

BTS SIO 2025

OPTION SLAM

ÉPREUVE E6

EPSI MONTPELLIER

Yousra ZAABAT

N° candidat : 10515619304

DOSSIER PROFESSIONNEL

SITUATION 1/2 : APPLICATION WEB AVEC GESTION DE PLAYLIST

Sommaire

Contexte.....	3
Présentation du client.....	3
Langages utilisés.....	5
Attendus des utilisateurs:.....	6
Attendus de la M2L:.....	6
Fonctionnalités principales:.....	6
La démarche.....	9
Développement.....	10
Modélisation.....	10
Architecture de la base de données.....	10
Authentification et gestion des profils.....	13
Explication du code.....	19
Interface utilisateur.....	19
Procédure de développement.....	24
Tests et validation.....	24
Conclusion.....	26

Contexte

La Maison des Ligues de Lorraine (M2L) a été créée en 2000 par suite de l'acquisition et la réhabilitation de la Maison Régionale des Sports de Lorraine par le Conseil Régional de Lorraine. Aujourd'hui, la M2L représente un vaste réseau sportif qui comprend environ 6 500 clubs, plus de 525 000 licenciés et près de 50 000 bénévoles.

Cet établissement joue un rôle central dans le développement et la cohésion du sport régional. Sa mission principale est d'héberger les structures sportives régionales et de leur fournir un ensemble complet de services administratifs, comptables et juridiques.

Le site de la M2L comprend plusieurs bâtiments qui offrent :

- Des bureaux pour les différentes ligues sportives
- Des espaces mutualisés incluant un amphithéâtre de 200 places
- Des salles de réunion adaptées aux besoins des organisations sportives
- Une salle de formation multimédia équipée
- Des locaux administratifs pour les différents organismes

Cette infrastructure permet aux dirigeants, bénévoles et salariés des associations sportives de se réunir, d'échanger et de se former dans des conditions optimales pour la gestion efficace de leurs ligues respectives.



Présentation du client

La Maison des Ligues de la Lorraine (M2L), 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.

Dans le cadre de ce projet, nous travaillons directement pour la Maison des Ligues de Lorraine (M2L), établissement situé dans la banlieue de Nancy. La M2L est un organisme financé par le Conseil Régional de Lorraine et administré par le Comité Régional Olympique et Sportif de Lorraine (CROSL).

Sa mission s'articule autour de quatre axes principaux :

Hébergement et soutien aux ligues sportives

La M2L met à disposition des espaces dédiés pour les ligues et comités sportifs régionaux, favorisant ainsi la collaboration inter-comités au sein de la structure.

Formation et accompagnement

L'établissement organise régulièrement des formations destinées aux bénévoles, dirigeants et salariés des différentes associations sportives. Ces formations couvrent des domaines variés tels que la gestion associative, la communication et les stratégies de financement.

Appui au développement du sport en Lorraine

La M2L participe activement à l'élaboration des politiques sportives régionales en collaboration avec les collectivités locales. Elle soutient également le développement de disciplines variées avec une attention particulière au handisport.

Mise à disposition d'équipements et de services

Outre les infrastructures physiques (salles de réunion, amphithéâtres, centres de formation), la M2L fournit des ressources immatérielles essentielles comme des logiciels spécialisés et un support technique et informatique pour l'ensemble des organismes sportifs de la région.

Projet actuel: La M2L nous a mandatés pour concevoir une application mobile e-commerce dédiée à la vente de matériel sportif. Cette application s'inscrit dans le cadre d'un nouveau partenariat commercial visant à offrir aux ligues et aux sportifs de la région un accès simplifié à l'équipement sportif.

Langages utilisés

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.

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**

 Music Library  Artistes  Albums  Morceaux  oui ▾

Mon profil

Firstname


Name

Email

Password

Update

[back to list](#)



Gestion des morceaux musicaux:

Ajout/Retrait de musique dans la bibliothèque

Catégorisation des morceaux par artiste, album, genre

Recherche et filtrage avancés

 Music Library  Artistes  Albums  Morceaux  oui ▾

 **Create new Artist**

Name

Thumbnail

Bio

Save

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 creating a new album. 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 and the name 'oui'. Below the navigation bar is a light purple header area with a back arrow icon and the title 'Create new Album'. The main form area is white and contains several input fields: 'Title' (a text input), 'Thumbnail' (a text input), 'Release date' (a date picker showing 'jj/mm/aaaa'), 'Artist' (a dropdown menu with 'Sköne' selected), and 'Type' (a dropdown menu with 'Album' selected). At the bottom left of the form is a red 'Save' button.

Administration:

Interface administrateur pour la gestion globale

Statistiques d'utilisation

Gestion des droits d'accès

La démarche

Pour la réalisation de cette application pour la M2L, nous avons suivi une démarche structurée et méthodique, garantissant l'adéquation de la solution aux besoins exprimés (qui va être détaillé par la suite).

Analyse des besoins:

- Entretiens avec les responsables de la M2L
- Identification des cas d'utilisation principaux
- Définition du périmètre fonctionnel

Conception de l'architecture:

- Choix des technologies (Symfony, Doctrine, SQLite)
- Conception du modèle de données
- Définition de l'architecture technique

Développement:

- Mise en place de l'environnement de développement
- Implémentation des fonctionnalités par itérations
- Tests unitaires et d'intégration

Design et ergonomie:

- Conception de l'interface utilisateur
- Optimisation de l'expérience utilisateur
- Adaptation responsive pour tous supports

Tests et recette:

- Tests fonctionnels
- Tests de performance
- Validation par les utilisateurs finaux

Déploiement et formation:

- Installation sur l'infrastructure de la M2L
- Formation des administrateurs
- Documentation utilisateur

Suivi et maintenance:

- Support technique
- Corrections et améliorations
- Évolutions futures

Développement

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:

User:

id: int

name: string

username: string

password: string(255) hash+salt

Track:

id: int

name: string

thumbnailURL: string

Artist:

id: int

name: string

thumbnailURL: string

Release:

id: int

name: string

thumbnailURL: string

price: float

category: int

Relations:

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

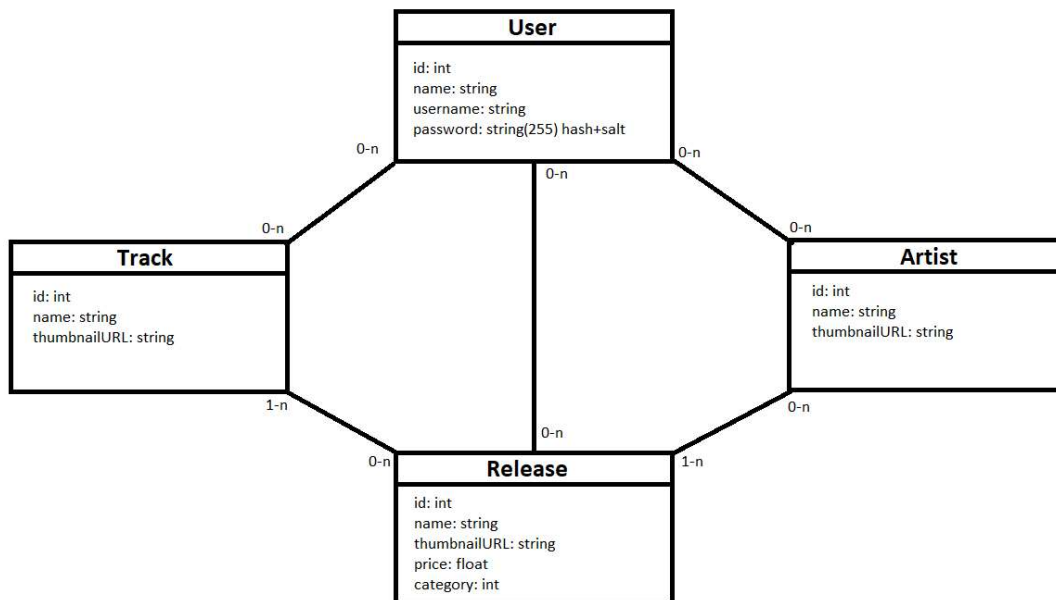
Un Track peut être lié à plusieurs Artist (relation 0-n)

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

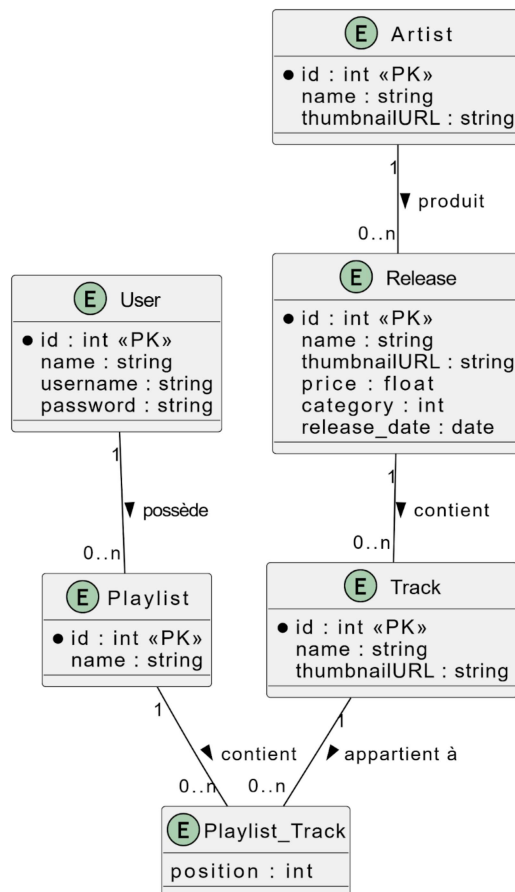
Un Track peut appartenir à un Release (relation 1-n)

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

(MCD.png)



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.

Authentification et gestion des profils

Diagramme de cas d'utilisation (Use Case)

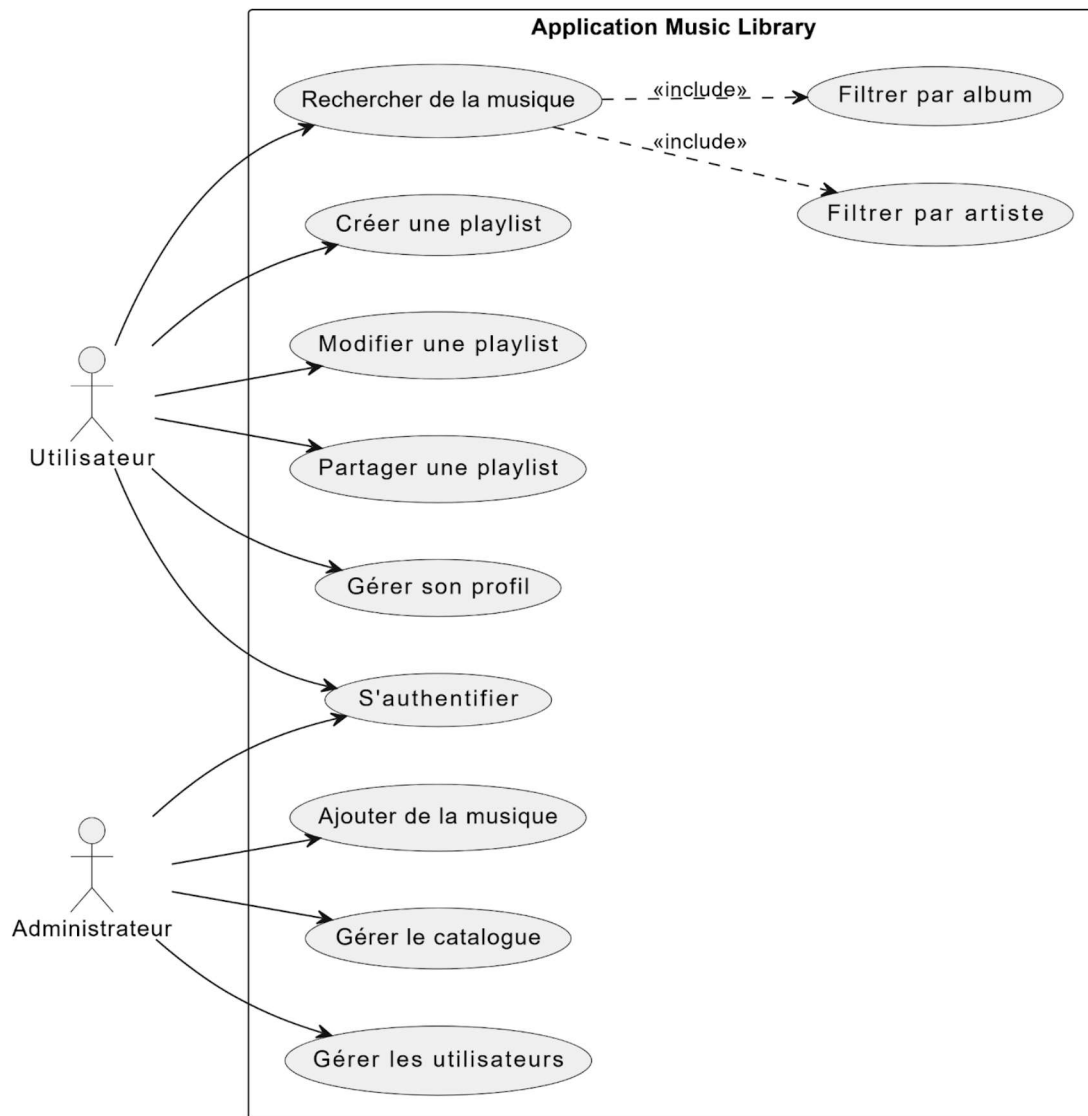


Diagramme de séquence - Création d'une playlist

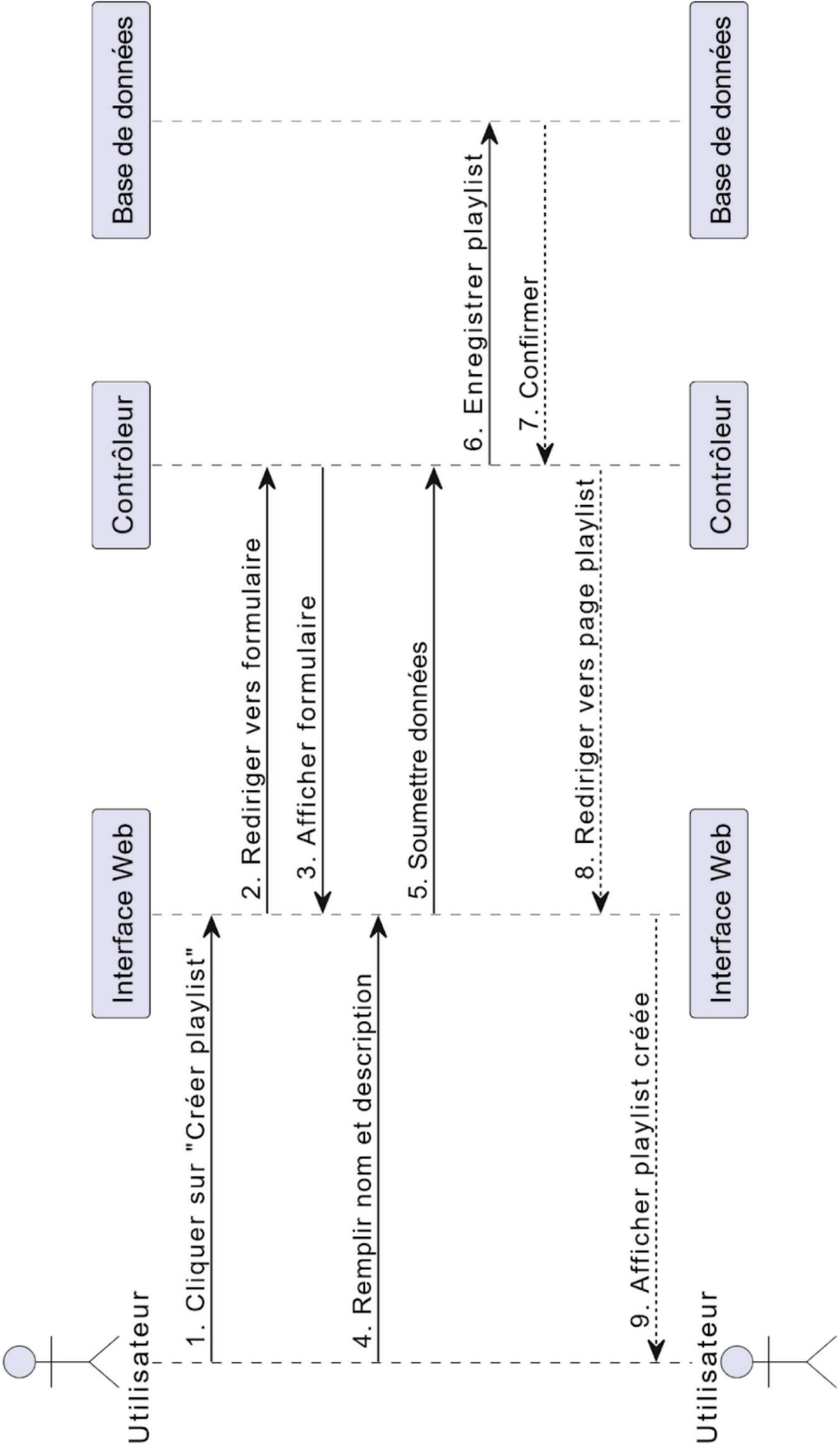


Diagramme de classes

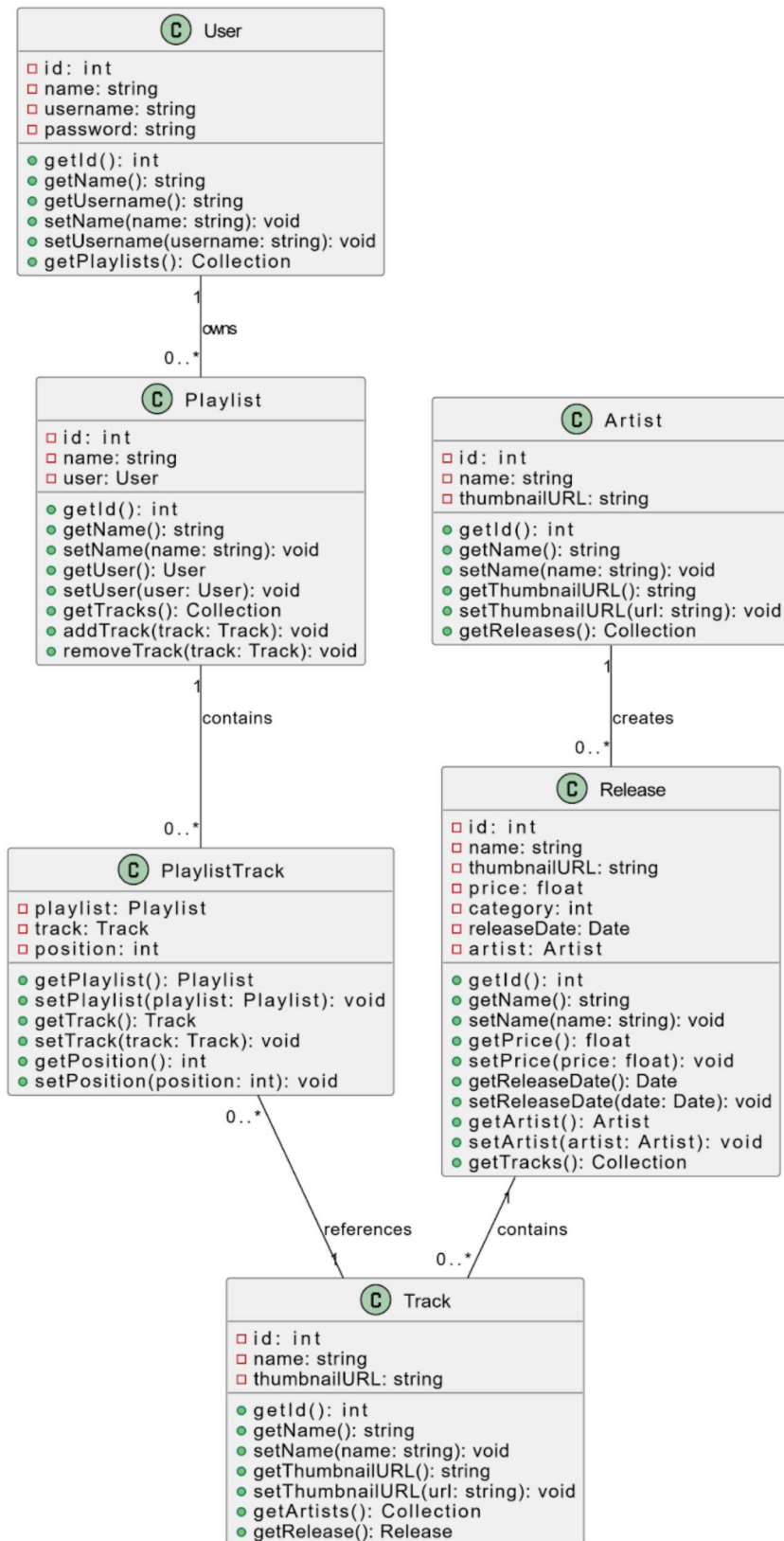


Diagramme d'activité - Processus d'authentification

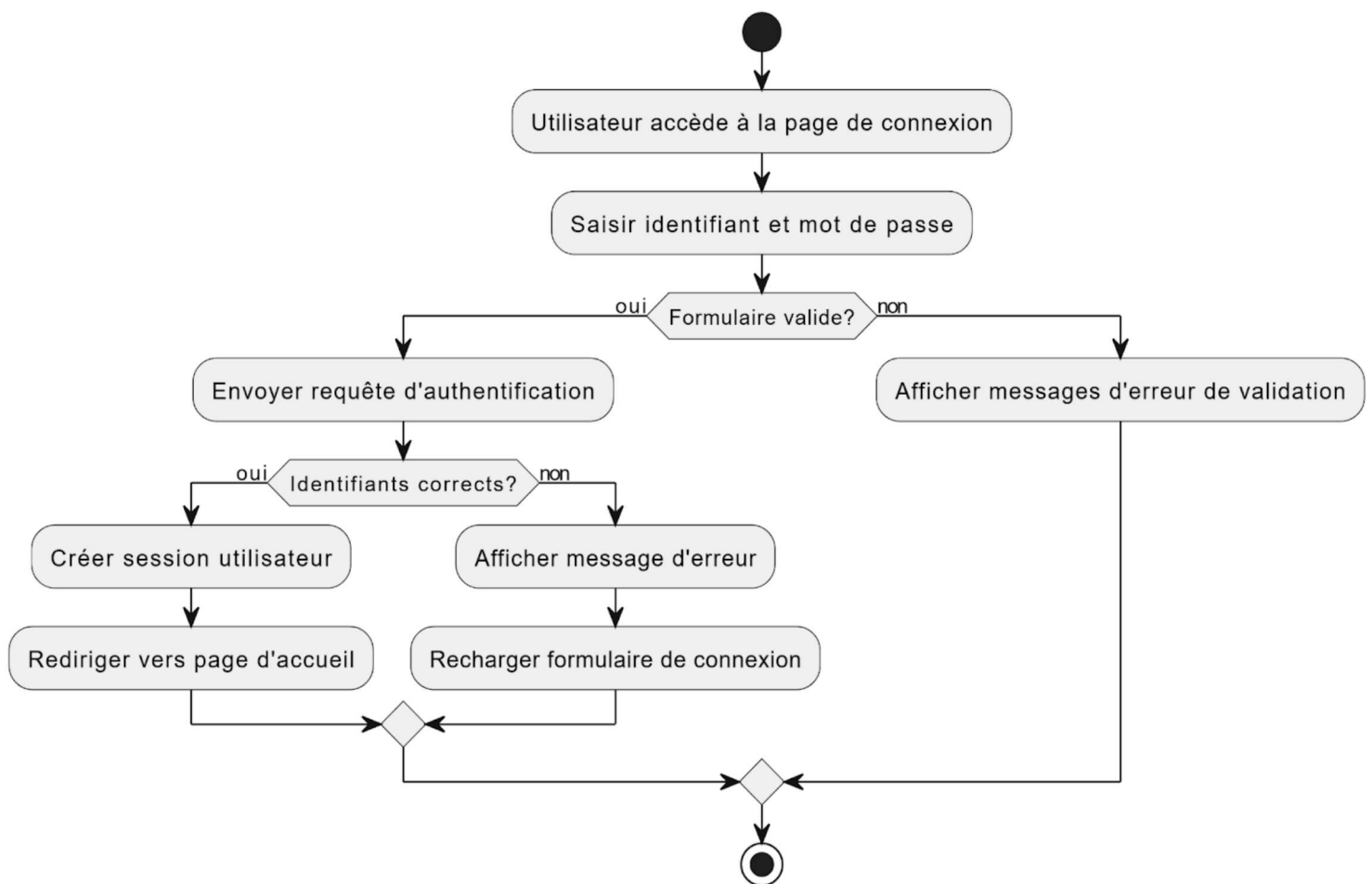


Diagramme d'état - Cycle de vie d'une playlist

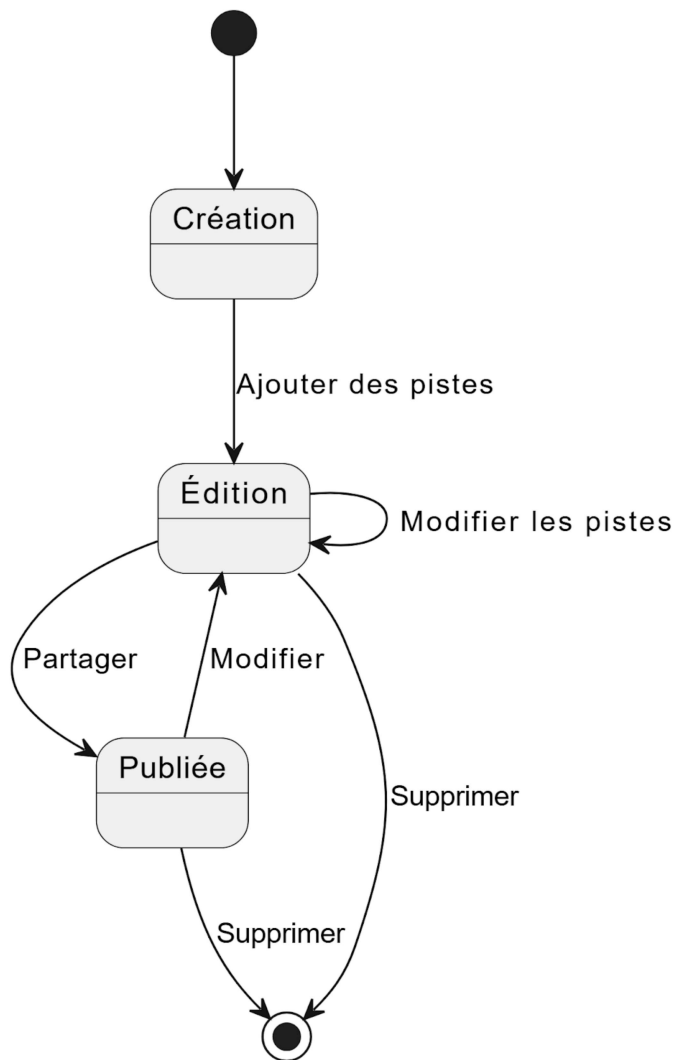
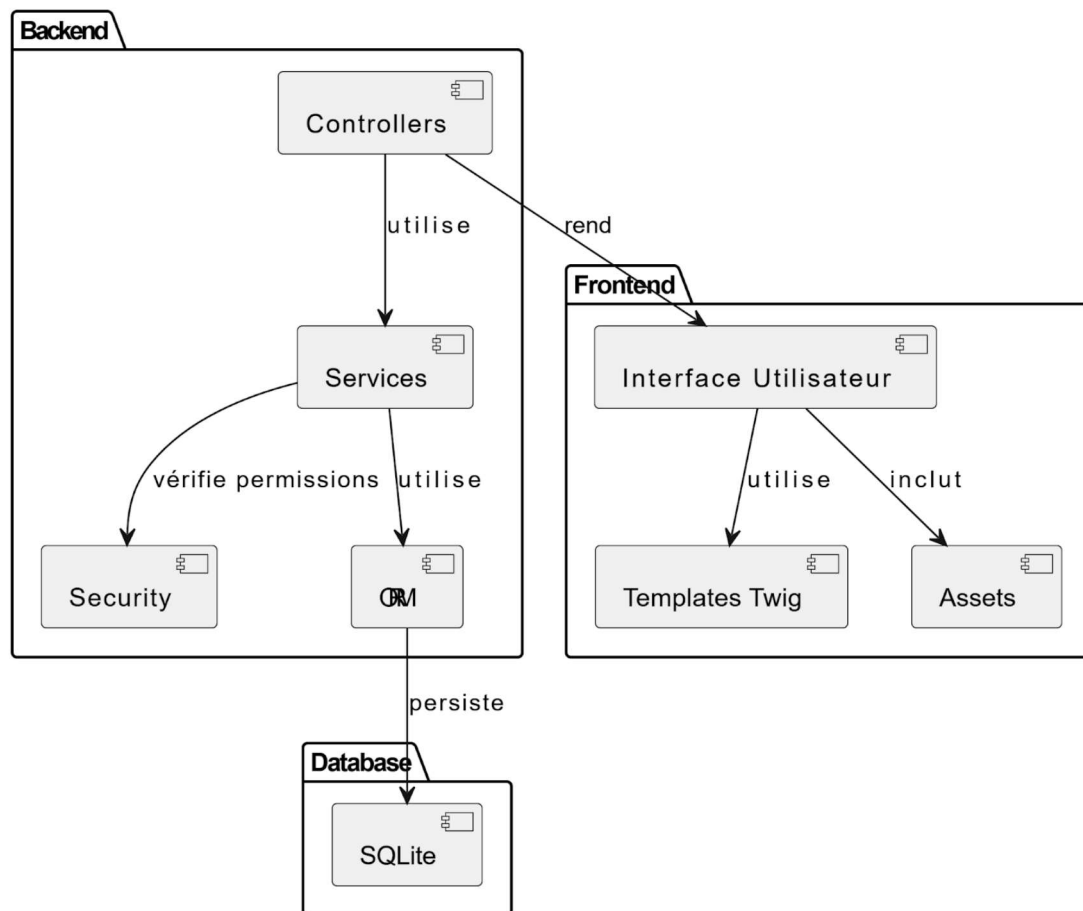


Diagramme de composants



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

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



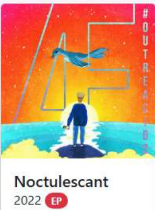
Vortek's

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.

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...
}
```

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(),
        ]);
    }
}
```

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>

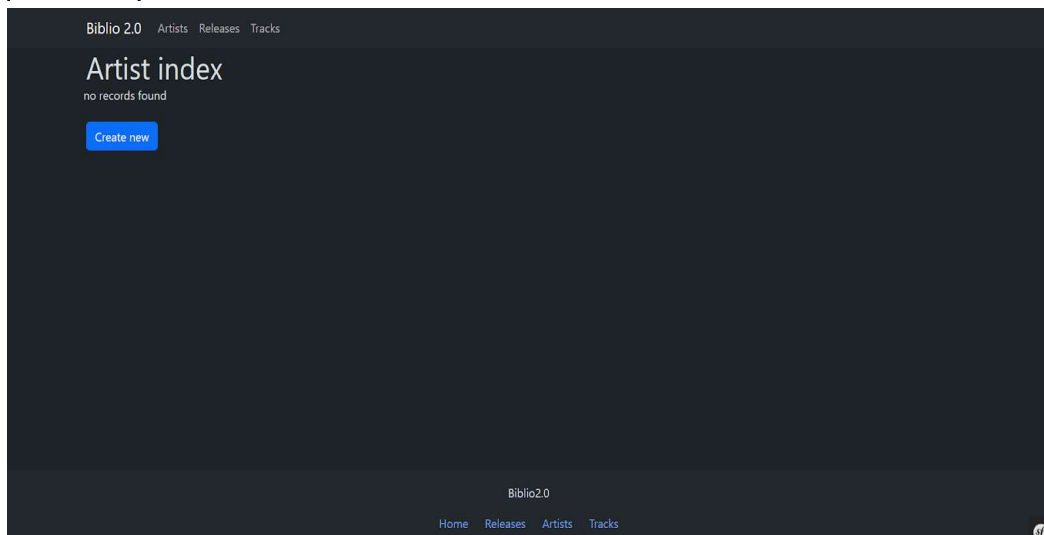
    <table class="table">
        <thead>
            <tr>
                <th>Id</th>
                <th>Nom</th>
                <th>Artiste</th>
                <th>Album</th>
                <th>Actions</th>
            </tr>
        </thead>
    </table>
{% endblock %}
```

```

    </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 %}

```



Procédure de développement

D'après le document PDF, voici la procédure de développement suivie:

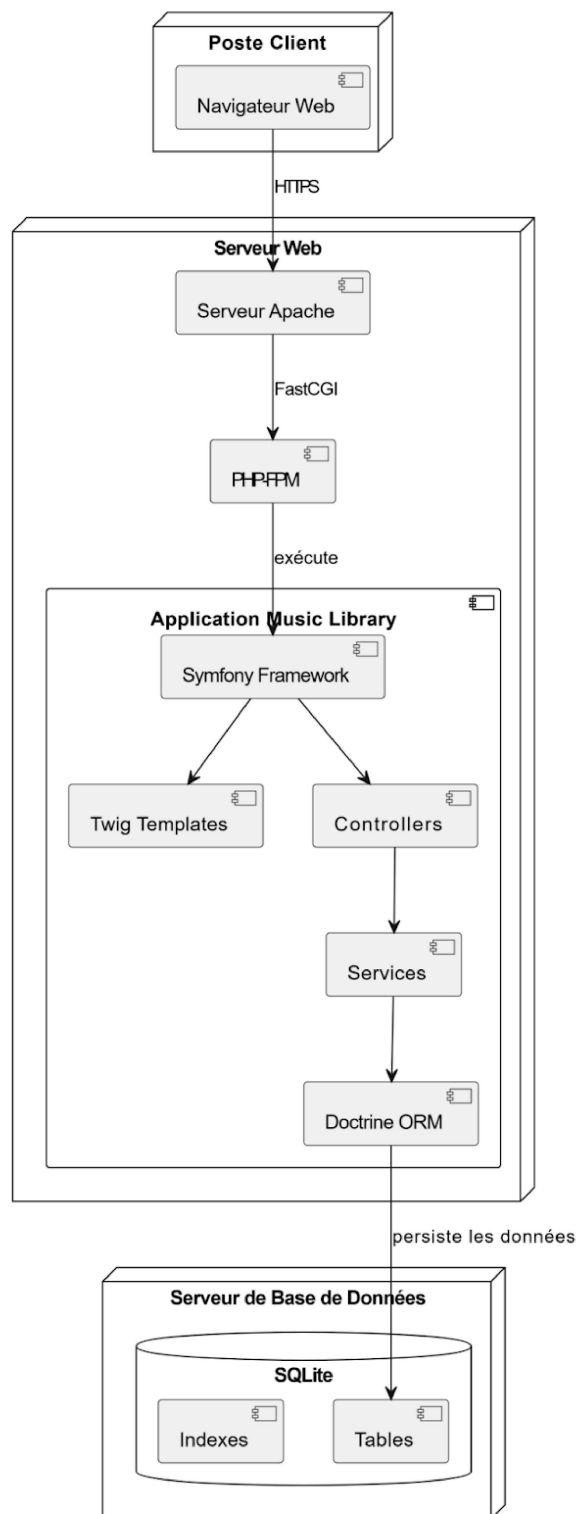
- Le front-end est développé avec Twig
- Le back-end est programmé en PHP (avec Symfony)
- La base de données utilisée est SQLite
- La gestion d'authentification est implémentée

Tests et validation

Des tests ont été réalisés pour vérifier:

- L'ajout/retrait de musique
- L'ajout/retrait d'utilisateurs
- L'accès à la page administrateur

Diagramme de déploiement



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 Symfony 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.

DESCRIPTION D'UNE RÉALISATION PROFESSIONNELLE		N° réalisation : 01		
Nom, prénom : ZAABAT Yousra		N° candidat : 10515619304		
<input checked="" type="checkbox"/> Épreuve ponctuelle <input type="checkbox"/> Contrôle en cours de formation		Date : date de passage		
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 pour la gestion d'une bibliothèque musicale.				
Intitulé de la réalisation professionnelle Music Library - 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 <ul style="list-style-type: none"> <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) <table border="0" style="width: 100%;"> <tr> <td style="vertical-align: top;"> Ressources fournies : <ul style="list-style-type: none"> Cahier des charges M2L PHP/ framework Symfony Doctrine </td> <td style="vertical-align: top;"> Résultats attendus : <ul style="list-style-type: none"> Une application Web fonctionnelle Gestion des données Gestion des événements </td> </tr> </table>			Ressources fournies : <ul style="list-style-type: none"> Cahier des charges M2L PHP/ framework Symfony Doctrine 	Résultats attendus : <ul style="list-style-type: none"> Une application Web fonctionnelle Gestion des données Gestion des événements
Ressources fournies : <ul style="list-style-type: none"> Cahier des charges M2L PHP/ framework Symfony Doctrine 	Résultats attendus : <ul style="list-style-type: none"> Une 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 de Symfony Documentation de SQLite 				
Modalités d'accès aux productions ³ et à leur documentation Lien de production : insh.xyz/6db744 Lien repo Git : insh.xyz/e12ec5 Lien de documentations : insh.xyz/6764d7				