Faculté des Sciences & Techniques

2016-2017

Conception d'application informatique

Thème : Gestion d'emploi du temps fixe et modulable d'une entreprise

Par :
CAMARA Kerfala
ENGBAKA-MOLEKA Michael

Encadré par : **Benoît CRESPIN**

Table des matières

I. Introduction 2

II. Présentation du concept MVC 4

1. Le modèle : 4

A. Les différents éléments constitutifs du modèle : 4

B. Les modèles utilisés : 6

2. Le contrôleur : 8

A. Les Contrôleurs utilisés : 9

B. Notion Validateur laravel : 10

3. La vue : 12

A. Notion héritage sur les vues laravel : 13

B. Outils web et Html : 14

III. Routage laravel : 21

IV. Les problèmes 22

1. Problèmes résolus : 22

2. Problèmes partiellement résolus : 23

V. Possible Amélioration 23

VI. Conclusion : 24

VII. Bibliographie 25

I. Introduction

Dans le cadre du cours de Conception d'application informatique , sur le choix de la réalisation d'une application informatique , nous avons choisis de réaliser une Application Web qui devra gérer les horaires fixes et modulables des employés et des étudiants au sein d'une Entreprise.

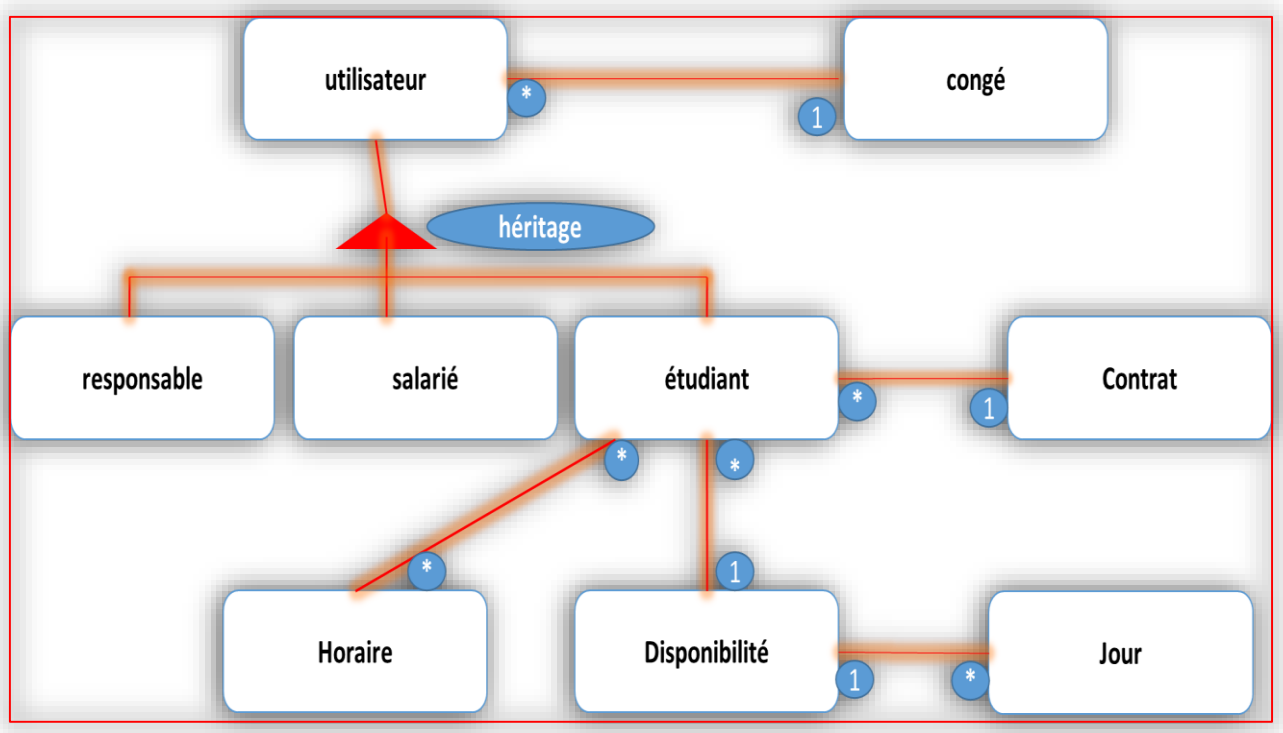
Pour concevoir cette application, nous avons besoin pour le système d'information, d'un Administrateur (gérant de l'application), des employés et des étudiants.

L'administrateur du système pourra donc créer un compte pour chaque employés et étudiants, lui fournir ses identifiants avec lesquels il pourra se connecter sur l'interface dédié à ce sujet. Il pourra aussi établir et opérer des modifications sur les plannings, valider des congés et aussi faire des propositions des heures supplémentaires si nécessaire aux étudiants en fonction de leurs disponibilités.

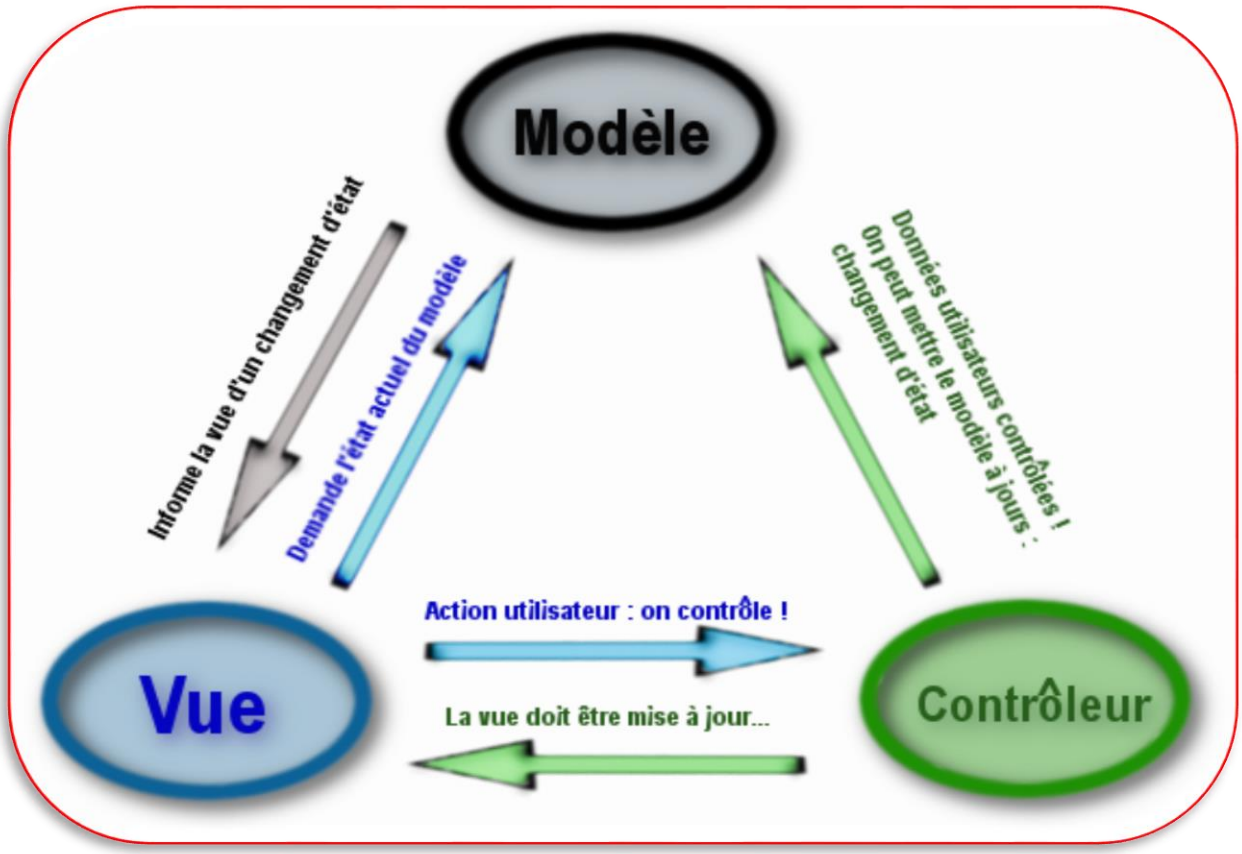
Une fois connecté, l'employé pourra ainsi voir son planning des deux prochaines semaines, il pourra sur l'interface proposer ses semaines de congé de l'année.

Et pour les étudiants, les horaires seront modulable, ils devront donc proposer leurs disponibilités sur base des quelles leurs plannings seront faits.

Modélisation UML :



II. *Présentation du concept MVC*



1. Le modèle :

La couche modèle représente la partie de l'application qui exécute le logique métier. Cela signifie qu'elle est responsable de récupérer les données, de les convertir selon des concepts chargés de sens pour votre application, tels que le traitement, la validation, l'association et beaucoup d'autres tâches concernant la manipulation des données. A première vue, l'objet modèle peut être vu comme la première couche d'interaction avec n'importe quelle base de données que vous pourriez utiliser pour une application. Mais plus globalement, il fait partie des concepts majeurs autour desquels vous allez exécuter votre application.

A. Les différents éléments constitutifs du modèle :

Le modèle est composé :

1. D'un attribut protégé, qui permet de préciser les attributs insérables dans d'une table par un utilisateur.

```php
protected $fillable = ['jour_id','etudiant_id','heureDebut','heureFin','valider'];
```

2. Des fonctions de liaison entre deux tables, qui permet de définir le type de liaison en deux tables (**1..n, n..m**, **1..1**) pour implémenter ses types de liaisons laravel dispose des mots clés :

   - **hasOne** « permet de gérer relation **1..1** »
   - **belongsTo** « quand une table contient la clé étrangère une autre table **1..n** »
   - **hasMany** « quand clé étrangère peut se retrouver plusieurs dans une autre table **1..n** »
   - **belongsToMany** « permet de gérer la relation **n..m** »

```php
public function utilisateur(){
    return $this->belongsTo('App\Utilisateur');
}
```

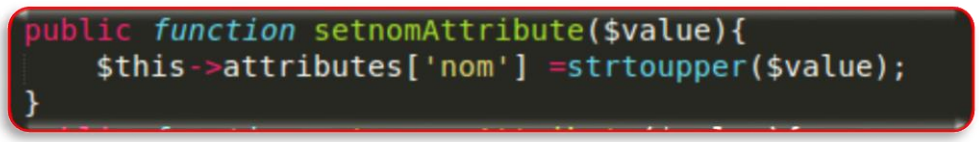
3. Des requêtes prédéfinis sur la table (Modèle) à l'aide du mot clé scopeNon_requête().

```php
public function scopeNonValidedisponible($query,$q){
    return $query->where('valider',false)->where('etudiant_id',$q);
}
```

4. Des fonctions simples qui peuvent être utilisé au besoin quand en instancie un objet de type Modèle

```php
public function db_calendrier($numero_jour){
    $la_date = carbon::now();
    if ($numero_jour =="1")
        return $la_date->next(carbon::MONDAY);
        else if ($numero_jour == "2")
            return $la_date->next(carbon::TUESDAY);
        else if ($numero_jour == "3")
            return $la_date->next(carbon::WEDNESDAY);
        else if ($numero_jour == "4")
            return $la_date->next(carbon::THURSDAY);
        else if ($numero_jour == "5")
            return $la_date->next(carbon::FRIDAY);
        else if ($numero_jour == "6")
            return $la_date->next(carbon::SATURDAY);
        else if ($numero_jour == "7")
            return $la_date->next(carbon::SUNDAY);
}
```

5. Getteurs et setteurs pour les modifications des données envoyées à la base de données.



```
public function setnomAttribute($value){
    $this->attributes['nom'] =strtoupper($value);
}
```

B. Les modèles utilisés :

Les différents modèles que nous avons utilisés sont les suivants :

- Utilisateur.php
  - ✓ **Attribut insérable** : nom, prenom, email, motDePasse
  - ✓ **Fonctions de liaison entre tables :**
    - ✓ **hasOne** : etudiant(), salarie(), responsable()
    - ✓ **hasMany**: conges().
  - ✓ **Requêtes prédéfinies** :
    SearchByNom(), SearchByPrenom(), SearchByNomPrenom()

- **Conges.php**
  - ✓ **Attribut insérable** : dateDebut, dateFin, Validaion, utilisateur_id
  - ✓ **Fonctions de liaison entre tables** :
    - ✓ belongsTo : utilisateur()
  - ✓ **Requêtes prédéfinies** :
    NonValide(), tousNonValide(), IdConge(), IdCongevaleur()

- Etudiant.php
  - ✓ **Attribut insérable** : utilisateur_id, contrat_id
  - ✓ **Fonctions de liaison entre tables** :
    - ✓ belongsTo : utilisateur()

- ✓ belongsToMany : jours(), horaire()

- ✓ **Requêtes prédéfinies** :
  SearchByutilisateur(), SearchByutilisateurId(), SearchByContratId()

❖ Disponibilite.php

- ✓ **Attribut insérable** : jour_id, etudiant_id, heureDebut, heureFin, valider
- ✓ **Fonctions de liaison entre tables** :
  - ✓ belongsToMany : etudiant() , jour()
- ✓ **Requêtes prédéfinies** :
  NonValidedisponible(), tousNonValidedisponible()
- ✓ **Autres fonctions :**
  Convertir ($jour) {…}, db_calendrier($numero_jour) { …}

❖ Jour.php

- ✓ **Attribut insérable** : nom_jour
- ✓ **Fonctions de liaison entre tables** :
  - ✓ belongsToMany : etudiants()

❖ Responsable.php

- ✓ **Attribut insérable** : utilisateur_id
- ✓ **Fonctions de liaison entre tables** :
  - ✓ belongsTo : utilisateur()

❖ Salarie.php

- ✓ **Attribut insérable** : utilisateur_id
- ✓ **Fonctions de liaison entre tables** :
  - ✓ belongsTo : utilisateur()

Requêtes prédéfinies

❖ Contrat.php

- ✓ **Attribut insérable** : type

- ✓ **Fonctions de liaison entre tables** :
  - ✓ belongsTo : etudiant**()**
- ✓ **Requêtes prédéfinies**

  SearchByContratId()

❖ Horaire.php

- ✓ **Attribut insérable** : jour, heureDebut, heurefin
- ✓ **Fonctions de liaison entre tables** :
  - ✓ belongsTo : etudiant()

2. Le contrôleur :

C'est le module qui traite les actions de l'utilisateur, modifie les données du modèle et de la vue. Un contrôleur en laravel est gérer par un objet Controller il suffit de le créer via le terminale par la commande php artisan make : controller nom_Controller, grâce à ce contrôleur nous pouvons écrire des fonctions spécifiques et appeler leurs appelées dans une routes voulues « fichier routes.php », il est à noter qu'un objet Controller laravel des fonctions prédéfinies qui sont :

- index (): envoi une vue par défaut avec toutes les données nécessaire
- edit (): permet d'éditer un modèle (une table) via cette fonction
- store (): enregistrement d'un modèle via un formulaire
- create (): création d'une ligne dans une table
- show ($id): afficher une autre vue hors l'index
- destroy ($id) : supprimer un enregistrement (modèle)
- update ($id,$request): modifier un enregistrement (modèle)

A. *Les Contrôleurs utilisés :*

Les différents contrôleurs que nous avons utilisés pour la réalisation de notre application sont les suivants :

- **connexionController**

c'est l'ensemble des gestions liées à la connexion des utilisateurs grâce aux fonctions store() et show(). La fonction store() permet de voir si les informations utilisateur sont correctes sinon renvoie une erreur « message flash erreur » show() permet de charger la pages correspondantes selon utilisateur soit étudiant, responsable ou salarié si le store() est correctes.

- **DisponibleController**

les fonctions store(), et update() sont utilisés dans disponibleController pour la gestion de la table disponibilité, store pour la création d'une disponibilité pour un étudiant et update pour la valider une disponibilité.

- **RechercheController**

la fonction store() permet de traiter la recherche fait dans les deux zones de saisies (valider congé et valider disponibilité) puis redirige vers la pages correspondantes avec les données nécessaires pour ces pages.

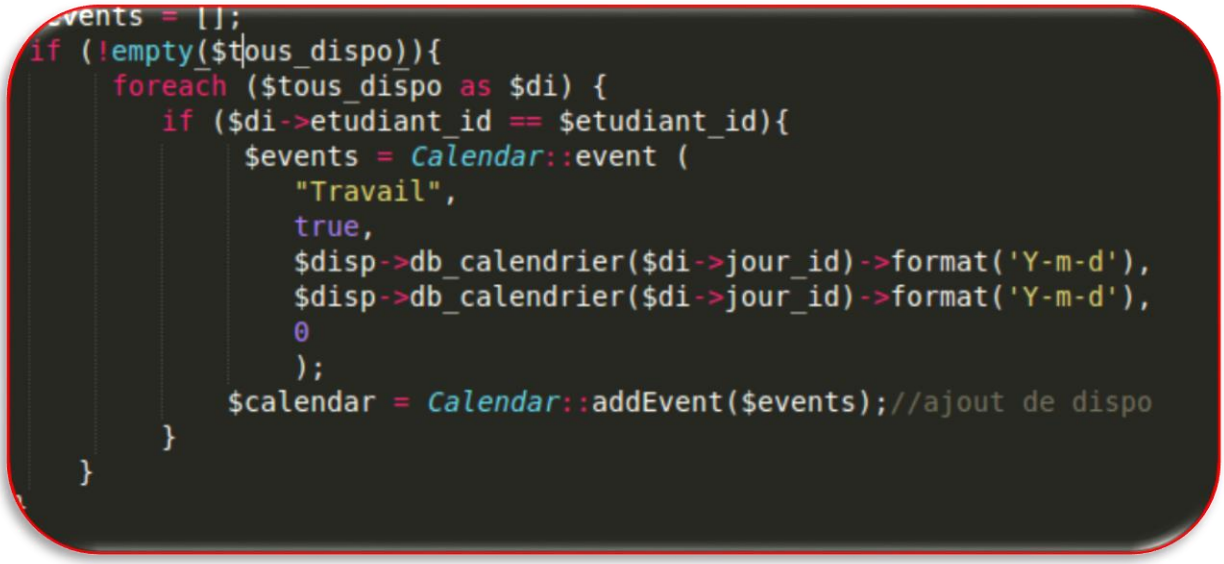
- **UtilisateurController**

Il utilise plusieurs fonctions : le store() qui permet l'enregistrement d'un utilisateur ; il est à noter que le store() se fait en fonction du type d'utilisateur choisi par exemple pour un étudiant il aura création d'un contrat et les autres types d'utilisateur non, le

update() pour la modification des informations concernant un utilisateur, le edit() charge tout simplement la vue editUtilisateur qui nous permettras d'éditer.

- **EtudiantController**

Les fonctions store() et show() ont été utilisé, la fonction ici sont but est à partir d'une liste d'étudiants quand on sélectionne une personne, elle nous renvoie sur la vue permettant soit de valider les congés, soit les disponibilités ou modifier le profil utilisateur. La fonction show() est très importante dans ce contrôleur car c'est grâce à elle quand un étudiant est connecté ont gère tous ceux qui est disponibilité de l'étudiant et mettre dans un calendrier à l'aide la façade Calendar ajouter dans le projet, après il redirige à la vue pour étudiant avec la variable calendrier rempli et l'utilisateur concerné.



```
vents = [];
if (!empty($tous_dispo)){
    foreach ($tous_dispo as $di) {
        if ($di->etudiant_id == $etudiant_id){
            $events = Calendar::event (
                "Travail",
                true,
                $disp->db_calendrier($di->jour_id)->format('Y-m-d'),
                $disp->db_calendrier($di->jour_id)->format('Y-m-d'),
                0
                );
            $calendar = Calendar::addEvent($events);//ajout de dispo
        }
    }
```

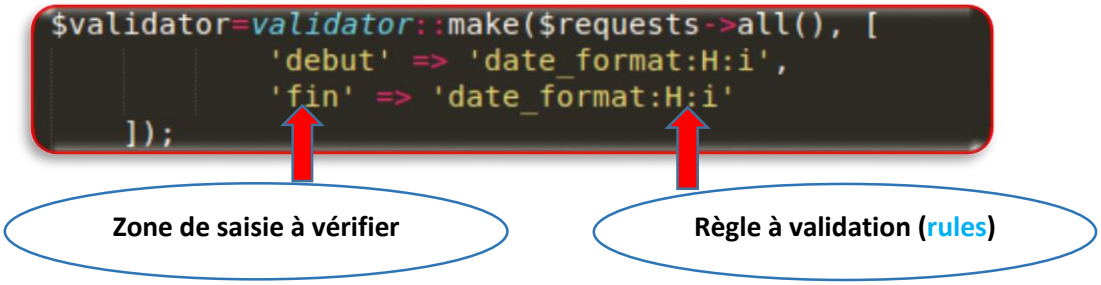
- **SalarieController**

Ce contrôleur utilise le même concept que EtudiantController sauf la différence un salarié n'a pas besoin de disponibilité, il travaille les 5 jours de la semaine.

B. Notion Validateur laravel :

Presque toutes les applications Web interactives doivent valider les données. Par exemple, un formulaire d'inscription nécessite probablement le mot de passe à confirmer. Peut-être que l'adresse e-mail doit être unique. Les validations vous permettent de s'assurer que seules les données valides sont enregistrées dans la base de données, la validation données permettra d'éviter les injections Sql ce qui est un atout pour la sécurité de l'application.

En résumé une validation laravel se fait par l'objet validator en faisant passer les zones saisies à des règles laravel prédéfinies « rules ».

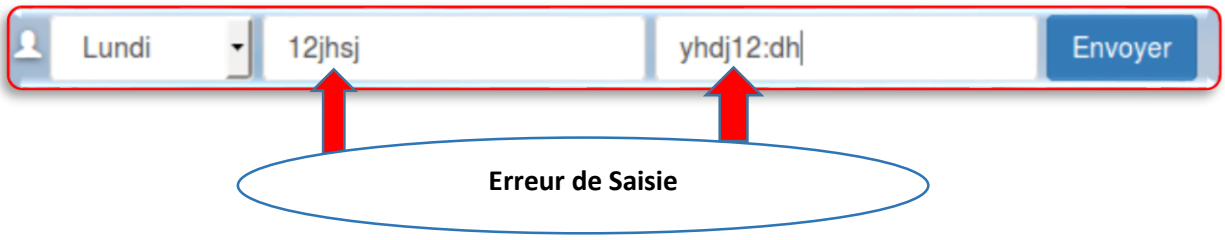


Après il suffit de vérifier si l'objet **validator** via sa methode **fails** est à vraie signifie qu'il y a une erreur sur les données sinon on peut enregistrer les données car elles sont valides

En pratique nous avons :

- ✓ **Avant saisie** :

- ✓ **Saisie** :



- ✓ **Clique bouton envoyer :**

3. La vue :

C'est la partie visible d'une interface graphique. La vue se sert du modèle, et peut être un diagramme, un formulaire, des boutons, etc. Une vue contient des éléments visuels ainsi que la logique nécessaire pour afficher les données provenant du modèle. Dans une application laravel une vue peut contenir du html, css, javascript, ajax, jquery…
Dans l'élaboration du projet, les vues utilisées dans le projet :
Avant de citer la liste des vues il faut noter que la spécificité d'une vue laravel est le nom de chaque vue est suivi par .blade.php exemple : editerUtilisateur.blade.php

- **Default,** la vue par défaut où la plupart des vues hérite.
- **editerUtilisateur,** permet d'éditer un utilisateur
  Donnée envoyer à cette vue : **$connecter**
- **etudiantDisponible,** permet de gérer les disponibilités non valides d'un étudiant
  - ✓ Donnée envoyer à cette vue :

  $connecter, utilisateur connecter, réalisant l'opération
  $users, tous les utilisateurs concernés
  $etudiants, tous les étudiants concernés
  $dispos, toutes les disponibilités non validées.

- **modifieEtudiant,** cette vue gère à la fois la validation des congés, les disponibilités si étudiant et éditer cet utilisateur.
  - ✓ Donnée envoyer à cette vue :

  $connecter, utilisateur connecter, réalisant l'opération
  $jours, tous les jours de la semaine (besoin pour la liste des disponibilités)
  $conges, tous les congés non validés.
  $dispos, toutes les disponibilités non validés.
  $personne, la personne qui a été sélectionner par utilisateur

- **Validation,** permet la validation des congés d'un utilisateur
  - ✓ Donnée envoyer à cette vue :

  $connecter, utilisateur connecter, réalisant l'opération

$users, tous les utilisateurs concernés

$conges, tous les congés non validés.

- **Responsable,** vue de gestion d'un utilisateur responsable
  - ✓ Donnée envoyer à cette vue :

    $connecter, utilisateur connecter, réalisant l'opération

    $users, tous les utilisateurs de la table utilisateur (besoin pour la liste)

    $etudiants, tous les étudiants de la table étudiant (besoin pour la liste)

- **Connection,** la vue permettant à un utilisateur de se connecter
- **Etudiant,** vue de gestion d'un utilisateur étudiant
  - ✓ Donnée envoyer à cette vue :

    $connecter, utilisateur connecter, réalisant l'opération

    $calendar, données qui remplissent le calendrier quand un étudiant se connecte.
- **Salarie,** vue de gestion d'un utilisateur salarié
  - ✓ Donnée envoyer à cette vue :

    $connecter, utilisateur connecter, réalisant l'opération

    $calendar, données qui remplissent le calendrier quand un salarié se connecte.

- **Succes,** permet d'afficher des messages de succès si une opération est correcte, cette vue à une spécificité car elle n'hérite pas de la vue default et en l'utilisant nous la incluons dans les autres vue par **:@include('succes)**
- **Index,** la vue qui contient le home page

A. Notion héritage sur les vues laravel :

L'héritage des vues comme son nom l'indique est un concept informatique qui consiste à éviter la réécriture du code, puisque les différentes pages utilisées dans un projet ont tendance à avoir des parties communes, l'idée est de les regrouper dans une seule et unique vue « default.blade.php » puis appeler cette vue quand on a en besoin.
Elle se fait dans laravel à partir de la commande :

@extends('default')

@section('partie_concernée ')

< !- - codage spécifique lié à la page concerné -- >

@endsection

Il faut préciser aussi dans **default.blade.php** dans quelle zone nous voulons afficher le code de la section en faisant à l'aide de la directive **@yield('partie_concernée')**

- enfin il faut aussi noter les inclusions d'une vue dans une autre grâce à la directive **@include('nom_de_la_vue')**, ceci est utilisé plus dans le cas de messages flash qui sont formatés dans une vue spécifique puis inclus dans une autre vue.

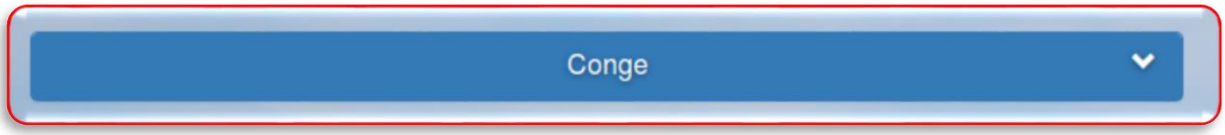
B. Outils web et Html :

Laravel est framework qui utilise par défaut : bootstrap (gestion responsive des pages web), et javascript (interactions dynamique côté machine).
Ce projet utilise :

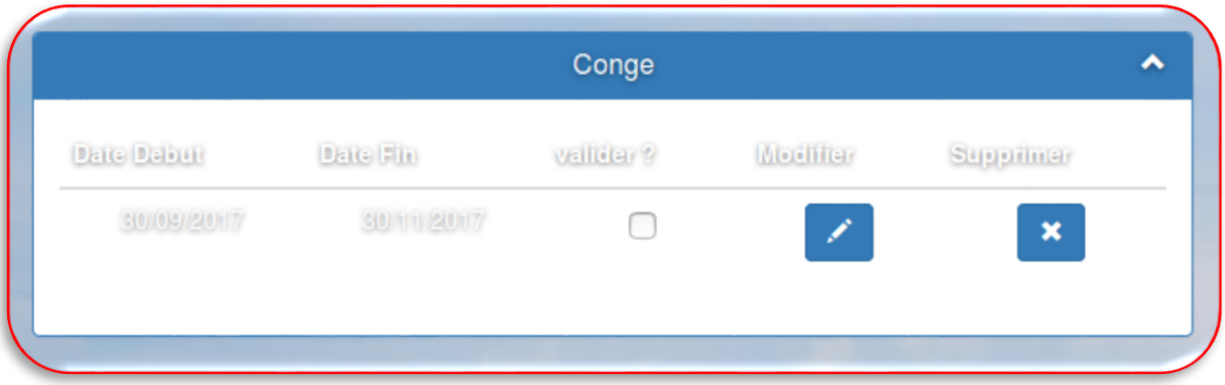
- **Jquey**

  Panel.js, permet la création d'un panel sur une personne en regroupant les informations dans le contenu du panel « capture »

  ✓ Avant clique :



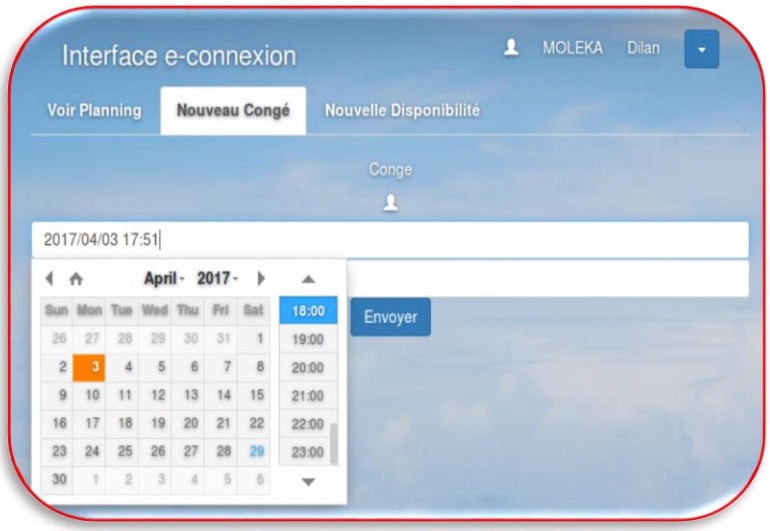
  ✓ Après clique :



  Datetimepicker.js, quand un utilisateur veut saisir les informations liées à d'une date, on propose à celui-ci un calendrier prédéfini.

✓ Avant clique

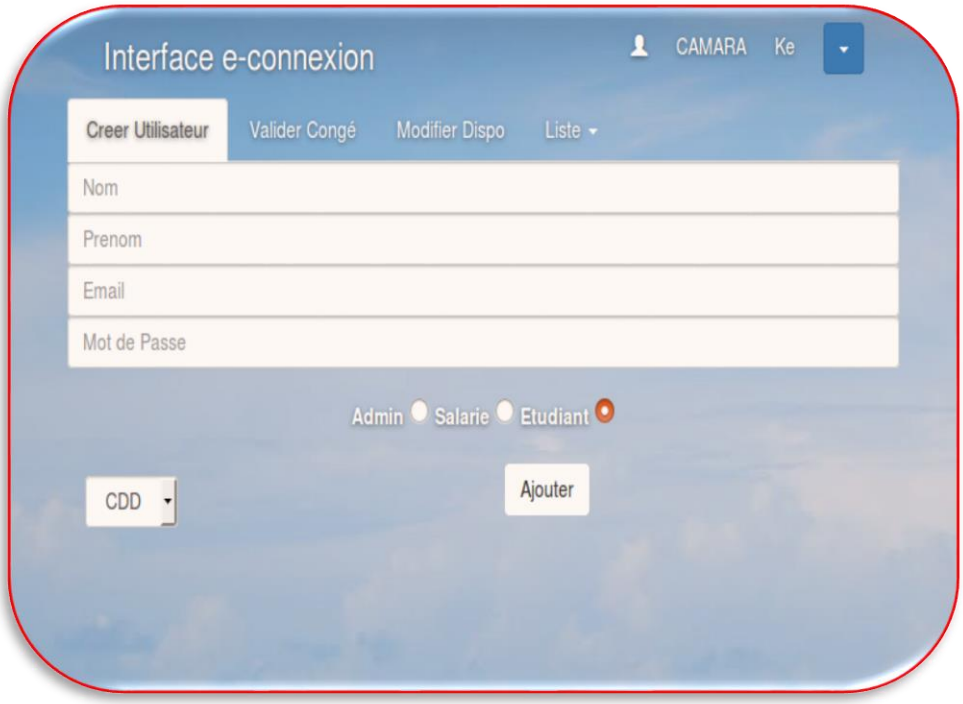
✓ Après clique



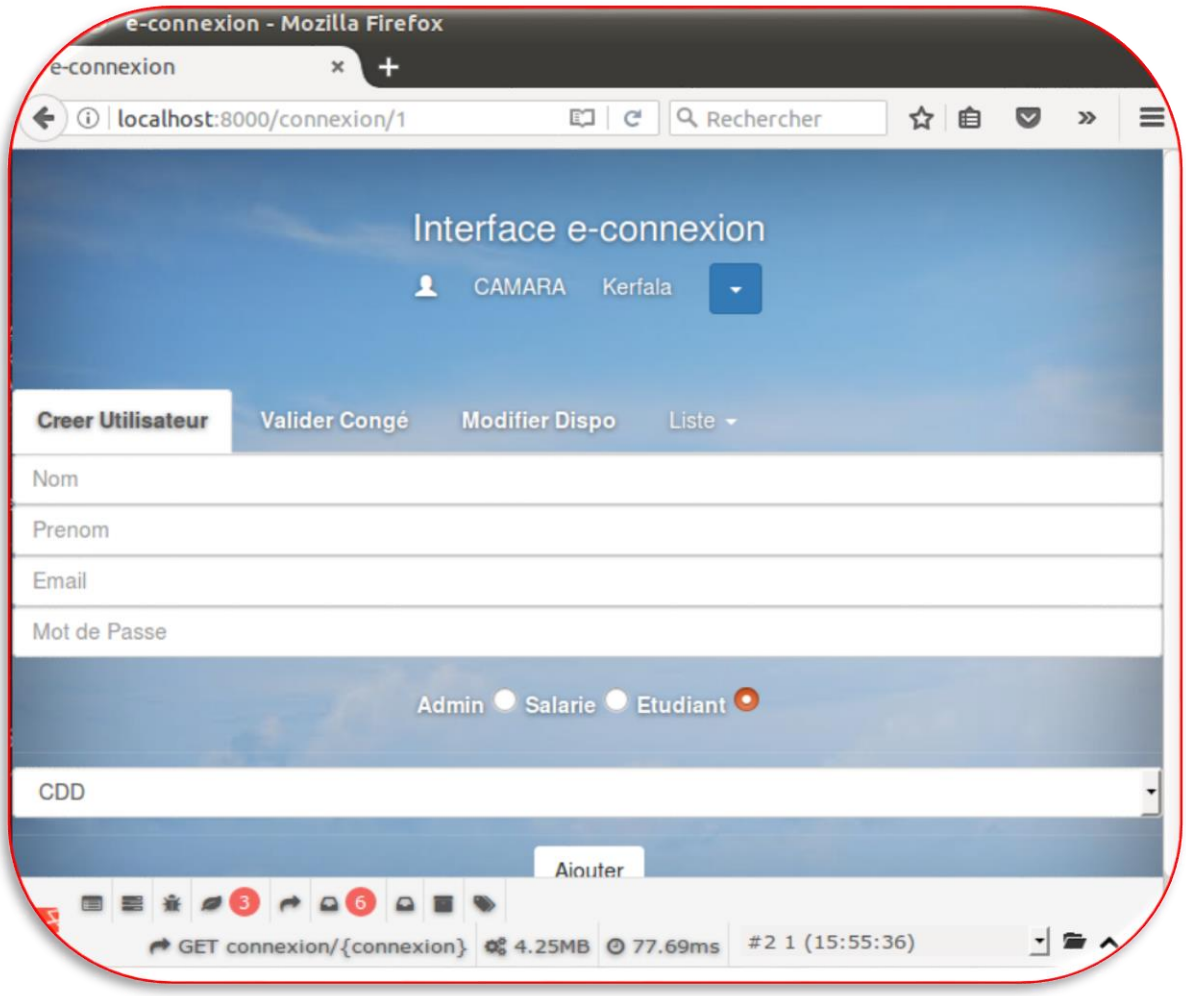
- **Bootstrap**

Bootstrap permet la gestion responsive des pages web de l'application, elle est créé avec le css et le jquery, ceci est élucidé dans notre projet par :

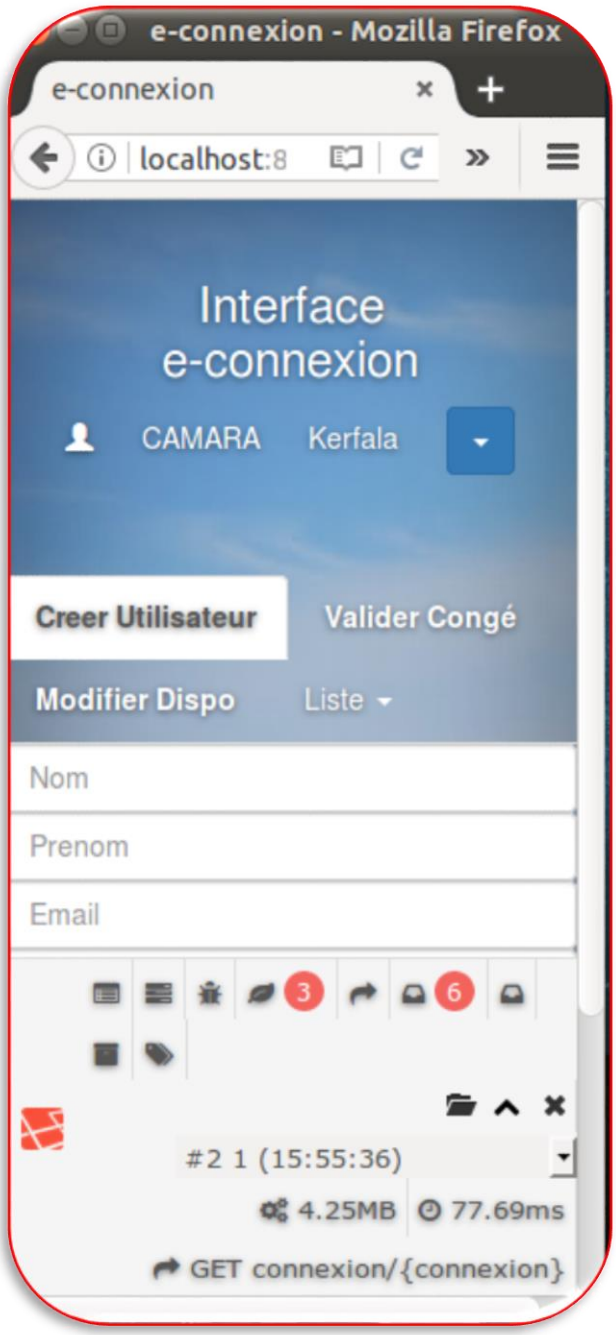
✓ Taille écran normale



✓ Taille écran tablette



✓ Taille écran mobile



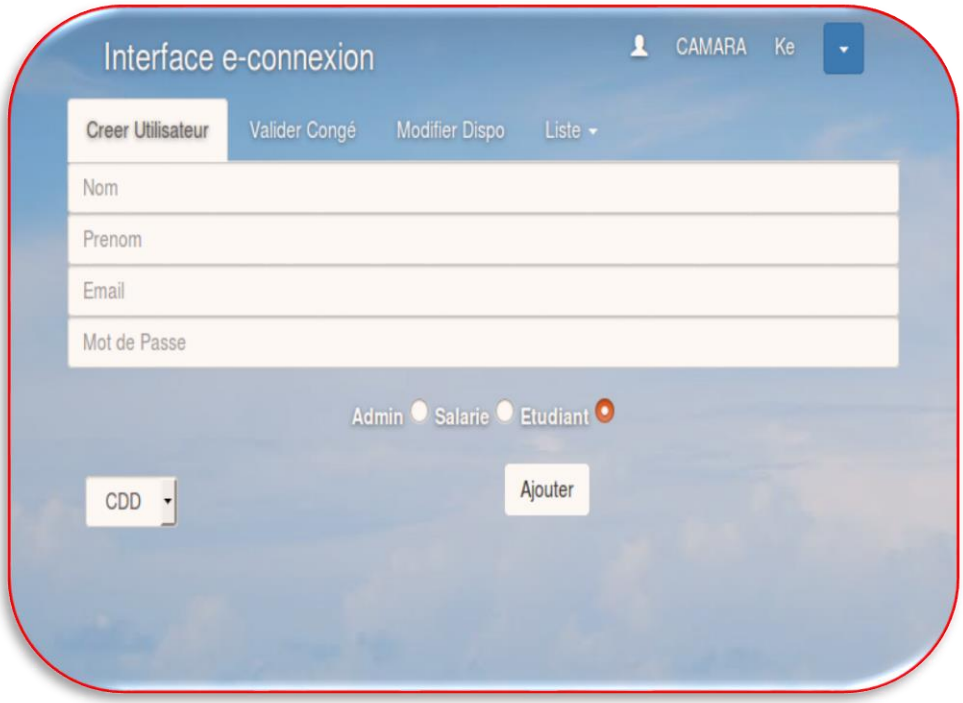
- **Css et Html**

Dans ce projet un seul fichier css cover.css, qui est utilisé pour la vue default.blade.php (design principale de l'application « où la plus part des vues hérite de celle-ci ») mais il faut noter aussi qu'on a utilisé du css dans d'autres vues pour des raisons spécifiques d'où il y avait pas nécessité d'utilisé un fichier css, le code était directement inclus dans la vue correspondante.

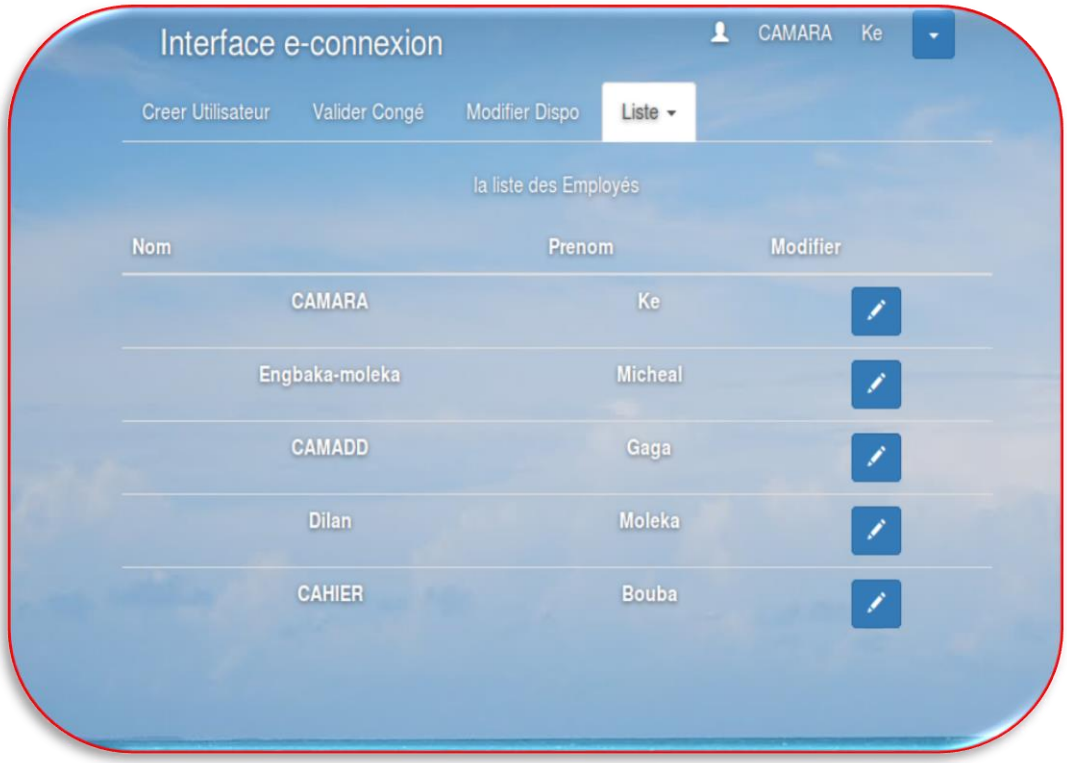
- **Quelques pages web utilisées dans le projet**

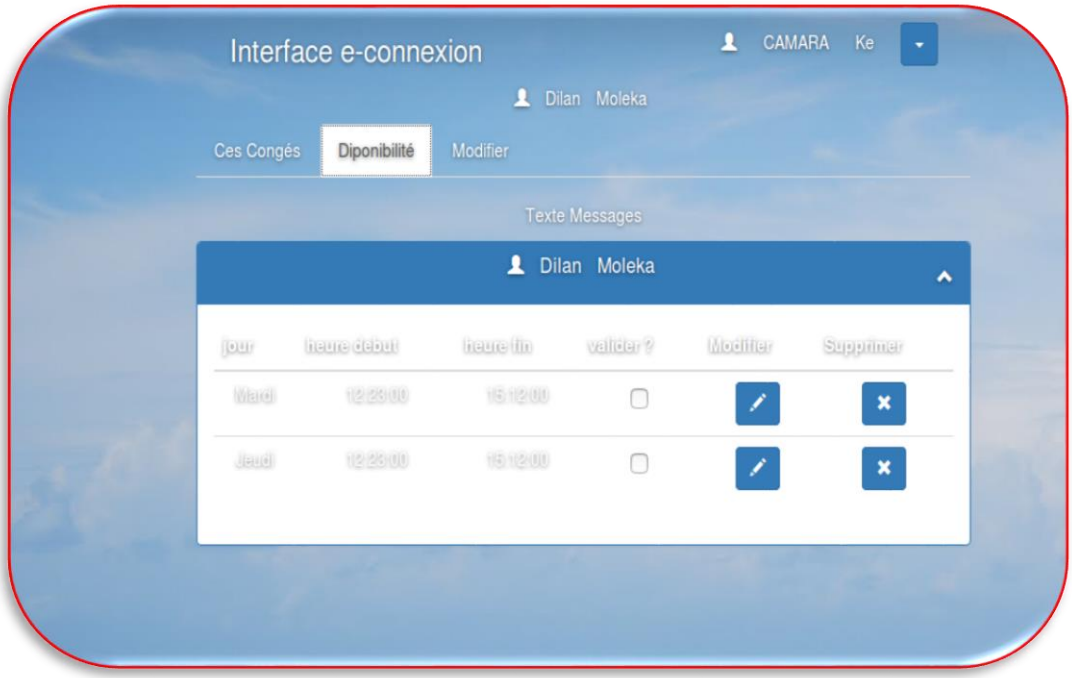
  Interface responsable

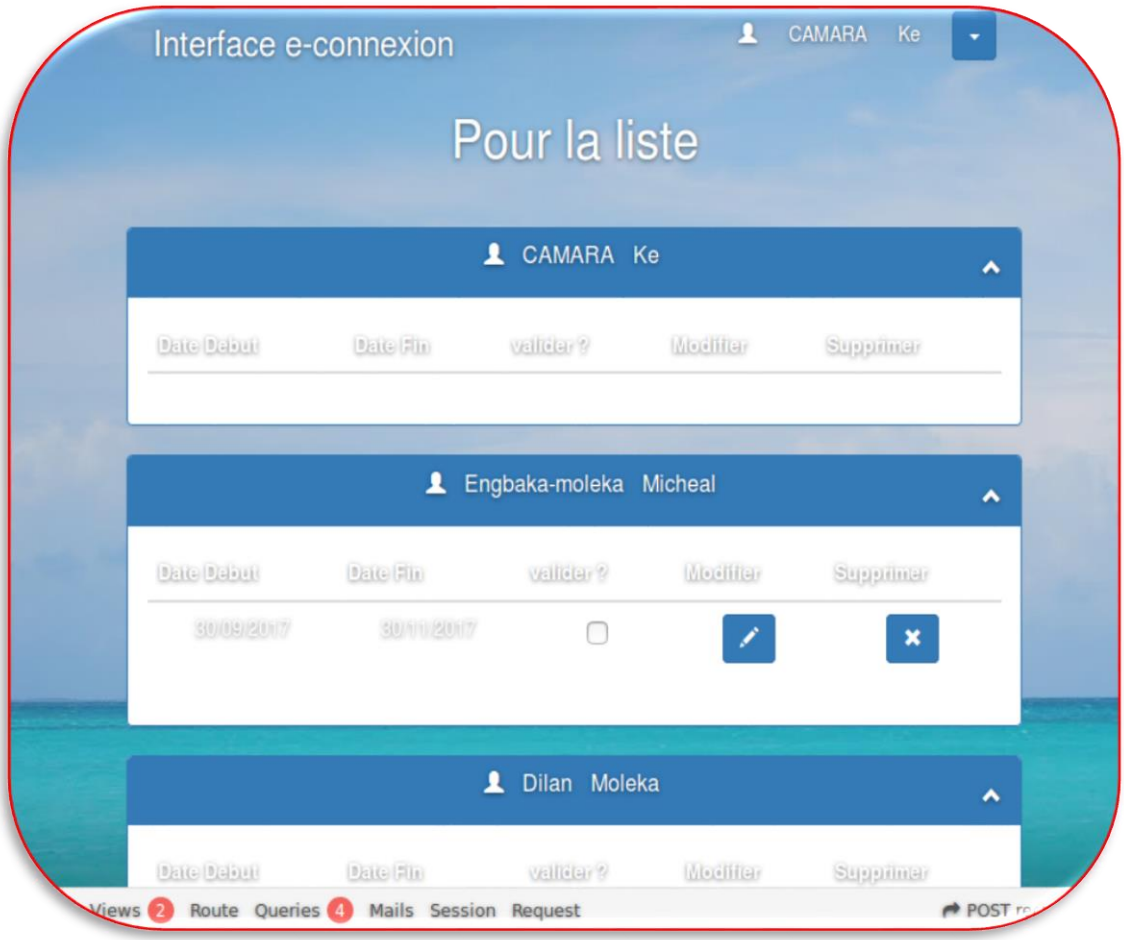
- Enregistrement employé

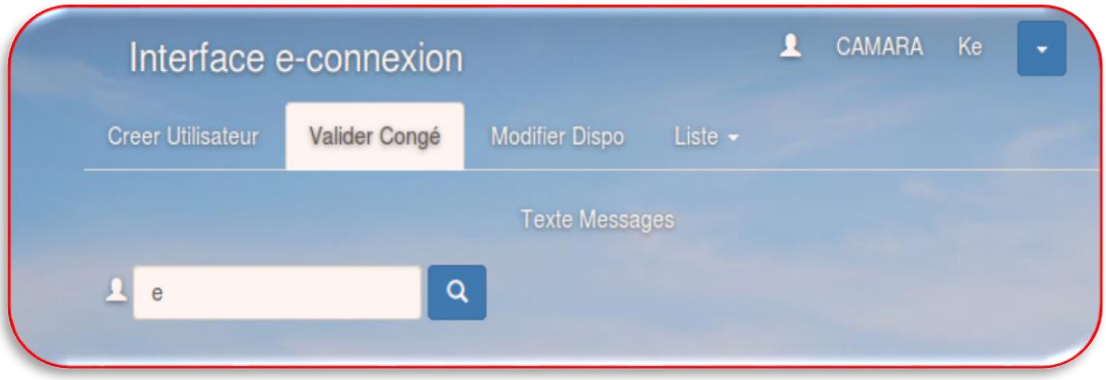


- Liste des employés

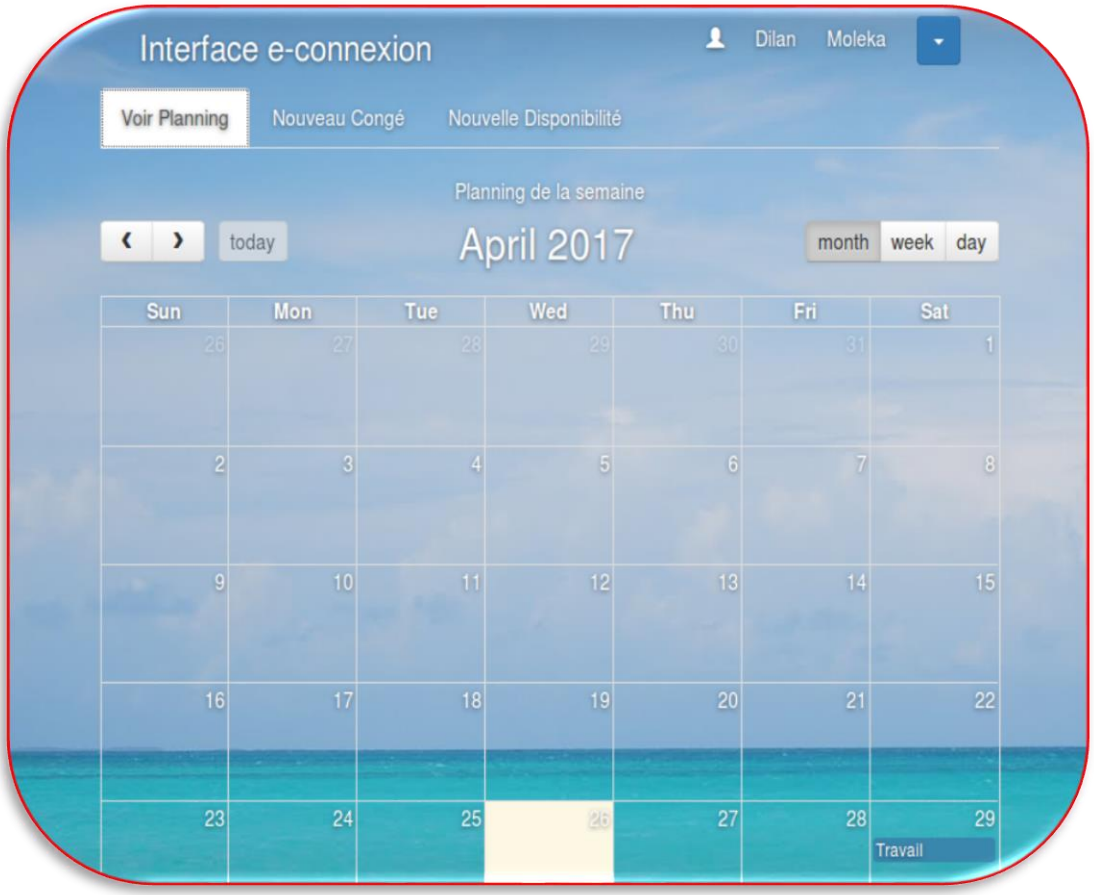



Interface e-connexion
CAMARA Ke
Dilan Moleka
Ces Congés
Diponibilité
Modifier
Texte Messages
Dilan Moleka
jour
heure debut
heure fin
valider ?
Modifier
Supprimer
Mardi
12:23:00
15:12:00
Jeudi
12:23:00
15:12:00

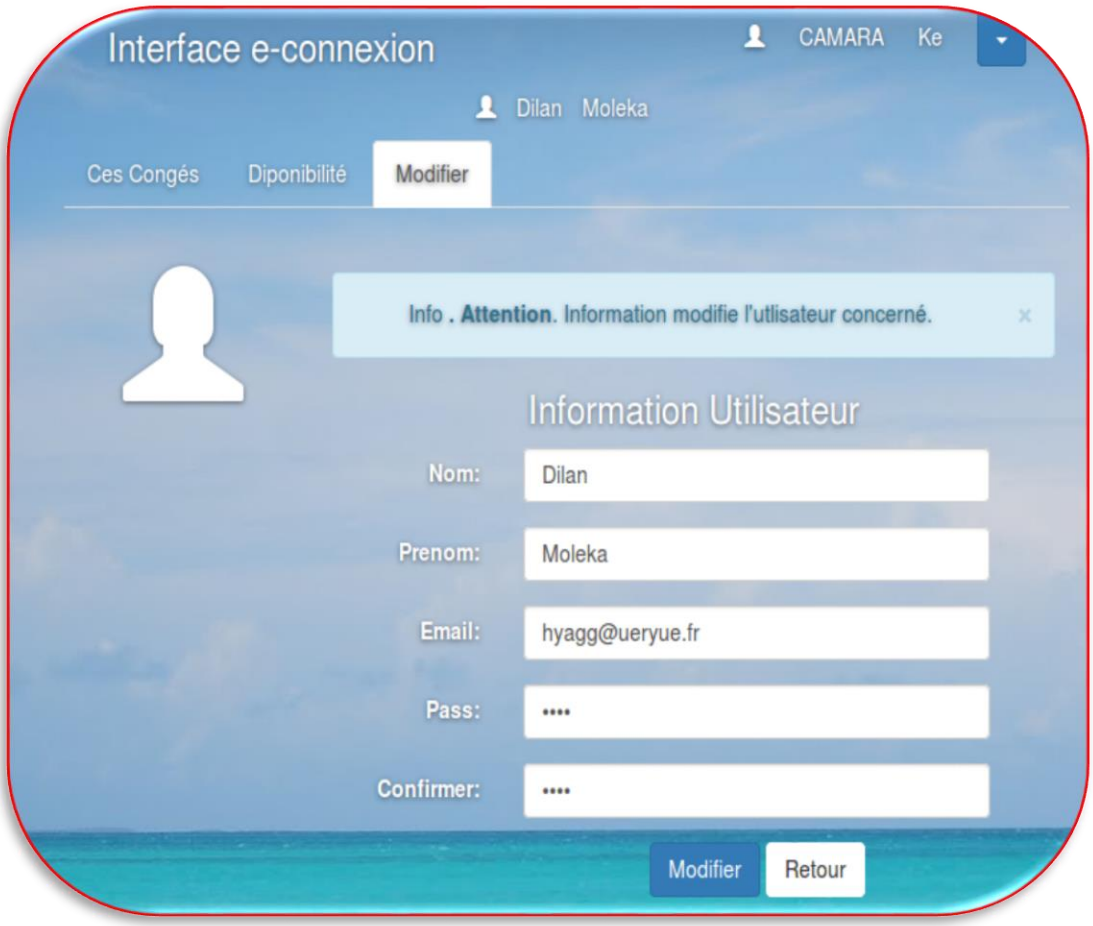

Interface e-connexion
CAMARA Ke
Pour la liste
CAMARA Ke
Date Debut
Date Fin
valider ?
Modifier
Supprimer
Engbaka-moleka Micheal
Date Debut
Date Fin
valider ?
Modifier
Supprimer
30/09/2017
30/11/2017
Dilan Moleka
Date Debut
Date Fin
valider ?
Modifier
Supprimer
Views 2
Route
Queries 4
Mails
Session
Request
POST



Interface étudiant ou salarié :



Interface commune :



III. *Routage laravel :*

Les routes en laravel se fait dans le fichier routes.php qui se trouve le répertoire **Controllers** d'un projet laravel.

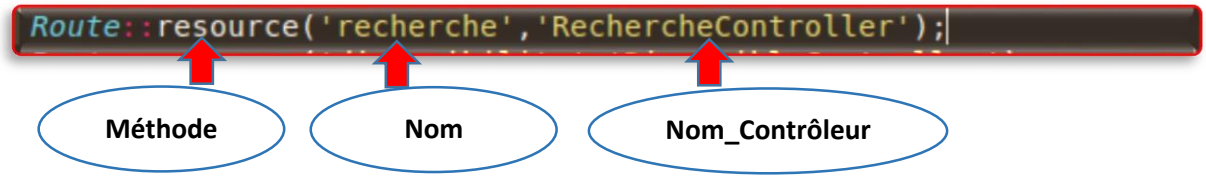
La spécificité avec laravel est qu'elle dispose un objet Route qui dispose des méthodes : **get**, **post**, **put**, ***resource**

Get : permet d'aller au contrôleur correspondant préciser dans la méthode.

Post : méthode post pour un contrôleur à préciser

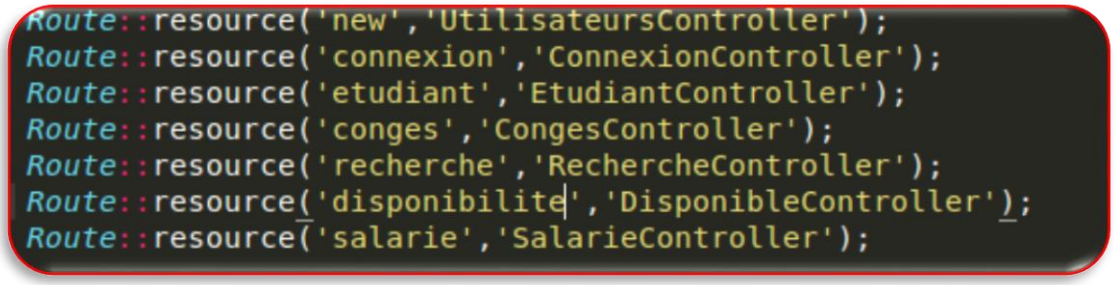
Put : méthode put pour un contrôleur à préciser

***Resource** : la méthode la plus utilisée dans le cadre du projet, elle permet de disposer de toutes les fonctions d'un contrôleur par le nom de cette ressource « '**nom.fonction_contrôleur**' »



Utilisation de cette ressource ce fait par **nom.fonction_controleur**

L'ensemble des ressources utilisées pour ce projet sont représentées par cette capture ci-dessous :

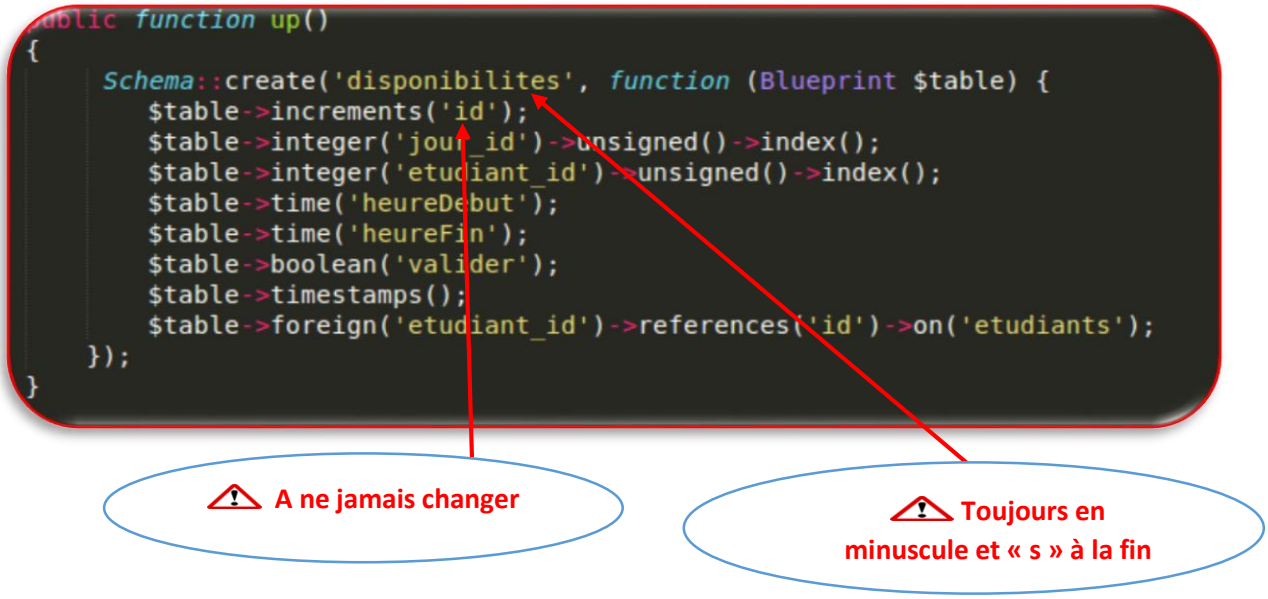


IV. Les problèmes

1. Problèmes résolus :

Créer les tables d'une base de données à partir d'un projet laravel est très simple en apparence mais peut s'avérer très délicate si on ne respecte les règles établies par laravel car c'est grâce à celles-ci que le framework communique avec la base de données ; les problèmes rencontrés qui nous a permis de connaître quelques règles sont :

- **Le nom des tables est en minuscule et se termine par « s »**. pour comprendre cette erreur il nous a fallu quelque jour avant de le résoudre
- **L'identificateur de chaque table est « id » et rien d'autres.** Utilisé un autre identifiant par exemple « **id_conges** ou **id_dispos** » certes peut passer à la création de la base de données ou même à l'insertion des enregistrements mais génère des erreurs pour les opérations de **suppression**, **modification**, ou la recherche d'un élément par la méthode **find** ou **findOrFail** de laravel car le framework se base sur « **id** » pour faire ces opérations. Il nous a fallu des semaines pour comprendre cette erreur.



2. Problèmes partiellement résolus :

Calendrier, bien que nous arrivons à afficher dans le calendrier le jour de travail d'un employé mais la précision au niveau de l'heure n'est pas défini.

Messages flash laravel, elle fonctionne pour aller d'une page à une autre mais ne fonction pas avec les redirections.

V. Possibles Améliorations

- Validation, essayer de faire la validation tous les formulaires possibles
- messages, envoyer des messages mails ou mobile aux utilisateurs quand leur demande a été validé.
- calendrier, faire de telle sorte que on puisse afficher à l'heure exactement dans l'intervalle horaire que l'étudiant travail.
- la base de données, nous pouvons faire améliorer notre base de données en gérant l'ensemble des horaires travaillées par étudiant chaque mois puis faire même des opérations statistique pour mieux comprendre quel est le jour de la semaine où il y a plus employé ou le mois, comparer la productivité par rapport aux années précédentes.

VI. Conclusion :

L'élaboration de ce projet nous a permis de savoir utilisé le framework **laravel** et nous a fait comprendre comment ce framework augmente la productivité d'un développeur web grâce à l'intégration par défaut de **bootstrap css** et **jquery** et la gestion simplifiée des routes pour l'accès des pages en passant par les modèles ou faire évoluer la base de données quand le projet est en cours de développement etc.

Ce projet peut évoluer en version suivante, avec des interfaces plus épurées, modifier la base de données pour les besoins spécifiques d'un client ou même pour gérer plus de données par exemple, les bulletins de salaire, contrat de travail, changement brusque de programme d'un employé à autre etc.

Enfin il faut retenir que nous avons choisi ce projet d'application web car la communication web « **Html** » est le moyen de transmission d'informations le plus utilisés au monde et le sera pour les 30 années à venir à moins qu'il est la téléportation ou autres (chose qui est probablement impossible), c'est pour cela en tant que futur informaticien « **quel que soit la spécialisation** », nous devons savoir-faire au minimum **un simple canal de transmission d'informations d'un bout à un autre** à travers le **langage web** ce que permet de faire **laravel et même plus** sans se prendre la tête à être un spécialiste du domaine web.

VII. Bibliographie

- https://www.grafikart.fr/formations/laravel
- http://elliottchiaradia.ch/blog/article/detail/laravel-5-les-dates
- https://laravel.com/docs/5.4/validation
- https://openclassrooms.com/courses/decouvrez-le-framework-php-laravel/presentation-generale-13
- http://askubuntu.com/questions/824485/how-to-activate-php7-0-pdo-in-16-04-lts
- http://stackoverflow.com/questions/2852748/pdoexception-could-not-find-driver
- https://bhavyanshu.me/tutorials/setup-laravel-project-on-xampp-linux/01/12/2015
- https://www.grafikart.fr/forum/topics/19798
- https://laracasts.com/discuss/channels/laravel/carbon-date-in-french
  http://stackoverflow.com/questions/3288257/how-to-convert-mm-dd-yyyy-to-yyyy-mm-dd