|  |
| --- |
| **Présentation du blog** |
| Réalisé par Céline LIEUTAUD |
| Pour le Bachelor 3 Informatique, spécialité Technologies Web. |

TABLE DES MATIERES

[CONTEXTE 2](#_Toc69715372)

[LE BLOG 2](#_Toc69715373)

[A. Technologies utilisées 2](#_Toc69715374)

[B. Modèle de données 2](#_Toc69715375)

[C. Utilisateurs 4](#_Toc69715376)

[D. Les routes 4](#_Toc69715377)

[PROBLEMES RENCONTRES 6](#_Toc69715378)

[A. Fichier .gitignore ne fonctionnant pas 6](#_Toc69715379)

[B. Problème de migrations 6](#_Toc69715380)

[CONCLUSION 8](#_Toc69715381)

# CONTEXTE

Dans le cadre du module de Développement Back-end et pour alimenter notre connaissance, nous devions réaliser un blog. Ce dernier devait répondre à un cahier des charges très détaillés sur ses fonctionnalités mais aussi sur les éléments obligatoires et/ou facultatifs qu’il devra contenir.

Pour ce faire, nous devons le produire avec Symfony, Composer et ReactJs. Ce document résume donc les fonctionnalités présentes dans le blog mais aussi les problèmes rencontrés lors de l’élaboration et du développement de ce dernier. En dernière partie du document, un retour d’expérience sera glissé à côté d’une conclusion.

# LE BLOG

## Technologies utilisées

En ce qui concerne les technologies utilisées, celles-ci se résument aux technologies suivantes :

* Le Framework Symfony, qui est un Framework open source et utilisé pour construire des applications web à haute performances.
* Composer, afin d’intégrer des paquets ou des bibliothèques externes dans le projet.
* Le Framework React, afin de s’occuper de l’application web et de ses pages
* Xampp comme serveur local afin de pouvoir gérer la base de données et la connecter à l’application web.
* Github, afin de pouvoir versionner et archiver le code en ligne

Le code total du projet se trouve dans le repository Github situé à l’adresse ci-après : <https://github.com/OphisDgn/projetBack_blog/tree/master>

Attention cependant, suite à un problème avec le .gitignore, la branche main n’est plus utilisable (voir la partie [Problème rencontrés : Fichier .gitignore ne fonctionnant pas](#_Fichier_.gitignore_ne)), ainsi il faut prendre la branche « master » où le code est donc le dernier mis à jour.

## Modèle de données

Le modèle de données du blog est le suivant :

Diagram

Description automatically generated

Figure 1 : Modèle de données du blog

La table User sert à répertorier les utilisateurs et à les différencier au travers de leur rôle contenu dans l’attribut « roles ». Ils ont chacun un nom, un prénom, une adresse email et un mot de passe afin qu’ils puissent se connecter et changer de mot de passe, commenter des articles, etc.

Les utilisateurs disposant du rôle « Admin » ou « Auteur » pourront créer des articles qui seront liés à eux via une relation ManyToOne[[1]](#footnote-1) entre la table User et la table Article. Cette dernière contiendra le titre de l’article, le contenu, le lien vers l’image et son statut (si l’article est visible ou non) mais aussi la date à laquelle l’article sera publié s’il n’est pas visible (contenu dans l’attribut « publish\_at ») et la date à laquelle il est posté s’il est direct mis en visible.

Les articles sont reliés avec une relation ManyToOne à Category, qui répertorie toutes les catégories grâce à leur nom. Il n’est possible de mettre qu’une seule catégorie à un article mais une catégorie peut avoir plusieurs articles, d’où la relation ManyToOne.

Comme pour tout blog, il est possible de rajouter des commentaires sur les articles. Ces derniers sont répertoriés dans la table Comment avec leur contenu, la date à laquelle ils ont été écrit et leur statut, qui correspond à l’approbation de l’administrateur sur le commentaire. Les commentaires ont aussi une relation ManyToOne aux utilisateurs mais aussi aux articles.

## Utilisateurs

Pour avoir accès à toutes les fonctionnalités du site, deux utilisateurs bien distincts ont été créés.

Un utilisateur simple, qui n’a aucun droit sur les articles ou les catégories mais qui peut cependant créer des commentaires sur les articles.

Et un autre utilisateur, qui sera lui l’administrateur du site. Il pourra pour sa part créer, modifier et/ou supprimer des articles mais aussi des catégories et en plus, il pourra valider ou supprimer des commentaires d’articles. Tout cela sera administrable dans le backoffice du site.

Les deux types d’utilisateurs doivent se connecter avec leur adresse mail et le mot de passe dont voici, ci-dessous les informations :

* L’utilisateur simple :
  + Adresse email : [user@ex.com](mailto:user@ex.com)
  + Mot de passe : user
* L’utilisateur administrateur :
  + Adresse email : [admin@ex.com](mailto:admin@ex.com)
  + Mot de passe : admin

## Les routes

Afin de pouvoir accéder aux différentes parties du site, plusieurs routes ont été créées. Certaines sont accessibles sans être connectés et d’autres nécessitent une connexion simple ou spécifique, à savoir qu’une connexion simple ici correspond à un utilisateur lambda et une connexion spécifique correspond à un compte Administrateur.

Voici un rapide tour des différentes routes, leurs correspondances et leurs restrictions :

|  |  |  |
| --- | --- | --- |
| Routes | Correspondance | Restrictions |
| «  / » | La page d’accueil |  |
| « /about » | La page de « A propos » |  |
| « /contact » | La page de contact |  |
| « /login » | La page de connexion |  |
| « /logout » | La page de déconnexion |  |
| « /article/{id} » | La page de détail d’un article | Où {id} est l’id de l’article ciblé |
| « /admin » | La page d’administration du site | Seuls les utilisateurs disposant du rôle admin et/ou auteur y auront accès |

# FONCTIONNALITES

Concernant les fonctionnalités présentes dans le blog, voici un résumé de chacune :

# PROBLEMES RENCONTRES

## Fichier .gitignore ne fonctionnant pas

Comme pour tout projet utilisant Git, nous avons la possibilité de renseigner dans un fichier les éléments que nous ne voulons pas pousser sur le repository Git. C’est le fichier : .gitignore.

J’avais donc utilisé ce fichier pour ne pas diffuser les dossiers de React et de Vscode (respectivement, node\_modules et .vscode) cependant le repository Git les avait quand même archivés. J’ai donc réalisé un test, en utilisant un fichier à ne pas diffuser et ça ne fonctionnait toujours pas.

J’ai donc cherché et trouvé un tutoriel[[2]](#footnote-2) pour le faire fonctionner mais j’ai dû faire une erreur lors du push car la branche main de mon repository ne comporte maintenant qu’un fichier README et un dossier vide.

Pour pallier ce problème, j’ai donc recréé un dossier en local où j’ai déplacé le code existant puis j’ai créé une nouvelle branche, nommé « Master », qui me sert maintenant comme branche principale. Par la suite, pour ne pas me mélanger, j’ai renommé la branche main en « Main\_error ».

J’ai remarqué que sur cette nouvelle branche, le fichier .gitignore fonctionne très bien, je n’ai donc pas très bien compris mais ça m’a permis de pouvoir utiliser ce fichier.

## Problème de migrations

Grâce à Doctrine, nous pouvons créer des Entités et des relations entre ces dernières afin de gérer au mieux notre application web. Les entités correspondent à des tables dans une base de données comme avec MySQL.

Ainsi, dès lors que l’on ajoute, modifie ou supprime des informations d’une entité ou carrément l’entité elle-même, on pourra réaliser une migration vers la base de données. C’est ce que j’ai fait lorsque j’ai ajouté l’entité Comment, qui regroupera tous les commentaires liés aux articles et à un utilisateur.

J’avais créé l’entité avec la commande symfony make:entity . J’ai renseigné mes attributs comme je l’avais fait pour User, Article et Category, aucune erreur n’était apparue pour ces derniers.

Par la suite j’ai réalisé la commande : php bin/console make :migrate , qui m’a servi à réaliser le fichier de migration. Et quand j’ai rentré la commande suivante : php bin/console doctrine:migrations:migrate , c’est là que cela m’a mis une erreur. Voici ci-après le screen du résultat obtenu :

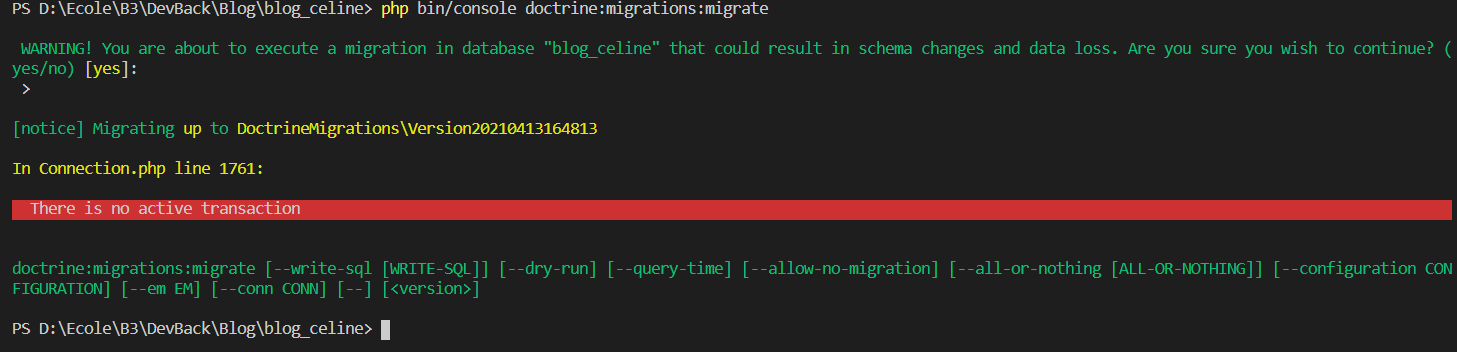


Figure 2 : Erreur pendant la migration

J’ai donc demandé de l’aide à l’intervenant du cours de Développement Back, Monsieur POUSSEL Olivier, qui m’a demandé de regarder les moteurs des tables de la base de données car le problème pourrait venir de cela et si ce n’est pas le cas, alors de réaliser des commandes pour mettre à jour composer car ça pourrait venir d’un bug de Symfony.

Après avoir regardé si les moteurs des tables de la bdd étaient bien en « innoDB » et non pas en « mylsam », ce qui était juste, j’ai donc décidé de réaliser les commandes suivantes :

* composer update
* composer dump-autoload

Et miracle, le problème s’est résolu, la migration n’a eu aucun problème. Ainsi il y a du avoir un bug de Symfony avec des ressources ou des éléments dont je n’ai pas fait attention ou pour d’autres raisons.

# CONCLUSION

1. Relation plusieurs à un. Par exemple, un article peut avoir plusieurs commentaires mais un commentaire ne peut avoir qu’un seul article. [↑](#footnote-ref-1)
2. Disponible à cette adresse : https://gist.github.com/Thithip/cfc058442e4e052aa817 [↑](#footnote-ref-2)