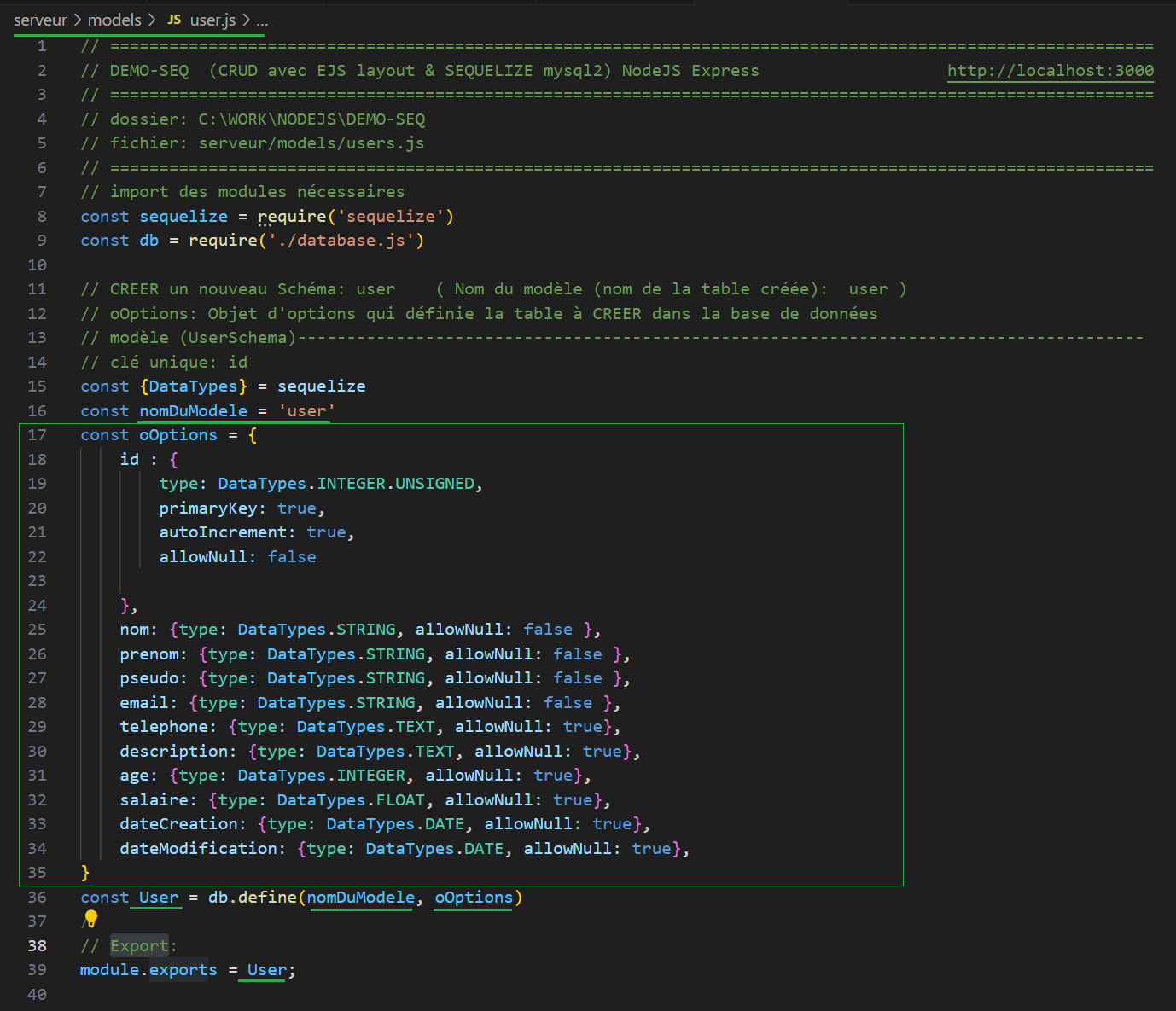
La mise en place de ce modèle de donnée user va également CREER la table **user** au premier lancement de l’application npm run dev

…

## Fichier base de données user

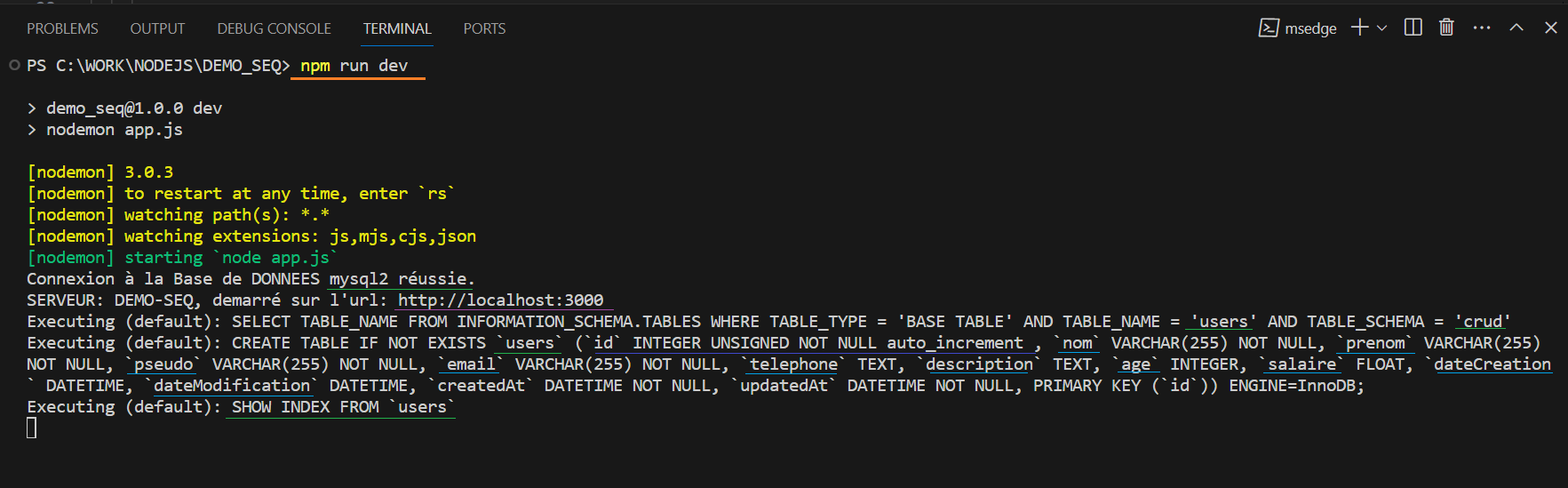
On va créer le Schéma du models mysql3 sequilize dans le fichier models/user.js

Dans notre cas **User** vat se comporter comme une **Class** **User**

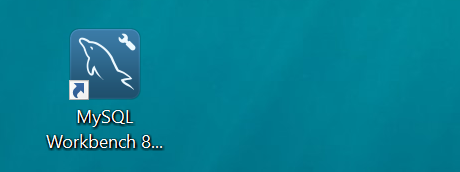


TEST/COMPILATION : le premier test va créer la table **user**

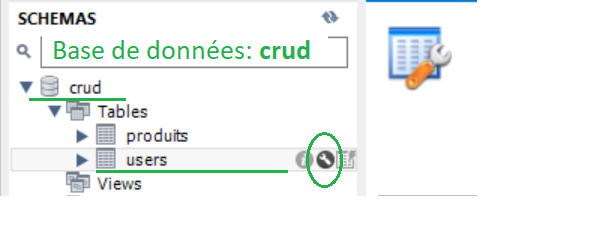
TEST : **npm run dev**



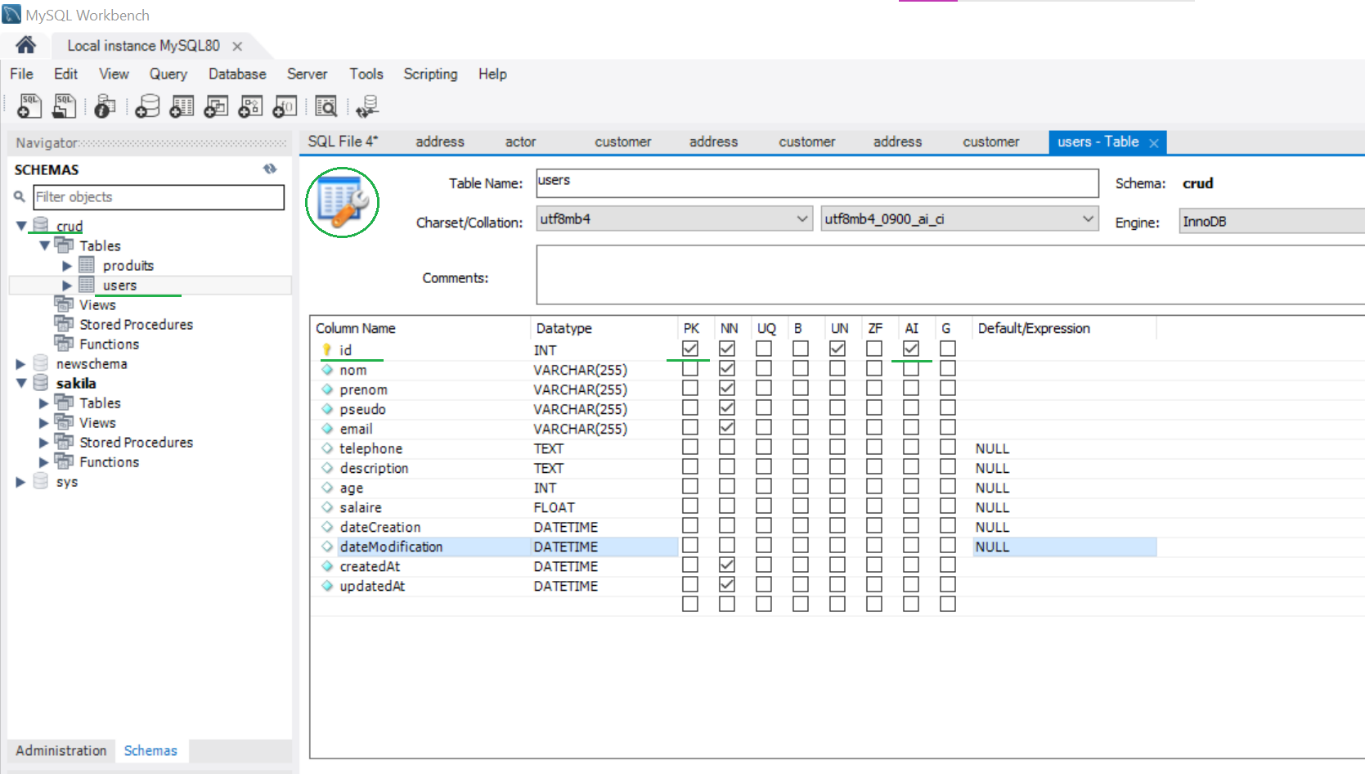
* La première compilation/exécution créé la table : **user**
* **Vérification** dans MYSQL WORKBENCH (mysql3)



…

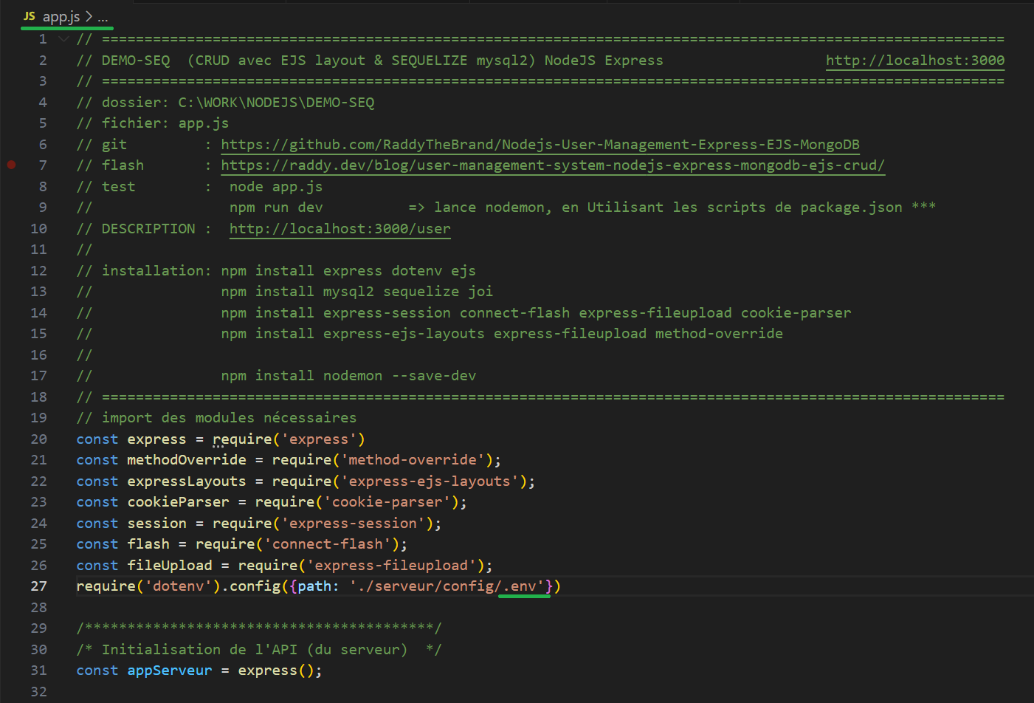


Cliquer sur la molette à droite de la table users

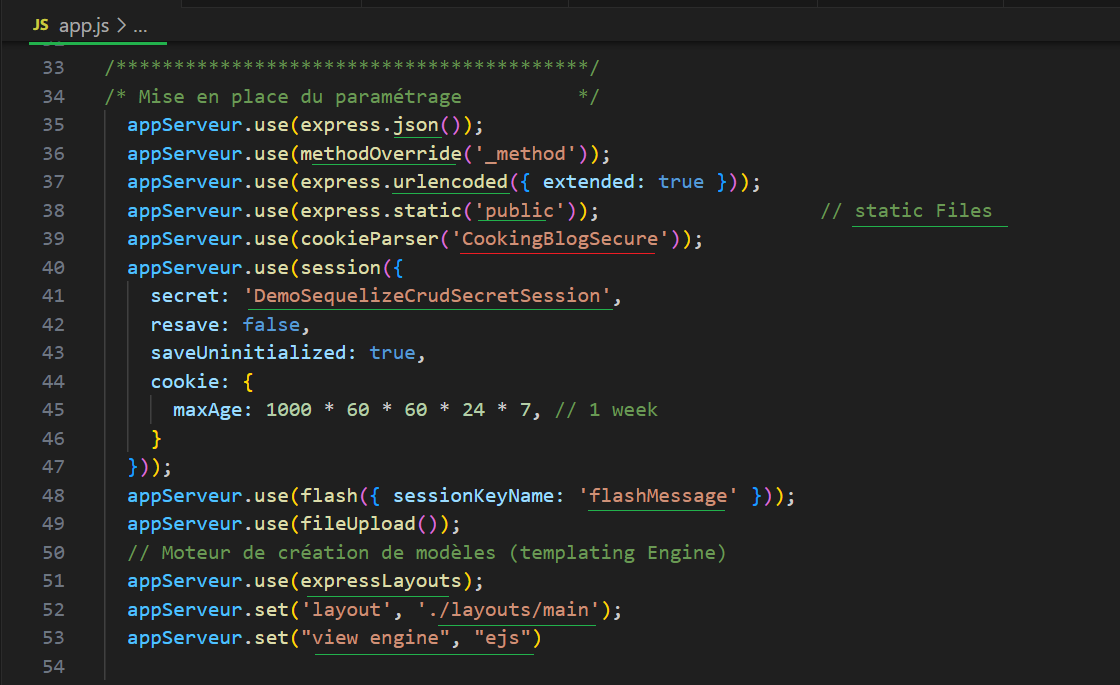


## Fichier app.js

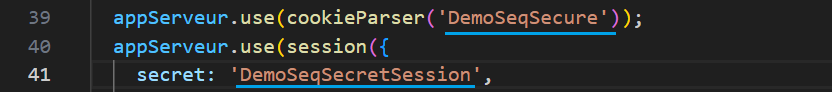
Fichier app.js ; Mise en place du serveur et définition des routes, ici userRoutes



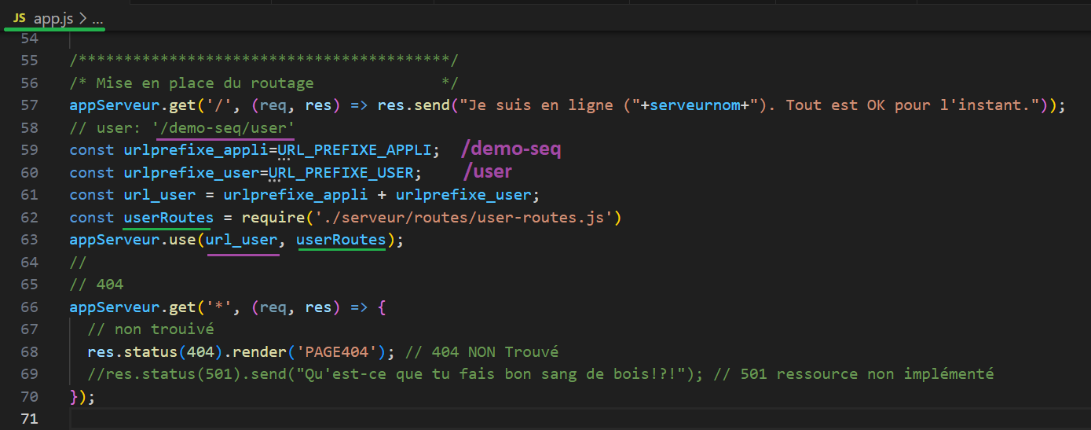
App.js suite, le nécessaire pour faire tourner l’application en mode : **ejs** **views layout**



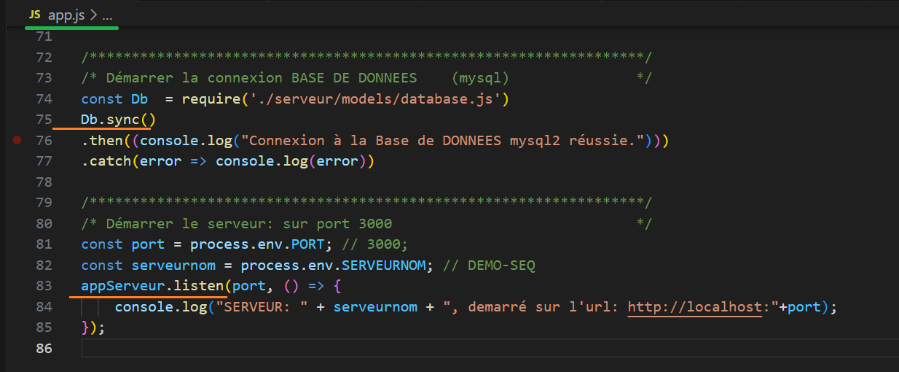
PENSER : à changer *CookingBlogSecure* en : DemoSeq**Secure** et secret : DemoSeq**SecretSession**



App.js suite, le nécessaire pour mettre en place le routage

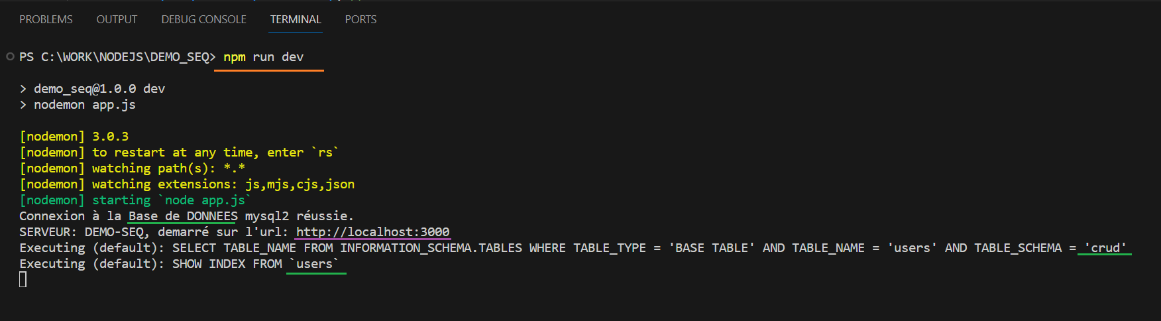


App.js suite, le nécessaire pour démarrer la base de données & le SERVEUR NODEJS

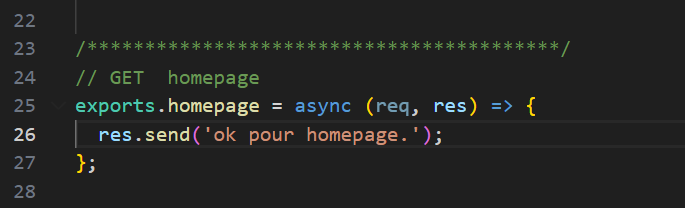


TEST/COMPILATION : le deuxième test va permettre de tester le paramétrage : **.env**

TEST : **npm run dev**



TEST routage url racines slash :

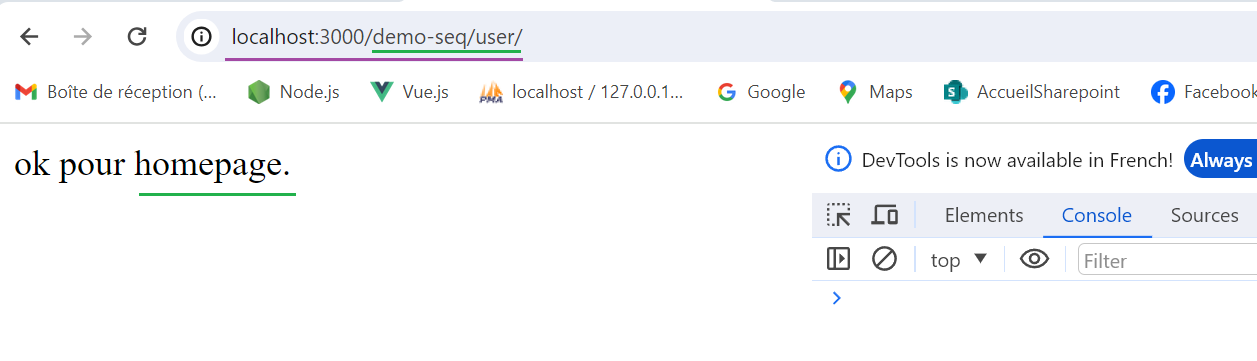


http://localhost:3000/

http://localhost:3000/demo-seq/user/



X

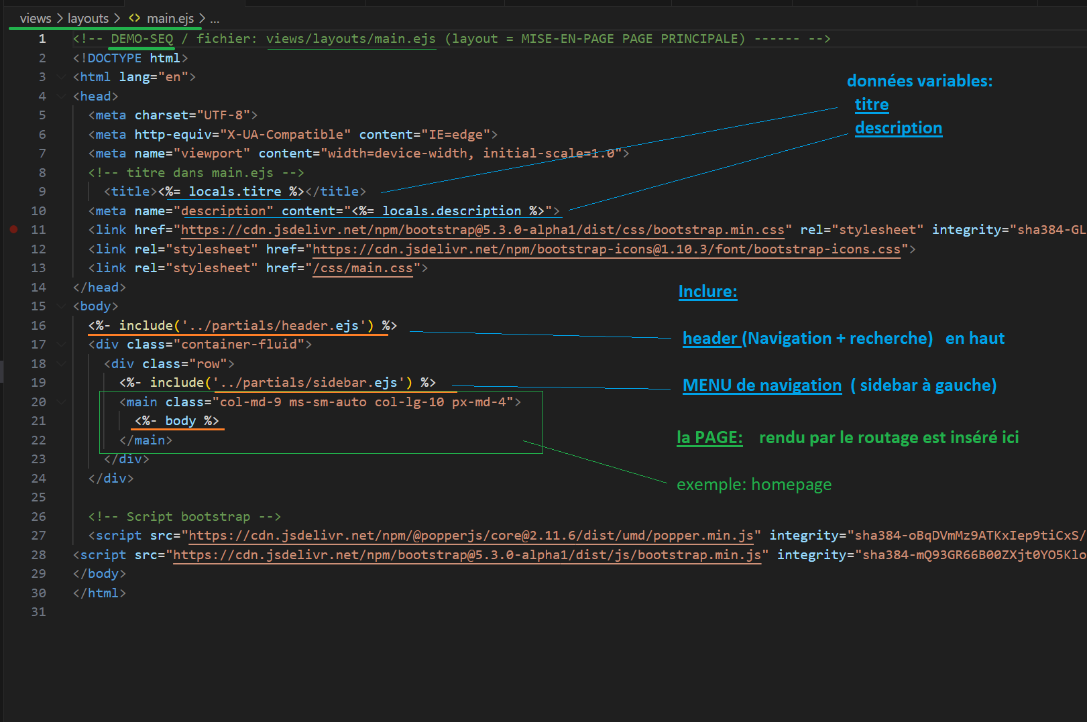


## Mise en place du layout ejs dans : views

La mise en place du layout ejs est constituée des fichiers suivants :

* Fichier main (qui inclue les deux partials) : views/layouts/main.ejs
* Partial entête (avec liens + recherche) : views/partials/header.ejs
* Partial navigation de gauche (MENU de gauche) : views/partials/sidebar.ejs
* Détail du fichier principal du **layout** (MISE-EN-PAGE): main.ejs

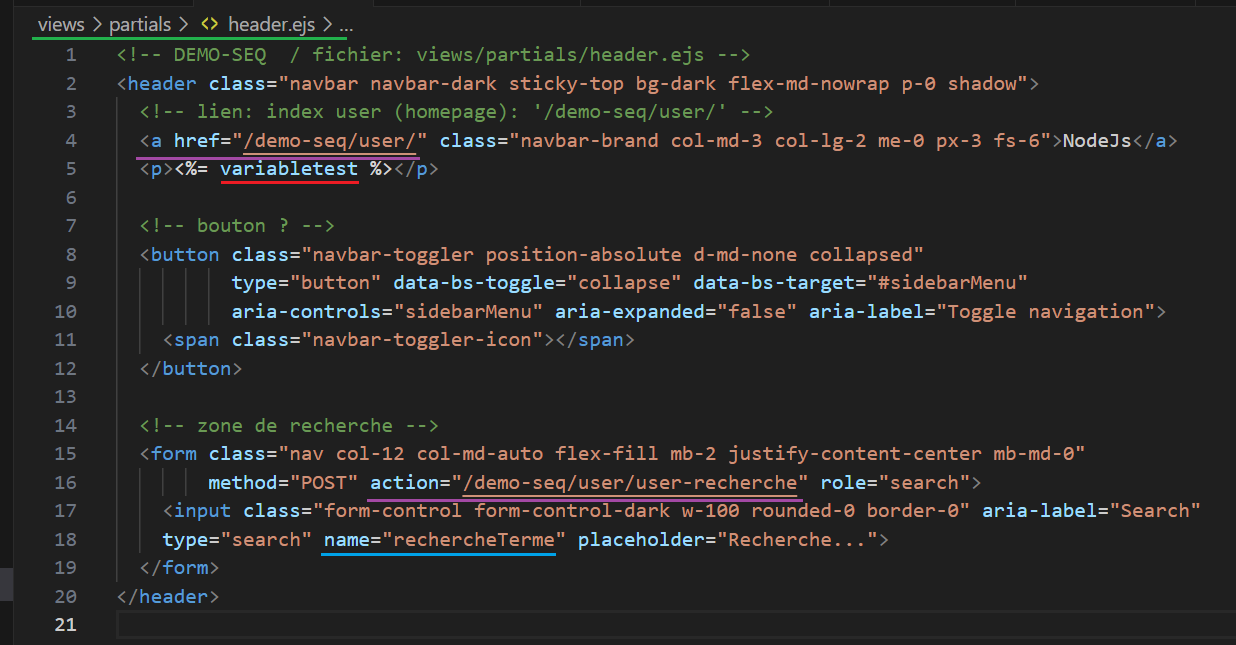
Pour ce main, c’est de la mise en place, rien à corriger dedans



* Détail du fichier partial (en-tête navigation/recherche) : header.ejs

C’est ici qu’on a des choses à **modifier** :

* + Lien vers index = homepage : ‘/demo-seq/user/’
  + Zone de recherche : ‘/demo-seq/user/user-recherche’ + rechercheTerme
  + La variable : variable test est à tester !

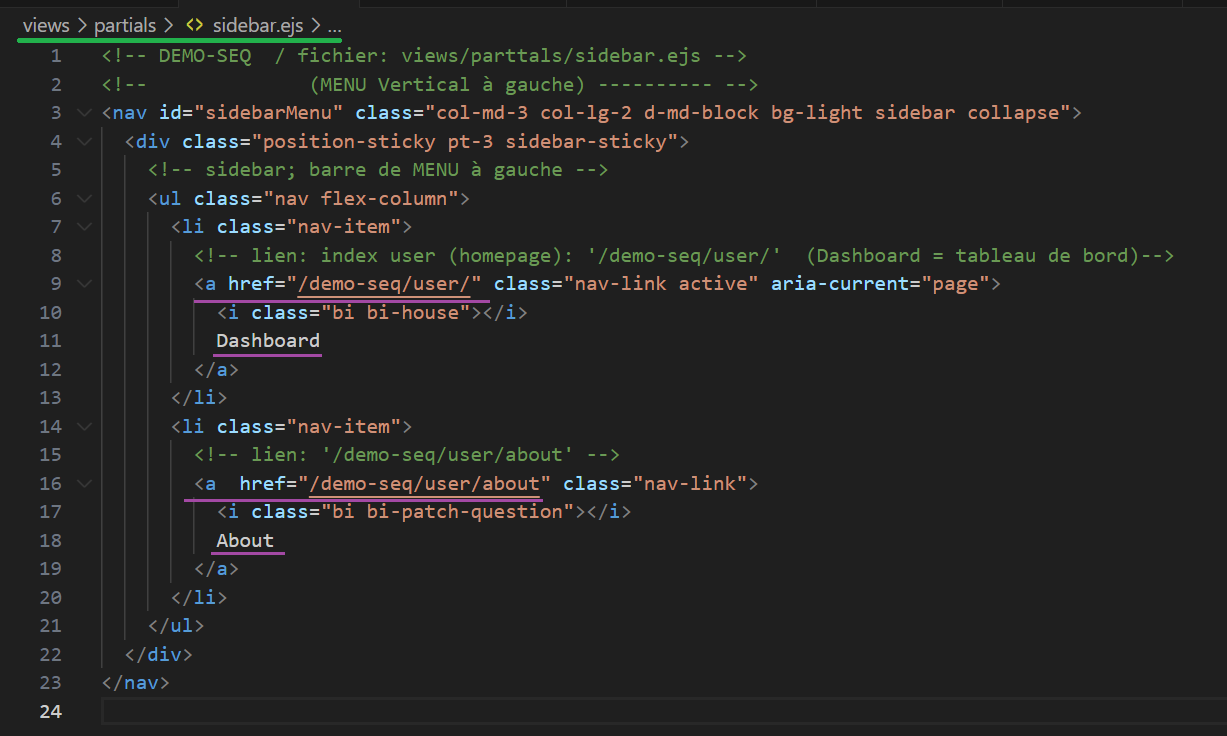


..

* Détail du fichier partial (MENU GAUCHE) : sidebar.ejs

C’est ici qu’on a des choses à **modifier** :

* + Lien vers index = homepage : ‘/demo-seq/user/’
  + Lien vers about ‘/demo-seq/user/about’
  + La variable : variable test est à tester !



## Mise en place du homepage : première PAGE du layout

C’est le lien ci-dessous qui doit rendre la PAGE du homepage c’est-à-dire index.ejs

http://localhost:3000/demo-seq/user/

SERVEUR + PORT : ‘http://localhost:3000’

RACINE de l’application : ‘/demo-seq’

RACINE de la table **user** : ‘/user’

Le slash index pour index.ejs = homepage : ‘**/**’

**ARCHITECTURE** du projet NodeJS EJS views layout (résumé)

[app.js]------------[routes] ---------------- [controleurs] ---------------- [models] ---------------- [views]

Démarre le serveur CRUD Méthodes Class PAGES

Listen main.ejs + header.ejs + sidebar.ejs

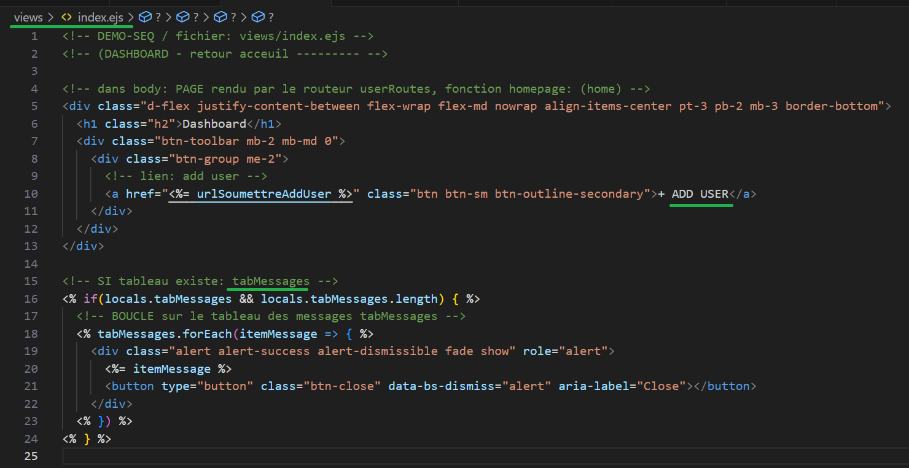
Et index.ejs

Met en place

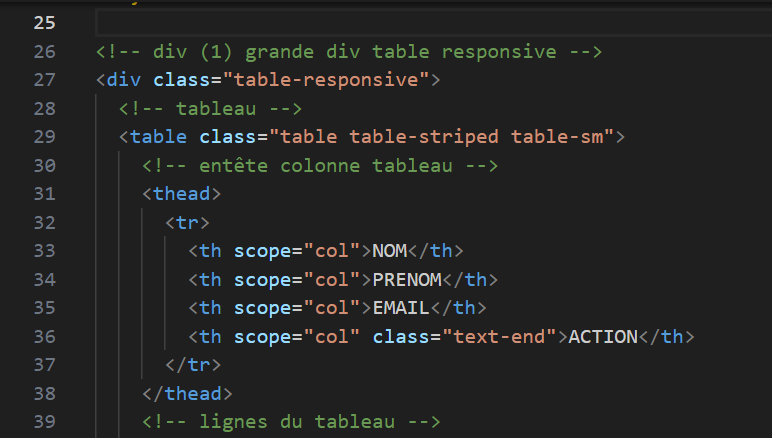
Le routage

Ficher de la PAGE homepage, index.ejs

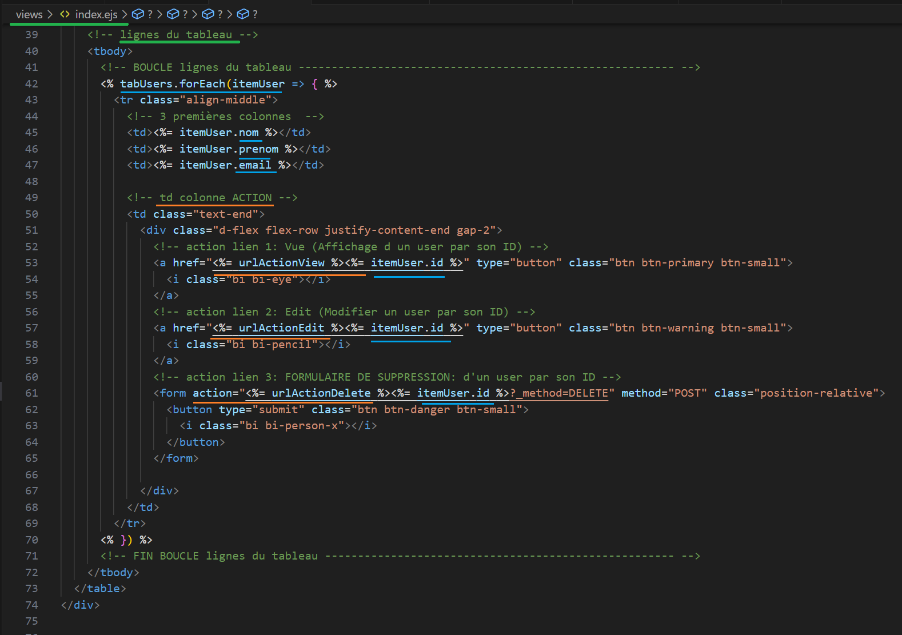
* Index.ejs : Ajouter user et tableau de message



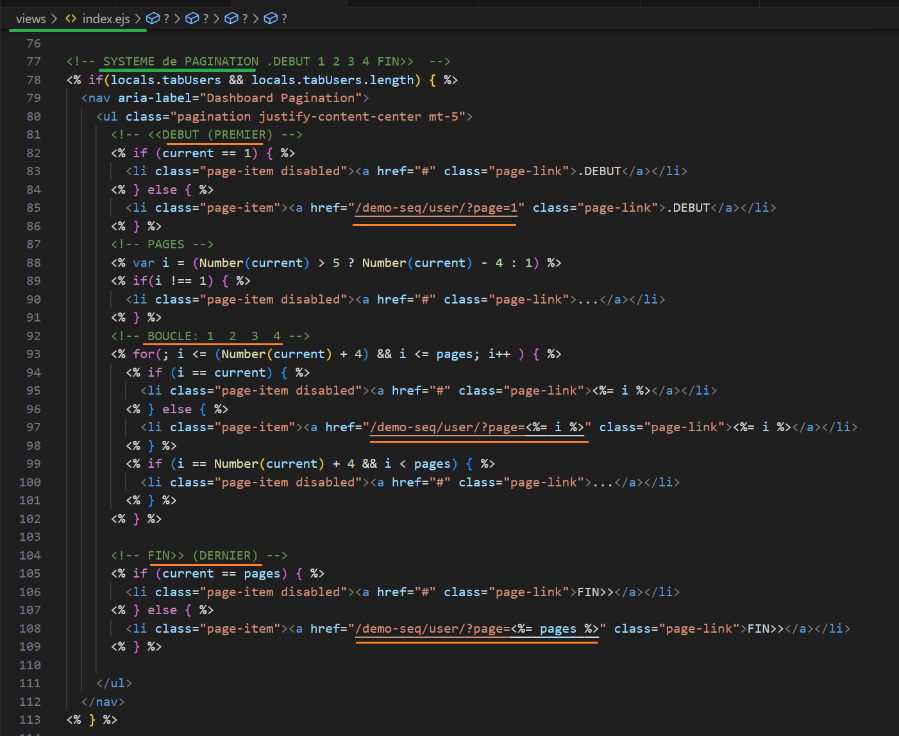
* Index.ejs suite, en-tête de colonne du tableau des users



* Index.ejs suite, lignes du tableau des users



* Index.ejs suite, système de pagination



\*

[app.js]------------[routes] ---------------- [controleurs] ---------------- [models] ---------------- [views]

Démarre le serveur CRUD Méthodes Class PAGES

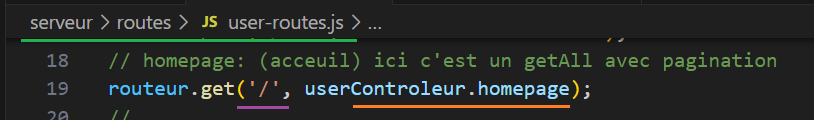
Listen main.ejs + header.ejs + sidebar.ejs

Et ‘**/**’ => homepage homepage() User index.ejs

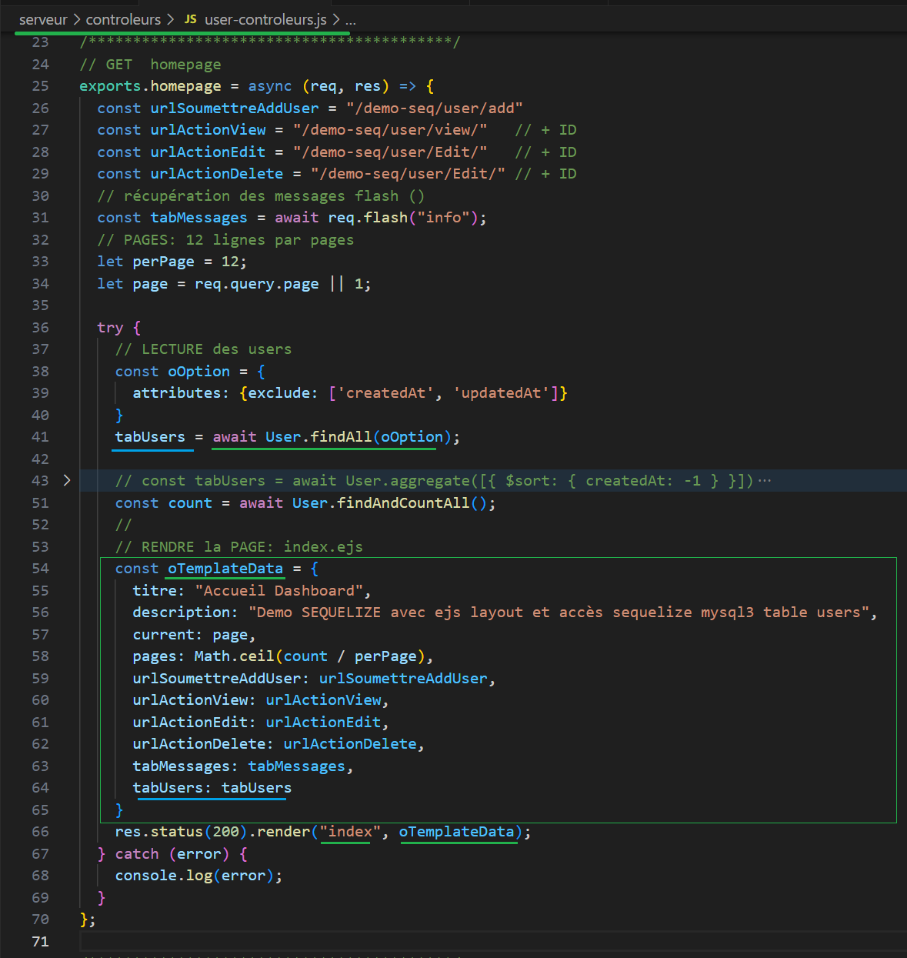
Met en place

Le routage

La route : ‘**/**’ avec les préfixes : ‘http://localhost:3000/demo-seq/user**/**’



Méthode du controleur user-controleurs : **homepage()**



Avec mysql sequelize aggregate ne fonctionne plus la pagination, comment faire ?

Ici iiiiiiiiiiiiiii en cours c’est pas simple pour la pagination

iciiiiiiiiiiiiiiiiiiiiiii

## La Homegape : doit être amélioré – Pagination !!!!!! \*\*\*\*\*\*

Debut de 12 en 12

Limit 12 lignes

Et gestion PAGES

Ici iiiiiiiii

Ici iiiiiiiii

…

## Paramétrage des url du header

Lorsqu’on affiche le Homepage, fichier index.ejs, grâce au layouts ejs (MISE-EN-PAGE ejs) ;

En fait le layouts ejs passe par main.ejs qui lui-même :

* Inclue : partials/header.ejs
* Inclue : partials/sidebar.ejs
* Gère via le routage dans <**%- body %**> : index.ejs

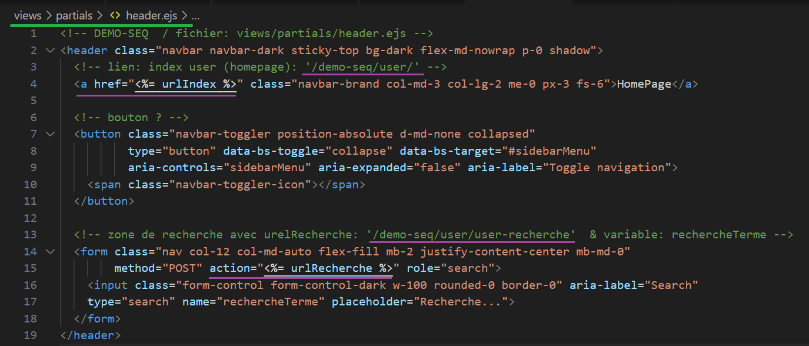
Tout ceci s’affiche donc en même temps au moment du res.render( index.ejs, {…} ) dans la méthode homepage()

C’est donc à cet endroit dans la méthode homepage() que l’on doit gérer les adresses url à mettre à la fois dans :

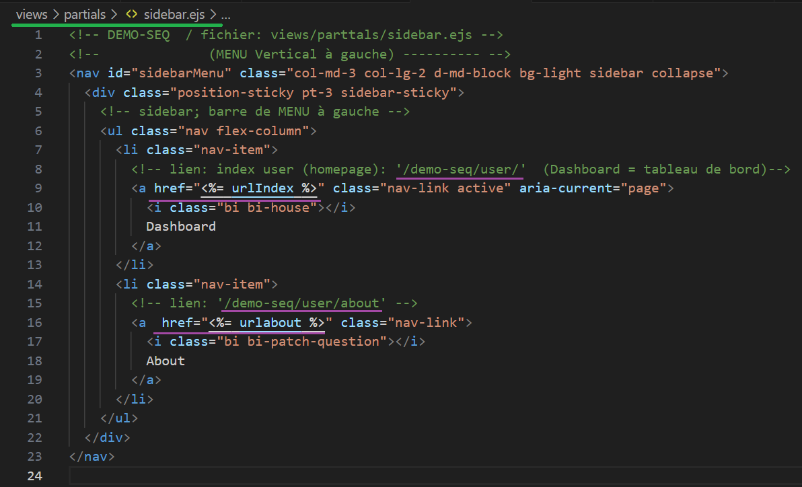
* Header.ejs
* Sidebar.ejs
* Index.ejs

**ATTENTION** : si on met en place des variables pour les url dans header.ejs c’est variable devront être géré partout où on fait un render d’une PAGE ejs même pour la PAGE 404

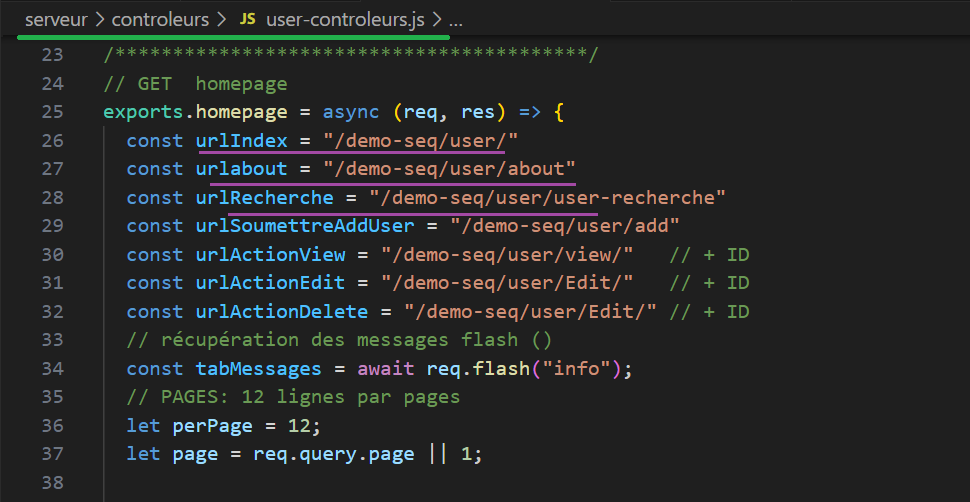
Exemple pour header.ejs

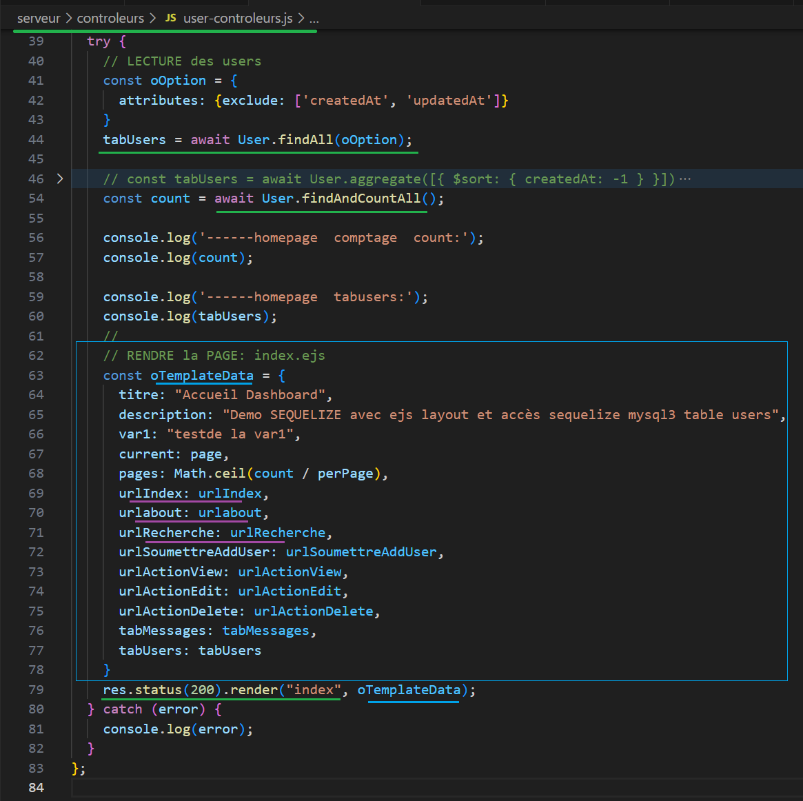


Exemple pour sidebar.ejs

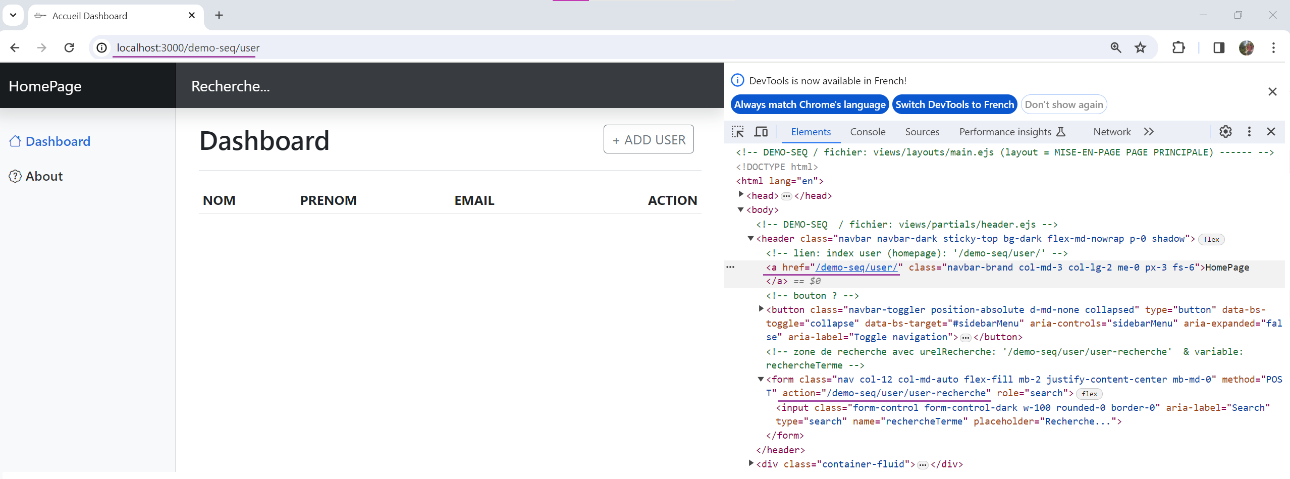


Pour les deux partials précédent { header.ejs, sidebar.ejs }, gestion des variables dans la méthode homepage()



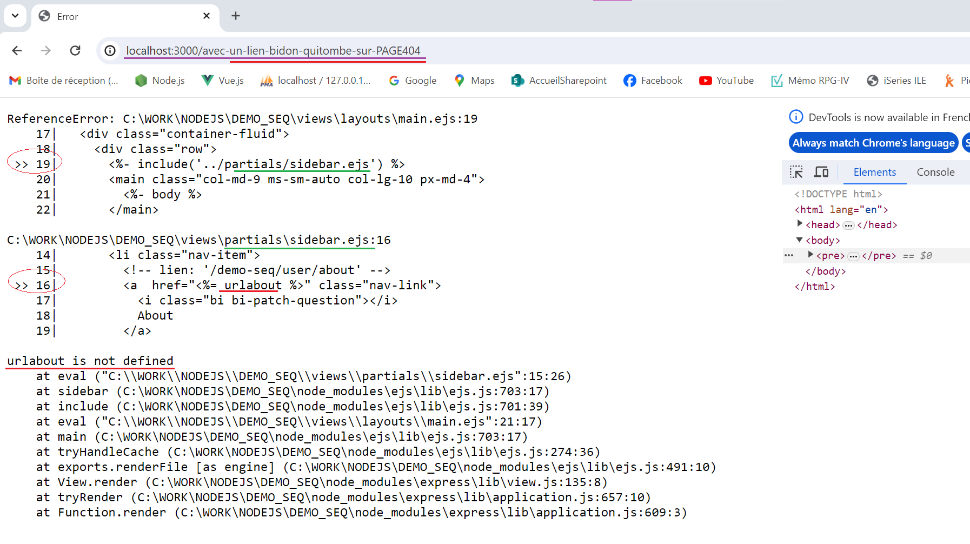


TESTER : **npm run dev** url : <http://localhost:3000/demo-seq/user>

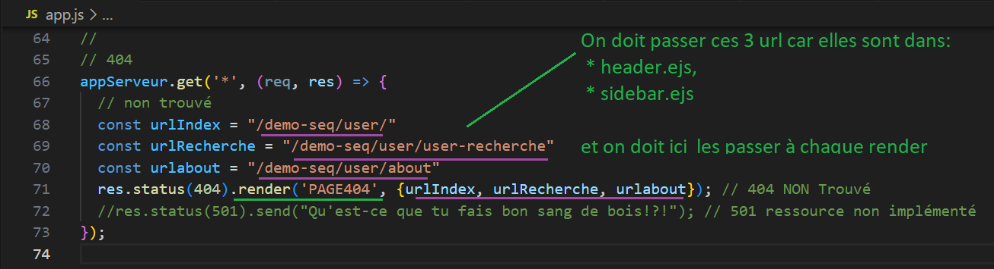


…

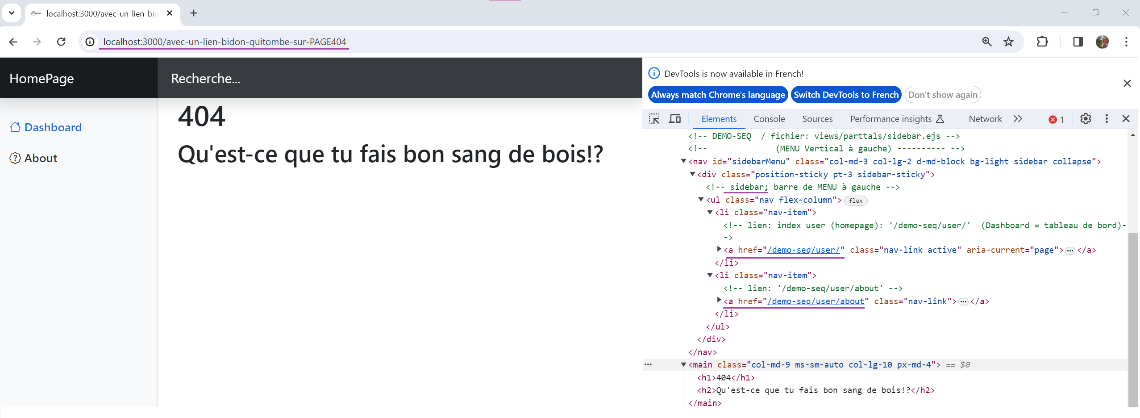
Tester avec erreur 404 url : <http://localhost:3000/avec-un-lien-bidon-quitombe-sur-PAGE404>



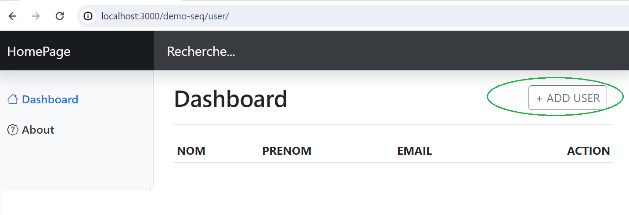
A cause du layouts ejs la PAGE 404 affiche aussi le headear & la sidebar, il faut donc aussi passer à chaque render même sur la PAGE404 les variables url définies dans le headear et dans le sidebar

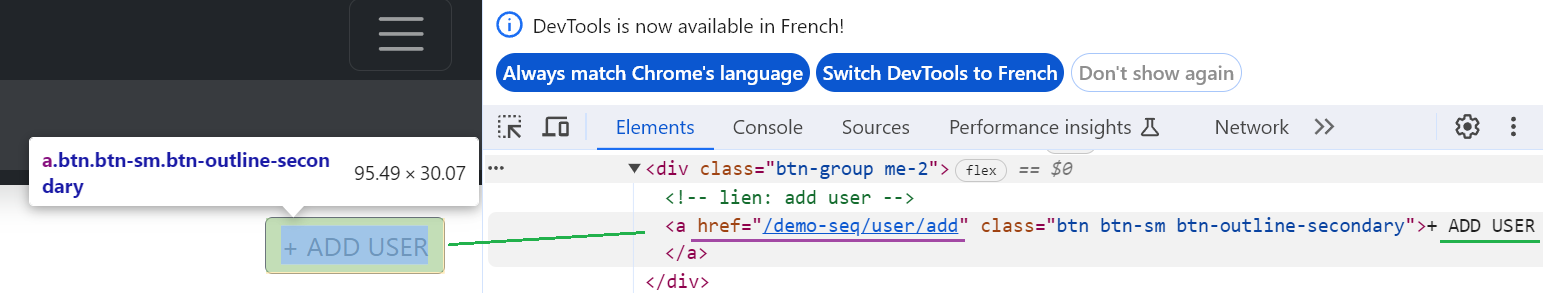


Tester avec erreur 404 url : <http://localhost:3000/avec-un-lien-bidon-quitombe-sur-PAGE404>

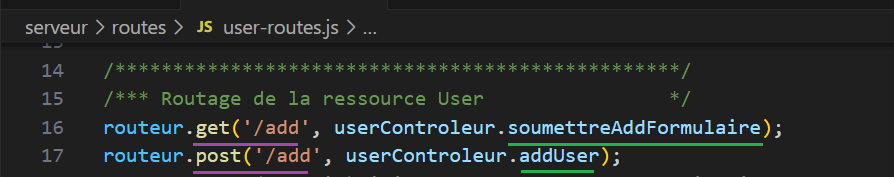


## Ajouter un User





Le routage :



Il fout deux routes, en ejs, pour ajouter un User :

* Une route **get** ‘/add’ pour afficher le formulaire de saisie du user
* Une route **post** ‘/add’ (posté) par le formulaire qui va ajouter le user en base de données

Le contrôleur, méthode : soumettreAddFormulaire

A écrire

Le contrôleur, méthode : addUser

A écrire

fv

## La

## La

## La

## La

## La

## La

## La

## La

## La

## La

## La

## La

## La

## La

## La

x

Vidéo 27’37’’ <https://www.youtube.com/watch?v=OEdPH4fV7vY>