Mise en route de la démo

Créer une base de données nommée "**moneyvalue**"dans xampp/mamp avec PHPmyadmin

Ouvrez le projet dans un terminal de vs code et mettez le prompt dans le dossier api.

> composer install
> php artisan migrate
> php artisan db:seed –class=CurrencySeeder
> php artisan db:seed –class=CurrencyPairSeeder

lancer le server:
> php artisan serve

puis rendezvous dans le dossier front.
> npm install
> npm run serve

l'application front fonctionne sur localhost:8080

L'application possède la fonctionnalité d'inscription afin de pouvoir procéder à la démo. Cette fonctionnalité sera désactivée pour la prod.
Les routes publiques de l'api (disponibles pour le développeur) ainsi que les route protégées destinées à l'administrateur sont spécifiées et décrites dans le document annexe.
Le dashboard administrateur possède trois composants permettant une gestion des devises, une gestion des paires de devises, et un convertisseur utilisant l'api publique (ce qui permet de vérifier que celle-ci fonctionne). Les développeurs ont cependant un endpoint leur permettant de vérifier la mise en service de l'API.

Configuration Fortify additionnelles

1. Aller dans **vendor\laravel\framework\src\Illuminate\Foundation\Http\Middleware\VerifyCsrfToken.php**

mettre un ' **!** ' devant la ligne ''$this→runningUnitTests()'' sinon la vérification du CSRF échouera toujours.

```php
    /**
     * Handle an incoming request.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Closure  $next
     * @return mixed
     *
     * @throws \Illuminate\Session\TokenMismatchException
     */
    public function handle($request, Closure $next)
    {
        if (
            $this->isReading($request) ||
            !$this->runningUnitTests() ||
            $this->inExceptArray($request) ||
            $this->tokensMatch($request)
        ) {
```

2. Dans **vendor\laravel\fortify\src\Http\Responses\LoginResponse.php**

```
public function toResponse($request)
{
    return $request->wantsJson()
                ? response()->json([
                    'message' => 'Login successful',
                    'user' => $request->user(),
                    'token' => $request->user()->createToken('your-token-name')->plainTextToken,])
                : redirect()->intended(Fortify::redirects('login'));

}
```

```php
<?php

namespace Laravel\Fortify\Http\Responses;

use Laravel\Fortify\Contracts\LoginResponse as LoginResponseContract;
use Laravel\Fortify\Fortify;

class LoginResponse implements LoginResponseContract
{
    /**
     * Create an HTTP response that represents the object.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Symfony\Component\HttpFoundation\Response
     */
    public function toResponse($request)
    {
        return $request->wantsJson()
                    ? response()->json([
                        'message' => 'Login successful',
                        'user' => $request->user(),
                        'token' => $request->user()->createToken('your-token-name')->plainTextToken,
                    ])
                    : redirect()->intended(Fortify::redirects('login'));
    }
}
```