Jean Forteroche, blog d'écrivain

CAHIER DES CHARGES

Marion BATTISTA

battista.marion@gmail.com

1. Suivi des versions

Cette section met en évidence les évolutions du document, et quelles parties ont été mises à jour pour facilité la relecture.

- ➔ v.1.1, 06 mars 2017 : Présentation et spécifications fonctionnelle
- ➔ v.1.2, 15 mars 2017 : Spécifications fonctionnelles et techniques Article / Comment / User
- ➔ v.1.3, 20 mars 2017 : Ajout des diagrammes Article et Comment
- ➔ v.1.4, 27 mars 2017 : Ajout des diagrammes User et /Model

2. PRÉSENTATION

2.1 Le projet

Mr Jean Forteroche, acteur et écrivain, rédige actuellement son prochain roman : Billet Simple pour l'Alaska. Il souhaite innover et publier son roman épisode par épisode sur son propre site. Mr Forteroche refuse que son site soit développer avec un CMS de type WordPress. Il souhaite avoir son propre outil de blog, depuis lequel il pourra écrire et publier le plus simplement possible son futur roman.

2.2 Enjeux

- ➔ Créer une interface « friendly », très facile à prendre en main pour le client.
- ➔ Fidéliser le lecteur par la publication d'articles sous forme de chapitre chaque semaine

3. SPÉCIFICATIONS FONCTIONNELLES

3.1 Nom du domaine et hébergement du site internet.

Le site du client est hébergé à l'adresse : http://jeanforteroche.wubiii.com/.

Modèle	Processeur	Mémoire vive	Stockage	Prix / mois
Dedibox® SC 2016	Intel® C2350 (Avoton) 2C / 2T @1,7 Ghz	4 Go DDR3	120 Go SSD	9€ 34 TTC
Bande passante		Basic illimité 1 Gbit/sec		
Basic illimité 1 Gbit/sec		Traffic Illimité		

3.2 Environnement

L'application sera développée en PHP et utilisera une base de données MySQL. Deux interfaces sont à prévoir :

- ➔ Frontend : permet l'affichage et la lecture des billets. Pour le développement du frontend, nous utilisons la librairie de Bootstrap.
- ➔ Backend : permet l'administration des billets et de leurs commentaires.

Le client souhaite une application de blog simple en PHP et avec une base de données MySQL. Elle doit fournir une interface frontend (lecture des billets) et une interface backend (administration des billets pour l'écriture). On doit y retrouver tous les éléments d'un CRUD :

- ➔ Create : création de billets
- ➔ Read : lecture de billets
- ➔ Update : mise à jour de billets
- ➔ Delete : suppression de billets

Chaque billet doit permettre l'ajout de commentaires, qui pourront être modérés dans l'interface d'administration au besoin. Un commentaire peut être une réponse à un autre commentaire : dans ce cas, il est légèrement décalé vers la droite pour montrer l'arborescence des commentaires. Il ne peut pas y avoir plus de 3 sous-niveaux de commentaires. Les lecteurs doivent pouvoir "signaler" les commentaires pour que ceux-ci remontent plus facilement dans l'interface d'administration pour être modérés.

L'interface d'administration sera protégée par mot de passe. La rédaction de billets se fera dans une interface WYSIWYG basée sur TinyMCE, pour que Jean n'ait pas besoin de rédiger son histoire en HTML (on comprend qu'il n'ait pas très envie !).

Le code sera construit sur une architecture MVC. Vous développerez autant que possible en orienté objet. Au minimum, le modèle doit être construit sous forme d'objet.

4. Articles

Depuis le dashboard l'auteur Jean Forteroche doit pouvoir :

- ➔ Créer un article
- ➔ Éditer un article
- ➔ Lire un article
- ➔ Supprimer un article

Sur la page d'accueil du dashboard, un bouton « Ajouter un article » permet de créer un nouvel article, et l'auteur retrouve la liste des articles publiés sur la page http://jeanforteroche.wubiii.com/dashboard/articles. Depuis cette même page il peut également créer, éditer ou supprimer un article. La technologie TinyMCE est utilisée, elle propose une interface pour l'éditeur de texte. L'auteur s'en servira pour créer et éditer ses articles.

Une fois l'article enregistré il est ajouté dans la base de données et est publié sur le site. Le lecteur pourra consulter la liste des articles sur http://jeanforteroche.wubiii.com/articles. Il pourra lire un article en particulier sur http://jeanforteroche.wubiii.com/article?articleId=3.

4.1 Base de données

La table d'un article est le suivant :

```
Id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
title VARCHAR(255) NOT NULL,
content TEXT NOT NULL,
author VARCHAR(40) NOT NULL,
date DATETIME NOT NULL,
PRIMARY KEY (id))';
```

4.2 Création

A chaque création d'un nouvel article, le constructeur vérifie que la table existe bien dans la base de donnée, dans le cas contraire il la créé. Il initialise également le champs « date » à chaque nouvelle création.

4.3 Enregistrement

Une seule fonction pour l'enregistrement :

- Si l'id de l'article est égal à 0, alors un nouvel article est enregistré dans la base de données
- Si l'id de l'article est différent de 0, alors l'article égal à l'id est modifié et enregistré ;

4.4 Suppression

Connexion et suppression de l'article retrouvé grâce à son id.

4.5 Lecture des articles

La méthode findAll() sélectionne dans la base de données tous les articles et les renvoie sous forme d'un tableau. Permet à l'auteur de consulter la liste de ses articles depuis le dashboard ('/dashboard/articles') et le site ('/articles').

La méthode findById() sélectionne dans la base de données l'article correspondant à l'id passé en paramètre de la méthode.

La fonction findByLastOne() renvoie le dernier article enregistré par l'auteur. Cet fonction permet d'afficher sur la page d'accueil du site un extrait du dernier article enregistré ainsi que son titre et sa date de publication.

4.5 Diagramme du modèle Article

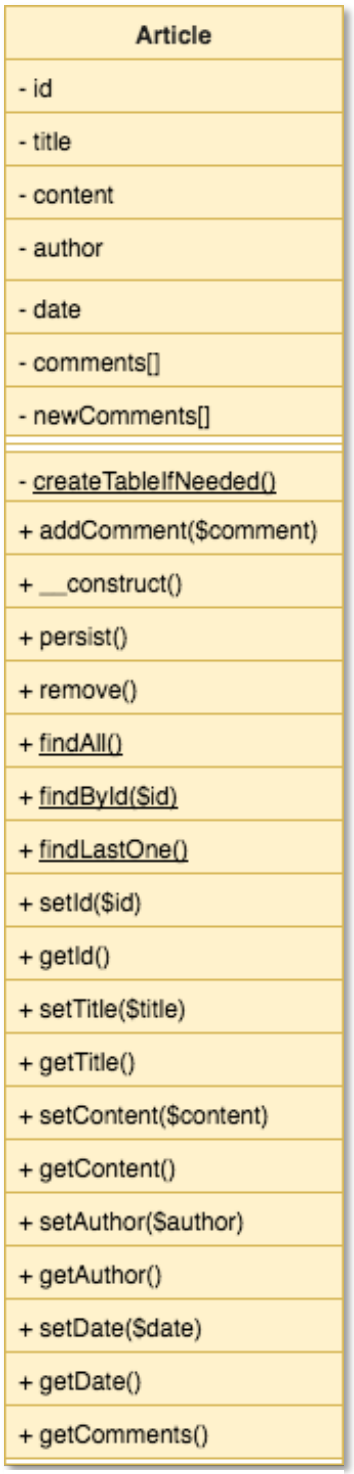


Figure 1. Article Model

5. Commentaires

Le formulaire situé en dessous de chaque article permet à l'utilisateur de poster un commentaire. Chaque commentaire dispose de quatre boutons :

- Modifier
- Supprimer
- Répondre
- Signaler : Retire le commentaire du site et le fait apparaître dans le dashboard, pour que l'auteur puisse le gérer.

L'auteur ne souhaite pas avoir plus de trois niveaux de commentaires, le troisième niveau de commentaire ne possède donc pas de bouton « Répondre ».

5.1 Base de données

La table d'un commentaire est le suivant :

```
id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
title VARCHAR(255) NOT NULL,
content TEXT NOT NULL,
pseudo VARCHAR(40) NOT NULL,
date DATETIME NOT NULL,
articleId SMALLINT UNSIGNED,
commentId SMALLINT UNSIGNED,
level SMALLINT UNSIGNED,
alert TINYINT UNSIGNED,
PRIMARY KEY (id))'
```

5.2 Création

A chaque création d'un nouveau commentaire, le constructeur vérifie que la table mb_comment existe bien dans la base de donnée, dans le cas contraire il la créé. Il initialise également le champs « date » à chaque nouvelle création. Le constructeur initialise l'id du commentaire parent (commentId) à 0 et le niveau du commentaire (level) à 1.

5.3 Enregistrement

Même fonctionnement que pour les articles, une seule fonction pour l'enregistrement :

- Si l'id du commentaire est égal à 0, alors un nouveau commentaire est enregistré dans la base de données
- Si l'id du commentaire est différent de 0, alors le commentaire égal à l'id est modifié.

Lors de l'enregistrement du commentaire, on enregistre l'id de l'article qui vient d'être commenté dans le champs articleId afin de pouvoir retrouver tous les commentaires associés à un article.

Pour enregistrer les différents niveaux de commentaires,

5.4 Suppression

Connexion et suppression du commentaire retrouvé grâce à son id. Si un commentaire de niveau 1, qui comporte des sous commentaires, est supprimé tous les commentaires enfants sont également supprimés.

5.5 Lecture des commentaires

La méthode findAll() sélectionne dans la base de données tous les commentaires et les renvoie sous forme d'un tableau. Permet la consultation de la liste des commentaires associés à un article depuis le site (/article?articleId=$articleId).

La méthode findById() sélectionne dans la base de données le commentaire correspondant à l'id passé en paramètre de la méthode.

La méthode findByAllByArticle() sélectionne dans la base de données les commentaires associés à un article retrouvé par son id.

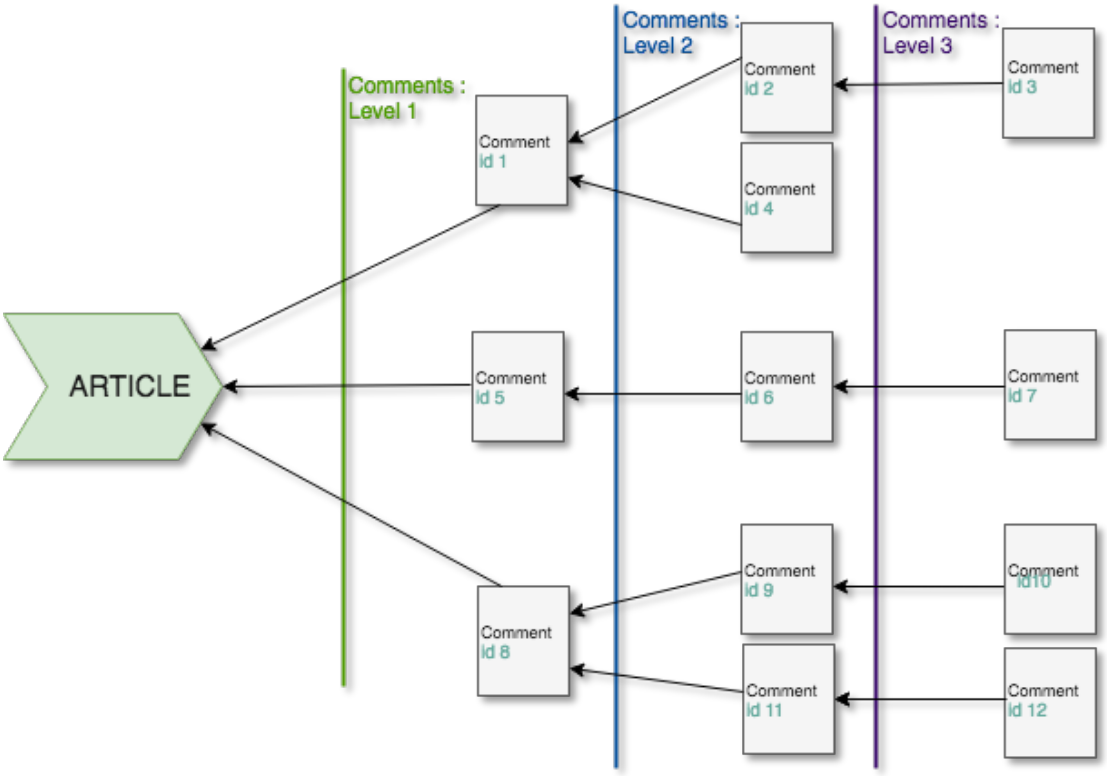
La méthode findByAlert() sélectionne dans la base de données tous les commentaires dont le champs alert est à 1. Permet à l'auteur de consulter depuis son dashboard ('/dashboard/comments') la liste des commentaires qui ont été signalés par les utilisateurs.

La méthode findAllByComment() permet un commentaire en particulier avec ses commentaires enfants. Le résultat est retourné sous forme de tableau.

5.6 Diagramme du modèle Comment

Figure 2. Comment model

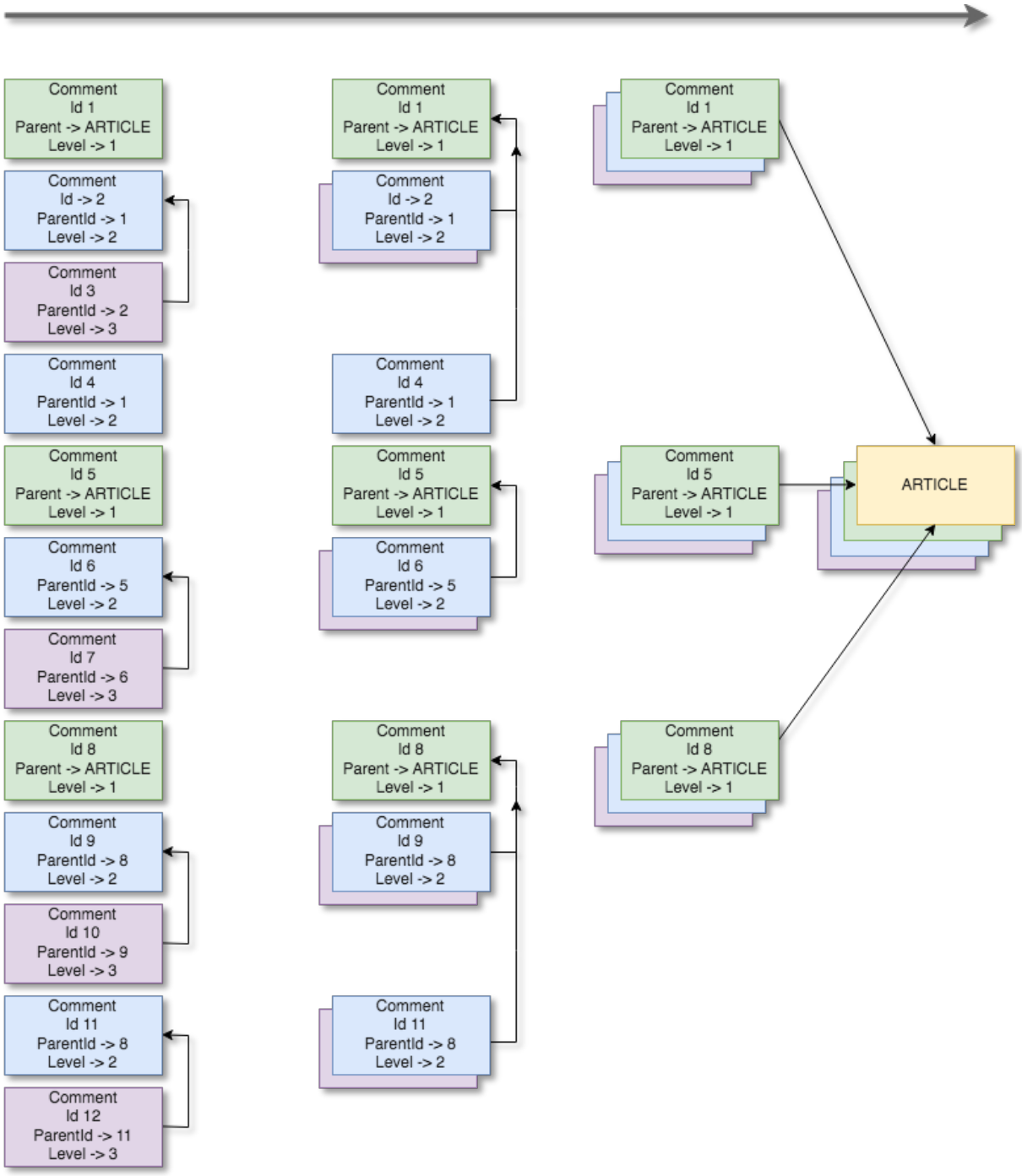
5.6 Diagramme d'enregistrement des commentaires



C'est un algorithme en deux passes :

1. On parcourt la liste de tous les commentaires et pour chaque élément de niveau trois on le relie à son parent de niveau deux et on le supprime de la liste
2. On parcourt une deuxième fois la liste. Pour chaque élément de niveau deux on le relie à son parent de niveau 1 et on le supprime de la liste.

On démarre d'une représentation sous forme de liste pour arriver à une représentation sous forme d'arbres.
Cet algorithme est extensible à souhait, il peut s'appliquer à des niveaux de commentaires 4, 5,6… en appliquant la même méthode.


Comment
Id 1
Parent -> ARTICLE
Level -> 1
Comment
Id -> 2
ParentId -> 1
Level -> 2
Comment
Id 3
ParentId -> 2
Level -> 3
Comment
Id 4
ParentId -> 1
Level -> 2
Comment
Id 5
Parent -> ARTICLE
Level -> 1
Comment
Id 6
ParentId -> 5
Level -> 2
Comment
Id 7
ParentId -> 6
Level -> 3
Comment
Id 8
Parent -> ARTICLE
Level -> 1
Comment
Id 9
ParentId -> 8
Level -> 2
Comment
Id 10
ParentId -> 9
Level -> 3
Comment
Id 11
ParentId -> 8
Level -> 2
Comment
Id 12
ParentId -> 11
Level -> 3
ARTICLE

6. Gestion des utilisateurs

Pour gérer la publication des commentaires associés aux articles, différents rôles ont été créés :

	Admin	Connected	Anonymous
Dashboard	X		
Rédaction de commentaires	X	X	
Lecture des articles	X	X	X

6.1 Base de données

La table d'un utilisateur est le suivant :

```
'CREATE TABLE IF NOT EXISTS mb_user (
    id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
    username VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL,
    password VARCHAR(40) NOT NULL,
    token VARCHAR(64),
    role SMALLINT UNSIGNED,
    locked SMALLINT UNSIGNED,
    PRIMARY KEY (id))'
```

6.2 Création d'un nouvel utilisateur

Lorsque l'utilisateur souhaite s'enregistrer sur le site, il clique sur l'onglet « Enregistrez-vous ». Le formulaire d'enregistrement est composé de trois champs :

- Un pseudo,
- Une adresse mail,
- Un mot de passe

Lorsque l'utilisateur valide le formulaire, un email de confirmation est envoyé à l'utilisateur qui devra cliquer sur le lien contenu dans le mail pour valider son inscription.

6.3 Connexion

Si l'utilisateur est déjà enregistré sur le site, il peut se connecter à tout moment en cliquant sur l'onglet « Connectez-vous ».

6.4 Gestion des utilisateurs

L'auteur peut gérer tous ses utilisateurs depuis le dashboard ('/dashboard/users') :

- Suppression du compte utilisateur,
- Blocage du compte utilisateur,
- Attribution d'un niveau de permission

La méthode findAll() sélectionne dans la base de données tous les utilisateurs et les renvoie sous forme d'un tableau. Permet la consultation de la liste des utilisateurs depuis le site (/dashboard/users).

La méthode findById() sélectionne dans la base de données l'utilisateur correspondant à l'id passé en paramètre de la méthode.

La méthode findByUsername() sélectionne dans la base de données l'utilisateur passé en paramètre dans la méthode.

La méthode findByUsernameOrEmail() sélectionne dans la base de données l'utilisateur ou l'email passé en paramètre dans la méthode.

La méthode lockToggle() permet de verrouiller le compte d'un utilisateur, il continue d'accéder au site mais ne peut plus laisser de commentaires .

6.5 Diagramme du modèle User

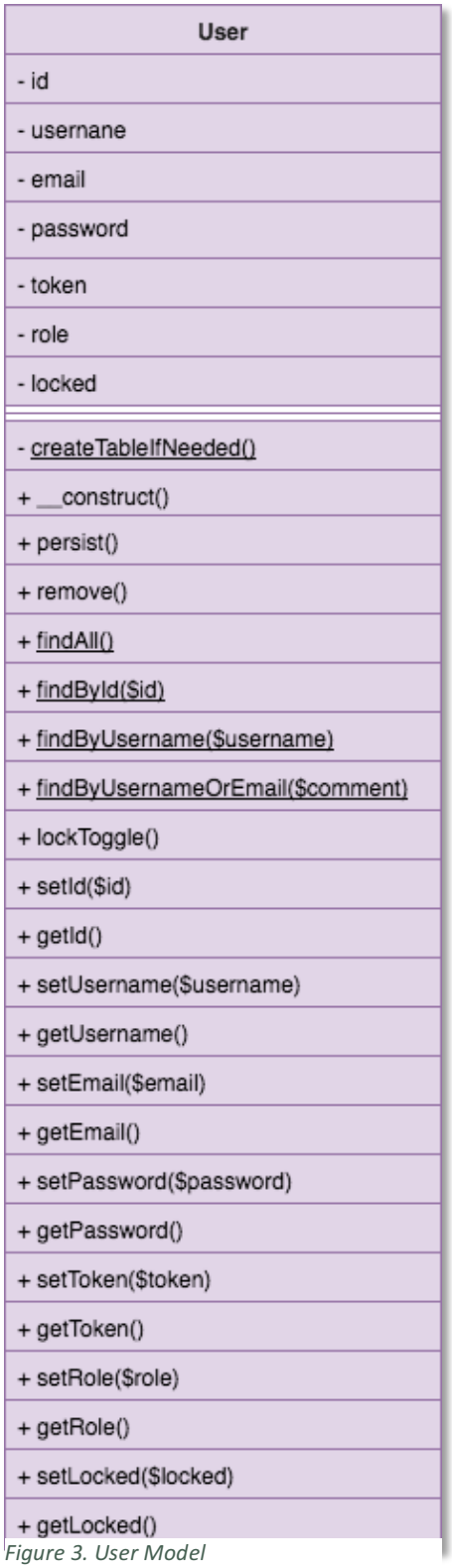


Figure 3. User Model

7. Diagramme des classes de « /Model »