**DOCUMENTATION PARKING :**

# *Attribution des places de parking :*

Ce projet a été réalisé en groupe de 3 personnes lors de ma 2ème année de BTS SIO durant lequel nous avons dû crée et programmé une application en Laravel.

Afin d’éviter le stationnement sauvage dans le labyrinthe qu’est le parking, il a été décidé d’attribuer à chaque membre qui le demandait une place de parking numérotée.

## *Le besoin attendu :*

* Le front-office doit être sécurisé et n’accepter que les demandes du personnel des ligues. Les inscriptions au service de réservation de place doivent être validées (ou créées) par un administrateur.
* L’administrateur, seul utilisateur du back-office, doit pouvoir éditer la liste des places et gérer les inscriptions des utilisateurs.
* Lorsqu’un utilisateur en fait la demande, une place libre lui est est attribuée aléatoirement et immédiatement par l’application, la réservation expire automatiquement au bout d’une durée par défaut déterminée par l’administrateur.
* Si une demande ne peut pas être satisfaite, l’utilisateur est placé en liste d’attente.
* L’utilisateur ne peut pas choisir la date à laquelle une place lui est attribué, les réservations sont toujours immédiates. Un utilisateur ne peut pas faire une demande de réservation s’il est en file d’attente ou qu’il occupe une place.
* Un utilisateur ou l’administateur peuvent fermer une réservation avant la date d’expiration prévue. Une fois celle-ci expirée, l’utilisateur doit refaire une demande s’il souhaite obtenir une place.

Caractéristiques à tenir compte lors du projet :

### *3.1  Sécurité*

* Protection des accès par mot de passe.
* Contrôles de saisie des données côté serveur.
* Contrôles de saisie côté client.
* Protection contre les attaques par injection.

### *3.2  Gestion des mots de passe*

* Fonction "mot de passe perdu ?".
* Hachage des mots de passe.

### *3.3  Espace utilisateur*

* Vérification de l’identité par saisie d’un mot de passe.
* Possibilité de visualiser le numéro de place attribuée, ainsi que l’historique des places précédemment attribuées.
* Possibilité de faire une demande de réservation.
* Possibilité de connaître son rang sur la file d’attente.
* Modification du mot de passe.

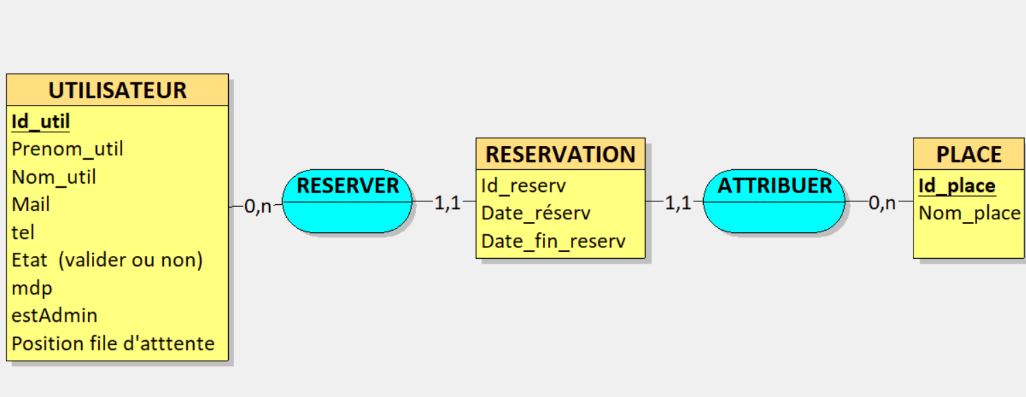
### *3.4  Espace administrateur*

* Protection de l’accès par mot de passe.
* Édition de la liste des utilisateurs, réinitialisation des mots de passe.
* Édition de la liste des places
* Consultation de la liste d’attente.
* Consultation de l’historique d’attribution des places.
* Attribution manuelle des places.
* Édition de la file d’attente (modification de la position des personnes en attente).

### *3.5  Pages web*

* Mise aux normes HTML5 et CSS des pages WEB

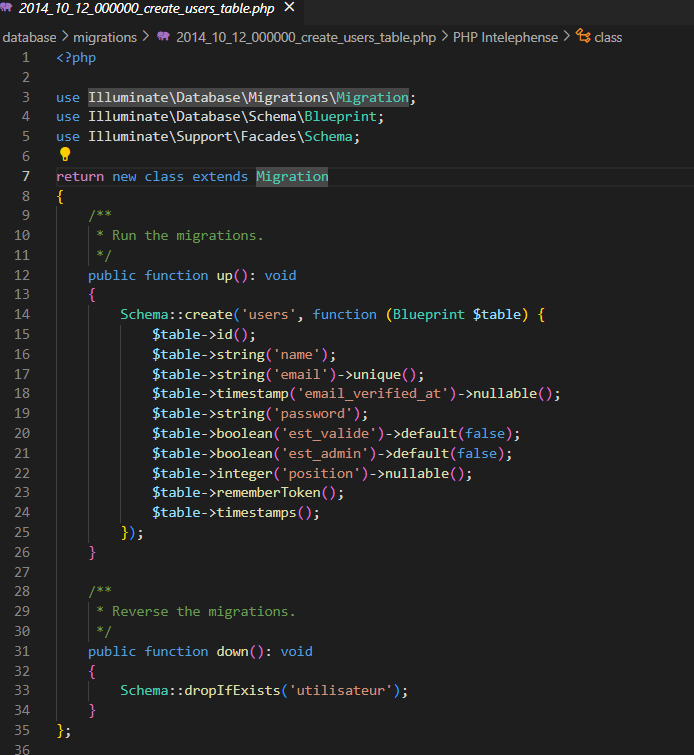
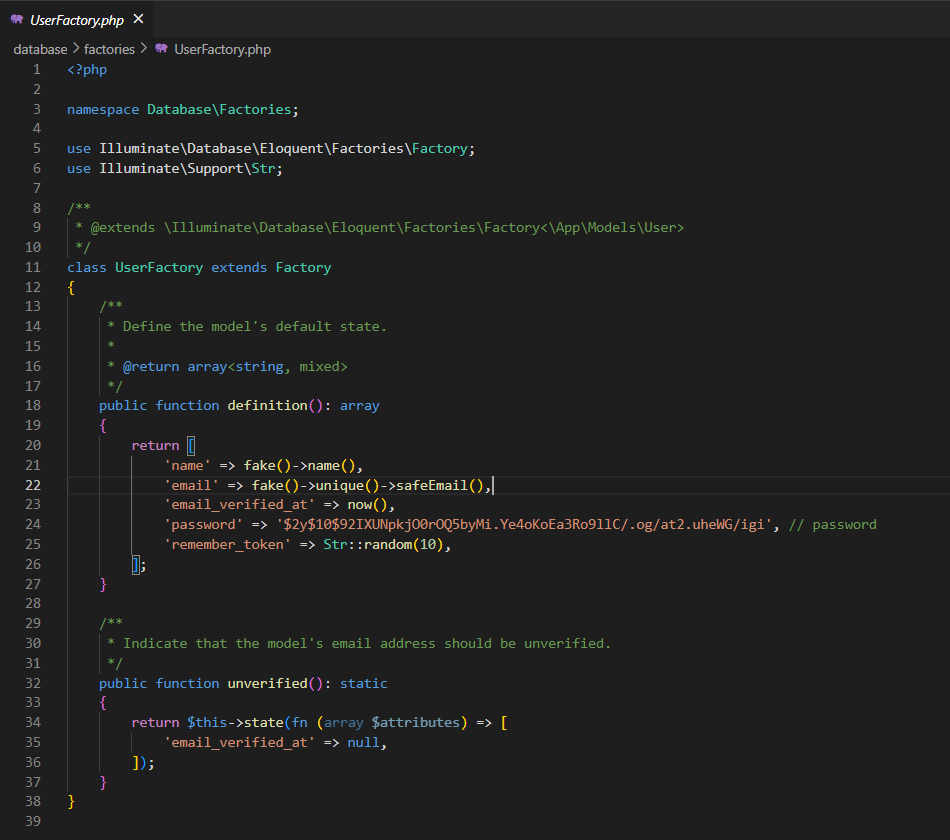
***MCD Réalisé pour cette application :***



Cette application a donc été programmée sur Laravel en PHP lors de différentes étapes :

La création des migrations spécifiques pour chaque table

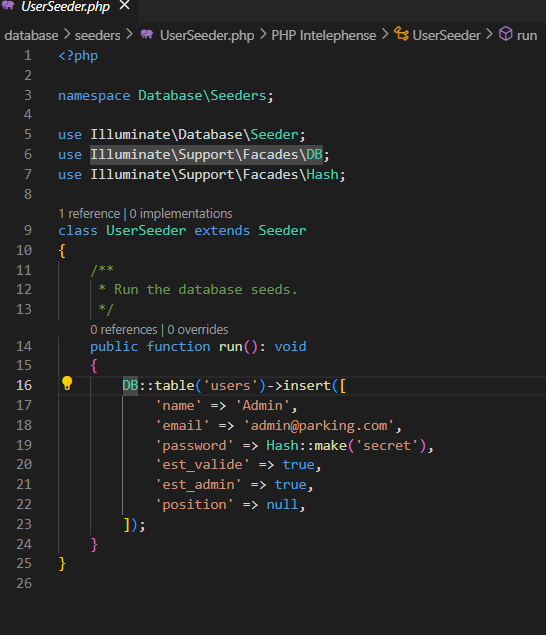
### Voici un exemple de migrations :

Dans une migration, il faut spécifier le type de données, si il peut être nul, si l’est unique, si il a une valeur par défaut etc.

Mise en place des Factories :

Une factorie contient des valeurs par défaut à renter pour chaque table crée.

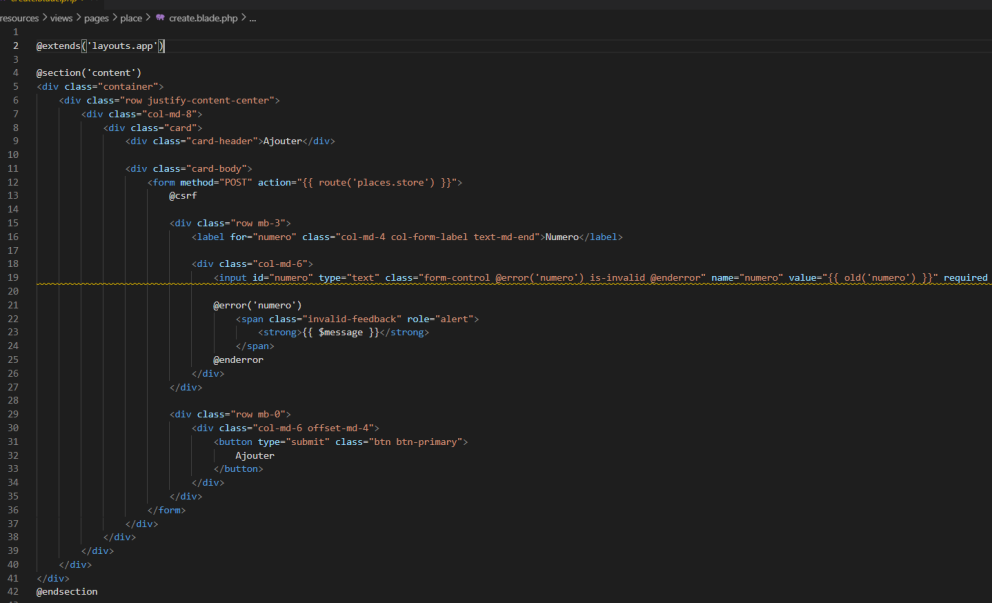
Ensuite il faut mettre en place les seeders :



Le seeder crée un jeu de données, pour l’environnement de développement ou pour les tests.

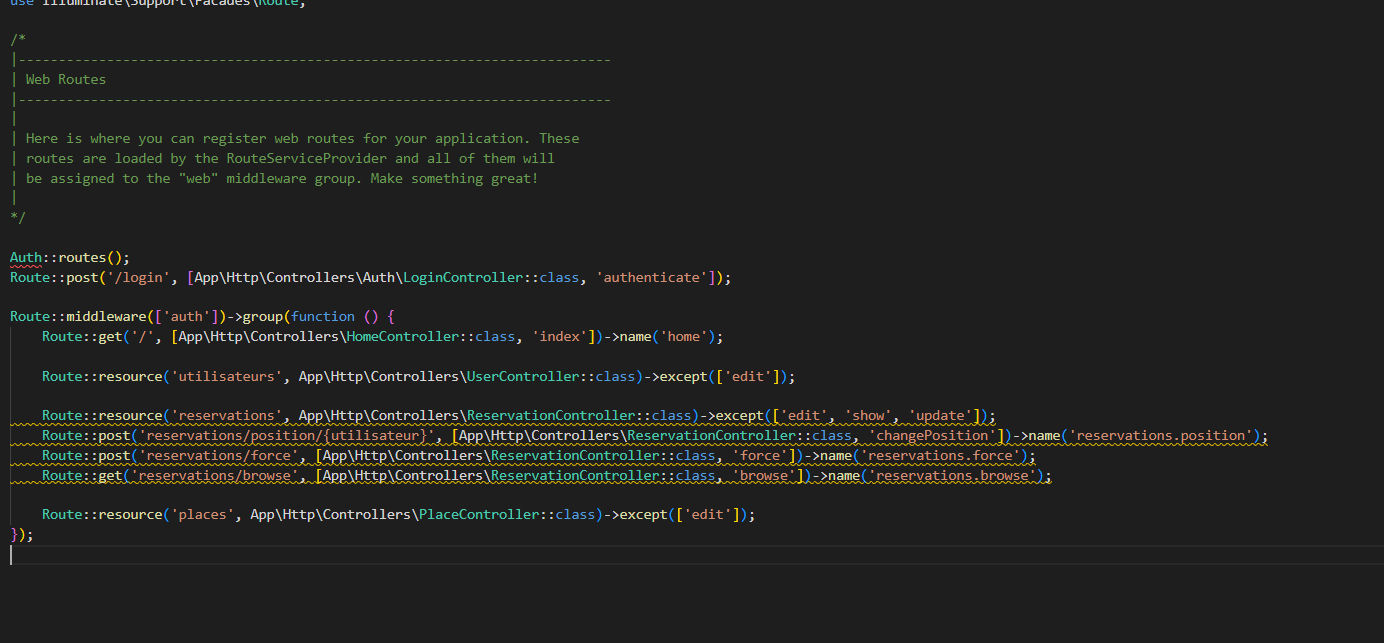
Ils se basent sur la définition des factories pour créer ses objets et remplir la base de données.

Et ici ajout de l’admin dans le seeder.

Initialisation et création des vues :

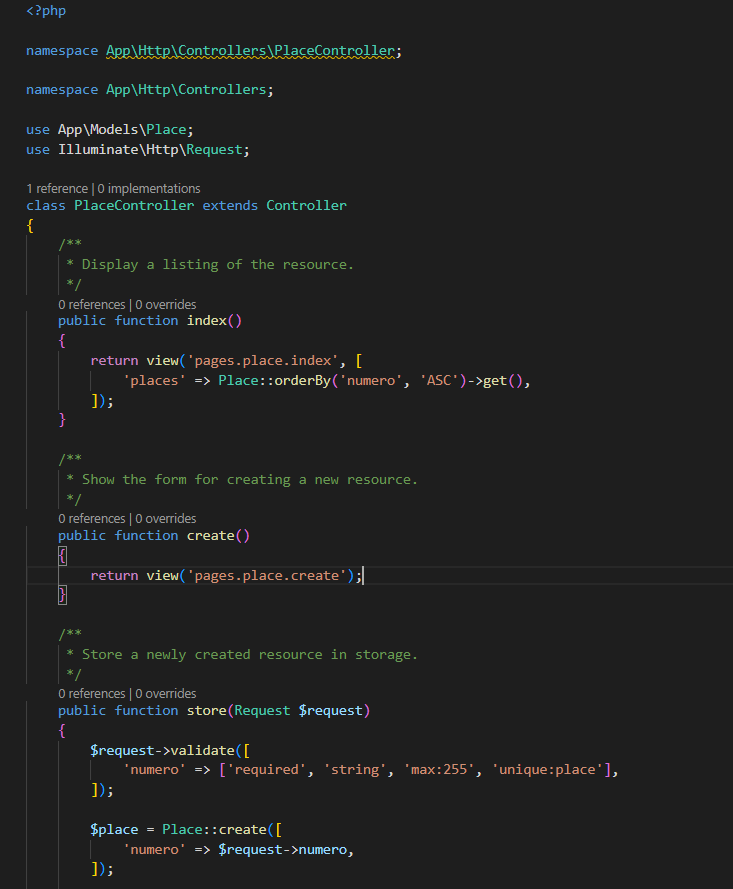
Les vues contiennent le code HTML du site, une vue c’est donc créer un fichier contenant du HTML qui sera retourné par une route.

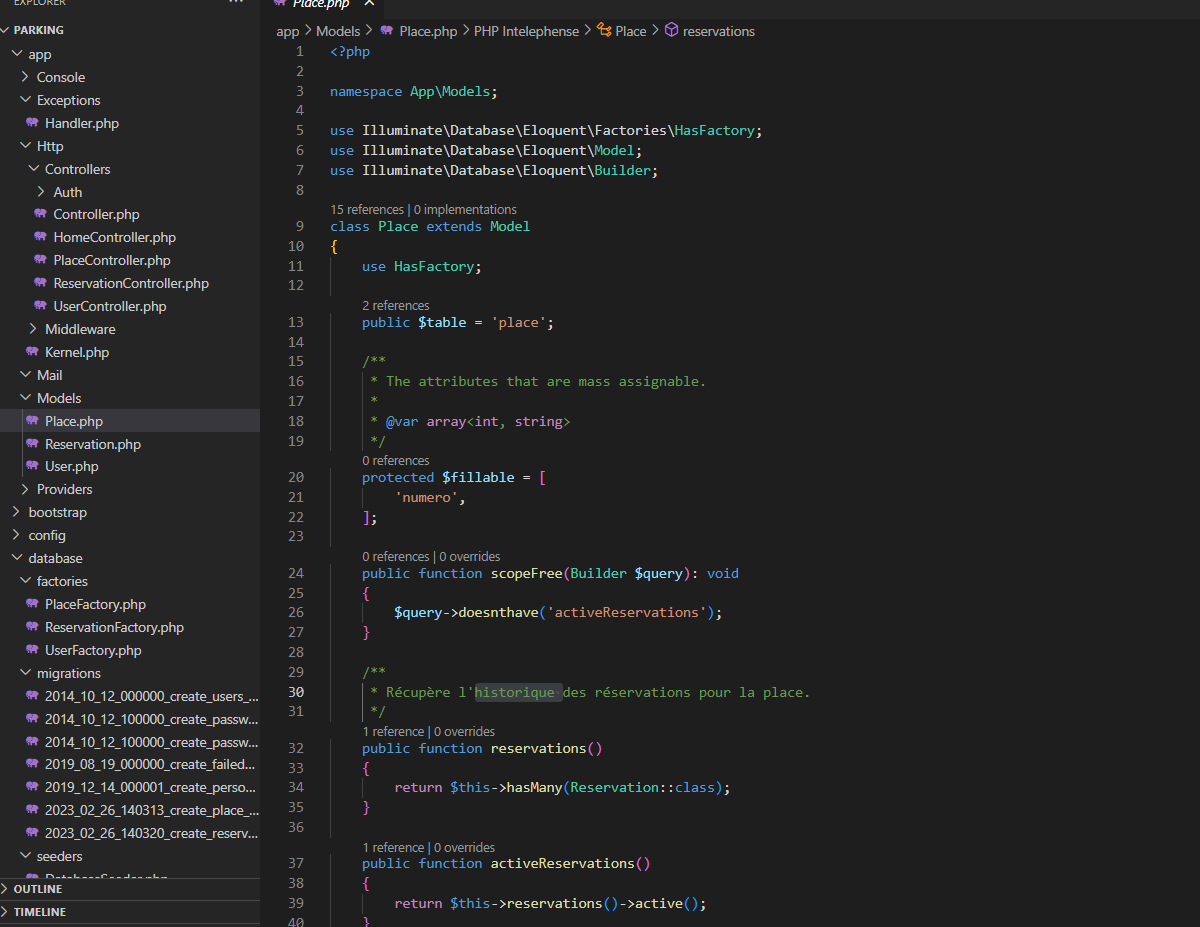
Il faut ensuite mettre en place les routes :



Le système de routes permet de faire correspondre un morceau de code aux différentes URL de l’application.

Initialisation et mise en place des controller :

Un controller est une classe qui va contenir différentes méthodes. Chaque méthode correspondant généralement à une opération (URL) de votre application. Il est ensuite possible de faire appelle à ce controller et cette méthode (appellée "action") en utilisant les routes.



Initialisation et mise en place des model :

Dans Laravel, chaque table de base de données a un “modèle” correspondant qui nous permet d'interagir avec cette table.