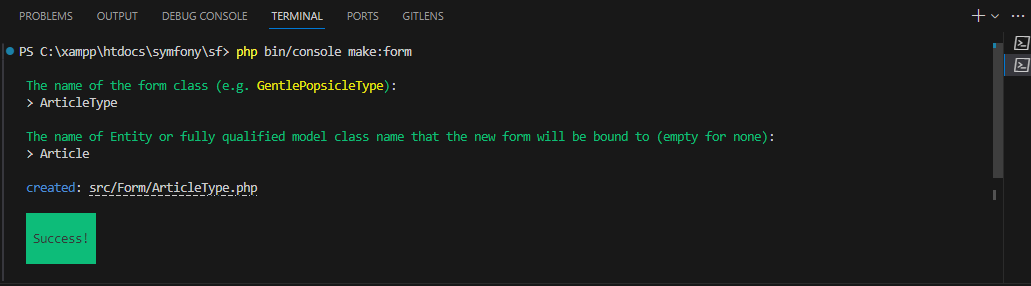
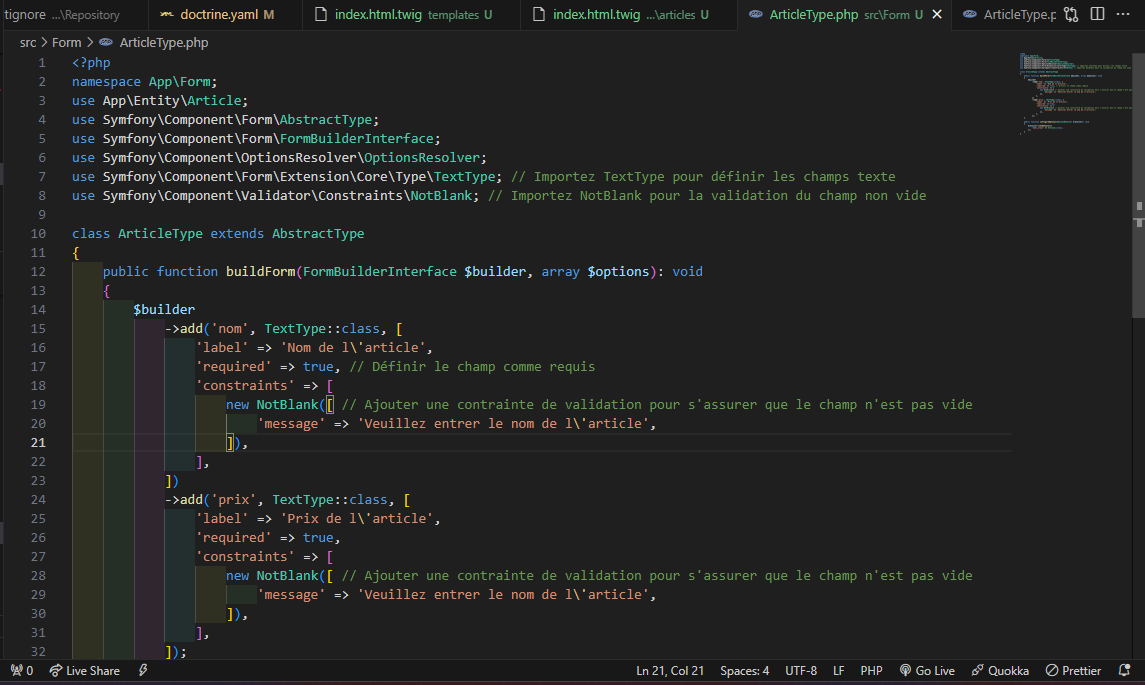
Compte rendu TP5

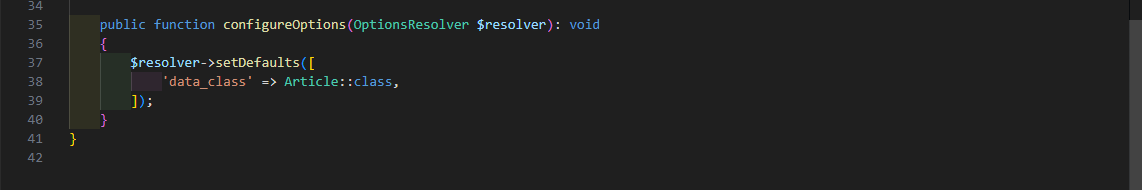
On doit tout d’abord créer la classe formulaire correspondant à l’entité Article en tapant la commande suivante : php bin/console make:form

Symfony 7 simplifie la création de formulaires en générant automatiquement des classes de formulaire pré-configurées à partir des entités Doctrine avec la commande make:form. En fournissant le nom de l'entité, Symfony crée des champs de formulaire correspondant aux propriétés de cette entité, réduisant ainsi la nécessité de définir manuellement chaque champ. Les classes de formulaire générées contiennent des méthodes pour configurer les champs et les validations, offrant une personnalisation facile et une cohérence dans la création des formulaires.



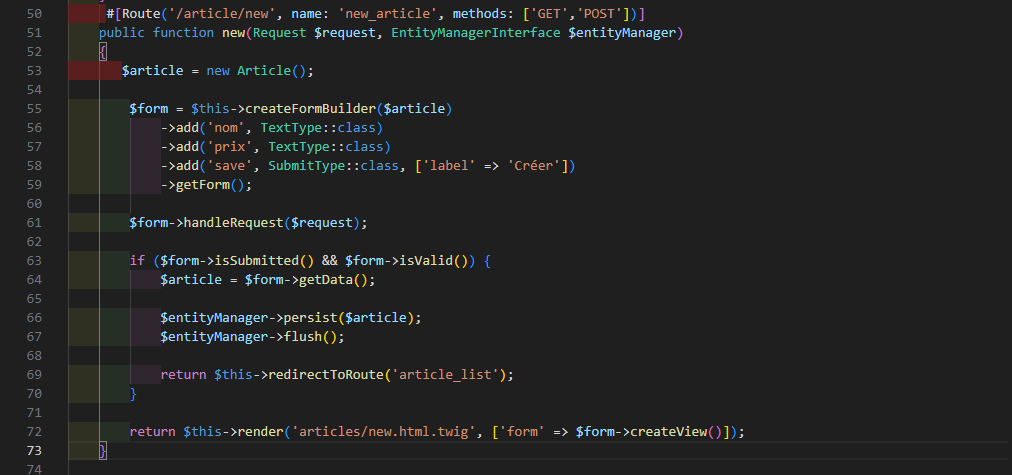
Ci-dessus se trouve une classe ArticleType utilisée pour définir un formulaire Symfony correspondant à l'entité Article. Cette classe utilise les composants de formulaire Symfony pour créer des champs de formulaire pour le nom et le prix de l'article. Des contraintes de validation sont également ajoutées pour s'assurer que ces champs ne sont pas vides.





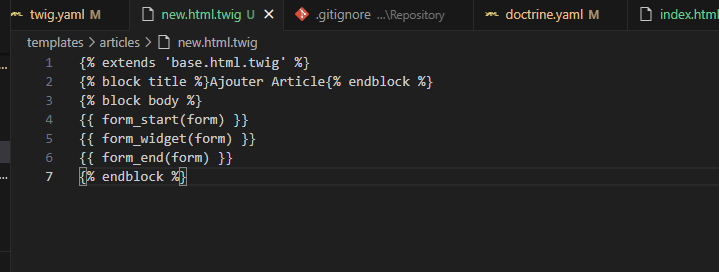
On ajoute “use App\Form\ArticleType “ au début de la page « indexController.php » pour accéder à la classe ArticleType.

La méthode **new** crée un formulaire pour ajouter un nouvel article. Lorsque le formulaire est soumis, il est vérifié. Si les données sont valides, un nouvel article est créé et enregistré dans la base de données. Ensuite, l'utilisateur est redirigé vers la liste des articles. Si le formulaire n'est pas valide ou n'est pas soumis, la page de création du formulaire indique une contrainte d'intégrité.

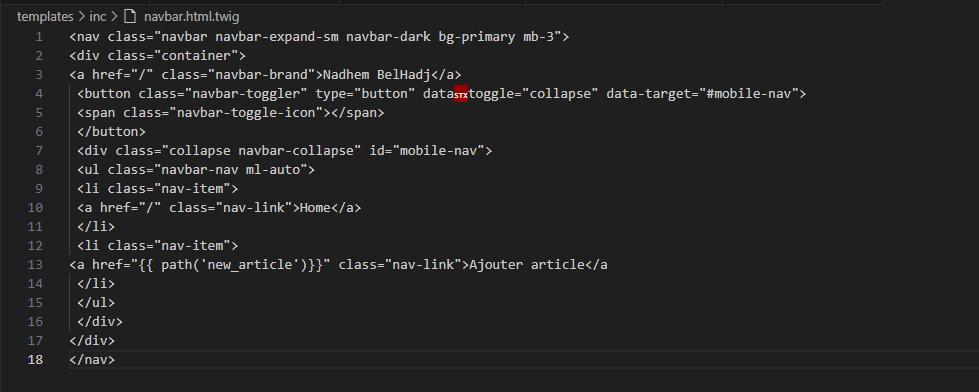


la vue new.htm.twig est une page qui affiche un formulaire pour ajouter un nouvel article, avec les champs correspondants définis dans le ArticleType associé au formulaire.

* form\_start(form) génère la balise de début du formulaire.
* form\_widget(form) affiche les champs du formulaire.
* form\_end(form) génère la balise de fin du formulaire.



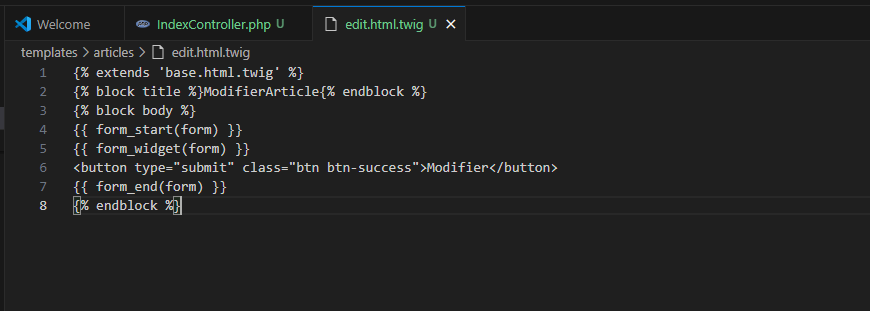
On ajoute maintenant la route ‘new\_articles’ dans le link ‘Ajouter article’ placé dans le navbar



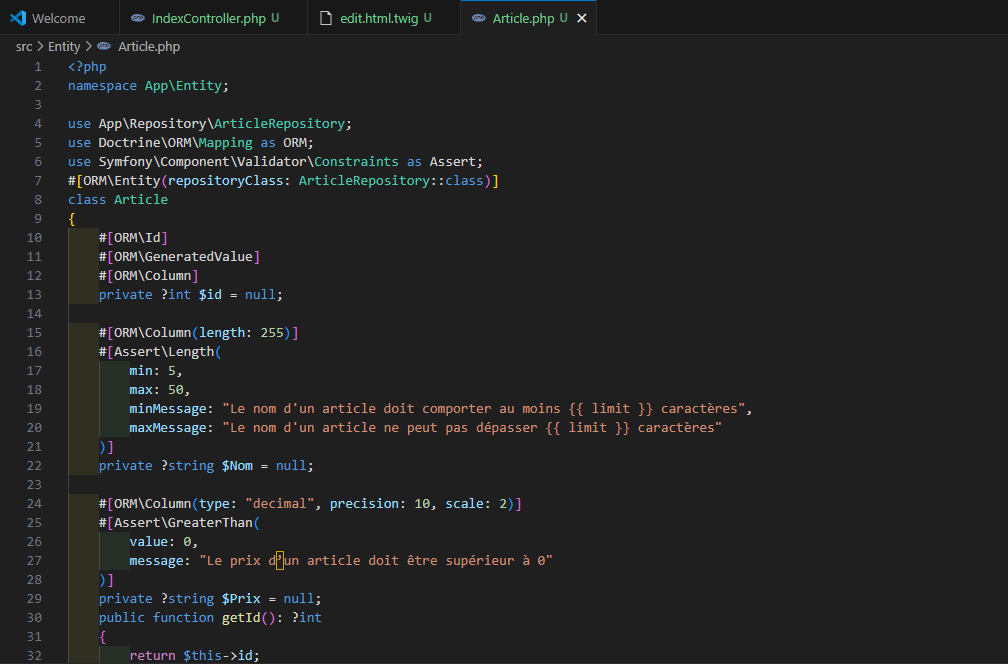
De même pour La fonction edit qui gère la modification des articles dans notre application. Elle récupère l'article à modifier, affiche un formulaire prérempli avec ses données, puis enregistre les modifications si le formulaire est soumis et valide. Sinon, elle affiche à nouveau le formulaire avec les erreurs éventuelles.

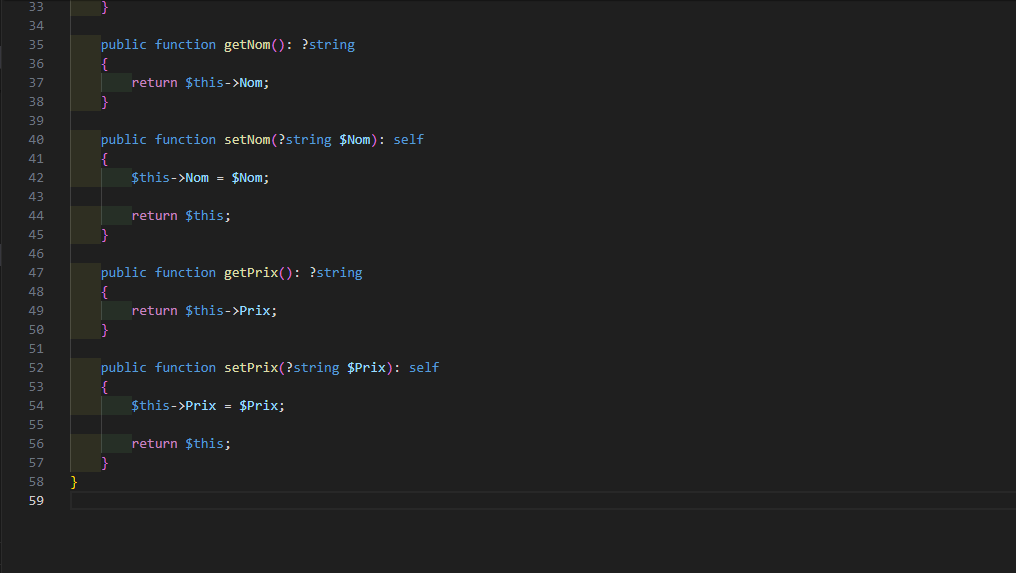


Ce code Twig définit la vue **edit.html.twig**, qui est utilisée pour afficher le formulaire de modification d'un article.

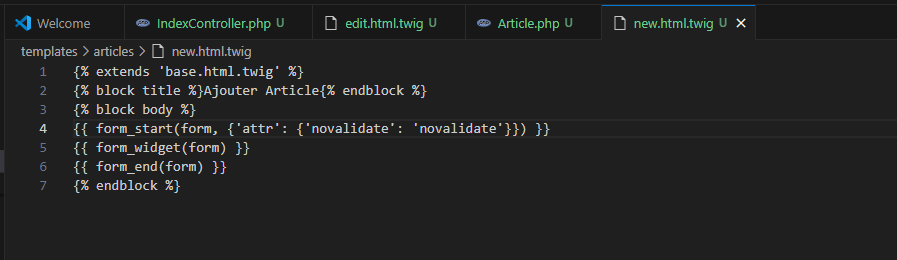


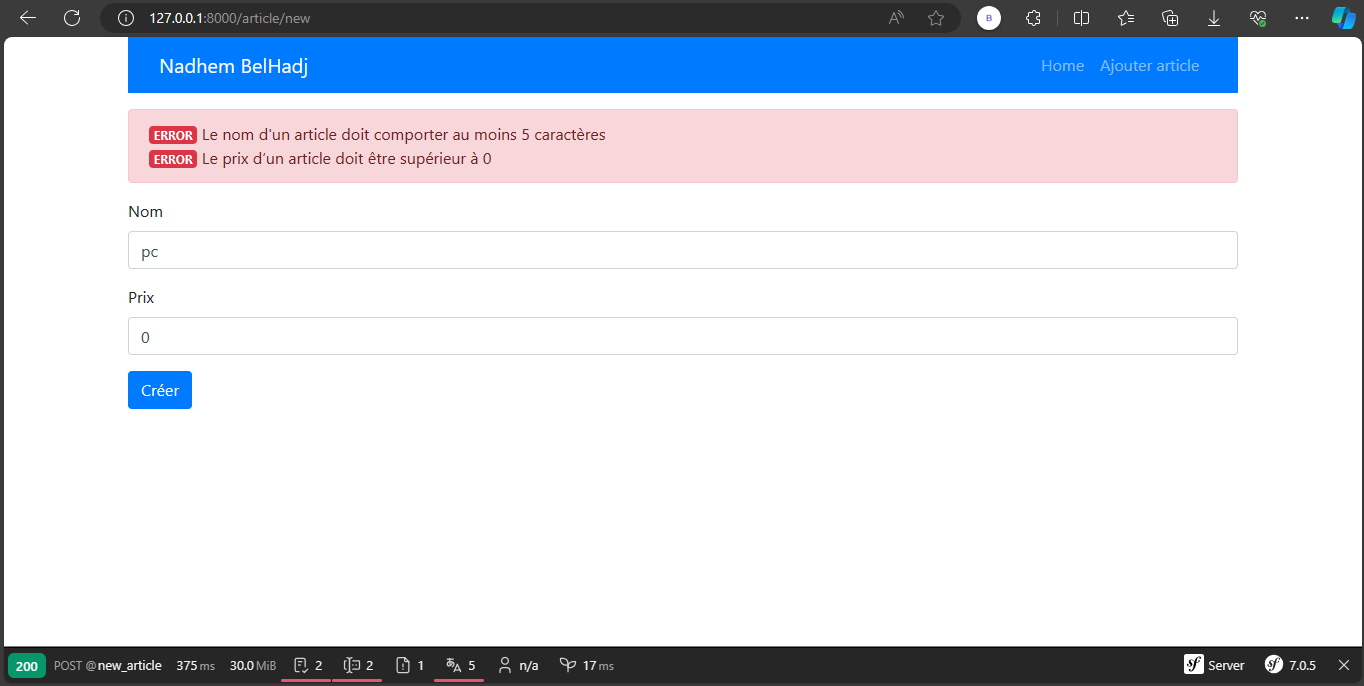
Pour ajouter des contraintes de validation sur les données entrées dans les formulaires, nous utilisons les annotations de validation fournies par Symfony au niveau de l'entité. Les contraintes requises sont : le nom d'un article doit comporter au moins 5 caractères et au maximum 50, et le prix d'un article ne doit pas être égal à 0.



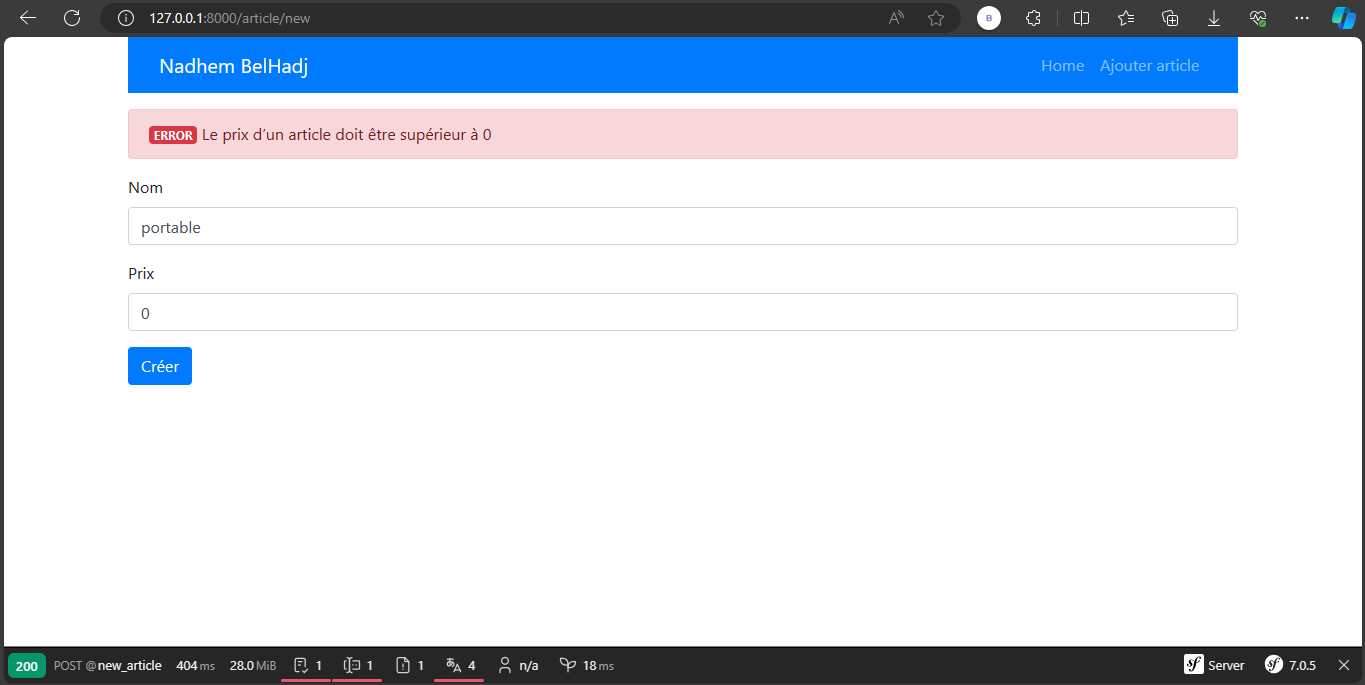


au niveau du fichier new.html.twig, on ajoute {{ form\_start(form, {'attr': {'novalidate': 'novalidate'}}) }} : **novalidate** est ajouté au formulaire avec la valeur **novalidate**. Cet attribut est utilisé pour désactiver la validation HTML5 par défaut du navigateur. Cela signifie que la validation côté client ne sera pas effectuée, permettant ainsi à Symfony d'appliquer ses propres contraintes de validation côté serveur lors de la soumission du formulaire.

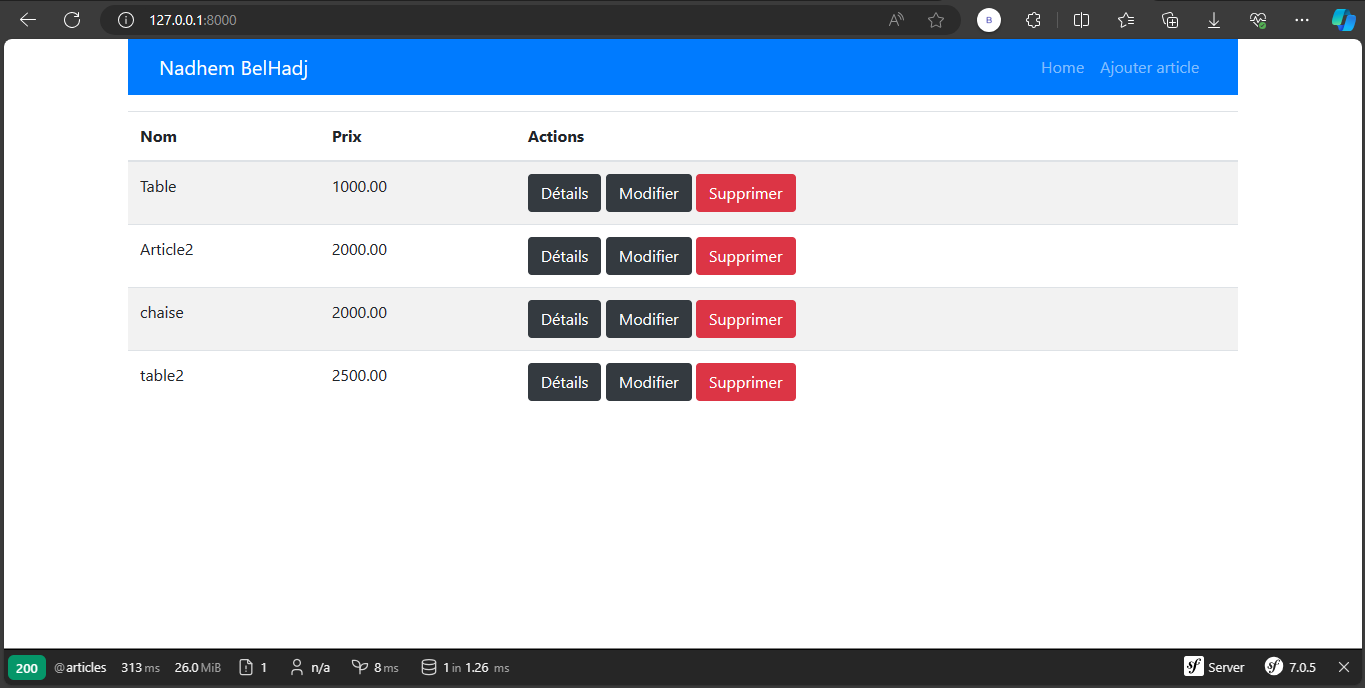


Nous allons maintenant tester notre travail en ajoutant un article avec un nom de moins de 5 caractères et un prix égal à 0. Ensuite, nous examinerons le résultat dans la capture d'écran ci-dessous.

Maintenant, nous allons vérifier uniquement le prix.



Nous allons également tester la fonction d'édition en modifiant l'article "table2"



nous vérifierons que lorsqu'on essaie de modifier le prix pour qu'il soit égal à 0, le message d'erreur s'affiche correctement.

