

Documentation technique FreeTube (version 1.6)

Prenom : Nicolas

Nom : Cupit

| | |
|--|----------|
| Documentation technique FreeTube (version 1.6) | 1 |
| Choix des outils | 2 |
| Les langages de programmation utilisés | 2 |
| Les Framework utilisés | 3 |
| Les bibliothèques utilisées | 4 |
| Diagrammes et schémas | 5 |
| Schéma et rôle des différentes tables | 5 |
| Diagramme UML de déploiement de l'application via docker | 8 |
| Diagramme UML de composants API REST et frontend | 8 |
| Guide de déploiement | 10 |
| Informations à renseigner nécessaires au bon fonctionnement de l'application | 10 |
| Déploiement de l'application via docker | 11 |
| Déploiement de l'application en local | 12 |
| URL d'accès aux applications | 13 |
| Bonus | 14 |
| Service PhpMyAdmin | 14 |
| Compte de Test | 14 |

Choix des outils

Les langages de programmation utilisés

Pour ce projet, seuls les langages de programmation **JavaScript** et **SQL** ont été utilisés. Au vu de la complexité du projet, mais également par rapport aux compétences techniques que je possède.

Pourquoi JavaScript ?

Bien qu'il s'agisse d'un langage initialement créé afin de programmer des applications web, y insérer des animations et plein d'autres choses, le langage JavaScript permet aujourd'hui de gérer efficacement le frontend mais également le backend.

Une flexibilité qui permet donc de réaliser l'entièreté des applications avec un seul et unique langage. D'autant plus qu'il s'agit d'un langage que j'ai énormément utilisé en entreprise.

Pourquoi SQL (Structured Query Language) et MariaDB ?

Pourquoi SQL ?

Mon choix d'utiliser une base de données SQL a été motivé par la complexité de ce projet, couplée à la structure des données qui allaient y être traitées. En effet, le langage SQL permet de gérer nativement et de manière automatisée, les données complexes ayant une structure prédéfinie, comme le suggère le cahier des charges de ce projet. Par exemple, il est possible de gérer automatiquement l'incrémentation des likes (ou j'aimes) d'une vidéo directement dans la base de données. Ce type de fonctionnalités permet aux développeurs d'avoir moins de complexité et de lignes de codes à implémenter au sein de ou des API(s), et autres composants de l'application concernée.

Pourquoi MariaDB ?

En ce qui concerne le choix de la base de données MariaDB, il s'agit de raisons purement financières. La base de données MySQL ayant été rachetée par Oracle, elle tend à devenir payante selon diverses sources sur internet, une décision qui a d'ailleurs causé sa suppression de certains systèmes d'exploitation tel que DEBIAN (distribution Linux). Afin de prévenir tous risques, il me paraît plus avisé de choisir MariaDB comme alternative. Base de données largement utilisée dans le monde de l'IT.

Les Framework utilisés :

- *NodeJS Express* : il s'agit d'un des Framework les plus populaires de la plateforme NodeJS en matière de création d'API REST, disposant d'une large documentation et d'une forte communauté de développeurs, d'où mon choix. D'autant plus qu'il permet de jouir des nombreuses fonctionnalités offertes par NodeJS en simplifiant le code nécessaire à l'utilisation de celles-ci. Fonctionnalités suggérées par le cahier des charges, telles que :
 - L'accès aux fichiers statiques via le protocole http/https (images, videos).
 - Récupération et exploitation des données stockées dans MariaDB.

Je tiens à préciser qu'il s'agit de Framework que j'utilise depuis plus de 3 ans pour certains projets. Je dispose donc d'une certaine affinité avec le Framework.

Lien de la documentation NodeJs Express : <https://expressjs.com/>

- *Vue.Js* : il s'agit là également d'un Framework JavaScript, permettant de créer des interfaces utilisateurs complexes avec peu de complexité au niveau du code. Vue.Js est une excellente alternative à la librairie React, qui, malgré sa popularité est assez complexe à utiliser, en plus de comporter une multitude de Framework tous différents les uns des autres.

Vue.Js à contrario, propose un environnement simple et compatible avec bon nombres des bibliothèques JavaScript annexes, en plus de permettre de créer des applications Single Page, permettant ainsi de charger la page côté utilisateur en une seule fois. Ce Framework est également compatible avec [Express](#).

Lien de la documentation Vue.Js : <https://vuejs.org/>

Les bibliothèques utilisées :

- **Argon2** : Permet de hacher les mots de passes utilisateurs dans la base de données conformément aux exigences du cahier des charges. Alternative au package Bcrypt qui est désormais déprécié. Permet le respect des normes RGPD et la confidentialité des données utilisateurs.

Lien de téléchargement : <https://www.npmjs.com/package/argon2>

- **Cryptr** : Permet de chiffrer et déchiffrer les données utilisateurs telles que l'adresse email. Permettant le respect des normes RGPD et la confidentialité des données utilisateurs.

Lien de la documentation : <https://nodejs.org/api/crypto.html>

- **Jwt** : Permet de gérer les sessions utilisateurs via un token Json. Le token contient l'IP de l'utilisateur, son ID, ainsi que la durée de validité du token.

Lien de la documentation : <https://www.npmjs.com/package/jsonwebtoken>

- **Multer** : Permet de gérer l'envoi de données via le body des requêtes http. Cet aspect n'est pas géré nativement par Express, ce qui requière donc l'utilisation de Multer...

Lien de la documentation : <https://www.npmjs.com/package/multer>

- **ffmpeg** : Ce package n'est pas utilisé dans cette version de l'application. Il sera utilisé dans la version 2.0, permettant de gérer l'encodage des vidéos afin de permettre la lecture d'une même vidéo sous différentes qualités (4K, 720 P), selon les désirs de l'utilisateur. Fonctionnalité Bonus.

Lien de la documentation : <https://www.npmjs.com/package/ffmpeg>

- **Axios** : Ce package permet d'effectuer les requêtes http vers l'API de l'application afin d'interagir avec la base de données, et d'afficher ou d'envoyer des données.

Lien de la documentation : <https://www.npmjs.com/package/axios>

- **Element Plus et Bosstrap** : Il s'agit des bibliothèques très utilisées dans le monde du développement Web, permettant la création d'interface utilisateurs responsives et compatibles avec la majorité des navigateurs Web. Un point non négligeable et primordial pour ce type d'application.

Schéma de la base de données



Rôle des différentes tables

• Utilisateurs

Cette table contient les informations relatives au compte utilisateur Freetube.

Elle permet de stocker :

- L'id utilisateur, permettant de l'identifier : u_id.
- Le nom de la chaîne Freetube : u_ftb_nom.
- Le nom de l'utilisateur : u_nom
- Le prénom de l'utilisateur : u_prenom
- L'email de l'utilisateur : u_email.
- Le mot de passe de l'utilisateur : u_passwd.
- Le path vers la photo de profil de l'utilisateur : u_pp.
- La description de la chaîne : u_description.
- Le total des commentaires de l'utilisateur : u_t_commentaires.
- Le total des "j'aime" reçus par les vidéos de l'utilisateur : u_t_jaimes.
- Le nombre d'abonnés à la chaîne : u_t_abos.
- Le total des vues de la chaîne : u_t_vues.
- L'âge de l'utilisateur : u_age.

• Videos

Cette table contient les informations sur les vidéos uploadés par les utilisateurs.

Elle permet de stocker :

- L'id de la vidéo, permettant de l'identifier : v_id.
- Le titre de la vidéo : v_titre.
- Le nom de la chaîne ayant publié la vidéo : v_auth_fk.
- Le path vers la miniature de la vidéo : v_miniaature.
- Le path vers le fichier vidéo : v_video.
- La description de la vidéo : v_description.
- Les tags associés à la vidéo (jusqu'à 10 tags) : v_tag1, v_tag2, ..., v_tag10.
- La catégorie de la vidéo : v_categorie_fk.
- Le total des "j'aime" de la vidéo : v_t_jaimes.
- Le nombre de commentaires associés à la vidéo : v_t_commentaires.
- Le nombre de vues de la vidéo : v_t_vues.
- Le statut de la vidéo (publique ou privée) : v_statut.
- La date de publication de la vidéo : v_DATE.
- Si la vidéo est pour un public majeur ou non : v_majeur.

- *Catfreetube (Catégories Freetube)*

Cette table contient les différentes catégories permettant le référencement des vidéos.

Elle permet de stocker :

- Le nom des catégories de vidéos : c_cat_nom.

- *Plalist_noms (sera renommée Playlist_noms)*

Cette table contient les informations relatives aux playlists créées par les utilisateurs.

Elle permet de stocker :

- L'id de la playlist : pn_id.
- Le nom de la playlist : pn_nom.
- L'utilisateur ayant créé la playlist : pn_auth_fk.

- *Playlists*

Cette table fait référence aux vidéos ajoutées dans les playlists des utilisateurs.

Elle permet de stocker :

- L'id de la playlist : p_id.
- L'id de la playlist à laquelle appartient la vidéo : p_pn_id_fk.
- L'id de la vidéo dans la playlist : p_v_id_fk.

- *Commentaires*

Cette table contient les commentaires laissés par les utilisateurs sur les vidéos.

Elle permet de stocker :

- L'id du commentaire : c_id.
- L'id de la vidéo à laquelle le commentaire est lié : c_video_id_fk.
- Le nom de l'utilisateur ayant publié le commentaire : c_auth_fk.
- Le texte du commentaire : c_txt.

- *Vtags*

Cette table contient les tags utilisés pour le référencement des vidéos.

Elle permet de stocker :

- L'id du tag : vt_id.
- Le nom du tag : vt_nom.
- Le compteur indiquant combien de fois ce tag a été utilisé : vt_t.

Diagramme UML de déploiement de l'application via Docker

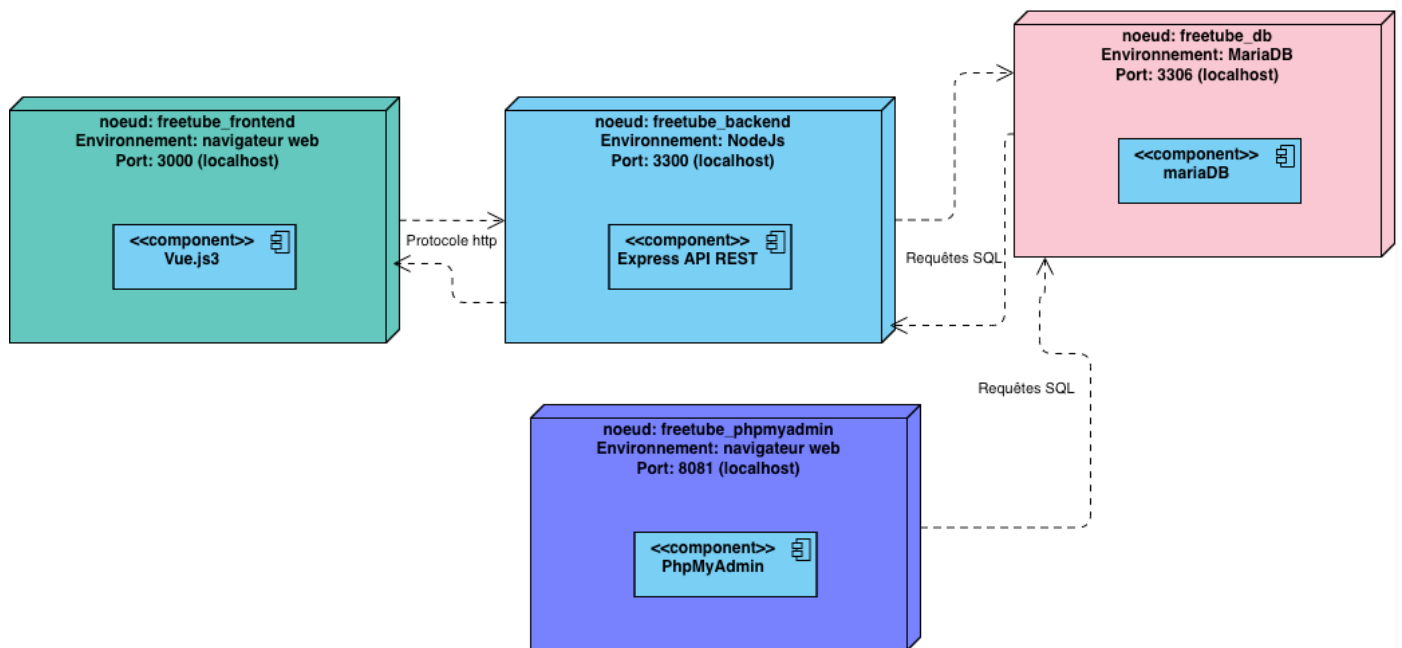


Diagramme représentant les différents nœuds de l'application et les protocoles utilisés (docker).

À noter que les nœuds sont tous interdépendants, excepté le nœud « phpmyadmin » qui lui est un serveur d'administration de bases de données.

Diagramme UML de composants | API REST et frontend

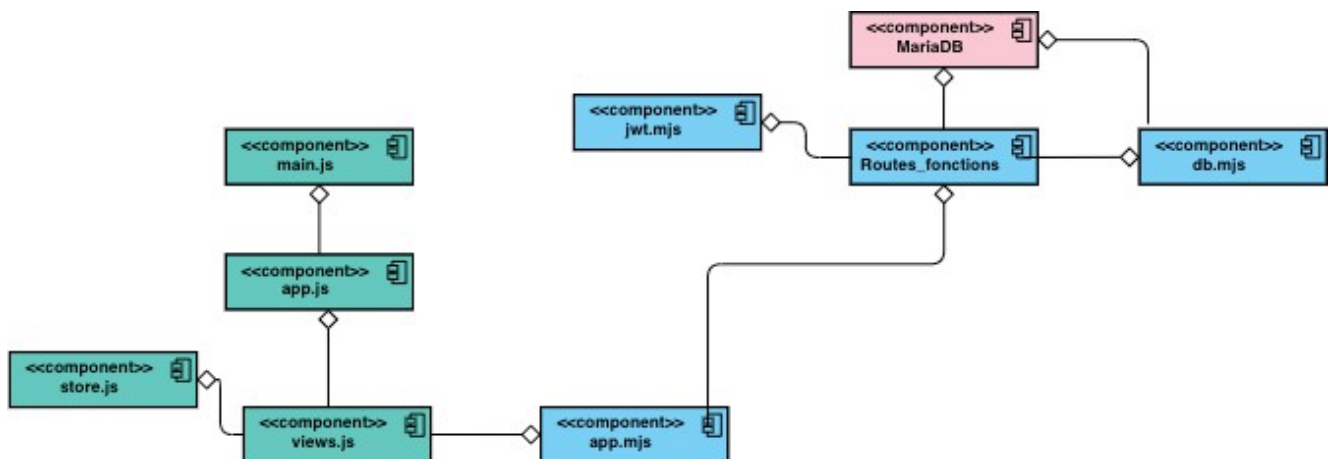


Diagramme représentant les composants principaux de l'application et leurs dépendances.

Le diagramme de composants, ayant pour but de décomplexifier l'architecture d'une application, a été représenté ici, uniquement les composants principaux de l'application (au vu de sa complexité).

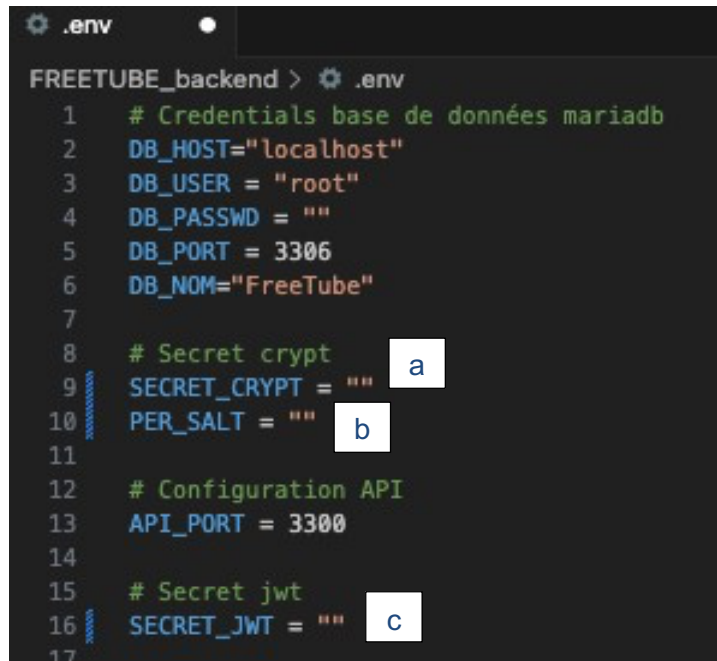
Description du diagramme (rôle des composants)

- **MariaDB** : base de données de l'application.
 - Dépendances : aucune
- **db.mjs** : permet de gérer les pools de connexion à la base de données.
 - Dépendance : **MariaDB**
- **Routes_fonctions** : permet de traduire les requêtes https en requêtes SQL en fonction de la route utilisée par le client. Permet par exemple d'uploader une vidéo ou de créer un compte utilisateur...
 - Dépendances : **MariaDB**, **jwt.mjs** et **db.mjs**
- **Jwt.mjs** : permet de générer un token, et vérifier son intégrité.
 - Dépendances : Aucune
- **app.mjs** : contient l'ensemble des routes de l'API.
 - Dépendances : **Routes_fonctions**
- **main.js** : permet d'initialiser l'application frontend.
 - Dépendances : aucune
- **app.js** : permet de lancer l'application frontend.
 - Dépendances : **main.js**
- **store.js** : permet de stocker le token utilisateur.
 - Dépendances : aucune
- **View.js** : contient toutes les pages de l'interface utilisateur de l'application.
 - Dépendances : aucune

Guide de déploiement

Informations à renseigner nécessaires au bon fonctionnement de l'application

1. Se rendre dans le répertoire FREETUBE_backend
2. Ouvrir le fichier « .env »
3. Renseigner les secrets comme indiqué ci-dessous :
 - a. dj230DEnPMLfÛ*12#@?.W098
 - b. P0@SW12DFvvP0XSSQZZDFNJLOX#2vf
 - c. 90L^M\$W23df*#}12QWKL



```
.env
FREETUBE_backend > .env
1 # Credentials base de données mariadb
2 DB_HOST="localhost"
3 DB_USER = "root"
4 DB_PASSWD = ""
5 DB_PORT = 3306
6 DB_NOM="FreeTube"
7
8 # Secret crypt
9 SECRET_CRYPT = "" a
10 PER_SALT = "" b
11
12 # Configuration API
13 API_PORT = 3300
14
15 # Secret jwt
16 SECRET_JWT = "" c
17
```

Veuillez prendre pour exemple l'image de référence ci-dessous.



```
# Credentials base de données mariadb
DB_HOST="localhost"
DB_USER = "root"
DB_PASSWD = ""
DB_PORT = 3306
DB_NOM="FreeTube"

# Secret crypt
SECRET_CRYPT = "dj230DEnPMLfÛ*12#@?.W098" a
PER_SALT = "P0@SW12DFvvP0XSSQZZDFNJLOX#2vf" b

# Configuration API
API_PORT = 3300

# Secret jwt
SECRET_JWT = "90L^M$W23df*#}12QWKL" c
```

Déploiement de l'application via docker

Étapes à suivre

1. **Lancer votre application Docker Desktop ou vos services Docker.**
2. **Assurez-vous d'avoir libéré les ports : 3000, 3300, 3306 et 8081**
3. **Aucun des répertoires ne doit contenir dans ses sous répertoires le dossier node_modules, ce qui est déjà le cas en l'état.**

Néanmoins, **si vous avez exécuté la commande npm install, veuillez supprimer ces dossiers node_modules** afin de ne pas compromettre certaines fonctionnalités de l'application dû à l'incompatibilité des binaires contenus dans ces dossiers, d'un système d'exploitation à un autre. Docker copie colle le répertoire du dossier de l'application durant la création des images, voilà pourquoi il faut veillez à respecter cette étape. Dans le cas où un fichier node_modules existe déjà avec les packages requis, la commande npm install effectué par docker ne s'exécutera pas et vous aurez une application dysfonctionnelle.

4. Ouvrir un terminal à la racine du projet (BetaFreeTube-by-Ncpt01-ReUploadOldSchoolProject).
5. Exécuter la commande : Docker compose build.
6. Attendre la fin de l'exécution de la commande spécifiée à l'étape 3.
7. Exécuter la commande : docker compose up.
8. Une fois la création du conteneur terminée, accédez aux applications via les URL fournies dans la documentation : [urls \(cliquez\)](#)

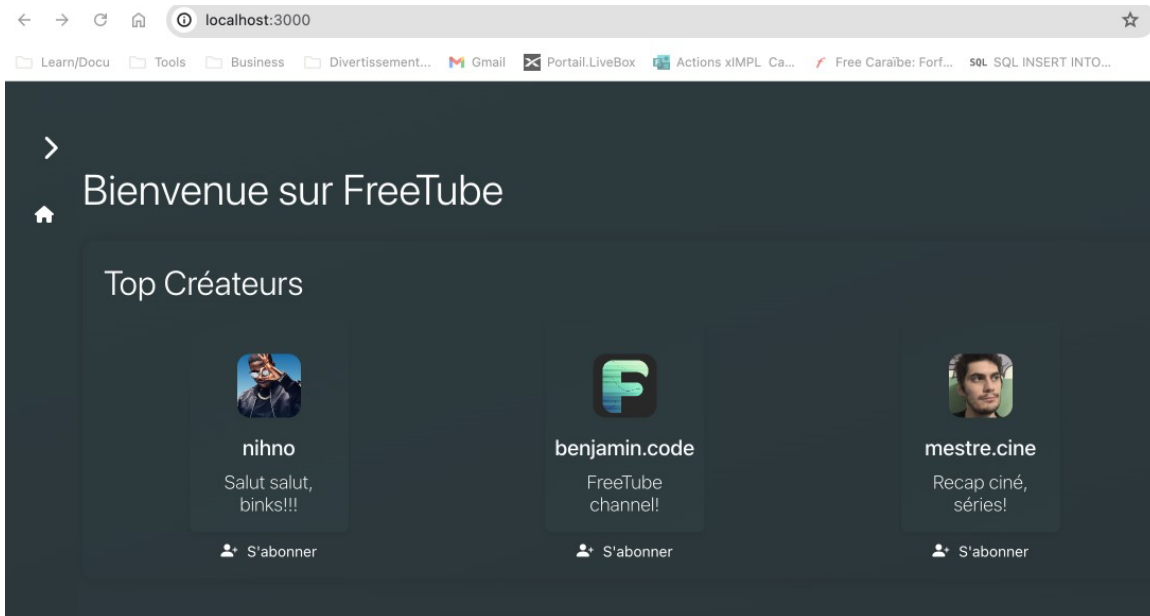
Déploiement de l'application en local

Étapes à suivre

1. **Assurez-vous d'avoir libéré les ports : 3000, 3300 et 3306.**
2. **Assurez-vous d'avoir une instance MariaDB version 5.2.1 ou ultérieur.**
3. **Assurez-vous que vos credentials pour la connexion à votre base de données soit en accords avec les données présentes dans le fichier «.env » se trouvant au sein du répertoire : BetaFreeTube-by-Ncpt01-ReUploadOldSchoolProject-FREETUBE_backend.**
4. **Assurez-vous d'avoir la version 8.2.4 du langage PHP ou version ultérieur.**
5. **Importer le fichier « YoutubeClone_Data.sql » présent à la racine du projet, dans votre base de données MariaDB.**
6. Ouvrir un terminal à la racine du projet (si ce n'est pas encore le cas), puis se rendre au répertoire : FREETUBE_backend.
7. Exécuter la commande : npm install.
8. Attendre la fin de l'exécution de la commande spécifiée à l'étape précédente, qui vous créera un dossier node_modules.
9. Exécuter la commande : npx nodemon app.mjs
10. Ouvrir un second terminal à la racine du projet, puis se rendre au répertoire : FREETUBE_frontend.
11. Exécuter la commande : npm install.
12. Attendre la fin de l'exécution de la commande spécifiée à l'étape précédente, qui vous créera un dossier node_modules.
13. Exécuter la commande : npm run dev
14. Une fois la création du conteneur terminée, accédez aux applications via les URL fournies dans la documentation : [urls \(cliquez\)](#)

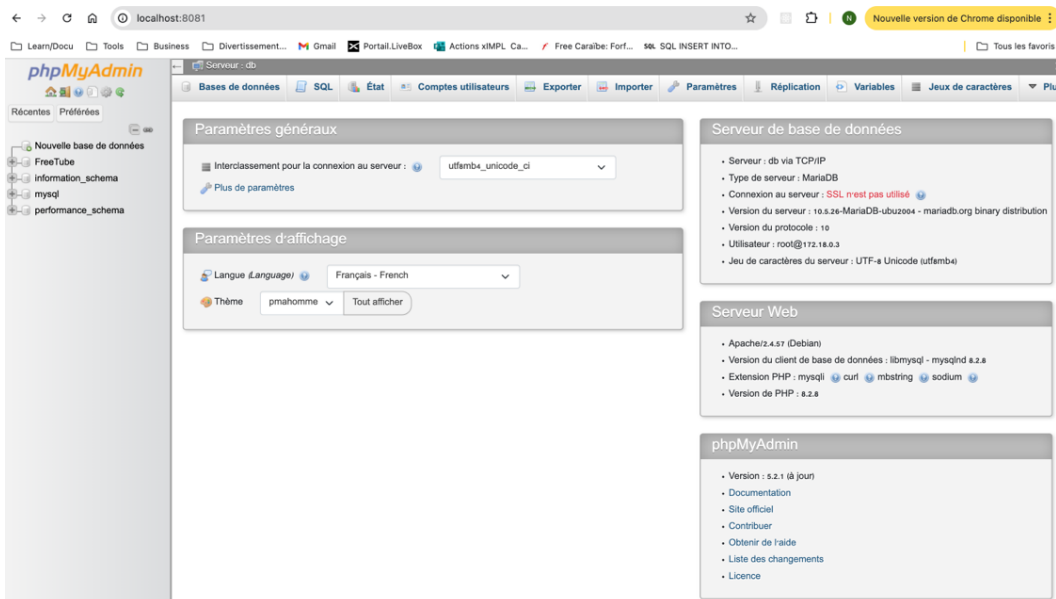
URL d'accès aux applications

- Application web : <http://localhost:3000/>



Interface utilisateur affiché par le navigateur

- PhpMyAdmin (déploiement via docker) : <http://localhost:8081/>



Interface administrateur PhpMyAdmin

- PhpMyAdmin (en local) : <http://localhost:{votre port}/>

Bonus

Service PhpMyAdmin

Le service PhpMyAdmin a été ajouté au conteneur Docker de l'application, afin de permettre aux ingénieurs systèmes, ainsi qu'aux développeurs, de maintenir la base de données plus facilement. Il permet également de visualiser le contenu de la base de données durant l'exécution de l'application (ajout de vidéos, créations utilisateurs...).

Compte de Test

Un compte de test a été créé en amont des nombreux comptes fictifs présents dans la base de données. Afin d'accéder aux credentials du compte, veuillez-vous rendre à la racine du projet, puis ouvrez le fichier « userTest.md »