ZINES

\* Créer un projet Symfony 6 nommé « Zines » \* Créer une table SQL au nom de « magazine » avec les champs suivants : \* id, name, price, created\_at \* Insérer quelques données de tests grâce à une fixture \* Sur la page d’accueil, afficher la totalité des données de cette table SQL, mais seulement les noms : \* Chacune des données doit posséder un lien « Voir plus de détails… » \* Ce lien redirige vers une route nommée « /magazine/{id} » : \* Afficher les détails du magazine selon l’ID transféré

symfony new --webapp zines

composer require symfony/apache-pack

symfony console make:controller HomeController

.env ligne 30

DATABASE\_URL="mysql://root@127.0.0.1:3306/magazine?serverVersion=5.7&charset=utf8mb4"

*symfony console doctrine:database:create*

cmd :

symfony console make:migration

symfony console doctrine:migrations:migrate

si il y a les modification qui est fait, il faut faire un update

* symfony console doctrine:schema:update –force

install un package

* composer require orm-fixtures –dev (deux tirets pas un)

et puis suprimer AppFixtures

Generer un fixture par CMD

symfony console make:fixtures MagazineFixtures

MagazineFixtures

class MagazineFixtures extends Fixture

{

    public function load(ObjectManager $manager): void

    {

        // Initialisation de Faker

        $faker = Faker\Factory::create();

        for ($i = 0; $i < 10; $i++) {

            // Date aléatoire

            $date = new DateTimeImmutable();

            $randDate = $date->modify('-'. rand(1, 600) .' days');

            // $randDate = $date->modify('-12 days');

            $magazine = new Magazine();

            $magazine->setName($faker->name);

            $magazine->setPrice($faker->numberBetween(6, 30));

            $magazine->setCreatedAt(DateTimeImmutable::createFromMutable($faker->dateTimeBetween('-10 years')));

            $manager->persist($magazine);

        }

        $manager->flush();

    }

}

* symfony console doctrine:fixtures:load

installer faker

composer require fzaninotto/faker

class HomeController extends AbstractController

{

    #[Route('/home', name: 'app\_home')]

    public function index(MagazineRepository $magazineRepository): Response

    {

        return $this->render('home/index.html.twig', [

            'magazines' => $magazineRepository->findAll()

        ]);

    }

    #[Route('/magazine/{id}', name: 'details\_magazine', requirements:['id' => '\d+'])]

    public function details(int $id, Magazine $magazine): Response

    {

        return $this->render('home/details.html.twig', [

            'magazine' => $magazine

        ]);

    }

}

\* Créer une entité « Category » contenant les champs suivants : id, name, color ; \* Relier l’entité « Magazine » à celle-ci ; \* Remplir de nouveau les tables grâce aux fixtures ; \* Dans le page affichant les détails d’un magazine, afficher le nom de la catégorie et mettre le texte en couleur grâce à la donnée enregistrée dans la propriété « color ».

symfony console make:entity Category

created: src/Entity/Category.php

created: src/Repository/CategoryRepository.php

Entity generated! Now let's add some fields!

You can always add more fields later manually or by re-running this command.

New property name (press <return> to stop adding fields):

> name

Field type (enter ? to see all types) [string]:

>

Field length [255]:

> 50

Can this field be null in the database (nullable) (yes/no) [no]:

> n

updated: src/Entity/Category.php

Add another property? Enter the property name (or press <return> to stop adding fields):

> color

Field type (enter ? to see all types) [string]:

>

Field length [255]:

> 20

> n

updated: src/Entity/Category.php

Add another property? Enter the property name (or press <return> to stop adding fields):

>

Next: When you're ready, create a migration with php bin/console make:migration

PS C:\laragon\www\zines> symfony console doctrine:schema:update --force

Updating database schema...

2 queries were executed

[OK] Database schema updated successfully!

PS C:\laragon\www\zines> symfony console make:entity Magazine

Your entity already exists! So let's add some new fields!

New property name (press <return> to stop adding fields):

> category

Field type (enter ? to see all types) [string]:

> relation

relation

on

What class should this entity be related to?:

> Category

Category

ry

What type of relationship is this?

------------ ------------------------------------------------------------------------

Type Description

------------ ------------------------------------------------------------------------

ManyToOne Each Magazine relates to (has) one Category.

Each Category can relate to (can have) many Magazine objects

OneToMany Each Magazine can relate to (can have) many Category objects.

Each Category relates to (has) one Magazine

ManyToMany Each Magazine can relate to (can have) many Category objects.

Each Category can also relate to (can also have) many Magazine objects

OneToOne Each Magazine relates to (has) exactly one Category.

Each Category also relates to (has) exactly one Magazine.

------------ ------------------------------------------------------------------------

Relation type? [ManyToOne, OneToMany, ManyToMany, OneToOne]:

> ManyToOne

ManyToOne

A new property will also be added to the Category class so that you can access and set the related Magazine object from it.

New field name inside Category [magazine]:

>

Is the Category.magazine property allowed to be null (nullable)? (yes/no) [yes]:

> n

Do you want to activate orphanRemoval on your relationship?

A Category is "orphaned" when it is removed from its related Magazine.

e.g. $magazine->removeCategory($category)

NOTE: If a Category may \*change\* from one Magazine to another, answer "no".

Do you want to automatically delete orphaned App\Entity\Category objects (orphanRemoval)? (yes/no) [no]:

updated: src/Entity/Magazine.php

updated: src/Entity/Category.php

Add another property? Enter the property name (or press <return> to stop adding fields):

>

Success!

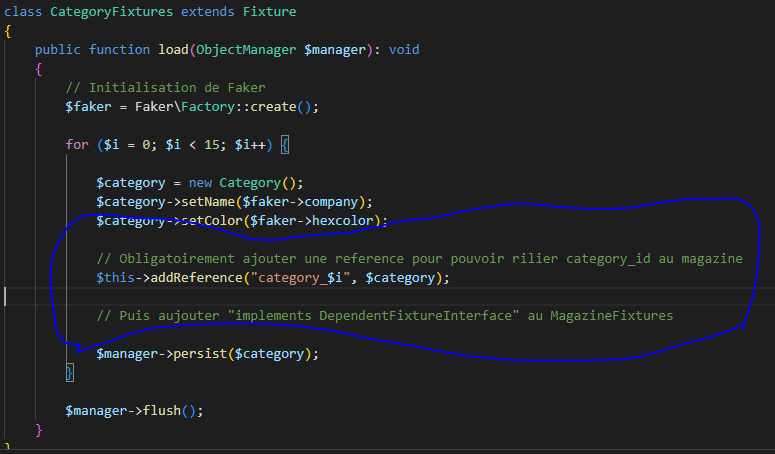
Next: When you're ready, create a migration with php bin/console make:migration

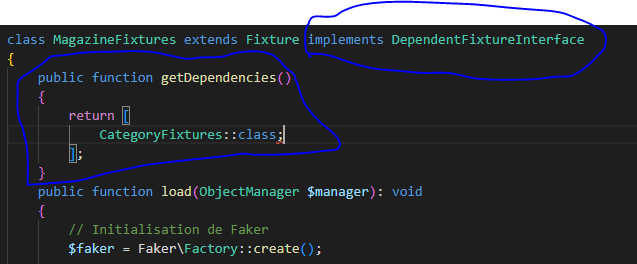
PS C:\laragon\www\zines> symfony console doctrine:schema:update --force

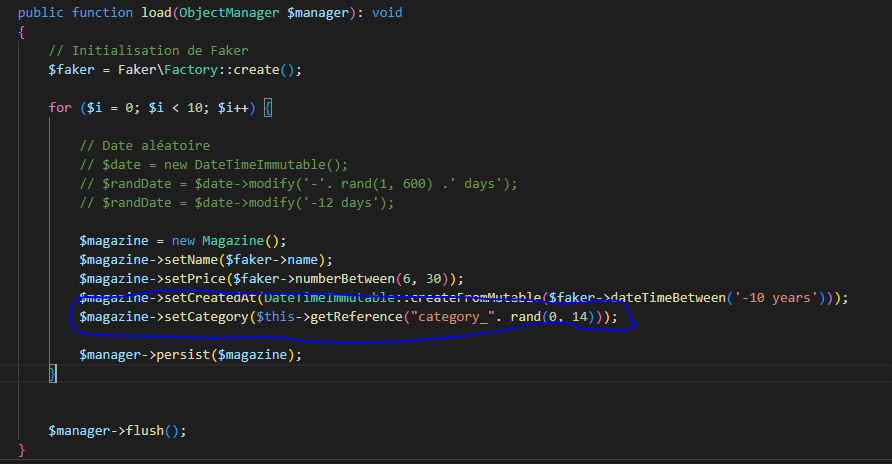
Updating database schema...

3 queries were executed

[OK] Database schema updated successfully!

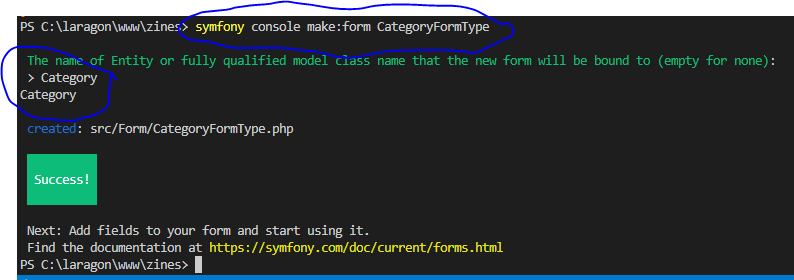




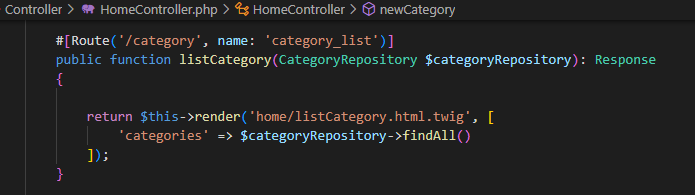


Retour au projet « Zines » : \* Créer une nouvelle page avec pour route : « /category », dans laquelle vous afficher la totalité des catégories avec deux liens pour chaque catégorie : « Editer » et « Supprimer »; \* Sur la page qui vient d’être créée, ajouter un bouton « Nouvelle catégorie » redirigeant vers une nouvelle page dont la route est « /category/new »; \* Générer un formulaire permettant d’ajouter de nouvelles catégories. Ne pas oublier les contraintes de validation; \* Revenir sur la page d’accueil du projet et ajouter un bouton « Nouveau magazine », redirigeant vers une page dont la route est « /magazine/new »; \* Générer un formulaire permettant d’ajouter de nouveaux magazines. À la validation du formulaire, si le magazine a bien été enregistré, afficher un message de succès, non pas sur le formulaire, mais sur la page d’accueil.

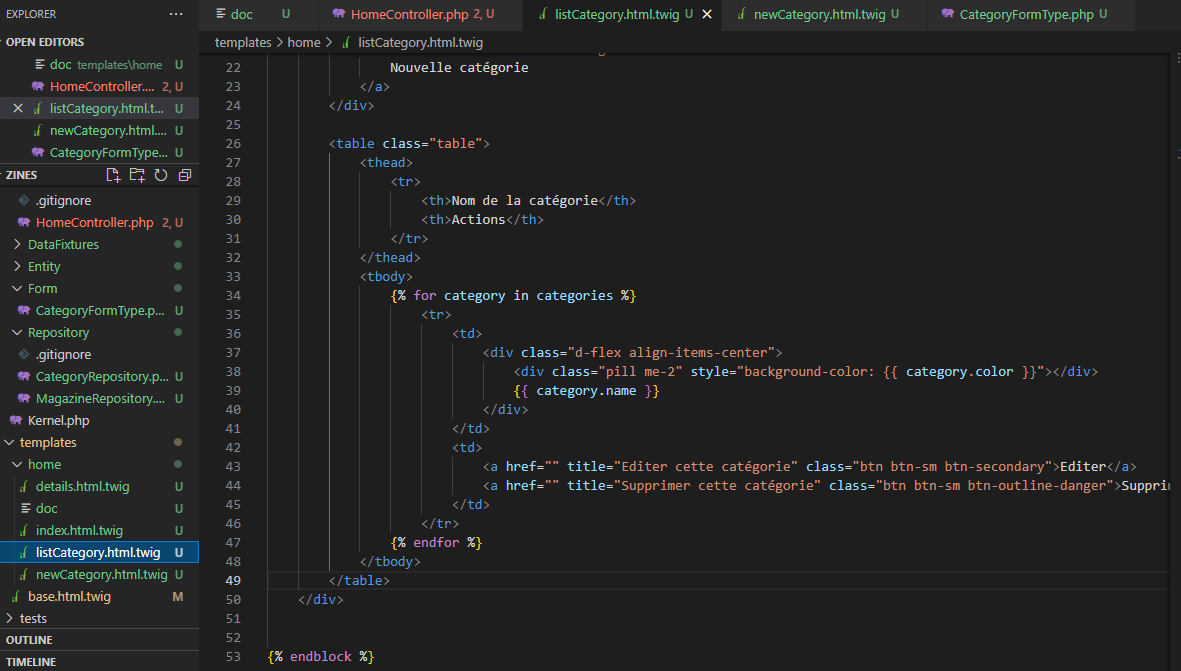
Formulaire Category



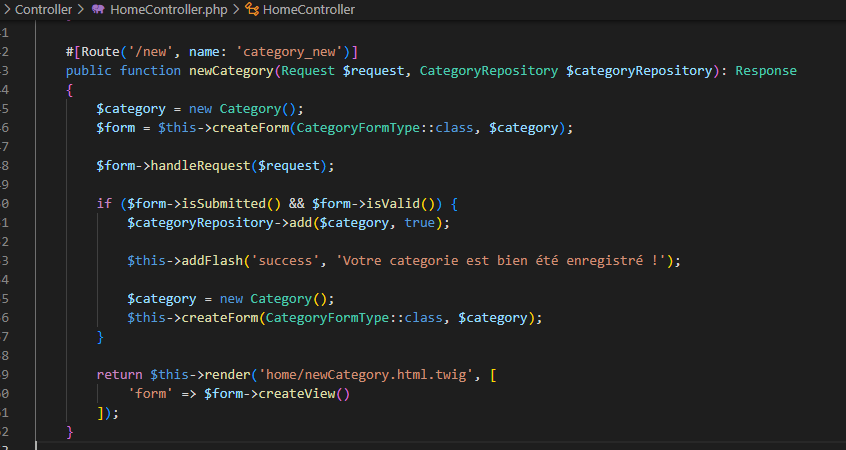
Création d’une route sur le controller



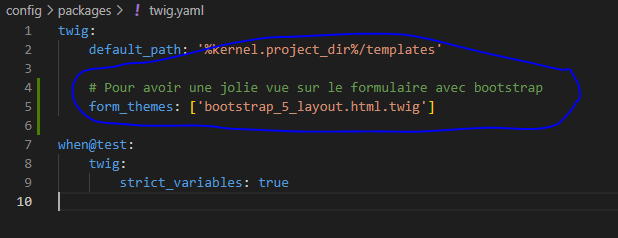
Création d’une page a affiché



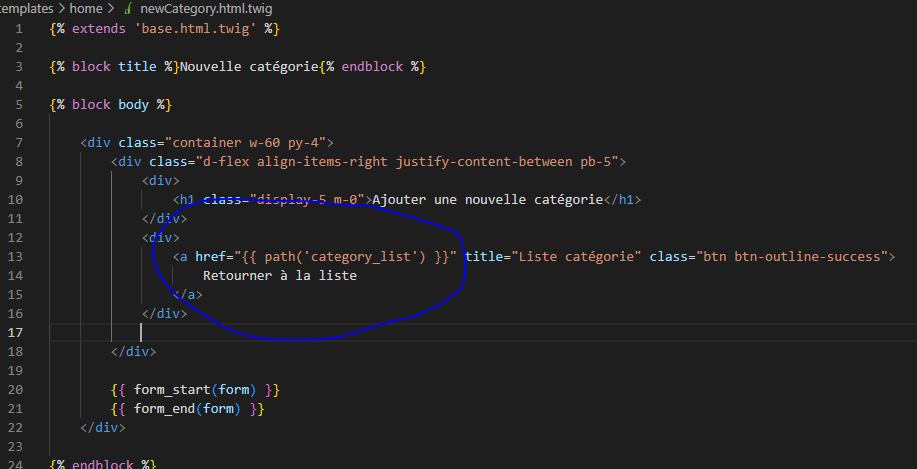
Une nouvelle route dans le controller



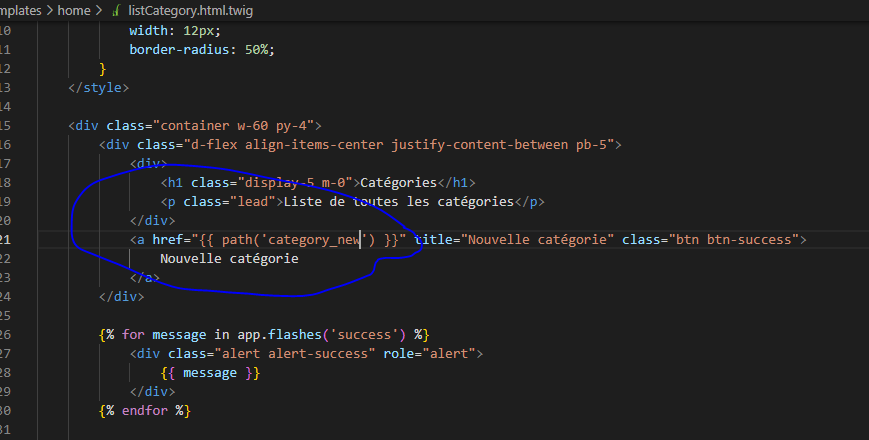
Modifier le twig.yaml



Retourner à la list des categories



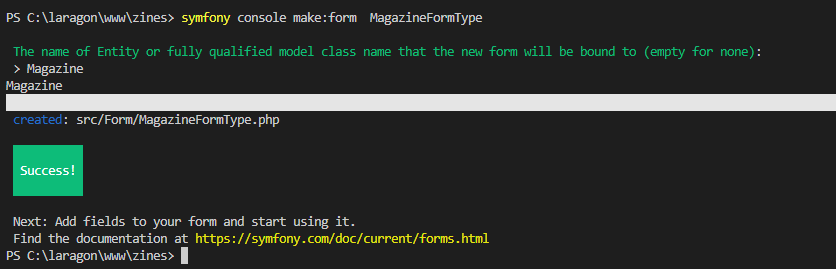
Le bouton ajouter une nouvelle category



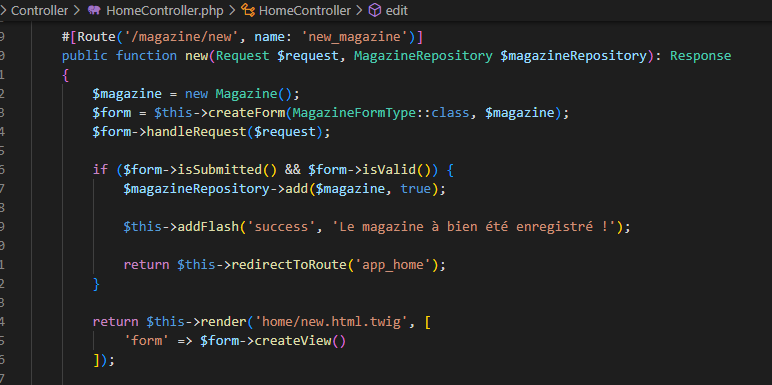
Elasticsearch apply pour recherche sur google (gratuit)

Pour supprimer ou editer

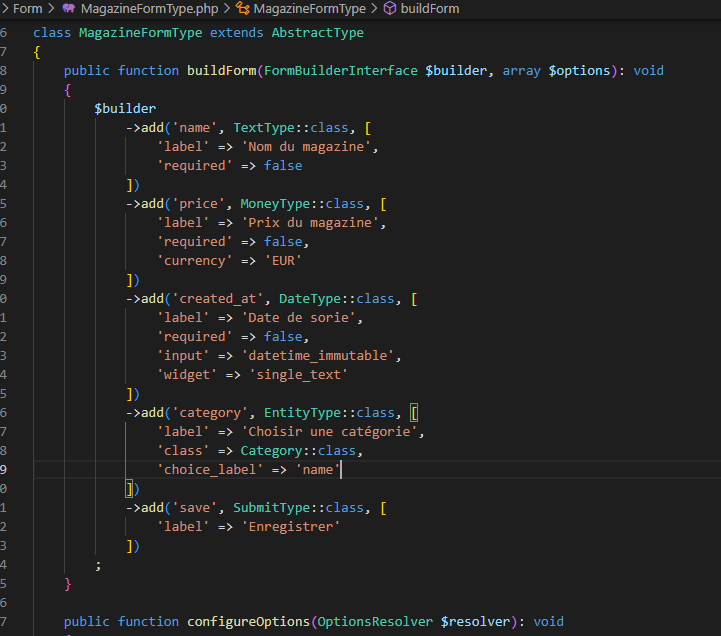
Création d’une form Magazine



Créer une nouvelle route



MagazineFormType

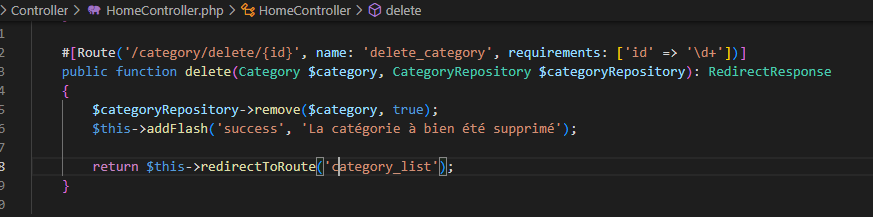


Edit

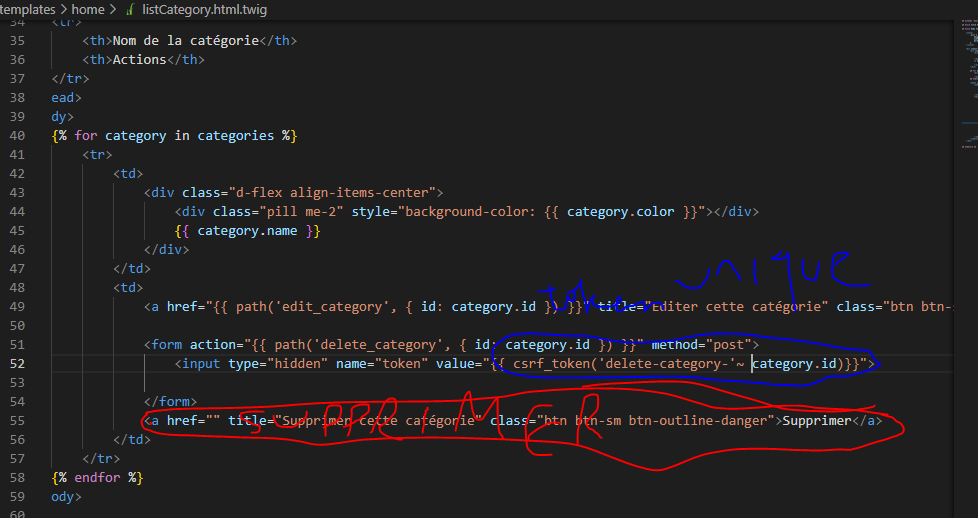


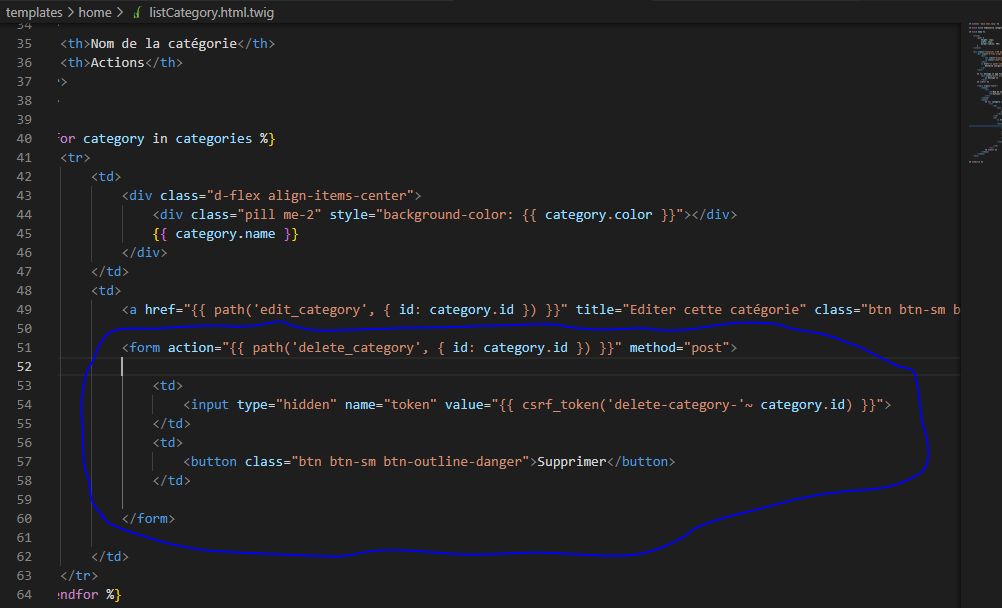
Crée la page edit.html.twig

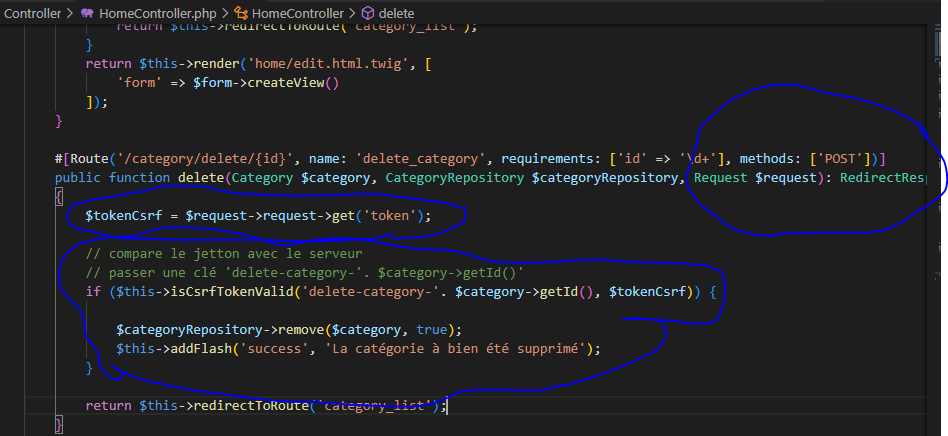
Supprimer



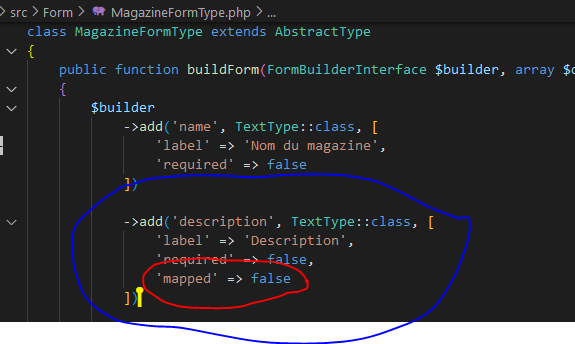
Pour supprimer en securite (pas par un pirate avec un lien) il faut créer un formulaire







Pour ajouter un champ d’un un formulaire si il n’y a pas dans la table BDD

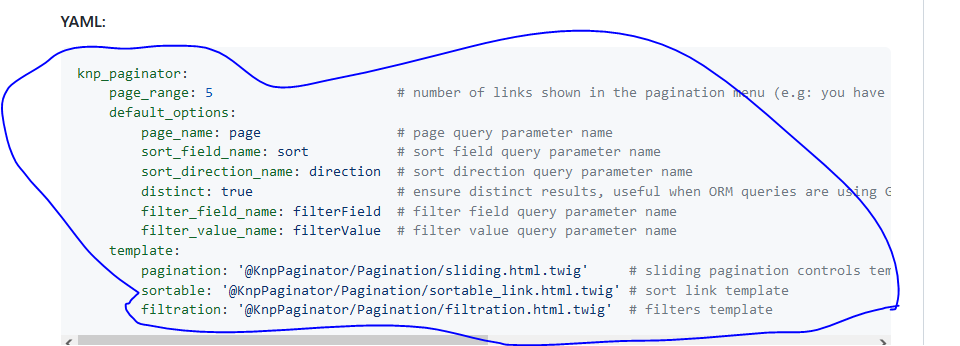
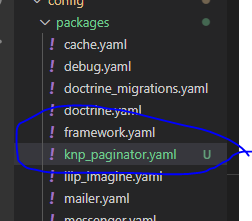


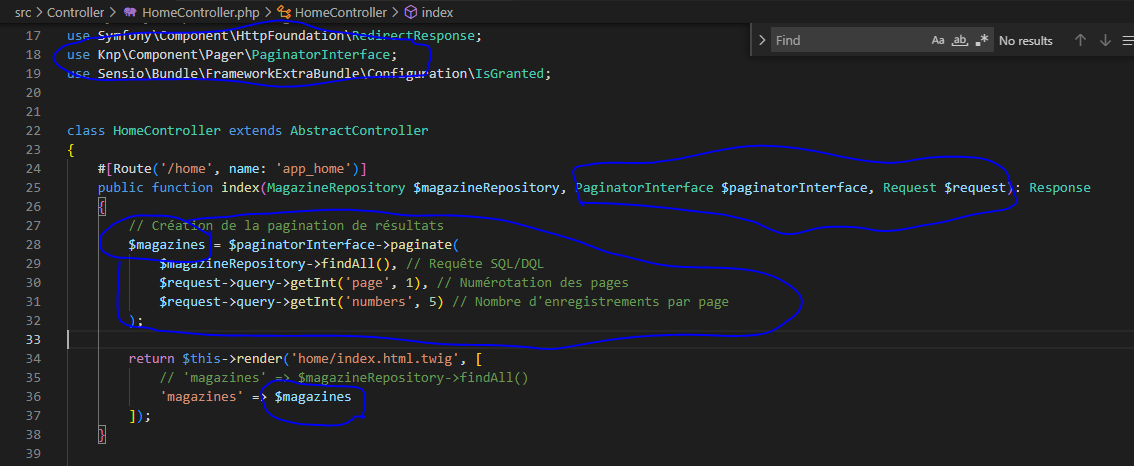
Sur le projet « Zines », ajouter deux nouveaux champs sur l’entité « Magazine » : description et cover.  Installez le bundle « Vichuploader » et modifier le formulaire d’ajout et d’édition des magazines afin de pouvoir ajouter une description et une image. L’image n’est pas obligatoire, mais si celle-ci existe, pensez à l’afficher en miniature dans le tableau des articles ;). Pensez aussi à paginer la liste des magazines et des catégories.

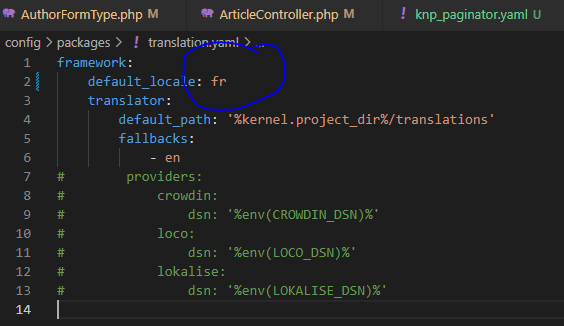
Pagination

composer require knplabs/knp-paginator-bundle

Création la page knp\_paginator.yaml dans le config\package





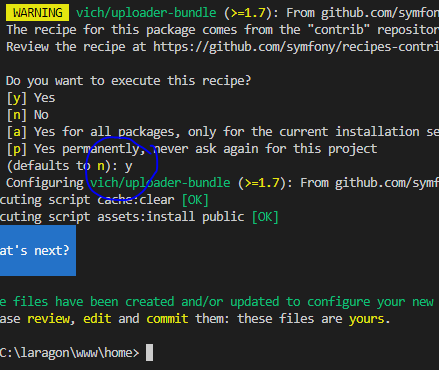


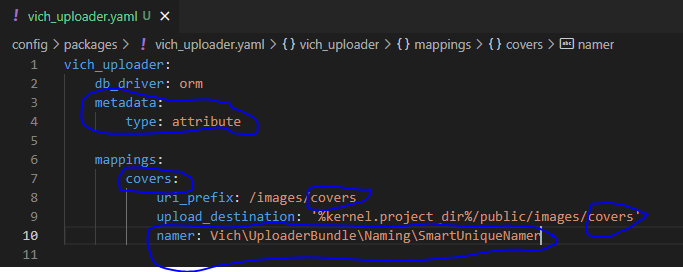
**Upload**

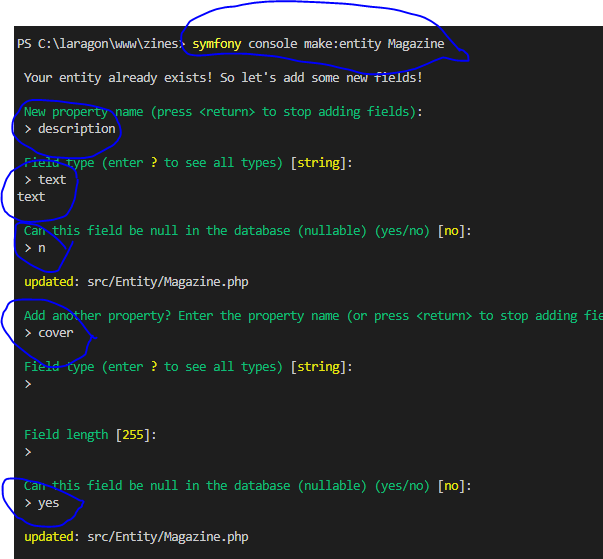
Installer bundle

https://github.com/dustin10/VichUploaderBundle/blob/master/docs/installation.md

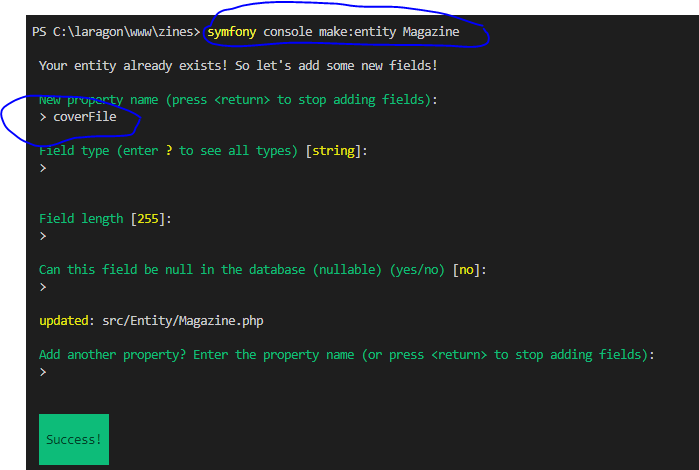
composer require vich/uploader-bundle



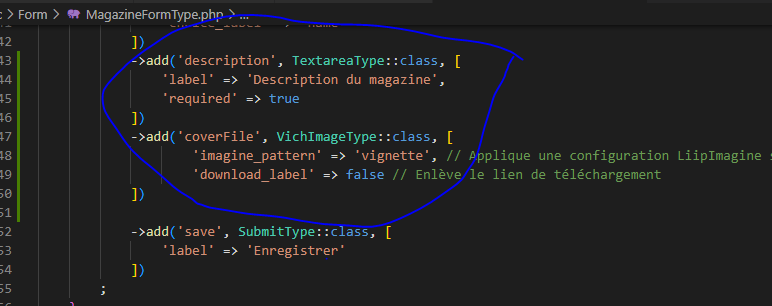




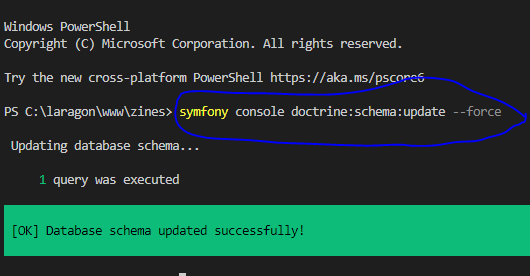
Ajouter un champs pour upload

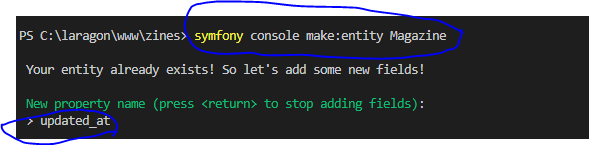


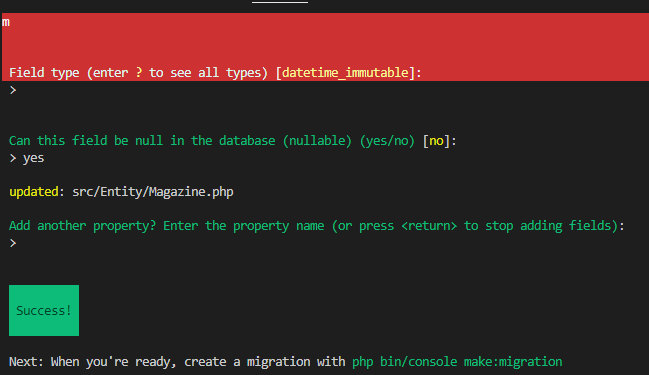
Ajouter au MagazineFormType

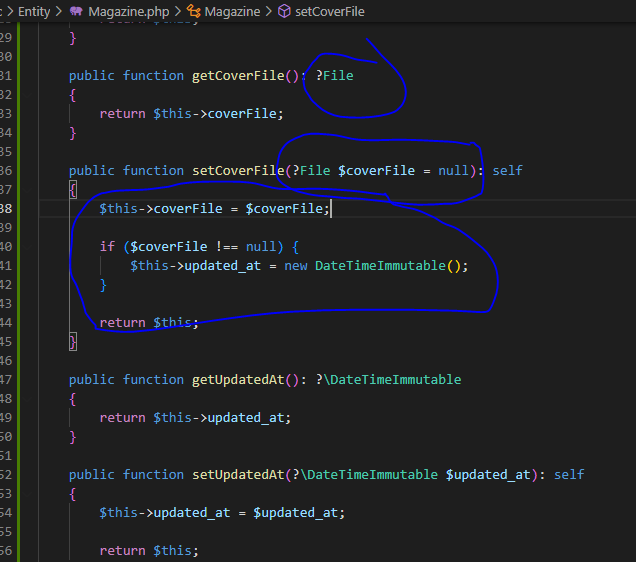


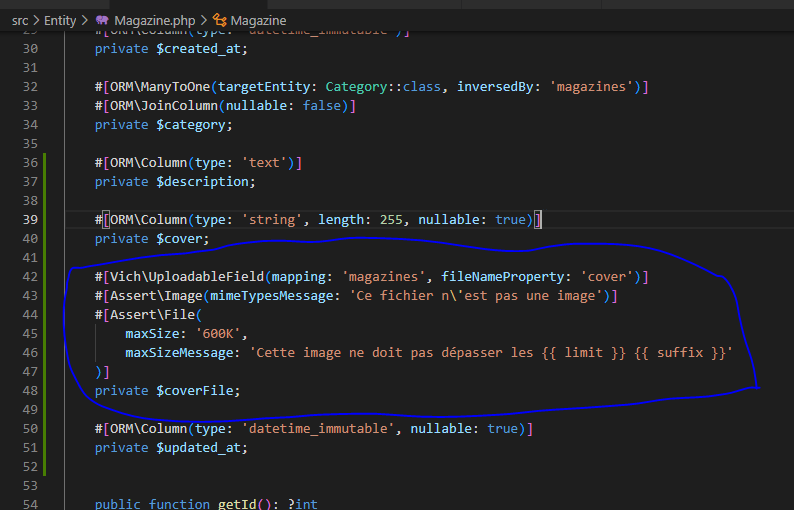
symfony console doctrine:schema:update --force



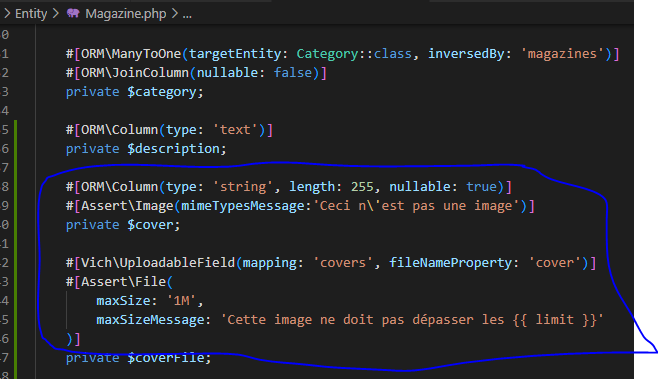


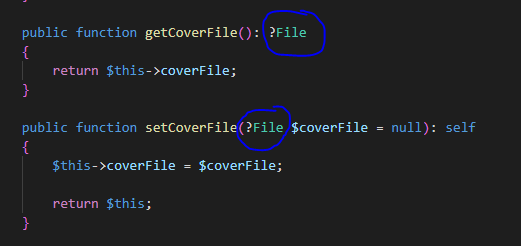




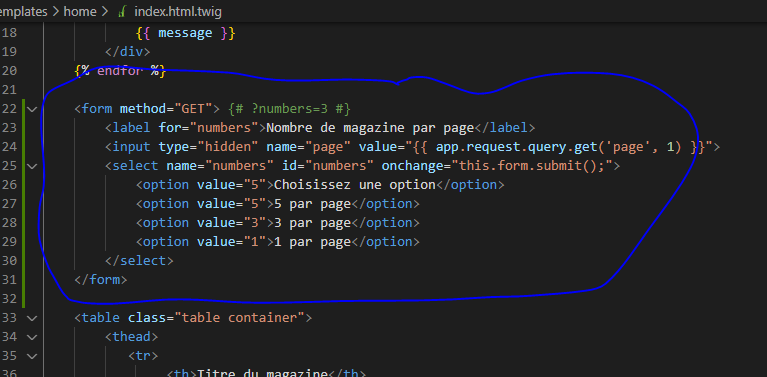


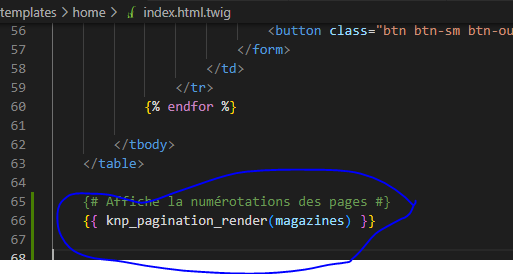
Magazine.php

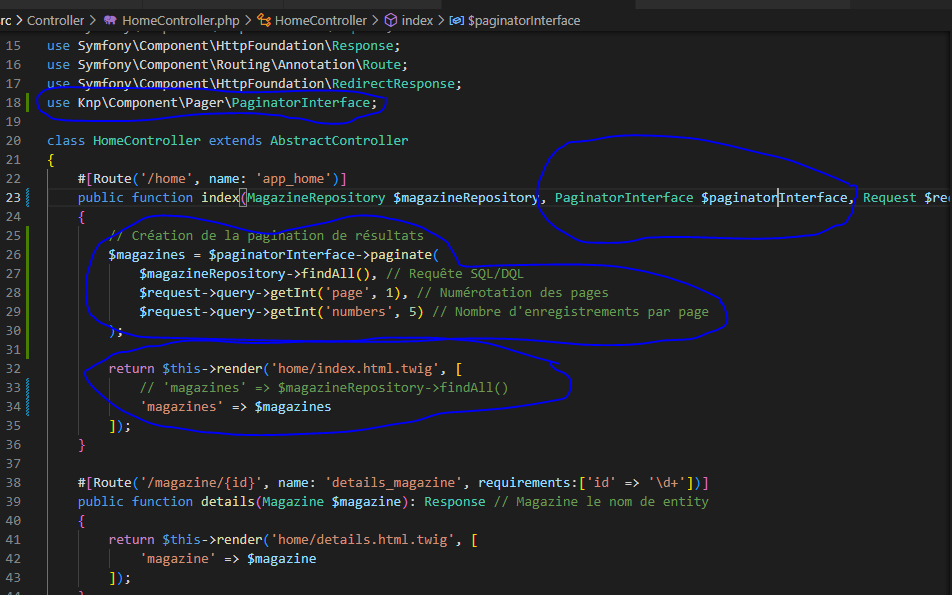




Page index







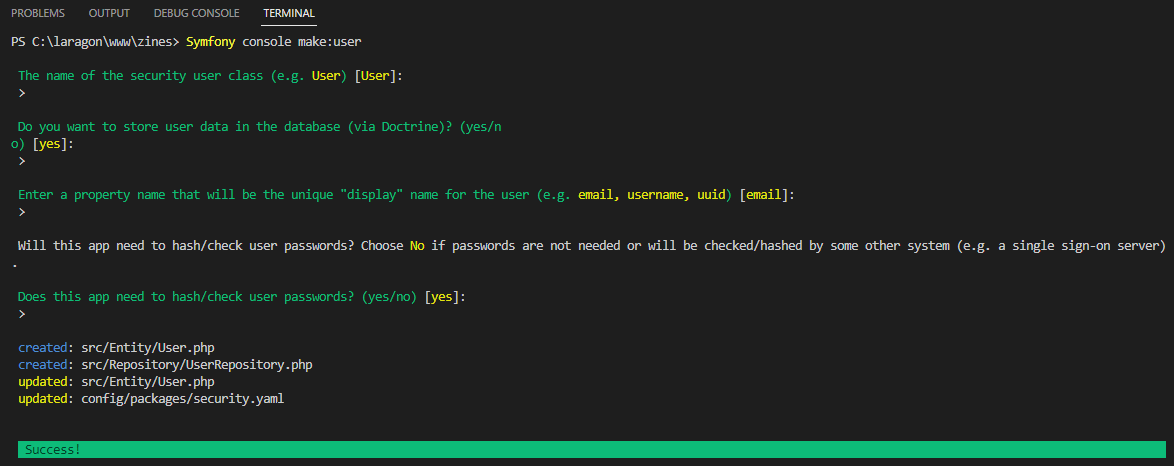
PS C:\laragon\www\zines> composer require liip/imagine-bundle

Repondre yes

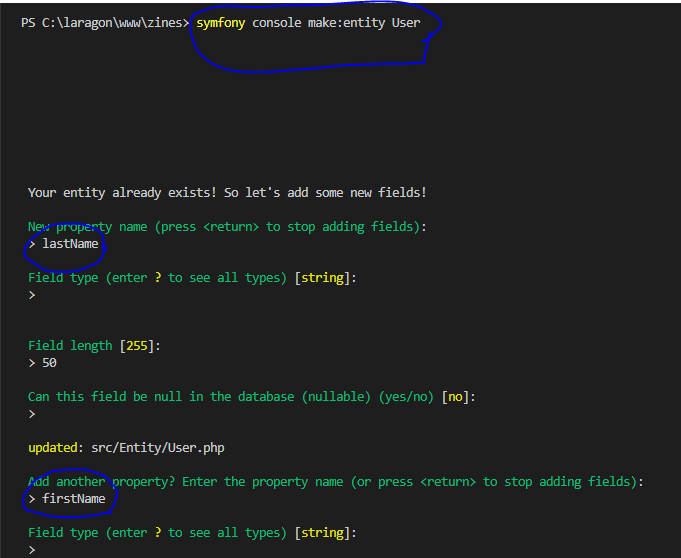
Hello Zines ! Ce projet a besoin de membres pour fonctionner. Pour cela, ils auront besoin de s’inscrire. Le formulaire comportera les champs suivants : nom, prénom, e-mail, mot de passe et confirmation de mot de passe. Une fois celui inscrit, redirigez-le vers la route « /account » que vous protègerais de façon à ce que seul un membre inscrit et connecté puisse y accéder. Sur cette page, afficher le nom, prénom et l’email de l’utilisateur connecté.

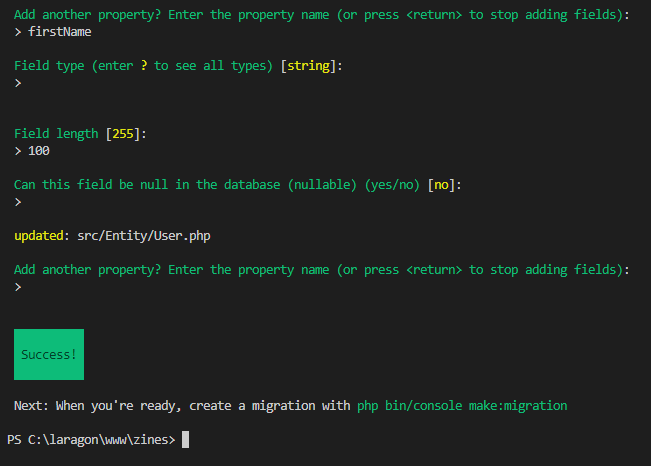
symfony console make:user

Faire entrer pour toute les questions



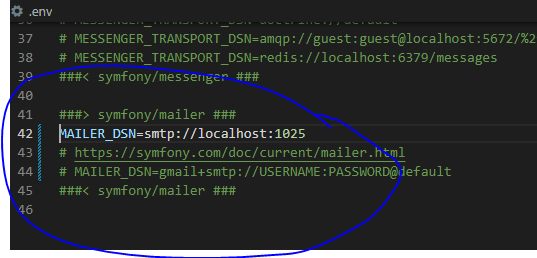
symfony console make:entity User

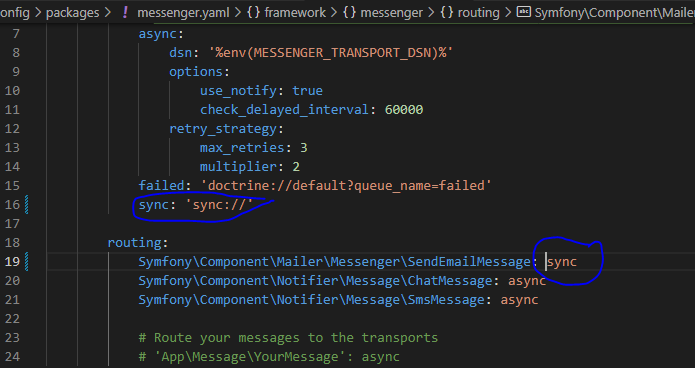


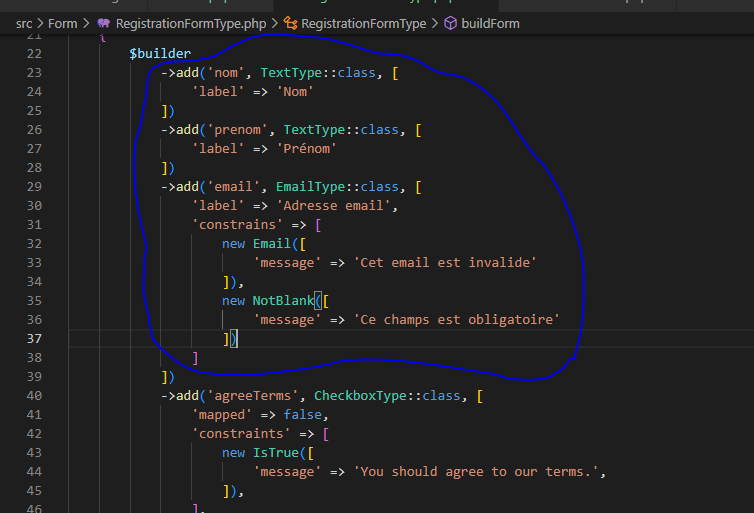


symfony console doctrine:schema:update --force

composer require symfonycasts/verify-email-bundle







'mapped' => false,

                'attr' => ['autocomplete' => 'new-password'],

                'constraints' => [

                    new NotBlank([

                        'message' => 'Please enter a password',

                    ]),

                    new Length([

                        'min' => 6,

                        'minMessage' => 'Your password should be at least {{ limit }} characters',

                        // max length allowed by Symfony for security reasons

                        'max' => 4096,

                    ]),

                ],

           ->add('password', RepeatedType::class, [

                'type' => PasswordType::class,

                'invalid\_message' => 'Mot de pass doit être identique',

                'options' => ['attr' => ['class' => 'password-field', 'autocomplete' => 'new-password']],

                'required' => true,

                'first\_options' => ['label' => 'Mot de pass'],

                'second\_options' => ['label' => 'Confirmer votre mot de pass'],

                'mapped' => false,

                'constraints' => [

                    new NotBlank([

                        'message' => 'Please enter a password',

                    ]),

                    new Length([

                        'min' => 6,

                        'minMessage' => 'Your password should be at least {{ limit }} characters',

                        // max length allowed by Symfony for security reasons

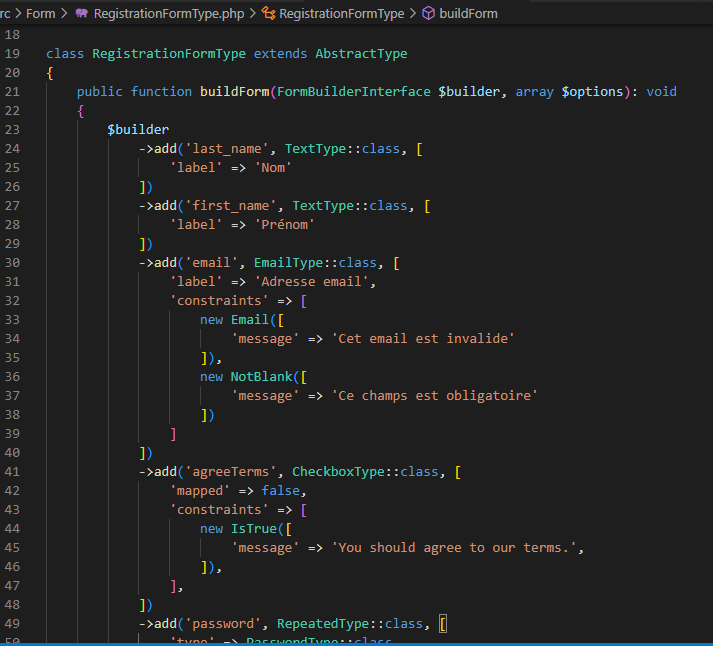
                        'max' => 4096,

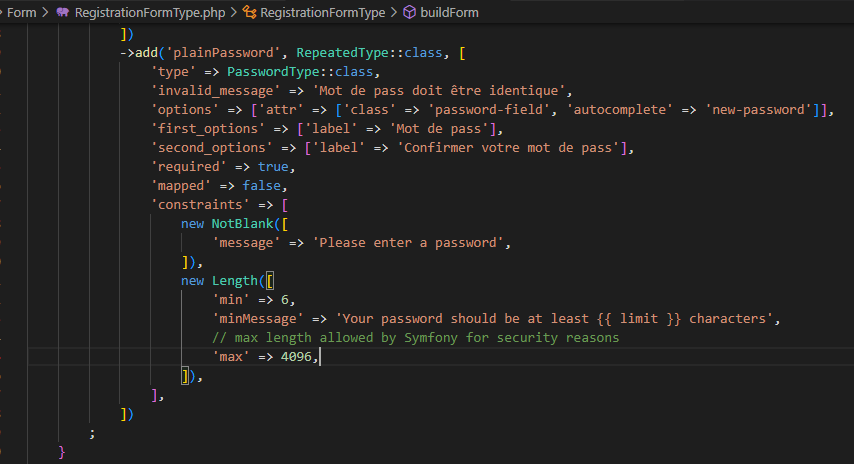
                    ]),

                ],

            ])

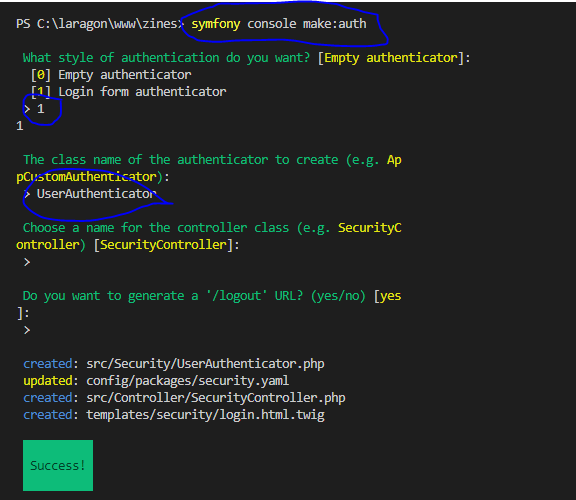






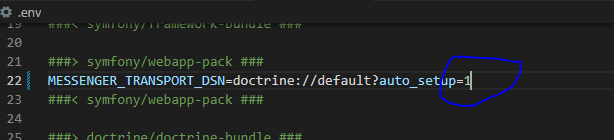
User authenticator

symfony console make:auth

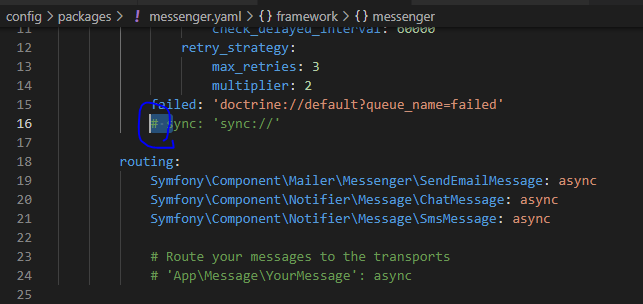


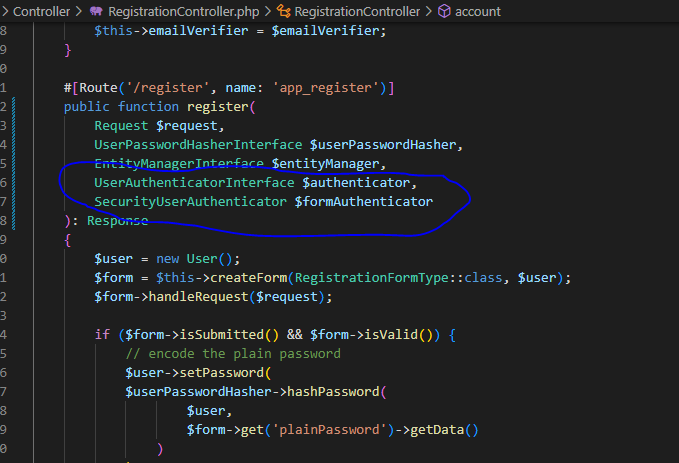
pour eviter les surcharge quand il y a trop d’email

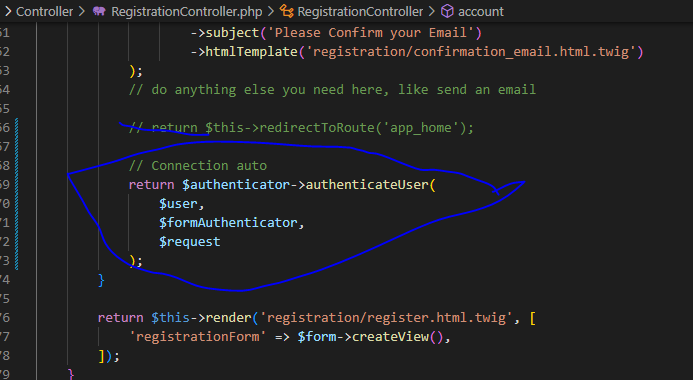
symfony console messenger:consume async -vv

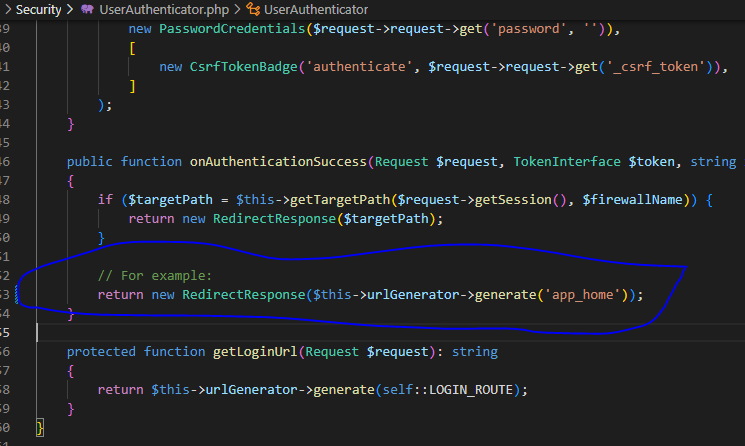


Decommenter ligne 16 messenger.yaml

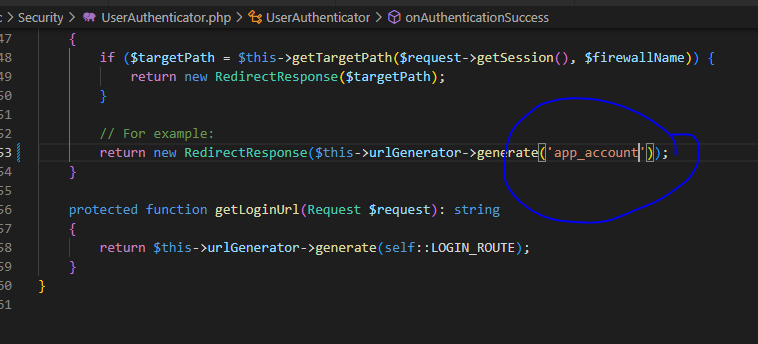


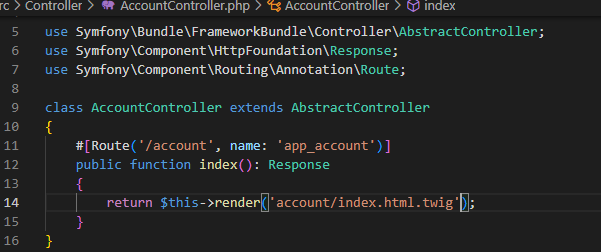




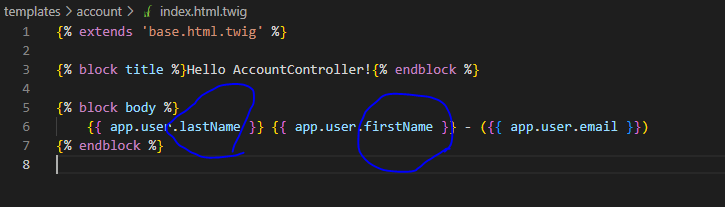


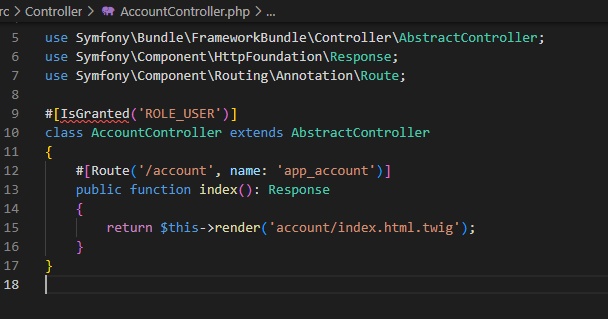
symfony console make:controller AccountController





lastName et firstName est écrire en camelCase car au moment de cration entity en camelCase même si sur la table est écrit avec underscore first\_name et last\_name

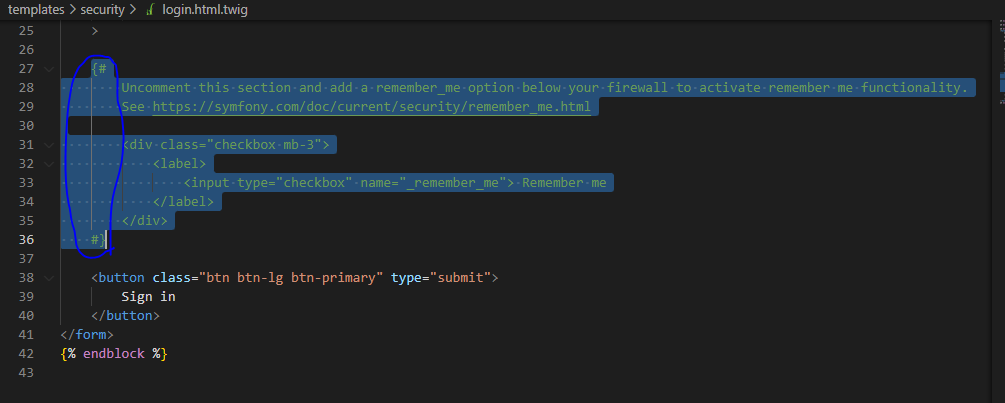


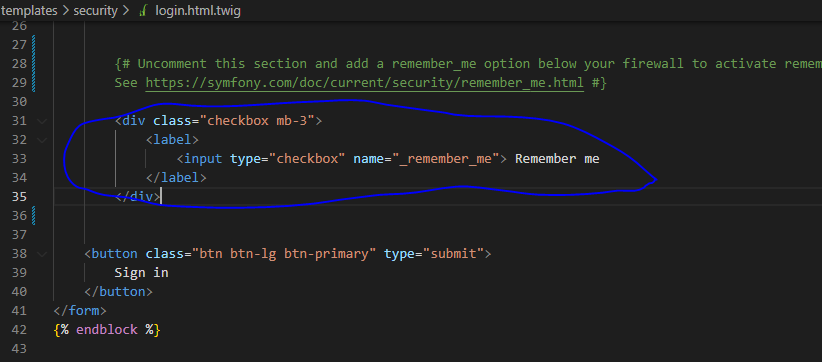


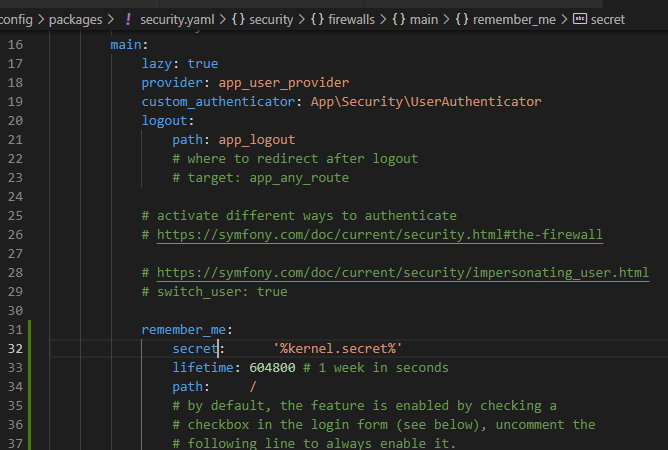
Sur le formulaire de connexion, ajouter un champ afin d'avoir l'option "Se souvenir de moi", pour éviter à l'utilisateur de se reconnecter à chaque visite. Ajouter un formulaire pour réinitialiser son mot de passe en cas de perte. La page "/category" ne doit pas être accessible à un utilisateur non connecté et sur la page d'accueil, les boutons suivants ne doivent pas être affichés à un utilisateur : nouveau magazine, éditer et supprimer.

Remember me

Unlever les commentaire

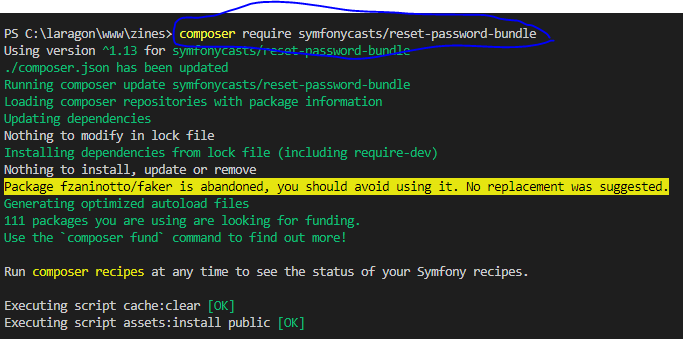






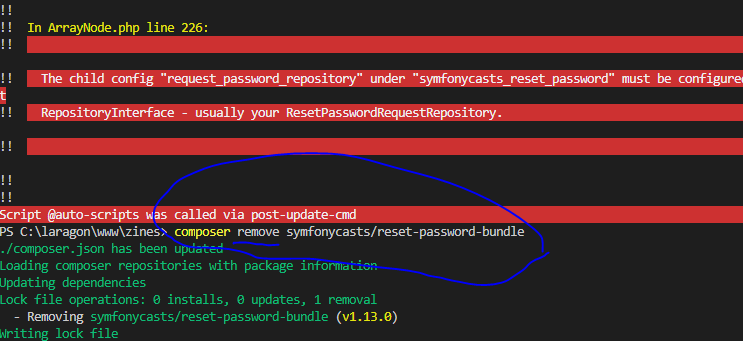
Reset password

composer require symfonycasts/reset-password-bundle



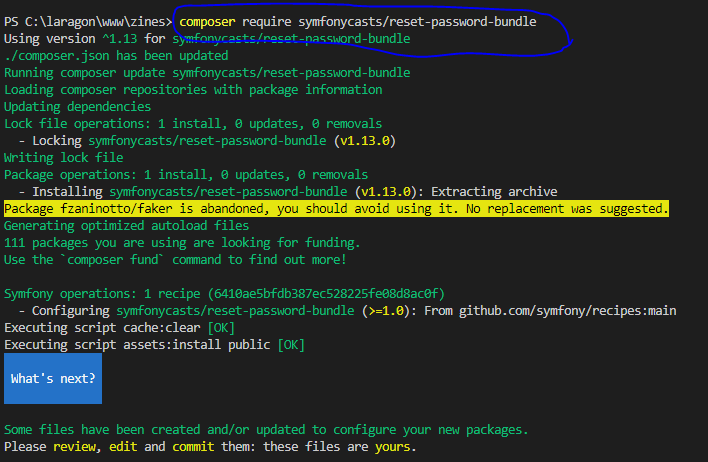
symfony console make:reset-password

un cas d’error

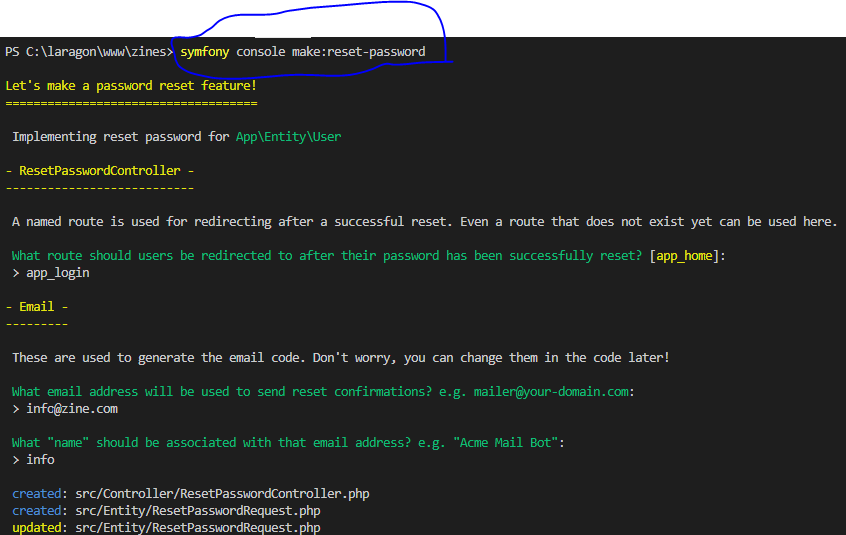


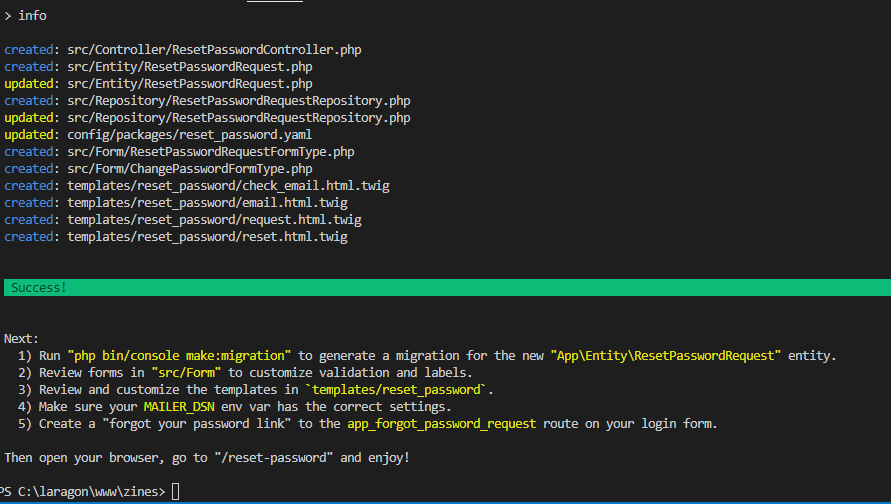
Relancer

composer require symfonycasts/reset-password-bundle

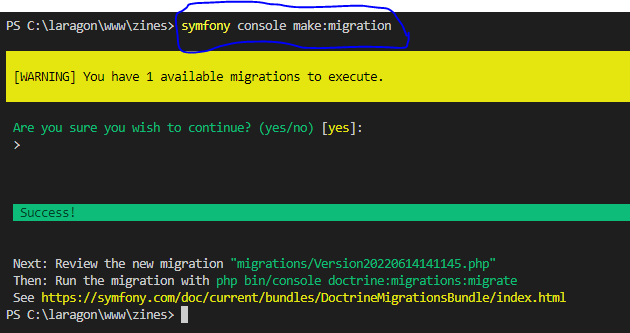


symfony console make:reset-password





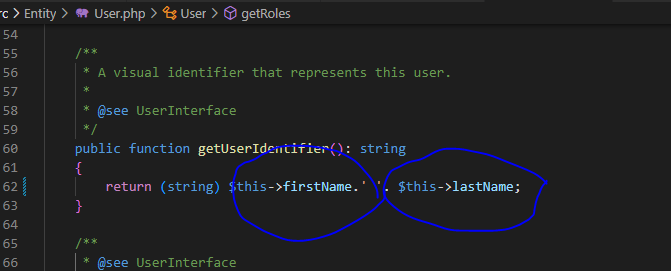
symfony console make:migration

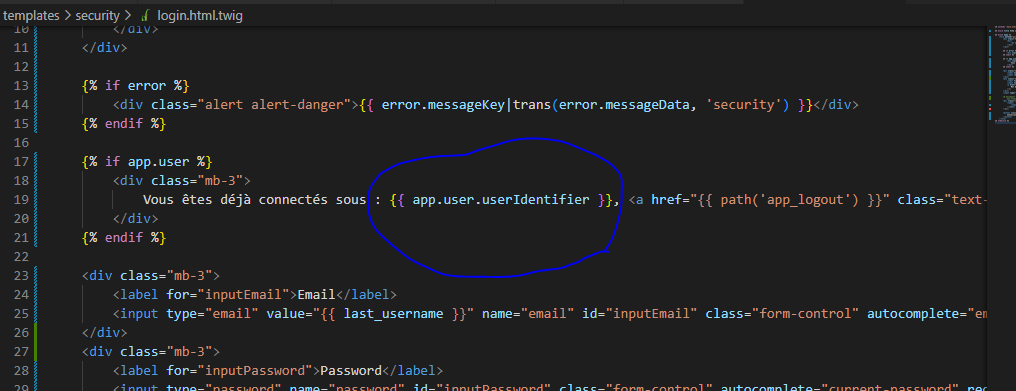


Met à jour la basse de donnée

symfony console d:s:u --force

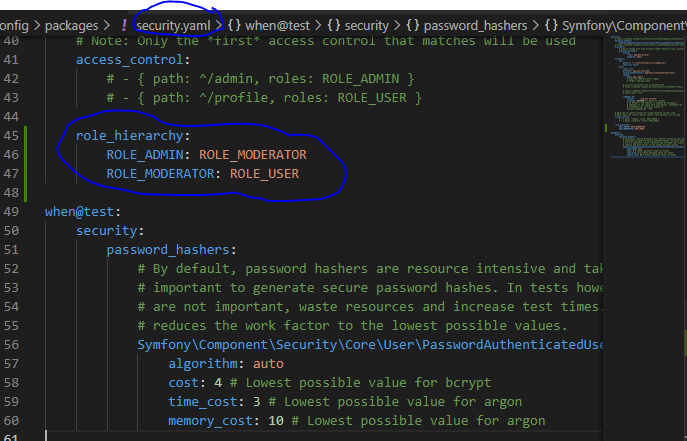
pour debuger



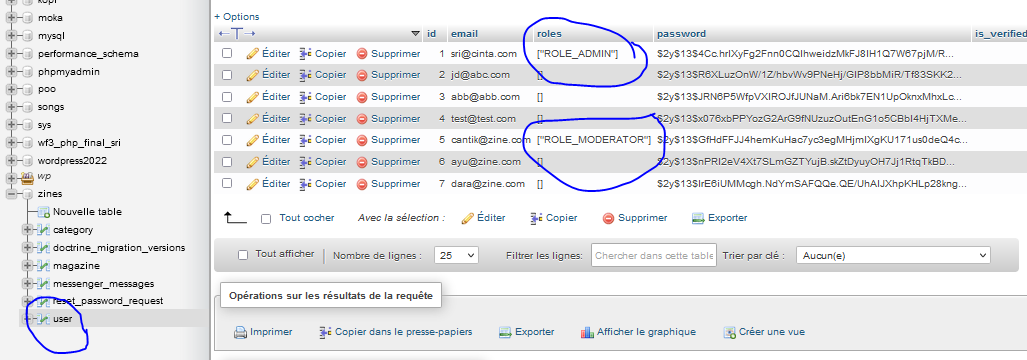


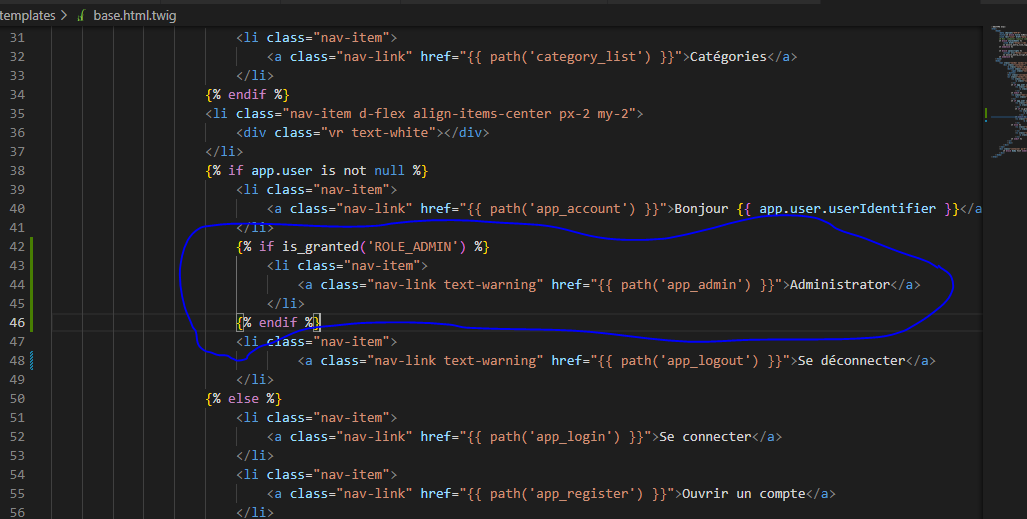
Créer une page d’administration accessible depuis l’adresse « /admin » dans laquelle se trouve une liste des membres du site paginés à 6 par pages. Cette liste contient les informations suivant des utilisateurs : id, prénom et nom, email, rôle et trois boutons : « Administrateur, modérateur et utilisateur », permettant de changer le rôle de l’utilisateur au clique grâce à une requête AJAX. Sur la page listant les magazines, modifier les accès aux boutons « nouveau magazine » et « supprimer » au ROLE\_ADMIN et « éditer » au ROLE\_MODERATOR. La page listant les catégories est seulement accessible au ROLE\_ADMIN.

Security.yaml

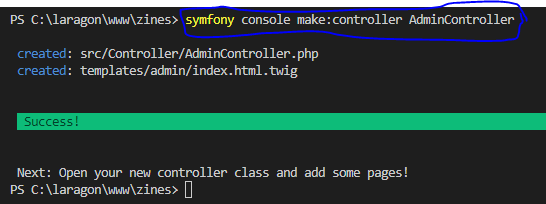


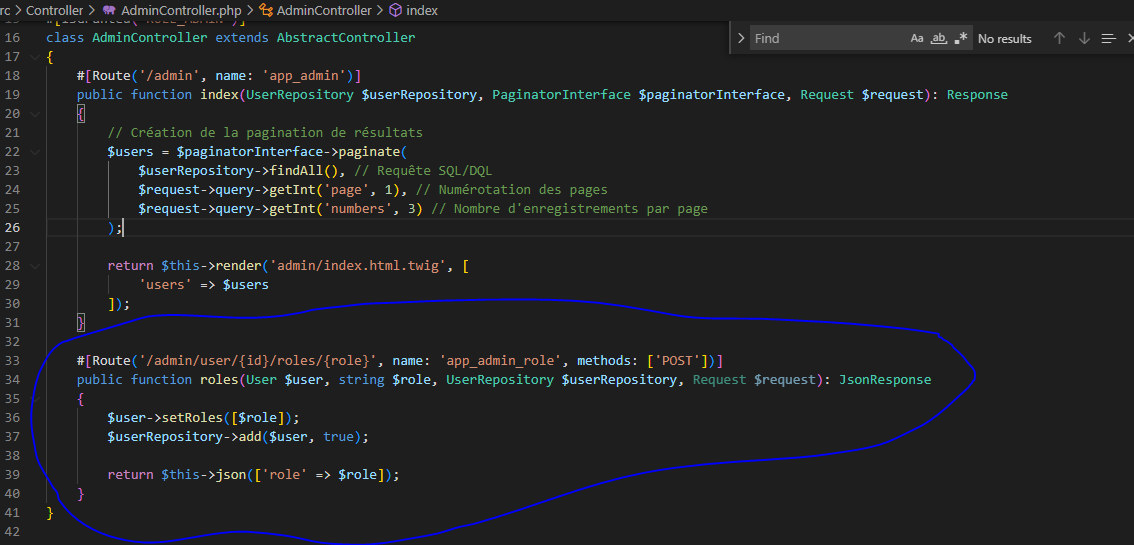
BDD

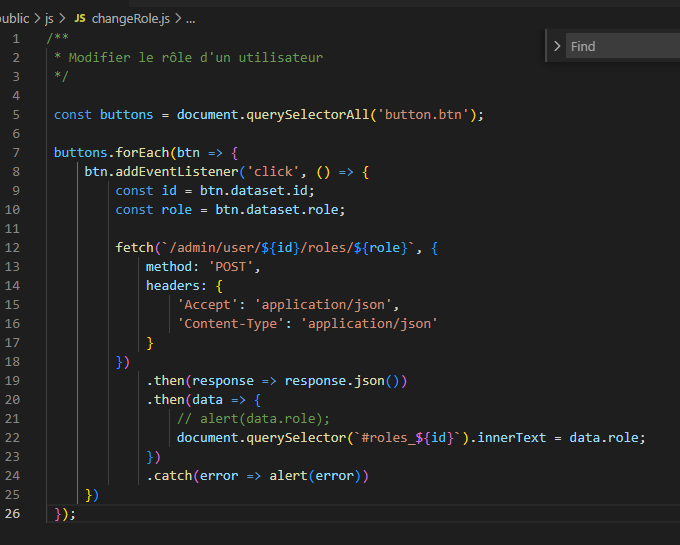




symfony console make:controller AdminController







Envoie email

Création template

