



Formation Développeur web et web mobile

Ma bibliothèque en ligne

[site](#)

REMERCIEMENTS

Je tiens à remercier en premier lieu celle qui m'a toujours soutenu ma femme Angelique, ainsi que mes amis et collègues dans cette aventure Abdel et Céline. Merci à Doranco et ses formateurs d'avoir pris autant de temps pour expliquer et détailler afin comprendre comment fonctionne la programmation. Un remerciement aussi envers ma belle soeur qui m'a permis de faire mon stage dans son entreprise. Et je remercie Sepho qui m'a donné l'inspiration de travailler sur le projet d'une bibliothèque en ligne.

Table des matières

Dufour Stéphane	Année 2023	1
sigle , abréviation et glossaire:		4
I - Presentation du projet		5
a) Les besoins fonctionnels		5
b) Les cibles de l'application		5
c) L'organisation		6
d) arborescence du site		7
e) Wireframe		9
II - Langages et Framework		12
III - Back End		14
a) diagramme de Class		16
b) parametrage de la BDD		16
c) Configuration Api platform		20
d) Mettre en place le token :		20
e) Gestion de la sécurité		25
IV - Front End		26
a) Le referencement Seo		27
b) Les obligations à respecter		28
c) design		28
V - la mise en ligne		29
a) La mise en ligne partie Back end :		29
b) La mise en ligne partie Front end :		31
VI - Conclusion et évolution		33

sigle , abréviation et glossaire:

-**isbn**: Numéro international normalisé du livre

-**BDD**: la base de donnée

-**MVC**: le modèle vue contrôleur est une architecture logicielle destiné à l'interface graphique

-**API**: interface de programmation d'application désigné par API "application programming interface " en anglais

-**PHP**: Hypertext Preprocessor.

-**HTML**: HyperText Markup Language.

-**SCSS / SASS**: Syntactically awesome stylesheets.

-**PMA** : phpMyAdmin.

-**Clés SSH** (Secure Shell) : est un protocole de communication sécurisé. Ce protocole de connexion impose un échange de clés de chiffrement en début de connexion.

-**SEO** (en anglais "*Search Engine Optimization*") : Optimisation pour les moteurs de recherche.

I - Presentation du projet

L' application "Ma blibliothèque en ligne" propose d'avoir acces à sa bibliothèque pour connaître les livres qu'on possède à tout moment et partout, le but est de répertorier les livres déjà acquis par l'utilisateur.

Pendant les déplacements et autres, les personnes peuvent mettre à jour leur liste tant qu'il y as une connexion internet.

Cette application est destinée à tous les mordus de lecture.

a) Les besoins fonctionnels

Les visiteurs non inscrits au site peuvent voir la liste des livres.

L'inscription est simple juste avec l'adresse email, le mot de passe et l'acceptation des conditions générale d'utilisation via le formulaire d'inscription.

La connexion a son compte se fait par l'email et le mot de passe, via le formulaire de connexion.

Une personne connectée peut voir sa liste de livre de sa bibliothèque.

Une barre de recherche permet de savoir si le livre est déjà connu dans la base avec son titre ou son isbn. Et apres , l'utilisateur peut l'ajouter à sa liste.

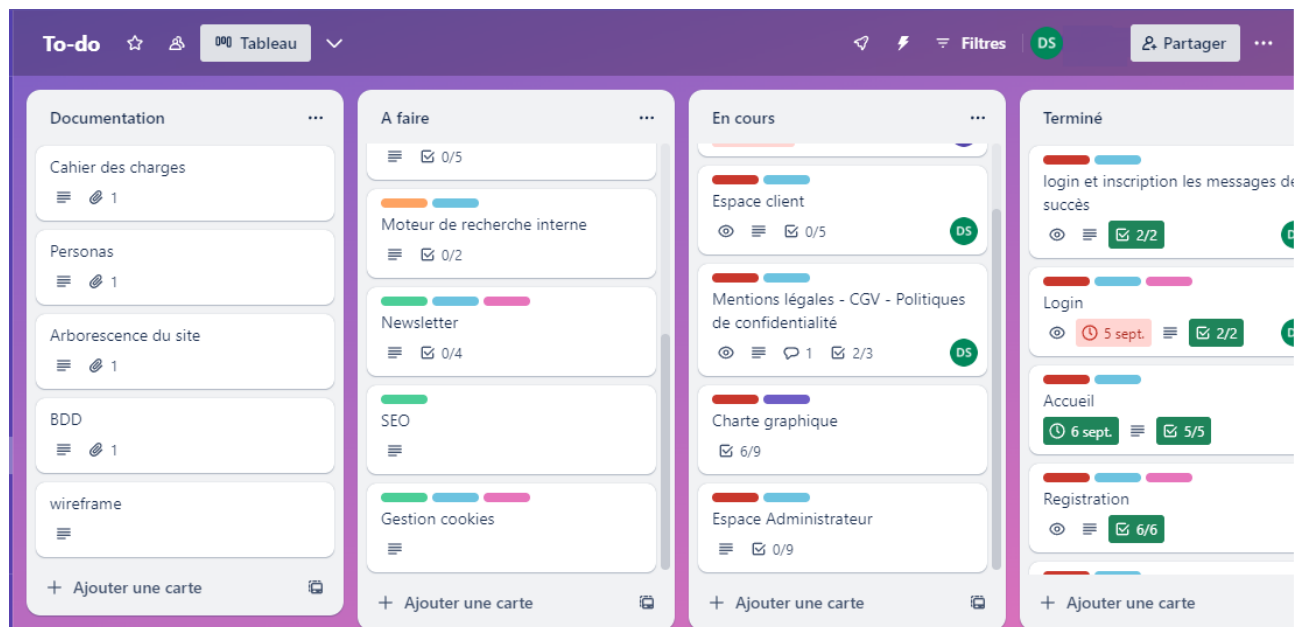
L'utilisateur peut demander l'ajout d'un livre via un formulaire ou la modification d'un livre déjà existant dans la BDD.

b) Les cibles de l'application

L'application est destinée à toute personne de la francophonie.

c) L'organisation

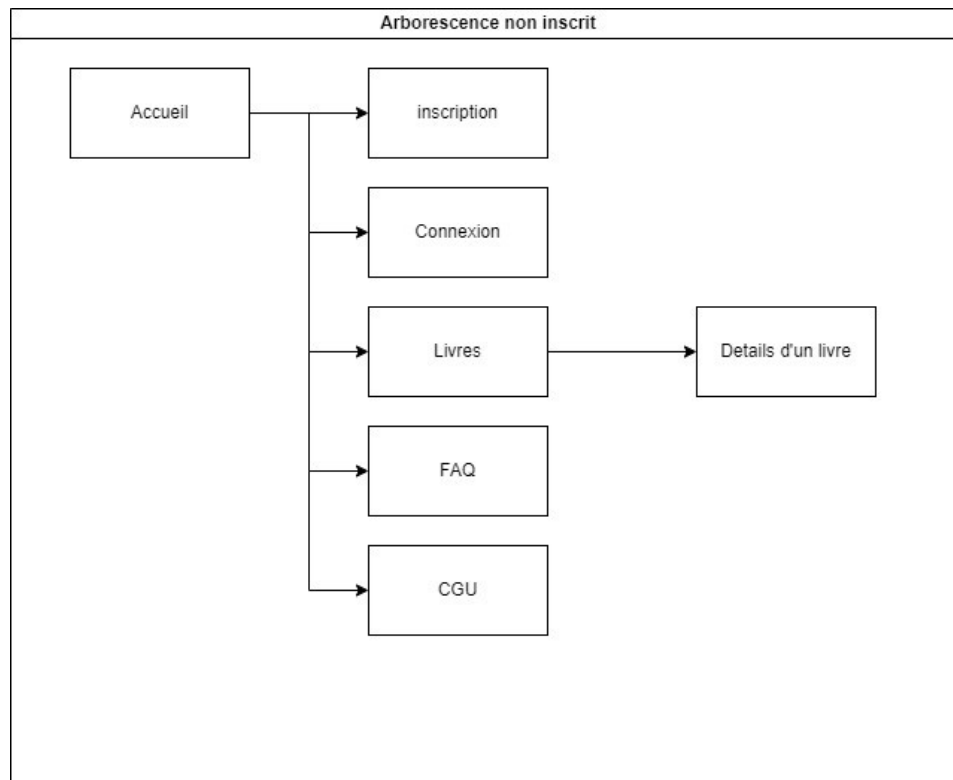
Pendant le projet, j'ai utilisé trello pour mettre en place une organisation pour savoir l'importance des fonctionnalités à développer :



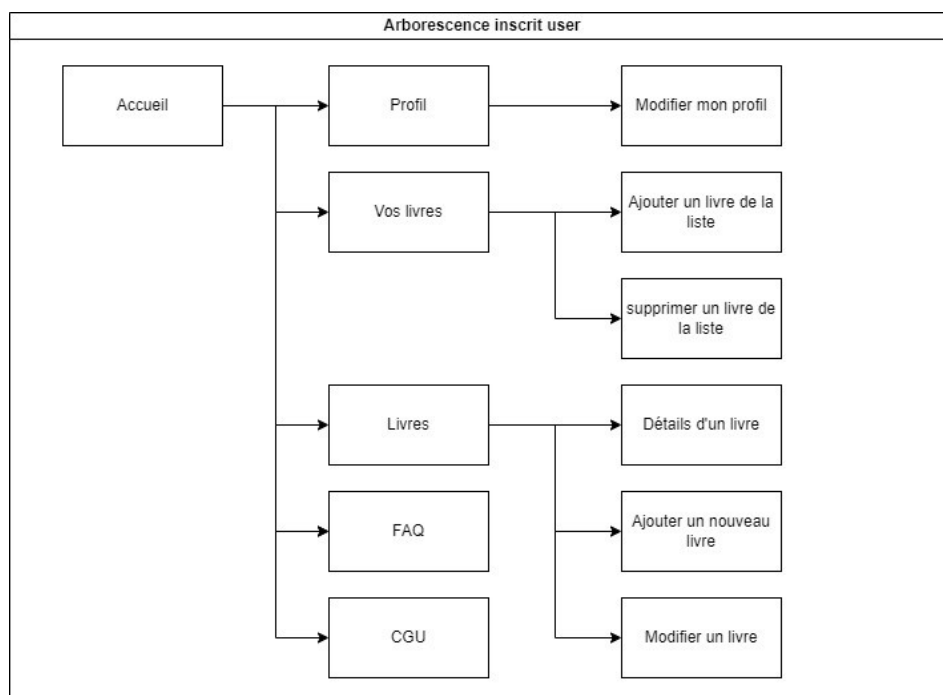
J'ai essayé de suivre au possible les dates pendant la programmation mais sur certaines tâches complexes à mettre en place les dates étaient largement dépasser.

d) arborescence du site

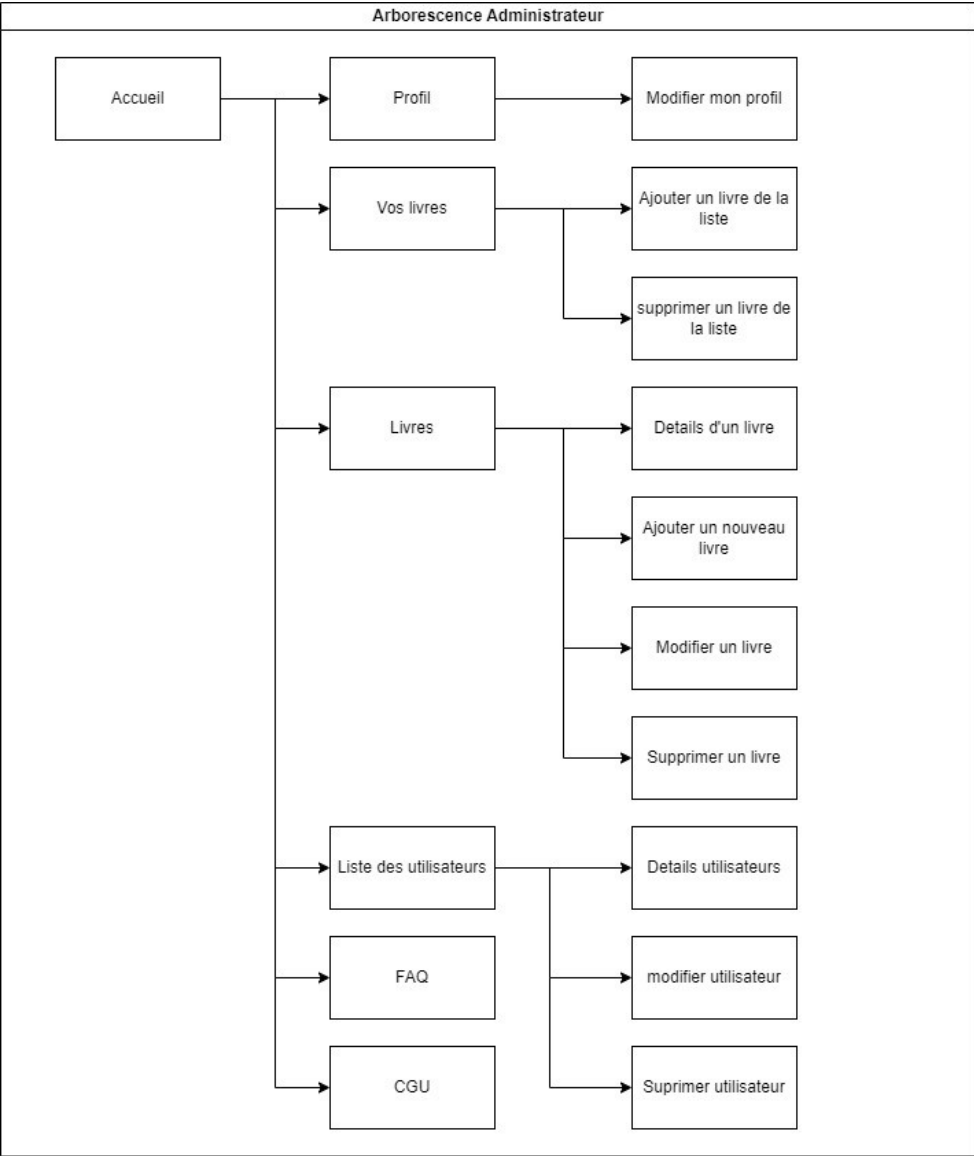
Les visiteurs non inscrits



Les utilisateurs inscrits



Les administrateurs

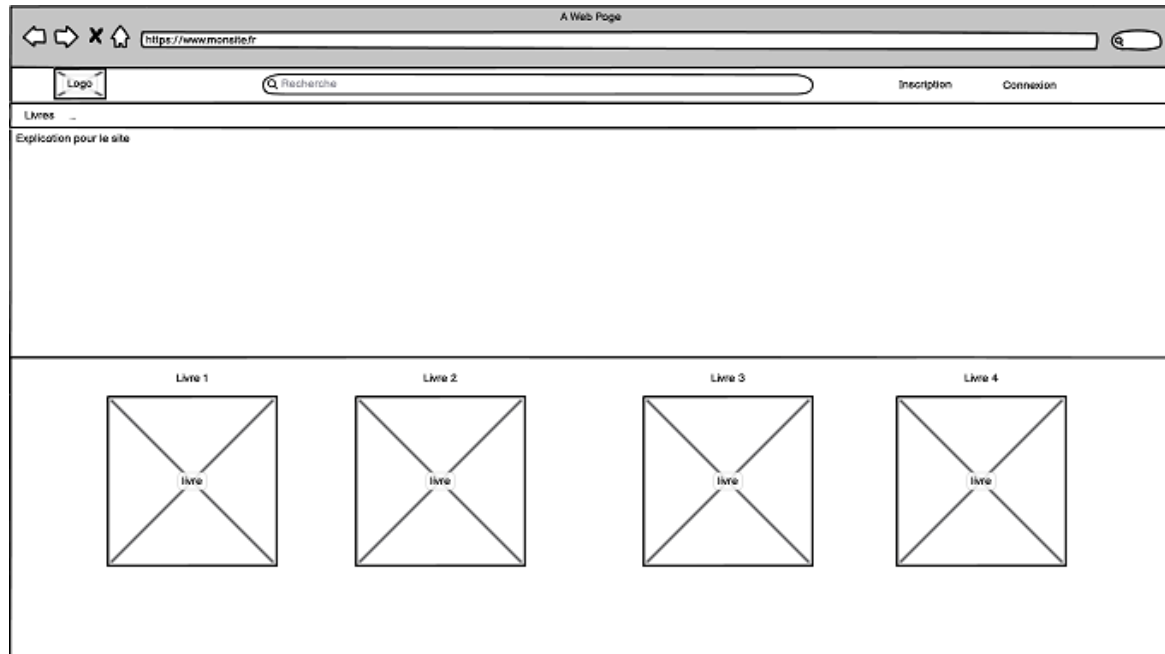


e) Wireframe

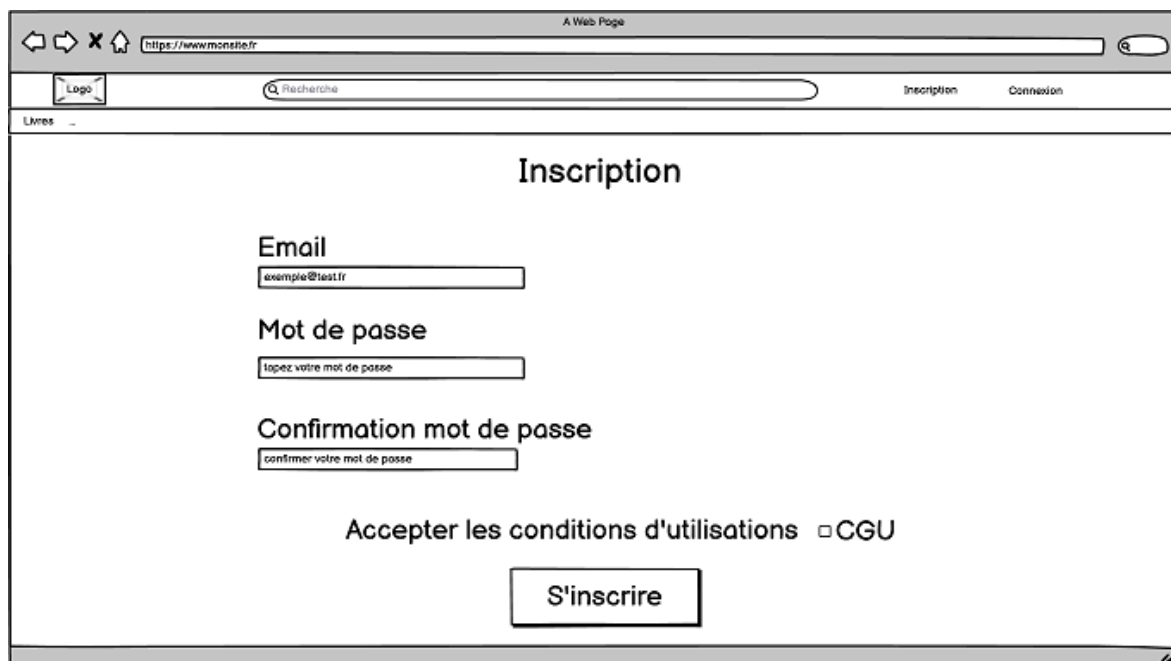
Pour le wireframe je vais utiliser le logiciel balsamiq



Page d'accueil



Page d'inscription



Page de connexion

The screenshot shows a web browser window with the address bar displaying 'https://www.monstie.fr'. The browser's address bar includes a search icon, a search input field with the placeholder 'Recherche', and a search button. The browser's title bar shows 'A Web Page'. The browser's tabs bar shows 'Lignes'. The browser's main content area displays the login page. The page has a header with a 'Logo' button, a search bar, and links for 'Inscription' and 'Connexion'. The main content area is titled 'Connexion' and contains two input fields: 'Email' with the value 'exemple@test.fr' and 'Mot de passe' with the placeholder 'tapez votre mot de passe'. Below the input fields is a 'Se connecter' button.

Connexion

Email
exemple@test.fr

Mot de passe
tapez votre mot de passe

Se connecter

Page du profil de l'utilisateur

The screenshot shows a web browser window with the address bar displaying 'https://www.monstie.fr'. The browser's address bar includes a search icon, a search input field with the placeholder 'Recherche', and a search button. The browser's title bar shows 'A Web Page'. The browser's tabs bar shows 'Lignes'. The browser's main content area displays the user profile page. The page has a header with a 'Logo' button, a search bar, and links for 'Mon compte' and 'Déconnexion'. The main content area is titled 'Mon profil' and contains three input fields: 'Email' with the value 'adresse@email.fr', 'Nom' with the value 'le nom', and 'Prenom' with the value 'Le prenom'. Below the input fields is a 'Modifier' button.

Mon profil

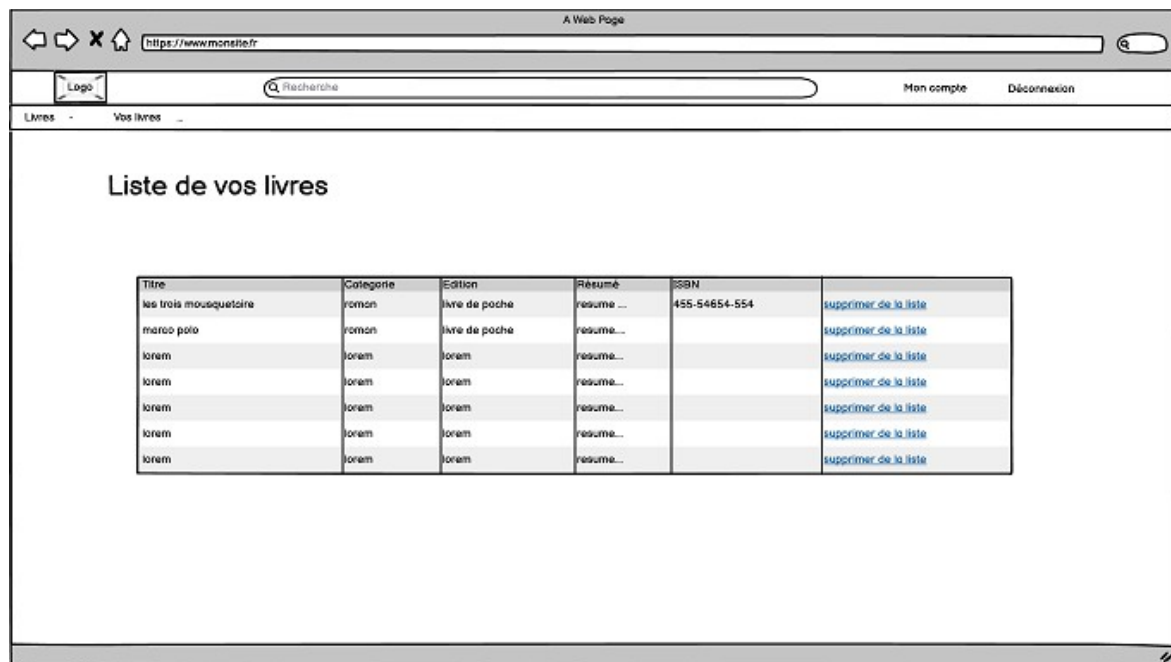
Email
adresse@email.fr

Nom
le nom

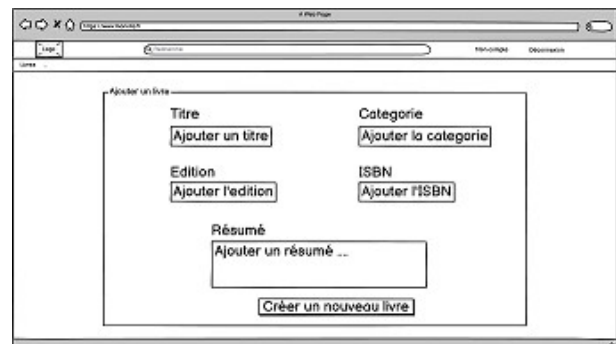
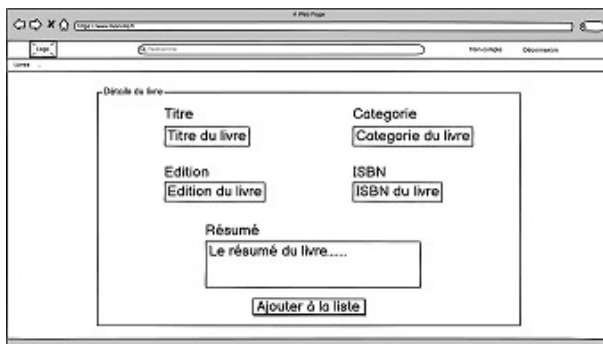
Prenom
Le prenom

Modifier

Page Liste des Livres



Page détails du livre et créer un nouveau livre



Coté responsive

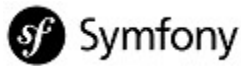


II - Langages et Framework

Back-end : PHP/Symfony, Api platform



PHP est un langage interprété par le serveur web principalement utilisé dans une page web dynamique. Le PHP est orienté objet.



symfony est un ensemble de composants PHP ainsi qu'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. Dans ce projet, je me sers de symfony comme d'une API c'est à dire recevoir des demandes, les traiter et renvoyer une réponse.



API platform est une dépendance pour simplifier les demandes pour les requetes API.

Front-end : HTML, JS/ React , SCSS/Tailwind



HTML 5 est langage de balisage. Ce langage permet de structurer sémantiquement une page web avec des liens Hypertextes.



SCSS est un langage de script pré compilé interprété en CSS.



Tailwind est un logiciel CSS open source utilisant les principes de Bootstrap, pour ajouter un style à chaque élément.



JavaScript est un langage de programmation utilisé pour rendre les pages web dynamique via les actions de l'internaute.



React est une bibliothèque JavaScript libre dont le but est de créer une page unique ou est injecté du contenu partiel via la création de composant dépendant d'un état et générant une page.

Framework : balsamiq, phpMyAdmin



phpMyAdmin est une application web de gestion pour les systèmes de gestion de base de données en MySQL et MariaDB, réalisée principalement en PHP.



Balsamiq est un logiciel servant à créer des wireframes.

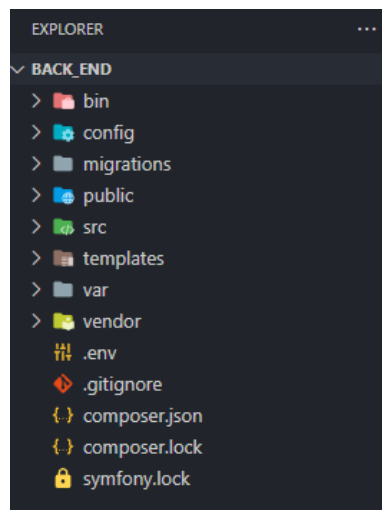
III - Back End

Création du projet avec la commande :

```
MINGW64:/c/Projet_bibliotheque  
DufSt@Pc-Djoudjin MINGW64 /c/Projet_bibliotheque  
$ composer create-project symfony/skeleton:"6.3.*" Back_End
```

la commande génère le projet symfony qui sera utilisé en API.

Structure du back :



Le dossier bin :

Il contient le fichier **console**.

Le fichier **console** comporte tous les exécutables de Symfony, toutes les commandes.

Le dossier config :

Il contient toutes les configurations des packages, services et les routes.

Le dossier public :

C'est le point d'entrée de l'api, il s'agit du dossier qui va communiquer avec le site depuis le navigateur.

Il contient :

- **Fichier index.php**

Les navigateurs web vont pointer sur le fichier pour communiquer.

Le dossier src :

Il s'agit du dossier principal de notre api, il contient l'ensemble de nos sources PHP et il contient les dossiers suivants :

- **Le dossier Controller**

il ne sera pas utilisé pour l'api car nous utilisons Api Platform.

- **Le dossier Entity**

Ce dossier contient les entités qui représentent les tables de notre BDD, elles permettent de mettre à jour notre BDD à chaque modification de l'Entity.

- **Le dossier Migration**

Contient les instructions pour faire nos tables de la BDD.

- **Le dossier vendor**

le dossier stock toutes les dépendances installées.

Ces dépendances sont installées via un gestionnaire "composer" qui permet d'installer des librairies.

- **Le dossier var**

Il est utilisé pour stocker le cache et les fichiers log.

- **Le fichier composer.json**

Il contient la liste des dépendances installées.

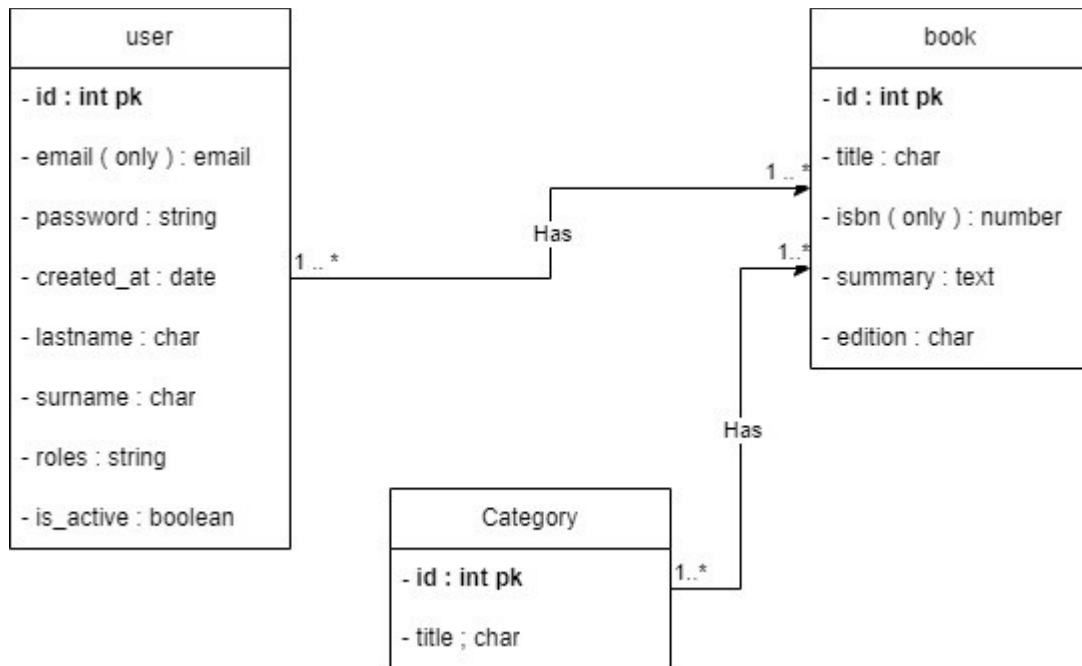
- **Le fichier .env**

Le fichier contient nos variables d'environnement.

On le parametre pour l'accès à la base de données.

a) diagramme de Class

Le diagramme de class pour le projet ma bibliothèque en ligne met en relation les deux tables user et book en Many to Many (car un utilisateur peut avoir une liste de livres mais un livre peut compter plusieurs utilisateurs liés au livre), et il s'agit de la meme relation entre book et category.



b) parametrage de la BDD

Intéressons nous, au fichier .env, changeons la ligne pour creer la BDD avec une commande symfony :

```
> templates
> var
> vendor
> .env
> .gitignore
> composer.json
25 #
26 # DATABASE_URL="sqlite:///kernel.project_dir%/var/data.db"
27 DATABASE_URL=mysql://root:@127.0.0.1:3306/librairy
28 # DATABASE_URL="mysql://app:!ChangeMe!@127.0.0.1:3306/app?serverVersion=10.11.2-MariaDB&charset=utf8mb4"
29 # DATABASE_URL="postgresql://app:!ChangeMe!@127.0.0.1:5432/app?serverVersion=15&charset=utf8"
30 ###< doctrine/doctrine-bundle ###
```

la commande symfony permettant la creation de la BDD

```
MINGW64:/c/Projet_bibliotheque/back_end
DufSt@Pc-Djoudjin MINGW64 /c/Projet_bibliotheque/back_end
$ symfony console doctrine:database:create
```


Création de deux entités :

L'entité user avec la commande :

```
MINGW64:/c/Projet_bibliotheque/back_end  
DufSt@Pc-Djoudjin MINGW64 /c/Projet_bibliotheque/back_end  
$ symfony console make:user
```

Et ensuite répondre au questionnaire de symfony :

```
MINGW64:/c/Projet_bibliotheque/back_end  
DufSt@Pc-Djoudjin MINGW64 /c/Projet_bibliotheque/back_end  
$ symfony console make:user  
  
The name of the security user class (e.g. User) [User]:  
> User  
  
Do you want to store user data in the database (via Doctrine)? (yes/no) [yes]:  
> yes  
  
Enter a property name that will be the unique "display" name for the user (e.g. email, username, uuid) [email]:  
>  
  
Will this app need to hash/check user passwords? Choose No if passwords are not needed or will be checked/hashed  
Does this app need to hash/check user passwords? (yes/no) [yes]:  
> yes
```

Une entité User a été créée, ensuite on lui affecte plusieurs variables de plus comme ;

- create_at pour la date de creation
- lastname pour le nom de famille
- firstname pour le prenom
- isActive pour activer le compte

L'entité Book avec la commande :

```
MINGW64:/c/Projet_bibliotheque/back_end  
DufSt@Pc-Djoudjin MINGW64 /c/Projet_bibliotheque/back_end  
$ symfony console make:entity  
  
Class name of the entity to create or update (e.g. GrumpyGnome):  
> Book  
  
New property name (press <return> to stop adding fields):  
> title
```

A l'entité Book on rajoute les variables suivante :

- isbn du livre
- summary pour un résumé
- edition du livre
- category pour le genre du livre

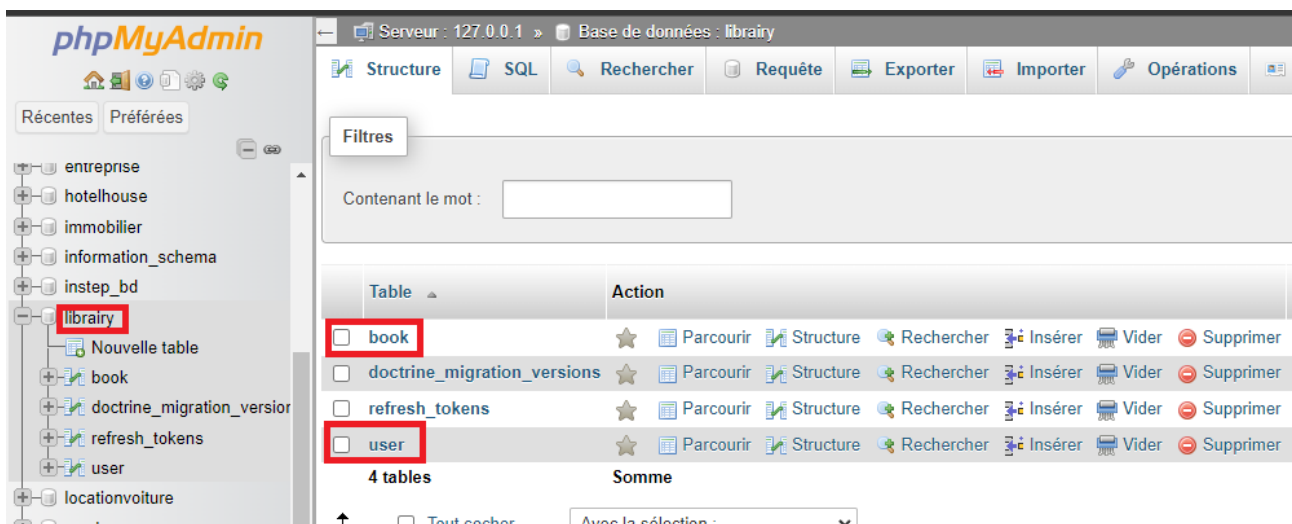
Pour mettre en place les entités dans la BDD, on crée le fichier de migration avec la commande suivante :

```
MINGW64:/c/Projet_bibliotheque/back_end  
DufSt@Pc-Djoudjin MINGW64 /c/Projet_bibliotheque/back_end  
$ symfony console make:migration
```

Ensuite on crée les tables dans la BDD avec la commande suivante :

```
MINGW64:/c/Projet_bibliotheque/back_end  
DufSt@Pc-Djoudjin MINGW64 /c/Projet_bibliotheque/back_end  
$ symfony console doctrine:migrations:migrate
```

Grace à Wampserver on vérifie si les bases sont créés :



Créons un lien de relation Many To Many entre la table User et la table Book, grâce aux commandes de symfony :

```

MINGW64:/c/Projet_bibliotheque/back_end

DufSt@Pc-Djoudjin MINGW64 /c/Projet_bibliotheque/back_end
$ symfony console make:entity User
Your entity already exists! So let's add some new fields!

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

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

What class should this entity be related to?:
> Book

What type of relationship is this?
-----
Type           Description
-----
ManyToOne      Each User relates to (has) one Book.
                Each Book can relate to (can have) many User objects.

OneToMany      Each User can relate to (can have) many Book objects.
                Each Book relates to (has) one User.

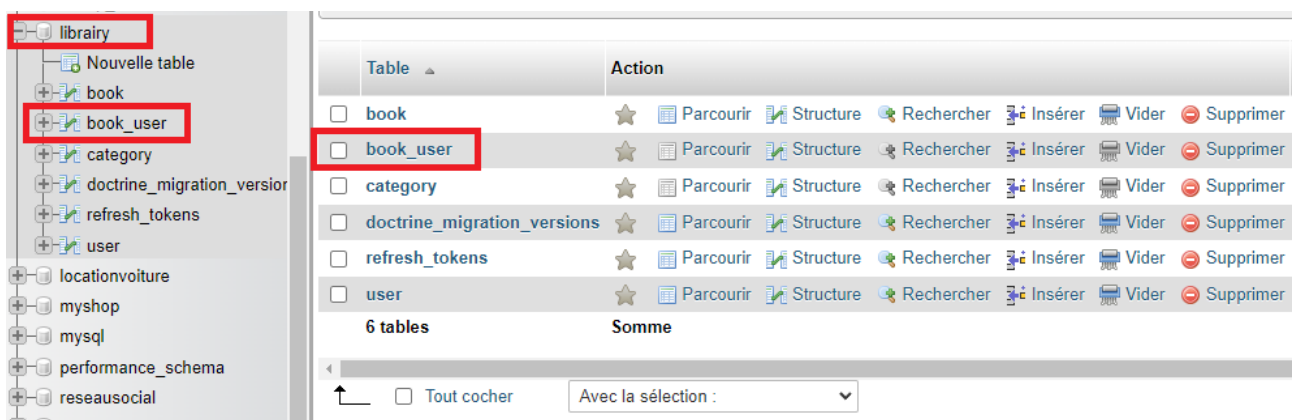
ManyToMany     Each User can relate to (can have) many Book objects.
                Each Book can also relate to (can also have) many User objects.

OneToOne       Each User relates to (has) exactly one Book.
                Each Book also relates to (has) exactly one User.
-----

Relation type? [ManyToOne, OneToMany, ManyToMany, OneToOne]:
> ManyToMany

```

Après avoir fait une migration nous avons une nouvelle table créée mettant en relation la table User et la table Book :



En créant ce lien on pourra récupérer les utilisateurs liés à des livres et inversement.

c) Configuration Api platform

La commande pour installer api platform : permet de simplifier l'utilisation d'une api.

```
MINGW64:/c/Projet_bibliotheque/back_end
DufSt@Pc-Djoudjin MINGW64 /c/Projet_bibliotheque/back_end
$ symfony composer require api|
```

Suivre les étapes fournies par Api platform sur leur site : [documentation Api Platform](#).

d) Mettre en place le token :

Pour la mise en place du token, il faut installer la dépendance lexik/jwt-authentication-bundle, servant pour la sécurité, permettant d'authentifier un utilisateur inscrit qui pourra avoir certain pouvoir que les visiteurs n'auront pas.

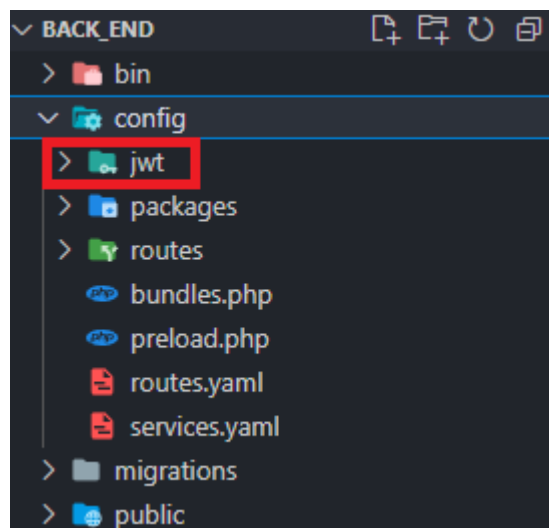
```
DufSt@Pc-Djoudjin MINGW64 /c/Projet_bibliotheque/back_end
$ composer require lexik/jwt-authentication-bundle
```

I – Creation du dossier jwt

Avec la commande suivant on crée le dossier jwt dans le dossier config du projet

```
DufSt@Pc-Djoudjin MINGW64 /c/Projet_bibliotheque/back_end
$ mkdir -p config/jwt
```

voici la partie back end :



II – générer les clés SSH

Regardons le fichier .env, qui a été crée avec la commande pour installer la dépendance lexik/jwt-authentication-bundle :

```
34  ###< nejmio/cors-bundle ###
35
36  ###> lexik/jwt-authentication-bundle ###
37  JWT_SECRET_KEY=%kernel.project_dir%/config/jwt/private.pem
38  JWT_PUBLIC_KEY=%kernel.project_dir%/config/jwt/public.pem
39  JWT_PASSPHRASE=yourpassword
40  ###< lexik/jwt-authentication-bundle ###
41
```

Modifions la ligne JWT_PASSPHRASE par notre mot de passe :

```
36  ###> lexik/jwt-authentication-bundle ###
37  JWT_SECRET_KEY=%kernel.project_dir%/config/jwt/private.pem
38  JWT_PUBLIC_KEY=%kernel.project_dir%/config/jwt/public.pem
39  JWT_PASSPHRASE=power
40  ###< lexik/jwt-authentication-bundle ###
41
```

Avec la commande suivante on génère la clé privée et publique :

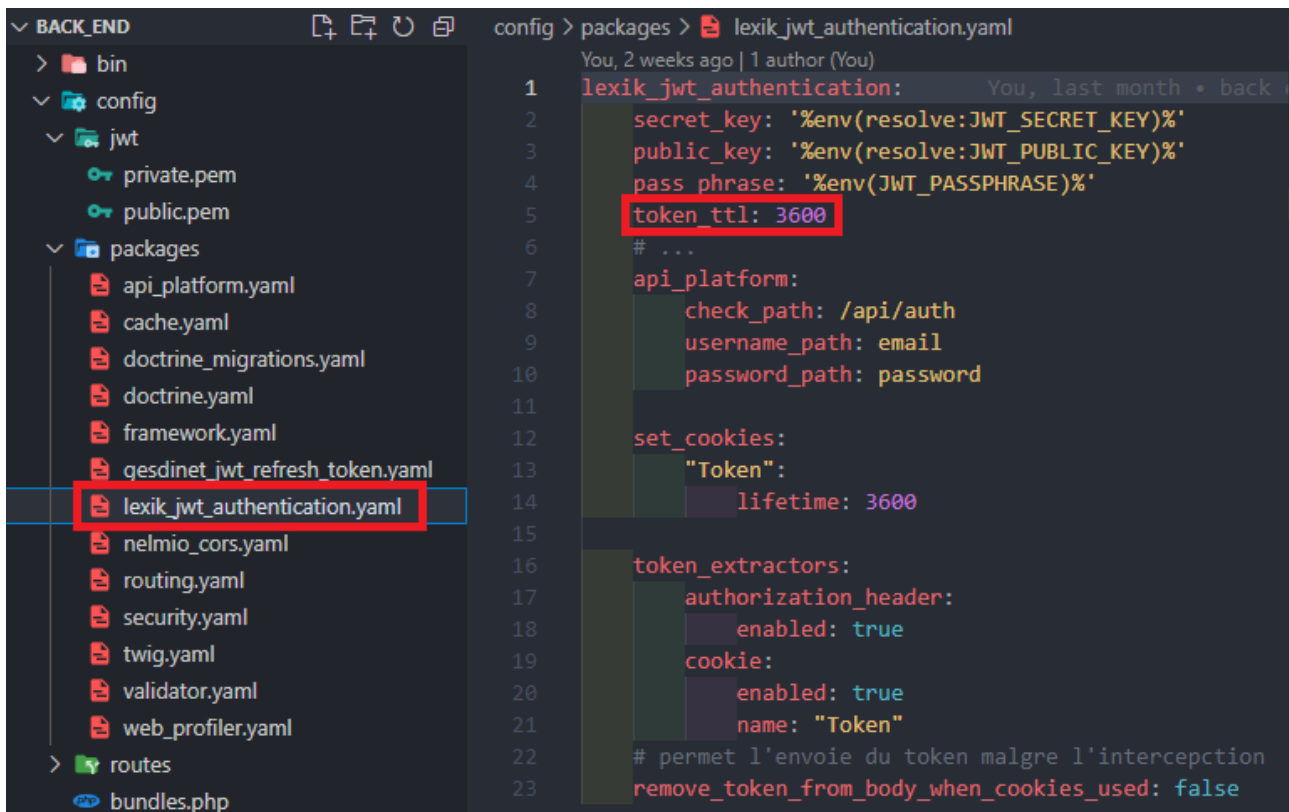
```
MINGW64:/c:/Projet_bibliotheque/back_end
DufSt@Pc-Djoudjin MINGW64 /c:/Projet_bibliotheque/back_end
$ php bin/console lexik:jwt:generate-keypair
```

Ci desous, les clés générées :

```
EXPLORER
BACK_END
  bin
  config
    jwt
      private.pem
      public.pem
  packages
  routes
  bundles.php
  preload.php
  routes.yaml
  services.yaml
  migrations
  public
  src
  templates
  var
  vendor
  .env

config > jwt > private.pem
1  -----BEGIN RSA PRIVATE KEY-----
2  Proc-Type: 4, ENCRYPTED
3  DEK-Info: AES-256-CBC, 5BD28BF6F1392552FC896681D8277AD3
4
5  zEvmN29QNUOeI1Wpk+hsMN79Jr6BQQtSnBblzU8COG0BpwEga+KxS+QJ1XFgff8i
6  h0IgKFfmpABIHh387hJ67EABzS9jD+OZmON6yrF+uQAqupzI2eN3pAKHEti+o2IP
7  3w3JfU1jgwSouONqsyfGpsMVgvD/+oc4Yzv4yslMmXmUH1SvnkuxzcIO1F+DdB53
8  e/Mubip1thyws9o/bpL2UTBC35oqno0N0lx3gqba6Bm0J48Ck+2mtqt1ofR33hEA
9  Iy2cQAHCOfczHhWj80ry0gEvETGygT0ZccJW/LeyGd8RNafwN4MQuoUPqD6JKJau
10 vUkITRY1uiZ577/21873aFkNfWInHtklShgWnCXYz+godzouxFukXBmH9rrtfs73
11 iZw0L1D+sHvsUh5QaSItr3XPleT4Sf7Mu6Tqh700YcRxQoxqpKySoneSoPoE4q
12 cmN45H+DmZ2dQ9fg0ICvkP4fv7ys0Iq6pc1JMPITH4ZcvY0f5rbmw4MHOQ5ne8dA
13 7yIrAVvDcxenOqWgYEK3md2HsvWJZlW2uzAvWwRj5xX6GTBpxyagPha7xNT4waUJ
14 I/6Y/JUOqdAIEFsirNsphyPKGtgrZAD0bdP7GwH+jwDL/5jiDaCNjBcVuD3/BN1j
15 V34Ad822QICy7gS3TUL1fk02cvVo80AN3ITxat+FxqeBUU5YjVF6ek8TV2A3AxW4
16 5T9qA13AkKJfFxxk1KzcjF/Rq3mJ5BxHj5iLWn1yhS1FxxMg8dbgdIEv1d90wBHFw
17 wIU02j3qwalYD1lKUP2BHK3w16u7/4Wr8f9E/cVG9fJ1NL98B4is7EjBL12vk3+J
18 wLxAhh9z/GIS/dErz+cwMTSc9FYcQ0TJWFqHkLEgm+PVLr22jNnENow40zLsE+t7
19 DgIeOhC8sdM1qLXzjak+eNY2XpBKaoUrfdqcP8R4jv/Ewsr2jSsBp33sTIk5H0o1
20 yW4t9LVhV3gN2DXjc54pdQK+jwlaNS16717Jq6P1KtIjoGLNIqhFjFFZau9M35TL
21 98ykavkhc0Z0u+SxTXLyp+oY0uPIYdJ5o5Fki8XMMVezPB0L7ZDyq5HIVH5DHPYB
```

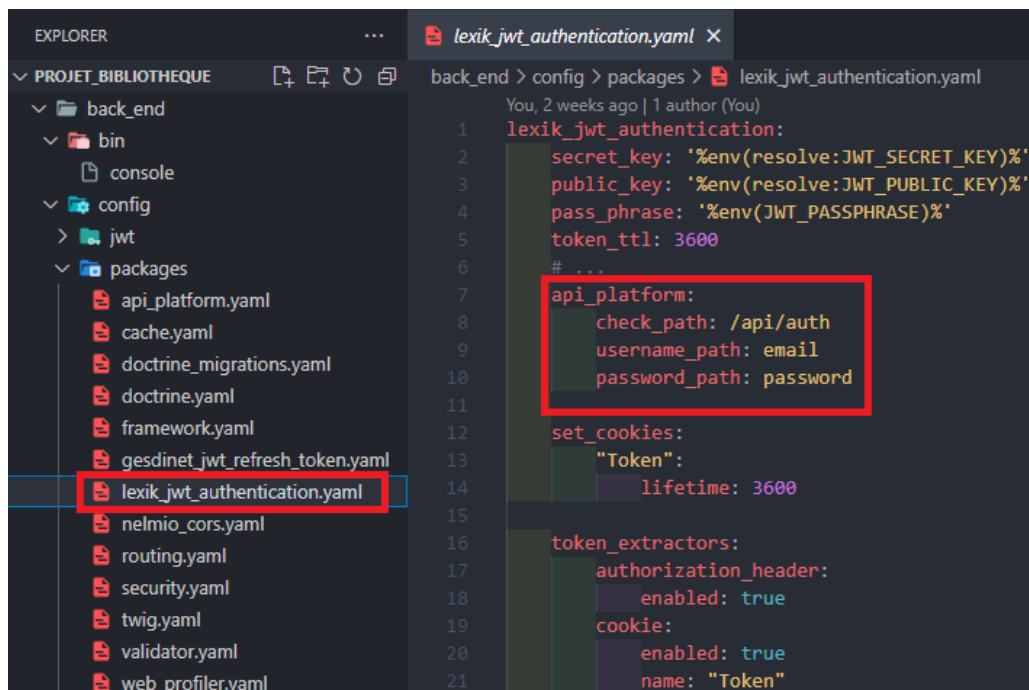
Nous pouvons mettre aussi en place un temps pendant laquelle le token sera valide pour le projet et les tests. J'ai mis un temps de 3600 secondes c'est à dire une heure et nous pouvons le modifier dans ce fichier



```
1 lexik_jwt_authentication:
2     secret_key: '%env(resolve:JWT_SECRET_KEY)%'
3     public_key: '%env(resolve:JWT_PUBLIC_KEY)%'
4     pass_phrase: '%env(JWT_PASSPHRASE)%'
5     token_ttl: 3600
6     # ...
7     api_platform:
8         check_path: /api/auth
9         username_path: email
10        password_path: password
11
12    set_cookies:
13        "Token":
14            lifetime: 3600
15
16    token_extractors:
17        authorization_header:
18            enabled: true
19        cookie:
20            enabled: true
21            name: "Token"
22    # permet l'envoi du token malgré l'interception
23    remove_token_from_body_when_cookies_used: false
```

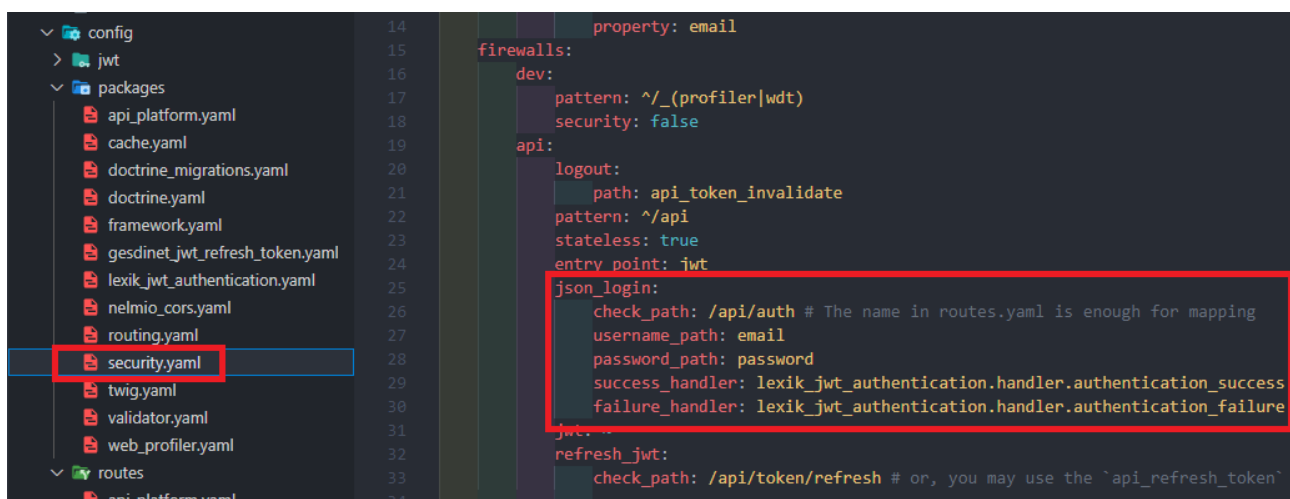
Continuons en mettant en place les chemins d'authentification pour api platform

Dans le fichier lexik_jwt_authentication.yaml :



```
1 lexik_jwt_authentication:
2     secret_key: '%env(resolve:JWT_SECRET_KEY)%'
3     public_key: '%env(resolve:JWT_PUBLIC_KEY)%'
4     pass_phrase: '%env(JWT_PASSPHRASE)%'
5     token_ttl: 3600
6     # ...
7     api_platform:
8         check_path: /api/auth
9         username_path: email
10        password_path: password
11
12    set_cookies:
13        "Token":
14            lifetime: 3600
15
16    token_extractors:
17        authorization_header:
18            enabled: true
19        cookie:
20            enabled: true
21            name: "Token"
```

Dans le fichier security.yaml:

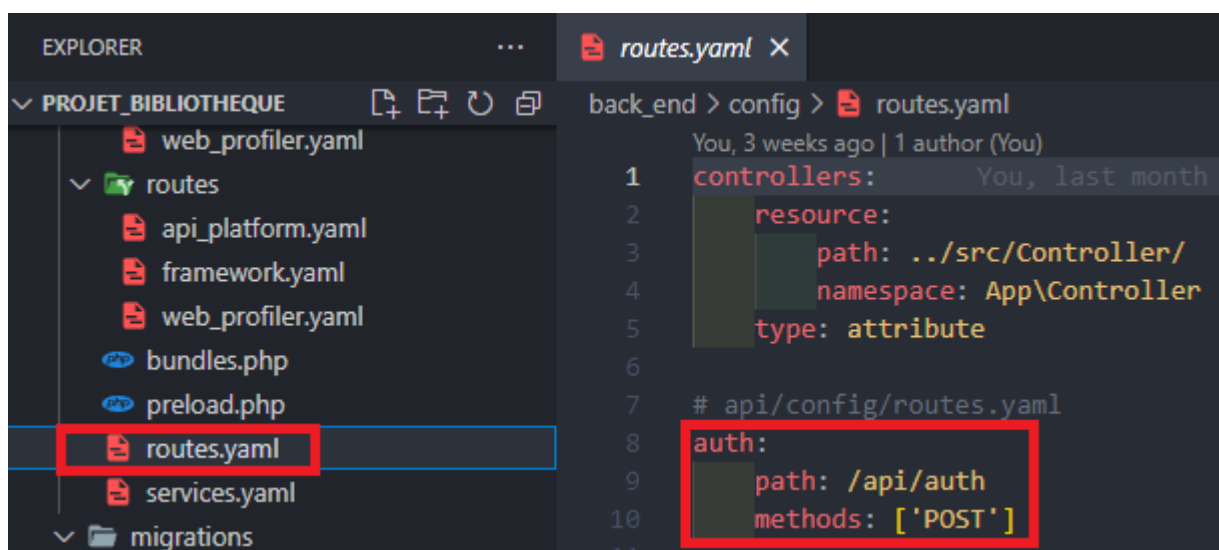


The screenshot shows the security.yaml file in a code editor. The left sidebar displays a file explorer with a tree structure: config, jwt, packages (containing api_platform.yaml, cache.yaml, doctrine_migrations.yaml, doctrine.yaml, framework.yaml, gesdinet_jwt_refresh_token.yaml, lexik_jwt_authentication.yaml, nelmio_cors.yaml, routing.yaml, security.yaml, twig.yaml, validator.yaml, and web_profiler.yaml), and routes (containing api_platform.yaml). The security.yaml file is highlighted in the sidebar. The main editor shows the configuration for the 'api' entry point, with the 'json_login' section highlighted by a red box. The 'json_login' section includes 'check_path', 'username_path', 'password_path', 'success_handler', and 'failure_handler'.

```
14     property: email
15     firewalls:
16     dev:
17         pattern: ^/_(profiler|wdt)
18         security: false
19     api:
20         logout:
21             path: api_token_invalidate
22         pattern: ^/api
23         stateless: true
24         entry_point: jwt
25     json_login:
26         check_path: /api/auth # The name in routes.yaml is enough for mapping
27         username_path: email
28         password_path: password
29         success_handler: lexik_jwt_authentication.handler.authentication_success
30         failure_handler: lexik_jwt_authentication.handler.authentication_failure
31     jwt:
32     refresh_jwt:
33         check_path: /api/token/refresh # or, you may use the `api_refresh_token`
```

Enfin dans la routes.yaml

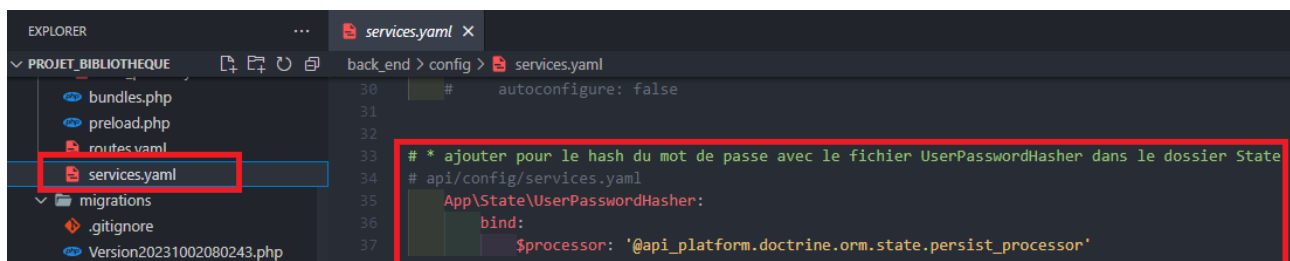
il faut changer aussi le chemin pour qu'il soit cohérent avec les autres changements :



The screenshot shows the routes.yaml file in a code editor. The left sidebar displays a file explorer with a tree structure: PROJÉT_BIBLIOTHEQUE (containing web_profiler.yaml, routes, api_platform.yaml, framework.yaml, web_profiler.yaml, bundles.php, preload.php, routes.yaml, services.yaml, and migrations). The routes.yaml file is highlighted in the sidebar. The main editor shows the configuration for the 'auth' resource, with the 'auth' section highlighted by a red box. The 'auth' section includes 'path' and 'methods'.

```
1 You, 3 weeks ago | 1 author (You)
2 controllers: You, last month
3 resource:
4     path: ../src/Controller/
5     namespace: App\Controller
6     type: attribute
7 # api/config/routes.yaml
8 auth:
9     path: /api/auth
10    methods: ['POST']
```

Ensuite nous allons passer au hashage du mot de passe pour qu'il soit stocké et crypté :



The screenshot shows the services.yaml file in a code editor. The left sidebar displays a file explorer with a tree structure: PROJÉT_BIBLIOTHEQUE (containing bundles.php, preload.php, routes.yaml, services.yaml, migrations, .gitignore, and Version20231002080243.php). The services.yaml file is highlighted in the sidebar. The main editor shows the configuration for the 'App\\State\\UserPasswordHasher' service, with the 'bind' section highlighted by a red box. The 'bind' section includes 'bind' and '\$processor'.

```
30 # autoconfigure: false
31
32
33 # * ajouter pour le hash du mot de passe avec le fichier UserPasswordHasher dans le dossier State
34 # api/config/services.yaml
35 App\\State\\UserPasswordHasher:
36     bind:
37         $processor: '@api_platform.doctrine.orm.state.persist_processor'
```

il faut rajouter le fichier UserPasswordHasher dans le dossier state qu'il faut créer aussi :

The screenshot shows the VS Code Explorer on the left with the 'State' directory highlighted in red. The main editor displays the code for 'UserPasswordHasher.php'. The code includes a namespace declaration, imports for 'ProcessorInterface' and 'UserPasswordHasherInterface', and a final class 'UserPasswordHasher' that implements 'ProcessorInterface'. The class has a constructor and a 'process' method that handles password hashing and token generation.

```
1 <?php
2 You, last month | 1 author (You)
3
4 /**
5  * fichier ajouter pour la hash du mot de passe les Plainpasswords ont été remplacé par Password
6  */
7 # api/src/State/UserPasswordHasher.php
8 namespace App\State;
9 use ApiPlatform\Metadata\Operation;
10 use ApiPlatform\State\ProcessorInterface;
11 use Symfony\Component\PasswordHasher\Hasher\UserPasswordHasherInterface;
12 final class UserPasswordHasher implements ProcessorInterface
13 {
14     public function __construct(private readonly ProcessorInterface $processor, private readonly UserPasswordHasherInterface $passwordHasher)
15     {
16     }
17     public function process($data, Operation $operation, array $uriVariables = [], array $context = [])
18     {
19         if (!$data->getPassword()) {
20             return $this->processor->process($data, $operation, $uriVariables, $context);
21         }
22         $hashedPassword = $this->passwordHasher->hashPassword(
23             $data->getPassword()
24         );
25         $data->setPassword($hashedPassword);
26         $data->eraseCredentials();
27         return $this->processor->process($data, $operation, $uriVariables, $context);
28     }
29 }
```

En réussissant une authentification, on peut renvoyer des informations via une interception :

The screenshot shows the VS Code Explorer on the left with 'security.yaml' highlighted in red. The main editor displays the configuration for 'security.yaml'. The configuration includes an 'entry_point' of 'jwt', a 'json_login' section with 'check_path' and 'password_path', and 'success_handler' and 'failure_handler' set to 'lexik_jwt_authentication.handler.authentication_success' and 'lexik_jwt_authentication.handler.authentication_failure' respectively. The 'jwt' section is also configured with 'refresh_jwt' and 'check_path'.

```
24 entry_point: jwt
25 json_login:
26     check_path: /api/auth # The name in routes.yaml is enough for mapping
27     username_path: email
28     password_path: password
29     success_handler: lexik_jwt_authentication.handler.authentication_success
30     failure_handler: lexik_jwt_authentication.handler.authentication_failure
31 jwt: ~
32 refresh_jwt:
33     check_path: /api/token/refresh # or, you may use the 'api refresh token'
```

L'interception :

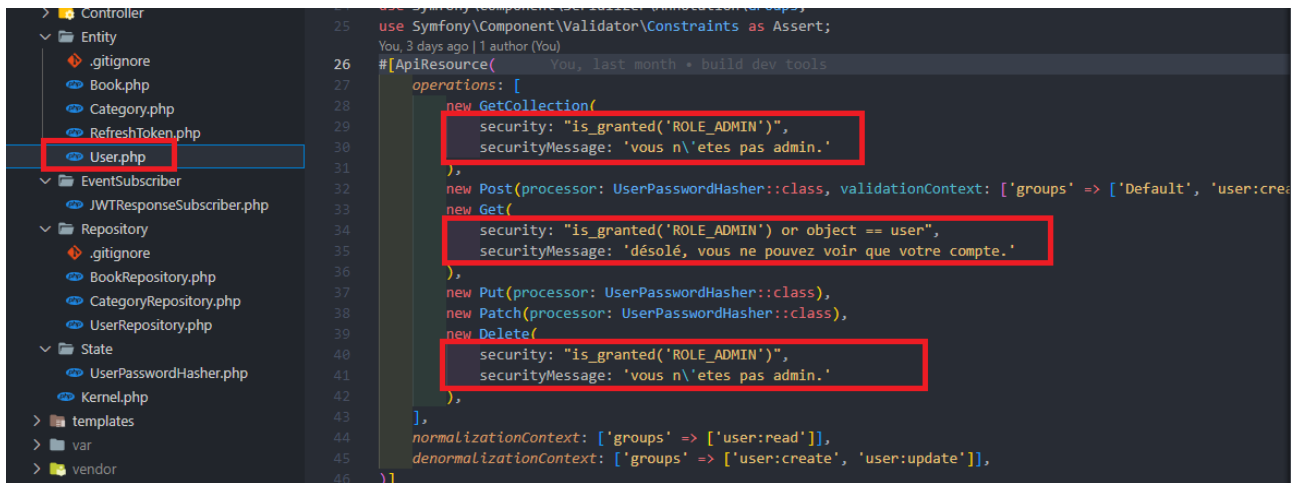
The screenshot shows the VS Code Explorer on the left with 'JWTResponseSubscriber.php' highlighted in red. The main editor displays the code for 'JWTResponseSubscriber.php'. The code includes a class 'JWTResponseSubscriber' that implements 'EventSubscriberInterface'. The class has a 'getSubscribedEvents' method and an 'onAuthenticationSuccessResponse' method. The 'onAuthenticationSuccessResponse' method extracts user information from the event and sets it on the response data. A red box highlights the data array being set, and a note on the right explains that this information is sent back to the user when they are connected.

```
15 You, last month | 1 author (You)
16 class JWTResponseSubscriber implements EventSubscriberInterface
17 {
18     public static function getSubscribedEvents()
19     {
20         return [
21             'lexik_jwt_authentication.on_authentication_success' => ['onAuthenticationSuccessResponse', 200],
22         ];
23     }
24     public function onAuthenticationSuccessResponse(AuthenticationSuccessEvent $event)
25     {
26         $data = $event->getData();
27         $user = $event->getUser();
28
29         if (!$user instanceof UserInterface) {
30             return;
31         }
32
33         $data['user'] = [
34             'roles' => $user->getRoles(),
35             'identity' => $user->getUserIdentifier(),
36             'id' => $user->getId(),
37         ];
38
39         $event->setData($data);
40     }
41 }
42 }
```

Les informations renvoyer quand une utilisateur est connecté

e) Gestion de la sécurité

Grace aux roles des utilisateurs, on peut ajouter une sécurité pour les différentes actions avec api platform :

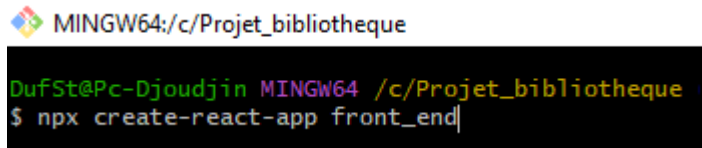


```
25 use Symfony\Component\Validator\Constraints as Assert;
26 #ApiResource(
27     operations: [
28         new GetCollection(
29             security: "is_granted('ROLE_ADMIN')",
30             securityMessage: 'vous n\'etes pas admin.'
31         ),
32         new Post(processor: UserPasswordHasher::class, validationContext: ['groups' => ['Default', 'user:create']],
33         new Get(
34             security: "is_granted('ROLE_ADMIN') or object == user",
35             securityMessage: 'désolé, vous ne pouvez voir que votre compte.'
36         ),
37         new Put(processor: UserPasswordHasher::class),
38         new Patch(processor: UserPasswordHasher::class),
39         new Delete(
40             security: "is_granted('ROLE_ADMIN')",
41             securityMessage: 'vous n\'etes pas admin.'
42         ),
43     ],
44     normalizationContext: ['groups' => ['user:read']],
45     denormalizationContext: ['groups' => ['user:create', 'user:update']],
46 )
```

IV - Front End

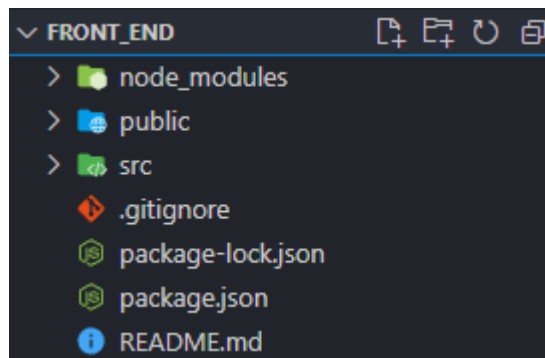
Pour la partie front end de notre site, nous allons utiliser React avec les dépendances suivantes : react-router-dom, Formik, Yup, axios et tailwind.

Creation du projet avec la commande suivante :



```
MINGW64:/c/Projet_bibliotheque
DufSt@Pc-Djoudjin MINGW64 /c/Projet_bibliotheque
$ npx create-react-app front_end
```

la commande crée les dossiers suivants :



Le dossier node_module :

il est un dossier qui est la source des dépendances utilisées dans le projet NodeJS.

Le dossier public :

C'est le point d'entrée du site, il s'agit du dossier qui va communiquer avec l'ensemble du site depuis le navigateur.

Le dossier src :

Il s'agit du dossier principal du site, il contient l'ensemble de nos pages web codées en REACT et il contient les dossiers suivants :

- Le dossier assets :

Ce dossier contient l'ensemble des images pour l'apparence du site dont le logo.

- Le dossier components :

Il est fait pour contenir les différents composants utilisés pour le site.

- Le dossier context :

Le dossier context est fait pour actualiser le site par rapport à un visiteur d'un membre inscrit.

- Le dossier page :

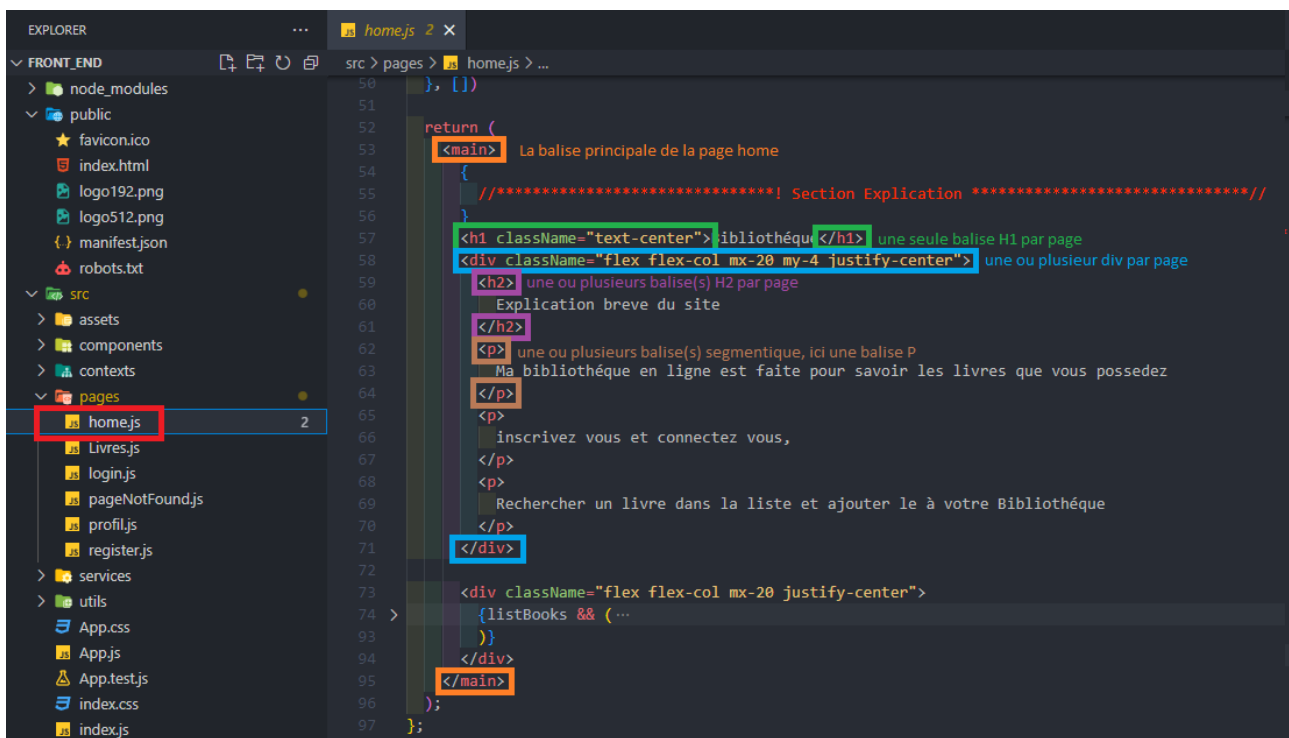
Il contient l'ensemble des pages web du site.

- Le dossier utils :

Il sert pour les constantes du site.

a) Le référencement Seo

L'optimisation pour les moteurs de recherche, appelé aussi le **SEO** (en anglais "*Search Engine Optimization*"). Il inclut l'ensemble des techniques visant à améliorer le positionnement d'une page, d'un site dans la page de résultats d'un moteur de recherche. L'utilisation des bonnes balises HTML est importante ainsi que les balises meta.



```
50 }, {})  
51  
52 return (  
53   <main> La balise principale de la page home  
54   {  
55     //*****! Section Explication *****//  
56   }  
57   <h1 className="text-center"> bibliothèque </h1> une seule balise H1 par page  
58   <div className="flex flex-col mx-20 my-4 justify-center"> une ou plusieurs div par page  
59     <h2> une ou plusieurs balise(s) H2 par page  
60     Explication breve du site  
61     </h2>  
62     <p> une ou plusieurs balise(s) segmentique, ici une balise P  
63     Ma bibliothèque en ligne est faite pour savoir les livres que vous possédez  
64     </p>  
65     <p>  
66     inscrivez vous et connectez vous,  
67     </p>  
68     <p>  
69     Rechercher un livre dans la liste et ajouter le à votre Bibliothèque  
70     </p>  
71   </div>  
72  
73   <div className="flex flex-col mx-20 justify-center">  
74     {listBooks && (...  
75     )}  
76   </div>  
77 </main>  
78 );
```

b) Les obligations à respecter

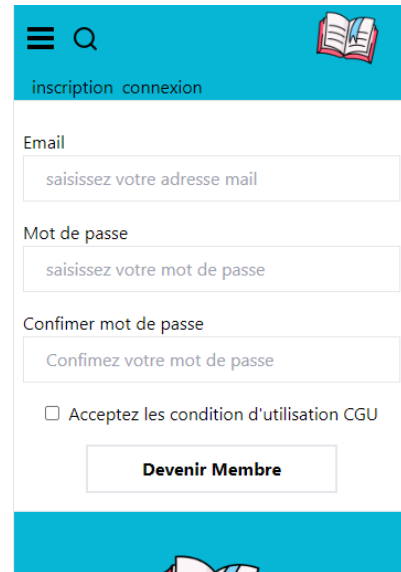
Par rapport à la loi, on ne doit pas tracer une personne avec les cookies ou autres informations stockées, donc les informations stockées ne doivent servir qu'à permettre d'identifier un utilisateur pour qu'il ait accès aux fonctionnalités des utilisateurs identifiés. Les informations ne doivent pas être utilisées pour cibler la personne et l'identifier concrètement.

c) design

Logo : 

Typographie : ui-sans-serif

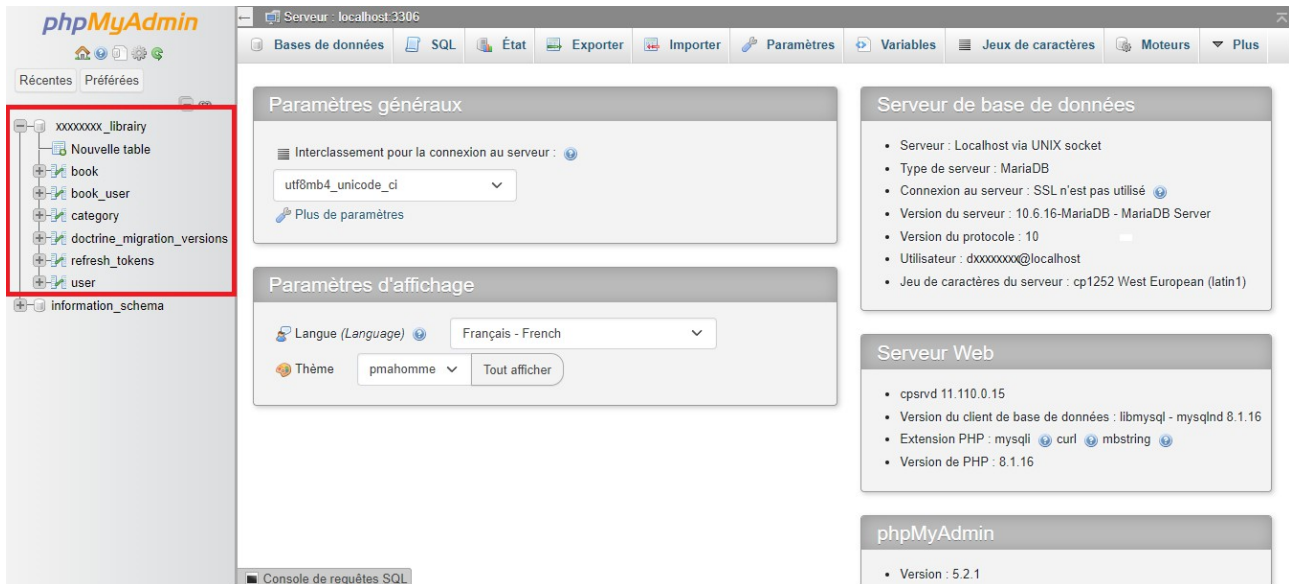
Couleur : rgb (6, 182, 212)



The mockup shows a web interface for a library. It features a blue header with a menu icon, a search icon, and a book icon. Below the header, there are links for 'inscription' and 'connexion'. The main form area is white and contains fields for 'Email' (with placeholder text 'saisissez votre adresse mail'), 'Mot de passe' (with placeholder text 'saisissez votre mot de passe'), and 'Confirmer mot de passe' (with placeholder text 'Confirmez votre mot de passe'). There is also a checkbox for 'Acceptez les condition d'utilisation CGU' and a 'Devenir Membre' button. The footer is blue and contains a book icon.

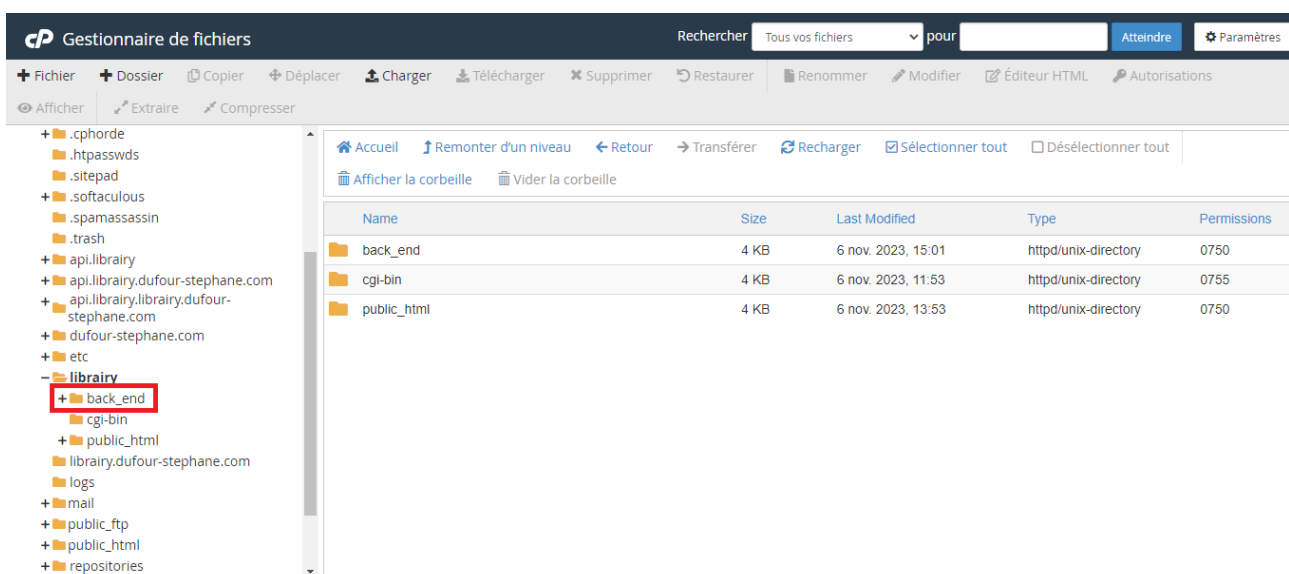
V - la mise en ligne

En utilisant [o2switch](#) pour la mise en ligne de mon site, j'ai pu exporter ma BDD locale et importer dans la BDD en ligne.



a) La mise en ligne partie Back end :

Pour la mise en ligne de la partie Back End, on doit déjà mettre le dossier back end dans le dossier back end créé dans o2switch :



Changeont le fichier .env de notre projet pour qu'il soit utilisé par notre BDD en ligne :

The screenshot shows the VS Code editor with the `.env` file open. The Explorer sidebar on the left shows the project structure. The `.env` file content is as follows:

```

17  ###> symfony/framework-bundle ###
18  APP_ENV=dev
19  APP_SECRET=57bbd8a0f9b7f7c39c0be247b47bfd73
20  ###< symfony/framework-bundle ###
21
22  ###> doctrine/doctrine-bundle ###
23  # Format described at https://www.doctrine-project.org/projects/doctrine-dbal/en/latest/reference/
24  # configuration.html#connecting-using-abstract-configuration
25  # IMPORTANT: You MUST configure your server version, either here or in config/packages/doctrine.yaml
26  #
27  DATABASE_URL="sqlite://%kernel.project_dir%/var/data.db"
28  DATABASE_URL="mysql://duXXXXXXXXXX:XXXXXXXXXX@127.0.0.1:3306/duXXXXXXXXXX_library"
29  DATABASE_URL="mysql://app:ChangeMe!@127.0.0.1:3306/app?serverVersion=10.11.2-MariaDB&charset=utf8mb4"
30  DATABASE_URL="postgresql://app:ChangeMe!@127.0.0.1:5432/app?serverVersion=15&charset=utf8"
31  ###< doctrine/doctrine-bundle ###
32
33  ###> nelmio/cors-bundle ###
34  CORS_ALLOW_ORIGIN='^https?:/(.*)?dufour-stephane\\.com(:[0-9]+)?$|^https?:/(localhost|127\\.0\\.0\\.1|:
35  [0-9]+)?$'
36  ###< nelmio/cors-bundle ###
37
38  ###> lexik/jwt-authentication-bundle ###
39  JWT_SECRET_KEY=%kernel.project_dir%/config/jwt/private.pem
40  JWT_PUBLIC_KEY=%kernel.project_dir%/config/jwt/public.pem
41  JWT_PASSPHRASE=power
42  ###< lexik/jwt-authentication-bundle ###

```

Annotations with arrows point to the following values in the `DATABASE_URL` on line 28:

- Utilisateur de la BDD**: Points to `duXXXXXXXXXX`
- Mot de passe de la BDD**: Points to `XXXXXXXXXX`
- Le nom de la BDD**: Points to `duXXXXXXXXXX_library`

Ensuite, il faut changer la variable `APP_ENV=dev` en `APP_ENV=prod` :

The screenshot shows the VS Code editor with the `.env` file open. The `APP_ENV=prod` line (line 18) is highlighted with a red box. The `.env` file content is as follows:

```

14  # Run "composer dump-env prod" to compile .env files for production u
15  # https://symfony.com/doc/current/best_practices.html#use-environment
16
17  ###> symfony/framework-bundle ###
18  APP_ENV=prod
19  APP_SECRET=57bbd8a0f9b7f7c39c0be247b47bfd73
20  ###< symfony/framework-bundle ###
21
22  ###> doctrine/doctrine-bundle ###
23  # Format described at https://www.doctrine-project.org/projects/doctr
24  # IMPORTANT: You MUST configure your server version, either here or i
25  #
26  DATABASE_URL="sqlite://%kernel.project_dir%/var/data.db"
27  DATABASE_URL="mysql://duXXXXXXXXXX:XXXXXXXXXX@127.0.0.1:3306/duXXXXXXXXXX

```

Avec `o2switch`, on a accès à un commande pour mettre à jour les sites, et faire certaines commandes particulieres :

The screenshot shows the `o2switch` web interface. On the left, there is a sidebar with the following links:

- Espace Technique
- Espace Client
- Documentation

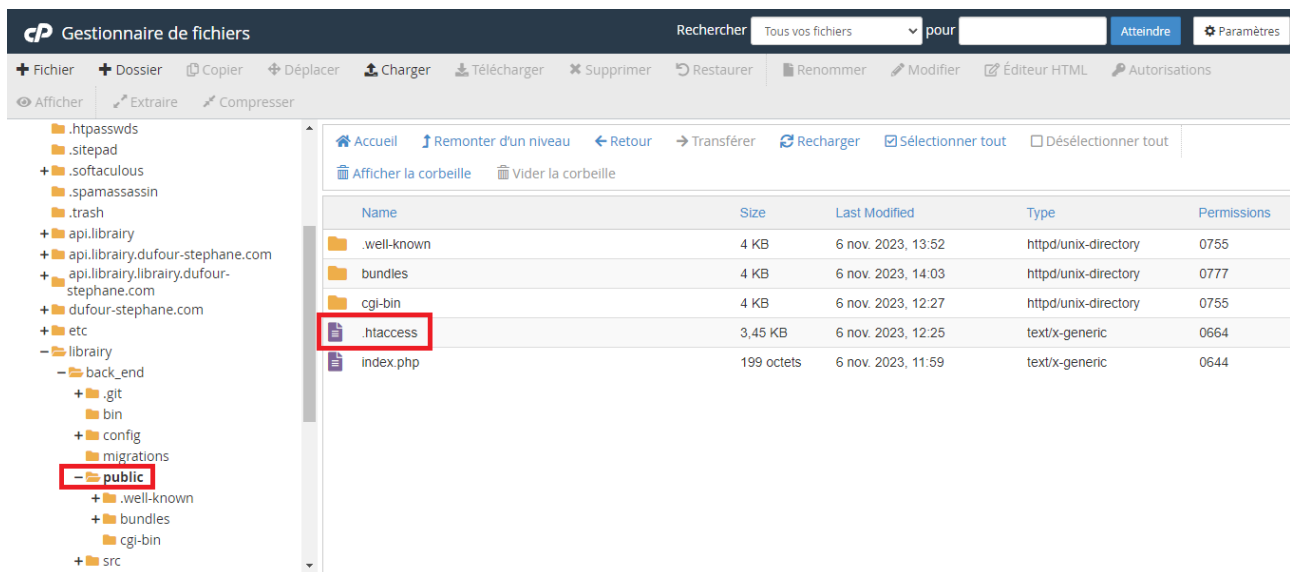
The main content area shows a terminal window with the following commands and output:

```

dxxxxxxx@bois:~/api.library/back_end
[dxxxxxxx@bois ~]$ ls
access-logs          dufour-stephane.com  logs                repositories
api.library          etc                  mail                ssl
api.library.dufour-stephane.com  library             public ftp          tmp
api.library.library.dufour-stephane.com  library.dufour-stephane.com  public_html        www
[dxxxxxxx@bois ~]$ cd api.library
[dxxxxxxx@bois api.library]$ ls
back_end
[dxxxxxxx@bois api.library]$ cd back_end/
[dxxxxxxx@bois back_end]$ ls
public
[dxxxxxxx@bois back_end]$ composer require symfony/apache-pack

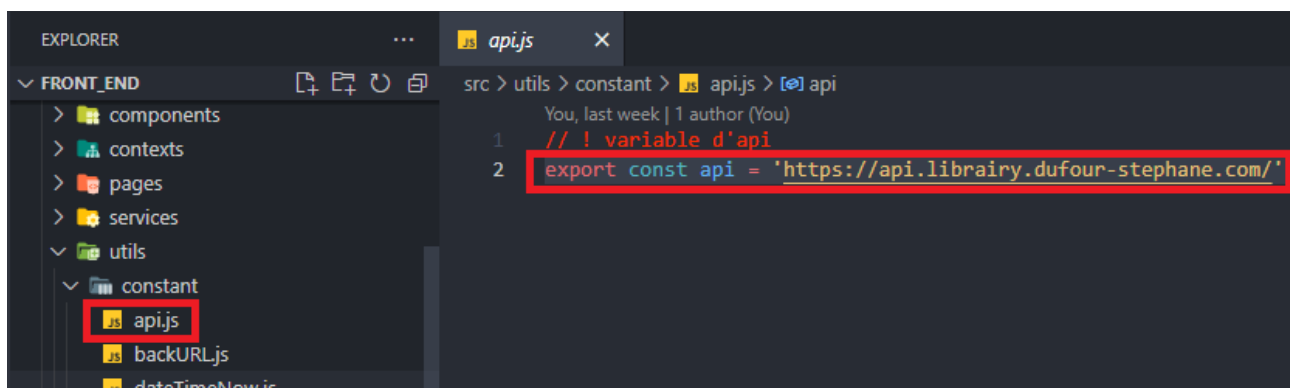
```

Avec la commande vu précédemment on a crée le fichier .htaccess dans le fichier public; le fichier .htaccess est la configuration du serveur apache :



b) La mise en ligne partie Front end :

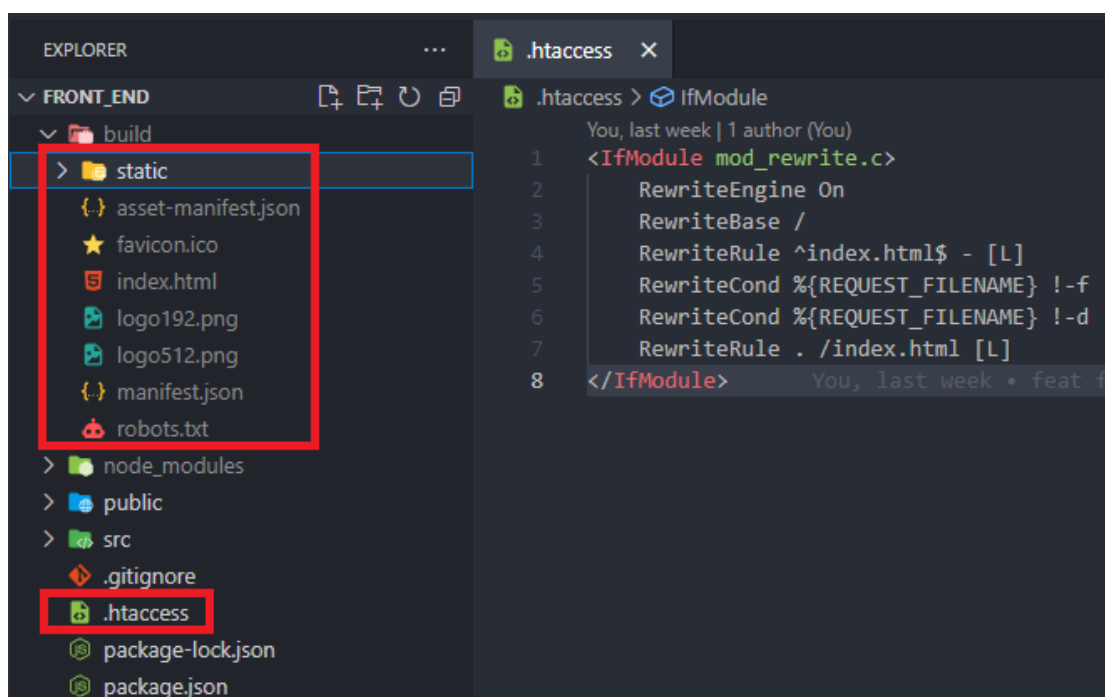
Pour la mise en ligne de cette partie, elle commence dans le projet en local en changeant la constante de l'api (j'ai choisis de mettre une constante globale car ca me permet de modifier qu'une seule variable). Avec la commande suivante, on crée le build qu'on mettra dans le dossier spécifique de o2switch **public_html** :



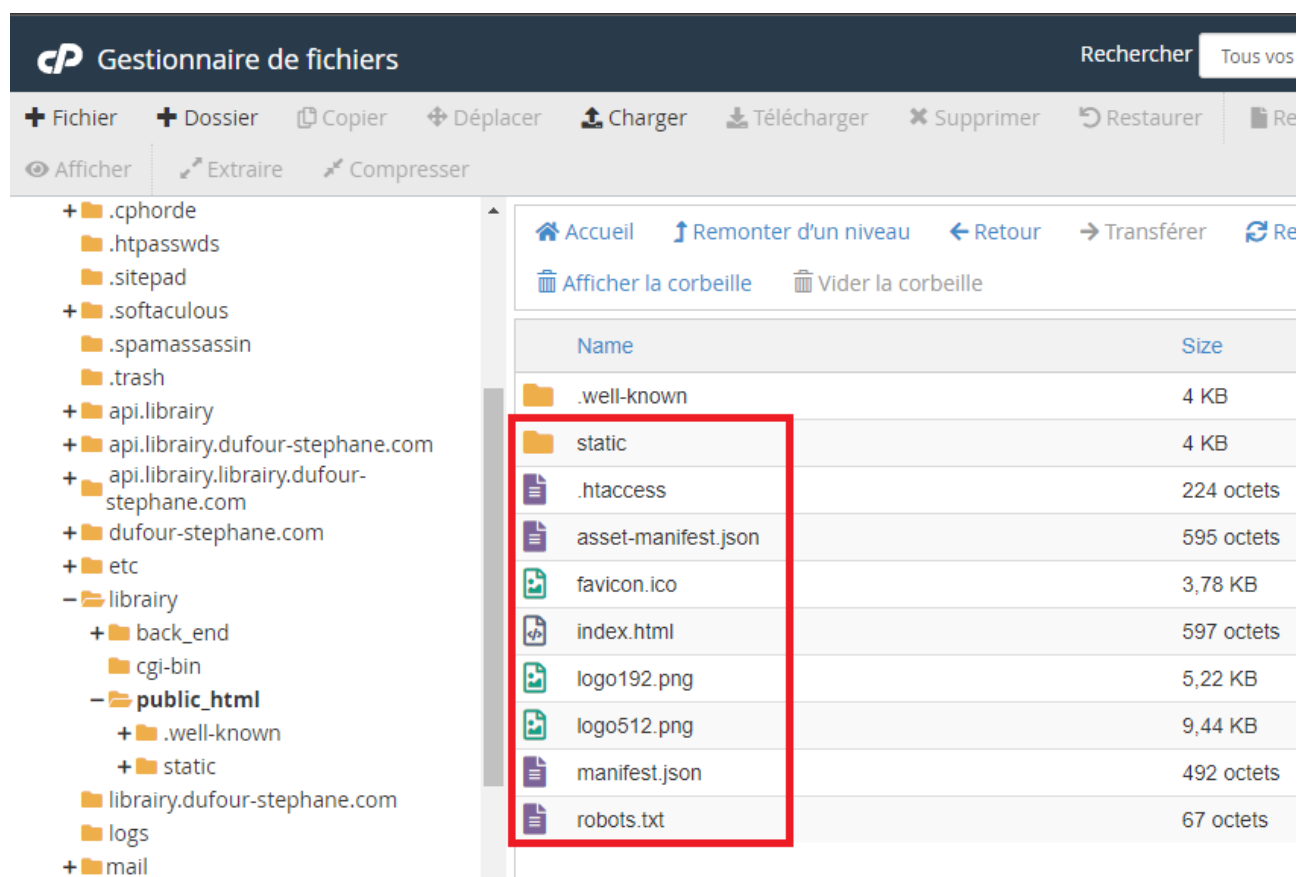
MINGW64:/c/Projet_bibliotheque/front_end

```
DufSt@Pc-Djoudjin MINGW64 /c/Projet_bibliotheque/front_end
$ npm run build
```

L'instruction a crée des dossiers et un .htaccess dans le dossier :



Ensuite, il suffit juste de mettre les fichiers créés dans le dossier spécifique pour la mise en ligne :



VI - Conclusion et évolution

Evolution :

Pour l'évolution de cette application, l'évolution possible est de mettre en place une prise de photo du titre et surtout de l'isbn pour l'ajouter automatiquement à la liste ou le créer directement avec la photo.

Conclusion :

Cette formation m'a permis d'acquérir des compétences et de comprendre le fonctionnement spécifique du Back end qui me manquait afin de me sentir à l'aise avec le back end. Et d'approfondir les connaissances que j'avais en front end avec la mise en place d'un projet. J'ai découvert pendant la formation les logiciels Adobe XD et photoshop.

Grace à mon projet, j'ai pu découvrir la mise en place d'une gestion de role, l'authentification et la mise en ligne d'un site via o2switch.