## **Vues :**

Les vues sont créer grâce au template *Blade [1].*

*Creation de layout blade permettant de ne pas répéter un contenu commun à plusieurs page.*

Les layaout sont place dans le dossier : app/views/layouts/master.blade.php

Par exemple tout la structure principale de la page HTML (header et body) commun à chaque page est dans le fichier master.balde dossier layouts

Ce contenu encapsule le contenu placé dans : « content » le contenu de chaque page est donc placé dans la section « content »

Les données sont directement validée escapé dans la vue grace au triple acolade {{{

Traduction [2] ,

## **Contrôleurs :**

Ajout du filtre CSRF avant l’exécution du contrôleur.

Voir le schéma des contrôleurs [3]

Création des ressources pour chaque contrôleur : php artisan controller:make UserController

index : pour afficher la liste des utilisateurs,  
create : pour envoyer le formulaire pour la création d'un nouvel utilisateur,  
store : pour créer un nouvel utilisateur,  
show : pour afficher les données d'un utilisateur,  
edit : pour envoyer le formulaire pour la modification d'un utilisateur,  
update : pour modifier les données d'un utilisateur,

destroy : pour supprimer un utilisateur.

**Validation des données**, injection des classes de validation dans le contrôleur permettant d’effectué la validation dans des fichiers séparé.  
Le contrôle de la validation, si echouée (fails) alors les erreurs et les entrée (inputs) sont retournée pour correction par l’utilisateur

Création des fichiers ValidatorInerface et BaseValidator defini le mécahnisme de validation

Chaque validation est faite grâce à un fichier de validation séparé permettant d’effectuer facilement les tests.

Les fichiers de validation sont placé dans un dossier ajouté a Laravel : Lib/Validation

Ce dossier doit être à la liste de dossier connu par Laravel grâce à composer.json puis la mise à jours : composer dumpautoload

Gestion des données

## **Base de données configuration**

**Migrations**

[4] configuration de database.php pour la base de donnee birdswink en mysql

Puis création de la table migration contenant l’historique des migrations avec Artisan : *php artisan migrate:install*

Création de la migration de la table users

php artisan migrate:make create\_users\_table   
php artisan migrate:make create\_boxes\_table  
php artisan migrate:make create\_pictures\_table  
php artisan migrate:make create\_comments\_table  
php artisan migrate:make create\_likes\_table

Edition des fichiers de migration (database/migration) (methodes up et down)   
afin de créer les colonnes des la tables (voir schéma bdd)

(Le format de la date à été choisi comme : Timestamp unix [5])

**Populations**

Creation de la population pour les tests

app/database/seeds  
php artisan migrate:rollback (suppression tables)  
php artisan migrate (creation tables)  
php artisan db:seed (creation populations)

**Relations**

Relation 1 :n dans le fichier migration de chaque table

Exemple liaison de la table boxes avec users

//create relation

Schema::table('boxes', function(Blueprint $table) {

$table->foreign('user\_id')->references('id')->on('users')

->onDelete('restrict')

->onUpdate('restrict');

});

Puis liaison dans le fichier modèle : (/model)

Exemple pour le users

## **Références**

*[1] :* [*http://laravel.com/docs/templates*](http://laravel.com/docs/templates)

*[2] :* [*http://docs.laravel.fr/3/localization*](http://docs.laravel.fr/3/localization)

*[3] :* [*http://fr.openclassrooms.com/informatique/cours/decouvrez-le-framework-php-laravel/les-controleurs-4*](http://fr.openclassrooms.com/informatique/cours/decouvrez-le-framework-php-laravel/les-controleurs-4)

*[4] :* [*http://fr.openclassrooms.com/informatique/cours/decouvrez-le-framework-php-laravel/migrations-et-modeles*](http://fr.openclassrooms.com/informatique/cours/decouvrez-le-framework-php-laravel/migrations-et-modeles) *[5]* [*http://dev.petitchevalroux.net/mysql/choisir-bon-format-date-mysql.225.html*](http://dev.petitchevalroux.net/mysql/choisir-bon-format-date-mysql.225.html)