haute école
neuchâtel berne jura
ingénierie
www.he-arc.ch

Documentation

Auteurs : Nicolas Huguenin, Vincent Déruaz, Kilian Brandt

Cours : Développement Web, HE-ARC

Encadrement :David Grünenwald

Dernière modification : 08.01.15

Table des matières

1. Introduction

Ce document est une petite documentation élémentaire qui décrit les aspects important de l'application *ArcNotes.*

ArcNotes est une application web développée avec le *framework Laravel* version 4 qui permet aux utilisateurs d'écrire et de partager des notes de cours avec leurs camarades.

2. Technologies

Les technologies utilisées au sein de ce projet sont

- PHP avec le *framework Laravel*
- MYSQL pour la base de données
- HTML 5/ CSS 3
- Javascript (avec jQuery)

2. Modèle

Le modèle constitue l'ensemble des informations que l'application traite, dans le cas d'*ArcNotes* le modèle correspond à la base de données. L'ensemble des modèles (tables) sont interfacées en PHP grâce à l'*ORM* du *framework Laravel* qui s'appelle *Eloquent.*

Vous trouverez le modèle de la base de donnée d'*ArcNotes* en annexes de ce document et pour de plus amples informations quand à *Eloquent* rendez-vous ici http://laravel.com/docs/4.2/eloquent.

3. Controllers

Les contrôleurs constituent le cœur de l'application, ce sont eux qui effectuent les traitements avant que le résultats ne soient transmis à la vue de l'utilisateur.

Ce chapitre contient une brève description des différents contrôleurs de l'application *ArcNotes* qui décrit leurs rôles principaux.

3.1. ClassController

Ce contrôleur se charge d'interfacer toutes les actions qui nécessitent d'interagir avec les classes.

Ce contrôleur assure les fonctions suivantes :

- Création de la vue de création d'une classe ;
- Création de la vue d'affichage des informations d'une classe ;
- Création de la vue qui liste les classes publiques ;
- Création de la vue qui liste les classes dont un utilisateur est membre ;
- Gestion de l'insertion d'une classe dans la BDD ;
- Modification des informations d'une classe ;

- Invitation d'un nouveau membre au sein d'une classe ;
- Accepter ou refuser l'adhésion d'un membre à une classe ;
- Suppression d'un cours qui appartient à une classe ;
- Retirer un membre d'une classe ;
- Modification des droits attribués à un membre d'une classe ;
- Modification de la visibilité d'une classe (publique ou privée) ;
- Suppression d'une classe ;
- Désinscription d'un membre à une classe ;
- Gère l'adhésion d'un membre à une classe.
- Rechercher au sein des classes publiques.

3.2. CourseController

Ce contrôleur effectue les traitements liés aux cours qu'une classe peut contenir.

Il assure les rôles suivants :

- Création de la vue pour créer un cours ;
- Création de la vue de modification d'un cours ;
- Création de la vue d'affichage d'un cours ;
- Traiter les requêtes d'ajout d'un cours et insertion dans la base de données ;
- Modification d'un cours ;
- Suppression d'un cours ;
- Rechercher au sein des cours appartenant à des classes publiques.

3.3. NoteController

Ce contrôleur permet la gestion des notes d'un cours. Qu'elles soient manuscrites ou sous forme de fichier (scanné ou *pdf* par exemple).

Ce contrôleur assure les rôles suivants :

- Création de la vue (formulaire) d'écriture d'une note manuscrite ;
- Création de la vue d'édition d'une note manuscrite ;
- Gérer l'insertion d'une nouvelle note manuscrite dans la BDD ;
- Modification d'une note manuscrite soit par requête POST standard ou alors avec requête AJAX et réponses en JSON ;
- Suppression d'une note manuscrite ;
- Création de la vue pour l'*upload* d'un fichier de note ;
- *Upload* d'un fichier de note ;
- Download d'un fichier de note ;
- Suppression d'un fichier de notes (physiquement y compris) ;

- Création de la vue qui permet de lire une note depuis le lien partagé *;
- *Download* d'un fichier de note en passant par le lien partagé *.

* Lorsqu'une note est partagée, aucuns droits sur la lecture de celui-ci n'est vérifié.

3.4. SchoolController

Ce contrôleur permet la gestion des écoles.

Il assure les rôles suivants.

- Création de la vue d'ajout d'une nouvelle école ;
- Insertion d'une nouvelle école.

Les écoles ne pouvant pas être supprimées par son créateur dans un but de réutilisation pour les classes ayant une école commune. Lorsqu'un utilisateur crée une classe, il sélectionne parmi les écoles existantes ou en crée une nouvelle.

3.5. UserController

Ce contrôleur permet la gestion des utilisateurs.

Il assure les rôles suivants :

- Création de la vue d'enregistrement d'un nouvel utilisateur à *ArcNotes* ;
- Ajouter un nouvel utilisateur dans la BDD et gérer l'envoi du mail d'activation du compte et l'activation d'un compte ;
- Connexion d'un utilisateur à *ArcNotes* ;
- Déconnexion d'un utilisateur ;
- Changement du mot-de-passe.

4. Vues

Les vues permettent d'afficher les informations des modèles à l'utilisateur une fois traitées par les contrôleurs.

Pour *ArcNotes* l'ensemble des vues ont été développée à l'aide du moteur de *templates Blade,* propre au *framework Laravel*. Ce moteur de *templates* permet d'éviter au maximum la duplication de code en plus de simplifier grandement l'écriture de code métier au sein du code HTML et de garantir un échappement des données utilisateur plus facile.

Toutes les vues du projet sont basées sur la vue views/layout/default.blade.php qui se trouve dans le dossier app/ de l'arborescence.

Cette vue par défaut comprend :

- Un header :
  - Image de titre/retour à l’accueil
  - Menus

    - Barre de recherche
- Un dock à gauche :
    - Profil ou login
    - Liste des classes et des cours dont l'utilisateur courant est membre
- Le contenu de la page :
    - Titre
    - Corps
    - Pied de page
- Une section (vide par défaut) qui est utilisée pour l'ajout de scripts.

Les sections titre (title) et corps (body) sont à remplir par les vues qui héritent de default. La section pied de page (footer) est à redéfinir si on désire mettre du contenu dans une fine barre horizontale située en bas de la page (typiquement pour la pagination).

La section réservée aux scripts (page-scripts) peut être peuplée de scripts si besoin. Le contenu de la section doit être entouré par des balises, car celle-ci est vide par défaut.

Pour plus d'informations concernant le moteur de *templates Blade* rendez-vous ici http://laravel.com/docs/4.2/templates

5. Système de droits / visibilité

ArcNotes dispose d'un système de droits permettant de limiter l'accès aux fichiers de notes aux utilisateurs.

5.1 Classe privée contre classe publique

Une classe peut être publique c'est à dire qu'on la trouve depuis le menu des classes publiques même sans être connecté, et que les cours qu'elle contient peuvent sortir dans les résultats de la recherche.

Si la classe est privée alors elle est invisible aux personnes qui ne connaissent pas son existence, et son contenu est introuvable depuis la recherche.

La visibilité d'une classe peut être modifiée à tout moment depuis le *manager (*/classes/owned).

5.2 Rejoindre une classe

Pour rejoindre une classe publique, il faut faire une demande en un clic depuis l'interface d'affichage des classes. Cette demande doit être acceptée par le créateur de la classe.

Pour rejoindre une classe privée il faut être invité à l'aide de l'adresse e-mail utilisée lors de l'inscription par le créateur de la classe

5.3 Droits d'ajout/édition/création au sein d'une classe

Lorsqu'un utilisateur est membre d'une classe celui-ci peut disposer de plusieurs droits au sein de celle-ci.

Les droits déterminent l'appartenance d'un utilisateur à une classe et sont codés sur 4 bits afin de pouvoir extraire facilement les 4 différents droits avec des masques et une

opération logique AND.

Les 4 droits existants sont les suivants :

15 (1111) => propriétaire : Le propriétaire est le seul à pouvoir supprimer des membres et accepter/refuser les demandes d'adhésion si la classe est publique. Il est le seul à pouvoir supprimer la classe. Bien sûr il dispose des droits de lecture/création/édition expliqués ci-dessous.

8 (1000) => lecture : Le droit de lecture permet à un utilisateur d'ouvrir les cours d'une classe et de lire le contenu des notes ainsi que de télécharger les fichiers de notes.

4 (0100) => édition : Le droit d'édition permet à un utilisateur de modifier le contenu des notes mais ils ne peux ni supprimer ni créer de nouvelles notes.

2 (0010) => création : Le droit de création permet à l'utilisateur de créer aussi bien des cours que des notes écrites et d'*uploader* des fichiers de notes au sein de la classe.

le **droit 0 (0000)** correspond à une demande d'adhésion en attente.

Ce système permet d'extraire facilement les droits d'un utilisateur au sein d'une classe, de plus cette table associative crée le lien entre la classe et l'utilisateur.

Les droits peuvent être gérés par le propriétaire pour chaque membre de manière individuelle.

6. Utiliser le *plugin* de *toasts* dans le projet ArcNotes

Pour faire apparaître de petits messages d'erreurs/d'informations/etc. afin d'informer l'utilisateur, *ArcNotes* dispose d'un système de popups temporaires (à la manière des *toasts* sous *Android)* qui utilise les sessions.

Pour s'en servir c'est très simple. Le layout blade de base va générer le code *javascript* qui permet d'afficher le message lorsque la session '*toast'* n'est pas vide puis il la videra. Dans le code si vous souhaitez faire apparaître un de ces messages ajoutez une session de la manière suivante.

```
Session::put('toast',array('[type]', "Text Message");
```

Il y a quatre [types] possibles :

1. success: Affiche un message avec un icône de succès ;
2. error : Affiche un message d'erreur ;
3. warning : affiche un message d'attention ;
4. notice : affiche un message informatif ;

Ainsi au prochain chargement d'une page de l'application *ArcNotes* le message est affiché.

Voici un petit exemple pour terminer :

```
Session::put('toast',array('success', "Vous êtes maintenant connecté');
```

L'exemple ci-dessus affiche un popup de confirmation (succès) avec le message « Vous êtes maintenant connecté » durant un court instant.

7. Configurer le serveur pour l'*upload* des fichiers

Comme *ArcNotes* dispose d'un système d'*upload* de fichiers, le serveur a besoin de quelques petites modifications de la configuration standard pour assurer le bon déroulement des opérations *d'upload*.

7.1 Serveur Apache

En considérant que votre serveur web soit Apache.

- Chargement du module *mod_mime.so*, normalement il s'agit de décommenter la ligne `LoadModule mime_module modules/mod_mime.so` dans le fichier *httpd.conf.* Il faut télécharger ce module s'il est manquant

Ce module permet de faire l'association entre les extensions des fichiers et les données qu'ils contiennent.

7.2 Configuration PHP

Il faut configurer PHP pour qu'il accepte des tailles de fichiers en *upload* plus grandes que 2Mo par défaut.

- Modifications des configurations dans le fichier *php.ini*
  - file_uploads = on
  - upload_max_filesize = 20M

ArcNotes accepte les fichiers jusqu'à 10'000Ko donc pour être certain on indique à *PHP* d'en accepter le double.

Si des problèmes persistent lors de l'upload vérifiez les droits sur le dossier */public/uploads/.*

8. Utilisation des routes et des filtres

Laravel propose un système de routage qui permet de rediriger les requêtes sur les bonnes méthodes des contrôleurs. Les filtres quant à eux permettent d'ajouter un pré-traitement/post-traitement avant en plus du traitement effectué par le contrôleur. Les filtres s'appliquent sur les routes.

ArcNotes utilise ces concepts. Ainsi vous trouverez dans le fichier *app/routes.php* les routes et les appels aux filtres.

Les principales utilisations des filtres sont la vérification que l'utilisateur est connecté ainsi que la vérification *CSRF* lors de l'envoi de formulaires.

9 Conclusion

ArcNotes est une application web qui exploite la puissance du *framework Laravel*. L'utilisation du *framework* Laravel nous a permis de faire abstraction de l'interfaçage de la base de données avec le code métier, d'assurer la bonne distribution des requêtes en utilisant la puissance des routes, de séparer d'une manière logique et structurée le code métier dans les différents contrôleurs ainsi que de simplifier l'écriture des vues grâce au moteur de *templates.*

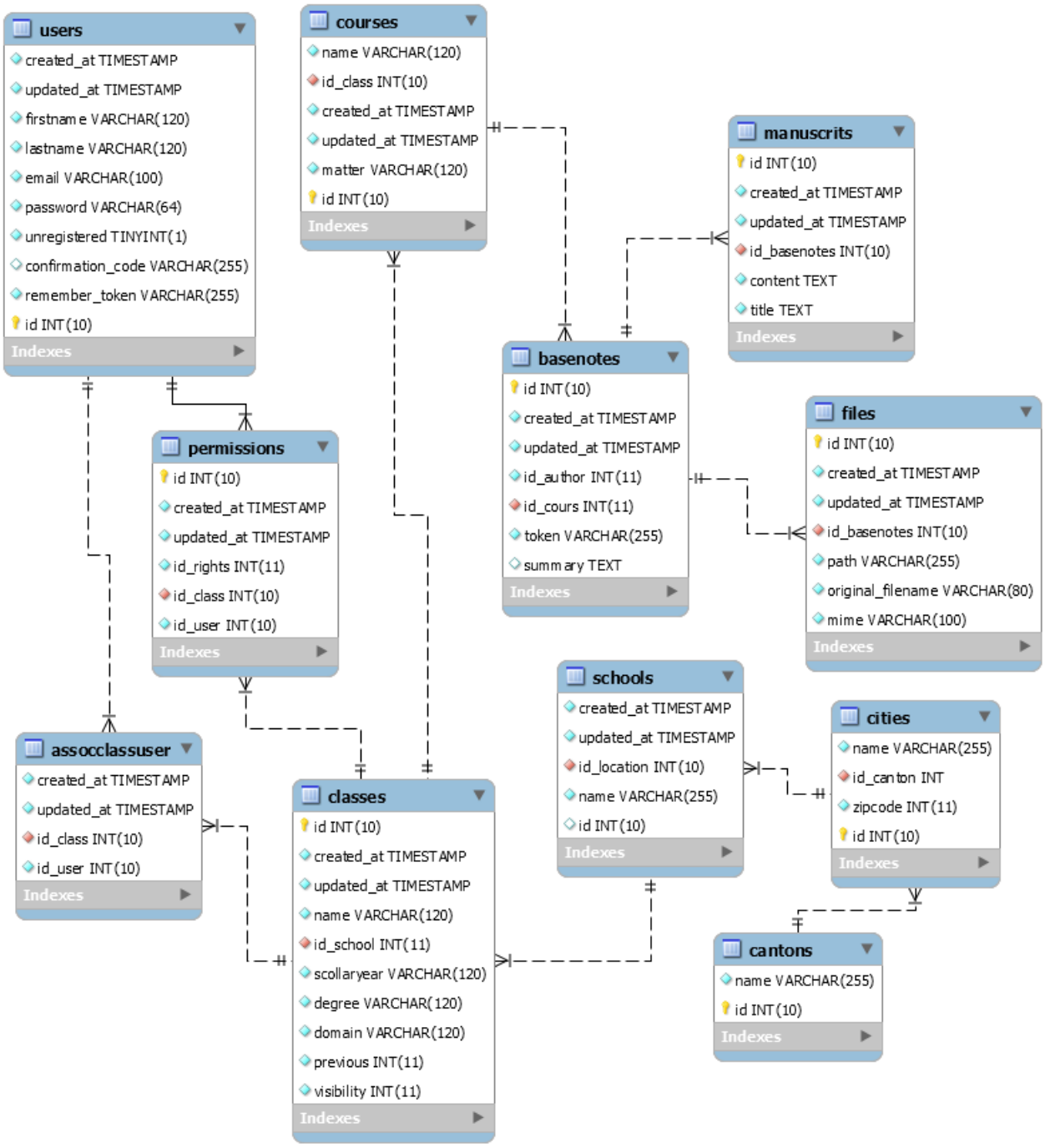
L'application permet de gérer l'écriture et le partage de notes aussi bien écrites que sous forme de fichier au sein d'une hiérarchie d'écoles, de classes et de cours en garantissant un certain contrôle grâce à un système de droits simple.

L'héritage des vues basées sur un *layout* par défaut permet la création de nouvelles interface en toute simplicité tandis que le *plugin toast* permet d'afficher aisément et de manière propre des messages à l'utilisateur sans alourdir le contenu des pages.

Pour rendre l'édition des notes plus aisée et en guise d'apprentissage nous avons développé en *AJAX* avec *jQuery* un système de sauvegarde des notes écrites en exploitant les réponses *JSON* de *Laravel*.

L'application nécessite néanmoins quelques configurations de PHP et d'Apache pour être entièrement fonctionnelle.

Annexes


users
created_at TIMESTAMP
updated_at TIMESTAMP
firstname VARCHAR(120)
lastname VARCHAR(120)
email VARCHAR(100)
password VARCHAR(64)
unregistered TINYINT(1)
confirmation_code VARCHAR(255)
remember_token VARCHAR(255)
id INT(10)
Indexes
courses
name VARCHAR(120)
id_class INT(10)
created_at TIMESTAMP
updated_at TIMESTAMP
matter VARCHAR(120)
id INT(10)
Indexes
manuscrits
id INT(10)
created_at TIMESTAMP
updated_at TIMESTAMP
id_basenotes INT(10)
content TEXT
title TEXT
Indexes
basenotes
id INT(10)
created_at TIMESTAMP
updated_at TIMESTAMP
id_author INT(11)
id_cours INT(11)
token VARCHAR(255)
summary TEXT
Indexes
files
id INT(10)
created_at TIMESTAMP
updated_at TIMESTAMP
id_basenotes INT(10)
path VARCHAR(255)
original_filename VARCHAR(80)
mime VARCHAR(100)
Indexes
permissions
id INT(10)
created_at TIMESTAMP
updated_at TIMESTAMP
id_rights INT(11)
id_class INT(10)
id_user INT(10)
Indexes
schools
created_at TIMESTAMP
updated_at TIMESTAMP
id_location INT(10)
name VARCHAR(255)
id INT(10)
Indexes
cities
name VARCHAR(255)
id_canton INT
zipcode INT(11)
id INT(10)
Indexes
assocclassuser
created_at TIMESTAMP
updated_at TIMESTAMP
id_class INT(10)
id_user INT(10)
Indexes
classes
id INT(10)
created_at TIMESTAMP
updated_at TIMESTAMP
name VARCHAR(120)
id_school INT(11)
scollaryear VARCHAR(120)
degree VARCHAR(120)
domain VARCHAR(120)
previous INT(11)
visibility INT(11)
Indexes
cantons
name VARCHAR(255)
id INT(10)
Indexes