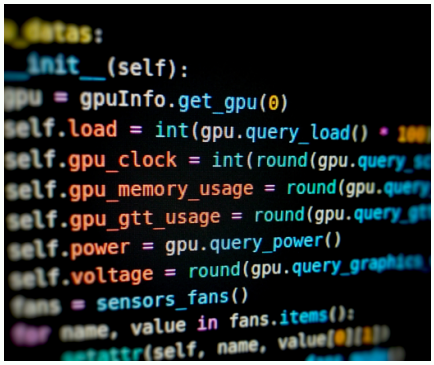
CAHIER DES CHARGES

Projet Partinoire



DC DEV8
Monchablon Hugo

StuD
CleaN

SOMMAIRE

StuD
CleaN

SOMMAIRE

1 - Présentation

Présentation Contexte Stud Clean

Faire la vaisselle, étendre le linge, repasser ses affaires... 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, 1/12 d'une année, c'est déjà trop, c'est pourquoi Stud Clean a été créé.

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

Le but principal de Stud 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 leur permettre de libérer du temps pour profiter de leurs familles ou encore de leurs loisirs.

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

Stud Clean est aussi destiné aux étudiants, en effet, seuls les étudiants pourront s'inscrire en tant que Cleaner (les utilisateurs qui pourront réaliser des prestations). En France la vie reste cher et c'est encore plus le cas pour les étudiants qui pour beaucoup doivent trouver un travail à côté de leurs études le week-end par exemple pour subvenir à leurs besoins, c'est pourquoi Stud Clean à en partie était créé, Stud Clean permettra aux étudiants de pouvoir se faire un peu d'argent quand ils le souhaitent et pouvoir manipuler leurs emplois du temps comme ils le désirent. Sur Stud Clean, c'est utilisateur de fixé le prix d'une prestation puisque Stud Clean n'a pas pour but de rentrer en concurrence avec des professionnelles du métier (femme de ménage) mais a pour but d'aider les étudiants en rendant des services rémunérer à d'autres utilisateurs.

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

1 - Présentation

Analyse de la concurrence

La plupart des sites internet permettant de mettre en relation des utilisateurs avec des personnes pour réaliser des prestations de la vie quotidienne demandent de réaliser des devis avec des professionnels. Sur ces sites, c'est à l'utilisateur de trouver une personne, généralement une femme de ménage. La plupart des utilisateurs demandent des certifications, et les personnes réalisant les prestations sont des professionnelles du métier de femme de ménage.

Sur mon site, les utilisateurs n'ont qu'à créer un événement (une "ménage party") et attendre que quelqu'un s'inscrive plus besoin de chercher la personne. Mon application serait destinée principalement aux étudiants, ce sont eux qui effectueront les prestations. Étant donné qu'un étudiant n'a pas forcément une formation professionnelle dans ce domaine, les utilisateurs sont ceux qui fixent les prix, et les étudiants (les "cleaners") sont libres de s'inscrire ou non.

Les sites proposant des services de ménage sont pour la plupart des sites qui proposent également des services plus variés, tels que le site La plupart des sites internet permettant de mettre en relation des utilisateurs avec des personnes pour réaliser des prestations de la vie quotidienne demande de réaliser des devis avec des professionnels, sur ces sites c'est au utilisateur d'aller trouver une personne (femme de ménage dans la plupart des cas).
La plupart des utilisateurs demandent des certifications et les personnes réalisant les prestations sont des personnes qui sont du métier (femme de ménage).

Sur mon site les utilisateurs ont juste à créer un évènement (ménage party) et attendre que quelqu'un s'inscrivent plus besoin de chercher la personne.

1 - Présentation

Analyse de la concurrence

Mon application serait destinée principalement pour les étudiants, en tout cas les étudiants seront ceux qui feront les prestations, forcément un étudiant n'est pas du métier, c'est pourquoi les utilisateurs sont ceux qui fixent les prix et les étudiants (Cleaner) sont libres de s'inscrire ou non.

Les sites proposant des services de ménages sont pour la plupart des sites qui proposent aussi des services plus vagues comme le site https://www.allovoisins.com/ qui propose certes des services de ménages (femme de ménage) mais qui propose aussi des services beaucoup plus généraux en passant des services de plomberies ou encore d'électricités. , qui propose non seulement des services de ménage (femme de ménage), mais aussi des services beaucoup plus généraux, comme la plomberie ou l'électricité.

1 - Présentation

Planning prévisionnel

Tâches à réaliser	Durée	Mars	Avril	Mai
Initialisation du projet Symfony/Tailwind.css/Twig	2j			Temps réservé pour prévoir d'éventuel bug problème durant le développement du projet Temps réservé pour la rédaction des différents éléments à rendre pour le passage de titre le 16 Juin
Création de la base de données	4j			
Création des liens entre les tables dans Symfony	3j			
Création de la navbar du site	2j			
Création du système d'inscription (connexion, inscription, déconnexion, formulaire, type de compte)	7j			
Création des pages de profils des différents utilisateurs	4j			
Création du système de ménage party	2j			
Création système de modification de compte	1j			
Création menu déroulant de la navbar (information sur l'utilisateur)	1j			
Création du footer du site	1j			
Système de modification de ménage party	3j			
Système d'inscription d'un cleaner à une ménage party	2j			
Système de vérification de compte par l'admin du site	2j			
Ajout du champ et du système de justificatif étudiant	1j			
Ajustement du style du site (Front-end)	4j			
Ajout système de modification du mot de passe suite à la vérification du mot de passe actuel du compte	2j			

2 - Charte Graphique

Couleurs

#6366F1 indigo 700	#fffff white	#E5E7EB gray 200

Logo

Typographie

Nunito, Sans Serif
Classe tailwind:
font-nunito et font-serif

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 ils permettent de réaliser les interactions du back-end (connexion utilisateur etc).
Les Vues sont basées sur Twig (j'utilise le framework tailwind.css pour le style des pages)
les Modèles sont basés sur le SQL pour le traitement logique des données.

J'ai utilisé GitHub pour pouvoir versionner mon code.

3- Partie technique

Avantages des Langages et Logiciels utilisés :

- 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 (Controllers, Templates, dossier pour les formulaires), tout est organisé sous forme de dossier.

- 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 telles que la création de la base de données avec ses relations.

- Doctrine est un Orm simple d'utilisation (ORM est un acronyme pour Object-Relational Mapper, ils permettent d'effectuer un mapping, une liaison entre le monde relationnel et le monde des objets)

- 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, tailwind fonctionne avec un système de classes utilitaires ce qui contrairement à BootStrap permet d'être plus libre dans le choix de son style.

```
<div class="flex items-center justify-center">
    <div class="mt-10 p-6 grid grid-cols-2">
```

Exemple : mt-10 permet d'ajouter un margin-top de 10px, p-6 pour ajouter un padding de 6px

3- Partie technique

Logiciels utilisés :

- PHP Storm
- MySQL Workbench
- Terminal Bash
- Wamp
- Adobe XD
- Illustrator
- Trello (https://trello.com/invite/b/XHqTomZe/ATTIacf7a27e69fd073718c032dd7700b6e32203BAF9/studclean)

Avantages des logiciels utilisés :

- PHP Storm est très pratique pour la prise en charge et l'intégration du Framework Symfony, il permet une bonne auto compression lors de la saisie du code.

- MySQL Workbench possède une bonne interface qui reste simple d'utilisation pour du debug ou de la gestion de base de données.

- Bash pour interagir avec le projet et le dépôt GitHub principalement. (commande créédans le package.json)

```
"watch": "encore dev --watch",
```

la commande "npm run watch" pour lancer tailwind.css grâce à encore (permet de charger le style en compilant le tailwind dans un fichier style.css.

Encore est un module qui permet d'optimiser et de compresser le code CSS.

- Wamp pour faire tourner le SQL. (aucune utilisation du localhost de wamp je passe pas le serveur php)

```
"server": "php -S localhost:8000 -t public -c php.ini",
```

en lançant la commande "npm run server" un serveur php est lancé sur le port 8000. J'ai créer cette commande dans mon package.json dans les scripts pour pouvoir lancer plus une meilleure productivité.

- Illustrator pour faire le Logo du site.

3- Partie technique

Logiciels utilisés:

Trello est une application très pratique elle 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).

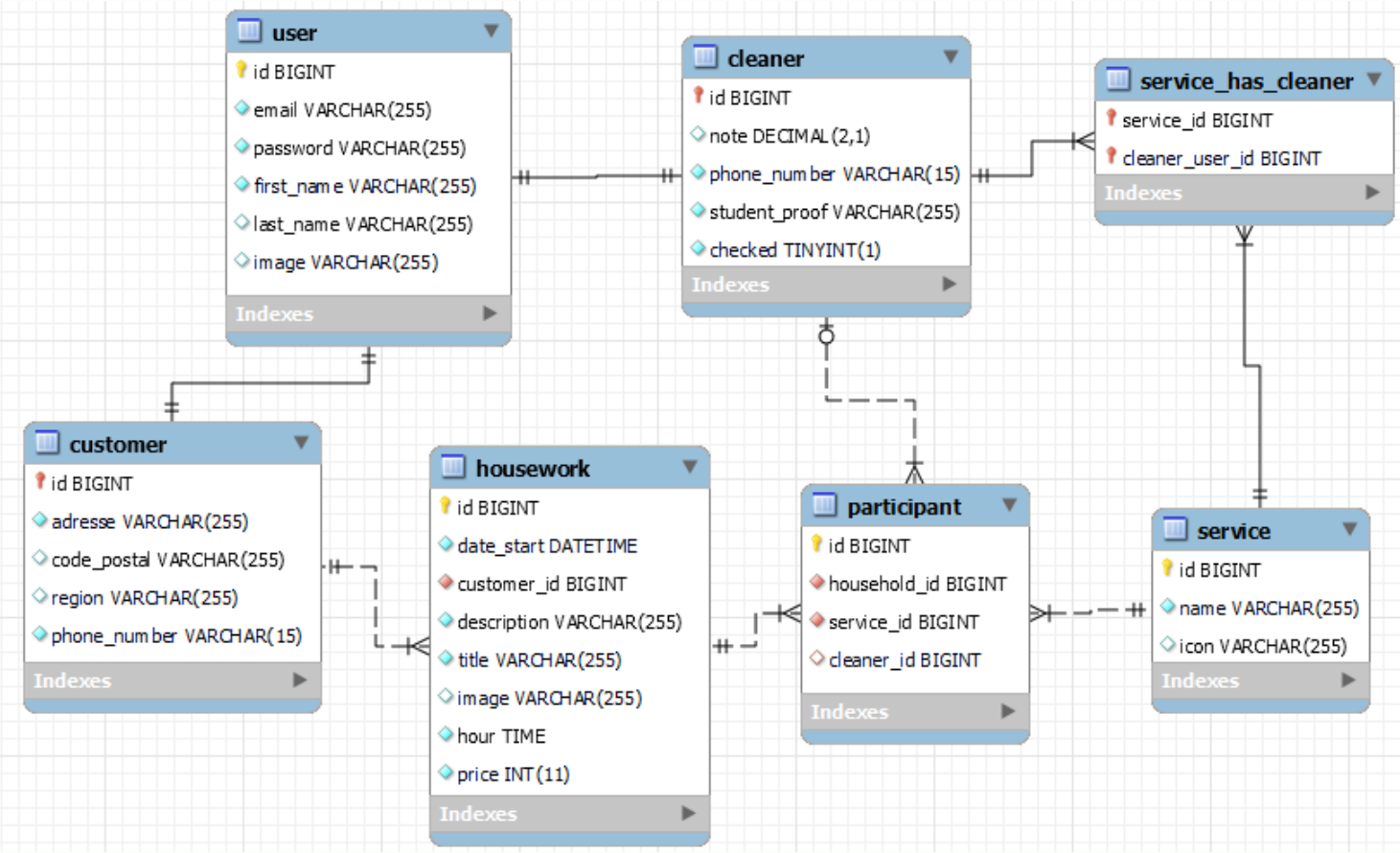
J'ai utilisé le terminal bash pour la plupart des commandes à réaliser comme lancer le serveur php, lancer les commandes npm et composer permettant l'installation des paquets au sein de Symfony.

```
"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 dépôt GitHub.

3- Partie technique

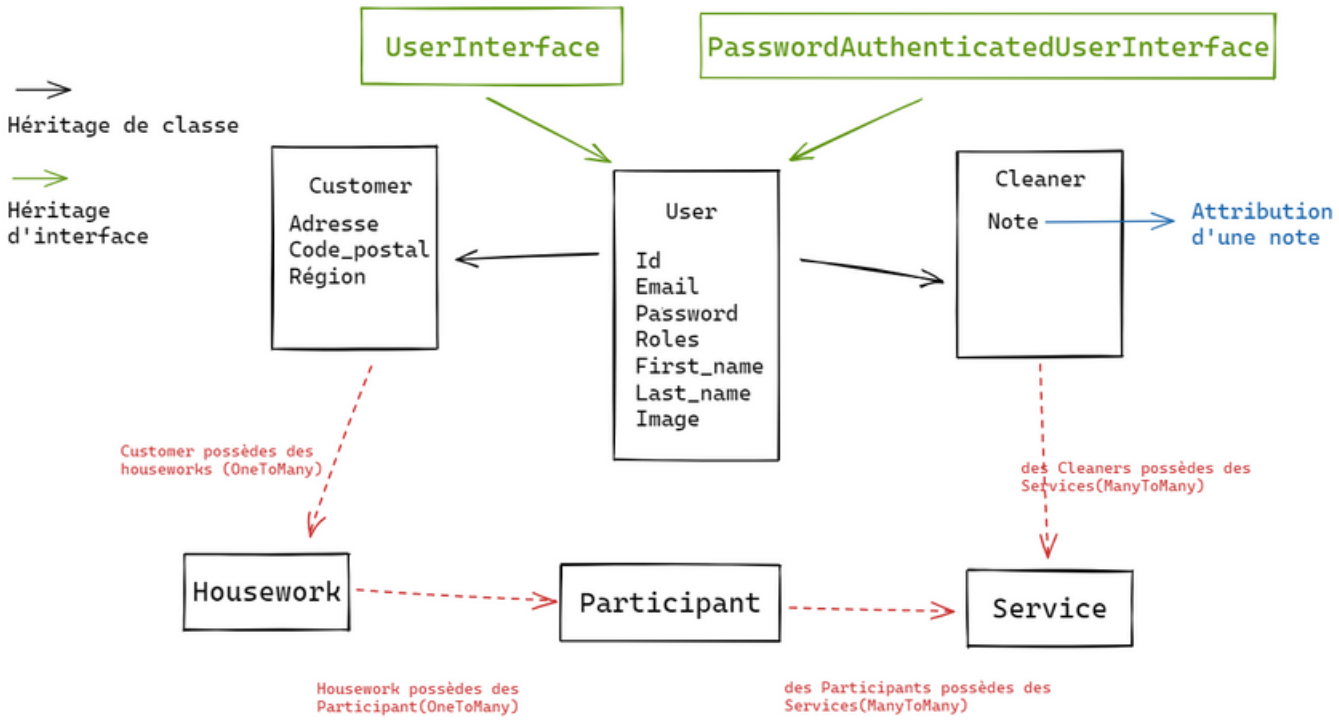
Base de données:



3- Partie technique

Diagramme de classe :

Diagramme de classe Stud Clean



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

Diagramme 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 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 nettement plus sécurisée que seulement entrer les mots de passes en clair dans la base de données.

Symfony utilise plusieurs algorithmes de hachage sécurisés tel que Bcrypt et Argon2 qui sont deux hachages très utilisés de nos jours et qui sont les plus sécurisés.

Dans mon application, j'ai utilisé l'algorithme Argon2 de Symfony afin de sécuriser les mots de passes des utilisateurs dans ma base de données, grâce à ça les mots de passes des utilisateurs de sites ne sont pas connus par l'administrateur ou encore des personnes mal intentionnées qui souhaiterai pirater la base de données.

3- Partie technique

Diagramme 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 :

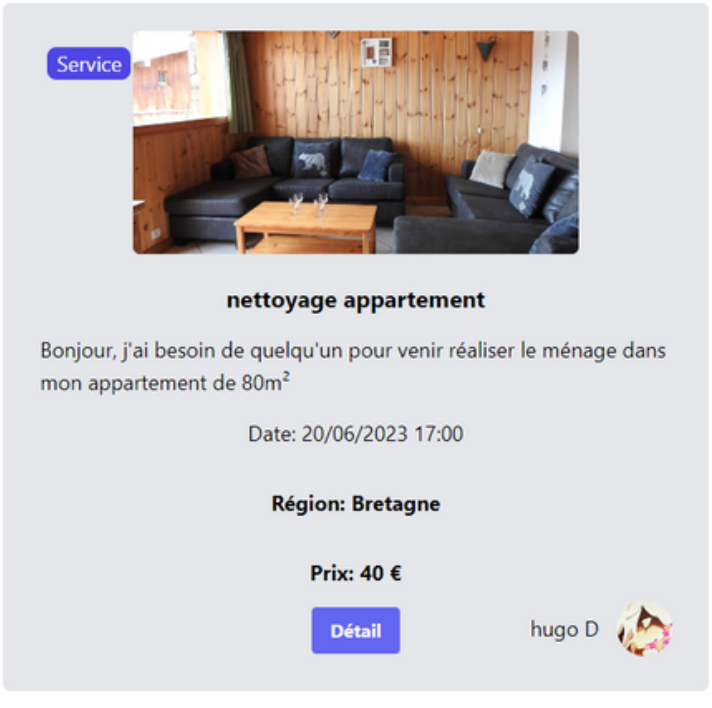
- Customer
- Cleaner

Pour mon application, seuls les étudiants pourront créer des comptes Cleaner c'est pourquoi j'ai préféré faire la distinction entre les deux entités Customer et Cleaner. Des personnes peuvent donc créer des comptes Customer pour poster des évènements, et les étudiants eux peuvent créer des comptes Cleaner s'ils souhaitent rendre des services à des Customer contre rémunérations, si jamais un utilisateur est un étudiant, mais souhaite poster des évènements et qui ne cherche pas à partager ses services, il peut simplement créer un compte Customer comme tout autres utilisateurs.

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

Ensuite, les Customers pourront créer des Houseworks ce que j'appelle aussi des Ménages Party qui sont simplement des évènements auxquels les Cleaners pourront s'inscrire.

Exemple de ménage party:



4- Présentation de l'application

Présentation des fonctionnalités de l'application :

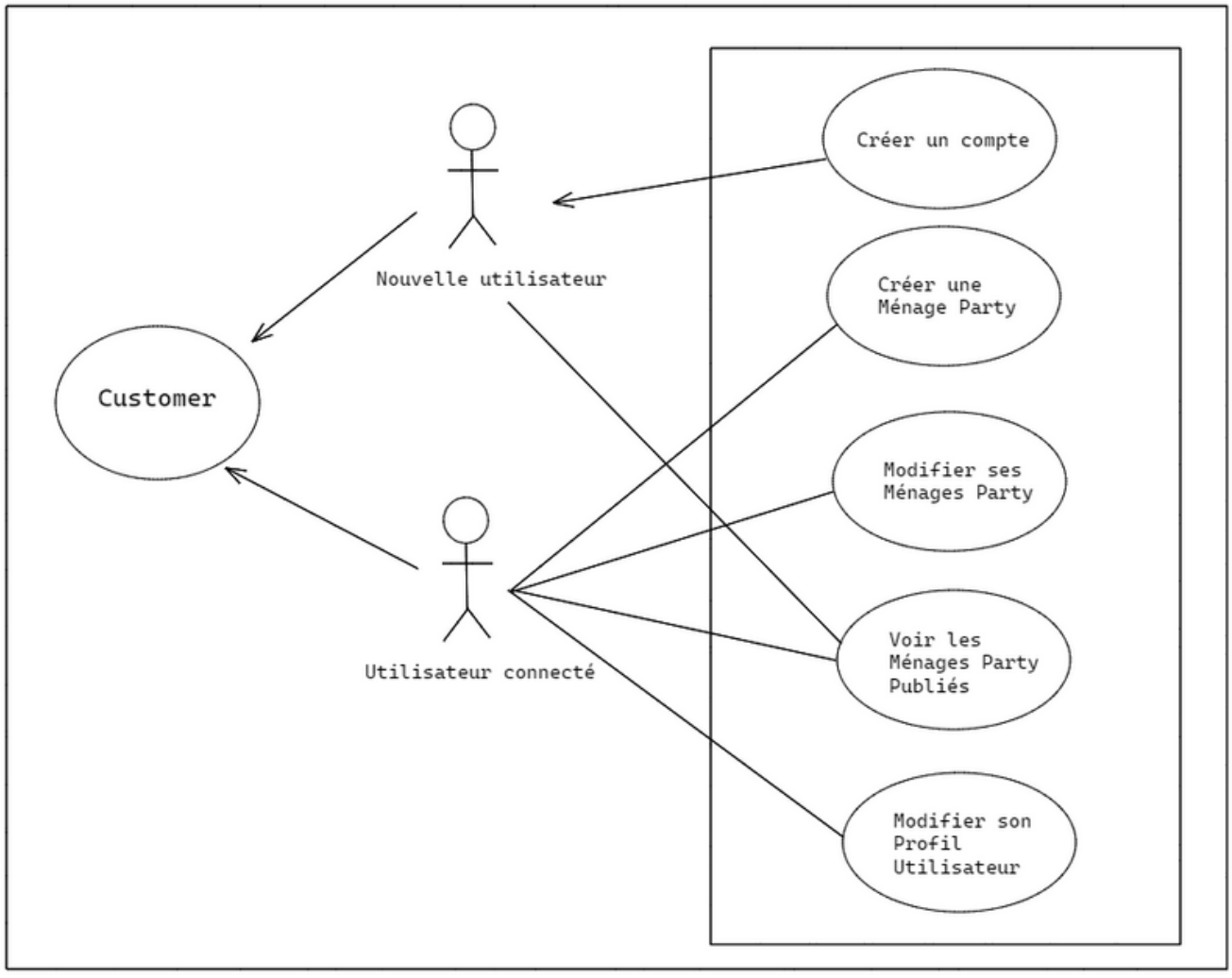
Stud Clean a pour objectif de répondre à la problématique développée précédemment, c'est-à-dire être en mesure de fournir les fonctionnalités suivantes :

- Créer un compte Customer pour poster et créer des Ménages Party
- Créer un compte Cleaner pour s'inscrire à des Ménages Party
- Profil différent pour chaque utilisateur (Cleaner/Customer)
- Customer: Accès aux Ménages Party publiés.
- Modification du profil et des informations relatives au type de compte (Cleaner/Customer)
- Système de vérification de compte Cleaner (justificatif étudiant par une photo) vérification réalisée par l'administrateur du site.
- Tous les utilisateurs peuvent voir les ménages partys des autres utilisateurs via l'onglet "Ménages Party".

4- Présentation de l'application

Interaction de l'application avec ses utilisateurs : Customer

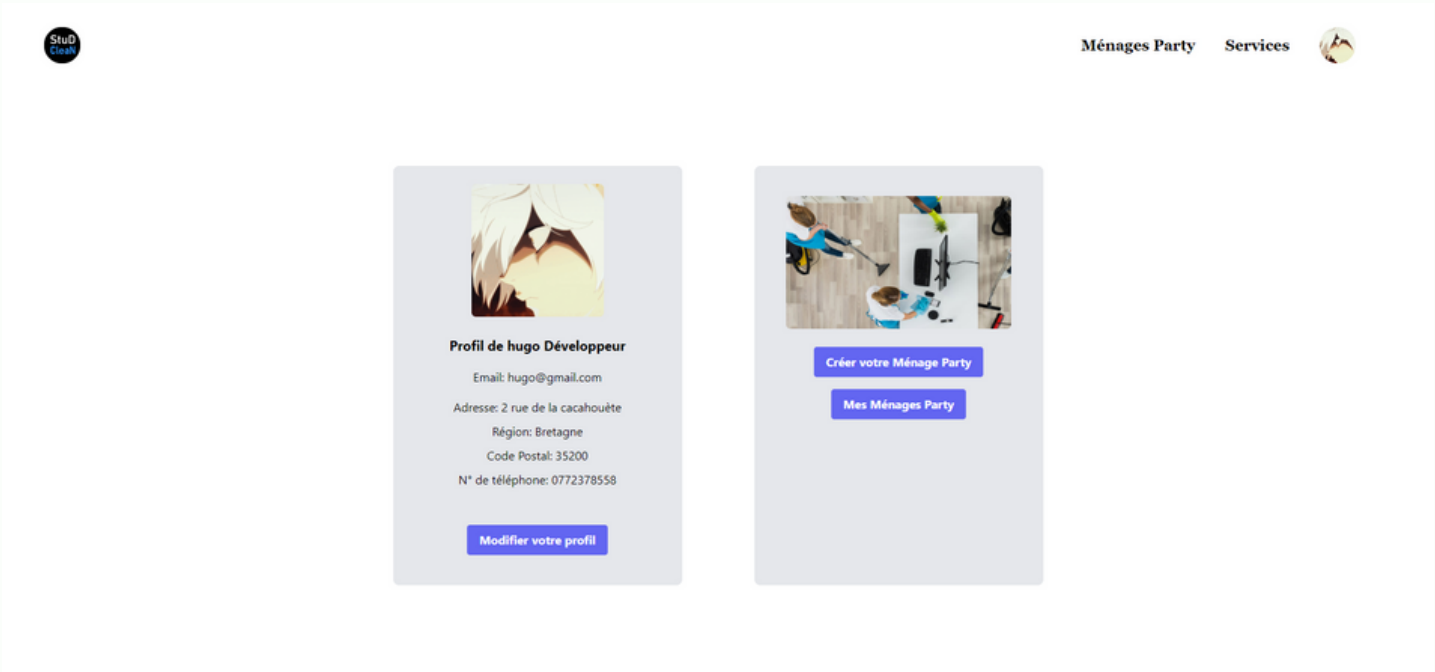
Diagramme d'utilisation d'un Customer:



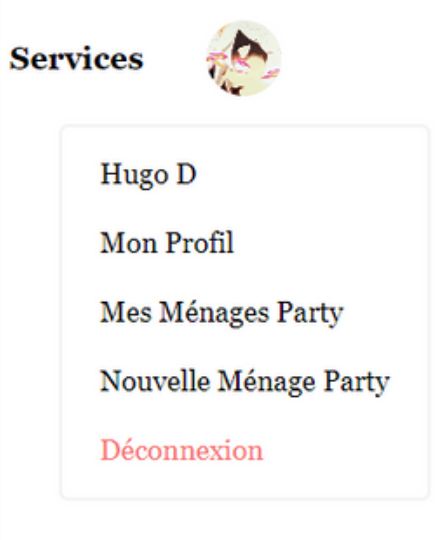
4- Présentation de l'application

Profil

profil d'un Customer



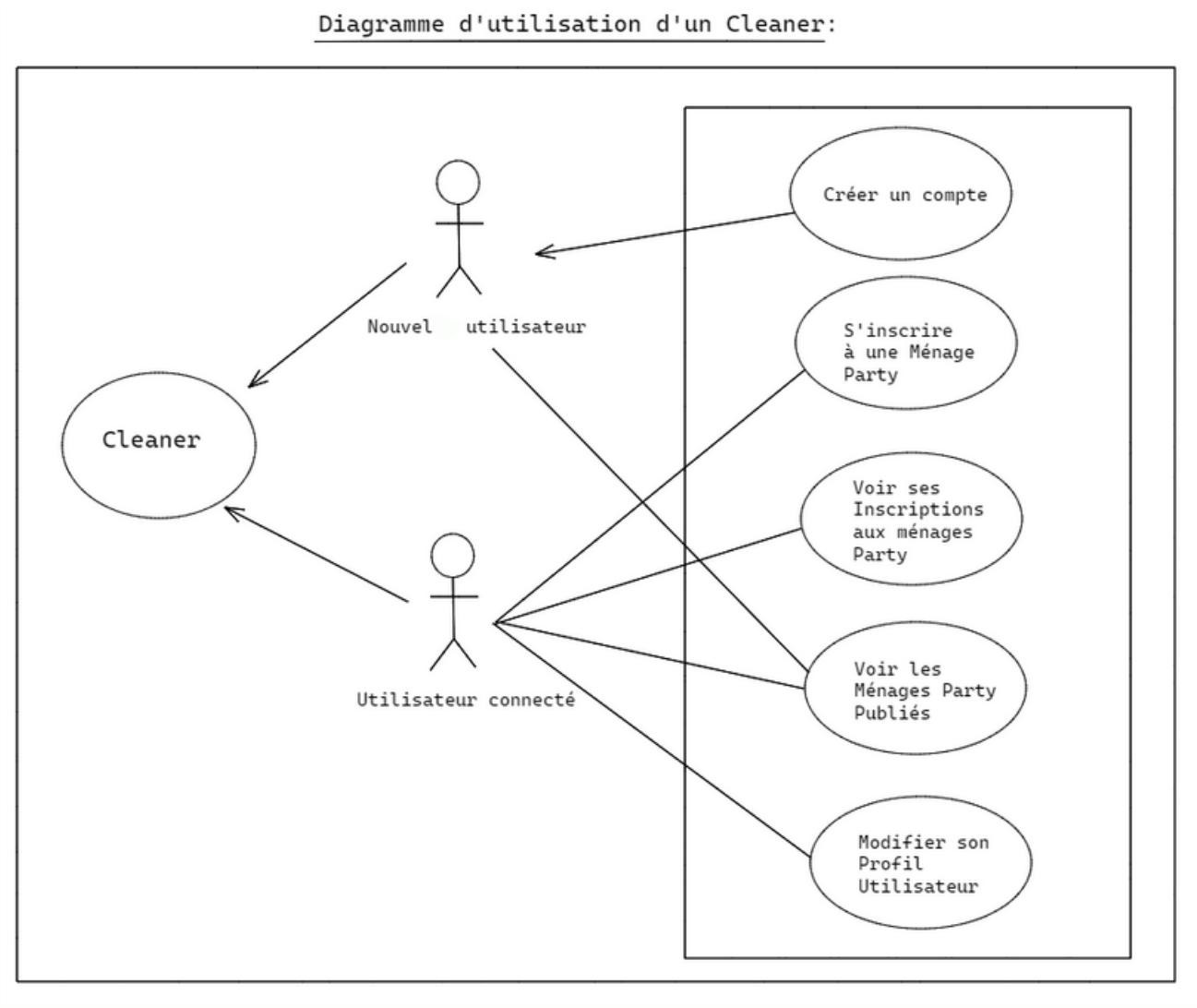
- Possibilité de voir les informations relatives au compte connecté
- Possibilité de modifier les informations du compte
- Possibilité de créer des Ménages Party
- Possibilité de voir ses Ménages Party créées.



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

4- Présentation de l'application

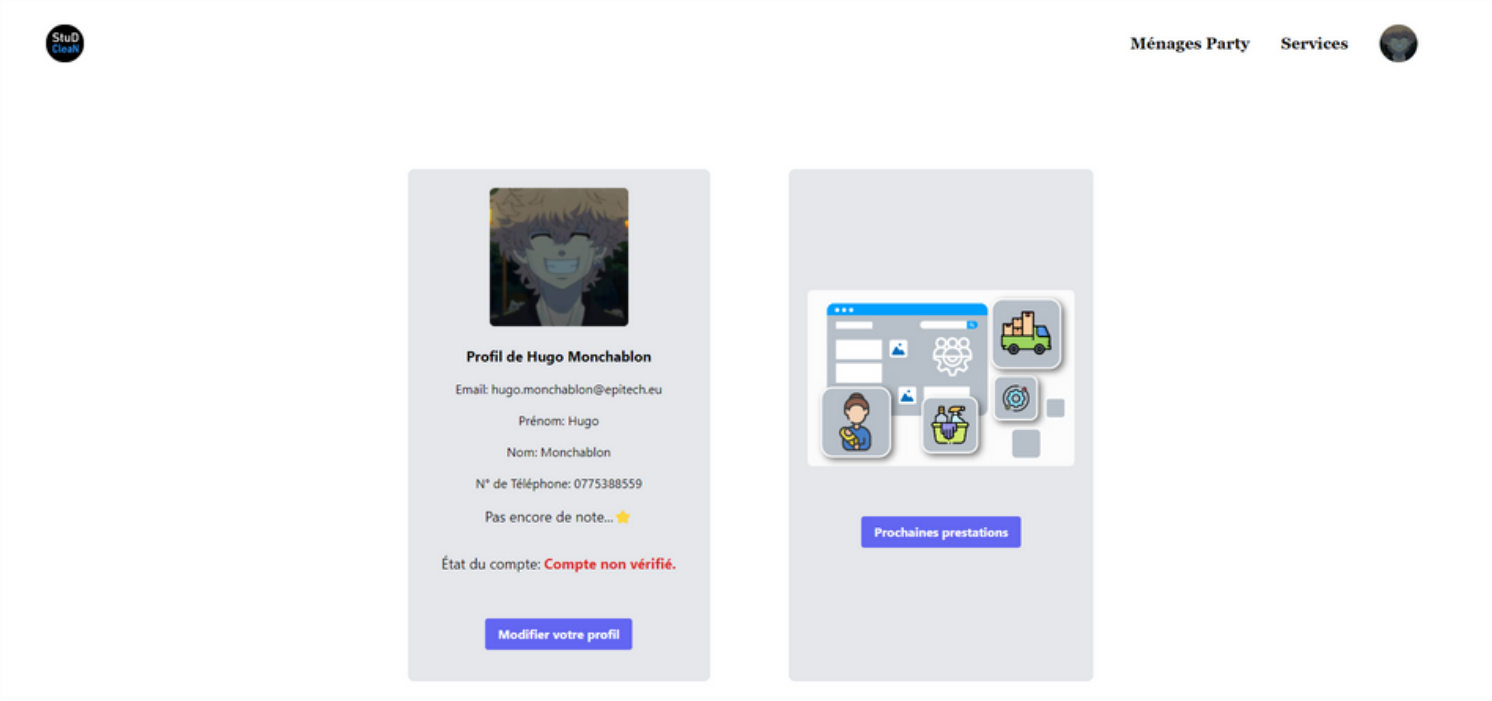
Interaction de l'application avec ses utilisateurs : Cleaner



4- Présentation de l'application

Profil

profil d'un Cleaner

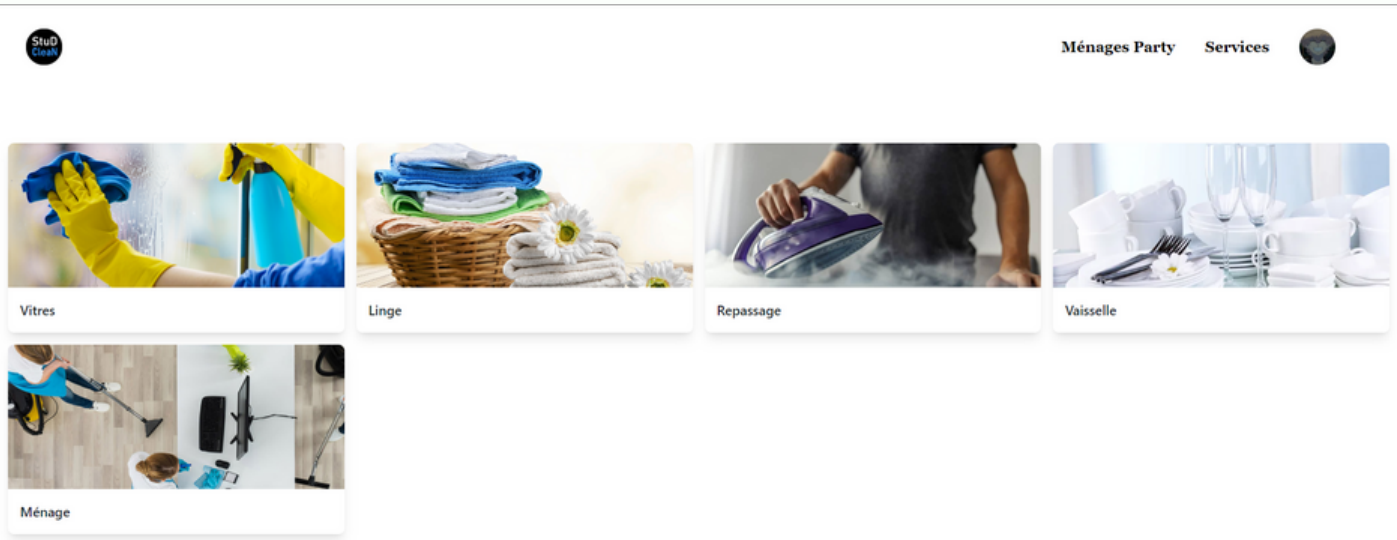


- Possibilité de voir les informations relatives au compte connecté
- Possibilité de modifier les informations du compte

- Menu déroulant au niveau de la barre navigation permettant de cliquer sur l'image de profil de l'utilisateur et de naviguer dans les différentes sections telles que :
- Voir le profil utilisateur
- Voir les prochaines prestations (les ménages party où l'utilisateur s'est inscrit).
- Un bouton déconnexion pour se déconnecter

4- Présentation de l'application

Page de Services



Sur le page de Services, on peut voir la liste des services que propose l'application telle que :

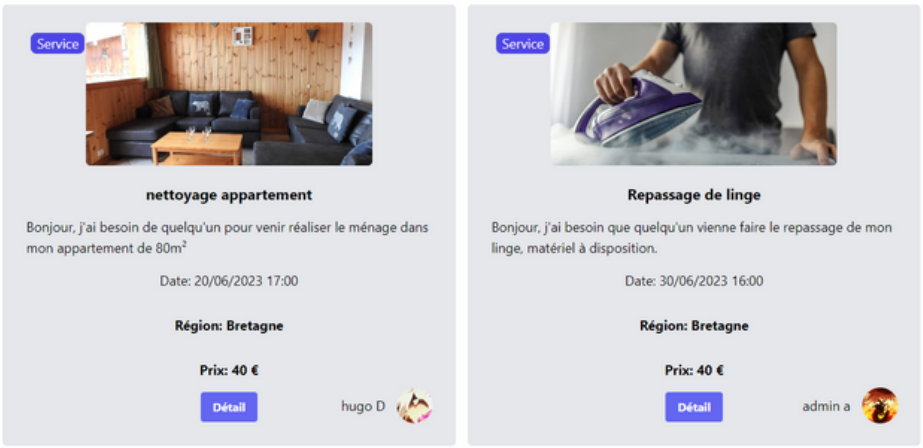
- Les Vitres (nettoyage des vitres)
- Le Linge (nettoyage du linge)
- Le Repassage (repassage du linge)
- La vaisselle (nettoyage de la vaisselle de manières général)

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

Pour l'instant, seul l'administrateur à la capacité de créer de nouveaux services.

4- Présentation de l'application

Page de Ménages Party



Sur le page de Ménages Party, on peut voir la liste des ménages party qu'ont posté 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 profil du créateur de la ménage party
- Le prix de la prestation demandé par le Customer

Je compte ajouter un système de transaction par carte bancaire. Cependant, l'inconvénient de cette question d'argent (paiement bancaire) est qu'il faut trouver un moyen sécurisé pour le paiement, étant donné que Stud 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 réaliser. C'est services que le Cleaner aura au préalablement choisis 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 profil d'un Customer permet à l'utilisateur de pouvoir voir les Ménages Party qu'il a créé avec notamment :

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

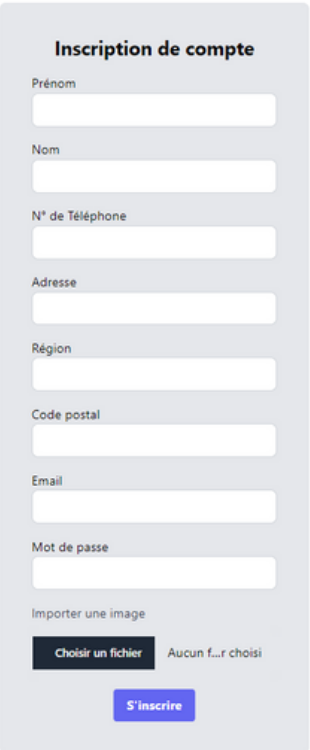
En cliquant sur le détail d'une ménage party, un utilisateur pourra s'il le désire modifier ou supprimer sa ou ses ménages party.

5- Review de codes

Page d'inscription:

La page d'inscription comporte 2 choix :

- soit créer un compte client (Customer) pour poster 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 Customer où il pourra renseigner son Prénom, Nom, N° de téléphone, Adresse, Région, Code Postal, Email, mot de passe ainsi qu'une image de profil.

Après avoir cliqué sur le bouton s'inscrire, le nouvel utilisateur Customer sera créé et envoyé en base de données pour pouvoir se connecter à chaque fois qu'il ira sur le site et renseignera ses identifiants.

5- Review de codes

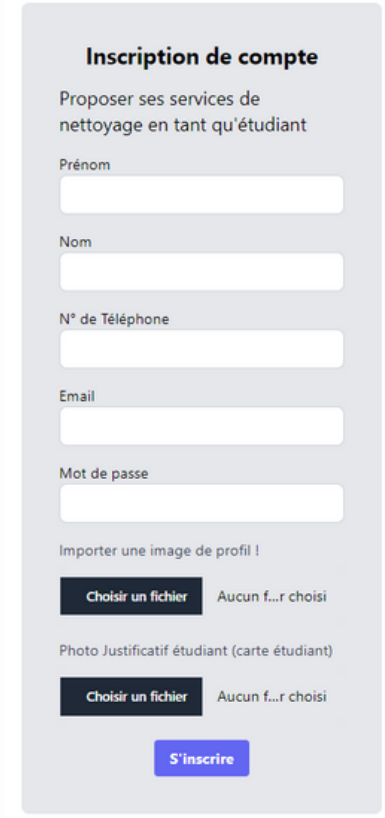
Page d'inscription:

Pour créer son compte étudiant (Cleaner) l'utilisateur tombera sur un formulaire différent. En effet,
au prénom, nom, email, N° de téléphone, mot de passe et à l'image de profil s'ajoutera les champs :

- Photo justificatif étudiant

L'utilisateur devra ajouter une photo d'un justificatif de scolarité d'étudiant, permettant à l'administrateur de pouvoir vérifier son compte.

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




```php
#[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'envoyé en base.

5- Review de codes

Page d'inscription:

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

$student_proof = $form->get('student_proof')->getData();
if ($student_proof) {
    $StudentProofFileName = $fileUploader->upload($student_proof);
    $user->setStudentProof($StudentProofFileName);
}
```

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

```php
$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 enregistré dans sur 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 profil :

```php
#[Route('/profile/edit', name: 'app_edit_profile')]
public function editProfile(CustomerRepository $customerRepository, CleanerRepository $cle
{
    $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, [
            'is_edit_profile' => true,
        ]);
        $form->handleRequest($request);
    }
    else {
        $form = $this->createForm( type: RegistrationCleanerFormType::class, $user, [
            'is_edit_profile' => true,
        ]);
        $form->handleRequest($request);
    }
    $currentPassword = $form->get('currentPassword')->getData();
    $userPassword = null;

    if ($form->isSubmitted() && $form->isValid()) {
        if ($customer) {
            $userPassword = $customer->getPassword();
            if (!password_verify($currentPassword, $userPassword)) {
                $form->get('currentPassword')->addError(new FormError( message: 'Incorrecte
            }
            elseif (password_verify($currentPassword, $userPassword)) {
                $image = $form->get('image')->getData();
                if ($form->get('plainPassword')->getData()) {
                    $customer->setPassword(
                        $userPasswordHasher->hashPassword(
                            $user,
                            $form->get('plainPassword')->getData()
```

Pour la modification de profil, 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 définir qu'elle type d'user essaye de modifier son profil pour éviter d'avoir 2 pages différentes avec deux 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 différents champs, c'est pourquoi j'utilise deux formulaires différents.

5- Review de codes

Modification de profil :

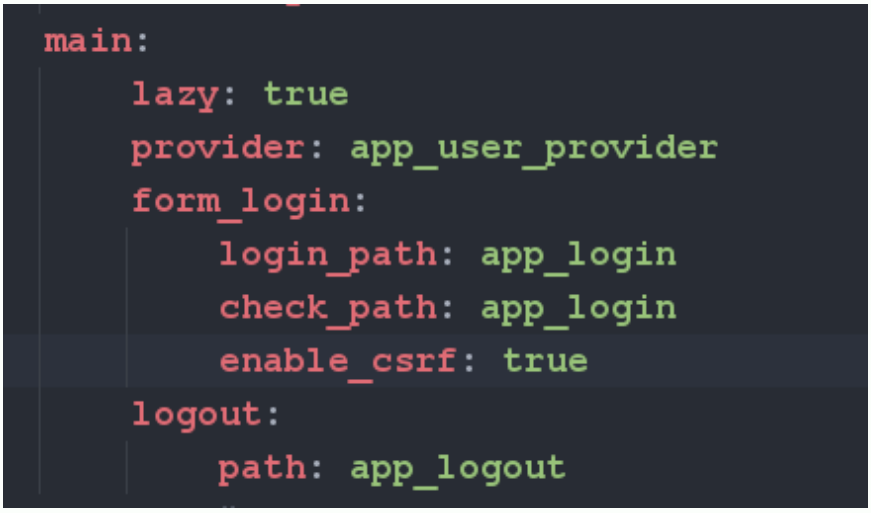
```php
if ($form->isSubmitted() && $form->isValid()) {
    if ($customer) {
        $userPassword = $customer->getPassword();
        if (!password_verify($currentPassword, $userPassword)) {
            $form->get('currentPassword')->addError(new FormError( message: 'Incorrecte!!'));
        }
        elseif (password_verify($currentPassword, $userPassword)) {
            $image = $form->get('image')->getData();
            if ($form->get('plainPassword')->getData()) {
                $customer->setPassword(
                    $userPasswordHasher->hashPassword(
                        $user,
                        $form->get('plainPassword')->getData()
                    )
                );
                $entityManager->persist($customer);
            }
            if ($image) {
                $fileName = $fileUploader->upload($image);
                $customer->setImage($fileName);
            }
            $entityManager->persist($customer);
```

Quand le formulaire est validé et envoyé, je récupère le mot de passe actuel que l'utilisateur a saisi pour pouvoir le vérifier avec celui dans la base de données, ce qui permet de vérifier si la personne qui essaye de modifier le compte soit le propriétaire du compte. Ensuite, je définis si c'est bon, je valide la modification en assignant le nouveau mot de passe saisi, s'il y en a un si jamais l'utilisateur n'a pas rempli le champ "nouveau mot de passe" alors, je ne change pas le mot de passe de l'utilisateur.

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

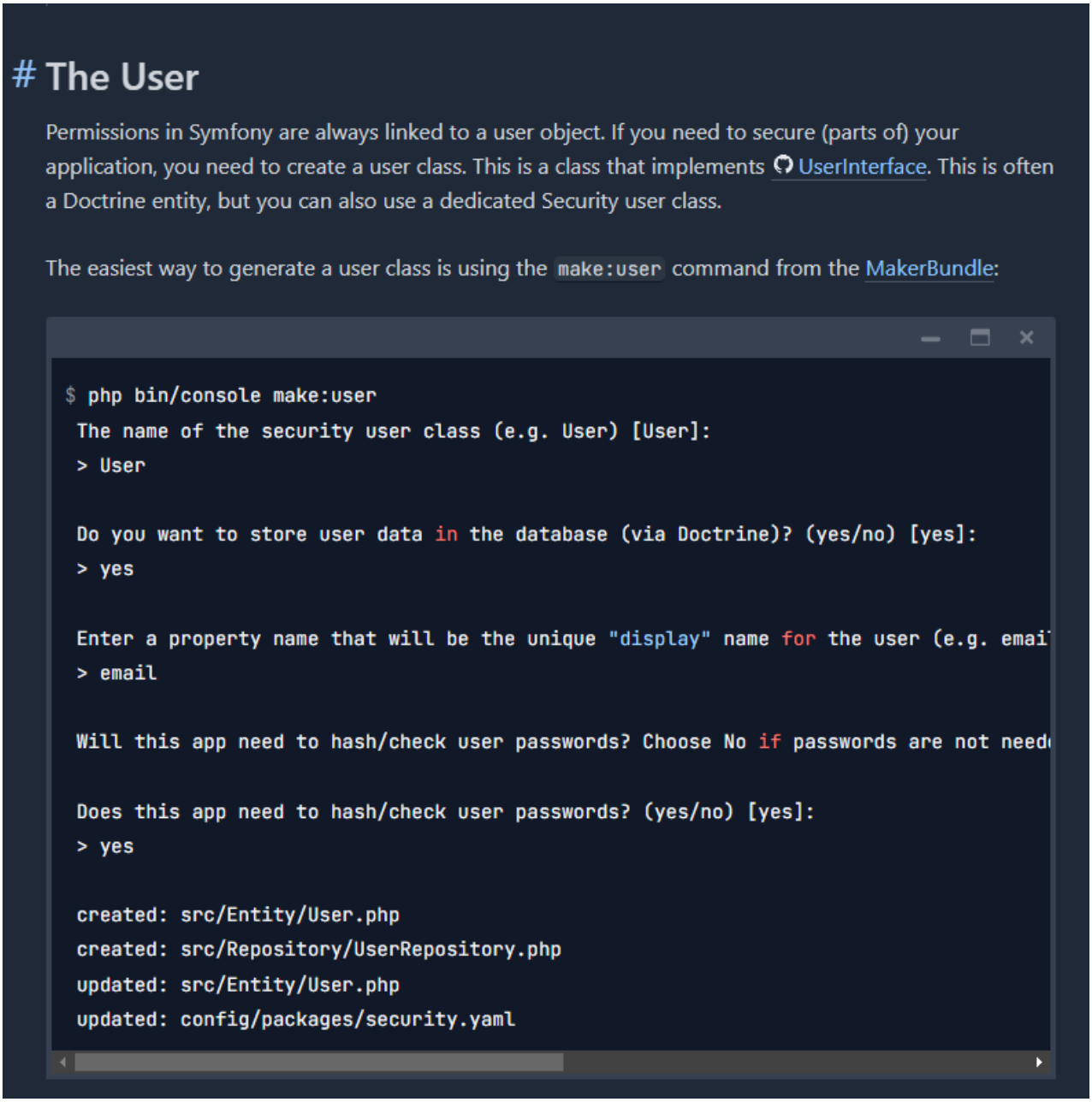
"CSRF" signifie Cross-Site Request Forgery, une attaque qui exploite la confiance de l'utilisateur dans une application web en utilisant des requêtes falsifiées pour accéder à des fonctionnalités non autorisées ou pour exécuter des actions malveillantes au nom de l'utilisateur.

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. emai
 > email

 Will this app need to hash/check user passwords? Choose No if passwords are not need

 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 (documentation officielle de Symfony). 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 "UserInterface" 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