

RAPPORT DE PROJET

DÉVELOPPEMENT

FRONT-BACK END

Groupe :

Zeina AL WAZZAN

Aurélien CLAUX

Myriam AZZAZ

M2 MIAGE SITN

2020-2021

Sommaire

1. Présentation du besoin et de sa solution	2
2. Explication des fonctionnalités	2
a. Le header	2
b. La page d'Accueil, « Le marché »	3
c. Le Panier	3
d. Confirmation de commande	4
e. L'Administration	4
f. Se connecter	5
g. S'inscrire	5
3. Composant et bibliothèques utilisées.....	6
4. Schéma d'architecture des composants et services Angular	7
5. Sessions	8
a. Partie Connexion.....	8
b. Partie Inscription.....	10
6. API Node.....	11
a. User :	11
b. Vegetable :	11
7. Base de données : MangoDB	11
a. Partie Utilisateur.....	12
b. Partie vegetables.....	12
c. Partie Payment.....	13

1. Présentation du besoin et de sa solution



M. Verger tient une épicerie qui vend des fruits et légumes frais, dénommée *Le Petit Maraîcher*. Dans le contexte de la crise sanitaire, et dans un monde qui se digitalise, M. Verger a fait appel à nos services afin de lui créer une plateforme sur laquelle les clients peuvent commander directement des fruits et légumes frais vendues par *Le Petit Maraîcher*, afin d'être livré à domicile. Il va ainsi pouvoir étendre sa clientèle.

L'application que nous avons développée est un site sur lequel on peut acheter des fruits et légumes dans le but d'être livré à la maison. Le site web du Petit Maraîcher a été conçu en Angular coté front-end et en Node JS coté back-end, avec la base de données hébergée par Mango DB. Nous entrerons dans plus de détails par la suite.

2. Explication des fonctionnalités

Avant de vous expliquer les bibliothèques et les technologies utilisées, nous allons présenter les différentes fonctionnalités qui constituent notre application.

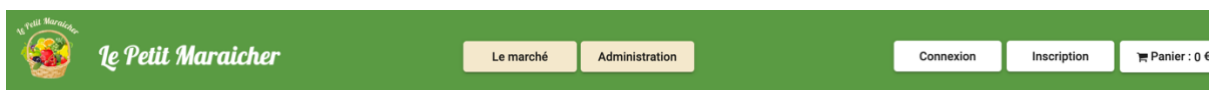
Le *Petit Maraîcher* est composé de **six pages** :

- La page d'Accueil, nommée « Le marché ». C'est sur cette page principale que l'utilisateur va pouvoir faire ses courses
- Le Panier : Qui s'actualise lorsqu'un utilisateur ajoute ou enlève des légumes
- La page de connexion
- La page d'inscription
- La page de confirmation de commande
- La page administration

Par soucis de clarté, nous allons passer en revue les fonctionnalités de chacune des pages.

a. Le header

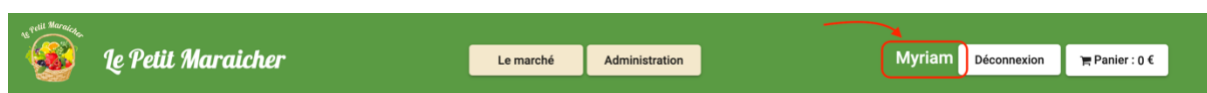
Il est présent sur chacune des pages.



Ceci est la vue admin (bouton administration visible), ce qui diffère pour le client, c'est qui verra tout ça, hormis l'onglet administration.

Le logo et le nom de l'épicerie de M. Verger figure en haut à gauche. Au centre, le bouton marché (on est sur cette page par défaut, mais on a mis ce bouton ici dans le cas où le client va dans le panier, ou souhaite se connecter, et veut retourner sur le marché). À droite, les boutons de connexion et d'inscription.

Lorsque l'utilisateur est connecté, voici la vue qu'il aura :



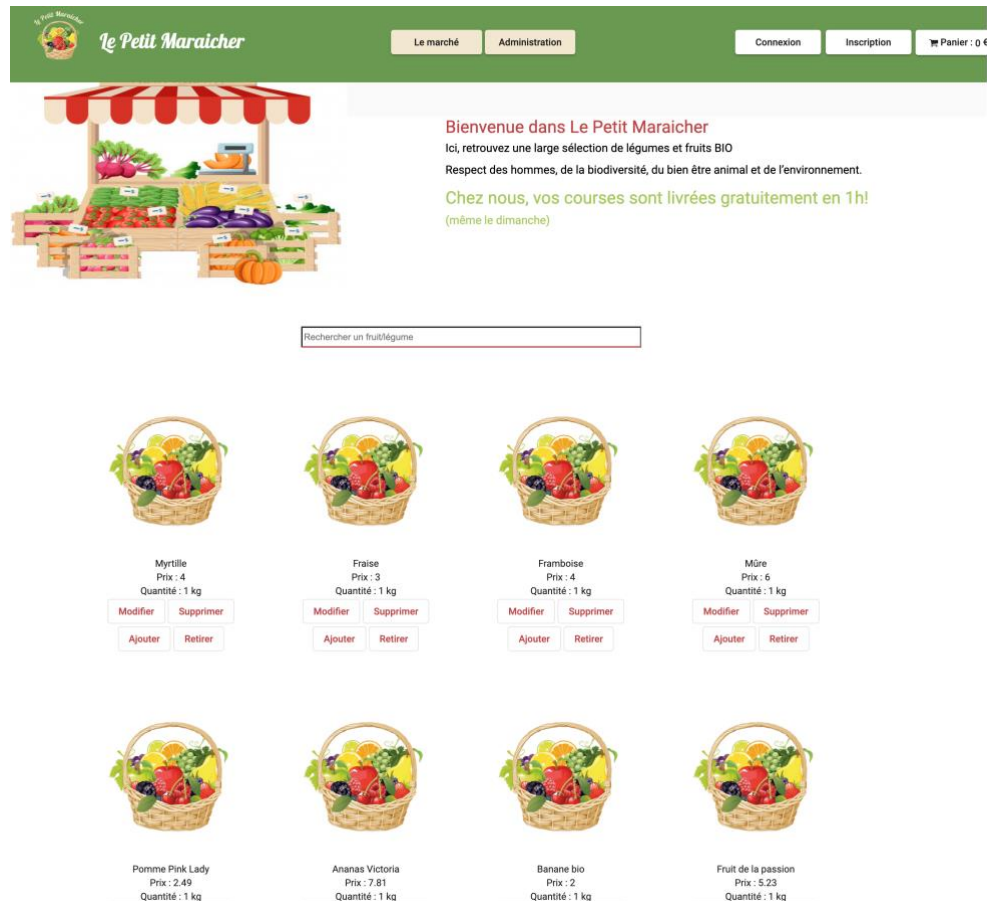
Son nom s'affichera, et il pourra également se déconnecter s'il le souhaite.

Enfin, le bouton panier se situe à l'extrême droite. Le prix du panier s'affiche en temps réel en fonction de ce qu'il y a dedans. Il est mis à jour à chaque fois que l'utilisateur ajoute un produit au panier.

b. La page d'Accueil, « Le marché »

C'est la première page qui apparaît à l'utilisateur. La gamme de produits de M. Verger étant limitée, on a décidé de tout regrouper sous l'enseigne du « marché ». On y trouve aussi une description de ce que propose Le Petit Maraîcher en tête de page, suivi des produits.

Chaque produit affiché a un nom et un prix, ainsi qu'un bouton « ajouter », qui permet à l'utilisateur, s'il clique dessus, de l'ajouter au panier dans le but de l'acheter.



c. Le Panier

Le bouton Panier est cliquable en haut à droite, dans le header. Il s'agit d'un récapitulatif des produits ajoutés par l'utilisateur dans le panier.

C'est via cette page qu'il va pouvoir confirmer sa commande et passer à la commande, en cliquant sur le bouton « Valider mon panier ». Ce qui le renverra vers la page de confirmation de commande.

Voici un panier bien fourni, avec le total qui correspond bien à la somme des prix des produits. Le total est également visible dans le header, sur le bouton panier, en temps réel.

Panier

Framboise Quantité : 2	Prix : 4 €
Mûre Quantité : 1	Prix : 6 €

Total : 14 €

Valider mon panier

d. Confirmation de commande

C'est l'étape finale du processus d'achat sur Le Petit Maraîcher. Le client, afin de passer commande, va devoir remplir un formulaire requérant son nom, prénom, adresse, adresse mail. Ces informations vont servir pour la livraison.

Sur la même page, le client communique les coordonnées de sa carte bancaire pour procéder au paiement.

Une fois que tout a bien été saisi, le client peut cliquer sur « valider le paiement » afin de passer commande.

Checkout Form

Adresse de facturation

Nom et Prénom
Robert VERGER

Email
robert.verger@wanadoo.fr

Adresse
2 rue des fraisiers

Ville
Paris

Commune
Paris

Code Postal
75002

Paiement

Cartes acceptées
VISA MASTERCARD AMEX DISCOVER

Nom sur votre carte
Robert VERGER

Numéro carte
4979 8722 2832 0023

Mois Exp
Juin

Annee Exp
2023

CVV
007

☒ Adresse de facturation = adresse de livraison

Valider le paiement

e. L'Administration

Cette page est censée être réservée aux utilisateurs de type admin, mais cette fonctionnalité n'est implémentée.

Via cette page, il peut ajouter des produits au catalogue des fruits et légumes, en saisissant son nom et son prix. Page d'administration des objets manipulées, nous en avons 2 : les utilisateurs et les légumes (Fruits/Légumes). Il est possible d'ajouter des produits, avec leur nom et prix.

En bas de page est présent le tableau des utilisateurs, il est possible de les modifier ou de les supprimer. La création d'utilisateur est gérée uniquement par l'inscription des utilisateurs

The screenshot shows the 'Le Petit Maraîcher' website interface. At the top, there is a green header with the logo, navigation links for 'Le marché' and 'Administration', and buttons for 'Connexion', 'Inscription', and 'Panier : 0'. The main content area features a form titled 'Ajouter un produit' with fields for 'Nom de l'aliment', 'Prix de l'aliment en €', and a file upload section. Below the form is a table titled 'Utilisateurs' with columns for 'Login', 'Nom complet', 'Ville', 'CodePostal', 'Type', and 'Modifier'.

Login	Nom complet	Ville	CodePostal	Type	Modifier
zeina	Zeina W.	Vincennes	94300		Modifier Supprimer
myriam	Myriam	paris	93390		Modifier Supprimer

f. Se connecter

La fonctionnalité de connexion est située dans le header, et accessible depuis le marché, l'administration, et le panier.

1. En tant qu'utilisateur, je peux me connecter au site du petit maraîcher en saisissant mon login et mot de passe.
2. En tant qu'admin, je peux me connecter au site du petit marché en saisissant mon login et mon mot de passe afin d'accéder au site.

The screenshot shows the 'Le Petit Maraîcher' website interface with the 'Connexion' form. The header is green and contains the logo, navigation links for 'Le marché' and 'Administration', and buttons for 'Login', 'Register', and 'Panier : 0 €'. The main content area features a form titled 'Connexion' with fields for 'Login' and 'Password', a 'Se connecter' button, and a link for 'Vous n'êtes pas inscrit?' leading to the 'S'identifier' button.

g. S'inscrire

Si l'utilisateur ne possède pas de compte, il peut en créer un depuis l'onglet inscription (et également depuis la fenêtre connexion, où en bas de page, figure un bouton renvoyant vers la page d'inscription).

Inscription

Full Name

Adresse

Code Postal

Ville

Login

Password

Register

[Vous êtes déjà inscrit?](#)

Se connecter

3. Components et bibliothèques utilisées

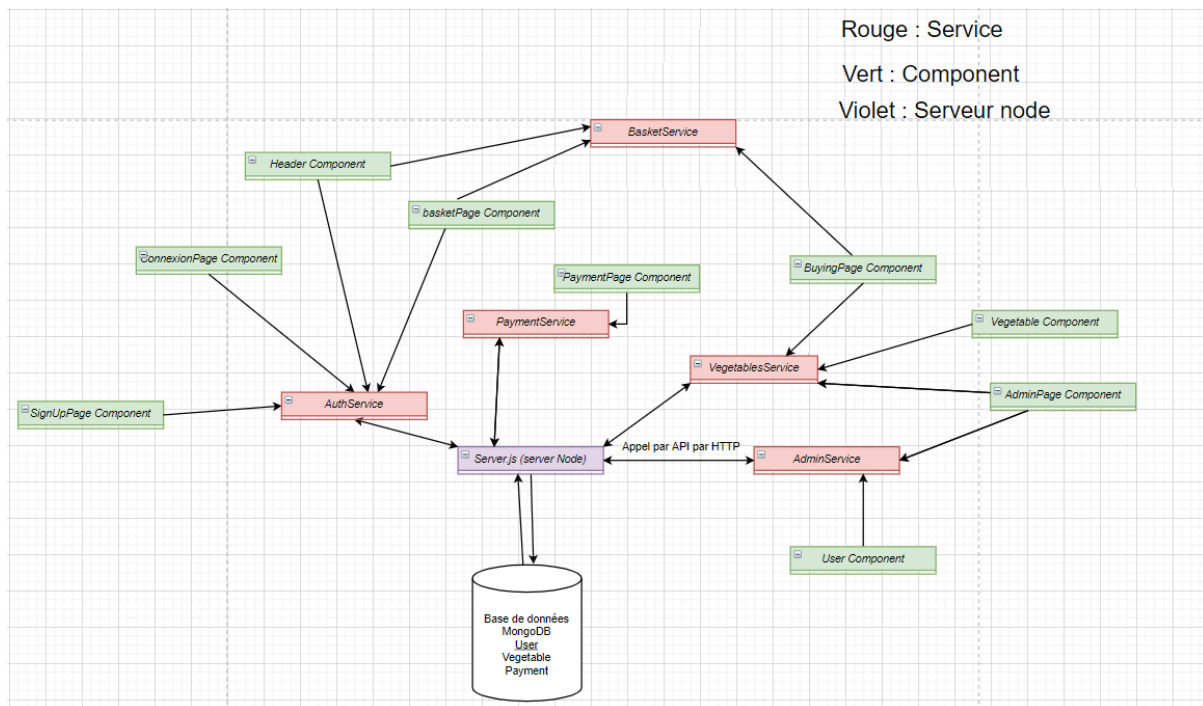
Objets :

- Vegetables
- User
- Payment

Component :

- **Buying-page** → Page d'Accueil : affichage des produits, Ajouter Produits, Retirer du panier Produits, Modifier et Supprimer produits du site
- **Adminpage** → Administration du site : Ajouter des Produits (nom, prix), Consulter les comptes utilisateurs (modifier, supprimer)
- **Basket-page** → Le Panier : récapitulatif des produits ajoutés, validation du panier
- **Payment-page** → Page de payment après validation du panier : form de payment
- **Connexion-page** → Page de connexion : login et password utilisateur
- **Signup-page** → Page de connexion : login et password utilisateur
- **Header** → le header appliqué sur toutes les pages du site
- **Models** → schémas de définition des objets

4. Schéma d'architecture des composants et services Angular



4 services permettent de communiquer avec le serveur Node :

- Auth : Correspond avec le serveur node via HTTP pour récupérer les informations de connexion.
- Payment : Correspond avec le serveur node via HTTP pour envoyer les informations de paiements
- Vegetable : Correspond avec le serveur node via HTTP pour envoyer et récupérer des informations sur les légumes
- Admin : Correspond avec le serveur node via HTTP pour envoyer et récupérer des informations liées aux utilisateurs et pour créer de nouveaux vegetable

Le service Basket est utilisé pour centraliser les fonctions liées au panier mais n'est ne correspond pas avec le serveur node.

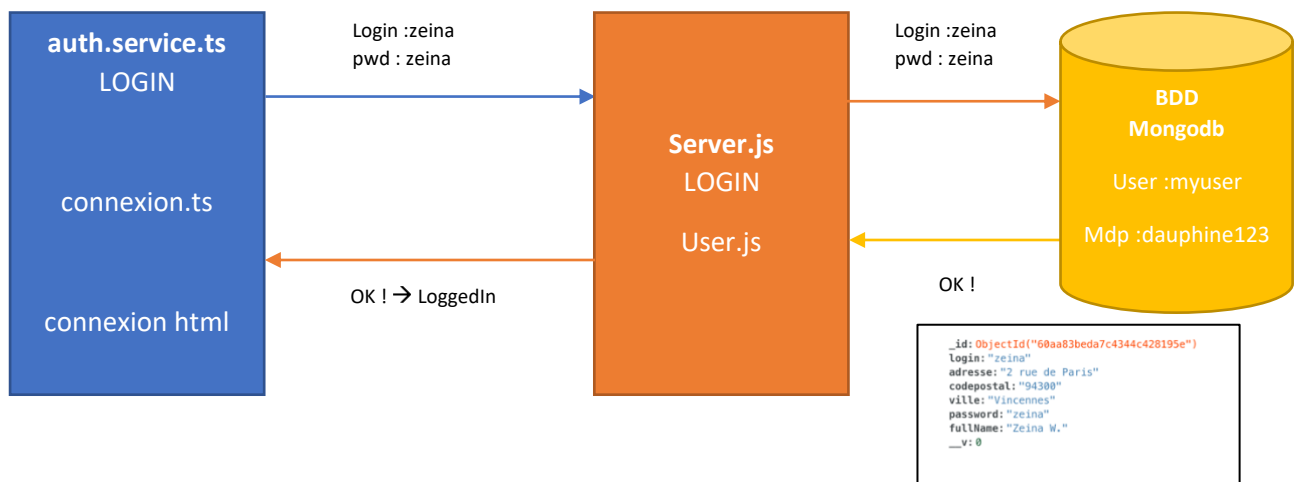
Les composants ont pour but d'interagir avec leur HTML suite aux données récupérées via l'utilisation des fonctionnalités des services pour récupérer les informations.

5. Sessions

Les fichiers connectés pour assurer la bonne session de la **connexion** et l'**inscription** :

	Partie Front	Partie Back
Connexion	<ul style="list-style-type: none"> auth.service.ts connexion-page.component.ts connexion.-page.component.html 	<ul style="list-style-type: none"> user.js server.js
Inscription	<ul style="list-style-type: none"> auth.service.ts signup-page.component.ts signup-page.component.html 	

Le Front et le back sont connectés à partir des fichiers cités ci-dessus. Voici un schéma qui résume le fonctionnement de la base de connexion :



a. Partie Connexion

A partir du front, les données saisies par l'utilisateur « login et password » seront communiquées à l'**auth.service.ts** à partir du **ngModel**.

```

<div class = "form">
  <div class = "titre"> Connexion </div>
  <mat-form-field>
    <mat-label>Login</mat-label>
    <input matInput type="text" [(ngModel)]="login">
  </mat-form-field>

  <br/> <br/>

  <mat-form-field>
    <mat-label>Password</mat-label>
    <input matInput type="password" [(ngModel)]="password">
  </mat-form-field>

```

Les attributs login et password sont aussi définis dans « **connexion-page.component.ts** ». Dans **connexion-page.component.ts** on utilise les bibliothèques suivantes :

```

src > app > connexion-page > TS connexion-page.component.ts > ConnexionPageComponent
1 import { Component, OnInit } from '@angular/core';
2 import { AuthService } from '../auth.service';
3 import { FormControl, Validators } from '@angular/forms';
4
5
6

```

```

src > app > connexion-page > TS connexion-page.component.ts > ...
1 import { Component, OnInit } from '@angular/core';
2 import { AuthService } from '../auth.service';
3 import { FormControl, Validators } from '@angular/forms';
4
5
6 @Component({
7   selector: 'app-connexion-page, form-field-error-example',
8   templateUrl: './connexion-page.component.html',
9   styleUrls: ['./connexion-page.component.scss']
10 })
11 export class ConnexionPageComponent implements OnInit {
12
13   login:any = "";
14   password:any = "";
15
16   constructor(public authService: AuthService) { }
17
18   ngOnInit(): void {
19   }
20
21   submit():any{
22     this.authService.login(this.login, this.password).subscribe(
23       (userInfo:any)=>{
24         this.authService.connectedUser = userInfo;
25       },
26       (error)=>{
27         console.log("error",error)
28       }
29     )
30   }
31
32 }

```

Ensuite, le fichier « **auth.service.ts** » va connecter le front avec le serveur **server.js** à partir du localhost 3000 afin de communiquer au serveur les données du login et password.

```

13
14 login(login:any, password:any):Observable<any>{
15   return this.http.post('http://localhost:3000/connexion', {login, password}, {withCredentials: true});
16 }
17

```

Dans **server.js** on est connectée à une bdd dans mongodb (user :myuser et mdp :dauphine123).

```

26
27 mongoose.connect('mongodb+srv://myuser:dauphine123@cluster0.mongodb.net/connexion?retryWrites=true&appName=connexion', {
28   useNewUrlParser: true,
29   useUnifiedTopology: true
30 }).then(() => {
31   console.log("Successfully connected to DB!");
32 }).catch((error) => {
33   console.log("Unable to connect to DB!");
34 });

```

Le serveur va recevoir les informations saisies dans le front. A son tour, il vérifie l'existence d'un utilisateur avec les informations saisies (ligne 178 ci-dessous), et il va répondre au client avec une erreur s'il ne trouve pas l'utilisateur ou avec une connexion avec succès.

```

175 {} package-lock.json
176 {} package.json
177 {} server.js
178 //login
179 app.post('/connexion', (request, response) => {
180   User.findOne({login: request.body.login, password: request.body.password }, (error, user) => {
181     if (error) return response.status(401).json({msg: 'Error'});
182     if (!user) return response.status(401).json({msg: 'Wrong login'});
183     request.session.userId = user._id;
184     response.status(200).json({login: user.login, fullName: user.fullName});
185   });
186 });

```

Si l'utilisateur est connecté avec succès, son login et un bouton Logout seront affichés sur le header.



b. Partie Inscription

A partir du front, les données saisies par l'utilisateur « fullname, adresse, codepostal, ville, login et password » seront communiquées à l'**auth.service.ts** à partir du **ngModel**.

```
src > app > signup-page > signup-page.component.html > body > div.form >
1 <body>
2
3   <div class = "form">
4     <div class = "titre"> Inscription </div>
5     <mat-form-field>
6       <mat-label>Full Name</mat-label>
7       <input matInput type="text" [(ngModel)]="fullName">
8     </mat-form-field>
9
10    <br/> <br/>
11
12    <mat-form-field>
13      <mat-label>Adresse</mat-label>
14      <input matInput type="text" [(ngModel)]="adresse">
15    </mat-form-field>
16
17    <br/> <br/>
18
19    <mat-form-field>
20      <mat-label>Code Postal</mat-label>
21      <input matInput type="text" [(ngModel)]="codepostal">
22    </mat-form-field>
23
24    <br/> <br/>
25
26    <mat-form-field>
27      <mat-label>Ville</mat-label>
28      <input matInput type="text" [(ngModel)]="ville">
29    </mat-form-field>
30
31    <br/> <br/>
```

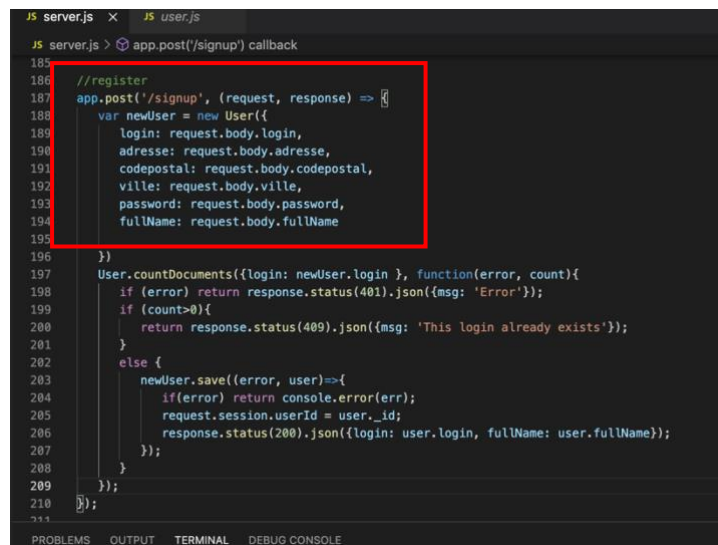
Ensuite, le fichier « **auth.service.ts** » va connecter le front avec le serveur **server.js** à partir du localhost 3000 afin de communiquer au serveur les données du login et password.

```
23
24 register(login:any, password:any, fullName:any, adresse:any, codepostal:any, ville:any):Observable<any>{
25   return this.http.post('http://localhost:3000/signup', {login: login, password:password, fullName:fullName, adresse:adresse, codepostal:codepostal});
26 }
27
```

Dans **server.js** on est connectée à une bdd dans mongodb (user : myuser et mdp :dauphine123).

```
26
27 mongoose.connect('mongodb+srv://myuser:dauphine123@cluster0.
28 .then(() =>{
29   console.log("Successfully connected to DB!");
30 })
31 .catch((error) => {
32   console.log("Unable to connect to DB!");
33 });
34
```

Le serveur va recevoir les informations saisies dans le front. A son tour, il va créer un nouvel utilisateur avec les attributs saisis et reçus. Il vérifie l'existence de l'utilisateur dans la base, s'il existe il renvoie une erreur, sinon il l'ajoute dans la base.



```
JS server.js x JS user.js
JS server.js > app.post('/signup') callback
185
186 //register
187 app.post('/signup', (request, response) => {
188   var newUser = new User({
189     login: request.body.login,
190     adresse: request.body.adresse,
191     codepostal: request.body.codepostal,
192     ville: request.body.ville,
193     password: request.body.password,
194     fullName: request.body.fullName
195   })
196   User.countDocuments({login: newUser.login }, function(error, count){
197     if (error) return response.status(401).json({msg: 'Error'});
198     if (count>0){
199       return response.status(409).json({msg: 'This login already exists'});
200     }
201     else {
202       newUser.save((error, user)=>{
203         if(error) return console.error(err);
204         request.session.userId = user._id;
205         response.status(200).json({login: user.login, fullName: user.fullName});
206       });
207     }
208   });
209 });
210
```

6. API Node

Il existe 2 types d'objets dans notre projet : les utilisateurs (User) et les fruits et légumes (Vegetable). Les deux objets sont gérés avec les actions CRUD.

Par objets les actions CRUD utilisées avec la méthode HTTP suivi du lien sont les suivantes :

a. User :

. **Create** : Le formulaire de connexion crée un nouvel utilisateur dans la base de données. Il utilise POST /signup

. **Read** : Depuis la page d'administration les utilisateurs sont restitués sous forme de tableaux, il utilise GET /admin/user

Ou depuis la visualisation de la modification GET /admin/users/:id

. **Update** : Depuis le bouton modifier un utilisateur de la page d'administration, PUT /admin/users/:id

. **Delete** : Depuis le bouton supprimer un utilisateur sur la page d'administration DELETE /admin/users/:id

b. Vegetable :

Create : Depuis le formulaire de la page d'administration pour créer des aliments. POST /buying

Read : Depuis le page Marché. Le tableau des fruits et légumes GET /buying

Ou depuis la visualisation de la modification sur le bouton modifier GET / buying/:id

Update : Depuis le bouton modifier d'un vegetable PUT /buying/:id

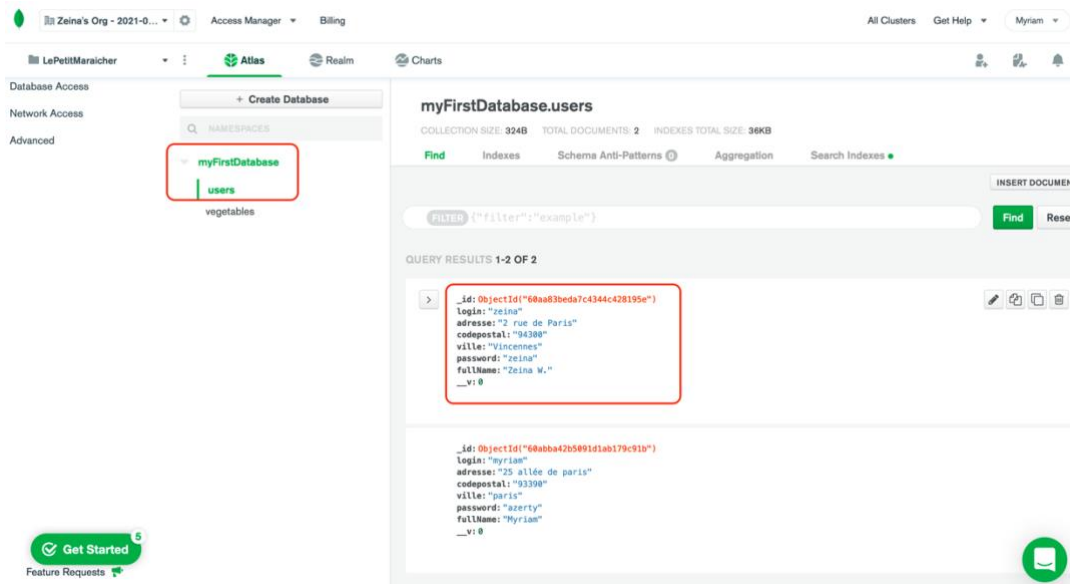
Delete : Depuis le bouton supprimer DELETE /buying/:id

7. Base de données : MangoDB

Nous avons choisi la solution de MangoDB pour la base de données du Petit Maraicher. Elle nous sert à conserver les données des utilisateurs lorsqu'ils s'inscrivent (afin qu'ils puissent se connecter par la suite) et les produits qui constituent le catalogue.

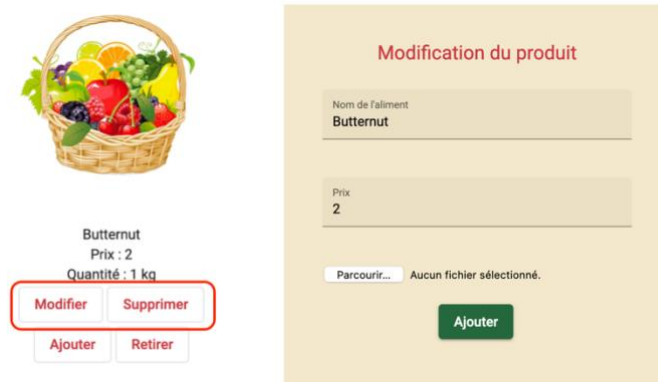
a. Partie Utilisateur

Voici la vue de nos deux objets gérés par Mango DB. D'abord, la partie utilisateurs nous permet de garder une trace des comptes créés lors de l'inscription, et surtout de permettre aux utilisateurs déjà inscrits de pouvoir se connecter.

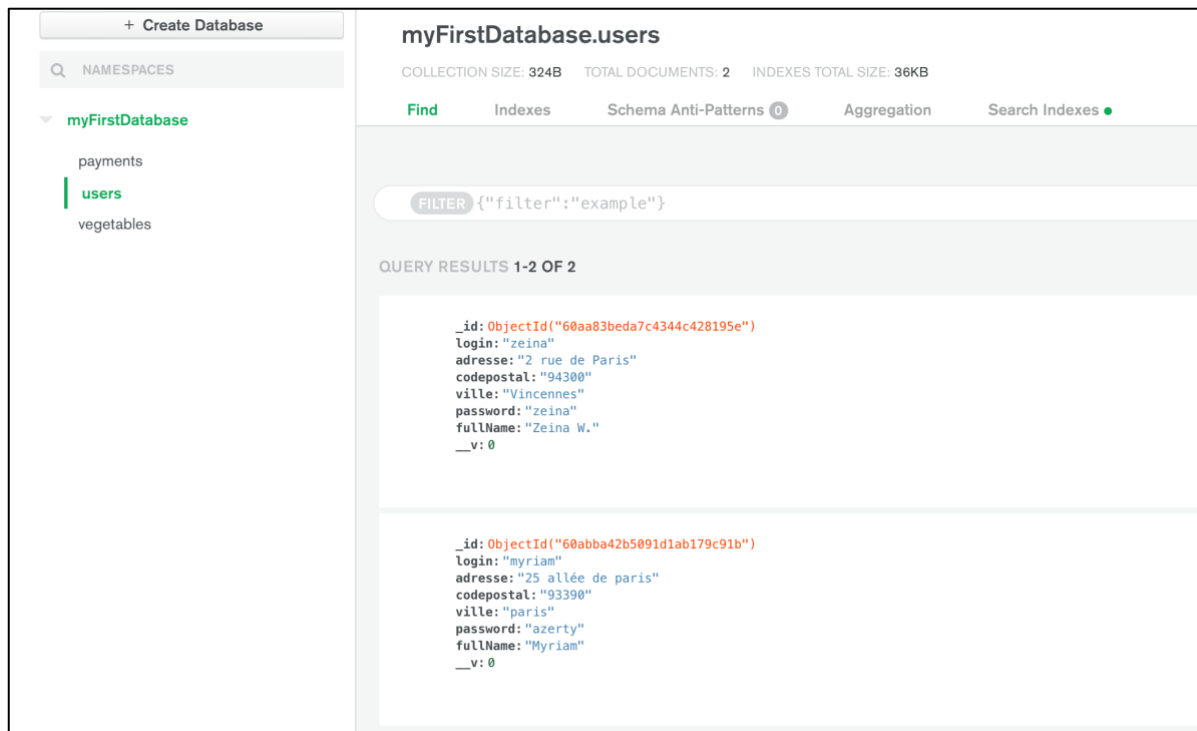


b. Partie vegetables

Ensuite, c'est dans la partie « vegetables » que se trouve l'ensemble des produits du catalogue du Petit Maraicher. Pour ajouter, modifier, supprimer un produit, cela se passe depuis le site du Petit maraicher, depuis la vue Administrateur. L'ajout se fait depuis l'onglet administration, et la modification/suppression est possible depuis la page du marché, sous chacun des produits :



Voici la vue dans MangoDB des produits, intitulées « vegetables » :



Lorsqu'on ajoute un produit au catalogue, on a vu qu'il fallait passer par l'onglet Administrateur sur l'application. On saisit le nom du produit, et son prix :

Ajouter un produit

Nom de l'aliment

Prune

Prix de l'aliment en €

2.99

Parcourir...

Aucun fichier sélectionné.

Ajouter

Dès que l'on clique sur « Ajouter », il s'ajoute à la liste des produits dans la base de données :



c. Partie Payment

Il s'agit ici de conserver les données bancaires des clients lorsqu'ils effectuent leurs commandes. Dès que le client valide le paiement, ses informations s'ajoutent dans la base de données.

+ Create Database

Q NAMESPACES

▼ myFirstDatabase

- payments
- users
- vegetables

myFirstDatabase.payments

COLLECTION SIZE: 264B TOTAL DOCUMENTS: 2 INDEXES TOTAL SIZE: 36KB

Find Indexes Schema Anti-Patterns ⓘ Aggregation Search Indexes ●

INSERT DOCUMENT

FILTER {"filter":"example"}

Find Reset

QUERY RESULTS 1-2 OF 2

>

```
_id: ObjectId("60ac071614acf57718e01435")
cnom: "Alexandre Banane"
cnum: "111-222-333-444"
cmois: "0ctobre"
cannee: "2024"
cvv: "222"
__v: 0
```

```
_id: ObjectId("60ac1728c46f9b2e94799c69")
cnom: "Robert Verger"
cnum: "4970 9833 4720 3342"
cmois: "Juin"
cannee: "2023"
cvv: "007"
__v: 0
```