

NOTE DE STAGE 2021

CHEZ COFLINE

SOMMAIRE

Mardi 25 Mai 2021	2
Missions :	2
Réalisations :	2
Mercredi 26 Mai 2021	6
Missions :	6
Realisations :	6
Jeudi 27 Mai 2021	8
Missions :	8
Réalisations :	8
Vendredi 28 Mai 2021	10
Missions :	10
Réalisations :	10

***Par soucis de confidentialité ce rapport ne contiendra pas de données sensibles.**

***Pré-stage notre maitre de stage nous a conseillé de regarder les vidéos de Nord Coders (YouTube) sur Laravel 8 afin d'avoir une première vision sur comment fonctionne ce Framework.**

Commented [VS(21)]: Signature accord de confidentialité

Commented [VS(22)]: Note : Un client = Etat civil/Info contact/Adresse identifie le client
Peut être lié à : Dossier/Facturation/Tache/Document

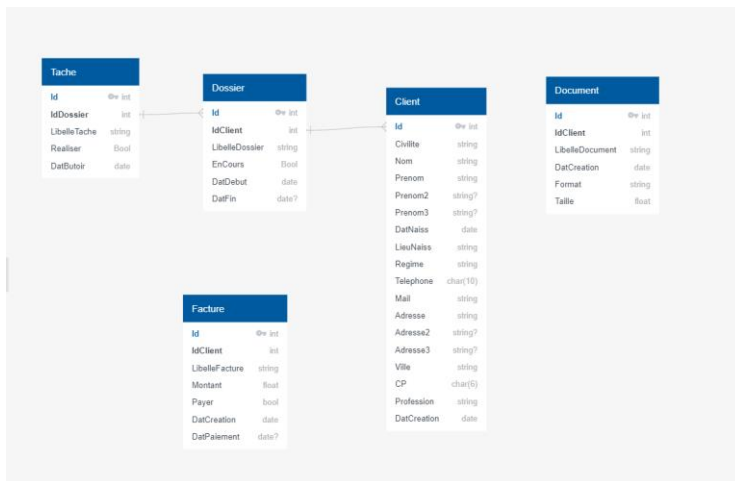
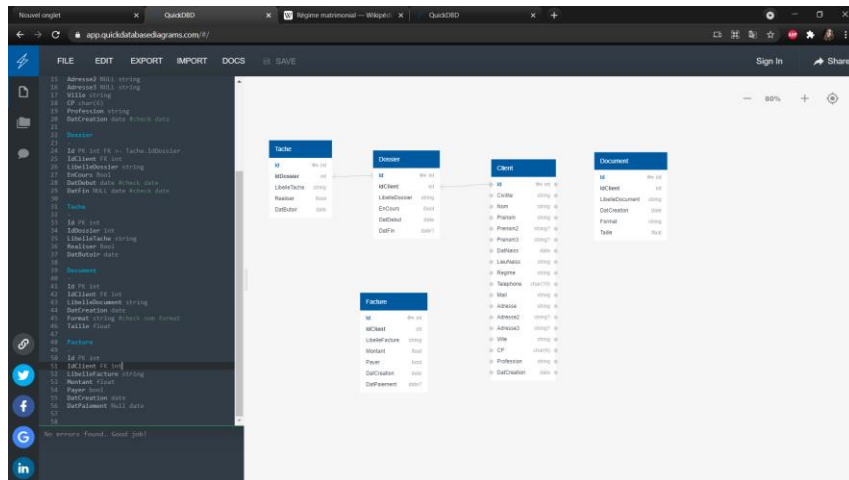
Commented [VS(23)]: Note :

- Evocation de React (Librairie JS) et de responsive.
- Faire autant de découpe qu'on pense qu'il y a de module et les mettre dans une archive zip.

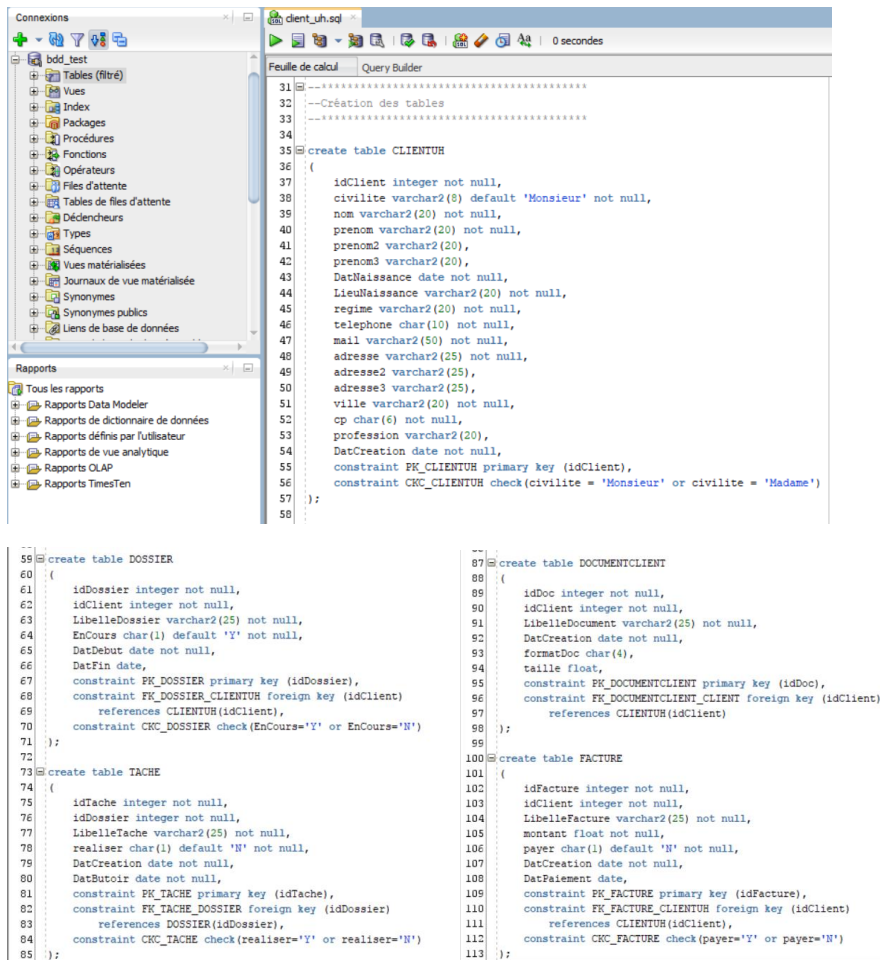
-

2 | Page

SCHALCKENS Valentine – BTS SIO – SLAM



b. Transposition de ce schéma en PL/SQL sur une base Oracle en local



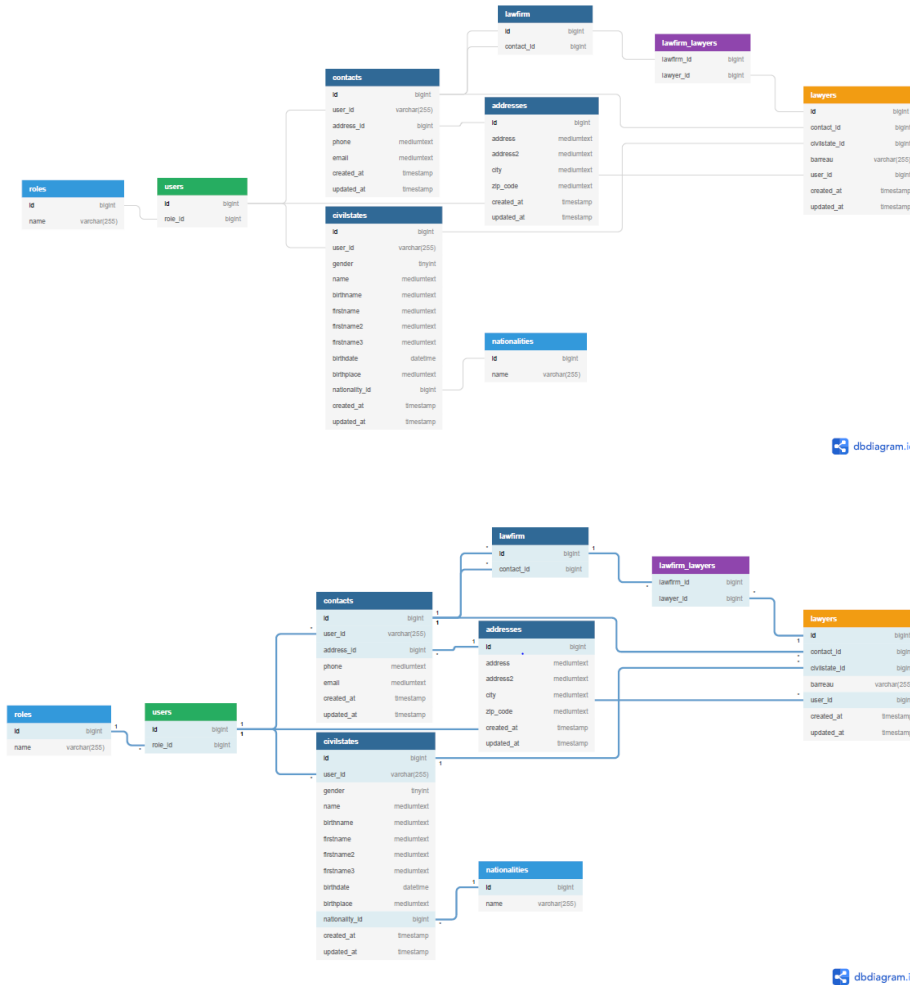
```

31 --Création des tables
32 -----
33
34
35 create table CLIENTUK
36 (
37     idClient integer not null,
38     civilite varchar2(8) default 'Monsieur' not null,
39     nom varchar2(20) not null,
40     prenom varchar2(20) not null,
41     prenom2 varchar2(20),
42     prenom3 varchar2(20),
43     DatNaissance date not null,
44     LieuNaissance varchar2(20) not null,
45     regime varchar2(20) not null,
46     telephone char(10) not null,
47     mail varchar2(50) not null,
48     adresse varchar2(25) not null,
49     adresse2 varchar2(25),
50     adresse3 varchar2(25),
51     ville varchar2(20) not null,
52     cp char(6) not null,
53     profession varchar2(20),
54     DatCreation date not null,
55     constraint PK_CLIENTUK primary key (idClient),
56     constraint CHK_CLIENTUK check(civilite = 'Monsieur' or civilite = 'Madame')
57 );
58
59 create table DOSSIER
60 (
61     idDossier integer not null,
62     idClient integer not null,
63     LibelleDossier varchar2(25) not null,
64     EnCours char(1) default 'Y' not null,
65     DatDebut date not null,
66     DatFin date,
67     constraint PK_DOSSIER primary key (idDossier),
68     constraint FK_DOSSIER_CLIENTUK foreign key (idClient)
69         references CLIENTUK(idClient),
70     constraint CHK_DOSSIER check(EnCours='Y' or EnCours='N')
71 );
72
73 create table TACHE
74 (
75     idTache integer not null,
76     idDossier integer not null,
77     LibelleTache varchar2(25) not null,
78     realiser char(1) default 'N' not null,
79     DatCreation date not null,
80     DatButoir date not null,
81     constraint PK_TACHE primary key (idTache),
82     constraint FK_TACHE_DOSSIER foreign key (idDossier)
83         references DOSSIER(idDossier),
84     constraint CHK_TACHE check(realiser='Y' or realiser='N')
85 );
86
87 create table DOCUMENTCLIENT
88 (
89     idDoc integer not null,
90     idClient integer not null,
91     LibelleDocument varchar2(25) not null,
92     DatCreation date not null,
93     formatDoc char(4),
94     taille float,
95     constraint PK_DOCUMENTCLIENT primary key (idDoc),
96     constraint FK_DOCUMENTCLIENT_CLIENT foreign key (idClient)
97         references CLIENTUK(idClient)
98 );
99
100 create table FACTURE
101 (
102     idFacture integer not null,
103     idClient integer not null,
104     LibelleFacture varchar2(25) not null,
105     montant float not null,
106     payer char(1) default 'N' not null,
107     DatCreation date not null,
108     DatPaiement date,
109     constraint PK_FACTURE primary key (idFacture),
110     constraint FK_FACTURE_CLIENTUK foreign key (idClient)
111         references CLIENTUK(idClient),
112     constraint CHK_FACTURE check(payer='Y' or payer='N')
113 );

```

SCHALCKENS Valentine – BTS SIO – SLAM

c. Correction avec le maitre de stage et schéma de donnée traité pour le moment



MERCREDI 26 MAI 2021

MISSIONS :

1. Création d'un projet Laravel 8 pour l'application.
2. Implémentation des Migrations.
3. Implémentation des Models et mapping relationnel avec Eloquent.

REALISATIONS :

1.

Création du projet à l'aide de Laragon et des vidéos de Nord Coders (YouTube). Installation de la nouvelle version de composer. Installation de Visual Studio Code (VSC).

2.

Utilisation de **php artisan** sur VSC pour générer automatiquement les migrations et leur modèle² :

php artisan make:model User -m

Ajout des données nécessaires dans les migrations ainsi que leur bon type en se basant sur le modèle relationnel fourni et la documentation Laravel.

Puis migration des tables dans la base de données à l'aide d'une commande artisan³ :

php artisan migrate(:fresh)

Commented [VS(24)]: En cas d'erreur ou d'oubli lors de la première migration

3.

Ajout des relations nécessaire dans les models précédemment créer avec les principes de OneToOne, OneToMany, et ManyToMany.

² <https://laravel.com/docs/8.x/eloquent#generating-model-classes>

³ <https://laravel.com/docs/8.x/migrations#drop-all-tables-migrate>

SCHALCKENS Valentine – BTS SIO – SLAM

```

1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  class CreateContactsTable extends Migration
8  {
9      /**
10       * Run the migrations.
11       *
12       * @return void
13       */
14     public function up()
15     {
16         Schema::create('contacts', function (Blueprint $table) {
17             $table->id();
18             $table->foreignId('user_id')->constrained();
19             $table->foreignId('address_id')->constrained();
20             $table->mediumText('phone');
21             $table->mediumText('email');
22             $table->timestamps();
23         });
24     }
25
26     /**
27      * Reverse the migrations.
28      *
29      * @return void
30      */
31     public function down()
32     {
33         Schema::dropIfExists('contacts');
34     }
35 }
36

```

```

1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Model;
6
7  class Contact extends Model
8  {
9      protected $guarded = [];
10
11     public function users()
12     {
13         return $this->belongsTo(User::class);
14     }
15     public function addresses()
16     {
17         return $this->belongsTo(Address::class);
18     }
19     public function lawfirms()
20     {
21         return $this->hasOne(Lawfirm::class);
22     }
23     public function lawyers()
24     {
25         return $this->hasOne(Lawyer::class);
26     }
27 }
28

```

JEUDI 27 MAI 2021

MISSIONS :

1. Implémenter les controllers, nous devons pouvoir créer, afficher, modifier, et supprimer des lignes des tables de la BDD.
2. Implémenter les routes détaillées (ne pas utilisé ressource) et réfléchir à post ou get.

REALISATIONS :

1.

Mise en place du principe CRUD, avec les fonctions create, store, show, edit, update, destroy. Aide de la documentation et des vidéos Nord Coders pour implémenter le code des fonctions.

2.

Début de l'implémentation des routes à l'aide de la documentation et des vidéos Nord Coders pour implémenter le code des fonctions.

```
83 Route::get('contacts',[ContactController::class, 'contacts'])->name('contacts');
84 Route::get('contacts/create',[ContactController::class, 'create'])->name('contacts.create');
85 Route::post('contacts/store',[ContactController::class, 'store'])->name('contacts.store');
86 Route::get('contacts/edit/{id}',[ContactController::class, 'edit'])->name('contacts.edit');
87 Route::post('contacts/update/{id}',[ContactController::class, 'update'])->name('contacts.update');
88 Route::post('contacts/delete/{id}',[ContactController::class, 'delete'])->name('contacts.delete');
89 Route::get('contacts/{id}',[ContactController::class, 'show'])->name('contacts.show');
```



```

1  <?php
2
3  namespace App\Http\Controllers;
4
5  use App\Models\Contact;
6  use App\Http\Controllers\Controller;
7  use Illuminate\Http\Request;
8
9  class ContactController extends Controller
10 {
11     public function contacts()
12     {
13         return view('contacts');
14     }
15
16     public function create()
17     {
18         return view('contacts.create');
19     }
20
21     public function store(Request $request)
22     {
23         Contact::create([
24             'user_id' => $request->user_id,
25             'address_id' => $request->address_id,
26             'phone' => $request->phone,
27             'email' => $request->email
28         ]);
29     }
30     public function show($id)
31     {
32         $contact = Contact::findOrFail($id);
33         return view('contacts.show', compact('contact'));
34     }
35
36     public function edit($id)
37     {
38         $contact = Contact::findOrFail($id);
39         return view('contacts.edit', compact('contact'));
40     }
41
42     public function update(Request $request, $id)
43     {
44         $contact = Contact::findOrFail($id);
45         $contact->update([
46             'user_id' => $request->user_id,
47             'address_id' => $request->address_id,
48             'phone' => $request->phone,
49             'email' => $request->email
50         ]);
51     }
52
53     public function destroy($id)
54     {
55         $contact = Contact::findOrFail($id);
56         $contact->delete();
57     }
58 }

```

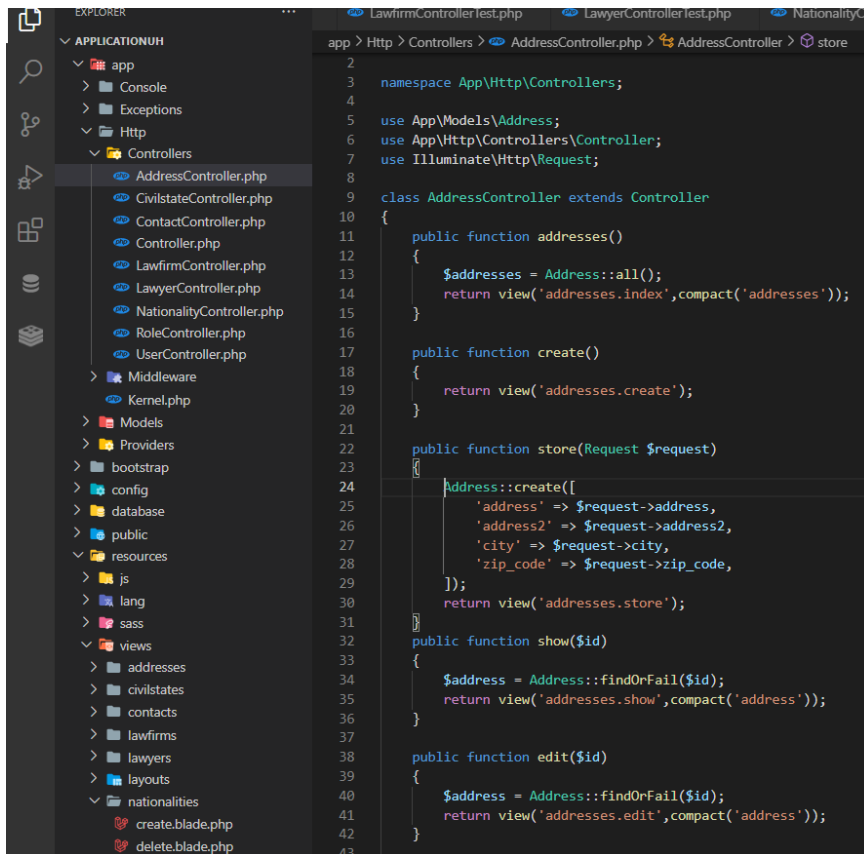
VENDREDI 28 MAI 2021

MISSIONS :

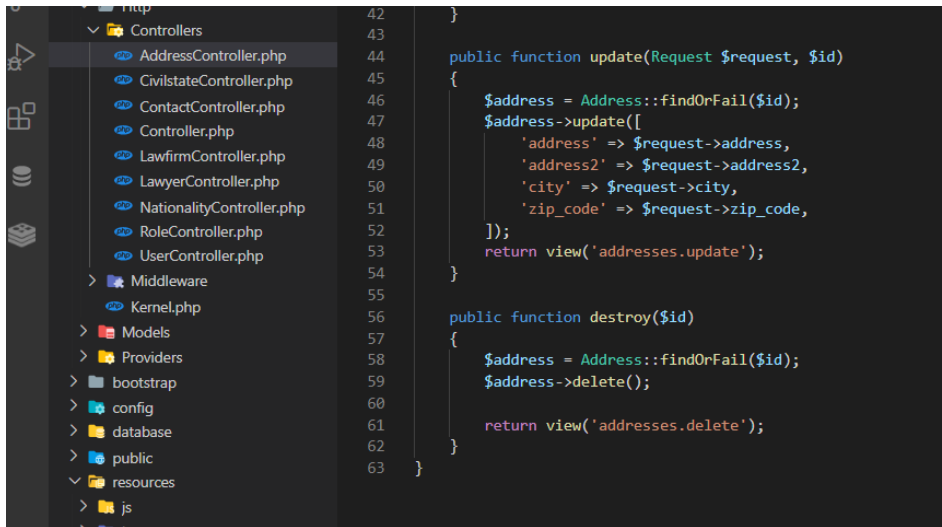
1. Continuer l'implémentations des routes et des controllers.
2. Implémenter les views de façon sommaire pour le moment.

REALISATIONS :

1. Rectification de certaines erreurs découvertes après test avec les views et des données arbitraires dans la base de données.



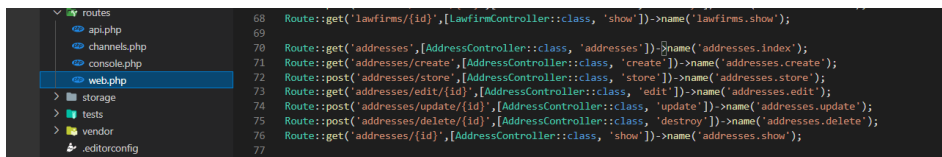
```
1 namespace App\Http\Controllers;
2
3 use App\Models\Address;
4 use App\Http\Controllers\Controller;
5 use Illuminate\Http\Request;
6
7 class AddressController extends Controller
8 {
9     public function addresses()
10     {
11         $addresses = Address::all();
12         return view('addresses.index', compact('addresses'));
13     }
14
15     public function create()
16     {
17         return view('addresses.create');
18     }
19
20     public function store(Request $request)
21     {
22         Address::create([
23             'address' => $request->address,
24             'address2' => $request->address2,
25             'city' => $request->city,
26             'zip_code' => $request->zip_code,
27         ]);
28         return view('addresses.store');
29     }
30
31     public function show($id)
32     {
33         $address = Address::findOrFail($id);
34         return view('addresses.show', compact('address'));
35     }
36
37     public function edit($id)
38     {
39         $address = Address::findOrFail($id);
40         return view('addresses.edit', compact('address'));
41     }
42 }
43
```



```

42 }
43
44 public function update(Request $request, $id)
45 {
46     $address = Address::findOrFail($id);
47     $address->update([
48         'address' => $request->address,
49         'address2' => $request->address2,
50         'city' => $request->city,
51         'zip_code' => $request->zip_code,
52     ]);
53     return view('addresses.update');
54 }
55
56 public function destroy($id)
57 {
58     $address = Address::findOrFail($id);
59     $address->delete();
60
61     return view('addresses.delete');
62 }
63

```

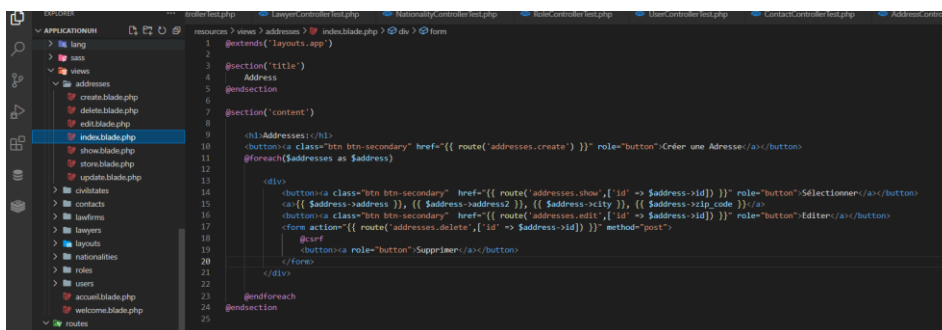


```

68 Route::get('lawfirms/{id}', [LawfirmController::class, 'show'])->name('lawfirms.show');
69
70 Route::get('addresses', [AddressController::class, 'index'])->name('addresses.index');
71 Route::get('addresses/create', [AddressController::class, 'create'])->name('addresses.create');
72 Route::post('addresses/store', [AddressController::class, 'store'])->name('addresses.store');
73 Route::get('addresses/edit/{id}', [AddressController::class, 'edit'])->name('addresses.edit');
74 Route::post('addresses/update/{id}', [AddressController::class, 'update'])->name('addresses.update');
75 Route::post('addresses/delete/{id}', [AddressController::class, 'destroy'])->name('addresses.delete');
76 Route::get('addresses/{id}', [AddressController::class, 'show'])->name('addresses.show');
77

```

2. Implémentation des vues à l'aide des vidéos de Nord Coders et de la documentation Laravel.
 - a. Codes

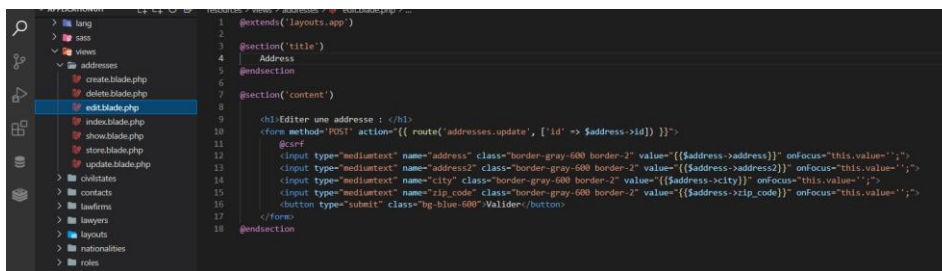
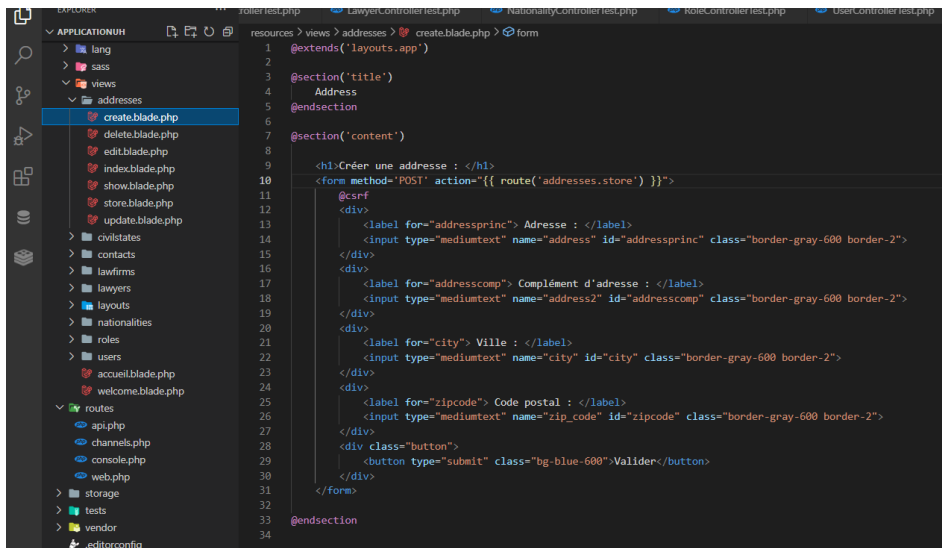


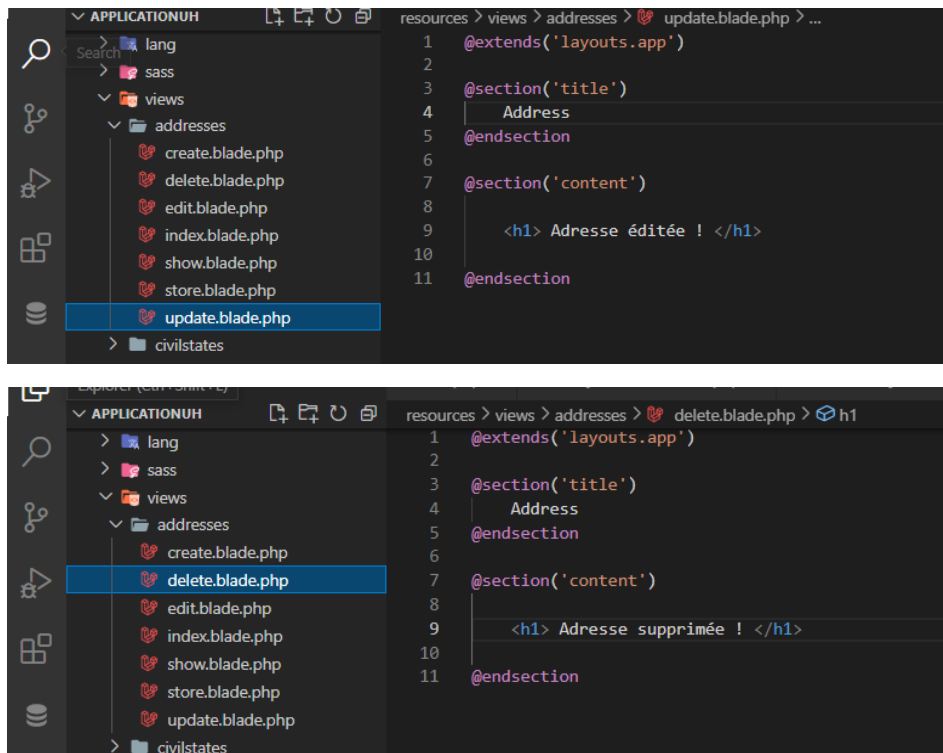
```

1 @extends('layouts.app')
2
3 @section('title')
4     Address
5 @endsection
6
7 @section('content')
8
9     <h1>Addresses</h1>
10     <button class="btn btn-secondary" href="{{ route('addresses.create') }}" role="button">Créer une Adresse</button>
11     @foreach($addresses as $address)
12
13         <div>
14             <button class="btn btn-secondary" href="{{ route('addresses.show', ['id' => $address->id]) }}" role="button">Sélectionner</button>
15             <a href="{{ route('addresses.show', ['id' => $address->id]) }}" role="button">{{ $address->address }}</a>
16             <button class="btn btn-secondary" href="{{ route('addresses.edit', ['id' => $address->id]) }}" role="button">Modifier</button>
17             <form action="{{ route('addresses.delete', ['id' => $address->id]) }}" method="post">
18                 @csrf
19                 <button class="btn btn-secondary" href="{{ route('addresses.delete', ['id' => $address->id]) }}" role="button">Supprimer</button>
20             </form>
21         </div>
22     @endforeach
23 @endsection
24

```

SCHALCKENS Valentine – BTS SIO – SLAM





b. Résultats

- [Accueil](#)
- [Role](#)
- [User](#)
- [Address](#)
- [Contact](#)
- [Nationality](#)
- [Civilstate](#)
- [Lawyer](#)
- [Lawfirm](#)

Addresses:

[Créer une Adresse](#)

Sélectionner	zefqsdf, sdfsd, fffffff, 54628	Editer
Supprimer		
Sélectionner	egrrsv, ervgvqs, qdvqvqc, 849849	Editer
Supprimer		
Sélectionner	ftfjbj, .n ,hjb, buigiug, 56625	Editer
Supprimer		

- [Accueil](#)
- [Role](#)
- [User](#)
- [Address](#)
- [Contact](#)
- [Nationality](#)
- [Civilstate](#)
- [Lawyer](#)
- [Lawfirm](#)

Créer une adresse :

Adresse :

Complément d'adresse :

Ville :

Code postal :

[Valider](#)

- [Accueil](#)
- [Role](#)
- [User](#)
- [Address](#)
- [Contact](#)
- [Nationality](#)
- [Civilstate](#)
- [Lawyer](#)
- [Lawfirm](#)

Editer une adresse :

[Valider](#)