Cahier Technique
Projet de fin d'année Web1 P2019

Groupe #

HÉTIC

1 Table des matières

1 Table des matières 1
2 Introduction 3
2.1 Objectifs 3
2.2 Périmètre 4
2.3 Définitions, acronymes, glossaire 5
2.4 Références 6
2.5 Vue d'ensemble 6
3 Description d'ensemble 6
3.1 Choix techniques 6
3.1.1 Base de données 6
3.1.2 Solution back-end 6
3.1.3 Solution front-end 7
3.2 Dépendances 7
4 Exigences spécifiques 7
4.1 Cas d'utilisations 7
4.2 Exigences supplémentaires 9
5 Base de données 10
5.1 Définition des entités 10
5.2 Modélisation 11
5.3 Projection de volumétrie 11
6 Architecture technique 11
6.1 Classes 11

6.2 Diagramme de classes 13
6.3 Interfaces externes 13
7 Sécurité 13
7.1 Étude des risques 13
7.2 Solutions 14
8 Installation et déploiement 14
8.1 Installation 14
8.2 Déploiement 15
9 Plan de reprise d'activité 15

2 Introduction

2.1 Objectifs

Yalla est une association créée en juillet 2013 qui a pour but de soutenir et d'aider des enfants à se scolariser au Liban en leur apportant une aide humanitaire.
Les objectifs de l'association Yalla se base sur **2 grands points** :

- Garantir le droit à l'éducation et au développement de l'enfant, en permettant son développement dans un environnement sécurisé et adapté, le remettre à niveau et le réhabituer à un rythme scolaire et l'intégrer dans un système scolaire classique afin qu'il puisse obtenir un diplôme et l'aider à construire son avenir.
- Développer des mécanismes de construction de la paix, avec des bénévoles syriens et libanais travaillant autour d'un projet fédérateur et positif, dans le respect et l'égalité et organiser des activités ludiques pour que les enfants syriens et libanais se découvrent.

Yalla étant une association, il lui faut récolter des fonds de particuliers comme d'organisations pour continuer à vivre et nourrir ses projets. Dans ce but, un site internet a été fondé, afin de faire connaître le projet et d'avoir une présence sur internet. Yalla possède également un compte Twitter et Facebook.
Le site actuel permet les fonctionnalités suivantes :

- se renseigner sur l'histoire de l'association, son équipe, ses projets, sa mission et ses partenaires
- contacter directement par mail ou par voie postale l'association
- faire des dons et devenir membre de Yalla

Yalla organise occasionnellement des ventes pour récolter les fonds. (ex : vente temporaire de vêtements)
Dans notre refonte, l'objectif est de **faciliter la navigation sur le site** en réduisant le nombre d'onglets dans la barre de navigation et de supprimer les menus déroulants. Le but est de rassembler le contenu :

- Page Nous connaître qui rassemblera « notre philosophie », « notre équipe » et « notre fonctionnement ».
- Page Nos projets qui rassemblera les « notre projet passé », « notre projet actuel » et « contexte ».
- Page Notre mission qui rassemblera « notre stratégie », « nos actions » et « nos objectifs ».

Notre refonte mettra également en place les fonctionnalités suivantes :

- un nouveau logo → pour contribuer au design moderne que nous voulons apporter.
- une nouvelle charte graphique → l'orange utilisé est trop agressif, d'autres couleurs seraient plus adaptées.
- possibilité de lire le site en anglais, français et arabe → pour cibler un plus large public à travers le monde.
- mettre en avant les réseaux sociaux et le fait de donner avec un bouton directement en haut du site ou dans la nav → inciter le visiteur

du site à donner et lui éviter de chercher à travers le site comment faire un don.

Proposition après paiement d'une page avec la possibilitée de remplir un champ "nom" et photo afin de faire un board des participants au projet.

Les fonctionnalités du back-office :

- Ajouter/supprimer/éditer/prévisualiser un article et le récupérer dans sa version antérieure si erreur.
- Visualiser les statistiques (nombre de vus des articles)
- Ajout de tags et catégories et les lier à un article.
- Possibilité de changer contenu de pages exemple : type page contact. Ajout de différent type de langue.

Nous gardons les anciennes fonctionnalités qui sont indispensables au bon fonctionnement du site.

2.2 Périmètre

Le périmètre du projet est défini par 3 grands types : besoins, délai et budget:

Tout d'abord notre délai. Le projet a été lancé le Lundi 19 juin à 9h du matin.
Il durera jusqu'aux vendredi 30 juin soit un temps effectif de 9 jours.
Le temps effectif est de 7h par jours et par personnes.
L'équipe est composé de 6 personnes soit un temps effectif de 42h par jour soit globalement 380h pour le groupe entier.
Ceci représente un budget de 32 400€.

Fonctionnalité :

Front office:

- Tri des articles
- Menu burger

Back office:

- Update
- Ajout
- Supprimer
- Prévisualisation
- Statistiques de visibilité
- Changer de langue
- Sauvegarde de la DB
- Hide/Show des articles
- (Messages)

2.3 Définitions, acronymes, glossaire

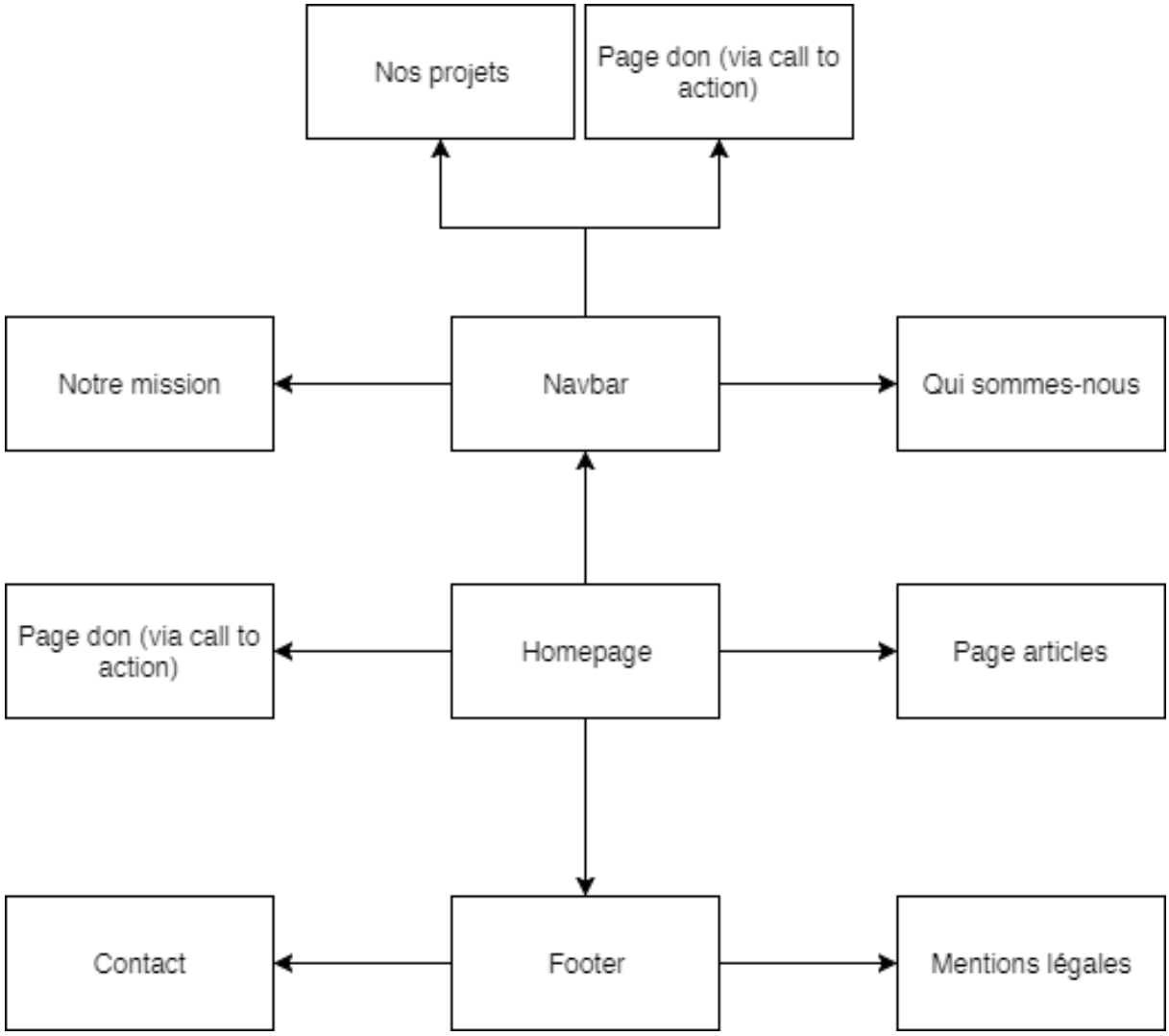
FR
Association humanitaire : Une association humanitaire fournit une aide d'urgence et ponctuelle lors d'une situation de crise exceptionnelle ou de la catastrophe naturelle.
Don : Action de donner, chose ou somme donnée, à une institution, œuvre ou association par exemple.
Crise humanitaire : Une situation dans laquelle la vie d'un grand nombre de personnes est menacée de façon exceptionnelle et immédiate.
Crowdfunding : Levée de fonds sur internet.

EN
Charity organization : A type of non-profit organization which provides an immediate help to an exceptional crisis or natural disaster.
Donation : Act of giving a sum of money or a thing to a fondation, an organization or a charity for instance.
Humanitarian crisis : A situation that puts the life of a large group of people at risk in an exceptional and immediate way.
Crowdfunding : Web fundraising.

Acronymes
W4 : Women's WorldWide Web
OFPRA : Office français de protection des réfugiés et des apatrides
CALAM : Cours d'arabe à Paris
CODSSY : Collectif De Développement Et Secours Syrien
UNICEF : Fonds des Nations unies pour l'enfance

2.4 Références

2.5 Vue d'ensemble



3 Description d'ensemble

3.1 Choix techniques

3.1.1 Base de données

Nous avons décidé d'utiliser une Base De Donnée MySQL.

3.1.2 Solution back-end

Afin de mettre en place une solution back-end adaptée à la demande, nous avons opté pour une architecture basée sur le framework Laravel (version 5.4), se basant sur une version de PHP supérieure ou égale à 5.6.
Nous avons porté notre choix sur ce framework car il est maintenu et le sera encore longtemps et possède une large communauté.

Pour compléter les fonctionnalités de bases mises en œuvre par Laravel, nous avons sélectionné différentes dépendances côté back-end telles que :
Laravel-translatable
Laravelcollective/html 5.4
L'usage de ces dépendances est détaillé un peu plus bas (3.2).

Bootstrap pour faciliter l'intégration du Backoffice
Disposition simple et efficace

Au niveau des dépendances Front, nous utiliserons chart.js (voir plus bas).

3.1.3 Solution front-end

Le framework AngularJS 1 nous semble être une solution intéressante pour gérer les différents états de l'interface, présentant une utilisation plus facile des mécaniques AJAX et même de routing front-end pour une partie plus petite de notre application (tel que le dashboard back-office).
Ce framework est très largement documenté et possède également une communauté active.

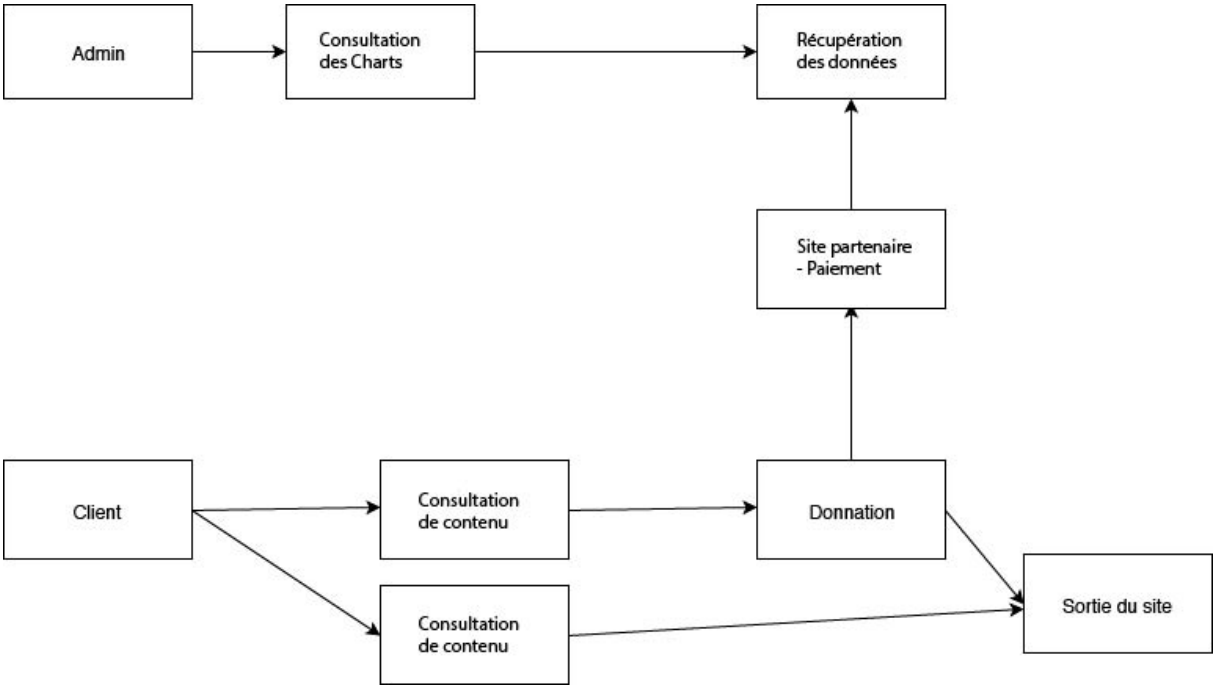
3.2 Dépendances

Back :
laravelcollective/html :
Permet la création et la gestion facile des formulaires, bien évidemment compatible avec le framework Laravel (et facilite le fonctionnement).
Front :
chart.js :
Permet une disposition sous forme de graphique des données
Ex : Tant de % de visites en plus tel mois par rapport à tel mois, tant de donneurs, parrains etc
Surtout utile pour le dashboard back-office.

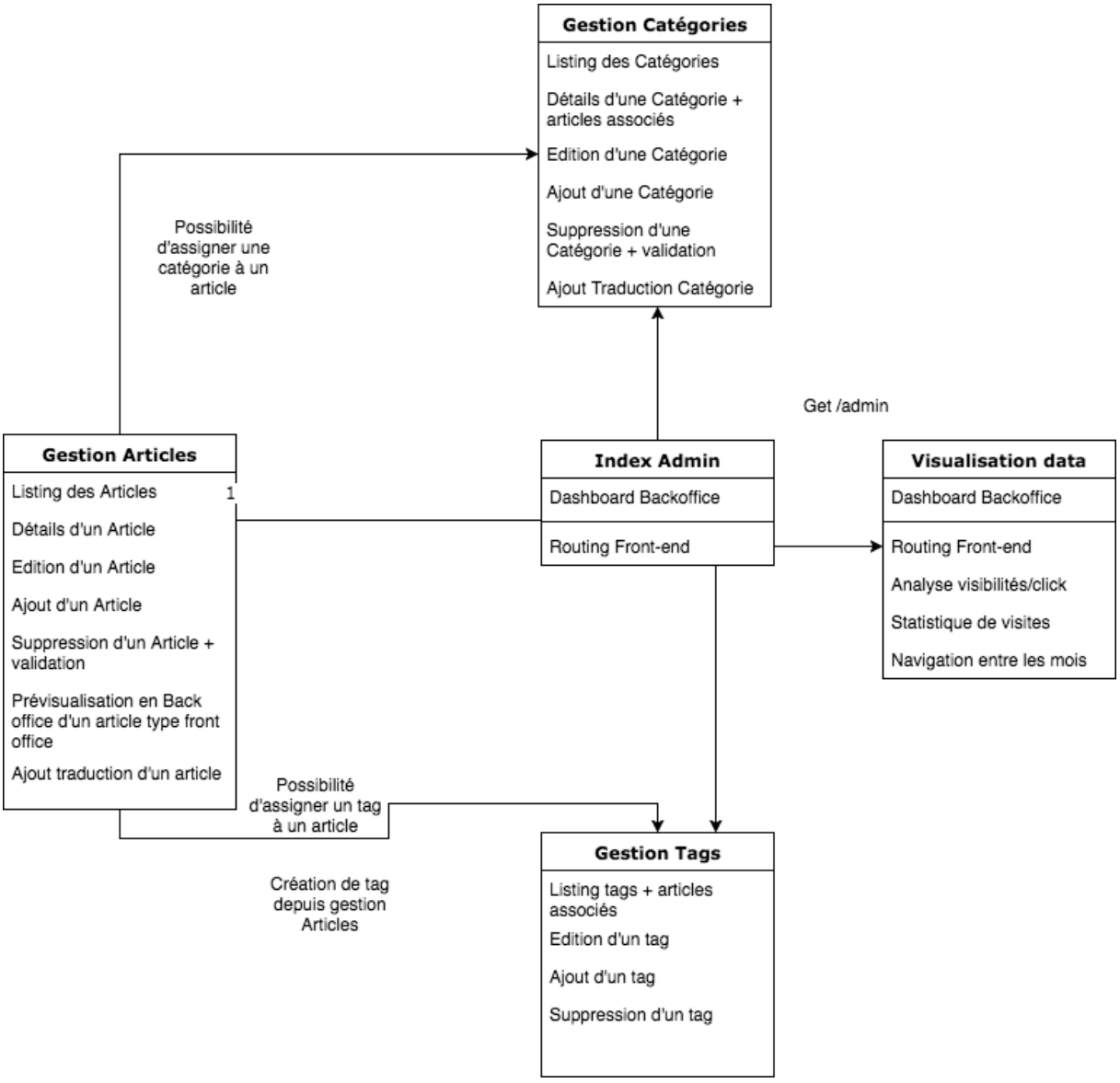
4 Exigences spécifiques

4.1 Cas d'utilisations

Front-office


Admin
Consultation des Charts
Récupération des données
Site partenaire - Paiement
Client
Consultation de contenu
Donnation
Consultation de contenu
Sortie du site

Back-office



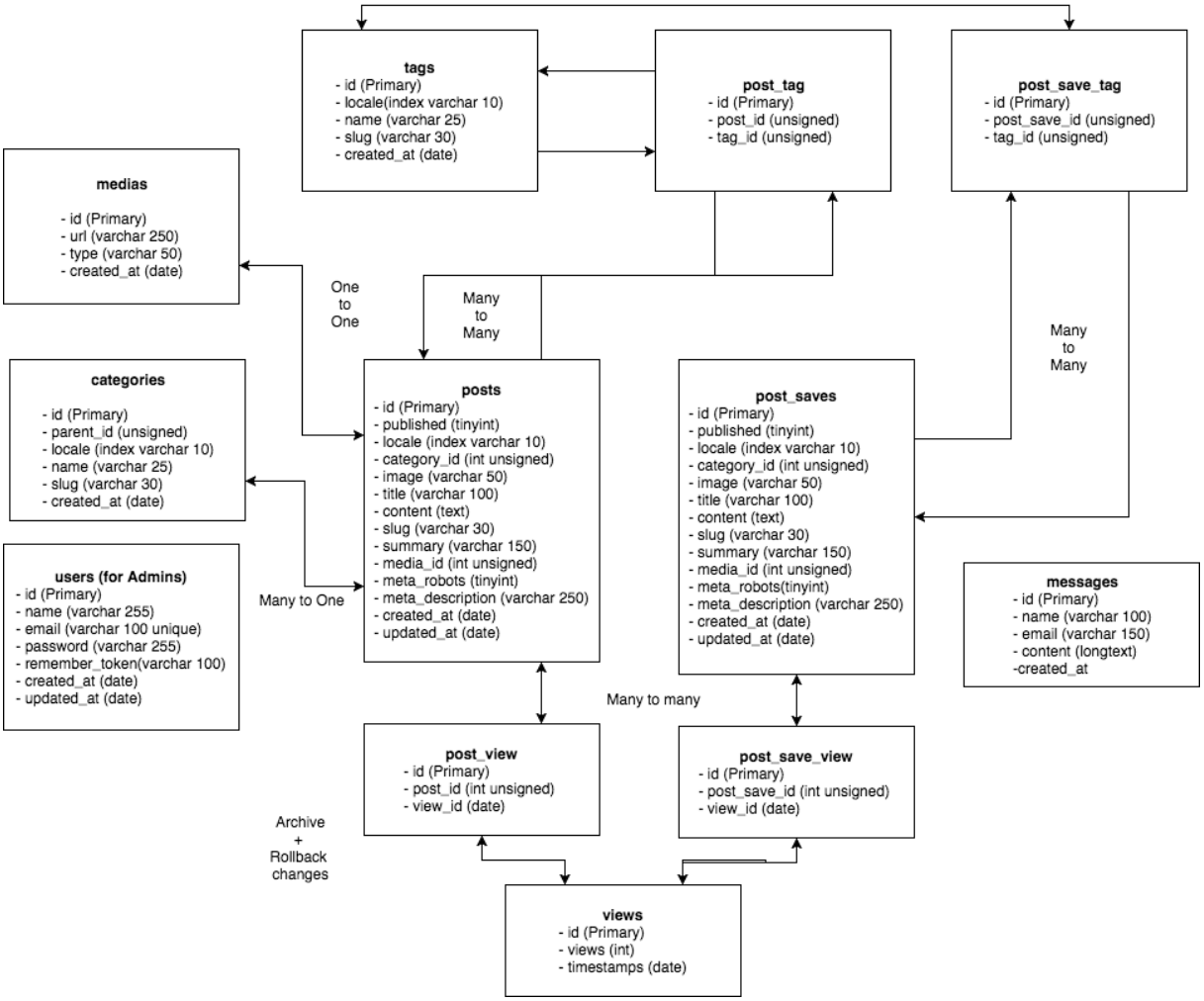
4.2 Exigences supplémentaires

5 Base de données

5.1 Définition des entités

- Posts : Entité représentant un contenu (article), il est caractérisé par ses propres données de référencements, médias (images/video etc), son état (publié ou non) et sera mis en ligne par un Administrateur.
- Tags : Entité représentant les tags qu'on pourra attribuer à des posts.
- Catégories : Entité représentant une catégorie qui regroupera plusieurs articles et permettra de classifier chaque post en fonction de son contenu.
- Metas : Entité contenant toutes les données nécessaires au bon référencement/partage sur les réseaux sociaux d'un article.
- Post_Saves : Permettra une sauvegarde des posts dans la base de données lors de suppression/edition d'un article → Permettra de garder un œil sur l'historique et de lancer une procédure de retour en arrière si mauvaise manipulation.
-Views : Compteur de view par post par mois ➔ contient le nombre de visites d'un article disposé sur un mois (champ created_at) et permettra d'établir des statistiques de visites sur plusieurs mois consécutifs.
- post_tag, post_save_tag, post_view et post_save_view représenteront des tables dites de « pivot » qui permettront de lier les différentes entités entre elles (un post/sauvegarde sera lié à plusieurs tags et plusieurs compteurs de vues, et un tag sera lié à plusieurs posts/sauvegardes).
- Users : contiendra les comptes des Admin pour accéder au Backoffice
- Medias : contient des médias classées par type(image, vidéo, lien etc) et sera lié à des articles par le biais d'une entité de pivot media_post (on retrouve le même cas pour post_tag).
- Messages contiendra les messages reçu depuis le formulaire de contact dans le front-office ainsi que les différentes informations concernant l'expéditeur (e-mail/nom) .

5.2 Modélisation



5.3 Projection de volumétrie

La disposition de la base de données que nous avons choisis d'adopter permet une maintenabilité simple et efficace au cours du temps. Effectivement, afin d'ajouter des posts / tags / categories ou même une traduction des champs concernant le référencement d'un post en particulier, nous avons pris la liberté de créer des tables spécifiques, ainsi on peut tout simplement ajouter une traduction pour une locale différente et tout le contenu du site dans la langue concernée sera disponible en fonction. Les nombreuses relations que nous avons instauré dans la BDD permettront une accessibilité facilité aux propriétés des différentes entités.
En terme de volume, ces différentes relations permettent une conservation d'un grand nombre de données sans risque de bug, et laissant une place accessible aux possibles évolutions ou/et ajouts de fonctionnalités.

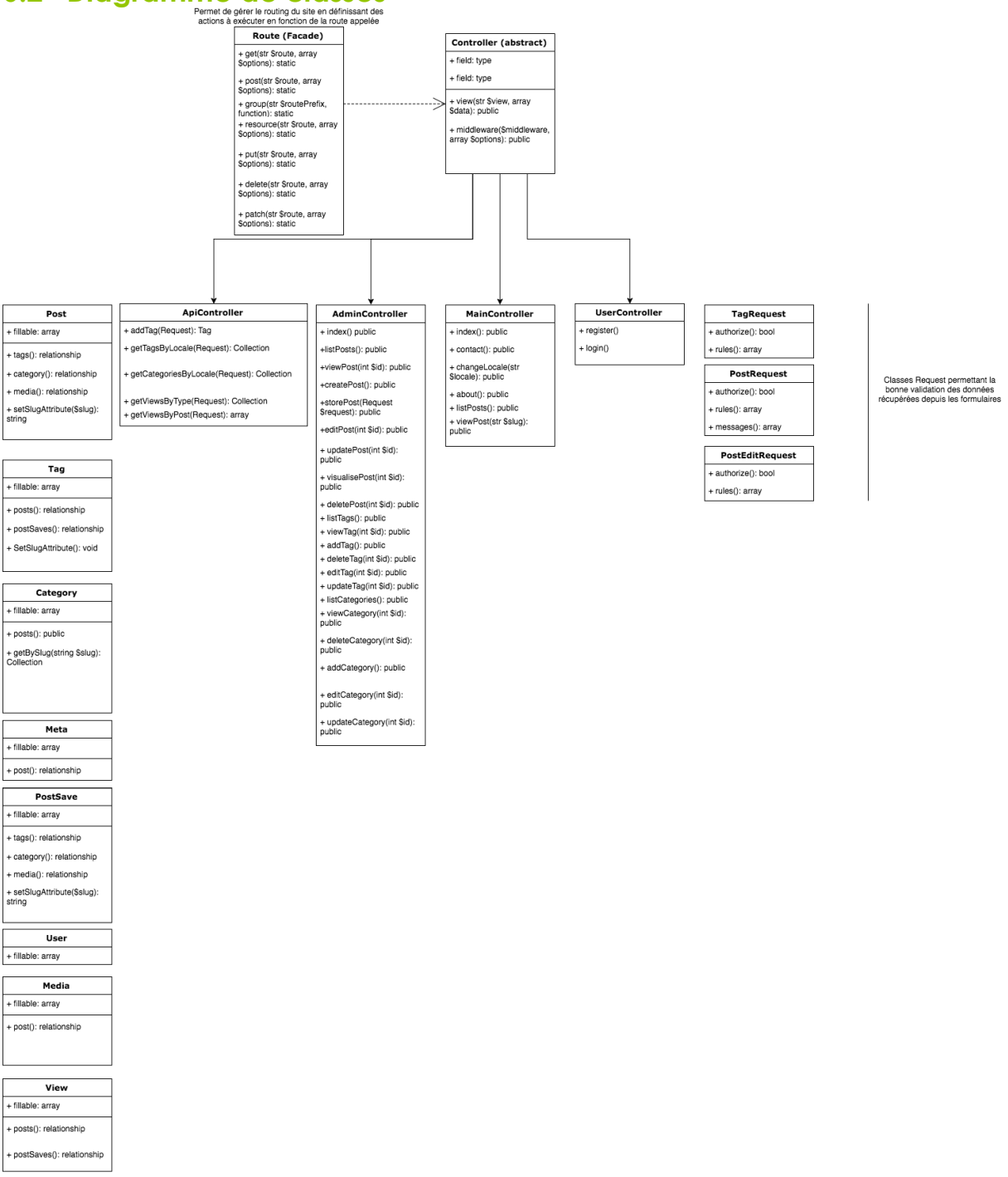
6 Architecture technique

6.1 Classes

En plus des classes internes au framework Laravel, nous créerons des classes servant à différents buts :

- Des controllers qui seront chargés de gérer toute la gestion propre à une partie du site (ex : AdminController pour tout le panel Admin, il sera le maître d'orchestre et appellera des entités de la BDD, sera chargé des vérifications (données/formulaires) et de rendre les vues adaptées.
- Des Model (entités) : modélisation sous forme d'objet PHP et représentant une table de la base de données, sera chargée de renvoyer des objets « Collection » contenant les données appelées.
- Les Request : Représenteront les vérifications à appliquer lors de la réception des données provenant d'un formulaire (par exemple à l'ajout d'un post dans la base de données), on précisera à l'intérerieur les champs à vérifier et la vérification à effectuer (ex : caractères alpha uniquement, entre 5 et 16 caractères).

6.2 Diagramme de classes



6.3 Interfaces externes

Nous utiliserons un trait externes au framework et propre à une dépendance que nous utilisons :

7 Sécurité

7.1 Étude des risques

Les risques principaux sont constitués dans le panel administrateur, si un utilisateur non administrateur réussi à accéder au panel administrateur, il pourra choisir de supprime, d'ajouter ou de modifier l'ensemble des éléments présents dans la BDD et impacter directement le site tel qu'on le voit dans le front office.
De la même façon, les actions qui touchent directement à la base de données doivent être protégés pour ne pas permettre à n'importe quel individu d'accéder à la Base de données.

7.2 Solutions

Afin de protéger le site des différents risques listés ci-dessus, plusieurs moyens seront mis en place :

- Protection du panel administrateur par un système d'authentification.
- Protection de la réception de données concernant l'altération de données par un token de vérification (csrf) géré directement par le framework Laravel.
- Vérification de toutes les données reçues par le biais d'instance de classes personnalisés de type « Request », déterminant les champs requis, et des règles concernant l'état des données avant de toucher directement aux entités.

8 Installation et déploiement

8.1 Installation

Dans un premier lieu, il faut s'assurer que la machine que vous utilisez possède php, si vous travaillez sur MAC, php sera installé par défaut et vous pourrez passer à l'étape suivante, dans un autre cas je vous recommande de vous orienter vers la documentation php visible sur la page ci-dessous :
http://php.net/manual/fr/install.php

Afin d'installer correctement le projet, il faut installer les dépendances du site et du framework en utilisant la ligne de commande :

```
$ composer install
```

Si composer est installé en globale sur votre machine.
Nous prendrons le soin de déposer le fichier composer.phar dans le dossier contenant le projet. Au quel cas il faudra utiliser la ligne de commande :

```
$ php composer.phar install
```

Si vous désirez refaire une installation pour la première fois sur un nouveau serveur ou en local, il faut d'abord s'assurer que composer est installé

→ dans le dossier du projet, utiliser la ligne de commande avant d'utiliser les commandes listées précédemment.

$ php composer.phar

8.2 Déploiement

Afin de procéder à un déploiement optimal du projet Yalla sur un hébergement.
Il vous faut dans un premier temps vous munir des informations concernant l'accès à la base de données de votre hébergeur (Utilisateur et Mot-de-passe, lien de l'hôte phpMyAdmin).
Il vous faudra modifier le fichier .env disponible à la racine du projet, la configuration de la base de donnée vous sera demandé.
Il faut ensuite modifier les variables d'environnement HOST par le lien de l'hôte phpMyAdmin, USER par l'utilisateur de votre hébergement, et PASSWORD par le mot de passe choisis pour accéder à la base de donnée.
Ensuite, à l'aide d'un logiciel d'upload de fichier (tel FileZilla), connectez vous sur votre client d'hébergement et déposez sous la racine le contenu du projet excepté le dossier public.
Vous pouvez si vous le voulez mettre en ligne le dossier vendor également, ou suivre les recommandations d'installation décrites plus haut afin d'installer les dépendances du projet.
Pour cette dernière étape il vous faudra lancer la commande suivante pour mettre à jours votre base de donnée dans un terminal, à la racine du serveur:

$ php artisan migrate

Ce qui permettra de remplir correctement le Base de données.
Pour cette dernière étape vous devrez tout simplement mettre en ligne dans votre logiciel d'Upload le contenu du dossier public (tous les fichiers/dossiers) à la racine de votre hébergement (souvent appelé "www").

9 Plan de reprise d'activité

Afin de faciliter une reprise d'activité en cas de problème sur le serveur ou dans la base de données, nous avons décidé de mettre en place un système de sauvegarde récurrent des composants de la Base de données sur le Github afin de permettre à un administrateur ou quelconque individu de relancer l'activité du site en suivant simplement les processus d'installation et de déploiement, avec un processus supplémentaire si le soucis a eu lieu dans la BDD :

Un fichier .sql sera mis à jours contenant l'ensemble des données et de la structure globale (tables) dans le dossier du projet sur le site github.

Ainsi, il faudra télécharger ce fichier et l'importer dans la base de données grace à l'outils phpMyAdmin disponible sur votre hébergeur.