ESPEUT SULLIVAN

Table des matières

Réalisation 1 : Jardin partagé 2
1. Présentation Générale du Projet 2
Identifiants de connexion 3
2. Users cases 3
2.1 Users cases écrit 3
2.2 UML diagramme use case 6
2.3 Navigation avec captures d'écran 6
Inscription 6
Connexion et navigation en tant qu'utilisateur 7
Gestion de ma parcelle (Ajout, modification, suppression de plantes) 9
S'inscrire à une réunion 9
Espace administrateur 10
3. Déploiement et Installation et lancement du site 12
- Prérequis : 12
- Étapes : 12
4. Création du projet 13
MCD 13
Commande création du projet dans symfony, controller 14
5. Fonctionnalités Principales 16
5.1Authentification et autorisations 16
5.2 Gestion de « ma parcelle » 19
5.3 Système de Posts & Commentaires 19
6. Partie Administration 21
7. Tests et Sécurité 25
7.1. Tests 25
7.2 Sécurité 28
8. Front-End & Expérience Utilisateur 30
9. Difficultés Rencontrées 31

Réalisation 1 : Jardin partagé

1. Présentation Générale du Projet

L'association des jardins partagés « Jardin Pota-geai » développe un site internet afin de lister les parcelles du jardin. Initialement prévu pour lister les adhérent(e)s participants et leur parcelle, il a été décidé de donner la possibilité aux adhérent(e)s de pouvoir mettre des plantes sur leur parcelle virtuelle afin de simuler les plantations.

Ils/elles peuvent planter et donner une période de croissance à leurs plantes.

Puis les utilisateurs ont voulu une partie pour échanger sur leurs activités au jardin alors une partie posts et commentaires a été ajoutée, l'utilisateur est redirigé directement vers cette partie après la connexion.

De plus les utilisateurs peuvent s'inscrire à des réunions en présentielles sur différents thèmes.

- **Nom du projet :** Jardin Partagé
- **Technos utilisées :** PHP, Symfony, Doctrine, Twig, CSS, VichUploaderBundle
- **Objectif :** Application de gestion de parcelles végétales dans un jardin partagé,

  Partie administrateur pour gérer les utilisateurs, les parcelles, les plantes, les posts, les réunions.

L'utilisation du framework symfony présente de nombreux avantages, il est reconnu pour sa robustesse, sa flexibilité et sa performance.

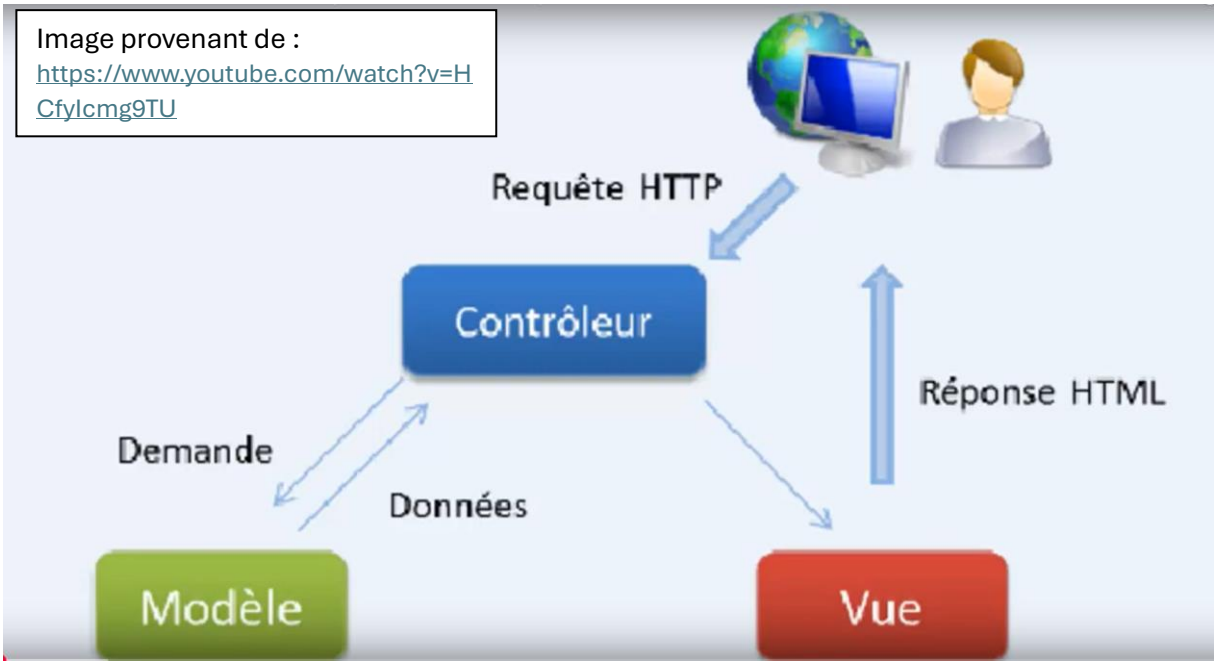
Lors de la création d'une application, symfony la divise en trois parties distinctes :

- Le Modèle (Model) : qui représente les données de l'application ainsi que la logique d'accès et de manipulation des données (Entity et Repository).

- La vue (View) : Elle gère l'affichage des données, elle est gérée par le moteur de template Twig et une syntaxe particulière.

- Le contrôleur (Controller) : traite les requêtes http et renvoie les réponses, c'est l'intermédiaire entre le modèle et la vue.

L'utilisation de Symfony renforce aussi la sécurité de l'application car des couches de protection y sont intégrées contre les failles XSS, les injections SQL et les attaques CSRF.



Il est important d'utiliser LTS (long terme support) car c'est la version supportée sur le long terme, avec des mises à jour et des correctifs.

Identifiants de connexion

Administrateur : test@demo.com, mot de passe : 1234

Utilisateur : aaa@a.com, mot de passe : Bonjourcava?1

Utilisateur : bbb@b.com, mot de passe : Bonjourcava?1

2. Users cases

2.1 Users cases écrit

S'inscrire :

- Sur la page d'accueil je clique sur « s'inscrire » dans la nav barre.
- Je rempli les champs avec les informations demandées (en respectant les regex pour l'adresse de courriel et le mot de passe de fort).
- Je clique sur le bouton « s'inscrire ».
- Je suis connecté et redirigé vers la page avec les posts (page par défaut après connexion).

Se connecter :

- Sur la page d'accueil je clique sur le bouton « se connecter » situé dans la nav barre.
- Je rempli le formulaire de connexion avec mes identifiants.
- Je soumets le formulaire.
- Je suis redirigé sur la page des posts en cas de succès.

Ajouter une plante dans le cas où j'ai une parcelle :

- Je me connecte.

- Je vais sur ma parcelle.
- Je clique sur « ajouter une plante ».
- Je rempli les informations sur la plante que je souhaite ajouter.
- Je soumets le formulaire.
- La plante est ajoutée en base de données.
- Je suis redirigé sur la page de ma parcelle .
- Un message de confirmation est affiché (addFlash).

S'inscrire à une réunion :

- Je vais sur la page des posts.
- Si une réunion est disponible, je peux m'inscrire en cliquant sur le bouton « s'inscrire ».
- L'inscription est ajoutée en base de données.
- La page est réactualisée.

Ajouter un post :

- Je me connecte.
- Je vais sur la page des posts.
- Je clique sur « ajouter ».
- J'écris un message, je peux ajouter une image.
- Je clique sur « envoyer ».
- Le post est enregistré en base de données.
- La page est rechargée et le message s'affiche.

Modifier un post :

- Je me connecte.
- Je vais sur la page des posts.
- Je clique sur le bouton « modifier » sur un de mes posts.
- Je modifie le contenu ou l'image du post.
- Je clique sur « envoyer ».
- La modification du post est enregistrée en base de données.
- La page est rechargée et le message modifié s'affiche.

Ajouter un commentaire :

- Je me connecte.
- Je vais sur la page des posts.
- Je clique sur le bouton « ajouter un commentaire » sur un post.
- J'écris un commentaire.
- Je clique sur « envoyer ».
- Le commentaire est enregistré en base de données.
- La page est rechargée et le commentaire s'affiche en dessous du post.

Pour administrateur

Création parcelle :

- Je me connecte en tant qu'administrateur.
- Je clique sur « tableau de bord » dans la nav barre.
- Je clique sur « parcelles ».
- Je renseigne les informations de la parcelle.
- La parcelle est enregistrée en base de données.
- La page est rechargée.

Afficher une parcelle :

- Je me connecte en tant qu'administrateur.
- Je clique sur « tableau de bord » dans la nav barre.
- Je clique sur « parcelles ».
- Sur une parcelle, je clique sur « afficher details ».
- Je suis redirigé sur une page sur laquelle les détails de la parcelle (numéro, size etc...) s'affichent.

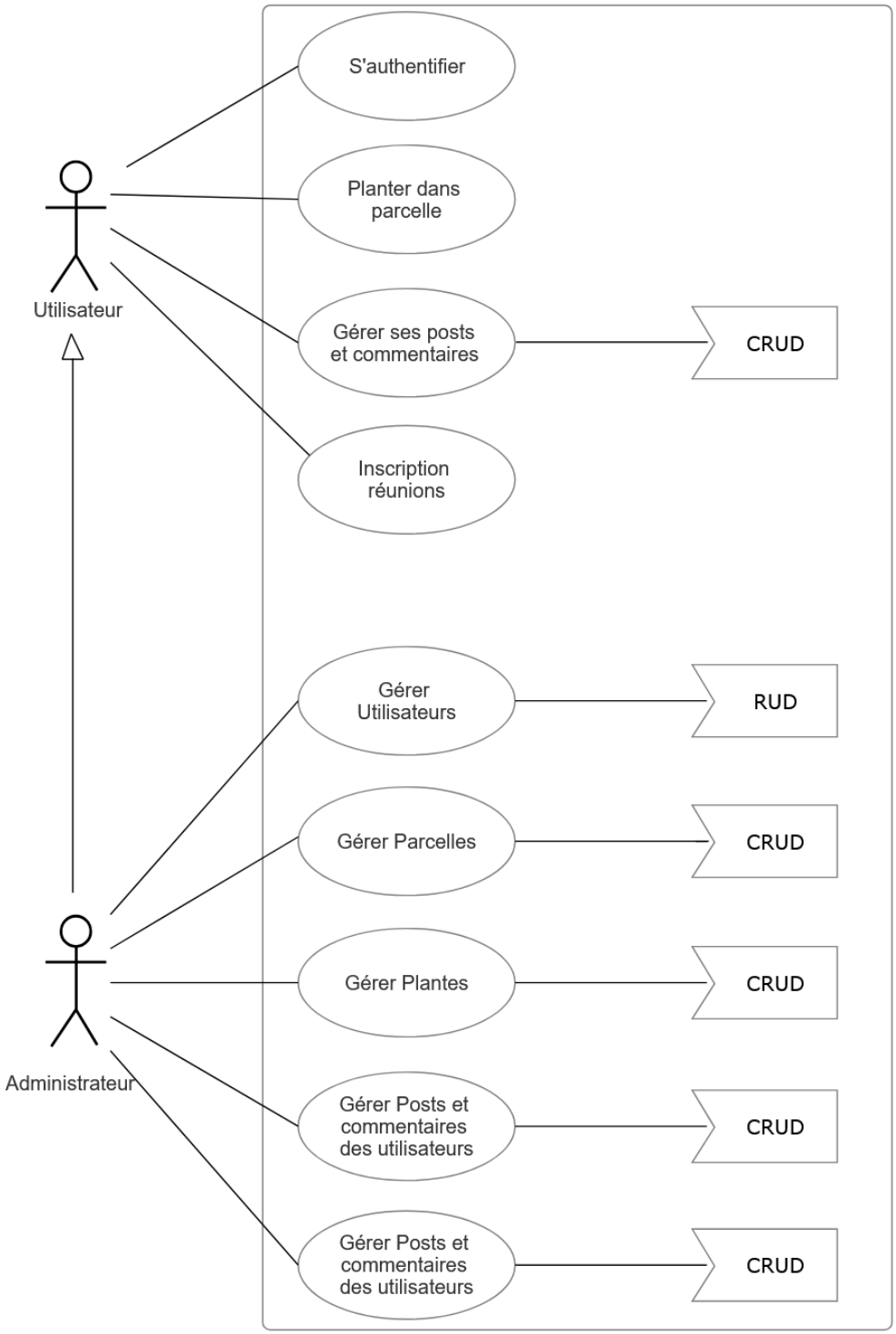
Modifier une parcelle :

- Je me connecte en tant qu'administrateur.
- Je clique sur tableau de bord dans la nav barre.
- Je clique sur « parcelles ».
- Sur une parcelle, je clique sur « modifier ».
- Je fais des changements.
- Je clique sur « modifier ».
- Les changements sont enregistrés en base de données.
- Je suis redirigé sur la page avec toutes les parcelles et je vois que les changements ont été appliqués.

Supprimer une parcelle :

- Je me connecte en tant qu'administrateur.
- Je clique sur tableau de bord dans la nav barre.
- Je clique sur parcelles.
- Je clique sur modifier.
- Je suis redirigé vers une page avec les informations de la parcelle.
- je clique sur supprimer.
- La parcelle est supprimée en base de données.
- Je suis redirigé vers la page des parcelles et je constate que la parcelle que j'ai supprimée a disparu.

2.2 UML diagramme use case

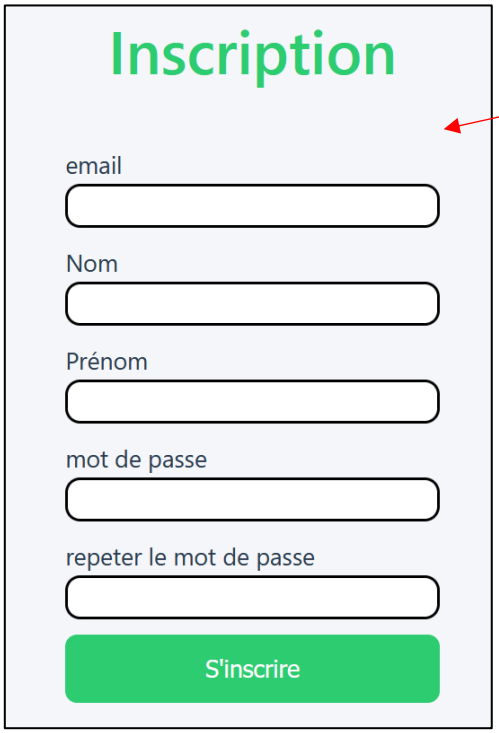


UML Use case de l'application jardin partagé fait avec SmartDraw

2.3 Navigation avec captures d'écran

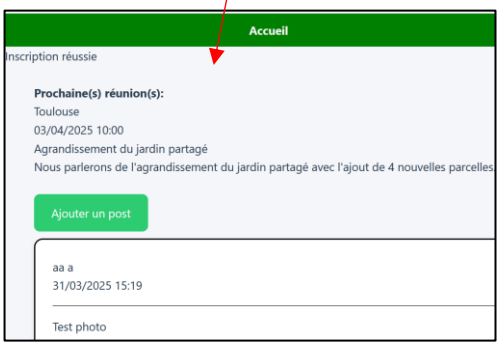
Inscription

Barre de navigation de la page d'accueil quand non connecté.

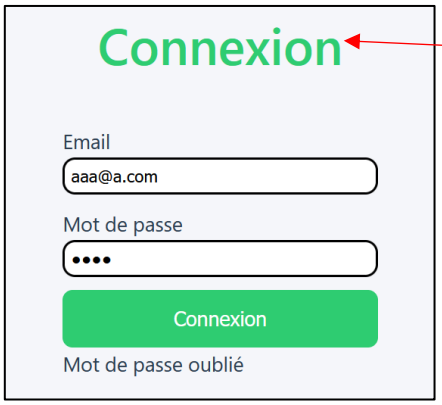


Formulaire d'inscription.

Redirection après inscription.

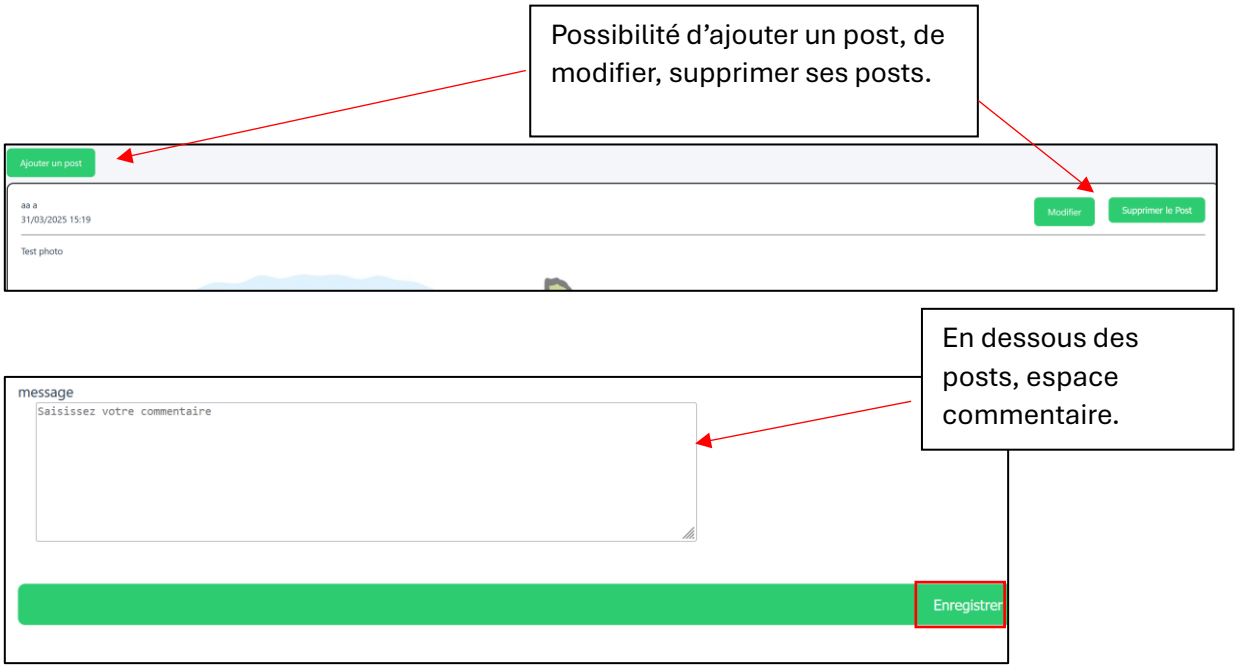


Connexion et navigation en tant qu'utilisateur

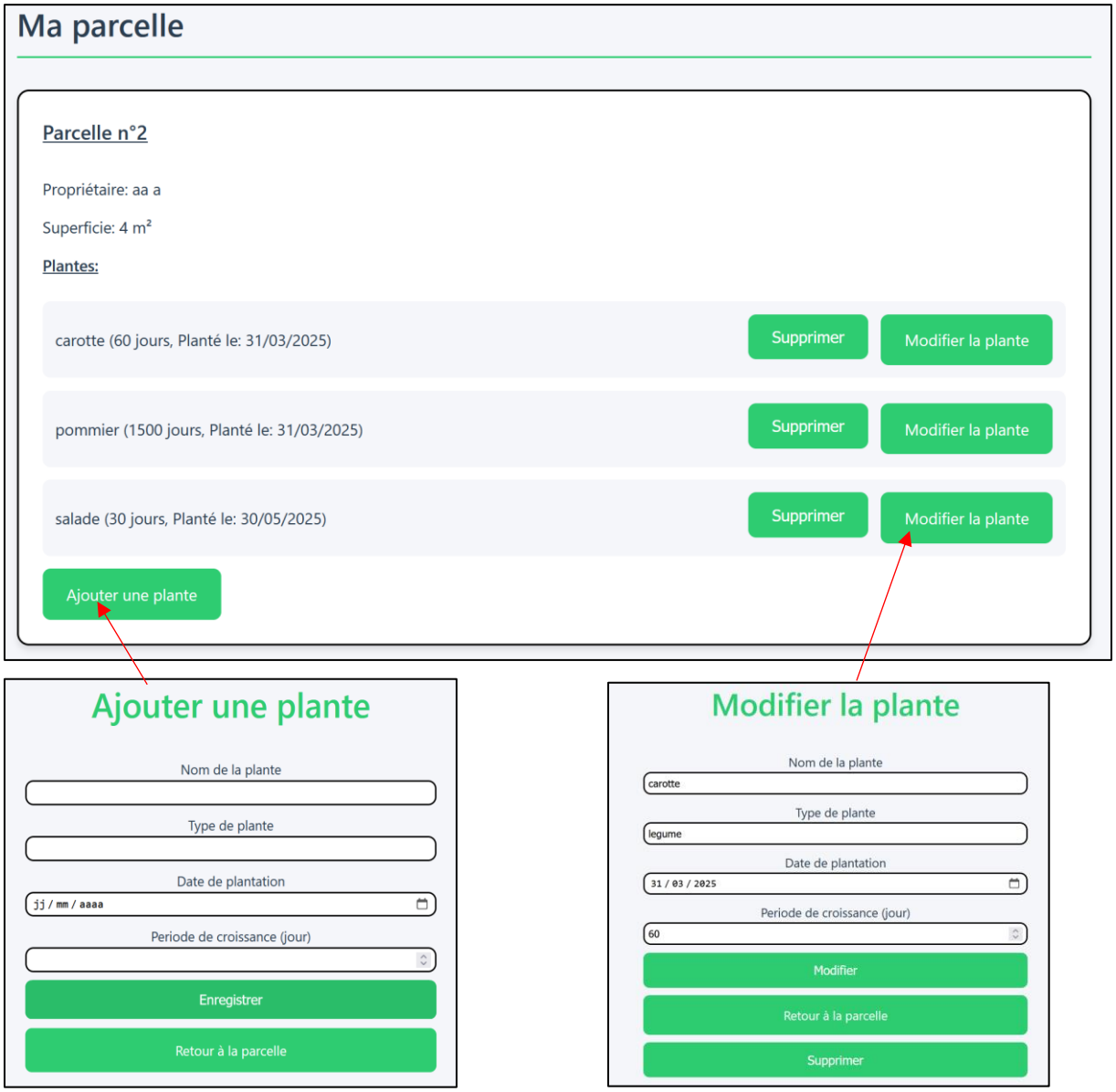


Formulaire de connexion.

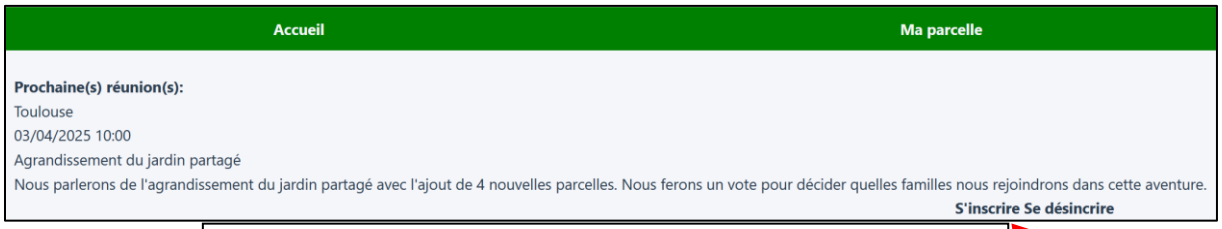
Modification de la barre de navigation.


Possibilité d’ajouter un post, de modifier, supprimer ses posts.
Ajouter un post
aa a
31/03/2025 15:19
Modifier
Supprimer le Post
Test photo
En dessous des posts, espace commentaire.
message
Saisissez votre commentaire
Enregistrer

Gestion de ma parcelle (Ajout, modification, suppression de plantes)



S’inscrire à une réunion

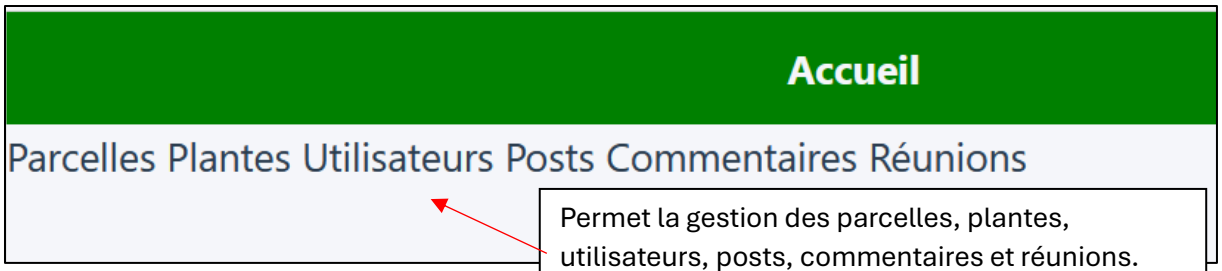


Sur la page d’accueil,

Possibilité de s’inscrire ou de se désinscrire à une réunion.

Espace administrateur

Un administrateur à accès à tout ce que peut faire un utilisateur, le Tableau de bord en plus



Permet la gestion des parcelles, plantes, utilisateurs, posts, commentaires et réunions.

Liste des parcelles

Identifiant	Numéro	Superficie	Date de création	actions
23	1	2	2025-02-21 13:37:18	Afficher Modifier
24	2	4	2025-02-21 13:37:18	Afficher Modifier
25	3	6	2025-02-21 13:37:18	Afficher Modifier
26	4	8	2025-02-21 13:37:18	Afficher Modifier
27	5	10	2025-02-21 13:37:18	Afficher Modifier
28	6	12	2025-02-21 13:37:18	Afficher Modifier
29	7	14	2025-02-21 13:37:18	Afficher Modifier
30	8	500	2025-02-21 13:37:18	Afficher Modifier
32	10	9999	2025-02-24 17:06:05	Afficher Modifier
33	11	50	2025-02-24 17:13:41	Afficher Modifier
34	27	85	2025-03-04 14:06:02	Afficher Modifier

Créer une nouvelle parcelle Tableau de Bord

L'administrateur peut naviguer facilement entre les différents menus grâce à des liens lui permettant de revenir en arrière.

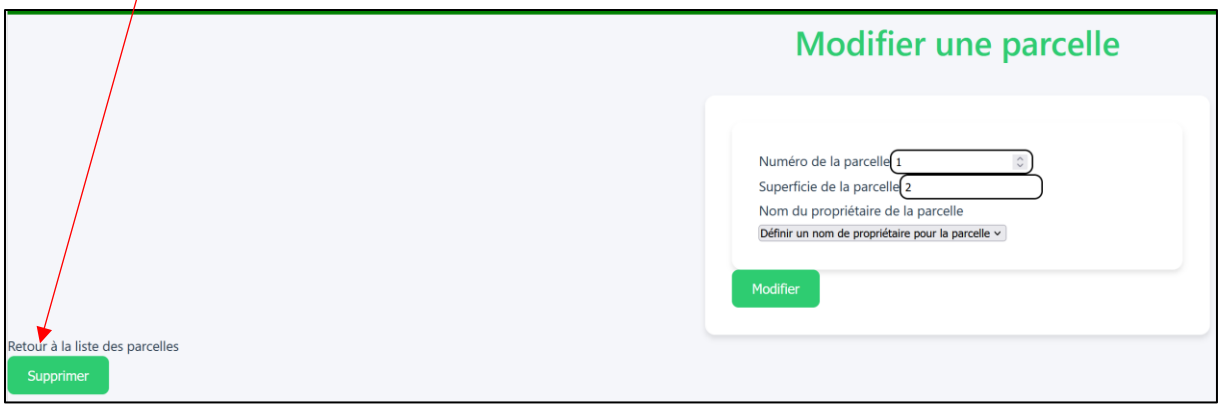
Parcelle

Identifiant de la parcelle :	34
Numéro de la parcelle :	27
Superficie :	85
Propriétaire :	Aucun propriétaire défini pour cette parcelle
Créé le :	04/03/2025 14:06

Retour Modifier

Supprimer

Afficher une parcelle



Liste des plantes

Id	Nom	Type	DatePlantation	PeriodeCroissance	Date de création	actions
5	salade	legume	2025-02-21 13:37:00	3	2025-02-21 13:37:00	Afficher Modifier
6	salade	legume	2025-02-21 13:37:19	4	2025-02-21 13:37:19	Afficher Modifier
7	salade	legume	2025-02-21 13:37:19	5	2025-02-21 13:37:19	Afficher Modifier
8	salade	legume	2025-02-21 13:37:19	6	2025-02-21 13:37:19	Afficher Modifier
9	salade	legume	2025-02-21 13:37:19	7	2025-02-21 13:37:19	Afficher Modifier
10	salade	legume	2025-02-21 13:37:19	8	2025-02-21 13:37:19	Afficher Modifier
12	salade	legume	2025-03-07 10:10:00	3	2025-03-10 10:10:00	Afficher Modifier
13	radis	legume	2025-03-07 00:00:00	5	2025-03-07 15:04:40	Afficher Modifier
15	carotte	legume	2025-03-07 00:00:00	1	2025-03-07 15:07:46	Afficher Modifier
16	radis	legume	2025-03-07 00:00:00	5	2025-03-07 15:49:29	Afficher Modifier
17	radis	legume	2025-03-07 00:00:00	5	2025-03-07 15:53:33	Afficher Modifier
18	salade	legume	2025-06-04 00:00:00	30	2025-03-27 14:44:18	Afficher Modifier
19	carotte	legume	2025-03-31 00:00:00	60	2025-03-31 15:22:12	Afficher Modifier
20	pommier	arbuste	2025-03-31 00:00:00	1500	2025-03-31 16:28:23	Afficher Modifier
21	salade	legume	2025-05-30 00:00:00	30	2025-04-08 15:48:05	Afficher Modifier

Créer une nouvelle plante Tableau de Bord

Même principe pour les plantes, les utilisateurs (sauf créer), les posts, les commentaires et les réunions.

Liste des utilisateurs

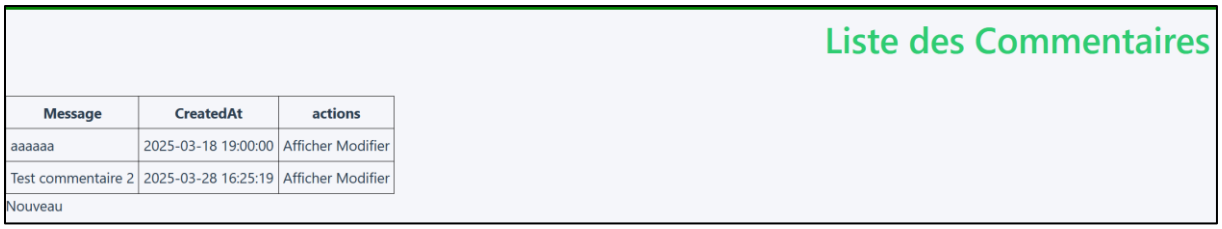
Id	Email	Roles	Lastname	Firstname	actions
3	test@demo.com	Administrateur , Utilisateur	Nom	Sullivan	Afficher Modifier
4	bbb@b.com	Utilisateur	bb	b	Afficher Modifier
5	aaa@a.com	Utilisateur	a	aa	Afficher Modifier
7	test2@test.com	Utilisateur	tesst	test	Afficher Modifier

Tableau de Bord

Liste des Posts

Message	CreatedAt	ImageName	UpdatedAt	actions
Bonjour Jordan. Merci de m'enseigner la programmation. Tu es un superprof!	2025-03-12 17:44:43			Afficher Modifier
3eme post.	2025-03-16 11:21:31			Afficher Modifier
Test Test	2025-03-18 18:01:06	bridge-9456745-1280-680a5cfe9d35e883654334.jpg	2025-04-24 15:47:10	Afficher Modifier
Test photo	2025-03-31 15:19:40	garden-7833569-1280-67dd953ca04a7280360915-67ea966c76af3541709569.jpg	2025-03-31 13:19:40	Afficher Modifier

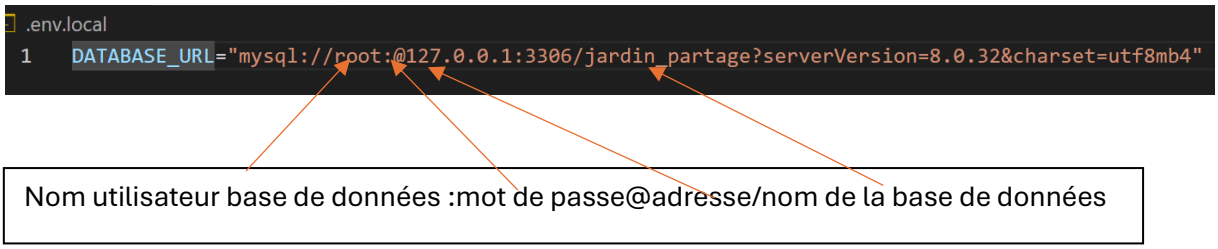
Nouveau



3. Déploiement et Installation et lancement du site

- Prérequis :
  - Version de PHP supérieure ou égale à 8.2
  - Symfony CLI
  - MySQL
  - Composer
  - phpMyAdmin
- Étapes :

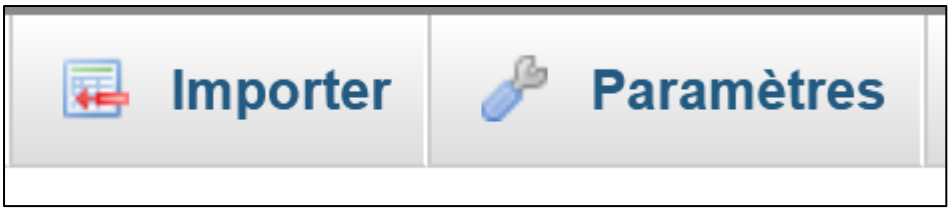
a) git clone https://github.com/Sulli573/jardin-partage.git
b) Aller dans le dossier jardin-partage
c) Puis commandes dans le terminal sur la racine du projet
composer install (va installer ce qu'il faut pour que le projet tourne = dépendances)
d) Créer un .env.local à la racine et rentrer vos informations de connexions à la base de données :



e) Maintenant que les identifiants sont renseignés créer la base de données avec la commande :

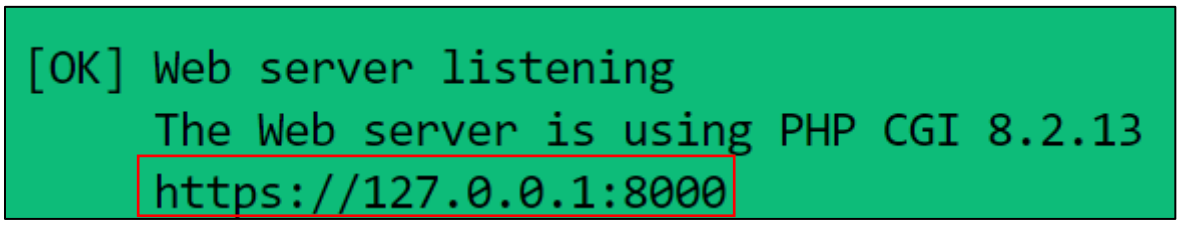
- php bin/console doctrine:database:create

f) Aller dans phpMyAdmin, sélectionner la base de données, aller dans l'onglet Importer



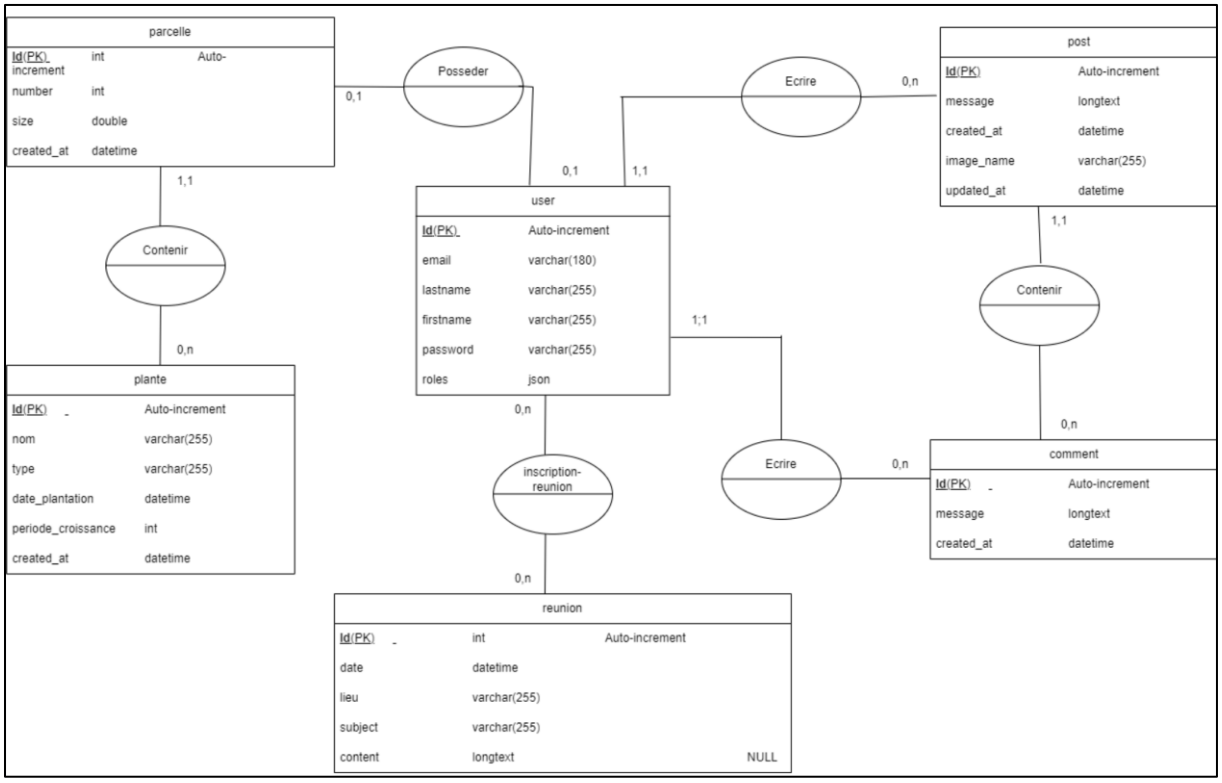
Faire parcourir et importer le fichier jardin_partage.sql qui est à la racine du projet.
Ou
Dans le terminal de votre ide :
mysql -u VOTRE_USER -pVOTRE_PASSWORD NOMDEVOTRE_BASEDEDONNEES < jardin_partage.sql

Une fois tout installer taper la commande symfony serve -d (-d afin de pouvoir taper d'autres commandes dans le terminal par la suite) et rendez-vous à l'adresse indiquée :

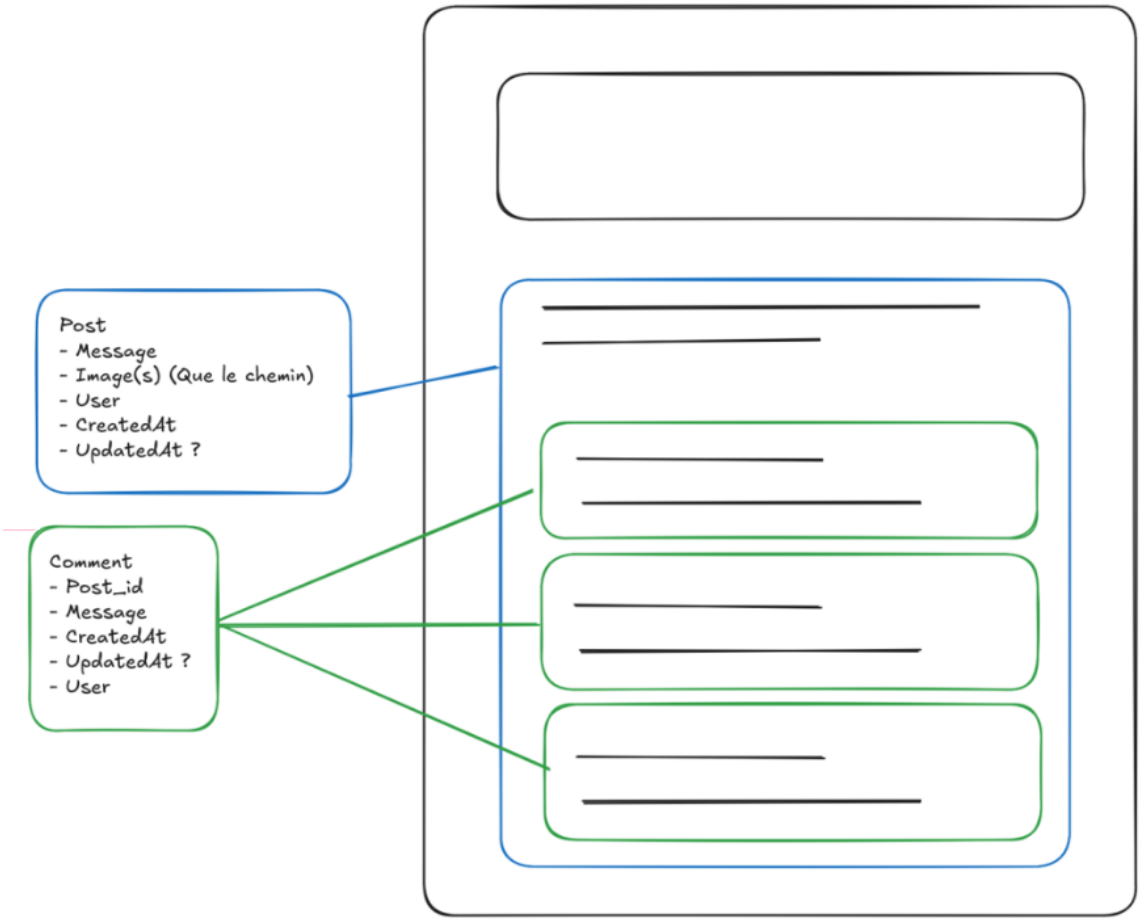


4. Création du projet

MCD



Mcd créé avec mcd.drawio (extension visual studio code)

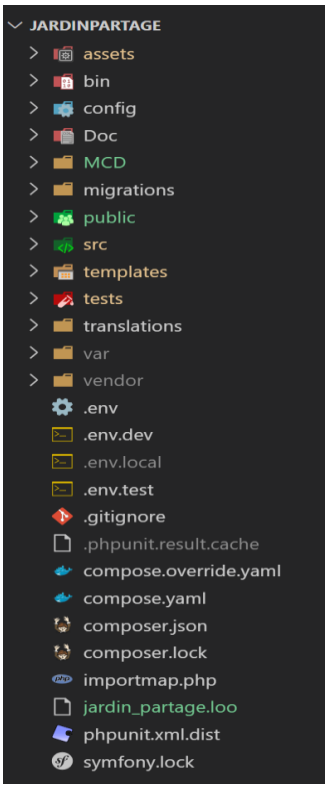


Réflexion, schématisation pour le front des posts et commentaires

Commande création du projet dans symfony, controller

Pour créer le projet, dans le terminal de mon ide visual studio code, j'ai taper la commande :

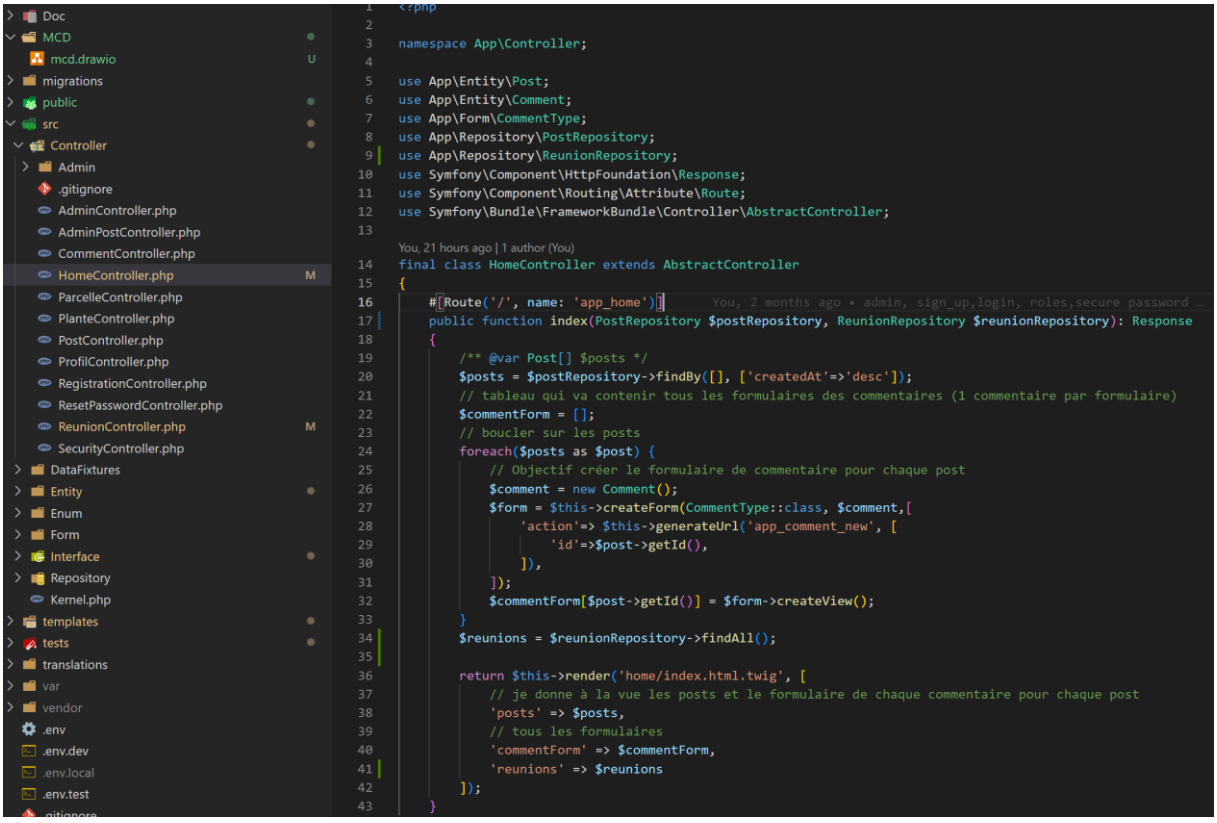
Symfony new JARDINPARTAGE --webapp cela à créé le dossier JARDINPARTAGE avec toutes les dépendances (grâce à –webapp, sans cela aurait été moins complet).



Puis j'ai créé mon premier controller

Exemple de controller : PlanteController.php avec la commande php bin/console make :controller

Va créer le controller et la vue associée. Le contoller permet d'associer une logique à une route spécifique



Il y a la logique avec la route ayant comme url / et comme nom app_home(à utiliser par exemple dans les liens).

La fonction index va créer les formulaires de création de commentaires pour chaque post puis l'envoyer à la vue (templates/home/index.html.twig):

```
<div class="card-coment">
    {# initialisation du formulaire #}      You, last month • modifications front
    {#   post.id en twig  équivaut à $post->getId() en php natif #}
    {{ form_start(commentForm[post.id]) }}
    {{ form_row(commentForm[post.id]) }}
        <button class="btn"> {{ button_label|default('Enregistrer') }}</button>
    {{ form_end(commentForm[post.id]) }}

    {% for comment in post.comments|sort((a, b) => b.createdAt <=> a.createdAt)%}

        <div class="card-comment-item">
            <div class="card-comment-message">
                <span>{{comment.user.fullName}}</span>
                <span>{{comment.message}}</span>
            </div>

            {% if comment.user == app.user %}
                <div class="card-comment-actions">
                    <a href="{{ path('app_comment_edit', {'id': comment.id}) }}" class="btn">Modifier</a>
                    {% include "comment/_delete_form.html.twig" %}
                </div>
            {% endif %}
```

Qui va afficher un formulaire pour saisir un commentaire sur le post.

Sera Affiché les commentaires des utilisateurs et si vous êtes l'utilisateur qui a écrit le commentaire les liens pour le modifier ou le supprimer apparaitrons.

C'est aussi rendu possible grâce au Repository qui « va chercher » les informations en base de données pour l'entity post.

5. Fonctionnalités Principales

5.1Authentification et autorisations

- Création utilisateur + mot de passe sécurisé (regex)

  Contrainte de force du mot de passe dans RegistrationFormType.php

```
'constraints' => [
    new Regex(
        pattern: '/^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{12,}$/',
        message: "Veuillez créer un mot de passe d'au moins 12 caractères, incluant une majuscule, une minuscule, un chiffre et un caractère spécial (@$!%*?&)."
    )
],      You, 2 months ago • admin, sign_up,login, roles,secure password
```

- Connexion / déconnexion

```
#[Route(path: '/login', name: 'app_login')]
public function login(AuthenticationUtils $authenticationUtils): Response
{
    if ($this->getUser()) {
        return $this->redirectToRoute("app_parcelle_index");
    }
```

  Le formulaire de connexion :

```twig
<div class="login-container">
    <form class="login-form" method="post">
        {% if error %}
            <div class="alert alert-danger">{{ error.messageKey|trans(error.messageData, 'security') }}</div>
        {% endif %}

        <h1 class="h3 mb-3 font-weight-normal">Connexion</h1>       You, last month • modifications front

        <div class="form-input">
            <label for="username">Email</label>
            {# Last_username sert  à garder l'email dans le champs lors d'une soumission de formulaire avec mauvais identifiants #}
            <input type="email" value="{{ last_username }}" name="_username" id="username" class="form-control" autocomplete="email" required autofocus placeholder="Insérer votre adresse email(courriel)">
        </div>

        <div class="form-input">
            <label for="password">Mot de passe</label>
            <input type="password" name="_password" id="password" class="form-control" autocomplete="current-password" required placeholder="Entrez votre mot de passe">
        </div>

        <input
        type="hidden" name="_csrf_token" data-controller="csrf-protection" value="{{ csrf_token('authenticate') }}">
```

Route de la deconnexion

```php
#[Route(path: '/logout', name: 'app_logout')]
public function logout(): void
{
    throw new \LogicException('This method can be blank - it will be intercepted by the logout key on your firewall.');
}
```

- Restriction des pages /admin aux administrateurs

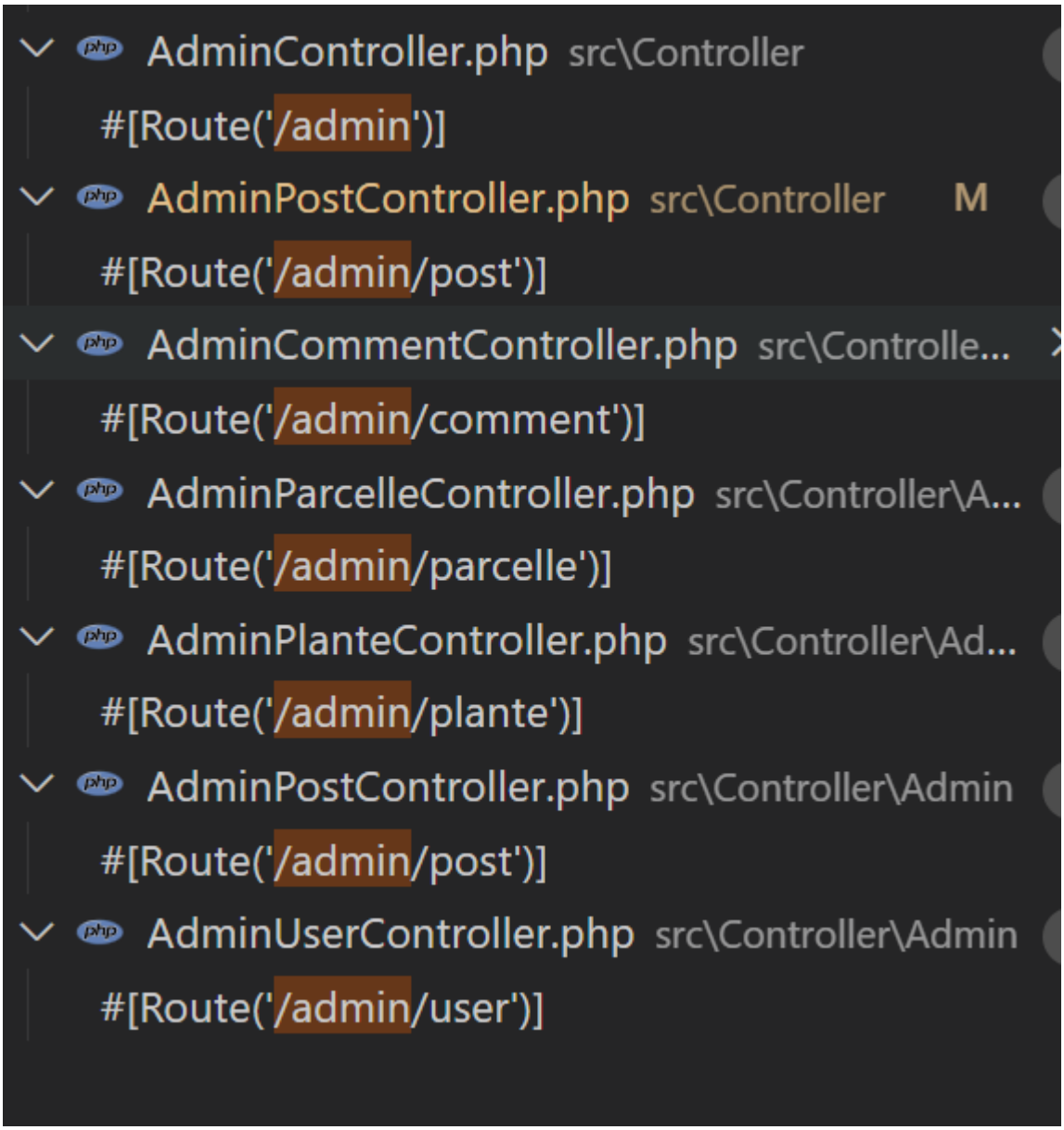
  Chemin :config/packages/security.yaml

  Décommenter :

```yaml
access_control:
    # - { path: ^/admin, roles: ROLE_ADMIN }
```

```yaml
access_control:
    - { path: ^/admin, roles: ROLE_ADMIN }
```

Tous les chemins commençant par admin sont interdits aux utilisateurs n’ayant pas le rôle ADMIN :



- Toutes les routes du controleur ParcelleController.php ne sont accessibles qu'à l'utilisateur authentifié :
  Dans ParcelleController.php grâce à IsGranted.

```php
#[IsGranted('ROLE_USER')]
final class ParcelleController extends AbstractController
{
    #[Route('/parcelle', name: 'app_parcelle_index', methods: ['GET'])]
    public function index(ParcelleRepository $parcelleRepository): Response
    {
        #recupérer l'utilisateur connecté
        /** @var User $user */
        $user = $this->getUser();
        #recupérer sa parcelle      You, 2 months ago • admin, sign_up,logir
        $parcelle = $user->getParcelle();
        return $this->render('parcelle/index.html.twig', [
            'parcelle' => $parcelle
        ]);
    }
```

5.2 Gestion de « ma parcelle »

- CRUD sur les plantes

Update

```php
#[Route('/{id}/edit', name: 'app_plante_edit', methods: ['GET', 'POST'])]
public function edit(Request $request, Plante $plante, EntityManagerInterface $entityManager): Response
{
    $form = $this->createForm(PlanteType::class, $plante);
    $form->handleRequest($request);

    if ($form->isSubmitted() && $form->isValid()) {
        $entityManager->flush();

        return $this->redirectToRoute('app_parcelle_index', [], Response::HTTP_SEE_OTHER);
    }

    return $this->render('plante/edit.html.twig', [      You, 3 months ago • CRUD plante,redirection apres suppressi
        'plante' => $plante,
        'form' => $form,
    ]);
}
```

Delete

```php
#[Route('/{id}', name: 'app_plante_delete', methods: ['POST'])]
public function delete(Request $request, Plante $plante, EntityManagerInterface $entityManager): Response
{
    if ($this->isCsrfTokenValid('delete'.$plante->getId(), $request->getPayload()->getString('_token'))) {
        $entityManager->remove($plante);
        $entityManager->flush();
    }

    return $this->redirectToRoute('app_parcelle_index', [], Response::HTTP_SEE_OTHER);
}
```

5.3 Système de Posts & Commentaires

- Création de posts et de commentaires (authentifiés)
- Limitation à l'auteur (édition/suppression)

  Montrer que posts et commentaires éditables et supprimables que par l'auteur

- Upload d'image dans les posts (VichUploader)

vich_uploader.yaml

config > packages > vich_uploader.yaml > { } vich_uploader > { } mappings > { } posts

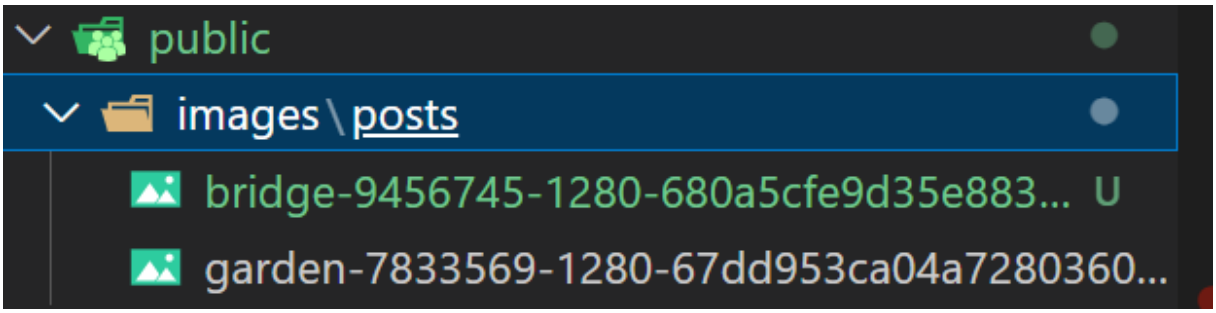
You, last month | 1 author (You)

```yaml
vich_uploader:
    db_driver: orm

    mappings:
        posts:
            uri_prefix: /images/posts
            upload_destination: '%kernel.project_dir%/public/images/posts'
            namer: Vich\UploaderBundle\Naming\SmartUniqueNamer

```

Et dans l'Entity Post.php :

```php
#[Vich\UploadableField(mapping: 'posts', fileNameProperty: 'imageName')]
private ?File $imageFile = null;

#[ORM\Column(nullable: true)]
private ?string $imageName = null;
```

Les images seront donc stockées dans :



- Affichage des posts/commentaires triés (du + récent au + ancien)

  Tri des posts dans HomeController.php

```php
$posts = $postRepository->findBy([], ['createdAt'=>'desc']);
```

  Affichage des commentaires par date de création dans templates/home/index.html.twig

```twig
{% for comment in post.comments|sort((a, b) => b.createdAt <=> a.createdAt)%}
```

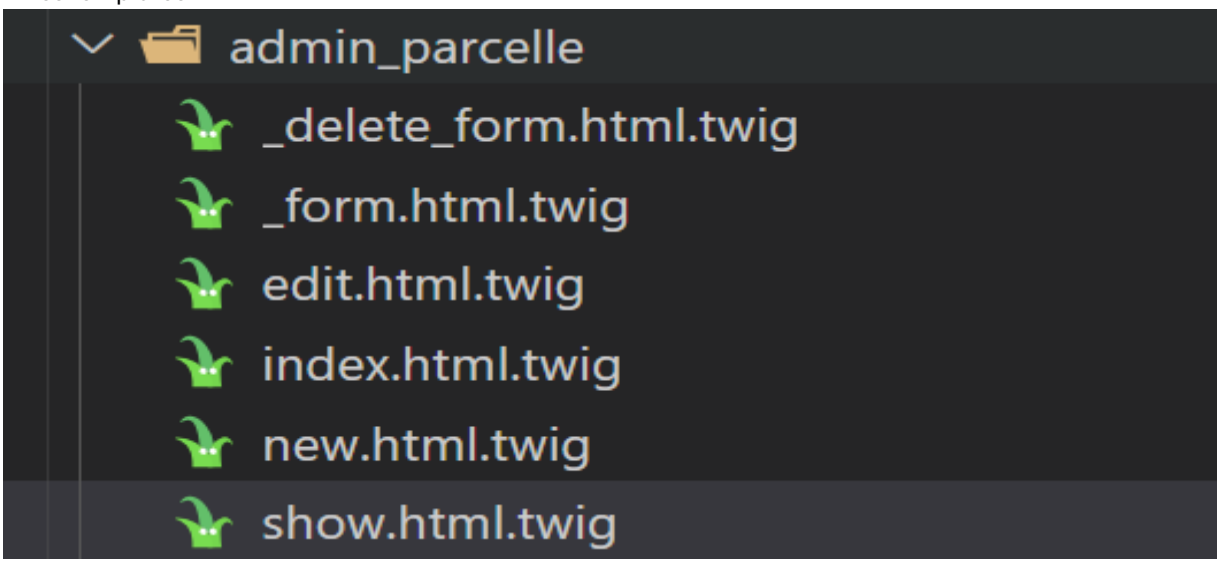
6. Partie Administration

- CRUD admin sur Users (l'admin ne peut pas créer d'utilisateur), Parcelles, Plantes, Posts, Commentaires, Réunions.

  Exemple pour parcelle

  Php bin/console make :crud
  Pour créer AdminParcelleController.php avec les routes et la logique de l'affichage, la création, la modification et la suppression de parcelles

Et les templates :



Ce qui donne comme résultat :

Identifiant	Numéro	Superficie	Date de création	actions
23	1	2	2025-02-21 13:37:18	Afficher Modifier
24	2	4	2025-02-21 13:37:18	Afficher Modifier
25	3	6	2025-02-21 13:37:18	Afficher Modifier
26	4	8	2025-02-21 13:37:18	Afficher Modifier
27	5	10	2025-02-21 13:37:18	Afficher Modifier
28	6	12	2025-02-21 13:37:18	Afficher Modifier
29	7	14	2025-02-21 13:37:18	Afficher Modifier
30	8	500	2025-02-21 13:37:18	Afficher Modifier
32	10	9999	2025-02-24 17:06:05	Afficher Modifier
33	11	50	2025-02-24 17:13:41	Afficher Modifier
34	27	85	2025-03-04 14:06:02	Afficher Modifier

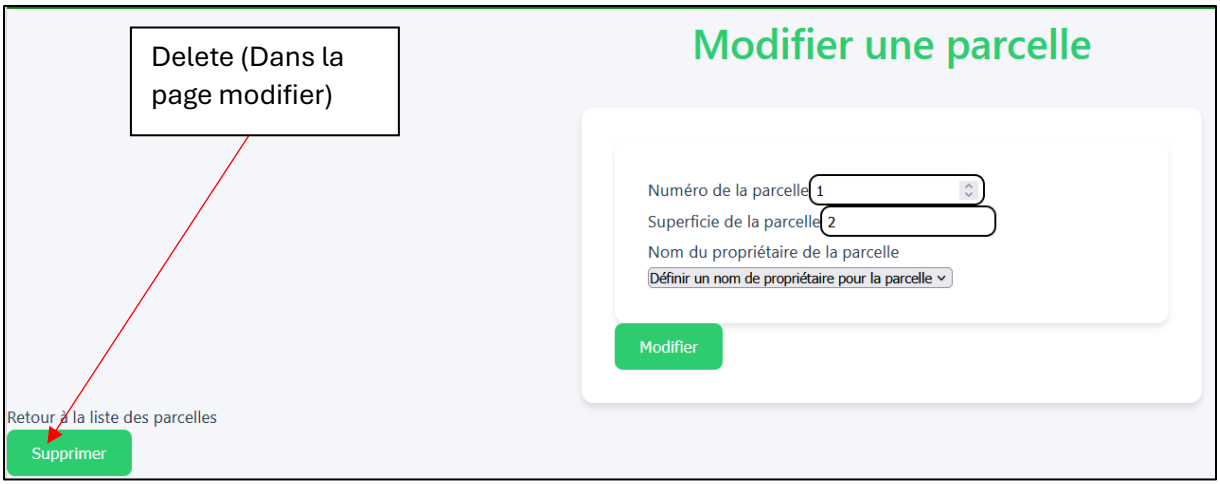
Créer une nouvelle parcelle Tableau de Bord

Create(New)

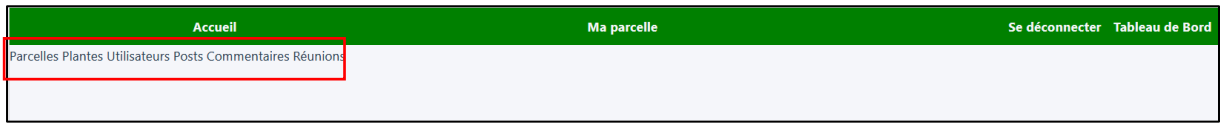
Read(Show)

Update(Edit)

Delete (Dans la page modifier)

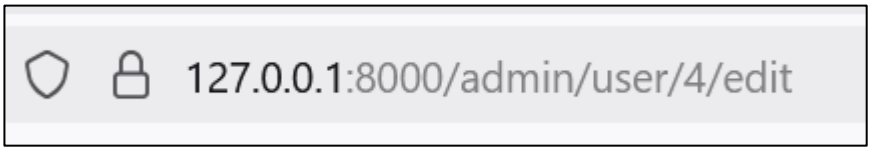


- Interface avec navigation claire



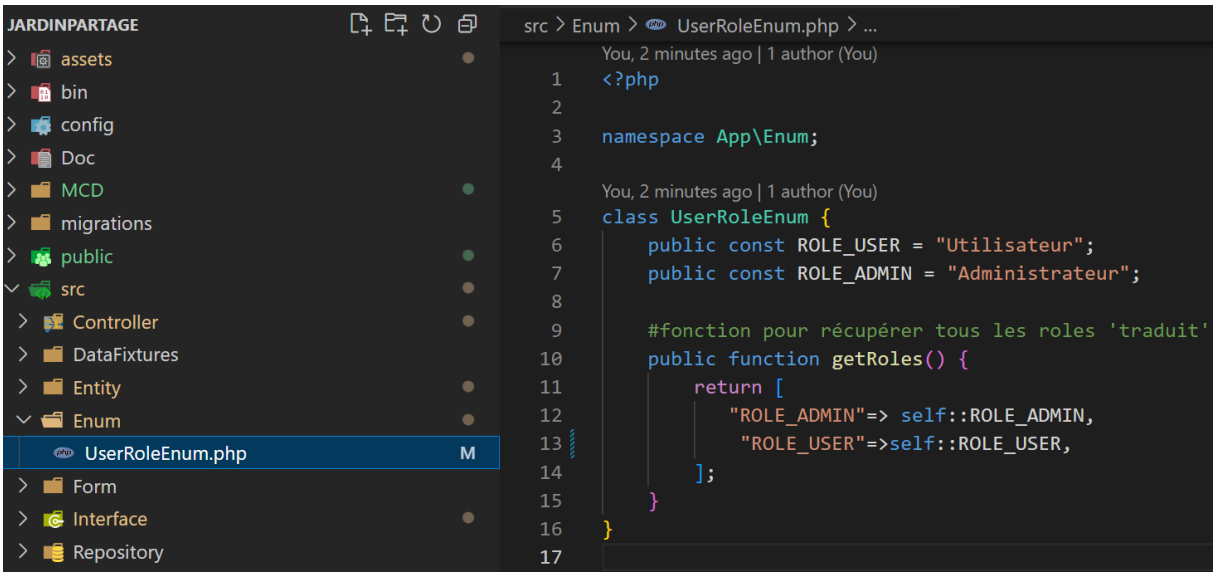
- Assignation des rôles et parcelles pour un utilisateur

Pour le rôle va d'abord récupérer l'utilisateur, créer le formulaire de modification avec la route edit Admin/AdminUserController.php et injecter dans le formulaire les données de l'utilisateur

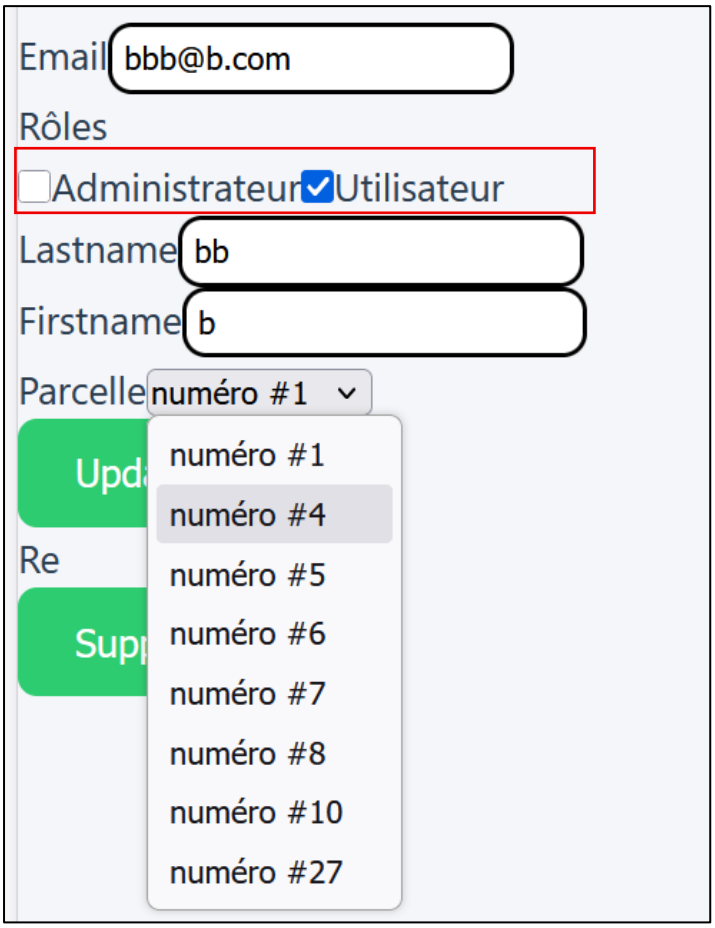



```php
#[Route('/{id}/edit', name: 'app_admin_user_edit', methods: ['GET', 'POST'])]
public function edit(Request $request, User $user, EntityManagerInterface $entityManager): Response
{
    $form = $this->createForm(UserType::class, $user, [
        "roles" => $this->getParameter("security.role_hierarchy.roles")
    ]);
    $form->handleRequest($request);

    if ($form->isSubmitted() && $form->isValid()) {
        try {
            $entityManager->flush();

            return $this->redirectToRoute('app_admin_user_index', [], Response::HTTP_SEE_OTHER);
            # \Exception équivaut à mettre en haut du fichier use Exception;
        } catch (\Exception $e) {
            $this->addFlash("error", "Une erreur est survenur lors de la modification de l'utilisateur");
            return $this->redirectToRoute('app_admin_user_edit');
        }
    }

    return $this->render('admin_user/edit.html.twig', [
        'user' => $user,
        'form' => $form,
    ]);
}
```

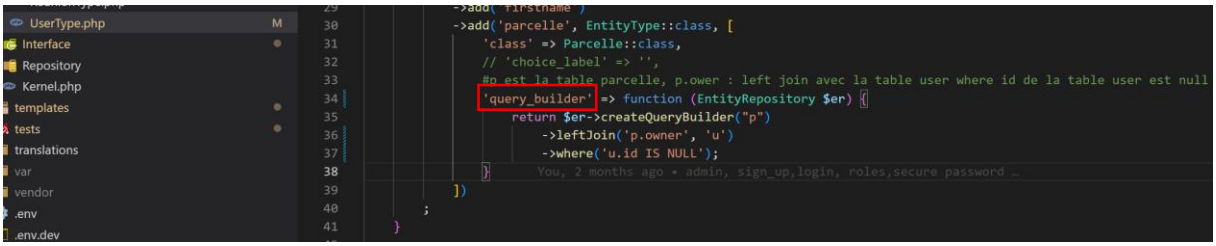
Pour l'affichage des rôles, on les récupère via un Enum :



À la soumission du formulaire les rôles sont modifiés en fonction des checkbox cochés :



Pour la parcelle on récupère les parcelles via la configuration du formulaire en spécifiant que le champ est un Entitytype cela va récupérer toutes les parcelles qui ne possèdent pas d'utilisateur associé (grâce à la query_builder) :

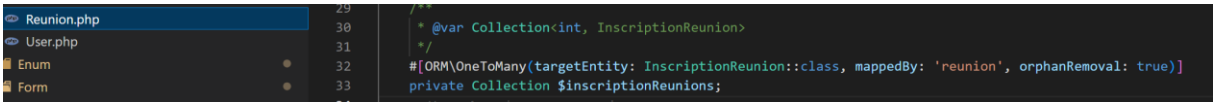


Réunions & Inscriptions

- Création de réunions avec formulaire d'inscription
- Suppression des orphelins (les inscriptions sans réunion)

  Si on supprime une réunion cela supprime aussi toutes les inscriptions associées à cette dernière.

  Cela est définit dans l'entité Reunion sur la propriété InscriptionReunion par orphanRemoval



7. Tests et Sécurité

7.1. Tests

Il faut une base de données test :

```
You, 2 weeks ago | 1 author (You)
# define your env variables for the test env here
KERNEL_CLASS='App\Kernel'
APP_SECRET='$ecretf0rt3st'
SYMFONY_DEPRECATIONS_HELPER=999999
PANTHER_APP_ENV=panther
PANTHER_ERROR_SCREENSHOT_DIR=./var/error-screenshots
DATABASE_URL="mysql://root:@127.0.0.1:3306/jardin_partage_test?serverVersion=8.0.32&charset=utf8mb4"
```

- **Tests unitaires : validation logique (ex : fonction calcul)**

  Test du numéro de la parcelle. Test que le setter récupère bien la bonne valeur et donc que la logique fonctionne correctement.

```php
class ParcelleControllerTest extends WebTestCase
{
    public function testIndex()
    {
        /** @var ParcelleRepository&\PHPUnit\Framework\MockObject\MockObject $parcelleRepository */
        $parcelleRepository = $this->createMock(ParcelleRepository::class);
        $tokenStorage = $this->createMock(TokenStorageInterface::class);
        /** @var User&\PHPUnit\Framework\MockObject\MockObject $user */
        $user = $this->createMock(User::class);
        $token = $this->createMock(TokenInterface::class);
        $parcelle = new Parcelle(); // Créez un objet Parcelle pour le test
        $parcelle->setNumber(1);
        $parcelle->setSize(100);
        $parcelle->setCreatedAt(new \DateTimeImmutable());
        $parcelle->setOwner($user);

        $tokenStorage->method('getToken')->willReturn($token);
        $token->method('getUser')->willReturn($user);
        $user->method('getParcelle')->willReturn($parcelle);
        $user->method('getFullname')->willReturn('John Doe');

        $controller = new ParcelleController();
        $container = $this->getContainer();
        $container->set('security.token_storage', $tokenStorage);
        $controller->setContainer($container);

        $response = $controller->index($parcelleRepository);

        $this->assertInstanceOf(Response::class, $response);
        $this->assertStringContainsString('Ma parcelle', $response->getContent());
    }
}
```

- **Tests fonctionnels : comportement global**

  Tester quand l'utilisateur n'est pas authentifié s'il est bien redirigé sur la page login :

```php
public function testIndexRedirectWhenNotAuthenticated(): void
{
    // Envoi d'une requête GET à la route /parcelle sans être authentifié
    $this->client->request('GET', '/parcelle');

    // Vérification que l'utilisateur est redirigé vers la page de connexion
    $this->assertResponseRedirects('/login');
}
```

- **Test d'intégration**

  test la page quand un utilisateur a une parcelle :

```php
public function testIndexWhenUserHasParcelle(): void
{
    // Créer un utilisateur
    $user = new User();
    $user->setEmail('test@example.com');
    $user->setPassword(self::$passwordHasher->hashPassword($user, 'password'));
    $user->setFirstname('John');
    $user->setLastname('Doe');
    $user->setRoles(['ROLE_USER']);

    // Créer une parcelle
    $parcelle = new Parcelle();
    $parcelle->setNumber(1);
    $parcelle->setSize(100);
    $parcelle->setCreatedAt(new \DateTimeImmutable());
    $parcelle->setOwner($user);

    // Sauvegarder en base de données
    self::$entityManager->persist($user);
    self::$entityManager->persist($parcelle);
    self::$entityManager->flush();

    // Connecter l'utilisateur
    self::$client->loginUser($user);

    // Faire la requête
    self::$client->request('GET', '/parcelle');

    // Vérifier la réponse
    $this->assertResponseIsSuccessful();
    $this->assertSelectorTextContains('h2', 'Ma parcelle');
    $this->assertSelectorTextContains('.card', 'Propriétaire: John Doe');
    $this->assertSelectorTextContains('.card', 'Superficie: 100 m²');
}
```

Pour exécuter les tests :

pour exécuter seulement les test unitaires :

php bin/phpunit tests/Units

```
PHPUnit 9.6.22 by Sebastian Bergmann and contributors.

Testing C:\Users\Utilisateur\Documents\SIO\BTS 2025\Projets 2025\JardinPartage\tests\Units
.......                                                             7 / 7 (100%)

Time: 00:00.658, Memory: 26.00 MB

OK (7 tests, 14 assertions)
```

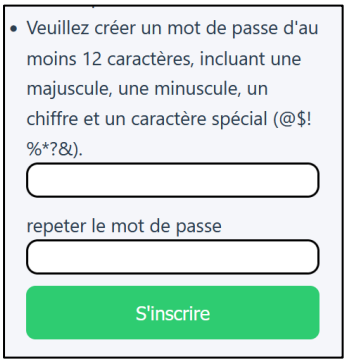
Pour exécuter tous les tests :

Php bin/phpunit

```
PHPUnit 9.6.22 by Sebastian Bergmann and contributors.

Testing
.........                                                         9 / 9 (100%)

Time: 00:01.231, Memory: 42.00 MB

OK (9 tests, 24 assertions)
```

7.2 Sécurité

- Mot de passe fort. Comme indiqué plus haut une regex impose un mot de passe fort à l'inscription



- CSRF Token sur formulaires

  Lors des suppressions ont utilise les tokens CSRF pour empêcher la suppression depuis des actions provenant de sites externes :

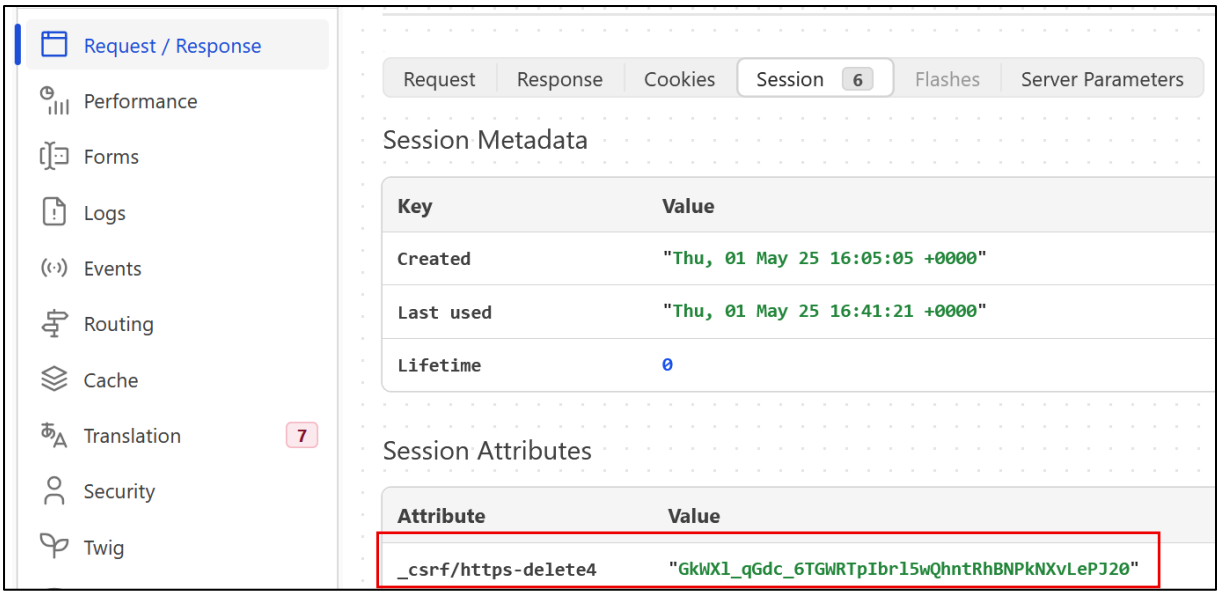
  On compare le token soumis dans le formulaire de suppression et celui en session :

  Côté formulaire

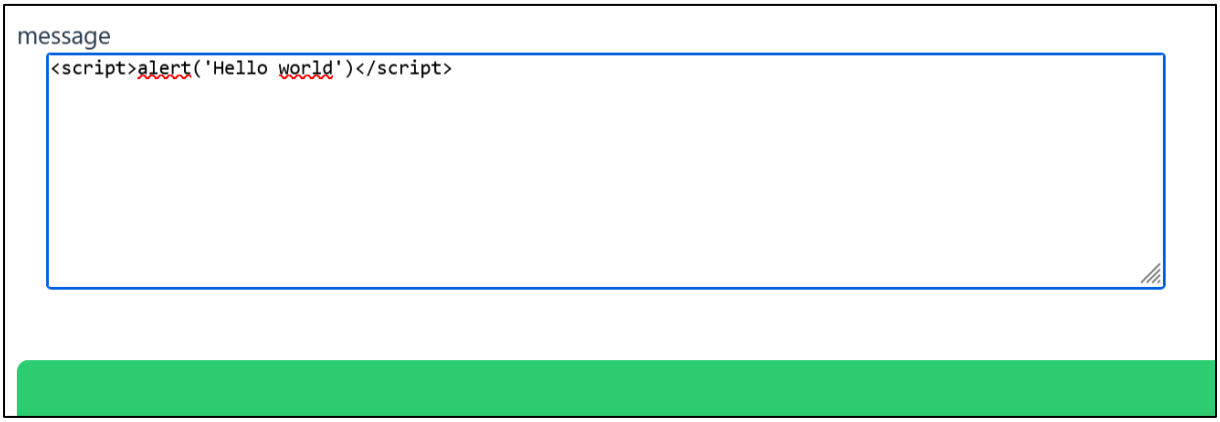
```
<form method="post" action="{{ path('app_admin_plante_delete', {'id': plante.id}) }}" onsubmit="return confirm('Are you sure you want to delete this item?')
    <input type="hidden" name="_token" value="{{ csrf_token('delete' ~ plante.id) }}">      You, 2 months ago • admin, sign_up,login, roles,secure password
    <button class="btn">Supprimer</button>
</form>
```

  Côté session,

  Symfony génère ce token lors de la génération de la page visible dans le profiler

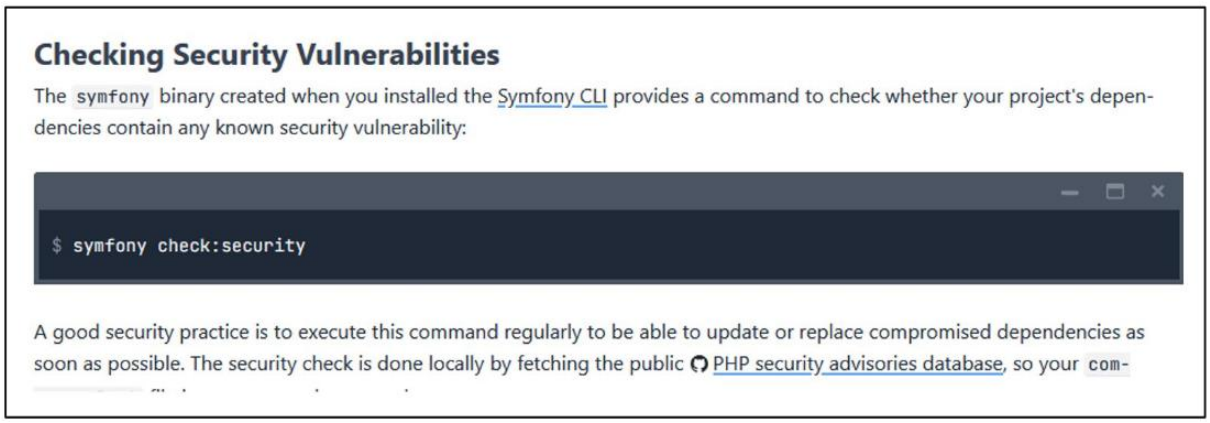


- XSS protection



Par défaut, Symfony empêche les injection XSS (code malvaillant pouvoir être exécuté par le navigateur).

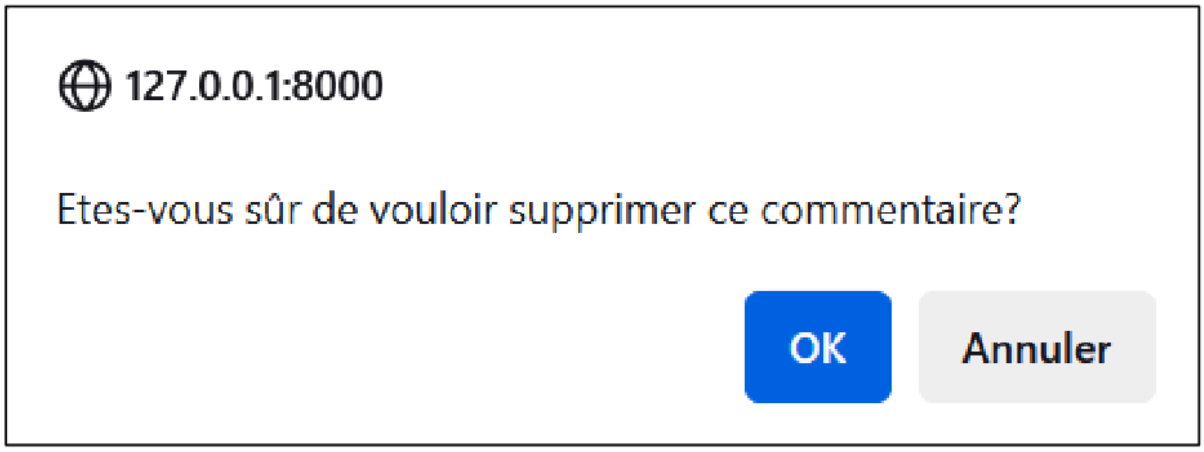
- Vérification bundle de sécurité Symfony (check :security)
  1fois par semaine environ faire la commande pour vérifier les vulnérabilités de symfony.



Et faire un composer update pour mettre régulièrement à jour les dépendances PHP.

8. Front-End & Expérience Utilisateur

- Utilisation de Twig et app.css.
- Formulaires stylisés et centrés.
- Feedback utilisateur (addFlash).



- Affichage conditionnel en fonction du rôle ou de la connexion :
  Voir les captures d'écran navigation (1.3).
  → Quand admin le tableau de bord s'affiche, quand aaa@a.com peut modifier ses messages mais pas ceux de bbb@b.com etc...
- Affichage d'images :
  Dans home/index.html.twig :

```
{% if post.imageName %}
    <div
        class="card-image">
        {# Va aller chercher l'image à l'intérieur de l'objet post #}

        <img src="{{ vich_uploader_asset(post) }}" alt="image">
    </div>
{% endif %}
<hr>
```

9. Difficultés Rencontrées

- Formulaires Symfony (ajout manuel des champs)
- Séparation logique back et front