



Documentation technique Sommaire :

Présentation Description Mise en place Présentation :

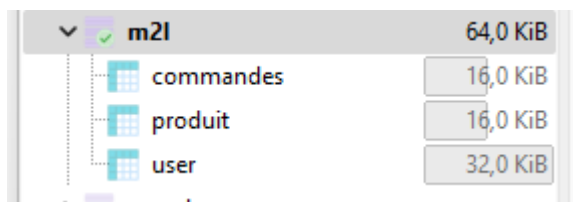
Suite à l'AP 3, nous avons pour but de réaliser l'application mobile en tant qu'application lourde pour la maison des ligues de lorraine M2L.

Description :

Cela sera une application mobile d'équipements sportifs réalisé en Dart avec Flutter ainsi qu'en Node Js. Les parties importantes sont une gestion des produits via un compte admin et principalement la connexion et déconnexion de l'admin.

Fonctionnalités principales :

Dashboard Admin - Gestion des Produits : le dashboard offre aux administrateurs un ensemble d'outils pour gérer efficacement les produits disponibles sur la plateforme. Les administrateurs peuvent facilement ajouter de nouveaux produits en remplissant un formulaire . Ils peuvent spécifier des informations telles que le nom du produit, la description, les images, les prix, les quantités disponibles, etc. Les administrateurs peuvent modifier les détails des produits existants à tout moment. Les administrateurs peuvent supprimer des produits de la plateforme.



The screenshot shows a database management interface with a tree view on the left and a table structure on the right. The tree view shows a folder named 'm2l' with a green checkmark icon. The table structure on the right lists three tables: 'commandes' (16,0 KiB), 'produit' (16,0 KiB), and 'user' (32,0 KiB). Each table is represented by a small grid icon.

Table	Size
commandes	16,0 KiB
produit	16,0 KiB
user	32,0 KiB

Notre base de données est faite sous HeidiSQL et est composée de trois tables

De base Options Index (0) Clés étrangères (0) Vérifier les contraintes (0) Partitions </> C

Nom : user

Commentaire :

Colonnes : + Ajouter x Supprimer ▲ Monter ▼ Descendre

#	Nom	Type de données	Taille/Ensem...	Non si...	ZERO...	Par défaut	Commentaire
1	uuid	UUID		<input type="checkbox"/>	<input type="checkbox"/>	NULL	
2	nom	VARCHAR	500	<input type="checkbox"/>	<input type="checkbox"/>	NULL	
3	email	VARCHAR	500	<input type="checkbox"/>	<input type="checkbox"/>	NULL	
4	mdp	VARCHAR	500	<input type="checkbox"/>	<input type="checkbox"/>	NULL	
5	admin	INT	11	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	NULL	

La table user stocke les informations des utilisateurs enregistrés sur la plateforme en afin les authentifier le rôle d'administrateur est à modifier à la main depuis la base de données.

De base Options Index (0) Clés étrangères (0) Vérifier les contraintes (0) Partitions </> C

Nom : produit

Commentaire :

Colonnes : + Ajouter x Supprimer ▲ Monter ▼ Descendre

#	Nom	Type de données	Taille/Ensem...	Non si...	ZERO...	Par défaut	Commentaire
1	puid	UUID		<input type="checkbox"/>	<input type="checkbox"/>	NULL	
2	nom	VARCHAR	500	<input type="checkbox"/>	<input type="checkbox"/>	NULL	
3	description	VARCHAR	500	<input type="checkbox"/>	<input type="checkbox"/>	NULL	
4	prix	FLOAT		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	NULL	
5	quantite	INT	11	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	NULL	
6	img	VARCHAR	500	<input type="checkbox"/>	<input type="checkbox"/>	NULL	

Aide Ignorer Enregistrer

La table produit stocke les informations sur les produits disponibles à l'achat afin d'afficher les produits sur le site, gérer les stocks.

The screenshot shows a database management interface for a table named 'commandes'. At the top, there are tabs for 'De base', 'Options', 'Index (0)', 'Clés étrangères (0)', 'Vérifier les contraintes (0)', and 'Partitions'. Below these, the 'Nom' field is set to 'commandes' and the 'Commentaire' field is empty. A section labeled 'Colonnes :' contains buttons for '+ Ajouter', 'x Supprimer', '▲ Monter', and '▼ Descendre'. Below this is a table with 8 columns: '#', 'Nom', 'Type de données', 'Taille/Ensem...', 'Non si...', 'ZERO...', 'Par défaut', and 'Commentaire'. The table lists 5 columns for the 'commandes' table.

#	Nom	Type de données	Taille/Ensem...	Non si...	ZERO...	Par défaut	Commentaire
1	cuid	UUID		<input type="checkbox"/>	<input type="checkbox"/>	NULL	
2	montant	FLOAT		<input type="checkbox"/>	<input type="checkbox"/>	NULL	
3	puiduser	UUID		<input type="checkbox"/>	<input type="checkbox"/>	NULL	
4	nom	VARCHAR	50	<input type="checkbox"/>	<input type="checkbox"/>	NULL	
5	produits	UUID		<input type="checkbox"/>	<input type="checkbox"/>	NULL	

La table Commandes stocke les détails des paniers d'achat des utilisateurs et est liée aux deux tables précédentes.

L'application communique avec la base de données via une API créée avec NodeJS et Express.

```
const express = require('express');
const cors = require('cors');
const userRoute = require('./routes/userroute');
const prodRoute = require('./routes/prodroute');
const commandeRoute = require('./routes/commandesroute');
const cookieParser = require('cookie-parser');
const authRoute = require('./routes/authroute');
```

Les données importantes sont cachées dans un fichier .env

```
BACK > database > {} database.js > ...
1  const mariadb = require('mariadb');
2  require('dotenv').config();
3
4
5
6  const pool = mariadb.createPool({
7    host: process.env.DB_HOST,
8    database: process.env.DB_DTB,
9    user: process.env.DB_USER,
10   password: process.env.DB_PWD,
11   port: process.env.DB_PORT,
12   connectionLimit: 100,
13   acquireTimeout: 30000,
14 });
15
16 module.exports = { pool: pool};
```

On utilise un middleware servant à authentifier les personnes connectés grâce à un jeton de connexion temporaire avec JsonWebToken. Ici on vérifie si c'est bien un administrateur en décodant le isAdmin qui vérifie en base de données si la valeur de l'admin est égale à 1.

```
const jwt = require('jsonwebtoken');

exports.authenticator = (req, res, next) => {
  try {
    const token = req.headers.authorization;
    console.log('Token récupéré depuis le cookie :', token);
    if (!token) {
      throw new Error('Accès refusé : Aucun jeton fourni.');
```

```
    }

    jwt.verify(token, process.env.API_KEY, (err, decoded) => {
      if (err) {
        throw new Error('Accès refusé : Jeton invalide.');
```

```
      }

      if (decoded.isAdmin) {
        req.user = decoded;

        next();
      } else {
        throw new Error('Accès refusé : L\'utilisateur n\'est pas un administrateur.');
```

```
      }
    });
  } catch (error) {
    res.status(401).json({ error: error.message });
  }
};
```

Les middlewares se placent sur les routes.

```
BACK > routes > {} useroute.js > ...
1  const express = require('express');
2  const router = express.Router();
3  const userController = require('../controllers/userController');
4  const { authenticator } = require('../middleware/middleware');
5
6  router.post('/ins', userController.postUser);
7  router.post('/conn', userController.conn);
8  router.delete('/user/:uuid', authenticator, userController.deleteUser);
9  router.post('/logout', userController.handleLogout);
10
11 router.get('/user', authenticator, userController.getAllUser);
12
13 module.exports = router;
14
```

```
BACK > routes > {} commandesroute.js > ...
1  const express = require('express');
2  const router = express.Router();
3  const commandeController = require('../controllers/commandeController');
4  const decodeToken = require('../middleware/uuid');
5
6  router.post('/valider', decodeToken, commandeController.valider);
7  router.get('/commandes', commandeController.getAllCommandes);
8
9  module.exports = router;
```

```
BACK > routes > {} authroute.js > ...
1  const express = require('express');
2  const router = express.Router();
3  const authController = require('../controllers/authController');
4  const { authenticator } = require('../middleware/middleware');
5
6  router.get('/conn', authenticator, authController.getUser);
7
8
9  module.exports = router;
```

On utilise Bcrypt pour comparer le mot de passe donné avec le mot de passe hashé dans la base de données.

```
exports.conn = async (req, res) => {  
  const { email, mdp } = req.body;  
  
  const query = 'SELECT uuid,email, mdp, admin FROM user WHERE email = ?';  
  
  try {  
    const conn = await pool.getConnection();  
  
    const rows = await conn.query(query, [email]);  
  
    if (rows && rows.length > 0) {  
      const user = rows[0];  
  
      const isValid = await bcrypt.compare(mdp, user.mdp);  
    }  
  }  
}
```

Marion TRINH BTS SIO SLAM

La gestion des produits tels que l'ajout, modification, suppression se font ici sans

middleware car ces fonctions sont appelées dans un composant qui lui-même s'affiche sur le dashboard admin que si la valeur isAdmin est vérifiée c'est pourquoi on peut s'en passer ici.

```
BACK > controllers > prodController.js > putProdbypuid > putProdbypuid
20
21
22 > exports.getAllProduits = async (req, res) => { ...
35 };
36
37 > exports.postProd = async (req, res) => { ...
66 };
67
68 > exports.putProdbypuid = async (req, res) => { ...
125 };
126
127 > exports.deleteProd = async (req, res) => { ...
142 };
143 > exports.decrementQuantity = async (req, res) => { ...
166 };
167
168 > exports.incrementQuantity = async (req, res) => { ...
186 };
187
188
```

uploads.

```
const express = require('express');
const router = express.Router();
const prodController = require('../controllers/prodController');
const multer = require('multer');
const upload = multer({ dest: 'uploads/' });
router.use('/uploads', express.static('uploads'))

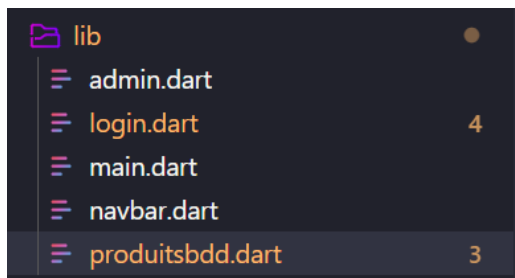
router.get('/produit', prodController.getAllProduits);
router.post('/produit', upload.single('image'), prodController.postProd);
router.put('/produit/:puid', upload.single('image'), prodController.putProdbypuid);
router.delete('/produit/:puid', prodController.deleteProd);
router.put('/produit/:puid/decrement', prodController.decrementQuantity);
router.put('/produit/:puid/increment', prodController.incrementQuantity);

module.exports = router;
```

useroute.js

uploads

- 1a8c2d30c94f12c5b7b096632dd1...
- 1a18e6e1d9d6b45a5557224c72d...
- 1b2cbc820d8e182deaddcb21390...
- 1c18fb98579b287c30954e7bec41...
- 1ce7434d68d1e4838490d85f... U
- 2bdd19284d4970586dc8a4e68bb...
- 3db184b6decbbdb1359ca5fc2dd...
- 3fa4a0cceb03abe97c63d5ea04bdf...
- 4e3bc4e77878db8f7f57c852db96...
- 4f55f30f90969e594a02fb8c6125e...
- 5be8ca9269352946df442401572d...
- 6a6683b6d36b75d6b42233fd11e...



Login est la page permettant la connexion.

Admin est la page qui permet de récupérer les produits.

Main est la page contenant le titre et la couleur de l'application.

Navbar est la page où est gérée la navigation.

Produitsbdd est la page regroupant tous les formulaires de modifications, d'ajouts et de suppressions de produits.