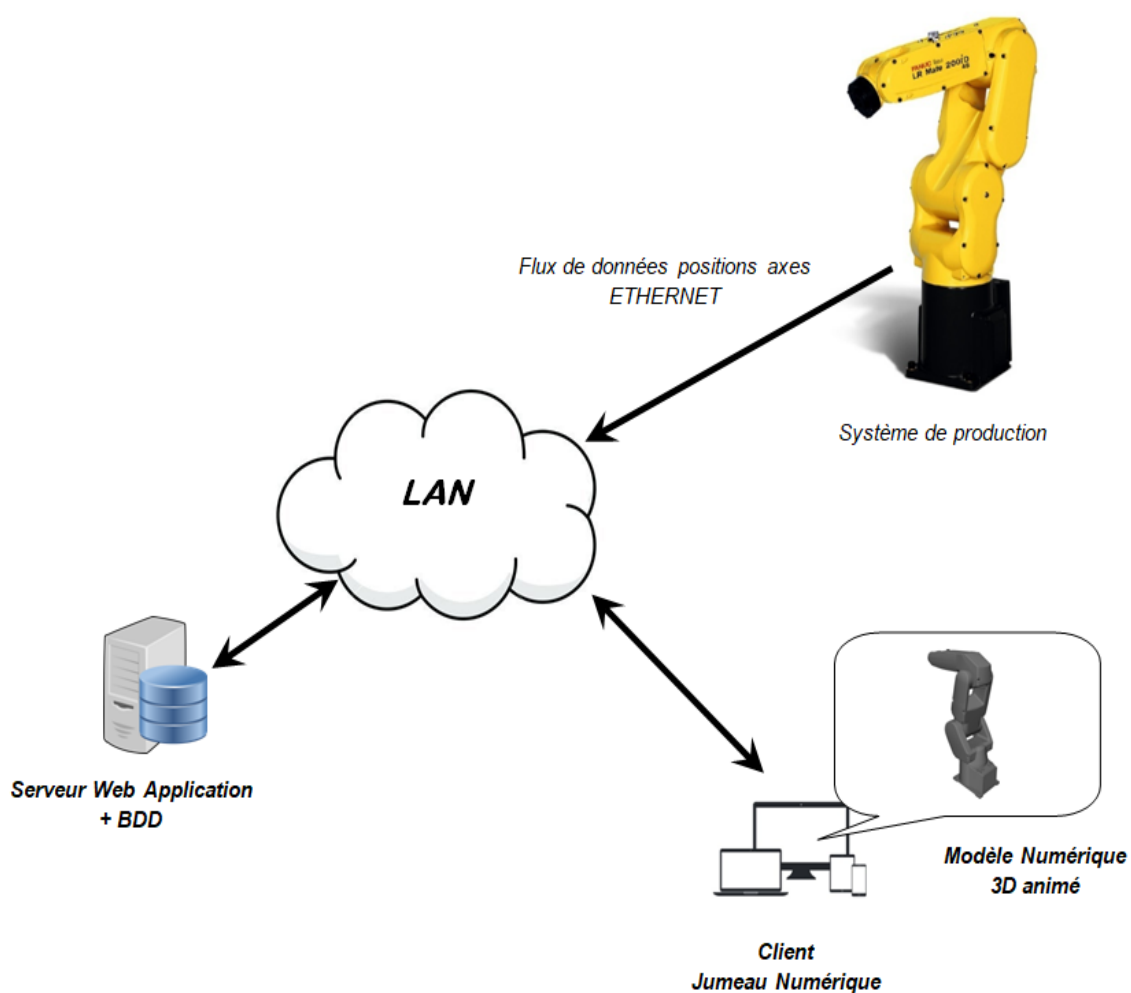




# **Projet Jumeau Numérique** **Présentation de l'application web**



Fraj Benoît

BTS - Systèmes Numériques 2018-2020

# Sommaire

I) Présentation du Projet.....	3
1 Les objectifs.....	3
2 Tâches à réaliser.....	4
II) Travaux réalisé sur l’application.....	5
1 Infrastructure du site.....	5
2 Table des utilisateurs.....	5
3 Le formulaire d’authentification.....	6
4 Gestion des utilisateurs.....	6
5 Configuration du robot.....	7

# I) Présentation du Projet

## 1 Les objectifs

L'objectif de notre projet est de créer une application web qui visualise en 3D le robot articulé FANUC LR Mate 200iD 4S en temps réel. L'application propose un système d'authentification à 2 facteurs (username, password). Deux types de comptes sont disponibles, les comptes utilisateurs et le(s) compte(s) administrateur(s).

L'administrateur a des privilèges différents de celui de l'utilisateur, il peut gérer les comptes utilisateurs (suppression, ajout, édition).

L'application permet également d'enregistrer ou de supprimer des séquences qui sont par la suite présentes dans la base de données.

La base de données contient une table utilisateurs, une table séquences et une table robot (contenant les informations du robot à visualiser).

Le projet sera développé sous le framework Symfony et la gestion du projet se fera sur GitHub, afin de pouvoir plus facilement communiquer et avoir une meilleure expérience de travail de groupe.

## 2 Tâches à réaliser

Je m'occupe personnellement des tâches effectués par l'étudiant 1.

Cependant nous avons opté pour une méthode de travail nommé « Scrum », ce qui signifie que les différentes tâches à réaliser ne sont pas définitives et tout au long du projet chaque personne pourra choisir des tâches totalement indépendantes. La méthode agile est utilisée, ce qui signifie que des objectifs à réaliser pour chaque sprint sont définis.

### Énoncé des tâches à réaliser par les étudiants : (Prévisionnel\*)

#### ETUDIANT 1 :

SERVEUR WEB / INTEGRATION INFRASTRUCTURE / SECURITE /  
PROFILS UTILISATEURS / COMPTE ADMINISTRATEUR  
*Framework Symfony*  
*PHP*



#### ETUDIANT 2 :

AFFICHAGE/ANIMATION 3D DU JUMENT NUMERIQUE  
*JavaScript*  
*Bibliothèque three.js*  
*Framework Symfony*



#### ETUDIANT 3 :

GESTION/TRAITEMENT FLUX DE DONNEES SYSTEME REEL  
*PHP*  
*JavaScript*  
*Framework Symfony*



#### ETUDIANT 4 :

GESTION BDD / TRAITEMENT REQUETES / ENREGISTREMENT  
LECTURE SEQUENCES JUMENT  
*PHP*  
*Java*  
*XML*  
*Framework Symfony*



## II) Travaux réalisé sur l'application




























### 1 Infrastructure du site

À l'aide de la librairie Bootstrap, j'ai pu introduire une barre de navigation avec un thème qui gère le CSS afin de rendre l'application plus agréable. Bootstrap permet également de rendre le site responsive et les pages s'adaptent alors à tout type d'appareils (smartphone, tablette...).

Bootstrap m'aide aussi à gérer les images, grâce aux assets et la mise en forme de certains éléments HTML.

### 2 Table des utilisateurs

La table « **user** » est composé de 9 champs, comme ci-dessous :

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
<input type="checkbox"/> 1	<b>id</b> 	int(11)			Non	Aucun(e)		AUTO_INCREMENT	 Modifier  Supprimer  Plus
<input type="checkbox"/> 2	<b>email</b>	varchar(255)	utf8mb4_unicode_ci		Non	Aucun(e)			 Modifier  Supprimer  Plus
<input type="checkbox"/> 3	<b>username</b>	varchar(255)	utf8mb4_unicode_ci		Non	Aucun(e)			 Modifier  Supprimer  Plus
<input type="checkbox"/> 4	<b>password</b>	varchar(255)	utf8mb4_unicode_ci		Non	Aucun(e)			 Modifier  Supprimer  Plus
<input type="checkbox"/> 5	<b>roles</b>	longtext	utf8mb4_bin		Non	Aucun(e)	(DC2Type:json)		 Modifier  Supprimer  Plus
<input type="checkbox"/> 6	<b>nom</b>	varchar(255)	utf8mb4_unicode_ci		Non	Aucun(e)			 Modifier  Supprimer  Plus
<input type="checkbox"/> 7	<b>prenom</b>	varchar(255)	utf8mb4_unicode_ci		Non	Aucun(e)			 Modifier  Supprimer  Plus
<input type="checkbox"/> 8	<b>reset_token</b>	varchar(255)	utf8mb4_unicode_ci		Oui	NULL			 Modifier  Supprimer  Plus
<input type="checkbox"/> 9	<b>password_requested_at</b>	datetime			Oui	NULL			 Modifier  Supprimer  Plus

Des contraintes de validation sont présentes sur certains champs à l'aide des asserts, le mot de passe par exemple doit avoir un minimum de 8 caractères.

Le champ « roles » ne possède que 2 valeurs, « ROLE\_USER » et « ROLE\_ADMIN ».

Le champ « reset\_token » sert à créer un lien provisoire, qui servira à l'utilisateur afin de réinitialiser son mot de passe.

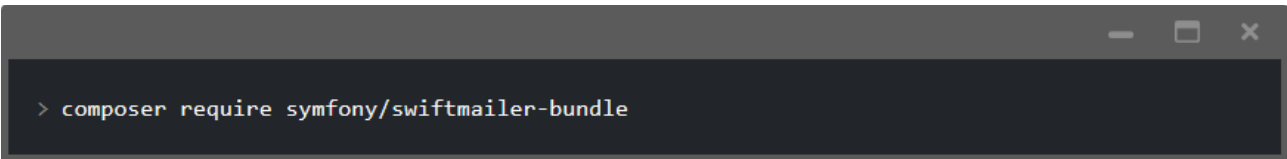
Le champ « password\_requested\_at » sert à définir si le lien créé par le reset\_token est encore valide ou non.

### 3 Le formulaire d'authentification

Sous Symfony le formulaire de connexion est géré par le composant sécurité de Symfony, de même pour la déconnexion. J'ai donc créé un formulaire avec les champs «pseudo» et « mot de passe », j'ai par la suite indiqué à Symfony avec quelle base de données il devait travailler (dans le fichier .env), afin de vérifier les données entrées par l'utilisateur dans le formulaire.

J'ai également mis une condition dans le controller, afin qu'un utilisateur soit renvoyé vers l'accueil lorsqu'il tente d'accéder à la page de connexion.

L'option « Mot de passe oublié » est disponible sous le formulaire de connexion, il permet à l'utilisateur de réinitialiser son mot de passe en cas d'oubli. J'ai utilisé le composant SwiftMailer pour l'envoi de mail, ainsi que le serveur smtp de Gmail.



```
> composer require symfony/swiftmailer-bundle
```

J'ai donc utilisé la commande ci-dessus, afin d'installer SwiftMailer et j'ai créé un compte Gmail pour l'envoi de mails aux utilisateurs.

Le mail est composé d'un lien ayant une durée de validité limitée, il renvoie l'utilisateur sur une page unique de l'application dont seuls les utilisateurs ayant une clé unique(token) peuvent y accéder.

**Voici la fonction qui permet de vérifier la validité du lien et d'en changer la durée :**

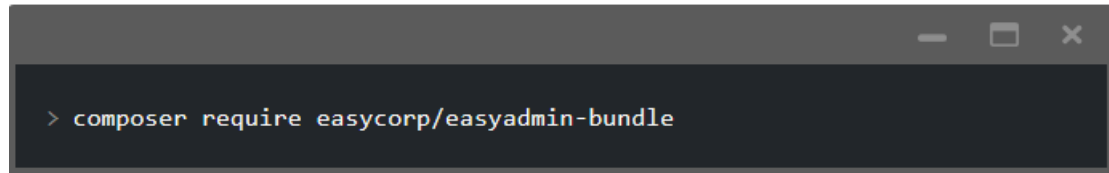
```
// si supérieur à 1h, retourne false
// sinon retourne true
private function isRequestInTime(\DateTime $passwordRequestedAt = null)
{
    if ($passwordRequestedAt === null)
    {
        return false;
    }

    $now = new \DateTime();
    $interval = $now->getTimestamp() - $passwordRequestedAt->getTimestamp();

    $daySeconds = 60 * 60;
    $response = $interval > $daySeconds ? false : $reponse = true;
    return $response;
}
```

## 4 Gestion des utilisateurs

Pour la gestion des utilisateurs j'ai utilisé le bundle EasyAdmin, disponible avec la commande ci-dessous :

A terminal window with a dark background and light gray text. The command `> composer require easycorp/easyadmin-bundle` is entered at the prompt.

```
> composer require easycorp/easyadmin-bundle
```

Après l'installation on atterrit sur le template généré par le bundle par toutes routes commençant par */admin*. Pour personnaliser l'affichage et les options, comme par exemple modifier ou supprimer un champ de la table, on modifie le fichier « *easy\_admin.yaml* ».

La gestion des utilisateurs n'est disponible que pour les utilisateurs ayant le rôle d'administrateur, cette condition se fait dans le fichier « *security.yaml* ».

J'ai également modifier le fichier de template du bundle, qui se situe dans le dossier « *vendor/easycorp/src/ressources/views/default/layout.html.twig* », afin de pouvoir intégrer la barre de navigation de l'application.

## 5 Configuration du robot

La page de configuration du robot sert à modifier, si besoin, les données du robot comme par exemple son nombre d'axes ou encore son adresse IP. J'ai donc créé une table robot afin que l'application puisse chercher dans la base de données le nombre d'axes, l'adresse IP, le nom ou le modèle du robot.

Avoir une table pourra servir également à l'avenir du projet afin de gérer potentiellement plusieurs robots dont les données sont différentes.

## 6 Modification du mot de passe

Étant donné que chaque utilisateur est créé par un administrateur, c'est donc l'administrateur qui définit un mot de passe pour chacun. J'ai alors mis en place une page pour la modification du mot de passe de chaque utilisateur de l'application, en allant dans l'onglet « *Mon Compte* » puis « *Modifier son mot de passe* ».

Le formulaire est composé de 3 champs :

## MODIFICATION DU MOT DE PASSE

Mot de passe actuel :

Nouveau mot de passe :

Confirmation du mot de passe

Enregistrer

La validation de l'ancien mot de passe est géré par Symfony grâce à une classe de Symfony que l'on ajoute avec `use App\Form\Model\ChangePassword;` dans le formulaire Symfony. On définit ensuite les champs avec le type password, voir ci-dessous.

```
$builder
    ->add('oldPassword', PasswordType::class)
    ->add('newPassword', PasswordType::class)
    ->add('confirm_password', PasswordType::class)
;
```

La confirmation du nouveau mot de passe est quant à elle validé par un assert de Symfony, nommée `EqualTo` et qui permet de vérifier l'égalité des 2 champs « Nouveau mot de passe ». Pour ajouter les asserts à une entité on utilise la commande `use Symfony\Component\Validator\Constraints as Assert;` en début de fichier.

J'ai ensuite utilisé les asserts comme ci-dessous :

```
/**
 * @var string
 *
 * @ORM\Column(name="password", type="string", length=255, nullable=false)
 * @Assert\Length(min="8", minMessage="Votre mot de passe doit faire minimum 8 caractères.")
 */
private $password;

/**
 * @Assert\EqualTo(propertyPath="password", message="Vous n'avez pas tapé le même mot de passe")
 */
public $confirm_password;
```



### III) Bibliographie

#### 1 Apparence de l'application

Bootstrap :

<https://getbootstrap.com/docs/4.0/layout/grid/>

<https://getbootstrap.com/docs/4.0/content/tables/>

#### 2 Sécurité

Formulaire de connexion:

[https://symfony.com/doc/current/security/form\\_login\\_setup.html](https://symfony.com/doc/current/security/form_login_setup.html)

Bundle EasyAdmin avec Symfony :

<https://symfony.com/doc/master/bundles/EasyAdminBundle/index.html>

Mot de passe oublié :

<https://geekco.fr/blog/mini-serie-reinitialisation-du-mot-de-passe-partie-4-5>

Modification du mot de passe en se basant sur l'ancien mot de passe :

<https://symfony.com/doc/current/reference/constraints/UserPassword.html>

<https://symfony.com/doc/current/reference/constraints/EqualTo.html>

Token CSRF (protection contre un type de vulnérabilité des services d'authentification web) :

<https://geekco.fr/blog/mini-serie-creation-d-un-espace-membre-sur-symfony-4-1-5>

<https://symfony.com/doc/current/security/csrf.html>