**ENSET-M 2021-2022**

**COMPTE RENDU**

**TP : Spring MVC Thymeleaf**

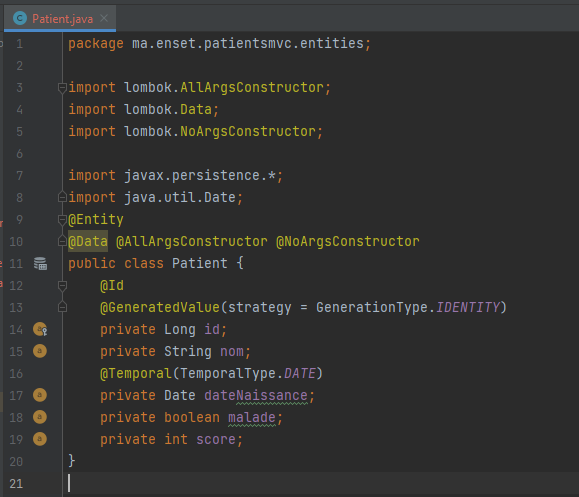
Par : **Assimi DIALLO** Encadrant : **M. YOUSSFI**

**Spring MVC Thymeleaf**

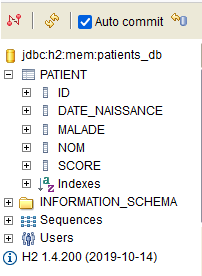
Dans ce TP nous allons faire la gestion des patients avec Spring MVC, le moteur de template Thymeleaf et Spring Data JPA.

L’application à réaliser doit afficher la liste de patients, permettre de rechercher un patient avec un mot-clé et supprimer un patient. Nous ajouterons aussi la pagination pour l’affichage de la liste des patients.

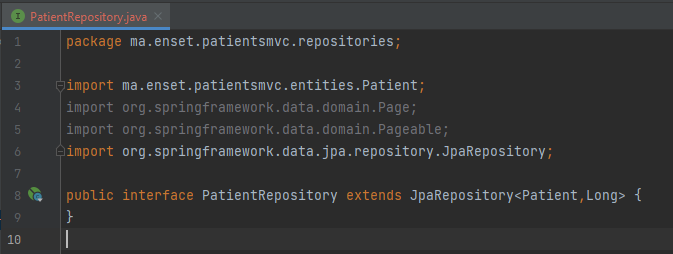
On crée donc l’entité **Patient** dans un package **entities** et grâce à Spring Data JPA nous aurons la génération de notre base de données H2.



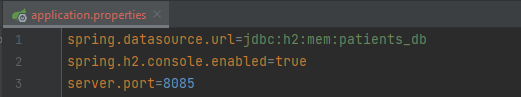
A l’exécution on obtient sur la base de données H2 :



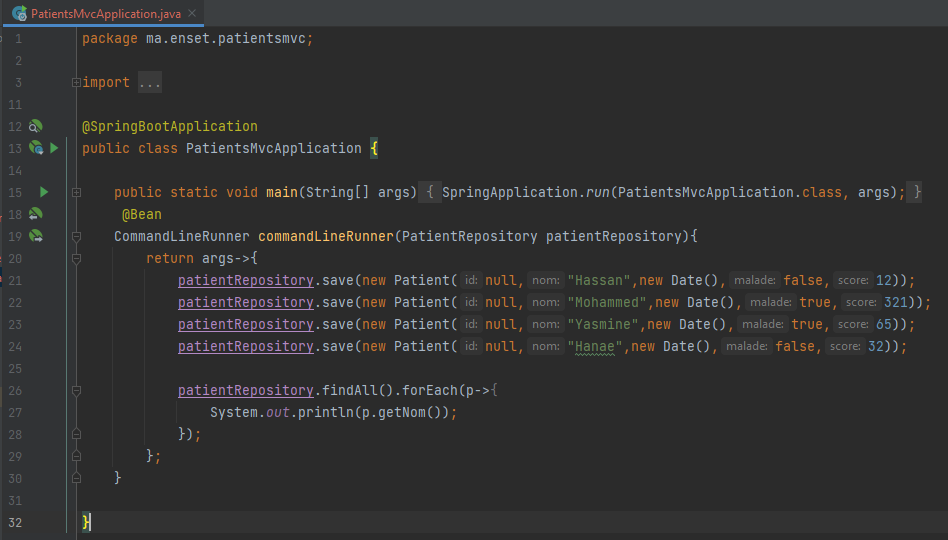
Par la suite on crée ue interface **PatientRepository** dans un package **repositories**. Cette interface hérite de **JpaRepository** et nous permettra donc d’interagir avec la base de données.

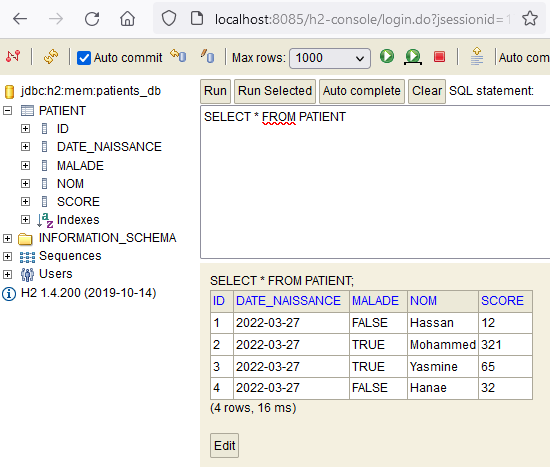


On a sur **applications.properties** les configurations suivantes :

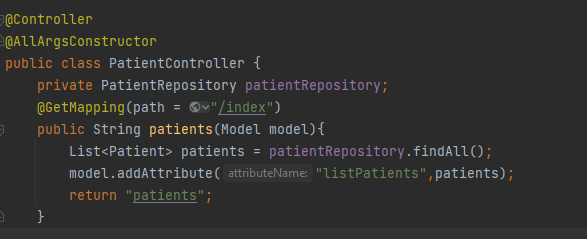


L’application va donc s’exécuter sur le port 8085.

Par la suite nous ajoutons les données dans la base de données :

Quand on exécute :

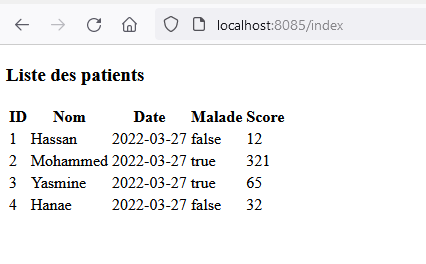
Puis nous déclarons un contrôleur



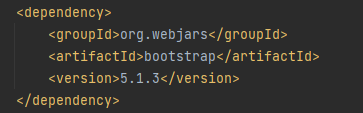
Le contrôleur va donc permettre la récupérer la liste de tous les patients et de la retourner dans la vue **patients** de **Thymeleaf.**

Puis on fait le traitement avec Thymeleaf en créant un fichier **patients.html** dans un dossier **template**s contenu dans le dossier **resources**. Dans **patients.html** on aura :

 L’affichage des patients se fera sur l’adresse : **localhost:8085/index**



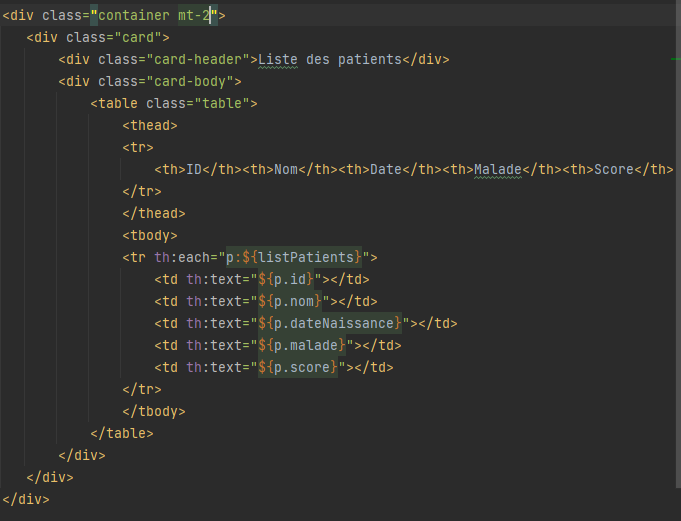
Pour améliorer l’affichage, on va ajouter bootstrap 5 :

Comme dépendance on aura :

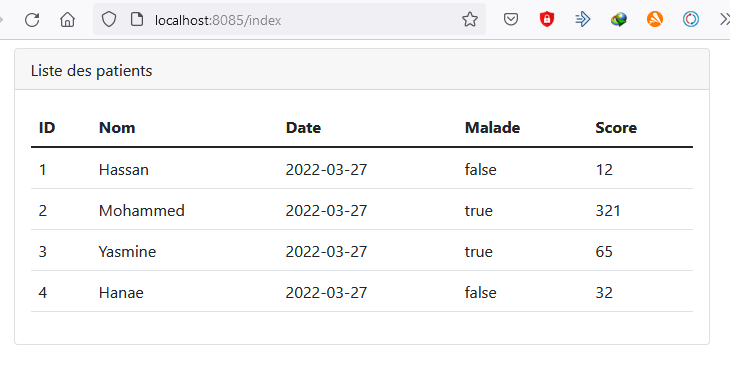
Puis on ajoute le lien sur le **patients.html** :



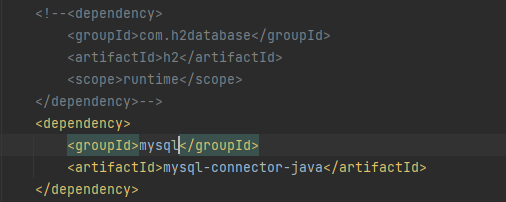
Par la suite on ajoute des classes bootstrap comme ci-dessous :

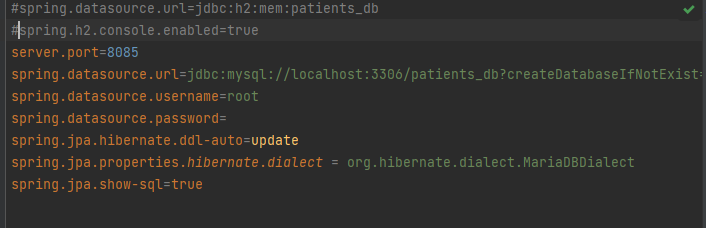


Puis quand on exécute on a :

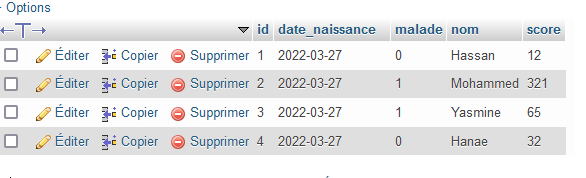


Pour pouvoir faire la pagination il nous faut plus de données. Nous allons donc passer sur une base de données MySQL de sorte qu’à chaque exécution de l’application des données soient ajoutées à la base de données :

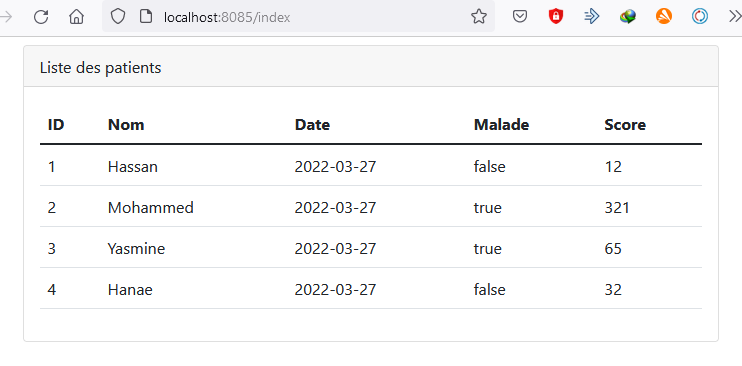




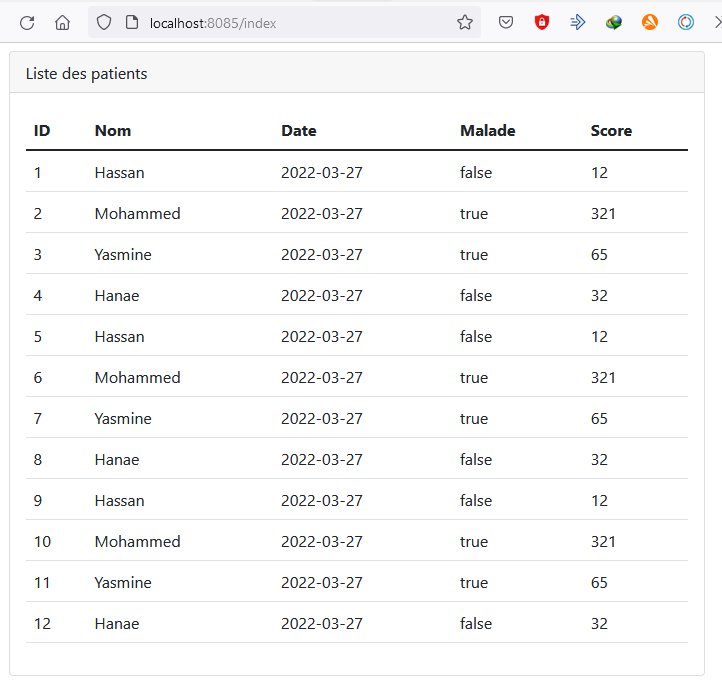
Quand on exécute l’application, on a l’ajout des données sur la base de données MySQL:



On remarque aussi que nous avons toujours les mêmes données sur l’adresse **localhost:8085/index**

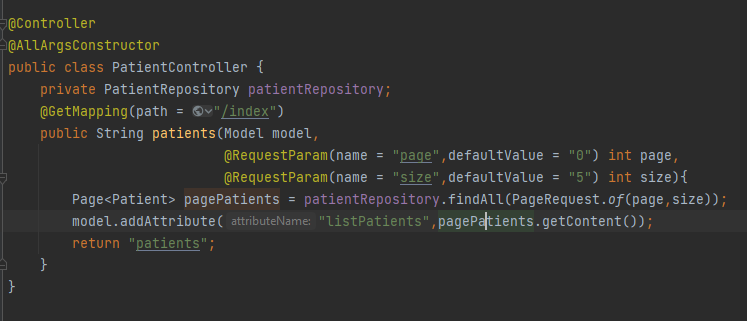


En exécutant deux (02) autres fois, on remarque que la table s’agrandit :

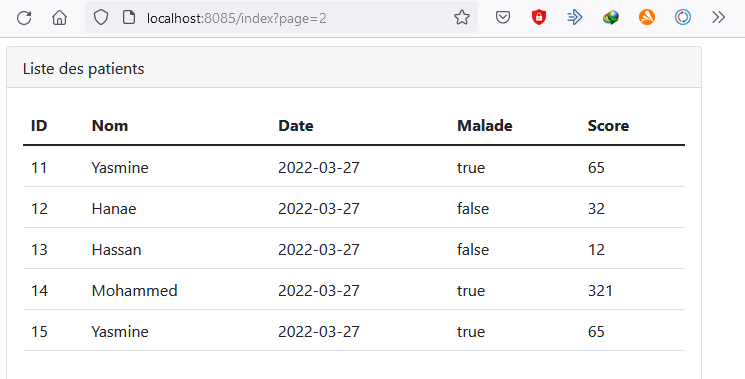


Maintenant que nous avons suffisamment de données, il devient intéressant de faire la pagination :

On aura donc dans le contrôleur :

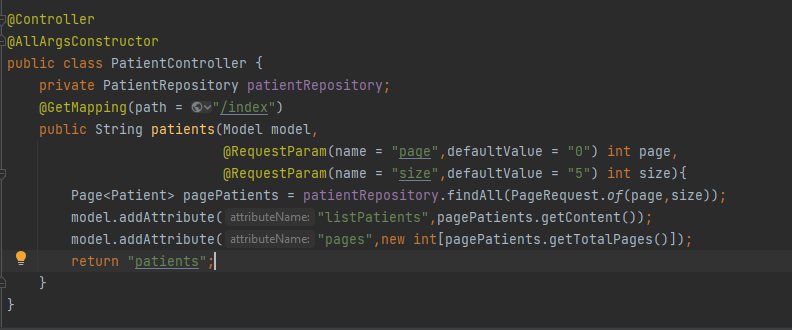


Quand on exécute en précisant **page=2** dans l’url on remarque que la table commence à **id =11**, donc on est bien sur la **page 2** étant donné que chaque page contient 5 patients :



Nous allons maintenant ajouter les numéros de pages juste après la table.

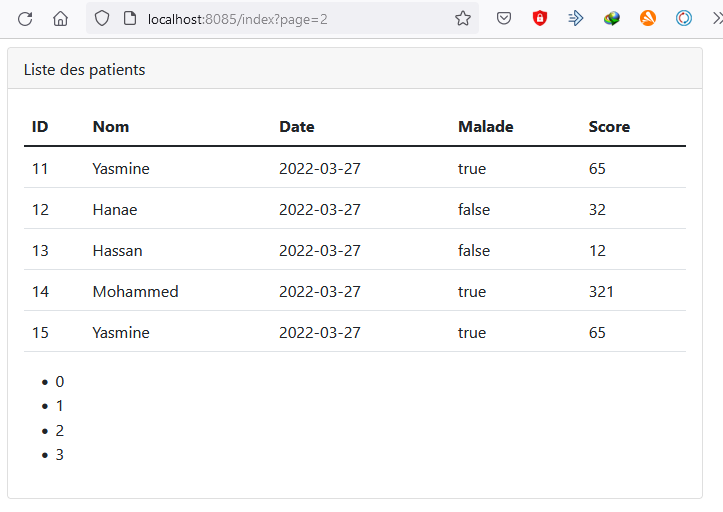
On aura dans le contrôleur :



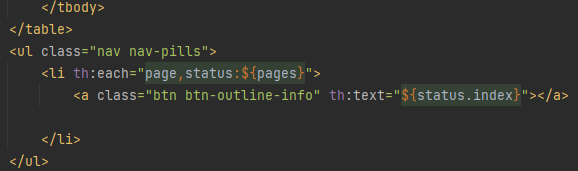
Et sur le fichier **patients.html :**



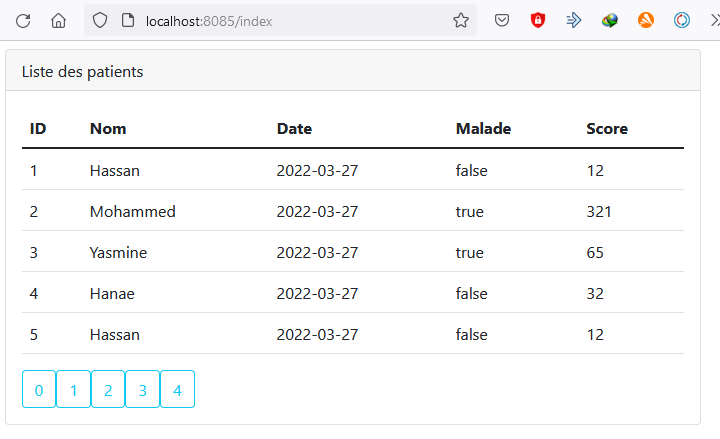
A l’exécution on a :



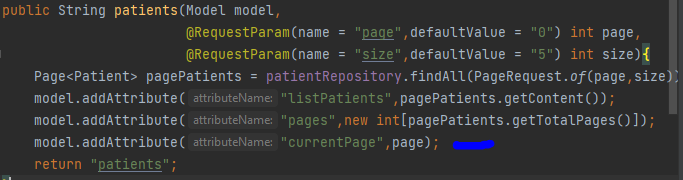
On arrange un peu plus l’affichage en faisant :



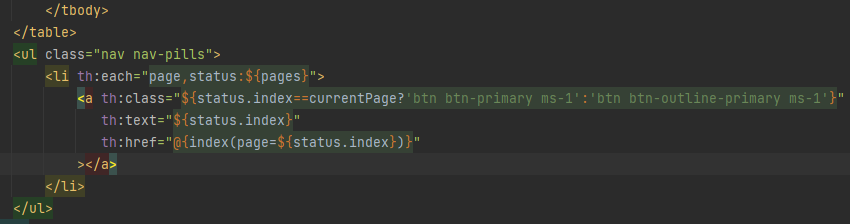
Et on aura :



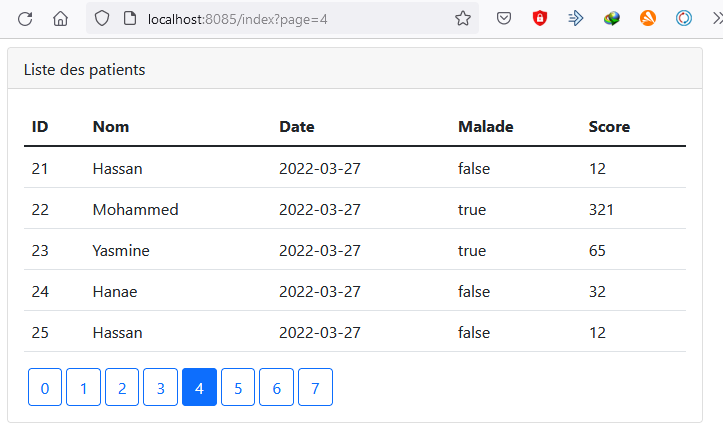
On voudra maintenant faire en sorte que quand l’utilisateur est sur une page donnée le numéro de cette page soit dans un carré bleu et ceci pour indiquer sur quelle page l’utilisateur est. On ajoute donc la page courante comme attribut du modèle comme suit :



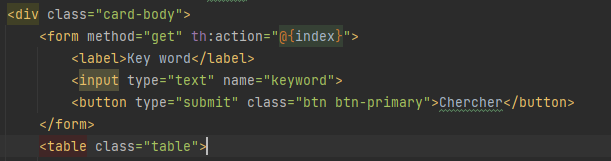
Sur le fichier patients.html on a :



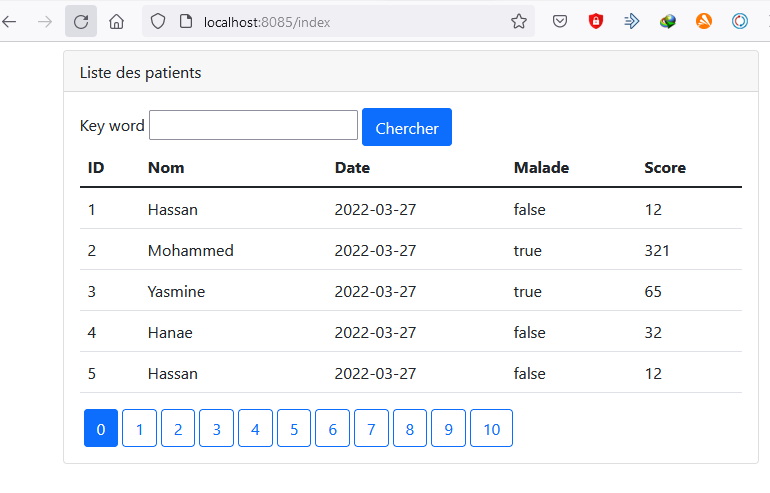
En exécutant on peut voir. Ici comme le montre l’url on est sur la page 4 et on peut le remarquer aussi sur les numéros de pages :



Nous allons maintenant passer à la partie de la recherche. L’utilisateur doit pouvoir rechercher un patient avec un mot clé. Pour cela on ajoute dans **patients.html** un formulaire offrant un champ de saisie et un bouton **Chercher** :



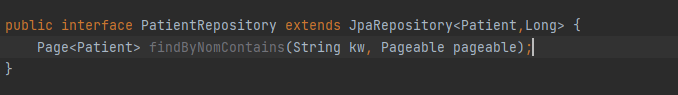
Quand on exécute :



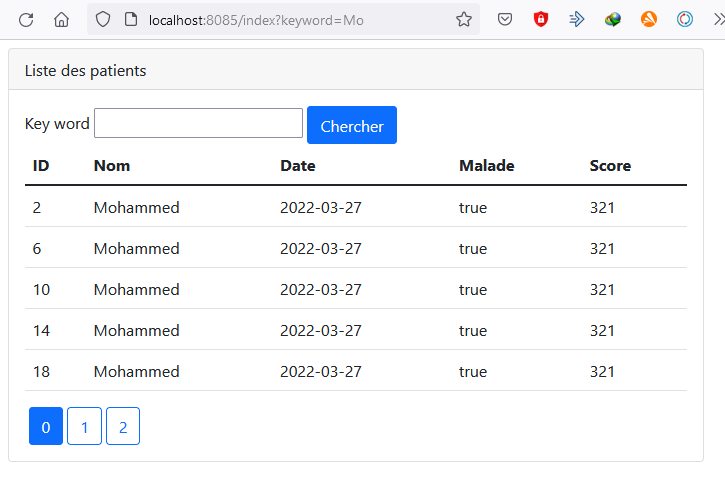
On ajoute keyword comme paramètre de l’url et un traitement se fera avec ce keyword. En effet on va prendre le keyword et l’utiliser dans une méthode de **PatientRepository** qui va nous permettre de faire la recherche. Elle va retourner tous les patients dont les noms contiennent keyword.

On aura :

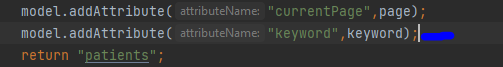




Quand on exécute et recherche **Mo,** comme on peut le constater dans l’url, on nous affiche la liste des patients dont les noms contiennent **Mo** :



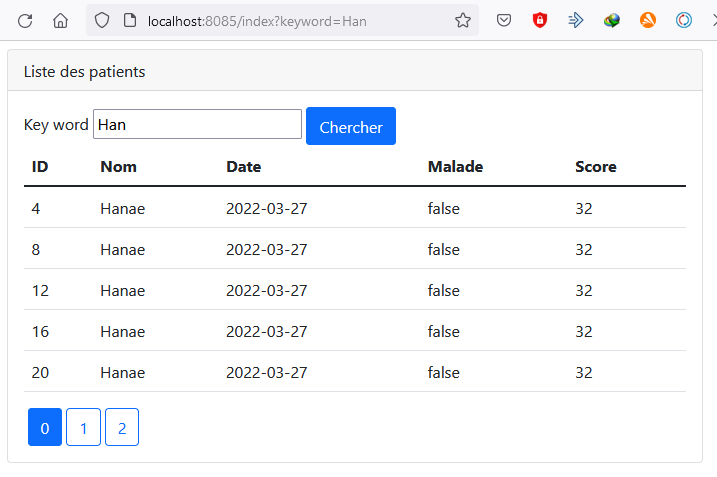
On remarque tout de même que le champ de saisie devient vide. Pour y remédier on peut ajouter un attribut dans le modèle, attribut dans lequel on va stocker keyword.



Puis on donne keyword comme valeur à l’input, le champ de saisie :

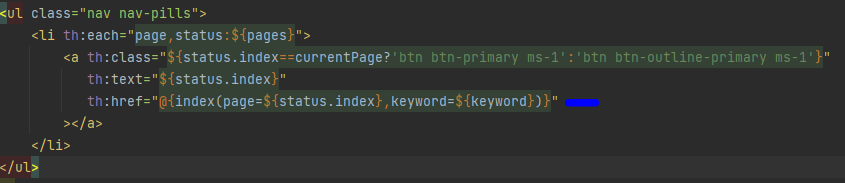


On peut voir maintenant que quand on fait une recherche, le mot clé ne disparait pas :

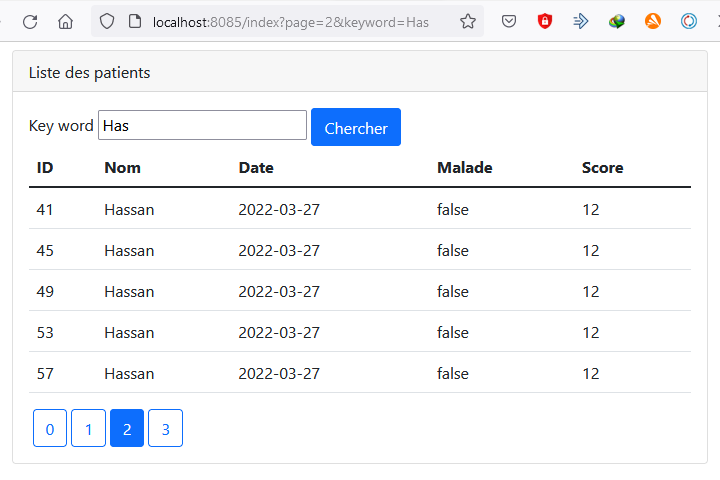


Mais on a toujours un problème. Quand par exemple ci-dessus, on clique sur 2 pour afficher la page 2, on perd le mot clé et on obtient l’affichage de la page 2 correspondant à tous les patients. On ne tient donc plus compte de la recherche.

Pour y remédier, sur les liens des numéros de page, on ajoute le keyword. Come suit :

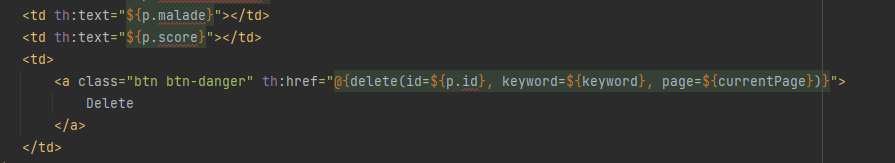


Maintenant quand on fait la recherche on peut parcourir les pages concernant la recherche, les pages correspondant au mot clé saisi, le keyword.



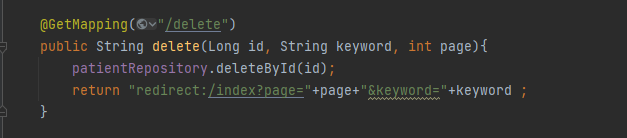
Maintenant passons à la suppression.

Commençons par ajouter une colonne après la colonne score. La colonne contiendra pour chaque patient un bouton **Delete** sur lequel on clique pour supprimer le patient :

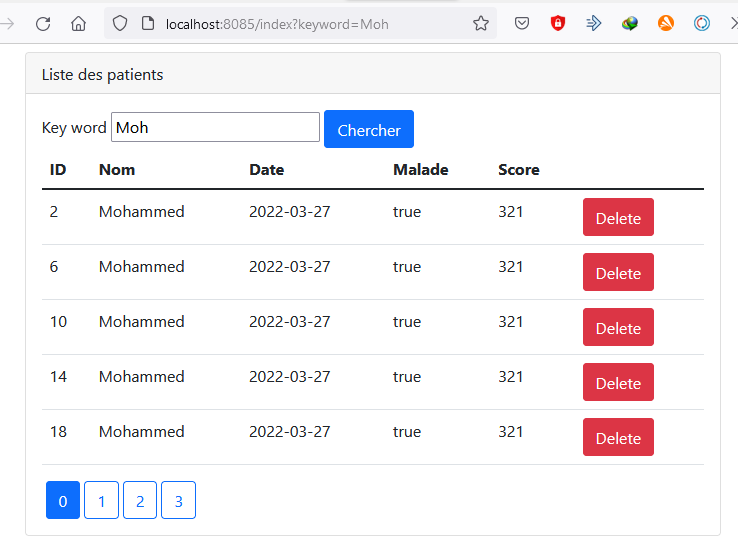


Par la suite on ajoute dans une contrôleur une fonction **delete** qui va être exécutée sur l’adresse **localhost:8085/delete** et qui va faire une redirection.

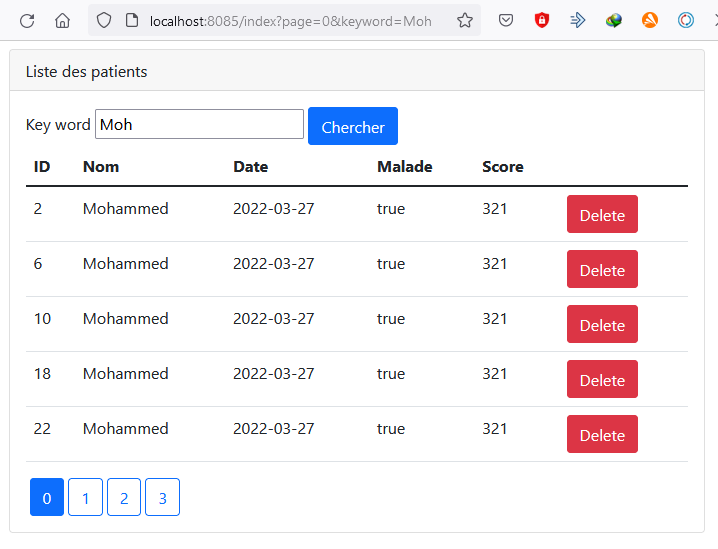
La fonction **delete** qui prend **id** parmi ses paramètres, supprime le patient ayant cet **id** et la redirection se fait aussi avec les autres paramètres comme **page** et **keyword** de sorte qu’après la suppression on reste sur la même page :



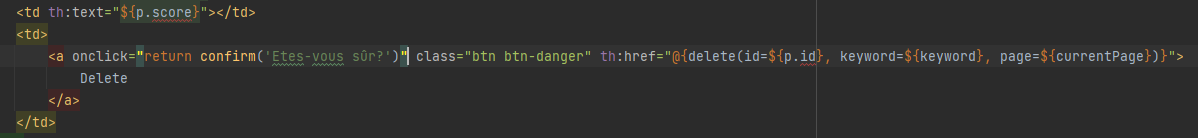
On va supprimer le patient qui le 14 :



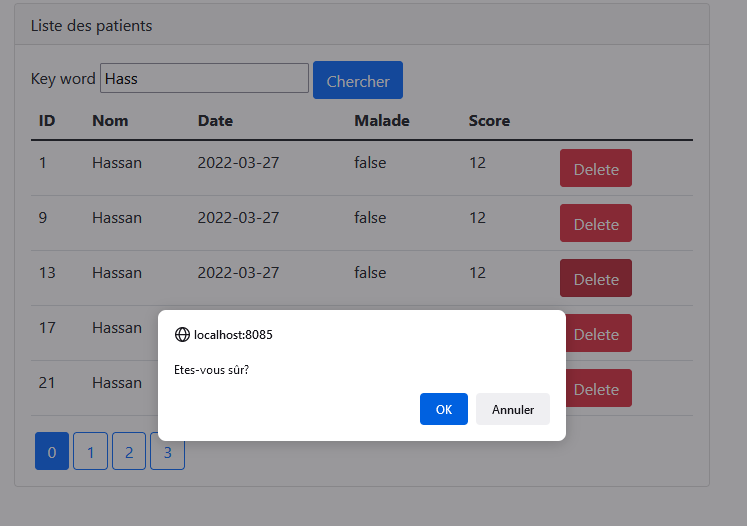
Après la suppression on reste sur la même page comme on peut le voir ci-dessous :



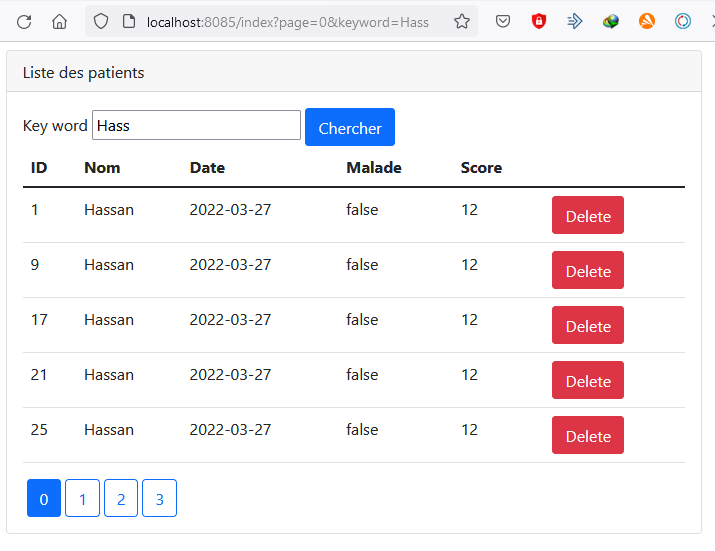
On va demander la confirmation à l’utilisateur avant la suppression. On fait donc :



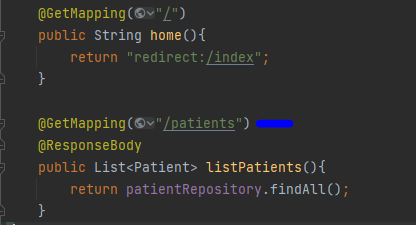
Maintenant quand on essaie de supprimer, une confirmation est demandée. Quand on fait annuler la suppression ne s’effectue pas. Si on clique sur OK le patient est supprimé :



On clique sur OK et on remarque que le patient qui a id = 13 est supprimé :



On peut aussi retourner sous format JSON la liste des patients. Ce qui nous sera utile lorsque nous feront le traitement côté client.



Sur l’adresse **localhost:8085/patients** on a :

