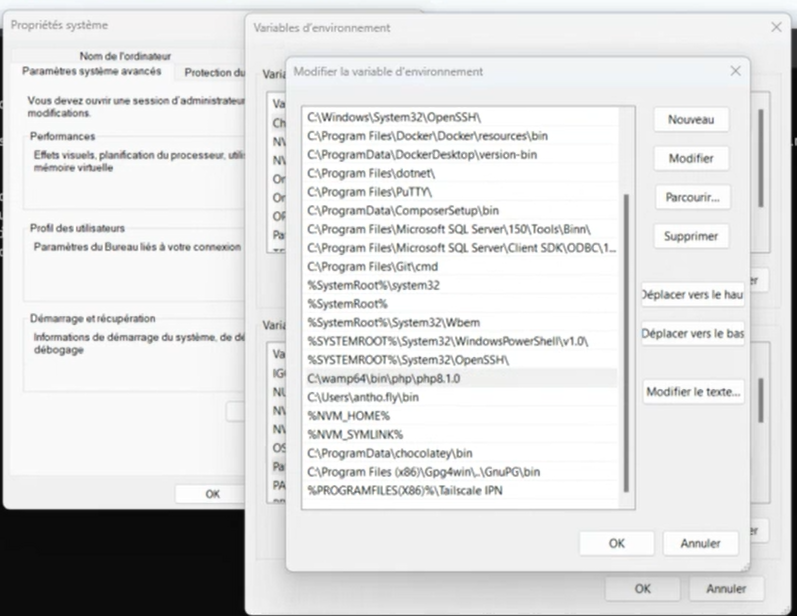
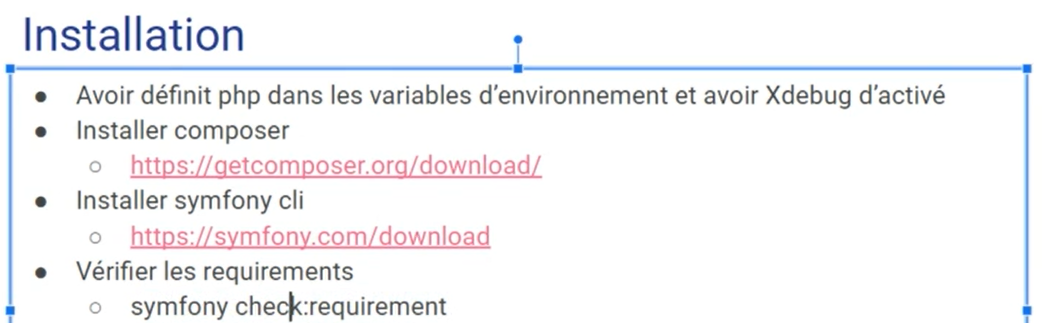


PHP

* Avoir PHP d’installé : en ligne de commande, taper php – v pour vérifier
* Variables d’environnement du système : path pointe vers php
* 

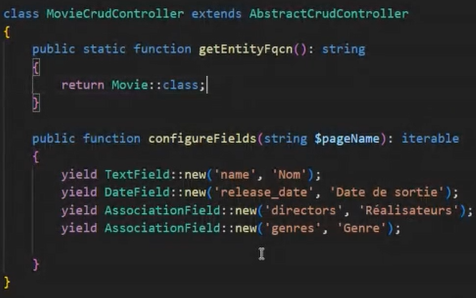
Installation du projet symfony



* Installer un nouveau projet symfony
  + symfony new my\_project\_directory --version="6.2.\*" –webapp
  + Aller dans le dossier créé
    - cd my\_project\_directory
* lancer le serveur
  + symfony server:start -d
* Afficher en français par défaut
  + Dans le fichier config=> packages => translation.yaml
    - framework: default\_locale: fr

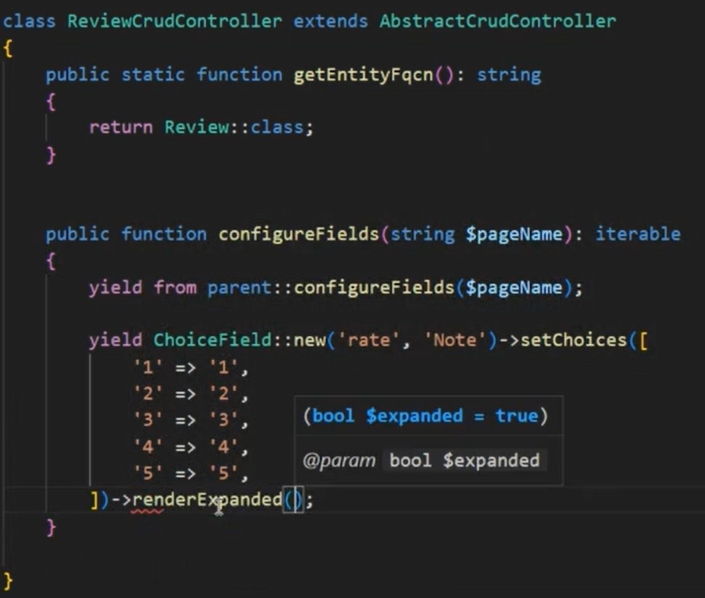


* Controlleur de page :
  + symfony console make:controller PageController
  + Définir cette page comme la page d’accueil
    - Dans PageController.php => route => remplacer /page par /
* Templates (twig)
  + Créer un dossier partials
    - Y mettre le header dans un fichier \_header.html.twig
    - Y mettre le footer dans un fichier \_footer.html.twig
  + Dans base.html.twig, insérer le header et le footer
    - {% include ‘partials/\_header.html.twig’ %}
    - {% include ′partials/\_footer.html.twig′ %}
  + Pour les images
    - Les mettre dans le dossier public
    - src=“{{ asset(‘images/logo.svg’) }}”
* Easyadmin
  + composer require easycorp/easyadmin-bundle
  + Création du dashboard
    - php bin/console make:admin:dashboard
      * on met les paramètres par défaut
    - Dans DashboardController.php => fonction index
      * Remplacer return parent ::index() ; par
        + return $this->render('admin/dashboard.html.twig');
    - Dans le dossier templates
      * Créer un dossier admin
      * Dans ce dossier admin, créer un nouveau fichier dashboard.html.twig’
        + {% extends '@EasyAdmin/page/content.html.twig' %}
        + {% block content\_title %}Tableau de bord{% endblock %}
        + {% block main %}
        + <h2>Bienvenue sur l'administration du garage Vincent Parrot</h2>
        + {% endblock %}
* Base de données
  + Dans le fichier .env : DATABASE\_URL="mysql://root:@127.0.0.1:3306/dB\_test?serverVersion=mariadb-10.4.32&charset=utf8mb4"
* Lancer XAMPP et cliquer sur admin
* Recharger la base de données à partir du fichier SQL (je viens de le modifier)
* Easyadmin : Sécuriser l’interface d’administration : <https://symfony.com/doc/current/the-fast-track/fr/15-security.html>
  + Créer l’entité Users
    - symfony console make:user Users
      * Valider les paramètres par défaut
  + Faire la migration vers la base de données (ma base existe déjà donc pas sûr)
    - symfony console make:migration
    - symfony console doctrine:migrations:migrate -n
  + Configurer le système d’authentification
    - symfony console make:auth
      * 1 => login
      * AppCustomAuthenticator
      * Paramètres par défauts (x3)
      * 0 activate when the user checks a box
  + Ajouter les règles de contrôle d’accès
    - Dans le fichier Config=>packages=> security.yaml
      * - { path: ^/admin, roles: ROLE\_ADMIN }
* Création des entités (ils apparaitront en anglais pour l’utilisateur mais tant pis)
  + symfony console make:entity
    - cars
      * image : blob
      * title : string
      * year : integer
      * fuel : string
      * kilometer : integer
      * price : integer
    - contacts
      * lastname : string
      * firstname : string
      * email : string
      * phone : string
      * subject : string
      * message : text
    - opening\_hours
      * day : string
      * hour : string
    - opinions
      * name : string
      * comment : text
      * grade : integer
      * status : boolean
    - services
      * title : string
      * image : blob
      * text : text
  + modifier les relations entre les entités (à voir si nécessaire)
    - symfony console make:entity
      * cars => car\_id => relation => many to many => users
      * opinions => opinion\_id => relation => many to many => users
      * contacts => contact\_id => relation => many to many => users
      * contacts => contact\_id => relation => many to one => cars
      * opening\_hours => opening\_id => relation => many to one => users
      * services => service\_id => relation => many to one => users
* Easyadmin : creation des CRUD
  + php bin/console make:admin:crud (validation des paramètres par défaut)
    - cars
    - opinions
    - contacts
    - opening\_hours
    - services
    - users
  + modifier le fichier DashboardController.php => fonction configureMenuItems
    - ajouter les champs : (si j’ai le temps, customiser les logos en modifiant ‘fas fa-list’ par une une valeur sur le site font awesome)
      * yield MenuItem ::linkToCrud(‘Users, ‘fas fa-list’, Users::class);
      * yield MenuItem ::linkToCrud(‘Cars, ‘fas fa-list’, Cars::class);
      * yield MenuItem ::linkToCrud(‘Opinions, ‘fas fa-list’, Opinions::class);
      * yield MenuItem ::linkToCrud(‘Contacts, ‘fas fa-list’, Contacts::class);
      * yield MenuItem ::linkToCrud(‘Opening\_hours, ‘fas fa-list’, Opening\_hours::class);
      * yield MenuItem ::linkToCrud(‘Services, ‘fas fa-list’, Services::class);
    - après le namespace, vérifier la presence de :
      * use App\Entity\Users
      * use App\Entity\Cars
      * use App\Entity\Opinions
      * use App\Entity\Contacts
      * use App\Entity\Opening\_hours
      * use App\Entity\Services
    - Si relation entre tables : ajouter la fonction configureFields dans le fichier MovieCrudController.php (si problème avec l’entité Movie)

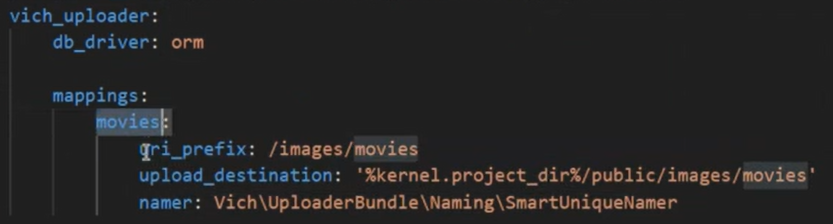


* Dans le OpinionsCrudController.php
  + public function configureFields(string $pageName): iterable
  + {
    - yield from parent::configureFields($pageName);
    - yield ChoiceField::new(‘grade)->setChoices([
      * ‘1’ => ‘1’,
      * ‘2’ => ‘2’,
      * ‘3’ => ‘3’,
      * ‘4’ => ‘4’,
      * ‘5’ => ‘5’,
    - ])->renderExpanded();
  + }
* Dans le UsersCrudController.php
  + Faire la modification indiquée dans <https://dev.to/nabbisen/symfony-6-and-easyadmin-4-hashing-password-3eec>
  + Attention : remplacer User par Users
  + Rajouter les roles : ROLE\_ADMIN, ROLE\_EMPLOYE, ROLE\_USER

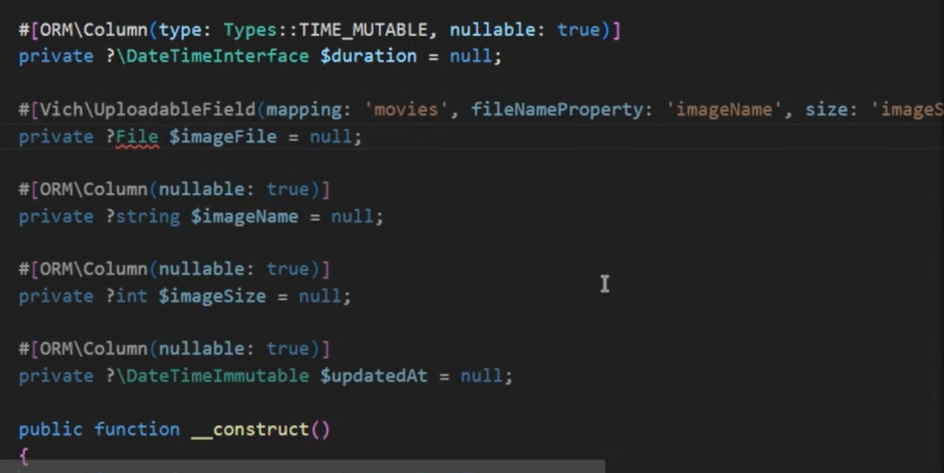




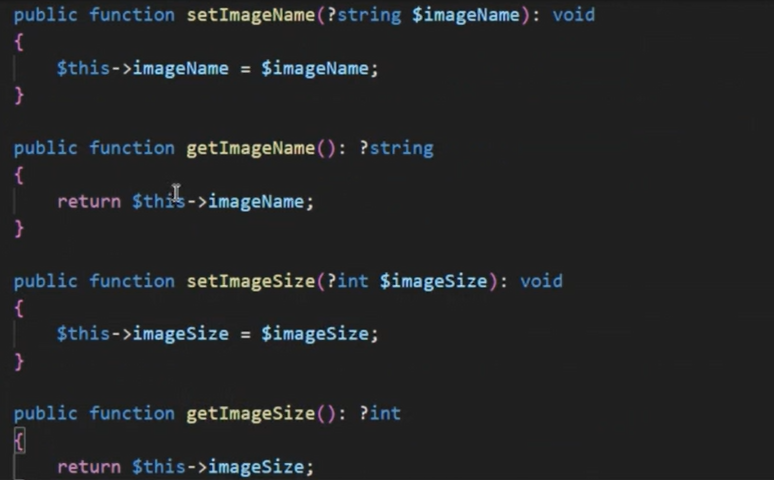
* Uploader un fichier avec le bundle suivant (<https://github.com/dustin10/VichUploaderBundle/blob/master/docs/index.md> )
  + Composer require vich/uploader-bundle
    - Yes execute this recipe
  + Dans public, créer un dossier images et y mettre toutes les images
  + Dans le fichier config => packages => vich\_uploader.yaml
    - Indiquer où se trouve les images (à partir du dossier public)
      * Cars :
        + uri\_prefix : /images/vente/cars
        + upload\_destination : ‘%kernel.project\_dir%/public/images/vente/cars’



* + Dans le fichier src => entity => Cars.php
    - Sous le namespace
    - use Symfony\Component\HttpFoundation\File\File;
    - use Vich\UploaderBundle\Mapping\Annotation as Vich;
  + Sous #{ORM\Entity,…
    - #[Vich\Uploadable]
  + Sous #[ORM\Column(…
    - #[Vich\UploadableField(mapping: cars, fileNameProperty: 'imageName', size: 'imageSize')]
    - private ?File $imageFile = null;
    - #[ORM\Column(nullable: true)]
    - private ?string $imageName = null;
    - #[ORM\Column(nullable: true)]
    - private ?int $imageSize = null;
    - #[ORM\Column(nullable: true)]
    - private ?\DateTimeImmutable $updatedAt = null;



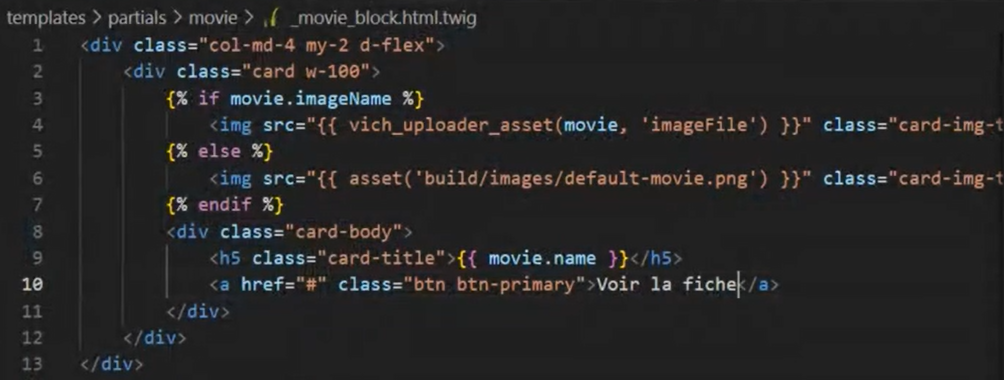
* Après les fonctions du fichier, coller toutes les fonctions de <https://github.com/dustin10/VichUploaderBundle/blob/master/docs/usage.md>



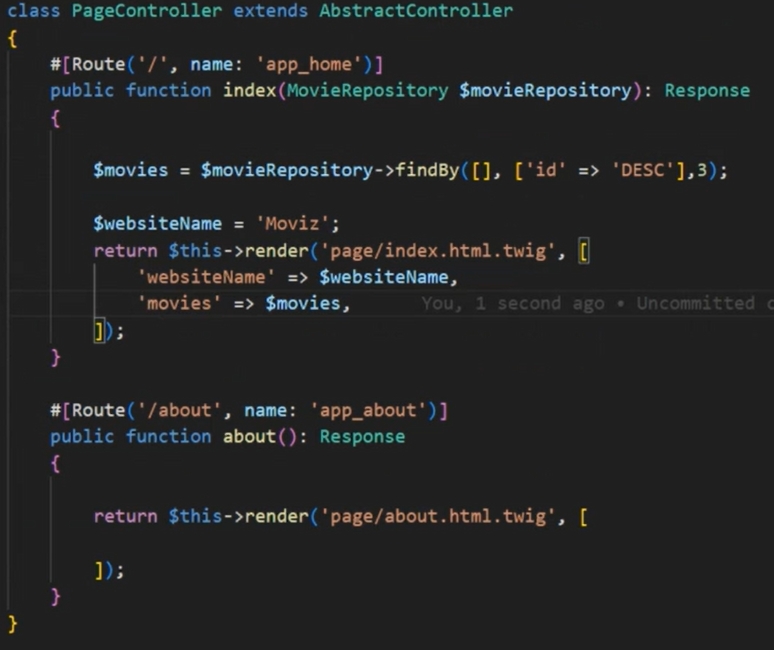
* + Dans le fichier src => admin => CarsCrudController dans la fonction configureFields
    - yield TextareaField ::new’imageFile’)->setFormType(VichImageType::class)->hideOnIndex();
    - yield ImageField::new(‘imageName’)->setBasePath(‘images/vente/cars/’)->hideOnForm();



* + Faire la migration
    - symfony console make:migration
    - symfony console doctrine:migrations:migrate
* Partie Front
  + symfony console make :controller
    - CarsController
  + Dans le fichier src =>Controller => CarsController.php => fonction index
    - Public function index(CarsRepository $carsRepository): Response
    - {
      * $cars = $carsRepository->findBy([],[‘id’ => ‘DESC’]);
      * return $this->render(‘cars/index.html.twig’, [‘cars’ => $cars, ]);
* }
  + Dans le fichier templates => car => index.html.twig
    - Faire une boucle for pour afficher la liste des voitures
      * Dans le container
        + {% for car in cars %}
    - Afficher car.car\_title, car.car\_image, car.car\_year, car.car\_fuel, car.car\_kilometer, car.car\_price.



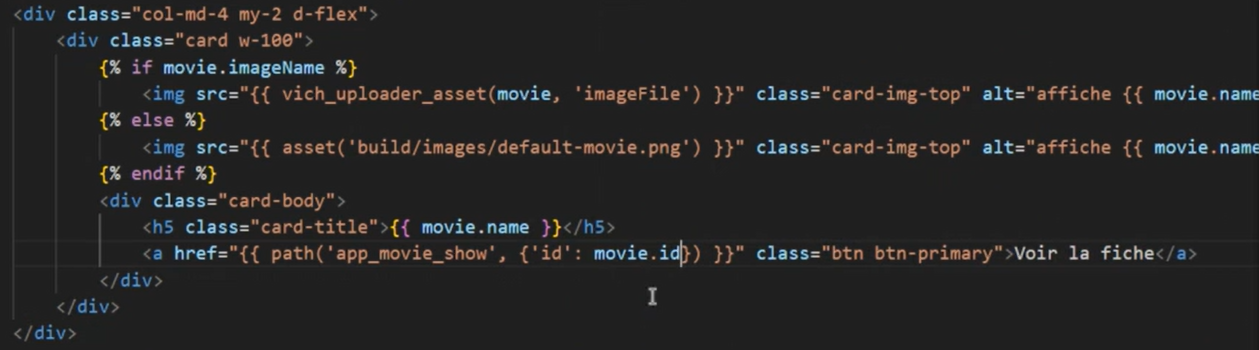
* + - * + {% endblock %}
  + Dans le fichier src => controller => PageController.php
    - On indique une limite pour le nombre de répétition de la boucle : ici 3 films



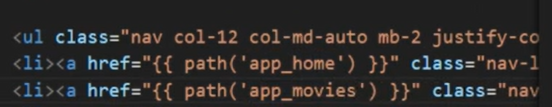
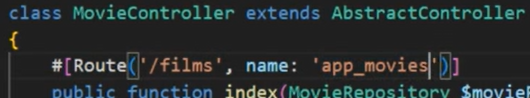
* + Dans le fichier templates => page => index.html.twig



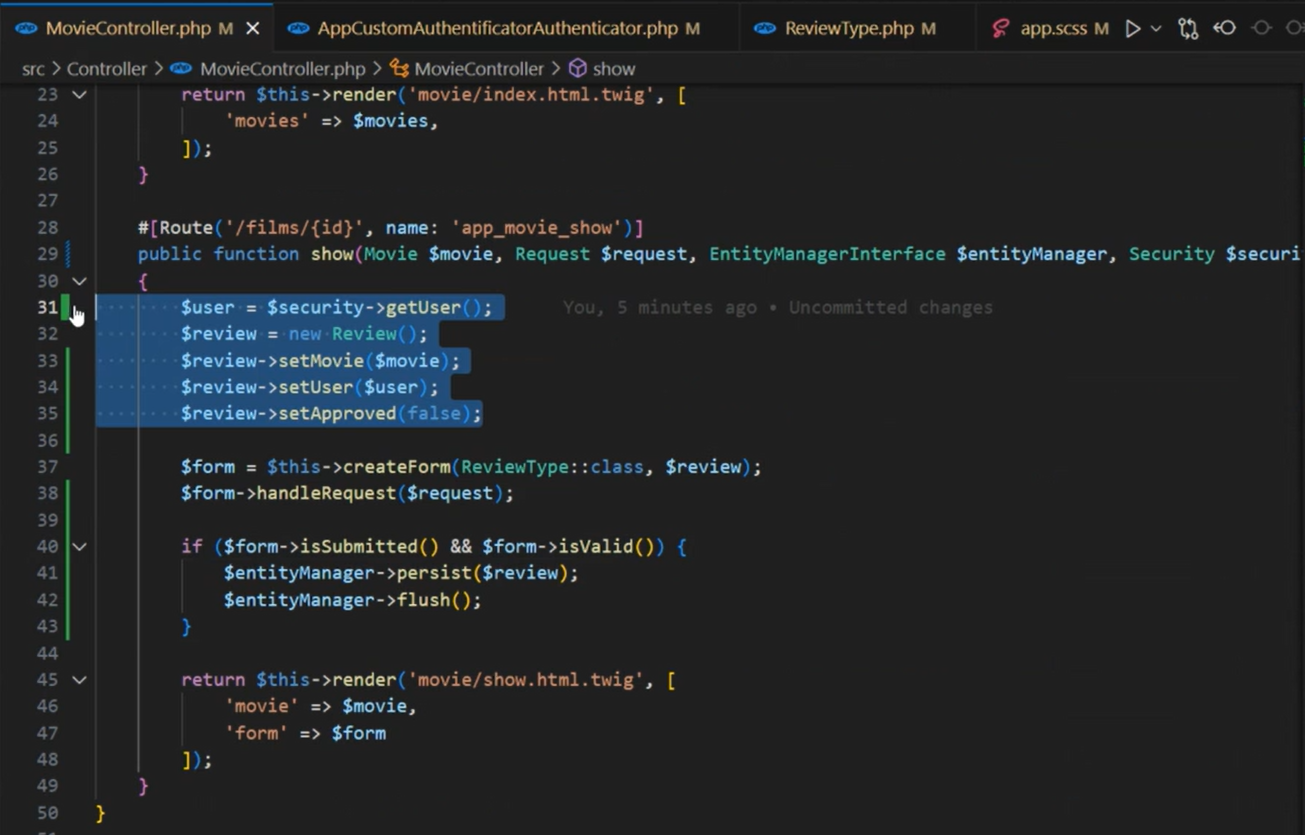
* + Pour faire des liens, écrire dans le href
    - href=″{{ path(‘chemin\_a\_afficher’, {‘parametre’: parameter}) }}″



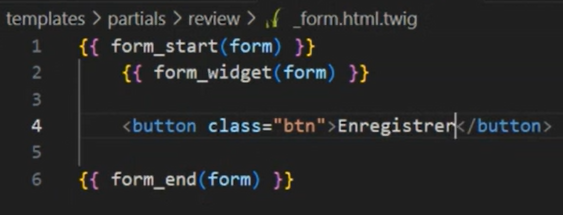
* + Même principe dans les liens de la navigation de la page d’accueil (\_header.html.twig)
    - On met le même path que le name de la route du controller



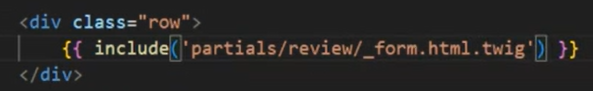
* Générer un form type <https://symfony.com/doc/current/reference/forms/types.html>
  + Dans le live, fichiers
    - Show.html.twig
    - MovieController.php
  + Création du formulaire
    - symfony console make:form
      * OpinionType (nom de la form)
      * Opinions (nom de l’entity)
    - Dans le fichier src => controller => OpinionsController.php
      * $rate = new Rate();
      * $form = $this->createForm(OpinionType :class, $rate);
      * Return $this->render(‘opinion.html.twig’, [
        + ‘opinion’ => $opinion,
        + ‘form’ => $form,
      * ]);



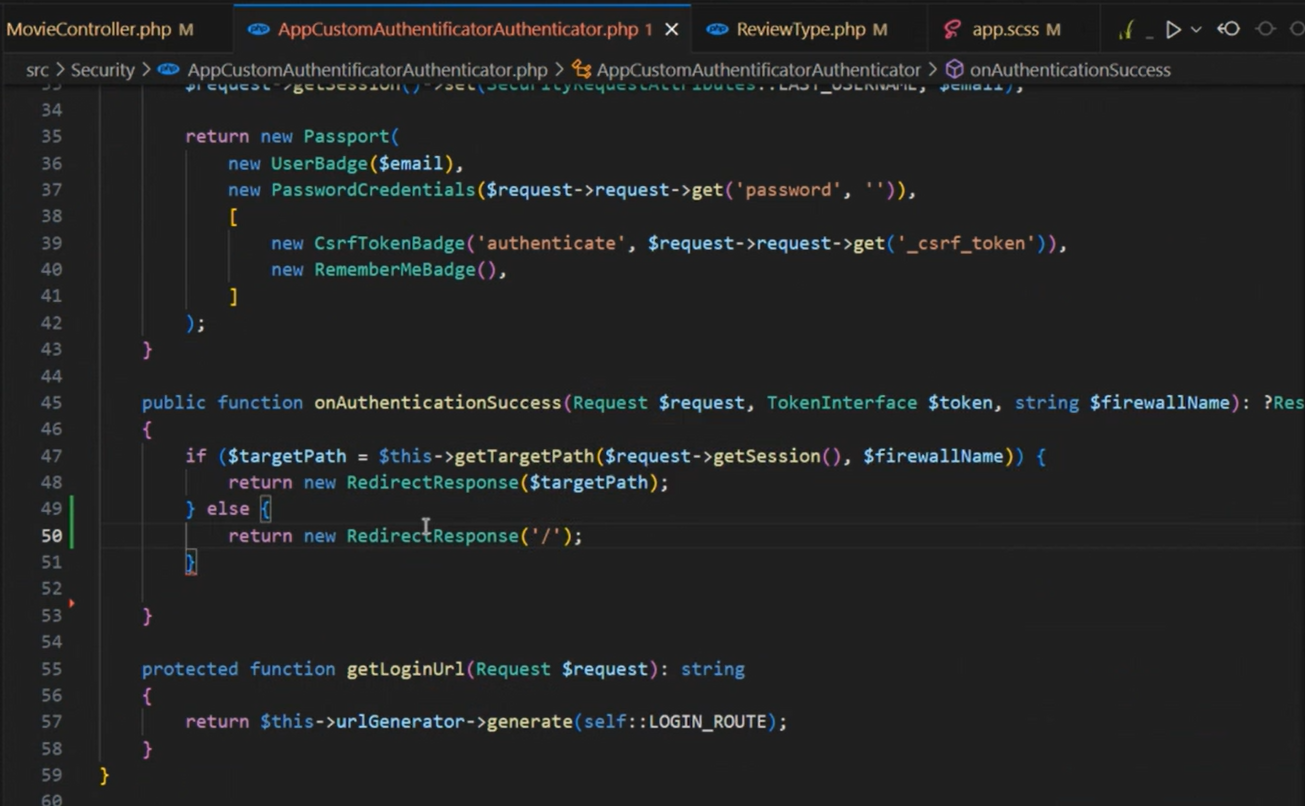
* + Affichage du formulaire
    - Créer un fichier templates => partials => opinions => \_form.html.twig
      * (form\_widget pour mettre tous les champs ou form\_label pour chacun des champs)



* + - Dans le fichier qui doit afficher le formulaire on va chercher le partial



Connexion



* Gestion des erreurs

