

29/03/2019

# logiciel gestion de don

Laravel

Yohanan HIRSCH

CERTIFICATION NIVEAU III DEVELOPEUR WEB



## TABLE DES MATIERES

Résumé du projet .....	4
<b>le projet</b> .....	5
Technologie utilisé .....	6
Cahier des charges .....	7
Cahier des charges fonctionnel .....	7
Divers.....	8
Le projet EN TANT que tel .....	9
Choix de départ : .....	9
Generalités sur laravel.....	10
Construction de la base de donnéeS du site.....	13
Opérateur de bits appliqué à notre travail et à laravel .....	15
Installation des tables et migration dans laravel.....	16
VALIDATION DANS LARAVEL .....	21
Étude des cas d'inscriptions d'une association et d'un DONATEUR PAR un client. ....	23
INSERTION D'UN DONATEUR PAR LE CLIENT .....	27
GESTION DU MASQUAGE DANS LARAVEL .....	31
GESTION DE LA CRÉATION D'UN CERFA DANS LE PROJET.....	35
Génération du cerfa en pdf et envoi d'email.....	40
<b>Test de l'application</b> .....	45
Conclusion, pointS à évoluer du projet.....	49
ANNEXES.....	50
LARAVEL.....	51
CONSOLE .....	51
glossaire .....	53
DICTIONNAIRE DES TABLES.....	57
Base de données laravel_association .....	57
Généralité sur les associations.....	62
Lise Reynaert, Aurélien d'Isanto, division Enquêtes thématiques et études transversales, Insee	62
Cadre juridique des associations.....	63
Article 238 bis du Code général des impôts.....	63
Mask of bits théorie .....	66
Utilisation du masquage par un exemple .....	67



### RESUME DU PROJET

Mon projet se fait dans le cadre de TekWS. Cette entreprise a été fondée par M. SCHULMANN M., programmeur analyste expert en informatique, Tekws se compose de deux grands pôles : la formation et la création de site internet

L'idée du projet est de travailler sous une autre forme, avec de nouvelles technologies, un logiciel existant de gestion de dons et d'envoi de cerfa. Le but principal étant de savoir si mon projet peut être plus performant en terme de rapidité et de sécurité.

La gestion de dons et l'envoi de cerfa demande dans un premier temps une réflexion du besoin de l'utilisateur et une recherche juridique à propos des cerfa ; en effet, ceux-ci donnent droit à une baisse d'impôt et sont donc encadrés par la loi. Ceci fait, le cahier des charges peut être finalisé.

Le choix de départ est de travailler avec le langage php et en particulier avec le Framework laravel. Celui-ci contient divers outils qui permettent de travailler beaucoup plus rapidement notamment un outil de login, validation, etc.

Dans ce projet précis, j'ai eu la possibilité de travailler avec le maskage de bit, notion qui permet de travailler différemment les relations entre tables. Enfin, autre point important du projet, l'utilisation de dompdf, qui permet de générer des pdf et l'envoi d'email gérés par laravel

## LE PROJET

### TECHNOLOGIE UTILISE

Lors de mon projet voici les technologies utilisées. Pour plus de précision sur celles-ci voir annexe et la suite du document :

- Php
- Sql
- laravel 5.7
- Html 5
- Css 3
- Fontawesome
- Bootstrap 4.1
- JQuery
- Ajax
- Maildev
- Visual code
- Mysql
- Phpmyadmin

## CAHIER DES CHARGES

### CAHIER DES CHARGES FONCTIONNEL

#### Évolution du logiciel

Le logiciel doit être fait de manière à pouvoir aisément évoluer, en particulier vers un web service

---

#### LES INTERVENANTS

Dans un premier temps ceux-ci sont aux nombres de 4 :

- l'administrateur : celui-ci peut gérer l'inscription des clients.
- Les clients : celui-ci peut gérer l'inscription des associations, des donateurs, et des clients restreints
- les clients restreints : peuvent gérer d'une à la totalité des associations ainsi que leurs donateurs
- les donateurs peuvent accéder aux informations les concernant,

---

#### ACCESSIBILITE DU LOGICIEL

Le logiciel est accessible au niveau du web, toute les fonctionnalités du logiciel se font à partir d'un compte validé par mail

---

#### SECURITE

- Sécurité des informations : les informations ne doivent pas être transparentes,
- Le logiciel ne doit pas être sensible aux attaques Cross-site scripting :

Le Cross-site scripting (abrégé XSS), est un type de faille de sécurité des sites web permettant d'injecter du contenu dans une page, provoquant ainsi des actions sur les navigateurs web visitant la page. Les possibilités des XSS sont très larges puisque l'attaquant peut utiliser tous les langages pris en charge par le navigateur (JavaScript, Java, Flash...) et de nouvelles possibilités sont régulièrement découvertes notamment avec l'arrivée de nouvelles technologies comme HTML 5. Il est possible par exemple, de rediriger vers un autre site pour de l'Hameçonnage ou encore de voler la session en récupérant les cookies.



- Le logiciel ne doit pas être sensible aux attaques du type « insertion de code »,

---

### FONCTIONNALITES DU PROJET

- Intégration de nouveaux intervenants
- Intégration de dons
- Génération de cerfa de dons sous format pdf
- Envoi par email du cerfa au donateur, une fois celui-ci validé
- Vue des différents dons
- Vue des différentes associations
- Vue des différents acteurs gérés par l'utilisateur
- Possibilité de modifier les acteurs gérés par l'utilisateur
- Possibilité de modifier les dons (ajout d'un message d'attention lorsque le cerfa est validé)
- Validation des utilisateurs par email
- Recherche via ajax des utilisateurs lors de l'ajout ou de la modification des dons
- Ajout de la signature de l'utilisateur dans le pdf
- Ajout de texte accompagnant le pdf
- Respect des normes à propos du cerfa

---

### DIVERS

Une attention importante est demandée à propos des données de l'utilisateur, un client ne peut avoir accès aux données d'un autre utilisateur. La modification des données par un utilisateur ne peut pas avoir d'influence sur les données d'un autre utilisateur. Un donateur d'une association peut être donateur d'une autre association. De plus un client restreint d'un client peut être client restreint d'un autre client.

Autre point particulier, si possible travailler avec le masquages de bit lors de relation de cardinalité (n,n) entre tables

## LE PROJET EN TANT QUE TEL

### CHOIX DE DEPART :

Le travail se fait à partir du langage php 7.2, qui est une des versions les plus récentes du langage php. J'utilise aussi dans le projet javascript ou plutôt jQuery ainsi que Ajax

Pour la bonne pratique du langage et pour faciliter le développement du logiciel j'utilise le framework Laravel.

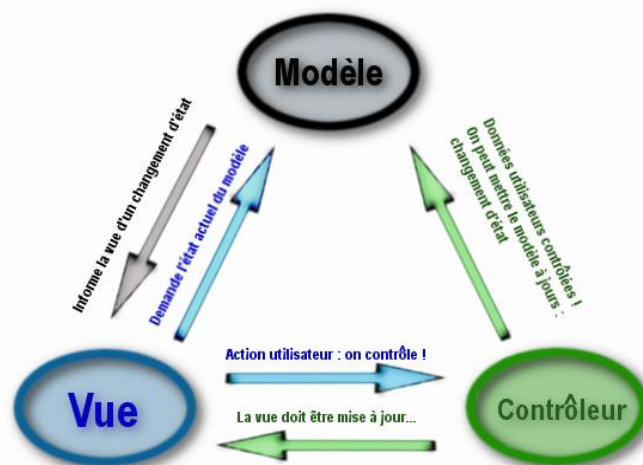
L'architecture de Laravel se base sur l'architecture mvc :

Le Modèle Vue Contrôleur (MVC) est un patron d'architecture logicielle, généralement utilisé pour implémenter des interfaces utilisateurs : c'est donc un choix populaire pour la conception d'applications web. En général, on sépare l'application en trois parts distinctes, permettant la modularité et rendant plus simple la collaboration et la réutilisation. Ce modèle rend aussi les applications plus flexibles et plus accueillantes aux itérations.

Le modèle détermine quelles données l'application doit contenir. Si l'état de ces données change, alors le modèle va généralement avertir la vue (donc l'affichage peut changer au besoin) et parfois le contrôleur (dans le cas où une logique différente est nécessaire pour contrôler la modification de la vue).

La vue détermine comment les données de l'application doivent être affichées.

Le contrôleur contient toute la logique de mise à jour du modèle et/ou la vue en réponse aux entrées (actions) de l'utilisateur sur l'application.



### GENERALITES SUR LARAVEL

(Source Maurice chavelli)

D'après Wikipédia, un Framework informatique est un « ensemble cohérent de composants logiciels structurels, qui sert à créer les fondations ainsi que les grandes lignes de tout ou d'une partie d'un logiciel » ; autrement dit, c'est une base cohérente avec des briques toutes prêtes à disposition, c'est une maison dans laquelle on a déjà construit les fondations et apporter la plupart du matériel de construction. L'utilité d'un Framework est d'éviter de passer du temps à développer ce qui a déjà été fait par d'autres, souvent plus compétents, et qui a en plus été utilisé et validé par de nombreux utilisateurs.

. Il existe des Framework pour tous les langages de programmation et en particulier pour PHP, les plus connus dans ce langage sont Symfony et Laravel.

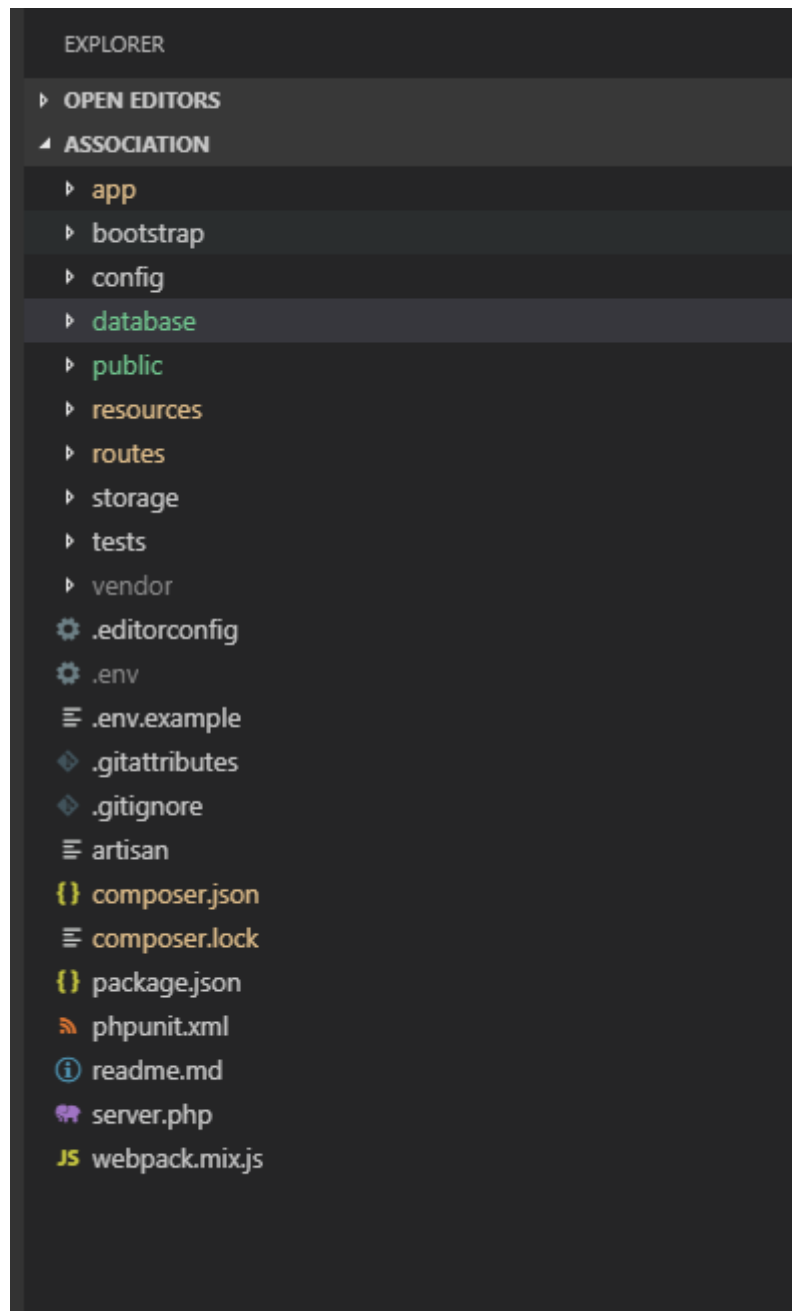
Laravel est un Framework qui utilise des composants d'autre source, dont symfony.

A l'intérieur il est possible de trouver :

- Un système de routage perfectionné (RESTFUL et ressources) ;
- Un créateur de requêtes SQL et un ORM performants (eloquent)
- Un moteur de templates efficace (Blade)
- Un système d'authentification pour les connexions
- Un système de validation
- Un système de pagination
- Un système de migration pour les bases de données
- Un système d'envoi de courriels
- Un système de cache
- Un système d'événements
- Un système d'autorisations (middleware)
- Une gestion des sessions...
- Et bien d'autres choses....

Il existe plusieurs façons d'installer Laravel, la plus courante est de passer par Composer. Composer est un gestionnaire de dépendance écrit en PHP, celui-ci se travail en ligne de commande. L'un des avantages de Composer et des gestionnaires de dépendance est de pouvoir gérer les mises à jour des composants

Une fois Laravel installé, l'organisation des dossiers se visualise ainsi :



Le dossier App, dans lequel on travaillera le plus souvent, contient quatre dossiers :

- **Console** : toutes les commandes en mode console,
- **Http** : tout ce qui concerne la communication : contrôleurs, middlewares (il y a 4 middlewares de base qui servent à filtrer les requêtes HTTP) et le kernel,
- **Providers** : tous les fournisseurs de services (providers), il y en a déjà 5 au départ. Les providers servent à initialiser les composants.
- **User** : un modèle qui concerne les utilisateurs pour la base de données.

On retrouve ensuite :

- **Bootstrap** : scripts d'initialisation de Laravel pour le chargement automatique des classes, la fixation de l'environnement et des chemins, et pour le démarrage de l'application,
- **Public** : tout ce qui doit apparaître dans le dossier public du site : images, CSS, scripts...
- **Config** : toutes les configurations : applications, authentications, caches, bases de données, espaces de noms, emails, systèmes de fichier, session...
- **Database** : migrations et populations,
- **Ressources** : vues, fichiers de langage et assets (par exemple les fichiers Sass),
- **Routes** : la gestion des urls d'entrée de l'application,
- **Storage** : données temporaires de l'application : vues compilées, caches, clés de session...
- **Tests** : fichiers de tests,
- **Vendor** : tous les composants de Laravel et de ses dépendances (créés par Composer),

Il y a un certain nombre de fichiers dans la racine dont voici les principaux :

- **Artisan** : outil en ligne de Laravel pour des tâches de gestion,
- **Composer.json** : fichier de référence de composer,
- **Package.json** : fichier de référence de npm pour les assets,
- **Phpunit.xml** : fichier de configuration de phpunit (pour les tests unitaires),
- **.env** : fichier pour spécifier l'environnement d'exécution.

Pour la suite du projet, je rajoute deux dossiers : un dossier Repository et un dossier Service ainsi que Requests.

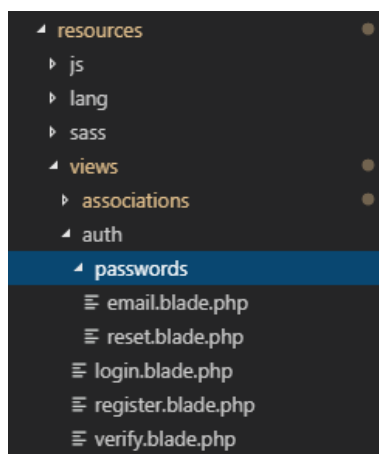
## CONSTRUCTION DE LA BASE DE DONNEES DU SITE

Mon site aura besoin d'un service d'authentification. J'utilise Laravel qui propose différents outils facilitant la construction d'un site.

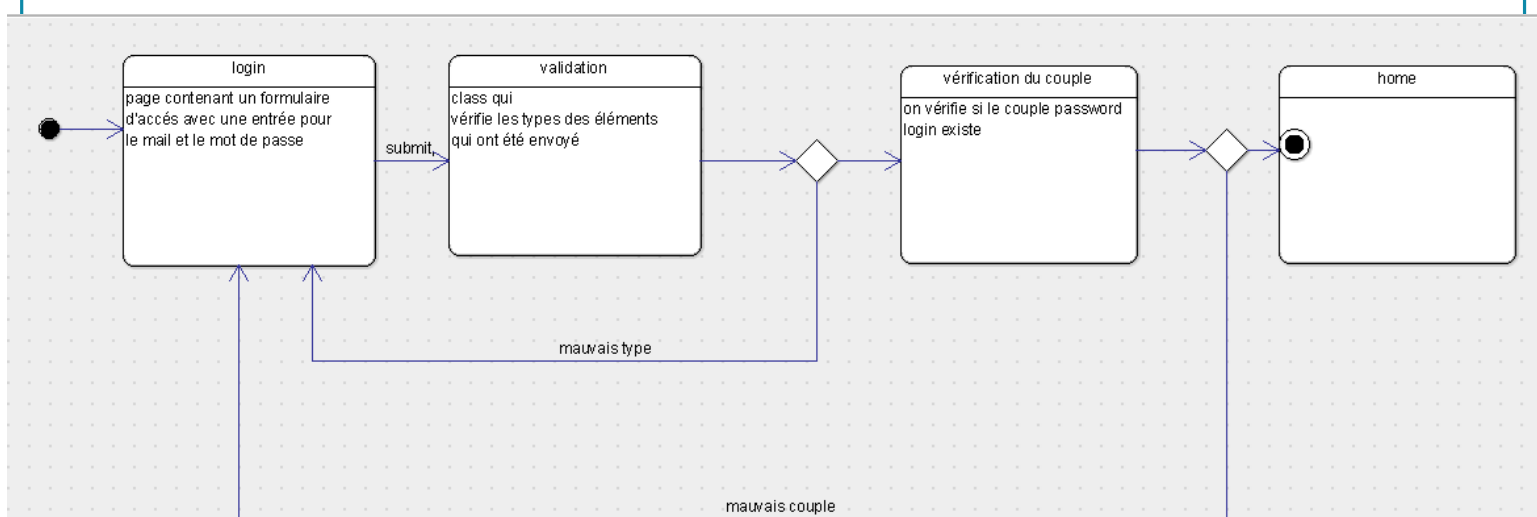
L'installation de ce service se fait en plusieurs étapes :

1) sur la commande on tape : `php artisan make:auth`

La commande génère un dossier dans view contenant les vues dont on aura besoin par la suite



Une fois l'installation faite, le schéma de transition du login se fait ainsi :



Ensuite, la construction des tables du site se déroule en fonction de l'évolution du site,

On se concentre d'abord sur deux acteurs : le donateur et le gestionnaire principal d'association.

Le gestionnaire pouvant créer des associations et entrer des dons,

On a donc 3 ou 4 tables :

- donateurs
- dons
- clients
- associations

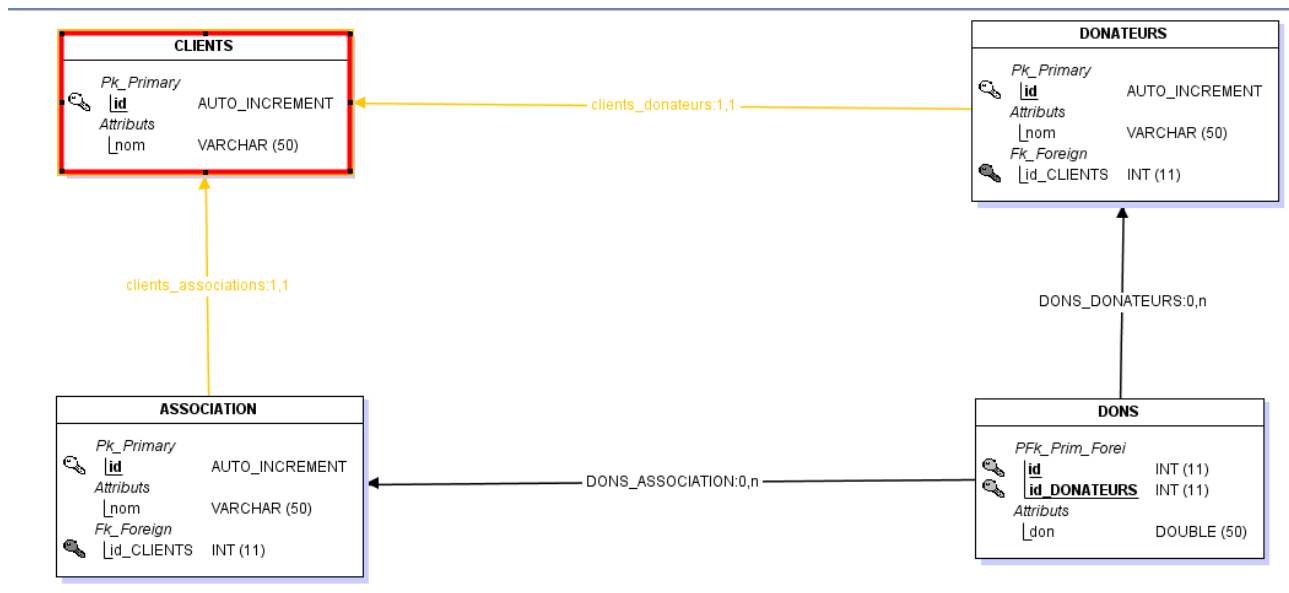
Dans un premier temps ma réflexion m'amène à avoir ce type de relation entre les différents protagonistes :

- donateurs et associations ont une relation de cardinal (n,n) donc une table de liaison où on peut ajouter les dons dans la table de liaison

- clients et donateurs ont une relation de cardinal (1,n)

- clients et association ont une relation de cardinal (1,n)

Ce qui nous donne le schéma suivant :

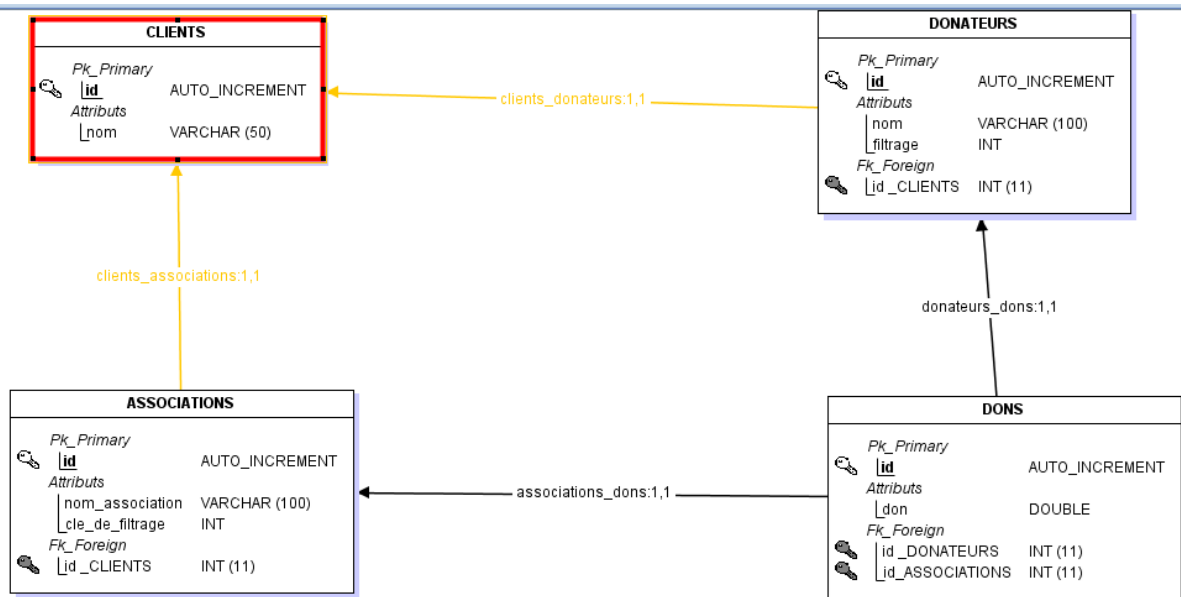


Dans un second temps, mon tuteur de stage me demande d'utiliser une autre façon de procéder : un « maskage de bit »

## OPERATEUR DE BITS APPLIQUE A NOTRE TRAVAIL ET A LARAVEL

Ainsi en appliquant les opérateurs sur les bits à notre travail, on peut se permettre de bien différencier le fait de faire un don et l'appartenance d'une personne à une association.

Cela donne alors :



Le nombre de tables reste le même par que précédemment, à la différence que l'appartenance d'un donateur à une association est indépendant du don, ce qui permet d'éviter de passer par la table « dons » pour rechercher l'appartenance d'un donateur à une association. Enfin la vérification de l'appartenance d'un donateur à une association se fait grâce à la clé de filtrage dans « associations » et filtrage dans « donateurs »,

Cette base établie reste à intégrer les tables dans la base de données ainsi que pouvoir les utiliser,

Ce sujet sera abordé dans la partie suivante ....



## INSTALLATION DES TABLES ET MIGRATION DANS LARAVEL

Il existe plusieurs types de bases de données, Pour mon travail je décide d'utiliser Mysql, dont je suis familier. L'installation des tables peut se faire manuellement à partir de Phpmyadmin, mais ceci est une mauvaise pratique pour le suivi du travail, et le travail en groupe. Laravel offre la possibilité de passer par les migrations. L'idée d'une migration est de pouvoir faire un suivi de l'évolution des tables et de pouvoir revenir en arrière le cas échéant.

L'installation des tables et l'utilisation des tables est expliqué sur le site de Laravel.

The screenshot shows the Laravel documentation page for generating migrations. The page title is "# Generating Migrations". It explains that to create a migration, you use the `make:migration` Artisan command. A code block shows the command: `php artisan make:migration create_users_table`. It states that the new migration will be placed in the `database/migrations` directory. It also mentions that the `--table` and `--create` options can be used to indicate the table name and whether to create a new table. Two code blocks show these options: `php artisan make:migration create_users_table --create=users` and `php artisan make:migration add_votes_to_users_table --table=users`. Finally, it notes that the `--path` option can be used to specify a custom output path.

**Traduction :** « pour utiliser une migration, utilisé l'Artisan commande `'make :migration'`

`php artisan make :migration create_users_table`

La nouvelle migration sera placée dans votre dossier : `database/migrations`. Chaque nom de fichier de migration contient une date qui permet à Laravel de déterminer l'ordre des migrations

Les options '`-table`' and '`-create`' peuvent être utilisées pour indiquer le nom de la table et si la migration crée une nouvelle table :

`Php artisan make :migration create_users_table --create=users`

`Php artisan make :migration add_votes_to_users_table --table=users`

*Si vous désirez, un chemin de sortie spécifique pour la génération de migration, vous pouvez utiliser option '- -path' quand vous exécutez la commande 'make :migration'. Le chemin donné devrait être relatif à votre dossier de base. »*

Ainsi, par exemple, la création du fichier permettant la génération de la table de nom « clients » se fait à partir de la commande :

« **php artisan make:migration create\_clients\_table --create=clients** », ce qui a pour effet de générer un fichier, après l'avoir rempli ceci donne :

```

2
3 use Illuminate\Support\Facades\Schema;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Database\Migrations\Migration;
6
7 class CreateClientsTable extends Migration
8 {
9     /**
10      * Run the migrations.
11      *
12      * @return void
13      */
14     public function up()
15     {
16         Schema::create('clients', function (Blueprint $table) {
17             $table->increments('id');
18             $table->string('client');
19             $table->timestamps();
20         });
21     }
22
23     /**
24      * Reverse the migrations.
25      *
26      * @return void
27      */
28     public function down()
29     {
30         Schema::dropIfExists('clients');
31     }

```

Une fois les premiers fichiers générés, les premières préparations de table sont faites, on peut passer à la création des tables à partir de la commande : « **php artisan:migrate** »

Seulement, un problème se pose alors : la commande donne une erreur de ce type :

La solution se trouve alors sur le site Starckoverflow :

https://stackoverflow.com/questions/42244541/laravel-migration-error-syntax-

Search...

Thanks for the suggestion @can-vural, I did. – absiddiqueLive Feb 15 '17 at 9:57

add a comment

25 Answers

active oldest votes

120

According to the [official documentation](#), you can solve this quite easily.

Add following code to **AppServiceProvider.php** (/app/Providers/AppServiceProvider.php)

```
use Illuminate\Support\Facades\Schema; //NEW: Import Schema

function boot()
{
    Schema::defaultStringLength(191); //NEW: Increase StringLength
}
```

MySQL reserves always the max amount for a UTF8 field which is 4 bytes so with 255 + 255 with your DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4\_unicode\_ci, you are over the 767 max key length limit. By @scaisedge

share improve this answer

edited 2 days ago

answered Feb 15 '17 at 9:52

Chuck Le Butt 28.3k ● 46 ● 151 ● 227

absiddiqueLive 3,468 ● 2 ● 13 ● 26

Be careful about this solution. If you index email fields for example, stored emails can only have a max length

**Traduction :** « selon la documentation officielle, vous pouvez résoudre le problème assez rapidement. Ajoutez cette suite de code dans

AppServiceProvider.php(/app/Providers/AppServiceProvider.php)

Use Illuminate\Support\Facades\Schema ; //Nouvelle import de Schema

function boot()

{

Schema ::defaultStringLength(191) ;// Nouvelle majoration StringLength

}

Mysql réserve toujours un montant maximum pour des donnée UTF-8 lequel est 4 bytes. Aussi avec 255 + 255, avec vos jeux de caractère par défaut utf8mb4 ajouter à utf8mb4\_unicode\_ci ; vous êtes au-dessus de la taille maximal limite (par @scaidage) »

De plus la solution se trouve dans la documentation de Laravel :

### Index Lengths & MySQL / MariaDB

Laravel uses the `utf8mb4` character set by default, which includes support for storing "emojis" in the database. If you are running a version of MySQL older than the 5.7.7 release or MariaDB older than the 10.2.2 release, you may need to manually configure the default string length generated by migrations in order for MySQL to create indexes for them. You may configure this by calling the `Schema::defaultStringLength` method within your `AppServiceProvider`:

```
use Illuminate\Support\Facades\Schema;

/**
 * Bootstrap any application services.
 *
 * @return void
 */
public function boot()
{
    Schema::defaultStringLength(191);
}
```

Alternatively, you may enable the `innodb_large_prefix` option for your database. Refer to your database's documentation for instructions on how to properly enable this option.

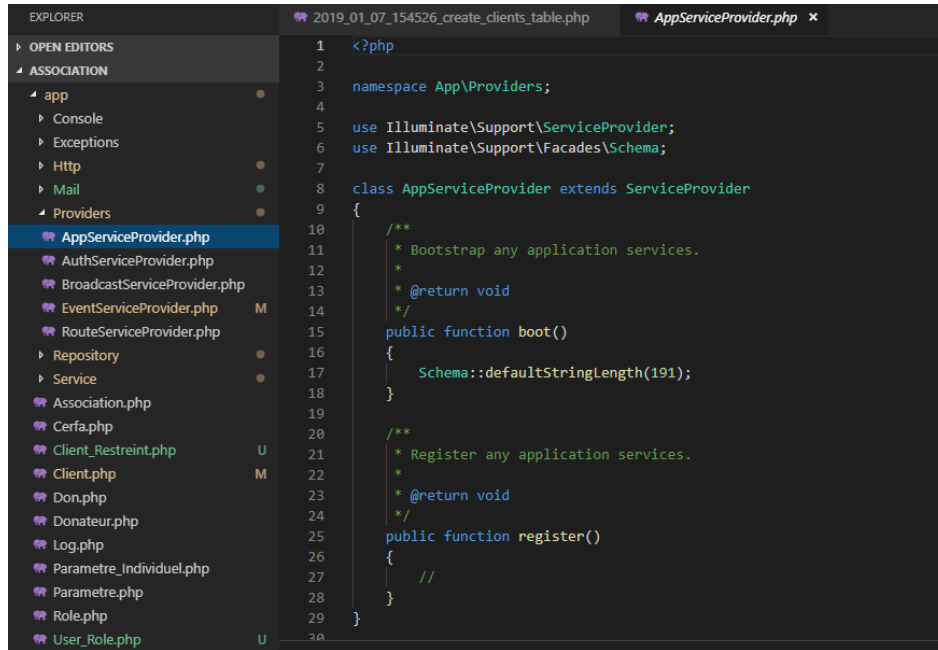
### Traduction : « longueur d'index et Mysql/MariaDB

Laravel utilise le jeu de caractère `utf8mb4` par défaut, qui inclus un soutien pour ranger les 'emojis' dans les bases de données. Si vous travaillez avec une version de Mysql plus vieille que 5.7.7 ou MariaDB plus vieille que 10.2.2, vous pouvez avoir besoin de configurer manuellement la taille par défaut du string généré par les migrations pour Mysql pour créer l'index pour eux. Vous pouvez configurer ça en appelant la méthode `Schema::defaultStringLength` à l'intérieur de votre fichier `AppServiceProvider`

Use .....

Une solution alternative, vous pourriez effacer l'option `innodb_large_prefix` de votre base de données. Référez-vous à la documentation de votre base de données pour les instructions sur « comment effacer proprement ces options. »

Ainsi en modifiant le fichier `appServiceProvider`, en ajoutant `Schema::defaultStringLength(191)` à la fonction `boot`, on solutionne le problème



```
1 <?php
2
3 namespace App\Providers;
4
5 use Illuminate\Support\ServiceProvider;
6 use Illuminate\Support\Facades\Schema;
7
8 class AppServiceProvider extends ServiceProvider
9 {
10     /**
11      * Bootstrap any application services.
12      *
13      * @return void
14      */
15     public function boot()
16     {
17         Schema::defaultStringLength(191);
18     }
19
20     /**
21      * Register any application services.
22      *
23      * @return void
24      */
25     public function register()
26     {
27         //
28     }
29 }
```

## VALIDATION DANS LARAVEL

## # Introduction

Laravel makes it easy to protect your application from [cross-site request forgery](#) (CSRF) attacks. Cross-site request forgeries are a type of malicious exploit whereby unauthorized commands are performed on behalf of an authenticated user.

Laravel automatically generates a CSRF "token" for each active user session managed by the application. This token is used to verify that the authenticated user is the one actually making the requests to the application.

Anytime you define an HTML form in your application, you should include a hidden CSRF token field in the form so that the CSRF protection middleware can validate the request. You may use the `@csrf` Blade directive to generate the token field:

```
<form method="POST" action="/profile">
    @csrf
    ...
</form>
```

The `VerifyCsrfToken` middleware, which is included in the `web` middleware group, will automatically verify that the token in the request input matches the token stored in the session.

### CSRF Tokens & JavaScript

When building JavaScript driven applications, it is convenient to have your JavaScript HTTP library automatically attach the CSRF token to every outgoing request. By default, the `resources/js/bootstrap.js` file registers the value of the `csrf-token` meta tag with the Axios HTTP library. If you are not using this library, you will need to manually configure this behavior for your application.

## # Excluding URIs From CSRF Protection

Sometimes you may wish to exclude a set of URIs from CSRF protection. For example, if you are using [Stripe](#) to process payments and are utilizing their webhook system, you will need to exclude your Stripe webhook handler route from CSRF protection since Stripe will not know what CSRF token to send to your routes.

Laravel facilite la protection de votre application contre les attaques de type CSRF (falsification de requêtes) entre sites. Les falsifications de requêtes intersites sont un type d'exploit malveillant par lequel des commandes non autorisées sont exécutées pour le compte d'un utilisateur authentifié.

Laravel génère automatiquement un "jeton" CSRF pour chaque session d'utilisateur actif gérée par l'application. Ce jeton est utilisé pour vérifier que l'utilisateur authentifié est celui qui envoie réellement les demandes à l'application.

Chaque fois que vous définissez un formulaire HTML dans votre application, vous devez inclure un champ de jeton CSRF masqué dans le formulaire afin que le middleware de protection CSRF puisse valider la demande. Vous pouvez utiliser la directive @csrf Blade pour générer le champ de jeton :

```
<form method= «POST» action= «/profile »>@csrf ... </form>
```

Le middleware VerifyCsrfToken, qui fait partie du groupe de middleware Web, vérifie automatiquement que le jeton contenu dans l'entrée de requête correspond au jeton stocké dans la session.

### **Jetons CSRF et JavaScript**

Lors de la création d'applications JavaScript, il est pratique que votre bibliothèque HTTP HTTP attache automatiquement le jeton CSRF à chaque demande sortante. Par défaut, le fichier ressources / js / bootstrap.js enregistre la valeur de la balise méta csrf-token dans la bibliothèque HTTP Axios. Si vous n'utilisez pas cette bibliothèque, vous devrez configurer manuellement ce comportement pour votre application.

## ÉTUDE DES CAS D'INSCRIPTIONS D'UNE ASSOCIATION ET D'UN DONATEUR PAR UN CLIENT.

Nous avons vu précédemment que chaque association était associée à une clé de filtrage qui lui est propre. L'une des difficultés de l'insertion d'une association dans la base de donnée est de savoir quelle clé de filtrage lui associer. De plus le calcul de la clé de l'association d'un client X ne doit prendre en compte que les associations de ce client particulier,

En effet dans le cas contraire on pourrait arriver à une clé bien trop grande pour la base de données :

On doit éviter ce type de tableau :

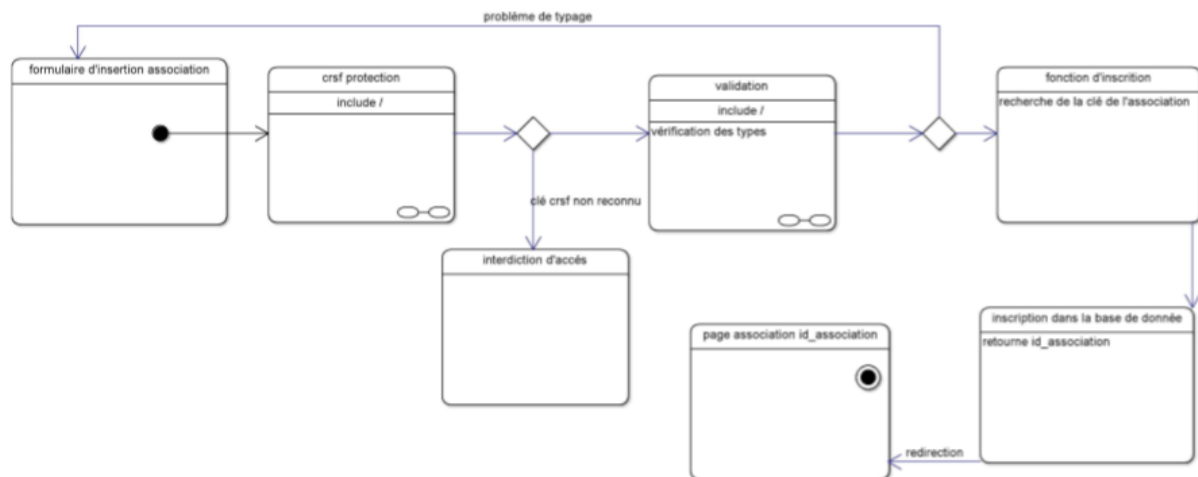
client_id	association	cle_de_filtrage
1	association_1_client_1	1
1	association_2_client_1	2
2	association_1_client_2	4
2	association_2_client_2	8
1	association_3_client_1	16
1	association_4_client_1	32
2	association_3_client_1	64

On veut arriver à ce type de tableau

client_id	association	cle_de_filtrage
1	association_1_client_1	1
1	association_2_client_1	2
2	association_1_client_2	1
2	association_2_client_2	2
1	association_3_client_1	4
1	association_4_client_1	8
2	association_3_client_1	4



Le schéma état transition de l'inscription peut se voir ainsi :



Dans un premier temps l'inscription passe par un formulaire en html5. Les entrées de formulaire en Html5 disposent de différents attributs qui permettent de sécuriser un formulaire

On a par exemple les balises :

- <input type='mail'> ;
- <input type='text'> ;
- <input type='date'>

Qui obligent l'utilisateur à entrer un certain type de données dans les entrées de formulaire ; de plus on dispose de l'attribut « required » qui oblige à remplir les entrées de formulaire. Html5 dispose d'autre attributs de sécurité que je n'ai pas intégré pour le moment. La sécurité html5 ne sert pas en elle-même à sécuriser entièrement site ; en effet un internaute bien informé peut par exemple utiliser l'inspecteur de site pour changer le formulaire, c'est la raison pour laquelle on travaillera la sécurité du site dans la partie back-end. Partie sur laquelle on reviendra ultérieurement

Au niveau du design j'utilise le framework Bootstrap 4 ; il permet d'avoir un bon rendu et, dans un premier temps, de travailler rapidement le design.

Le code html du formulaire se présente de cette manière :

```

<!-- debut du formulaire -->
<!-- donnée envoyé vers association.store en post -->
<div class="card">
  <div class="card-body">
    <form action="{{ route('association.store') }}" method="POST">
      @csrf
    <div class="form-group">
      <label for="association">association</label>
      <input type="text" class="form-control" name="association"
        id="association" placeholder="entrée le nom de votre association" value="{{ old('association') }}" required>
    </div>
    <div class="form-group">
      <label for="adresse">adresse</label>
      <input type="text" class="form-control" name="adresse"
        id="adresse" placeholder="entrée l'adresse de votre association" value="{{ old('adresse') }}" required>
    </div>
    <div class="form-group">
      <label for="ville">ville</label>
      <input type="text" class="form-control" name="ville"
        id="ville" placeholder="entrée la ville de votre association" value="{{ old('ville') }}" required>
    </div>
    <div class="form-group">
      <label for="code_postal">code postal</label>
      <input type="text" class="form-control" name="code_postal"
        id="code_postal" placeholder="entrée le code postal de votre association" value="{{ old('code_postal') }}" required>
    </div>
  </div>
</div>

```

@csrf est une balise obligatoire dans tout formulaire Laravel. Elle permet à Laravel de se protéger d'attaques du type Cross-site request forgery.

L'objet de cette attaque est de transmettre à un utilisateur authentifié, une requête HTTP falsifiée qui pointe sur une action interne au site, afin qu'il l'exécute sans en avoir conscience et en utilisant ses propres droits. L'utilisateur devient donc complice d'une attaque sans même s'en rendre compte. L'attaque étant actionnée par l'utilisateur, un grand nombre de systèmes d'authentification sont contournés.

(source: [https://fr.wikipedia.org/wiki/Cross-site\\_request\\_forgery](https://fr.wikipedia.org/wiki/Cross-site_request_forgery))

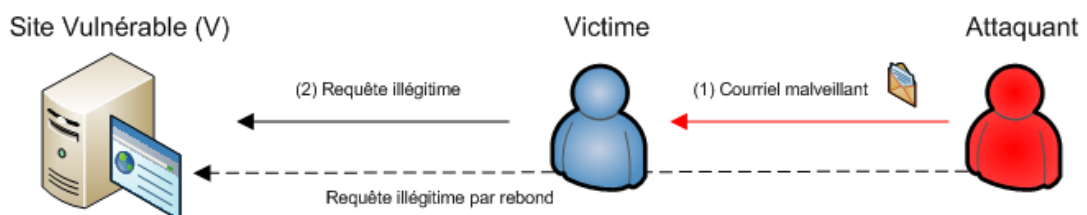
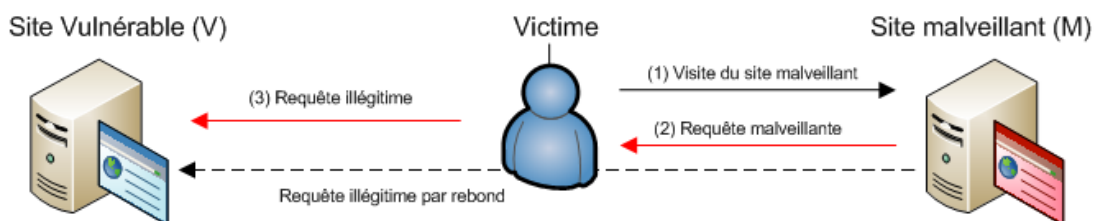


Figure 1: L'attaquant envoie un courriel qui provoque l'envoi d'une requête illégitime



(source : <https://www.cert.ssi.gouv.fr/information/CERTA-2008-INF-003/>) Laravel génère une clé d'authentification sur le formulaire, qui lui permet de reconnaître le formulaire et de contrecarrer ce type d'attaques ,

Autre étape dans la sécurité, la validation de formulaire proposé par Laravel,

```
* @return bool
*/
public function authorize()
{
    return true;
}

/**
 * Get the validation rules that apply to the request.
 *
 * @return array
 */
public function rules()
{
    return [
        'association'=>'required|alpha',
        'adresse'=>'required',
        'ville'=>'required|alpha|max:30',
        'code_postal'=>'required|alpha_num',
        'pays'=>'required|alpha|max:20',
        'telephone_1'=>'required|numeric|max:12',
        'adresse_email'=>'required|email',
        'rcs'=>'required',
        'ape'=>'required',
        'siret'=>'required|max:12',
    ];
}
```

La validation se fait par une classe externe qui permet de filtrer les bonnes données des mauvaises

Dans la fonction « rule ») on donne le type de données qui sont permises, si les types ne sont pas permis on fait une redirection vers le formulaire. Laravel dispose de types de filtrage tels que « alpha, alpha\_numeric, email », etc. Le développeur peut bien sûr intégrer d'autre type » pour améliorer la sécurité du site.

Au niveau du Controller, on vérifie que l'utilisateur a bien un « id client », ce qui confirmera son rôle en tant que client, (ceci est aussi vérifié à l'aide de la table « rôle » est la valeur du rôle qui est inséré en session)

Une fois que les données sont validées, que le rôle a été vérifié et qu'on a récupéré l'« id client » on cherche la dernière clé de filtrage des associations appartenant aux clients ayant soumis le formulaire,

```

public function findMaxCle_de_filtrageByClient_id($client_id)
{
    return DB::table('associations')
        ->where('client_id',$client_id)
        ->max('cle_de_filtrage')
        ;
}

```

si le client n'a pas de fonction la fonction retourne 0 et donc la clé de filtrage est 1

```

        $cle_de_filtrage = $this->associations->findMaxCle_de_filtrageByClient_id($id);
        if (!empty($cle_de_filtrage))
        {
            $cle_de_filtrage = 2*$cle_de_filtrage;

$id_association = $association->addAssociation([
    'client_id'=>$id,
    'nom_association'=>$request->input('association'),
    'cle_de_filtrage'=>$cle_de_filtrage,
    'ville'=>$request->input('ville'),
    'adresse'=>$request->input('adresse'),
    'code_postal'=>$request->input('code_postal'),
    'pays'=>$request->input('pays'),
    'telephone_1'=>$request->input('telephone_1'),
    'adresse_email'=>$request->input('adresse_email'),
    'ape'=>$request->input('ape'),
    'rccs'=>$request->input('rccs'),
    'siret'=>$request->input('siret'),
]);
}else{
    $id_association = $association->addAssociation([
        'client_id'=>$id,
        'nom_association'=>$request->input('association'),
        'cle_de_filtrage'=>1,
        'ville'=>$request->input('ville'),
        'adresse'=>$request->input('adresse'),
        'code_postal'=>$request->input('code_postal'),

```

## INSERTION D'UN DONATEUR PAR LE CLIENT

L'insertion d'un donateur est un peu plus complexe que l'insertion d'une association.

Plusieurs choses sont à prendre en compte dans cette insertion, en effet un donateur peut appartenir à plusieurs clients, les informations qui appartiennent à un client ne sont pas étanches et ne doivent pas être modifiables. Pour gérer cela on est obligé de bien mettre « l'id client » à chaque donateur mais aussi faire des répétitions d'informations ; c'est à dire que dans notre table « donateurs » on pourra avoir :

nom	prénom	client_id	user_id
Gérard	Oury	1	5
Longtarin	Gaston	1	6
Gérard	Oury	2	5

Ceci assure une étanchéité des informations.

Autre chose à prendre en compte l'inscription d'un utilisateur se fait à partir d'un email unique, il ne peut pas avoir dans la table « user » qui gère les connexions deux fois la même adresse e-mail.

De plus comme nous avons vu précédemment un filtre est associé au donateur qui permet d'associer les associations au donateur qui est inscrit

Au niveau de la validation et de la gestion d'envoi des données de formulaire ceci se fait de manière analogue à précédemment

L'inscription en tant que soi a donc plusieurs étapes :

- première étape :

Récupération des données et calcul de la valeur du filtre associé au donateur

- deuxième étape :

Inscription du donateur en tant qu'utilisateur

```

    }
    $userR = new UserRepository();
    // création de l'utilisateur
    $user_id = $userR->IdUserEmail(['name'=>$nom, 'password'=>Hash::make('123456'), 'email'=>$email]);

```

Cette étape se fait en deux fois :

```

public function IdUserEmail(array $array)
{
    // on recherche si un utilisateur à l'email donné par le formulaire
    $userCheck = DB::table('users')
    ->where('email',$array['email'])->first();
    // si on a pas trouvé d'utilisateur
    if(empty($userCheck->id))
    {
        // on crée l'utilisateur
        $user = /*DB::table('users')*/User::create($array);
        /*    ->insertGetId($array)*/

        // un email est envoyé pour que le donateur valide sont inscription
        $user->sendEmailVerificationNotification();
        // on retourne l'id de l'utilisateur
        return $user->id;
    }else{
        // on retourne l'id de l'utilisateur
        return DB::table('users')
        ->where('email',$array['email'])
        ->select('id')
        ->first()->id;
    }
}

```

- Dans un premier temps, on vérifie s'il existe un utilisateur avec l'e-mail concerné
- si l'utilisateur existe déjà on renvoie l'id de l'utilisateur concerné
- si l'utilisateur n'existe pas on l'inscrit dans la table « users » et on envoie un email de validation de compte pour le donateur (l'envoi d'email sera abordé par la suite). ; cela fait, on renvoie l'id de l'utilisateur inscrit.

Après, il reste trois étapes pour l'inscription globale du donateur

```

$donateurCheck = DB::table('donateurs')
    ->where('user_id',$array['user_id'])
    ->where('client_id',$array['client_id'])
    ->first()
;
// on verifie que le client n'a pas déjà l'utilisateur avec l'id user
if(empty($donateurCheck->id))
{
    // renvoie l'id de l'inscription
    return DB::table('donateurs')->insertGetId(
        [
            'nom'=>$array['nom'],
            'prenom'=>$array['prenom'],
            'adresse'=>$array['adresse'],
            'ville'=>$array['ville'],
            'code_postal'=>$array['code_postal'],
            'telephone_1'=>$array['telephone_1'],
            'telephone_2'=>$array['telephone_2'],
            'date_de_naissance'=>$array['date_de_naissance'],
            'commentaire'=>$array['commentaire'],
            'client_id'=>$array['client_id'],
            'user_id'=>$array['user_id'],
            'active'=>1,

```

On vérifie si le client a déjà le donateur concerné. Si le client a déjà le donateur on renvoie 0 sinon on inscrit le donateur et on renvoie son id.

Enfin on donne le rôle de donateur à l'utilisateur en ayant vérifié que ce rôle n'existait pas déjà pour lui.

## GESTION DU MASQUAGE DANS LARAVEL

La gestion du masquage a demandé plusieurs recherches dans Laravel, en effet il n'existe pas d'objet spécifique pour la gestion du masquage.

Le masquage est utile dans le cas de la recherche d'associations appartenant à un donateur

-

« DB::select » est la première manière pour utiliser la méthode des filtrage dans Laravel. Elle n'est pas conseillée car difficilement réutilisable et modifiable.

Dans la fonction on remarque que le filtrage et le client\_id est la condition de jointure entre associations et donateur.

En testant la requête sur phpmyadmin avec pour donateur id 1 et client id 1, on a

```
SELECT associations.nom_association, associations.adresse, associations.ville, associations.id as association_id, donateurs.id as donateur_id FROM
associations INNER JOIN donateurs ON (associations.cle_de_filtrage & donateurs.filtrage_association = associations.cle_de_filtrage AND
associations.client_id = donateurs.client_id) WHERE donateurs.id = 1 AND associations.client_id = 1
```

☐ Profilage [Éditer en ligne] [Éditer] [Expliquer SQL] [Créer le code source PHP] [Actualiser]

☐ Tout afficher | Nombre de lignes : 25 | Filtrer les lignes : Chercher dans cette tabl | Trier sur l'index: Aucun(e)

+ Options

nom_association	adresse	ville	association_id	donateur_id
ASSOCIATION DES CLOWNS PAS RIGOLO	5 ure du clown	bourg en bresse	1	1
ASSOCIATION DES CLOWNS RIGOLO	5 rue du rire	hipsbourg	2	1

La deuxième manière de gérer les masquages a été faite plus tard, elle utilise les objets beaucoup plus proprement.

Comme dit précédemment Laravel n'est pas futilisé en première intention pour les « bitwise operator ». Il existe quelques pistes d'utilisation des « bitwise » sur internet : par exemple sur « starckoverflow » j'ai pu trouver une réponse évoquant ce sujet :



The screenshot shows a Stack Overflow page for the question "bitwise-and-condition-in-a-lara". The URL is <https://stackoverflow.com/questions/34392332/bitwise-and-condition-in-a-lara>. The page has 4,930 views, 1 answer, 24 votes, and 47 comments. The question is about using bitwise AND in Eloquent where clauses. The answer, by Victor Axelsson, explains that Eloquent doesn't support fancy where clauses like bitwise AND, but you can use `whereRaw` to execute raw where clauses. It provides an example: `$belongsToMany->whereRaw('(type & 1<<25) != 0')`. It also shows how to debug SQL queries using `DB::enableQueryLog()` and `dd(DB::getQueryLog())`. The answer is marked as the best answer.

### Traduction :

« Vous pouvez utiliser 'whereRaw' pour exécuter des clauses where brut, complexes

Eloquent ne supporte pas plusieurs clauses atypiques, elle supporte les clauses du type like, = ou standard. Je suggère que vous jetiez un coup d'œil à la documentation.

Dans votre cas j'utiliserais probablement quelque chose comme :

```
$belongsToMany->whereRaw('(type & 1<<25) != 0')
```

Et ajoutez à cela votre select clause

Si vous voulez déboguer votre sql vous pouvez utiliser :

```
DB::enableQueryLog();
```

```
dd(DB::getQueryLog());
```

De cette façon vous pouvez aussi voir, si vous mettez, n'importe quoi comme bitmasks à l'intérieur de la clause "où" si c'est remplacé par un '=' »

Ainsi la méthode `whereRaw` peut aider dans les opérateurs de bit. Cependant ces opérateurs sont utilisés dans des jointures, la seule manière de les faire, sauf en utilisant la méthode « `select()` » qui permet d'intégrer une requête sql directement, et de passer par la méthode `join()`.

La documentation de Laravel peut aider à résoudre ce problème :

## # Raw Expressions

Sometimes you may need to use a raw expression in a query. To create a raw expression, you may use the `DB::raw` method:

```
$users = DB::table('users')
    ->select(DB::raw('count(*) as user_count, status'))
    ->where('status', '<>', 1)
    ->groupBy('status')
    ->get();
```



Raw statements will be injected into the query as strings, so you should be extremely careful to not create SQL injection vulnerabilities.

L'avantage de `raw()` par rapport à `whereRaw` est que son utilisation est beaucoup plus souple

Ceci ne résout pas entièrement notre problème, puisque pour travailler avec notre cas nous avons besoin de deux conditions dans la jointure

Ceci est résolu toujours à l'aide de la documentation de Laravel :

### Advanced Join Clauses

You may also specify more advanced join clauses. To get started, pass a `Closure` as the second argument into the `join` method. The `Closure` will receive a `JoinClause` object which allows you to specify constraints on the `join` clause:

```
DB::table('users')
    ->join('contacts', function ($join) {
        $join->on('users.id', '=', 'contacts.user_id')->orOn(...);
    })
    ->get();
```

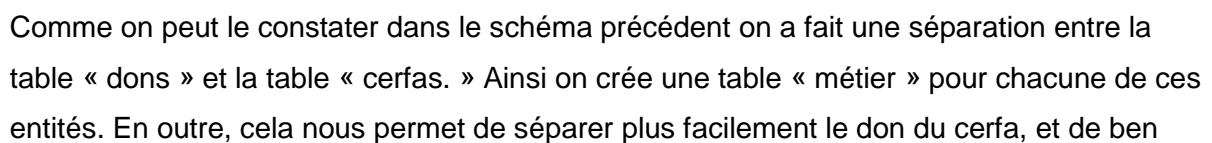
If you would like to use a "where" style clause on your joins, you may use the `where` and `orWhere` methods on a join. Instead of comparing two columns, these methods will compare the column against a value:

Dans notre cas la fonction permettant de trouver les données d'un donateur avec une association sera celle-ci :

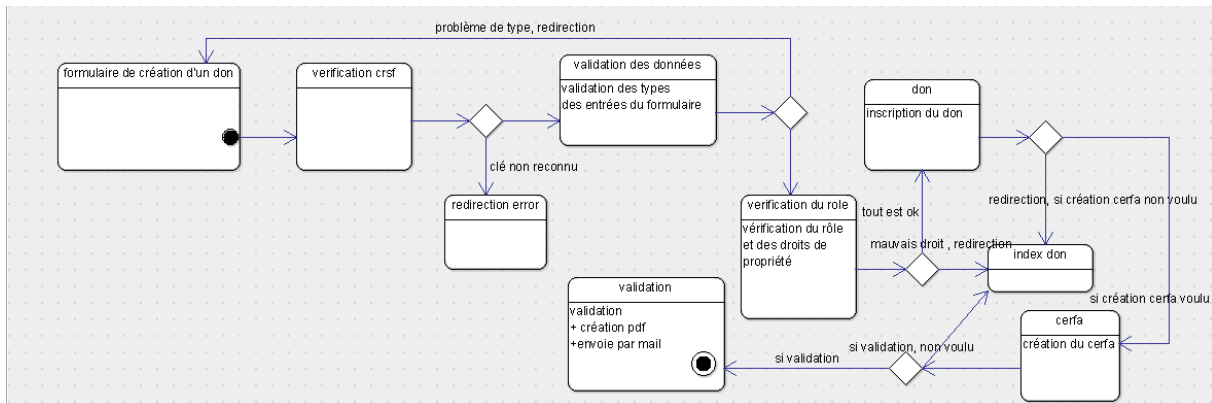
```
public function findAssociationBydonateur_idAndAssociationId($donateur_id,$id_association)
{
    $select = $this->selectVueDonateur();
    $result = DB::table('associations')
        ->where('associations.id',$id_association)
        ->join('donateurs',function($join)
        {
            $join
                ->on(DB::raw(
                    'associations.cle_de_filtrage & donateurs.filtrage_association'),
                    '=',
                    'associations.cle_de_filtrage');
            $join->on('donateurs.client_id','=','associations.client_id');
        })
        ->where('donateurs.id',$donateur_id)
        ->select($select);
    return $result->get();
}
```

Comme vu précédemment le cerfa doit contenir certain points clé tel que :

- Autre point, lors de la création du cerfa, celui-ci doit être envoyé au donateur par mail sous format pdf. La création d'un cerfa demande donc de savoir générer un document pdf et savoir l'envoyer par mail. Les relations entre tables à ce stade du travail sont de cet ordre :



marquer les étapes de la création d'un cerfa.



La création du cerfa demande donc plusieurs étapes, parmi celle-ci on retrouve dans la fonction d'insertion :

- la recherche de propriété des associations et donateurs, selon le rôle de l'utilisateur :

```
if($role->contains('role','client'))
{
    //verification de la propriété
    $id_client = DB::table('clients')
    ->where('clients.id',$client_id)
    ->join('donateurs','donateurs.client_id','=', 'clients.id')
    ->join('associations','associations.client_id','=', 'clients.id')
    ->where('donateurs.id',$id_donateur)
    ->where('associations.id',$id_association)
    ->select('clients.id')
    ->pluck('clients.id')
    ->first()
};
}elseif ($role->contains('role','client_restreint')) {
    $id_client = DB::table('clients_restreints')
    ->where('clients_restreints.id',$client_id)
    ->join('associations',function($join){
        $join->on(DB::raw('clients_restreints.filtrage_association_client_restreint&associations.cle_de_filtrage'),
        '=',
        'associations.cle_de_filtrage');
        $join->on('associations.client_id','=', 'clients_restreints.client_id');
    })
    ->where('associations.id',$id_association)
    ->join('donateurs',function($join){
        $join->on(DB::raw('donateurs.filtrage_association&associations.cle_de_filtrage'),
        '=',
        'associations.cle_de_filtrage');
        $join->on('associations.client_id','=', 'donateurs.client_id');
    })
    ->where('donateurs.id',$id_donateur)
    ->select('clients_restreints.id')
    ->pluck('clients_restreints.id')
    ->first();
}
```

Si l'utilisateur est un client, on recherche si l'id du client est bien dans la ligne du donateur et de l'association Si l'utilisateur est un client restreint on passe indirectement par le client id

propriétaire du client restreint et on passe par le maskage pour vérifier la propriété des associations et des donateurs aux clients restreints.

Une fois accomplie l'étape de vérification on peut passer à l'inscription du don qui ne demande pas de commentaire supplémentaire.

L'étape suivante consiste à créer et valider le cerfa si voulu :

```
if(!empty($array['creation_cerfa']))&&($array['creation_cerfa']=='1'))
{
    // initialisation du tableau pour la création du cerfa
    $tab = array('edit'=>'voici le cerfa','don_id'=>$don_id);
    // on utilise la classe CerfaRepository
    $cerfas = new CerfaRepository();
    // on utilise la méthode addCerfa de l'objet crée précédemment
    // pour la création du cerfa. la méthode renvoie l'id de la ligne crée

    if(!empty($don_id))
    {
        // ajout de cerfa en fonction du role
        $cerfa_id = $cerfas->bigAddCerfa($tab,$client_id,$role);

        Log::info('cerfa id envoyé par la fonction bigaddcerfa :'.$cerfa_id);
    }
    if(!empty($array['validation_cerfa']))&&($array['validation_cerfa']=='2'))
    {
        if(!empty($cerfa_id))
        {
            // validation du cerfa selon role |
            $valid = $cerfas->bigValidationCerfa($cerfa_id, $client_id,$role);
            // renvoi les id des données créées avec message
            return array('don_id'=>$don_id,'cerfa_id'=>$cerfa_id,'message'=>'cerfa et don crée ainsi ');
        }
    }
    else
    {
        // renvoi les id des données créées avec message
        return array('don_id'=>$don_id,'cerfa_id'=>$cerfa_id,'message'=>'cerfa et don crée');
```

\$array['creation\_cerfa'] et \$['validation\_cerfa'] sont des données envoyées par l'utilisateur à partir de <input type="checkbox" name='creation\_cerfa'>,

La validation du cerfa requiert, elle aussi, plusieurs étapes :

```

public function BigvalidationCerfa($cerfa_id, $client_id,$role)
{
    if ($this->BigClientisProprietaire($cerfa_id, $client_id,$role))
    {
        Log::info('cerfa_id à l entrée de bigvalidation : '.$cerfa_id);
        try
        {
            if(!$this->isValid($cerfa_id))
            {
                $oldNo = $this->dernierNumero($cerfa_id) + 1;
                Log::info('dernier numero : '.$oldNo);

                $cerfavar = Cerfa::where('cerfas.id',$cerfa_id);
                $cerfa = $cerfavar
                    ->update(
                        ['confirmed'=>1,
                        'oldNo'=>$oldNo,
                        'date_signature'=>\date("Y-m-d")]
                    );

                //
                \App\Don::where('cerfa_id',$cerfa_id)
                    ->update(array('validate'=>1));
            }
        }
    }
}

```

La fonction pouvant être utilisée indépendamment de la création du don, celle-ci vérifie la propriété du cerfa de l'utilisateur.

Puis on recherche le dernier numéro du cerfa de l'association de l'utilisateur :

```

public function dernierNumero($cerfa_id)
{
    Log::info('cerfa id entrée dernier numéro : '.$cerfa_id);

    $association_id = DB::table('cerfas')->where('cerfas.id',$cerfa_id)
        ->join('dons','dons.cerfa_id','=', 'cerfas.id')
        ->select('dons.association_id')
        ->first()
        ->association_id;

    Log::info('association_id trouvé : '.$association_id);

    $max = DB::table('cerfas')
        ->join('dons','dons.cerfa_id','=', 'cerfas.id')
        ->where('dons.association_id',$association_id)
        ->select(DB::raw('max(cerfas.oldNo) as num'))
        ->first()
        ->num;

    if(empty($max))
    { Log::info('max est vide : '.$max);
      return 0;
    }else{
      Log::info('max n est pas vide : '.$max);
      return $max;
    }
}

```

Cela terminé, le cerfa est validé, le pdf généré et l'email envoyé :

```

if($cerfa == 1)
{
    // generation du pdf
    $pdfs = new PdfService();
    $pdf = $pdfs->generatePdf(array('cerfa_id'=>$cerfa_id,'client_id'=>$client_id,'role'=>$role));
    // envoi du cerfa par mail
    //recuperation de l'email
    $email_donneur = $pdf['email'];
    //recupertaion du pdf
    $pdf_a_envoye = $pdf['pdf'];
    // preparation de l'email
    $data = new Contact([
        'nom' => 'association : ',
        'email' => $email_donneur,
        'message' => "ceci est un mail automatique de la part de l'association : ".$pdf['association'].",
        un cerfa , un grand merci pour votre don",
        'document'=> $pdf_a_envoye->output()
    ]);
    // envoi de l'email
    Mail::to('administrateur@chezmoi.com')
    ->send($data);
}

return $cerfa;
}else{
    Log::info('cerfa deja valide');
    return 0;
}
}

```



## GENERATION DU CERFA EN PDF ET ENVOI D'EMAIL

La génération d'un pdf sur Laravel n'est pas chose compliquée mais elle nécessite de faire une recherche de l'outil approprié. La plupart des réponses pour la génération de pdf m'ont dirigé vers « Dompdf ». L'ajout de « Dompdf » se fait à partir de Composer et de la commande :

« Composer require barryvdh/laravel-dompdf »

Une fois cela fait on ajoute dans le fichier « config/app.php »

So go to the **config >> app.php** and add the following configuration.

```
'providers' => [
    ....
    Barryvdh\DomPDF\ServiceProvider::class,
],
'aliases' => [
    ....
    'PDF' => Barryvdh\DomPDF\Facade::class,
],
```

Après, l'outil de génération de pdf est prêt à être utilisé. (plusieurs tuto aident à la compréhension de Dompdf, mais la plupart des options de cette outil se trouvent sur le lien suivant : <https://github.com/barryvdh/laravel-dompdf> )

La génération du cerfa demande dans un premier temps la récupération des données lié au cerfa :

```
$cerfas = new CerfaRepository();
$cerfa = $cerfas->bigShowCerfa($array["cerfa_id"],$array["client_id"],$array["role"]);
$parametres = new ParametreIndividuelRepository();
if($array['role']->contains("role","client"))
{
    $parametreTexte = $parametres->parametreIndividuel(2,$array['client_id']);
    $parametreImage = $parametres->parametreIndividuel(1,$array['client_id']);
}elseif ($array['role']->contains("role","client_restreint")){
    $parametreTexte = $parametres->parametreIndividuelByIdClientRestreint(2,$array["client_id"]);
    $parametreImage = $parametres->parametreIndividuelByIdClientRestreint(1,$array["client_id"]);
}
```

\$cerfa contient les données relatives au cerfa lui-même et la variable \$parametre contient les paramètres de l'utilisateur tels que l'image de la signature du propriétaire.

Une fois ces données récupérées on peut commencer à créer le pdf

```
$pdf = App::make('dompdf.wrapper');
if(!empty($cerfa))
{
    if($cerfas->isValid($array["cerfa_id"]))
    {
        $lien2 = public_path()."/images/cerfa2.jpeg";
        $lienImage = public_path()."/thumbs/".$parametreImage->option_image_individuel;
        $pdf->loadHTML("<html>
        <meta http-equiv='Content-Type' content='text/html; charset=utf-8'/>
        <title>cerfa don </title>
        <style>

            p{
                text-align: center;
            }

            .egale{
                width: 25em;
                text-align: center;
                position: relative;
                height:100%;
            }

            table {
                font-family: arial, sans-serif;
                border-collapse: collapse;
                width: 60%;
                margin:auto;
            }

            td, th {
```

Le pdf se crée à partir de balise html et de css . Il est à noter que Dompdf a des difficultés avec Bootstrap (ce qui peut se comprendre puisque bootstrap demande une bibliothèque extérieure) mais aussi avec certaines valeurs en Css tel que Display grid . Il est préférable de travailler avec un document html simple utilisant les tableaux pour positionner les éléments et des attributs basiques de css

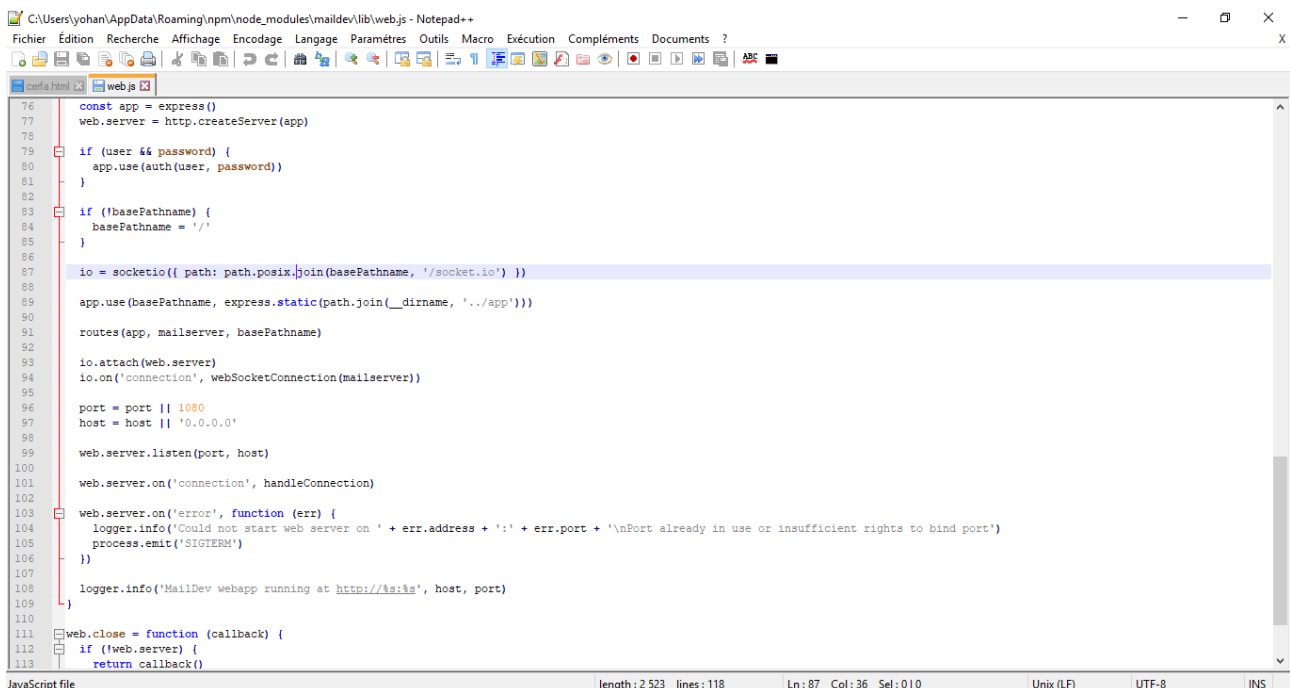
L'envoi d'emails classiques en php peut se faire à partir de la fonction mail() .Mais en utilisant cette fonction, la construction classique de cet email est complexe, et peut bien

souvent finir en courrier indésirable si celle-ci est mal faite. C'est la raison pour laquelle on utilise un composant de Laravel, provenant de Symfony, pour l'envoi d'email : « swiftmailer »

L'envoi d'email exige plusieurs étapes sur Laravel, Après, l'a procédure est assez simple. Par contre, tester l'envoi d'email en local m'a demandé quelques recherches pour trouver l'outil adapté. Le plus simple à installer et à utiliser est Maildev : celui s'installe à partir de npm et à partir de la commande « npm install -g maildev »

Le lancement de Maildev se fait à partir de la commande « Maildev -o »

Dans mon cas Maildev n'a pas marché directement : après recherche et différents tutos sur YouTube, il s'est avéré que la manipulation à faire pour que Maildev se lance est allée dans le fichier lib/web.js de Maildev et changé une ligne du code comme l'image suivante :



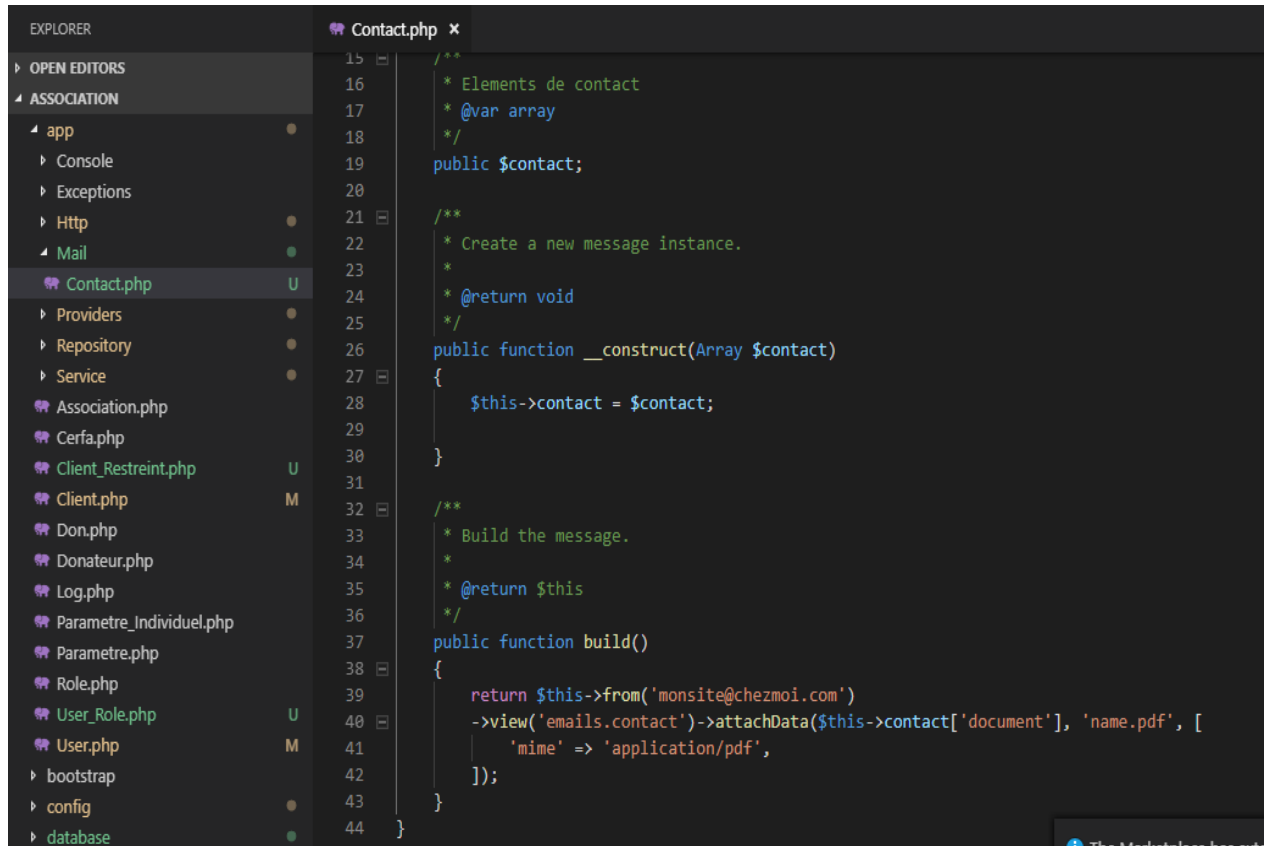
```
76 const app = express()
77 web.server = http.createServer(app)
78
79 if (user && password) {
80   app.use(auth(user, password))
81 }
82
83 if (!basePathname) {
84   basePathname = '/'
85 }
86
87 io = socketio({ path: path.posix.join(basePathname, '/socket.io') })
88
89 app.use(basePathname, express.static(path.join(__dirname, '../app')))
90
91 routes(app, mailserver, basePathname)
92
93 io.attach(web.server)
94 io.on('connection', websocketConnection(mailserver))
95
96 port = port || 1080
97 host = host || '0.0.0.0'
98
99 web.server.listen(port, host)
100
101 web.server.on('connection', handleConnection)
102
103 web.server.on('error', function (err) {
104   logger.info('Could not start web server on ' + err.address + ':' + err.port + '\nPort already in use or insufficient rights to bind port')
105   process.emit('SIGTERM')
106 })
107
108 logger.info('MailDev webapp running at http://0.0.0.0:' + host + ':' + port)
109
110
111 web.close = function (callback) {
112   if (!web.server) {
113     return callback()
114   }
115 }
```

Pour effectuer l'envoi d'e-mail, dans le cas de l'utilisation de Maildev , nous devons encore paramétrer le fichier .env de Laravel comme suit :

```
MAIL_DRIVER=smtp
MAIL_HOST=127.0.0.1
MAIL_PORT=1025
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null
```

Une fois fait, on utilise la commande « php artisan make:mail Contact »

Qui génère une « class » nommée ici Contact.



Contact est alors la classe qui centralise la construction est le contenu de l'email

Le constructeur de la classe Contact contient les données de l'email envoyé par l'utilisateur.

La méthode Build va construire l'email : celle-ci renvoie en particulier une vue qui contient le format de l'email envoyé

```

<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="utf-8">
  </head>
  <body>
    <h2>Prise de contact sur mon beau site</h2>
    <p>Réception d'une prise de contact avec les éléments suivants :</p>
    <ul>
      <li><strong>Nom</strong> : {{ $contact['nom'] }}</li>
      <li><strong>Email</strong> : {{ $contact['email'] }}</li>
      <li><strong>Message</strong> : {{ $contact['texte'] }}</li>
      <li><strong>Document</strong> : {{ $contact['document'] }}</li>
    </ul>
  </body>
</html>

```

De plus la méthode AttachData associée à la vue permet d'attacher un document dans notre cas au format pdf au mail

Il est à noter qu'un envoi d'email particulier a posé problème et a échoué, lors de son utilisation sur le site en ligne hébergé par « one and one ».

Une recherche sur internet m'a donné la solution

« Déjà dans le fichier env pour one & one :

```
MAIL_DRIVER=mail
MAIL_HOST=smtp.ionos.fr
```

ensuite dans config/mail.php

```
changer 'driver' => env('MAIL_DRIVER', 'smtp'), pour 'driver' => 'mail',
```

Et le mode de transport pour swift doit aussi être changer car ne va pas pour 1&1 dans

```
vendor/swifmailer/swifmailer/lib/classes/Swift/SendmailTransport.php
```

Changer le constructeur :

```
public function __construct($command = '/usr/sbin/sendmail -bs')
par public function __construct($command = '/usr/sbin/sendmail -t -i')
```

Et là miracle les mails fonctionne! merci 1&1 pour m'avoir occupé jamais eu de problèmes chez ovh ou online ... »

## TEST DE L'APPLICATION

Inscription d'un donateur :

Vue formulaire :

mail@donateur.fr

Adresse  
18 rue test

Telephone  
0388659878

Telephone 2  
0659897812

Ville  
strasbourg

Code Postal  
67000

pays  
france

date de naissance  
13 / 03 / 1985

Commentaire  
<iframe src="google.fr"></iframe>  
<script>  
console.log("coucou");  
document.location.reload(true);  
</script>

Association  
ASSOCIATION LES PETITS PONNETS  
ASSOCIATION DE RUFUS

Sign in

On notera que pour le test une tentative d'insertion de code est tenté. La validation au moment où j'écris le dossier le permet, mais est géré au niveau client par Laravel comme le montre la suite :

index donateur creation donateur

INFORMATION DONNATEUR 5

nom : nomdonateur

prenom : prenomdonateur

adresse : 18 rue test

ville : strasbourg

code postal : 67000

telephone : 0388659878

telephone : 0659897812

<iframe src="google.fr"></iframe> <script> console.log("coucou"); document.location.reload(true); </script>

modification

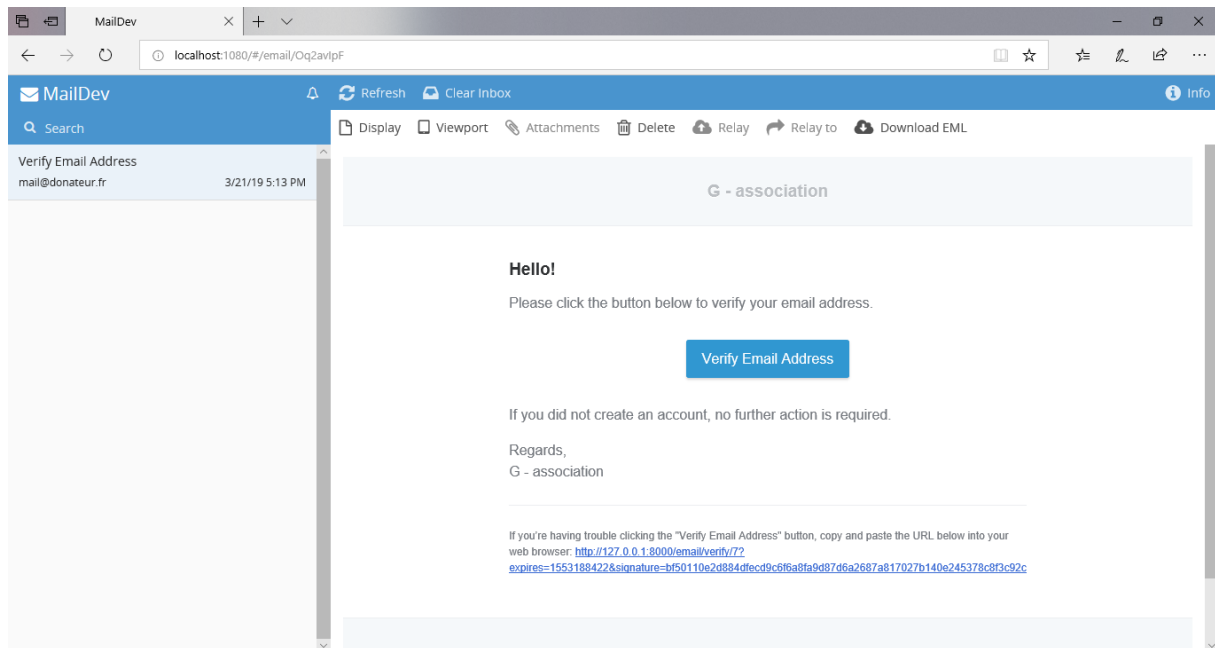
liste des associations  
ASSOCIATION DE RUFUS

liste des dons repertorié

L'évolution du code pourra interdire ce genre d'insertion à l'aide d'un regex du type

« `^[^<>%$]*$` »

Lorsque le donateur est inscrit un mail lui est envoyé :



Dans la suite le mail devra évoluer, ce mail permet de faire une étape supplémentaire de sécurité dans l'accès du site.

### Test de l'insertion du don et de l'envoi d'un cerfa à un donateur :

Inscription du don dans le formulaire :

A screenshot of a web application interface for 'G - association'. The browser's address bar shows '127.0.0.1:8000/don/create'. The interface has a sidebar with icons for 'DONATEUR', 'ASSOCIATION', 'PARAMETRE INDIVIDUEL', and 'CLIENT RESTREINT'. The main form contains the following fields: 'nom de l'association' (ASSOCIATION LES PETITS PONNETS), 'choix du donateur, barre de recherche' (bensoussan jose), 'don' (145), 'don en lettre' (cent quarante cinq), 'forme du don' (billet), 'nature du don' (espèce), 'mode versement du don' (manuel), 'date versement' (04 / 02 / 2019), and checkboxes for 'création cerfa' and 'validation cerfa'. A blue 'valider' button is at the bottom.

Affichage du don sur l'écran :

donateur	don	association	date du don	cerfa	validation	cerfa id
1	1560	ASSOCIATION LES PETITS PONNETS	2019-03-06 09:42:46	<a href="#">lien vers cerfa</a>	1	1
1	154	ASSOCIATION DE RUFUS	2019-03-06 13:38:21	<a href="#">lien vers cerfa</a>	1	2
1	10	ASSOCIATION LES PETITS PONNETS	2019-03-06 13:41:44	<a href="#">lien vers cerfa</a>	1	3
1	123	ASSOCIATION LES PETITS PONNETS	2019-03-22 08:01:45	<a href="#">lien vers cerfa</a>	1	4
1	145	ASSOCIATION LES PETITS PONNETS	2019-03-22 08:03:36	<a href="#">lien vers cerfa</a>	1	5

Mail envoyé au donateur :


**Prise de contact sur mon beau site**

Réception d'une prise de contact avec les éléments suivants :

- **Nom** : association :
- **Email** : bensoussan@gmail.com
- **Message** : ceci est un mail automatique de la part de l'association : ASSOCIATION LES PETITS PONNETS, en pièce jointe veuillez trouvez un cerfa , un grand merci pour votre don
- **document** :



Le cerfa envoyé est en pièce jointe il se présente ainsi :

 N° 11580*03	<b>Reçu au titre des dons à certains organismes d'intérêt général</b>  Articles 200 et 238 bis du code général des impôts (CGI)	RECU N° 4
--	--	-----------

Bénéficiaire des versements
<b>Organisme : ASSOCIATION LES PETITS PONNETS</b> <b>Adresse : 1 rue du cheval</b> <b>Siret : 10025632884</b> <b>APE : 5AE</b> <b>RCS : PARIS</b> Oeuvre ou organisme d'intérêt général

Donateur
<b>Société :</b> <b>Nom : bensoussan jose</b> <b>adresse : 1 avenue alsace</b>

Le bénéficiaire reconnaît avoir reçu au titre des dons et versements ouvrant droit à réduction d'impôt, la somme de :

<b>somme en chiffre : 145</b>	somme en lettre : cent quarante cinq
-------------------------------	--------------------------------------

**Date du versement: 2019-02-04**  
**Forme du don : billet**  
**Nature du don : espèce**  
**Mode de versement du don : manuel**

date :2019-03-22 08:03:36

nom :

signature :



### CONCLUSION, POINTS A EVOLUER DU PROJET

Mon projet à l'état actuel n'est pas entièrement abouti, certains points devront encore évoluer :

- Les formulaires ne sont pas encore aboutis (par exemple certain attributs de formatage pourraient être ajoutés ; au niveau des adresses on pourrait mettre une case numéro puis une case adresse, etc...)
- Au niveau de la validation, une réflexion est encore à faire pour mieux formater (réflexion sur le formatage du siret d'une entreprise, etc.)
- Le précédent logiciel à des fonctionnalités en plus que je n'ai pas (par exemple au niveau des rôles)
- Pour le moment un administrateur ne peut pas aller dans le compte client, un client ne peut pas voir son compte donateur au cas où celui-ci l'est (une variable session pourrait arranger cela)
- A l'intérieur du logiciel les utilisateurs ne peuvent pas encore changer de mot de passe et autre ...
- Le mot de passe de départ est fixe
- On pourrait ajouter un formulaire de préinscription au logiciel
- Seul les mots de passe sont hachés, pour une meilleure sécurité de l'information il faudrait aussi hacher d'autre données telles que nom, prénom, etc, ...

Aujourd'hui d'autre logiciels de don existent sur internet ; pour que ce logiciel soit plus performant il faudrait que celui-ci soit un logiciel plus global de gestion d'une association.

De plus une réflexion est encore à faire au niveau du web service, et plus généralement sur sa définition : qu'est-ce qu'un web service

Enfin pour conclure ce travail je voudrais adresser tous mes remerciements à mes tuteurs de formation et de stage.

# ANNEXES

Créer un projet laravel

**composer create-project --prefer-dist laravel/laravel blog**

Connaître toutes les routes de son application

**php artisan route:list**

Tester une application

**php artisan serve**

Installation des migrations

**php artisan migrate:install**

Exemple de création d'une migration pour une table emails

**php artisan make:migration create\_emails\_table**

Génération d'un modèle lié à une table Clients

**php artisan make:model Clients**

Génération d'un Controller lié à une entité spécifique (ici l'entité se nomme Reponses)

**php artisan make:Controller ReponsesController --resource**

*Hors console on met sur le fichier de routage :*

*Route::resource('categories', 'ReponsesController');*

Permet de vérifier la liste des routes du site

**php artisan route:list**

Génération de vues pour l'authentification

## **php artisan make:auth**

*Génération de routing pour l'authentification :*

*Route::auth();*

*Route::get('/home', 'HomeController@index');*

### GLOSSAIRE

(Source Wikipédia)

#### PATTERN DESIGN :

---

##### *Patron de conception*

*En informatique, et plus particulièrement en développement logiciel, un patron de conception (souvent appelé design pattern) est un arrangement caractéristique de modules, reconnu comme bonne pratique en réponse à un problème de conception d'un logiciel. Il décrit une solution standard, utilisable dans la conception de différents logiciels.*

*(Source Wikipédia)*

*Un patron de conception est donc, un point de départ dans la réflexion de la création d'un logiciel, c'est une architecture de logiciel, une organisation des différents composants d'un logiciel. L'avantage principal d'un design pattern est la facilitation du travail en équipe, de l'évolution du logiciel*

*Exemple de design pattern : Modèle-vue-contrôleur(mvc), model-view-presenter (MVP) et model-view-view model*

#### SERVEUR :

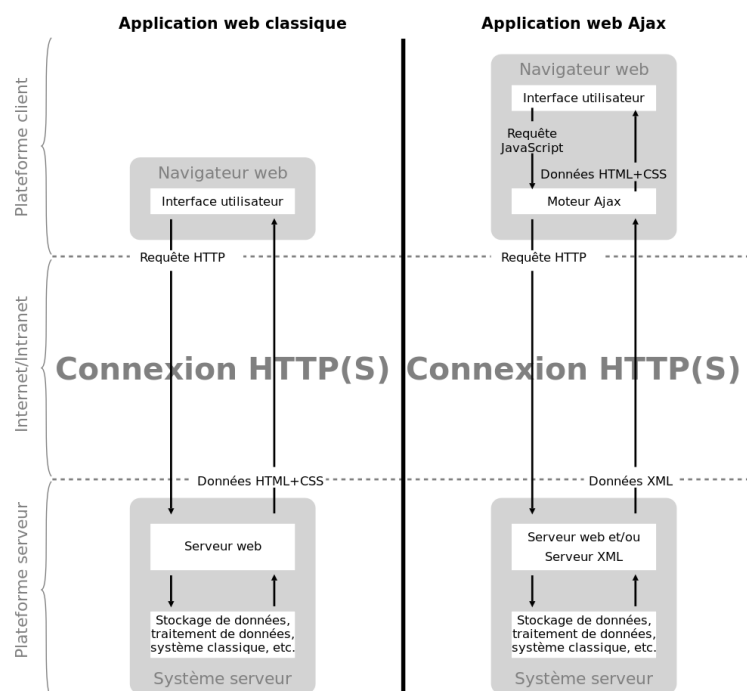
---

Un serveur informatique est un dispositif informatique (matériel ou logiciel) qui offre des services, à un ou plusieurs clients (parfois des milliers). Les services les plus courants sont :

- L'accès aux informations du World Wide Web ;
- Le courrier électronique ;
- Le partage d'imprimantes ;
- Le commerce électronique ;
- Le stockage en base de données ;
- La gestion de l'authentification et du contrôle d'accès ;
- Le jeu et la mise à disposition de logiciels applicatifs (optique Logiciel en tant que service).

#### AJAX :

---



L'architecture informatique ajax (acronyme d'asynchronous JavaScript and XML : JavaScript et XML asynchrones) permet de construire des applications Web et des sites web dynamiques interactifs sur le poste client en se servant de différentes technologies ajoutées aux navigateurs web entre 1995 et 2005.

Ajax combine JavaScript, les requêtes de type XMLHttpRequest, les manipulations du DOM, ainsi qu'un format de données (XML ou JSON), afin d'améliorer maniabilité et confort d'utilisation des applications internet riches :

DOM et JavaScript permettent de modifier l'information présentée dans le navigateur en respectant sa structure ;

L'objet XMLHttpRequest sert au dialogue asynchrone avec le serveur Web ;

XML, cité dans l'acronyme, était historiquement le moyen privilégié pour structurer les informations transmises entre serveur Web et navigateur, de nos jours le JSON tend à le remplacer pour cet usage.

L'usage d'Ajax fonctionne sur tous les navigateurs Web courants : Google Chrome, Safari, Mozilla Firefox, Internet Explorer, Microsoft Edge, Opera, etc.

**MYSQL :**

système de gestion de bases de données relationnelles (SGBDR). Il est distribué sous une double licence GPL et propriétaire. Il fait partie des logiciels de gestion de base de données les plus utilisés au monde<sup>3</sup>, autant par le grand public (applications web principalement) que par des professionnels, en concurrence avec Oracle, PostgreSQL et Microsoft SQL Server.

### SQL :

---

SQL (sigle de Structured Query Language, en français langage de requête structurée) est un langage informatique normalisé servant à exploiter des bases de données relationnelles. La partie langage de manipulation des données de SQL permet de rechercher, d'ajouter, de modifier ou de supprimer des données dans les bases de données relationnelles.

Outre le langage de manipulation des données, la partie langage de définition des données permet de créer et de modifier l'organisation des données dans la base de données, la partie langage de contrôle de transaction permet de commencer et de terminer des transactions, et la partie langage de contrôle des données permet d'autoriser ou d'interdire l'accès à certaines données à certaines personnes.

Créé en 1974, normalisé depuis 1986, le langage est reconnu par la grande majorité des systèmes de gestion de bases de données relationnelles (abrégié SGBDR) du marché.

SQL fait partie de la même famille que les langages SEQUEL (dont il est le descendant), QUEL (intégré à Ingres) ou QBE (Zloof).

### MERISE :

---

Merise (prononcer /mɛ.ʁiz/) est une méthode d'analyse, de conception et de gestion de projet informatique.

Merise a été très utilisée dans les années 1970 et 1980 pour l'informatisation massive des organisations. Cette méthode reste adaptée pour la gestion des projets internes aux organisations, se limitant à un domaine précis. Elle est en revanche moins adaptée aux projets transverses aux organisations, qui gèrent le plus souvent des informations à caractère sociétal (environnemental et social) avec des parties prenantes.

### UML :

---

Le Langage de Modélisation Unifié, de l'anglais Unified Modeling Language (UML), est un langage de modélisation graphique à base de pictogrammes conçu pour fournir une



méthode normalisée pour visualiser la conception d'un système. Il est couramment utilisé en développement logiciel et en conception orientée objet.

L'UML est le résultat de la fusion de précédents langages de modélisation objet : Booch, OMT, OOSE. Principalement issu des travaux de Grady Booch, James Rumbaugh et Ivar Jacobson, UML est à présent un standard adopté par l'Object Management Group (OMG).

### BOOTSTRAP :

---

Bootstrap est une collection d'outils utiles à la création du design (graphisme, animation et interactions avec la page dans le navigateur, etc.) de sites et d'applications web. C'est un ensemble qui contient des codes HTML et CSS, des formulaires, boutons, outils de navigation et autres éléments interactifs, ainsi que des extensions JavaScript en option. C'est l'un des projets les plus populaires sur la plate-forme de gestion de développement GitHub.

### JQUERY :

---

jQuery est une bibliothèque JavaScript libre et multiplateforme créée pour faciliter l'écriture de scripts côté client dans le code HTML des pages web3. La première version est lancée en janvier 2006 par John Resig.

La bibliothèque contient notamment les fonctionnalités suivantes :

- Parcours et modification du DOM (y compris le support des sélecteurs CSS 1 à 3 et un support basique de XPath) ;
- Événements ;
- Effets visuels et animations ;
- Manipulations des feuilles de style en cascade (ajout/suppression des classes, d'attributs...) ;
- Ajax ;
- Plugins ;
- Utilitaires (version du navigateur web...).

Depuis sa création en 2006 et notamment à cause de la complexification croissante des interfaces Web, jQuery a connu un large succès auprès des développeurs Web et son apprentissage est aujourd'hui un des fondamentaux de la formation aux technologies du Web. Il est à l'heure actuelle la librairie front-end la plus utilisée au monde (plus de la moitié des sites Internet en ligne intègrent jQuery).

## DICTIONNAIRE DES TABLES

## BASE DE DONNEES LARAVEL\_ASSOCIATION

**Structure de la table associations**

Colonne	Type	Null	Valeur par défaut
id	int(10)	Non	
client_id	int(10)	Non	
nom_association	varchar(191)	Non	
cle_de_filtage	int(11)	Non	
created_at	timestamp	Oui	NULL
updated_at	timestamp	Oui	NULL
adresse	varchar(191)	Non	
ville	varchar(191)	Non	
code_postal	varchar(100)	Non	
pays	varchar(100)	Non	
telephone_1	varchar(100)	Non	
adresse_email	varchar(100)	Non	
active	tinyint(1)	Non	
siret	varchar(191)	Oui	NULL
ape	varchar(191)	Oui	NULL
rsc	varchar(191)	Oui	NULL

**Structure de la table cerfas**

Colonne	Type	Null	Valeur par défaut
id	int(10)	Non	
oldNo	int(11)	Non	0
pre_edit	text	Non	
edit	text	Non	
confirmed	tinyint(1)	Non	
date_signature	date	Oui	NULL
don_id	int(10)	Non	
created_at	timestamp	Oui	NULL
updated_at	timestamp	Oui	NULL

**Structure de la table clients**

Colonne	Type	Null	Valeur par défaut
---------	------	------	-------------------

<b>id</b>	int(10)	Non
client	varchar(191)	Non
created_at	timestamp	Oui NULL
updated_at	timestamp	Oui NULL
user_id	int(10)	Non
nom	varchar(100)	Non
prenom	varchar(100)	Non
adresse	varchar(191)	Non
ville	varchar(191)	Non
code_postal	varchar(100)	Non
telephone_1	varchar(100)	Non
telephone_2	varchar(100)	Non
date_de_naissance	datetime	Oui NULL
commentaire	longtext	Non
active	tinyint(1)	Non
email	varchar(191)	Non

### Structure de la table clients\_restreints

Colonne	Type	Null	Valeur par défaut
<b>id</b>	int(10)	Non	
user_id	int(10)	Non	
client_id	int(10)	Non	
nom_client_restreint	varchar(100)	Non	
prenom_client_restreint	varchar(100)	Non	
adresse_client_restreint	varchar(191)	Non	
ville_client_restreint	varchar(191)	Non	
code_postal_client_restreint	varchar(100)	Non	
pays_client_restreint	varchar(191)	Non	
email_client_restreint	varchar(100)	Non	
telephone_1_client_restreint	varchar(100)	Non	
telephone_2_client_restreint	varchar(100)	Non	
date_de_naissance_client_restreint	date	Oui	NULL
active_client_restreint	tinyint(1)	Non	1
commentaire_client_restreint	text	Oui	NULL
filtrage_association_client_restreint	int(11)	Non	
created_at	timestamp	Oui	NULL

updated\_at timestamp Oui NULL

### Structure de la table donateurs

Colonne	Type	Null	Valeur par défaut
id	int(10)	Non	
client_id	int(10)	Non	
filtrage_association	int(11)	Non	
created_at	timestamp	Oui	NULL
updated_at	timestamp	Oui	NULL
user_id	int(10)	Non	
nom	varchar(100)	Non	
prenom	varchar(100)	Non	
adresse	varchar(191)	Non	
ville	varchar(191)	Non	
code_postal	varchar(100)	Non	
telephone_1	varchar(100)	Non	
telephone_2	varchar(100)	Non	
date_de_naissance	datetime	Oui	NULL
commentaire	longtext	Non	
active	tinyint(1)	Non	
email	varchar(191)	Non	
societe	varchar(191)	Oui	NULL

### Structure de la table dons

Colonne	Type	Null	Valeur par défaut
id	int(10)	Non	
donateur_id	int(10)	Non	
association_id	int(10)	Non	
don	double	Non	
created_at	timestamp	Oui	NULL
updated_at	timestamp	Oui	NULL
valider	tinyint(1)	Non	0
cerfa_id	int(10)	Oui	NULL
don_en_lettre	varchar(191)	Oui	NULL
don_forme	varchar(191)	Oui	NULL
don_date_versement	date	Oui	NULL
don_nature	varchar(191)	Oui	NULL

don\_mode\_versement varchar(191) Oui NULL

### Structure de la table logs

Colonne	Type	Null	Valeur par défaut
id	int(10)	Non	
log	text	Non	
user_id	int(10)	Non	
created_at	timestamp	Oui	NULL
updated_at	timestamp	Oui	NULL

### Structure de la table migrations

Colonne	Type	Null	Valeur par défaut
id	int(10)	Non	
migration	varchar(191)	Non	
batch	int(11)	Non	

### Structure de la table parametres

Colonne	Type	Null	Valeur par défaut
id	int(10)	Non	
parametre	varchar(191)	Non	
valeur	varchar(191)	Non	
created_at	timestamp	Oui	NULL
updated_at	timestamp	Oui	NULL

### Structure de la table parametres\_individuels

Colonne	Type	Null	Valeur par défaut
id	int(10)	Non	
valeur	varchar(191)	Non	
parametre_id	int(10)	Non	
client_id	int(10)	Non	
association_id	int(10)	Non	
created_at	timestamp	Oui	NULL
updated_at	timestamp	Oui	NULL
option_texte_individuel	text	Oui	NULL
option_image_individuel	varchar(191)	Oui	NULL

### Structure de la table password\_resets

Colonne	Type	Null	Valeur par défaut
email	varchar(191)	Non	
token	varchar(191)	Non	
created_at	timestamp	Oui	NULL

**Structure de la table roles**

Colonne	Type	Null	Valeur par défaut
id	int(10)	Non	
role	varchar(191)	Non	
created_at	timestamp	Oui	NULL
updated_at	timestamp	Oui	NULL

**Structure de la table sessions**

Colonne	Type	Null	Valeur par défaut
id	varchar(191)	Non	
user_id	int(10)	Oui	NULL
ip_address	varchar(45)	Oui	NULL
user_agent	text	Oui	NULL
payload	text	Non	
last_activity	int(11)	Non	

**Structure de la table users**

Colonne	Type	Null	Valeur par défaut
id	int(10)	Non	
name	varchar(191)	Non	
email	varchar(191)	Non	
email_verified_at	timestamp	Oui	NULL
password	varchar(191)	Non	
remember_token	varchar(100)	Oui	NULL
created_at	timestamp	Oui	NULL
updated_at	timestamp	Oui	NULL
active	tinyint(1)	Non	1

**Structure de la table users\_roles**

Colonne	Type	Null	Valeur par défaut
id	int(10)	Non	
role_id	int(10)	Non	
user_id	int(10)	Non	
created_at	timestamp	Oui	NULL
updated_at	timestamp	Oui	NULL

GENERALITE SUR LES ASSOCIATIONS

---

LISE REYNAERT, AURELIEN D'ISANTO, DIVISION ENQUETES THEMATIQUES  
ET ETUDES TRANSVERSALES, INSEE

«

*La France compte 1,3 million d'associations actives en 2013. Elles interviennent principalement dans quatre domaines : le sport, les loisirs, la culture et la défense de causes, de droits ou d'intérêts.*

*Seulement 12 % des associations emploient des salariés, le plus souvent un ou deux. La majorité des salariés des associations se concentre dans l'action sociale, humanitaire ou caritative, l'hébergement social ou médico-social et la santé. Par ailleurs, du personnel mis à disposition par d'autres organismes concourt souvent à l'activité des associations employeuses de l'enseignement et de la formation. Au total, le travail salarié effectué dans les associations représenterait 7 % de celui réalisé dans l'ensemble de l'économie, en équivalent temps plein.*

*Les bénévoles interviennent dans l'ensemble des domaines d'activité et dans la quasi-totalité des associations, même employeuses. Le volume de travail qu'ils y consacrent correspondrait à 680 000 emplois en équivalent temps plein.*

*Évaluées au total à 104 milliards d'euros, les ressources financières sont très concentrées dans les associations employeuses, en particulier dans l'action sociale, humanitaire ou caritative, l'hébergement social ou médico-social, la santé et l'enseignement. Elles résultent essentiellement de recettes d'activité, d'origine privée ou publique.*

»

CADRE JURIDIQUE DES ASSOCIATIONS

---

ARTICLE 238 BIS DU CODE GENERAL DES IMPOTS

1. Ouvrent droit à une réduction d'impôt égale à 60 % de leur montant les versements, pris dans la limite de 5 pour mille du chiffre d'affaires, effectués par les entreprises assujetties à l'impôt sur le revenu ou à l'impôt sur les sociétés au profit :

a) D'œuvres ou d'organismes d'intérêt général ayant un caractère philanthropique, éducatif, scientifique, social, humanitaire, sportif, familial, culturel ou concourant à la mise en valeur du patrimoine artistique, à la défense de l'environnement naturel ou à la diffusion de la culture...

b) De fondations ou associations reconnues d'utilité publique ou des musées de France et répondant aux conditions fixées au a, ainsi que d'associations culturelles ou de bienfaisance et des établissements publics des cultes reconnus d'Alsace-Moselle....

c) Des établissements d'enseignement supérieur ou d'enseignement artistique publics ou privés, d'intérêt général, à but non lucratif ;

.....

5 B-1-045 janvier 2004-2-I. Principe :

Les reçus sont établis et délivrés par les organismes bénéficiaires des versements. Ils doivent comporter toutes les mentions concernant l'organisme et figurant dans le nouveau modèle de reçu fixé par arrêté du 1<sup>er</sup> décembre 2003 (J.O. du 7 décembre 2003) dont une reproduction est publiée en annexe. Toutefois, ce reçu ne constitue qu'un modèle permettant de matérialiser le contenu du document. Sa présentation peut être aménagée. En raison de la diversité des organismes bénéficiaires et de leurs modalités de gestion, l'administration n'assure pas la fourniture d'imprimés permettant l'établissement des reçus. Chaque organisme ou association peut faire éditer par un imprimeur, se procurer auprès de son fournisseur habituel ou éditer lui-même par procédé informatique des reçus adaptés à sa situation et à ses propres contraintes de gestion. Les associations sont également autorisées à transmettre par internet les reçus aux donateurs qui les impriment eux-mêmes, sous réserve que les



reçus ainsi délivrés soient conformes aux normes décrites ci-dessous. Dans ce cas, le logiciel utilisé par l'association doit seulement permettre l'édition du reçu sans qu'aucune modification puisse être effectuée par le donateur.

Bien qu'il n'existe pas de format prescrit, les reçus fiscaux doivent impérativement contenir toutes les informations présentes dans le modèle de reçu fiscal officiel :

- **Un numéro d'ordre** : Numéro unique pour chaque reçu, comme sur une facture.
- **Organisme bénéficiaire** : Nom complet, adresse et objet explicite de l'association.
- **Qualité de l'organisme** : A quel titre l'organisme peut-il délivrer des reçus fiscaux ? Il s'agit de préciser « organisme d'intérêt général » pour la grande majorité des associations loi 1901. Les organismes reconnus d'utilité publique doivent préciser la date du décret et de publication au journal officiel de la reconnaissance de leur statut.
- **Coordonnées du donateur** : Nom et adresse complète.
- **Date du versement** : Format jj/mm/aaaa accepté.
- **Montant de la somme versée** : A indiquer en toutes lettres et en chiffres sur les reçus papier. Pour les reçus établis sur ordinateur, il n'est pas nécessaire d'écrire la somme en toutes lettres, mais le montant en chiffres doit être entouré de 3 astérisques. Exemple : \*\*\*200,00€\*\*\*
- **Nature du versement** : Don numéraire réglé par espèces, chèque ou virement, voire même titres de sociétés ou autre chose. S'il s'agit du renoncement au remboursement de frais ou à l'abandon de produits, cochez la case « Autres » dans le modèle officiel et précisez à côté de quoi il s'agit.
- **Signature d'une personne habilitée** : Signature lisible et qualité de la personne signataire, de préférence le président ou le trésorier de l'association. La signature peut être imprimée ou apposée à l'aide d'un tampon.

**En conclusion** les associations sont donc soumis à différentes lois au niveau de l'état français.

Celles-ci peuvent bénéficier de dons qui permettent aux donateurs d'accéder à des baisses d'impôt.

En contre parti du don et pour permettre d'accéder à la baisse d'impôt les associations se doivent de donner un cerfa normé selon les préceptes donnés auparavant. La gestion des cerfas est donc un facteur clé dans la survie d'une association c'est la raison pour laquelle il m'a été demandé durant mon stage de créer un logiciel de suivi et d'édition de cerfa.

## MASK OF BITS THEORIE

*Quelques mots de mathématiques sur l'écriture d'un nombre en base  $n$  :*

**Lemme 1.2.1** Soient  $k \geq 0$  un entier et  $c_0, \dots, c_k \in \{0, \dots, b-1\}$ . On suppose que  $c_k \neq 0$ . L'entier  $a := c_k b^k + \dots + c_0 b^0$  vérifie alors :

$$(1.2.1.1) \quad b^k \leq a \leq b^{k+1} - 1.$$

**Théorème 1.2.2** Soit  $a \in \mathbb{N}^*$  un entier naturel non nul. Il existe alors un unique entier  $k \geq 0$  et une unique suite finie  $(c_0, \dots, c_k)$  tels que :

$$(1.2.2.1) \quad \begin{cases} a = c_k b^k + \dots + c_0 b^0, \\ 0 \leq c_i \leq b-1 \text{ pour } i = 0, \dots, k, \\ c_k \neq 0. \end{cases}$$

On dit alors que  $(c_k \dots c_0)_b$  est l'écriture de  $a$  en base  $b$ . Les  $c_i$  sont les chiffres de cet écriture. On précise parfois que cette écriture se fait poids forts à gauche.

On en conclue que tout nombre s'écrit d'une manière unique en base  $k$  où  $k$  est un entier naturel plus grand ou égale à 2

1101 est un nombre en base 2, en base 10 se nombre donne  $1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$   
 $= 8 + 4 + 1 = 13$

1101 est la représentation unique en base 2 de 13 en base 10 (qui lui-même est la représentation unique de ce nombre en base 10 qui peut aussi s'écrire ainsi :  $1 \cdot 10^1 + 3 \cdot 10^0$ )

## UTILISATION DU MASQUAGE PAR UN EXEMPLE

Supposons qu'un client gère 4 associations, l'association A, l'association B, l'association C et l'association D nous voulons pouvoir gérer le fait qu'un bénévole soit ou non dans l'association A, B, C ou D

Nous pourrions alors gérer cela par un tableau, avec des valeurs booléennes représentées par les nombres 1 et 0, 1 un bénévole appartient à l'association et 0 le bénévole n'appartient pas à l'association.

Ce qui donnerait alors pour un bénévole particulier

Association A	Association B	Association C	Association D
1	0	0	1

Dans cet exemple le bénévole appartient à l'association A et D mais n'appartient pas à B et C

Comme vu la suite de nombres (1,0,0,1) peut être vue comme la représentation unique d'un nombre en base 2

En traduisant cette suite en base 10, cela donnerait  $1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$

$$= 8 + 1 = 9$$

Le fait qu'un bénévole appartienne à l'association A et D mais n'appartient pas à B et C peut donc être traduit par le nombre 9

de plus en base 10 appartenir à l'association A peut-être représenter par le nombre  $2^3 = 8$

en base 10 appartenir à l'association B peut être représenter par le nombre  $2^2 = 4$

en base 10 appartenir à l'association C peut être représenter par le nombre  $2^1 = 2$

en base 10 appartenir à l'association D peut être représenter par le nombre  $2^0 = 1$

Le langage PHP nous offre l'opportunité de travailler avec des opérateurs sur les bits, ainsi si nous continuons dans notre exemple :

- le bénévole appartient-il à l'association A ?  $9 \& 8 = 8$  renvoie true
- le bénévole appartient-il à l'association B ?  $9 \& 4 = 4$  renvoie false
- le bénévole appartient-il à l'association C ?  $9 \& 2 = 2$  renvoie false
- le bénévole appartient-il à l'association B ?  $9 \& 1 = 1$  renvoie true

Un tableau d'appartenance de bénévole aux association A, B, C, D donnerait alors

Nom	Appartenance
Dupond	9
Didier	6
Roland	14
Robert	1

Ainsi Dupond appartient à l'association A et D

Didier appartient à l'association B et C

Roland appartient à l'association A, B, et C

Et Robert appartient à l'association D