

# Atelier de Professionnalisation 3

## Maison des Liges



Contexte :

Le contexte proposé est celui de la Maison de Liges 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:

Inscription Se connecter

### Connexion

Adresse email

Mot de passe

Envoyer

Adresse: Bikini Bottom Num: 06 12 34 56 78 [Mention Légale](#) Nous Contacter

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:

```

94 app.post('/connexion', async (req, res) => {
95   const id = parseInt(req.params.id);
96   const { mail, mdp } = req.body;
97   let conn;
98   try {
99     console.log("Lancement de la connexion");
100    conn = await pool.getConnection();
101    console.log("Lancement de la requête");
102    // Interroger la base de données pour récupérer l'utilisateur
103    const rows = await conn.query('SELECT * FROM ap2 WHERE mail = ?', [mail]);
104    console.log("connexion", rows[0].mdp);
105    // Si un utilisateur est trouvé, le renvoyer, sinon renvoyer une erreur d'authentification
106    const match = await bcrypt.compare(mdp, rows[0].mdp);
107    console.log(match);
108
109    if (rows.length === 1) {
110      res.status(200).json({ id: rows[0].id, mail: rows[0].mail, role: rows[0].role });
111    } else {
112      console.log("pas correct")
113      res.status(401).json({ message: "Nom d'utilisateur ou mot de passe incorrect." });
114    }
115  } catch (err) {
116    console.log(err);
117    res.status(500).json({ message: "Une erreur s'est produite lors de la connexion à la base de données." });
118  } finally {
119    if (conn) {
120      conn.release(); // Libérer la connexion à la base de données
121    }
122  }
123 });

```

Tout d'abord nous vérifions que l'identifiant rentré 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@gmail.cc

Mdp

.....

Envoyer

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

id	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:

```


63 app.post('/inscr', async (req, res, hashedPassword) => {
64   let conn;
65   try {
66     console.log("lancement de la connexion")
67     conn = await pool.getConnection();
68     console.log("lancement de la requete insert")
69     console.log(req.body);
70     const bcrypt = require('bcrypt');
71     const saltRounds = 10;
72     const plainPassword = req.body.mdp;
73     hashedPassword = await bcrypt.hash(plainPassword, saltRounds);
74     let requete = 'INSERT INTO ap2 (mail, mdp) VALUES (?, ?)';
75     let rows = await conn.query(requete, [req.body.mail, hashedPassword]);
76     console.log(rows);
77     res.status(200).json(rows.affectedRows)
78   }
79   catch (err) {
80     console.log(err);
81   }
82 })

```


Ici 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:


 [Accueil](#) [Se Déconnecter](#)

Balle 10




+

Chaussure 75



+

Pannier 100



+

Panier

Total : 0 €

Valider



Adresse: Bikini Bottom

Num: 06 12 34 56 78

[Mention Légale](#)

[Nous Contacter](#)

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

```
117 app.post('/Ajt', async(req,res) => {
118   let conn;
119   try{
120     console.log("lancement de la connexion")
121     conn = await pool.getConnection();
122     console.log("lancement de la requete")
123     // Insérer un nouveau produit dans la base de données
124     const rows = await conn.query ('INSERT INTO produit (Articles, Image, Prix, Quantite) VALUES (?, ?, ?, ?)',
125     [[req.body.Articles, req.body.Image, req.body.Prix, req.body.Quantite]]);
126     console.log(rows);
127     res.status(200).json(rows.affectedRows)
128   }
129   catch(err){
130     console.log(err)
131   }
132 })
```

et avons récupéré les valeur dans notre formulaire :

```
42 return (
43   <div className='container' style={{ marginTop:'200px'}}>
44     <h2> Ajouter un article</h2>
45
46     <form onSubmit={handleSubmit(AjoutArticles)}>
47       <label>Articles </label>
48       <input {...register("Articles", { required: true })} onChange={(e) => setArticles(e.target.value)} />
49
50       <label>Lien de l'images </label>
51       <input {...register("Image", { required: true })} onChange={(e) => setImage(e.target.value)} />
52
53       <label>Prix </label>
54       <input {...register("Prix", { required: true })} onChange={(e) => setPrix(e.target.value)} />
55
56       <label>Quantité </label>
57       <input {...register("Quantite", { required: true })} onChange={(e) => setQuantite(e.target.value)} />
58
59       {(errors.Articles || errors.Image || errors.Prix || errors.Quantite) ? <span>Tous les champs doivent être remplis</span> : ""}
60
61       <input type="submit" />
62
63       <p><Link to="/Sup"><input type='button' value='Supprimer un article' /></Link>
64
65       <Link to="/Modifier"><input type='button' value='Modifier un article' /></Link></p>
66     </form>
67   </div>
```



## MODIFICATION :

Pour modifier un produit la requête était différente

```
169 app.put('/:modification/:id', async (req, res) => {
170
171   const id = parseInt(req.params.id)
172   let conn;
173   try {
174     console.log("lancement de la connexion")
175     conn = await pool.getConnection();
176     console.log("lancement de la requete update")
177     let requete = 'UPDATE produit SET Articles = ?, Image = ?, Prix = ?, Quantite = ? WHERE id = ?;'
178     let rows = await conn.query(requete, [req.body.Articles, req.body.Image, req.body.Prix, req.body.Quantite, id]);
179     console.log(rows);
180     res.status(200).json(rows.affectedRows)
181   }
182   catch (err) {
183     console.log(err);
184   }
185
186 })
```

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

```
25   return (
26     <div className='body'>
27       <h2> Les produits a modifier</h2>
28       <div className="box">
29         {affichage ?
30           quiz.map(produit => (
31             <div>
32               <div className='box-title' style={{ marginTop: '200px'}}>
33                 Produit n°{produit.id}
34               </div>
35               <div className='box-body'>
36                 {produit.Articles} {produit.Prix} €
37                 <img src={` ${process.env.PUBLIC_URL}/images/${produit.Image}`} width="200px"/>
38               </div>
39               <div className='box-footer'>
40                 <Link to={` /modification/` + produit.id}><FaPen /></Link>
41               </div>
42             </div>
43           ))
44           : <p>Changement...</p>
45         }
```

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

```
90   return (
91     <div className='container'>
92
93       <h2> Modifier votre article</h2>
94
95       <form onSubmit={handleSubmit(editQuestion)}>
96         <label>Articles </label>
97         <input defaultVale={Articles} onChange={(e) => setArticles(e.target.value)} />
98         <label>Image </label>
99         <input defaultVale={Image} onChange={(e) => setImage(e.target.value)} />
100        <label>Prix </label>
101        <input defaultVale={Prix} onChange={(e) => setPrix(e.target.value)} />
102        <label>Quantité </label>
103        <input defaultVale={Quantite} onChange={(e) => setQuantite(e.target.value)} />
104        {(errors.Articles || errors.Image || errors.Prix || errors.Quantite) ? <span>Tous les champs doivent être remplis</span> : ""}
105        <input type="submit" />
106      </form>
107    </div>
108  )
109 }
110
111 }
```

## 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:

```
134 app.delete('/Del/:id', async(req,res) => {
135   const id = parseInt(req.params.id)
136   let conn;
137   try{
138     console.log("lancement de la connexion")
139     conn = await pool.getConnection();
140     console.log("lancement de la requete")
141     // Supprimer un produit de la base de données en fonction de son ID
142     const rows = await conn.query ('DELETE FROM produit WHERE id = ?', [id]);
143     console.log(rows);
144     res.status(200).json(rows.affectedRows)
145   }
146   catch(err){
147     console.log(err)
148   }
}
```