Atelier de Professionnalisation 3 Maison des Ligues



Contexte:

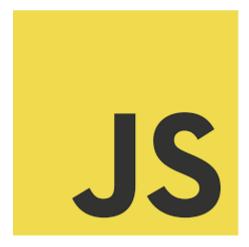
Le contexte proposé est celui de la Maison de Ligues de Lorraine (M2L) qui a pour mission de fournir des espaces et des services aux différentes ligues sportives régionales et à d'autres structures hébergées.

SOMMAIRE:

- OUTILS UTILISÉS
- CONNEXION
- INSCRIPTION
- PRODUIT
- AJOUTER
- MODIFICATION
- SUPPRIMER

OUTILS UTILISÉS:

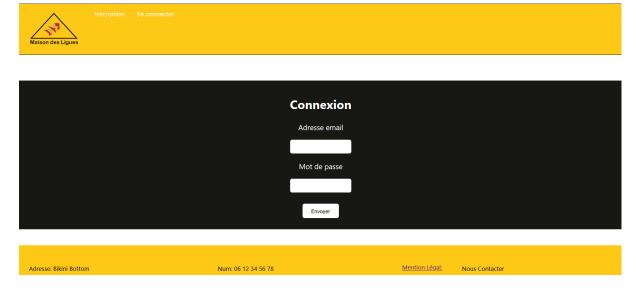






CONNEXION:

Comme expliquer dans la documentation Utilisateur nous avons tout d'abord la page de connexion:



Lorsque l'utilisateur rentre ses informations et qu'elles sont renseignées dans notre base de donnée alors il a accès à notre page "Produit". Pour ce faire nous avons mit dans notre API "serveur.js" le code suivant:

Tout d'abord nous vérifions que l'identifiant rentrée existe bien dans notre base de donnée, ensuite les mot de passe étant crypté dans notre base de donnée nous devons comparer le mot de passe rentré avec les mots de passe dans notre base de donnée.

Si les informations de l'utilisateur sont correctes alors on récupère son id, son mail, et son rôle sinon on renvoie un message disant que le nom d'utilisateur ou le mot de passe est incorrect.

INSCRIPTION:

Dans le cas où l'utilisateur souhaite s'inscrire il lui suffit de rentrer un mail et un mot de passe pour accéder au produit:



Inscrivez-vous							
	E_mail						
	DocTchenique@gm	nail.cc					
	Mdp						
	Envoyer						
Adresse: Bikini Bottom	Num: 06 12 34 56 78	Mention Légal;	Nous Contacter				

Lorsqu'il s'inscrit, les information rentrée vont dans notre base de donnée:

id	7	mail	role	mdp
	18	Yassine	0	\$2b\$10\$Ikjr9bbb.izv89iL0cZ3.uEi
	19	Admin	1	\$2b\$10\$ppH2SNiAKsDbBs1egCbi
	20	Documentation	0	\$2b\$10\$BsIAi7g4DXPWiRS7wG
	21	DocTchenique@gmail.com	0	\$2b\$10\$9XcWf5iw9V7rodMWSar

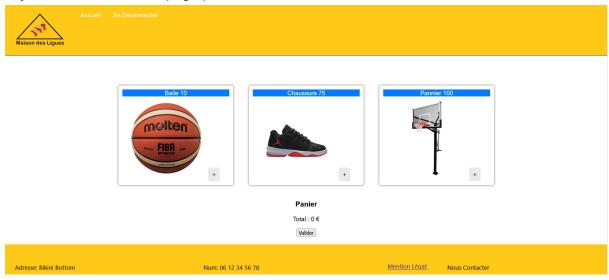
Voici le code utiliser pour l'inscription:

```
app.post('/inscr', async (req, res, hashedPassword) => {
    let conn;
    try {
        console.log("lancement de la connexion")
        conn = await pool.getConnection();
        console.log("lancement de la requete insert")
        console.log(req.body);
        const bcrypt = require('bcrypt');
        const saltRounds = 10;
        const plainPassword = req.body.mdp;
        hashedPassword = await bcrypt.hash(plainPassword, saltRounds);
        let requete = 'INSERT INTO ap2 (mail, mdp) VALUES (?, ?);'
        let rows = await conn.query(requete, [req.body.mail, hashedPassword]);
        console.log(rows);
        res.status(200).json(rows.affectedRows)
    }
    catch (err) {
        console.log(err);
    }
}
```

lci on voit que le mot de passe est haché à l'aide de bcrypt on le récupère dans notre body puis le "mélange" 10 fois pour ensuite le rentrer dans notre base de donnée.

PRODUIT:

Voyant maintenant notre page produit:



Le nom, le prix et l'image de nos produits sont importés de notre bdd afin que nous puissions en ajouter, modifier et supprimer.

id 🥊	Articles 🔻	Image	Prix	Quantite
16	Pannier	/Pannier.png	100	29
15	Chaussure	/Chaussure.png	75	37
1	Balle	/balle.png	10	994

AJOUTER:

Pour ajouter un produit nous avons fait une requête "INSERT INTO" dans la table produit

et avons récupéré les valeur dans notre formulaire :

MODIFICATION:

Pour modifier un produit la requête était différente

```
app.put()'/modification/:id', async (req, res) => {

const id = parseInt(req.params.id)
let conn;
try {
    console.log("lancement de la connexion")
    conn = await pool.getConnection();
    console.log("lancement de la requete update")
    let requete = 'UPDATE produit SET Articles = ?, Image = ?, Prix = ?, Quantite = ? WHERE id = ?;'
    let rows = await conn.query(requete, [req.body.Articles, req.body.Image, req.body.Prix, req.body.Quantite, id]);
    console.log(rows);
    res.status(200).json(rows.affectedRows)
}

catch (err) {
    console.log(err);
}
}
```

et la méthode que j'ai utilisé nécessite deux composant, le premier récupère l'identifiant de l'article à modifier:

Et c'est dans le second composant que l'on fait appelle a la requête pour modifier mit dans notre API:

```
return (

div className='container'>

chip Modifier votre article</h2>

chip Modifier votre article</hd>

chip Modifier votre article chip Modifier
```

SUPPRIMER:

Pour supprimer un produit la méthode est très semblable nous faisons aussi deux étape avant de pouvoir supprimer un produit il n'y a globalement que la requête qui change:

```
app.delete('/Del/:id', async(req,res) => {
    const id = parseInt(req.params.id)
    let conn;

try{
    console.log("lancement de la connexion")
    conn = await pool.getConnection();
    console.log("lancement de la requete")

// Supprimer un produit de la base de données en fonction de son ID
    const rows = await conn.query ('DELETE FROM produit WHERE id = ?', [id]);
    console.log(rows);
    res.status(200).json(rows.affectedRows)

// Catch(err){
    console.log(err)
```