Documentation technique

I- Version de la documentation technique

II- Comprendre quels fichiers modifier et pourquoi ?

III- Comment s’opère l’authentification ?

IV- Où sont stockés les utilisateurs ?

I- Version

Support	Version	Dates
Documentation technique	1.0	09/04/2021

II- Comprendre quels fichiers modifier et pourquoi ?

Les différents types de dossier :

Controller :

Module qui traite les actions de l'utilisateur, modifie les données du modèle et de la vue.
Initialement dans une architecture MVC un Controller contient la logique concernant les actions effectuées par l'utilisateur. Symfony dans une bonne pratique préconise de ne pas mettre de logique métier dans les Controllers, ils appelleront alors d'autres dossiers comme Form ou Entity qui permettrons d'avoir un code plus simple dans les Controllers.
Les Controllers correspondent également aux routes de l'application, une fois appelés, ils donnent l'information aux Repository pour aller chercher des informations en base de données et les ils envoient à la vue qui s'occupera de l'affichage. Les Controllers servent de passerelle entre les données et l'affichage.

Les différents Controllers :

DefaultController :

- indexAction : Permet d'afficher la page d'accueil.

SecurityController :

- loginAction : Permet d'afficher la page de connexion.

TaskController :

- listAction : Permet d'afficher toutes les tâches.
- listActionFinished : Permet d'afficher les tâches finies.
- createAction : Permet la création d'une tâche.
- editAction : Permet de modifier une tâche.
- toggleAction : Permet de changer le statut d'une tâche.
- deleteAction : Permet de supprimer une tâche.

UserController :

- listAction : Permet d'afficher tous les utilisateurs
- createAction : Permet la création d'un nouvel utilisateur.
- editAction : Permet de modifier un utilisateur.

DataFixtures :

Les Fixtures sont utilisées pour charger un « faux » ensemble de données dans une base de données qui peut ensuite être utilisée pour des tests ou pour vous aider à vous fournir des données intéressantes pendant que vous développez votre application.

Entity :

Une entité, c'est ce que l'ORM de Symfony (Doctrine) va manipuler et enregistrer dans la base de données.
Les entités sont donc des représentations de tables de la base de données, on y retrouve les champs hydratés de la base de données, et permettent de modifier ou créer des nouvelles données.

User :

- id : Identifiant unique attribué à un nouvel utilisateur.
- username : Nom d'utilisateur en string, cette valeur est unique.
- password : Mot de passe d'un utilisateur en string, cette valeur est encrypté par le service Security de Symfony
- email : Email d'un utilisateur en string, cette valeur est unique.
- tasks : clef étrangère, relation avec l'Entity Task. Il regroupe les tâches créées par l'utilisateur.
- rôles : Rôle d'un utilisateur en json, par défauts les utilisateurs ont le rôles : ROLE_USER.

Task :

- id : Identifiant unique attribué a une nouvelle tâche.
- createdAt : Date de création d'une tâche au format DateTime.

- title : Titre d'une tâche en string.
- content : Contenu d'une tâche en text.
- isDone : Champ booléen (correspond à 1 ou 0) qui indique si une tâche est finie ou non.
- user : Ce champ sert de relation à l'entité Task, et correspond à l'utilisateur qui a créé la tâche.

Form :

Permet de paramétrer les formulaires html, gérer le rendu des champs du formulaire HTML, définir le type de données, la validation des données soumises par l'utilisateur, le mappage des données du formulaire en objets.

TaskType :

- buildForm : Création d'un formulaire Task via Symfony Form ainsi que la configuration des différents champs du formulaire, on y retrouve :
  - Le title, type TextType;
  - Le content, type TextareaType.

UserType :

- BuildForm : Création d'un formulaire User via Symfony Form ainsi que la configuration des différents champs du formulaire, on y retrouve :
  - Le username, type TextType;
  - Le password, type RepeatedType pour une confirmation, les deux champs sont de type PasswordType;
  - L'email, type EmailType;
  - Le rôle de l'utilisateur, type ChoiceType permet de choisir entre ROLE_USER ou ROLE_ADMIN.

Repository :

Permet de faire des appels à la base données et ainsi récupérer des informations via certains filtres, il est possible aussi de créer des requêtes.

TaskRepository :

- Cette classe n'a pas été modifié, elle hérite de la classe ServiceEntityRepository de symfony, il existe plusieurs méthodes pour récupérer des données (find, findOneBy, findBy, findAll).

UserRepository :

- Cette classe n'a pas été modifié, elle hérite de la classe ServiceEntityRepository de symfony, il existe plusieurs méthodes pour récupérer des données (find, findOneBy, findBy, findAll).

Security :

Permet de gérer toute la partie authentification, la gestions des accès, on y retrouve aussi des Voters qui permettent d'ajouter une restriction a certaine page du site web.

LoginFormAuthenticator :

- Supports : Vérifie la source de la connexion (venant bien de la page login via une requête POST.
- getCredentials : Récupérer les informations du formulaire, dans le cas où les informations soient incorrectes, on garde en mémoire le nom d'utilisateur pour le réaffecter dans le bon champ du formulaire.
- getUser : On vérifie que le token généré par le formulaire n'est pas été modifié ou supprimé, puis on vérifie que le nom d'utilisateur soit bien relié à un utilisateur.
- checkCredentials : On vérifie que le mot de passe indiqué soit le bon par rapport à celui enregistré en base de données.
- getPassword : est utilisé lorsqu'on change de cryptage.
- onAuthenticationSuccess : Si la connexion est un succès, on effectue une redirection soit vers la page d'accueil, soit vers la page demandé avant connexion (si elle existe et si l'utilisateur a les droits).
- getLoginUrl : Génère l'url complète vers la page de login.

Voter/TaskVoter :

- supports : Vérifie que l'on est bien en train de faire une demande liée aux tâches, on y retrouve deux paramètres EDIT : lorsqu'on veut faire une édition sur une tâche existante, et DELETE : lorsqu'on veut supprimer une tâche existante
- voteOnAttribute : s'occupe de vérifier si on est bien connecté et appel d'autre fonction selon l'action que l'on souhaite effectué.
- canDelete : vérifie qu'on est les privilège pour accéder à la suppression d'une tache
- canEdit : vérifie qu'on est les privilège pour accéder la modification d'une tache

Voter/UserVoter :

- Supports : Vérifie que l'on est bien en train de faire une demande liée aux utilisateurs, on y retrouve trois paramètres CREATE : lorsqu'on veut créer un utilisateur, UPDATE : lorsqu'on veut faire modifier un utilisateur et VIEW : lorsqu'on veut voir des informations sur un utilisateur.
- voteOnAttribute : s'occupe de vérifier si on est bien connecté et appel d'autre fonction selon l'action que l'on souhaite effectué.
- canView : vérifie qu'on est les privilège pour accéder à l(affichage des utilisateurs
- canEdit : vérifie qu'on est les privilège pour accéder à la modification d'un utilisateur
- canCreate : vérifie qu'on est les privilège pour accéder à la création d'un utilisateur

Template :

Regroupe les fichiers d'affichage html/twig.

Public :

Regroupe les dossiers css/js/etc, on y retrouve aussi le dossier regroupant les résultats des tests fonctionnels.

Config :

Le dossier config regroupe les différentes configurations en .yaml des services rajouter par Composer, on peut y paramétrer des routes, des services ou des bundles.

III- Comment s'opère l'authentification ?

L'authentification (s'authentifier) veut dire confirmer son identité en fournissant par exemple son nom d'utilisateur ou son email et son mot de passe sur un système.

Les informations sur les utilisateurs vont être enregistrées en base de données et c'est là-bas que nous allons récupérer son nom d'utilisateur et mot de passe pour l'identifier.

Comment mettre en place le système de sécurité de Symfony ?

Pour ce système d'authentification il faudra alors créer une entité User où il faudra implémenter UserInterface.

Lorsque l'on installera le bundle symfony/security-bundle pour l'authentification, un fichier security.yaml sera généré.

Dans le fichier security.yaml la configuration de base est constituée du providers qui va permettre d'indiquer comment on va récupérer les utilisateurs et les informations d'authentification comme le nom d'utilisateur et le mot de passe. On récupère les informations d'authentification en base de données généralement.

```
providers:
        app_user_provider:
            entity:
```

```
            class: App\Entity\User
            property: username
```

Le fichier securiry.yaml est constitué de l'encoders qui permettra de choisir le type d'encodage pour le mot de passe, l'algorithme défini sur auto prendra le meilleur algorithme de cryptage actuel de symfony.

```
encoders:
    App\Entity\User:
        algorithm: auto
```

Il y a également le firewall qui permet de définir les composants qui vont pouvoir authentifier l'utilisateur. Cela peut être dans le firewall dev (désactivé la securité dans l'environnement de développement) ou main.
Dans le firewall main, anonymous sera à true si l'on souhaite que chaque utilisateur accède au site. Sinon il faudra changer cette valeur. Il faudra également renseigner le provider qu'on a configuré auparavant.

Il faudra également spécifier dans le guard l'utilisation du App/Security/LoginFormAuthenticator qui permettra de récupérer les données soumises par l'utilisateur, de vérifier le mot de passe et le token csrf, vérifier si l'utilisateur existe bien et rediriger l'utilisateur après l'authentification.

```
firewalls:
    dev:
        pattern: ^/(_(profiler|wdt)|css|images|js)/
        security: false
    main:
        anonymous: lazy
        provider: app_user_provider
        guard:
            authenticators:
                - App\Security\LoginFormAuthenticator
```

On pourra également renseigner un niveau d'accès suivant le rôle des utilisateurs dans access_control.

```
access_control:
    - { path: ^/login, roles: IS_AUTHENTICATED_ANONYMOUSLY }
    - { path: ^/users/, roles: ROLE_ADMIN }
    - { path: ^/, roles: ["ROLE_USER", "ROLE_ADMIN"] }
```

Lorsque l'on se rend sur /users sans être connecté par exemple on est redirigé sur /Login car l'access_contrôle est activé.
La route /login doit contenir un formulaire permettant de s'authentifié, il faudra alors le créer dans les templates twig.

Il faudra créer un controller SecurityController pour ajouter une nouvelle route /login pour authentifier l'utilisateur. Ce controller renverra le formulaire précédemment créer
On pourra ajouter la classe AuthentificationUtils dans ce controller pour avoir le lastUsername et les possibles erreurs de connexion.

Comment procéder à l’authentification ?

Une fois le formulaire de connexion soumis, Symfony et Security vérifie les données envoyées par l’utilisateur, grâce à la classe LoginFormAuthenticator, on vérifie que le token généré par le formulaire soit valide. On vérifie ensuite que l’utilisateur existe bien en base de données et également le mot de passe.
Si tout s’est bien passé, l’utilisateur est à présent connecté, et l’utilisateur est gardé en session, il sera alors redirigé vers la page d’accueil.

IV- Où sont stockés les utilisateurs ?

Qu’est-ce qu’une base de données ?

Les utilisateurs sont stockés dans une base de données. Une base de données est une collection structurée de données.

C’est quoi PHPMYADMIN ?

PHPMYADMIN est un système de gestion de base de données que j’ai utilisé pour ce projet. PHPMYADMIN est un outil d'administration gratuit et open source pour MySQL et MariaDB. Son installation configure automatiquement Apache, MySQL, PHP et Perl sur votre ordinateur.

Comment stocker les utilisateurs dans PHPMYADMIN ?

Grace à PHPMYADMIN, on pourra alors créer une nouvelle base de données que l’on nommera par exemple Todoco. Et à partir de cette base de données il sera possible de créer

des tables. Des tables correspondent à un ensemble de données organisées sous forme d'un tableau où les colonnes correspondent à des catégories d'informations. Dans notre cas pour notre base de données Todoco il y aura deux tables, User et Task. Il y aura dans ces deux tables des lignes d'enregistrement où seront présentes toutes les données des utilisateurs (comme le mot de passe, l'email, le pseudo) et des tâches.