



Université Abdelmalek Essaadi

Faculté des sciences et techniques de Tanger

Département d'Informatique

Mini Projet

CYCLE D'INGÉNIEUR

LOGICIELS ET SYSTÈMES INTELLIGENTS

**Sujet: Application Web
gestion LSI.**

Réalisé par :

ALI SALMI.

LAMIAE MOUDDENE.

Encadré par:

EL AACHAK LOTFI

Remerciements

Avant de commencer la présentation de ce travail, nous profitons de l'occasion pour remercier ALLAH pour la volonté, la force et la santé qu'il nous a donné afin de réaliser ce travail.

Au terme de ce projet, nous tenons à remercier toute personne ayant contribué, de près ou de loin, à l'aboutissement de ce travail, en particulier :

Nos respectueuses gratitudee à notre cher professeur Monsieur EL AACHAK pour nous avoir donné le goût de l'apprentissage et d'avoir partagé avec nous sa passion pour l'enseignement. Nous avons grandement apprécié son implication et sa qualité d'enseignement dont nous avons bénéficié tout au long du semestre.

C'est grâce à lui que nous avons acquis de nouvelles compétences indispensables pour la réalisation de ce projet.

Merci à nos camarades de classe pour avoir partagé avec nous leurs connaissances.

Table des matières :

Remerciements	2
Introduction.....	5
Conception UML.....	6
Diagramme de cas d'utilisation :	6
Diagramme de classe :	7
Mapping :BD en PhpMySQL :	7
Back-end Laravel	8
Authentification Json Web Token « JWT » :	8
Migration de base de données:.....	9
Les tableaux de base de données :	10
Table d'utilisateurs :	10
Table des étudiants :	10
Table des professeurs :	11
Table des administrateurs :	11
Table des semestres :	11
Table des modules :	12
Table des notes :	12
Table des emplois du temps :	12
Table des PFE :	13
Les contrôleurs :	13
Les modèles :	15
REST API :	16
Front-end Vue Js.....	17
Axios :	17
Les composants(components) :	18
Les vues :	18
Page d'accueil :	19
Espace Admin :	19
Espace Prof :	22
Espace Etudiant :	23

Introduction

L'objectif de ce travail est de réaliser une application web orientée service pour garantir une bonne gestion du cycle d'ingénieur LSI de la FSTT.

Notre application sera composée de deux parties, une partie Front end «VueJS», et la partie Back end « Laravel ». pour accéder à l'application on a utilisé l'authentification avec « JWT ». L'admin de l'application gère les utilisateurs « étudiants et Professeurs », les Modules ,les PFE,les notes (Etudiants) et les Emplois du temps.

Notre application est développée à l'aide de plusieurs langages et Framework :



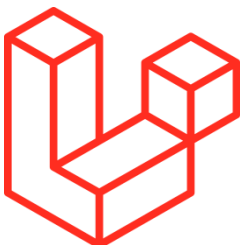
HTML 5 : est un langage informatique qui permet à un créateur de sites Web de gérer la manière dont le contenu de ses pages Web va s'afficher sur un écran, via le navigateur .



CSS 3 :un langage qui permet de gérer la présentation d'une page Web, se présente comme une alternative à la mise en forme via des balises HTML



JavaScript : un langage de développement informatique, et plus précisément un **langage de script orienté objet**. On le retrouve principalement dans les pages Internet. Il permet, entre autres, d'introduire sur une page web ou HTML des petites animations ou des effets.



Laravel est un Framework web open-source écrit en PHP respectant le principe modèle-vue-contrôleur et entièrement développé en programmation orientée objet



Vue.js est un framework JavaScript open-source utilisé pour construire des interfaces utilisateur et des applications web monopages.



JSON Web Token (JWT) est un standard ouvert défini dans la RFC 7519. Il permet l'échange sécurisé de jetons (tokens) entre plusieurs parties. Cette sécurité de l'échange se traduit par la vérification de l'intégrité et de l'authenticité des données.

Conception UML

Diagramme de cas d'utilisation :

Le diagramme de cas d'utilisation montre les différentes tâches des 3 utilisateurs de l'application « admin » « professeur » et « étudiant »

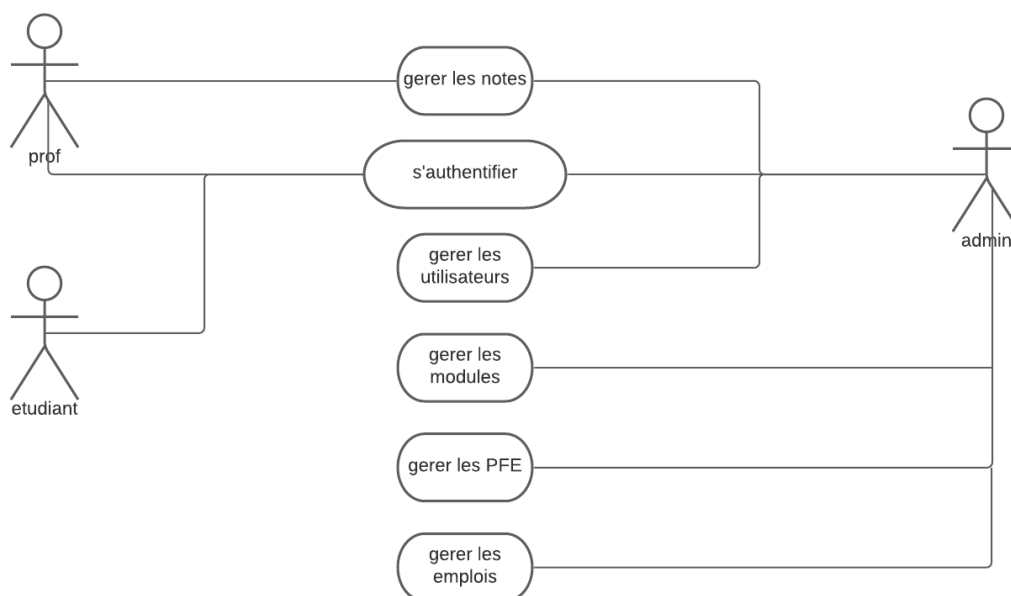
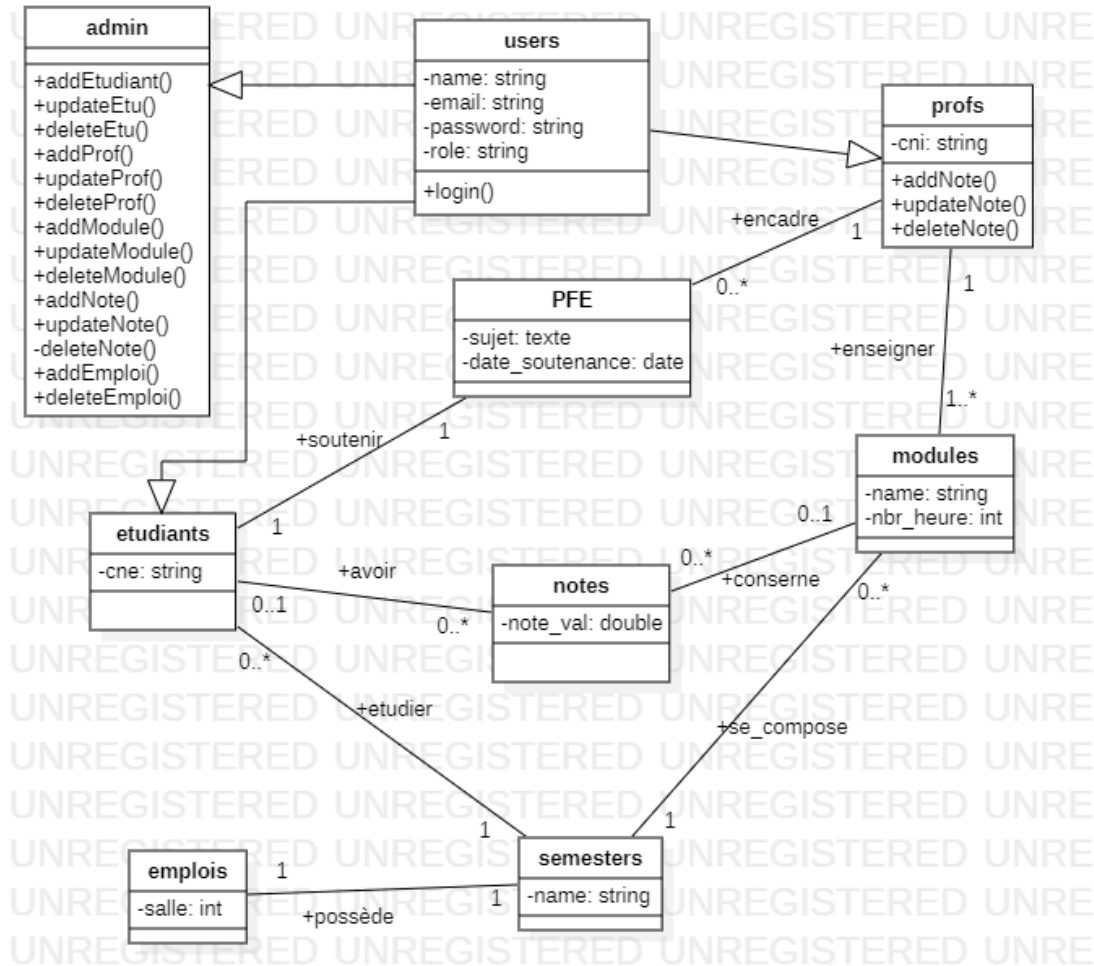


Diagramme de classe :

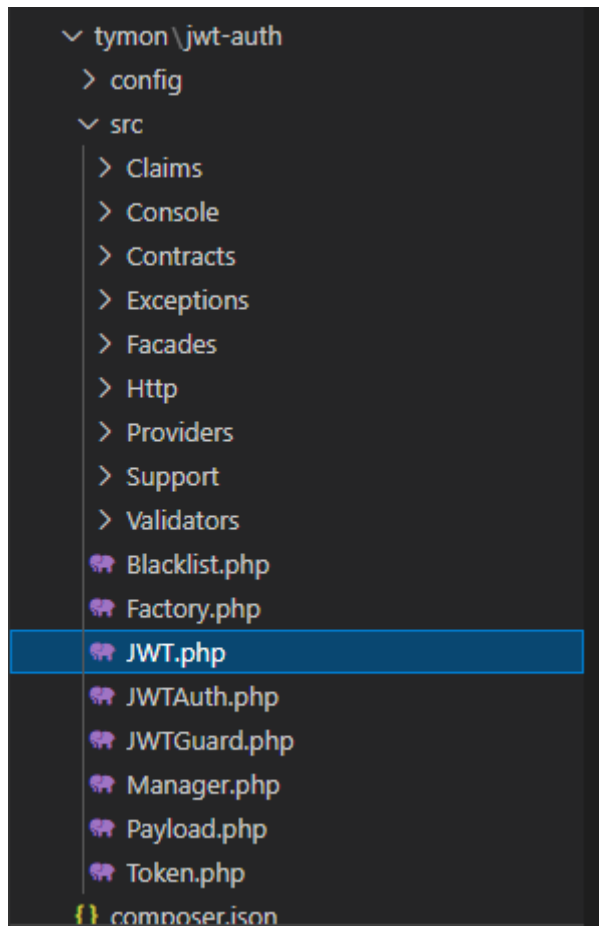
Diagramme de classe possède les différentes classes utilisées, on note que les classes « admin » « professeur » et « étudiant » héritent d'une classe qui s'appelle « user », où on a réalisé l'authentification avec JWT



Mapping :BD en PhpMySql :

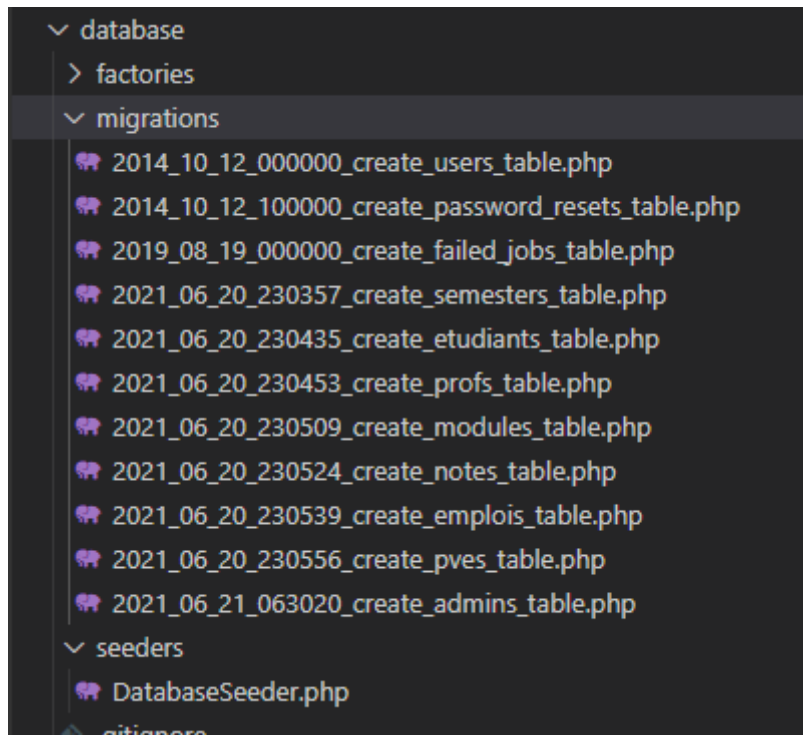
Table	Action	Lignes	Type	Interclassement	Taille	Perte
<input type="checkbox"/> admins	★ Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_unicode_ci	32,0 kio	-
<input type="checkbox"/> emplois	★ Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_unicode_ci	32,0 kio	-
<input type="checkbox"/> etudiants	★ Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_unicode_ci	48,0 kio	-
<input type="checkbox"/> failed_jobs	★ Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_unicode_ci	32,0 kio	-
<input type="checkbox"/> migrations	★ Parcourir Structure Rechercher Insérer Vider Supprimer	11	InnoDB	utf8mb4_unicode_ci	16,0 kio	-
<input type="checkbox"/> modules	★ Parcourir Structure Rechercher Insérer Vider Supprimer	3	InnoDB	utf8mb4_unicode_ci	48,0 kio	-
<input type="checkbox"/> notes	★ Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_unicode_ci	48,0 kio	-
<input type="checkbox"/> password_resets	★ Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_unicode_ci	32,0 kio	-
<input type="checkbox"/> profs	★ Parcourir Structure Rechercher Insérer Vider Supprimer	3	InnoDB	utf8mb4_unicode_ci	32,0 kio	-
<input type="checkbox"/> pves	★ Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_unicode_ci	48,0 kio	-
<input type="checkbox"/> semesters	★ Parcourir Structure Rechercher Insérer Vider Supprimer	6	InnoDB	utf8mb4_unicode_ci	32,0 kio	-
<input type="checkbox"/> users	★ Parcourir Structure Rechercher Insérer Vider Supprimer	3	InnoDB	utf8mb4_unicode_ci	16,0 kio	-
12 tables	Somme	26	InnoDB	utf8mb4_general_ci	416,0 kio	0 0

- Autorisation : il s'agit du scénario le plus courant pour l'utilisation de JWT. Une fois que l'utilisateur est connecté, chaque demande suivante inclura le JWT, permettant à l'utilisateur d'accéder aux routes, services et ressources autorisés avec ce jeton. L'authentification unique est une fonctionnalité qui utilise largement JWT de nos jours, en raison de sa faible surcharge et de sa capacité à être facilement utilisée dans différents domaines.
- Echange d'informations : les jetons Web JSON sont un bon moyen de transmettre des informations en toute sécurité entre les parties. Parce que les JWT peuvent être signés, par exemple, en utilisant



Migration de base de données:

Les migrations sont comme le contrôle de version de votre base de données, permettant de modifier et de partager le schéma de base de données de l'application. Les migrations sont associées au générateur de schéma de Laravel pour créer le schéma de base de données de notre application. La façade Laravel **Schema** fournit un support indépendant de la base de données pour la création et la manipulation de tables sur tous les systèmes de base de données pris en charge par Laravel.



Les tableaux de base de données :

Table d'utilisateurs :

```
*/  
public function up()  
{  
    Schema::create('users', function (Blueprint $table) {  
        $table->id();  
        $table->string('name');  
        $table->string('role');  
        $table->string('email');  
        $table->string('password');  
        $table->rememberToken();  
        $table->timestamps();  
    });  
}
```

Table des étudiants :

```
public function up()  
{  
    Schema::create('etudiants', function (Blueprint $table) {  
        $table->id();  
        $table->string('cne');  
        $table->unsignedBigInteger('user_id');  
        $table->unsignedBigInteger('semester_id');  
        $table->foreign('user_id')->references('id')->on('users')->onUpdate('cascade')->onDelete('cascade');  
        $table->foreign('semester_id')->references('id')->on('semesters')->onUpdate('cascade')->onDelete('cascade');  
        $table->timestamps();  
    });  
}
```

Table des professeurs :

```
public function up()
{
    Schema::create('profs', function (Blueprint $table) {
        $table->id();
        $table->string('cni');

        $table->unsignedBigInteger('user_id');
        $table->foreign('user_id')->references('id')->on('users')->onUpdate('cascade')->onDelete('cascade');
        $table->timestamps();
    });
}
```

Table des administrateurs :

```
public function up()
{
    Schema::create('admins', function (Blueprint $table) {
        $table->id();

        $table->unsignedBigInteger('user_id');

        $table->foreign('user_id')->references('id')->on('users')->onUpdate('cascade')->onDelete('cascade');

        $table->timestamps();
    });
}
```

Table des semestres :

```
public function up()
{
    Schema::create('semesters', function (Blueprint $table) {
        $table->id();
        $table->string('name')->unique();
        $table->timestamps();
    });
}
```

Table des modules :

```
public function up()
{
    Schema::create('modules', function (Blueprint $table) {
        $table->id();
        $table->string('name');
        $table->integer('nbr_heure');

        $table->unsignedBigInteger('prof_id');
        $table->unsignedBigInteger('semester_id');
        $table->foreign('prof_id')->references('id')->on('profs')->onUpdate('cascade')->onDelete('cascade');
        $table->foreign('semester_id')->references('id')->on('semesters')->onUpdate('cascade')->onDelete('cascade');

        $table->timestamps();
    });
}
```

Table des notes :

```
public function up()
{
    Schema::create('notes', function (Blueprint $table) {
        $table->id();
        $table->double('note_val',4,2);
        $table->unsignedBigInteger('etu_id');
        $table->unsignedBigInteger('module_id');
        $table->foreign('etu_id')->references('id')->on('etudiants')->onUpdate('cascade')->onDelete('cascade');
        $table->foreign('module_id')->references('id')->on('modules')->onUpdate('cascade')->onDelete('cascade');

        $table->timestamps();
    });
}
```

Table des emplois du temps :

```
public function up()
{
    Schema::create('emplois', function (Blueprint $table) {
        $table->id();
        $table->integer('salle');
        $table->unsignedBigInteger('semester_id');
        $table->foreign('semester_id')->references('id')->on('semesters')->onUpdate('cascade')->onDelete('cascade');

        $table->timestamps();
    });
}
```

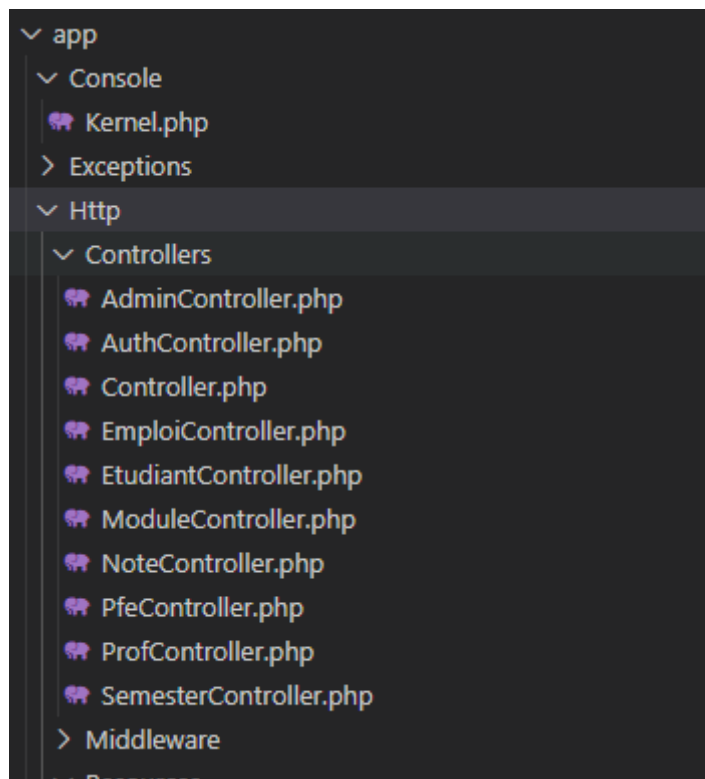
Table des PFE :

```
public function up()
{
    Schema::create('pves', function (Blueprint $table) {
        $table->id();
        $table->string('sujet');
        $table->unsignedBigInteger('etu_id');
        $table->unsignedBigInteger('prof_id');
        $table->foreign('etu_id')->references('id')->on('etudiants') -
        >onUpdate('cascade')->onDelete('cascade');
        $table->foreign('prof_id')->references('id')->on('profs') -
        >onUpdate('cascade')->onDelete('cascade');

        $table->timestamps();
    });
}
```

Les contrôleurs :

Pour bien organiser son code dans une application Laravel il faut bien répartir les tâches. Les routes sont juste un système d'aiguillage pour trier les requêtes qui arrivent , La tâche d'un contrôleur est de réceptionner une requête (qui a déjà été triée par une route) et de définir la réponse appropriée, rien de moins et rien de plus.



Voici un exemple d'un contrôleur : AuthController.php

```
public function register(Request $request) {
```

```

        $validator = Validator::make($request-
>only('name','role','email','password'), [
            'name' => 'required|string|between:2,100',
            'role' => 'required|string',
            'email' => 'required|string|email|max:100|unique:users',
            'password' => 'required|string|min:6',
        ]);

        if($validator->fails()){
            return response()->json($validator->errors()->toJson(), 400);
        }

        $user = User::create(array_merge(
            $validator->validated(),
            ['password' => bcrypt($request->password)]
        ));

        return response()->json([
            'message' => 'User successfully registered',
            'user' => $user
        ], 201);
    }

    protected function createNewToken($token){
        return response()->json([
            'access_token' => $token,
            'token_type' => 'bearer',
            'expires_in' => auth()->factory()->getTTL() * 60,
            'user' => auth()->user()
        ]);
    }
}

```

AdminController.php :

```

        public function addModule(Request $request){
            Module::create([
                'name'=>$request['name'],
                'nbr_heure'=>$request['nbr_heure'],
                'semester_id'=>$request['semester_id'],
                'prof_id'=>$request['prof_id'],
            ]);
        }

        public function updateModule($id,Request $request){

```

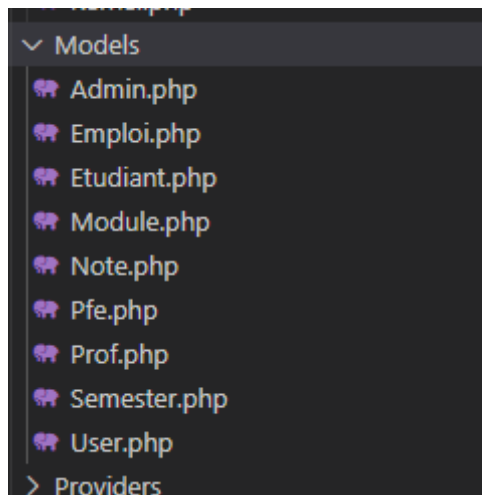
```

        $mdl=Module::findOrFail($id);
        $mdl->name=$request->name;
        $mdl->nbr_heure=$request->nbr_heure;
        $mdl->prof_id=$request->prof_id;
        $mdl->semester_id=$request->semester_id;
        $mdl->save();
    }

```

Les modèles :

Laravel propose de gérer nos données par l'intermédiaire d'un outil appelé **Eloquent** qui est un **ORM** (Object Relational Mapping) qui permet de simplifier les interactions avec la base de données en créant des extensions à la **classe Model** qui portent le **nom de la table** sur laquelle nous souhaitons interagir.



Voici un exemple de model Etudiant.php :

```

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Etudiant extends Model
{
    use HasFactory;
    protected $fillable = [
        'cne',
        'semester_id',
        'user_id',
    ];
}

```

```
];  
}
```

REST API :

REST est un ensemble de contraintes architecturales. Il ne s'agit ni d'un protocole, ni d'une norme. Les développeurs d'API peuvent mettre en œuvre REST de nombreuses manières.

Lorsqu'un client émet une requête par le biais d'une API RESTful, celle-ci transfère une représentation de l'état de la ressource au demandeur ou point de terminaison. Cette information, ou représentation, est fournie via le protocole HTTP dans l'un des formats suivants : JSON (JavaScript Object Notation), HTML, XML, Python, PHP ou texte brut. Le langage de programmation le plus communément utilisé est JSON

Voici exemple de nos route dans le fichier api.php :

```
Route::post('/login', [AuthController::class, 'login']);  
Route::post('/logout', [AuthController::class, 'logout']);  
Route::post('/refresh', [AuthController::class, 'refresh']);  
Route::get('/user-profile', [AuthController::class, 'userProfile']);  
Route::post('/registerUser', [AdminController::class, 'register2']);  
  
Route::get('user', [AuthController::class, 'index']);  
Route::get('user/{id}', [AuthController::class, 'show']);  
  
//*****etudiant***** */  
Route::put('updateEtu/{id}', [AdminController::class, 'updateEtu']);  
Route::post('deleteEtu/{id}', [AdminController::class, 'deleteEtu']);  
  
//*****prof*****  
** */  
Route::put('updateProf/{id}', [AdminController::class, 'updateProf']);  
Route::post('deleteProf/{id}', [AdminController::class, 'deleteProf']);  
  
//*****module***** */  
Route::post('addModule', [AdminController::class, 'addModule']);  
Route::put('updateModule/{id}', [AdminController::class, 'updateModule']);  
;  
Route::post('deleteModule/{id}', [AdminController::class, 'deleteModule']  
);
```


Front-end Vue Js

VueJS est un framework JavaScript progressif et un framework Front End utilisé pour développer des interfaces Web interactives. L'accent est davantage mis sur la partie vue, qui est l'extrémité avant. Il est très facile à intégrer avec d'autres projets et bibliothèques. D'un autre côté, Vue est également parfaitement capable d'alimenter des applications sophistiquées à page unique lorsqu'il est utilisé en combinaison avec des outils modernes et des bibliothèques de support.

Axios :

Axios est une bibliothèque JavaScript fonctionnant comme un client HTTP. Elle permet de communiquer avec des API en utilisant des requêtes, elle est basée sur Promise, ce qui vous permet de profiter des avantages d'async de JavaScript et await pour un code asynchrone plus lisible.

Vous pouvez également intercepter et annuler des demandes, et il existe une protection intégrée côté client contre la falsification des demandes intersites.

Exemple :

```
created() {  
    axios  
      .get('http://127.0.0.1:8000/api/auth/showEtu')  
      .then(response => {  
        this.etus = response.data;  
      });  
}
```

Cette méthode nous a envoyé la liste des étudiants ;

```
deleteE: function(i){  
    axios.post('http://127.0.0.1:8000/api/auth/deleteEtu/'+i).then  
(alert('Etudiant is deleted'));  
    window.location.reload(false);  
},
```

Cette méthode nous permet de supprimer un étudiant ;

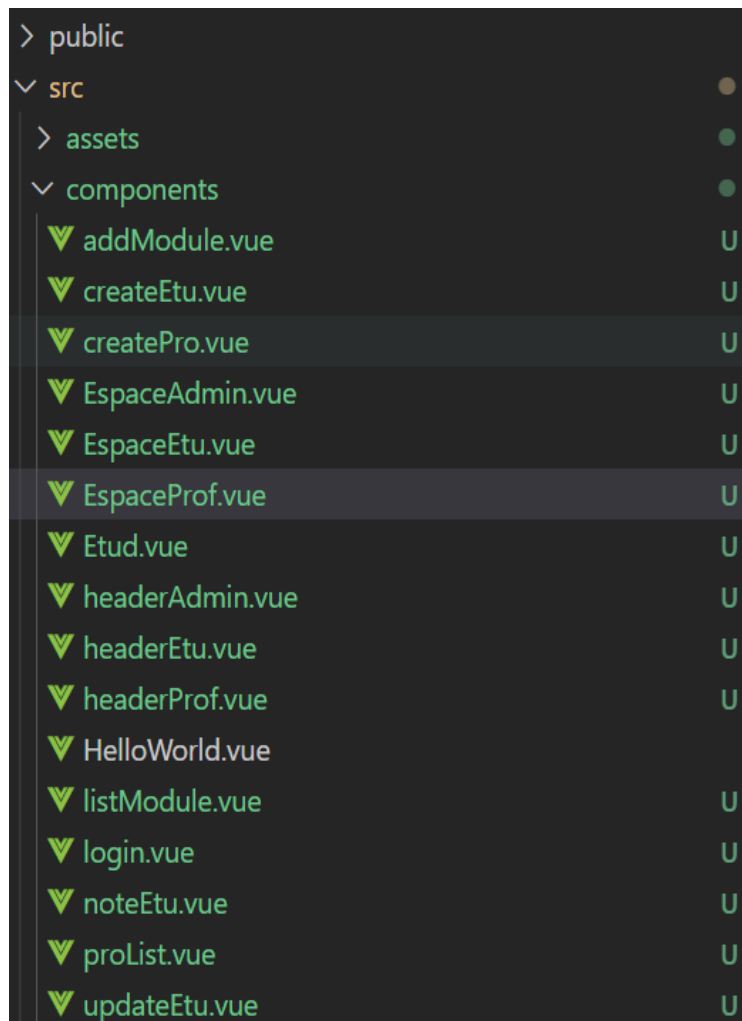
```
submit(){  
    axios.post('http://127.0.0.1:8000/api/auth/addUser', this.user)  
      .then(  
        this.message1='User added successfully'  
      )  
  
      .finally(()=>this.loading=true);
```

```
}
```

Cette méthode nous permet de ajouter un étudiant ;

Les composants(components) :

Les composants sont l'une des fonctionnalités les plus puissantes de Vue.js. Ils vous aident à étendre les éléments HTML de base pour encapsuler du code réutilisable. À un niveau élevé, les composants sont des éléments personnalisés auxquels le compilateur de Vue.js attacherait un comportement spécifié. Dans certains cas, ils peuvent également apparaître comme un élément HTML natif étendu avec le « is » attribut spécial



Les vues :

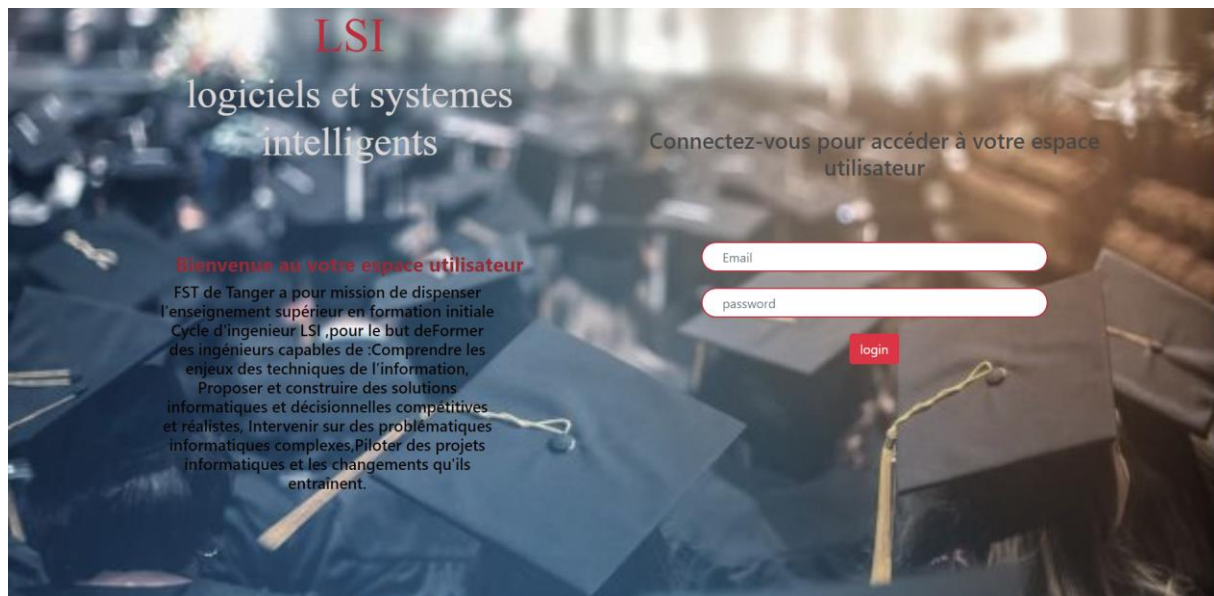
Exemple :

▼ addModule.vue	U
▼ createEtu.vue	U
▼ createPro.vue	U
▼ espaceAdmin.vue	U
▼ espaceEtu.vue	U
▼ EspaceProf.vue	U
▼ etud.vue	U
▼ Home.vue	M
▼ listModule.vue	U
▼ login.vue	U
▼ proList.vue	U
▼ updateEtu.vue	U

Page d'accueil :

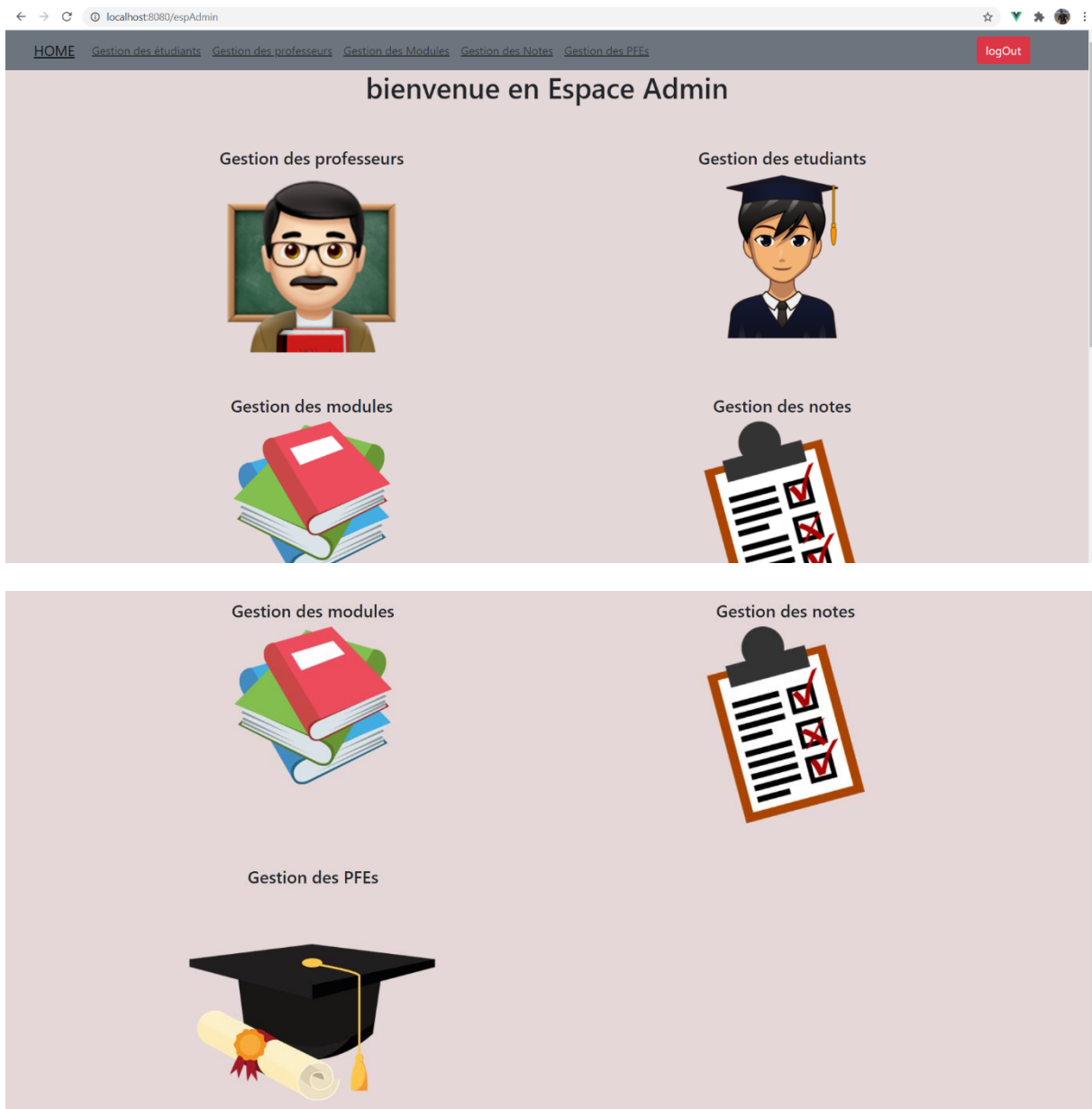
La page d'accueil représente un espace de login ,chaque utilisateur se connecte avec son email et mot de passe

Après réussir la connexion ,un espace personnel de l'utilisateur s'affiche, et dépend de son rôle (prof ou admin ou étudiant)



Espace Admin :

L'admin de l'application ses taches est de gérer (CRUD) les professeurs, les étudiants , les modules et les notes ainsi que les PFE .



Exemple :Gestion des étudiants :

La gestion des étudiants, permet de ajouter un étudiant , modifier ses données ou le supprimer :

Lors de l'ajout d'un étudiant, l'admin remplit un formulaire par le nom , email ,mot de passe, semestre et CNE, si tous les champs sont valides un message de succès s'affiche

[HOME](#)
[Gestion des étudiants](#)
[Gestion des professeurs](#)
[Gestion des Modules](#)
[Gestion des Notes](#)
[Gestion des PFEs](#)
[logOut](#)

User added successfully

Nom:

fatima zahiz

Email:

fatima@gmail.com

Mot de passe:

Semestre:

1

CNE:

R132355345

ajouter étudiant

Etudiants List

Alors en tout moment on peut afficher la liste des étudiants inscrits ,avec la possibilité de modifier et supprimer un étudiant choisit

[HOME](#)
[Gestion des étudiants](#)
[Gestion des professeurs](#)
[Gestion des Modules](#)
[Gestion des Notes](#)
[Gestion des PFEs](#)
[logOut](#)

ajouter un étudiant

Name	Email	CNE	Action	
lamiae	lamiae@gmail.com	R3656565265	Update	delete
ali	www@gmail.com	d535415241	Update	delete
samir	samir@gmail.com	K165315615	Update	delete
fatima zahiz	fatima@gmail.com	R132355345	Update	delete
ayoub	ayoub@gmail.com	M684684565	Update	delete
saad	saad@gmail.com	P685416531	Update	delete
asmae	asmae@gmail.com	I6854316541	Update	delete

Exemple : Gestion des professeurs :

[HOME](#)
[Gestion des étudiants](#)
[Gestion des professeurs](#)
[Gestion des Modules](#)
[Gestion des Notes](#)
[Gestion des PFEs](#)
[logOut](#)

ajouter un Professeur

Name	Email	CNE	Action	
ghadi	ghadi@gmail.com	d535415241	Update	delete
eIAchaq	eIAchaq@gmail.com	lkk58456	Update	delete
elbrak	elbrak@gmail.com	dd55421	Update	delete
fennan	fennan@gmail.com	f685362	Update	delete

Espace Prof :

[HOME](#) [Modules](#) [Gestion des notes](#) [PFF](#) [logOut](#)

bienvenue en Espace Professeur

Mes informations



Nom: Fennan Cni: LA12345

Email: Fennan@gmail.com

Mes modules



Module	Semester	nombre d'heure
Système d'exploitation	S1	24h
Programmation Mobile	S3	34h
POO en JAVA	S2	28h


[Voir Plus](#)

Email: Fennan@gmail.com

POO en JAVA S2 28h


[Voir Plus](#)

Notes des Etudiants



Module	Action
Système d'exploitation	Gérer notes
Programmation Mobile	Gérer notes
POO en JAVA	Gérer notes

Projet Fin d'étude




Etudiant	Sujet	Date de soutenance
mohammed adam	deep learning	10/07/2021

Espace Etudiant :

[HOME](#) [MODULES](#) [NOTES](#) [PFEs](#) [logOut](#)

bienvenue en Espace Etudiant



Vos informations




Nom: samir Cne: K165315615

Email: samir@gmail.com Semester: S2

Emploi du temps




votre relevé des notes



Projet Fin d'étude


Email: saad@gmail.com Semester: S6

votre relevé des notes



[Première Année](#)[Deuxième Année](#)[Troisième Année](#)

Projet Fin d'étude



Sujet: App Web
Encadrant: El achag
Date de soutenance: 24-06-2021

Exemple : consulter les notes d'un étudiant :

HOME MODULES NOTES PFE		logOut
Nom Module		Note
POO en JAVA		13.5
Dev Web&Framworks		10
Théories des graphes		11.5
UML		12
Statistique et probabilité		17
Mangement & comptabilité		13

Conclusion :

Grace à ce projet que nous avons eu l'opportunité de cumuler les connaissances théoriques avec celles de la pratique, ceci permet également de rentrer dans la vie active et découvrir plus précisément le milieu professionnel. Ce projet consiste à découvrir les outils de réalisation d'une Application Web, à partir de notre vision et point de vue pour objectif de réaliser des différentes parties (backend, frontend), à l'aide de plusieurs outils comme **laravel 8** et **vueJs**. L'élaboration de ce rapport a pour principale source nos connaissances acquises tout le long de notre formation scolaire et nos recherches personnelles.