|  |
| --- |
| Institut Limayrac |
| Serveur de Tickets |
| Document projet |

|  |
| --- |
| Yohann  30/01/2022 |

Table des matières

[I. Présentation d 1](#_Toc94469352)

[II. Scénarios possibles 1](#_Toc94469353)

[A. L’ajout d’un utilisateur 1](#_Toc94469354)

[B. Se connecter & consulter les tickets 4](#_Toc94469355)

[C. Création de tickets 7](#_Toc94469356)

[D. Sécurisation de l’application 9](#_Toc94469357)

[E. Gestionnaire d’erreurs 11](#_Toc94469358)

[III. Bilan 11](#_Toc94469359)

1. Présentation du projet

Le projet fait initialement suite aux cours de Java Spring allant de l’initiation au framework à la mise place d’une application permettant l’envoi d’une page web à la suite d’une requête d’un client mais également l’échange de données avec une base.

Il a ainsi été convenu avec le professeur d’un objectif mettant en oeuvre ces points vus en cours, en y ajoutant les inconnues qu’amène n’importe quel projet viant l’étude d’un nouveau framework.

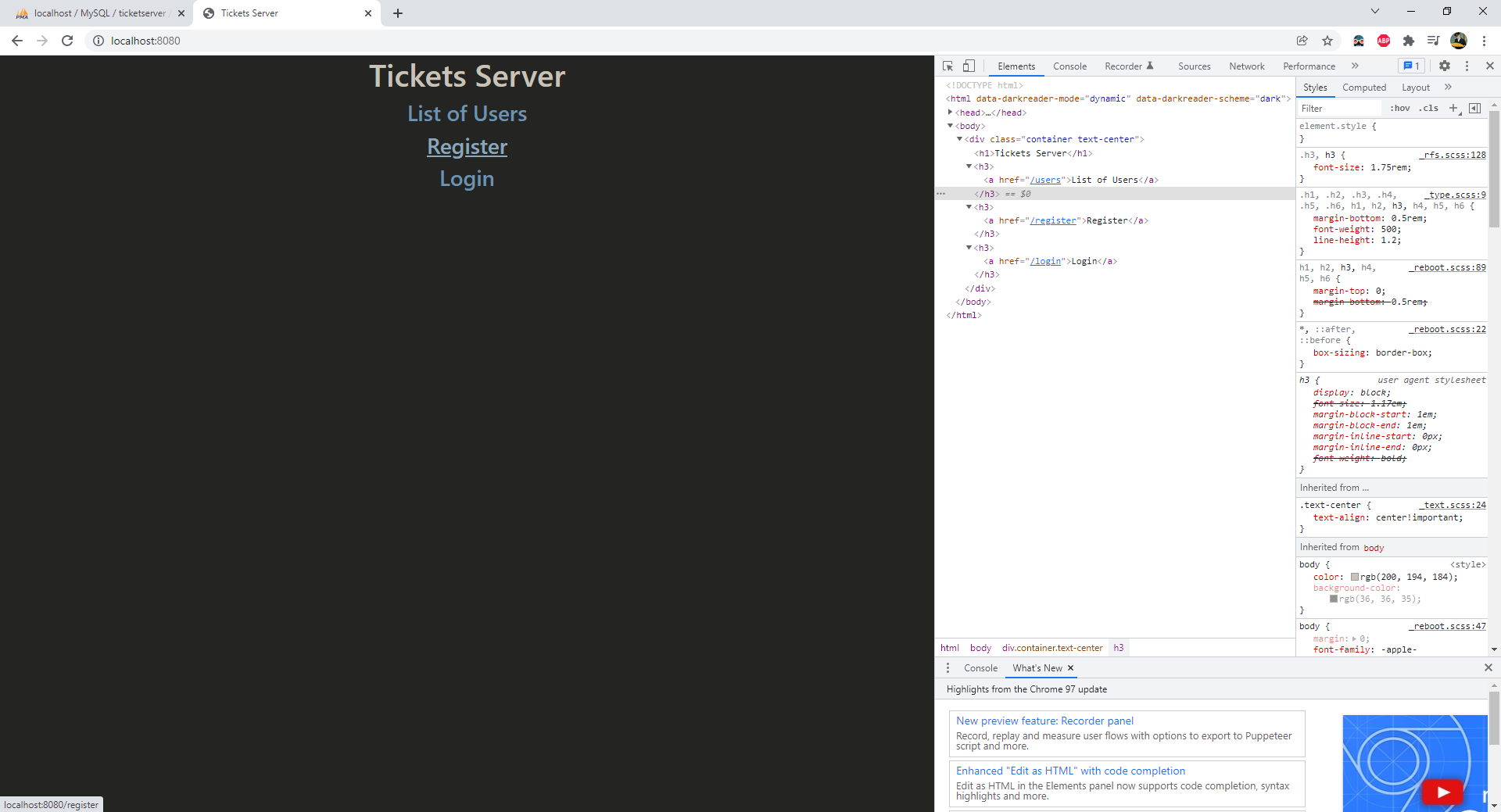
# Scénarios possibles

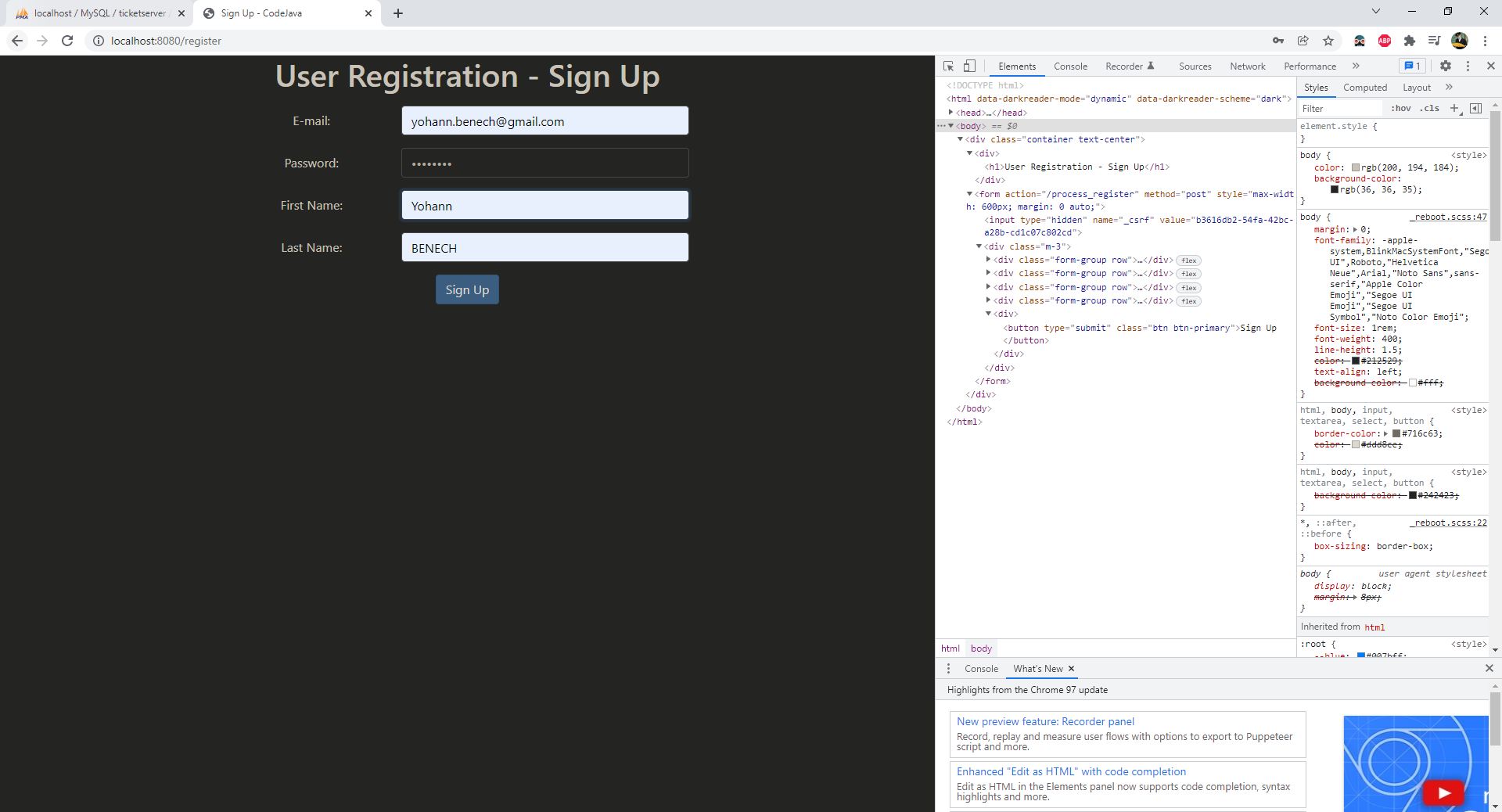
Le 30/01/2022, l’application permet les actions suivantes :

L’ajouts d’utilisateurs, la connexion via un utilisateur, la consultation des tickets, la création de tickets. L’application possède en outre une sécurisation basique de son arborescence, afin d’éviter aux personnes non-identifiées d’accèder aux ressources sensibles. Le serveur, comme vous le verrez possède également un système de gestion des erreurs (sous la forme d’un contrôleur).

Ces fonctionnalités seront ci-dessous présentées dans l’ordre.

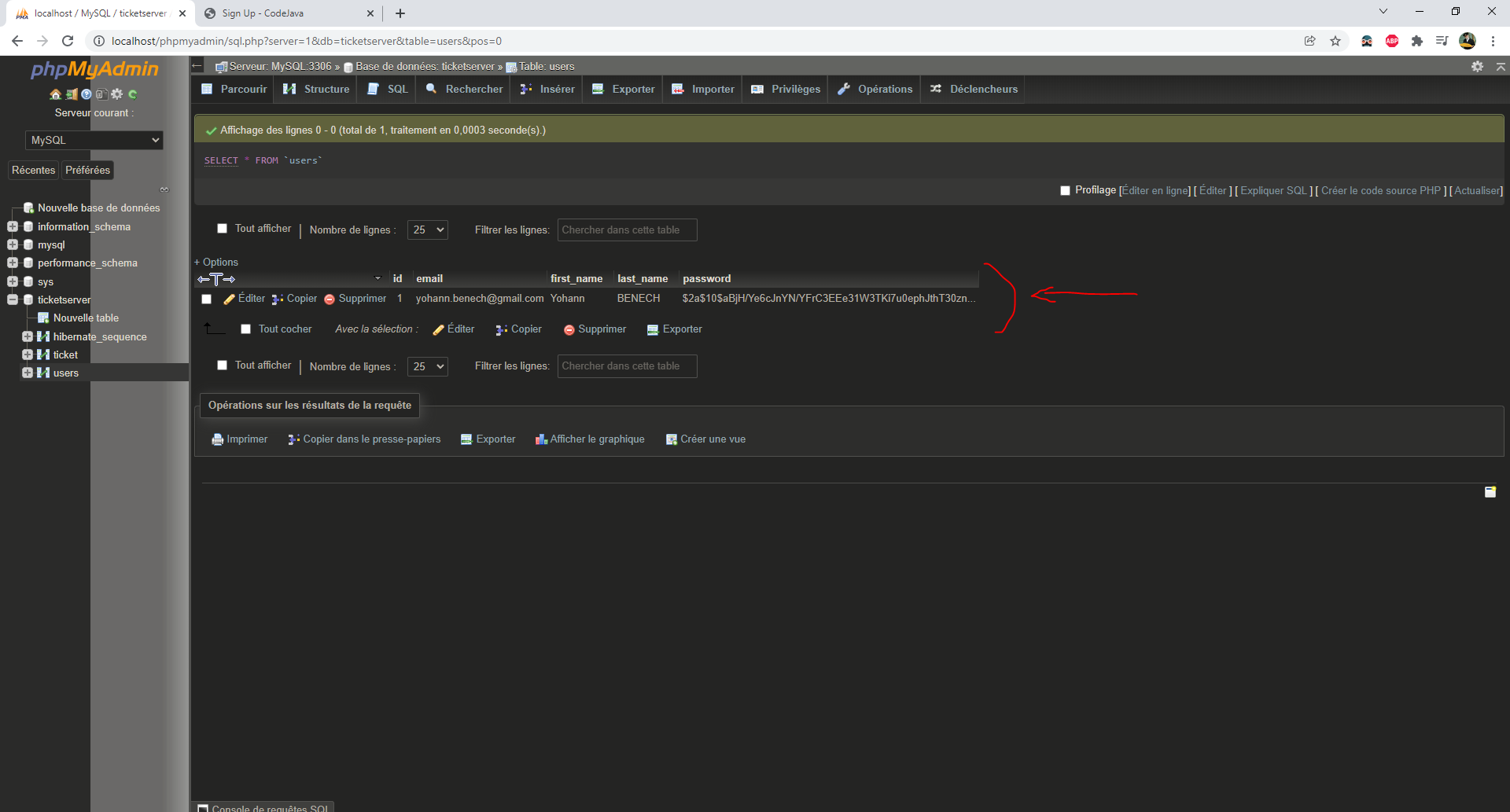
## L’ajout d’un utilisateur



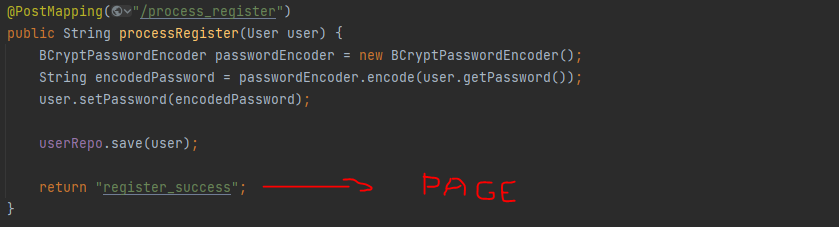


Le mot de passe caché utilisé est « password ». Je n’ai pas jugé utile de mettre une verification bien que sachant le mettre en place et nécéssaire par la suite du déroulement du projet.

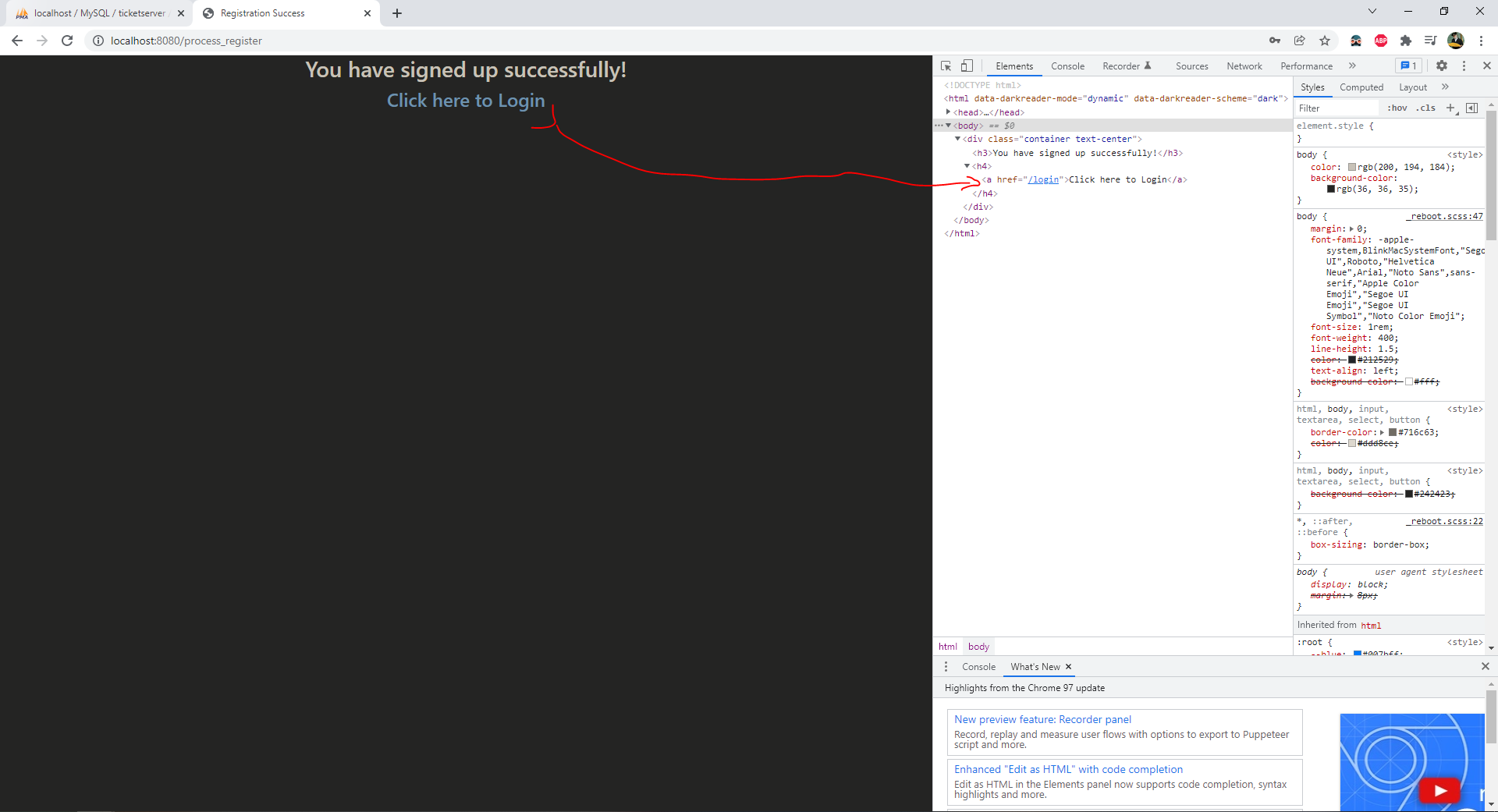
Résultat BDD :



Comme vu ci-dessus le mot de passe est haché (hash), question de sécurité... Ceci à l‘aide de la classe BcryptPasswordEncoder implémentant la classe org.springframework.security.crypto.password.PasswordEncoder. Code ci-joint :

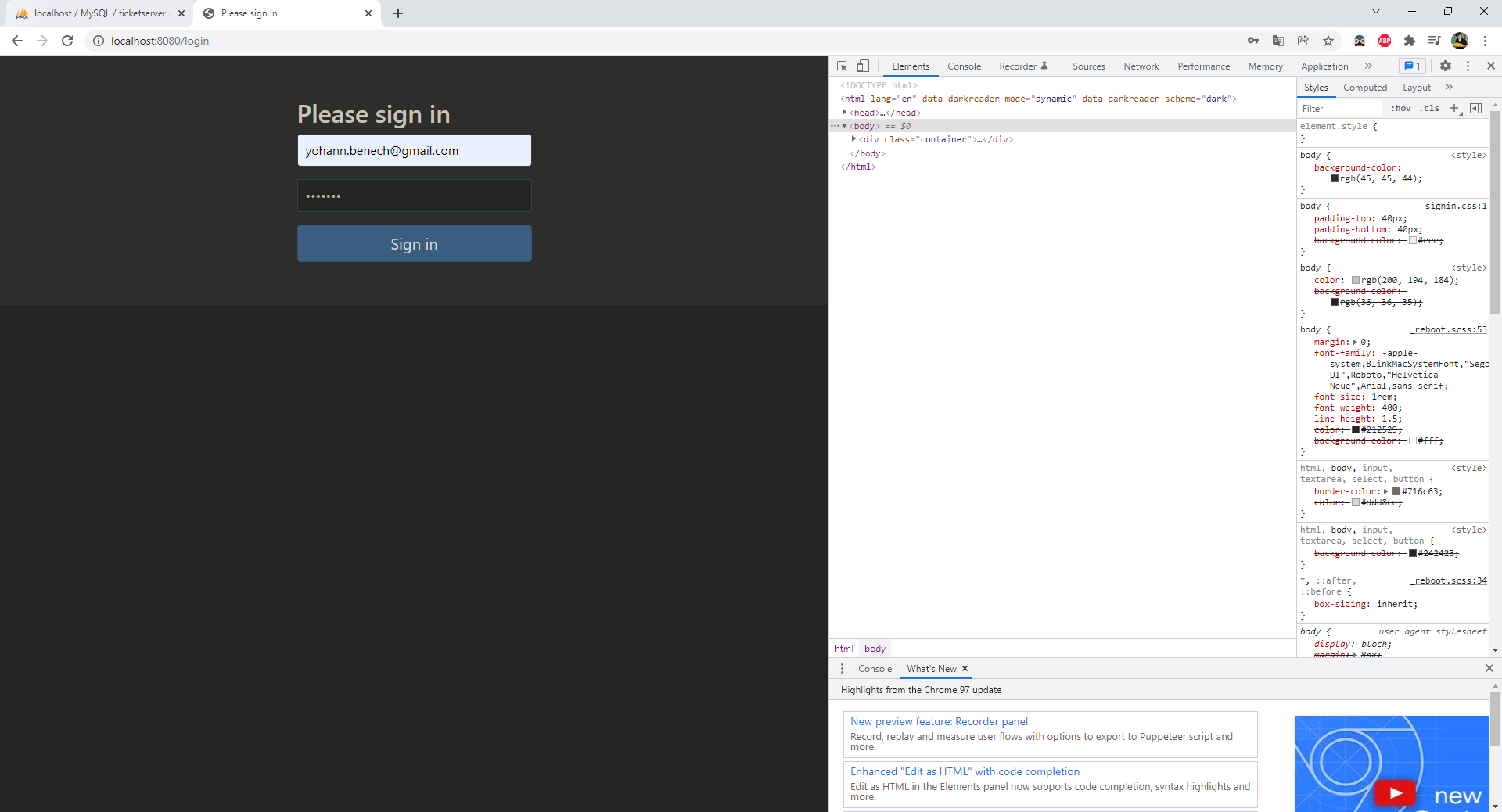


Une fois l’utilisateur crée, vous atterirez sur une page de confirmation.

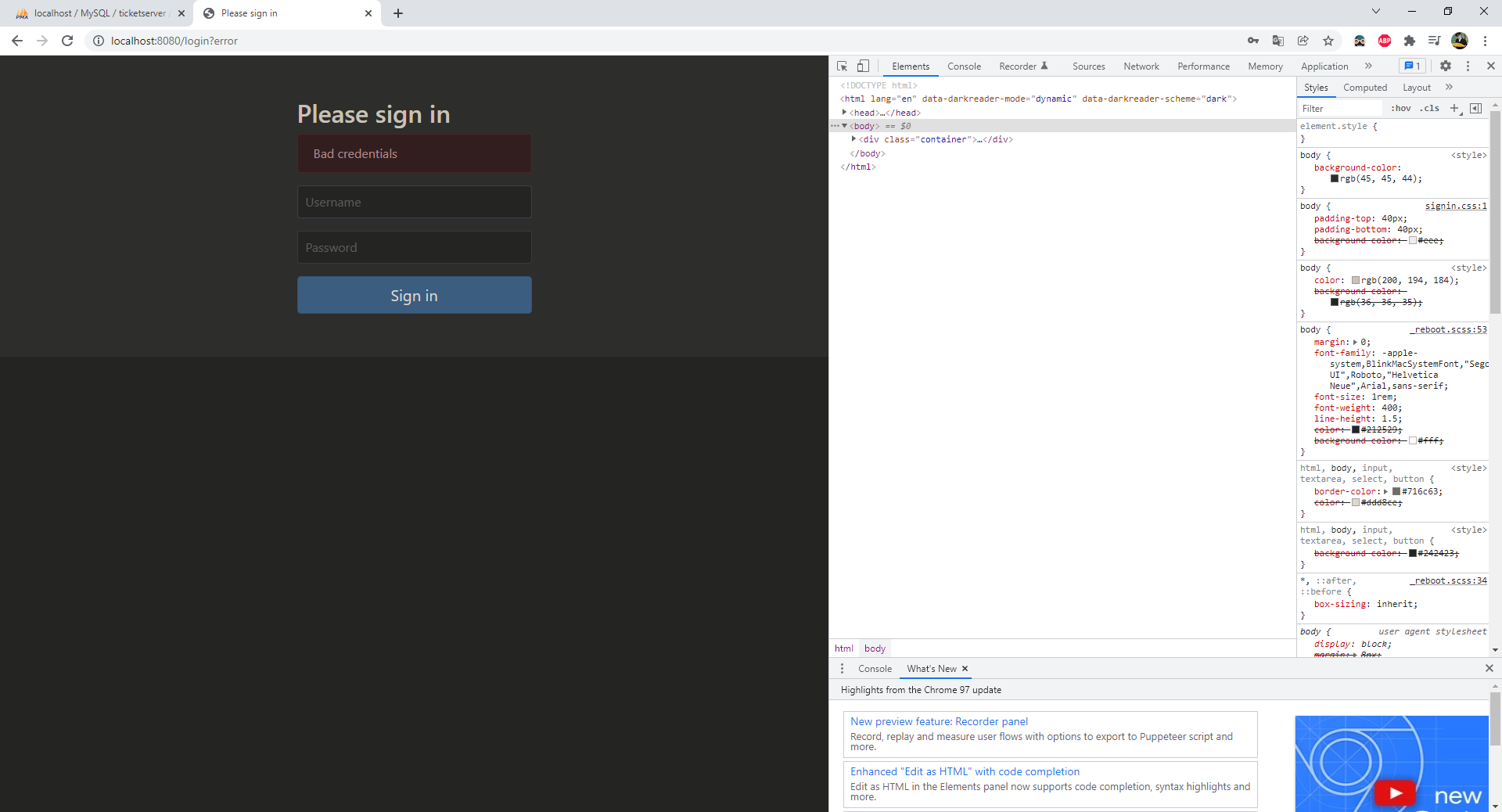


Vous pourrez cliquer sur le lien indiqué pour vous connecter.

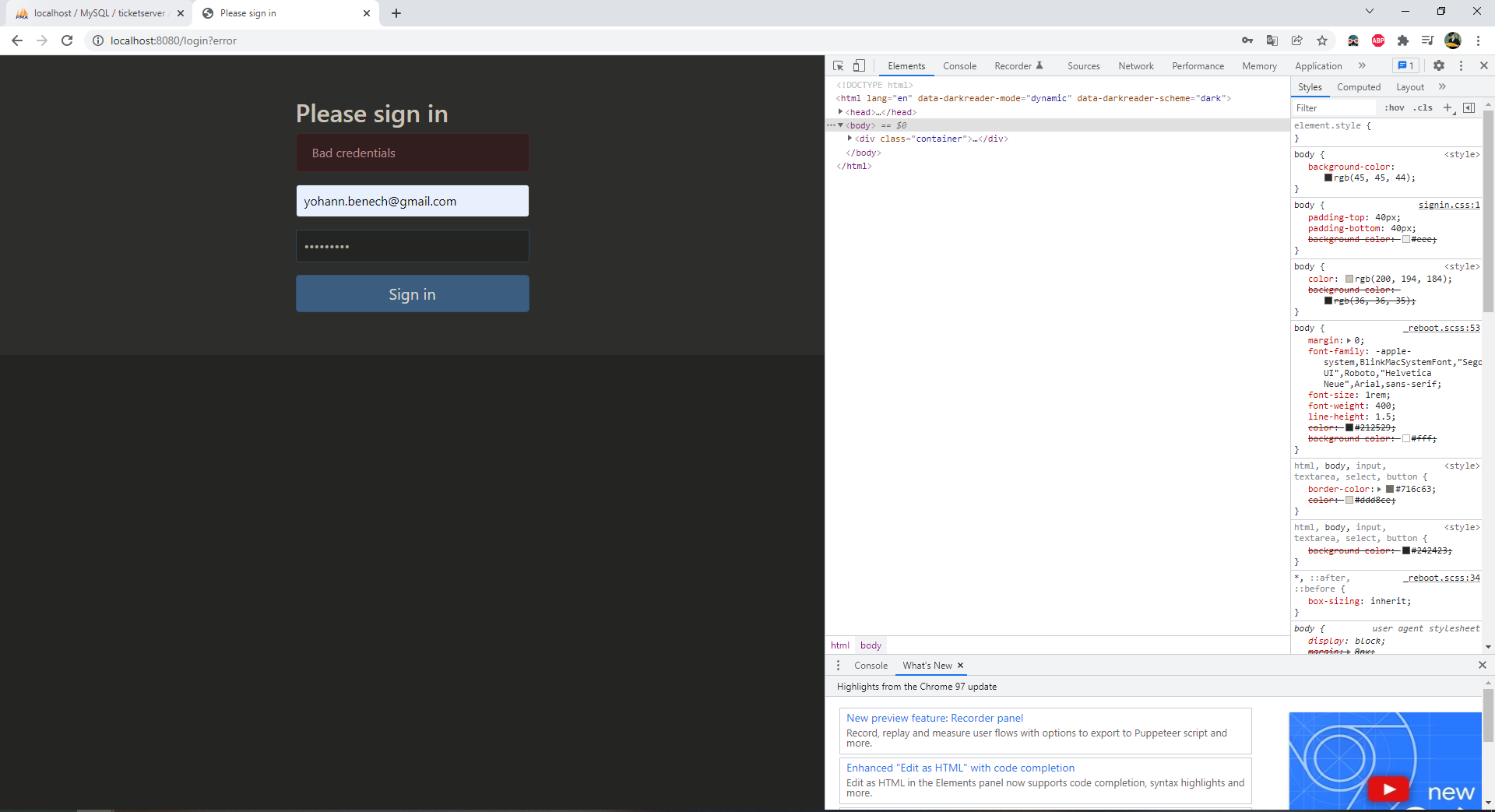
## Se connecter & consulter les tickets



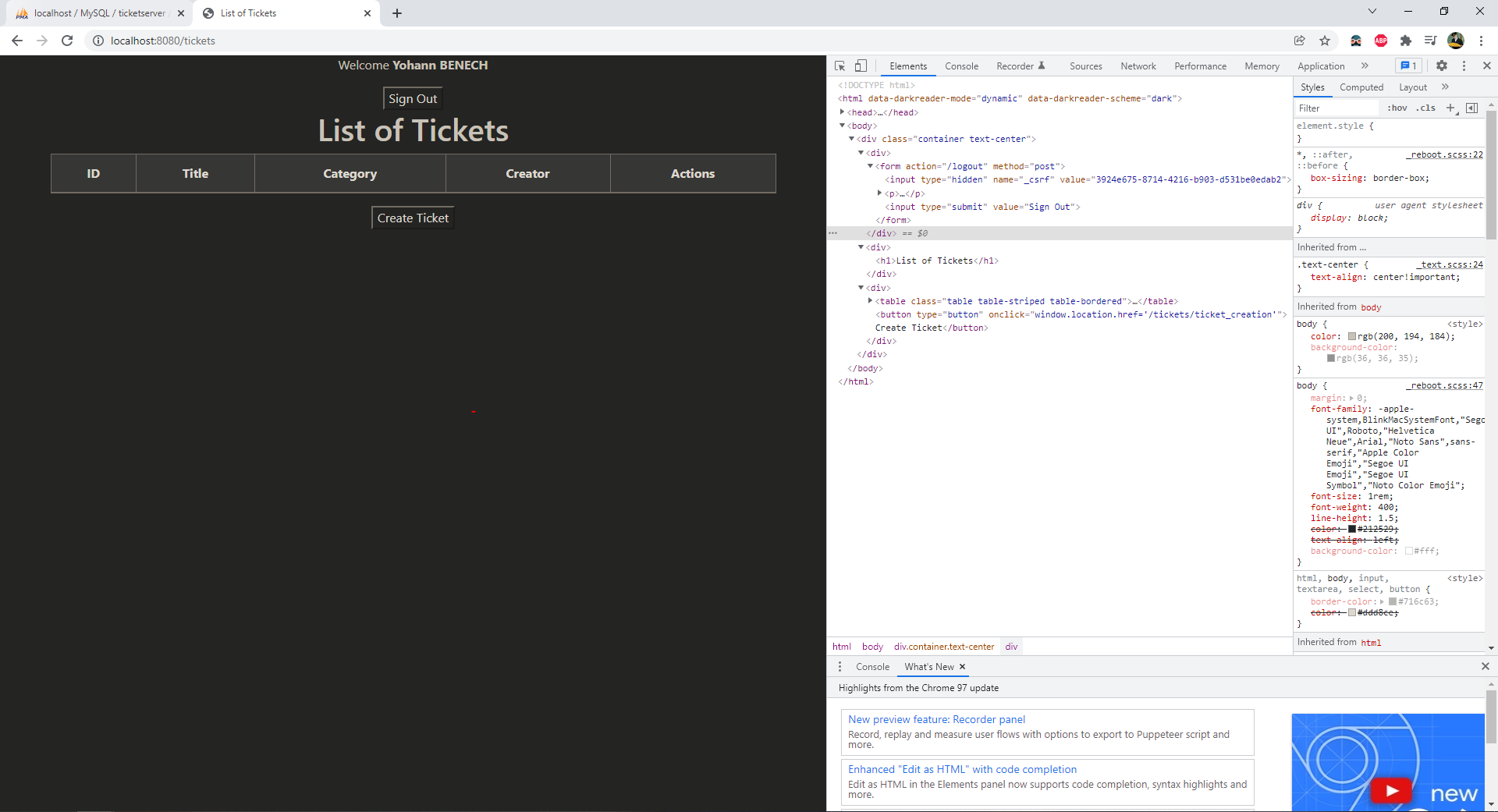
Le mot de passe que l’on va essayer est « passwor », il manque le « d ». Le but est de montrer que le serveur refuse l’authentification à moins que le bon mot de passe soit utilisé, à savoir ici, « password » en n’oubliant pas le « d ».



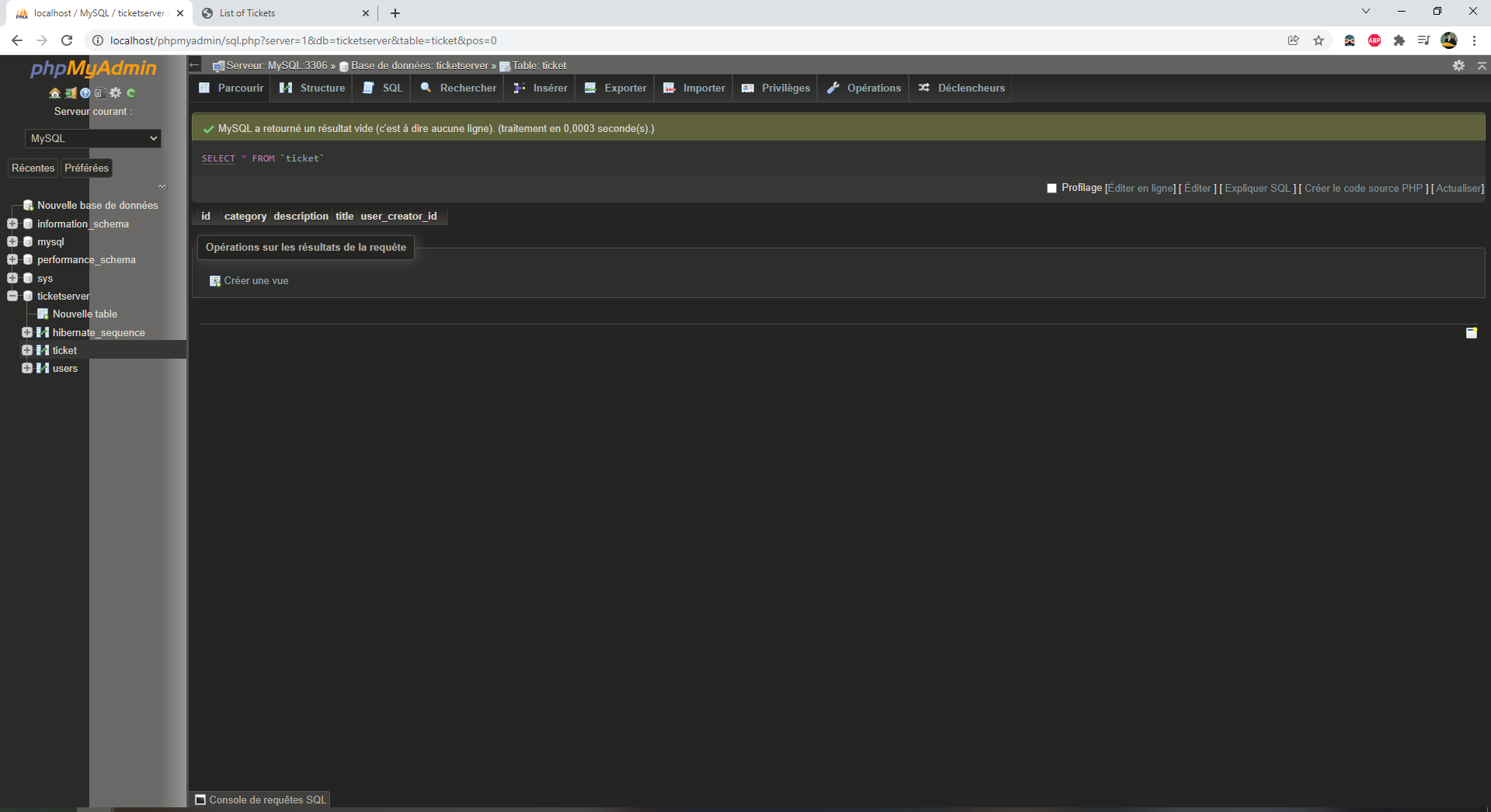
Le serveur a bien refusé l’authentification. Nous essayons alors avec le bon mot de passe.



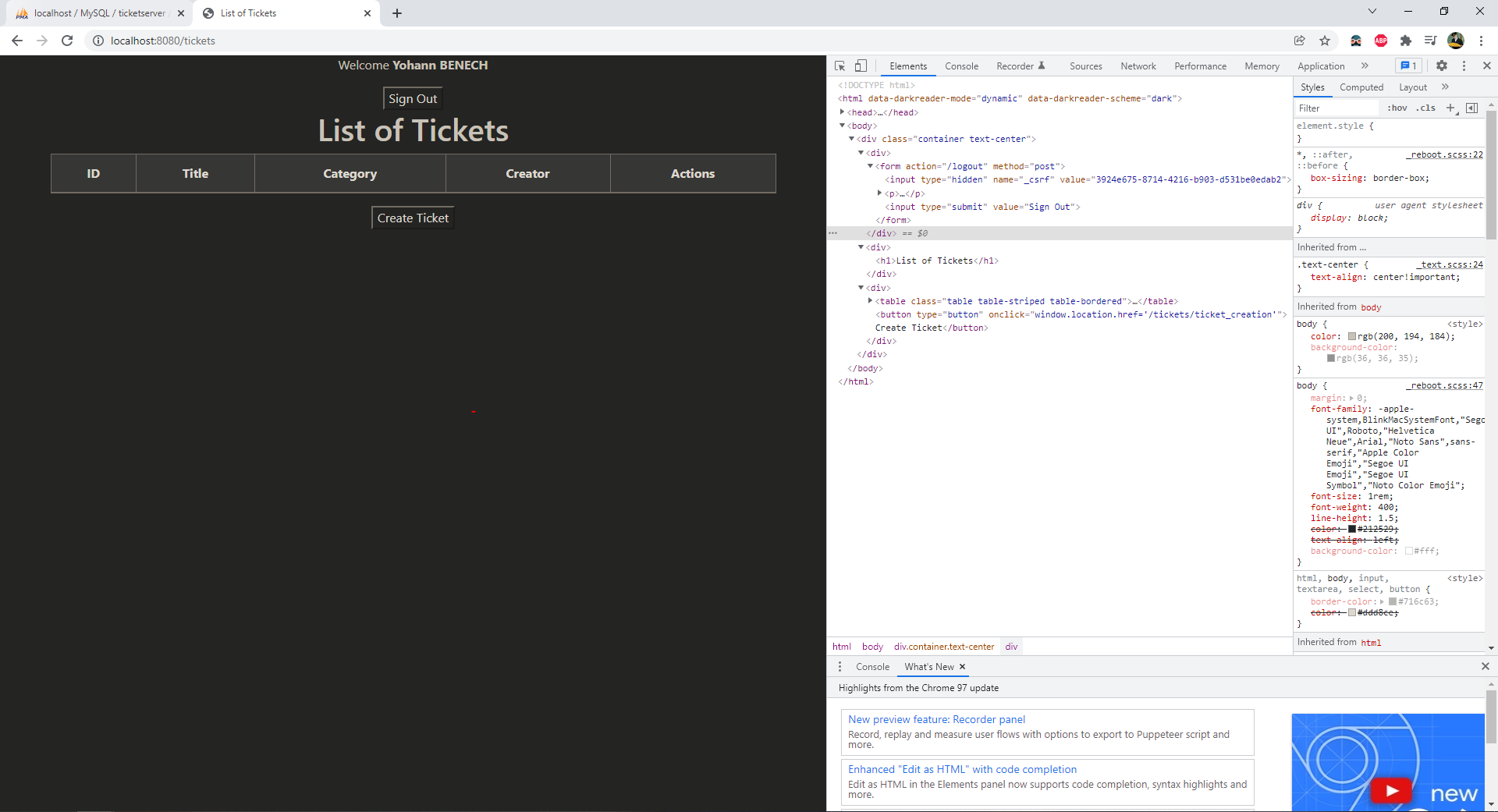
On atterit donc sur la page suivante.



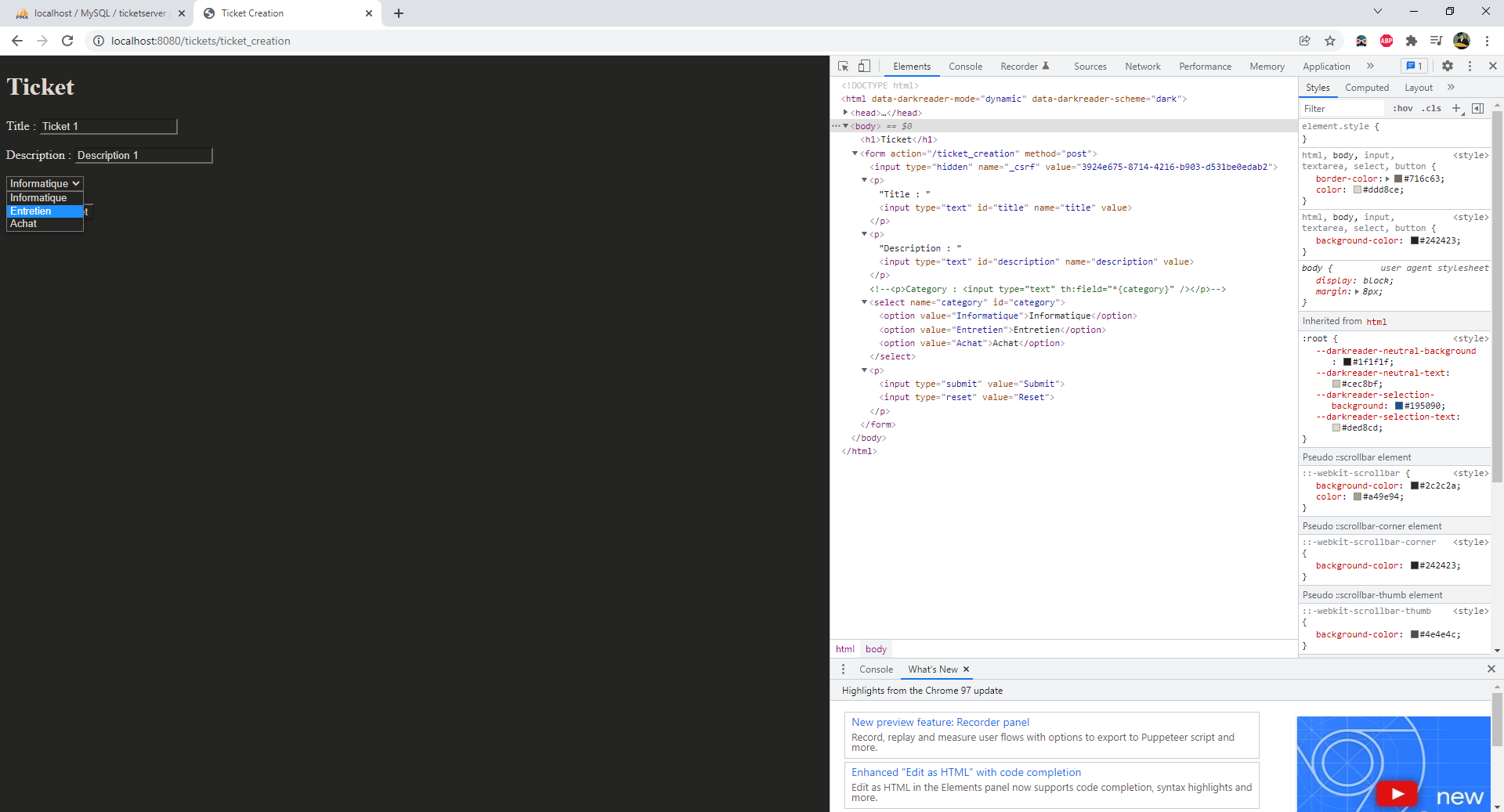
Cette page est censée afficher la liste de tous les tickets dans la BDD. Elle est logiquement vide :



## Création de tickets

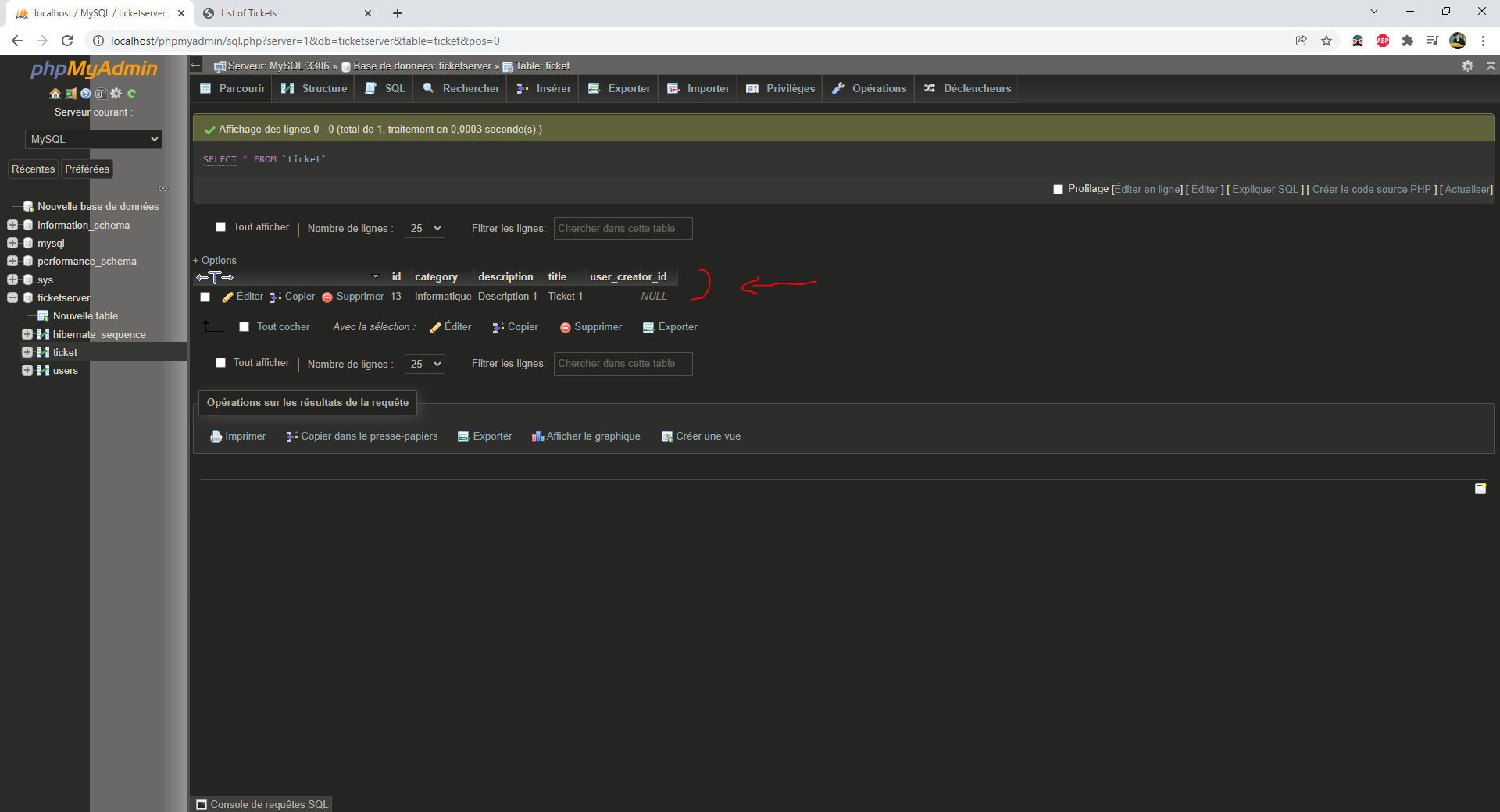


Cliquez sur « Create Ticket ».

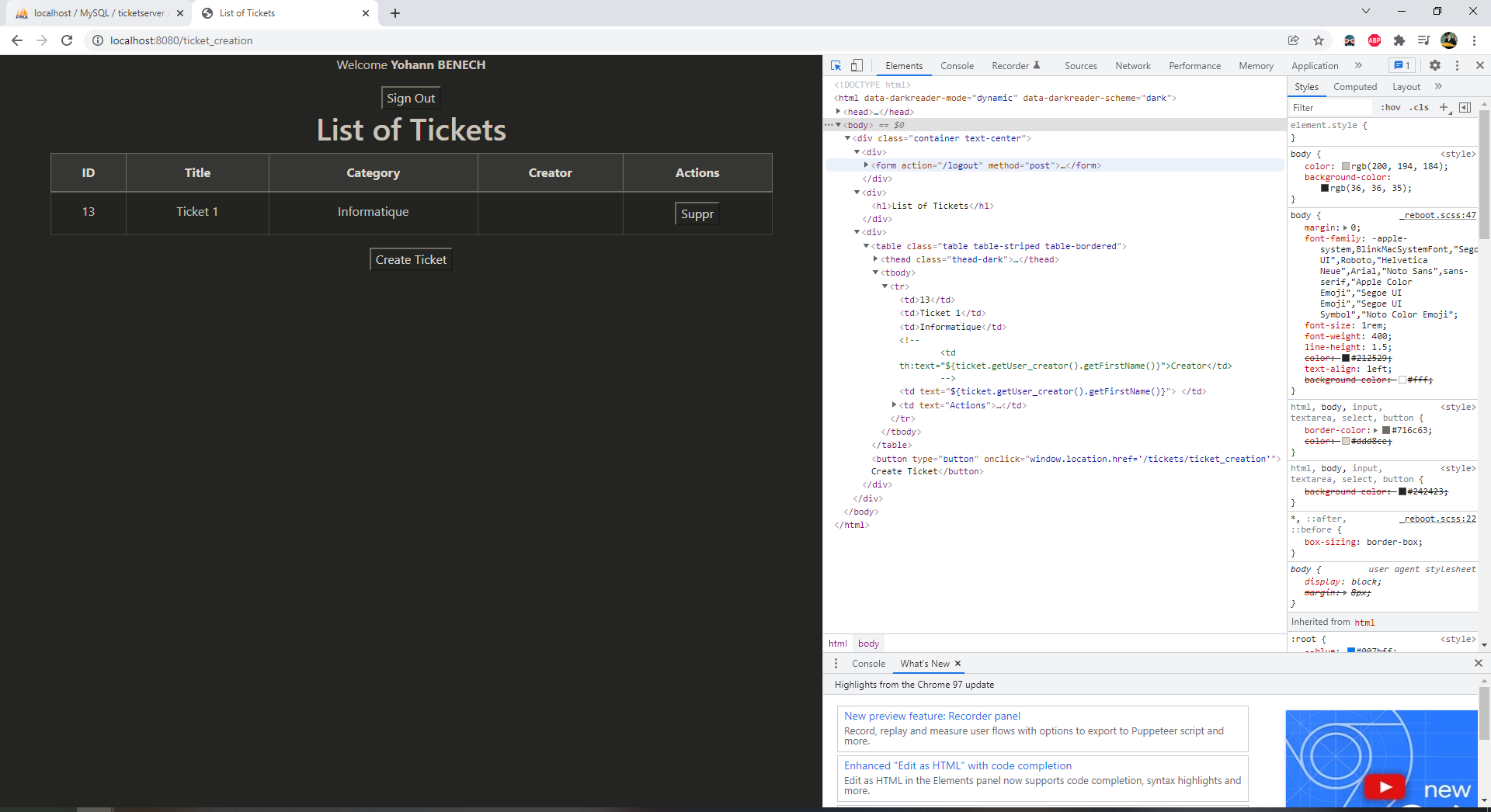


On atterit sur cette page ou l’on peut compléter les champs du ticket. Une liste des categorie a étée mise en place dans le code HTML via un « select » et ses options. On pourrait cependant imaginer à l’avenir lister automatiquement ces dernieres en passant le repository d’une classe « category » à la vue.

Cliquez sur submit afin d’ajouter le ticket à la base. Avant d’être redirigé sur la vue générale des tickets. Verifions la BDD :

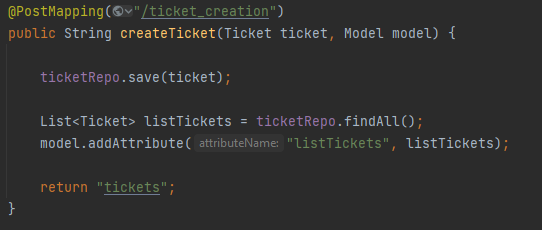


Le ticket a bien été ajouté ! Verifions que la vue a bien étée actualisée aussi.



C’est bon ! Comme vous le voyez le champs créateur est vide. En effet j’ai eu un soucis d’autorisations en essayant d’acceder au nom d’un utilisateur passé à la vue. Le bouton de suppression n’est pas encore fonctionnel.

Code permettant l’actualisation de la vue :



On actualise en fait le repository avant de le re-passer à la vue !

## Sécurisation de l’application

La sécurisation de l’application passe par l’utilisation des classes :

CustomUserDetails 🡨 UserDetails :

Défini la classe utilisateur (« User ») qui sera utilisée pour la partie authetification. En fait, elle permet de faire le lien entre notre classe « User » et la clase « User » utilisée par la partie authentification.

CustomUserDetailsService 🡨 UserDetailsService :

Défini la couche Service de la partie authentification, on y defini un repository pour nos utilisateurs. Qui nous permettra par la suite de manipuler nos utilisateurs.

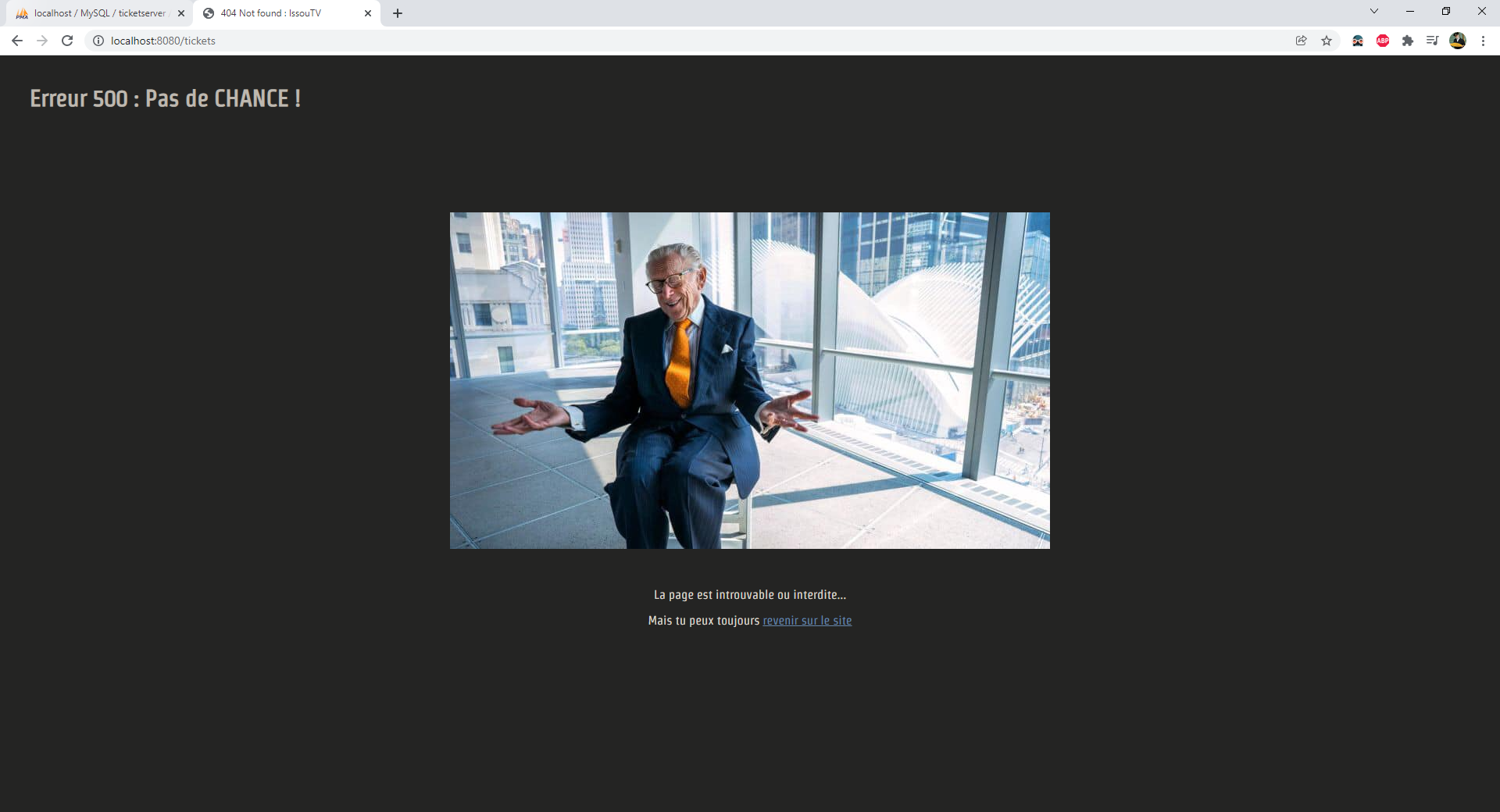
WebSecurityConfig 🡨 WebSecurityConfigureAdapter :

Configure notre application, y figurent notamment à partir de quelle arborescence il est nécessaire de s’authentifier et quels droits y sont appliqués.

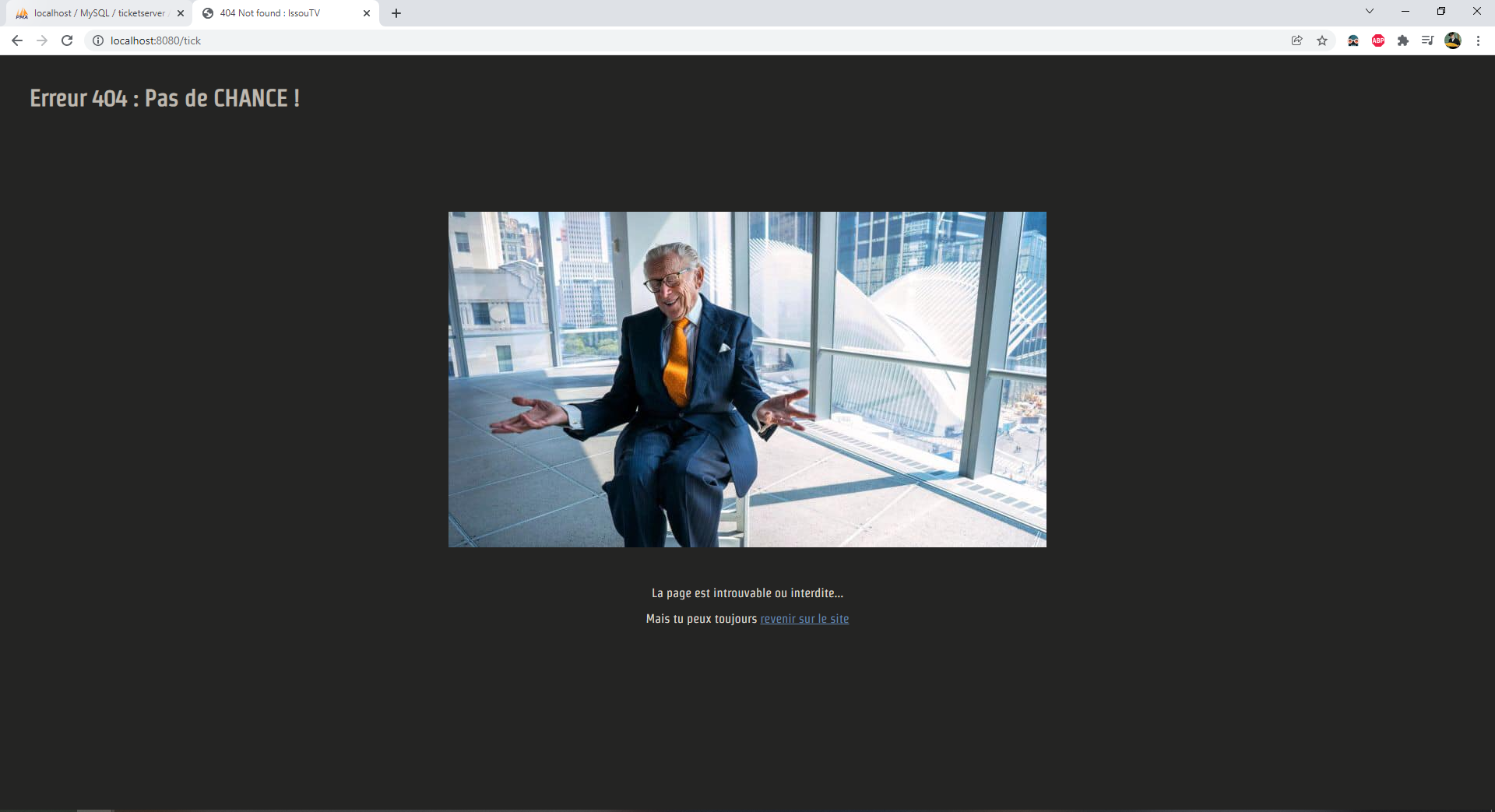
**Pour une explication complète, qui fut une de mes source d’apprentissage consultez directement ce** [**lien**](https://www.codejava.net/frameworks/spring-boot/user-registration-and-login-tutorial)**.**

Précision, le bouton « Sign Out » permet effectivement de se déconnecter de l’utilisateur courant tout en vous retournant sur la page d’accueil du serveur. Vous ‘aurez accès à rien à moins d’avoir été authentifié une fois de plus.

Test sans authentification d’acceder aux tickets (erreur 500 donc) :



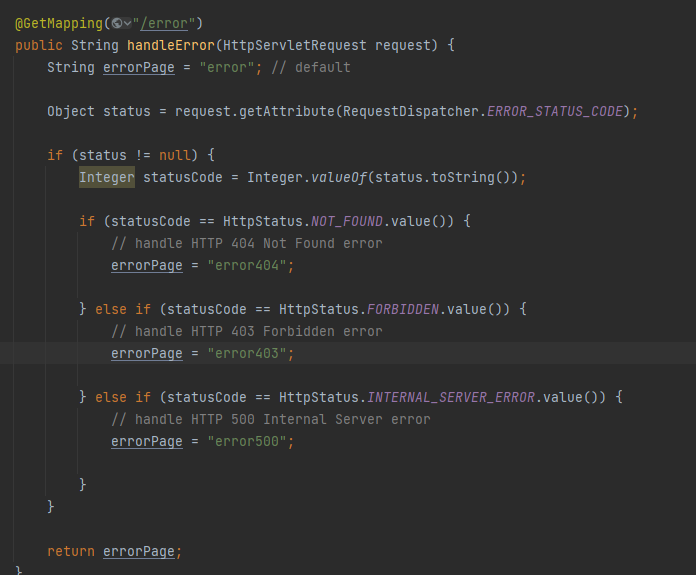
Test d’accès à une ressource non existante (erreur 404 donc) :



Le lien en bas permet de retourner sur la page d’accueil du serveur.

## Gestionnaire d’erreurs

Simplement un controller récupérant les erreurs renvoyées par le serveur et renvoyant les pages adéquates. Code ci-joint :



Comme vous le voyez seules 3 erreurs sont reellement prises en compte. Mai rien n’empeche d’en gerer plus ! Il suffira d’ajouter un « elseif » testant le status HTTP et le tour est joué.

Pour le test de ces erreurs (sauf 403) se référer à la partie précédente.

# Bilan

Le projet m’a permit d’apprendre de nouvelles chose, que se soit vis à vis des cours, fonctionnement de base d’un projet, sa structure, ou bien l’utilisation de thymeleaf pour l’interprétation de pages. Cependant dans mon cas personnel, ayant loupé pas mal de cours j’ai du compter sur de nombreuses heures de cours en ligne pour rattraper le niveau de la classe (ou du moins le niveau qu’elle est censée avoir). Ci-dessous quelques unes de mes sources :

[Spring Boot Tutorial | Full Course [2021] [NEW]](https://www.youtube.com/watch?v=9SGDpanrc8U)

[Spring Boot and Spring Security with JWT including Access and Refresh Tokens](https://www.youtube.com/watch?v=VVn9OG9nfH0)

[Spring Boot Roadmap - How To Master Spring Boot](https://www.youtube.com/watch?v=cehTm_oSrqA)

En effet j’ai testé pas mal de choses comme la mise en place de Web Services renvoyant le contenu de la BDD mis en forme (JSONs), la génération de tokens (jetons), mais je n‘ai pas jugé utile de tout garder, privilégiant la livraison d’une application simple.

Malheureusement je n’ai pas réussi à mettre tout en place dans les temps, suppression/affectation des tickets. Cependant la partie sécurisation du serveur est de loin ce qui m’a pris le plus de temps et le plus mis en difficulté mais au final cela aura payé.