2021/2022

UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

FORMEDITOR

Projet INFO0303

Résumé

Premier projet avec le framework Laravel Version 8.x

SAMI DRIOUCHE & RAHIM HAYAT

S3F3B & S3F5B

Intervenant : Cyril RABAT

Sommaire

Introduction

Le site FormEditor est un site qui vous permettra de créer des formulaires en ligne et de les poster de façon permanente, pour une meilleure gestion de tout ce qui est paperasses administratives.

Vous pourrez former des groupes pour que vos correspondant puisse remplir les champs qui leur sont attribué de façon rapide et simple.

Les utilisateurs peuvent créer autant de formulaire qu'ils le souhaitent sans contraintes ni limites.

FormEditor est un site accessible par tous. Facile d'utilisation, chaque utilisateur possède des rôles.

Utile à la fois pour les professionnels et le monde du travail comme pour étudiant ou simple particulier.

Prérequis

Voici quelques informations nécessaires à savoir avant d'utiliser FormEditor :

Le projet est installé sur GitLab ainsi que la VM de DRIOUCHE Sami (drio0004) et HAYAT Rahim (haya0002).

- Lien de téléchargement GitLab :

https://gitlab-mmi.univ-reims.fr/RSFE/formeditor.git

- Accès à FormEditor :

http://10.5.2.25/~drio0004/FormEditor/public/

http://10.5.2.25/~haya0002/FormEditor/public/

- Accès à phpmyadmin :

http://10.5.2.25/phpmyadmin/index.php

- Accès à la base de données (Sami DRIOUCHE) :

DB_DATABASE=DRIOU1DB

DB_USERNAME=DRIOU1

DB_PASSWORD=%R23bG2j|

- Accès à la base de données (Rahim HAYAT):

DB_DATABASE=HAYAT1DB

DB_USERNAME=HAYAT1

DB_PASSWORD=!62p=LYe4

Pour récupérer le projet laravel du site à l'aide du lien GitLab :

-depuis la VM veuillez-vous situez dans le répertoire

-il vous suffit d'ouvrir un terminal, taper les commandes :

git clone https://gitlab-mmi.univ-reims.fr/RSFE/formeditor.git FormEditor

cd FormEditor/

composer install

Utilisez la commande cp si vous êtes sous linux sinon copy sous Windows :

cp .env.example .env

php artisan key:generate

Si vous avez cloné le projet sur la VM, il faut permettre l'accès à ces 2 répertoires à l'aide de la commande suivante :

chmod -R 777 storage/ bootstrap/cache

La première fois il vous faudra vous connecter avec le compte admin qui vous permettra par la suite de créer des comptes utilisateur (professionnel, particulier et étudiant) et pouvoir exploiter pleinement le site :

- Compte admin (Chuck NORRIS) :

Email : chuck.norris@toto.fr

Mot de passe : totototo

Puis quelque compte par default pour chacun des rôles :

- Compte modérateur :

Email : moderateur@formeditor.com

Mot de passe : password

- Compte étudiant :

Email : etudiant@formeditor.com

Mot de passe : password

- Compte professionnel :

Email : professionnel@formeditor.com

Mot de passe : password

- Compte particulier :

Email : particulier@formeditor.com

Mot de passe : password

Easter eggs 😊 :

- Essayez d'accéder à une page qui n'existe pas.
- Essayez d'accéder à un formulaire qui n'existe pas dans la Barre de recherche.

Cahier des charges

Avant de commencer le développement du site web, nous avons dû réfléchir à la modélisation du site. En effet nous avons réfléchis au cas d'utilisation, diagramme de navigation et ainsi qu'au modèle conceptuel de donnée (MCD) et modèle logique de donnée (MLD).

Pour modéliser nos données nous avons choisi d'utiliser les logiciels darw.io ainsi que looping.

Cas d'utilisation

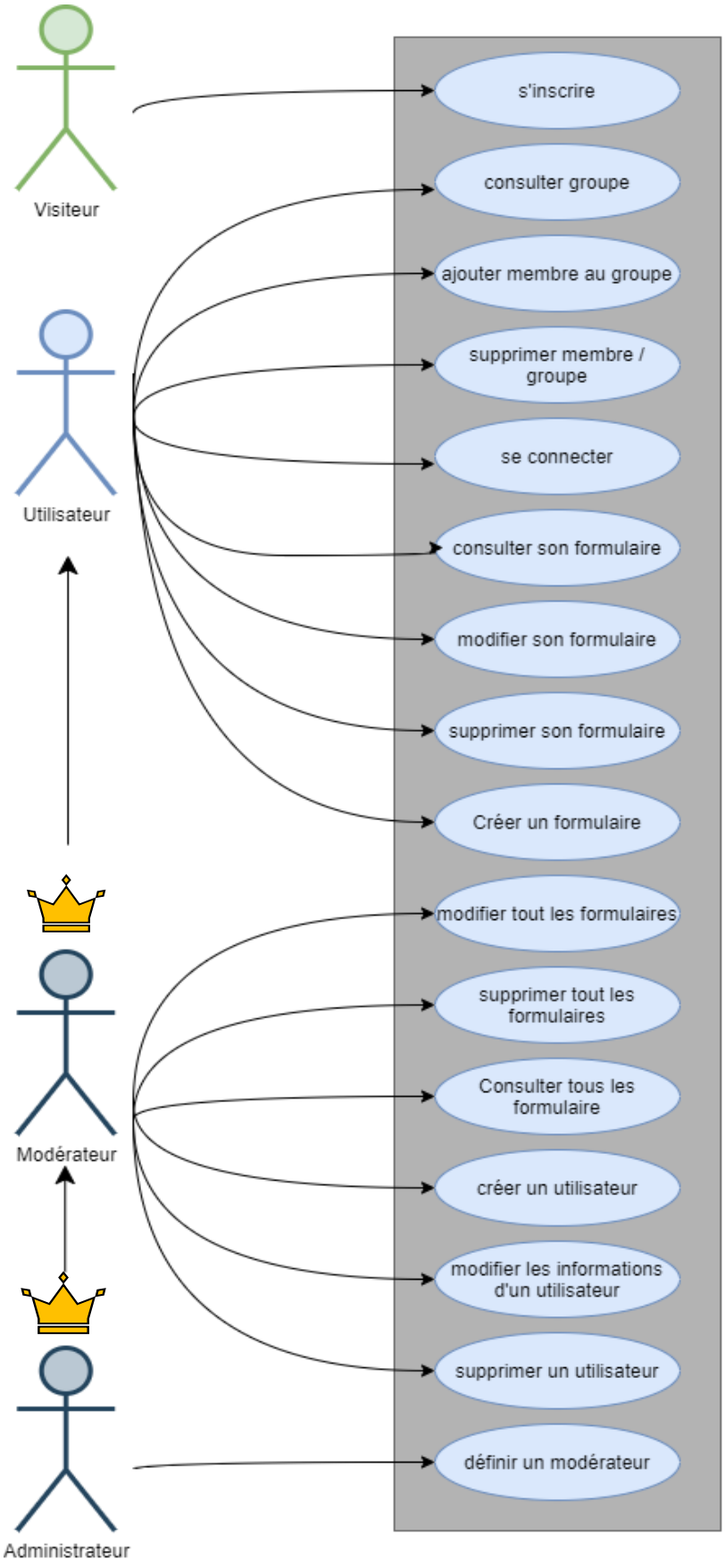


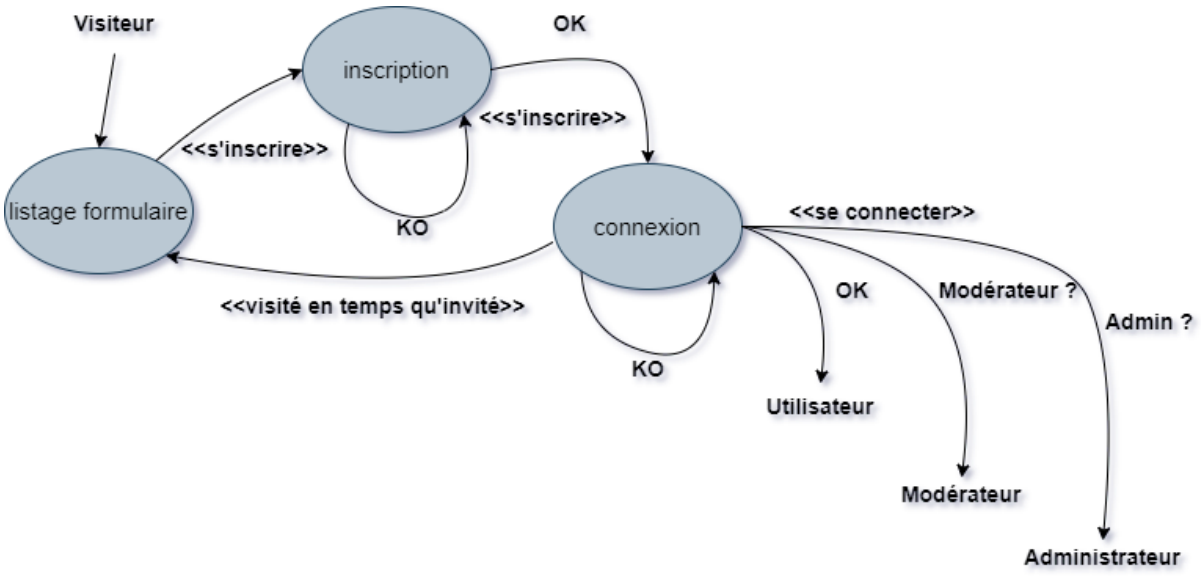
Diagramme de navigation

Pour la connexion nous avons utilisé le système même de Laravel qui est « Auth ». Donc tout est géré par celui-ci. Si c'est un succès il sera redirigé sur l'accueil du site, sinon il restera sur la page de connexion.

Quelques explications :

Visiteur

Le visiteur peut se s'inscrire et consulter la liste des formulaires.



- Inscription :

Controller : RegisteredUserController

Méthode : create store

Route :

Vue : auth.register

Requête :

- Connexion :

Controller : RegisteredUserController

Méthode : create store

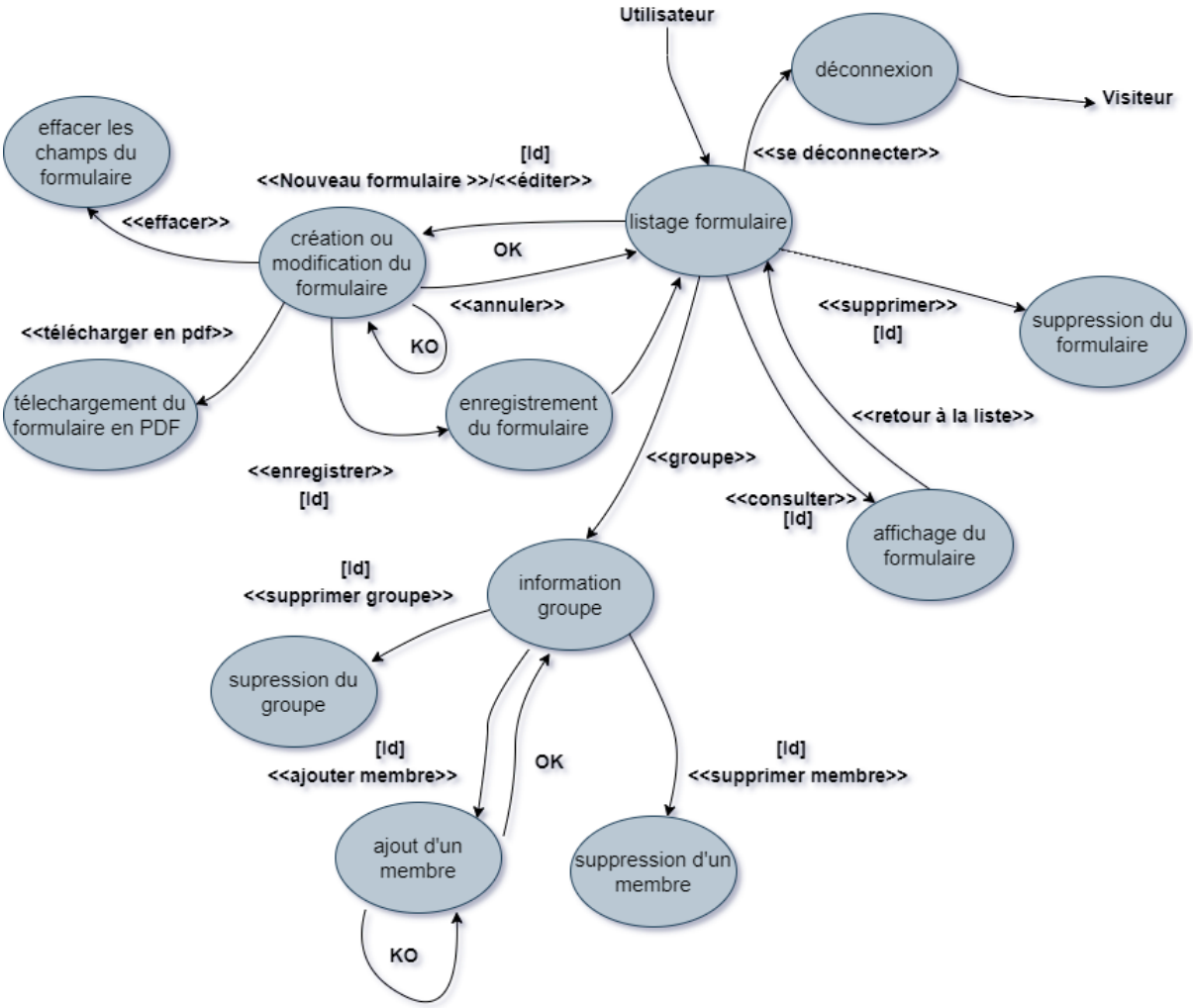
Route :

Vue : auth.login

Requête :

Utilisateur

L'utilisateur (étudiant, particulier, professionnel) peut se connecter et se déconnecter consulter des formulaires et créer, modifier ainsi que supprimer les siens.



- Liste formulaire :

Controller : FormsController

Méthode : forms.index

Vue : forms.list

- Consulter un formulaire :

Controller : FormsController

Méthode : forms.show

Vue : forms.consult

- Creation de formulaire :

Controller : FormsController

Méthode : forms.create

Vue : forms.create

Requête : StoreFormsRequest

- Editer un formulaire :

Controller : FormsController

Méthode : forms.edit

Vue : forms.edit

Requête : StoreFormsRequest

- Supprimer un formulaire :

Controller : FormsController

Méthode : forms.destroy

- Rechercher un formulaire :

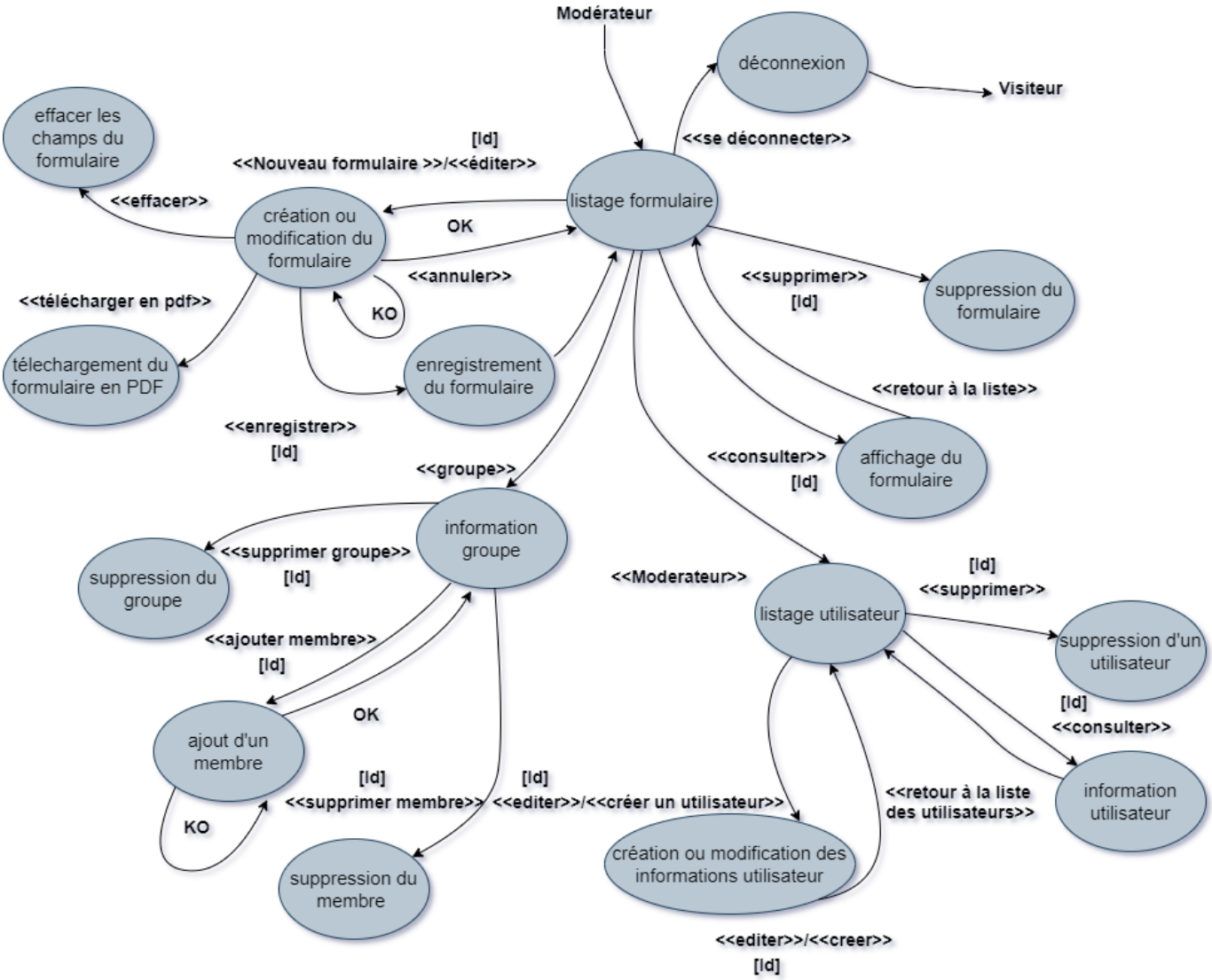
Controller : RechercheAjaxController

Méthode : action

Vue : forms.list

Modérateur

Le modérateur peut se connecter et se déconnecter. Il a la possibilité de consulter, créer, modifier et supprimer des formulaires d'utilisateurs, ainsi que consulter la liste des utilisateurs modifier leur information et pouvoir créer ou supprimer des compte utilisateur du site.



Gérer les utilisateurs.

- Liste utilisateur :

Controller : AdminController

Méthode : admin.index

Vue : admin.list

- Consulter les informations d'un utilisateur :

Controller : AdminController

Méthode : admin.show

Vue : admin.consult

- Création d'un utilisateur :

Controller : AdminController

Route : admin.create

Vue : admin.create

Méthode : admin.store

Requête : StoreAdminRequest

- Editer un utilisateur :

Controller : AdminController

Route : admin.edit

Vue : admin.edit

Requête : StoreAdminRequest

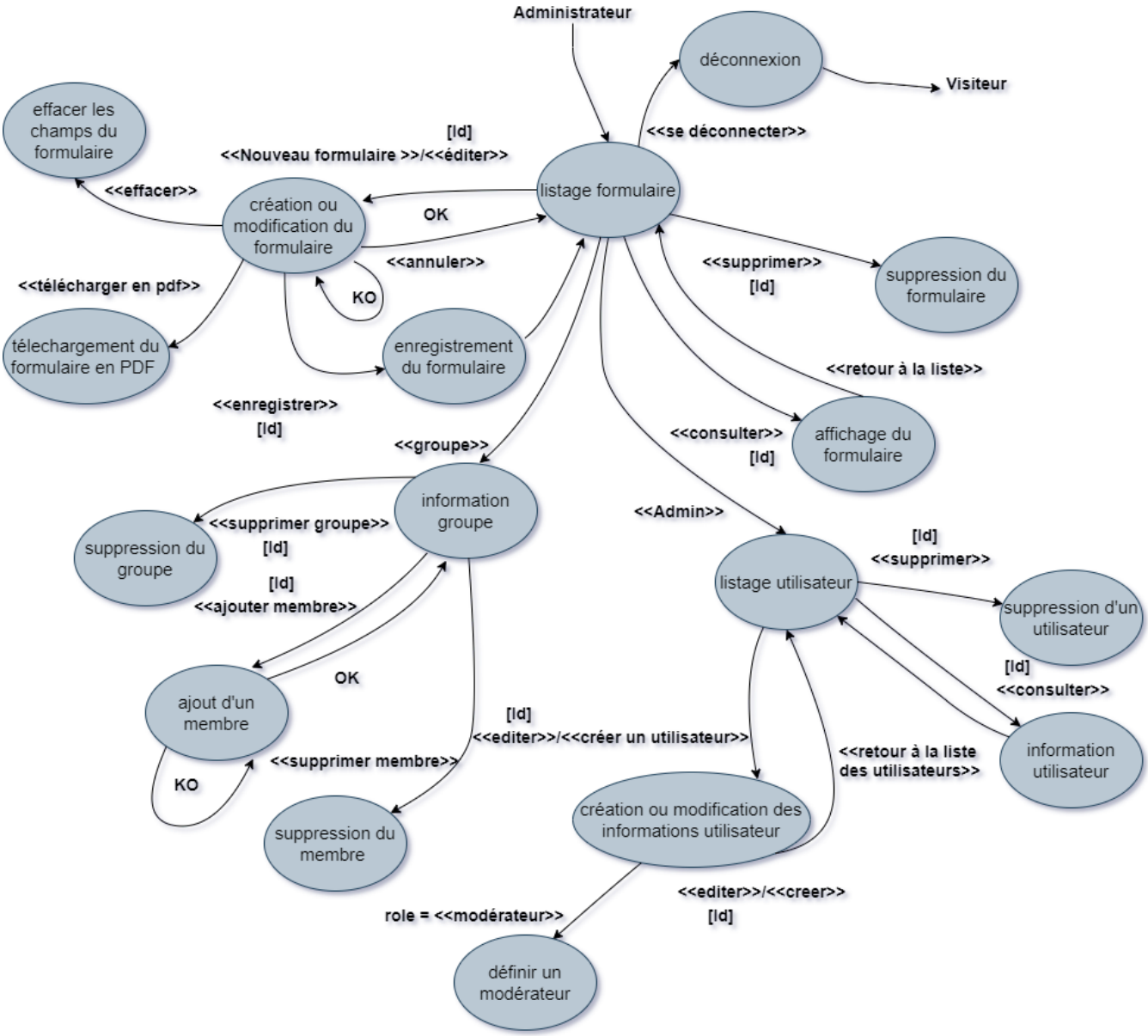
- Supprimer un utilisateur :

Controller : AdminController

Méthode : admin.destroy

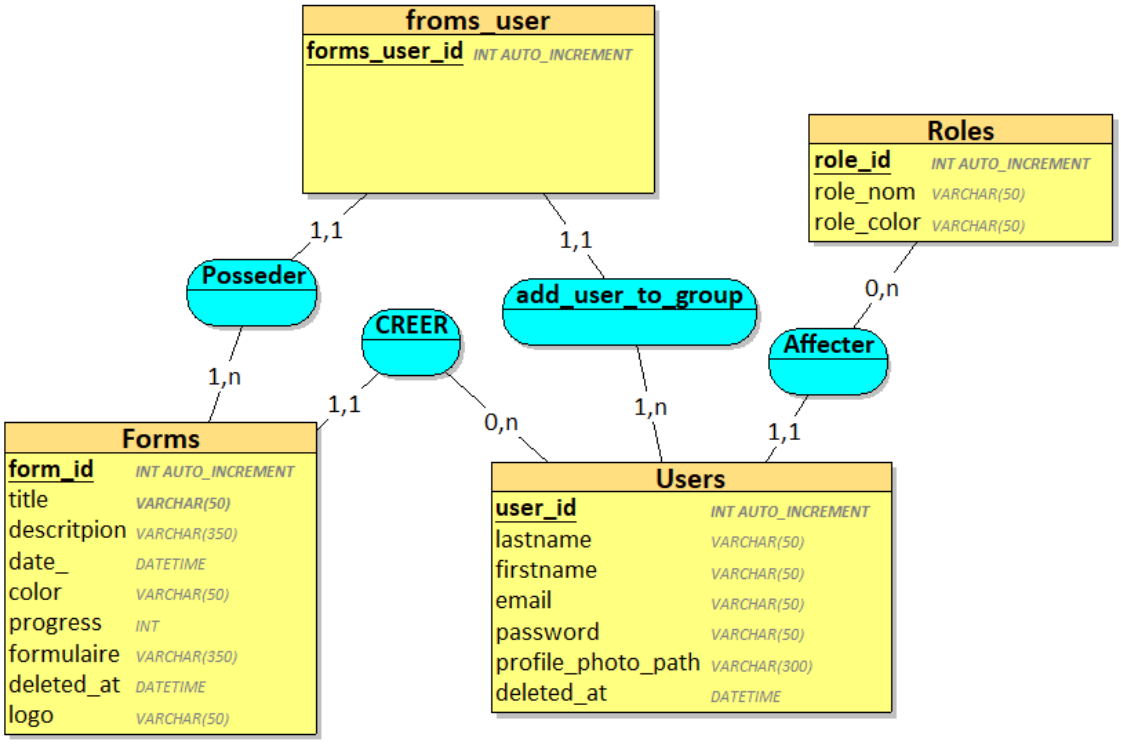
Administrateur

L'admin lui peut en plus définir des modérateurs et modifier leur information.



Modèle conceptuel de donnée (M.C.D)

Le MCD est une représentation graphique de haut niveau qui permet facilement et simplement de comprendre comment les différents éléments sont liés entre eux à l'aide de diagrammes codifiés.

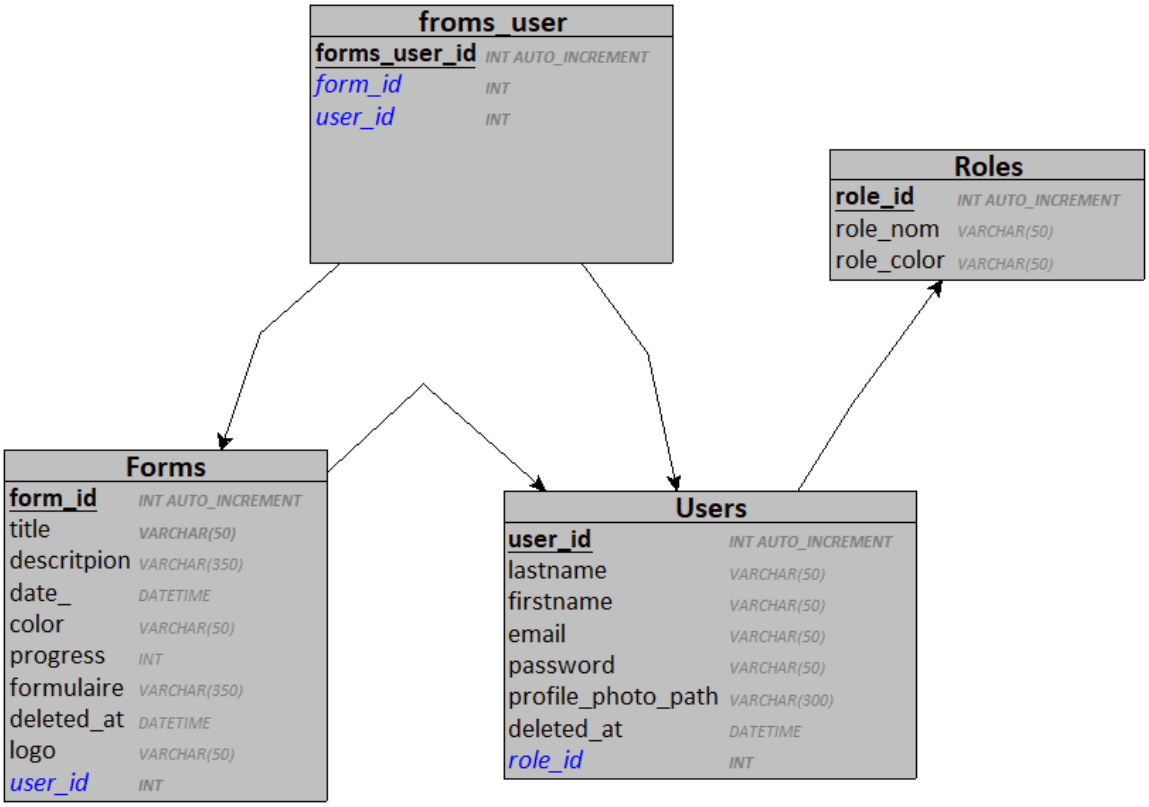


Certaines relations de ce MCD entraineront l'apparition de table pivot (ou cross table). Sur le MCD elles sont représentées en bleu. C'est notamment dû aux relation « belongs to many » (0 :n). En effet pour ce genre de relation nous avons besoin d'une table supplémentaire qui permet de faire le lien entre les deux tables car en BDD nous ne pouvons pas ajouter de colonnes aux tables mais des lignes.

Un utilisateur est affecté à un seul rôle, cet utilisateur peut créer autant de formulaire qu'il le souhaite, un formulaire est créé par seulement un utilisateur celui-ci peut sois être public et accessible pour tous sois privé et il appartient donc à un groupe d'utilisateur.

Modèle logique de donnée (M.L.D)

Le MLD est un modèle relationnel qui utilise des tables mais pas d'associations, et convertit la forme de MCD pour afficher différentes clés primaires et secondaires.



Réalisation

Pour le choix du développement de notre application Laravel FromEditor nous avons fait le choix de créer un Model et d'y associé une ressource, un controller ainsi que la migration associée.

Nous avons fait le choix de peupler notre base de données avec les outils de Laravel à notre disposition, la Factory et le Seeder pour en créer en grande quantité ainsi que de faker pour permettre d'avoir des informations et donnée fictives, non redondante et aléatoire pour rendre l'application beaucoup plus réaliste.

Les données sont affichées avec une pagination au niveau des formulaires ainsi que la possibilité de rechercher un formulaire via la barre de recherche, ce qui nous facilite grandement la tâche si l'on recherche quelque chose de précis rapidement.

Problème rencontré

La réalisation de ce projet fut très longue et nous avons rencontré de nombreux problème de taille qui nous ont fait perdre pas mal de temps.

A chacun de ces problèmes il nous a fallu tester et chercher des solutions par nous-même.

Et si par malheur nous n'avons pas trouvé la solution nous demandions en premier temps une aide dans le serveur discord en deuxième temps une aide à Monsieur Rabat par mail ou pendant les TP et finalement dans les forums tel que stackoverflow par exemple.

1. Problème dans la méthode update et store du formulaire car « forms » en anglais peut ne pas contenir de caractère « s » à la fin ce qui donne « form ». Tandis que par exemple « news » contient par default un 's' et Laravel le comprend.

2. Dans la méthode update de notre controller pour la gestion des utilisateurs nous devions également mettre l'id en paramètre en plus de l'objet user il récupérait bien toutes les informations mais malheureusement pas l'id.

3. Lors du développement de la barre de recherche en ajax nous avons homis certain détail dans la sortie en json de notre html, comme devoir mettre des backslashs (antislashs) pour échapper les caractères comme les guillemets simple et double.

4. Chaque champ de notre formulaire doit avoir un id unique afin d'enregistrer sa valeur et pouvoir l'afficher au moment de la modification du formulaire.
   J'ai utilisé javascript et créer une boucle pour sélectionner tous mes champs via leur classe et concaténer une clé générer aléatoirement à chaque champ id.

5. Nous avions eu un souci pour les Groupes, il nous fallait créer une table pivot entre les formulaires et utilisateurs via un belongsToMany des 2 côtés.
   Mais le souci était de créer un Models FormsUser qui devais avoir une migrations forms_user sans caractère 's' à la fin (l'ordre est important pour la convention Laravel)
   Le Model devait absolument être associé avec la migration.
   Après beaucoup d'acharnement le problème a pu être résolu, la donnée n'est donc plus stockée par l'intermédiaire du Model pivot mais via une requête vers la table migrée soit forms_user directement.

Regret

Malheureusement il y a plein de fonctionnalité qui nous aurions aimé implémenter mais par faute de temps elle non pas étais faites. Comme créer un système de chat entre utilisateur du même groupe, pouvoir modifier les informations de l'utilisateur dans les paramètres utilisateur notamment la photo de profil. Rajouter quelque animation à l'aide de CSS et javascript dans la page d'accueil, pouvoir configurer la barre de recherche pour les utilisateurs à l'aide de ajax, pouvoir insérer une image dans son formulaire ou encore sauvegarder les préférences de l'utilisateur pour le thème sombre par exemple.

Conclusion

Pour finir ce projet a été long à réaliser tant au niveau de la modélisation que de la production.

En effet, nous ne nous rendions pas compte de sa complexité et de son timing de réalisation.

Cependant, c'est notre premier projet web avec l'utilisation de Laravel dont nous sommes assez fiers, bien sûr à l'époque ce n'est pas parfait mais il est fonctionnel et nous pensons que c'est le plus important.

En effet nous aurions pu l'améliorer au niveau de l'utilisation de la visualisation des données avec plus de AJAX par exemple.

Finalement, nous nous sommes plus amusés avec le projet que nous ne le pensions au départ.

Webographie

- https://laravel.com/docs/8.x/installation
- https://stackoverflow.com/
- https://asana.com/fr
- https://stackoverflow.com/users/17430556/stack-realtek
- https://codepen.io/trending
- https://codepen.io/aybukeceylan/full/OJRNbZp
- https://youtu.be/piG91X4sV2U

Annexe

Commande

Voici quelque commande que nous avons utilisé pour le projet sur gitlab:

```
git config --global user.name "haya0002"
git config --global user.email "rahim.hayat@etudiant.univ-reims.fr"
```

New Repository :

```
git clone https://gitlab-mmi.univ-reims.fr/RSFE/formeditor.git FormEditor
cd FormEditorV1.0
git push -u origin master
```

Existing Folder :

```
git init
git remote add origin https://gitlab-mmi.univ-reims.fr/RSFE/formeditor.git
git add *
git commit -m "Commentaire"
git push -u origin master
```

Existing Git Repository :

```
git remote rename origin old-origin
git remote add origin https://gitlab-mmi.univ-reims.fr/RSFE/formeditor.git
```

git push -u origin --all

git push -u origin --tags

Create Branch :

git branch hayat

Switch to Other Branch

git checkout hayat

git merge master // fusion des branch sur la branch master

verification MAJ?

simple git push par la suite !

git push --set-upstream origin hayat // faire un push depuis/vers? la branch hayat

git push -u origin hayat / push vers la branch hayat

Timeline

Vérification des routes avec :

php artisan route:list -c

FormEditorV1.0 : Projet vierge configuré + DebugBar + langue FR {

composer create-project --prefer-dist laravel/laravel FormEditor

cd FormEditor

composer require barryvdh/laravel-debugbar --dev

/* si erreur */ : composer require maximebf/debugbar -W

php artisan vendor:publish // option 1

php artisan storage:link

composer require laravel-lang/lang:~8.0

// dossier vendor larav-lang/src/ copier le dossier fr et le placer dans resource/lang à coté de en .

Configurer le .env et dossier app/config/app.php et database.php

}

FormEditorV1.1 : Configuration des Permission pour Laravel 8 (VM) + Ajout de l'Authentication (Jetstream Livewire) {

/* Jetstream Livewire */

composer require laravel/jetstream --dev

php artisan jetstream:install livewire

npm install

npm run dev

//modifier la route existante :

Route::middleware(['auth:sanctum', 'verified'])->get(

'/dashboard',

[App\Http\Controllers\NewsController::class, 'index'])->name('dashboard');

php artisan migrate:fresh --seed

}

FormEditorV1.2 : Ajout Forms => [Model, Seeder associé, Factory, Controller, Request] + Design Front-End minimum {

resources/view/forms => et les page associées au Formulaire + layouts associé

php artisan make:model Forms --migration

php artisan make:seed FormsSeeder

php artisan make:factory FormsFactory

php artisan db:seed --class=FormsSeeder // commande pour seeder un Seeder en particuliers

php artisan make:controller FormsController --resource

php artisan make:request StoreFormsRequest

php artisan make:migration add_user_to_forms --table=forms // tapper à la main

php artisan make:seed UserSeeder // si elle n'existe pas déjà par défaut

}

FormEditorV1.3 : Ajout Pagination + random profile photo {

Dans le namespace App\Providers :

- ajout de la Pagination dans la vue du forms/list.blade.php et modification du fichier AppServiceProvider

namespace Database\Factories :

UserFactory : ajout photo de profil aléatoire

}

FormEditorV1.4 : Ajout : Role => [Relation One To Many (Relationships)], Gate & Permission, Pagination, Design & Some other! {

php artisan make:model Role --migration

Dans le namespace App\Providers :

- ajout des Gates dans le fichier : AuthServiceProvider

}

FormEditorV1.5 : Front-End => [login, register, error 404], Role => Modérateur, Form => Design {

}

FormEditorV1.6 : Ajout Admin => [Middleware, Controller, Request, Resource] + Réglage BUG {

resources/view/admin => et les page associées au Panel Admin + layouts associé

php artisan make:controller AdminController

}

FormEditorV1.7 : Ajout Barre de recherche AJAX => [RechercheAjaxController], Modification + Ajout FormBuilder et conversion PDF {

// Pour la partie Ajax

php artisan make:controller RechercheAjaxController

// Pour le FormBuilder

npm install html2canvas

https://html2canvas.hertzen.com/

npm install jspdf –save

https://artskydj.github.io/jsPDF/docs/index.html

dossier dist {chercher le lien sur internet}

}