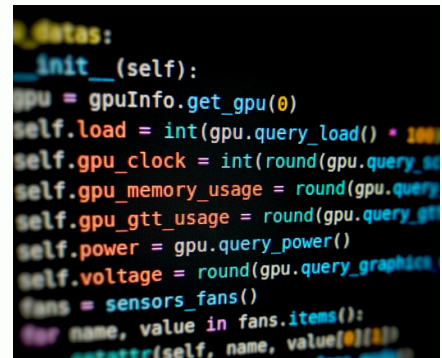


CAHIER DES CHARGES



Projet Partinoire



DC DEV8
Monchablon Hugo

SOMMAIRE

1. Présentation

Présentation Contexte Uber Clean	1-2
----------------------------------	-----

2. Charte Graphique

Présentation de la charte Graphique	3
-------------------------------------	---

3. Partie Technique

Langages et Logiciels utilisés:	4-7
Avantages des Langages et Logiciels utilisés:	4-7
Base de données:	8
Diagram de classe :	9-11

4. Présentation de l'application

Présentation des fonctionnalités de l'application:	12
Interaction de l'application avec ses utilisateurs : Customer	13
Profile Customer	14
Interaction de l'application avec ses utilisateurs : Cleaner	15
Profile Cleaner	16
Page de Services	17
Page de Ménages Party	18
Page de Ménages Party d'un Customer	19

5. Code Review

Page d'inscription:	20-22
Modification de profile	23-24

6. Sécurité

Les failles CSRF en symfony	25
-----------------------------	----

7. Extrait de documentation symfony

Faire le user en symfony :	26-27
----------------------------	-------

8. Annexes et ressources utilisés

28

1 - Présentation

Faire la cuisine, étendre le linge, s'occuper des enfants... les tâches ménagères nous prennent beaucoup de temps. Nous y avons passé, en moyenne, 2 heures et 7 minutes par jour en 2010, selon une étude de l'Insee. Sur l'ensemble de l'année, cela équivaut à plus d'un mois de travail (31,9 jours). Les tâches ménagères nous prennent beaucoup de temps 1/12mois c'est déjà trop, c'est pourquoi Uber Clean a été créé.

Uber Clean est une application web qui permettra à ses futurs utilisateurs de pouvoir partager des services autour des tâches quotidiennes.

Le but principal de Uber Clean est d'aider les gens qui n'ont pas forcément le temps ou l'envie de faire les tâches de la vie quotidienne de leurs permettent de libérés du temps pour profité de leurs familles ou encore de leurs loisirs par exemple.

Nous sommes aujourd'hui malheureusement dans une société où on a plus le temps pour profité de sa famille, de ses loisirs et de profité de la vie sans avoir à toujours se préoccupé du travail et des tâches ménagères par exemple.



Pas le temps ou l'envie de faire les fastidieuses tâches de la vie quotidienne ? N'attendez plus Uber Clean est là pour vous aidé !

1 - Présentation

Uber Clean n'a pas seulement pour but d'aider les gens à se soulager des tâches de la vie quotidienne mais vise aussi à aider les personnes cherchant un moyen de rémunération sans qualification, qui permet de choisir et d'organiser un emploi du temps en fonction de ses envies et de la demande.

Par exemple un étudiant cherchant un petit travail à côté des cours qui lui permettra de ne pas gêner et interférer avec sa scolarité et aussi qui lui permettra de se faire rémunérer contre des services avec d'autres utilisateurs ce qui pourrait l'aider dans sa vie étudiante, et inversement pour par exemple une personne gagnant bien sa vie, qui n'aura pas le temps de faire les tâches ménagères et qui cherche à s'en soulager Uber Clean sera pour lui une solution tout comme pour l'étudiant.

De même, les mères au foyer ou les parents qui cherchent à travailler tout en s'occupant de leurs enfants peuvent trouver une solution adaptée avec Uber Clean. Ils pourraient organiser leur temps de travail autour de leur emploi du temps familial et éviter ainsi les coûts supplémentaires de la garde d'enfants.

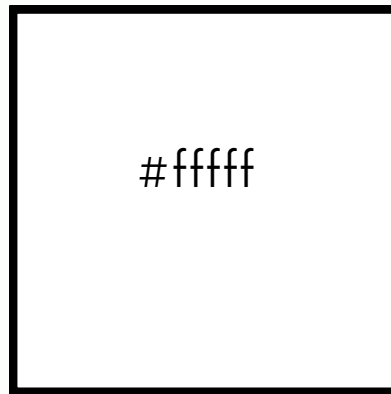
Enfin, Uber Clean pourrait également offrir des solutions pour les personnes vivant dans des zones éloignées ou rurales, où il peut être difficile de trouver un emploi disponible localement. En leur permettant de travailler à distance et de se connecter avec les utilisateurs locaux, Uber Clean pourrait aider à stimuler l'emploi dans ces zones et à offrir des opportunités de travail flexibles et adaptées à leurs besoins.

En effet, peu importe l'endroit où l'on se trouve tant qu'il y a des foyers il y aura des tâches ménagères.

En résumé, Uber Clean pourrait offrir des solutions pour les personnes cherchant un moyen de rémunération flexible et adapté à leur situation personnelle. Que ce soit pour des étudiants, des parents, ou des personnes vivant dans des zones éloignées, Uber Clean pourrait offrir des opportunités de travail adaptées et offrir une plus grande flexibilité dans l'organisation de leur emploi du temps.

2 - Charte Graphique

Couleurs



Logo



Typographie

Nunito

3- Partie technique

Langages et Logiciels utilisés:

PHP + Framework Symfony
CSS + Framework Tailwind.css

Symfony est un Framework PHP basé sur le modèle MVC
(Modèle Vue Contrôleur)

Les Controllers sont basés sur PHP.
Les Vues sont basés sur Twig (avec tailwind.css)
les Modèles sont basés sur le SQL pour le traitement logique des données.

J'ai utilisé Github pour pouvoir versionné mon code.



3- Partie technique

Avantages des Langages et Logiciels utilisés:

- PHP et Symfony sont gratuit.
- PHP et Symfony possèdent beaucoup de documentation et d'utilisateurs ce qui permet d'avoir beaucoup de ressources à disposition.
- Symfony permet une bonne organisation du code
- Symfony est open source
- Symfony est un des plus puissants Framework à ce jour.
- Permet de ne pas perdre du temps sur des tâches fastidieuses tels que la création de la base de données avec ses relations.
- Doctrine est un Orm simple d'utilisation
- Tailwind Css est un très bon Framework qui nécessite des bases en Css mais qui permet de rester libre dans le choix de son style.



3- Partie technique

Logiciels utilisés:

- PHP Storm
- Mysql Workbench
- Terminal Bash
- Wamp
- Adobe XD
- Photoshop
- Trello (<https://trello.com/b/XHqTomZe/uberclean>)

Avantages des logiciels utilisés:

- PHP Storm très pratique pour la prise en charge et l'intégration du Framework Symfony, bonne auto compression.
- Mysql Workbench bonne interface reste simple d'utilisation pour du debug ou de la gestion de DB.
- Bash pour interagir avec le projet et le repo principalement.
- Wamp pour faire tourner le sql. (aucune utilisation du localhost de wamp je passe pas le serveur php)
- Adobe XD pour faire les maquettes du site
- Photoshop pour faire le Logo du site principalement.

3- Partie technique



Logiciels utilisés:

Trello est une application mobile très pratique il permet de s'organiser dans ses tâches.

Il m'a été utile pour voir par quelles tâches je devais commencer (en définissant un ordre d'importance).

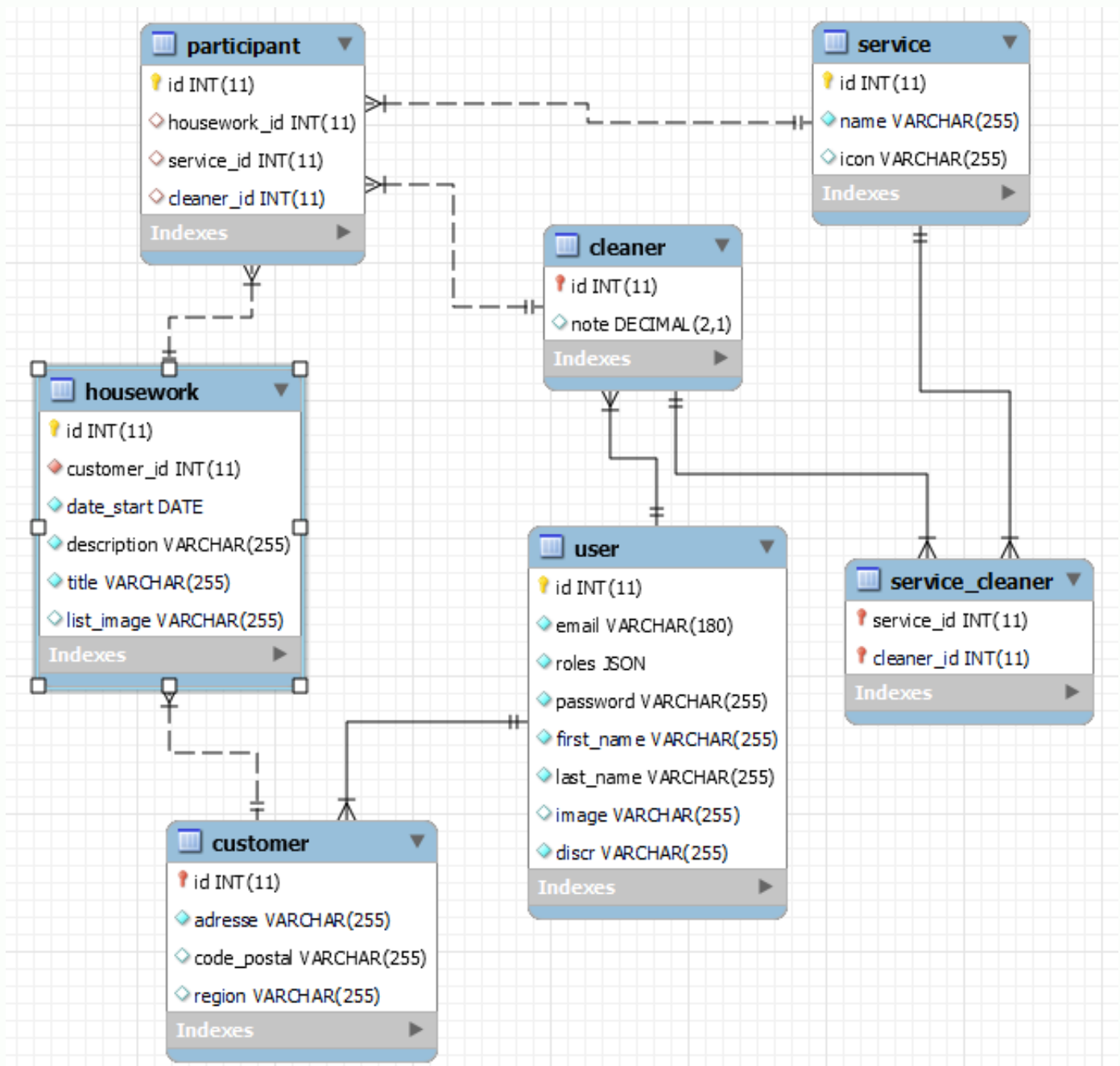
J'ai utilisé le terminal bash pour la plupart des commandes à réaliser comme lancer le serveur php, lancer les commandes npm et composer permettant l'installation des paquets au sein de symfony.

```
"scripts": {  
  "server": "php -S localhost:8000 -t public",  
  "reload-db": "php bin/console doctrine:database:drop --force && php bin/console doctrine:database:create && php bin/console doctrine:mig",  
  "dev-server": "encore dev-server",  
  "dev": "encore dev",  
  "watch": "encore dev --watch",  
  "build": "encore production --progress",  
  "push": "git add . && git commit -m $1 && git push"  
},
```

Par exemple dans mon package.json j'ai créé des scripts personnalisés pour gagner du temps de production comme la commande "server" permettant de lancer le serveur php sur le port 8000, ou encore le script "reload-db" qui me permet de reconstruire ma base de données en faisant un "npm run reload-db". J'aime aussi utilisé le terminal bash pour gérer mon repository github.

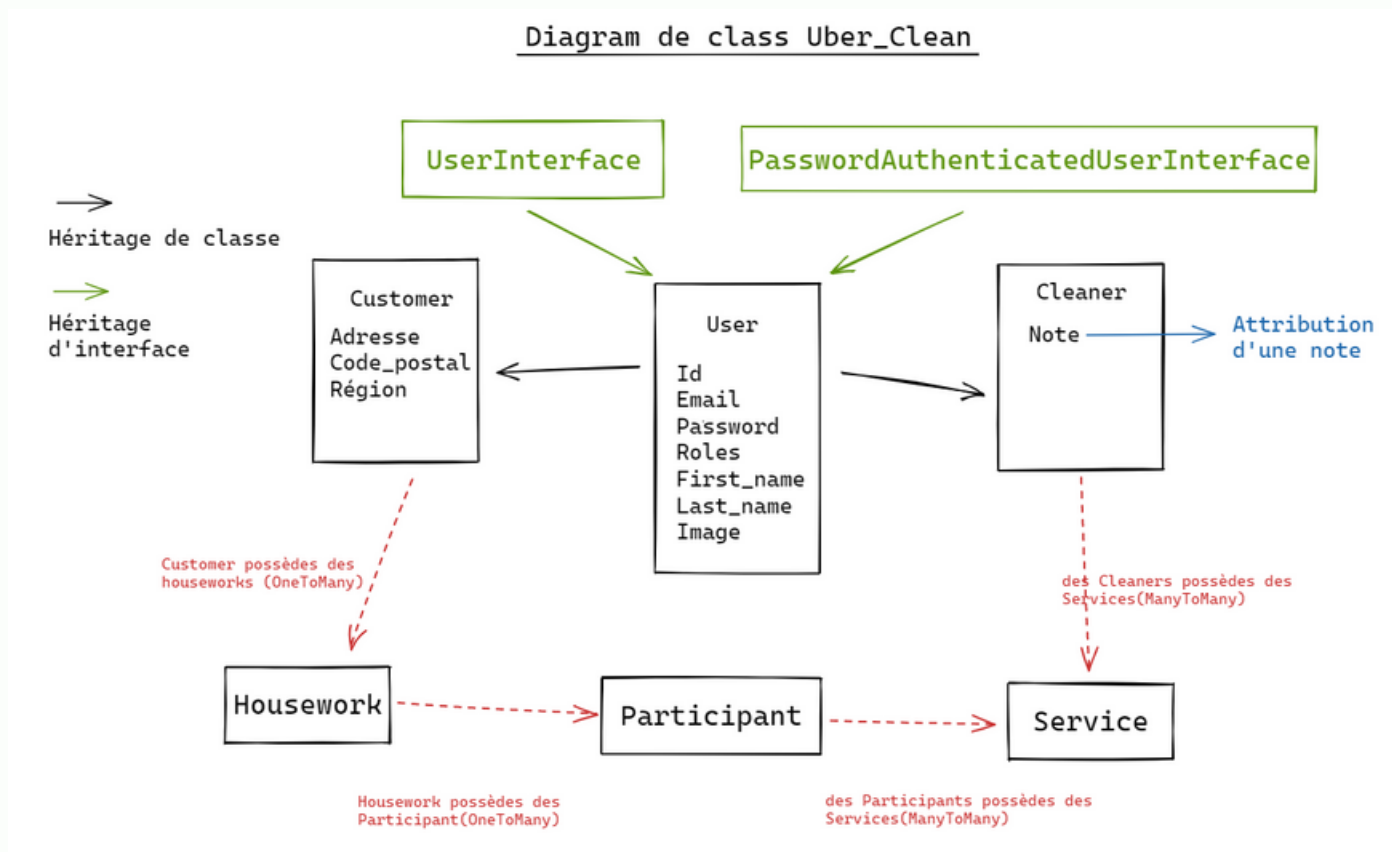
3- Partie technique

Base de données:



3- Partie technique

Diagram de classe :



Dans mon application l'entité User hérite de 2 classes de Symfony : "UserInterface" et la class "PasswordAuthenticatedUserInterface" fournie une méthode commune pour récupérer le mot de passe de l'utilisateur lors de l'authentification.

Cette interface peut être utilisée pour les utilisateurs qui stockent leur mot de passe en clair dans la base de données.

3- Partie technique

Diagram de classe :

Lorsqu'on implémente l'interface `PasswordAuthenticatedUserInterface` dans une classe enfant comme la classe `User` par exemple, cela nous permet d'avoir accès à la méthode `getPassword()` qui permet de récupérer en clair le mot de passe entré par l'utilisateur.

Il convient de noter que stocker les mots de passe en clair en base de données est une pratique de sécurité dangereuse, en effet aujourd'hui on passe par des hachages pour stocker les mots de passes des utilisateurs pour avoir une manière bien plus sécurisée que seulement entrer les mots de passes en clair dans la base.

Symfony utilise plusieurs algorithmes de hachage sécurisés tel que Bcrypt et Argon2 qui sont 2 hachage très utilisé de nos jours.



3- Partie technique

Diagram de classe :

Les classes "Customer" et "Cleaner" héritent de la classe "User", en effet pour l'application je suis parti sur l'optique de créer 2 types de comptes :

- Customer
- Cleaner

Je pars du principe qu'un utilisateur qui propose ses services de tâches ménagères ne va pas non plus proposer que l'on vienne faire ces mêmes tâches chez lui, et inversement pour l'utilisateur qui cherche à se libérer de ses tâches de la vie quotidienne, il ne va pas proposer ses services pour le faire chez les autres.

C'est pour ces raisons que j'ai décidé de séparer les 2 types d'entités (Cleaner, Customer) dans 2 classes bien distinctes.

Ensuite, les Customers pourront créer des Houseworks ou encore des Ménages Party qui sont simplement des événements que les Customers pourront créer pour que les Cleaners puissent s'inscrire dessus.

4- Présentation de l'application

Présentation des fonctionnalités de l'application:

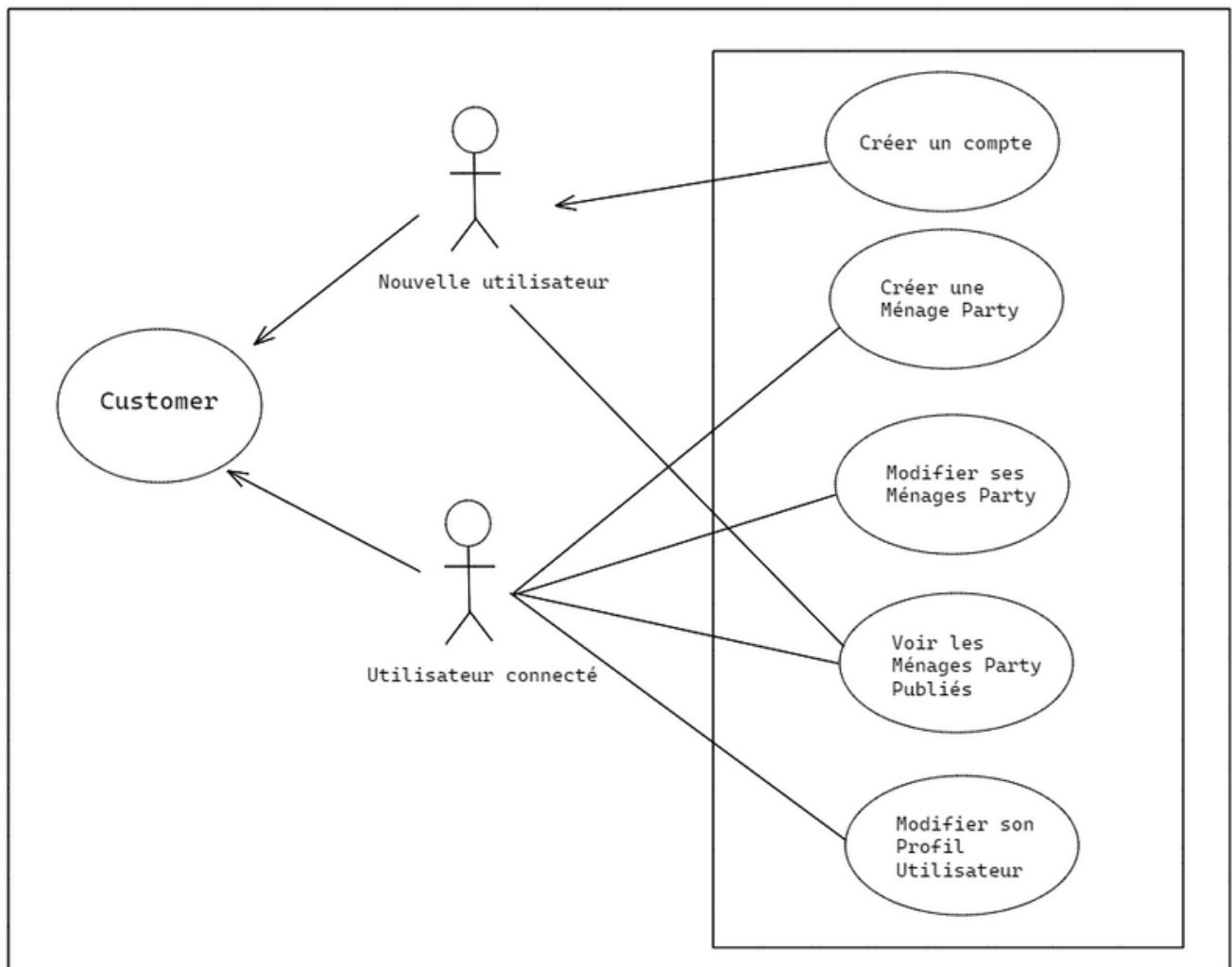
Uber Clean a pour objectif de répondre à la problématique développée précédemment, c'est à dire être en mesure de fournir les fonctionnalités suivantes :

- Créer un compte Customer pour posté des Ménages Party
- Créer un compte Cleaner pour s'inscrire à des Ménages Party
- Profile différent pour chaque utilisateur (Cleaner/Customer)
- Customer: Accès aux Ménages Party publiés.
- Modification du profile et des informations relatives au type de compte (Cleaner/Customer)
- Possibilité de voir les profils des autres utilisateurs
- Possibilité d'ajouter des notes d'évaluation sur les services d'un Cleaner

4- Présentation de l'application

Interaction de l'application avec ses utilisateurs :
Customer

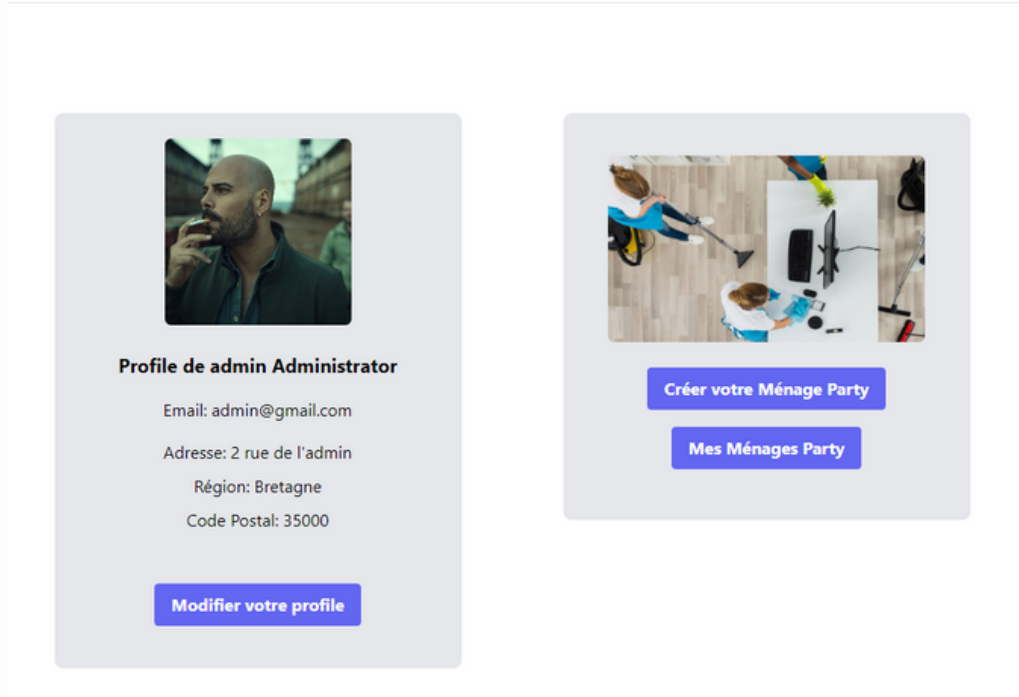
Diagramme d'utilisation d'un Customer:



4- Présentation de l'application

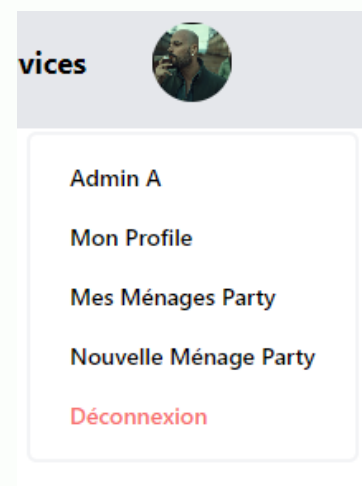
Profile

Profile d'un Customer



- Possibilité de voir les informations relatives au compte connectés
- Possibilité de modifier les informations du compte
- Possibilité de créer des Ménages Party
- Possibilité de voir ses Ménages Party créées.

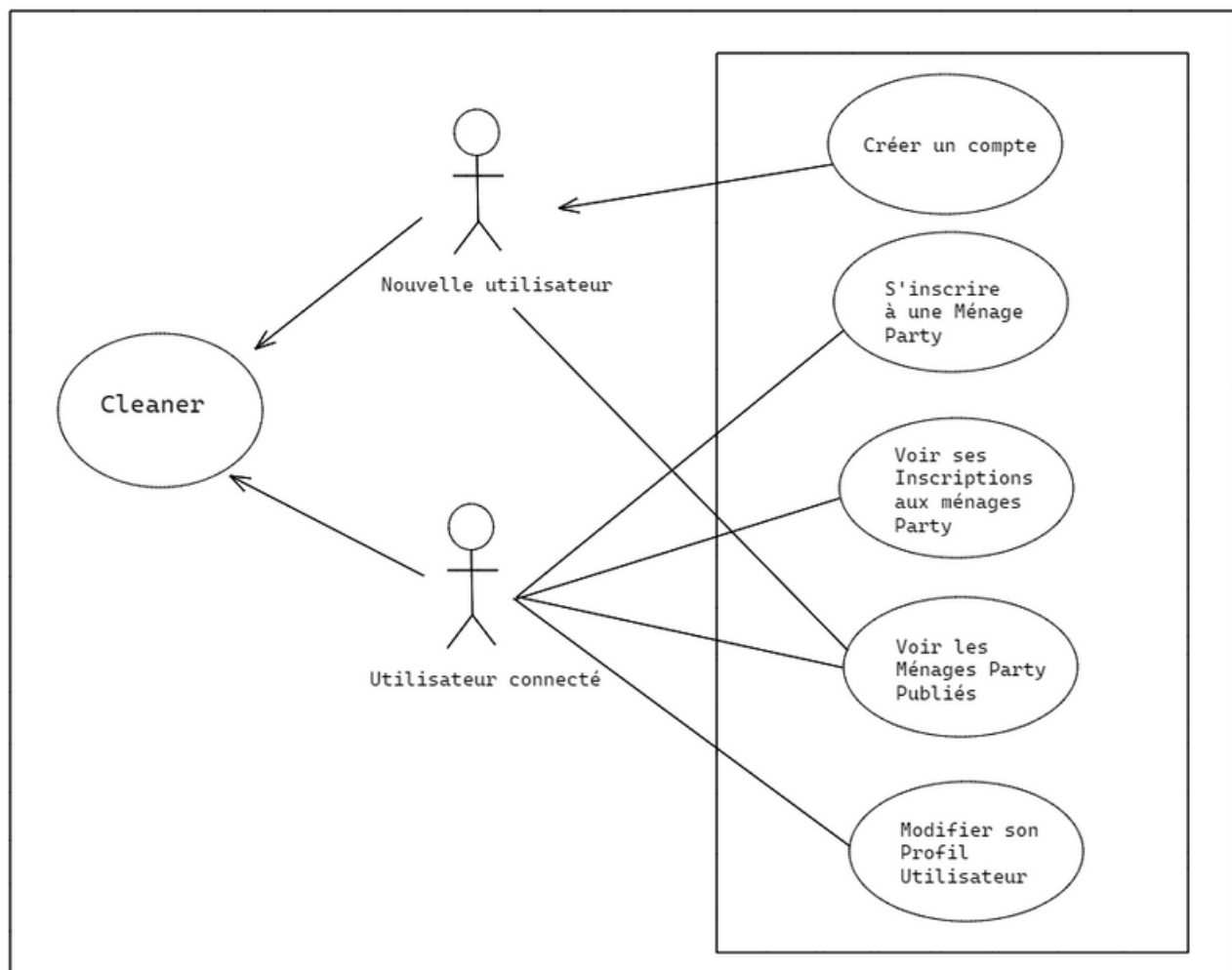
- Menu déroulant au niveau de la barre navigation permettant de cliquer sur l'image de profile de l'utilisateur et de naviguer dans les différentes sections tels que:
- Voir le profile utilisateur
- Voir les Ménages Party de l'utilisateur
- Créer de nouvelles Ménages Party
- Un bouton déconnexion pour se déconnecter



4- Présentation de l'application

Interaction de l'application avec ses utilisateurs :
Cleaner

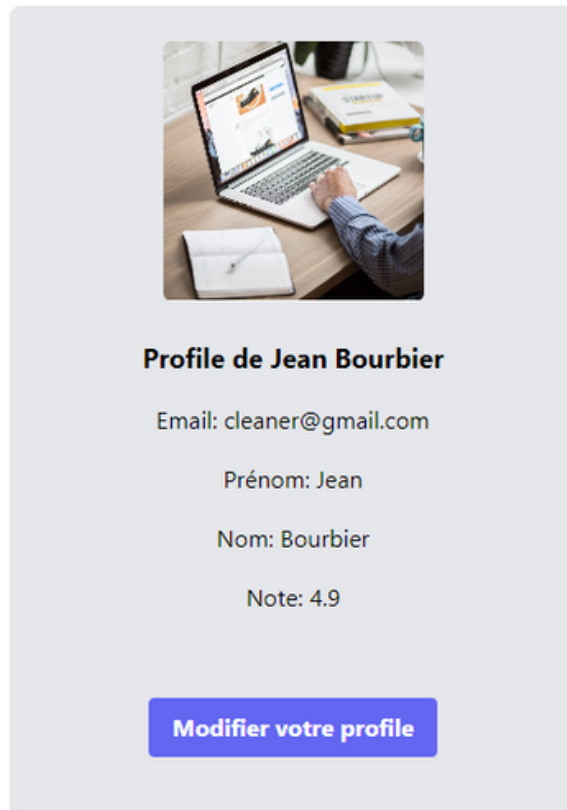
Diagramme d'utilisation d'un Cleaner:



4- Présentation de l'application

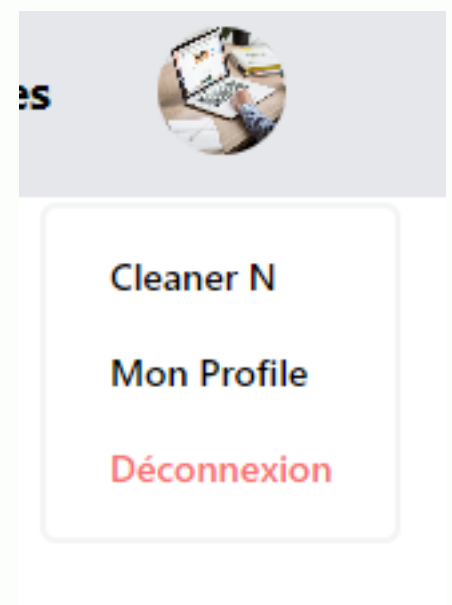
Profile

Profile d'un Cleaner



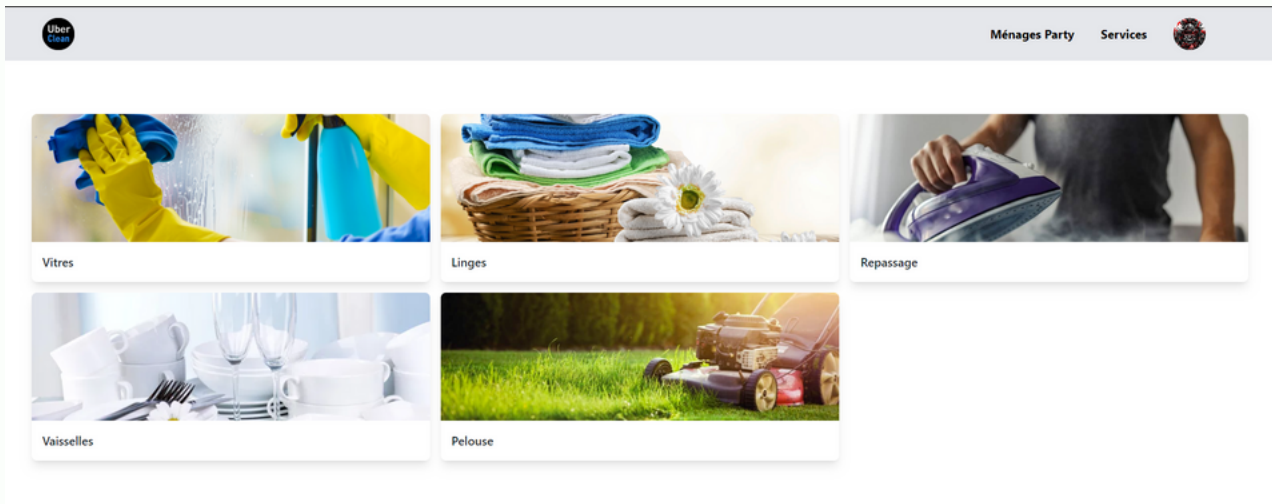
- Possibilité de voir les informations relatives au compte connectés
- Possibilité de modifier les informations du compte

- Menu déroulant au niveau de la barre navigation permettant de cliquer sur l'image de profile de l'utilisateur et de naviguer dans les différentes sections tels que:
- Voir le profile utilisateur
- Un bouton déconnexion pour se déconnecter



4- Présentation de l'application

Page de Services



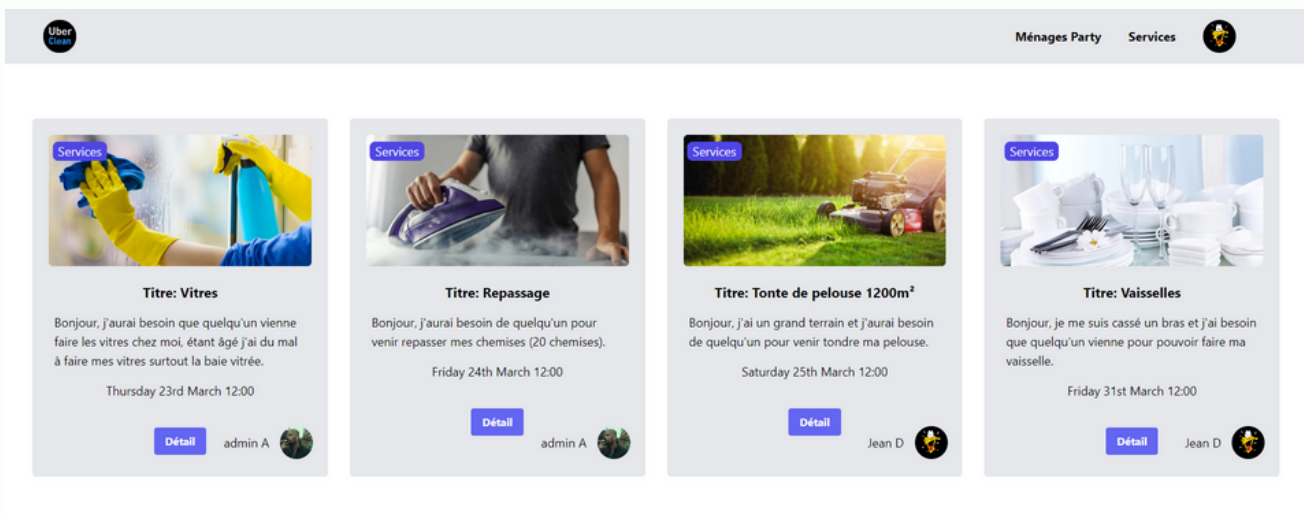
Sur le page de Services on peut voir la listes des services que proposent l'application tels que:

- Les Vitres (nettoyage des vitres)
- Le Linge (nettoyage du linge)
- Le Repassage (repassage du linge)
- La vaisselle (nettoyage de la vaisselles de manières général)
- La pelouse (tonte de la pelouse)

La page services est une page qui pourra voir ses services changés au fil du temps.

4- Présentation de l'application

Page de Ménages Party



Sur le page de Ménages Party on peut voir la listes des ménages party qu'on postés les différents Customers sur le site les ménages party sont affichés avec:

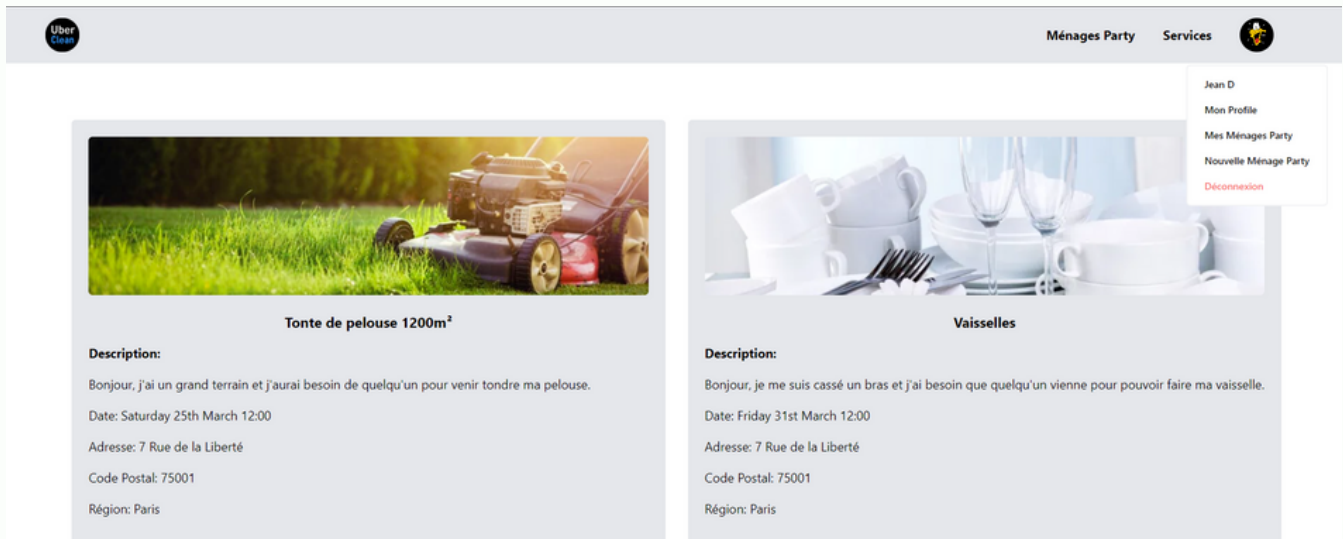
- Un Titre
- Une Description
- Une Date
- Un bouton pour accéder au détail de la ménage party
- Le nom et l'image de profile du créateur de la ménage party

Je compte ajouter le prix à présenter pour la ménage party. Cependant, l'inconvénient de cette question d'argent est qu'il faut trouver un moyen sécurisé pour le paiement, étant donné qu'Uber Clean est basé sur un service entre particuliers. Cela soulève des questions de sécurité, telles que le vol, la dégradation ou encore d'agression par exemple.

Je compte implémenter un système de recherche basé sur les services, c'est à dire qu'un Cleaner pourra à la création de son compte choisir les différents services qu'il souhaite réalisé. C'est services que le Cleaner aura au préalable choisi permettront une recherche filtrée des Ménages party.

4- Présentation de l'application

Page des Ménages Party d'un Customer



La page "Mes Ménages Party" accessible dans le profile d'un Customer permet l'utilisateur de pouvoir voir les Ménages Party qu'il a crée avec notamment :

- Le titre
- La description
- La date
- L'Adresse
- Le code Postal
- La région

La fonctionnalité pour permettre à un Customer de pouvoir supprimé une ménage party sera bientôt ajouté ainsi que la possibilité de pouvoir modifié une ménage party. Prochainement le Customer pourra voir quel Cleaners sont inscrit à ses Ménages Party.

5- Review de codes

Page d'inscription:

La page d'inscription comporte 2 choix:

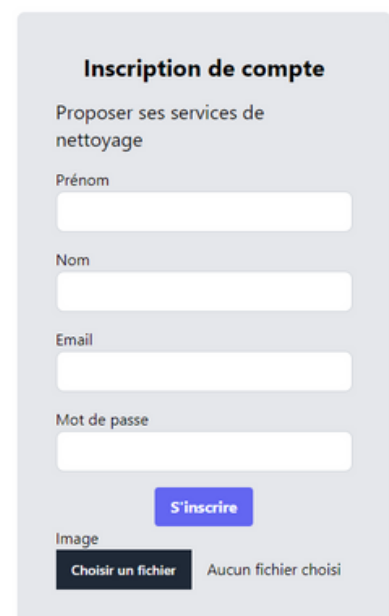
- soit créer un compte client (Customer) pour posté des ménages party par exemple
- soit créer un compte nettoyeur (Cleaner) pour s'inscrire à des ménages party que des clients (Customers) auraient créés.



Ensuite, après avoir fait son choix l'utilisateur tombera sur des pages de connexion différentes :

La première sera la page d'inscription d'un compte Cleaner où il pourra renseigner son Prénom, Nom, Email, mot de passe ainsi qu'une image de profile.

Après avoir cliquer sur le bouton s'inscrire, le nouvelle utilisateur Cleaner sera créé et envoyé en base de données pour pouvoir se connecter à chaque fois qu'il ira sur le site.



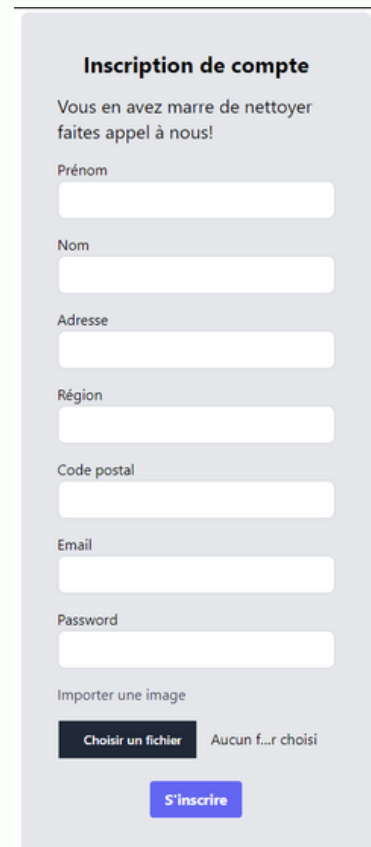
5- Review de codes

Page d'inscription:

Pour créer son compte client (Customer) l'utilisateur tombera sur un formulaire différent. En effet, au prénom, nom, email, mot de passe et à l'image de profile s'ajoutera les champs :

- Adresse
- Région
- Code Postal

Tout comme pour le Cleaner, lorsque l'utilisateur cliquera sur le bouton s'inscrire le compte client (Customer) sera créé et il pourra le réutiliser en se connectant.



```
#[Route('/registration/cleaner', name: 'app_registration_cleaner')]
public function registerCleaner(Request $request, UserPasswordHasherInterface $userP
{
    $user = new Cleaner();
    $form = $this->createForm( type: RegistrationCleanerFormType::class, $user);
    $form->handleRequest($request);
    if ($form->isSubmitted() && $form->isValid()) {
        // encode the plain password
        $user->setPassword(
            $userPasswordHasher->hashPassword(
                $user,
                $form->get('plainPassword')->getData()
            )
        );
    }
}
```

Création d'un nouveau Cleaner.

On crée le formulaire d'inscription grâce au `$form = $this->createForm`

Ensuite, on vérifie si le formulaire est valide et bien rempli (pas d'erreur de champ vide, email bon format etc...).

Ensuite on peut hacher le mot de passe de l'utilisateur avant de l'envoyer en base.

5- Review de codes

Page d'inscription:

```
);  
$image = $form->get('image')->getData();  
if ($image) {  
    $fileName = $fileUploader->upload($image);  
    $user->setImage($fileName);  
}  
$user->setRoles(["ROLE_CLEANER"]);  
$entityManager->persist($user);  
$entityManager->flush();  
return $this->redirectToRoute('route: '/' );  
}  
return $this->render('view: registration/registerCleaner', [  
    'registrationForm' => $form->createView(),  
]);
```

Après avoir hacher le mot de passe on passe à l'upload de l'image grâce à la classe fileUploader qu'on a au préalable passé en paramètre. On va vérifier qu'on a bien récupéré l'image que l'utilisateur a entré dans le formulaire puis on passe à son upload en utilisant la fonction "upload".

```
$image = $form->get('image')->getData();  
if ($image) {  
    $fileName = $fileUploader->upload($image);
```

Ensuite, on définit l'image de l'utilisateur `$user->setImage($fileName);` afin qu'elle soit enregistrée dans son compte utilisateur.

Enfin on définit le rôle du compte "ROLE_CLEANER" et on passe à l'envoi en base grâce à la méthode "persist" de l'entityManager qui va nous permettre d'envoyer l'objet qu'on lui a passé en paramètre. Enfin, on fait un flush puis on redirige l'utilisateur vers la page d'accueil.

La méthode **flush()** de l'EntityManager permet de synchroniser les modifications apportées aux entités avec la base de données. En d'autres termes, elle sert à enregistrer les modifications en attente de toutes les entités gérées par l'EntityManager dans la base de données.

5- Review de codes

Modification de profile :

```
#[Route('/profile/edit', name: 'app_edit_profile')]
public function editProfile(CustomerRepository $customerRepository, CleanerRepository $cleanerRepo
{
    $user = $this->getUser();

    if (!$user || !($user instanceof PasswordAuthenticatedUserInterface)) {
        return $this->redirectToRoute(route: '/login');
    }

    $customer = $customerRepository->findOneBy(['email' => $user->getUserIdentifier()]);
    $cleaner = $cleanerRepository->findOneBy(['email' => $user->getUserIdentifier()]);

    if ($customer) {
        $form = $this->createForm(type: RegistrationCustomerFormType::class, $user);
        $form->handleRequest($request);
    }
    else {
        $form = $this->createForm(type: RegistrationCleanerFormType::class, $user);
        $form->handleRequest($request);
    }

    if ($form->isSubmitted() && $form->isValid()) {
        // encode the plain password

        if ($customer) {
            $customer->setPassword(
                $userPasswordHasher->hashPassword(
                    $user,
                    $form->get('plainPassword')->getData()
                )
            )
        }
    }
}
```

Pour la modification de profile la toute première chose que je fais c'est de bien vérifier si l'utilisateur est bien connecté, ensuite je fais une requête dans la table Customer et Cleaner pour par la suite essayer de définir qu'elle type d'user essaye de modifier son profile pour éviter d'avoir 2 pages différentes avec 2 routes différentes.

Si un Customer est trouvé un formulaire d'inscription de Customer est créé sinon un formulaire de Cleaner est créé. Les Cleaners et les Customers possèdent des champs différents c'est pourquoi j'utilise 2 formulaires différents.

5- Review de codes

Modification de profile :

```
);  
$image = $form->get('image')->getData();  
if ($image) {  
    $fileName = $fileUploader->upload($image);  
    $customer->setImage($fileName);  
}  
$entityManager->persist($customer);  
}  
elseif ($cleaner) {  
    $cleaner->setPassword(  
        $userPasswordHasher->hashPassword(  
            $user,  
            $form->get('plainPassword')->getData()  
        )  
    );  
  
    $image = $form->get('image')->getData();  
    if ($image) {  
        $fileName = $fileUploader->upload($image);  
        $cleaner->setImage($fileName);  
    }  
    $entityManager->persist($cleaner);  
}  
  
$entityManager->flush();
```

Ensuite je récupère les données du formulaire pour les assignés à l'utilisateur connecté puis j'utilise `$entityManager->persist($cleaner);` afin de modifier le profile de l'utilisateur dans la base de données.

6- Sécurité

les Failles CSRF :

```
main:
    lazy: true
    provider: app_user_provider
    form_login:
        login_path: app_login
        check_path: app_login
        enable_csrf: true
    logout:
        path: app_logout
```

"CSRF" signifie Cross-Site Request Forgery, une attaque qui exploite la confiance de l'utilisateur dans une application web en utilisant des requêtes falsifiées pour accéder à des fonctionnalités non autorisées ou pour exécuter des actions malveillantes au nom de l'utilisateur.

Symfony est un framework PHP populaire qui fournit une fonctionnalité de protection CSRF pour aider à prévenir les attaques CSRF. Lorsque l'option "enable_csrf" est définie sur "true" dans le formulaire de connexion, Symfony ajoute un jeton CSRF à chaque demande POST envoyée à ce formulaire. Ce jeton est ensuite vérifié par le serveur pour s'assurer que la demande provient bien de l'utilisateur autorisé.

En résumé, en activant l'option "enable_csrf" pour le formulaire de connexion dans Symfony, vous ajoutez une couche de sécurité supplémentaire pour empêcher les attaques CSRF et assurer la confidentialité et l'intégrité des informations d'identification de l'utilisateur.

7- Extrait de documentation en symfony

Faire le user en symfony :

The User

Permissions in Symfony are always linked to a user object. If you need to secure (parts of) your application, you need to create a user class. This is a class that implements [UserInterface](#). This is often a Doctrine entity, but you can also use a dedicated Security user class.

The easiest way to generate a user class is using the `make:user` command from the [MakerBundle](#):

```
$ php bin/console make:user
The name of the security user class (e.g. User) [User]:
> User

Do you want to store user data in the database (via Doctrine)? (yes/no) [yes]:
> yes

Enter a property name that will be the unique "display" name for the user (e.g. email)
> email

Will this app need to hash/check user passwords? Choose No if passwords are not needed
Does this app need to hash/check user passwords? (yes/no) [yes]:
> yes

created: src/Entity/User.php
created: src/Repository/UserRepository.php
updated: src/Entity/User.php
updated: config/packages/security.yaml
```

7- Extrait de documentation en symfony

Faire le user en symfony :

Au début, j'ai eu du mal pour refaire de tête le système avec le User, je me suis donc rendu sur la documentation de symfony et je suis tombé sur cette page. Cette page explique que les permissions dans symfony sont toujours liés à un objet User et que si on a besoin de sécuriser cette partie de notre application on va avoir besoin de créer une Classe User.

La documentation explique ensuite que cette classe implémente l'interface "ManagerInterface" de symfony, c'est une entité de Doctrine mais on nous dit aussi qu'on peut utiliser une classe "security user" à la place.

Ensuite, ils nous expliquent que le moyen le plus simple de générer une classe utilisateur et d'utiliser la commande "make:user" provenant de MakerBundle.

Ensuite sur la documentation on nous montre les commandes à utiliser pour ce faire. Tout d'abord il utilise la commande "php bin/console make:user" cette commande nous demande de rentrer le nom de la classe de sécurité de l'utilisateur qui est par défaut "User".

On nous demande si on veut stocker les données utilisateurs dans la base de données avec Doctrine, puis on doit renseigner le nom des propriétés qu'on veut ajouter à notre utilisateur, dans l'exemple il ajoute "email". Ensuite il est demandé si l'application a besoin de hasher ou de vérifier le mot de passe des utilisateurs.

Enfin, la commande se termine en créant plusieurs fichiers :

```
src/Entity/User.php  
src/Repository/UserRepository.php  
src/Entity/User.php  
config/packages/security.yaml
```

8- Annexes et ressources utilisés

