

# **DOSSIER PROJET**

**Kévin CHALLIT**

Médiathèque La Chapelle-Cureaux

« Développeur Web et Web mobile »

***École Studi***

# Sommaire

• <b><u>Sommaire</u></b> .....	<b>p.2</b>
• <b><u>Introduction</u></b> .....	<b>p.3</b>
1. Compétence du référentiel couvertes par le projet ..	p.3
2. Résumé du projet .....	p.4
• <b><u>Conceptualisation de l'application</u></b> .....	<b>p.5</b>
1. Cahier de charges et spécifications fonctionnelles ....	p.5
2. Environnement technique .....	p.7
• <b><u>Mise en place de l'application</u></b> .....	<b>p.8</b>
1. Diagramme de classe .....	p.8
2. Diagrammes de séquence.....	p.9
3. Wireframe .....	p.10
4. Feuille de style .....	p.11
• <b><u>Phase de développement</u></b> .....	<b>p.12</b>
1. Initialisation de l'environnement de travail .....	p.12
2. Authentification & Inscription .....	p.14
3. Gestion des utilisateurs .....	p.16
4. Le catalogue & sa gestion .....	p.18
5. Les emprunts & leurs gestions .....	p.21
6. La barre de recherche dynamique .....	p.24
7. Les vues Twig .....	p.25
8. La sécurité de l'application .....	p.27
9. Le déploiement de l'application .....	p.29
• <b><u>Traduction d'un article anglophone</u></b> .....	<b>p.30</b>
• <b><u>Conclusion</u></b> .....	<b>p.32</b>
• <b><u>Annexes</u></b> .....	<b>p.33</b>

# Introduction

## 1. Compétences du référentiel couvertes par le projet

a. Développer la partie front-end d'une application web ou web\_mobile en intégrant les recommandations de sécurité :

- Maquetter une application.
- Développer une interface utilisateur web dynamique.
- Réaliser une interface utilisateur web statique et adaptable.

Le langage de programmation PHP permet de produire des pages web dynamiques créer à partir d'un serveur HTTP. Les pages servies seront différentes selon l'utilisateur.

L'application doit pouvoir proposer une expérience utilisateur similaire quelle que soit la taille de l'écran. J'ai donc fait en sorte que l'interface s'adapte aux différentes tailles de supports utilisés.

b. Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité :

- Créer une base de données.
- Développer les composants d'accès aux données.
- Développer la partie back-end d'une application web ou web mobile.

L'application fait appel à une base de données relationnelle que j'ai pensée selon le cahier des charges et développée.

Certaines fonctionnalités réalisées en asynchrone nécessiteront de faire appel à une API. J'ai donc pu utiliser le bundle « Api-Platform ».

De plus, mes différents appels à la base de données seront fait via Doctrine, un ORM (Object-Relational Mapping) pour PHP.

## **2. Résumé du projet**

Dans le cadre de ma formation de développeur web / web-mobile, l'évaluation en cours de formation (ECF) qui a lieu du 29 septembre au 21 octobre 2021, j'ai réalisé la plateforme web de la Médiathèque « La Chapelle-Cureaux ».

La plateforme est un espace réservé aux clients et aux employés de la médiathèque, l'utilisateur à accès au catalogue de la médiathèque et il peut réserver des livres en ligne afin de venir les chercher sur place.

L'employé a accès à un espace sécurisé où il pourra suivre et gérer les différentes fonctionnalités de la plateforme.

J'ai développé l'application avec le Framework PHP Symfony qui permet de concevoir rapidement des applications web sécurisées, et grâce à ses différents packages, de générer du code basique telle que le CRUD par exemple.

De plus, la communauté de développeurs gravitant autour de ce Framework me donne accès à une multitude de ressources, car travaillant seul sur ce projet avoir une documentation riche m'a grandement aidé.

Pour ce projet, j'ai un cahier des charges à respecter précisant les différentes fonctionnalités et règles métiers attendues au sein de l'application web, cependant nous avions une totale liberté sur les styles appliqués à condition qu'ils restent cohérents et minimalistes.

Ce projet m'a permis de perfectionner mes compétences sur ce Framework ainsi que mon anglais, ma logique algorithmique et de proposer aux recruteurs un projet complet sur mon GitHub.

# Conceptualisation de l'application

## 1. Cahier des charges et spécifications fonctionnelles

### a. Rôle au sein de l'application :

- Utilisateur non inscrit/connecté/validé : Cet utilisateur n'aura accès qu'à la partie login de l'application et à la page de création de compte, nommé « PUBLIC\_ACCESS » dans l'app.
- Utilisateur inscrit/connecté : L'utilisateur suivant aura accès au catalogue ainsi que ses fonctionnalités, il pourra aussi emprunter un livre, suivre ses emprunts en cours et modifier son profil, nommé « ROLE\_USER » dans l'app.
- L'employé : Cet utilisateur est un employé de la médiathèque, il sera le gestionnaire du catalogue, il pourra suivre et gérer les emprunts utilisateurs, accepter ou refuser les demandes d'adhésion, nommé « ROLE\_EMPLOYEE ».
- L'administrateur : Correspondra généralement à la personne qui s'occupera de développer l'application, il aura accès à toutes les fonctionnalités à des finalités de test, nommé « ROLE\_ADMIN ».

### b. Hiérarchisation des rôles :

- L'utilisateur inscrit mais non validé aura les mêmes droits que l'utilisateur non inscrit/connecté.
- L'employé héritera du rôle de l'utilisateur inscrit/connecté.
- L'administrateur héritera du rôle de l'employé (et donc aussi du rôle de l'utilisateur).

### c. Spécifications fonctionnelles & droits associés :

- **Créer un compte : Utilisateur non inscrit/connecté/validé**
  - L'utilisateur désirant s'inscrire devra renseigner : un nom, un prénom, un email valide, une date de naissance, son adresse postale et un mot de passe sécurisé.
  - L'utilisateur pourra choisir de cacher ou non son mot de passe.
- **Se connecter à la plateforme : Utilisateur non inscrit/connecté/validé**
  - L'utilisateur pourra choisir de cacher ou non son mot de passe.
- **Découvrir le catalogue : Utilisateur inscrit/connecté – employé – administrateur**
  - L'inscrit aura accès à l'intégralité du catalogue de la médiathèque. Un visuel permettra de différencier les livres disponibles de ceux qui ne le sont pas.
  - Si besoin, il peut rechercher un livre précis par son titre grâce à une barre de recherche ou filtrer par genre.

- **Emprunter un livre : Utilisateur inscrit/connecté – employé – administrateur**
  - Si le livre est disponible, un bouton associé permettra alors de l'emprunter.
  - Si l'inscrit clique sur le bouton, le livre ne sera plus accessible pour d'autres utilisateurs.
- **Voir ses emprunts en cours : Utilisateur inscrit/connecté – employé – administrateur**
  - Si l'utilisateur est en retard dans un rendu, une notification de rappel sera affichée.
- **Ajouter un livre dans le catalogue : employé - administrateur**
  - Chaque livre possède un titre, une image de la première de couverture, une date de parution, une description, un auteur, un genre.
- **Gestion du profil utilisateur : Utilisateur inscrit/connecté – employé – administrateur**
  - L'utilisateur pourra changer son adresse ainsi que son mot de passe après avoir confirmé son identité en écrivant son ancien mot de passe.
  - L'utilisateur pourra choisir de cacher ou non son mot de passe.
- **Gestion des emprunts : employé – administrateur**
  - Les employés de la médiathèque voient la liste de tous les emprunts en cours. Tout livre dont la date de récupération par un inscrit date de plus 3 semaines devra être affiché en priorité.
  - Lorsqu'un utilisateur vient rendre un livre, un employé confirme la remise et clôture l'emprunt, automatiquement, le livre devient à nouveau accessible dans le catalogue.
- **Gestion des utilisateurs : employé – administrateur**
  - Un employé devra obligatoirement vérifier les informations entrées lors de l'enregistrement d'un nouvel utilisateur. S'il décide de confirmer la création du compte, alors l'utilisateur pourra se connecter sur la plateforme, sinon le compte sera clôturé.

## 2. Environnement Technique

Je travaille à mon domicile sur un ordinateur de bureau avec Windows 10, il m'arrive aussi de travailler sur un Mac lors de certains de mes déplacements.

Pour l'écriture de l'application, j'utilises PHP Storm.

- **Serveur local :**
  - J'utilise le serveur de développement local fourni par Symfony.
  - PHP v8.0.6
  - Serveur HTTP Apache
- **Serveur de production :**
  - J'utilise la solution de service offert par Heroku pour déployer l'application sur le web.
  - PHP 8.0.11(cli)
  - Add-on « JawsDB Maria » pour la base de données en ligne.
- **Partie Front :**
  - HTML 5
  - CSS 3
  - Framework CSS « MaterializeCSS » orienté Material Design.
  - Javascript Vanilla ES6 minimum
- **Partie Back :**
  - Symfony v5.3.9
  - PHP 7.4 à 8.0
  - Base de données relationnelle Maria DB v10.4.19 avec le moteur InnoDB.
- **Outils tiers nécessaire lors du projet :**
  - Git pour le versionning de l'application, et GitHub pour la sauvegarde sur le cloud.
  - Postman pour le test des différentes routes de mon API et la gestion des données envoyé par elle.
  - Composer pour l'installation des packages du projet.
  - Wireframe.cc pour la réalisation des wireframes.
  - LucidChart et Visual Paradigm Online pour la réalisation des différents diagrammes UML.
  - XAMPP : Serveur web local, utilisé uniquement pour la base de données.

# Mise en place de l'application

## 1. Diagramme de classe

La première étape fut la réalisation d'un diagramme de classe (schéma de représentation de ma future base de données) afin de définir les entités ainsi que leurs associations et leurs cardinalités, je me suis appuyé sur le cahier des charges et de LucidChart pour la réalisation de ce document.

### a. Les tables & leurs colonnes :

- **Table « Users »** : contiendra tous les utilisateurs, leurs rôles pour la sécurité, leurs différentes coordonnées, le statut de validation de leur compte.
- **Table « Books »** : contiendra tous les livres du catalogue, leurs disponibilités et le chemin vers l'image.
- **Table « BooksReservations »** : contiendra tous les emprunts utilisateur, le statut de collecte du livre, la date de réservations du livre et le cas échéant la date de collecte de celui-ci.
- **Table « Categories »** : contiendra le nom des genres des livres.
- **Table « Authors »** : contiendra les auteurs des différents livres.

### b. Associations & cardinalités :

- Un « **Users** » pourra emprunter plusieurs « **Books** » (One-To-Many) et un « **Books** » ne pourra être emprunter que par un utilisateur (Many-to-One).
- Chaque emprunt utilisateur créera une ligne dans la table d'association « **BooksReservations** » qui ne pourra seulement détenir qu'un unique « **Books** » (One-To-One)

- Un « **Books** » détiendra au minimum une « **Categories** » et un « **Authors** » (Many-To-Many), une table d'association sera créée par Symfony lors de la migration Doctrine.

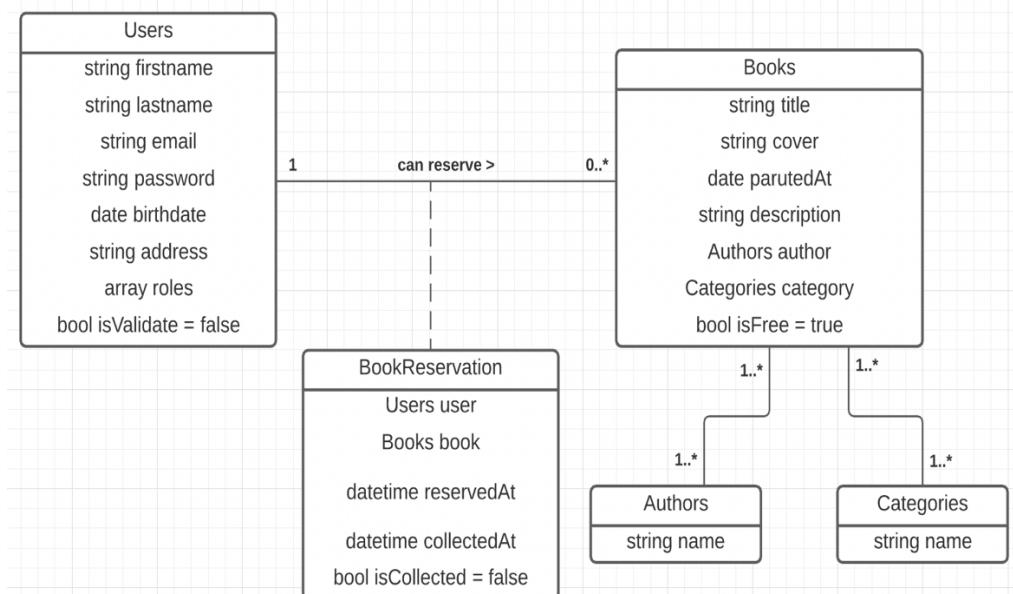


Diagramme de classe

## 2. Diagrammes de séquence

J'ai ensuite réalisé plusieurs diagrammes de séquence pour déterminer comment les différents acteurs et mon application réagiront entre eux.

### a. Cas n°1 - Rendre un livre :

L'acteur de ce diagramme est l'employé.

L'acteur souhaitant accéder à cette fonctionnalité devra se rendre dans la page de gestion des emprunts, l'application requêtera la base de données afin d'obtenir la liste de tous les emprunts de livre en cours qui lui renverra dans la vue.

Quand l'employé confirme le rendu du livre, l'application supprimera l'emprunt et demandera le changement de disponibilité du livre à la base de données qui le rendra disponible.

Un message sera affiché à l'utilisateur lui confirmera la clôture de l'emprunt.

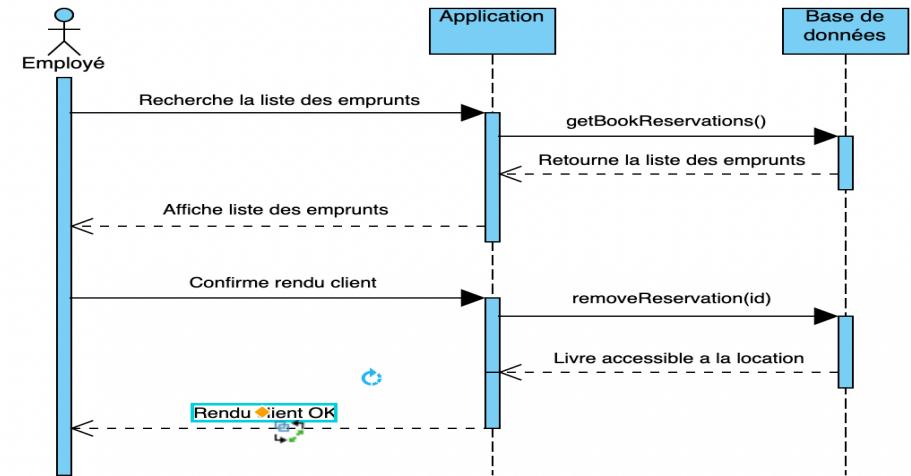


Diagramme de séquence 1 – Rendre un livre

### b. Cas n°2 - Emprunter un livre :

Les acteurs dans cette séquence sont les utilisateurs inscrit, validé et connecté ainsi que les employés (et l'administrateur à des finalités de tests fonctionnelles).

L'acteur se situant dans la page de catalogue recherchera un livre (par l'intermédiaire de la barre de recherche ou en filtrant les livres, ou simplement dans la liste entière).

Une fois que l'acteur aura choisi son livre dans la liste, l'application ira le rechercher en base de données et renverra la page de détail du livre.

L'utilisateur décidant d'emprunter le livre, l'application interrogera la base de données qui confirmera la disponibilité du livre et mettra le statut du livre à non disponible puis créera l'emprunt dans la base de données et pour finir renverra un message de succès à l'utilisateur.

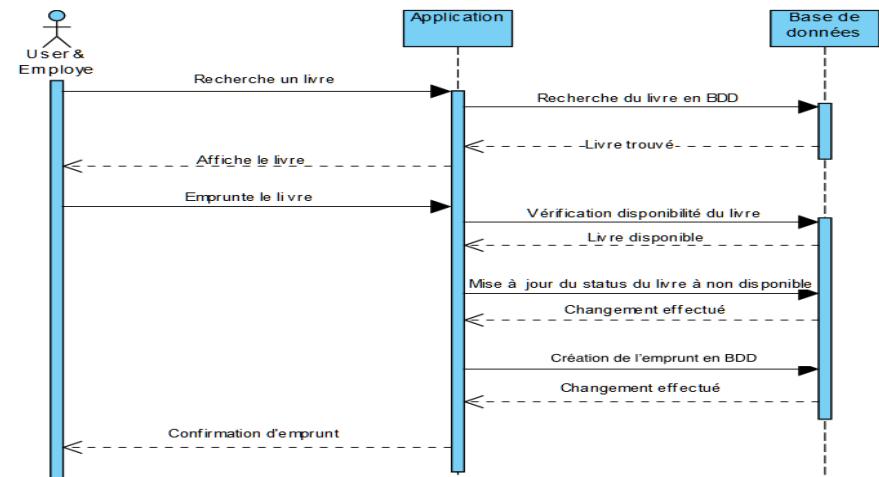


Diagramme de séquence 2 – Emprunter un livre

### 3. Wireframes

Les diagrammes concernant les spécifications fonctionnelles achevés, j'ai entrepris la conception des wireframes pour définir les zones et composants que chaque page doit contenir.

J'ai procédé à la réalisation des wireframes suivant sur le site « wireframe.cc » :

- Page de connexion
- Page de création de compte
- Page du catalogue
- Page de détail du livre
- Les pages de gitations administrateur qui ont le même template.

Les éléments sont représentés par des traits fin et noir, le texte avec du lorem ipsum exception des éléments importants et de l'entête et les images sont dans un rectangle blanc barré

The wireframe shows a desktop browser window. At the top, there's a header with a logo (crossed books), the text "Médiathèque La Chapelle-Cureaux", and navigation links: "Catalogue", "Administration", "Mon compte", and "Déconnexion". Below the header is a search bar containing "Rechercher un livre", a dropdown menu labeled "Genre", and a clear button. The main content area displays two book entries. Each entry has a thumbnail icon (crossed books), the title "Lorem ipsum dolor sit amet", a short description, and a status indicator: "Disponible" with a radio button (selected) and "Non disponible" with a radio button (unchecked). Both entries have a "Voir le livre" button at the bottom. At the very bottom of the page is a footer with the text "Made By Kevin CHALLIT".

Wireframe version bureau – Page catalogue

The wireframe shows a mobile device screen. At the top, there's a header with a logo (crossed books), the text "Médiathèque La Chapelle-Cureaux", and a "Menu" button. Below the header is a search bar with "Rechercher un livre" and a dropdown menu labeled "Genre". The main content area displays three book entries. Each entry has a thumbnail icon (crossed books), the title "Lorem ipsum dolor", a short description, and a status indicator: "Disponible" with a radio button (unchecked). The first two entries have a "Voir le livre" button at the bottom. At the very bottom is a footer with the text "Made By Kevin CHALLIT".

Wireframe version mobile – Page catalog

## 4. Feuille de style

Dernière étape dans la mise en place du projet, j'ai établi la charte graphique, nous n'avions qu'une seule consigne qui était que le design du site soit simple, épuré et minimaliste mais il se devait de rester dans le thème.

Les polices d'écritures ont été trouvé sur le site de Google Fonts.

Elles sont simples et non fantaisistes.

La police « Righteous » est utilisé seulement pour l'entête du site, tandis que la police « Overlock » sera utilisé pour le reste.

La couleur de police peut être noire ou blanche selon l'opacité du bleu (blanc sur du foncé et noir sur le fond plus claire).

La couleur principale est le bleu clair « #0DC9C9 » et je réduis l'opacité de 90% pour la couleur de fond des pages.

### Feuille de style Médiathèque La Chapelle-Curreaux

#### Police de 1er niveau: Righteous



#000000



#0DC9C9



#FFFFFF

#### Police de 2eme niveau: Overlock

  Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur eget auctor diam. Phasellus nec eleifend neque, nec posuere felis. Duis id pretium ex

BUTTON 1

BUTTON 2

# Développement de l'application

## 1. Initialisation de l'environnement de travail

### a. Génération du projet & de ses dépendances :

La génération du projet requiert d'avoir Composer installer sur son PC. Après ouverture d'un terminal de commande, j'ai simplement tapé la commande : « **composer create-project symfony/skeleton mediatheque-la-chapelle-cureaux** » qui génère automatiquement un projet Symfony vierge.

J'ai ouvert le projet et j'ai également installé les bundles nécessaires au projet avec la commande Composer « **composer req nomDuBundle** » :

- **Twig** : Moteur de Template pour PHP, permet la génération de mes vues.
- **Doctrine** : ORM pour PHP, gestion de mes entités et de la communication avec la base de données.
- **Symfony Forms** : Générateur de formulaire, permettra de sécuriser les données entrantes.
- **Validator** : Fonctionne avec Symfony Forms et Doctrine en apposant des contraintes sur les données entrantes.
- **Security Bundle** : Gestion de la sécurité de l'application (ex : authentification, protections des routes...)
- **Api-Platform** : Génération de l'API REST.
- **Annotation** : Permet l'annotation au-dessus des contrôleurs et des entités pour leurs configurations.
- **Maker (--dev)** : Générateur d'entités, de formulaires, de contrôleurs etc...
- **Profiler (--dev)** : Débuggeur Symfony, permet d'accéder aux détails de la requête exécuté.
- **& d'autres...**

Lors de l'installation des différents bundles, différents fichiers ont été générés, je me suis porté sur le fichier « **.env.local** » créer à la racine du projet lors de l'installation de Doctrine pour y rajouter les identifiants de ma base de données locale.

Un fichier « **security.yaml** » est aussi créer dans le dossier « config » lors de l'installation de « Security », j'y enregistre mes différents rôle au sein de l'application. (Liste des rôles disponible plus haut)

## b. Génération des entités & migration en base de données :

Une entité est une classe PHP représentant une table, ses champs et ses propriétés dans la base de données.

Avec le Maker-Bundle, la génération des entités est simplifiée et rapide avec seulement un formulaire à remplir après l'exécution de la commande PHP :

**« php bin/console make:entity nomDeLentity »**

Je me suis donc appuyé sur le diagramme de classe pour générer les différentes entités.

J'ai pris ensuite le temps de relire et de corriger certaines annotations.

Exemple : générer l'API à partir de l'entité ou encore changer la stratégie de génération de l'ID des utilisateurs (cf. photo 1).

Avant d'effectuer la migration en base de données, j'ai créé la base de données hébergée sur mon serveur locale XAMPP nommé «mediatheque »

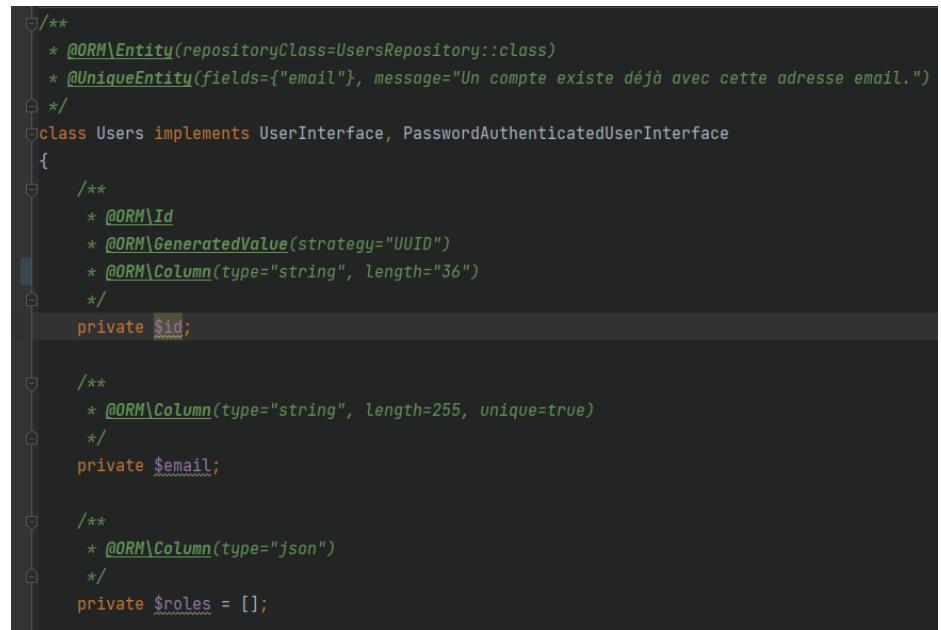
Grâce aux commandes Doctrine, j'ai généré le fichier de migrations à partir des entités créées précédemment contenant les requêtes SQL (cf. photo 2).

**« php bin/console make:migrations »**

Après vérification et correction, la commande Doctrine suivante m'a permis d'exécuter le fichier de migration.

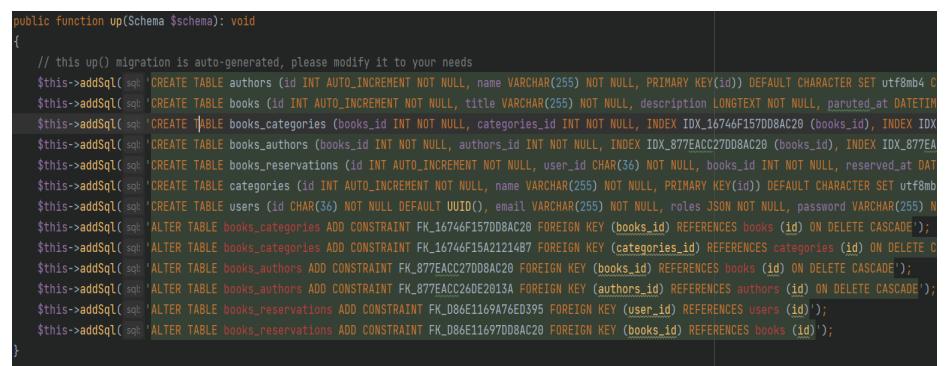
**« php bin/console doctrine:migrations:migrate »**

La base de données est créée et connecté à mon application.



```
/** * @ORM\Entity(repositoryClass=UsersRepository::class) * @UniqueEntity(fields={"email"}, message="Un compte existe déjà avec cette adresse email.") */ class Users implements UserInterface, PasswordAuthenticatedUserInterface { /** * @ORM\Id(strategy="UUID") * @ORM\Column(type="string", length=36) */ private $id; /** * @ORM\Column(type="string", length=255, unique=true) */ private $email; /** * @ORM\Column(type="json") */ private $roles = []; }
```

Entité Users – Stratégie de génération d'ID défini sur UUID



```
public function up(Schema $schema): void { // this up() migration is auto-generated, please modify it to your needs $this->addSql($sql: 'CREATE TABLE authors (id INT AUTO_INCREMENT NOT NULL, name VARCHAR(255) NOT NULL, PRIMARY KEY(id)) DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci ENGINE=InnoDB'); $this->addSql($sql: 'CREATE TABLE books (id INT AUTO_INCREMENT NOT NULL, title VARCHAR(255) NOT NULL, description LONGTEXT NOT NULL, paruted_at DATETIME);'); $this->addSql($sql: 'CREATE TABLE books_categories (books_id INT NOT NULL, categories_id INT NOT NULL, INDEX IDX_16746f1570D8AC28 (books_id), INDEX IDX_877eACC270D8AC28 (categories_id));'); $this->addSql($sql: 'CREATE TABLE books_authors (books_id INT NOT NULL, authors_id INT NOT NULL, INDEX IDX_877eACC270D8AC28 (books_id), INDEX IDX_877eACC270D8AC28 (authors_id));'); $this->addSql($sql: 'CREATE TABLE books_reservations (id INT AUTO_INCREMENT NOT NULL, user_id CHAR(36) NOT NULL, books_id INT NOT NULL, reserved_at DATETIME);'); $this->addSql($sql: 'CREATE TABLE categories (id INT AUTO_INCREMENT NOT NULL, name VARCHAR(255) NOT NULL, PRIMARY KEY(id)) DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci ENGINE=InnoDB'); $this->addSql($sql: 'CREATE TABLE users (id CHAR(36) NOT NULL DEFAULT UUID(), email VARCHAR(255) NOT NULL, roles JSON NOT NULL, password VARCHAR(255) NOT NULL);'); $this->addSql($sql: 'ALTER TABLE books_categories ADD CONSTRAINT FK_16746f15A2121A87 FOREIGN KEY (books_id) REFERENCES books (id) ON DELETE CASCADE'); $this->addSql($sql: 'ALTER TABLE books_categories ADD CONSTRAINT FK_16746f15A2121A87 FOREIGN KEY (categories_id) REFERENCES categories (id) ON DELETE CASCADE'); $this->addSql($sql: 'ALTER TABLE books_authors ADD CONSTRAINT FK_877eACC270D8AC28 FOREIGN KEY (books_id) REFERENCES books (id) ON DELETE CASCADE'); $this->addSql($sql: 'ALTER TABLE books_authors ADD CONSTRAINT FK_877eACC26DE2013A FOREIGN KEY (authors_id) REFERENCES authors (id) ON DELETE CASCADE'); $this->addSql($sql: 'ALTER TABLE books_reservations ADD CONSTRAINT FK_D86E1169A76ED395 FOREIGN KEY (user_id) REFERENCES users (id) ON DELETE CASCADE'); $this->addSql($sql: 'ALTER TABLE books_reservations ADD CONSTRAINT FK_D86E11697DD8AC28 FOREIGN KEY (books_id) REFERENCES books (id) ON DELETE CASCADE'); }
```

Extrait fichier de migration généré par Doctrine.

## 2. Authentification & Inscription

Le Maker-Bundle offre un gain de temps considérable dans le développement de ces 2 fonctionnalités, il nous offre plusieurs commandes nous permettant de générer plusieurs fichiers que l'on va pouvoir personnaliser par la suite.

Cette fonctionnalité est rattachée à l'entité « Users » qui doit obligatoirement implémenter « UserInterface ».

### a. Authentification

Après avoir tapé la commande « **php bin/console make:auth** », un questionnaire se lance :

- Style d'authentication ? Choix « login form » car la connexion à l'application passera par un formulaire de connexion.
- Nom du service d'authentication ? « Authenticator »
- Nom du contrôleur : « SecurityController »
- Génération du logout : « yes »

A la fin du questionnaire, Symfony :

- génère le Template Twig de login,
- le contrôleur,
- met à jour le fichier « security.yaml » en créant un nouvel User Provider (configuration de l'entité à vérifier lors de l'authentification),
- créer dans le dossier « Security », le service d'authentication.

Dans le contrôleur j'ai modifié la route de la page à « / », cette page sera la page d'accueil.

J'ai fini en modifiant la vue pour la mettre en conformité avec la charte graphique et le wireframe avec « MaterializeCSS » et du CSS.

```
kevin@MacBook-Air-de-Kevin skeleton % php bin/console make:auth
What style of authentication do you want? [Empty authenticator]:
[0] Empty authenticator
[1] Login form authenticator
> 1

The class name of the authenticator to create (e.g. AppCustomAuthenticator):
> Authenticator

Choose a name for the controller class (e.g. SecurityController) [SecurityController]:
>

Do you want to generate a '/logout' URL? (yes/no) [yes]:
>

created: src/Security/Authenticator.php
updated: config/packages/security.yaml
created: src/Controller/SecurityController.php
created: templates/security/login.html.twig

Success!

Next:
- Customize your new authenticator.
- Finish the redirect "TODO" in the App\Security\Authenticator::onAuthenticationSuccess() method.
- Check the user's password in App\Security\Authenticator::checkCredentials().
- Review & adapt the login template: templates/security/login.html.twig.
kevin@MacBook-Air-de-Kevin skeleton %
```

Formulaire Maker création de l'authentification.

```
providers:
    app_user_provider:
        entity:
            class: App\Entity\Users
            property: email
```

Paramètres User Provider

## b. Inscription

Comme pour l'authentification, le Maker-bundle contient une commande permettant de générer tout le code minimum pour créer cette fonctionnalité en fonction d'un questionnaire.

La commande : « **php bin/console make:registration-form** »

La configuration établit lors du questionnaire est :

- Ajout d'un `@UniqueEntity` sur le champ « email » dans l'entité « Users ».
- Pas d'email de vérification.
- Redirection sur la page de login à la création du compte.

Symfony génère à la fin du questionnaire :

- Le Template de la page d'inscription,
- Le « `FormBuilder` »
- Le contrôleur « `RegistrationController` »

J'ai configuré le formulaire en précisant le type de champ (ex : « `TextType::class` »), les options propres à chacun d'eux (les labels, les attributs etc...).

J'ai rajouté aussi les contraintes telles que « `NotBlank` » ou « `Length` » afin de conserver un contrôle et une cohérence sur les données entrantes lors de la soumission du formulaire d'inscription.

Lors de son inscription, l'utilisateur tapera son mot de passe que l'application recevra et hashera avant de l'insérer en BDD.

Il pourra aussi voir son mot de passe en clair grâce à un fonction Javascript qui à chaque clique sur le bouton « œil » changera le type de champ du mot de passe.

```
>>>add( child: 'plainPassword', type: RepeatedType::class, [
  'type' => PasswordType::class,
  'mapped' => false,
  'required' => !$options['is_edit'],
  'invalid_message' => 'Les 2 mots de passe doivent être identiques',
  'first_options' => [
    'label' => 'Mot de passe',
    'attr' => ['autocomplete' => 'new-password'],
    'required' => !$options['is_edit'],
    'constraints' => [
      new Length([
        'min' => 6,
        'minMessage' => 'Votre mot de passe doit contenir au moins {{ limit }} caractères.',
        'max' => 255,
        'maxMessage' => 'Votre mot de passe doit contenir au maximum {{ limit }} caractères.',
      ]),
    ],
  ],
  'second_options' => [
    'label' => 'Confirmer votre mot de passe',
    'required' => !$options['is_edit'],
  ],
])
```

Extrait de la configuration du champ de mot de passe en utilisant un double champ identique en contrainte.

```
if ($form->isSubmitted() && $form->isValid()) {
  //Add constraint NotBlank in controller for protect HTML "required"
  $validator->validate($form->get('plainPassword')->getData(), new NotBlank(['message' => 'Vous devez remplir ce champ.']));
  $user->setPassword($userPasswordHasherInterface->hashPassword($user, $form->get('plainPassword')->getData()));

  $entityManager = $this->getDoctrine()->getManager();
  $entityManager->persist($user);
  $entityManager->flush();
  $this->addFlash(type: 'success', message: 'Inscription enregistrée avec succès, vous pourrez vous connecter quand votre compte sera validé par un administrateur');

  return $this->redirectToRoute('app_login');
}
```

Hachage du mot de passe lors de la soumission du formulaire.

```
const input1 = document.getElementById('registration_form_plainPassword_first');
const btnShow1 = document.getElementById('view1');
const btnHide1 = document.getElementById('hide1')

btnShow1.addEventListener( type: 'click', listener: () => {
  input1.type = 'text';
  btnShow1.style.display = 'none';
  btnHide1.style.display = 'block';
})

btnHide1.addEventListener( type: 'click', listener: () => {
  input1.type = 'password';
  btnShow1.style.display = 'block';
  btnHide1.style.display = 'none';
})
```

PassViewerIcon.js

### 3. Gestion des utilisateurs

Comme voulu par le cahier des charges, un employé pourra confirmer ou refuser un utilisateur en vérifiant les données qu'il a entrés lors de son inscription, cela se passe dans un contrôleur nommé « **UsersController** ».

Les différentes méthodes de cette fonctionnalité sont protégées par la méthode « **is\_granted** » obtenu grâce au bundle « **Security** ».

- **Methode panelAdmin – Route « /users/admin »**

Cette méthode retourne la page du panel de gestion des utilisateurs qui n'ont toujours pas été validés par un employé, elle contient la liste des utilisateurs ainsi que leurs coordonnées obtenu grâce à la requête Doctrine « **findBy** ».

```
// Return user no validate by employee
#[Route('/admin', name: 'users_index')]
#[IsGranted('ROLE_EMPLOYEE', message: 'Vous n\'êtes pas autorisé à accéder à cette page')]
public function panelAdmin (UsersRepository $usersRepository) : Response
{
    return $this->render( view: 'users/admin-index', [
        'users' => $usersRepository->findBy(['isValidate' => false])
    ]);
}
```

Méthode panelAdmin

- **Méthode accept – Route « /users/accept/{id} »**

La fonctionnalité n'est disponible qu'en méthode POST, et est protégée par un jeton CSRF.

Lorsque l'employé accepte l'utilisateur la méthode change le statut du booléen « **isValidate** » et le place à true. Grâce à ça, l'utilisateur obtiendra le « **ROLE\_USER** » et pourra ainsi accéder au site.

```
//Accept user and change validate bool at true
#[Route('/accept/{id}', name: 'user_accept', methods: ['POST'])]
#[IsGranted('ROLE_EMPLOYEE', message: 'Vous n\'êtes pas autorisé à effectué cette action')]
public function accept (Request $request, Users $user) : Response
{
    if ($this->isCsrfTokenValid( id: 'accept' . $user->getId(), $request->request->get( key: '_token'))) {
        $user->setIsValidate( isValidate: true);

        $entityManager = $this->getDoctrine()->getManager();
        $entityManager->flush();
        $this->addFlash( type: 'success', message: 'Action bien prise en compte.');
    }
    return $this->redirectToRoute( route: '/admin');
}
```

Méthode accept

- **Méthode delete – Route « /users/remove/{id} »**

La fonctionnalité n'est disponible qu'en méthode POST, et est protégée par un jeton CSRF.

Si l'employé refuse le nouvel utilisateur, il n'a qu'à cliquer sur le bouton « **Refuser** » et confirmer son choix puis l'application vérifiera le token CSRF et effacera l'utilisateur.

```
public function getRoles(): array
{
    $roles = $this->roles;
    if($this->isValidate) {
        $roles[] = 'ROLE_USER';
    }

    return array_unique($roles);
}
```

Si l'utilisateur est validé, le « **ROLE\_USER** » est ajouté dans le tableau de rôles

- Méthode `userUpdate` – Route « /users/ »

Une fois l'utilisateur validé par un employé, il a accès au catalogue et aux diverses fonctionnalités définies dans le cahier des charges dont la modification de son profil.

La méthode fonctionne en méthode GET et POST.

Grâce à la méthode « `getUser` » de l'« **AbstractController** », j'ai obtenu les infos de l'utilisateur sans avoir à l'afficher dans le lien.

#### Méthode GET :

- Renvoie le formulaire pré-rempli contenant ses coordonnées, seulement l'adresse et le mot de passe sont modifiable, le reste des champs sont désactivés grâce au tableau « `is_edit` » passé dans la méthode « `createForm` » et la valeur y est placée sur les différents champs à désactivés.
- Un champ de confirmation de l'ancien mot de passe est créé grâce à un **EventListener** qui se charge de vérifier avant le chargement du formulaire si l'utilisateur possède une adresse email, dans ce cas le champ sera chargé.
- Coté vue, une balise `fieldset` et une légende sont ajoutées contenant les 2 champs de mot de passe, ainsi que la modal contenant le champ de vérification de l'ancien mot de passe pour confirmer les changements.

#### Méthode POST :

- Traitement du formulaire, le champ mot de passe est vérifié s'il est rempli, et dans le cas où il est, il est hasher.

L'utilisateur pourra aussi voir les deux champs de mot de passe en clair grâce à la fonction JS vu précédemment.

```
// Get user details and form for update user details.  
#[Route('/', name: 'user_update', methods: ['GET', 'POST'])]  
public function userUpdate (Request $request, UsersRepository $usersRepository, UserPasswordHasherInterface $hasher) : Response  
{  
    $user = $usersRepository->find($this->getUser());  
    $form = $this->createForm( type: RegistrationFormType::class, $user, ['is_edit' => true]);  
    $form->handleRequest($request);  
  
    if ($form->isSubmitted() && $form->isValid()) {  
        if (!$form->get('plainPassword')->isEmpty()) {  
            $user->setPassword($hasher->hashPassword($user, $form->get('plainPassword')->getData()));  
        }  
  
        $entityManager = $this->getDoctrine()->getManager();  
        $entityManager->flush();  
        $this->addFlash( type: 'success', message: 'Votre profil a été mis à jour avec succès.');  
    }  
  
    return $this->render( view: 'users/user-index',[  
        'form' => $form->createView(),  
    ]);  
}
```

Méthode `userUpdate`

```
->addEventListener( eventName: FormEvents::PRE_SET_DATA, function (FormEvent $events){  
    $form = $events->getForm();  
    $user = $events->getData();  
  
    if (!empty($user->getEmail())) {  
        $form->add( child: 'oldPassword', type: PasswordType::class, [  
            'mapped' => false,  
            'label' => 'Mot de passe',  
            'error_bubbling' => true,  
            'constraints' => [  
                new UserPassword([  
                    'message' => 'Mot de passe invalide.'  
                ])  
            ]  
        ]);  
    }  
});
```

EventListener au pré-chargement du formulaire d'enregistrement

## 4. Le catalogue & sa gestion

Le catalogue référence l'entité « **Books** » qui a également deux relations non nulles « **Authors** » et « **Categories** », j'ai donc également mise en place les 2 options permettant de les créer et de les supprimer lors de la création d'une entité « **Books** ».

### a. Le Contrôleur « BooksController » :

- Méthode new - Route « /books/new »

Page accessible uniquement aux employés et +, elle est protégée par la méthode « **is\_granted** ».

- Méthode **GET** : Retourne le formulaire de création d'entité et intègre les formulaires de création d'entités « **Authors** » et « **Categories** » avec les actions pointant sur leurs contrôleurs respectifs pour le traitement.

- Méthode **POST** : Si le formulaire est valide, l'image de couverture est renommée (pour éviter tout conflit de nom) puis déplacé dans le dossier « /public/upload » par le service « **ImgUploader** » injecté dans la fonction.

```
#[Route('/new', name: 'books_new', methods: ['GET', 'POST'])]
#[IsGranted('ROLE_EMPLOYEE', message: 'Vous n\'êtes pas autorisé à accéder à cette page')]
public function new(Request $request, ImgUploader $uploader, CategoriesRepository $categoriesRepository, AuthorsRepository $authorsRepository): Response
{
    $book = new Books();
    $form = $this->createForm(BooksType::class, $book);
    $newAuthorForm = $this->createForm(AuthorsType::class, [], [
        'action' => $this->generateUrl('authors_new')
    ]);
    $newCategoryForm = $this->createForm(CategoriesType::class, [], [
        'action' => $this->generateUrl('categories_new')
    ]);
    $form->handleRequest($request);

    if ($form->isSubmitted() && $form->isValid()) {
        // Give new name if an image has been uploaded //
        $file = $form['cover']->getData();
        if($file instanceof UploadedFile){
            $filename = $uploader->getFileName($file);
            $book->setCover('uploads/' . $filename);
        }

        $entityManager = $this->getDoctrine()->getManager();
        $entityManager->persist($book);
        $entityManager->flush();

        $this->addFlash('success', 'Livre ajouté au catalogue avec succès');
        return $this->redirectToRoute('books_index', [], Response::HTTP_SEE_OTHER);
    }

    return $this->render('books/new.html.twig', [
        'authors' => $authorsRepository->findAll(),
        'categories' => $categoriesRepository->findAll(),
        'form' => $form->createView(),
    ]);
}
```

Méthode new

```
class ImgUploader
{
    private $targetDirectory;

    public function __construct($targetDirectory)
    {
        $this->targetDirectory = $targetDirectory;
    }

    public function getTargetDirectory()
    {
        return $this->targetDirectory;
    }

    #Create a new filename and move the file in public/image
    public function getFileName (UploadedFile $file): string
    {
        $newFileName = 'book_'.uniqid().'.'.$file->guessExtension();
        $file->move($this->getTargetDirectory(), $newFileName);
        return $newFileName;
    }
}
```

Service « **ImgUploader** » qui renommera par un uniqid l'image de couverture

La valeur de la propriété « **\$targetDirectory** » est passé par injection d'argument configuré dans le fichier de configuration des services situé dans « config/services.yaml » (généré lors de l'initialisation du projet).

```
parameters:
    upload_directory: '%kernel.project_dir%/public/uploads'
Paramètre contenant le chemin du dossier d'upload
```

```
App\Services\ImgUploader:
    arguments:
        $targetDirectory: '%upload_directory%' targetDirectory
```

Injection du paramètre « **upload\_directory** » à la propriété **\$targetDirectory**

- Méthode index – Route « books/ »

Fonction du catalogue de livre.

- Méthode GET :

Retourne tous les livres avec affichage de ceux disponible à l'emprunt, les pages sont paginés grâce à la méthode que me fournit le bundle « KnpPaginator ».

Je n'ai qu'a injecté dans la fonction le « PaginatorInterface » puis utilisé la méthode « paginate » en lui donnant :

- Un objet de type « **Query** » qui est la requête Doctrine,
- La requête GET avec le numéro de la page et celle par défaut est la page 1,
- La limite de livre par page qui sera de 5.

Elle retourne aussi le formulaire de filtrage par genre.

- Méthode POST :

L'utilisateur souhaitera faire un filtrage par genres de livres.

Pour ce faire :

- je crée une classe « **FiltersBooks** » contenant la propriété de type « **Categories** », ainsi qu'un formulaire qui me retournera les catégories de livres dans un sélecteur.
- je crée une méthode dans « **BooksRepository** » contenant une requête par étape qui effectue une jointure sur la table/entité « **Categories** » et renvoi les livre par ID dans l'ordre croissant.

```
class FiltersBooks
{
    public Categories $category;

    /**
     * @return Categories
     */
    public function getCategory(): Categories
    {
        return $this->category;
    }
}
```

Classe FiltersBooks située « App/src/Data »

```
// Get the catalogue or search filter page
#[Route('/', name: 'books_index', methods: ['GET', 'POST'])]
public function index(BooksRepository $booksRepository, Request $request, PaginatorInterface $paginator):
Response
{
    $filterBooks = new FiltersBooks();
    $filterForm = $this->createForm( type: FiltersType::class, $filterBooks);
    $filterForm->handleRequest($request);

    if($filterForm->isSubmitted() && $filterForm->isValid()) {
        return $this->render( view: 'books/index.html.twig', [
            'books' => $booksRepository->getBookByCategory($filterBooks)->getResult(),
            'filterForm' => $filterForm->createView()
        ]);
    }

    return $this->render( view: 'books/index.html.twig',
        [
            'books' => $paginator->paginate($booksRepository->getBooksByIsFree(), $request->query->getInt(key: 'page', default: 1), limit: 5),
            'filterForm' => $filterForm->createView()
        ]
    );
}
```

Méthode index

```
public function getBookByCategory(FiltersBooks $filtersBooks) : Query
{
    $query = $this->createQueryBuilder(alias: 'b')
        ->select(select: 'c', 'b')
        ->join(join: 'b.categories', alias: 'c');

    if(!empty($filtersBooks->getCategory())){
        $query = $query
            ->andWhere('c.id IN (:categories)')
            ->setParameter(key: 'categories', $filtersBooks->category)
            ->orderBy(sort: 'b.id', order: 'DESC');
    }
    return $query->getQuery();
}
```

Requête par étape « BooksRepository » - jointure sur table « Categories »

- Méthode show – Route « books/{id} »

Méthode permettant l'affichage des détails d'un livre.

- Grâce à l'ID récupéré dans le lien, la méthode récupérera l'entité dans la base de données.

```
#[Route('/{id}', name: 'books_show', methods: ['GET'])]
public function show(Books $book): Response
{
    return $this->render( view: 'books/show.html.twig', [
        'book' => $book,
    ]);
}
```

Méthode show

- Méthode delete – Route « /books/remove/{id} »

Fonction d'effacement du livre.

- Disponible uniquement avec la méthode « POST ».
- Protégé par la méthode « **is\_granted** » et un token CSRF.
- Vérification de la disponibilité du livre à l'emprunt, retourne un message d'erreur si le livre est emprunté et ne supprimera pas l'entité.

Pour accéder à cette fonctionnalité, vous aurez besoin d'être sur la page de détail d'un livre.

```
#[Route('/remove/{id}', name: 'books_delete', methods: ['POST'])]
public function delete(Request $request, Books $book, BooksReservationsRepository $reservationsRepository): Response
{
    $reservations = $reservationsRepository->findOneBy(['books' => $book->getId()]);

    if($reservations) {
        $this->addFlash( type: 'errors', message: 'Un emprunt est en cours pour ce livre');
    } elseif ($this->isCsrfTokenValid( id: 'delete' . $book->getId(), $request->request->get( key: '_token'))) {
        $entityManager = $this->getDoctrine()->getManager();
        $entityManager->remove($book);
        $entityManager->flush();

        $this->addFlash( type: 'success', message: 'Livre effacé du catalogue avec succès');
    }

    return $this->redirectToRoute( route: '/', [], status: Response::HTTP_SEE_OTHER);
}
```

Méthode delete

## b. Les formulaires « BooksType », « CategoriesType » et « AuthorsBooks »

Les formulaires sont concentrés dans la page de création de l'entité « **Books** » donc uniquement accessible aux utilisateurs de grade employé minimum.

Ils contiennent tous les champs de l'entité concerné à remplir, exception de l'ID généré automatiquement lors de la création, les types de champs et leurs options (contraintes, labels, select multiple ou non etc...)

Les formulaires d'ajout/suppression de « **Categories** » et « **Authors** » sont contenus dans une modal, pour y accéder une icône est disponible à côté du choix de la « **Categories** » ou de l'« **Authors** » du livre

```
>>>add( child: 'categories', type: EntityType::class, [
    'label' => 'Type(s) de livre',
    'class' => Categories::class,
    'choice_label' => 'name',
    'multiple' => true,
    'attr' => ['class' => 'multiple'],
    'constraints' => [
        new NotBlank([
            'message' => 'Vous devez remplir ce champ.'
        ])
    ]
])
>>>add( child: 'authors', type: EntityType::class, [
    'label' => 'Auteur(s)',
    'class' => Authors::class,
    'choice_label' => 'name',
    'multiple' => true,
    'attr' => ['class' => 'multiple'],
    'constraints' => [
        new NotBlank([
            'message' => 'Vous devez remplir ce champ.'
        ])
    ]
]);

```

Configuration du champ « categories » et « authors »

## Vue champ « Categories » & « Authors »

Auteur(s)



Type(s) de livre



## 5. Les emprunts & leurs gestions

Le contrôleur « **BooksReservations** » contient toutes les méthodes permettant de créer et de gérer les emprunts contenus dans l'entité homonyme que ce soit du côté administrateur, ou coté utilisateurs.

a. Le contrôleur « BooksReservations » :

- **Méthode getUsersReservations – Route « /reservations/ »**

Gère l'affichage des livres emprunté/réservation coté utilisateurs en récupérant l'utilisateur connecté grâce à la méthode « getUser ». Pour l'affichage des livres empruntés depuis + de 3 semaines, j'ai créé une fonction dans un service vérifiant chaque emprunt et renvoyant un tableau des emprunts dont la date de collecte + 3 semaines est inférieure à la date du jour (la date de réservation n'est pas utilisée, ni rendu dans la vue).

```
#Route('/', name: 'user_books_reservations', methods: ['GET'])
public function getUserReservation (BooksReservationsRepository $reservationsRepository, OutdatedReservations $outdatedReservations) : Response
{
    $reservations = $reservationsRepository->findBy(['user' => $this->getUser(), ['reservedAt' => 'ASC']]);
    $outdatedReservations = $outdatedReservations->getOutdatedReservation($reservations);

    return $this->render( view: 'books_reservations/user-index', [
        'reservations' => $reservations,
        'outdatedReservations' => $outdatedReservations
    ]);
}
```

Méthode « getUsersReservations »

```
public function getOutdatedReservation (array $reservations) : array
{
    $outdatedReservations = [];

    foreach ($reservations as $reservation) {
        if($reservation->getIsCollected()) {
            $reservedAt = new \DateTime($reservation->getCollectedAt()->format('Y-m-d H:m:s'));
            $date_now = new \DateTime();
            if($reservation->getIsCollected() && ($reservedAt->add(new \DateInterval( duration: 'P3W')) < $date_now) ) {
                $outdatedReservations[] = $reservation;
            }
        }
    }
    return $outdatedReservations;
}
```

Fonction « GetOutdatedReservation » dans le service « OutdatedReservations »

- **Méthode new – Route « /reservations/new/{id} »**

Création d'un nouvel emprunt, utilisation du bouton situé dans la page de détail d'un livre, le bouton envoi l'ID du livre voulant être emprunté dans le lien.

La fonctionnalité est uniquement disponible en POST (donc utilisation d'un formulaire) et est protégée contre les failles CSRF grâce à un jeton.

Avant la création d'un emprunt, la méthode vérifiera si le livre est disponible et si l'utilisateur à moins de 10 livres empruntés, le cas échéant elle renverra une erreur à l'utilisateur, sinon le livre est marqué indisponible et l'emprunt est créé.

```
#Route('/new/{id}', name: 'books_reservations_new', methods: ['POST'])
public function newBooks ($book, UsersRepository $usersRepository, Request $request): Response
{
    if ($this->isCsrfTokenValid( id: 'create' . $book->getId(), $request->request->get( key: '_token') )) {

        $user = $usersRepository->find($this->getUser());

        if (! $book->getIsFree()) {
            $this->addFlash( type: 'errors', message: 'Le livre n\'est pas disponible');
        }

        if ($user->getBooksReservations()->count() > 10) {
            $this->addFlash( type: 'errors', message: 'Vous avez déjà atteint la limite de livre emprunté');
        }

        if ($book->getIsFree() && $user->getBooksReservations()->count() < 10) {
            $booksReservation = new BooksReservations();
            $booksReservation->setBooks($book)
                ->setUser($user)
                ->setReservedAt(new \DateTime( datetime: 'now', new \DateTimeZone( timezone: 'Europe/Paris')));
            $book->setIsFree( isFree: false);

            $entityManager = $this->getDoctrine()->getManager();
            $entityManager->persist($booksReservation);
            $entityManager->flush();
            $this->addFlash( type: 'success', message: 'Vous pouvez venir chercher votre livre dès maintenant à la médiathèque.');
        }
    }

    return $this->redirectToRoute( route: '/{id}', [ 'id' => $book->getId()]);
}
```

Méthode new

- Méthode cancelReservation – « /reservations/cancel-reservation/{id} »**

Annulation d'un emprunt via le bouton (disponible uniquement si l'utilisateur n'a pas récolté son livre) situé dans la page de gestion des emprunts (« /reservations »). La méthode, uniquement disponible en POST, vérifiera le token CSRF, s'assurera que l'emprunt appartient bien à l'utilisateur et que le livre n'est pas collecté avant de supprimer l'emprunt et de repasser le livre à disponible, sinon elle renverra une exception. L'ID de la réservation est passé dans le lien.

```
#[Route('/cancel-reservation/{id}', name: 'books_reservations_cancel', methods: ['POST'])]
public function cancelReservation(Request $request, BooksReservations $booksReservations, BooksReservationsRepository $reservationsRepository, BooksRepository {
    if ($this->isCsrfTokenValid( id: 'cancelReservation', $booksReservations->getId(), $request->request->get( key: '_token'))) {
        $book = $booksRepository->find($booksReservations->getBooks()->getId());

        if (($this->getUser() === $booksReservations->getUser()) && !$booksReservations->getIsCollected()) {
            $book->setIsFree( isFree: true);

            $entityManager = $this->getDoctrine()->getManager();
            $entityManager->remove($booksReservations);
            $entityManager->flush();

            $this->addFlash( type: 'success', message: 'Votre emprunt a bien été annulé');
        } else {
            throw new AccessDeniedException( message: "Vous n'avez pas le droit d'annuler cette emprunt");
        }
    }

    return $this->redirectToRoute( route: 'user_books_reservations', [
        'reservations' => $reservationsRepository->findBy([ 'user' => $this->getUser()])
    ]);
}
```

Méthode cancelReservation

Mes Emprunts			
Livre	Date de réservation	Date d'emprunt maximum	
Dieu - La science Les preuves	30/10/2021 14:10	20/11/2021 14:11	<button>ANNULER L'EMPRUNT</button>
The Witcher Sorceleur - : Le Sorceleur : La carte de l'univers	30/10/2021 15:10	-	<button>ANNULER L'EMPRUNT</button>
Louca - Tome 9 : Louca - Game Over	30/10/2021 15:10	-	<button>ANNULER L'EMPRUNT</button>

Vue page d'emprunt utilisateur avec bouton d'annulation

- Méthode adminPanel – Route « /reservations/admin »**

La page est disponible uniquement aux employés et +, et est protégée par la méthode « `is_granted` ». Cette méthode renvoi la liste des réservations et des emprunts en cours, elle effectue aussi le travail de suppressions des réservations de plus de 3 jours via la fonction « `getOutdatedReservationAdminPanel` » dans le service « `OutdatedReservations` » qui retourne dans un tableau les réservations non collectées dont les dates de réservation + 3 jours est supérieur à la date d'aujourd'hui. Pour finir, la méthode `adminPanel` bouclera sur chaque réservation retournée par le service, supprimera les emprunts et remettra le livre disponible dans le catalogue.

```
#[Route('/admin', name: 'books_reservations_index', methods: ['GET'])]
#[isGranted('ROLE_EMPLOYEE', message: 'Vous n\'êtes pas autorisé à accéder à cette page')]
public function adminPanel(BooksReservationsRepository $booksReservationsRepository, OutdatedReservations $outdatedReservations): Response {
    // Check the date and delete the reservation not recollected since 3 days
    $reservations = $booksReservationsRepository->findAll();

    $outdatedReservation = $outdatedReservations->getOutdatedReservationAdminPanel($reservations);
    $entityManager = $this->getDoctrine()->getManager();

    foreach($outdatedReservation as $reservation) {
        $reservation->setBooks()->setIsFree(true);
        $entityManager->remove($reservation);
        $entityManager->flush();
    }

    return $this->render( view: 'books_reservations/admin-index.html.twig', [
        'reservations' => $booksReservationsRepository->getReservationByDate(),
    ]);
}
```

Méthode adminPanel

```
public function getOutdatedReservationAdminPanel (array $reservations) : array
{
    $outdatedReservations = [];
    foreach($reservations as $reservation) {
        $reservedAt = new \DateTime($reservation->getReservedAt()->format('Y-m-d H:m:s'));
        $dateNow = new \DateTime();
        if(!$reservation->getIsCollected() && ($reservedAt->add(new \DateInterval( duration: 'P3D')) < $dateNow)){
            $outdatedReservations[] = $reservation;
        }
    }
    return $outdatedReservations;
}
```

Fonction « `getOutdatedReservationAdminPanel` dans le service « `getOutdatedReservation` »

- Méthode `changeCollectStatut` –

### **Route « /reservations/customer-collect/{id} »**

Fonctionnalité disponible en méthode POST pour les employés et +, protégée par la méthode « `is_granted` ». L'ID de la réservation est passé par le lien.

Après vérification du token CSRF, la méthode alternera selon le statut de collecte du livre « `isCollected` » entre :

- True : créer la date de collecte avec le bon fuseau horaire.
- False : remettra la date de collecte à nulle.

Renvoi un message à l'utilisateur si l'opération s'est bien déroulée.

```
##[Route('customer-collect/{id}', name: 'books-update-collect', methods: ['POST'])]
#[IsGranted('ROLE_EMPLOYEE', message: 'Vous n\'êtes pas autorisé à effectué cette action')]
public function changeCollectStatus (BooksReservations $booksReservations, Request $request) : Response
{
    if ($this->isCsrfTokenValid( id: 'updateCollect' . $booksReservations->getId(), $request->request->get( key: '_token'))) {
        if($booksReservations->getIsCollected()) {
            $booksReservations->setIsCollected( isCollected: false);
            $booksReservations->setCollectedAt( collectedAt: null);
        } else {
            $booksReservations->setIsCollected( isCollected: true);
            $booksReservations->setCollectedAt(new \DateTime( datetime: 'now', new \DateTimeZone( timezone: 'Europe/Paris')));
        }

        $entityManager = $this->getDoctrine()->getManager();
        $entityManager->flush();

        $this->addFlash( type: 'success', message: 'Mise à jour du statut effectué avec succès');
    }

    return $this->redirectToRoute( route: 'books_reservations_index');
}
```

Méthode `changeCollectStatus`

- Méthode `delete` –

### **Route « /reservations/customer-collect/{id} »**

Lorsqu'un utilisateur vient rendre son emprunt à la médiathèque, un employé réceptionnera le livre et clôturera l'emprunt en le supprimant.

Méthode uniquement disponible en POST, pour les employés et + grâce à la méthode « `is_granted` ».

Après vérification du jeton CSRF, la méthode s'occupera de supprimer l'emprunt en base de données et rendra le livre disponible sur le catalogue.

```
##[Route('remove/{id}', name: 'books_reservations_delete', methods: ['POST'])]
#[IsGranted('ROLE_EMPLOYEE', message: 'Vous n\'êtes pas autorisé à effectué cette action')]
public function delete(Request $request, BooksReservations $booksReservation, BooksRepository $booksRepository): Response
{
    if ($this->isCsrfTokenValid( id: 'delete' . $booksReservation->getId(), $request->request->get( key: '_token'))) {
        $book = $booksRepository->find($booksReservation->getBooks()->getId());

        $book->setIsFree( isFree: true);

        $entityManager = $this->getDoctrine()->getManager();
        $entityManager->remove($booksReservation);
        $entityManager->flush();

        $this->addFlash( type: 'success', message: 'Emprunt clôturé avec succès');
    }

    return $this->redirectToRoute( route: 'books_reservations_index', [], status: Response::HTTP_SEE_OTHER);
}
```

Méthode `delete`

## 6. La barre de recherche dynamique

Cette fonctionnalité permettra aux utilisateurs de rechercher un livre du catalogue par son titre par la suggestion de titres du catalogue contenant en son sein le mot recherché et cela de façon asynchrone en rajoutant dynamiquement au DOM un volet déroulant grâce à JS.

### a. Mise en place de l'API

Avec le bundle API-Platform, j'ai généré une API REST avec l'entité « **Books** » ainsi que ses relations (« **Categories** », « **Authors** ») avec la méthode « **ApiRessource** ».

J'ai rajouté dans les paramètres de l'annotation la protection des routes POST/DELETE/PUT/PATCH en les fermant et le format de retour en JSON.

```
namespace App\Entity;

use ...;

/**
 * @ApiResource(collectionOperations={"GET"}, itemOperations={"GET"}, formats={"json"})
 * @ORM\Entity(repositoryClass=BooksRepository::class)
 * @UniqueEntity(fields={"title"}, message="Un livre avec ce titre existe déjà.")
 */
class Books
```

Entité « Books » convertie en API

### b. La fonction JS asynchrone « search »

Un écouteur d'événement placé sur la barre de recherche permet d'exécuter la fonction à chaque appui sur le clavier (événement « **keyup** »).

La fonction contient la requête **fetch** permettant de récupérer le catalogue de livre, de filtrer les livres renvoyé par la requête contenant le mot de recherché.

Elle gère aussi l'affichage du volet déroulant, de la fonction à appeler selon l'état et le contenu de la réponse et de l'animation de chargement de la requête.

```
const search = async () => {
    let searchWord = input.value;
    await cleanDropdown();
    containerDropdown.style.display = 'flex';
    loadingIcon.style.display = 'inline-block';

    if (searchWord.length > 0 && searchWord.length <= 2) {
        displayHelpText( content: 'Continuer à taper ...');
    } else if (searchWord.length === 0) {
        containerDropdown.style.display = 'none';
        loadingIcon.style.display = 'none';
    }

    if(searchWord.length > 2){
        let response = await fetch( input: 'https://mediatheque-chapelle-cureaux.herokuapp.com/api/books');
        let data = await response.json();
        const searchResult = await data.filter(book => book.title.includes(searchWord));
        loadingIcon.style.display = 'none';

        if((response.ok && response.status === 200) || response.status === 304) {
            searchResult.length !== 0 ? searchResult.forEach(result => displayResults(result)) : displayHelpText( content: 'Aucun titre ne ressemble à votre recherche.');
        } else {
            displayHelpText( content: 'Une erreur est survenue.');
        }
    }
}
```

Fonction search

```
input.addEventListener( type: 'keyup', search);

const displayResults = (content) => {
    const li = document.createElement( tagName: 'li');
    li.insertAdjacentHTML( where: 'afterbegin', html: '<a href="https://mediatheque-chapelle-cureaux.herokuapp.com/books/${content.id}"> ${content.title} </a>');
    dropdown.append(li);
}

const displayHelpText = (content) => {
    const li = document.createElement( tagName: 'li');
    li.textContent = content;
    dropdown.append(li);
}
```

Les fonctions de création d'élément dans le DOM

## 7. Les Vues Twig

Le Framework **Materialize** est un Framework CSS utilisant les règles du Materiel Design créé par Google.

Il me permet de créer le layout de mes pages facilement et rapidement en enfermant mes différents composants de page dans un « container » ou une « row », il dispose aussi d'une bonne palette de composants utilisant JS dont les formulaires multiples, les menus déroulants et la modal que j'utilise.

- **Page d'accueil : Page de connexion**

Page du formulaire de connexion contenant le champ caché de protection CSRF.

```
<form class="center z-depth-3" method="POST">
    <h1>Connexion</h1>
    <div class="input-field">
        <input id="email" value="{{ last_username }}" name="email" required autofocus type="email" />
        <label for="email">Email</label>
    </div>
    <div class="input-field">
        <input id="password" type="password" name="password" required />
        <label for="password">Mot de passe</label>
    </div>
    <input type="hidden" name="_csrf_token" value="{{ csrf_token('authenticate') }}>
    <button class="btn waves-effect waves-light" type="submit" name="action">Connexion
        <i class="material-icons right">send</i>
    </button>
    {% if error %}
        <span class="error-msg">{{ error.messageKey|trans(error.messageData, 'security') }}</span>
    {% endif %}
</form>
```

Formulaire de connexion

- **Page d'inscription**

Inclus le formulaire d'inscription.

```
{{ form_start(form, {'attr': {'class': 'z-depth-3'}}) }}
<div class="civility-container">
    <div class="input-field">
        {{ form_label(form.firstname) }}
        {{ form_widget(form.firstname) }}
        {{ form_errors(form.firstname) }}
    </div>
    <div class="input-field">
        {{ form_label(form.lastname) }}
        {{ form_widget(form.lastname) }}
        {{ form_errors(form.lastname) }}
    </div>
</div>
```

Champs nom et prénom du formulaire d'inscription

- **Page de gestion du profil côté utilisateur**

Inclus le formulaire d'inscription prérempli, rajoute aussi la modal du champ de vérification de l'ancien mot de passe.

```
{% if app.user is not null %}
<fieldset>
    <legend>Changez votre mot de passe</legend>
    {% endif %}

    <div class="input-field">
        {{ form_label(form.plainPassword.first) }}
        {{ form_widget(form.plainPassword.first) }}
        <i class="material-icons icon-pass-view" id="view1">visibility</i>
        <i class="material-icons icon-pass-hide" id="hide1">visibility_off</i>
        {{ form_errors(form.plainPassword.first) }}
    </div>
    <div class="input-field">
        {{ form_label(form.plainPassword.second) }}
        {{ form_widget(form.plainPassword.second) }}
        <i class="material-icons icon-pass-view" id="view2">visibility</i>
        <i class="material-icons icon-pass-hide" id="hide2">visibility_off</i>
        {{ form_errors(form.plainPassword.second) }}
    </div>

    {% if app.user is not null %}
    </fieldset>
    {% endif %}

    {% if app.user is null %}
        <button type="submit" class="btn waves-effect waves-light"> Enregistrer <i class="material-icons right">send</i></button>
    {% else %}
        <div class="center btn-container">
            <a class="waves-effect waves-light btn modal-trigger" href="#oldPasswordModal"> Valider mes changements <i class="material-icons right">edit</i></a>
        </div>
        {% include('users/_oldPasswordForm-modal') %}
    {% endif %}
```

Si un utilisateur est connecté, ajoute le fieldset est la modal.

- **Page de gestion des utilisateurs coté administration**

Affiche la liste des utilisateurs non validé, ainsi que leurs différents boutons « Accepter » et « Refuser ».

```
{% for user in users %}
    <div class="container responsive-card z-depth-2">
        <div class="row">
            <p id="status"><u><b>Nom:</b></u> {{ user.firstname|upper }} </p>
            <p><u><b>Prénom:</b></u> {{ user.lastname }} </p>
            <p><u><b>Adresse email:</b></u> <a href="mailto:{{ user.email }}"> {{ user.email }} </a></p>
            <p><u><b>Date de naissance:</b></u> {{ user.birthdate|date('d/m/Y') }}</p>
            <p><u><b>Adresse:</b></u> {{ user.address }} </p>
        </div>
        <div class="btn-container">
            {% include('users/_accept-registration') %}
            {% include('users/_reject-registration') %}
        </div>
    </div>
```

Liste des utilisateurs non validé

## • Page du catalogue

Inclus les champs de filtrage, et gères l'affichage de tous les livres.

```

<% if (is_granted("ROLE_EMPLOYEE")) %>
  {% include('headers/header-employee') %}
<% else %>
  {% include('headers/header-user') %}
<% endif %>
<main>
  {% include('../app-flashes') %}
  {% include('books/_filters-book.html.twig') %}

  {% for book in books %}
    <div class="container z-depth-2">
      {% include('books/_btn-dispo-book') %}
      <div class="row">
        <div class="col l3 s4 book-img-container">
          
        </div>
        <div class="book-text-container col l9 s8">
          <h1>{{ book.title }}</h1>
          <p>{{ book.description }}</p>
          <p> Genre:<br>
            {% for category in book.categories %}
              {{ category.name }}<br>
            {% endfor %}
          </p>
        </div>
      </div>
    </div>
  {% endfor %}
</main>

```

Gestion d'affichage du header + Liste catalogue

## • Page de détail du livre

Affiche le livre demandé par l'utilisateur, affiche aussi le bouton « Emprunter », et pour l'employé, il affiche aussi le bouton « Supprimer » aussi.

```

<div class="btn-container">
  {% if (is_granted("ROLE_EMPLOYEE")) %>
    {% include('books/_delete_form') %}
  {% endif %}
  <a class="waves-effect waves-light btn modal-trigger btn-small {{ book.isFree ? '' : 'disabled' }}" href="#emprunte-modal">
    <i>Emprunter</i>
    <span>{{ book.title }}</span>
  </a>
</div>
</main>
{% include('../_footer') %}

<div id="emprunte-modal" class="modal">
  <div class="modal-content">
    <h4>Confirmation d'emprunt</h4>
    <p>NOTE: Vous vous engagez à récupérer le livre dans un délai de 3 jours sous peine d'annulation de votre emprunt.</p>
  </div>
  <div class="modal-footer">
    <a href="#" class="btn btn-small waves-effect waves-light modal-close">Annuler</a>
    <span>{{ book.title }}</span>
  </div>
</div>

```

Gestion de l'affichage des boutons + modal confirmation d'emprunt

## • Page de gestion des emprunt côté administration

Affiche tous les emprunts et les réservations.

Grâce à la fonction JS, met en évidence les emprunts dont les délais sont dépassés en vérifiant tous les textes de statut de réservations et ajoute une classe CSS sur la ligne.

```

<tbody>
  {% for reservation in reservations %}
    <tr>
      {% include('books_reservations/_status-text') %}
      <td style="...> <a href="{{ path('/id', { 'id': reservation.books.id }) }}" target="_blank">{{ reservation.books.title }}</a> </td>
      <td style="...> <a href="mailto:{{ reservation.user_email }}">{{ reservation.user.lastname }} {{ reservation.user.firstname|upper }}</a></td>
      <td style="...>{{ reservation.reservedAt|date('d/m/Y H:m') }}</td>
      <td style="...>{{ reservation.isCollected ? reservation.collectedAt|date_modify('+3 weeks')|date('d/m/Y H:m') : ' - ' }}</td>
      <td class="actions">
        {% include('books_reservations/_delete_form') %}
        {% include('books_reservations/_update-book-status') %}
      </td>
    </tr>
    {% else %}
      <td colspan="6">Aucun emprunt n'est en cours</td>
    {% endif %}
  {% endfor %}
</tbody>

```

Liste des emprunts

```

const statuCheck = () => {
  const status = document.querySelectorAll( selectors: '#status' );
  for(let i = 0; i < status.length; i++){
    if(status[i].textContent.includes('Dépassement')){
      status[i].parentNode.classList.add('alert-reservation')
    }
  }
}

```

Fonction JS vérifiant la présence du texte « Dépassement »

## • Page de gestion des emprunts côté utilisateur :

Affiche les emprunts de l'utilisateur, ainsi qu'une alerte contenant le nom des livres en retard, le bouton d'annulation d'emprunt est désactivé si le livre est récolté.

```

<% if outdatedReservations %>
  <% for reservation in outdatedReservations %>
    <div class="row alert-container z-depth-1">
      <div class="alert-img-container">
        
      </div>
      <div>
        <h2>Vous avez dépasser le délai de garde de :</h2>
        <ul>
          <li>{{ reservation.books.title }}</li>
        </ul>
      </div>
    </div>
  <% endfor %>
<% endif %>

```

Alerte utilisateur indiquant les livres en retard

## 8. La sécurité de l'application

Au fur et à mesure de ce document, j'ai égrainé les différentes mesures de sécurité dans l'application, dans cette partie, vous pourrez trouver les détails de ces mesures et celles dont je n'ai pas eu l'occasion de mentionner

- **Rôle utilisateur et protection des routes avec Security**

Grâce au bundle Security, je créé des rôles utilisateurs, ce qui permettra de gérer les accès aux pages et fonctionnalités suivant le rôle de l'utilisateur.

Les rôles sont hiérarchisés, ce qui veux dire qu'un rôle hérite de celui d'en dessous.

Toutes les pages sont ouvertes au « ROLE\_USER » minimum (exception de la page de login et d'inscription qui sont ouvertes à tout le monde), les différentes fonctionnalités seulement accessibles aux employés et + sont annotés sur leurs contrôleurs avec la méthode « IsGranted »

```
security:  
    role_hierarchy:  
        ROLE_USER: ROLE_USER  
        ROLE_EMPLOYEE: ROLE_USER  
        ROLE_ADMIN: ROLE_EMPLOYEE
```

Hiérarchie des rôles au sein de l'application

```
access_control:  
    - { path: ^/books, roles: ROLE_USER }  
    - { path: ^/reservations, roles: ROLE_USER }  
    - { path: ^/categories, roles: ROLE_EMPLOYEE }  
    - { path: ^/authors, roles: ROLE_EMPLOYEE }  
    - { path: ^/users, roles: ROLE_USER }  
    - { path: ^/, roles: PUBLIC_ACCESS }
```

Gestion des accès aux différents contrôleurs

- **Utilisation de l'UUID pour l'ID des utilisateurs**

Permettra dans le futur la fusion de bases de données utilisateur sans risque (ou très peu) conflit de doublons de clé primaire.

Les IDs ne seront jamais montrés dans les liens de page, pour récupérer l'utilisateur, j'utilise la méthode « `getUser` ».

- **Hachage des mots de passe utilisateur en BDD**

Aura pour effet de protégé les mots de passe des utilisateurs en cas d'une fuite de données. **Security** est configuré de façon à laisser Symfony s'occuper de gérer le type de cryptage (par défaut c'est BCRYPT).

```
password_hashers:  
    Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface: 'auto'  
    App\Entity\Users:  
        algorithm: auto
```

L'algorithme d'encryptage des mots de passe utilisateur est défini automatiquement

- Masquage des différentes pages et fonctionnalités administratives**

Dans la vue, les différents liens vers les pages et boutons de fonctionnalités administratif seront cachés à l'utilisateur qui n'a pas les droits.

```
{% if (is_granted("ROLE_EMPLOYEE")) %}
    {% include('headers/header-employee') %}
{% else %}
    {% include('headers/header-user') %}
{% endif %}
```

Gestion de l'affichage des menus de navigation du site

```
{% if (is_granted("ROLE_EMPLOYEE")) %}
    {% include('books/_delete_form') %}
{% endif %}
<a class="waves-effect waves-light btn modal-trigger btn-small {{ book.isFree ? '' : 'disabled' }}" href="#emprunte-modal">
    <i class="material-icons right">add</i>Emprunter</a>
```

Affiche le bouton de suppression du livre pour les employés et +

- Contraintes sur les formulaires & protection CSRF**

La faille CSRF est l'action de faire exécuter une action nécessitant un certain niveau d'accréditation sur un site web à un autre utilisateur authentifié et ayant les droits sur l'application.

Avec Validation, j'ai typé les champs et placé des contraintes sur les différents champs pour avoir une sécurité sur les injections SQL, les failles XSS et aussi avoir une cohérence dans les données entrantes.

Les formulaires Symfony créés avec le bundle « **Symfony Form** » intègrent automatiquement la protection contre ce type de faille, mais je l'ai mise en place sur toutes les actions nécessitant au minimum d'être un utilisateur validé par un employé (l'emprunt, l'annulation d'emprunt, la validation de nouveaux clients etc...).

Pour ce faire, je crée un formulaire contenant un champ caché contenant le jeton CSRF qui pointe vers le contrôleur de l'action protégé qui vérifiera avant d'effectuer l'action que le token est valide.

```
<form method="post" action="{{ path('books_reservations_new', {'id': book.id}) }}">
    <input type="hidden" name="_token" value="{{ csrf_token('create' ~ book.id) }}"/>
    <button type="submit" class="btn waves-effect waves-light"> Je confirme </button>
</form>
```

Bouton de confirmation envoyant le formulaire contenant le jeton CSRF.

```
public function buildForm(FormBuilderInterface $builder, array $options): void
{
    $builder
        ->add('child', 'name', [
            'label' => 'Nom de la catégorie',
            'constraints' => [
                new NotBlank([
                    'message' => 'ERREUR AJOUT CATEGORIE: Vous devez remplir ce champ.'
                ]),
                new Length([
                    'max' => 255,
                    'maxMessage' => 'ERREUR AJOUT CATEGORIE: Le titre doit contenir au maximum {{ limit }} caractères.'
                ]),
                new Type([
                    'type' => 'string',
                    'message' => 'ERREUR AJOUT CATEGORIE: Veuillez utiliser seulement des lettres.'
                ])
            ]
        ]);
}
```

Configuration du formulaire d'ajout de catégories

## 9. Le déploiement en production sur Heroku

La dernière étape fut le test de toutes les fonctionnalités, le traçage et la correction du moindre bug avant d'effectuer le déploiement sur Heroku. Heroku déploie les contenus via son répertoire **Github** créé en même temps que le projet.

Pour déployer l'application :

- Je crée un nouveau projet Heroku avec la commande tapée sur le terminal de mon pc « **heroku create mediatheque-chapelle-cureaux** », ce qui me créera une connexion avec le répertoire Git du projet.
- J'y ajoute l'addon « **JawsDB Maria** » qui créera la base de données en ligne.
- J'enregistre les identifiants de la nouvelle base de données dans une variable d'environnement de mon projet qui se nomme « **DATABASE\_URL** ».
- Je passe le serveur Heroku en mode production en ajoutant la variable d'environnement « **APP\_ENV** »
- J'ajoute aussi la variable « **APP\_SECRET** » qui servira pour les fonctions de cryptages par exemple.

Config Vars

KEY	VALUE	Action
APP_ENV	prod	
APP_SECRET	9846f67d8ccf24564c0fa07ee8457a75	
DATABASE_URL	mysql://i4hs5v8mzx3tzz8:jskldhdhhjek	
KEY	VALUE	Add

Variable d'environnement

- Je crée le fichier « **Procfile** » à la racine du projet, qui se chargera de pointer le fichier dans « **public/index.php** » comme point d'entrée de l'application.

```
Procfile
1   web: heroku-php-apache2 public/
```

Fichier Procfile

- Je génère le fichier « **.htaccess** » qui s'occupera de la gestion des redirections avec la commande Composer « **composer req symfony/apache-pack** »
- Je commit et push mon projet sur le répertoire Git du projet Heroku.
- J'effectue la migration Doctrine des différentes tables avec la commande « **heroku run php bin/console make:migrations:migrate** »

L'application est maintenant en ligne, il ne me restera plus qu'à importer un jeu de données.

# Traduction d'un article anglophone

Lors du développement de l'application, j'ai été confronté à une chose dont je n'avais pas une grande compréhension : les failles CSRF, mes différentes recherches m'ont mené vers des articles anglophones dont un particulièrement intéressant qui explique le fonctionnement de cette faille.

## a. Partie anglophone

### What is CSRF ?

A typical Cross-Site Request Forgery (CSRF or XSRF) attack aims to perform an operation in a web application on behalf of a user without their explicit consent. In general, it doesn't directly steal the user's identity, but it exploits the user to carry out an action without their will. For example, it can lead the user to change their email address or password in their profile or even perform a money transfer.

In a nutshell, a typical CSRF attack happens as follows:

1. The attacker leads the user to perform an action, like visiting a web page, clicking a link, or similar.
2. This action sends an HTTP request to a website on behalf of the user.
3. If the user has an active authenticated session on the trusted website, the request is processed as a legitimate request sent by the user.

As you can see, having the website affected by a CSRF vulnerability is not enough to make the attack successful. The user must also have an active session on the website. In fact, the CSRF vulnerability relies on the authenticated session management.

Typically, session management in a web application is based on cookies. With each request to the server, the browser sends the related cookie that identifies the current user's session. This usually happens even if the request is originated from a different website. This is the issue exploited by the attacker.

### Using a CSRF token

The typical approach to validate requests is using a CSRF token, sometimes also called anti-CSRF token. A CSRF token is a value proving that you're sending a request from a form or a link generated by the server. In other words, when the server sends a form to the client, it attaches a unique random value (the CSRF token) to it that the client needs to send back. When the server receives the request from that form, it compares the received token value with the previously generated value. If they match, it assumes that the request is valid.

## b. Partie anglophone traduite

### **Qu'est-ce que le CSRF ?**

L'attaque typique « Cross-Site Request Forgery » (CSRF ou XSRF) a pour but d'exécuter une opération sur une application web au nom d'un utilisateur sans son consentement explicite. En général, elle ne vole pas directement l'identité de l'utilisateur, mais elle l'exploite pour qu'il réalise une action sans qu'il le veuille. Par exemple, il peut obliger l'utilisateur à changer son adresse email ou son mot de passe dans son profil ou même réaliser un transfert d'argent.

En résumé, une attaque CSRF de base arrive comme ceci :

1. L'attaquant oblige l'utilisateur à effectuer une action, comme visité une page web, cliqué sur un lien, ou quelque chose de similaire.
2. Cette action envoie une requête HTTP à un site web à la place de l'utilisateur.
3. Si l'utilisateur a une session authentifiée active sur un site de confiance, la requête est exécutée comme une requête légitime envoyée par l'utilisateur.

Comme vous pouvez voir, avoir un site internet affecté par une vulnérabilité CSRF n'est pas suffisant pour que cette attaque soit accomplie avec succès. L'utilisateur doit aussi avoir une session active sur le site web. En fait, la faille CSRF repose sur la gestion de la session authentifiée

Typiquement, la gestion de session utilisateur dans une application web est basée sur les cookies. A chaque requête au serveur, le navigateur envoie le cookie qui identifie la session utilisateur courante. Ceci arrive généralement même si la requête est originaire d'un autre site. C'est le problème exploité par l'attaquant.

### **Utiliser un jeton CSRF**

L'approche typique pour valider une requête est d'utiliser un jeton CSRF, parfois il est aussi appelé un jeton anti-CSRF. Un token CSRF est une valeur prouvant que vous envoyez une requête provenant d'un formulaire ou d'un lien généré par le serveur. En d'autres mots, quand le serveur envoie un formulaire au client, il lui attache une valeur aléatoire unique (le jeton CSRF) que le client devra renvoyer. Quand le serveur reçoit la requête du formulaire, il compare la valeur du jeton reçu avec la valeur de celui généré précédemment. Si elles sont pareilles, ça veut dire que la requête est valide.

## Conclusion

Lors de la mise en place du projet, j'ai immédiatement compris que c'était l'étape la plus importante, en effet, tous les documents comme les diagrammes, les wireframes etc. m'ont aidé à construire de solides fondations pour la réalisation de l'application.

Auparavant j'avais déjà réalisé lors d'une évaluation d'entraînement un petit projet avec le Framework Symfony et j'y ai acquis de bonnes bases ainsi qu'une certaine attirance pour celui-ci et pour le back-end en général, le domaine étant très vaste et en perpétuelles mutations, je suis conscient que ce sera un métier où la veille technologique devra être effectué tout au long de notre carrière.

La réalisation de ce projet m'a aussi fait comprendre que le développement d'une application, même minime, était une chose assez longue à mettre en place tellement le nombre de tâches étaient importantes, d'où le fait d'avoir une organisation impeccable.

Bien gérer son temps est aussi une compétence obligatoire pour ce métier car nous avions un délai à respecter pour rendre cette évaluation ainsi que tous les documents annexes et il en sera ainsi tout du long de notre carrière de développeur.

# Annexes

**Connexion**

Email \_\_\_\_\_

Mot de passe \_\_\_\_\_

**CONNEXION >**

Écran de connexion

**Inscription**

Nom \_\_\_\_\_

Prénom \_\_\_\_\_

Date de naissance  
jj / mm / aaaa  
\_\_\_\_\_

Adresse \_\_\_\_\_

Adresse e-mail \_\_\_\_\_

Mot de passe \_\_\_\_\_

Confirmer votre mot de passe \_\_\_\_\_

**ENREGISTRER >**

Page d'inscription

**Médiathèque La Chapelle-Cureaux**

**RECHERCHE**

Rechercher un livre \_\_\_\_\_

Genre \_\_\_\_\_

**Disponible**

**Le monde de l'esclavage**

Cet ouvrage d'une ambition exceptionnelle présente sous une forme accessible à un large public une histoire inédite de l'esclavage depuis la Préhistoire jusqu'au présent. Il paraît vingt ans après le vote de la loi Taubira, alors que la prise de conscience du passé esclavagiste est chaque jour plus aiguisée au sein de la société française.

Genre: Histoire Sciences

**VOIR LE LIVRE**

**Disponible**

**L'institut**

En pleine nuit, à Minneapolis, des intrus pénètrent dans la maison de Luke Ellis, un surdoué de douze ans, tuent ses parents et le kidnappent. Quand le jeune garçon se réveille, à l'Institut, la chambre où il se trouve est semblable à la sienne – si ce n'est l'absence de fenêtre. Dans le couloir, d'autres portes cachent d'autres enfants, dotés comme lui de pouvoirs psychiques. Que font-ils là ? Qu'attendent-ils d'eux ? Et pourquoi ne cherchent-ils pas à s'enfuir ?

**NOUVEAUTÉ**

Page du Catalogue

 Médiathèque La Chapelle-Cureaux

**MENU**

### Ajouter un livre dans le catalogue

Titre \_\_\_\_\_

Date de parution  
jj / mm / aaaa  
\_\_\_\_\_

Auteur(s)  
\_\_\_\_\_ ▾ 

Type(s) de livre  
\_\_\_\_\_ ▾ 

Synopsis  
\_\_\_\_\_

**COUVERTURE** \_\_\_\_\_

**AJOUTER +**

[Admin] Page d'ajout de livre

 Médiathèque La Chapelle-Cureaux

**MENU**

### Mes emprunts

**Livre:** Louca - Tome 9 : Louca - Game Over  
**Date d'emprunt:** 30/10/2021 à 15:10  
**Date d'emprunt maximum:** 22/11/2021 20:11

**ANNULER L'EMPRUNT**

© 2021 Made By Kevin CHALLIT 

[Utilisateur] Liste des emprunts

 Médiathèque La Chapelle-Cureaux

**MENU**

### Liste des emprunts

**Statut:** Collecté (Dépassement)

**Livres:** La Vie invisible d'Addie Larue

**Utilisateurs:** Maria ORVALD

**Dates d'emprunt:** 08/10/2021 à 16:10

**Dates d'emprunt maximum:** 30/10/2021 17:10

**RENDU CLIENT**

**Statut:** Collecté (En cours)

**Livres:** Ils sont passés à l'acte

**Utilisateurs:** David WEBB

**Dates d'emprunt:** 20/10/2021 à 20:10

**Dates d'emprunt maximum:** 10/11/2021 21:11

[Admin] Page de gestion des emprunts



**MENU**

## Mon profil

Gérer votre *profil* et changer certaines informations personnelles de votre compte.

Nom

mojopojo

Prénom

sdfsdfsdf

Date de naissance

22 / 10 / 2015

Adresse

102 rue des coquelicots

Adresse e-mail

adminxxx@admin.fr

[Changez votre mot de passe](#)

Mot de passe



Confirmer votre mot de passe



[Page de gestion du profil utilisateur](#)



**MENU**

## Demandes d'adhésion

**Nom:** SFSDFSDF

**Prénom:** sdfsdfsdf

**Adresse email:** risdful@dani.fr

**Date de naissance:** 26/10/2021

**Adresse:** sdfsdfsdf

**ACCEPTER**

**REFUSER**

© 2021 Made By Kevin CHALLIT



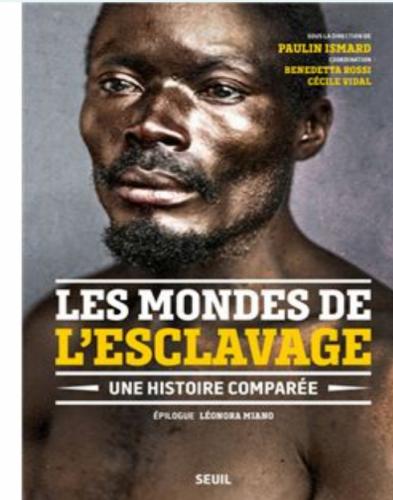
[\[Admin\] Page des demandes d'adhésion](#)



**MENU**

Disponible

[Le monde de l'esclavage](#)



**Synopsis:** Cet ouvrage d'une ambition exceptionnelle présente sous une forme accessible à un large public une histoire inédite de l'esclavage depuis la Préhistoire jusqu'au présent. Il paraît vingt ans après le vote de la loi Taubira, alors que la prise de conscience du passé esclavagiste est chaque jour plus aiguisée au sein de la société française.

**Date de parution:** 19/08/2021

**Auteur(s):**



[Page de détail d'un livre](#)