LARAVEL (8-2021) – FRAMEWORK PHP 2

SOMMAIRE

https://laravel.com/
https://laravel.com/docs/10.x
https://laravel.com/docs/11.x

ETAPE 2

https://laravel.com/
https://laravel.com/docs/10.x
https://laravel.com/docs/11.x

0 – Objectifs de l'étape 2

⇒ L'étape 2 :

⇒ Présente le package Breeze pour l'authentification (qui remplace UI)

⇒ se termine par le TP du chapitre 7 : installer le package Breeze dans un nouveau projet

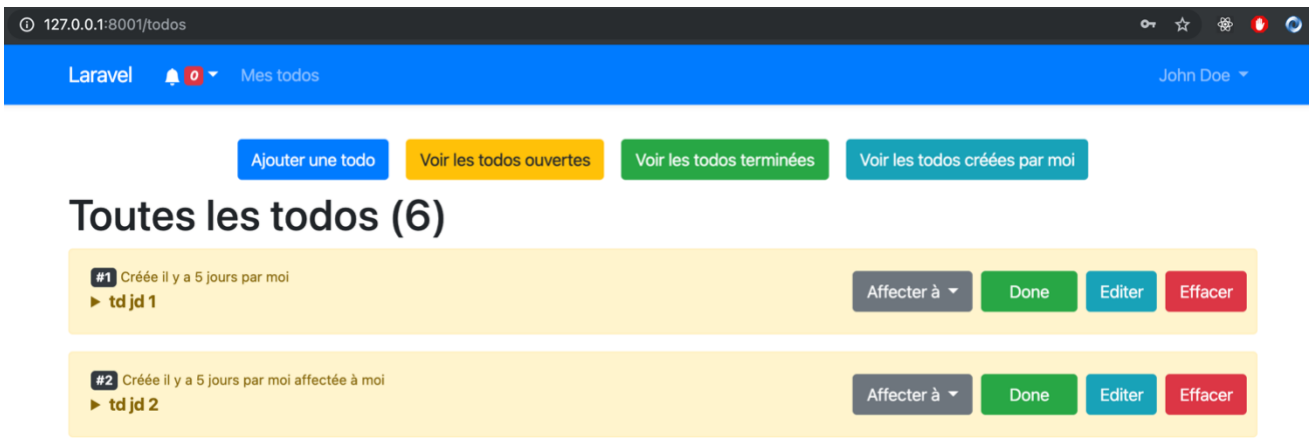
⇒ L'objectif de l'étape 2 est de se familiariser avec une application, la notion de migration, la notion de route, l'installation d'un package

Principe du document

- Les parties surlignées de bleu correspondent à des installation ou exercices à faire.
- Les parties surlignées en jaune sont des concepts importants.
- Les parties marquées (on peut sauter) peuvent être sautées.

Objectifs à atteindre pour l'ensemble de la formation

- Le cours et les différentes étapes de TP à partir de l'étape 2 consistent à produire une application ressemblant à :



1 - Installer le package d'authentification Breeze dans une application Laravel (v5)

Principes d'installation et de fonctionnement du package d'authentification Breeze

- Laravel nous permet de mettre en place des fonctionnalités plus facilement en installant des packages tout prêt.
- On va installer un package pour l'authentification (utilisateur, login, mot de passe, etc):

  ⇒ Le package « Breeze » remplace le package « UI » depuis la version 9 de Laravel.

  ⇒ C'est un package d'authentification des utilisateurs : https://github.com/laravel/breeze)

- Qui dit authentification dit enregistrement d'utilisateurs dans la BD
- On va donc installer les tables de la BD pout gérer les utilisateurs :

  ⇒ ça passe par des « migrations »

  ⇒ https://laravel.com/docs/11.x/migrations#main-content

- L'installation du package installe plusieurs routes.

  ⇒ Les routes sont des urls d'accès à l'application (au site WEB).

  ⇒ https://laravel.com/docs/11.x/routing#basic-routing

  ⇒ On va créer une page « à la main » avec une seule route.

2 – Installation- création de l'application – première migration

Création du projet Laravel matodolist

On repart d'une application vierge : « matodolist » qu'on va créer :

```
>composer create-project laravel/laravel matodolist
```

On teste le projet

```
>cd matodolist
>php artisan serve
```

Création de la base de données associée : matodolist_2025

- Notre application va utiliser une base de données. Des utilisateurs pourront se connecter.
- Le package Breeze qu'on va installer nous permettra d'installer des fonctionnalités pour les utilisateurs (enregistrement, login, logout, gestion des mots de passe).
- Pour faire ça, il faut avoir un serveur de BD qui fonctionne.

Installation d'un serveur de BD

- On installe un environnement pour la programmation WEB type WAMPP ou un équivalent.

  ⇒ Installer WAMP : https://www.wampserver.com/

  ⇒ Installer XAMPP : https://www.apachefriends.org/fr/index.html

Création de la BD « matodolist_2025 »

- Dans un client mysql (ça peut être phpMyAdmin) créez la BD matodolist_2025 :

```
Mysql> create database matodolist_2025;
Mysql> use matodolist_2025;
Mysql> show tables;
Empty set (0,00 sec)
```

⇒ Il faut démarrer un serveur MySQL : on peut le faire à partir d'un XAMPP (à éviter sur Mac pour Laravel : l'environnement est compliqué à mettre en place), WAMP (sur PC), MAMP (sur Mac) ou Laragon (que sur PC) ou ou « à la main.

⇒ Vérifier la cohérence du port de MySQL : probablement 3306

Mise à jour de l'environnement Laravel : .env et App/Providers/AppServicesProvider.php

Mise à jour du fichier .env

- Dans le fichier .env, mettez les bons DB_DATABASE, DB_USERNAME et DB_PASSWORD:

```
DB_DATABASE = matodolist_2025
DB_USERNAME = root
DB_PASSWORD =
```

Avant la mise à jour de la BD : App/Providers/AppServicesProvider.php

- Il est prudent de commencer par mettre à jour le fichier App/Providers/AppServicesProvider.php

  ⇒ Mais c'est peut-être déjà fait !!!

  ⇒ On ajoute la ligne suivante juste après le 1er use :

```
use Illuminate\Support\Facades\Schema; // juste après le 1er use
```

  ⇒ On ajoute la ligne suivante dans la fonction boot

```
Schema::defaultStringLength(191);       //  dans la fonction boot
```

- Une fois cela fait, on peut faire le php artisan migrate

Création des tables de la BD : php artisan migrate

Mise à jour de la BD : php artisan migrate

- On peut créer la BD avec les commandes suivantes :

```
>php artisan config:cache     # par prudence, on peut vider les caches
>php artisan cache:clear      # par prudence, on peut vider les caches
>php artisan migrate
```

```
  INFO  Preparing database.
  Creating migration table.............................. 74.66ms DONE

  INFO  Running migrations.
  0001_01_01_000000_create_users_table.................. 331.98ms DONE
  0001_01_01_000001_create_cache_table.................... 98.54ms DONE
  0001_01_01_000002_create_jobs_table.................... 435.29ms DONE
```

Consultation des tables créées dans la BD

- La commande « php artisan migrate » a créé 5 tables :

```
mysql> show tables;
+---------------------------+
| Tables_in_matodolist_2025 |
+---------------------------+
| cache                     |
| cache_locks               |
| failed_jobs               |
| job_batches               |
| jobs                      |
| migrations                | -> 3 migrations
| password_reset_tokens     |
| sessions                  |
| users                     | -> vide
+---------------------------+
```

  ⇒ Tout est vide sauf migrations.

- La table migration enregistre les migrations effectuées : c'est la première table créée.
- Les migrations sont les 3 premières commandes passées : créer 3 tables dont celles des users.
- On verra plus tard comment créer des migrations.

```
mysql> select * from migrations;
+----+-----------------------------------------+-------+
| id | migration                               | batch |
+----+-----------------------------------------+-------+
|  1 | 0001_01_01_000000_create_users_table    |     1 |
|  2 | 0001_01_01_000001_create_cache_table    |     1 |
|  3 | 0001_01_01_000002_create_jobs_table     |     1 |
+----+-----------------------------------------+-------+
```

- La table users est une table dans laquelle on stockera les caractéristiques des utilisateurs. Elle sera utilisée par le package Breeze.
- Pour le moment, il n'y a pas d'utilisateur.

Annexes concernant divers problèmes de BD (on peut sauter si pas de problème !)

Erreur la plus commune ;

C:\Users\bertrandliaudet\Desktop\test\vite>php artisan migrate
Illuminate\Database\QueryException

SQLSTATE[42000]: Syntax error or access violation: 1071 La clé est trop longue. Longueur maximale: 1000 (SQL: alter table `users` add unique `users_email_unique`(`email`))

=> Solution la plus commune : avant le php artisan migrate

- Il est possible qu'il faille commencer par mettre à jour le fichier App/Providers/AppServicesProvider.php

  ⇒ Mais c'est peut-être déjà fait !!!

  ⇒ On ajoute la ligne suivante juste après le 1er use :

```
use Illuminate\Support\Facades\Schema; // juste après le 1er use
```

  ⇒ On ajoute la ligne suivante dans la fonction boot

```
Schema::defaultStringLength(191);      //  dans la fonction boot
```

En cas d'erreur Illuminate\Database\QueryException

- Il faudra peut-être supprimer la BD et la recréer (pour la vider) :

```
Mysql> drop database matodolist
Mysql> create database matodolist
Mysql> use matodolist
```

- Il faudra peut-être appeler la BD : « laravel »

Problème de BD : Connection Refused

- Application http://127.0.0.1:8000/, menu Register : route : http://127.0.0.1:8000/register
- On saisit le formulaire : connection refused.
- Solution : Le serveur de BD ne tourne pas ! Il faut le démarrer avec XAMPP ou d'autres techniques.

Autres problèmes de BD :

- Application http://127.0.0.1:8000/, menu Register : route : http://127.0.0.1:8000/register
- On saisit le formulaire :

  ⇒ erreur dans le fichier vendor/laravel/framework/src/Illuminate/Database/Connection.php
- Solution : La BD n'existe pas : dans le fichier « .env », il faut choisir un nom pour la BD, vérifier que le username de la BD est bon ainsi que son passaword (root, vide sur PC, root, root sur MAC). Il faut ensuite créer un client mysql : create database laraveltodo ;
- Fichier .env sur MAC :

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=matodolist
DB_USERNAME=root
DB_PASSWORD=root
```

- Création de la BD dans un client mysql :

```
mysql>create database matodolist;
```

  – From MySQL 8.0, utf8mb4 is the default character set.

  ⇒ On peut aussi écrire : create database laraveltodo CHARACTER SET = utf8mb4 COLLATE = utf8mb4_ general_ci;
- Quand on met à jour le fichier .env, mieux vaut relancer le serveur puis l'application.

```
>php artisan serve
```

Problème de BD : les tables n'existent pas.

- Application http://127.0.0.1:8000/, menu Register : route : http://127.0.0.1:8000/register
- On saisit le formulaire : la table users n'existe pas.
- On a un message qui propose de faire un « php artisan migrate » ou de passer par un bouton qui va le faire pour nous.
- Une fois le migrate fait, toutes les tables de la BD sont créées. On peut les regarder en faisant un « show tables » en console mysql.

Problème fonctionnel : le mot de passe doit avoir au moins 8 caractères

- Application http://127.0.0.1:8000/, menu Register : route : http://127.0.0.1:8000/register
- On saisit le formulaire : ça fonctionne. On crée un utilisateur John Doe et un utilisateur Fred.
- On peut mettre les adresse mail qu'on veut. Si on met une adresse mail fonctionnelle, on recevra des mails de l'application (c'est vite lassant !).
- Le mot de passe doit avoir au moins 8 caractères : exemple : laravelmdp
- On peut regarder l'utilisateur saisi dans la BD en console MySQL :

```
mysql>desc laraveltodo.users ;
mysql>select id, name, email, password from laraveltodo.users;
```

3 - Installation du package laravel Breeze

Principes techniques pour l'authentification

- Un package à installer : le package Breeze.

Le package Breeze

- packagist.org : chercher breeze

  ⇒ https://packagist.org/packages/laravel/breeze : on trouve des informations sur l'installation.

- On peut aussi regarder sur github, on y trouve des informations sur le package breeze

  ⇒ https://github.com/laravel/breeze : on trouve aussi des informations sur github.

Charger Breeze avec composer

- On trouve les informations sur le package breezz et son installation sur packagist.org, ici : https://packagist.org/packages/laravel/breeze
- La doc est directement dans la doc laravel : https://laravel.com/docs/11.x/starter-kits#laravel-breeze
- Pour installer le package Breeze, on tape, en ligne de commande, dans le dossier « matodolist », la commande « composer require » :

```
> composer require laravel/breeze --dev
```

⇒ On utilise require --dev pour le développement. Pour la production on peut s'en passer.

⇒ La commande « composer require » permet de charger un package : ici Breeze.

Résultats de l'installation : php artisan

- Après l'installation, php artisan montre de nouvelles commandes : breeze :install
- php artisan c'est pareil que php artisan list

```
> php artisan
...
breeze
  breeze:install        Install the Breeze controllers and resources
...
```

⇒ On pourra passer la commande php artisan breeze :install

⇒ On verra ce que ça fait.

- Aide : https://laravel.com/docs/11.x/starter-kits#laravel-breeze

Installation de l'authentification dans notre projet :

```
> php artisan breeze:install
```

⇒ On choisit blade

⇒ Puis les 2 choix par défaut (ou le dark mode)

Bilan d'architecture

⇒ On a ajouté le dossier node_modules

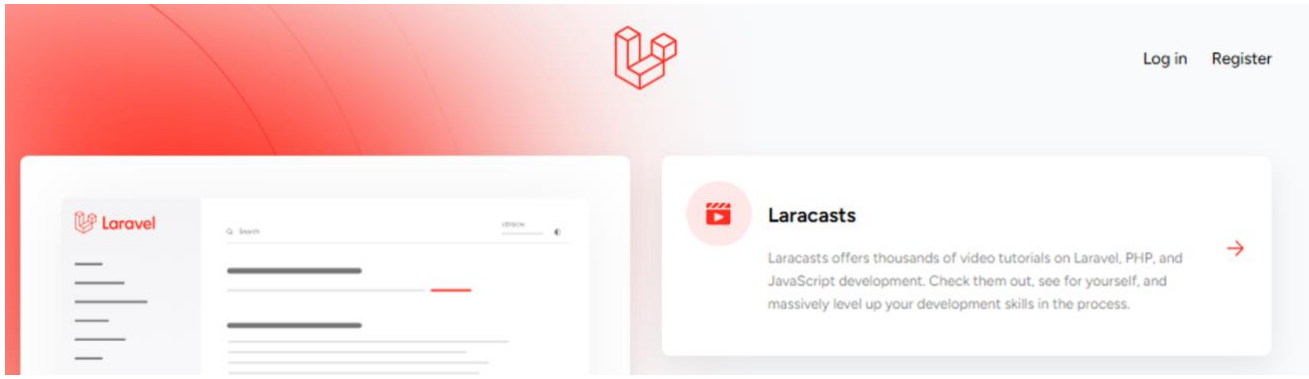
4 - Test de l'installation de l'authentification

Démarrage du serveur et d'un client :

```
> php artisan serve
Starting Laravel development server: http://127.0.0.1:8000
```

Route / : Localhost :8000

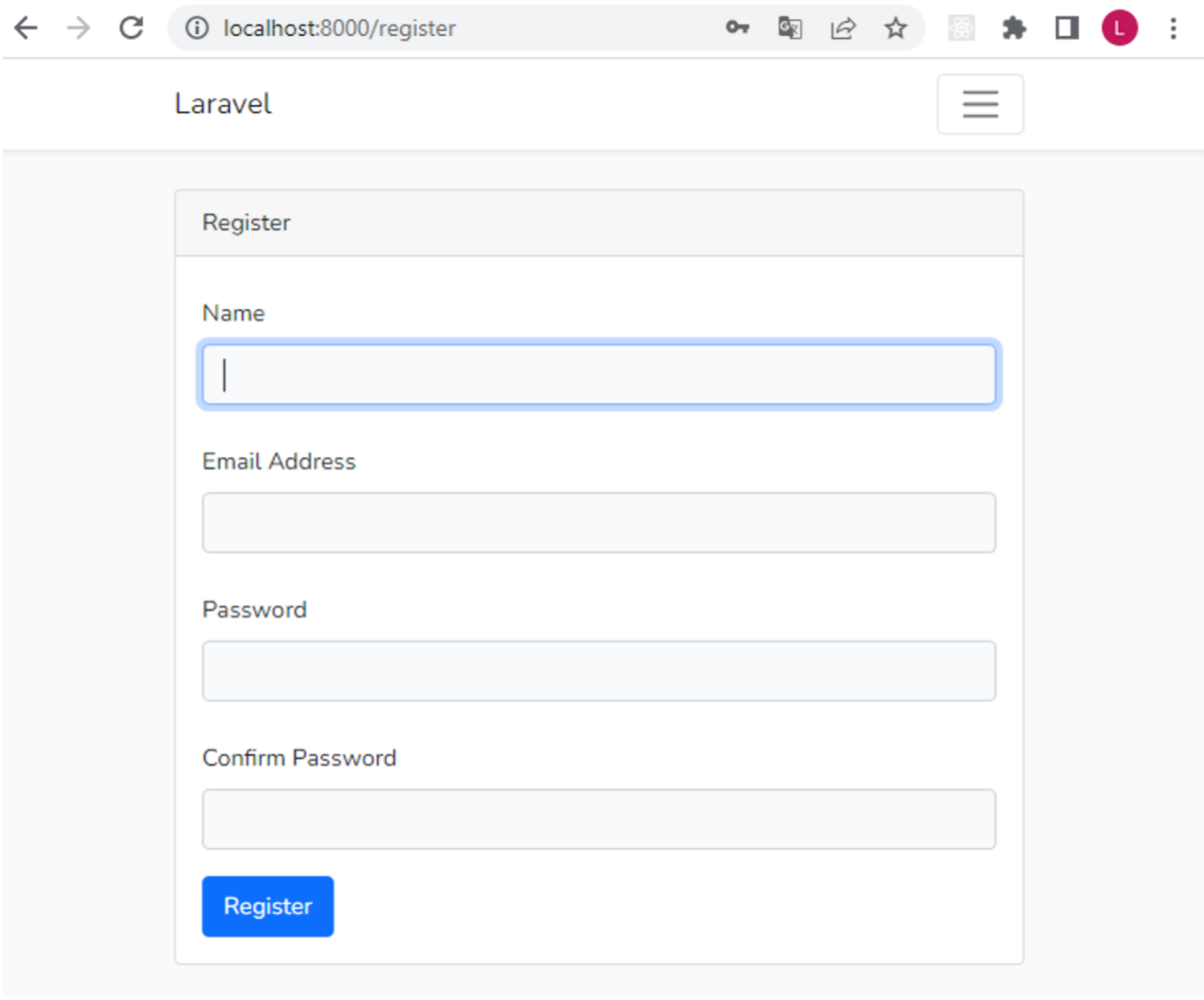
⇒ On a le « login » et le « register » :



Version sombre :

Route /register -> Enregistrement d'un utilisateur

- on clique sur Register :

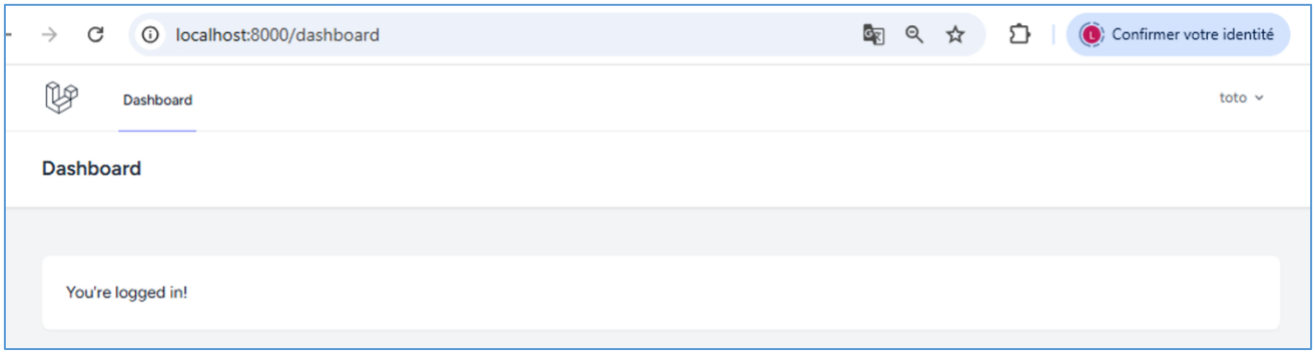


⇒ On a la route : localhost :8000/register

⇒ On met un nom, un mail, un password d'au moins 8 caractères.

Route /dashboard -> Un utilisateur est logué

- On est logué :



⇒ On a la route : localhost :8000/dashboard : c'est la route de la page d'accueil de l'utilisateur loggué, ici « toto ».

⇒ Le menu «toto » permet un logout.

⇒ Le menu « Laravel » ramène à la page l'accueil.

- Tester tous les usages possibles à ce stade.

Vérification en BD

- On peut vérifier la présence des utilisateurs créés dans la BD.
- Dans une console mysql :

```
mysql> select * from users;
+----+----------+--------------+---------------------+
| id | name     | email        | created_at          |
+----+----------+--------------+---------------------+
|  1 | bertrand | bl@gmail.com | 2021-02-27 17:43:14 |
+----+----------+--------------+---------------------+
1 row in set (0,00 sec)
```

Routes

Voir toutes les routes :

- La commande suivante permet d'afficher toutes les routes :

```
> php artisan route:list
```

- Seuls les routes avec un méthode GET (ou HEAD) sont accessibles par l'URL : on peut toutes les tester :

  ⇒ http://localhost:8000/ : amène à la page d'accueil. La route d'entrée est donc : http://127.0.0.1:8000/ ou http://localhost:8000/

  ⇒ http://localhost:8000/register : amène sur la page de création d'un utilisateur si personne n'est logué, ou sur la page home de l'utilisateur logué.

  ⇒ http://localhost:8000/login : amène sur la page de login

  ⇒ http://localhost:8000/home : amène sur la page de login si personne n'est logué, ou sur la page home de l'utilisateur logué.

  ⇒ http://localhost:8000/password/reset : amène sur une page de reset d'un password.

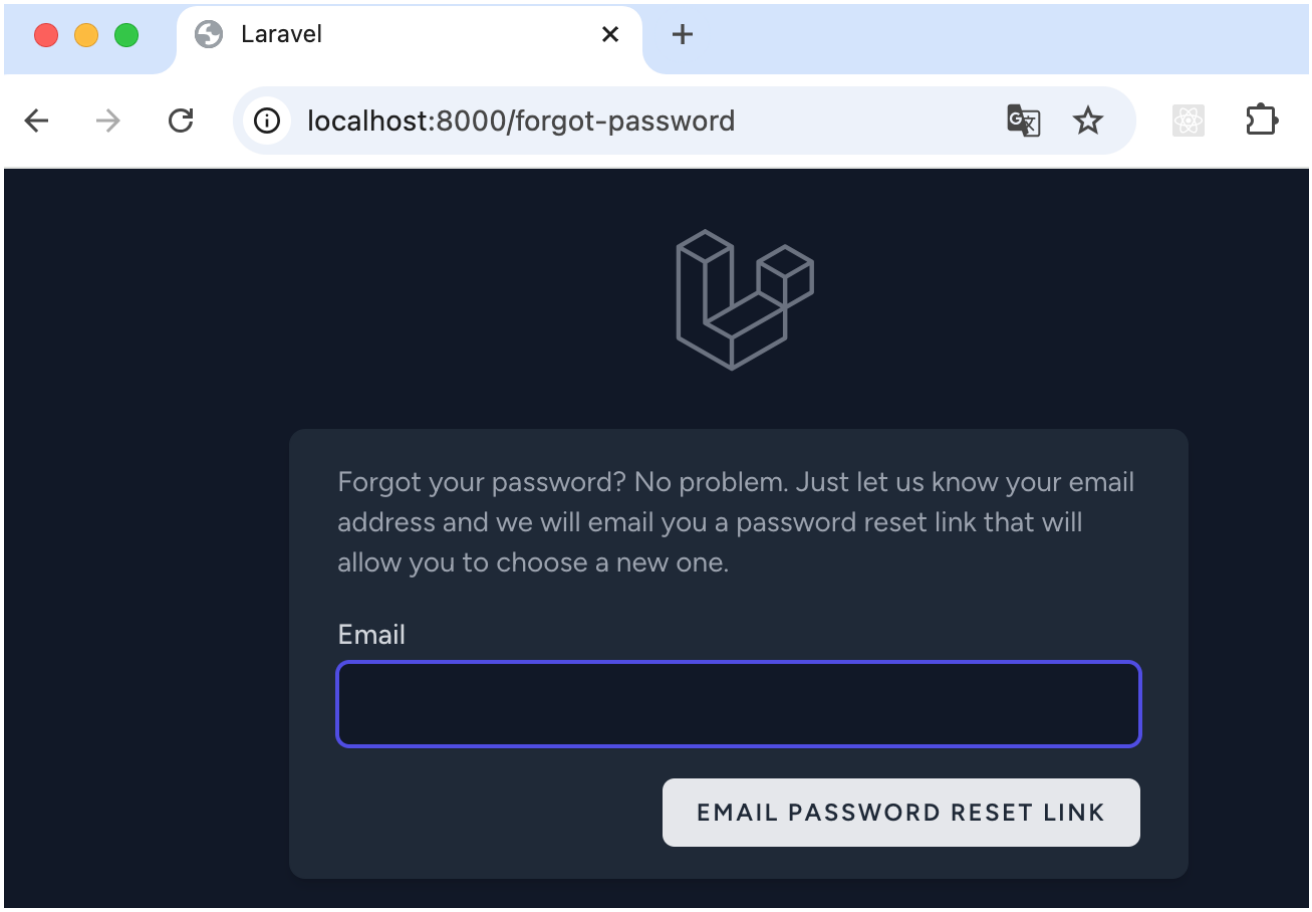
  ⇒ http://localhost:8000/password/reset/1 : amène sur une page de changement de password.

  ⇒ http://localhost:8000/logout : ne marche pas : c'est une route avec une méthode POST.

Password oublié

- La commande suivante permet d'afficher toutes les routes :
- On peut aller sur la route « forgot-password » si on a oublié son password.
- Le paramétrage est compliqué : dans .env il faut mettre à jour la partie SMTP
- On peut créer un compte gmail dédié. Mais c'est compliqué.
- Il faut créer un compte dédié et mettre à jour le fichier .env.
- Celui-ci-dessous marche (le password reste à mettre !)

```
MAIL_MAILER=smtp
MAIL_HOST=smtp.gmail.com
MAIL_PORT=587
MAIL_USERNAME=laravelbl@gmail.com
MAIL_PASSWORD="xxxxxxxxxxxxxxxxxxx"
MAIL_ENCRYPTION=tls
MAIL_FROM_ADDRESS=laravelbl@gmail.com
MAIL_FROM_NAME="${APP_NAME}"
```



5 – Architecture : fichiers modifies : routes et MVC

Routes

Fichier des routes : routes/web.php

- Au départ, le fichier ne contient une seule route : « / ».

```
Route::get('/', function () {
    return view('welcome');
});
```

- Une route appelle un contrôleur : ici c'est la fonction anonyme « function ».
- Le contrôleur fait des calculs puis appelle une vue : ici la vue « welcome » (fichier ressources/views/welcome.blade.php). C'est un fichier « blade » : il y a un « moteur de template » blade qui permet de gérer du code type PHP dans le HTML de la vue.

Après installation du package Breeze

- On a 1 nouvelle route get :

```
Route::get('/dashboard', function () {
    return view('dashboard');
})->middleware(['auth', 'verified'])->name('dashboard');
```

⇒ La route « '/dashboard » : c'est la page d'accueil une fois logué : elle retourne la vue « dashboard ».

⇒ Les routes de Auth : ce sont toutes les routes d'authentification. On retrouve le nom « Auth » dans le dossier des actions des routes dans la commande php artisan routes :list.

- On a 3 routes de middleware :

```
Route::middleware('auth')->group(function () {
    Route::get('/profile', [ProfileController::class, 'edit'])-
>name('profile.edit');
    Route::patch('/profile', [ProfileController::class, 'update'])-
>name('profile.update');
    Route::delete('/profile', [ProfileController::class, 'destroy'])-
>name('profile.destroy');
});
```

⇒ Le middleware, c'est la couche du milieu : les routes qui ne sont pas « principale ».

⇒ Ici il y en a 3 :

⇒ Une route **get**, accessible directement dans l'url, pour accéder aux données du profile de l'utilisateur connecté.

⇒ Une route **patch** qui permet de modifier les données du profile de l'utilisateur connecté.

⇒ Une route **delete** qui permet de supprimer un utilisateur connecté.

Controller

Fichier App\Http\Controllers\ProfileController.php

- C'est le controller appelé par les routes du middleware.
- Selon les méthodes (edit, update, destroy), il appelle des vues ou redirige vers des routes.

Fichiers dans App\Http\Controllers\Auth

- Tous les fichiers dans ce dossier Auth sont les controllers pour l'authentification.

View

Fichier des vues initial : ressources/views/welcome.blade.php

- La vue « welcome » correspond au ficher « welcome.blade.php » dans le dossier « ressources/views ».

  ⇒ C'est un fichier « blade », c'est-à-dire un fichier template html avec du code php qui sera compilé par blade en pur html (comme babel traduit du html en javascript).

  ⇒ On constate la présence de PHP précédé de « @ »

Fichier ressources/views/dashboard.blade.php

- C'est la vue appelée par le controller.
- C'est un fichier « blade » : un template qui intègre du HTML et du PHP.

Fichiers dans ressources/views/Auth

- Tous les fichiers dans ce dossier Auth sont les vues pour l'authentification.

6 - Synthèse

Commandes

```
>composer create-project laravel/laravel matodolist
>cd matodolist
>php artisan serve

Mysql> create database matodolist

> # mettre à jour le fichier App/Providers/AppServicesProvider.php
  # On ajoute la ligne suivante juste après le 1er use :
  # use Illuminate\Support\Facades\Schema; // juste après le 1er use
  # On ajoute la ligne suivante dans la fonction boot
  # Schema::defaultStringLength(191);      //  dans la fonction boot

>php artisan config:cache     # par prudence, on vide les caches
>php artisan cache:clear      # par prudence, on vide les caches
>php artisan migrate

Mysql> use matodolist
Mysql> show tables

>composer require laravel/breeze --dev

>php artisan route:list
```

Fichiers modifiés

Route

- routes/web.php

Controller

- App\Http\Controllers\HomeController.php
- App\Http\Controllers\Auth*

View

- ressources/views/home.blade.php
- ressources/views/Auth*

node_modules

- on a créé un dossier node_modules

Manipulation d'un projet laravel

Réduction de la taille du dossier

➢ ***Suppression des fichiers lock***

➢ ***Suppression du dossier vendor et du fichier composer.lock***

- On peut les refabriquer :

```
>composer update
```

➢ ***Suppression du dossier node_modules et npm update***

- On peut les refabriquer :

```
>npm update
```

- A noter que à ce stade, l'application marche sans le dossier node_modules.

7 - TP de synthèse : installer le package Breeze dans un nouveau projet

Installer le package Breeze dans un nouveau projet Laravel qu'on appellera matodolist.

Suivre les étapes présentées dans le cours et qui sont résumées ici :

1. `C:> composer create-project laravel/laravel matodolist`

   ⇒ `C:/matodolist>php artisan serve` pour vérifier

2. `Mysql> create database matodolist;`

   ⇒ `Mysql> show databases ; pour vérifier`

   ⇒ `Mysql> use databases ; pour vérifier`

   ⇒ `Mysql> show tables ; pour vérifier`

3. SublimeText> mettre à jour le fichier .env

   ⇒ `14 : DB_DATABASE=matodolist`

   ⇒ `Il faut aussi vérifier : Host, Port, Username, Password`

4. Dans le fichier App/Providers/AppServicesProvider.php : avant la migration

   ⇒ On ajoute la ligne suivante juste après le 1er use :

   ```
   use Illuminate\Support\Facades\Schema; // pour migration - juste après le 1er use
   ```

   ⇒ On ajoute la ligne suivante dans la fonction boot

   ```
   Schema::defaultStringLength(191);      // pour migration - dans la fonction boot
   ```

5. `C:/matodolist> php artisan migrate`

   ⇒ show tables dans MySQL pour vérifier

6. `C:/matodolist> composer require --dev laravel/breeze`

   ⇒ php artisan pour vérifier

7. `C:/matodolist> php artisan breeze:install`

   ⇒ Ça ajoute un dossier node_modules

Tester l'application

1. `C:/matodolist> php artisan serve`
2. Chrome> localhost:8000

   ⇒ on crée un utilisateur.

   ⇒ On passe par « register »

   ⇒ Exemple : toto, toto@gmail.com, mot de passe : totototo
3. `Mysql> select * from users ;` pour vérifier dans MySQL
4. On est sur le « dashboard » = « tableau de bord »
   a. On peut regarder le profile du user : route profile
   b. On peut se déloguer : retour à la route /
5. Quand on est sur la route / : on peut se connecter « toto »
   a. On arrive sur la route : login
6. On peut lister toutes les routes :

   ⇒ php artisan route :list
7. Testez l'envoi du mail en cas d'oubli du password :

   ⇒ C'est compliqué ! Cf. §5 Password oublié.
8. Supprimez vendor, node_modules et les lock :

   ⇒ Vérifier que ça ne marche plus.
9. Reachargez vendor :

   ⇒ composez update

   ⇒ Vérifiez que vendor est de nouveau là.

   ⇒ Vérifiez que ça marche.
10. Rechargez node_modules :

    ⇒ npm update

    ⇒ Véririfiez que node_modules est de nouveau là.

    ⇒ Vérifiez que ça marche toujours.