**Contexte** : PPE 2ème année – Framework PHP Laravel

Découverte d’un des plus importants Framework PHP : Laravel

Pré requis – Installation – Notions d’architecture

Projet de découverte avec **une application de demande de carte d’étudiant ENC**

**Objectif du PPE**

L’objectif est créer un 1er projet de découverte du framework pour se familiariser avec les grands principes.

L’objectif est de s’approprier les concepts génériques d’un Framework, tels que :

* Le pattern (ou motif, ou paradigm) MVC, pour modèle – Vue – Contrôleur
* Les principes du routage, des vues, l’accès aux bases de données via un ORM (Object Relational mapping) …

|  |
| --- |
| **Pré requis :**   1. Installation de Laragon (pour domicile ou portable)   Pour créer des projets Laravel, le mieux est soit d’utiliser le logiciel Laragon, qui est de la famille WAMP (c’est-à-dire qui contient un ensemble de modules packagés pour le développement Web, en environnement Windows), soit d’utiliser une machine virtuelle dédiée (avec Homestead / Vagrant). A l’ENC, nous utiliserons Laragon.  Question : quel est le nom équivalent en environnement Linux ?   1. Création d’un projet Laravel   **Etape 1** : paramétrage Proxy  Plus besoin de paramétrage avec la nouvelle connexion Free de l’ENC  **Etape 2** : création d’un projet Laravel   1. Dans Laragon : Start All , puis 2. Toujours dans Laragon   Clic droit dans Laragon puis « Créer un site Web rapidement » 🡪 Laravel : crée un projet dans LARAGON\www où LARAGON est le répertoire d’installation de laragon (usuellement c:\laragon)  (plus exactement il crée le projet à l’emplacement du DOCUMENT\_ROOT pour ceux qui l’auraient modifié).    Vous pouvez nommer ce projet **carteEnc**.  L’initialisation du projet, via le téléchargement des fichiers composant le Framework Laravel, doit prendre quelques minutes maximum.  Ensuite vous avez **2 possibilités** :   * Travailler avec un éditeur « classique », type Notepad++ ou VisualStudioCode. Les commandes Artisan (outil de tooling de Laravel * **Utiliser PHPStorm**   On préférera utiliser PHPStorm (cf document associé) car la maîtrise d’un IDE est importante en milieu professionnel, vous pouvez profiter de l’intégration de PHP Storm avec Laravel, les Helpers associés, les commandes artisan et l’outil de gestion de packages et dépendances Composer.  Par la suite je donne 2 possibilités à chaque fois, avec IDE ou sans IDE.  **Question n°1** :   1. Lorsque vous créer le projet Laravel via Laragon, quelle commande est exécutée (l’équivalent de ce que vous avez fait en utilisant l’interface Laragon si vous aviez voulu le faire « à la main » en ligne de commande) ? 2. Est-ce qu’une base de données a été crée ? Si oui, où ça ? 3. A quelle URL est disponible l’application ?   **Partie n°1** : Création d’une application de demande de carte d’étudiant ENC : partie Données et Modèle  L’objectif est de permettre à un utilisateur de demander via un formulaire une carte d’étudiant pour l’Ecole Nationale de Commerce.  Les données saisies vont être persistées dans une base de données MySQL.  **Etape 3** : création du mécanisme d’authentification  Analyse : tout d’abord, examinez les routes avant de lancer la commande mettant en œuvre l’authentification dans Laravel.  Le fichier définissant les routes web (HTML) se nomme **routes\web.php**  Avec PHPStorm :  Dans la fenêtre Terminal :    Puis ensuite :  Touche CTRL 2 fois puis :    Sans PHPStorm :  Dans la fenêtre Terminal :    Puis ensuite :     * Pour exécuter cette commande, via l’explorateur Windows, positionnez-vous dans le répertoire créé, par exemple : C:\laragon\www\carteENC puis ouvrir une ligne de commande (tapez cmd dans la fenêtre) * Saisissez les instructions dans la fenêtre DOS   Pour examiner les routes :    **Etape 4** : données : base et migration  4.a la base de données nommée carteEnc est normalement automatiquement créée par Laragon, au format UTF8MB4  4.b Modifiez le fichier .env à la racine du projet pour refléter votre configuration    4.c exécutez une commande ***artisan*** qui va créer l’enveloppe de la table et exécuter la migration des données  Avec PHPStorm :  Touche CTRL 2 fois puis :    Sans PHPStorm :  Dans le répertoire du projet :    Cette commande migre les données dans la base de données.  Le répertoire sources des données migrées est dans : **database\migrations**  Remarque : artisan est le nom du langage de commande des Laravel  **Etape 5** : installation partie Front End (CSS essentiellement)  Dans le répertoire du projet :    **1er Checkpoint :** vérification de l’authentification   * Vérifiez que l’URL <http://carteEnc.test/> vous donne accès à la page d’accueil de Laravel * Vérifiez que l’URL <http://carteEnc.test/register> vous permet de crée un compte * Créez un compte * Vérifiez qu’il est présent dans la base de données * Vérifiez que l’URL <http://carteEnc.test/login> est accessible * Vérifiez que vous pouvez vous logger avec le compte que vous avez créé   Atelier : customiser les vues par défaut de JetStream  Etape 1 : on commence par publier les vues JetStream pour pouvoir les modifier    Etape 2 : on va chercher les vues que l’on souhaite adapter dans le répertoire resources\views\vendor\jetstream\components  Exemple : modifiez le contenu du fichier Welcome.blade.php pour changer le texte d’accueil d’un utilisateur une fois loggé  4.d création d’un modèle pour la Carte d’étudiant  Avec PHPStorm :  Touche CTRL 2 fois puis :    Sans PHPStorm :  Entrez la commande suivante : |



La commande artisan make : model permet de créer un modèle, qui va être une fabrique d’objets de la classe mentionnée, ici CarteEtudiant, qui vont correspondre à une table dans la base de données.

Cette correspondance est assurée via l’option –m , qui est l’abréviation de migration.

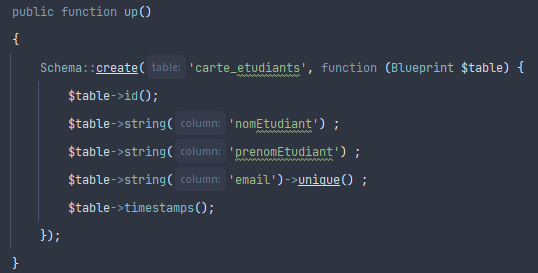
En français : effectuez une création d’un modèle de CarteEtudiant et préparez la migration correspondante.

Cette commande crée donc 2 fichiers :

1. Un fichier modèle, **app\CarteEnc.php**
2. Un fichier de migration : **database\migrations**\XXXX\_ create\_carte\_etudiants\_table.php

4.e modification de la structure de la table pour la carte d’étudiant

Editez le fichier **database\migrations**\XXXX\_create\_carte\_etudiants\_table.php, et modifiez la méthode (fonction) up() de la manière suivante.



4.f ; relancez la migration

Effectuez la commande suivante :

Avec PHPStorm :

Touche CTRL 2 fois puis :



Sans PHPStorm :



Cette commande équivaut à un ***rollback***, c’est-à-dire l’annulation de toutes les transactions en base de données, et à un ***migrate***, donc elle va réinitialiser la base de données en tenant compte des modifications apportées à la table.

**Partie n°2** : Création des vues

**Etape 5** : vues – création d’un composant LiveWire permettant d’ajouter une carte d’étudiant dans la base

**Blade** est le nom du moteur de templating permettant de faire des vues dynamique en **Laravel**.

C’est l’équivalent de **Twig** pour le framework **Symfony**, autre framework PHP très connu et utilisé.

Un outil de templating permet de générer du code pour les vues (HTML visualisé dans les navigateurs Web) de façon dynamique.

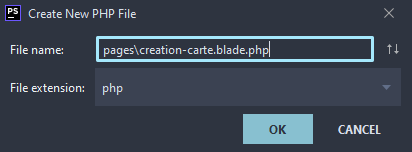
Le fichier principal se situe dans **resources**\**views\layouts\app.blade.php**

A la fin de ce fichier, vous trouverez une directive {{ slot }} , le mécanisme est similaire à celui étudié avec les composants Vue.js.

Le contenu inséré dans ce Slot est identifié avec des balises ouvrante <**x-app-layout**> et fermante </**x-app-layout**>

Etape 5.1 : Création d’un composant Blade

Dans resources\views, clic droit et File🡪 New PHP File :



Un composant nommé creation-carte.blade.php va être créé dans views\pages

Etape 5.2 : Modification du composant

Remplacez par le contenu du fichier **create.blade.php** fourni

**Partie n°3** : Création de la route et lien entre vue et contrôleur

**Etape 6** : création d’un contrôleur

Avec PHPStorm :

Exécutez la commande suivante :



Sans PHPStorm :

Exécutez la commande suivante :



Vous devez obtenir le message suivant :



**Etape 7** : ajout d’une route qui envoie vers le controleur :

Dans le fichier **routes\web.php** , ajoutez l’instruction suivante :



Remarque : une alternative est de :

1. Ecrire le code suivant à la place pour la route



1. Dans le fichier **App\Providers\RouteServiceProvider.php**, décommentez la ligne 29 (définition de l’attribut $namespace).

* Cela permet à Laravel de connaître l’emplacement des contrôleurs.

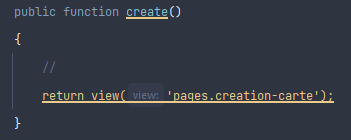
Rappel :

Vérification des routes en ligne de commande : ****

**Etape 8** : lien entre le contrôleur et la vue

Le contrôleur créé à l’étape 6 est situé dans : **app\Http\Controllers**

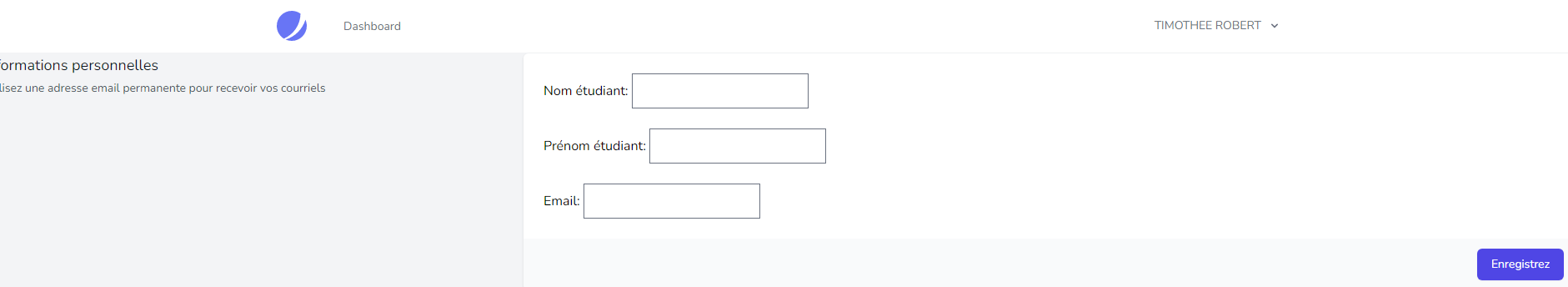
Modifiez la fonction create du fichier **CarteEncController.php** en ajoutant le code suivant :



**2ème checkpoint**

Adresse pour valider l’application à ce stade:

<http://carteenc.test/demandeCarte/create> : vous devez obtenir le formulaire suivant



**Partie n°4** : Persistance des données dans la base de données – Récupération des données sauvegardées

Dans le contrôleur, une méthode est déjà prête pour permettre de faire persister les données dans la base de données.

Il reste à compléter son code :

**Etape 9** : Enregistrement des données du formulaire

Ouvrir le fichier **CarteEncController.php** dans le répertoire **app\Http\Controllers** et

1. En haut du fichier on référence le modèle CarteEtudiant :



1. ajoutez le code suivant à la méthode (fonction) **store** :



**3ème checkpoint**

Saisissez des données dans le formulaire, enregistrez et vérifier que l’application vous redirige vers le Dashboard et que la demande a bien été enregistrée dans la base de données.

**Exercice** : changement de langue et ajout du lien pour accéder à la création de carte dans le DropDown

1. Changement de langue

Nous allons changer de langue de l’anglais au français, nous ne faisons pas de multilangue pour le moment.

Dans le fichier **config\app.php**, changez les entrées locale et fallback\_locale de en vers fr.

Ensuite créez un fichier resources\lang\fr.json et alimentez le.

Pour démarrer :

{  
 "Welcome to our website": "Bienvenue sur notre site",  
 "Dashboard" : "Tableau de bord"  
}

A chaque fois dans un fichier Blade que vous trouverez une variable de la forme suivante : {{ \_\_’nomVar’ }} vous pourrez l’ajouter à ce fichier avec une traduction française.

1. Ajout du lien dans le menu

Dans le menu resources\views\navigation-menu.blade.php,

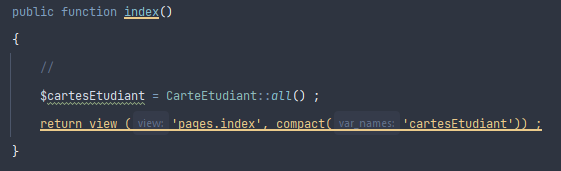
* Ajoutez un lien pour aller vers la page de demande de carte
* Supprimez ce qui est inutile (Teams, Api) : testez régulièrement ce que vous enlevez

**Etape 10** : Récupération des demandes de cartes effectuées sauvegardées dans la base de données.

1. on crée une nouvelle page blade dans le répertoire pages nommée index.blade.php

Pour récupérer des données déjà existantes, persistées dans la base de données, une méthode du contrôleur est déjà prévue, qu’il s’agit de compléter : il s’agit de la méthode index .

1. Méthode **index** du contrôleur :



Après avoir récupéré les données, cette méthode renvoie vers la vue **index.blade.php**. Le code est fourni, il s’agit de copier le code.

# Questions sur le PPE

Question 2 : à quoi sert le fichier web.php ? ouvrez le et indiquez quelles instructions ont été ajoutées à ce fichier ?

Quelle commande permet d’afficher l’ensemble des routes d’un projet Laravel ?

Question 3 : BDD et migration

1. Quel fichier permet de définir la BDD utilisée en développement ?
2. Quelle commande permet de créer un fichier de migration ? Quel est le fichier associé ?
3. Quelle commande permet de charger les données dans la BDD MySQL ?

Question 4 : Modèle

1. Quelle commande permet de créer un modèle ?
2. A quoi sert l’option -m ?

Question 5 : Controleur

1. Quelle commande permet de créer un contrôleur dans Laravel ?
2. A quoi sert l’option --resource

Question 6 : Framework MVC - pour l’ensemble des fichiers ci-dessous, indiquez s’il fait partie de Modèle, Vue ou Contrôleur ?

CarteEncController.php

create.blade.php

web.php

index.blade.php

User.php

2019\_11\_07\_HHMMSS\_create\_carte\_etudiants\_table.php

CarteEtudiant.php

edit.blade.php

Question 7 : Vues

1. Dans les vues du projet, les fichiers de type .blade.php, par exemple dans index.blade.php: à quoi servent les {{ }} ? En a t’on déjà rencontrées ?
2. Avec quel paramètre la vue **index.blade.php** est elle appelée ? que fait la fonction Php **compact** ?

Question 8 : Framework Laravel – Héritage

Prenons par exemple le code permettant de récupérer l’ensemble des cartes étudiant :



D’où vient la méthode all ? Récupérez le code de cette méthode.