

# **DOSSIER DE PROJET**

Projet réalisé dans le cadre de la présentation au Titre Professionnel  
Développeur Web & Web Mobile

**Réalisé par Gautier FENAUX**

# SOMMAIRE

I. Présentation de Tellrz .....p.4

II. Cahier des charges.....p.4

A/ Produit minimum viable .....p.6

B/ Les cibles.....p.6

III. Benchmark.....p.7

IV. Étapes de préproduction .....p.9

A/ Arborescence .....p.9

B/ User story .....p.10

C/ Charte graphique .....p.10

D/ Zoning et Maquettes .....p.12

VI. Technologies utilisées.....p.12

A/ Argumentaire des choix techniques .....p.13

VII. Compétences du référentiel

Partie front-end : CP1 - Maquetter une application .....p.14

Partie front-end : CP2 - Réaliser une interface utilisateur web statique et adaptable.....p.15

I/ Réalisation de la base.....p.15

II/ Réalisation d'un composant HTML.....p.16

Partie front-end : CP3 - Réaliser une interface utilisateur web dynamique.....p.20

I/ Réalisation d'un drag'n'drop.....p.20

Partie front-end : CP4 – Réaliser une interface utilisateur avec une solution de gestion de contenu ou e-commerce.....p.24

I/ Les fonctionnalités de gestion de contenu .....p.24

Partie back-end : CP5 – Créer une base de données.....p.27

Partie back-end : CP6 – Développer les composants d'accès aux données.....p.31

I/ Création des entity et repository .....p.31

II/ Création des C.R.U.D.....p.33

III/ Création d'une requête DQL.....p.35

IV/ Gestion du changement de mot de passe.....p.35

Partie back-end : CP7 – Développer la partie back-end d'une application web ou web mobile .....p.38

I/ Gestion des attaques de type CSRF .....p.38

II/ Hachage des données sensibles.....p.39

III/ Sécurisation des routes.....p.41

IV/Gestion des attaques de force-brute.....p.41

Partie back-end : CP8 – Elaborer et mettre en œuvre des composants dans une application de gestion de contenu ou e-commerce.....p.42

I/ Gestion du réagencement des chapitres .....p.42

II/ Gestion de la liste de favoris.....p.43

A/ Ajout à la liste de favoris .....p.43

B/ Accès aux données.....p.45

Recherche en anglais .....p.49

## **I. Présentation de Tellrz**

Le projet Tellrz, est une plateforme d'édition en ligne permettant aux auteurs de pouvoir présenter leurs œuvres et de créer une communauté de lecteurs. Le but est d'offrir une visibilité aux personnes rencontrant des difficultés à se faire publier via le monde classique de l'édition.

A terme le projet Tellrz est de proposer une « gamification » de la lecture au fur à mesure de la lecture, l'auteur peut poser des questions sur divers passages sur lesquels il souhaite avoir le retour de ses lecteurs. De cette manière, ces derniers deviennent force de proposition sans pour autant imposer leur point de vue.

D'autre part, des auteurs souhaitant rendre une partie de leur travail payant auront la possibilité de le faire.

L'idée est de développer une communauté de lecteurs/critiques autour de différents auteurs ceci pourront être soutenus, grâce à un système de dons.

Afin de continuer leur lecture, les utilisateurs devront répondre à des questions qui pourront être prises en compte ou non en compte par l'auteur afin que celui-ci est un retour sur son travail.

## **II. Cahier des charges**

### A - Les spécifications fonctionnelles

#### Landing page (visiteur)

- Accès aux pages « À propos », « Contact » et « mentions légales »
- Bouton de connexion utilisateur
- FAQ
- Accès partiel aux œuvres

#### Landing page (utilisateur)

- Accès aux outils d'éditions
- Accès à la liste de lecture

#### Page de login

- Formulaire de connexion
- Accès à « Mot de passe oublié »

#### Profil utilisateur

- Visualisation de ses informations personnelles
- Visualisation de ses manuscrits et de sa liste de lecture
- Bouton de création d'un nouveau livre

#### Pages de création de manuscrit

- Création des informations générales sur le manuscrit
- Création des chapitres

#### Page manuscrit

- Visualisation des informations générales
- Visualisation des chapitres
- Bouton de création de chapitre

#### Page chapitre

- Visualisation des chapitres

#### Page « Qui sommes-nous ? » (visiteur et utilisateur)

- Présente les personnes qui ont contribué au projet, leurs rôles, ainsi que leurs moyens de contact.

#### Page contact pour support technique

- Formulaire de contact pour assistance

#### Page mentions légales

Evolution des fonctionnalités :

Une sélection aléatoire de livres pour les lecteurs souhaitant découvrir des nouveaux univers. Un concours avec thème,

Un format de série hebdomadaire par catégorie, en changeant de lecteur de manière régulière (mois, trimestre, à définir... ).

Les utilisateurs pourront laisser des commentaires, envoyer des messages à l'auteur.

## **B/ Produit minimum viable**

La première version de Tellrz comprendra les fonctionnalités de bases de l'application séparé en deux versants : utilisateur connecté et utilisateur non-connecté.

### **Utilisateur non-connecté :**

- Landing page présentant l'application + accès partiel aux manuscrits
- Page de création de compte
- Page de login
- Page de contact
- FAQ
- Mentions légales

### **Utilisateur connecté :**

- Landing page avec accès total aux manuscrits
- Accès au profil utilisateur avec fonctionnalités CRUD
- Accès aux outils d'édition de manuscrits
- Accès aux manuscrits créés : chapitres et informations générales
- Accès à la liste de lecture

## **C/ Les cibles**

La plateforme s'adresse aussi bien aux auteurs qu'aux lecteurs. Nous aimerions nous démarquer des sites comme Wattpad en proposant un contenu de qualité c'est pourquoi l'identité du site aura un aspect sérieux, des couleurs sobres. L'âge du public visé ne sera pas en dessous de huit ans.

Peut-être mettre en place des membres correcteurs (payants), mettre en place des membres régulateurs ?

Le placement du produit sera plus axé sur les csp+, esthétiquement il sera sobre et élégant.

### **III. Benchmark**

#### **LIBRINOVA :**

Le site librinova propose une prise en charge complète de l'auteur allant de l'aide à la rédaction la plus basique, aux graphismes de la couverture, à l'envoi de manuscrit jusqu'aux services d'un agent littéraire.

Le site librinova propose des offres commerciales d'accompagnement de l'auteur à partir de 39euros pour suivre une masterclass en ligne, ce sont des vidéos non personnalisées qui traitent de différents sujets tels que : l'écriture d'une description, la trouvaille de l'inspiration ou encore le rôle de l'éditeur, etc.

Une deuxième offre personnalisée à 120 euros qui permet la lecture de son manuscrit par une bêta lectrice , celui-ci retourne un rapport de 2, 3 pages de critiques positives et négatives.

Le site s'adresse à des personnes qui ne peuvent avoir de retour sur leurs manuscrits dans le milieu de l'édition classique sans investissement financier. Les commentaires peuvent être ajoutés sur un paragraphe.

Enfin, une troisième offre personnalisée à 195 euros se décomposant de la manière suivante : un questionnaire est envoyé afin de cibler les problématiques de l'auteur, ensuite un coach dédié propose deux séances d'une demi-heure pour répondre à celles-ci.

Des lecteurs professionnels lisent et élaborent une fiche de lecture pour pour un manuscrit de 100 000 mots, cela pour un prix de 390 euros et 490 euros si le manuscrit est plus long.

Ils offrent également des packs entre 230 euros à 2515 euros, selon le prix les offres de prise en charge varient entre ces différentes options:

- les services de publication,
- les services commerciaux,
- les services d'impression,
- les services graphiques,
- les services promotionnels,
- les services éditoriaux,

Trop étoffé, navigation difficile (surtout pour un public âgé), il faut aller chercher l'information, l'information ne vient pas à l'utilisateur. Beaucoup de texte.

Avantages nombreux services, mais qui ont tendance à perdre l'utilisateur, difficile de définir lequel est le plus approprié.

#### **WATTPAD :**

Wattpad est le site qui se rapproche le plus de ce que je veux faire dans sa philosophie. Cependant il ne met pas en place une gamification pour lire les œuvres.

En octobre 2021, Wattpad compte 94 millions d'utilisateurs dont 5 millions sont des auteurs. Pour un total de plus de 23 milliards de minutes chaque mois mis au profit d'histoires originelles. L'application constitue aujourd'hui l'archive de plus d'un milliard d'histoires 4. 90 % des utilisateurs font partie de la Gen Z ou Millennial.

La durée moyenne d'une séance de lecture est de 37 minutes 6 secondes.

Permet aux auteurs de partager leurs histoires et de les ouvrir à de nouveaux publics en explorant les possibilités créatives offertes par les commentaires et le travail collaboratif. Les lecteurs sont notamment invités à voter pour les textes, en évaluer le contenu et à émettre des conseils. Ils peuvent également signaler des fautes ou encore souligner un passage particulièrement réussi. Les commentaires constructifs peuvent servir pour l'amélioration du texte et faire évoluer l'auteur dans son écriture. L'application tire donc son originalité d'une écriture tributaire d'autrui, au cœur de laquelle l'interaction ouverte entre auteur et lecteur tient une place cruciale. Les lecteurs ont la possibilité de les rémunérer s'ils le souhaitent.

Inconvénients : Couleurs criardes, manuscrits mal écrits, contenu de basse qualité, voir médiocre.

Système de coin pour acheter des livres.

Lorsque qu'on arrive sur la home page, proposition de livre en fonction des thèmes cochés lors de l'inscription. Sélection d'extraits de livre payant ou manuscrit complet non-payant.

Offre commerciales premium classiques, contenu hors ligne illimité, personnalisation du thème, pas de publicité, pièce en bonus lors d'achat de livre.

## **TELLRZ :**

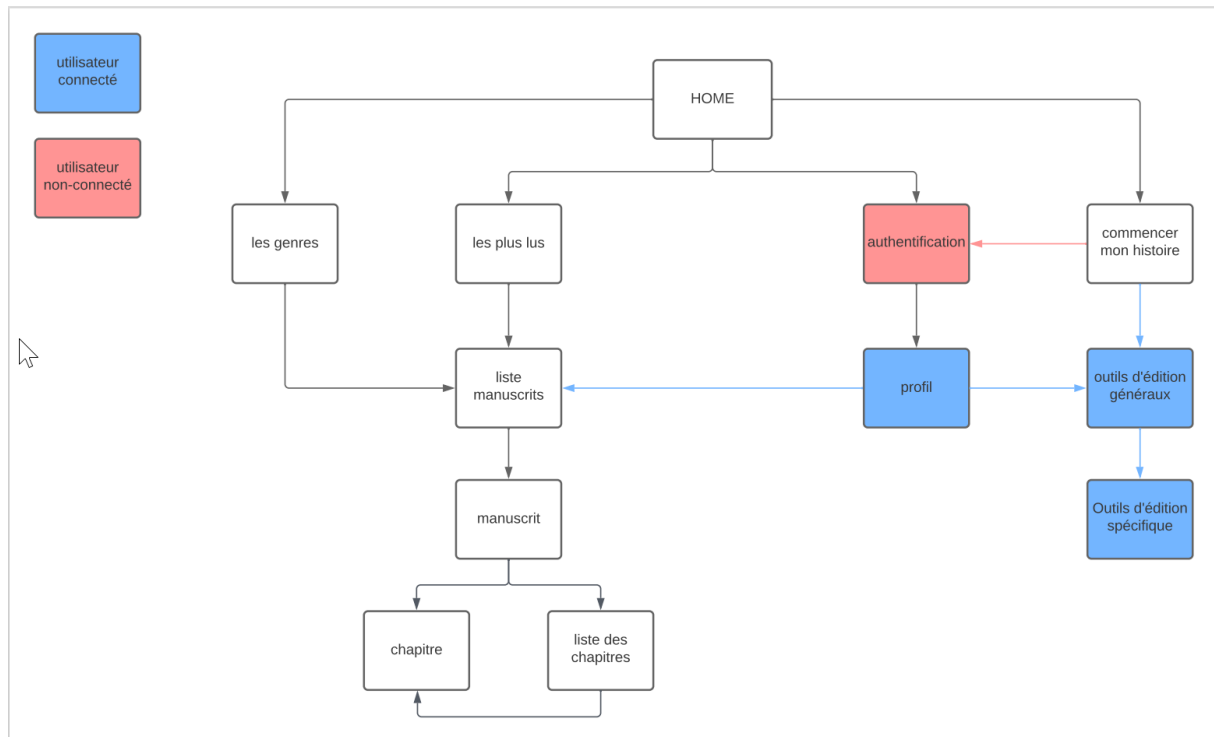
Le site s'adresse à des personnes étant refusés dans les maisons d'éditions traditionnelles By-passant le rôle du beta lecteurs de librinova. Contenu de meilleure qualité que sur wattpad. Eviter à l'auteur d'engranger des frais pour pouvoir avoir un retour sur son travail.

L'entièreté du manuscrit n'est pas à disposition de tous les utilisateurs, car ceux-ci doivent donner une contre-partie à l'auteur par le biais de leurs critiques.



## IV. Etapes de préproduction

### A/ Arborescence



Cette arborescence a été faite en toute première étape du projet, elle a permis de schématiser l'accès aux typologies de page en fonction du statut de l'utilisateur.

### B/ User story

	En tant que...	Je veux pouvoir...	Afin de...
Utilisateur 1	Visiteur	Lire des livres et écrire des commentaires et les noter	Découvrir des nouveaux auteurs et donner mon avis pour veiller à un contenu de qualité
Utilisateur 2	Membre connecté	Ecrire des manuscrits et recevoir les critiques de mes lecteurs	Faire connaître mes œuvres et créer une communauté de lecteurs

## C/ Charte graphique

L'application Tellrz se voulant une alternative à son concurrent Wattpad, il a été décidé de créer une esthétique sobre et élégante autour de deux couleurs principales :



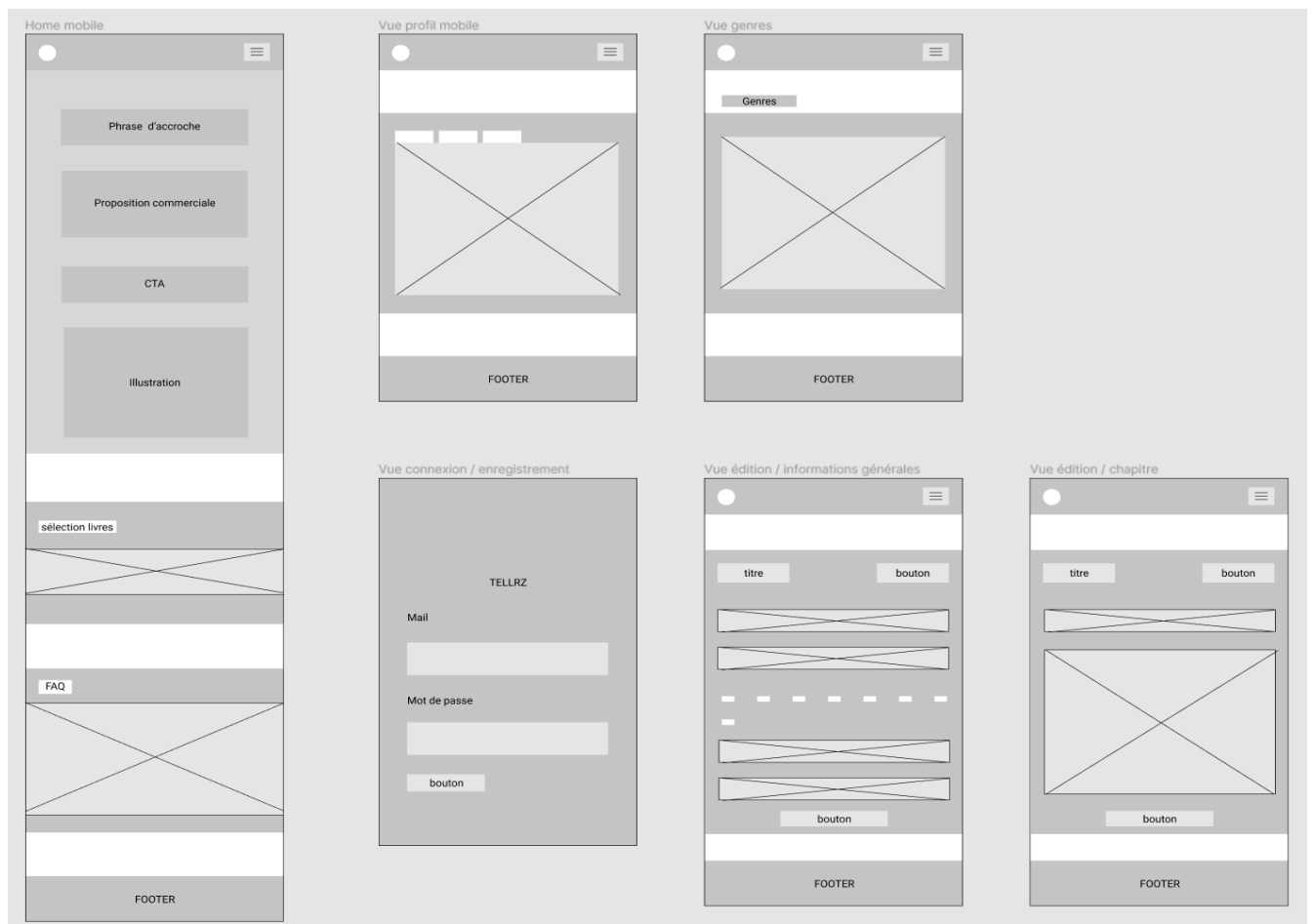
Ce bleu particulier invite à l'évasion, au voyage et donc à l'imaginaire du lecteur que l'on souhaite mettre en éveil.

Cet orange contraste au niveau des tons et complète la première couleur, il vient apporter de la chaleur et de lumière par rapport au premier.

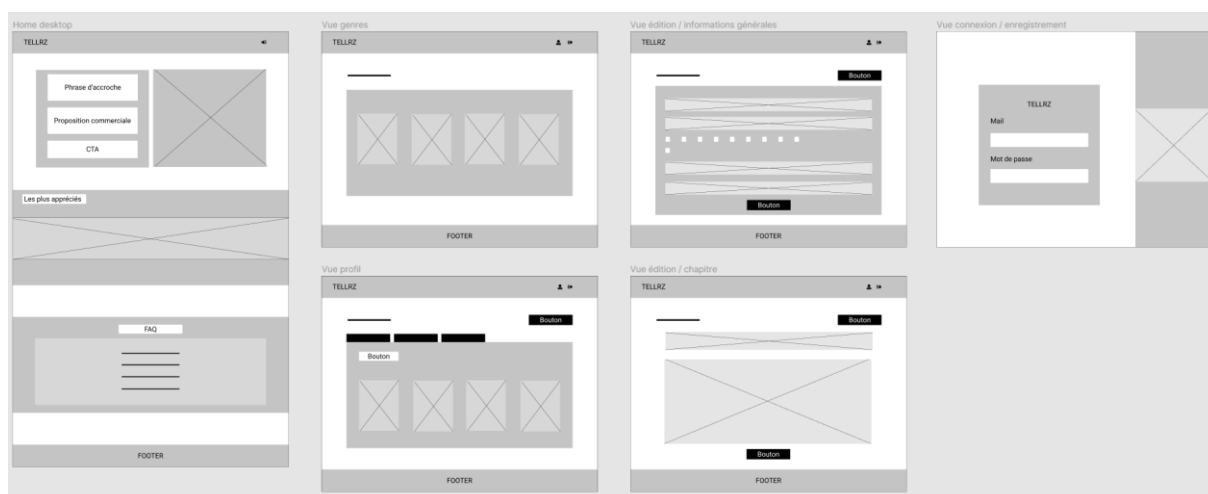
## D/ Zoning et maquettes

J'ai ensuite décliné les différentes typologies de page sous forme de zoning (mobile et desktop). Ces différentes vues permettront l'accès au produit minimum viable.

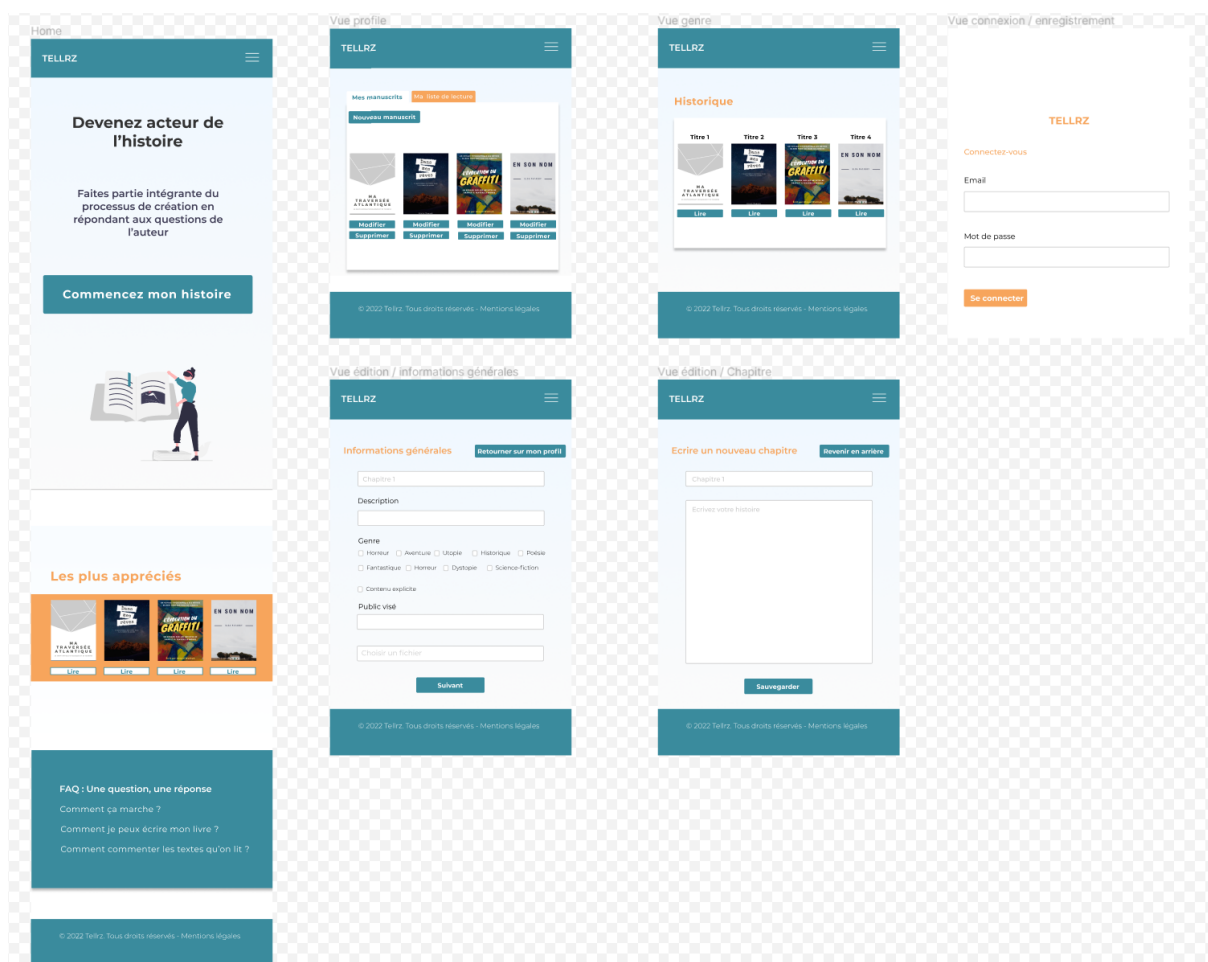
### Zoning version mobile



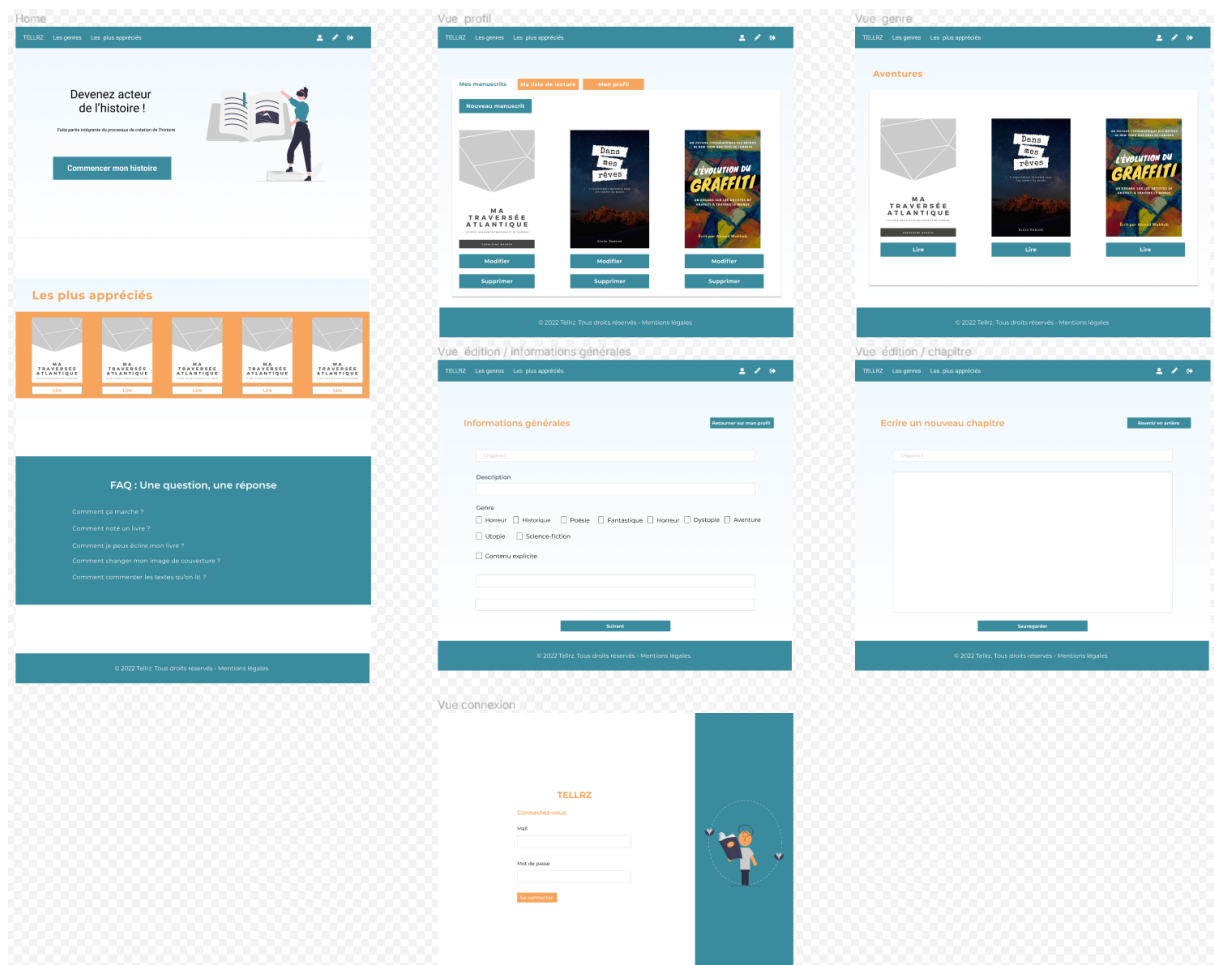
## Zoning version desktop



## Maquette version mobile



## Maquette version desktop



## V. Technologies utilisées

Figma : Pour la mise en place du zoning et de la maquette.

JMerise : Pour la création des MCD.

IDE : Visual Studio code.

HTML 5 : Utilisé pour réaliser la maquette statique du site web.

CSS 3 : Pour l'habillage du site.

Bootstrap 5 : Pour la réalisation de la partie front-end orientée mobile first

jQueryv3.6.0: Bibliothèque JavaScript utile à la manipulation des données du projet et facilite l'écriture de JavaScript.

Framework Symfony : Pour la réalisation de la partie back-end et de la partie front-end

Chrome : Nécessaire pour les différents tests et à la veille informationnelle.

## A/ Argumentaire des choix techniques

Maquettage et arborescence :

Figma est un outil très souple et facile à prendre en main pour un débutant.

### Front-end :

J'ai fait le choix d'utiliser le framework css Bootstrap 5 développé par les équipes de Twitter en 2011 puis déposé par la suite sur GitHub en open-source.

Il contient des codes HTML 5, CSS 3 et Javascript réutilisables et des composants graphiques comme des boutons, des formulaires, ou encore des miniatures.

Son utilisation favorise la mise en place d'un design responsive orienté mobile first avec son système de grille en 12 colonnes et ses nombreux breakpoint qui permettent une souplesse dans la disposition des éléments. Il permet un gain de temps considérable dans la mise en page des éléments.

De plus, bootstrap bénéficie d'une documentation simple à comprendre avec de nombreux exemples et des composants directement réutilisables à partir de la documentation.

J'ai opté pour l'utiliser avec CDN (Content Delivery Network), celui favorise l'expérience utilisateur, grâce au délestage. En effet, il agit comme une couche supplémentaire située à proximité physique du réseau de l'utilisateur final (au lieu du serveur d'où provient le contenu). Cela permet un temps de chargement plus rapide.

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
```

Concernant la manipulation du DOM, j'utilise une bibliothèque jQuery 3.6.0 et du javascript.

Je passe par l'IDE Visual Studio Code pour développer ce projet pour les différentes fonctionnalités qu'il propose ainsi que les diverses extensions qui aident à la visualisation des bugs tels intelephense pour le php, ou qui permettent une bonne indentation et une auto-complétion efficace.

### Back-end :

Concernant Symfony c'est un framework php complet avec une architecture MVC(model, view, controller). Ce framework permet un gain de temps dans la mise en place du projet. De plus, le moteur de template Twig permet une manipulation des données souple avec la possibilité d'insertion de logique dans la partie front-end directement. De nombreux bundles sont disponibles et rendus accessible via des commandes simples : make :entity, make :user, make :crud, make :controller , etc...

# Partie front-end : CP1 - Maquetter une application

En 2021, le trafic Internet mobile représentait 55,56 % du trafic Internet mondial total, soit plus de la moitié de la navigation totale. Fort de ce constat, j'ai décidé de réaliser la maquette d'abord en version mobile first. Les éléments que l'on retrouve dans le zoning sont classiques et participent à la bonne lecture de l'information.

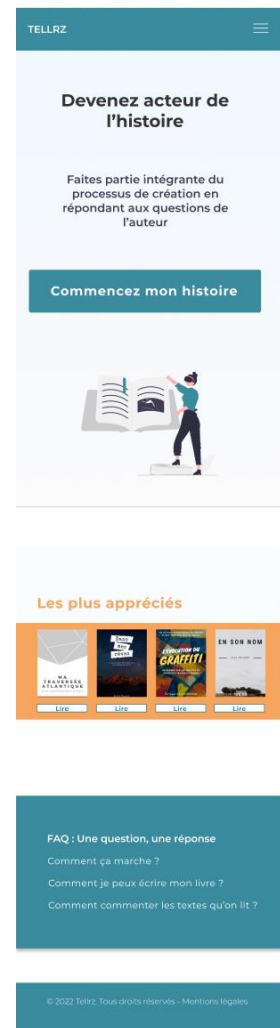
On retrouve le logo du site en haut à gauche et la navigation sous forme de burger menu déroulant sur la droite.

Le header présente une phrase d'accroche, une proposition commerciale et tout de suite après un CTA (Call to action), un bouton très visible qui donne envie à l'utilisateur de cliquer dessus. D'un point de vue objectif, cet enchaînement est logique : il accompagne le parcours utilisateur jusqu'à l'objectif souhaité, soit la création d'un compte sur notre site.

Juste en dessous de celui-ci on trouve une liste des livres les plus appréciés par les lecteurs. Cela permet à l'utilisateur d'accéder au contenu les mieux répertoriés par la communauté, il peut ainsi se faire une idée du type de contenu qu'il peut trouver sur la plateforme.

S'ensuit une FAQ avec toutes les questions les plus récurrentes autour de l'utilisation de l'application.

Puis un footer classique avec accès aux mentions légales.



# Partie front-end: CP2 - Réaliser une interface utilisateur web statique et adaptable

## I – Réalisation de la base

Afin de réaliser l'interface web j'ai utilisé le framework Bootstrap, j'ai pris soin de respecter la charte graphique des maquettes.

J'ai tout d'abord créé une base, ici un extrait du code de la barre de navigation :

```
<nav class="navbar navbar-expand-lg navbar-light">
  <div class="container-fluid">
    <a class="nav-link active text-white bold" aria-current="page" href="{{ path('home') }}" >TELLRZ</a>
    <button class="navbar-toggler shadow-none" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNavDropdown" aria-controls="navbarNavDropdown" aria-expanded="false" data-keyboard="true" data-toggle="tooltip">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNavDropdown">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0 w-100">
        <li class="nav-item dropdown">
          <a class="nav-link dropdown-toggle text-white" href="#" id="navbarDropdownMenuLink" role="button" data-bs-toggle="dropdown" aria-expanded="false">
            Les genres
          </a>
          <ul class="dropdown-menu" aria-labelledby="navbarDropdownMenuLink">
            {# <div class="nav-dropdown-style d-flex flex-wrap justify-content-center"> #}
            <li>
              <a class="dropdown-item" href="{{ path('genre-detail', {'slug' : 'horreur'}) }}">Horreur</a>
            </li>
            <li>
              <a class="dropdown-item" href="{{ path('genre-detail', {'slug' : 'aventure'}) }}">Aventure</a>
            </li>
            <li>
              <a class="dropdown-item" href="{{ path('genre-detail', {'slug' : 'science-fiction'}) }}">Science-fiction</a>
            </li>
            <li>
              <a class="dropdown-item" href="{{ path('genre-detail', {'slug' : 'utopie'}) }}">Utopie</a>
            </li>
            <li>
              <a class="dropdown-item" href="{{ path('genre-detail', {'slug' : 'historique'}) }}">Historique</a>
            </li>
            <li>
              <a class="dropdown-item" href="{{ path('genre-detail', {'slug' : 'fantastique'}) }}">Fantastique</a>
            </li>
            <li>
              <a class="dropdown-item" href="{{ path('genre-detail', {'slug' : 'poesie'}) }}">Poésie</a>
            </li>
            <li>
              <a class="dropdown-item" href="{{ path('genre-detail', {'slug' : 'action'}) }}">Action</a>
            </li>
            <li>
              <a class="dropdown-item" href="{{ path('genre-detail', {'slug' : 'nouvelle'}) }}">Nouvelle</a>
            </li>
            <li>
              <a class="dropdown-item" href="{{ path('genre-detail', {'slug' : 'classique'}) }}">Classique</a>
            </li>
          </ul>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#" style="color: white">Les plus appréciés</a>
        </li>
      </ul>
    </div>
  </div>
</nav>
```

Extrait du footer :

```
{% block body %}{% endblock %}
<footer class="background-color">

  <div class="text-center p-3 d-flex justify-content-center align-items-center mh-100">
    <p class="me-2 text-white">© 2022 Tellrz. Tous droits réservés -
    </p>
    <p>
      <a class="text-white" href="https://mdbootstrap.com/">
        Mentions légales</a>
    </p>
    <p></p>
  </div>

</footer>
```

On remarque ici les deux balises de block body twig dans lesquelles on va inclure les éléments HTML des autres templates en faisant un extends de la base en début de la plupart des fichiers html.twig.

```
{% extends 'base.html.twig' %}
```

## II – Réalisation d'un composant html :

```
<div class="global pt-5 pb-5 " id="content">
  <div class="container">
    <div class="wrapper mt-3">
      <input class="radio" id="one" name="group" type="radio" checked>
      <input class="radio" id="two" name="group" type="radio">

      <div class="tabs">
        <label class="tab" id="one-tab" for="one">Mes manuscrits</label>
        <label class="tab" id="two-tab" for="two">Ma liste de lectures</label>
      </div>

      <div class="panels">
        <div class="panel" id="one-panel">
          <div class="panel-body">
            <a href="{{ path('edition_tools_new') }}" class="btn mt-3 ms-3">Nouveau manuscrit</a>
            {% if manuscripts|length > 0 %}
              <div class="mt-5 d-flex horizontalScroll">
                {% for manuscript in manuscripts %}
                  <div class="">
                    <!-- Mettre une height fixe ? pour avoir un objet de meme largeur et hauteur mettre le style dans le css -->
                    <div class="card-body d-flex flex-wrap justify-content-center">
                      <h5 class="card-title overflow-hidden">{{ manuscript.title }}
                      </h5>
                      
                      <a href="{{ path('chapters_list', {'id': manuscript.id}) }}" class="btn btn-sm w-100 align-self-end mt-3">Modifier</a>
                    </div>
                  </div>
                {% endfor %}
              </div>
            {% endif %}
          </div>
        </div>

        <div class="panel" id="two-panel">
          <div class="panel-title">Take-Away Skills</div>
          <p>You will learn many aspects of styling web pages! You'll be able to set up the correct file structure, edit text and colors, and create attractive
          </p>
        </div>
      </div>
    </div>
  </div>
```

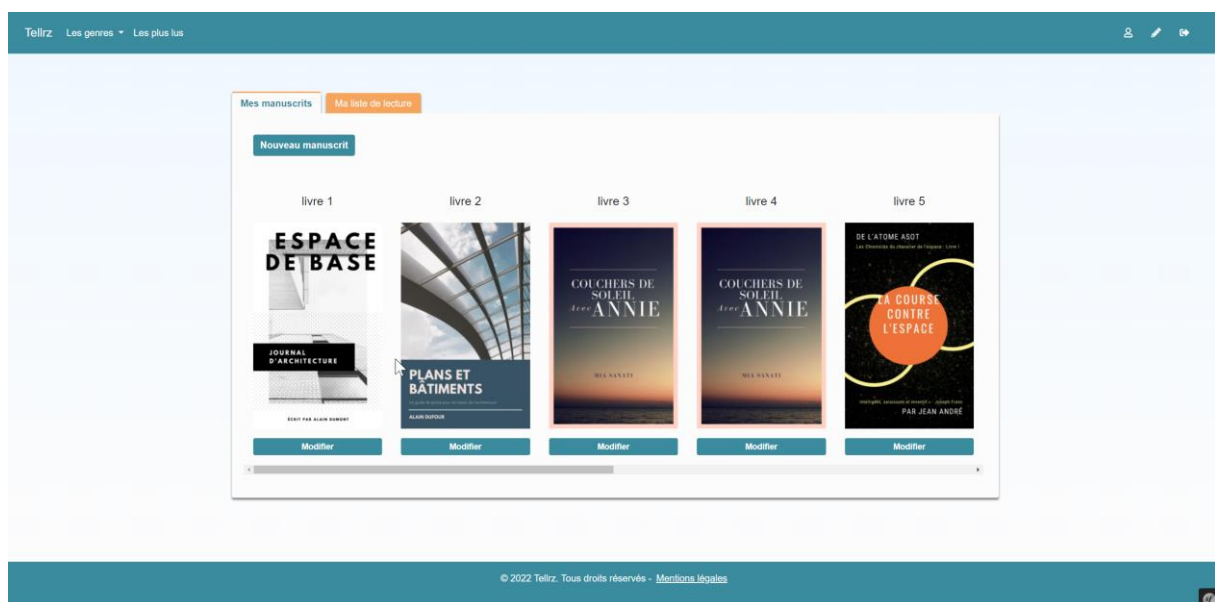
Afin de réaliser le panneau, je suis passé par du css et un système d'input radio et de div en display none.

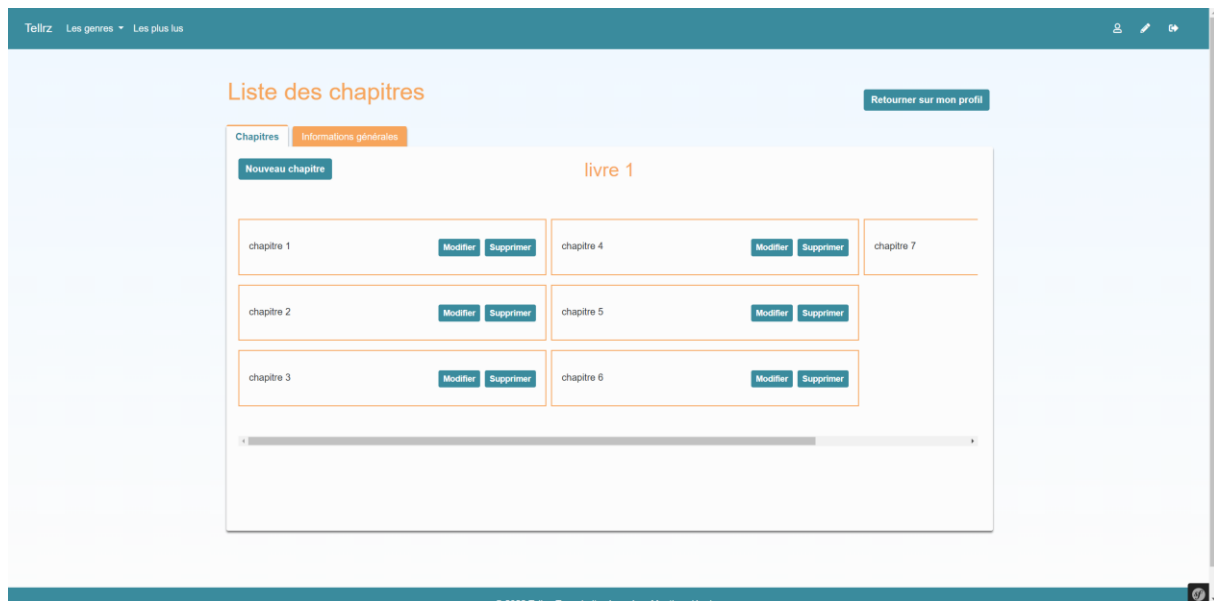
Les tabs étant des labels elles sont reliées aux inputs avec l'attribut for qui prend en valeur l'id de l'input.



Ainsi les balises input peuvent changer d'état (checked) et permettent d'appliquer le style : le panel qui correspond au bouton cliqué fait passer l'input radio en état : checked et cela fait passer en display block et le panel avec l'id one-panel ou two-panel selon le label cliqué. En effet on va venir sélectionner la div avec la class .panels avec le sélecteur de voisin ~ puis, une fois celui-ci ciblé, on pointe par exemple vers la div avec l'id #one-panel avec le sélecteur de descendant qui est '' (espace). C'est la même chose pour changer le style d'une tab.

```
.radio {
  display: none;
}
#one:checked ~ .panels #one-panel,
#two:checked ~ .panels #two-panel
{
  display: block;
}
#one:checked ~ .tabs #one-tab,
#two:checked ~ .tabs #two-tab
{
  background: #fcfcfc;
  color: #3a8b9d;
  border-top: 3px solid #F7A55C;
  font-weight: bold;
}
```





Concernant l'aspect responsive pour l'affichage des chapitres j'ai également utilisé le framework css bootstrap.

Néanmoins il a fallu prendre en compte l'orientation différente de l'affichage sur desktop (horizontale) et mobile (verticale).

Afin de rendre le contenu responsif, j'utilise le système de colonne pour passer le contenu sur une seule colonne j'utilise le col-12 (le contenu prend la largeur des douze colonnes, ce qui visuellement donne un élément sur une colonne). Ensuite, pour avoir plusieurs colonnes à partir du moment où le viewport est supérieur ou égal à 768px (breakpoint) j'utilise la class col-md-5, chaque div prendra donc 2,4 colonnes.

Concernant l'affichage des titres, j'ai positionné les titres au-dessus des boutons en vue mobile et en début de div sur les grand-écran à partir de 992px grâce à la class d-lg-flex. Les deux boutons servant à modifier ou supprimer un chapitre sont placés l'un à côté de l'autre grâce à la class d-flex également.

Le titre et les boutons sont également en d-lg-flex (992px) pour être placés en ligne, avec en plus l'ajout d'une marge supérieure à partir de mt-md-3 (768px) pour un meilleur rendu graphique.

```

{% if chapters is not null %}
<ul class="mt-5 d-md-flex flex-column align-items-center flex-wrap m-0 wrapperChapters horizontalScroll pt-3 box">
  {% for key, value in chapters %}
    <li class="border border-2 p-lg-3 mb-3 col-12 col-md-5 me-2 max-height-chapter " data-id="{{ value.id }}">
      <div class="d-lg-flex justify-content-between col-12 mt-md-3 dataBox" data-id="{{ value.id }}">
        <div class="w-100 overflowParent ">
          <div class="overflow-ellipsis">
            <p class="value text-center text-lg-start">{{ value.title }}</p>
          </div>
        </div>
        <div class="mt-2 mt-md-0 d-flex justify-content-center">
          <a href="{{ path('chapter_edit', {'id': value.id}) }}" class="text-center btn btn-sm me-2 height-button">Modifier</a>
          <form method="post" action="{{ path('chapter_delete', {'id': value.id}) }}" onsubmit="return confirm('Etes-vous sur de vouloir supprimer ce chapitre?');">
            <input type="hidden" name="_token" value="{{ csrf_token('delete' ~ value.id) }}">
            <button class="btn btn-sm">Supprimer</button>
          </form>
        </div>
      </div>
    </li>
  {% endfor %}
</ul>

```

Liste des chapitres

Retourner sur mon profil

Chapitres

Informations générales

livre 1

Nouveau chapitre

chapitre 1

Modifier Supprimer

chapitre 2

Modifier Supprimer

chapitre 3

Modifier Supprimer

chapitre 4

# Partie front-end : CP3 - Réaliser une interface utilisateur web dynamique

## I – Réalisation d'un drag'n'drop

Afin de réaliser une interface dynamique j'ai utilisé à la fois le langage script client javascript et le framework JQuery. J'ai complété le traitement des données reçues au niveau de mon back-end.

Afin que ma plateforme soit optimisée pour l'expérience utilisateur (UX), j'ai souhaité mettre en place un draggable au niveau de mes chapitres et que celui-ci soit traité de façon asynchrone. Cette fonctionnalité permettra à l'utilisateur de réagencer l'ordre de ces chapitres de façon fluide sans rechargement de page.

Pour mettre en place cela il m'a fallu tout d'abord paramétrer une position sur mon entité chapitre avec ses getter et ses setter pour pouvoir paramétrer la propriété position.

```
#[ORM\Column(name: 'position', type: 'integer')]
private int $position;
```

```
public function setPosition(int $position): void
{
    $this->position = $position;
}

public function getPosition(): int
{
    return $this->position;
}
```

J'ai mis en place un script javascript qui fonctionne en cascade.

```
((() => {enableDragSort('box')}) ());
```

Cette syntaxe entre parenthèse de la fonction permet au parser du navigateur de comprendre qu'il s'agit d'une expression de fonction, puis les parenthèses après les accolades permettent d'appeler la fonction automatiquement lors du parsing, c'est ce qu'on appelle une **Immediately invoked function execution (IIFE)**.

```
function enableDragSort(listClass) {
  const chaptersArray = document.getElementsByClassName(listClass);
  Array.prototype.map.call(chaptersArray, (list) => {enableDragList(list)});
}
```

La fonction enableDragSort() prend en argument un nom de classe, dans le cas présent il n'y a qu'une liste avec la classe box, celle des chapitres, représentée dans la balise <ul></ul>. On manipule le DOM afin de récupérer les éléments par la classe. Puis sur chaque liste qui est une collection de type HTMLCollection on appelle la méthode map qui va permettre de renvoyer une liste sur laquelle la fonction enableDragList() va être appelé.

La méthode map() crée un nouveau tableau avec les résultats de l'appel d'une fonction fournie en argument sur chaque élément du tableau appelant. Les éléments renvoyer par le DOM sont des nœuds c'est pourquoi il faut passer par la méthode map du Array.prototype.map et le .call permet d'appeler cette méthode sur la HTMLCollection

```
function enableDragList(list) {
  Array.prototype.map.call(list.children, (item) => {enableDragItem(item)});
}
```

La fonction enableDragList() récupère une liste en argument, sur cette dernière on récupère la propriété children grâce à la notation pointée qui va intercepter toutes les balises <li></li> et les mettre dans une HTMLCollection, puis, celui-ci va être mappée selon le même process que la fonction précédente, ainsi chaque élément va retourner le résultat de la fonction enableDragItem().

```
function enableDragItem(item) {
  item.setAttribute('draggable', true)
  item.ondrag = handleDrag;
  item.ondragend = handleDrop;
}
```

La fonction EnableDragItem() prend un item en argument, puis on utilise la méthode setAttribute de l'objet récupéré pour ajouter l'attribut html universel draggable que l'on initialise à true, ce qui permet de rendre draggable une balise.

Enfin on initialise les écouteurs d'événement ondrag et ondragend, respectivement par les fonctions handleDrag et handleDrop

```
function handleDrag(itemEvent) {  
    const selectedItem = itemEvent.target,  
          list = selectedItem.parentNode,  
          x = event.clientX,  
          y = event.clientY;  
  
    let swapItem = document.elementFromPoint(x, y) === null ? selectedItem : document.elementFromPoint(x, y);  
  
    if (list === swapItem.parentNode) {  
        swapItem = swapItem !== selectedItem.nextSibling ? swapItem : swapItem.nextSibling;  
        list.insertBefore(selectedItem, swapItem);  
    }  
}
```

Toute la gestion du drag'n'drop réside autour de la méthode insertBefore. Cette méthode reçoit en argument un nouveau nœud à insérer et un nœud de référence.

Deux éléments sont récupérés au moment du onDrag. L'élément cliqué (nouveau nœud à insérer) et un élément évolutif (nœud de référence) dont la valeur change tout au long du Drag'n'Drop en fonction de la position du curseur de la souris récupérée au moment de l'événement sur le DOM (document.elementFromPoint(x,y)). On vérifie si le nœud de l'élément de référence et l'élément cliqué ont le même parent, dans notre cas cela est toujours vrai.

Dans le cas où le swapItem est différent du prochain parent de l'élément sélectionné alors la valeur de référence reste le swapItem situé au niveau du curseur, ainsi le nœud du selectedItem vient se placer à la place de celui du swapItem du fait qu'ils aient le même parent.

Dans l'autre cas si le swapItem est le même que le frère de l'élément cliqué alors le swapItem change de valeur et prend celle de son prochain parent. Ainsi la valeur de référence change et permet d'insérer le nœud de l'élément sélectionné devant celui de référence.

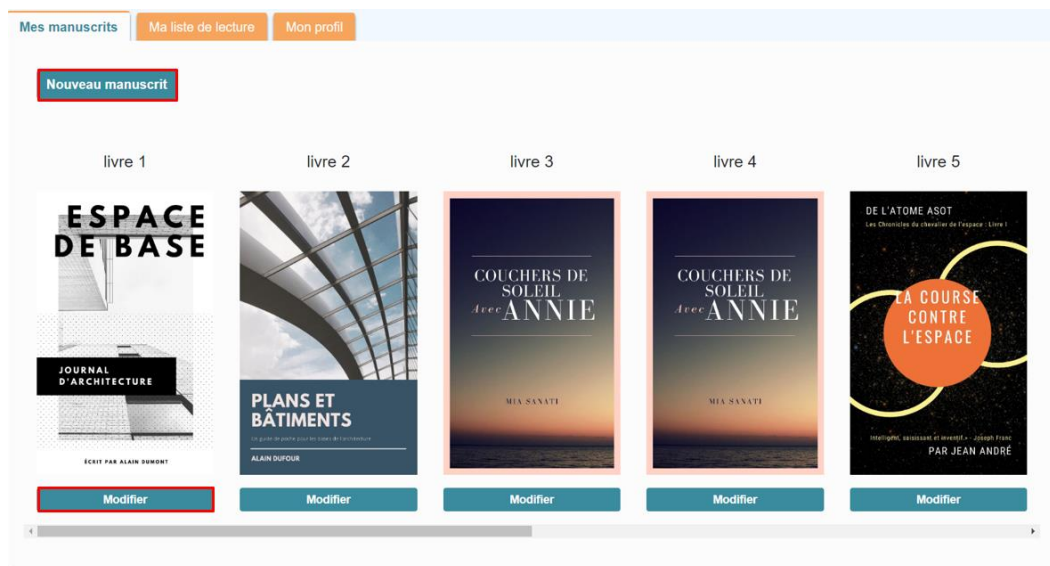
Enfin la gestion du drop est effectuée via la méthode handleDrop dont je détaillerai le fonctionnement dans la partie qui traite de la compétence professionnelle n°8. Cette fonction permet le réagencement de l'ordre en base de données, en effet dans le controller les chapitres sont récupérés par ordre de position, ce qui permet l'affichage dans le bon ordre dans la vue.



# Partie front-end: CP4 – Réaliser une interface utilisateur avec une solution de gestion de contenu ou e-commerce

## I – Les fonctionnalités de gestion de contenu

La gestion du contenu est faite par les utilisateurs du site. Pour cela j'ai mis en place de nombreux C.R.U.D qui permettent aux utilisateurs de créer leurs manuscrits.



Pour l'ajout des manuscrits sur son espace personnel chaque utilisateur a accès à des outils pour créer ou modifier des manuscrits.



S'ensuit une fenêtre avec un formulaire pour donner les informations générales concernant l'œuvre.

The screenshot shows a web form titled "Informations générales" in orange text. In the top right corner, there is a button labeled "Retourner sur mon profil". The form contains several sections: "Titre" with a text input field containing the placeholder "Ajouter un titre"; "Description" with a text input field containing the placeholder "Ajouter une description"; "Genre" with a row of checkboxes for "Horreur", "Aventure", "Science-fiction", "Utopie", "Historique", "Fantastique", "Poésie", "Action", "Nouvelle", and "Classique", followed by a "Contenu explicite" checkbox; "Public visé" with a dropdown menu currently showing "Enfant : 8-13 ans"; and a file selection area with a button "Choisir un fichier" and a text field "Aucun fichier choisi". At the bottom center, there is a blue button labeled "Suivant" which is highlighted with a red border.

Une fois validé cette première étape l'utilisateur arrive sur le formulaire de création de chapitre.

The screenshot shows a web form titled "Ecrire un nouveau chapitre" in orange text. In the top right corner, there is a button labeled "Revenir en arrière". The form contains two main sections: "Titre" with a text input field containing the placeholder "Ajouter un titre"; and a large text area for the chapter content with the placeholder "Écrivez votre histoire...". At the bottom center, there is a blue button labeled "Sauvegarder" which is highlighted with a red border.

La possibilité également de créer des listes de lecture avec le bouton présent sur la page d'informations générales du manuscrit.

Sélectionner un chapitre ▾Ajouter à ma liste de lecture

livre 1

Intrigue : Nouvel utilisateur test

Lectorat visé : Adulte : 18 ans et plus

Genres : Horreur Science-fiction

Ce manuscrit peut contenir des scènes choquantes de tout ordre et ne doit pas être lu par un public mineur ou sensible

Commencer ma lecture

Ce qui donne cela pour la vue permettant l'accès au livre ajouter


Mes manuscritsMa liste de lectureMon profil

Liste des manuscrits


test pour voir si tu es bie

livre 1

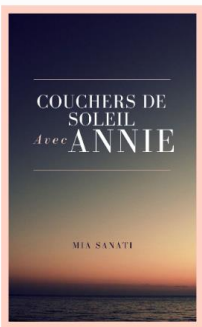
livre 2



Lire



Lire



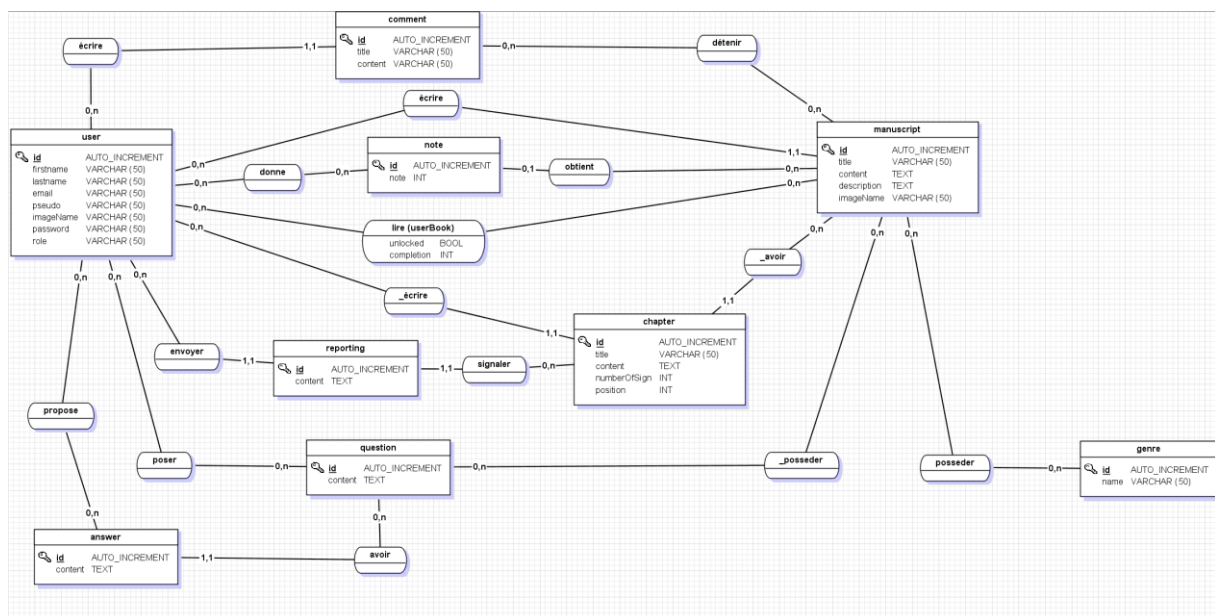
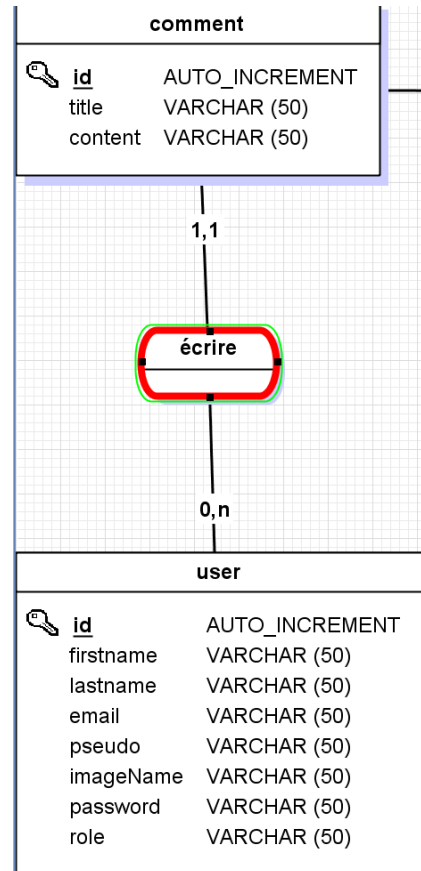
Lire

## Partie back-end : CP5 – Créer une base de données

Dans un premier temps il a fallu organiser le modèle logique de données selon la méthodologie MERISE. Pour ce faire, j'ai utilisé le logiciel JMerise qui permet facilement de créer des entités représentant les tables qui structureront ma base de données. Les relations sont faites à travers des associations qui prennent en titre un verbe, ceci afin de visualiser les liens entre les différentes entités (futures tables).

Enfin les cardinalités permettent de définir la nature du lien entre les entités, ManyToOne, OneToOne, etc... Par exemple combien de commentaire un utilisateur pourrait-il écrire ? En cardinalités cela nous donnerait : 0,n

Ou encore Combien d'utilisateur peuvent écrire un même commentaire? En cardinalités cela nous donnerait : 1,1



Une fois ce modèle logique créé j'ai pu commencer mon projet et créer, via Symfony, ma base données grâce à la commande `symfony console doctrine:database:create`. J'ai configuré mon fichier `.env` qui permet d'ajouter l'URL de la base de données, cela permet à l'ORM Doctrine de connaître l'endroit dans lequel il doit créer sa base de données.

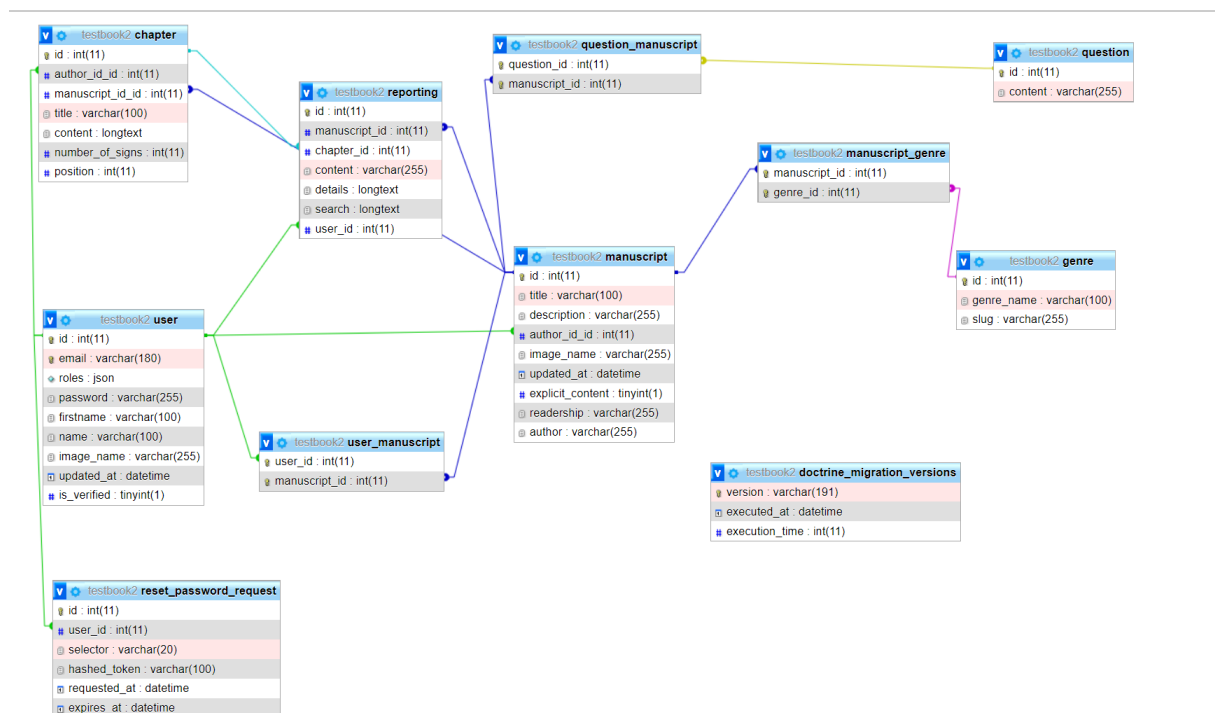
```
PS C:\wamp64\www\tellrz> symfony console doctrine:database:create
Created database `tellrz` for connection named default
PS C:\wamp64\www\tellrz> |
```

Voici une première étape avec un modèle physique de données, une fois les composants d'accès aux données créés. Dans ce schéma on peut voir les relations de type ManyToMany avec les tables de relations user\_manuscript et question\_manuscript.

La table user est reliée à trois autres tables dans des relations OneToMany (chapter, manuscript et reporting), et une relation de type ManyToMany.

La table manuscript est reliée à deux tables dans une relation de type OneToMany (chapter et reporting) et trois relations de type ManyToMany (user, question, genres).

La table reporting est reliée à trois tables dans une relation de type ManytoOne (chapter, manuscript et user)



## Relations dans l'entité Reporting :

```
// Reporting est propriétaire de la relation
#[ORM\ManyToOne(targetEntity: Manuscript::class, inversedBy: 'reportings')]
private $manuscript;

// Reporting est propriétaire de la relation
#[ORM\ManyToOne(targetEntity: Chapter::class, inversedBy: 'reportings')]
private $chapter;

#[ORM\ManyToOne(targetEntity: User::class, inversedBy: 'reportings')]
private $user;
```

## Relations dans l'entité User :

```
// User est propriétaire de la relation avec les manuscrits
#[ORM\ManyToOne(targetEntity: Manuscript::class, inversedBy: 'users')]
private $manuscript;

#[ORM\OneToMany(mappedBy: 'authorId', targetEntity: Manuscript::class, cascade: ["persist", "remove"])]
private $authorManuscript;

#[ORM\OneToMany(mappedBy: 'authorId', targetEntity: Chapter::class)]
private $chapter;

#[ORM\OneToMany(mappedBy: 'author', targetEntity: Reporting::class)]
private $reportings;
```

Relations dans l'entité Manuscript :

```
// User est propriétaire de la relation
#[ORM\ManyToMany(targetEntity: User::class, mappedBy: 'manuscript')]
private $users;

// Question est propriétaire de la relation
#[ORM\ManyToMany(targetEntity: Question::class, mappedBy: 'manuscripts')]
private $questions;

#[ORM\OneToMany(mappedBy: 'manuscriptId', targetEntity: Chapter::class)]
private $chapters;

//Manuscript est propriétaire de la relation
#[ORM\ManyToOne(targetEntity: User::class, inversedBy: 'manuscripts' )]
private $author_id;

// Manuscript est propriétaire de la relation
#[ORM\ManyToMany(targetEntity: Genre::class, inversedBy: 'manuscripts')]
private $genres;

//Reporting est propriétaire de la relation
#[ORM\OneToMany(mappedBy: 'manuscript', targetEntity: Reporting::class)]
private $reportings;
```

# Partie back-end: CP6 – Développer les composants d'accès aux données

## I – Création des Entity et repository

Afin de développer les composants d'accès aux données, j'ai utilisé le système d'entité propre à Symfony (entity). J'ai modélisé mes entités grâce à la commande « symfony console make:entity ». Cette commande permet de créer une classe du nom que l'on souhaite qui va schématiser la représentation d'une table.

Symfony propose alors d'ajouter des propriétés à notre Entity, il faut ensuite préciser le type et la longueur du champ. Une autre entité est créée et est disponible dans le dossier des entités. Ce fichier contient les propriétés déclarées ainsi que leurs getters/setters et les méthodes obligatoires à implémenter.

Pour permettre le mapping et la gestion des ArrayCollection, j'ai utilisé l'ORM (Object-relationnal mapper) Doctrine. De plus, cet ORM permet de construire rapidement des requêtes SQL (Structured Query Language) en remplissant un formulaire de question. Puis, il permet également d'accéder aux données via les repository.

Visuel de la commande « symfony console make:entity » :

```
PS C:\wamp64\www\tellrz> symfony console make:entity

Class name of the entity to create or update (e.g. GentleKangaroo):
> Manuscript
created: src/Entity/Manuscript.php
created: src/Repository/ManuscriptRepository.php
Entity generated! Now let's add some fields!
You can always add more fields later manually or by re-running this command.

New property name (press <return> to stop adding fields):
> title

Field type (enter ? to see all types) [string]:
>

Field length [255]:
>

Can this field be null in the database (nullable) (yes/no) [no]:
> no

updated: src/Entity/Manuscript.php

Add another property? Enter the property name (or press <return> to stop adding fields):
> description

Field type (enter ? to see all types) [string]:
>

Field length [255]:
>

Can this field be null in the database (nullable) (yes/no) [no]:
> yes

updated: src/Entity/Manuscript.php

Add another property? Enter the property name (or press <return> to stop adding fields):
> explicitContent

Field type (enter ? to see all types) [string]:
> bool

Main types
```

Visuels du résultat de la commande :

```
<?php

namespace App\Entity;

use App\Repository\ManuscriptRepository;
use Doctrine\ORM\Mapping as ORM;

#[ORM\Entity(repositoryClass: ManuscriptRepository::class)]
class Manuscript
{
    #[ORM\Id]
    #[ORM\GeneratedValue]
    #[ORM\Column(type: 'integer')]
    private $id;

    #[ORM\Column(type: 'string', length: 255)]
    private $title;

    #[ORM\Column(type: 'string', length: 255, nullable: true)]
    private $description;

    #[ORM\Column(type: 'boolean')]
    private $explicitContent;

    public function getId(): ?int
    {
        return $this->id;
    }

    public function getTitle(): ?string
    {
        return $this->title;
    }

    public function setTitle(string $title): self
    {
        $this->title = $title;

        return $this;
    }
}
```

Lors de la création de l'entité est également créé le fichier entityNameRepository qui permet de faire des requêtes sur les données récupérées dans les Controller.

```
<?php

namespace App\Repository;

use App\Entity\Manuscript;
use Doctrine\Bundle\DoctrineBundle\Repository\ServiceEntityRepository;
use Doctrine\Persistence\ManagerRegistry;

/**
 * @method Manuscript|null find($id, $lockMode = null, $lockVersion = null)
 * @method Manuscript|null findOneBy(array $criteria, array $orderBy = null)
 * @method Manuscript[]    findAll()
 * @method Manuscript[]    findBy(array $criteria, array $orderBy = null, $limit = null, $offset = null)
 */
class ManuscriptRepository extends ServiceEntityRepository
{
    public function __construct(ManagerRegistry $registry)
    {
        parent::__construct($registry, Manuscript::class);
    }

    /**
     * @return Manuscript[] Returns an array of Manuscript objects
     */
    public function findByExampleField($value)
    {
        return $this->createQueryBuilder('m')
            ->andWhere('m.exampleField = :val')
            ->setParameter('val', $value)
            ->orderBy('m.id', 'ASC')
            ->setMaxResults(10)
            ->getQuery()
            ->getResult();
    }
}
```

La seconde étape après la création de l'entité est d'exécuter la commande suivante : « symfony console make:migration » dans le terminal, celle-ci crée un objet contenant la requête SQL pour injecter les tables dans notre base de données.



## Visuel d'un fichier de migration pour l'entité Manuscript.php

```
1 <?php
2
3 declare(strict_types=1);
4
5 namespace Doctrine\Migrations;
6
7 use Doctrine\DBAL\Schema\Schema;
8 use Doctrine\Migrations\AbstractMigration;
9
10 /**
11  * Auto-generated Migration: Please modify to your needs!
12  */
13 final class Version20220222212647 extends AbstractMigration
14 {
15     public function getDescription(): string
16     {
17         return '';
18     }
19
20     public function up(Schema $schema): void
21     {
22         // this up() migration is auto-generated, please modify it to your needs
23         $this->addSql('CREATE TABLE manuscript (id INT AUTO_INCREMENT NOT NULL, title VARCHAR(255) NOT NULL, description VARCHAR(255) DEFAULT NULL, explicit_content TINYINT(1) NOT NULL, PRIMARY KEY(id)) DEFAULT CHARACTER SET utf8mb4 COLLATE `utf8mb4_unicode_ci` ENGINE = InnoDB');
24     }
25
26     public function down(Schema $schema): void
27     {
28         // this down() migration is auto-generated, please modify it to your needs
29         $this->addSql('DROP TABLE manuscript');
30     }
31 }
32 }
```

Afin de permettre à Doctrine de mapper notre base de données on exécute la commande « symfony console doctrine:migrations:migrate ».

## II – Création des C.R.U.D

L'acronyme CRUD, signifie Create, Read, Update, Delete cela définit l'ensemble des procédures qu'il est possible de faire sur une donnée. Bien entendu, toutes les procédures ne sont pas en accès libre et dépendent des rôles des utilisateurs.

Afin de permettre aux utilisateurs de créer un manuscrit, j'ai utilisé la commande « symfony console make:crud », cela permet de créer un controller avec toutes les méthodes permettant l'ajout, la modification, la lecture et la suppression des données.

```
PS C:\wamp64\www\tellrz> symfony console make:crud

The class name of the entity to create CRUD (e.g. DeliciousPopsicle):
> Manuscript

Choose a name for your controller class (e.g. ManuscriptController) [ManuscriptController]:
> EditionTools

created: src/Controller/EditionToolsController.php
created: src/Form/ManuscriptType.php
created: templates/edition_tools/_delete_form.html.twig
created: templates/edition_tools/_form.html.twig
created: templates/edition_tools/edit.html.twig
created: templates/edition_tools/index.html.twig
created: templates/edition_tools/new.html.twig
created: templates/edition_tools/show.html.twig

Success!

Next: Check your new CRUD by going to /edition/tools/
```

Ce controller est nommé EditionTools.php

Explication de la méthode new() :

```
25     #[Route('/new', name: 'edition_tools_new', methods: ['GET', 'POST'])]
26     public function new(Request $request, EntityManagerInterface $entityManager): Response
27     {
28         $manuscript = new Manuscript();
29         $form = $this->createForm(ManuscriptType::class, $manuscript);
30         $form->handleRequest($request);
31
32         if ($form->isSubmitted() && $form->isValid()) {
33             $entityManager->persist($manuscript);
34             $entityManager->flush();
35
36             // Passer le nouveau manuscrit dans la route pour accéder à l'étape de création des chapitres
37             // avec le manuscrit nouvellement créé.
38             return $this->redirectToRoute('chapter_new', [ 'manuscriptId' => $manuscript->getId(),
39             ], Response::HTTP_SEE_OTHER);
40         }
41
42         return $this->renderForm('edition_tools/new.html.twig', [
43             'manuscript' => $manuscript,
44             'form' => $form,
45         ]);
46     }
```

Les routes sont définies en attribut (php8), l'URI de la route est /new. L'option name permet de faire référence à cette méthode lors d'une redirection de route par exemple.

La méthode new prend donc en argument plusieurs injection de dépendance : la requête passée par l'utilisateur, dans notre cas, en POST (ici définie par l'option methods) et un gestionnaire d'entité pour assurer la persistance des données et la mise en base de données des data envoyées par l'utilisateur à travers la requête.

Avant de persister les données on vérifie qu'elles sont valides et bien soumises par l'utilisateur avec la condition If et les méthodes passées en arguments.

La méthode persist permet de rendre à l'état persistant l'entité passée en argument, celle-ci sera ensuite correctement synchronisée avec la base de données lorsque la méthode flush() sera invoquée.

Une fois le flush effectué, l'utilisateur sera redirigé vers l'étape suivante de création des chapitres, cela en récupérant l'id du manuscrit avec la méthode getId de l'entité Manuscript.

La persistance réfère au mécanisme responsable de la sauvegarde et de la restauration des données.

Doctrine applique une stratégie appelée écriture différée transactionnelle, ce qui signifie qu'elle retardera la plupart des commandes SQL jusqu'à ce que la méthode flush() soit invoquée.

### III - Création d'une requête DQL

Pour permettre à l'admin de filtrer ses recherches il a fallu créer un système de filtre. J'ai donc utilisé le système de query builder propre à Doctrine.

```
public function findAuthorBy($search)
{
    $query = $this
        ->createQueryBuilder('r')
        ->join('r.manuscript', 'm')
        ->join('m.author_id', 'u')
        ->andWhere('u.email = :val')
        ->orWhere('u.name = :val')
        ->setParameter('val', $search->authors)
        ;
    return $query->getQuery()->getResult();
}
```

Je vais donc décortiquer la construction de la requête DQL en requête SQL.

Par exemple, pour permettre à l'admin de trouver tous les signalements faits sur un auteur. L'explication de la requête se fera du général vers le particulier.

Tout d'abord on va rechercher tous les manuscrits qui ont un signalement avec la requête SQL suivante :

```
SELECT * FROM `reporting` as r INNER JOIN manuscript as m ON r.manuscript_id = m.id
```

Puis après cette première jointure une seconde qui va permettre de chercher tous les auteurs des manuscrits :

```
SELECT * FROM `reporting` as r INNER JOIN manuscript as m ON r.manuscript_id = m.id  
INNER JOIN `user` u ON m.author_id_id = u.id
```

Enfin, l'ajout de la clause WHERE qui permet de sélectionner par l'email ou le nom en fonction de ce qui est envoyé par l'admin. :

```
SELECT * FROM `reporting` as r INNER JOIN manuscript as m ON r.manuscript_id = m.id  
INNER JOIN `user` u ON m.author_id_id = u.id WHERE u.email = :val OR u.name = :val
```

### IV – Gestion du changement de mot de passe

Mise en place d'un système de changement de mot de passe :

Pour permettre à l'utilisateur de réinitialiser son mot de passe en cas de perte, j'ai installé le bundle symfonycasts/reset-password-bundle avec la commande composer require.

```

PS C:\wamp64\www\testBook2-2> composer require symfonycasts/reset-password-bundle
Info from https://repo.packagist.org: #StandWithUkraine
Using version ^1.13 for symfonycasts/reset-password-bundle
./composer.json has been updated
Running composer update symfonycasts/reset-password-bundle
Loading composer repositories with package information
Updating dependencies
Lock file operations: 1 install, 0 updates, 0 removals
  - Locking symfonycasts/reset-password-bundle (v1.13.0)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 1 install, 0 updates, 0 removals
  - Installing symfonycasts/reset-password-bundle (v1.13.0): Extracting archive
Generating optimized autoload files
110 packages you are using are looking for funding.
Use the `composer fund` command to find out more!

Symfony operations: 1 recipe (4d6a000af329070d3b1c3a62d853033a)
  - Configuring symfonycasts/reset-password-bundle (>=1.0): From github.com/symfony/recipes:main
Executing script cache:clear [OK]
Executing script assets:install public [OK]

What's next?

Some files have been created and/or updated to configure your new packages.
Please review, edit and commit them: these files are yours.

```

Ensuite, afin de générer les template, les controller, les entitie et les repository permettant la gestion de changement de mot de passe j'ai utilisé la commande symfony console make :reset-password.

```

Let's make a password reset feature!
=====

Implementing reset password for App\Entity\User

- ResetPasswordController -
-----

A named route is used for redirecting after a successful reset. Even a route that does not exist yet can be used here.

What route should users be redirected to after their password has been successfully reset? [app_home]:
>

- Email -
-----

These are used to generate the email code. Don't worry, you can change them in the code later!

What email address will be used to send reset confirmations? e.g. mailer@your-domain.com:
> admin@tellrz.com

What "name" should be associated with that email address? e.g. "Acme Mail Bot":
> Jean Dupont

created: src/Controller/ResetPasswordController.php
created: src/Entity/ResetPasswordRequest.php
updated: src/Entity/ResetPasswordRequest.php
created: src/Repository/ResetPasswordRequestRepository.php
updated: src/Repository/ResetPasswordRequestRepository.php
updated: config/packages/reset_password.yaml
created: src/Form/ResetPasswordRequestFormType.php
created: src/Form/ChangePasswordFormType.php
created: templates/reset_password/check_email.html.twig
created: templates/reset_password/email.html.twig
created: templates/reset_password/request.html.twig
created: templates/reset_password/reset.html.twig

Success!

```

# TELLRZ

## Connectez-vous

Email

test@test.com

Mot de passe

.....

[Mot de passe oublié ?](#)

Se connecter

Ajout au niveau de la vue pour permettre à l'utilisateur de changer de mot de passe en cas d'oubli.

## Renouveler votre mot de passe

Email

Entrez votre adresse mail, un lien pour modifier votre mot de passe vous sera envoyé.

Envoyer le lien

L'utilisateur envoie son mail et obtient dans sa boîte mail un lien de reconfiguration de son mot de passe.

# Partie back-end : CP7 – Développer la partie back-end d'une application web ou web mobile

## I – Gestion des attaques de types CSRF :

Afin d'éviter une attaque de type Cross-Site Request Forgery (CSRF), dans le fichier login.html.twig, la fonction csrf\_token permet de générer un jeton unique pour identifier la requête, celle-ci doit donc posséder ce token en session.

```
<input type="hidden" name="_csrf_token" value="{{ csrf_token('authenticate') }}">
```

Le fichier AppAuthenticator.php avec la méthode authenticate a pour but d'intercepter la requête afin de définir un passeport avec les informations prises dans celle-ci. Il est enregistré au niveau du security.yaml

```
public function authenticate(Request $request): Passport
{
    $email = $request->request->get('email', '');

    $request->getSession()->set(Security::LAST_USERNAME, $email);
    dd($request);
    return new Passport(
        new UserBadge($email),
        new PasswordCredentials($request->request->get('password', '')),
        [
            new CsrfTokenBadge('authenticate', $request->request->get('_csrf_token')),
        ]
    );
}
```

```
main:
    lazy: true
    provider: app_user_provider
    custom_authenticator: App\Security\AppAuthenticator
    logout:
```

```
AppAuthenticator.php on line 36:
Symfony\Component\HttpFoundation\Request {#50 ▾
  +attributes: Symfony\Component\HttpFoundation\ParameterBag {#95 ▶}
  +request: Symfony\Component\HttpFoundation\InputBag {#100 ▾
    #parameters: array:3 [▾
      "email" => "utilisateur@utilisateur.com"
      "password" => "test123"
      "_csrf_token" => "6868a9be60305f.LbavJH3C1"
    ]
  }
}
```

Le token est stocké dans un objet de type Request avec l'entrée parameters.

## II – Hachage des données sensibles

```
public function register(Request $request, UserPasswordHasherInterface $userPasswordHasher,
{
    $user = new User();
    $form = $this->createForm(RegistrationFormType::class, $user);
    //permet d'hydrater le formulaire se trouvant dans la requête
    $form->handleRequest($request);

    if ($form->isSubmitted() && $form->isValid()) {
        // encode the plain password
        $user->setRoles(["ROLE_USER"]);
        $user->setPassword(
            $userPasswordHasher->hashPassword(
                $user,
                $form->get('plainPassword')->getData()
            )
        );

        $entityManager->persist($user);
        $entityManager->flush();
    }
}
```

Utilisation du hachage de mot de passe lors de la création d'un utilisateur à l'aide de l'interface UserPasswordHasherInterface.

```
password
$2y$13$/vfgwSZILcqLaJ0Zq1UTW.6bHno4rkKa2eKrE7wICbB...
$2y$13$FZSeTuxb0B5E92FVHPCN.uXBGlqwWa/N2rgOqARKIDA...
pass123
```

\$2y fait référence à la fonction de hachage php : bcrypt.

Le hachage, est une fonction qui, à partir d'une donnée fournie en entrée, calcule une empreinte numérique.

L'empreinte numérique est une suite de caractères retourner par la fonction de taille limitée et souvent fixe.

Ainsi, le hachage d'un fichier texte de 300 pages ou bien d'un simple caractère, pourra, par exemple, retourner une suite de 32 caractères. La longueur des données passées en entrée amenées à être hachées n'influence aucunement la valeur retournée par la fonction de hachage. La longueur de la valeur retournée par la fonction de hachage dépend de la fonction elle-même.

Cependant, la valeur retournée sera totalement différente si un seul caractère du fichier est modifié et ce quel qu'en soit la taille. Il s'agit donc de créer une empreinte qui correspond au fichier. Une fois haché, il est impossible de créer une fonction permettant de récupérer le texte original.

Toutefois, en admettant que l'on connaisse la fonction ayant servie à hacher un mot de passe, il est possible de créer une table qui d'un côté stocke un texte et de l'autre côté stocke son hachage. On stocke ainsi plusieurs millions voir plusieurs milliards de possibilités. Il suffit ensuite de rechercher dans la table, le hachage d'un mot de passe pour vérifier s'il se trouve dans la table et s'il existe une correspondance, alors on récupère le texte associé et on a ainsi trouvé le mot de passe.

Pour contrer ce genre d'attaque il existe ce qu'on appelle le salage. Le salage consiste à ajouter un texte au mot de passe avant de le hacher et rend ce genre d'attaque plus difficile à opérer.



### III – Sécurisation des routes

L'accès aux pages est géré selon les rôles de l'utilisateur dans le fichier security.yaml :

```
access_control:
  - { path: ^/admin, roles: ROLE_ADMIN }
  - { path: ^/edition/tools, roles: ROLE_USER }
  - { path: ^/profile, roles: ROLE_USER }
```

L'utilisateur non-connecté essayant de se connecter via l'url à ces pages sera systématiquement redirigé vers la page de login.

### IV – Gestion des attaques de force-brute

```
PS C:\wamp64\www\testBook2> composer require symfony/rate-limiter
./composer.json has been updated
Running composer update symfony/rate-limiter
Loading composer repositories with package information
Restricting packages listed in "symfony/symfony" to "6.0.*"
Updating dependencies
Lock file operations: 2 installs, 0 updates, 0 removals
  - Locking symfony/lock (v6.0.5)
  - Locking symfony/rate-limiter (v6.0.3)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 2 installs, 0 updates, 0 removals
  - Downloading symfony/lock (v6.0.5)
  - Downloading symfony/rate-limiter (v6.0.3)
  - Installing symfony/lock (v6.0.5): Extracting archive
  - Installing symfony/rate-limiter (v6.0.3): Extracting archive
Generating optimized autoload files
```

J'ai installé le bundle rate-limiter via la commande `composer require symfony/rate-limiter` pour empêcher les attaques par force brute (brute-force attack).

Grâce à ce code mis dans le fichier security.yaml il est possible de bloquer les attaques par force brute en définissant un goulot d'étranglement au fur et à mesure des tentatives de connexions infructueuses.

'login\_throttling' prendre donc en argument un tableau, avec en clé 'max\_attempts' qui permet de définir le nombre de tentatives auquel l'utilisateur à droit pour se connecter, dans notre cas trois ; puis la clé 'interval' représente l'intervalle de temps écoulé après lequel l'utilisateur pourra retenter de se connecter, ici trente minutes.

```
login_throttling:
  max_attempts: 3
  interval: '30 minutes'
```

# Partie back-end : CP8 – Elaborer et mettre en œuvre des composants dans une application de gestion de contenu ou e-commerce

## I - Gestion du réagencement de chapitre

Afin de permettre la gestion du drop la fonction `handleDrop` envoie un tableau avec le bon réagencement des chapitres. En effet dans le controller les chapitres sont récupérés par ordre de position.

```
178 function handleDrop(itemEvent) {
179     console.log(itemEvent)
180
181     // On récupère le nouveau tableau au moment du drop
182     let sortedChaptersArray = itemEvent.target.parentNode.children;
183     const arrayOfPositionAndId = []
184     // A chaque fois que l'objet chapter est itéré dans la boucle on récupère son attribut data-id
185     // et on l'initialise son index dans le nouveau tableau avec la variable i dans le tableau arrayOfPositionAndId
186     let i = 0
187     for(let chapter of sortedChaptersArray) {
188         arrayOfPositionAndId[i] = chapter.getAttribute('data-id')
189         i++
190     }
191
192     // Encodage de l'array pour le passer dans une requête en ajax
193     const jsonArray = JSON.stringify(Object.assign({}, arrayOfPositionAndId))
194     $.ajax({
195         url: '/chapters/reorder',
196         method: 'POST',
197
198         data: jsonArray
199     })
200     .fail(function(err) {
201         alert("Une erreur est survenue veuillez rafraîchir la page et recommencer.")
202     })
203 }
204
```

Il s'agit donc de récupérer la liste de nœud triée qui donne le nouvel ordre des chapitres (`sortedChaptersArray`). Ensuite, on crée un tableau vide qui va permettre de ranger les id et les position (`arrayOfPositionAndId`), puis on initialise une variable `i` à zéro pour l'utiliser dans la boucle qui va suivre.

La boucle parcourt chaque élément du tableau `sortedChaptersArray` et on insère dans le tableau `arrayOfPositionAndId` en réinitialisant la variable `i` en tant que clé de chaque valeur du nouveau tableau, cette dernière sera égale à l'id d'un chapitre, et ceux à chaque boucle du tableau.

Enfin, pour effectuer la requête AJAX le tableau `arrayOfPositionAndId` doit être transformé en JSON object via la méthode `stringify` de l'objet JSON. Lors de cette dernière, on appelle l'url du controller qui va traiter le tableau passé en `data`.

```

#[Route('/chapters/reorder', name: 'chapters_draggable')]
public function sortAction(Request $request, ChapterRepository $chapterRepository, EntityManagerInterface $entityManager)
{
    // Récupération et décodage du json en tableau php
    $positionArray = json_decode($request->getContent(), true);

    // Réinitialisation des données pour chaque chapitre,
    // en récupérant celui-ci via la $value et en paramétrant la position avec la $key
    foreach ($positionArray as $key => $value) {
        $chapter = $chapterRepository->findOneById($value);
        $chapter->setPosition($key);
    }

    $entityManager->persist($chapter);
    $entityManager->flush();

    return new Response(
        Response::HTTP_OK
    );
}

```

Le contrôleur récupère le tableau via la méthode `sortAction` récupère la requête et le met sous forme de tableau PHP.

Ensuite pour chaque élément du tableau on récupère le chapitre via la variable `$value` et on paramètre la position avec la variable `$key`. On utilise dans le corps de la boucle le `ChapterRepository` et sa méthode `findOneById` pour trouver un chapitre et on utilise son setter pour lui donner sa nouvelle position.

Enfin, on persist le chapitre et on l'envoie dans la base de données via le `flush`.

## II – Gestion de la liste de favoris

### A/ Ajout à la liste de favoris

Afin de permettre l'ajout de livre dans la liste de lecture, il faut stocker les livres et les utilisateurs dans la table de relation (`user_manuscript`) créé au moment de l'élaboration de l'entité `user`.

```

<button class="button-favourite btn" type="button" data-id="{{manuscript.id}}">
    Ajouter à ma liste de lecture
</button>

```

`$('.button-favourite')` permet de récupérer le bouton sous forme d'un objet jQuery auquel on adjoint un event listener : `on('click')`, pendant lequel on récupère l'id du manuscrit, puis on appelle la route via une requête AJAX.

```
$('.button-favourite').on('click', () => {  
  let manuscriptId = $('.button-favourite').attr('data-id');  
  $.ajax({  
    url: '/manuscript/' + manuscriptId + '/addToFavourite',  
    method: 'POST',  
  })  
})
```













L'appelle de cette route déclenche la fonction `addToFavourite`.

```
#[Route('/manuscript/{id}/addToFavourite', name: 'manuscript_favourite')]  
public function addToFavourite(int $id, ManuscriptRepository $manuscriptRepository, EntityManagerInterface $entityManagerInterface): Response  
{  
  // ajouter un message on success  
  $manuscript = $manuscriptRepository->findOneBy(['id' => $id]);  
  
  $user = $this->getUser();  
  $manuscript->addUser($user);  
  
  $entityManagerInterface->persist($manuscript);  
  $entityManagerInterface->flush();  
  
  return new Response(  
    Response::HTTP_OK  
  );  
}
```

Cette route prend dans son url l'id et celui-ci va permettre de récupérer le manuscrit et de lui ajouter un user que l'on obtient très simplement grâce à la classe `ManuscriptController` et son mot clé `$this`.

Enfin, pour stocker le manuscrit avec ses nouvelles informations on persist le manuscrit et on le flush.

Ce qui donne le résultat suivant en base de données :

<div><div>←T→</div><div></div></div>					user_id	manuscript_id
<input type="checkbox"/>	 Edit	 Copy	 Delete	39	24	
<input type="checkbox"/>	 Edit	 Copy	 Delete	41	77	
<input type="checkbox"/>	 Edit	 Copy	 Delete	41	104	
<input type="checkbox"/>	 Edit	 Copy	 Delete	41	105	

## B/ Accès aux données

Une fois cette étape passée, l'utilisateur a accès à sa liste, pour afficher celle-ci il a fallu créer une requête DQL qui permet de récupérer tous les manuscrits avec l'id de l'utilisateur.

```
public function findAllManuscriptByUser($id)
{
    return $this->createQueryBuilder('m')
        ->Join('m.users', 'u')
        ->andWhere('u.id = :val')
        ->setParameter('val', $id)
        ->getQuery()
        ->getResult()
    ;
}
```

Ce qui donne la requête DQL suivante :

```
SELECT m FROM App\Entity\Manuscript m INNER JOIN m.users u WHERE u.id = :val
```

Et la requête SQL suivante :

```
SELECT m0_id AS id_0, m0_title AS title_1, m0_description AS description_2,
m0_explicit_content AS explicit_content_3, m0_readership AS readership_4,
m0_image_name AS image_name_5, m0_updated_at AS updated_at_6, m0_author AS
author_7, m0_author_id AS author_id_id_8
```

FROM manuscript m0\_

INNER JOIN user\_manuscript u2\_ ON m0\_id = u2\_manuscript\_id (jointure n°1)

On récupère toutes les informations concernant le manuscrit à partir de la table manuscript en joignant, dans un premier temps la table de relation user\_manuscript pour récupérer tous les manuscrits qui ont une relation.

id_0	title_1	user_id	manuscript_id
24	Fixtures	39	24
77	test pour voir si tu es bien rentré dans la bdd	41	77
104	livre 1	41	104
105	livre 2	41	105

INNER JOIN `user` u1\_ ON u1\_.id = u2\_.user\_id WHERE u1\_.id = :val (ici 41) (jointure n°2)

Dans un second temps on joint la table user pour récupérer les user qui ont une relation avec la table user\_manuscript avec le ON u1\_.id = u2\_.user\_id.

		<div>⏪ ⏴ ⏵ ⏩</div>			▼	user_id	manuscript_id
id_0	title_1	<input type="checkbox"/>					
77	test pour voir si tu es bien rentré dans la bdd	<input type="checkbox"/>				39	24
104	livre 1	<input type="checkbox"/>				41	77
105	livre 2	<input type="checkbox"/>				41	104
		<input type="checkbox"/>				41	105

On se sert de cette requête dans le UserController pour renvoyer les manuscrits à la vue.

```
class UserController extends AbstractController
{
    #[Route('/profile', name: 'profile')]
    public function index(ManuscriptRepository $manuscriptRepository): Response
    {
        // récupérer les livres dont l'id correspond à l'id du user
        $user = $this->getUser();
        // Récupération des identifiants du user
        $user ->getUserIdentifiant();

        // récupérer le manuscrit cliqué
        $favouriteManuscripts = $manuscriptRepository->findAllManuscriptByUser($user);

        // lors de la requête on récupère l'id du user que l'on compare avec celui de l'auteur a
        $manuscripts = $manuscriptRepository->findBy([
            'author_id' => $user,
        ]);

        return $this->render('user/index.html.twig', [
            'manuscripts' => $manuscripts,
            'favouriteManuscripts' => $favouriteManuscripts
        ]);
    }
}
```

Ensuite on itère sur la variable `$favouriteManuscripts` via une boucle dans le fichier twig `user.html.twig` afin de récupérer chaque manuscrit et afficher ces différentes propriétés via la notation pointée.

```
<div class="panel" id="two-panel">
  <div class="panel-title">Liste des manuscrits</div>
  <div class="panel-body">
    {% if favouriteManuscripts|length > 0 %}

      <div class="mt-5 d-flex horizontalScroll">
        {% for favouriteManuscript in favouriteManuscripts %}
          <div class="card-style">
            <div class="card-body d-flex flex-wrap justify-content-center">
              <h5 class="card-title overflow-hidden" alt="{{ favouriteManuscript.title }}">{{ favouriteManuscript.title }}
            </h5>
            
            <a href="{{ path('manuscript', {'id': favouriteManuscript.id}) }}" class="btn btn-sm w-100 align-self-end mt-3">Lire</a>
          </div>
        {% endfor %}
      </div>
    {% endif %}
  </div>
</div>
```

Mise en place de la récupération des chapitres pour permettre d'aller à chaque fois au chapitre suivant. Pour ce faire j'ai créé une méthode `showChapter` qui récupère les tous les chapitres d'un manuscrit et qui récupère le chapitre en cours de lecture.

```
#[Route('/manuscript/{manuscriptId}/chapter/{chapterId}', name: 'manuscript_chapter')]

public function showChapter(int $manuscriptId, int $chapterId, ManuscriptRepository $manuscriptRepository, ChapterRepository $chapterRepository)
{
    $manuscript = $manuscriptRepository->findOneBy(['id' => $manuscriptId]);
    $chapter = $chapterRepository->findOneBy(['id' => $chapterId]);

    //Récupération de tous les chapitres qui ont l'id du manuscrit passé en url
    $chapters = $chapterRepository->findBy(['manuscriptId' => $manuscriptId], ['position' => 'ASC']);

    // trouver dans le tableau le chapitre qui correspond avec l'id du manuscrit passé en url
    $currentChapterIndex = array_search($chapter, $chapters);

    // On vérifie que le prochain chapitre est bien dans le tableau, si oui on lui stocke sa valeur dans la variable
    for($i = 0; $i < count($chapters); $i++) {
        if(isset($chapters[$currentChapterIndex + 1])) {
            $nextChapter = $chapters[$currentChapterIndex + 1];
        } else {
            $nextChapter = null;
        }
    };

    return $this->render('manuscript/manuscript_chapter.html.twig', [
        'chapter' => $chapter,
        'chapters' => $chapters,
        'manuscript' => $manuscript,
        'next_chapter' => $nextChapter,
        'chapter_id' => $chapterId,
        'manuscript_id' => $manuscriptId
    ]);
}
```

Pour trouver un moyen de récupérer l'index du chapitre en cours de lecture j'ai trouvé la méthode `array_search()`.

Exemple de recherche que j'ai effectué tout au long du projet en langue anglaise : `array_search` — Searches the array for a given value and returns the first matching key if successful.

Cette fonction permet de rechercher dans un tableau un élément (ou une valeur) donné et retourne la clé de celui-ci.

Ici, je stocke donc dans la variable `$currentChapterIndex` l'index du chapitre en cours. Puis je parcours à l'aide d'une boucle le tableau des chapitres récupéré dans la variable `$chapters`, dans le corps de cette boucle je vérifie si l'index du prochain chapitre à partir de mon `$currentChapterIndex` existe, si oui j'assigne la valeur `$nextChapter` à la valeur de mon chapitre en cours de lecture +1.

Puis je renvoie mon élément dans ma vue twig sous le nom de `next_chapter` afin de créer un bouton renvoyant vers le chapitre suivant.

```
{% if next_chapter is not null %}

    <a href="{{ path('manuscript_chapter', {'chapterId': next_chapter.id,
'manuscriptId': manuscript.id}) }}" class="btn justify-content-end border-
dropdown">Chapitre suivant</a>

{% else %}

    <p>FIN</p>

{% endif %}
```

Sélectionner un chapitre ▾

chapitre 3 ▾

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent blandit et urna sed dapibus. Fusce placerat faucibus mi sed mattis. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque cursus pretium elit eget blandit. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Sed a metus ex. Nullam et dui blandit, ultrices eros nec, suscipit diam. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aenean dictum nulla urna, non tincidunt quam tempor blandit. Donec eget tellus sit amet ante molestie malesuada interdum et augue. Pellentesque ante metus, vehicula et justo id, egestas ultricies mauris. Aliquam erat volutpat. Sed blandit, libero id mattis venenatis, velit erat aliquet eros, sed semper augue enim ac nisl. Proin ut accumsan risus, quis rutrum mauris. Ut lorem odio, condimentum ut lectus quis, efficitur varius enim. Proin euismod fermentum posuere. Duis dictum mi mauris, id pulvinar magna lacinia sed. Aliquam tempus viverra magna, eu venenatis nulla tempus sed. Sed porta lorem leo, ut tincidunt nulla pellentesque sed. Pellentesque convallis tellus in eros consequat molestie. Maecenas at turpis vehicula, varius massa a, blandit dui. Donec mi lorem, porttitor sed nisi et, vehicula venenatis leo. Aenean vehicula risus quis viverra vestibulum. Quisque sodales est non ligula egestas, non euismod orci ultricies. In venenatis ornare velit, in maximus ante scelerisque tincidunt. Maecenas id velit sagittis, viverra nibh at, lobortis ex. Nam congue, urna sit amet eleifend lacinia, justo erat egestas leo, a rhoncus libero lorem a turpis. Phasellus sagittis ut nunc non ornare. Aenean scelerisque ut felis id vehicula. Praesent commodo nisl in augue aliquet, id imperdiet massa sagittis. In diam lacus, euismod in fringilla in, iaculis sagittis eros. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque sed enim tortor. Proin et iaculis nisl, pulvinar ultrices neque. Nam a justo tincidunt, viverra turpis id, lobortis odio. Integer dictum sollicitudin erat. Quisque magna nunc, varius nec dolor sit amet, feugiat condimentum lectus. Mauris in diam dui. Aliquam congue viverra dui, a ullamcorper leo consequat a. Etiam vulputate nibh sit amet viverra rutrum. Nullam dignissim tempus gravida. Fusce bibendum pellentesque magna a porttitor. Vivamus non diam vitae est tristique convallis. Phasellus sed mauris lacus. Nam imperdiet neque neque, ac aliquam sem aliquet nec. Nunc eget eros egestas, luctus neque sit amet, pretium magna. Etiam eu suscipit risus, sit amet accumsan lorem. Mauris id mollis quam, nec suscipit sem. Phasellus auctor magna nec condimentum varius. Proin pharetra aliquam metus. Duis laoreet pretium tortor, at lobortis dolor egestas vel. Ut pharetra lectus at porttitor porta. Nunc at est ante. Nunc non velit fermentum, congue dolor quis, congue enim.

Signaler

88 Chapitre suivant



## Recherche en anglais

Afin de trouver comment rendre mes entités dans mon type, j'ai recherché dans google comment rendre des entités dans un formulaire en symfony. J'ai effectué la recherche avec cette question : how to render entity in a symfony type.

Je suis tombé sur la documentation officielle de Symfony en première page de résultat.

<https://symfony.com> > ... > Types ▾ [Traduire cette page](#)

### EntityType Field (Symfony Docs)

Rendered as, can be various tags (see [ChoiceType Field](#) (select drop-downs, ... The **entity type** has just one required option: the **entity** which should be ...

A special `ChoiceType` field that's designed to load options from a Doctrine entity. For example, if you have a `Category` entity, you could use this field to display a `select` field of all, or some, of the `Category` objects from the database.

Rendered as	can be various tags (see <a href="#">ChoiceType Field</a> (select drop-downs, radio buttons & checkboxes))
Parent type	<code>ChoiceType</code>
Class	<code>EntityType</code> <a href="#">↗</a>

Un champ spécial `ChoiceType` est conçu pour charger des options à partir d'une entité Doctrine. Par exemple, si vous avez une entité `Catégorie`, vous pouvez utiliser ce champ pour afficher un champ de sélection affichant tous les objets de la base données de l'entité `Category` ou bien, une partie seulement.

## Basic Usage

The `entity` type has just one required option: the entity which should be listed inside the choice field:

```
1 use App\Entity\User;
2 use Symfony\Bridge\Doctrine\Form\Type\EntityType;
3 // ...
4
5 $builder->add('users', EntityType::class, [
6     // looks for choices from this entity
7     'class' => User::class,
8
9     // uses the User.username property as the visible option string
10    'choice_label' => 'username',
11
12    // used to render a select box, check boxes or radios
13    // 'multiple' => true,
14    // 'expanded' => true,
15]);
```

This will build a `select` drop-down containing *all* of the `User` objects in the database. To render radio buttons or checkboxes instead, change the `multiple` and `expanded` options.

Ici, l'usage basique qui spécifie que l'entity type à une seule option requise : l'entité qui va être listée dans le champ de choix.

J'ai ensuite regardé quelles configurations il était possible de faire dans le tableau de configuration de la méthode `add`.

Dans mon cas, j'ai voulu rendre sous forme de case à cocher les genres possibles d'ajouter à un manuscrit lors de la création de ce dernier.

J'ai donc regardé les différentes clés qu'il est possible de passer à ce tableau.

### multiple

type: `boolean` default: `false`

If true, the user will be able to select multiple options (as opposed to choosing just one option). Depending on the value of the `expanded` option, this will render either a select tag or checkboxes if true and a select tag or radio buttons if false. The returned value will be an array.

Si vrai, l'utilisateur sera capable de sélectionner de multiples options (contrairement au fait de choisir une seule option). En fonction de la valeur de l'option `expanded`, cela rendra soit une balise de sélection soit des cases à cocher si elle vraie, soit une balise de sélection ou un bouton radio si elle est fausse. La valeur retournée sera dans un tableau.