Projet de Frameworks Front-End :

Site de E-commerce avec API

Encadreur :

M Kuassi Israel,
enseignant associé à l'ESMT

Étudiant :

Papa Omar SYLLA

Année universitaire 2021-2022, MI - ISI

Sommaire :

Introduction

Vue.js est un framework JavaScript pour le développement d'interfaces web. La première version a été publiée en 2014 par Evan You, un ancien de chez Google. Vue.js permet de construire son application par un assemblage de plusieurs composants. Ce composant permet de créer de nouvelles balises HTML avec un comportement spécifique. La finalité est de pouvoir facilement réutiliser un composant.

Dans le cadre de notre formation en frameworks frontend, nous allons réaliser un petit site de E-commerce. L'objectif de ce mini-projet est de permettre à chaque groupe de deux étudiants de développer une API REST de gestion de contacts avec Express JS et Mongo DB.

Et pour ce faire, D'abord nous allons effectuer la réalisation et montrerons le fonctionnement de l'API, ensuite nous ferons la réalisation et montrerons le fonctionnement du site.

1. Conception du modèle de données

Avant de réaliser le système il faut connaitre la SGBD utilisé et la structuration des données.

1.1. Mysql

MySQL est un Système de Gestion de Base de Données (SGBD) parmi les plus populaires au monde. Il est distribué sous double licence, un licence publique générale GNU et une propriétaire selon l'utilisation qui en est faites. La première version de MySQL est apparue en 1995 et l'outil est régulièrement entretenu.

MySQL fonctionne sur de nombreux systèmes d'exploitation (dont Linux, Mac OS X, Windows, Solaris, FreeBSD…) et qui est accessible en écriture par de nombreux langages de programmation, incluant notamment PHP, Java, Ruby, C, C++, .NET, Python …

Nous l'utiliserons donc pour alimenter l'API .

1.2 Structure des données

Dans le modèle de donnée nous identifions les entités suivantes :

user : qui permet d'identifier un utilisateur inscrit ;

user

nom de clé	Type de donnée	description
email	varchar(50), unique	identifiant de l'utilisateur
phone	varchar(45)	numéro de telephone
password	text	mot de passe
category	varchar(45), default 'marchand'	soit client soit vendeur
create_at	Timestamp, Default CURRENT_TIMESTAMP	date de création
updated_at	Timestamp, Default CURRENT_TIMESTAMP, ON UPDATE CURRENT_TIMESTAMP	date de modification

Titre : Structure du modèle user

profil : qui contient les informations personnelles de l'utilisateur ;

profil

nom de clé	Type de donnée	description
firstname	varchar(100)	prenom
lastname	varchar(100)	nom
age	int	Age

country	varchar(100)	pays
userId	varchar(100)	Identifiant de l'utilisateur
create_at	Timestamp, Default CURRENT_TIMESTAMP	date de création
updated_at	Timestamp, Default CURRENT_TIMESTAMP, ON UPDATE CURRENT_TIMESTAMP	date de modification

Titre : Structure du modèle profil

account : qui présente le compte de l'utilisateur , elle informe sur la somme qu'il détient et la devise ;

account

nom de clé	Type de données	description
Id_account	int, Auto incremental	identifiant du compte
balance	decimal(19,9), default '0'	Somme présent dans le compte
userId	varchar(100)	Identifiant de l'utilisateur
currency	varchar(45), default 'XOF'	Devise de la somme
create_at	Timestamp, Default CURRENT_TIMESTAMP	date de création
updated_at	Timestamp, Default CURRENT_TIMESTAMP,	date de modification

	ON UPDATE CURRENT_TIMESTAMP	

Titre : Structure du modèle account

article : qui représente un produit disponible à la vente ;

article

nom de clé	Type de donnée	description
id_article	int, Auto incremental	identifiant du produit
quantity	int	Quantité du produit
reference	varchar(100)	Nom du produit
unitPrice	decimal(19,9)	Montant de l'unité
sellerId	varchar(100)	Identifiant du vendeur
totalPrice	decimal(19,9)	Montant total
create_at	Timestamp, Default CURRENT_TIMESTAMP	date de création
updated_at	Timestamp, Default CURRENT_TIMESTAMP, ON UPDATE CURRENT_TIMESTAMP	date de modification

Titre : Structure du modèle article

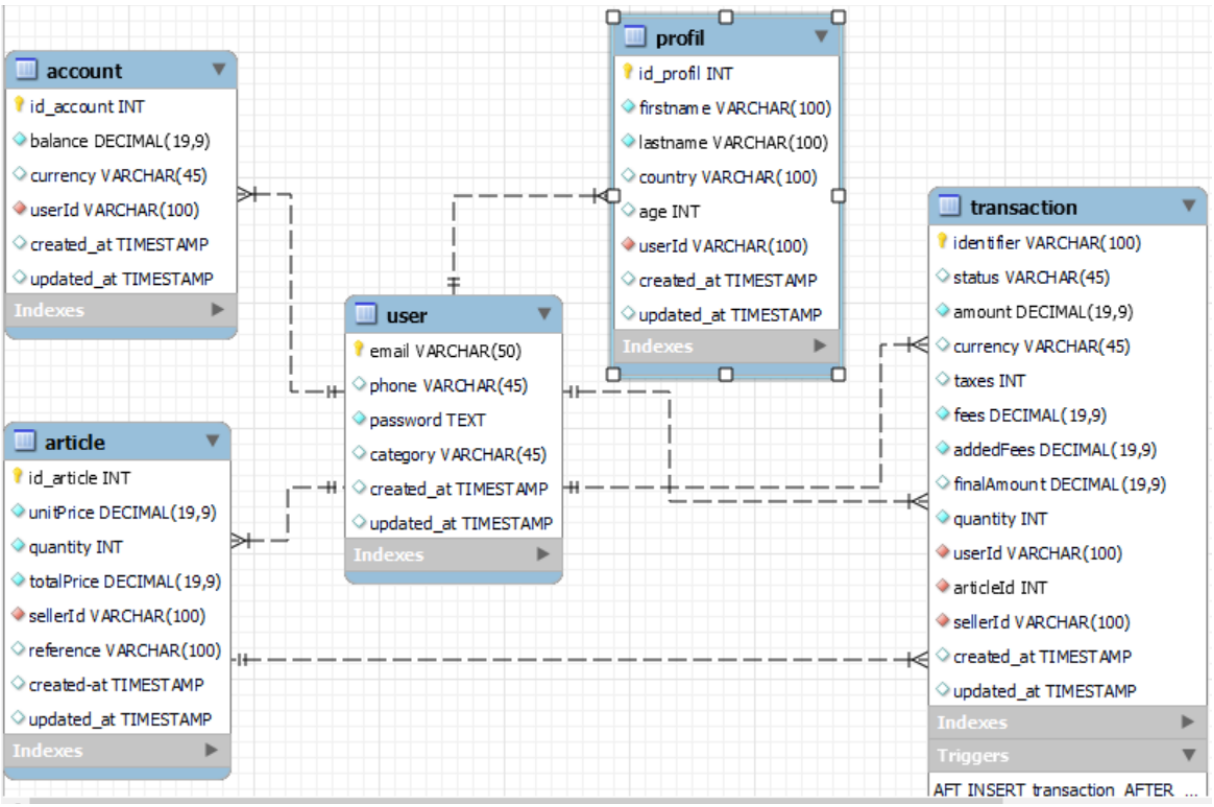
transaction : qui représente les transactions entre acheteur et vendeurs;

transaction

nom de clé	Type de donnée	description
identifier	varchar(100), not null	identifiant de la transaction
status	varchar(45)	Status de la transaction
amount	decimal(19,9)	Montant d'une unité du produit
currency	varchar(45)	Devise de la transaction
taxes	int	taxes
fees	decimal(19,9)	frais
addedfees	decimal(19,9)	Frais supplémentaires
finalAmount	decimal(19,9)	Calcul du montant total
quantity	int	Quantité du produit
userId	varchar(100)	Identifiant de l'acheteur
articleId	int	Identifiant du produit
sellerId	varchar(100)	Identifiant du vendeur
create_at	Timestamp, Default CURRENT_TIMESTAMP	date de création
updated_at	Timestamp, Default CURRENT_TIMESTAMP, ON UPDATE CURRENT_TIMESTAMP	date de modification

Titre : Structure du modèle transaction

Nous proposons donc le modèle de données suivant :



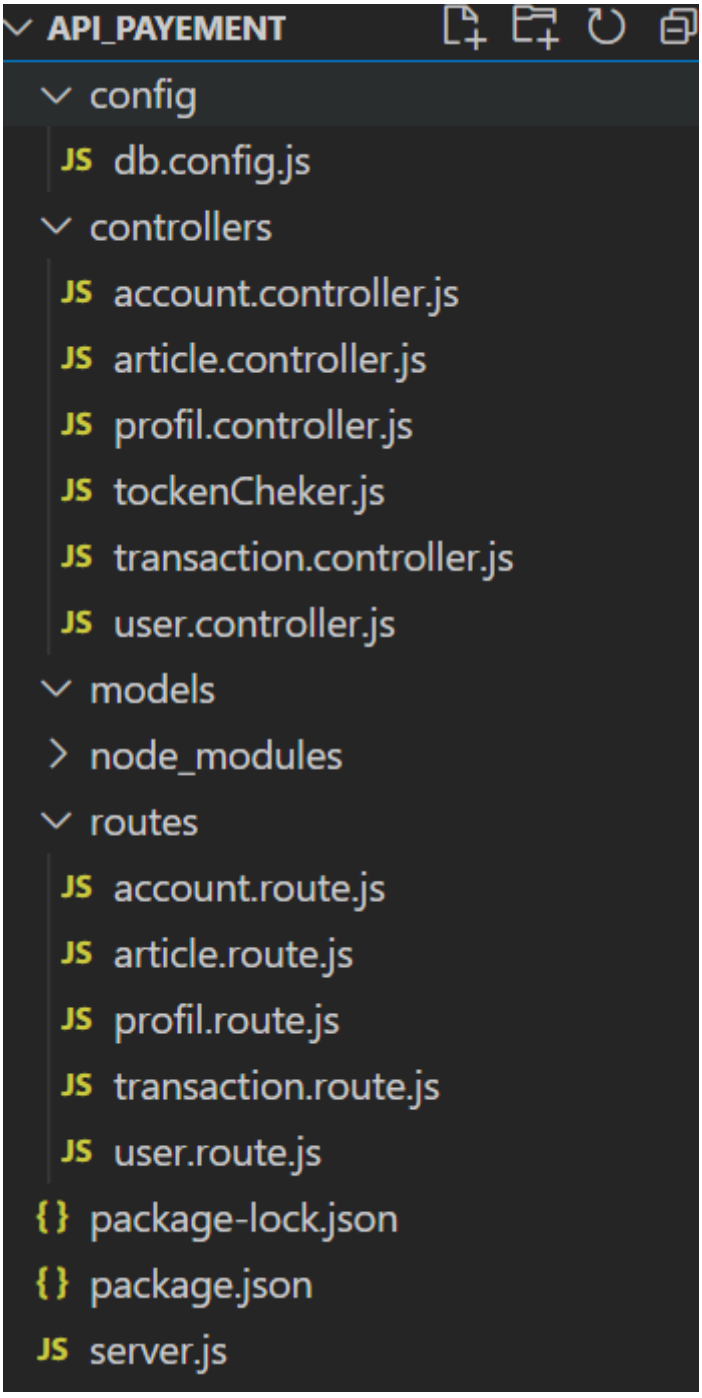
Titre : Structure du modèle de données

1. Réalisation et Fonctionnement de l'API

Pour réaliser la solution la première étape est l'installation de l'environnement.

1.1 Architecture du projet

Pour faciliter le débogage et rendre les composants réutilisables il est essentiel de décomposer les composants en modules :



Capture 5 : Architecture ou structure de l'API

Ici nous avons divisé le système en 4 modules distincts :

- Config : qui contient le fichier db.config.js qui exporte l'url de une connexion à notre base de donnée ;
- Models : qui contient les fichiers qui exportent les modèles de données ;

- Controllers : qui contient les fichiers qui définissent et exporte les opérations CRUD du modèle;
- Routes : qui contient des fichiers qui créent des routes qui permettent d'accéder aux opérations CRUD et les exportent ;

Server.js est le point d'entrée du server donc il utilise les modules pour configurer le server .

1.2 Captures du code

A. db.config.js

config > JS db.config.js > [@] <unknown>

```
1   const mysql = require('mysql')
2   db = mysql.createConnection({
3     host: 'localhost',
4     user: 'root',
5     password: 'user',
6     database: 'payment'
7   })
8
9
10
11  console.log("type de db:"+typeof(db));
12  console.log("detail de db:"+db);
13
14
15  tokenConf = {
16    "secret": "(-è_àç - secret - 87392",
17    "tokenLife": 900000,
18  }
19  module.exports = {
20    db,
21    tokenConf
22  }
```

B. server.js

JS server.js > app.use() callback

```
1   // Importation du module express pour pouvoir l'utilisation
2   const express = require("express");
3   const conf = require("./config/db.config");
4   const app = express();
5
6   app.use(express.urlencoded({ extended: false }));
7   app.use(express.json({ extended: false }));
8
9   app.use(function (req, res, next) {
10    res.header("Access-Control-Allow-Origin", "*");
11    res.header(
12      "Access-Control-Allow-Headers",
13      "Origin, X-Requested-With, Content-Type, Accept, Authorization"
14    );
15    next();
16  });
17
18  require("./routes/account.route")(app);
19
20  require("./routes/article.route")(app);
21
22  require("./routes/profil.route")(app);
23
24  require("./routes/transaction.route")(app);
25
26  require("./routes/user.route")(app);
27
28  // set port, listen for requests
29  const PORT = process.env.PORT || 8089;
30  app.listen(PORT, () => {
31    console.log(`Server is running on port ${PORT}.`);
32  });
33
```

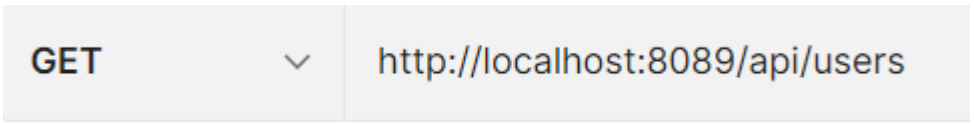
```
JS server.js > ...
29
30  // Ajouter les routes de contact.route
31  require("./routes/contact.route")(app);
32
33
34  // set port, listen for requests
35  const PORT = process.env.PORT || 8080;
36  app.listen(PORT, () => {
37    console.log(`Server is running on port ${PORT}.`);
38  });
39
```

1.3 Fonctionnement de l'API

A. Points de terminaisons

Montrons le fonctionnement des endpoints utilisés.

- Retourner tous les utilisateurs
  - Endpoint :

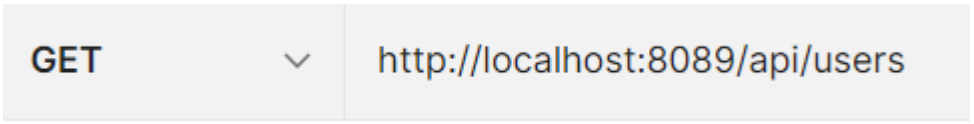


  - Résultat :

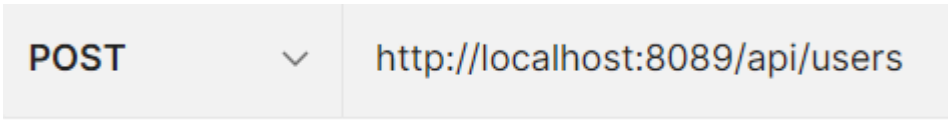

```
{
    "status": 200,
    "data": [
        {
            "email": "atrona8@gmail.com",
            "phone": "0785825702",
            "password": "$2b$10$yTCuifzonr8pnrZN.PiDSOHuL1l8.u9FM47raxrZbPREMRPmNFFqS",
            "category": "client",
            "created_at": "2022-08-03T05:49:52.000Z",
            "updated_at": "2022-08-03T05:49:52.000Z"
        },
```

```
            {
                "email": "test",
                "phone": "712234455",
                "password": "$2b$10$yNtNA7Fjzv0lvH1QUKBSTO9aR./WKtk6FdmkwZ7UMymGW
unRzcS16",
                "category": "client",
                "created_at": "2022-07-29T01:17:32.000Z",
                "updated_at": "2022-07-29T01:17:32.000Z"
            }
        ],
        "message": "User lists retrieved successfully"
}
```


- Créer un utilisateur
    - Endpoint :



    - Résultat :
- Retourner tous les utilisateurs
    - Endpoint :



    - Paramétres (body) :
        - password:testtest
        - email:test13
        - phone:712234455
        - category:client
        - password_confirmation:testtest
    -
    - Résultat :

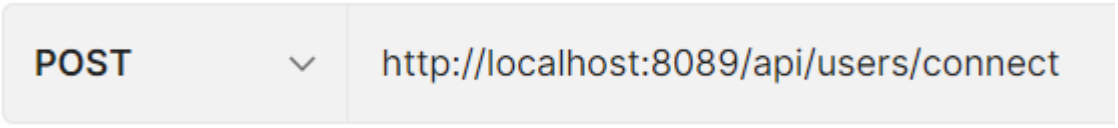

```
{
    "status": 200,
    "data": {
        "fieldCount": 0,
        "affectedRows": 1,
        "insertId": 0,
        "serverStatus": 2,
        "warningCount": 0,
```

```
            "message": "",
            "protocol41": true,
            "changedRows": 0
        },
        "message": "New user added successfully"
    }
```


- Retourner l’utilisateur si les identifians sont correctes

Si le système ne le trouve pas une resonse au status 400 et détailles de l’erreur sont retournés

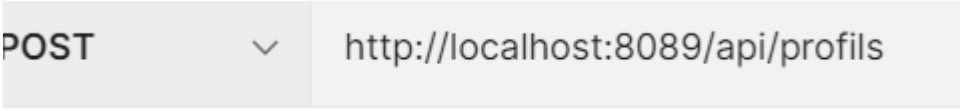
- Endpoint :



- paramètres :
  - password:test
  - email:test
- Résultat :


```
{
    "status": 201,
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6InRlc3QiLCJwYXNzd29yZCI6InRlc3QiLCJpYXQiOjE2NTk1MzI0MTcsImV4cCI6MTY2MDQzMjQxN30.vOu9nYml0tvR1BzlMUNmT_uH-LGVpd-o2fXEl9024b4",
    "user": {
        "email": "test",
        "phone": "712234455",
        "password": "$2b$10$yNtNA7Fjzv0lvH1QUKBSTO9aR./WKtk6FdmkwZ7UMymGWunRzcS16",
        "category": "client",
        "created_at": "2022-07-29T01:17:32.000Z",
        "updated_at": "2022-07-29T01:17:32.000Z"
    },
    "error": false,
    "message": "User test connected successfully"
}
```

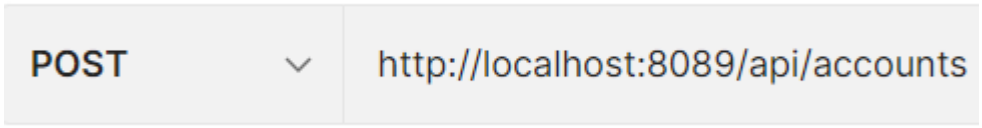

- Créer un profil
  - Endpoint :



- paramètres :
  - firstname:nom modif
  - lastname:test
  - country:71223445
  - age:18
  - userid:test12
- Résultat :


```
{
    "status": 200,
    "data": {
        "fieldCount": 0,
        "affectedRows": 1,
        "insertId": 74,
        "serverStatus": 2,
        "warningCount": 0,
        "message": "",
        "protocol41": true,
        "changedRows": 0
    },
    "message": "New profil added successfully"
}
```


- Créer le compte d’un utilisateur
  - Endpoint :



  - Paramètre :
    - userid:test12
    - balance:1000000
  - Résultat :

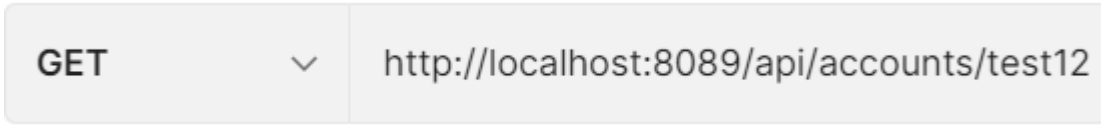

```
{
    "status": 200,
    "data": {
        "fieldCount": 0,
        "affectedRows": 1,
        "insertId": 11,
        "serverStatus": 2,
        "warningCount": 0,
        "message": "",
        "protocol41": true,
        "changedRows": 0
    },
    "message": "New account added successfully"
```

- }
    -

- Retourner le compte d’un utilisateur

Si le système ne le trouve pas une resonse au status 400 et détailles de l’erreur sont retournés

- Endpoint :

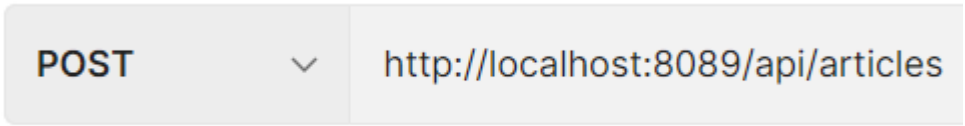


- Résultat :

- {
- "status": 200,
- "data": [
- {
- "id_account": 11,
- "balance": 1000000,
- "currency": null,
- "userId": "test12",
- "created_at": "2022-08-03T13:22:36.000Z",
- "updated_at": "2022-08-03T13:22:36.000Z"
- }
-],
- "message": "account test12 retrieved successfully"
- }

    -

- Créer un article
    - Endpoint :

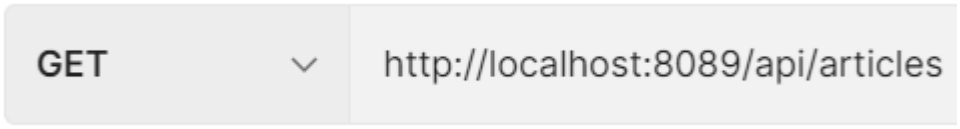


    - Paramètres :
        - reference:Produit 500
        - unitprice:1000
        - quantity:7
        - totalprice:7000
        - sellerid:test
    - Résultat :

- {

```
    "status": 200,
    "data": {
        "fieldCount": 0,
        "affectedRows": 1,
        "insertId": 9,
        "serverStatus": 2,
        "warningCount": 0,
        "message": "",
        "protocol41": true,
        "changedRows": 0
    },
    "message": "New article added successfully"
}
```


- Retourner toutes les articles
  - Endpoint :

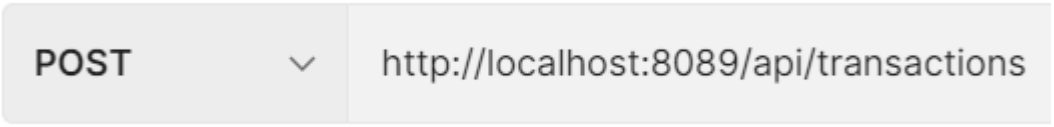


  - Résultat :


```
{
    "status": 200,
    "data": [
        {
            "id_article": 5,
            "unitPrice": 1000,
            "quantity": 0,
            "totalPrice": 2000,
            "sellerId": "test",
            "reference": "Nom article",
            "created-at": "2022-08-03T06:37:49.000Z",
            "updated_at": "2022-08-03T08:14:46.000Z"
        },
        {
            "id_article": 6,
            "unitPrice": 1000,
            "quantity": 7,
            "totalPrice": 2000,
            "sellerId": "test",
            "reference": "Iphone 19",
            "created-at": "2022-08-03T10:42:32.000Z",
            "updated_at": "2022-08-03T10:42:32.000Z"
        },
        {
            "id_article": 7,
            "unitPrice": 900,
            "quantity": 3,
            "totalPrice": 2700,
```

```
            "sellerId": "test",
            "reference": "Adidas azertyu",
            "created-at": "2022-08-03T10:43:22.000Z",
            "updated_at": "2022-08-03T10:43:22.000Z"
        },
        {
            "id_article": 8,
            "unitPrice": 900,
            "quantity": 3,
            "totalPrice": 2700,
            "sellerId": "test",
            "reference": "Mac",
            "created-at": "2022-08-03T10:52:00.000Z",
            "updated_at": "2022-08-03T10:52:00.000Z"
        },
        {
            "id_article": 9,
            "unitPrice": 1000,
            "quantity": 7,
            "totalPrice": 7000,
            "sellerId": "test",
            "reference": "Produit 500",
            "created-at": "2022-08-03T13:35:54.000Z",
            "updated_at": "2022-08-03T13:35:54.000Z"
        }
    ],
    "message": "article lists retrieved successfully"
}
```


- Créer un transaction
  - Endpoint :



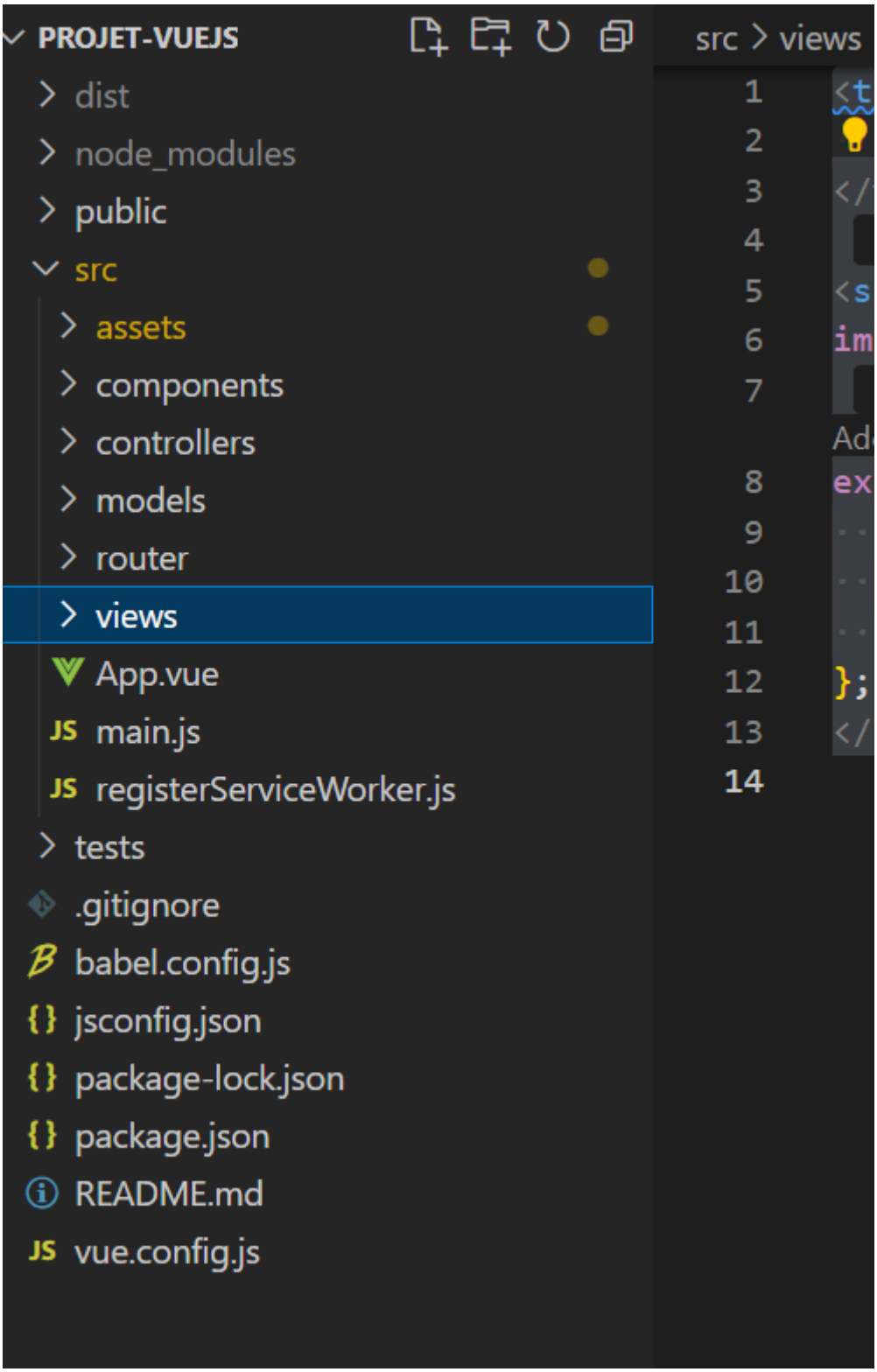
  - paramètres :
    - identifier:45364T76GHJI7
    - amount:1000
    - currency:F
    - taxes:18
    - fees:10
    - addedfees:10
    - quantity:0
    - userid:atrona8@gmail.com

        - articleid:5
        - sellerid:test
    - Résultat :


```
{
    "status": 200,
    "data": {
        "fieldCount": 0,
        "affectedRows": 1,
        "insertId": 0,
        "serverStatus": 34,
        "warningCount": 0,
        "message": "",
        "protocol41": true,
        "changedRows": 0
    },
    "message": "New transaction added successfully"
}
```


2. Réalisation et Fonctionnement du site

3.1 Architecture



Titre : Architecture du projet vuejs

Ici le projet est séparé en 6 parties principaux :

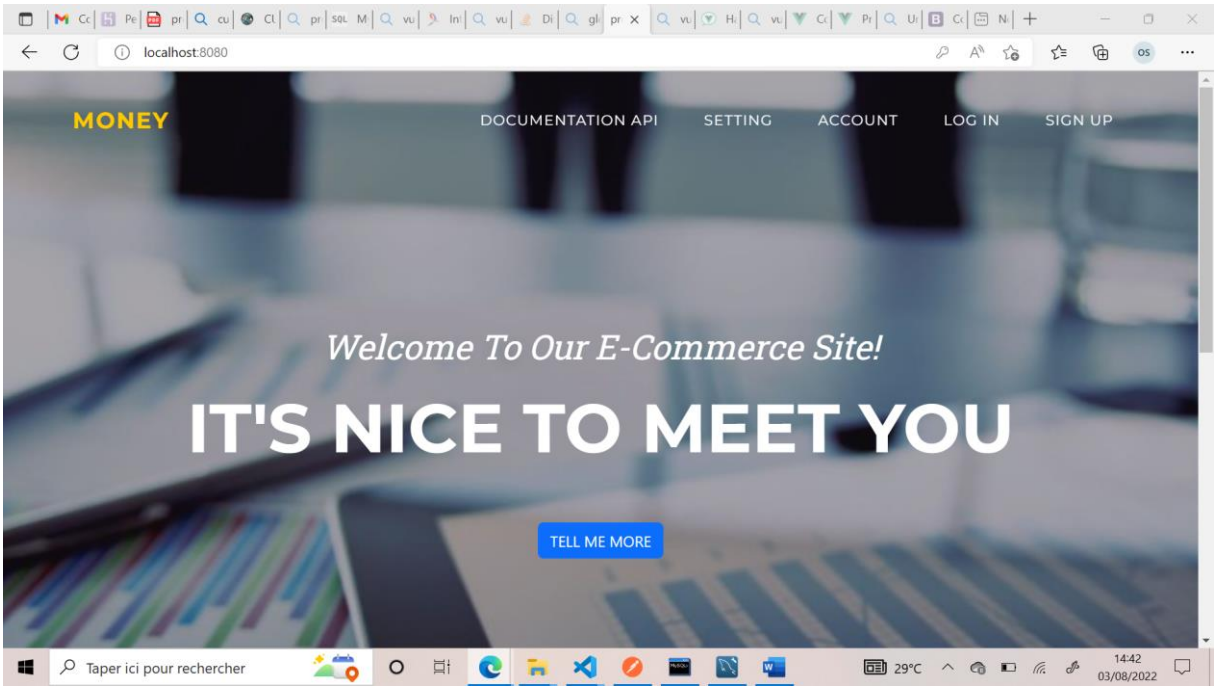
- Assets : qui contient des fichiers de style et des images;
- Models : qui contient les différents modèles et les exporte ;

- Controllers : qui contient les fichiers qui contiennent les fonction CRUD;
- Router : il crée des routes qui permettent d'accéder aux parties du site;
- Components : contient les composants vuejs ;
- Vuews : contient les fichiers responsable de la disposition des composants

3.2 Fonctionnement

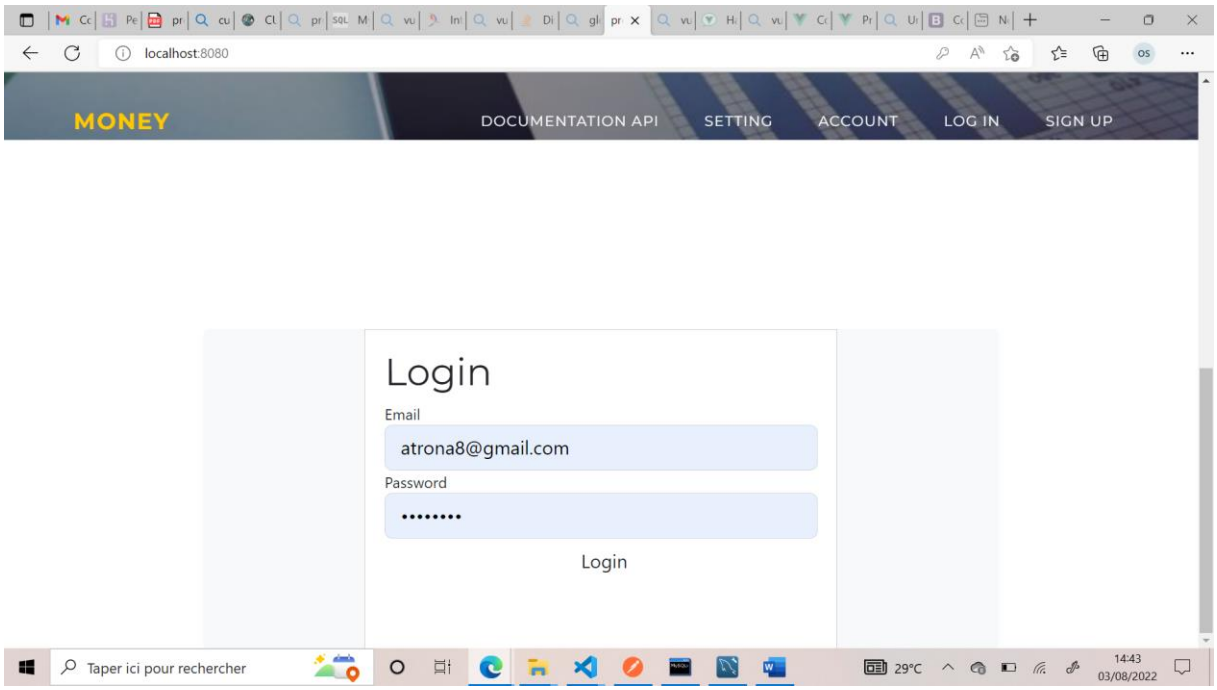
Nous montrerons les fonctionnalités du site avec l'aide de captures :

A. Connexion



Capture : page d'accueil

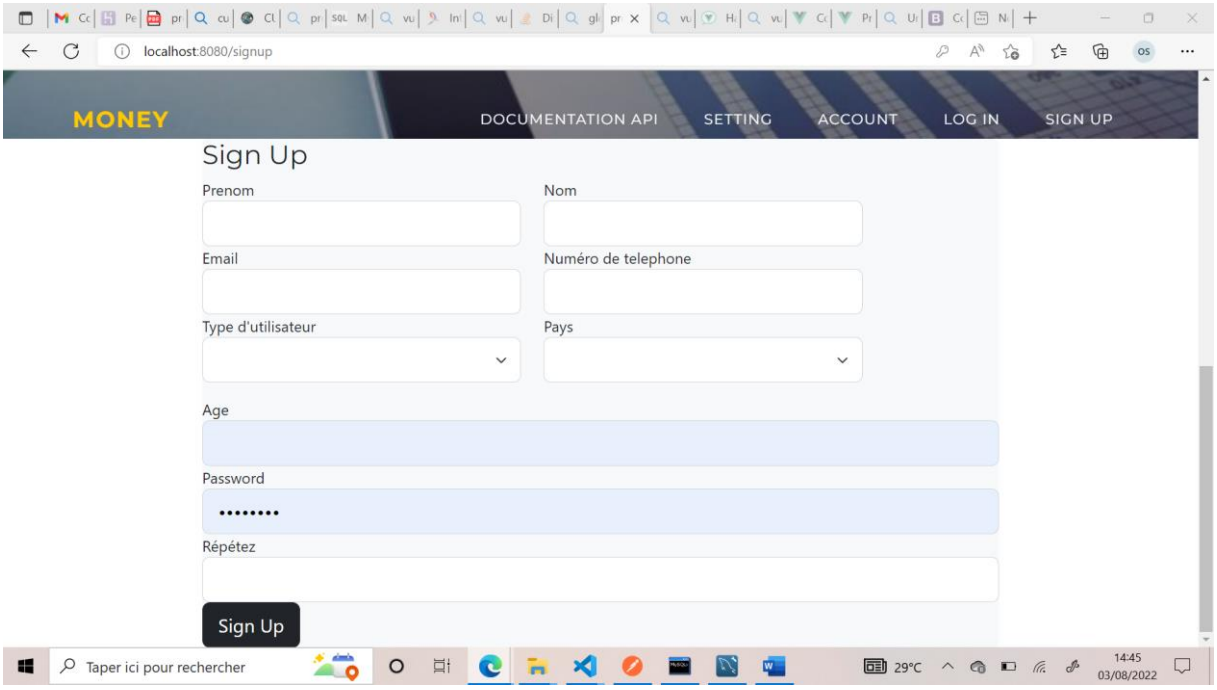
B. Inscription



Capture : page de connexion

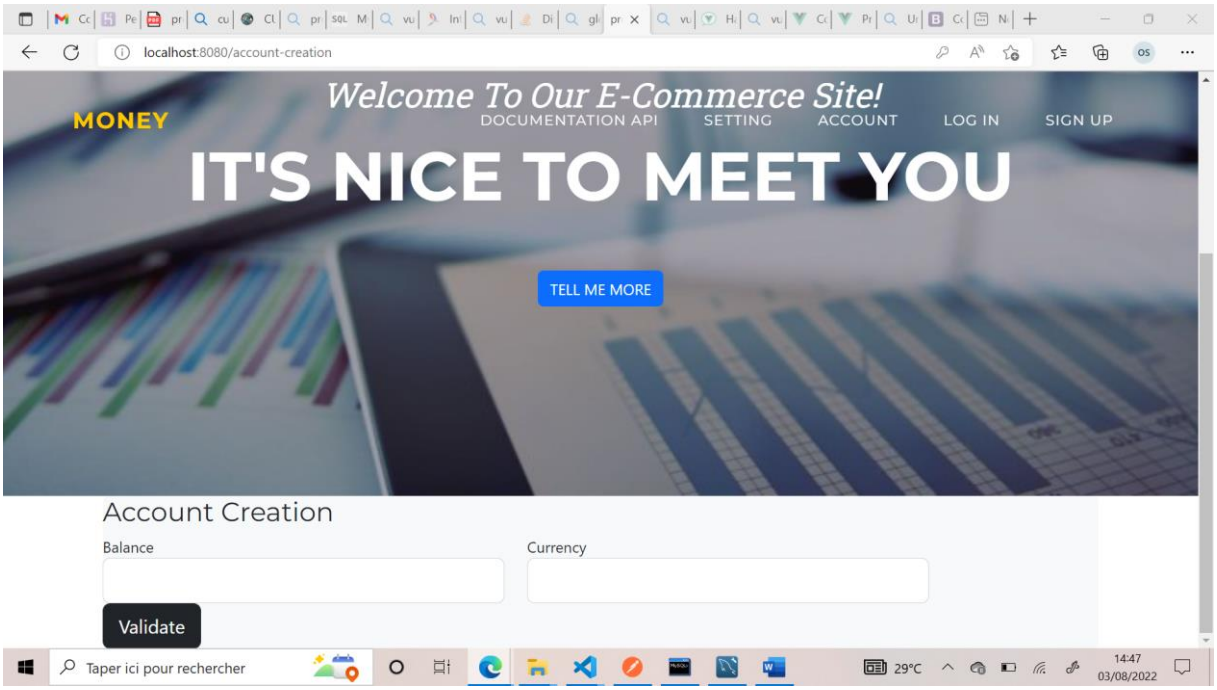
Dans la page de connexion l'utilisateur est obligé de renseigner son email et mot de passe, puis le système vérifie que les informations sont correctes, si un utilisateur 'client' est identifié il est redirigé vers la liste des articles, sinon, si un utilisateur 'marchant est identifié il est redirigé vers la liste des articles sinon un message d'erreur « The email and / or password is incorrect » apparait.

C. Ajout compte



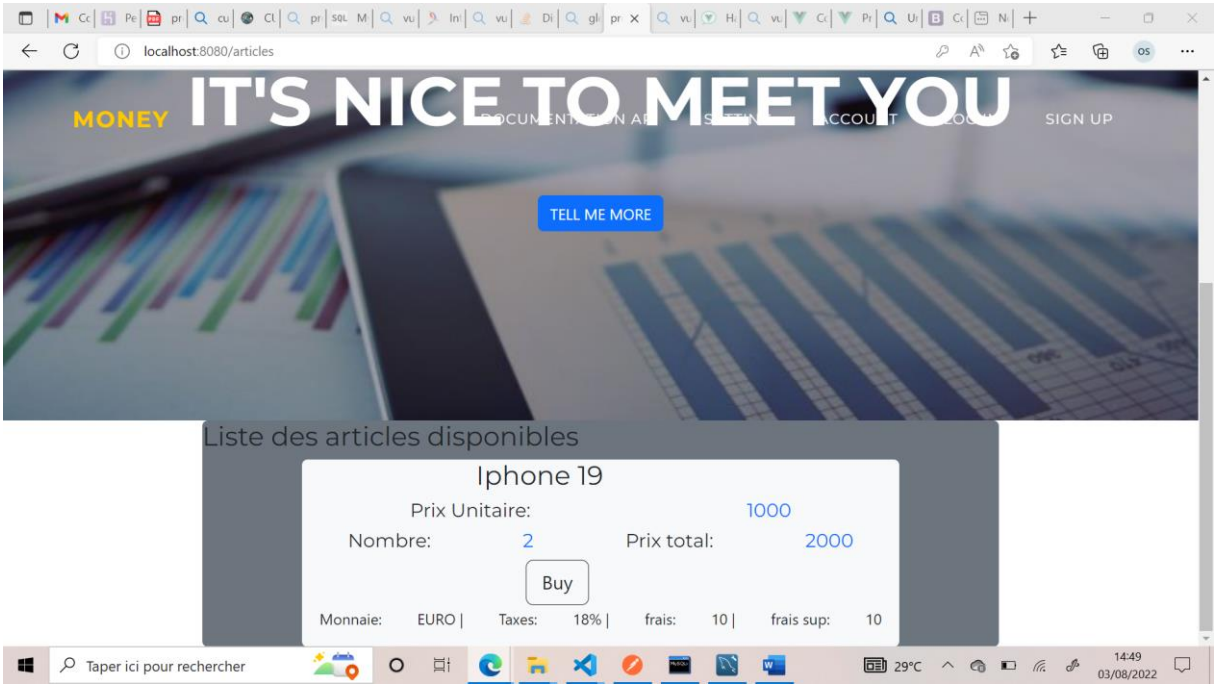
Capture : page d’inscription

Dans la page d’inscription d’utilisateur renseigne ses informations personnelles, puis si l’utilisateur est enregistré avec succès, il est redirigé dans la page de modification du ‘account’



Capture : page de modification du ‘account’

D. Achat article

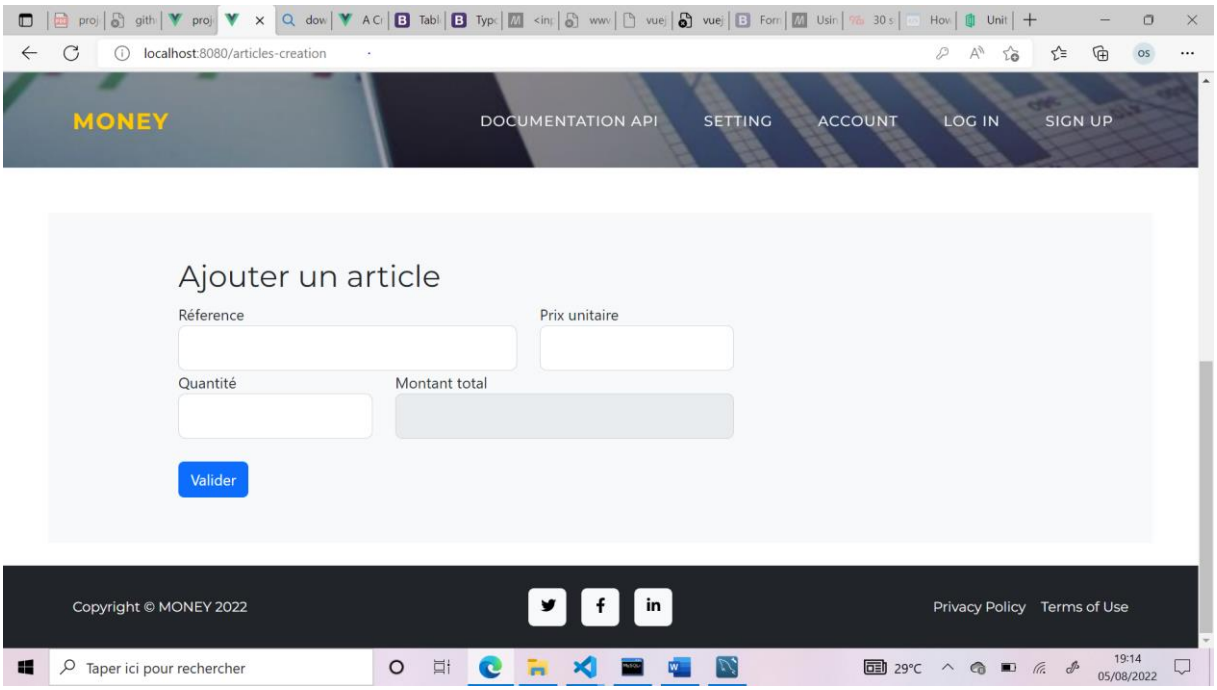


Capture : page d'achat d'article

Ici, le client peux choisir d'acheter un article, puis le système vérifie si la somme qu'il a suffi, si oui la transaction est créée, sinon une alerte lui informe que son solde est insuffisant .

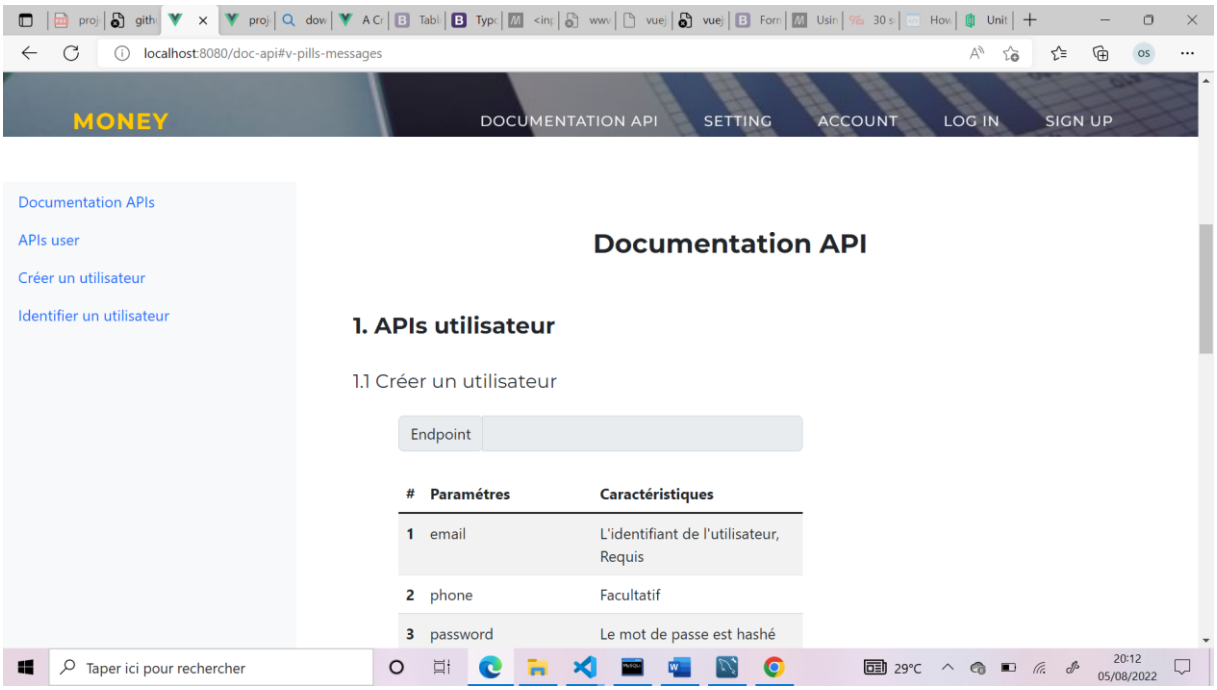
E. Ajout article

Si c'est un marchant qui s'est authentifié, il est dirigé vers la page de création de produit, où il peut informer du nom du produit, du prix unitaire et de la quantité, le montant total les calculé en même temp qu'il remplit.



Capture : page de création d'article

F. Documentation APIs



Capture : page de documentation d'API

3. Annexe

3.3. Liste des ressources utilisés

RESSOURCE	ROLE
NODEJS	Framework backend du site
VUE-CLI	Framework frontend du site
EXPRESSJS	Gestionnaire de packages
AXIOS	Package de gestion de requêtes HTTP
BOOTSTRAP	Framework css
NODE-SASS	Librairie de binding entre nodejs et le Sass
PATH	Libraisie de gestion de ‘path’
JEST	Librairie de gestion des test

3.4. Difficultés rencontrés et solutions

DIFFICULTES	SOLUTIONS
MANQUE DE TEMPS	Se concentrer sur des objectifs réalisables
LACUNES EN JAVASCRIPT ET CSS	Consulter régulièrement les documentations
BESOIN DE DISPOSER DES INFORMATIONS DE L’UTILISATEUR CONNECTE SUR L’ENSEMBLE DU SITE	Consulter la documentation de vuejs sur les variables globales
ERREURS	Consulter les documentations

3.5. Acquis

ACQUIS	LIEN DOCUMENTATION
RENFORCEMENT DES CONNAISSANCES EN BOOTSTRAP	Introduction · Bootstrap (getbootstrap.com)
RENFORCEMENT DES CONNAISSANCES EN VUEJS	Introduction | Vue.js (vuejs.org)

DECOUVERTE DE HEROKU & VUEJS	Heroku Node.js Support | Heroku Dev Center
TESTS UNITAIRES AVEC VUEJS	A Crash Course | Vue Test Utils (vuejs.org)