



WAG
INTERACTIVE

DOSSIER DE SYNTHÈSE



08 Avril 2016 – 07 Avril 2017

Next Formation | Concepteur Développeur Informatique

Sommaire

1. Remerciements
2. Présentation de l'entreprise
3. Projet
4. Environnement de travail
5. Choix des technologies
6. Conception
7. Développement et intégration
8. Conclusion

1. Remerciements

Je tiens à remercier toutes les personnes qui ont contribué au succès de mon alternance et qui m'ont aidé lors de la rédaction de ce dossier.

Tout d'abord, j'adresse mes remerciements à mon formateur, **Jérôme Ambroise** qui m'a beaucoup aidé dans le développement de mon projet, notamment pour le Framework Symfony3 et son ORM Doctrine que je ne connaissais pas du tout.

Je tiens à remercier vivement mon employeur, **Olivier Marcou** pour son accueil, le temps passé ensemble et le partage de son expertise au quotidien. Grâce aussi à sa confiance j'ai pu m'épanouir totalement dans mes missions. Il fut d'une aide précieuse dans les moments les plus délicats.

Enfin, je tiens à remercier également les différents membres de l'entreprise pour leur accueil, le partage de leurs connaissances techniques, aussi variées soient elles.

2. Présentation de l'entreprise

Web agency depuis plus de 15 ans



Wag Interactive est une agence de communication digitale depuis 1997.

Comme de nombreuses agences de communication, Wag Interactive a traversé l'explosion de la bulle Internet du début des années 2000, puis la crise économique de 2008 qui a ébranlé toutes les TPE. Cette conjoncture difficile lui a permis d'aiguiser ses champs d'expertises, de fidéliser ses clients et surtout d'accroître ses réseaux de compétences.

L'ADN de l'agence



Wag Interactive est une agence qui se distingue de part ses prestations, notamment celles en *consulting*¹ et celle en expertise technique et éditoriale.

Quelques chiffres



Wag Interactive en quelques chiffres c'est :

- Plus de 120 sites internet
- Plus de 30 projets en TYPO3
- Plus de 25 intranets/extranets

Mais c'est aussi :

- 20 freelances en renfort
- 5 structures partenaires
- Plus de 100 clients satisfaits des solutions apportées

¹ Consulting : Action de conseil et accompagnement du client au long du projet.

3. Projet

N'ayant pas eu un projet attiré mais plutôt plusieurs tâches distinctes, j'ai donc décidé de mettre sur pied un projet personnel afin de valider les compétences demandées pour le passage de ce titre.

Le projet sur lequel j'ai décidé de travailler est celui d'un vidéoclub. En effet, ils se font de plus en plus rare et sont en voie de disparition suite à la montée du streaming et de la vidéo à la demande. C'est pourquoi je me suis penchée sur cette problématique en réalisant ce projet.

➤ Les objectifs du projet :

- Gestion des films
- Gestion des utilisateurs
- Gestion du stock
- Gestion de la location des films

Par conséquent, il doit structurer les informations utiles aux utilisateurs et permettre d'y accéder rapidement. Ces informations sont utilisées dans les différentes fonctionnalités du projet.

Le projet doit permettre l'adhésion de nouveaux utilisateurs, mais également la gestion du stock des films, à savoir s'ils sont disponibles ou non, pour être loués par la suite.

4. Environnement de travail

Pendant ce projet j'ai donc été amené à travailler à mi-temps sur un ordinateur avec Windows et l'autre mi-temps sur Mac OS. J'ai d'ailleurs mis quelques temps à m'y habituer, n'ayant jamais développé sur ce dernier. Après l'avoir pris en main, j'ai installé les logiciels nécessaires au développement.

➤ Sublime Text 3



C'est un éditeur de texte générique codé en C++ et Python, disponible sur tout système d'exploitation. Son usage est courant, voire incontournable pour certaines tâches informatiques de base comme l'administration de système et le développement de logiciels.

➤ Mamp



Mamp est l'acronyme informatique qui signifie **M**acintosh **A**pache **M**ySQL et **P**HP. Il permet d'installer Apache, PHP et MySQL sous Mac OS. Il existe également pour Windows sous le nom de Wamp, et sous Linux sous Lamp.

➤ phpMyAdmin



PhpMyAdmin (PMA) est une application web de gestion pour les bases de données MySQL. Je l'ai utilisé pour créer, modifier et insérer des données dans les tables de ma base de données.

5. Technologies utilisées

➤ Symfony3

Pour le développement de ce projet, j'ai choisi d'utiliser un framework et plus particulièrement Symfony3. C'est un puissant framework qui permet la réalisation accélérée d'applications complexes, de façon structurée et facilement maintenable par la suite.

Qu'est-ce qu'un framework ?

Le mot « framework » provient de l'anglais « frame » qui signifie « cadre » en français et « work » qui signifie « travail ». C'est donc un cadre de travail, c'est-à-dire un environnement de travail avec un ensemble de composants qui servent à créer les grandes lignes d'un logiciel.

Parmi les différents frameworks PHP, Symfony3 est sans doute le plus populaires et le plus utilisé. Il dispose d'une large communauté qui met à jour et optimise le contenu de ses différents composants, ce qui en facilite la maintenance.

Symfony organise son code en Bundle. Ce sont des briques de l'application, regroupant du code à un même endroit. Nous aurons par exemple, un bundle lié à l'utilisateur dans lequel sera stocké le code concernant l'utilisateur. Ces bundles sont souvent structurés sous le modèle MVC.

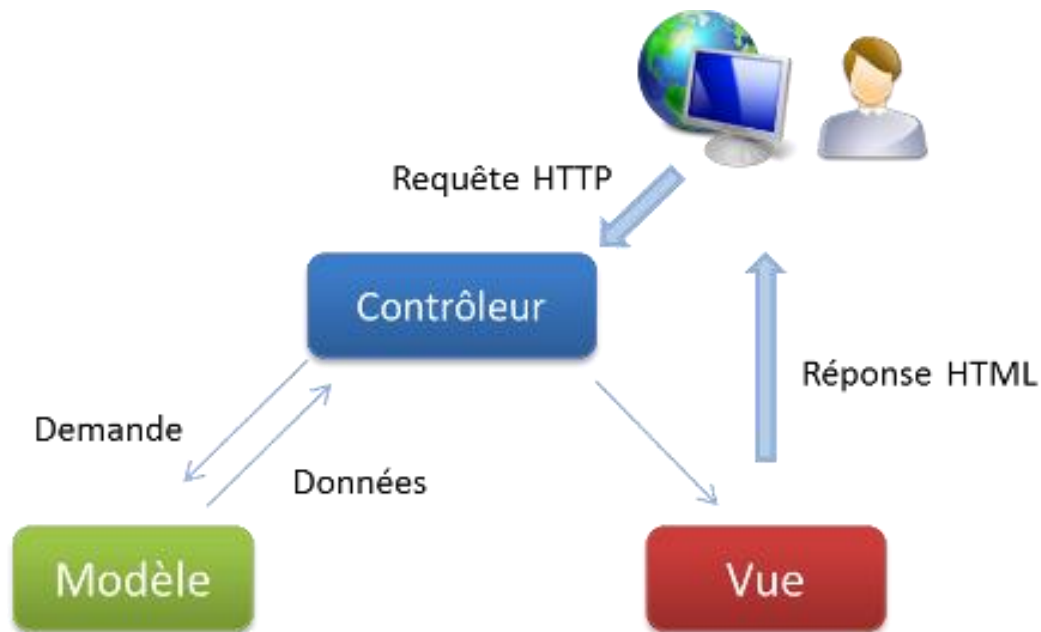
Qu'est-ce que le modèle MVC ?

Le modèle MVC (Modèle – Vue – Contrôleur) est très répandu dans le développement d'applications et occupe également une place importante dans le développement web. Il permet de structurer une application en distinguant la partie présentation d'une part, et le code applicatif d'autre part, ce qui facilite le développement en équipe, la relecture ainsi que la maintenance. Dans le contexte d'une application web, on obtient les éléments suivants :

- **Modèle** : c'est la partie qui contient les données manipulées par le site. Autrement dit, bien souvent, ce sont des objets (dans le langage objet) qui s'apparentent à ce qu'on appelle souvent la « couche métier ». En d'autres mots, il s'agit du cœur du programme.
- **Vue** : elles représentent les différentes pages du site qui affichent les informations demandées. Le rôle de « vue » est souvent rempli par des templates.
- **Contrôleur** : c'est l'endroit où sera réunie la logique métier, autrement dit, c'est ici que ce fera le traitement que l'on veut appliquer sur nos données. Il sert de passerelle entre la vue et le modèle. Le contrôleur frontal (Front Controller)

reçoit directement les requêtes utilisateur (URL et paramètres) puis, il se charge de les dispatcher aux contrôleurs concernés qui vont exécuter les actions liées.

Ci-dessous, voici une illustration du fonctionnement du modèle MVC :



Génération de code

Symfony3 permet aussi une génération de code via la console grâce à des lignes de commandes comme :

```
php bin/console generate:bundle
```

Ce type de commande permet, par exemple de générer directement des fichiers avec le code lié à des fonctionnalités courantes comme le développement portant sur le *CRUD*¹

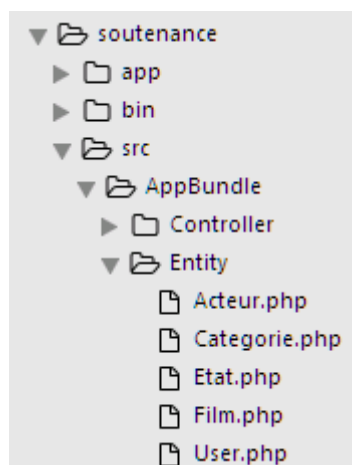
¹ CRUD : Désignation des quatre opérations de base pour la persistance des données, en particulier le stockage d'information en base de données.

(Create, Read, Update, Delete) par rapport à une table de la base de données. La génération du code permet un gain de temps non négligeable sur la conception d'un projet de grande ampleur.

➤ L'ORM Doctrine

Doctrine est un ORM (Object-Relational Mapping) permettant de faciliter la manipulation de données stockées dans un système de gestion de base de données relationnelles (SGBD) au sein des langages de programmation objet (PHP ou Java par exemple).

Avec Doctrine, nous manipulons donc des objets mappés à des tables en base de données. C'est l'ORM qui se charge de faire évoluer l'état des données en base en fonction de l'état de nos objets. Ces objets sont appelés « Entité » (Entity en anglais) et sont situées dans le dossier portant le même nom, situé lui-même à l'intérieur d'un Bundle (cf. arborescence ci-dessous).



➤ Un moteur de Template : Twig

Comme tout moteur de *template*¹, il permet la séparation du code PHP de la mise en page HTML. On se retrouve alors avec, d'un côté, le script qui fait une certaine chose (par exemple, récupération des données de la base de données), et de l'autre, la mise en page avec des zones prédéfinies où seront placées les données générées par le script.

L'avantage est de pouvoir travailler uniquement sur la mise en page, sans modifier quoique ce soit dans le script PHP et inversement. Ce qui permet alors de diviser efficacement le travail à faire.

Les avantages de Twig sont la rapidité, la flexibilité ainsi que la sécurité.

¹ Template : Mot anglais désignant un modèle de conception de logiciel ou de présentation de données.

6. Conception

6.1. Cas d'utilisation

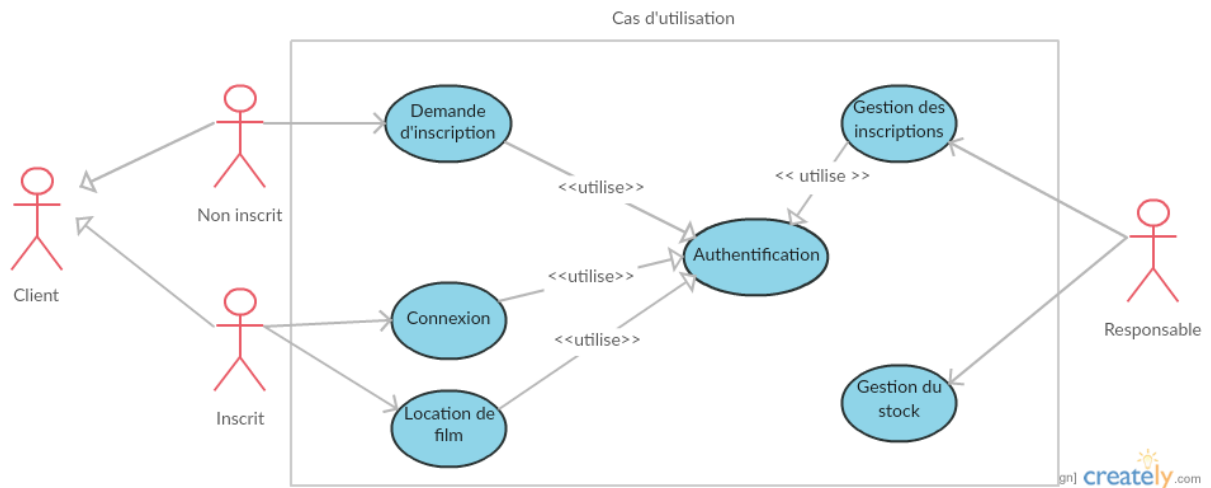


Diagramme créé avec createely (logiciel en ligne).

Ici, 2 acteurs ont été identifiés :

- Le client non inscrit peut faire une demande d'inscription. Celui qui est déjà inscrit peut quant à lui, se connecter et louer des films.
- Le responsable qui gère les inscriptions et la gestion du stock.

6.2. Diagramme de séquence

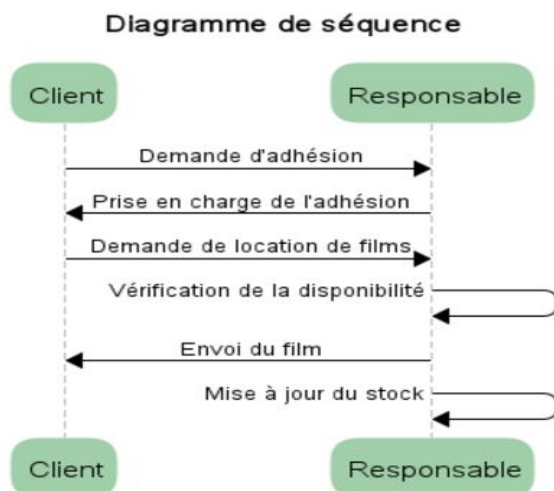


Diagramme réalisé avec websequencediagrams (logiciel en ligne)

Un client qui n'est pas inscrit fait une demande d'adhésion. Celle-ci est alors gérée par le responsable qui la prend en charge. Une fois inscrit, le client peut louer un film, le responsable vérifie avant toute chose la disponibilité de celui-ci avant d'envoyer le film au client puis de mettre à jour le stock.

6.3. MCD (Modèle Concepteur des Données)

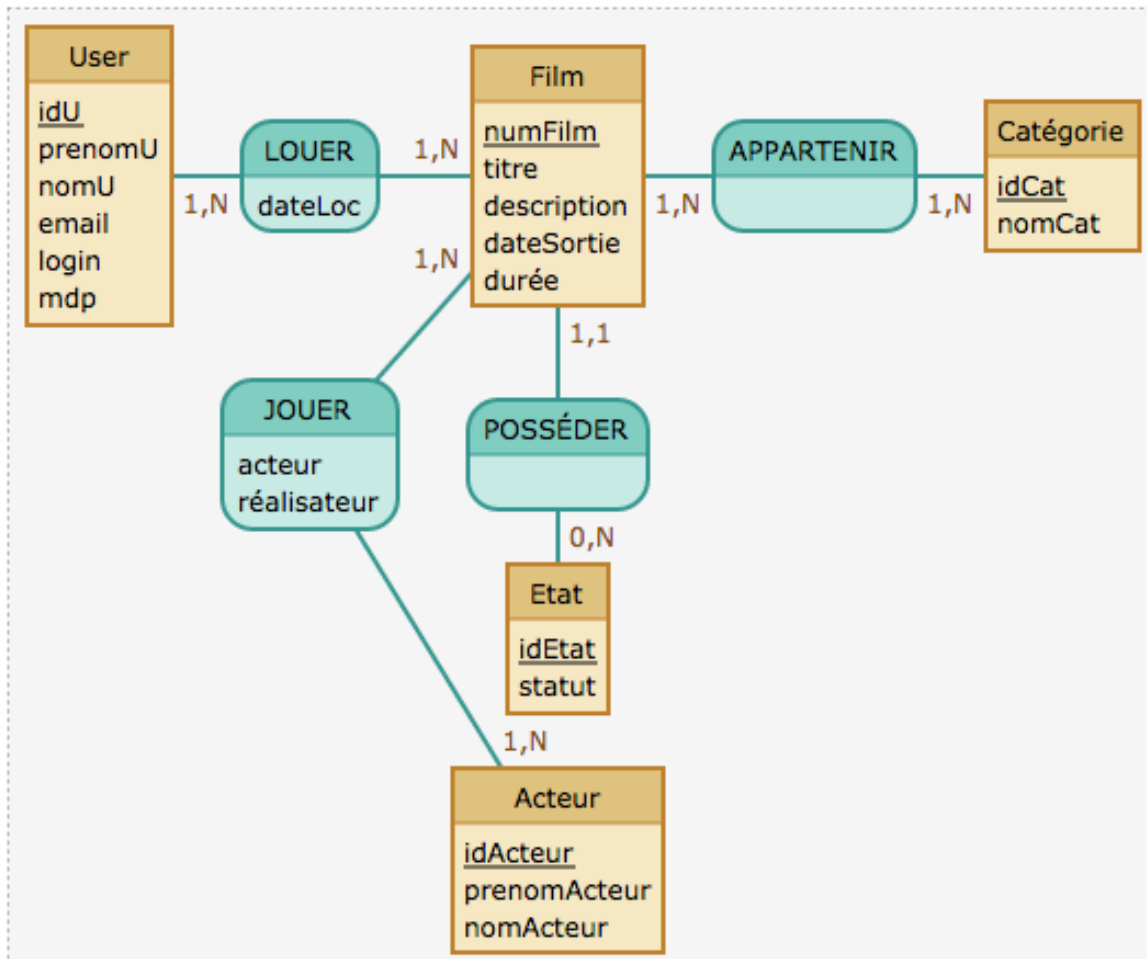


Schéma réalisé avec Mocodo (logiciel en ligne).

Une explication brève du schéma :

- L'association « Louer » dispose d'un attribut date pour pouvoir définir à quelle date l'emprunt le client a emprunté le film.
- L'association « Jouer » comporte deux attributs. Ce sont des booléens permettant de définir pour un film donné et un acteur, si l'acteur est simplement acteur ou acteur **ET** réalisateur.

6.4. MLD (Modèle Logique des Données)

User (idU, prenomU, nomU, email, login, mdp)
LOUER (#idU, #numFilm, dateLoc)
Film (numFilm, titre, description, dateSortie, durée, #idEtat)
APPARTENIR (#idCat, #numFilm)
Catégorie (idCat, nomCat)
JOUER (#numFilm, #idActeur, acteur, réalisateur)
Etat (idEtat, statut)
Acteur (idActeur, prenomActeur, nomActeur)

7. Développement et intégration

7.1. Installation de Symfony3

Pour que Symfony fonctionne, il faut déjà qu'un serveur web soit installé sur la machine. Pour ma part, il s'agit de MAMP. Une fois le serveur installé, voici les étapes par lesquelles je suis passée :

- Étape 1

```
$ sudo mkdir -p /usr/local/bin
$ sudo curl -Ls https://symfony.com/installer -o /usr/local/bin/symfony
$ sudo chmod a+x /usr/local/bin/symfony
```

- Étape 2

```
$ symfony new my_project_name
```

Dans cette étape, nous allons créer le projet symfony.

- Étape 3

```
$ cd projet_symfony/
```

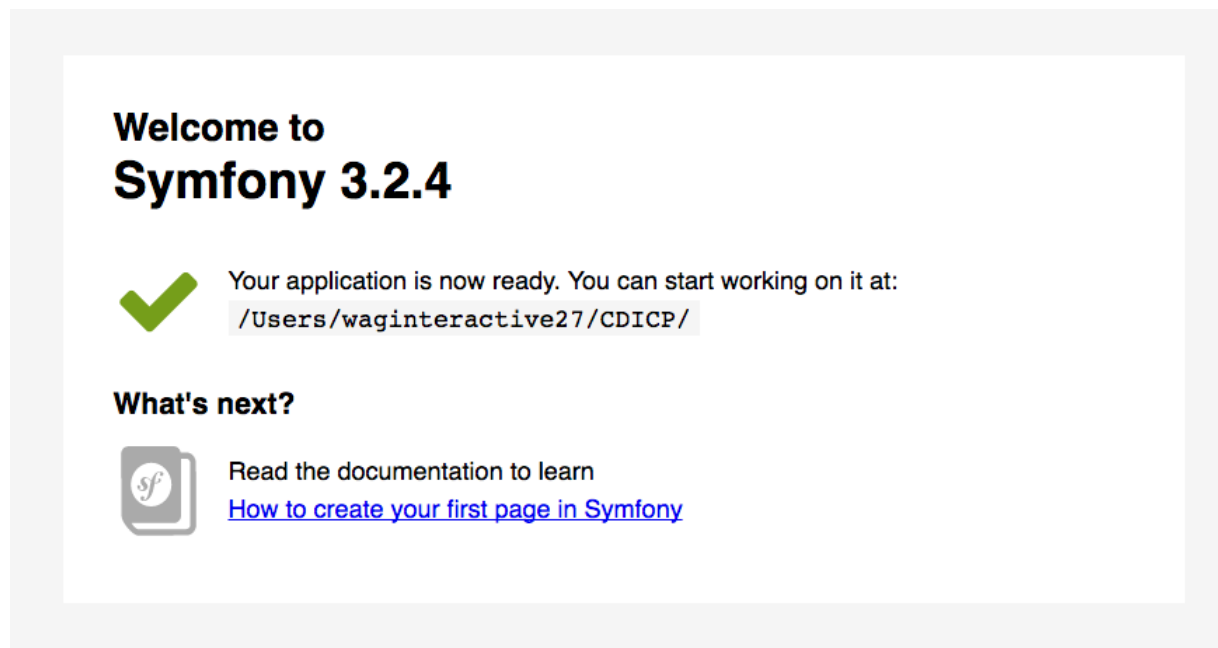
Une fois le dossier créé, on se place dedans.

- Étape 4

```
$ php bin/console server:run
```

Et enfin, on démarre le serveur.

- Étape 5



Après l'installation de Symfony, il suffit de se rendre à l'adresse suivante : <http://localhost:8000> pour obtenir cette page, signe que Symfony est bien installée.

- Étape 6

Une fois installé, Symfony nécessite quelques paramétrages, comme par exemple le nom de la base de données à laquelle se connecter ou bien le port qu'il faut utiliser. Pour cela, tout se passe dans le fichier « parameters.yml », ce fichier est généré automatiquement lors de l'installation de symfony.

```
2  parameters:
3      database_host: 127.0.0.1
4      database_port: 8889
5      database_name: movie_website
6      database_user: root
7      database_password: root
8      mailer_transport: smtp
9      mailer_host: 127.0.0.1
10     mailer_user: mailer_user
11     mailer_password: null
12     secret: 33dd5a2ba1785e46cd34715155320c84caa09b6f
13
```

Tous les paramètres concernant la base de données se feront dans ce fichier.

7.2. Symfony3 et Doctrine2

Voyons maintenant comment utiliser Doctrine. Pour commencer, nous allons d'abord créer une base de données en ligne de commande grâce à la commande :

```
php bin/console doctrine:database:create
```

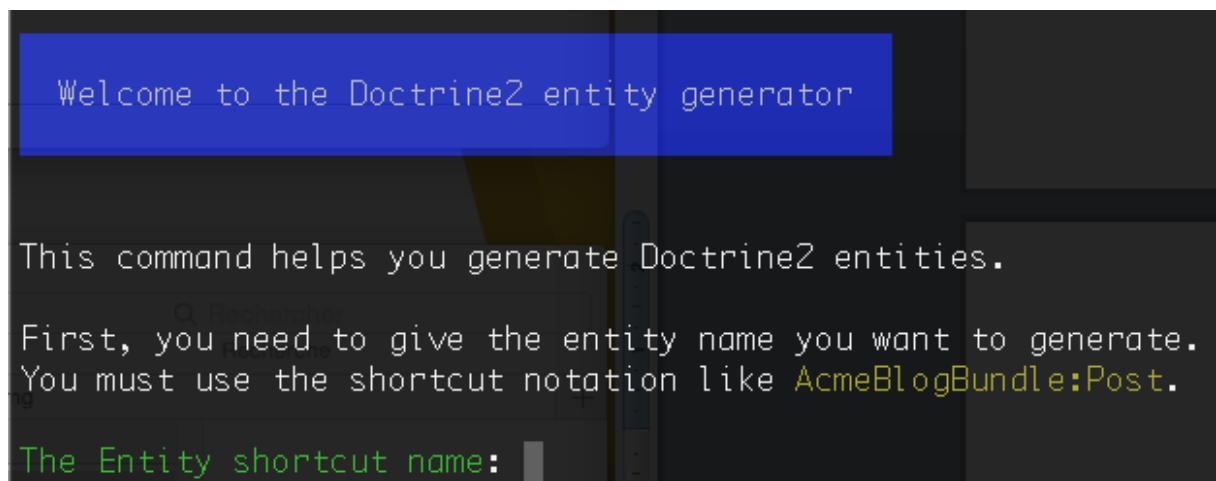
Ceci permettra par la suite de pouvoir mettre à jour le schéma de notre base de données de manière « dynamique ».

○ Création du modèle

La couche modèle est un ensemble de classe PHP placé dans le dossier « Entity » du Bundle que l'on aura créé auparavant. Pour se faire, nous allons générer nos entités, correspondant aux noms de classe en somme.

```
php bin/console doctrine:generate:entity
```

Une fois cette commande entrée dans l'invite de commande, cet écran apparaît.



Il suffit d'entrer un nom d'entité correspondant au format suivant : NomDuBundle:NomEntité. Ensuite, nous avons la possibilité d'entrer plusieurs noms de champs avec ses propriétés (son type, s'il est null ou non...). Lorsque l'on a terminé, si on va dans le dossier « Entity » on se rend compte qu'un fichier portant le nom de l'entité créée est présent. Si on l'ouvre, on peut voir qu'il y a déjà les *getters*¹ et *setters*²

Pour valider le modèle, il suffit de faire la commande suivante :

```
php bin/console doctrine:schema:validate
```

¹ Getter : C'est une méthode qui permet de récupérer un attribut dans une classe.

² Setter : C'est une méthode qui permet de modifier la valeur d'un attribut dans une classe.

Si tout se passe bien, le message suivant apparaîtra :

```
[Mapping] OK - The mapping files are correct.
[Database] OK - The database schema is in sync with the mapping files.
```

Dans le cas contraire, les erreurs apparaîtront ici. Une fois la modèle validé, nous pouvons mettre à jour notre base de données. On va d'abord vérifier que tout est bon et seulement ensuite, on appliquera pour que les modifications soient prises en compte dans notre base de données.

```
php bin/console doctrine:schema:update --dump-sql
```

S'il n'y a aucune erreur mentionnée alors on peut exécuter la seconde commande

```
php bin/console doctrine:schema:update --force
```

Et maintenant, si on retourne dans notre base de données, on peut voir les modifications effectuées.

- Peupler le modèle dans un contrôleur

Passons désormais au contrôleur, pour qu'il puisse être lié avec le modèle, il y a quelques lignes à rajouter.

- Référencer le modèle :

```
8 //Modèle Film
9 use AppBundle\Entity\Film;
```

- Récupérer le manager Doctrine :

```
$em = $this->getDoctrine()->getManager();
```

- Récupérer le *repository*¹ :

```
//Récupération du repository
$repository = $this->getDoctrine()->getRepository('AppBundle:Film');
```

¹ Repository : Ce dossier est créé automatiquement lorsque l'on crée les entités avec Doctrine en ligne de commandes. Dans celui-ci se trouvent les fichiers correspondant aux entités. Son but est double : d'une part, il contient du code qui pourra être réutilisable par la suite, et d'autre part, éviter de surcharger le contrôleur.

Une fois ces deux lignes ajoutées, il est alors possible de commencer le développement. Il faudra faire cette manipulation dans chacun des contrôleurs que nous allons créer tout au long du projet.

- FOSUser Bundle

Pour la suite du projet, nous allons avoir besoin d'un bundle tiers pour gérer les utilisateurs, notamment pour l'inscription, l'authentification etc.

Avant toute chose, nous devons installer *Composer*¹ qui permet de gérer les bundles.

```
◆ php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
◆ php -r "if (hash_file('SHA384', 'composer-setup.php') ===
    '55d6eade61b29c7bdee5cccfb50076874187bd9f21f65d8991d46ec5cc90518
    f447387fb9f76ebae1fbbacf329e583e30') { echo 'Installer verified'; } else {
    echo 'Installer corrupt'; unlink('composer-setup.php'); } echo PHP_EOL;"
◆ php composer-setup.php
◆ php -r "unlink('composer-setup.php');"
```

Après avoir tapé ces quatre commandes, nous allons vérifier que le fichier « composer.phar » est dans notre projet. Si c'est bel et bien le cas, nous pouvons alors, à la racine de notre projet dans la console, taper la commande suivante :

```
php composer.phar selfupdate
```

Cette dernière permet de mettre à jour le fameux fichier « composer.phar » que nous venons d'installer.

```
php composer.phar update
```

Celle-ci va nous permettre de mettre à jour tous les bundles.

Et enfin, les derniers pré requis avant de pouvoir utiliser ce bundle, c'est d'ajouter cette ligne dans le fichier de config.

```
framework:
    translator: ~
```

Nous allons donc récupérer le bundle en question grâce à cette commande :

```
php composer.phar require friendsofsymfony/user-bundle "~2.0@dev"
```

Il nous reste alors quelques configurations à effectuer pour que tout cela soit fonctionnel.

¹ Composer : C'est un outil qui permet de gérer les dépendances (bibliothèques) PHP. Il ne fait en aucun cas partie de Doctrine.


```
new FOS\UserBundle\FOSUserBundle(),
```

Notamment l'ajout de cette ligne dans « AppKernel.php » afin de le référencer.

Ensuite, nous alors devons créer un fichier nommé « User.php » dans le dossier « Entity » de « AppBundle ». Il étendra du fichier « BaseUser » du bundle.

Reste maintenant à gérer la sécurité. On va alors récupérer la sécurité par défaut de FOSUserBundle

```
fos_user:
  db_driver: orm # other valid values are 'mongodb' and 'couchdb'
  firewall_name: main
  user_class: AppBundle\Entity\User
  from_email:
    address: "%mailer_user%"
    sender_name: "%mailer_user%"
```

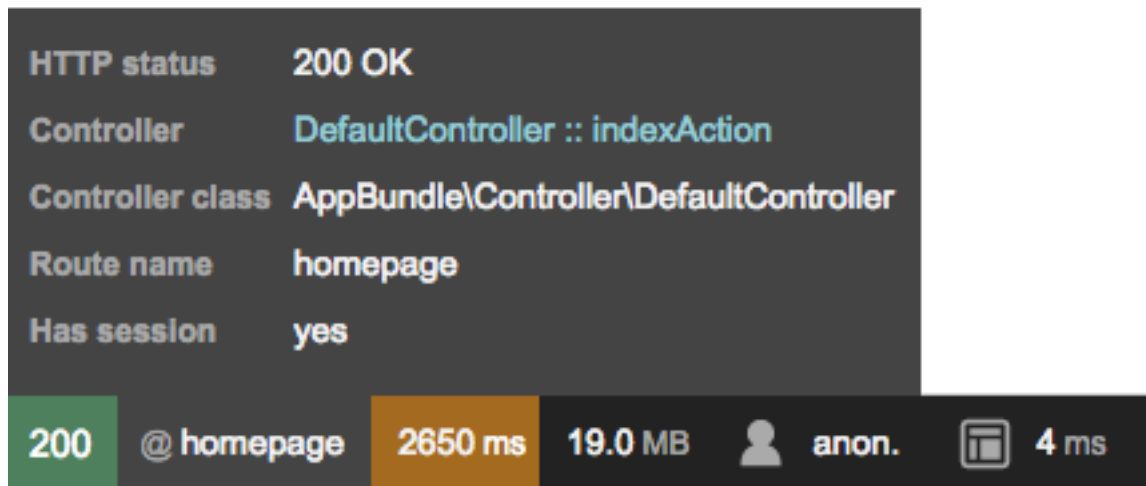
Il ne faut bien évidemment pas oublier de déclarer « %mailer_user % » dans le fichier des variables globales.

Dernière étape, le routage du bundle. Pour cela, nous allons ajouter la commande suivante dans le fichier « routing.yml »

```
fos_user:
  resource: "@FOSUserBundle/Resources/config/routing/all.xml"
```

Il faut désormais mettre à jour la base de données avec les commandes qu'on a vu plus haut et ensuite, on peut tester les fonctionnalités. Pour cela, il suffit de mettre l'URL du projet « localhost:8000/app_dev.php/nomDeLaRouteSouhaitée » pour voir le résultat. Nous avons alors le loisir de surcharger certaines des vues, en les plaçant selon la norme de nommage définie pour pouvoir insérer des éléments personnels.

Remarque : Dans l'URL du projet, le « app_dev.php » sert en mode développement pour voir les actions réalisées par le contrôleur ou encore les éventuelles erreurs. Cela donne naissance à cette barre :



Le statut « 200 » ici nous indique que tout s'est bien passé. Lorsqu'il y aura une erreur, cette barre de notification nous sera très utile pour trouver d'où elle provient.

7.3. Développement du projet

○ Modèle

Après avoir étudié le modèle de la base de données, j'ai commencé par créer les relations entre les différentes tables de celle-ci. Comme on a pu le voir auparavant, la table Film est au cœur de tout, par conséquent, il était évident que je commence par elle.

Nous avons vu sur le schéma qu'un film pouvait avoir un seul état, tandis qu'un état pouvait être relié à plusieurs films. C'est donc une relation que l'on appellera ManyToOne (définie par Doctrine). D'après la documentation officielle, dans ce type de relation, une propriété doit être rajouté du côté où la cardinalité est à 1. Ici, c'est donc notre table Film.

```
/**
 * @var string
 *
 * @ORM\Column(name="filmEtat", type="string", length=255)
 *
 * Un film a un état. Un état correspond à plusieurs films
 * @ORM\ManyToOne(targetEntity="Etat")
 * @ORM\JoinColumn(name="filmEtat", referencedColumnName="idEtat")
 */
private $filmEtat;
```

Voici donc la syntaxe pour relier la table Film et la table Etat par une clé étrangère. Le résultat sera donc un nouveau champ (ici nommé « filmEtat ») présent dans la table Film afin de savoir pour un film donné quel est son état. Autrement dit s'il est disponible ou déjà emprunté.

```

/**
 * Set filmEtat
 *
 * @param \AppBundle\Entity\Etat $filmEtat
 *
 * @return Film
 */
public function setFilmEtat(\AppBundle\Entity\Etat $filmEtat = null){
    $this->filmEtat = $filmEtat;

    return $this;
}

/**
 * Get filmEtat
 *
 * @return \AppBundle\Entity\Etat
 */
public function getFilmEtat(){
    return $this->filmEtat;
}

```

Grâce à Doctrine, nous avons la possibilité de créer automatiquement les getters et setters à l'aide d'une simple commande. Une fois que les accesseurs sont créés, nous en avons fini avec le modèle. Passons au contrôleur.

○ Contrôleur

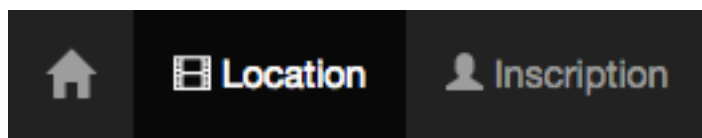
```

1  <?php
2
3  namespace AppBundle\Controller;
4
5  use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;
6  use Symfony\Bundle\FrameworkBundle\Controller\Controller;
7
8  //Modèle Film
9  use AppBundle\Entity\Film;
10
11 class FilmController extends Controller{
12
13     /**
14      * @Route("/film", name="film")
15      */
16     public function affichageFilmAction(){
17
18         //Récupération du repository
19         $repository = $this->getDoctrine()->getRepository('AppBundle:Film');
20
21         //Récupération de tous les films de la BDD
22         $films = $repository->findAll();
23
24         //Retourne une vue
25         return $this->render('AppBundle:film:film.html.twig', array('films' => $films));
26         //nom de la variable qu'on veut récupérer
27     }
28 }
29 ?>

```

Voici une explication du code :

- *Ligne 9* : On référence le modèle pour pouvoir faire appel aux propriétés présentes dans celui-ci.
- *Lignes 13 à 15* : Ce sont les annotations définies par Doctrine, il y a aussi la possibilité de les faire dans d'autres langages (yaml, xml..). On définit ici la route, c'est-à-dire le chemin qu'on indiquera dans l'URL ainsi qu'un nom, qui sera utile en cas d'éventuel lien vers cette même route.
- *Ligne 25* : Une fois qu'on a codé la logique métier que l'on désirait, on va ensuite pouvoir dire au contrôleur le nom de la vue vers laquelle on désire qu'il affiche le résultat. La fonction `render()` prend deux paramètres, le premier est le chemin souhaité répondant à une convention de nommage bien particulière dont voici la correspondance : `NomDuBundle:dossierIntermédiaire (s'il y a):nomDeLaVue`. Quant au second paramètre il correspond à la variable que l'on veut récupérer (donc ici la variable `film`).



- [Liste des films](#)
- [Liste des acteurs](#)

Voici donc l'aperçu lorsqu'on arrive sur le site. On a donc la possibilité de consulter la liste des films ou bien celles des acteurs.

```

1  {% extends 'base.html.twig' %}
2
3  {% block body %}
4      <div id="wrapper">
5          <div id="contentIndex">
6
7              <ul>
8                  <li><a href="{{ path('liste_film') }}">Liste des films</a></li>
9                  <li><a href="{{ path('liste_acteur') }}">Liste des acteurs</a></li>
10             </ul>
11
12         </div>
13     </div>
14 {% endblock %}
15
16 {% block stylesheets %}{% endblock %}

```

Pour cela, il a suffit d'ajouter dans le code le chemin relatif aux routes qu'on a défini dans le contrôleur film.

○ Vue

```



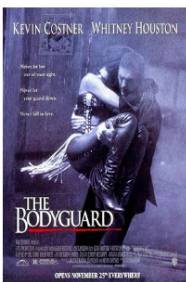

1  {% extends 'base.html.twig' %}
2
3  {% block body %}
4
5      <h1>Liste des films</h1>
6
7      <table class="table table-bordered table-inverse">
8          <thead>
9              <th class="contentTable">Couverture</th>
10             <th class="contentTable">Titre</th>
11             <th class="contentTable">Description</th>
12             <th class="contentTable">Date de sortie</th>
13             <th class="contentTable">Durée</th>
14             <th class="contentTable">État</th>
15             <th class="contentTable">Catégorie</th>
16          </thead>
17
18          {% for film in films %}
19              <tbody>
20                  <td></td>
21                  <td id="title">{{ film.titre }}</td>
22                  <td id="desc">{{ film.description }}</td>
23                  <td id="date">{{ film.dateSortie|date("Y") }}</td>
24                  <td id="time">{{ film.duree|date("H:i") }}</td>
25                  <td id="state">{{ film.filmEtat }}</td>
26                  <td id="category">{{ film.moviesByCategory }}</td>
27              </tbody>
28          {% endfor %}
29      </table>
30
31  {% endblock %}
32

```

Voyons donc le code présent dans la vue maintenant. Il est relativement simple à comprendre puisque ce n'est que du HTML. La seule petite particularité avec Twig, c'est la syntaxe. Notamment la première ligne qui représente un héritage de la vue « base.html.twig ». Twig est beaucoup plus lisible et compréhensible dans sa structure, puisqu'il fonctionne en bloc.

Et pour finir, voici le rendu sur le site.

Liste des films

Couverture	Titre	Description	Date de sortie	Durée	État	Catégorie
	La fille du train	Rachel prend tous les jours le même train et passe tous les jours devant la même maison. Dévastée par son divorce, elle fantasme sur le couple qui y vit et leur imagine une vie parfaite... jusqu'au jour où elle est le témoin d'un événement extrêmement choquant et se retrouve malgré elle étroitement mêlée à un angoissant mystère.	1970	01:53	Emprunté	Thriller, Action
	Le professionnel	Issu de l'élite de l'armée française, Joss Beaumont est chargé d'exécuter le président de la Malagawi. Un contre-ordre tombe, la cible est devenue un ami de l'Etat. Pour l'empêcher de nuire, Beaumont est incarcéré, mais ne tarde pas à s'évader, décidé à mener à bien l'opération malgré l'opposition de sa hiérarchie.	1970	01:45	Disponible	Action, Policier
	Bodyguard	Frank Farmer, ancien agent des services secrets, est un garde du corps emerite qui a mis ses talents a la disposition de deux presidents et de nombreux financiers et politiciens de reputation internationale. Un jour l'impresario Bill Devaney lui propose un contrat avantageux pour assurer la protection de sa cliente Rachel, comedienne et chanteuse en pleine ascension, menacée par un fan inconnu.	1970	02:09	Emprunté	Action, Film d'aventure
	Pretty Woman	Edward Lewis, homme d'affaires performant, rencontre par hasard Vivian Ward, beaute fatale qui arpente chaque nuit les trottoirs d'Hollywood Boulevard. La jeune femme ne fera qu'une bouchée du brillant PDG.	1970	01:59	Emprunté	Action

L'important est la dernière colonne dans laquelle figure l'état du film.

La suite est sensiblement identique à quelques petits détails.

```

/**
 * @var string
 *
 * @ORM\Column(name="idCat", type="string", length=255)
 *
 * Many movies have many categories
 * @ORM\ManyToMany(targetEntity="Categorie")
 * @ORM\JoinTable(name="film_categorie",
 *     joinColumns={@ORM\JoinColumn(name="numFilm", referencedColumnName="numFilm")},
 *     inverseJoinColumns={@ORM\JoinColumn(name="idCat", referencedColumnName="idCat")})
 */
private $moviesByCategory;

```

Modèle

```
/**
 * Set moviesByCategory
 *
 * @param string $moviesByCategory
 *
 * @return Film
 */
public function setMoviesByCategory($moviesByCategory){
    $this->moviesByCategory = $moviesByCategory;

    return $this;
}


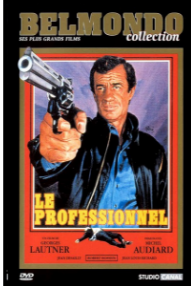
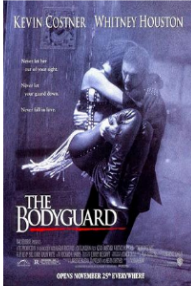

/**
 * Get moviesByCategory
 *
 * @return string
 */
public function getMoviesByCategory(){
    return $this->moviesByCategory;
}
```

Getters/setters

	Table ▲
<input type="checkbox"/>	acteur
<input type="checkbox"/>	categorie
<input type="checkbox"/>	etat
<input type="checkbox"/>	film
<input type="checkbox"/>	film_categorie
<input type="checkbox"/>	fos_user

Petit supplément par rapport à tout à l'heure, ici nous avons une table intermédiaire qui s'est créée à cause de la relation ManyToMany.

Liste des films

Couverture	Titre	Description	Date de sortie	Durée	État	Catégorie
	La fille du train	Rachel prend tous les jours le même train et passe tous les jours devant la même maison. Dévastée par son divorce, elle fantasme sur le couple qui y vit et leur imagine une vie parfaite... jusqu'au jour où elle est le témoin d'un événement extrêmement choquant et se retrouve malgré elle étroitement mêlée à un angoissant mystère.	1970	01:53	Emprunté	Thriller, Action
	Le professionnel	Issu de l'élite de l'armée française, Joss Beaumont est chargé d'exécuter le président de la Malagawi. Un contre-ordre tombe, la cible est devenue un ami de l'Etat. Pour l'empêcher de nuire, Beaumont est incarcéré, mais ne tarde pas à s'évader, décidé à mener à bien l'opération malgré l'opposition de sa hiérarchie.	1970	01:45	Disponible	Action, Policier
	Bodyguard	Frank Farmer, ancien agent des services secrets, est un garde du corps emerite qui a mis ses talents a la disposition de deux presidents et de nombreux financiers et politiciens de reputation internationale. Un jour l'impresario Bill Devaney lui propose un contrat avantageux pour assurer la protection de sa cliente Rachel, comedienne et chanteuse en pleine ascension, menacee par un fan inconnu.	1970	02:09	Emprunté	Action, Film d'aventure
	Pretty Woman	Edward Lewis, homme d'affaires performant, rencontre par hasard Vivian Ward, beaute fatale qui arpente chaque nuit les trottoirs d'Hollywood Boulevard. La jeune femme ne fera qu'une bouchée du brillant PDG.	1970	01:59	Emprunté	Action

Nous nous retrouvons donc avec un rendu similaire au précédent, avec la colonne « Catégorie » en plus.

- FOSUserBundle

Grâce à ce bundle, nous avons déjà plusieurs méthodes qui sont fournies avec. Voici un petit aperçu des routes qui ont été déclarées au sein du bundle.

fos_user_security_login	GET POST	ANY	ANY	/login
fos_user_security_check	POST	ANY	ANY	/login_check
fos_user_security_logout	GET POST	ANY	ANY	/logout
fos_user_profile_show	GET	ANY	ANY	/profile/
fos_user_profile_edit	GET POST	ANY	ANY	/profile/edit
fos_user_registration_register	GET POST	ANY	ANY	/register/
fos_user_registration_check_email	GET	ANY	ANY	/register/check-email
fos_user_registration_confirm	GET	ANY	ANY	/register/confirm/{token}
fos_user_registration_confirmed	GET	ANY	ANY	/register/confirmed
fos_user_resetting_request	GET	ANY	ANY	/resetting/request
fos_user_resetting_send_email	POST	ANY	ANY	/resetting/send-email
fos_user_resetting_check_email	GET	ANY	ANY	/resetting/check-email
fos_user_resetting_reset	GET POST	ANY	ANY	/resetting/reset/{token}
fos_user_change_password	GET POST	ANY	ANY	/profile/change-password

C'est donc avec ces routes que nous pourrions circuler dans le projet pour obtenir ce que l'on désire.

Pour commencer, intéressons nous à la connexion. Voici le layout prédéfini dans le bundle FOSUser.

```

1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="UTF-8" />
5
6          <!-- Fichiers CSS personnel -->
7          <link rel="stylesheet" type="text/css" href="{{ asset('bundles/app/css/style.css') }}" />
8      </head>
9      <body>
10         <div>
11             {% if is_granted("IS_AUTHENTICATED_REMEMBERED") %}
12                 {{ 'layout.logged_in_as'|trans({'%username%': app.user.username}, 'FOSUserBundle') }} |
13                 <a href="{{ path('fos_user_security_logout') }}">
14                     {{ 'layout.logout'|trans({}, 'FOSUserBundle') }}
15                 </a>
16             {% else %}
17                 <a href="{{ path('fos_user_security_login') }}">{{ 'layout.login'|trans({}, 'FOSUserBundle') }}</a>
18             {% endif %}
19         </div>
20
21         {% if app.request.hasPreviousSession %}
22             {% for type, messages in app.session.flashbag.all() %}
23                 {% for message in messages %}
24                     <div class="flash-{{ type }}">
25                         {{ message }}
26                     </div>
27                 {% endfor %}
28             {% endfor %}
29         {% endif %}
30
31         <div>
32             {% block fos_user_content %}
33             {% endblock fos_user_content %}
34         </div>
35     </body>
36 </html>

```

Ici c'est la vue qui affichera le formulaire de connexion.

```

1  {% trans_default_domain 'FOSUserBundle' %}
2
3  {% if error %}
4      <div>{{ error.messageKey|trans(error.messageData, 'security') }}</div>
5  {% endif %}
6
7  <form action="{{ path('fos_user_security_check') }}" method="post">
8      {% if csrf_token %}
9          <input type="hidden" name="_csrf_token" value="{{ csrf_token }}" />
10     {% endif %}
11
12     <label for="username">{{ 'security.login.username'|trans }}</label>
13     <input type="text" id="username" name="_username" value="{{ last_username }}" required="required" /><br>
14
15     <label for="password">{{ 'security.login.password'|trans }}</label>
16     <input type="password" id="password" name="_password" required="required" /><br>
17
18     <input type="checkbox" id="remember_me" name="_remember_me" value="on" />
19     <label for="remember_me">{{ 'security.login.remember_me'|trans }}</label><br><br>
20
21     <input type="submit" id="_submit" name="_submit" value="{{ 'security.login.submit'|trans }}" />
22 </form>

```

Ce qui donne ceci au niveau du front-end.

[Log in](#)

Username Password ☐ Remember me

Une fois connecté, l'utilisateur peut alors louer un film, et le processus est le même pour chacune des locations de films.

8. Conclusion

This year of alternation was very enriching for me because she allowed me to discover in detail the sector of the Web development, her actors, the constraints and existence of some aspects which we sometimes ignore with a simple school vision. It also allowed me to participate concretely in the stakes through my varied missions.

This year also allowed me to have an overview on the skills of the various members of the team, in particular the fact that they complement each other perfectly on certain tasks. I learn a lot of the part of customers, where I was able to attend one of the meetings to see the process.

With this experience and in answer to its stakes, I shall afterward like trying very much to direct me to the graphic side which interests me a lot, without disregard the development in itself. I shall like having the possibility of evolving in the development front-end which is the one which really attracts me, in spite of during my previous formation, it is the back-end which dominated.