

|  |
| --- |
| 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,  MPSI - ISI

## **Sommaire :**

1. Réalisation et Fonctionnement de l’API………………………………………………...
2. Réalisation et Fonctionnement du site……………………………………………………

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. 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 | description |
| email | identifiant de l’utilisateur |
| phone | numéro de telephone |
| password | mot de passe |
| category | soit client soit vendeur |
| create\_at | date de création |
| updated\_at | date de modification |

Titre : Structure du modèle user

**profil** : qui contient les informations personnelles de l’utilisateur ;

|  |  |
| --- | --- |
| profil | |
| nom de clé | description |
| firstname | prenom |
| lastname | nom |
| age | Age |
| country | pays |
| userId | Identifiant de l’utilisateur |
| category | soit client soit vendeur |
| create\_at | date de création |
| updated\_at | 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 | description |
| Id\_account | identifiant du compte |
| balance | Somme présent dans le compte |
| userId | Identifiant de l’utilisateur |
| currency | Devise de la somme |
| create\_at | date de création |
| updated\_at | date de modification |

Titre : Structure du modèle account

**article** : qui représente un produit disponible à la vente ;

|  |  |
| --- | --- |
| article | |
| nom de clé | description |
| id\_article | identifiant du produit |
| quantity | Quantité du produit |
| reference | Nom du produit |
| unitPrice | Montant de l’unité |
| sellerId | Identifiant du vendeur |
| totalPrice | Montant total |
| create\_at | date de création |
| updated\_at | date de modification |

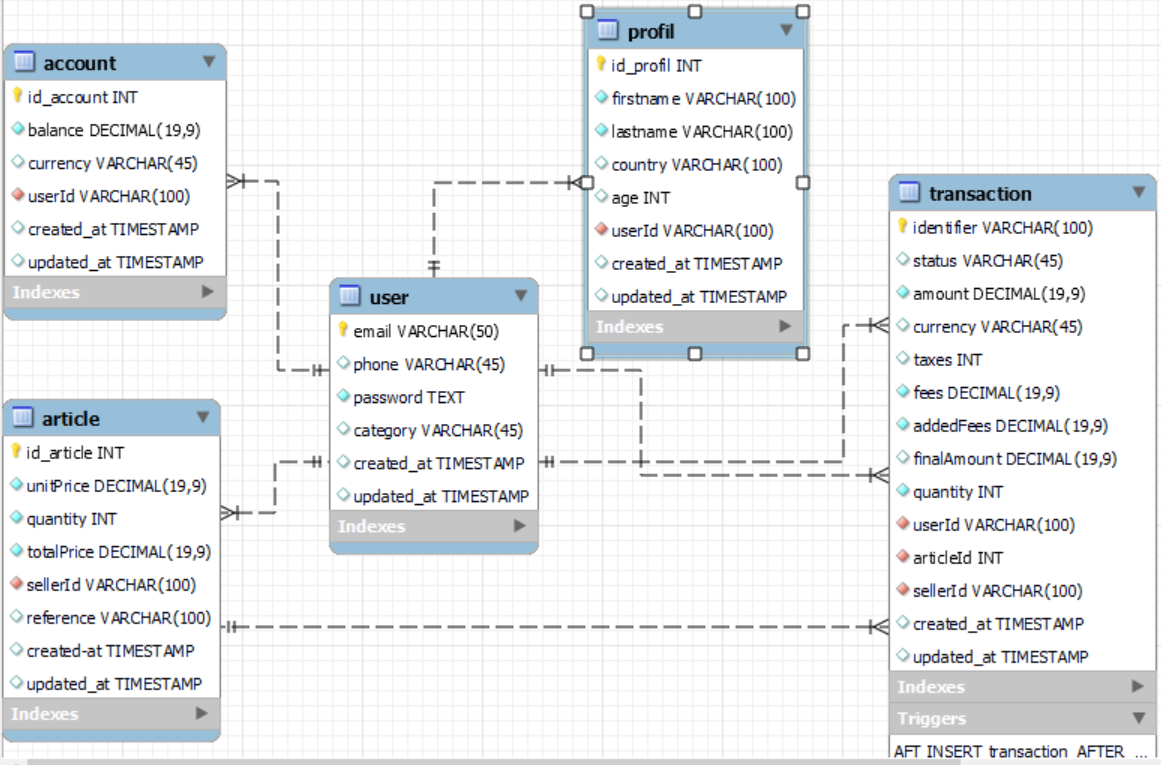
Titre : Structure du modèle article

**transaction** : qui représente les transactions entre acheteur et vendeurs;

|  |  |
| --- | --- |
| transaction | |
| nom de clé | description |
| identifier | identifiant de la transaction |
| status | Status de la transaction |
| amount | Montant d’une unité du produit |
| currency | Devise de la transaction |
| taxes | taxes |
| fees | frais |
| addedfees | Frais supplémentaires |
| finalAmount | Calcul du montant total |
| quantity | Quantité du produit |
| userId | Identifiant de l’acheteur |
| articleId | Identifiant du produit |
| sellerId | Identifiant du vendeur |
| create\_at | date de création |
| updated\_at | date de modification |

Titre : Structure du modèle transaction

Nous proposons donc le modèle de données suivant :



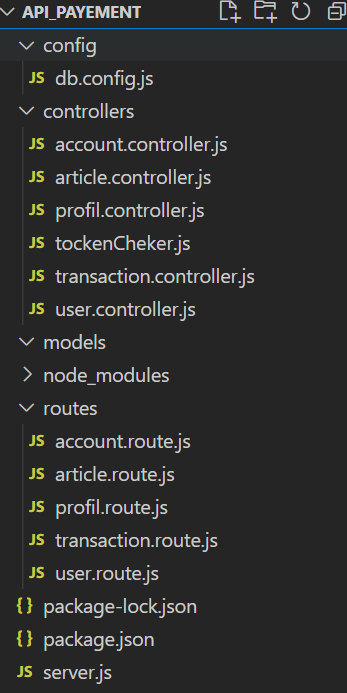
Titre : Structure du modèle de données

1. Réalisation et Fonctionnement de l’API

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

* 1. 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 ;

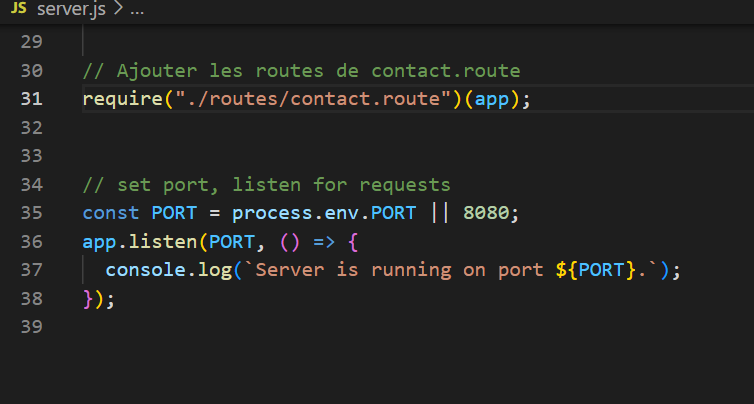
Server.js est le point d’entrée du server donc il utilise les modules pour configurer le server .

* 1. Captures du code

1. db.config.js



1. server.js

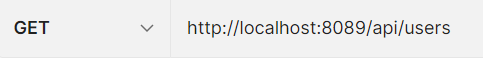


* 1. Fonctionnement de l’API

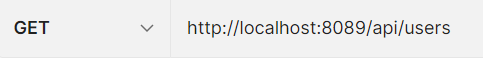
1. Points de terminaisons

Montrons le fonctionnement des endpoints utilisés.

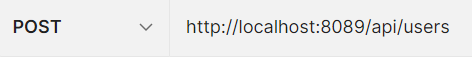
* Retourner tous les utilisateurs
  + Endpoint :



* + 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./WKtk6FdmkwZ7UMymGWunRzcS16",
* "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 :



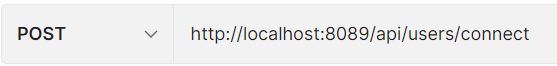
* + Résultat :
* Retourner tous les utilisateurs
  + Endpoint :



* + 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

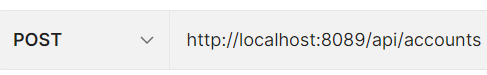
* + Endpoint :



* + 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 :



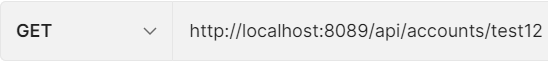
* + paramètres :
    - firstname:nom modif
    - lastname:test
    - country:71223445
    - age:18
    - userid:test12
  + Résultat :
* {
* "status": 200,
* "data": {
* "fieldCount": 0,
* "affectedRows": 1,
* "insertId": 74,
* "serverStatus": 2,
* "warningCount": 0,
* "message": "",
* "protocol41": **true**,
* "changedRows": 0
* },
* "message": "New profil added successfully"
* }
* Créer le compte d’un utilisateur
  + Endpoint :



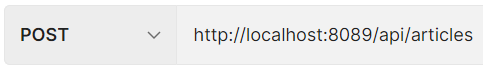
* + Paramètre :
    - userid:test12
    - balance:1000000
  + Résultat :
* {
* "status": 200,
* "data": {
* "fieldCount": 0,
* "affectedRows": 1,
* "insertId": 11,
* "serverStatus": 2,
* "warningCount": 0,
* "message": "",
* "protocol41": **true**,
* "changedRows": 0
* },
* "message": "New account added 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

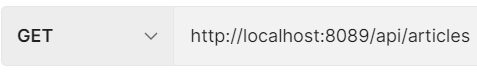
* + Endpoint :

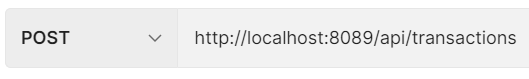


* + Résultat :
* {
* "status": 200,
* "data": [
* {
* "id\_account": 11,
* "balance": 1000000,
* "currency": **null**,
* "userId": "test12",
* "created\_at": "2022-08-03T13:22:36.000Z",
* "updated\_at": "2022-08-03T13:22:36.000Z"
* }
* ],
* "message": "account test12 retrieved successfully"
* }
* Créer un article
  + Endpoint :

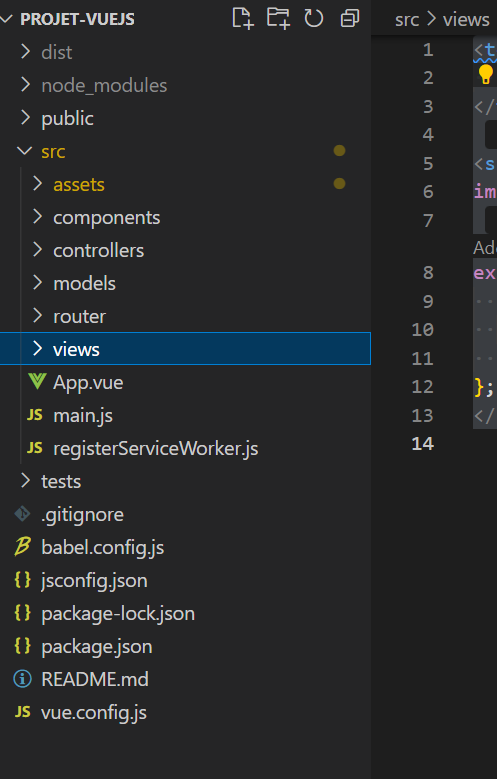


* + Paramètres :
    - reference:Produit 500
    - unitprice:1000
    - quantity:7
    - totalprice:7000
    - sellerid:test
  + Résultat :
* {
* "status": 200,
* "data": {
* "fieldCount": 0,
* "affectedRows": 1,
* "insertId": 9,
* "serverStatus": 2,
* "warningCount": 0,
* "message": "",
* "protocol41": **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"
* }

1. Réalisation et Fonctionnement du site
   1. Architecture



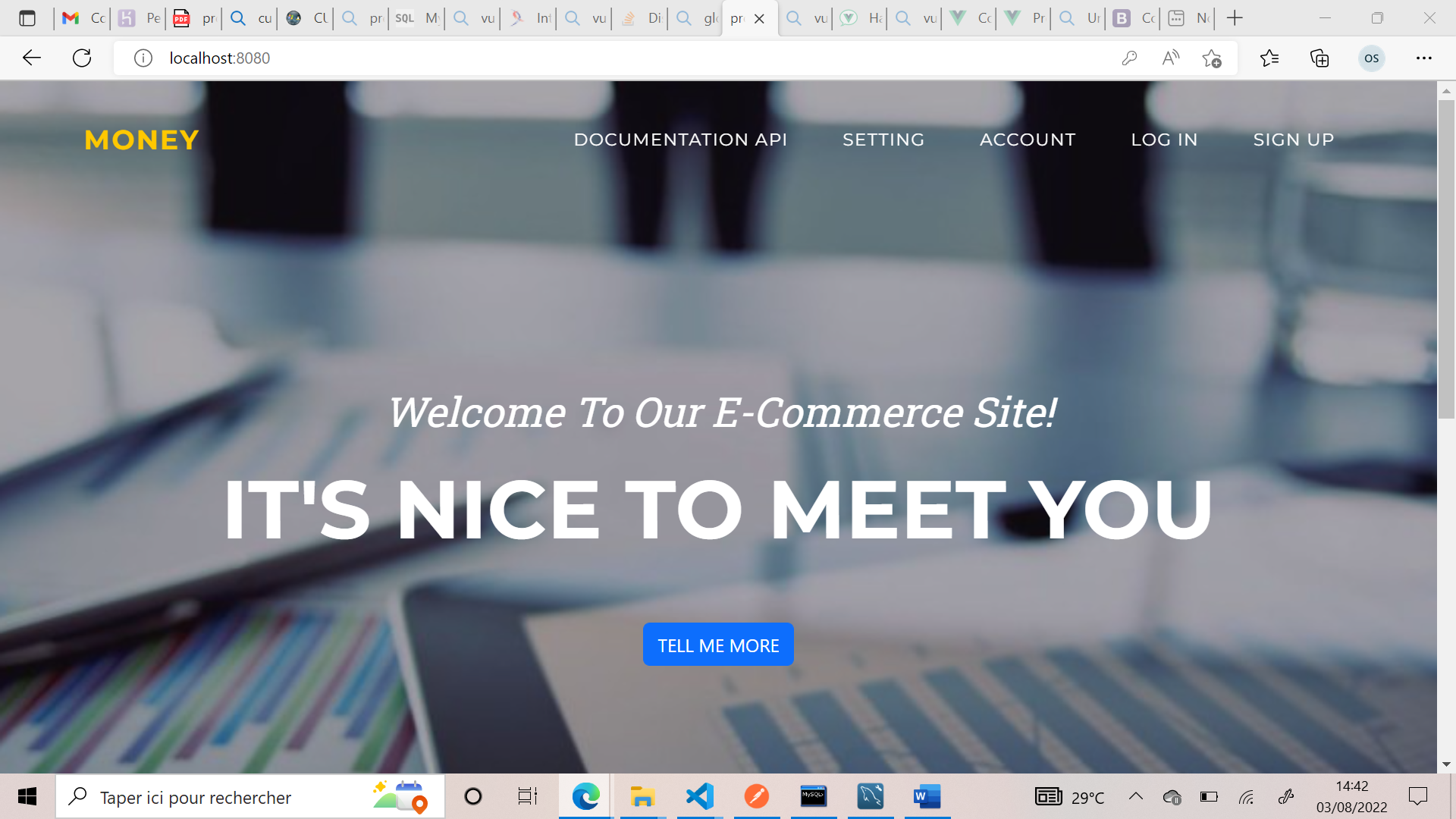
Titre : Architecture du projet vuejs

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
  1. Fonctionnement

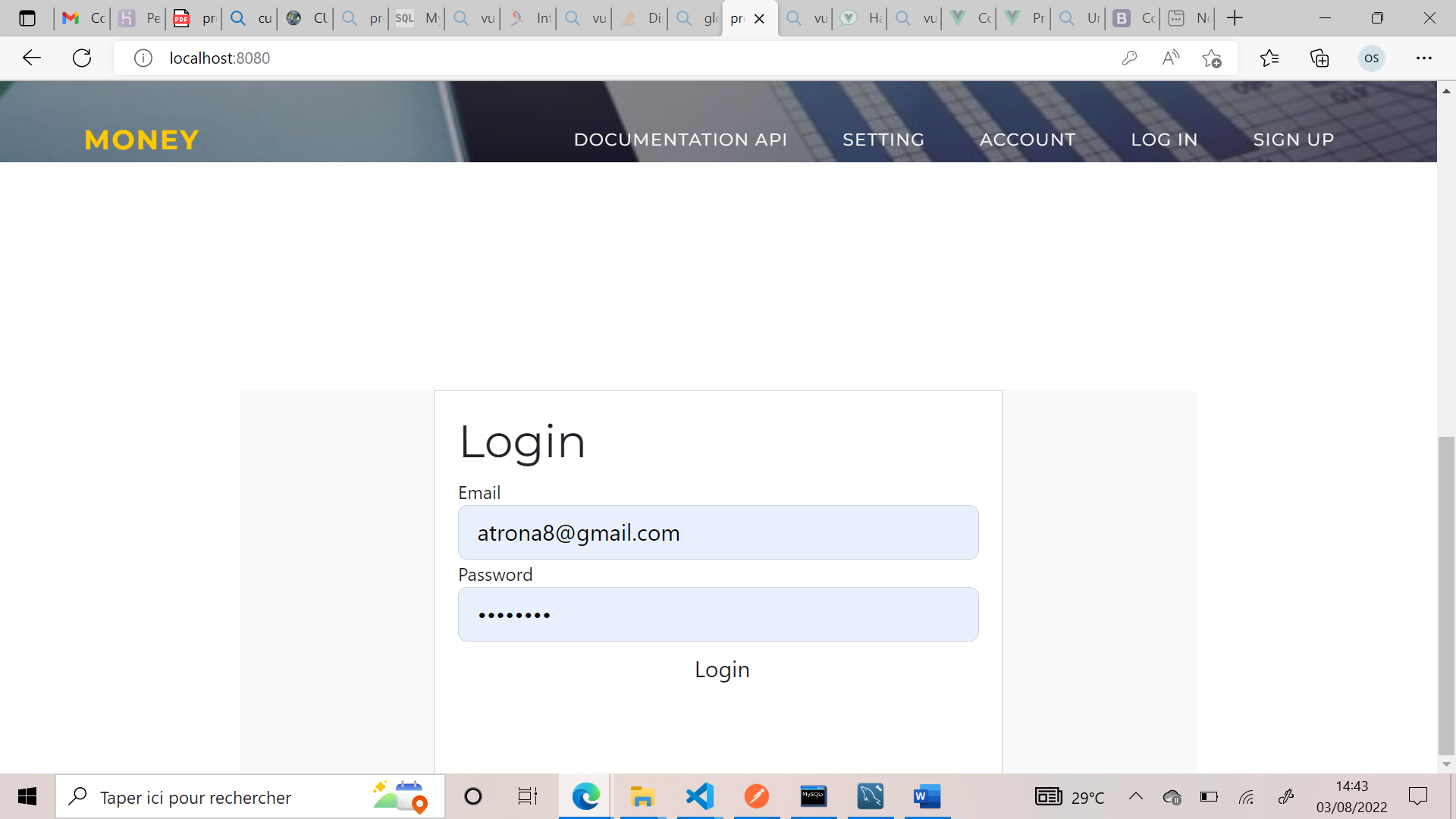
Nous montrerons les fonctionnalités du site avec l’aide de captures :

1. Connexion



Capture : page d’accueil

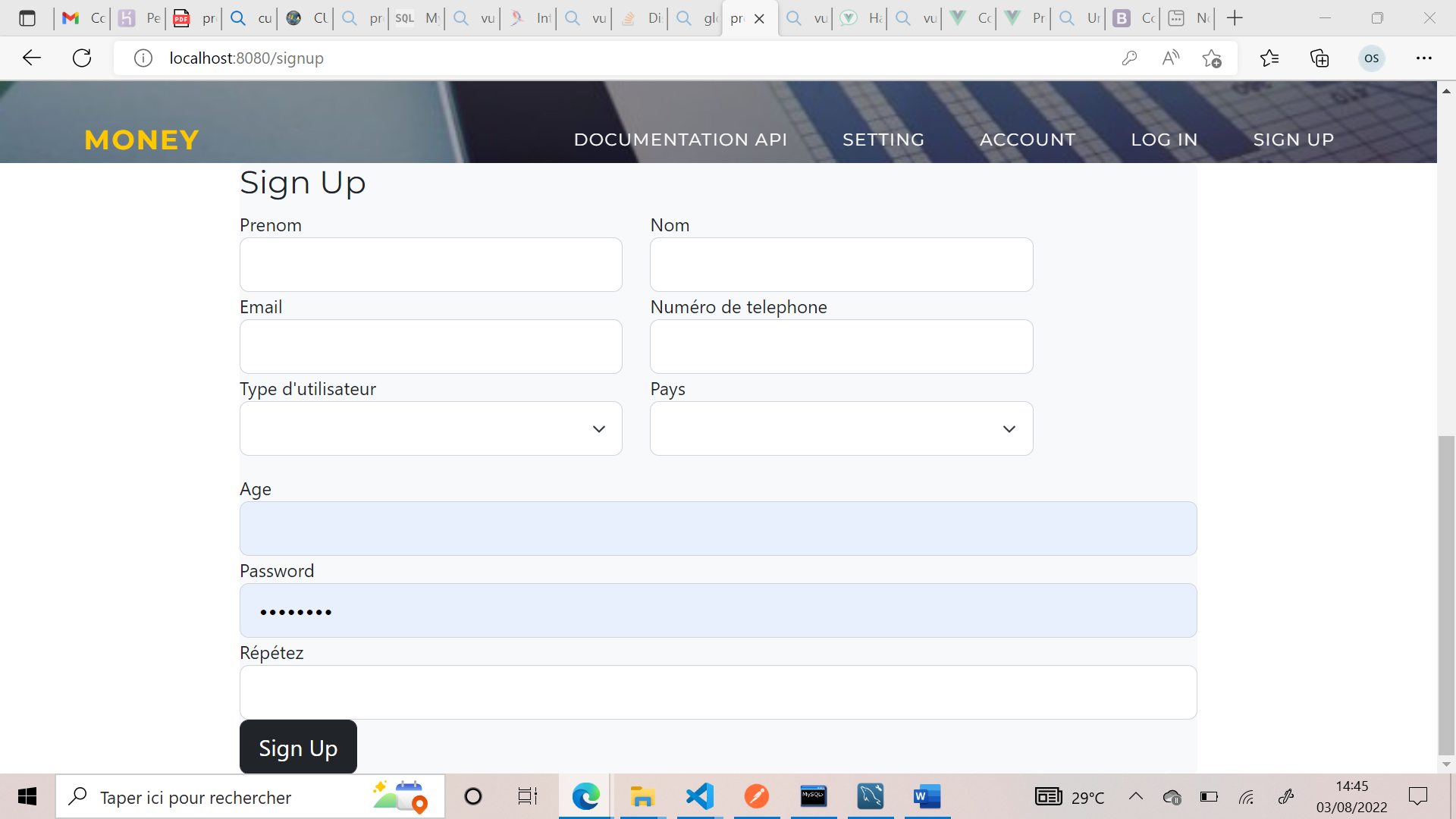
1. Inscription



Capture : page de connexion

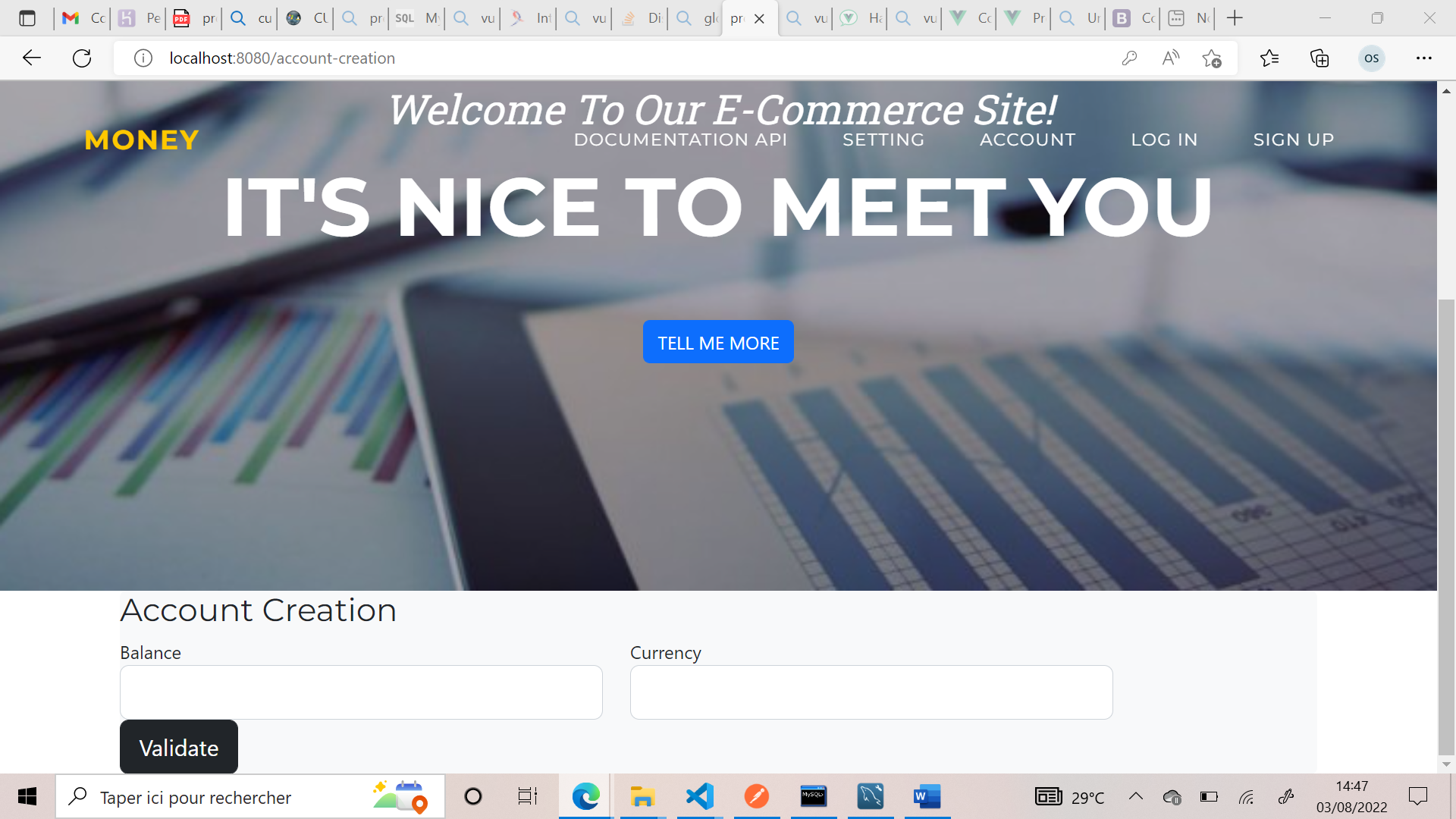
Dans la page de connexion l’utilisateur est obligé de renseigner son email et mot de passe, puis le système vérifie que les informations sont correctes, si l’utilisateur est identifié il est redirigé vers la liste des articles, sinon un message d’erreur apparait.

1. Ajout compte



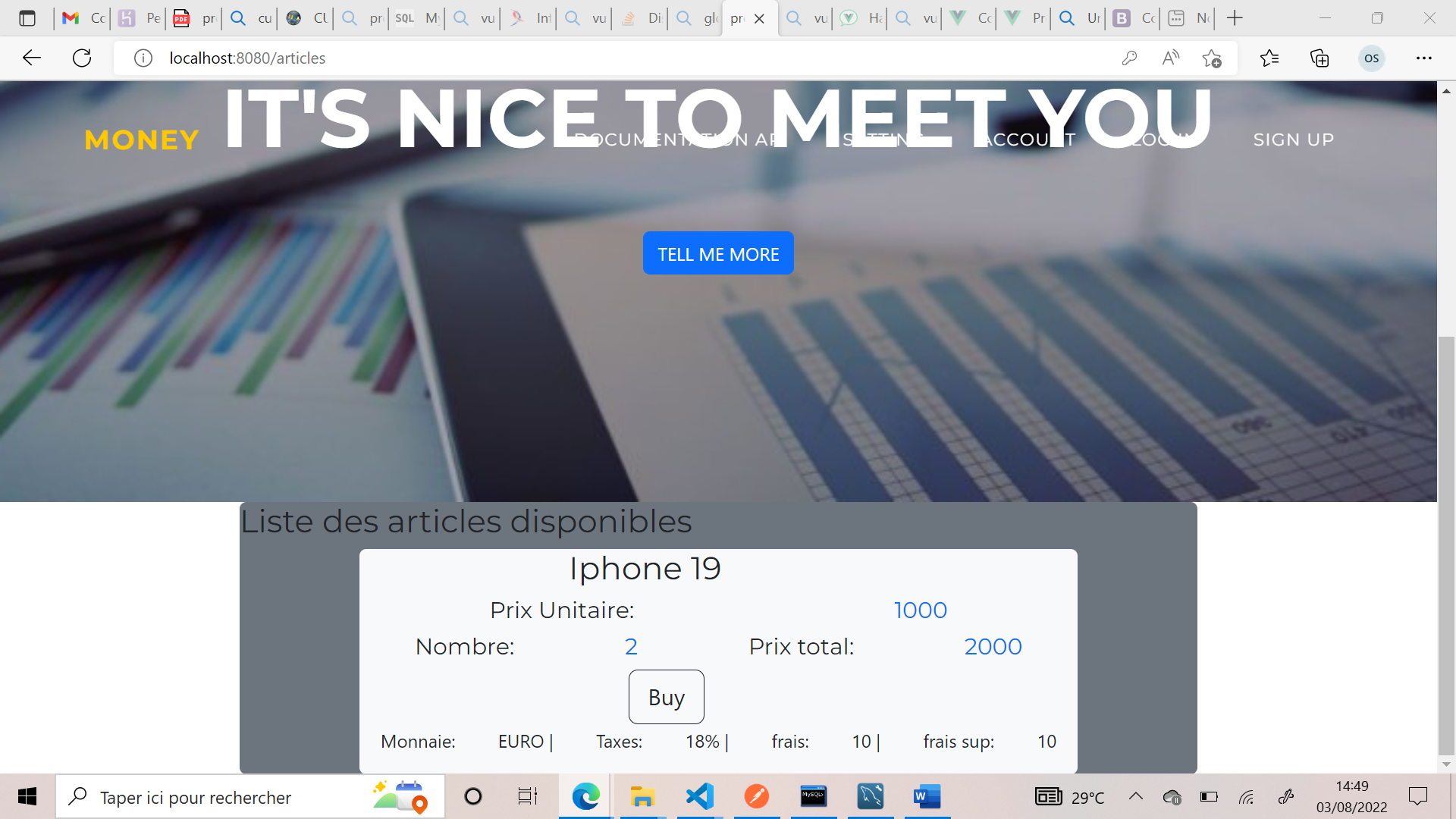
Capture : page d’inscription

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 création du ‘account’



Capture : page de création du ‘account’

1. 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 .