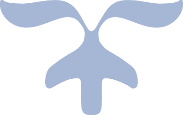


LICENCE 3 INFO MIAGE : PROJET S6

CONCEPTION D’UN SITE DE COMMANDE DE NOURRITURE : FAST - DELIVERY



08 AVRIL 2022

UNIVERSITE PICARDIE JULES VERNE AMIENS

UFR - SCIENCES

Réalisé par :

**DIAKITE Djénéba Kaba**

**KEITA Bougouri**

Enseignants :

**FURST FREDERIC**

**LI YU**

Table des matières

[Introduction 2](#_Toc100528444)

[Analyse & Conception 3](#_Toc100528445)

[Étude des produits existants du point de vue technique : 3](#_Toc100528446)

[Présentation des choix techniques : 3](#_Toc100528447)

[Modélisation des données 3](#_Toc100528448)

[La modélisation de données sur MongoDB 3](#_Toc100528449)

[1.2 Dictionnaire des données 4](#_Toc100528450)

[diagramme de classe 5](#_Toc100528451)

[Choix techniques et justification de votre choix : 5](#_Toc100528452)

[Structures des données et algorithmes : 5](#_Toc100528453)

[Codage 6](#_Toc100528454)

[Bilan & Conclusion : 11](#_Toc100528455)

[Annexes : 12](#_Toc100528456)

[Reference : 18](#_Toc100528457)

# 

# Introduction

Dans le cadre de notre formation à l’Université Picardie Jules Verne Amiens, nous sommes amenés à réaliser un projet en groupe de deux personnes pour le module projet UE afin de mettre en pratique les connaissances théoriques acquises lors du S5 (MACSI, MO) et en S6.

La réalisation s’est déroulée dans les meilleures conditions et a été une occasion pour complémenter la phase théorique.

Notre projet est de réaliser un site web pour la commande de nourriture qui va permettre au client d’organiser en amont et en ligne ses commandes, limiter ses déplacements et maximiser son temps à la maison ou dans son bureau. Le restaurateur quant à lui pourra organiser au mieux le flux de personnes devant et dans son établissement et limiter le risque sanitaire.

Ce rapport décrit les différentes étapes par lesquelles nous sommes passés afin de réaliser le travail qui nous a été confié en commençant par spécifier les différents changements que nous avons effectué au niveau de notre cahier de charge. Nous ferons une analyse de l’existant du point de vue technique, ensuite de la conception. Après le test et le bilan nous allons conclure.

# Analyse & Conception

## Étude des produits existants du point de vue technique :

Avec nos recherches nous avons trouvé plusieurs sites de commandes en ligne qui correspondent à nos attentes. Ces sites englobent toutes les fonctionnalités de notre futur site et peuvent être une version améliorée. Comme énumérer dans notre cahier de charges fonctionnelles, nous allons présenter les choix techniques de deux sites : Uber Eat (solution 1) et Deliveroo (solution 2)

## Présentation des choix techniques :

Solution1 : Uber Eats

Uber Eats est le service de livraison de plats de restaurants de la société Uber. Il assure le service de commande d’un plat auprès d’un restaurant et la livraison de celui-ci par un livreur indépendant à travers un site web ou une application mobile.

Uber Eats à adopter une architecture basée sur le micro service orienté domaine.

Il utilise une base dite : "Schemaless" pour le stockage de données de longue durée. Cette infrastructure vient remplacer progressivement les instances [MySQL](https://www.journaldunet.fr/web-tech/dictionnaire-du-webmastering/1203595-mysql-my-structured-query-language-definition/) et PostgreSQL

*Langage de programmation : d’Uber écrivent principalement avec Python, Node.js, Go et Java. La plupart des services exécutés chez Uber aujourd’hui utilisent encore ces 2 langages.*

Quelles sont les forces et faiblesses de l'existant ? Quelles en sont les limites ?

Pour le stockage niveau transaction il utilise Ledger Store Docstore, NoSQL (MONGODB)

Pour le mode de paiement utilisé : PayPal, cash, carte de bancaire

Solution1 : Deliveroo

Nous n’avons pas trouvé plus de détails concernant ce dernier mais il a un site web et mobile. Ces informations est celui de leur site d’emploi

Les bases de données : PostGreSQL, MySQL

# Modélisation des données

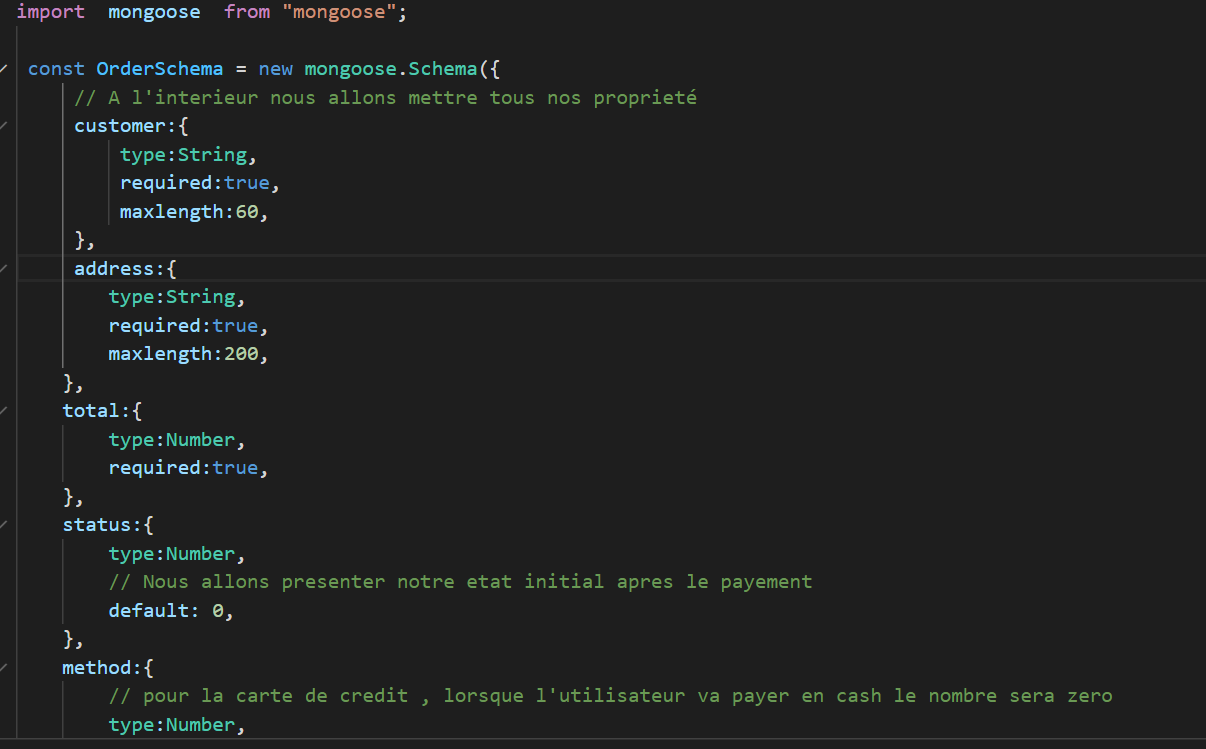
Cette rubrique est consacrée aux étapes fondamentales pour le développement de notre système de gestion de commande en ligne. Nous avons choisi de modéliser avec un NoSQL.

## La modélisation de données sur MongoDB

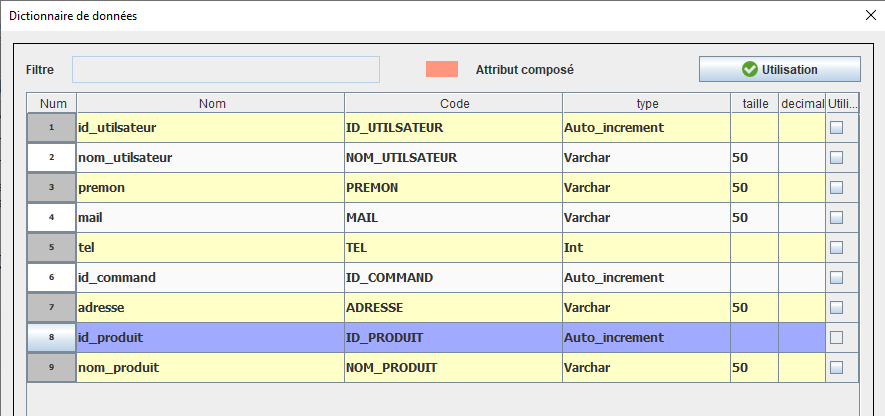
Contrairement aux bases de données SQL, MongoDB n’implique **aucune contrainte en termes de structure**des documents. Les données n’ont pas de schéma préconçu et c’est cette flexibilité qui rend MongoDB si puissant et performant.

La **modélisation des données** et la structure des documents doivent uniquement répondre aux besoins de l’utilisateur. Il convient de prendre en compte les besoins de l’application, et donc quelles seront les données et les types de données nécessaires.

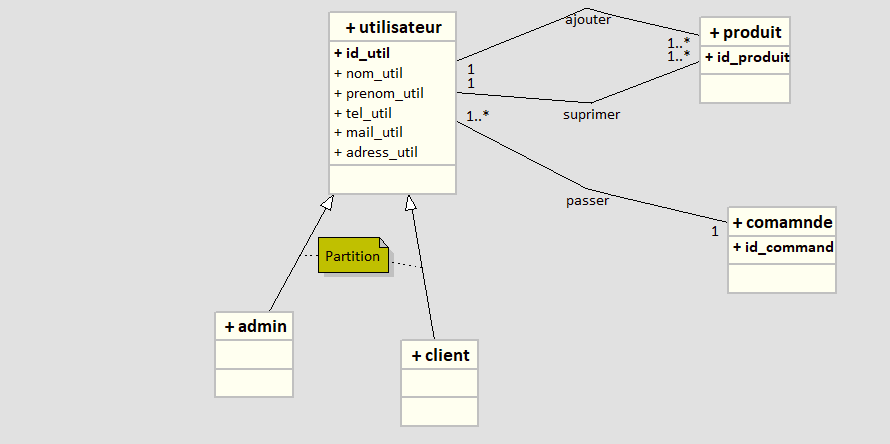
Si de nombreuses requêtes sont à prévoir, il est pertinent d’utiliser **des index** dans le modèle de données pour améliorer l’efficacité des requêtes. Enfin, en cas d’ajouts, de mises à jour et de suppressions de données fréquentes, on utilisera là encore des index et le système de sharding pour améliorer l’efficacité générale de l’environnement.



## 1.2 Dictionnaire des données



## diagramme de classe



Choix techniques et justification de votre choix :

Les choix techniques qui sont judicieux pour notre projet de mise en place d’un site web de commande en ligne.

**Pour le langage de programmation** nous avons opté pour Next js un Framework de développement frontend le plus populaire de react. Next js offre une solution simple et tout en un.

Backend : Next js se base sur [le file system node](https://practicalprogramming.fr/boite-a-outils-node-js/how-to-use-node-fs/) pour définir les routes frontend et backend de notre notre application.

Pour un projet web de très petite envergure tel que celui-ci, nous allons opter pour **Mongo DB Atlas**, une option de service MongoDB Enterprise hébergée dans le cloud qui ne nécessite aucune installation supplémentaire et offre un niveau gratuit pour commencer. On peut également mettre à jour ou appliquer notre schéma (schéma JSON) à tout moment. MongoDB va permettre de stocker les documents dans [des collections](https://www.mongodb.com/docs/manual/core/databases-and-collections/#std-label-collections) . Les collections sont analogues aux tables dans les bases de données relationnelles.

**Des API** pour faire des enregistrements dans notre Base de données tel que Postman, et PayPal développer pour traiter les encaissements pour notre compte et de faire en sorte que chaque règlement arrive directement dans le compte PayPal de notre boutique (pour nos achats tout en créant un compte personnel et business sur le site sandbox PayPal développer).

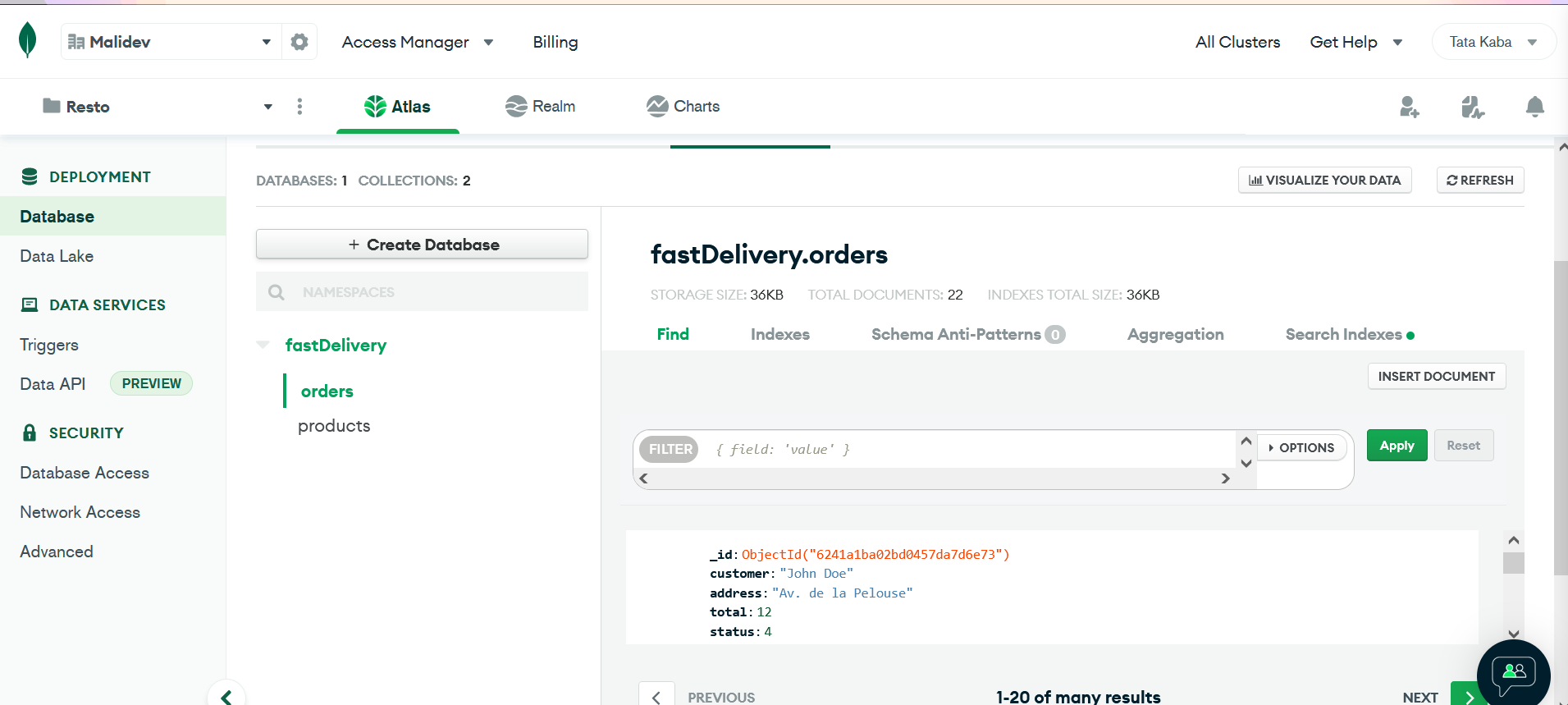
**Nous allons utiliser cloudinary** : qui fournit des services de gestion d'images basés sur le cloud (à chaque ajout de nouveau produit) qui serait un plus, par rapport à notre existant.

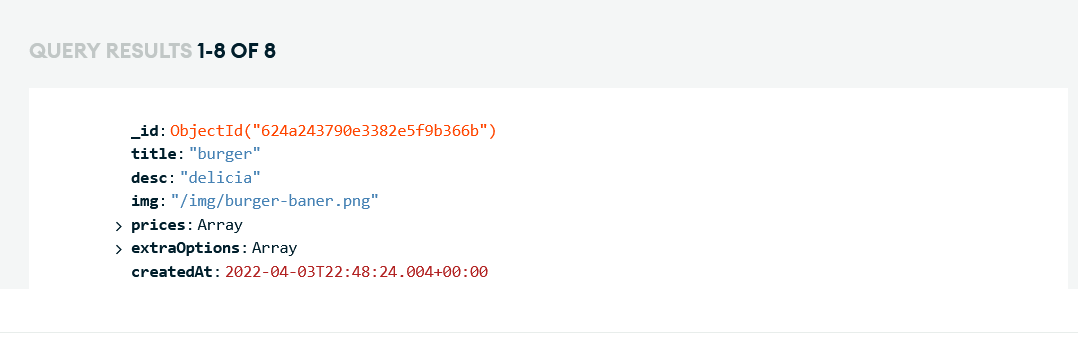
Et le déploiement se ferait sur Vercel.

## Structures des données et algorithmes :

Contrairement à une base de données [relationnelle SQL](https://datascientest.com/sql-tout-savoir) traditionnelle, MongoDB ne repose pas sur des tableaux et des colonnes. Nos données sont stockées sous forme de collections et de documents (objets JSON ) qui est un format en texte clair pour exprimer les données structurées. Et nous allons ajouter Mongoose une librairie va nous permettre d'imposer une structure à vos données tout en simplifiant les requêtes.

Des images des données ci dessous

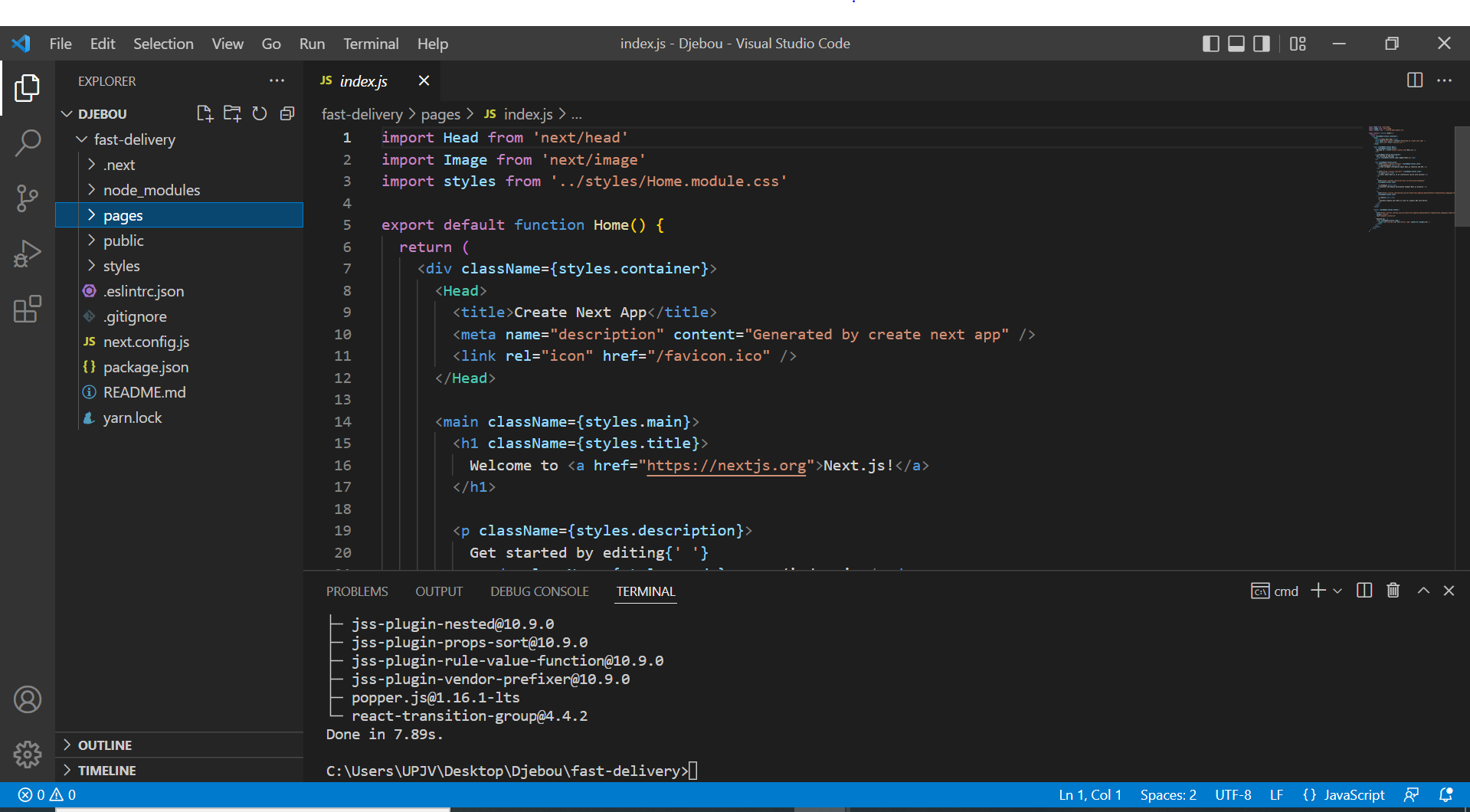




# Codage

Next JS est un Framework JavaScript qui permet de créer des sites web statiques, performants, rapides et conviviaux, mais également des applications web à l’aide de la bibliothèque React.

Dans la structuration de notre code nous avons :



**yarn.lock** - par défaut, create-next-app utilise le gestionnaire de paquets yarn, donc le fichier yarn.lock contient des informations sur toutes les dépendances tierces installées

**next.config.js** - est principalement utilisé pour les paramètres Next.

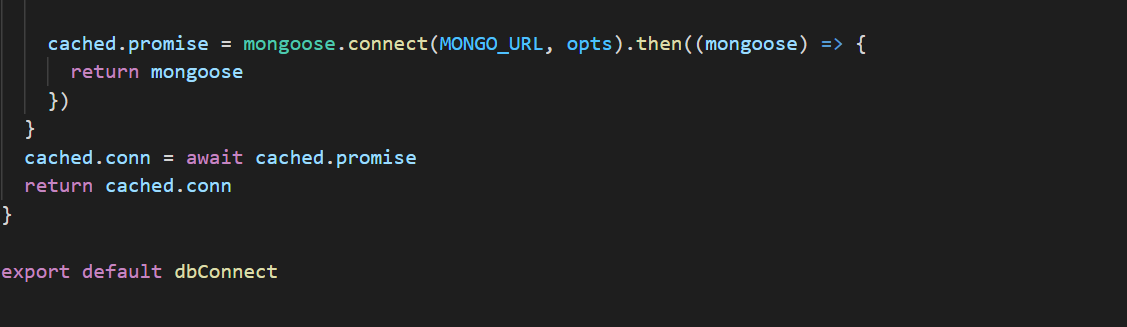
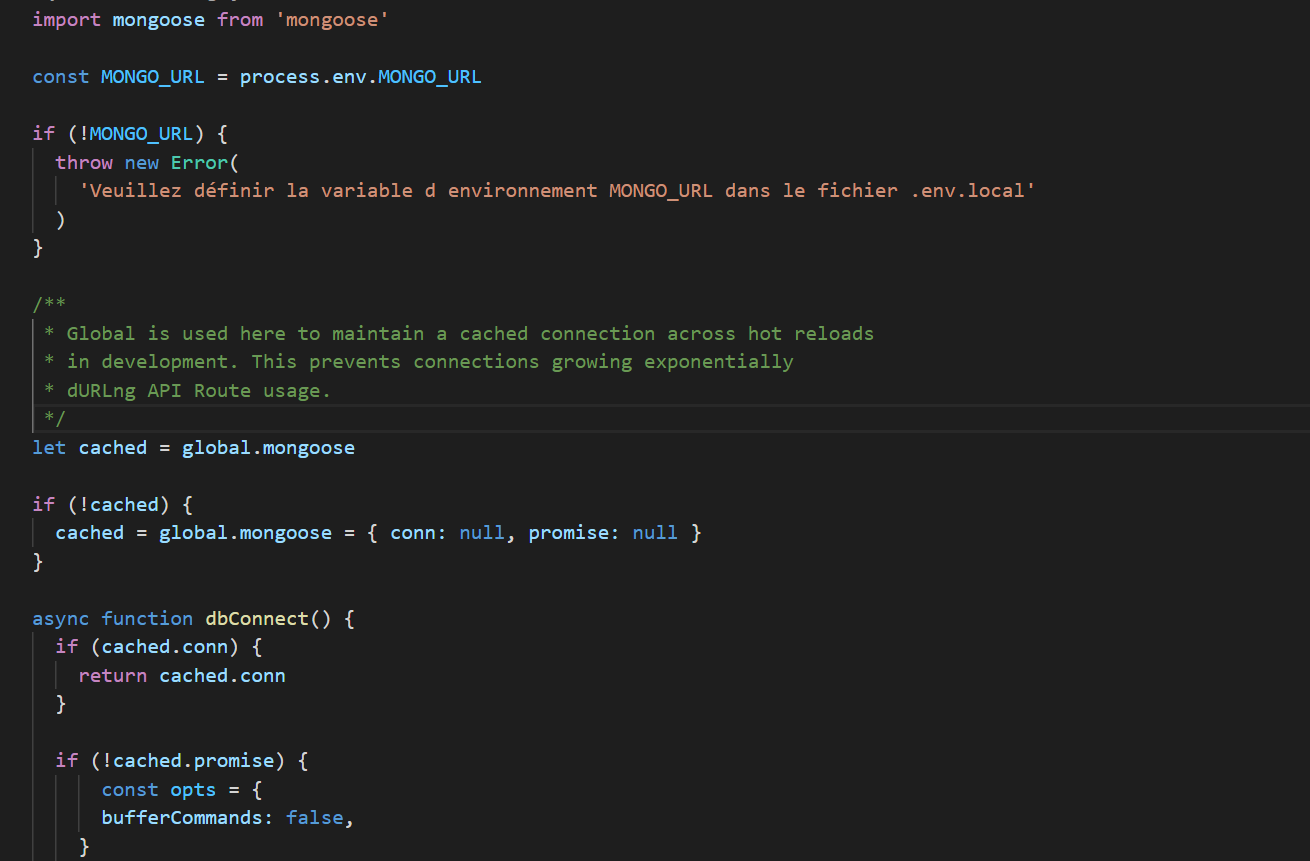
**styles/** - pour les styles de notre application (côté design)

**public/** - pour le contenu statique qui sera publiquement disponible sur https://votre-domaine/, par exemple un logo et nos images

**Pages/** - itinéraires de nos pages tel que nos produits , nos commandes , et nos utilisateurs (admin) présents dans notre base de données.

**Pages/api** - routes API elle mettra en relation nos  **interfaces** et leur permettra d’échanger des données nous avons des verbes http pour gérer les actions que nous voulons effectuer(get , post, put , delete etc…), envoi et réponse pour une api .Et un dossier component pour énumérer nos différentes interfaces ( navbar, contenu, pied de page)

**Description de quelques bouts de code :**

* + - 1. **Connexion à la base de données en utilisant mongoDB**

Après la création d’un fichier mongo.js, nous avons ajouté notre chaîne de connexion dans le code de notre application dans le fichier .env(contenant , nom user , password user, nom du cluster ainsi que le nom de la base de donnée) pour se connecter à la base de données.



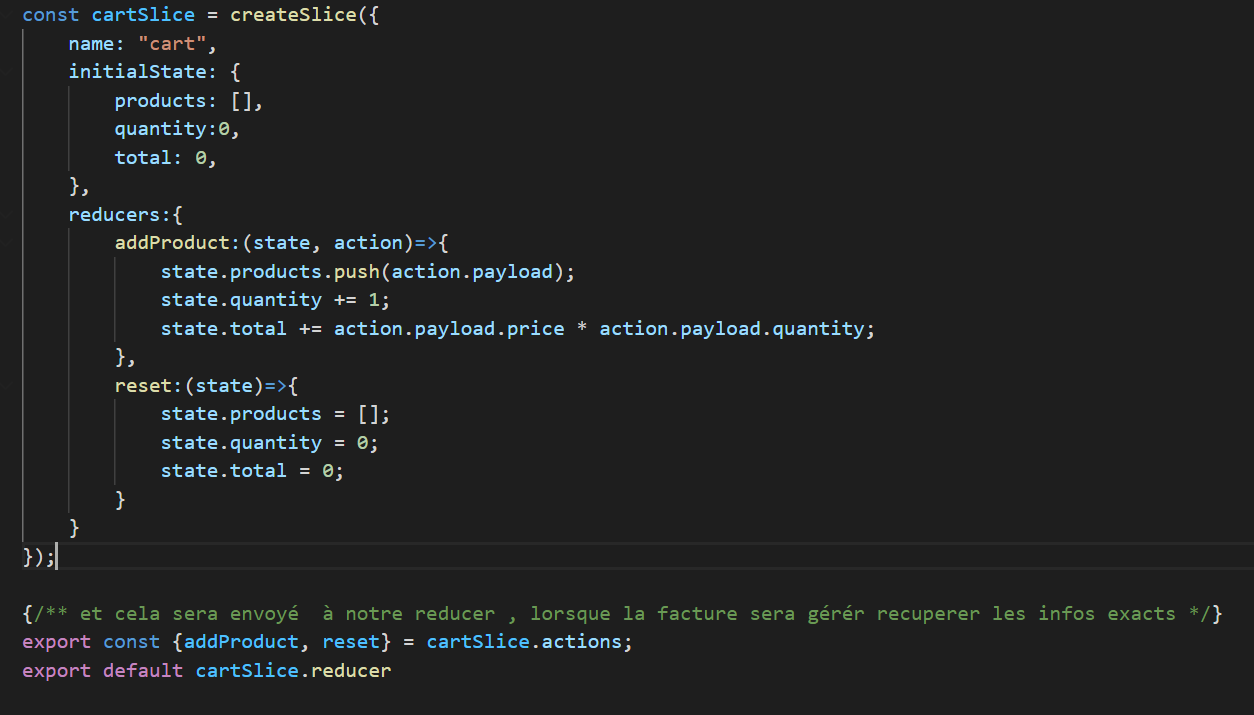
* + - 1. **Gestion de boutique**

**Nous allons configurer** une boutique **grâce à redux toolkit** pour faciliter la récupération ( panier - CartSlice) et de mise en cache des données que nous avons lors de la finalisation de notre commande

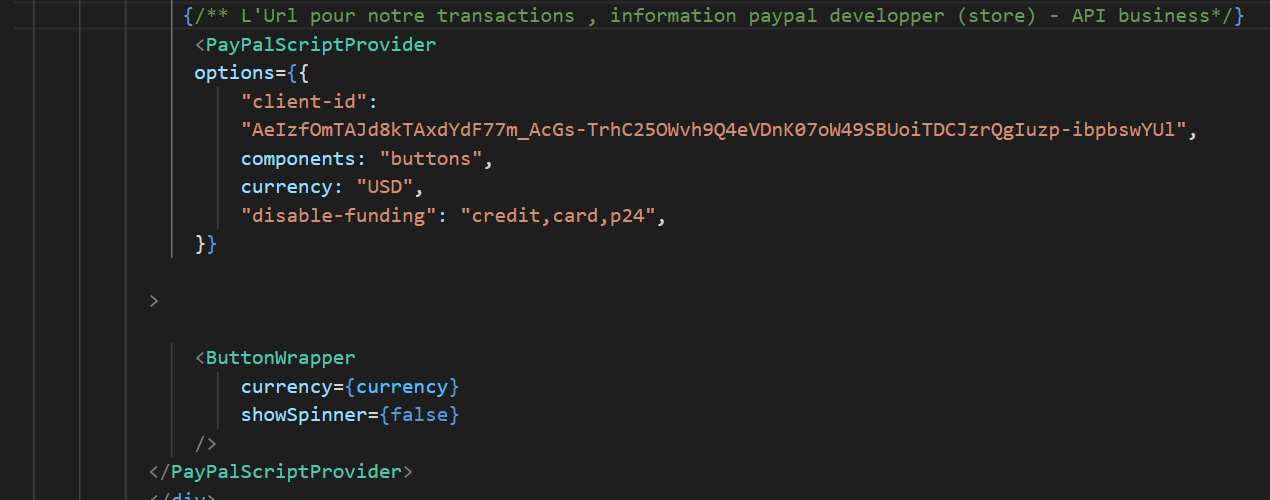
****

* + - 1. **Notre Panier:**

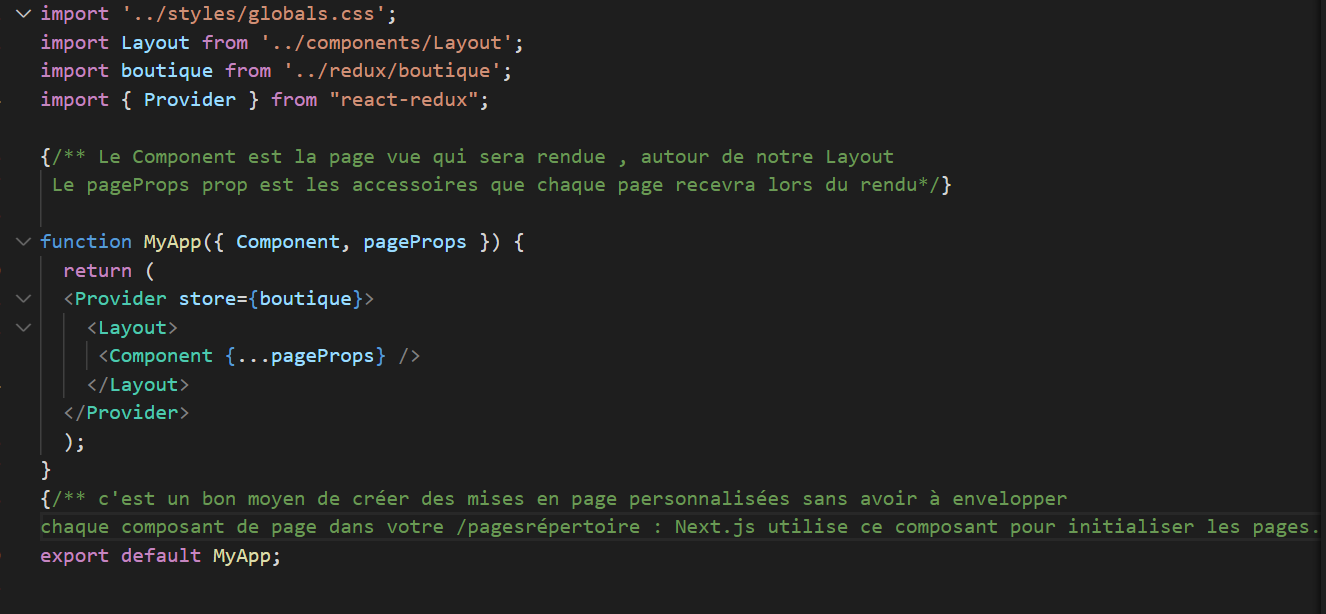
Ce bout de code va nous permettre de gérer notre portefeuille de commande en gérant le prix du produit et les changements que nous allons effectuer avant la finalisation de la commande tels que le choix de la quantité ou soit l’annulation d’un ingrédient etc…



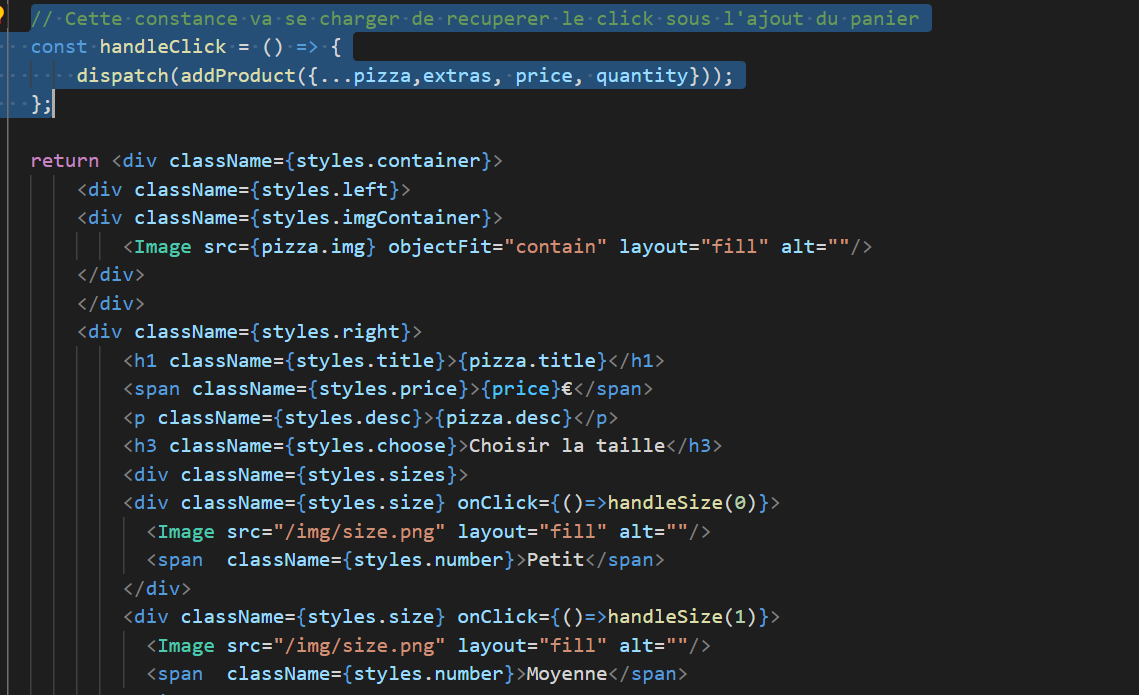
* + - 1. **Un bout de code pour le paiement paypal lié à un api store de paypal**



\_app.js C’est la première page vue qui sera rendue , qui récupère autour de notre Layout les pages selon l’ordre de sélection dans notre component ( voir index.js dans page )

* + - 1. **Ajout d’un produit dans le panier : **

La constance handleClick va nous permettre de récupérer le clic sur le produit et l’envoyer dans la partie sélection du produit avec ses différentes variables ( prix , quantité etc…).



# Bilan & Conclusion :

A l'heure du bilan, nous dressons un bilan positif. Dans l’ensemble nous avons pu  réaliser 90% des fonctionnalités. Nous avons fait face à des difficultés pendant la réalisation de ce projet. En premier lieu nous avons eu du mal à respecter le planning, avec le nombre de projets et de cours que nous avons ainsi que des imprévus liés au fonctionnement de nos machines. En second lieu, il a été difficile de faire fonctionner notre base de données, notre API ne fonctionnait pas. Cependant le déroulement du projet a été productif pour nous, nous avons appris beaucoup de technologies et élargi nos connaissances (niveau base de donnée NoSql , niveau langage de programmation web et aussi manipuler des api pour un site très professionnels ).  Chacun de nous a apporté un plus à l’autre et approfondit  ses connaissances en framework JavaScript.

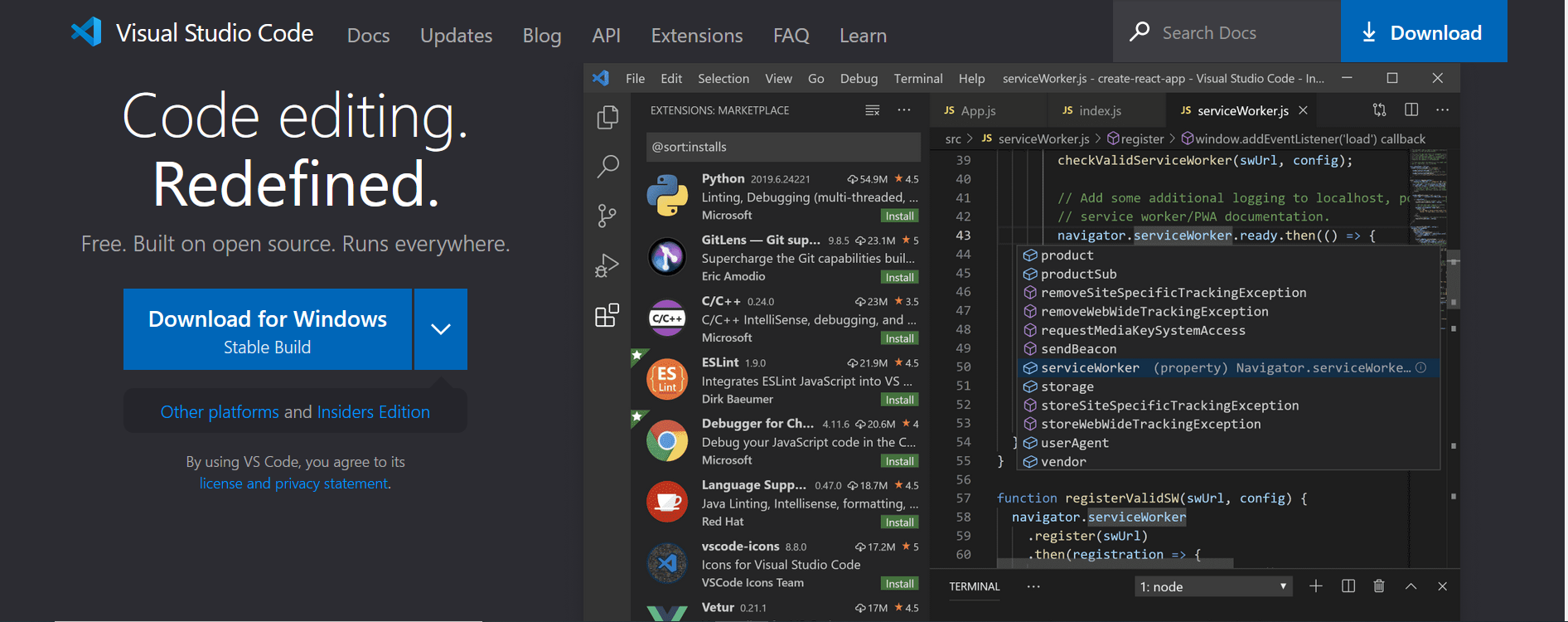
En conclusion, ce  travail d'équipe fructueux et agile nous à permis d'appréhender  l’analyse et la réalisation d’un projet, de faire face aux différentes contraintes requises en même temps nous aider à développer notre sens d’analyse pour l’accomplissement à terme de ce travail à savoir : partir de nos propres idées jusqu’à la conception d’une application.

# Annexes :

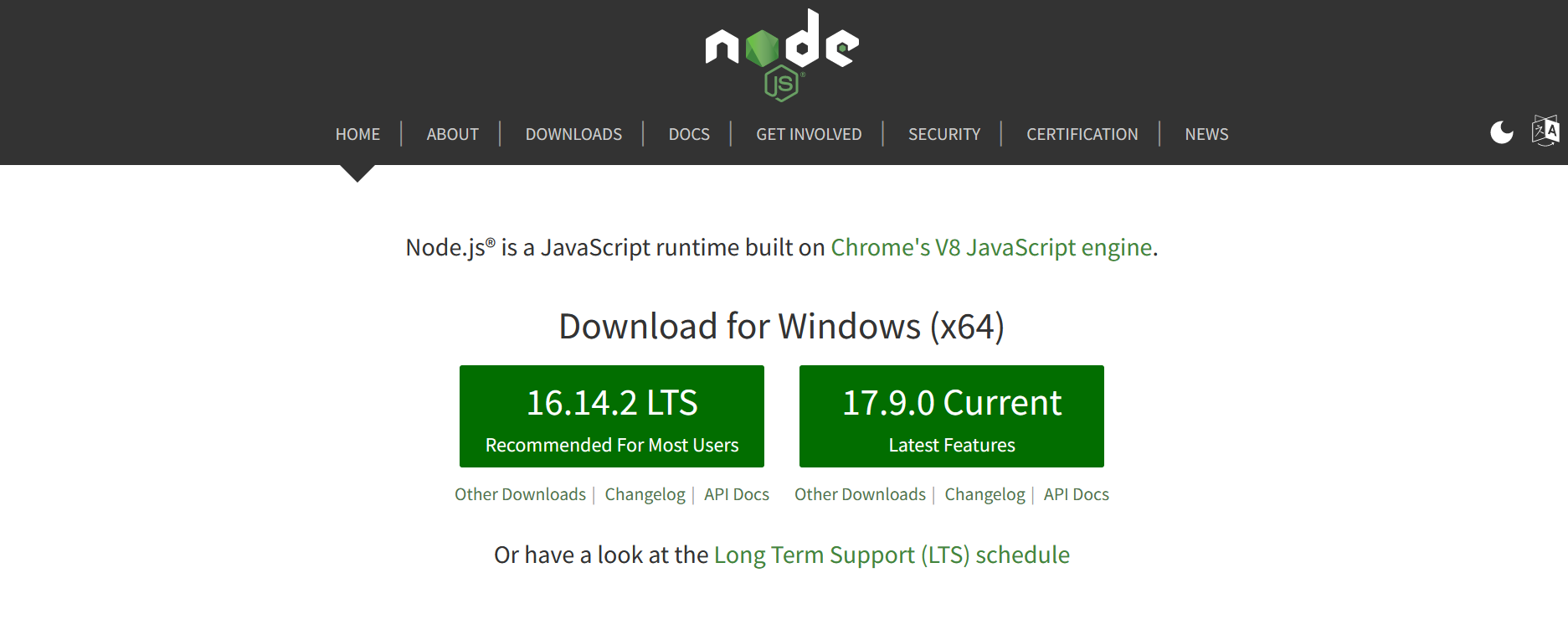
Les documents annexés (guide d'installation du produit, guide de l'utilisateur, etc.) sont-ils pertinents, corrects et complets ?

Guide d’installation :

Éditeur de Texte Visual Studio Code



Node JS : installer



Actuellement nous travaillons avec un NoSQL appelé MongoDB.

Mongodb : Créer un compte, un projet ensuite un cluster (groupe), une base de données , par la suite un utilisateur ainsi qu'une adresse IP. Connecter votre cluster à votre application : et vous verrez votre URL de connexion à copier.

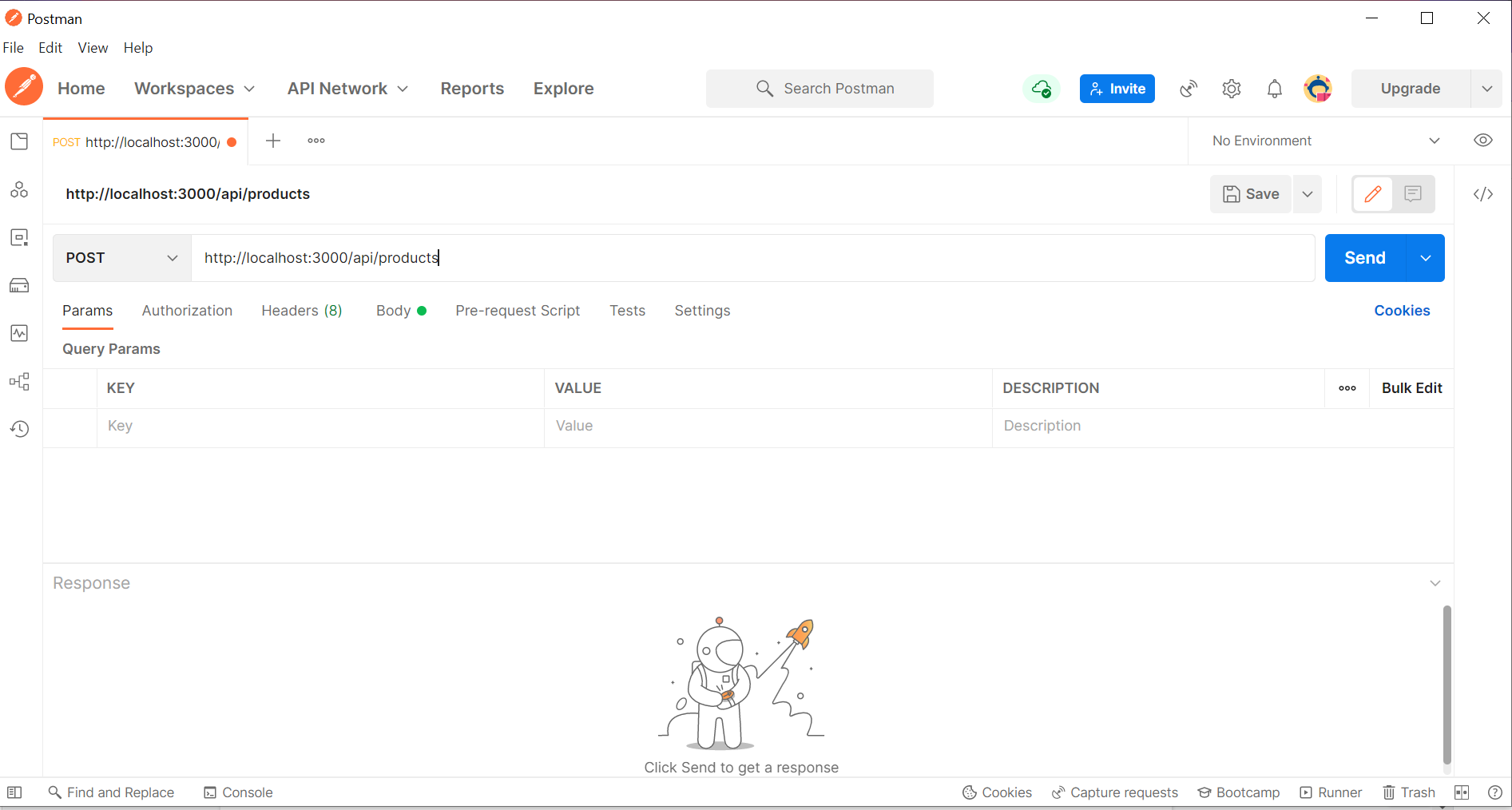
Installer mongodb en utilisant le mongoose package pour connecter à notre bd de mongodb: npx yarn add mongoose



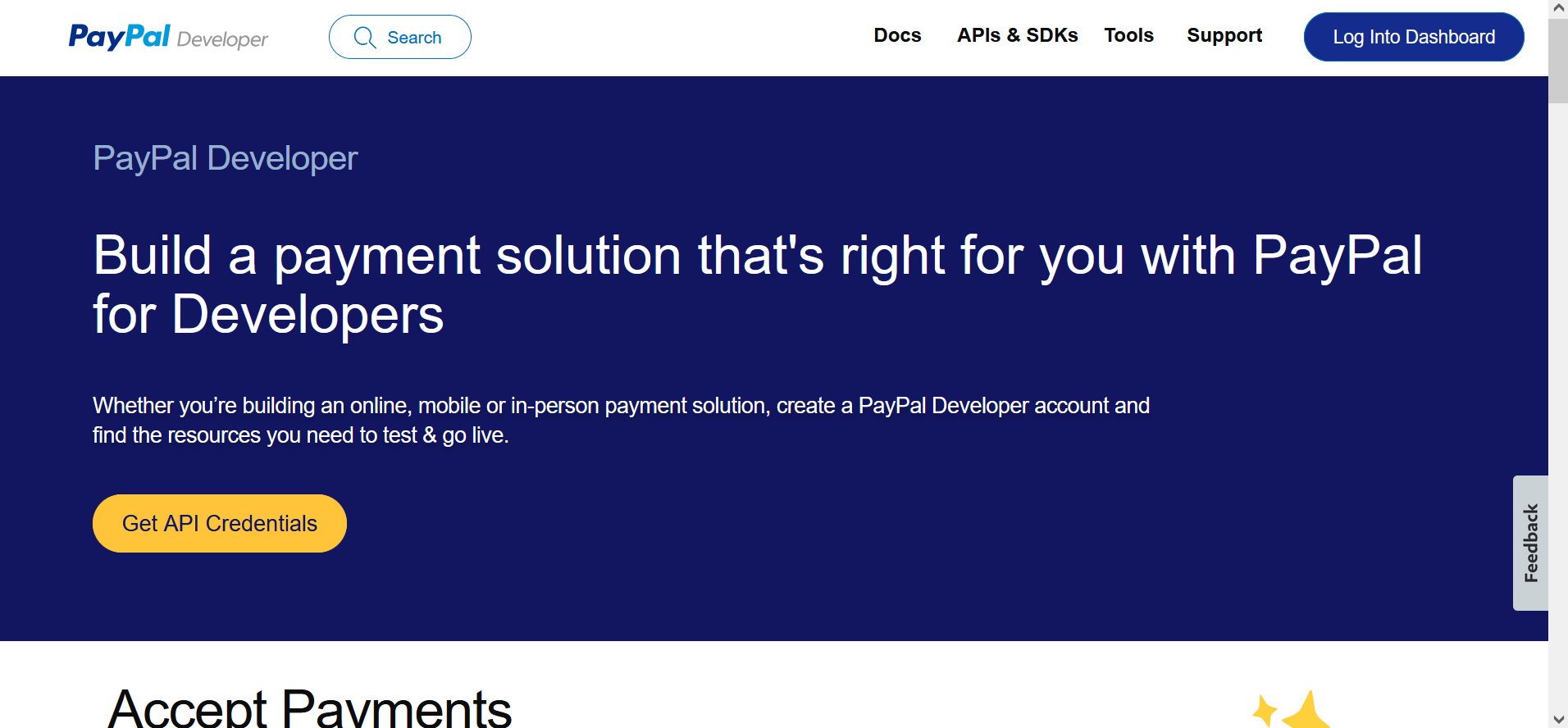
Utilisation des API : Postman et Paypal developer

Postman est un outil permettant de manipuler une API depuis une interface graphique agréable.

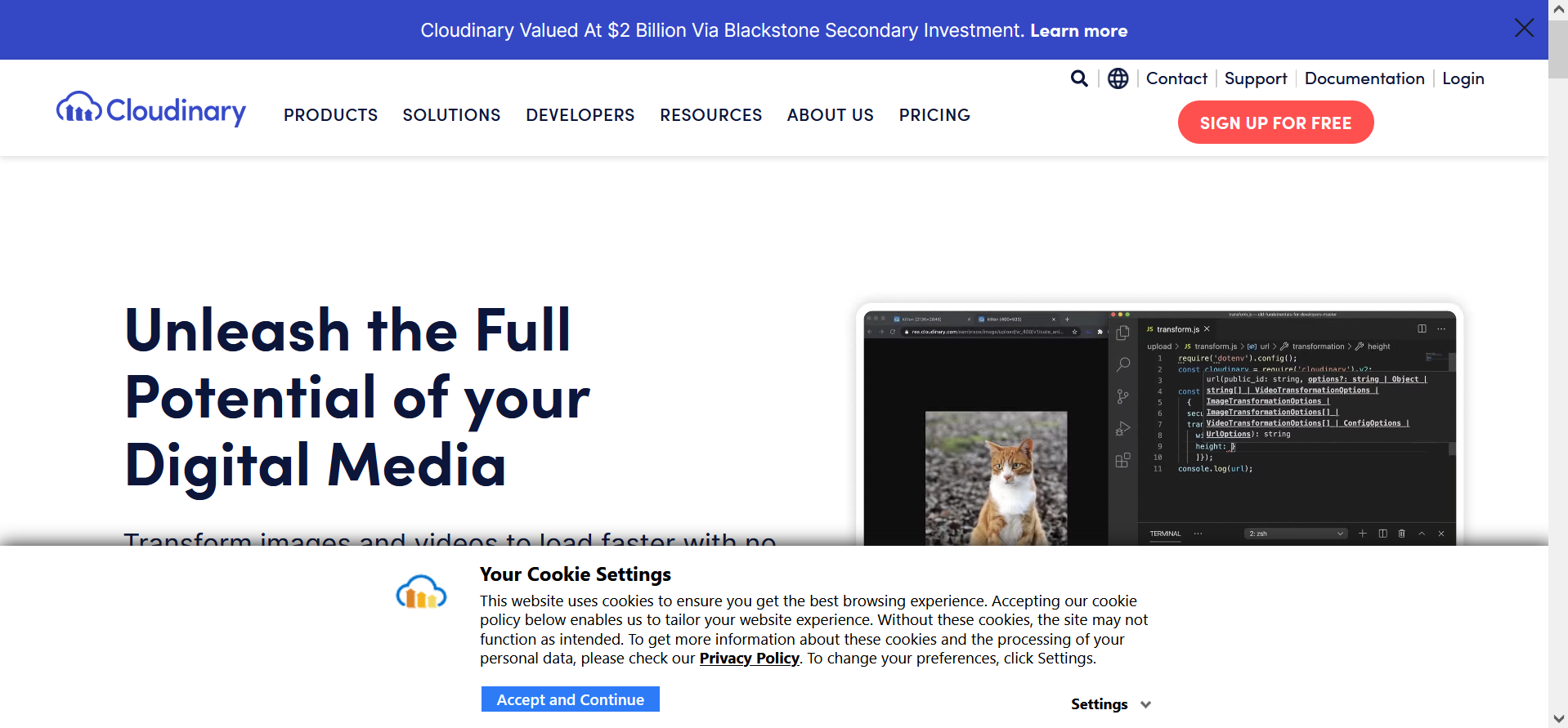
Installer et tester votre base de données



Paypal Developer et le profil sandbox ou nous allons faire des test de paiement



cloudinary : fournit des services de gestion d'images basé sur le cloud



Installation des librairies et création du projet

npx create-next-app : creation de l'application

npm install --global yarn

npx yarn add mongoose

npx yarn add @reduxjs/toolkit

npx yarn add redux react-redux

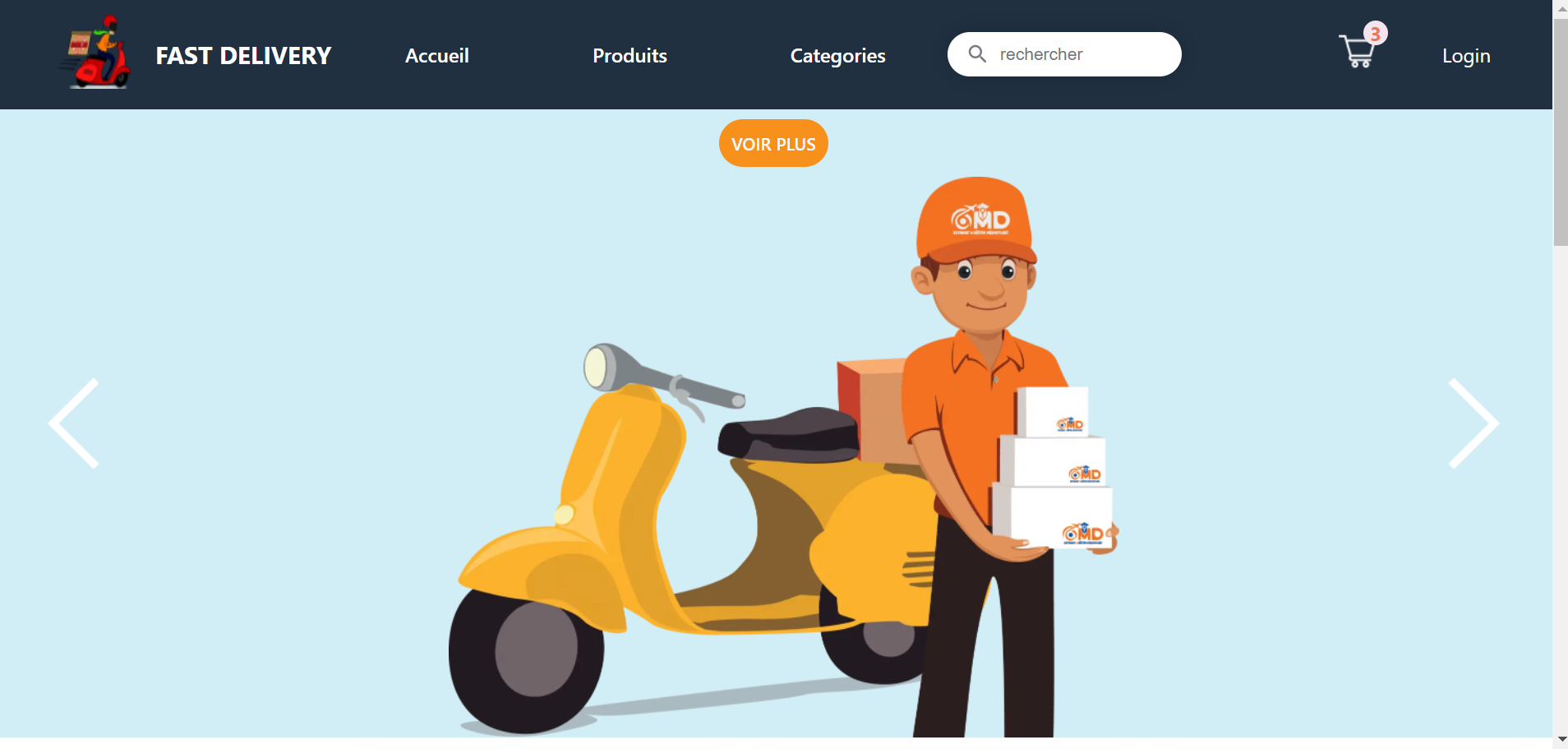
npx yarn add @paypal/react-paypal-js

yarn add @material-ui/core

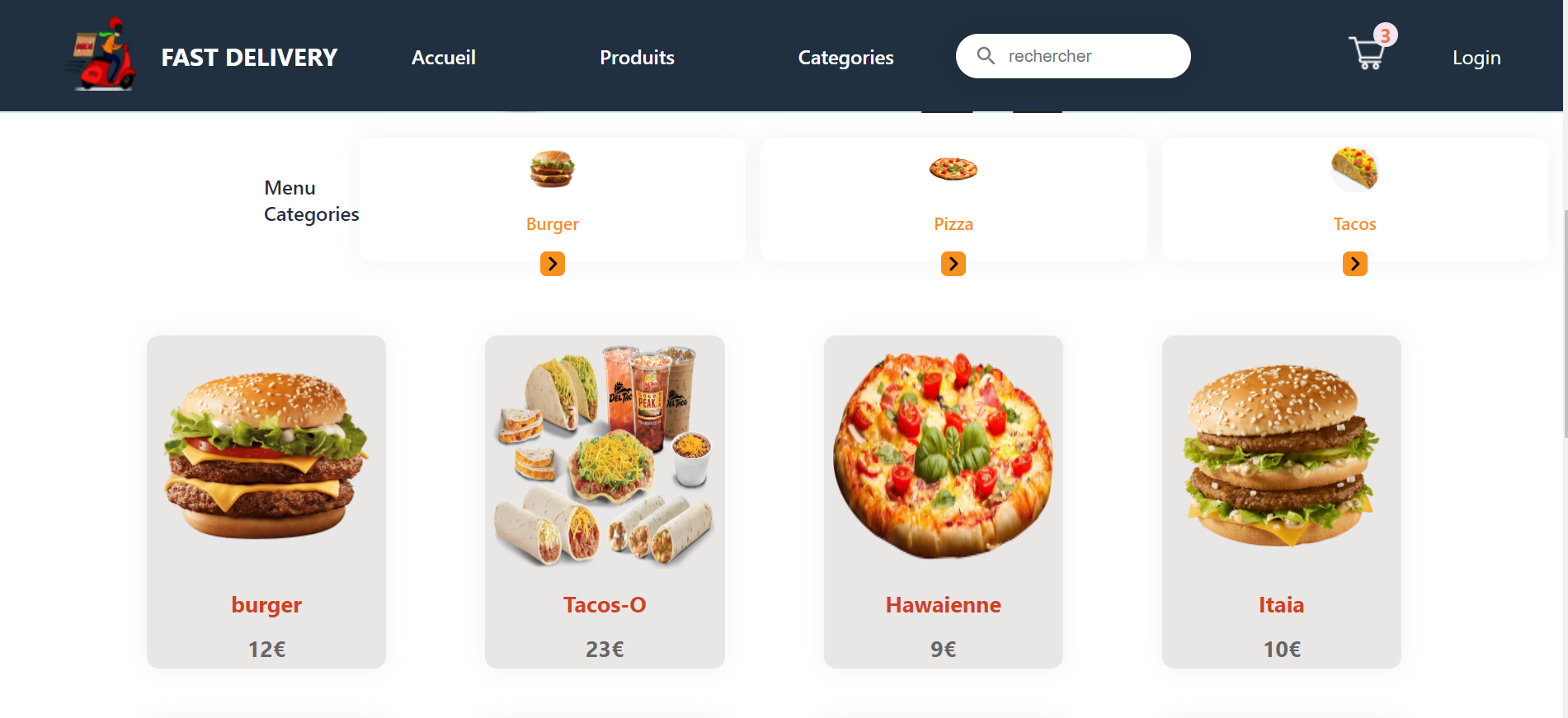
yarn add next-connect

guide de l'utilisateur

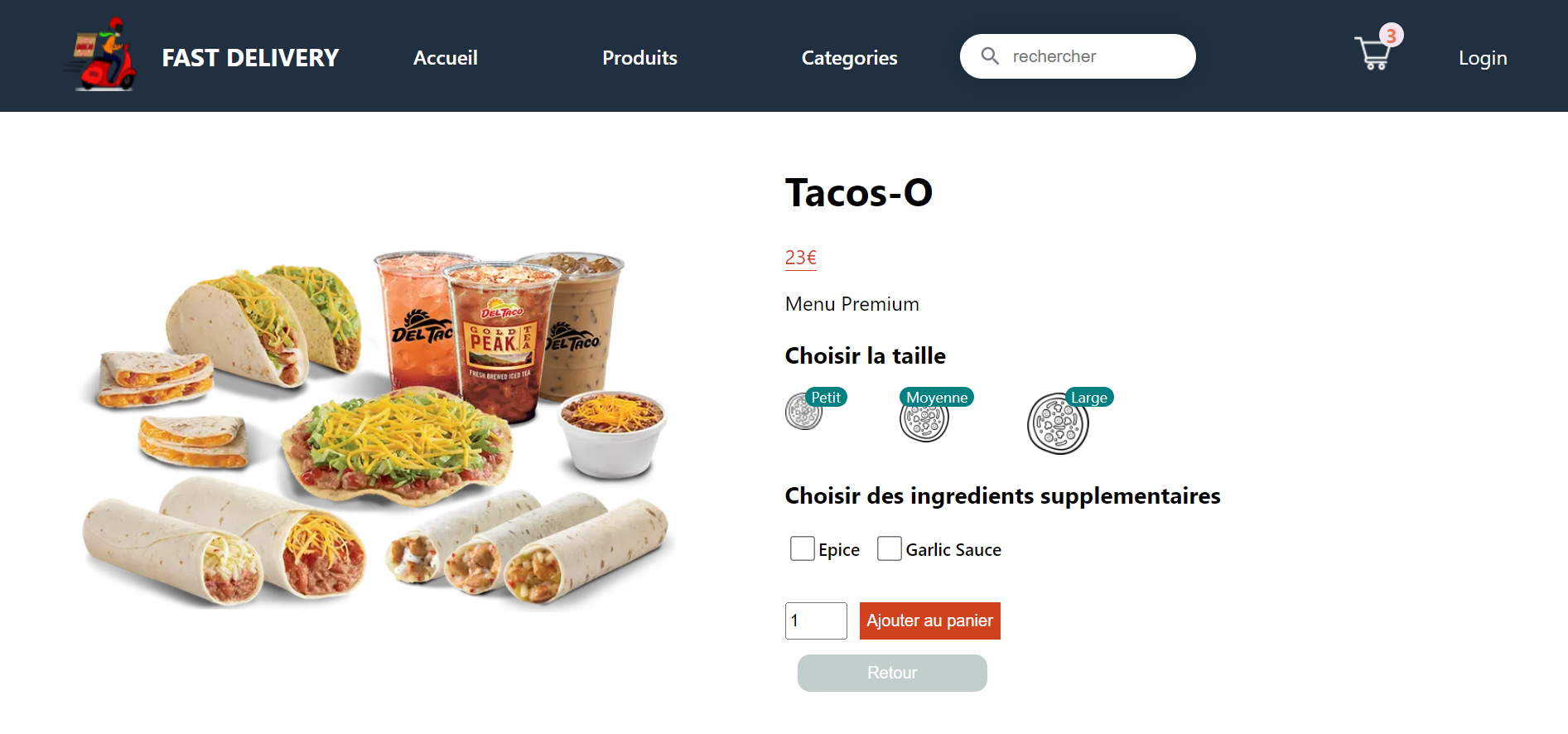
La partie Accueil de l’utilisateur



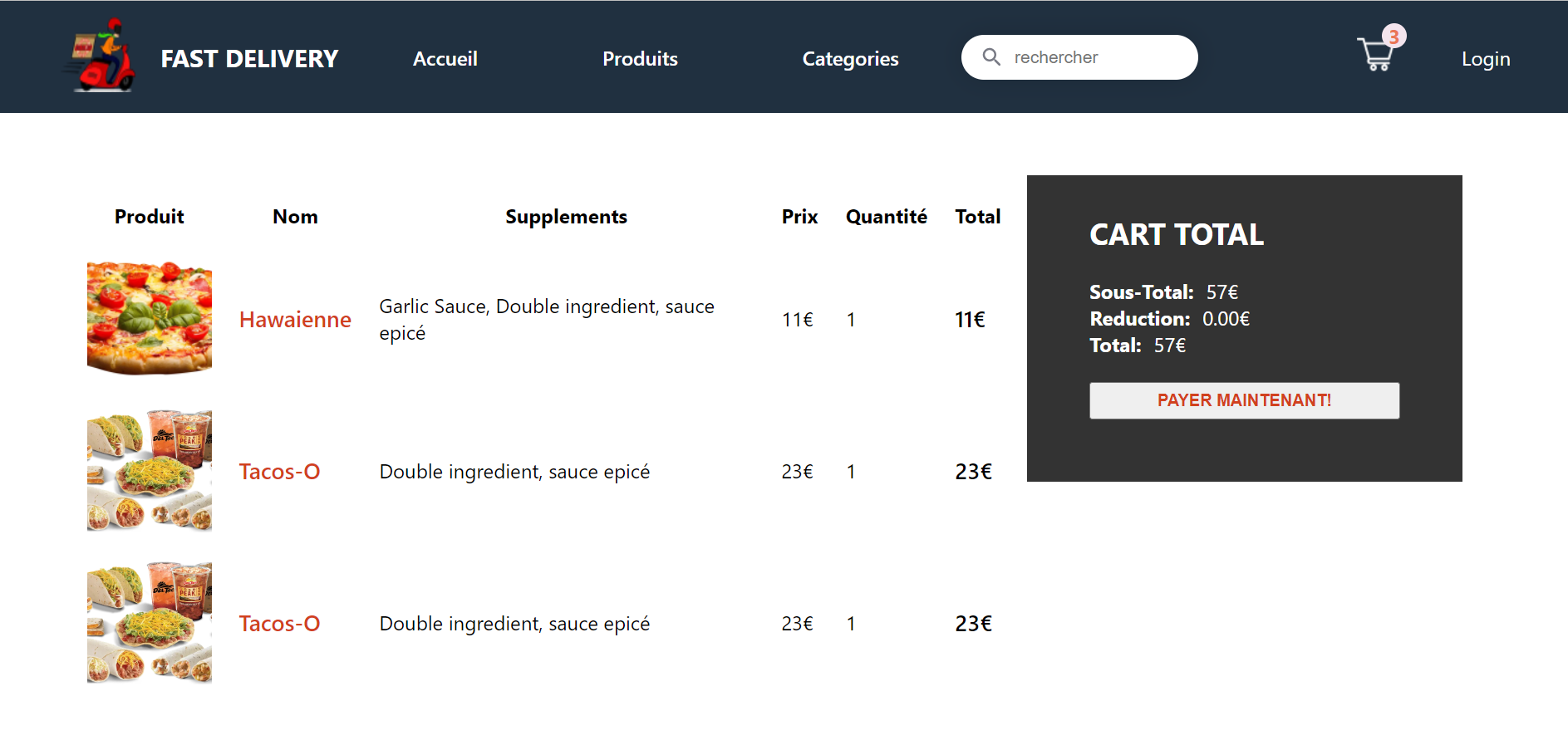
Présentation des produits :



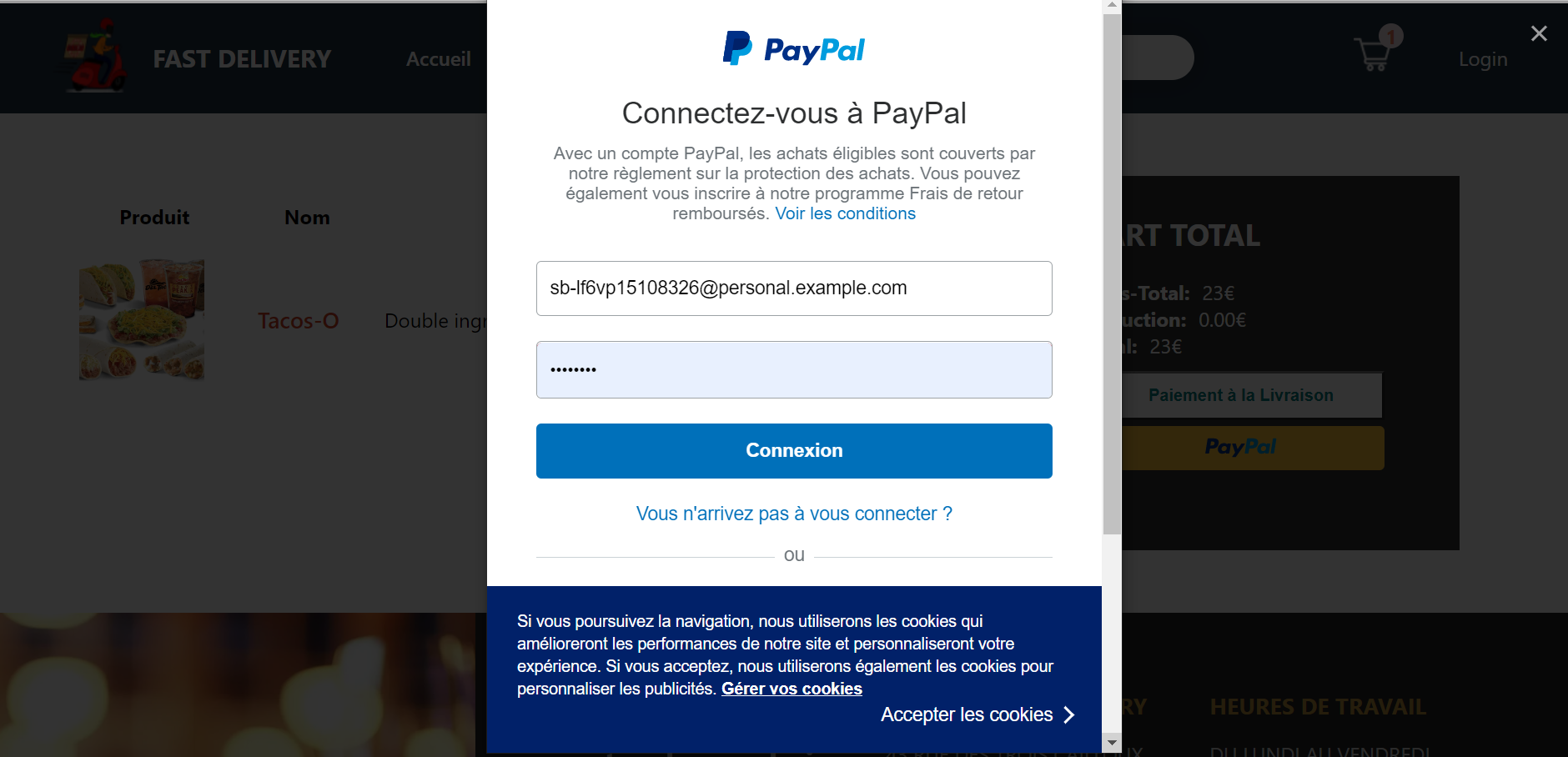
Il faut choisir un Produit qui nous mènera sur :



Après on configure et on ajoute au panier:



Procédure de paiement avec paypal et Finalisation de la commande :



# Reference :

<https://www.journaldunet.com/solutions/dsi/1194648-les-coulisses-techniques-d-uber/>

<https://eng.uber.com/category/articles/architecture/>

<https://eng.uber.com/microservice-architecture/>

<https://www.mongodb.com/docs/manual/introduction/>

<https://practicalprogramming.fr/nodejs-mongodb>

<https://practicalprogramming.fr/pourquoi-next-js>

<https://datascientest.com/mongodb>

<https://www.affdu.com/fr/next-js-tutorial.html>

<https://redux-toolkit.js.org/>