4.Spécifications techniques

Voici les différentes technologies que j'ai utilisé pour ce projet :

Environnement de travail :

- Ordinateur de bureau avec windows 10
- Le logiciel visual studio code (logiciel complet avec beaucoup extension pour tous les langages).

Extension : github extension pack

- Outil de version et de partage de code github, qui me permet de déposer mes commit après chaque fonctionnalité de l'application et de pouvoir créer une nouvelle branche.
- le logiciel xampp pour le serveur apache et la base donnée mysql
- un serveur linux hebergé chez ionos pour la mise en production des site web.
- Outil de transfert de fichiers filezilla pour le transfert du site de dev vers le site de production. Transfert que j'ai effectué régulièrement afin de m'assurer du fonctionnement de l'application.
- Outil web de suivi de projet TRELLO (https://trello.com/), qui m'a permis de planifier et de suivre toutes les taches a effectuer pour mon projet.

-Suivi des taches ECF David MARCAIS :
https://trello.com/invite/b/Sf55i8wO/0837745ec78a6aac00b3838920b56e51/ecf

Front-End :

HTML5, CSS3, JQUERY, Ces technologies sont indispensables aujoudh'iu pour le codage côté front.

Le thème LANKIT – est Template basé sur les composants Bootstrap, pour la construction de site moderne et flexible. Très simple utilsation et m'a permis de construite des pages front rapidement en adaptant le code à mo projet.

https://landkit.goodthemes.co/

Le Framework Bootstrap 5 , compatible avec la majorité des navigateurs, permet de rendre un site web attrayant aves ses nombreux modules, et surtout adaptable à tous les types d'écrans.

Back-End

Php , Ce langage est très populaire pour la créationde site web , disponible sur la plupart des systèmes exploitations et permet de se connecter à une grande quantité de bases de donnée.

Le Framework symfony est un outil très puissant et populaire avec de nombreuses mises à jours et un suivi de la communauté, il est modulable et adaptable, et il m'a permis de développer des fonctionnalités très rapidement.

Extension symfony:

-fixtures avec faker php pour la création de données aléatoire et alimentation automatique de la base de données

-KnpPaginatorBundle, afin de créer des pages avec de la pagination. Très simple à mettre en place dans un projet.

-EasyAdminBundle, afin de créer un backoffice rapidement et utilisable rapidement par les administrateurs du site et aussi les employés dans le cas de mon projet.

-VichUploader, afin de faire upload des photos dans easyadmin.

-Security-bundle, pour la mise en place de la sécurité et authentification des utilisateurs du site

-Reset-password-bundle pour régénérer un mot de passe en cas d'oubli ou de perte.

-Webpack encore, est un modules bundle open source. Son objectif principal est de regrouper des fichiers JavaScript pour les utiliser dans un navigateur

La base de donnée mysql fournit avec xampp, une distribution facile à installer et populaire dans le monde du developpement web.

5.Description du travail

a. Environnement de travail

Mise en place de mon environnement de développement avec le logiciel visual studio code. J'ai choisi ce logiciel car il est complet au niveau des langages de programmation, il a de nombreuses extensions qui facilite la vie d'un développeur et de nombreuses mises à jour sont disponibles.

J'ai installé ensuite le logiciel xampp pour mettre en place un serveur Web local et une base de donnée mysql.

Ensuite j'ai installé le logiciel filezilla pour le transfert de mes fichiers vers mon serveur en ligne disponible chez l'hébergeur https://www.ionos.fr/. Le serveur est un linux Ubuntu version 20.4 avec serveur apache, base de données MySQL (lampp). Je dispose aussi d'un nom de domaine davidmarcais.fr que j'utilise pour ce projet. Mon serveur en ligne est protégé en permanence par ionos. J'ai choisi ce fournisseur car il a une infrastructure qui est l'une des plus sécurisées au monde depuis des années. J'ai déjà eu des attaques sur mon adresse email et ionos a fait le nécessaire rapidement pour bloquer l'envoi de mails. Mon serveur web est bien protégé avec un mot de passe très sécurisé. J'ai également sécurisé ma base de donnée avec un mot de passe.

J'ai donc commencé par faire la création de mon projet ECF avec l'installation du framework symfony. J'ai décidé d'utiliser cette technologie car j'ai étudié ce framework dans le cadre de ma formation chez study. J'ai réalisé également une évaluation d'entrainement avec ce framework. Je ne voulais pas vu le temps imparti me lancer dans un autre framework que je ne connaissais pas.

J'ai donc initialisé un nouveau projet sous symfony :

> symfony new ECF

J'ai fait la création d'un dépôt git sous github afin de synchroniser chaque soir mes sources et pouvoir les récupérer en cas de problème sur mon ordinateur personnel.

Source disponible et accessible en public à l'adresses :

- https://github.com/dms70/ECF

B. Développement

J’ai installé les 2 packages suivants pour la communication entre symfony et ma base de données :

> composer require symfony/orm-pack

composer require --dev symfony/maker-bundle

Edition du fichier .env sous symfony avec la mise à jour de la variable DATABASE_URL

Création de la base de données avec la commande :

> php bin/console doctrine:database:create

J’ai ensuite réalisé la création de mes entités avec la commande make :entity

Je me suis donc servi du diagramme de classe pour faire la création de mes entités :

User, Book, Category et Genre.

Et lancement de la commande pour la création des entités dans ma base de données:

> php bin/console doctrine:schema:update –force

Dans le cours du projet j’ai dû ajouter des champs que je n’avais pas prévus initialement lors d’élaboration du diagramme de classe.

Le projet s’est fait avec du développement par moment en back end et d’autres en front-end

Pour la partie front, J’ai donc installé le package Webpack-encore, qui est une api qi permet de traiter les fichiers css et js , et de les compiler dans le projet. Il permet de réduire le code. J’ai installé le module bootstrap 5 et le module landkit qui est un des rares template compatible avec bootstrap 5 et très complet avec de nombreux modules réutilisables.

Grace à ces modules, j’ai pu commencer la création de la page principale avec un pied de page, haut de page et une barre de menu en respectant les wireframme.

Ensuite j’ai commencé intégration de la partie sécurité du site avec la mise en place d’une page de connexion sécurisée.

J’ai installé le package security-bundle de symfony, qui fournit toutes les fonctionnalités d'authentification et d'autorisation nécessaires pour sécuriser mon application médiathèque.

J’ai utilisé l’adresse email d’utilisateur comme identifiant pour se connecter sur le site. Lors de la création de l’entité « user », j’ai demandé à utiliser la méthode « hash » pour sécuriser le mot de passe afin qu’il ne soit pas visible dans la base de donnée et la vérification du mot de passe lors de la connexion.

J’ai remplacé dans toutes les sections nécessitant la méthode par défaut de symfony par la nouvelle la méthode :

> use Symfony\Component\PasswordHasher\Hasher\UserPasswordHasherInterface

Au fil du j’ai configuré la partie pare-feu de symfony avec dans un premier temps avec le contrôle d’accès au site avec vérification du compte. J’ai utilisé la clé « IsVerfied » et ajouter un script de vérification « App\Security\UserChecker ». Seul employé est habilité à valider l’accès au site.

J’ai aussi ajouté en fin de projet une double authentification avec le package « scheb/2fa » , ce package permet de faire un contrôle supplémentaire (voir section pour explication du concept) à partir d’un code que utilisateur recevra par email et lui permettant de lui donner l’accès au site en saisissant ce code sur le site. Cet accès est valide plusieurs jours sans avoir à retaper un code à chaque fois ? Il garantit que le poste sur lequel on se connecte est confiant.

Pour le contrôle d’accès (autorisation) j’ai utilisé le système de rôle de symfony. J’ai Crée 3 rôles avec des autorisations différentes et un niveau hiérarchique suivant les utilisateurs. Certaines url ne seront pas accessibles aux habitants.

Extrait Fichier : SECURITY.YAML

```
role_hierarchy:
    ROLE_ADMIN:       [ROLE_HABITANT,ROLE_EMPLOYE,ROLE_ADMIN]
    ROLE_EMPLOYE:     [ROLE_EMPLOYE,ROLE_ADMIN]
    ROLE_HABITANT:    [ROLE_HABITANT]
  access_control:
    - { path: ^/2fa, role: IS_AUTHENTICATED_2FA_IN_PROGRESS }
    - { path: ^/logout, role: IS_AUTHENTICATED_ANONYMOUSLY }
    - { path: ^/profil, roles: ROLE_HABITANT}
    - { path: ^/meslivres, roles: ROLE_HABITANT}
    - { path: ^/catalogue, roles: ROLE_HABITANT}
```

- { path: ^/admin, roles: ROLE_EMPLOYE }

- { path: ^/superadmin, roles: ROLE_ADMIN }

- { path: ^/categoriesRomans, roles: ROLE_HABITANT }

- { path: ^/categoriesEnfants, roles: ROLE_HABITANT }

- { path: ^/categoriesBD, roles: ROLE_HABITANT }

- { path: ^/categoriesDocumentaires, roles: ROLE_HABITANT }

- { path: ^/returnerdbook, roles: ROLE_EMPLOYE }

- { path: ^/validationbook, roles: ROLE_EMPLOYE }

- { path: ^/removereservation, roles: ROLE_HABITANT }

- { path: ^/registered, roles: ROLE_EMPLOYE }

J'ai ensuite utilisé la méthode « php bin/console make:auth » pour la création de mon formulaire de connexion sécurisée.

Dans mon formulaire de connexion j'ai ajouté une double saisie du mot de passe afin de vérifier le mot de passe est identique. J'ai utilisé un script JavaScript « password.js » afin d'afficher une alerte en cas de mauvaise saisie.

J'ai utilisé la méthode standard de symfony avec la vérification des champs de saisie afin d'éviter des injections sql. Il est possible d'ajouter des contraintes personnalisées pour sécuriser encore plus le projet mais je ne l'ai pas fait pour ce projet. Doctrine de symfony est déjà bien fournit pour se protéger à minima.

J'ai créé un formulaire pour demander la réinitialisation du mot de passe en cas de perte, d'oubli ou de vol. Un email est envoyé à utilisateur pour ressaisir un nouveau mot de passe. Dans l'application il y a aussi un menu pour changer son mot de passe. J'ai ajouté une méthode captcha a tous ces formulaires pour éviter que les robots de saisi automatique.

Afin de faire des tests de remplissage de ma base de données j'ai utilisé le module dev orm-fixtures, faker php et FakerPicsumImagesProvider. J'ai donc fait la création d'un fichier php (AppFixtures.php) adapté à mes besoins. Ce fichier permet de faire la création automtique d'un administrateur, d'employés, d'utilisateurs, de livres, des catégories et de genre de livres. J'ai dû trouver une astuce pour le champ image car le module insérer le fichier avec son chemin complet. J'ai donc crée un script php qui change modifie le champ avec le nom uniquement du fichier image.

```php
<?php

$pdo = new PDO('mysql:host=localhost;dbname=mediatheque', 'root', '');
$statement = $pdo->prepare('SELECT image,id FROM book');

if ($statement->execute()) {
    while ($book = $statement->fetch(PDO::FETCH_NUM)) {
        $info = new SplFileInfo($book[0]);
        $filenameimage = $info->getFilename();
```

```
        echo '<image>';
        print_r($filenameimage);
        print_r($book[1]);
        echo '<image>';
        $image = $pdo-
>prepare("UPDATE book SET image = '$filenameimage' WHERE id = $book[1] ");
        $image->execute();
    }
}
```

J'ai fait la création d'un backoffice pour la gestion d'ajout de categories, de genre de livres, de validation de comptes accessible uniquement par les employés et administrateur. Symfony dispose du easyadmin-bundle package à adapter suivant ses besoins. Création d'un backoffice (/admin) pour les employés et un backoffice pour administrateur (/superadmin). Seul administrateur du site peut faire la création de compte.

> **composer require easycorp/easyadmin-bundle**

J'ai ajouté un contrôler « crud » pour les livres, catégorie, genre et utilisateurs. J'ai modifié ces contrôleurs afin de spécifier les champs de saisie que je voulais pour la création.

J'ai dû intégrer pour la création des utilisateurs\employés un souscripteur a des d'évènements « dont celui du mot de passe. Sans ajout de ce service le mot de passe est envoyé en clair dans la base de donnée. Seul administrateur peut faire cette tâche.

Création du répertoire EventSubscriber avec le fichier EasyAdminSubscriber qui permet de faire « hash » du mot de passe avant l'envoi dans la base de donnée.

Pour la gestion des photos de couverture de livres, J'ai ajouté le module « VichUploaderBundle » qui permet de faire facilement upload et de mettre le nom du fichier dans la base de donnée. Il nomme et sauvegarde automatiquement le fichier dans un répertoire donnée ' /uploads/couverture).

J'ai ajouté de la pagination pour la visualisation du catalogue, j'ai donc choisi d'utiliser le module « KnpPaginatorBundle », afin de créer des pages paginées. Ce module est très simple à mettre en place et à adapter dans une fonction.

J'ai créé des contrôleurs symfony dans le répertoire contrôler pour chaque page rendu sur le site et situé dans répertoire template. Les contrôleurs est la partie qui orchestre mon application.

Pour la gestion de mes livres j'ai donc crée un contrôler : **MesLivresController**

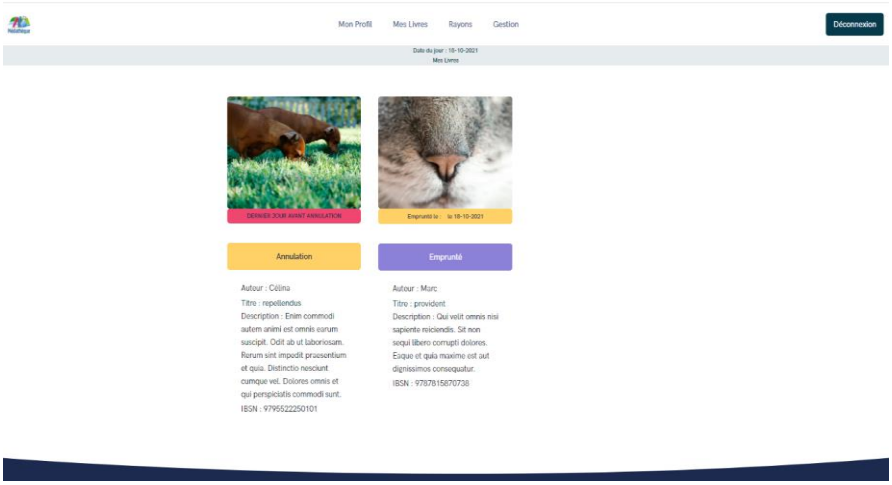
```php
<?php

namespace App\Controller;
use App\Entity\Book;
use App\Repository\BookedRepository;
use App\Repository\BookRepository;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;
class MesLivresController extends AbstractController
{

    #[Route('/meslivres', name: 'mes_livres')]

    public function meslivres(BookedRepository $BookedRepository, BookReposito
ry $BookRepository): Response
    {
        $user = $this->getUser();
        $userId = $user->getId();

        /** @var ActivityRepository */
        $entityManager = $this->getDoctrine()->getManager();
        $repository = $entityManager->getRepository(Book::class);

        return $this->render('mes_livres/meslivres.html.twig', [
           'Books' => $repository->findAllWithQB(user_id : $userId),
        ]);

    }
}
```

Celui-ci va afficher la page « meslivres ('mes_livres/meslivres.html.twig') » avec tous les livres réservés ou empruntés par un utilisateur. Dans ma méthode , le contrôler va lancer une fonction de recherche qui se situe dans un fichier dans le répertoire « repository ». Dans ce cas présent je vais rechercher la fonction » findAllWithQB » dans le fichier BookRepository.php.

```php
<?php

namespace App\Repository;

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

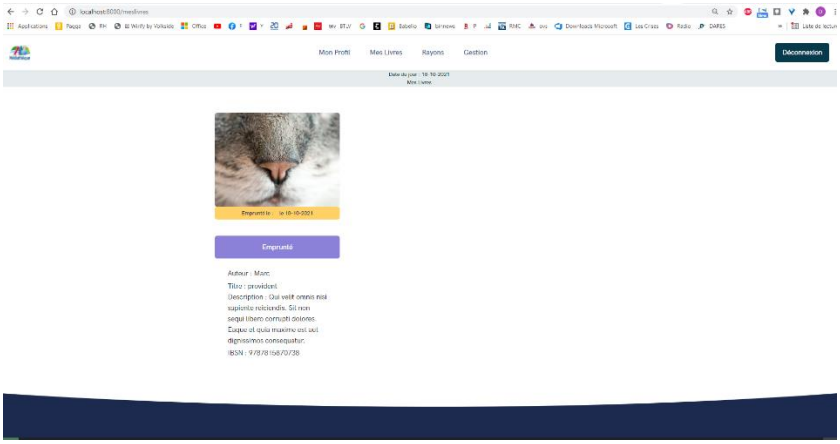
```php
use Doctrine\Migrations\Query\Query;
use phpDocumentor\Reflection\Types\Integer;

/**
 * @method Book|null find($id, $lockMode = null, $lockVersion = null)
 * @method Book|null findOneBy(array $criteria, array $orderBy = null)
 * @method Book[]    findAll()
 * @method Book[]    findBy(array $criteria, array $orderBy = null, $limit = n
ull, $offset = null)
 */
class BookRepository extends ServiceEntityRepository
{
    public function __construct(ManagerRegistry $registry)
    {
        parent::__construct($registry, Book::class);
    }
.....
    public function findAllWithQB(int $user_id)
    {
        return $this->createQueryBuilder('e')
            ->where('e.user = :user')
            ->setParameter('user', $user_id)
            ->getQuery()
            ->getResult();
    }
.....
}
```

Le contrôler se charge d'afficher la page « mes Livres » :



Pour la gestion des boutons j'ai créé des fichiers services dans le répertoire « service « qui s'occupent de faire l'annulation du livre en mettant à jour la table « user » et « book ». Le livre disparait de l'affichage.

```
  public function removereservation(int $isbn): void
    {

        //$user = $this->getUser();
       // $userId = $user->getId();

        /** @var ActivityRepository */
        $entityManager = $this->getDoctrine()->getManager();
        $book = $entityManager->getRepository(Book::class)-
>findOneByisbn($isbn);
        dump($book);
        $book->setReserved(false);
        $book->setBookeddate(NULL);
        $book->setBookeds(NULL);

        $id = $book->GetUser();
        $userfound = $entityManager->getRepository(User::class)-
>findOneByid($id);

       //$user ->getUser();
        $Bookborrowed= $userfound ->getBookborrowed();
        $Bookborrowed = --$Bookborrowed;
        $userfound->setBookborrowed($Bookborrowed);

        $book->setUser(NULL);

        //$user = $this->getUser();

        $entityManager->persist($book);
        $entityManager->persist($userfound);

        $entityManager->flush();


    }
```



Pour la communication entre le client et le serveur, j'ai installé le module webpack encore afin d'utiliser du JavaScript encapsule. J'ai fait la création de quelques fonctions JavaScript qui permettent de faire des contrôles ou d'éviter de recharger la page avec des requêtes exécutées par l'utilisation du client http axios.

Exemple du contrôle du mot de passe lors de la saisie des 2 champs. Si les mots de passe saisies sont différents, javasript se charge de faire le contrôle et d'envoyer un pop up avertissement.

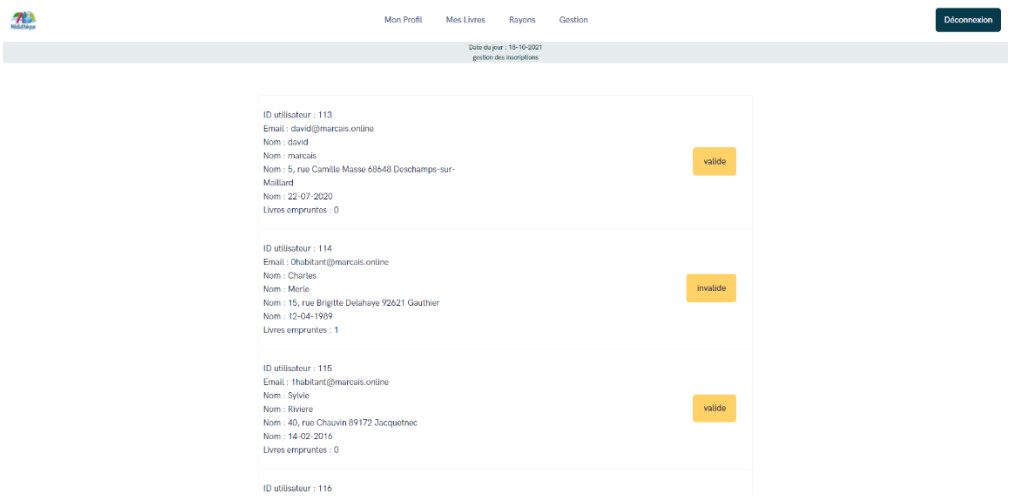
```
import jquery from 'jquery';

const alertify = require('alertifyjs');
import 'alertifyjs/build/css/alertify.min.css';

jquery("#new_edit_utilisateur").on('submit', function(){
if(jquery("#passwordfirst").val() != jquery("#verifpass").val()) {
alert("Les 2 mots de passe saisies sont différents");

return false;
}
})
```

Exemple du cas de la page de validation des comptes : Le contrôler affiche tous les comptes avec le statut validé ou non. Le bouton de validation est géré par javascript qui se charger de récupérer le résultat puis de la réponse avec la méthode axios.

```
const axios = require('axios').default;

function onclickremove(event){

event.preventDefault();

const url = this.href;
```

```
axios.get(url).then(function(response) {

    })
}

document.querySelectorAll('a.js-cancelbook').forEach(function(link){

link.addEventListener('click', onclickremove);

});

function oncheckedremove(event){

    event.preventDefault();

    const url = this.href;

    const span = this.querySelector('span');
    const a = this.querySelector('a');

    axios.get(url).then(function(response) {

        //console.log(response);
        if (span.classList.contains('valide'))
            {span.classList.replace('valide','invalide')
             span.textContent="invalide";
             console.log("pass ici");
             }

        else
            {span.classList.replace('invalide','valide')
            span.textContent="valide";
            console.log("pass ici valide");
        };

        })

    }

document.querySelectorAll('a.js-checked').forEach(function(link){

    link.addEventListener('click', oncheckedremove);

    });
```

Page de la validation des comptes :



Pour la vérification des livres réservés au-delà de 3 jours et où il n'a toujours pas été récupéré par l'inscrit, j'ai fait la création d'un script php autonome qui s'exécutera tous les soirs avec une cron afin de faire cette vérification. Le script parcours l'ensemble de des livres réservés et regarde si la date dépasse 3 jours, dans ce cas le script fait une modification dans la table en changeant la valeur du champ « reserved » à false , la valeur du champ « bookedddate » à NULL, le champ « user_id » à NULL, puis modification dans la table « user » afin de mettre à jour le champ « bookborrowed » pour enlever un livre du nombre total de livres réservés par utilisateur. J'utilise dans ce script L'extension PHP Data Objects (PDO) pour accéder à une base de données depuis PHP.

Script « reservation3days.php «

Pour la vérification empruntée au-delà de 3 semaines, L'utilisateur reçoit un avertissement dans sa fiche « mes livres » sur le livre en question. J'ai aussi fait la création d'un script automatique lancé à partir d'une cron qui vérifie les livres au-delà de 3 semaines, puis ce script envoi un mail à l'inscrit avec le titre du livre qui dépasse la limite d'emprunt de 3 semaines. Le script n'est pas encore fonctionnel car j'ai un problème avec la couche transport
«$transport = new EsmtpTransport('localhost'); » qui permet d'envoyer le mail.

```php
<?php

use Symfony\Component\Mailer\Exception\TransportExceptionInterface;
use Symfony\Component\Mime\Email;
use Symfony\Component\Mailer\Transport\Smtp\EsmtpTransport;
use Symfony\Component\Mailer\Mailer;



$pdo = new PDO('mysql:host=localhost;dbname=mediatheque', 'root', '');
$statement = $pdo-
>prepare('SELECT id,title,isbn FROM book WHERE  bookeddate < now()  - INTERVAL
 21 day AND borrowed = true');

    if ($statement->execute()) {
```

```
        while ($book = $statement->fetch(PDO::FETCH_NUM)) {
            echo '<prebook>';
            print_r($book);
            echo '<afterebook>';

            $getuserid = $pdo-
>prepare("SELECT user_id FROM book where id = $book[0] ");
            $getuserid->execute();
            $userid = $getuserid->fetch(PDO::FETCH_NUM);
            echo '<preuserid>';
            print_r($userid);
            echo '<afteruserid>';

            $getmail = $pdo-
>prepare("SELECT email FROM user where id = $userid[0] ");
            $getmail->execute();
            $emailuser = $getmail->fetch(PDO::FETCH_NUM);
            echo '<pregettotalbookuser>';
            print_r($emailuser);
            echo '</pregettotalbookuser>';

            $email = (new Email())
            ->from('david@marcais.online')
            ->to($emailuser)
            ->subject('depassement emprunt 3 semaines livre ISBN',$book[2])
            -
>text('depassement emprunt, merci de rapporter le livre' ,$book[1], 'le plus r
apidement , Merci');

            try {
                $transport = new EsmtpTransport('localhost');
                $mailer = new Mailer($transport);
                $mailer->send($email);
            } catch (TransportExceptionInterface $e) {
                // some error prevented the email sending; display an
                // error message or try to resend the message
            }
        }

    }
```