

Bibliothèque Musicale

Mars-Mai 2025

Table des matières

Bibliothèque musicale

1	Le contexte	7	Les fonctionnalités
2	Le client		<ul style="list-style-type: none">• Authentification et gestion des utilisateurs• Gestion des morceaux• Gestion des playlists• Administration• Code Symfony (exemples)
3	L'objectif de la mission	8	Conclusion
4	La réalisation	9	Annexe
5	L'architecture technique		
6	Le modèle de données		
	<ul style="list-style-type: none">• Entités principales• Modèle conceptuel des données (MCD)• Diagrammes (cas d'utilisation, séquence, classes, activité, état, composants)• Interface utilisateur et interface administrateur		

01 Le contexte.

La M2L (Maison des Ligues de Lorraine) est un établissement du Conseil Régional de Lorraine, situé dans la banlieue de Nancy.

Elle est administrée par le Comité Régional Olympique et Sportif de Lorraine (CROSL) et chargée de fournir aux ligues sportives de la région Grand Est des moyens matériels, logistiques et techniques.

Sa mission s'articule autour de quatre axes principaux :

- Hébergement et soutien aux ligues sportives : mise à disposition d'espaces pour les comités et ligues régionales, favorisant la collaboration entre structures sportives.
- Formation et accompagnement : organisation régulière de formations pour les bénévoles, dirigeants et salariés associatifs (gestion, communication, financement).
- Appui au développement du sport en Lorraine : participation aux politiques sportives régionales, soutien au développement des disciplines, avec une attention particulière portée au handisport.
- Mise à disposition d'équipements et de services : infrastructures physiques (salles, amphithéâtres), ressources immatérielles (logiciels spécialisés, support technique et informatique).

01 Le contexte.

Dans le cadre d'un nouveau partenariat commercial, la M2L souhaite proposer aux clubs affiliés une application mobile e-commerce intuitive facilitant les achats de matériel sportif.

L'application mobile permettra :

- la consultation du catalogue de produits,
- l'ajout au panier,
- la création de compte,
- la gestion des commandes.



M2L_Logo.png

02 Le client.

À propos

La Maison des Ligues de Lorraine (M2L) joue un rôle central dans le paysage sportif régional. Elle héberge une grande majorité des associations sportives régionales, ainsi que plusieurs comités départementaux, notamment le CROSL (Comité Régional Olympique et Sportif de Lorraine) et le CDOS (Comité Départemental Olympique et Sportif).

Au sein de ses locaux, la M2L met à disposition des structures sportives des espaces de travail fonctionnels (bureaux, salles de réunion, centres de formation), mais également un accès à des services numériques visant à améliorer leur gestion au quotidien. Ces services incluent l'utilisation d'outils informatiques, le support technique, ainsi que des solutions logicielles partagées.

Dans ce contexte, notre projet vise à répondre à une problématique concrète rencontrée par la M2L : la nécessité de mieux organiser la gestion des ressources (matériel, logiciels, services) mises à disposition des ligues et associations hébergées. Grâce à notre solution, la M2L pourra centraliser ces services dans une application mobile moderne, améliorant ainsi la lisibilité, l'accessibilité et l'efficacité de son offre à destination des acteurs sportifs régionaux.

02 Le client.

Pour développer notre bibliothèque musicale, nous utilisons un ensemble de technologies modernes et complémentaires :

PHP

PHP (Hypertext Preprocessor) est un langage de programmation libre, principalement utilisé pour produire des pages Web dynamiques via un serveur web. PHP est un langage impératif orienté objet qui a permis de créer un grand nombre de sites web célèbres. Il est considéré comme une des bases de la création de sites web dynamiques et d'applications web.

HTML

Le HyperText Markup Language (HTML) est le langage de balisage conçu pour écrire les pages web. Nous utilisons la dernière version HTML5 pour structurer le contenu de notre application.

Framework Symfony

Nous avons choisi d'utiliser Symfony, un framework MVC libre écrit en PHP. Il fournit des fonctionnalités modulables et adaptables qui permettent de faciliter et d'accélérer le développement d'un site web, notamment:

- Doctrine: Un ORM (Object-Relational Mapping) de base de données qui est un logiciel libre sous licence GNU LGPL. Cet ORM permet la persistance transparente des objets PHP. C'est l'interface qui permet de faire le lien ou "mapping" entre les objets et les éléments de la base de données.
- Architecture MVC: Le modèle MVC (Model-View-Controller ou Modèle-Vue-Contrôleur) met l'accent sur la séparation entre la logique métier et l'affichage du logiciel. Cette "séparation des préoccupations" permet une meilleure répartition du travail et une maintenance améliorée.

Composer

Composer est un logiciel gestionnaire de dépendances libre écrit en PHP. Il permet à ses utilisateurs de déclarer et d'installer les bibliothèques dont le projet principal a besoin.

SQLite

Selon les informations du PDF, pour le projet de la bibliothèque musicale, nous utilisons SQLite comme base de données.

Twig

Le front-end est développé avec Twig, un moteur de templates pour PHP moderne, flexible et sécurisé.

03 L'objectif de la mission.

Attendus des utilisateurs:

- Une navigation simple et intuitive
- Une interface sécurisée avec authentification
- Une visualisation claire des ressources musicales disponibles
- Des possibilités de filtrage par artiste, album, date de sortie
- Un système de création de playlists personnalisées
- Un accès adapté à tous les appareils

Attendus de la M2L:

- Une interface d'administration complète et intuitive
- Une gestion centralisée des ressources musicales
- Un suivi de l'utilisation des playlists lors des événements
- Un système de droits d'accès permettant de contrôler qui peut modifier les playlists officielles

Fonctionnalités principales:

Gestion des utilisateurs:

Authentification sécurisée

Gestion des profils utilisateurs avec différents niveaux d'accès

Inscription et connexion des utilisateurs

Se connecter

Email

Password

[Je n'ai pas encore de compte.](#) Se connecter

Mon profil

Firstname

CYPRIEN

Name

BROCHE

Email

cyp971@gmail.com

Password

Update

[back to list](#)

03 L'objectif de la mission.

Gestion des morceaux musicaux:

Ajout/Retrait de musique dans la bibliothèque

Catégorisation des morceaux par artiste, album, genre

Recherche et filtrage avancés



The screenshot shows a web interface for a music library. At the top is a red navigation bar with links: "Music Library", "Artistes", "Albums", and "Morceaux". On the right of the bar is a user profile icon labeled "oui". Below the navigation bar is a light blue header area with a back arrow icon and the title "Create new Artist". The form contains three input fields: "Name", "Thumbnail", and "Bio". At the bottom left of the form is a red "Save" button.

Gestion des playlists:

Création et modification de playlists personnalisées

Partage de playlists entre utilisateurs

Organisation des morceaux dans les playlists



The screenshot shows a web interface for a music library. At the top is a red navigation bar with links: "Music Library", "Artistes", "Albums", and "Morceaux". On the right of the bar is a user profile icon labeled "oui". Below the navigation bar is a light blue header area with a back arrow icon and the title "Create new Album". The form contains five input fields: "Title", "Thumbnail", "Release date" (with a date picker icon), "Artist" (a dropdown menu showing "Sköne"), and "Type" (a dropdown menu showing "Album"). At the bottom left of the form is a red "Save" button.

03 L'objectif de la mission.

Administration:

Interface administrateur pour la gestion globale

Statistiques d'utilisation

Gestion des droits d'accès

Modélisation

Architecture de la base de données

On a conçu un modèle de données adapté aux besoins de la bibliothèque musicale. D'après le diagramme de classes présent dans le mcd ci-dessous, notre modèle comprend les entités suivantes:

```
1  User:
2  id: int
3  name: string
4  username: string
5  password: string(255) hash+salt
6
7  Track:
8  id: int
9  name: string
10 thumbnailURL: string
11
12 Artist:
13 id: int
14 name: string
15 thumbnailURL: string
16
17 Release:
18 id: int
19 name: string
20 thumbnailURL: string
21 price: float
22 category: int
```

04 La réalisation

Relations:

Un User peut avoir plusieurs Track (relation 0-n)

Un Track peut être lié à plusieurs Artist (relation 0-n)

Un Artist peut avoir plusieurs Release (relation 1-n)

Un Track peut appartenir à un Release (relation 1-n)

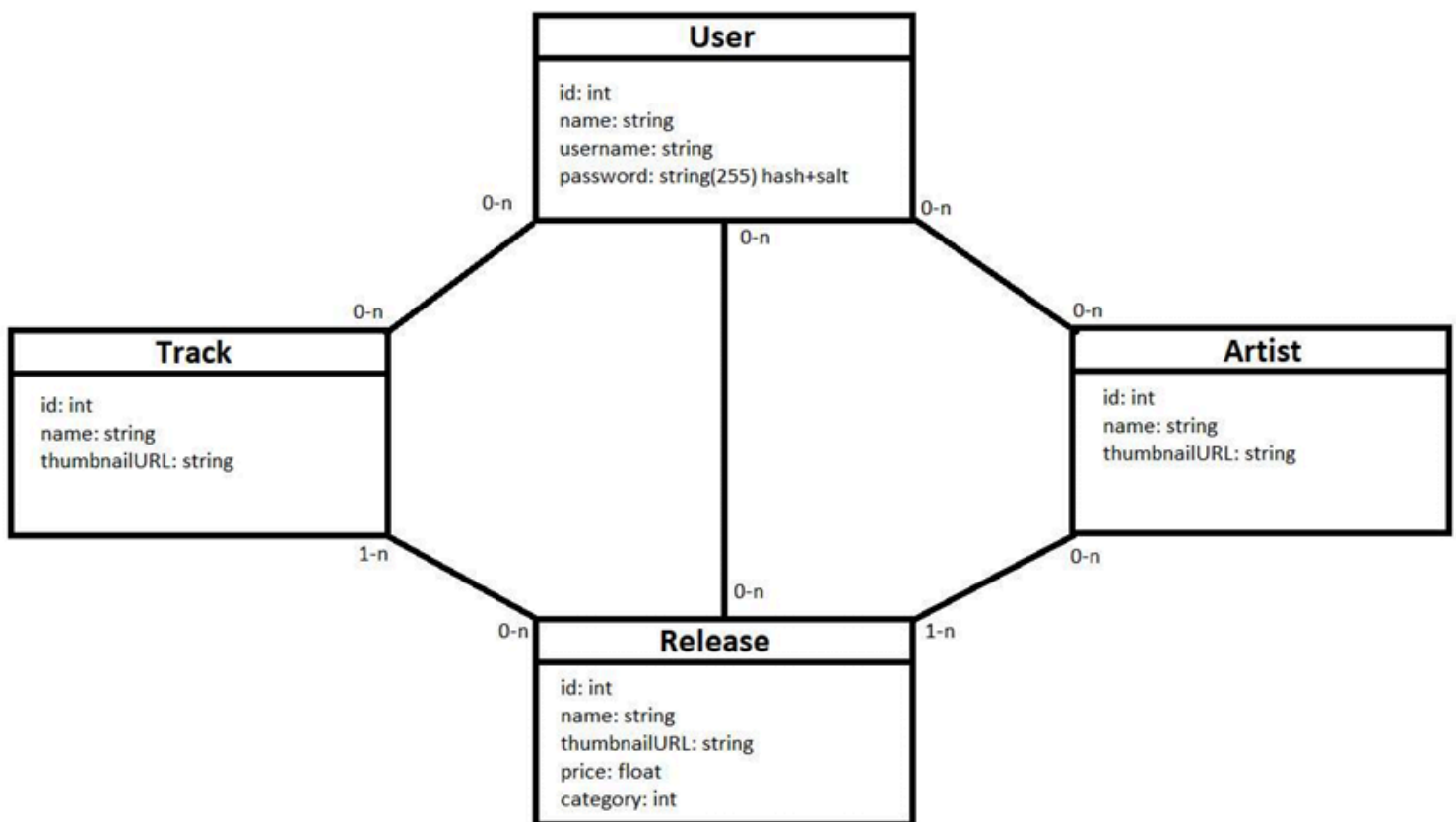
Cette structure permet une organisation efficace des données musicales, facilitant la recherche et la création de playlists.

05 L'Architecture technique

Les entités principales du projet sont :

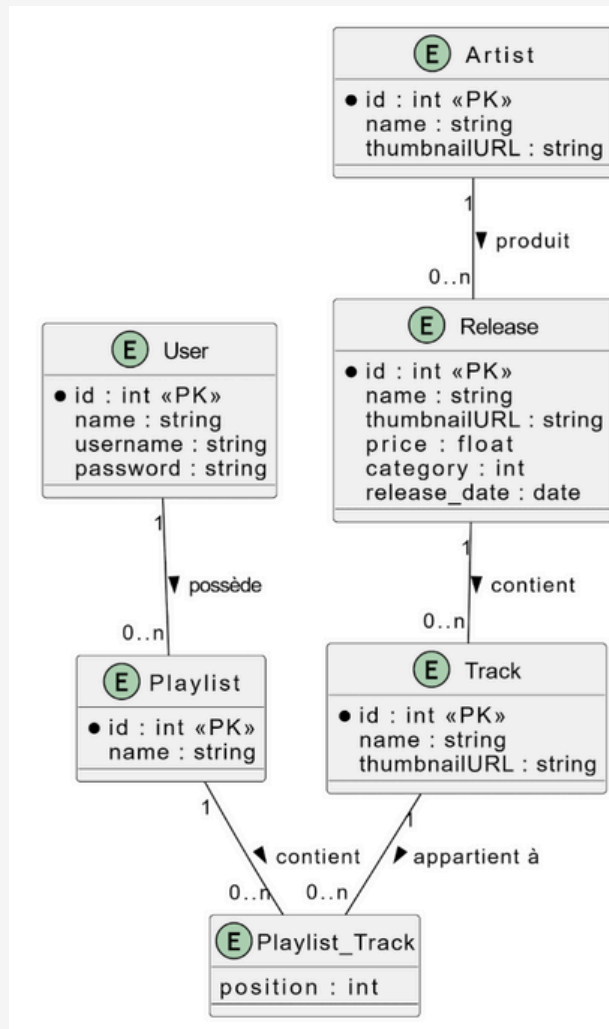
- Artiste : contient le nom, la biographie, le pays d'origine ;
- Musique : titre, date de sortie, durée, genre, lien vers l'album et l'artiste ;
- Album : titre, pochette, date de publication, type (album, EP, single).

Doctrine permet de gérer automatiquement la création des tables, les jointures, et d'assurer la cohérence de la base.



06 Le modèle de données

Modèle Conceptuel des Données (MCD)



Notre application de bibliothèque musicale a été développée en utilisant le framework Symfony pour offrir une solution robuste et évolutive à la M2L. La réalisation s'est déroulée conformément au planning établi et a permis de livrer une application répondant aux besoins exprimés.

L'architecture de l'application suit le modèle MVC propre à Symfony:

Modèle: Entités Doctrine pour la gestion des données

Vue: Templates Twig pour l'affichage

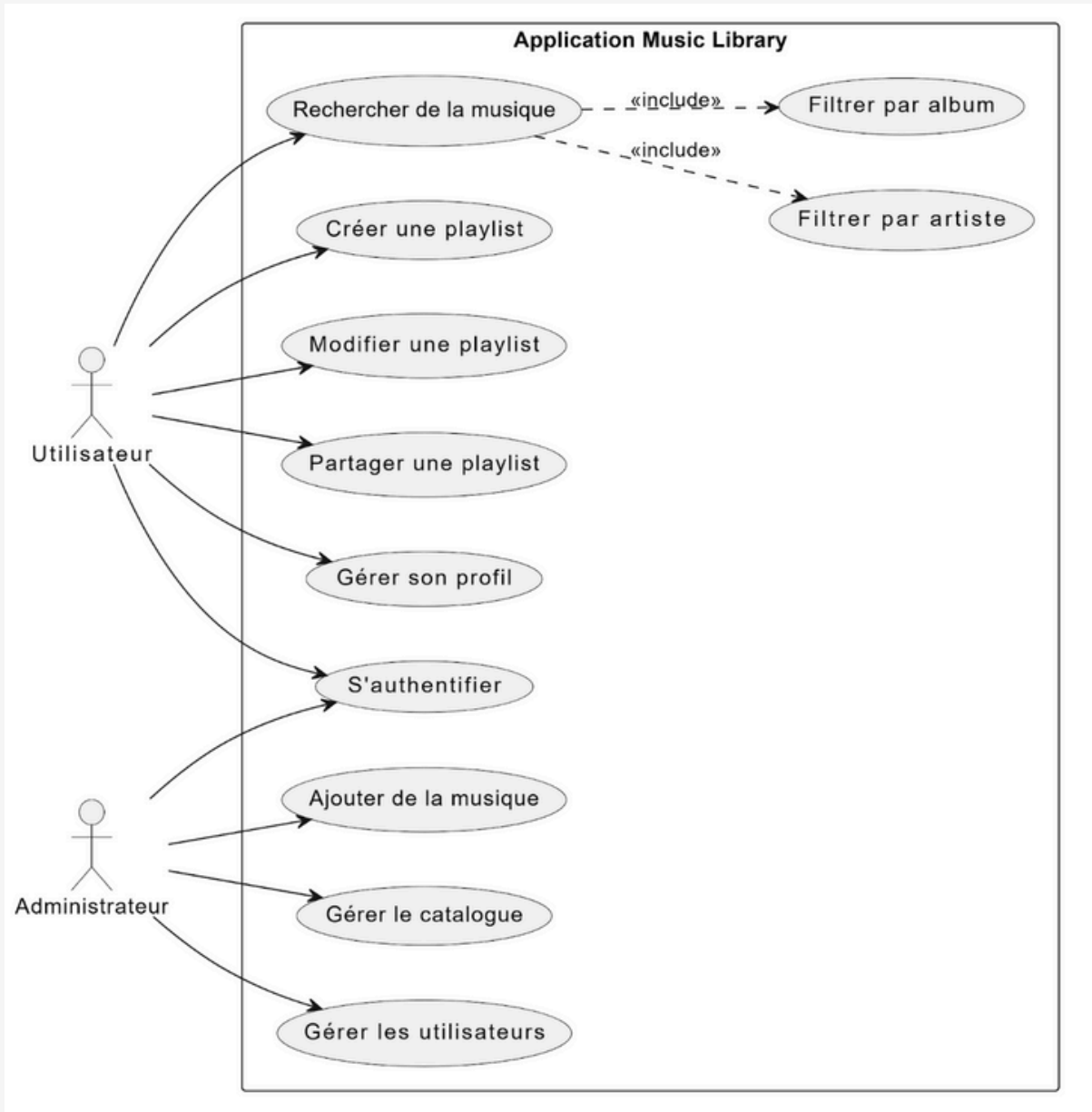
Contrôleur: Classes PHP gérant la logique métier

La sécurité a été une préoccupation majeure tout au long du développement, avec une implémentation robuste de l'authentification et de la gestion des droits.

06 Le modèle de données

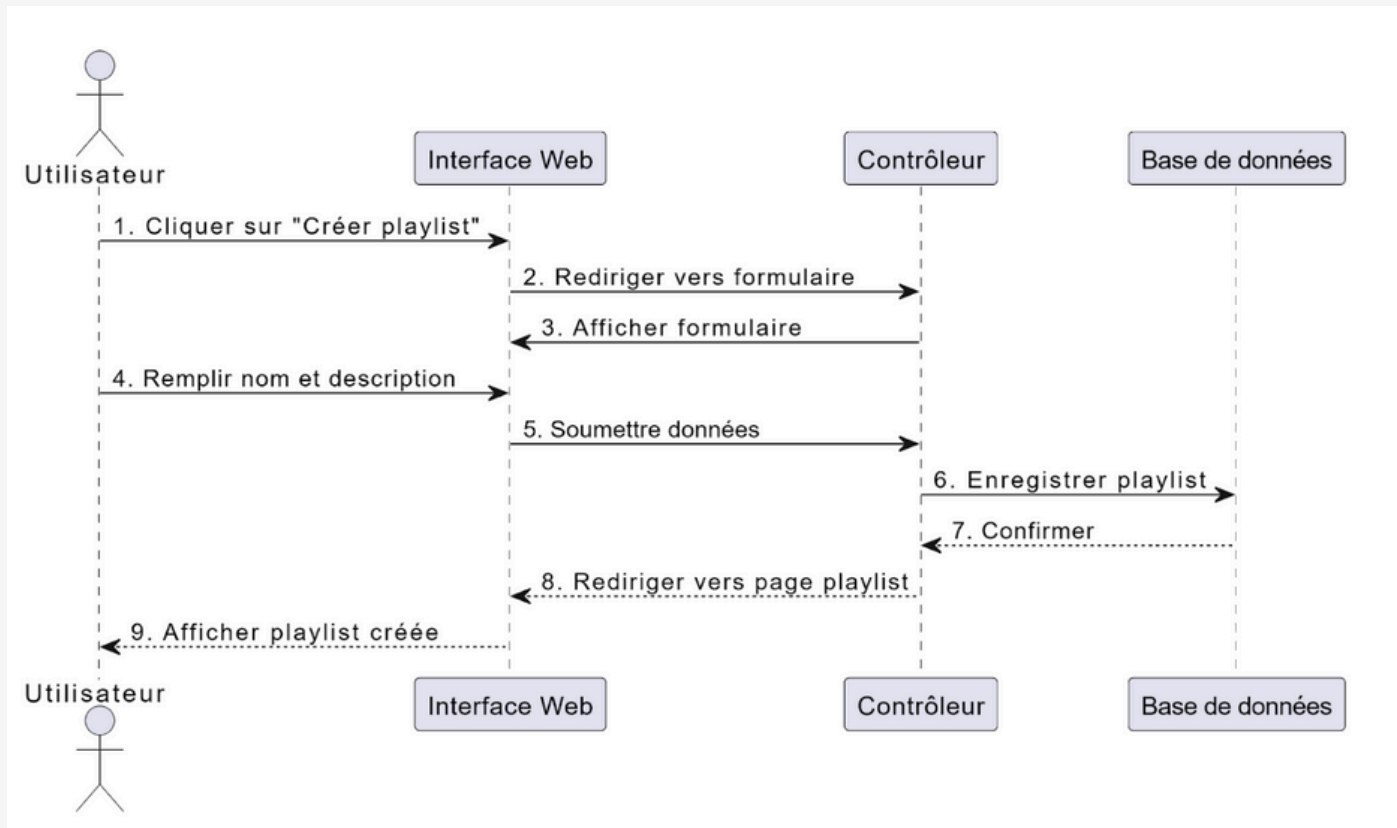
Authentification et gestion des profils

Diagramme de cas d'utilisation (Use Case)



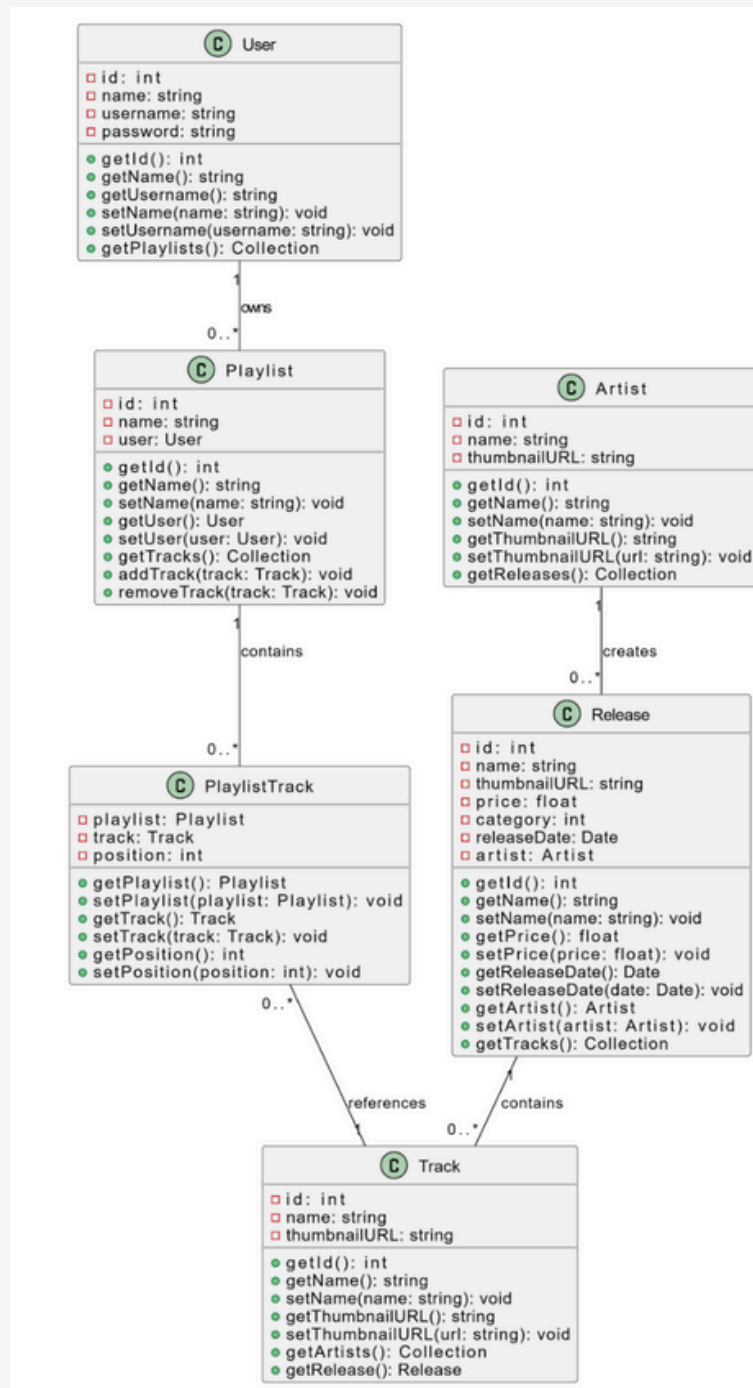
06 Le modèle de données

Diagramme de séquence - Création d'une playlist



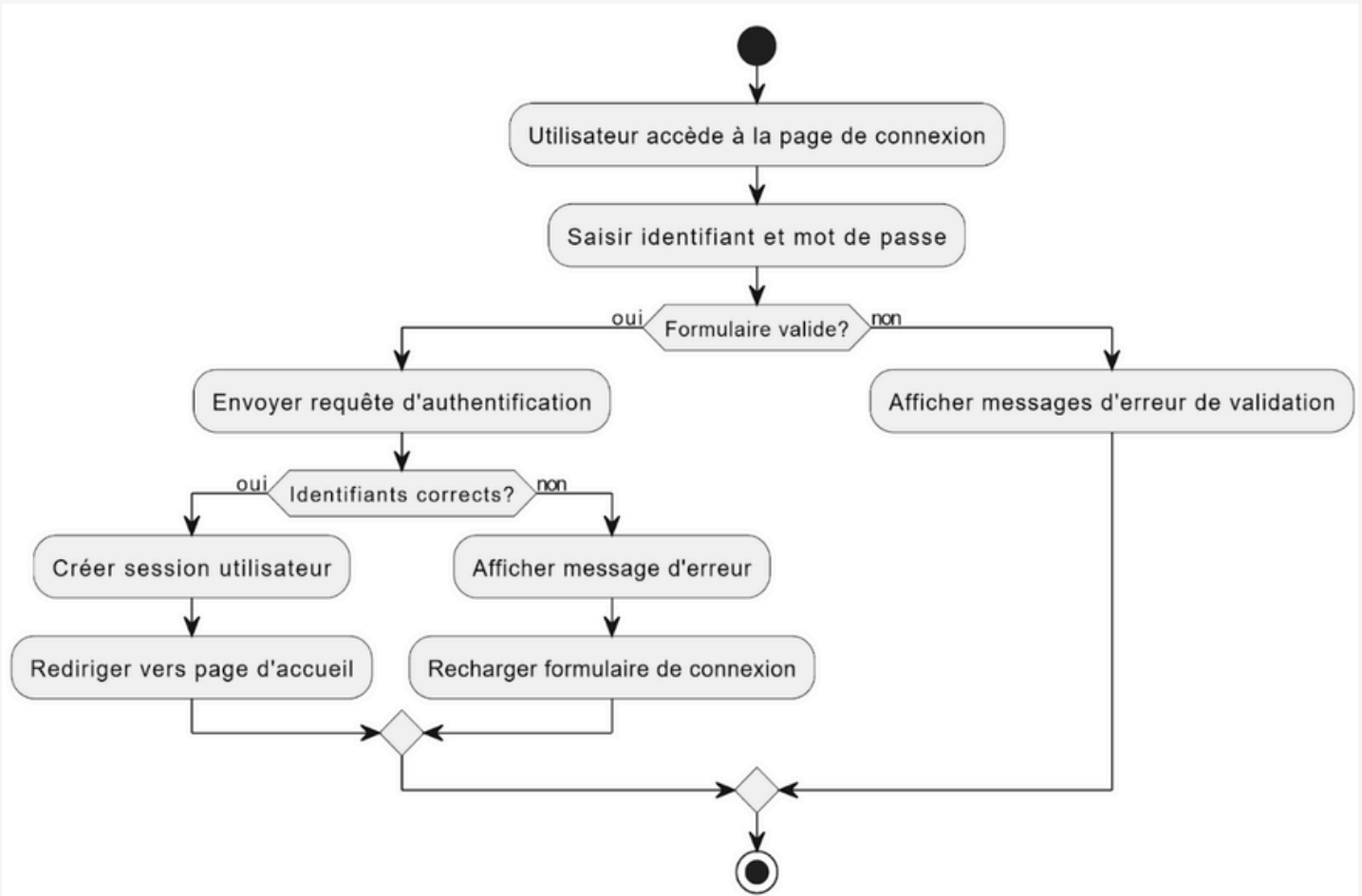
06 Le modèle de données

Diagramme de classes



06 Le modèle de données

Diagramme d'activité - Processus d'authentification



06 Le modèle de données

Diagramme d'état - Cycle de vie d'une playlist

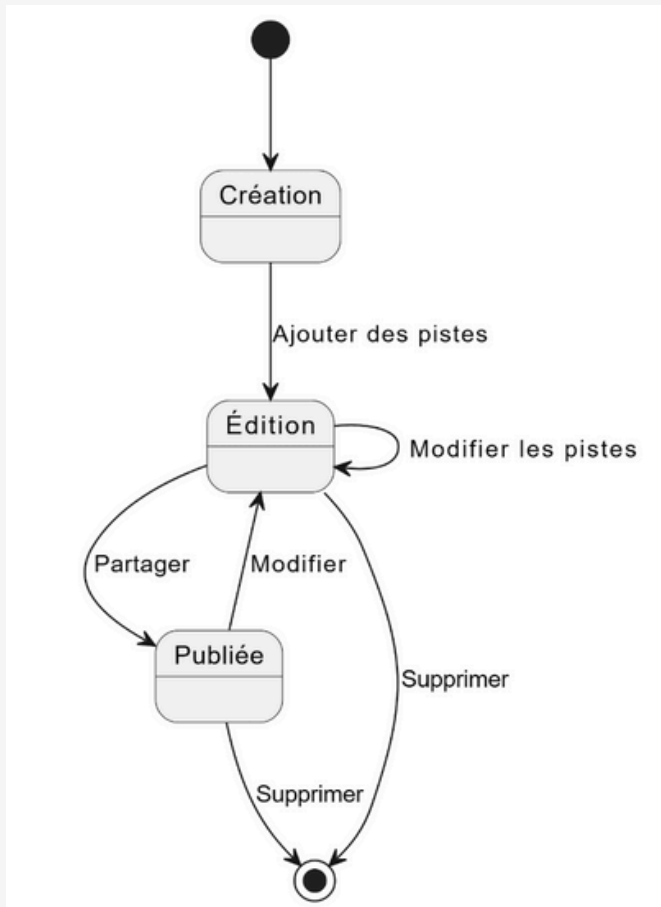
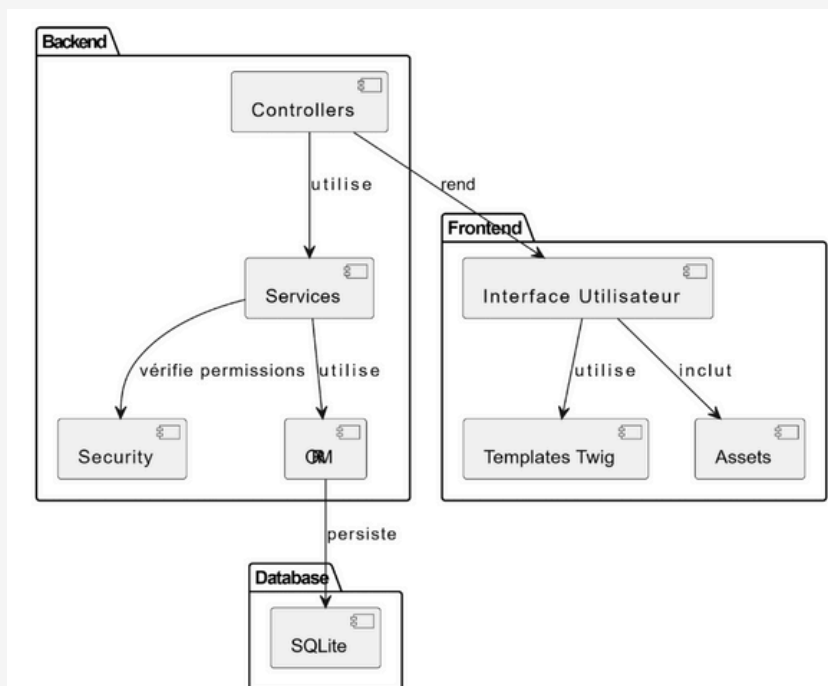


Diagramme de composants



06 Le modèle de données

Explication du code

Interface utilisateur

L'interface utilisateur a été conçue pour être intuitive et responsive, offrant une expérience optimale sur tous les types d'appareils. Voici les principales fonctionnalités de l'interface:

Page principale:

- Liste des morceaux récents
- Accès rapide aux playlists personnelles
- Barre de recherche



Page administrateur:

Comme indiqué dans le document original, nous avons développé une interface administrateur qui permet de créer des instances de l'entité artiste sans passer par du code, idéale pour un administrateur néophyte en informatique.

Gestion des playlists:



Création/modification de playlists

Ajout/suppression de morceaux

Organisation par glisser-déposer

06 Le modèle de données

[Music Library](#) [Artistes](#) [Albums](#) [Morceaux](#) [Se connecter](#)




Vortek's

Vortek's is a DJ composer who started music at the age of 13. He discovers the free world which inspires him enormously. Its sounds are particularly characterized by twisting ambient sounds and noises - but also by bewitching kicks. Vortek's draws its influences from the world of Rave, Hard music and Tekno. He sets no limits, no rules and leaves room for his imagination.

Releases

Il n'y a pas encore de release.

Apparaît dans



Profil utilisateur:

- Informations personnelles
- Historique d'activité
- Préférences

Comme mentionné dans le document original, Doctrine nous a permis de créer une entité "release" qui correspond aux différents albums d'un artiste. Cette entité possède des attributs tels qu'une date de sortie et un titre.

06 Le modèle de données

Voici un exemple du code Doctrine pour définir l'entité Release:

```
/**
 * @ORM\Entity(repositoryClass=ReleaseRepository::class)
 */
class Release
{
    /**
     * @ORM\Id
     * @ORM\GeneratedValue
     * @ORM\Column(type="integer")
     */
    private $id;
    /**
     * @ORM\Column(type="string", length=255)
     */
    private $name;
    /**
     * @ORM\Column(type="string", length=255, nullable=true)
     */
    private $thumbnailURL;
    /**
     * @ORM\Column(type="float", nullable=true)
     */
    private $price;
    /**
     * @ORM\Column(type="integer")
     */
    private $category;
    /**
     * @ORM\Column(type="date")
     */
    private $releaseDate;
    /**
     * @ORM\OneToMany(targetEntity=Track::class, mappedBy="release")
     */
    private $tracks;
    /**
     * @ORM\ManyToOne(targetEntity=Artist::class, inversedBy="releases")
     * @ORM\JoinColumn(nullable=false)
     */
    private $artist;

    // Getters and setters...
}
```

06 Le modèle de données

Architecture MVC

Notre application suit rigoureusement le modèle MVC proposé par Symfony:

Contrôleurs: Les contrôleurs gèrent les requêtes HTTP et coordonnent l'interaction entre le modèle et la vue.

```
/**
 * @Route("/tracks", name="track_")
 */
class TrackController extends AbstractController
{
    /**
     * @Route("/", name="index", methods={"GET"})
     */
    public function index(TrackRepository $trackRepository): Response
    {
        return $this->render('track/index.html.twig', [
            'tracks' => $trackRepository->findAll(),
        ]);
    }

    /**
     * @Route("/new", name="new", methods={"GET", "POST"})
     */
    public function new(Request $request): Response
    {
        $track = new Track();
        $form = $this->createForm(TrackType::class, $track);
        $form->handleRequest($request);
        if ($form->isSubmitted() && $form->isValid()) {
            $entityManager = $this->getDoctrine()->getManager();
            $entityManager->persist($track);
            $entityManager->flush();
            return $this->redirectToRoute('track_index');
        }
        return $this->render('track/new.html.twig', [
            'track' => $track,
            'form' => $form->createView(),
        ]);
    }
}
```

07 Les fonctionnalités

Modèles: Les entités Doctrine représentent les modèles de données.

Vues: Les templates Twig gèrent l'affichage des données.

```
{# templates/track/index.html.twig #}
{% extends 'base.html.twig' %}

{% block title %}Liste des morceaux{% endblock %}

{% block body %}
    <h1>Liste des morceaux</h1>

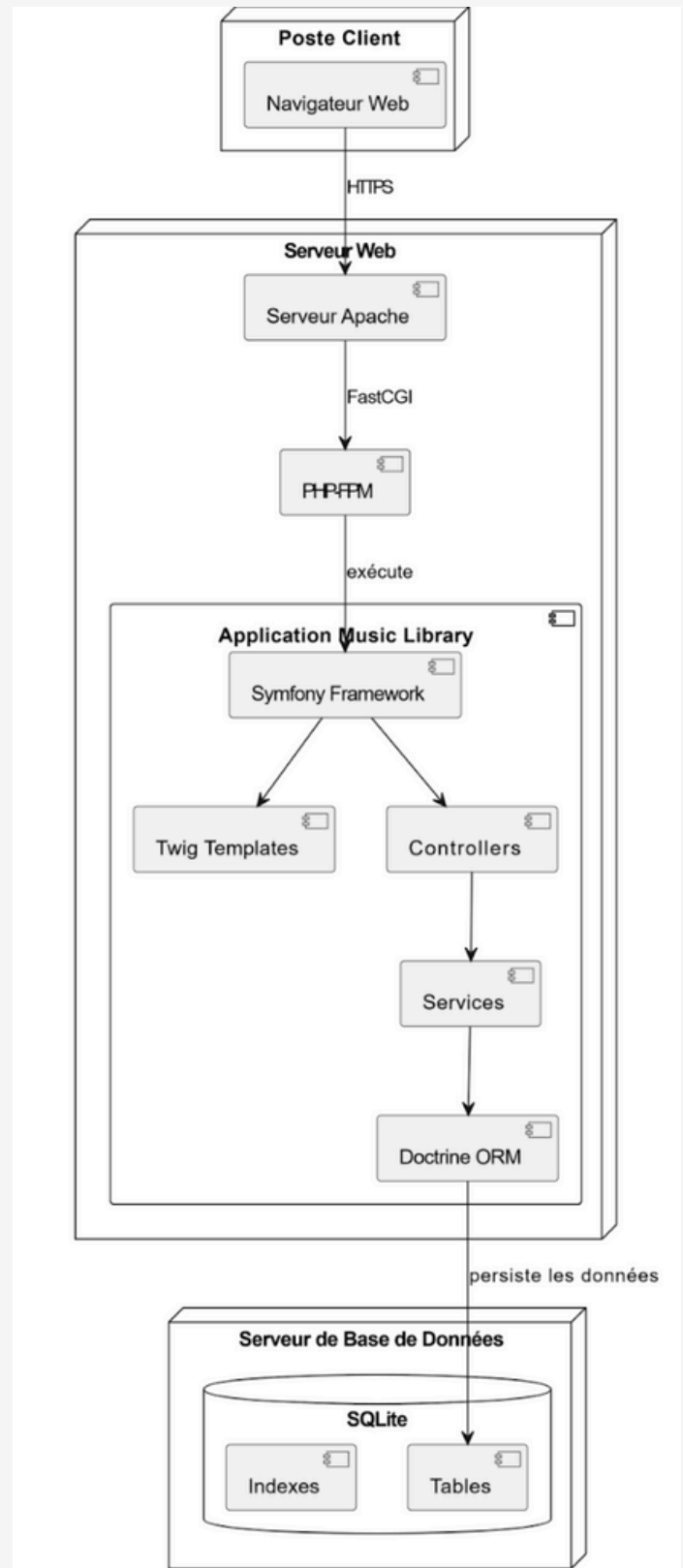
    <table class="table">
        <thead>
            <tr>
                <th>Id</th>
                <th>Nom</th>
                <th>Artiste</th>
                <th>Album</th>
                <th>Actions</th>
            </tr>
        </thead>
        <tbody>
            {% for track in tracks %}
                <tr>
                    <td>{{ track.id }}</td>
                    <td>{{ track.name }}</td>
                    <td>{{ track.artist.name }}</td>
                    <td>{{ track.release.name }}</td>
                    <td>
                        <a href="{{ path('track_show', {'id': track.id}) }}">Voir</a>
                        <a href="{{ path('track_edit', {'id': track.id}) }}">Modifier</a>
                    </td>
                </tr>
            {% else %}
                <tr>
                    <td colspan="5">Aucun morceau trouvé</td>
                </tr>
            {% endfor %}
        </tbody>
    </table>

    <a href="{{ path('track_new') }}">Ajouter un nouveau morceau</a>
{% endblock %}
```

07 Les fonctionnalités

Les tests ont été réalisés manuellement sur l'environnement local. Les points testés :

- Inscription / connexion utilisateur
- Accès restreint aux pages selon le rôle (utilisateur ou admin)
- Ajout de contenus musicaux via l'admin
- Consultation de la base par filtres
- Robustesse de la recherche et navigation
- Tests d'interface sur mobile (responsive OK)



08 Conclusion

Ce projet de bibliothèque musicale pour la M2L constitue une solution adaptée aux besoins spécifiques de l'organisation. En utilisant des technologies modernes comme Symphony et Doctrine, nous avons pu créer une application robuste, évolutive et facile à maintenir.

L'application permet désormais à la M2L de constituer et gérer facilement des playlists pour leurs événements sportifs, avec une interface intuitive accessible aussi bien aux administrateurs qu'aux utilisateurs finaux.

Les perspectives d'évolution incluent:

- L'intégration avec des services de streaming musical
- Des fonctionnalités avancées de recommandation
- Une application mobile complémentaire

Ce projet illustre notre capacité à comprendre les besoins spécifiques d'un client et à y répondre par une solution technique adaptée, en utilisant les meilleures pratiques de développement.

Note: Ce document est une ébauche du rapport final qui sera présenté lors de l'oral de l'épreuve E6.

09 Annexe

DESCRIPTION D'UNE RÉALISATION PROFESSIONNELLE		N° réalisation : 01
Nom, prénom : BROCHE Cyprien		N° candidat : 02443855526
<input checked="" type="checkbox"/> Épreuve ponctuelle <input type="checkbox"/> Contrôle en cours de formation		Date :
Organisation support de la réalisation professionnelle La Maison des Ligues de la Lorraine, établissement du Conseil Régional de Lorraine, est responsable de la gestion du service des sports et en particulier des ligues sportives ainsi que d'autres structures hébergées. La M2L doit fournir les infrastructures matérielles, logistiques et des services à l'ensemble des ligues sportives installées. Elle assure l'offre de services et de support technique aux différentes ligues déjà implantées (ou à venir) dans la région. M2L souhaite mettre en place une solution SaaS pour la gestion d'une bibliothèque musicale.		
Intitulé de la réalisation professionnelle Création d'une solution SaaS pour la gestion d'une bibliothèque musicale		
Période de réalisation : 01/10/2024 - 24/11/2024		Lieu : EPSI MONTPELLIER
Modalité : <input type="checkbox"/> Seul <input checked="" type="checkbox"/> En équipe		
Compétences travaillées <input checked="" type="checkbox"/> Concevoir et développer une solution applicative <input checked="" type="checkbox"/> Assurer la maintenance corrective ou évolutive d'une solution applicative <input checked="" type="checkbox"/> Gérer les données		
Conditions de réalisation ¹ (ressources fournies, résultats attendus)		
Ressources fournies : <ul style="list-style-type: none">• Cahier des charges M2L• PHP/framework Symfony• Bun		Résultats attendus : <ul style="list-style-type: none">• Application web fonctionnelle• Gestion des données• Gestion des événements
Description des ressources documentaires, matérielles et logicielles utilisées ² <ul style="list-style-type: none">• Cahier de charge M2L• Documentation technique de Symfony• Documentation d'utilisation de SQLite		
Modalités d'accès aux productions ³ et à leur documentation Lien de production : Insh.xyz/47af1b Lien repo Git : Insh.xyz/39ee7e Lien de documentations : Insh.xyz/6a1937		