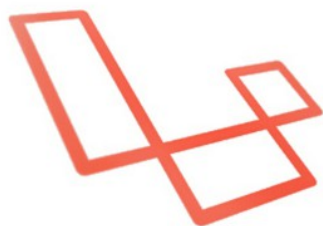




HAUTE ÉCOLE DE LA COMMUNAUTÉ FRANÇAISE EN HAINAUT
Catégorie Technique
8a Avenue Victor Maistriau, 7000 Mons

TR3TIN

Rapport de projet



laravel

Élèves :

Daubie Dylan
Poupon Martin

Titulaire : Mr. Tricarico

Année Académique : 2015 – 2016

Table des matières

Introduction.....	4
Design du site	5
Index.....	5
Inscription/Authentification.....	5
Home.....	6
Espace personnel.....	6
Création d'une liste de tâches.....	7
Création d'une tâche.....	7
Édition liste et tâche.....	8
About.....	8
Template.....	8
Les points importants du code.....	9
Création liste/tâche.....	9
Update liste/tâche.....	11
Home (point de vue contrôleur).....	13
Home (point de vue blade.php).....	14
Validation.....	15
Suppression.....	15
BDD.....	16
Sécurité mise en place.....	18
Point de vue sur Laravel.....	18
Conclusions.....	19

Introduction

Dans le cadre du cours de TR3TIN, nous avons dû mettre en place une « TodoList » développée avec l'aide du framework Laravel. Le site proposera aux utilisateurs possédant un compte de créer une liste de tâches à accomplir.

En plus de ça, il peut effectuer un suivi des tâches (tâches validées et non validées).

Le site doit comprendre plusieurs fonctionnalités, à savoir :

- Création d'un compte
- Connexion avec ce compte
- Créer et modifier une liste de tâches
- Créer et modifier une tâche

Design du site

Index

L'index fournit une brève explication du site, ainsi que deux boutons, l'un pour se connecter et l'autre s'inscrire.

Bienvenue sur TodoTracker !!!

Ici, vous pourrez créer votre liste de tâches à faire, les exécuter, les valider !

Qu'attendez-vous pour vous créer un compte !?

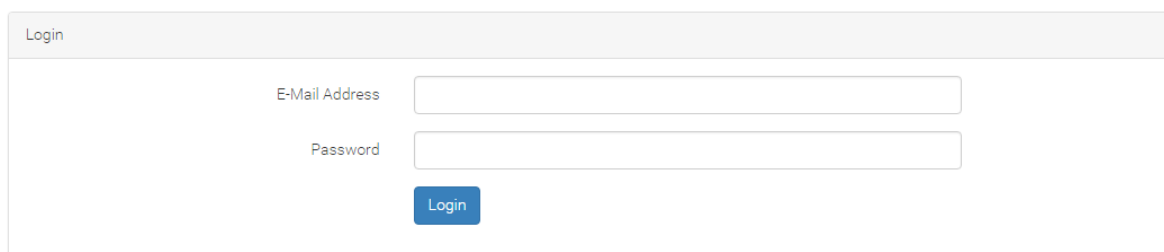
Inscription Connexion

Page de démarrage du site

Inscription/Authentification

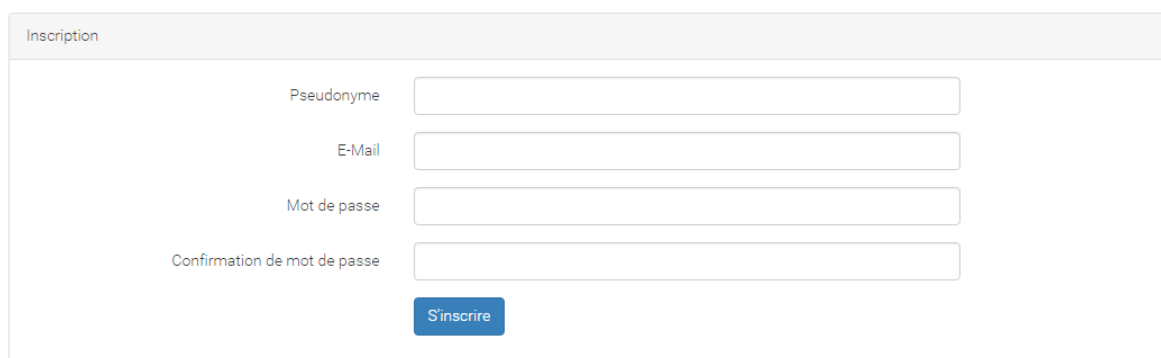
Pour accéder au contenu du site, l'utilisateur doit s'authentifier ou se créer un compte, via ces interfaces.

L'utilisateur doit entrer des identifiants valides (existants dans la base de données) pour pouvoir être connecté.



The login form is titled 'Login' in a light gray header. It contains two input fields: 'E-Mail Address' and 'Password'. Below the password field is a blue 'Login' button.

L'utilisateur peut s'authentifier via cette interface



The registration form is titled 'Inscription' in a light gray header. It contains four input fields: 'Pseudonyme', 'E-Mail', 'Mot de passe', and 'Confirmation de mot de passe'. Below the confirmation field is a blue 'S'inscrire' button.

L'utilisateur peut s'inscrire via cette interface

Home

Une fois connecté, l'utilisateur a accès à cette interface. Il peut y gérer ses listes de tâches, les valider, les supprimer et accéder à la page de modification des paramètres de la liste ainsi que celle des tâches. Les listes et les tâches associées sont affichées par ordre chronologique de création.

Liste de tâches en cours

TFE

Editer

Créer

Supprimer

Faire la page de garde

Editer

Valider

Supprimer

Echéance: 2016-01-07

L'utilisateur dispose de cette interface pour valider/supprimer des éléments

Espace personnel

Cette page permet d'avoir un aperçu détaillé des listes ainsi que de l'état des tâches validées ou encore en cours. C'est dans cette partie du site que l'utilisateur peut créer une liste de tâches via le bouton « Nouvelle Liste » situé en bas de la page. Un utilisateur ne peut pas créer deux fois la même liste.

Espace personnel

Nom de la liste : TFE

Editer

Description :

Création : 2016-01-06 15:16:11

Tâche(s) en cours :

Faire la page de garde

Tâche(s) terminée(s) :

Nouvelle Liste

L'utilisateur peut créer des listes de tâches

Création d'une liste de tâches

L'utilisateur a la possibilité de créer une liste et d'y ajouter une description (facultative). Il ne pourra pas créer de liste en laissant le champs « Liste de tâches » vide.

Nouvelle Liste

Liste de tâches :

Description :

Création de la liste

Création d'une tâche

L'utilisateur peut créer une tâche avec une date d'échéance qui lui est associée. Une tâche ne peut être créée sans un nom de tâche et une date d'échéance.

Ajout d'une tâche

Liste de tâche(s) : TFE

Tâche :

Échéance :

Création d'une tâche

Édition liste et tâche

Bien évidemment, l'utilisateur a la capacité de modifier entièrement les propriétés de la liste et des tâches.

Edition de tâche

Nom de la liste : TFE

Faire la page de garde

07-01-2016

Modifier

L'utilisateur peut éditer sa tâche

Edition de liste

Nom de la liste : TFE

Description :

Modifier

L'utilisateur peut éditer sa liste de tâches

About

Possibilité d'avoir accès au dépôt GIT des étudiants ayant participé au projet via le bouton présent en dessous .

A propos

Binôme :
Daubie Dylan
Poupon Martin

Intitulé du cours :
TR3TIN

Dépôt GIT du projet

Fournit des éléments supplémentaires

Template

Un template a été créé pour la navbar avec bootstrap, le contenu initial a été modifié de manière a pouvoir mettre les éléments « About », « Inscription » et « deconnexion » sur la droite de la page.

Les points importants du code

Création liste/tâche

Cette fonction permet de pouvoir ajouter une tâche dans la base de données. Une fois tous les paramètres entrés, la ligne est enregistrée dans la base de données et une redirection vers la route 'home' se met en place.

```
public function creation_tâche_soumission(Request $req)
{
    //le formulaire envoie tous les valeurs necessaires à la création de l'entrée dans la base de données.
    $param=$req->except(['_token']);
    $tache = new \App\Tache;
    $tache->liste=htmlentities($param['nomliste']);
    $tache->tache=htmlentities($param['nomtache']);
    $tache->date=htmlentities($param['date']);
    $tache->user_id=htmlentities($param['user_id']);
    $tache->done=htmlentities($param['done']);
    $tache->save();
    return redirect()->route('home');
}
```

Dans le cas de la liste, même procédure que pour la tâche, sauf qu'un contrôle de doublon s'applique. En effet, si un utilisateur souhaite créer deux fois la même liste, cela sera impossible.

```
public function creation_liste_soumission(Request $req)
{
    $param=$req->except(['_token']);
    $doublon = DB::table('listes')
        ->select('nomliste', 'user_id')
        ->where('nomliste' , '=', $param['nomliste'])
        ->where('user_id' , '=', Auth::user()->id)
        ->get();
    if (count($doublon) == 0)
    {
        $liste = new \App\Liste;
        $liste->user_id=htmlentities($param['user']);
        $liste->nomliste=htmlentities($param['nomliste']);
        $liste->description=htmlentities($param['description']);
        $liste->save();
        return redirect()->route('home');
    }
    else
    {
        return redirect()->route('creation_liste_errors');
    }
}
```

Update liste/tâche

Cette fonction, permet de mettre à jour la ligne de la base de données à modifier en réattribuant les valeurs « tâches » et « dates », puis en les sauvegardant. Et, par la même occasion écraser les anciennes valeurs.

```
public function updatetache(Request $req, $id)
{
    if(Auth::check())
    {
        $tache=Tache::find($id);
        if($req->isMethod('post'))
        {
            $param=$req->except(['_token']);
            $tache->tache=htmlentities($param['tache']);
            $tache->date=htmlentities($param['date']);
            $tache->save();
            return redirect()->route('home');
        }
        return view('pages/update_tache')->with('tache' , $tache);
    }
    else
    {
        return redirect()->route('index');
    }
}
```

Pour la modification de la liste, c'est la même fonction mis à part que l'on doit modifier le nom de la liste des tâches associé à la liste à modifier. Donc, une requête SQL a été mise en place pour rendre ceci possible.

```
public function updateliste(Request $req, $id)
{
    if(Auth::check())
    {
        //recupère l'id en cours de la liste et de la on la manipule
        //idem pour la fonction du dessous
        $liste=Liste::find($id);
        $tache_liste = $liste->nomliste;

        if($req->isMethod('post'))
        {
            $param=$req->except(['_token']);
            //reattribution des valeurs de nomliste et description de la table "listes"
            $liste->nomliste=htmlentities($param['liste']);
            $liste->description=htmlentities($param['description']);

            DB::table('taches')
                ->where('liste' , $tache_liste)
                ->update(['liste' => $liste->nomliste]);

            $liste->save();
            return redirect()->route('home');
        }
        return view('pages/update_liste')->with('liste' , $liste);
    }
    else
    {
        return redirect()->route('index');
    }
}
```

Home (point de vue contrôleur)

Pour ce qui est de la page « home », deux requêtes SQL ont été mises en place. Le premier, permet de sélectionner les tâches liées à une liste. Le deuxième, permet de faire une liste des noms de listes distincts. Une jointure est également créée pour lier la variable « liste » de la table « taches » et la variable « nomliste » de la table « liste ».

```
public function home()
{
    if(Auth::check())
    {
        $liste_home = DB::table('listes')
            ->join('taches', 'taches.liste', '=', 'listes.nomliste')
            ->select(
                'listes.user_id as id', 'listes.nomliste as nomliste', 'taches.tache as tache', 'taches.liste as liste',
                'taches.id as tache_id', 'taches.done as t_done', 'taches.date as date', 'taches.user_id'
            )
            ->where('listes.user_id', '=', Auth::user()->id) ->where('taches.done', '=', 0)
            ->where('date', '>=', new DateTime('today')) ->where('taches.user_id', '=', Auth::user()->id)
            ->orderBy('taches.date', 'ASC') ->get();

        $liste_unique = DB::table('listes')
            ->select('user_id', 'nomliste', 'id', 'created_at')
            ->groupBy('nomliste') ->where('user_id', '=', Auth::user()->id) ->orderBy('created_at', 'ASC')
            ->get();

        return view('pages/home', array(
            'liste_unique' => $liste_unique, 'selection' => $liste_home
        ));
    }
    else
    {
        return redirect()->route('index');
    }
}
```

Home (point de vue blade.php)

Pour pouvoir afficher l'ensemble des listes ainsi que leurs tâches respectives, je crée une première boucle qui a pour objectif de répertorier les listes. Puis dans cette boucle, je crée une seconde boucle qui a pour but de lister les tâches en lien avec la liste en cours de déploiement.

```
@foreach($liste_unique as $liste_home)
    <div id="conteneur_liste_home">
        <div id="liste_home">
            <b>{{ $liste_home->nomliste }}</b>
        </div>
        <div id="liste_boutons_home">
            <a href="{{ route('updatelist', ['id'=>$liste_home->id]) }}"><button class="btn btn-group-lg">Editer</button></a>
            <a href="{{ route('creation_tache', ['id'=>$liste_home->id]) }}"><button class="btn btn-info">Créer</button></a>
            <a href="{{ route('deletelist', ['id'=>$liste_home->id]) }}"><button class="btn btn-danger">Supprimer</button></a>
        </div>
    </div>

@endforeach

@if($tache_home->liste == $liste_home->nomliste)
    <div id="conteneur_tache_home">
        <div id="tache_home">{{ $tache_home->tache }}</div>

        <div id="boutons">
            <a href="{{ route('updatetache', ['id'=>$tache_home->tache_id]) }}"><button class="btn btn-group-lg">Editer</button></a>
            <a href="{{ route('validationtache', ['id'=>$tache_home->tache_id]) }}"><button class="btn btn-success">Valider</button></a>
            <a href="{{ route('deletetache', ['id'=>$tache_home->tache_id]) }}"><button class="btn btn-danger">Supprimer</button></a>
        </div>
        <div id="echéance"><b>Échéance:</b> {{ $tache_home->date }}</div>
    </div>
@endif
```

Validation

Cette fonction, va mettre à jour la ligne de la base de données liée à la tâche en modifiant la colonne « done » en 1. De cette manière, lors de l'affichage des tâches en cours, elle apparaîtra dans les tâches finies.

```
public function validationtache($id)
{
    $tache=Tache::find($id);

    $tache->done = 1;
    $tache->save();

    return redirect()->route('home');
}
```

Suppression

Pour la suppression des tâches, on fait appel à la méthode delete() présente dans Laravel et on lui cible la ligne de la base de données à supprimer. Pour ce qui est de la suppression de la liste. Il faut aussi supprimer toutes les tâches liées à cette liste, donc je fais une requête SQL qui regroupe toutes ces tâches et je les supprime également.

```
public function deletetache($id)
{
    $tache=Tache::find($id);
    $tache->delete();
    return redirect()->route('home');
}

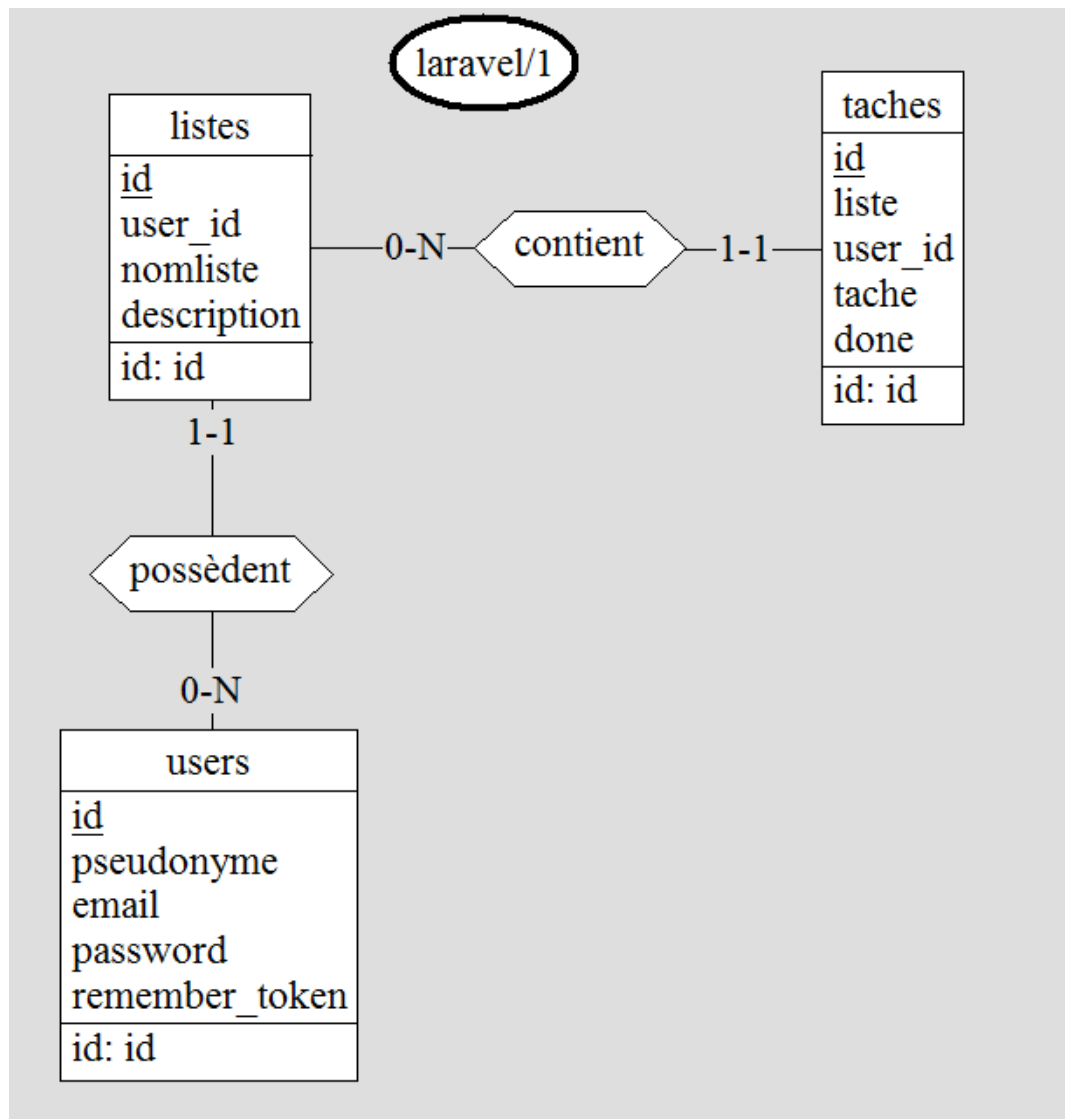
public function deleteliste($id)
{
    //suppression de la liste, MAIS, suppression également des tâches liées à ces listes. D'ou la query.
    $liste=Liste::find($id);
    $liste->delete();

    DB::table('taches')
        ->where('liste', '=', $liste->nomliste) ->where('user_id', '=', Auth::user()->id) ->delete();

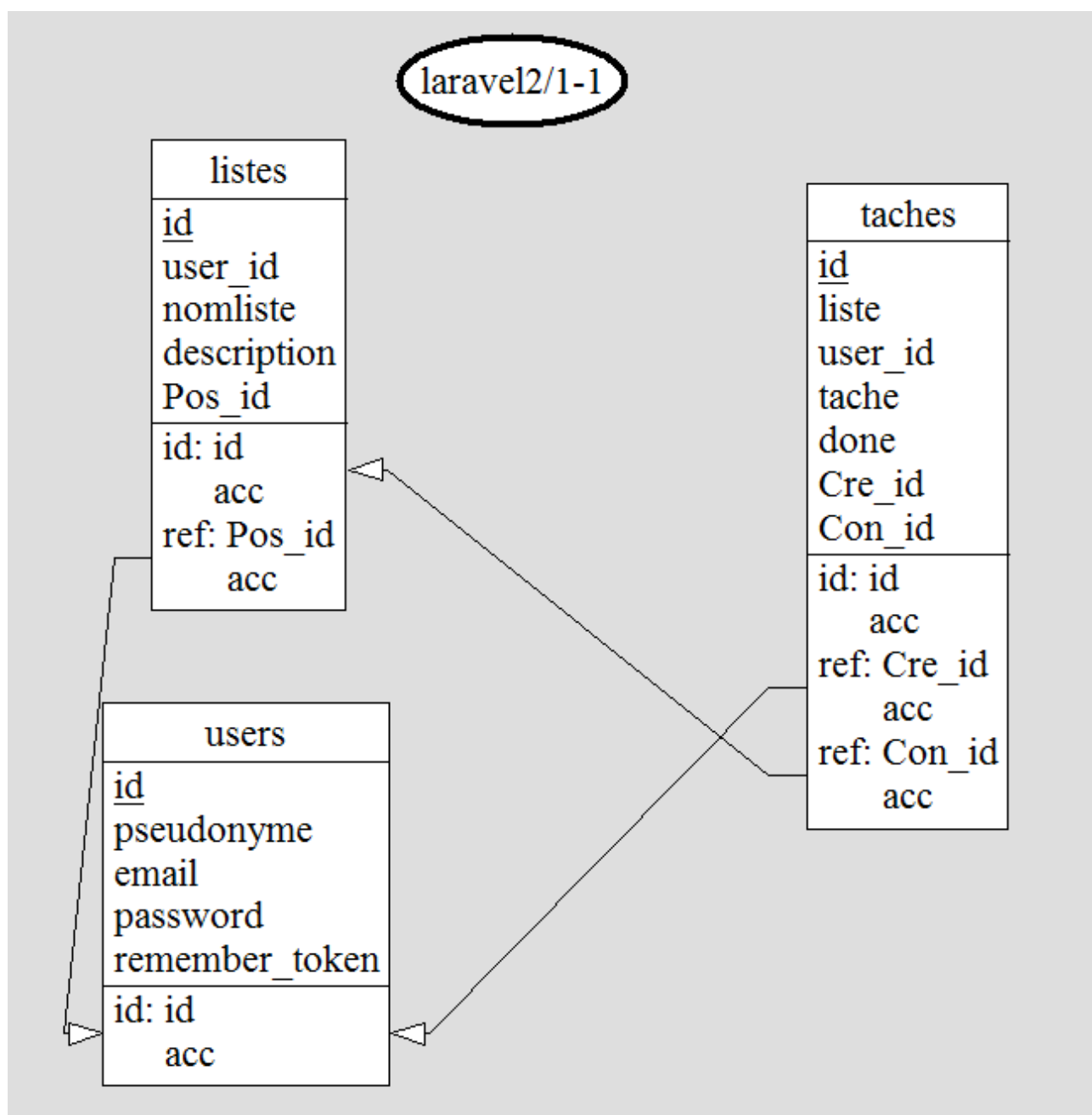
    return redirect()->route('home');
}
```


BDD

Trois tables ont été utilisées pour mettre en œuvre ce site, voici le schémas Entités-Associations.



Modèle conceptuel :



Sécurité mise en place

Pour pallier aux attaques par script, la balise `htmlentities()` a été utilisée pour toutes les valeurs enregistrées dans la base de données. De plus, la base de données est protégée par un mot de passe et le numéro de port par défaut de MAMP a été modifié (aussi bien pour le service Apache que pour le service SQL).

Point de vue sur Laravel

Laravel est un bon framework pour développer des sites de manière assez rapide et complète. Il met en place une architecture de programmation liée au MVC. Mais il nécessite une bonne connaissance de base en PHP, ce qui n'était pas mon cas. De ce fait nous avons éprouvé beaucoup de difficultés à cause de ces lacunes pour pouvoir arriver au bout de ce projet.

De plus, la documentation était bien présente, mais, ne nous aidait pas forcément pour ce que nous voulions faire.

La gestion des bases de données a été particulièrement pénible. Le modèle MVC n'étant pas notre point fort, nous avons également eu du mal avec cette partie du framework.

Conclusions

Le site a été conçu en respectant toutes les contraintes imposées par notre professeur. Un utilisateur a la possibilité de se créer un compte, une liste de tâches, de supprimer une liste ou une tâche et de valider une tâche.

Le site est un minimum sécurisé, les bugs via URL ne sont pas possible, car des redirections ont été mises en place pour pallier à ce problème.

Nous avons apporté toutes nos connaissances concernant les base de données et la programmation web sur ce projet.

Par contre, ce serait intéressant de retravailler le code pour que le site soit *mobile responsive*. Dans le cas présent, seul la barre de navigation l'est.

La liaison via un compte gmail pour avoir des notifications, l'envoi de nouveau mot de passe par mail sont des points importants à mettre en place pour que le site soit entièrement opérationnel.

Ce projet nous a permis d'approcher de près Laravel, qui est un framework puissant mais très difficile à maîtriser.