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

TABLE DES MATIERES

[CONTEXTE 2](#_Toc69334961)

[LE BLOG 3](#_Toc69334962)

[A. Technologies utilisées 3](#_Toc69334963)

[B. Modèle de données 3](#_Toc69334964)

[PROBLEMES RENCONTRES 5](#_Toc69334965)

[A. Fichier .gitignore ne fonctionnant pas 5](#_Toc69334966)

[B. Problème de migrations 5](#_Toc69334967)

[CONCLUSION 7](#_Toc69334968)

# 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.

## Modèle de données

Le modèle de données du blog est le suivant :

Diagram

Description automatically generated

Figure : Modèle de données du blog

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

Les utilisateurs disposant du rôle « Admin » ou « Auteur » pourront créer des articles qui seront liés à eux via une relation ManyToOne 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.

# 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. 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 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 palier à 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.

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 réalisé 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 après la commande :

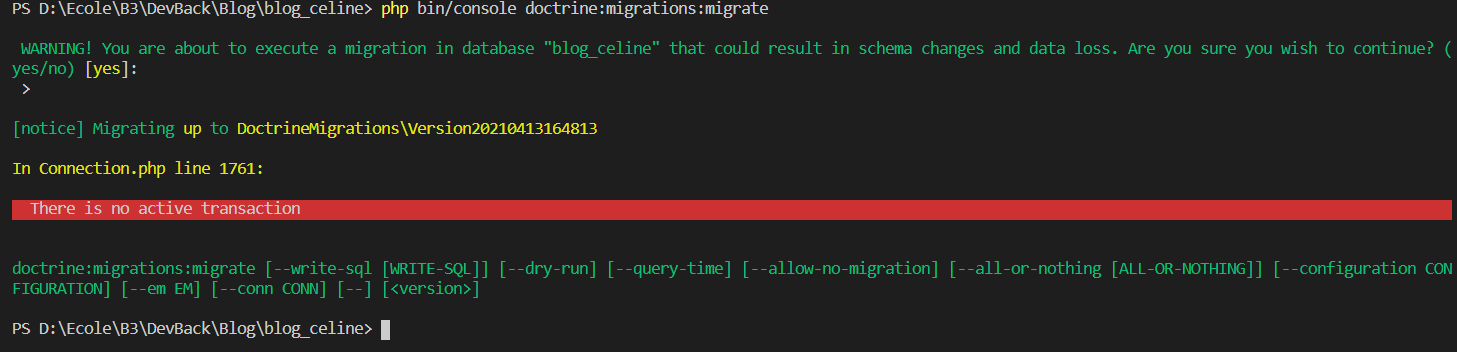


Figure : Erreur pendant la migration

# CONCLUSION