

Kelyan KARAOUNI  
Ibrahim KARAMANLIAN  
B2 DEV

# Mini-Rapport Projet DevOps

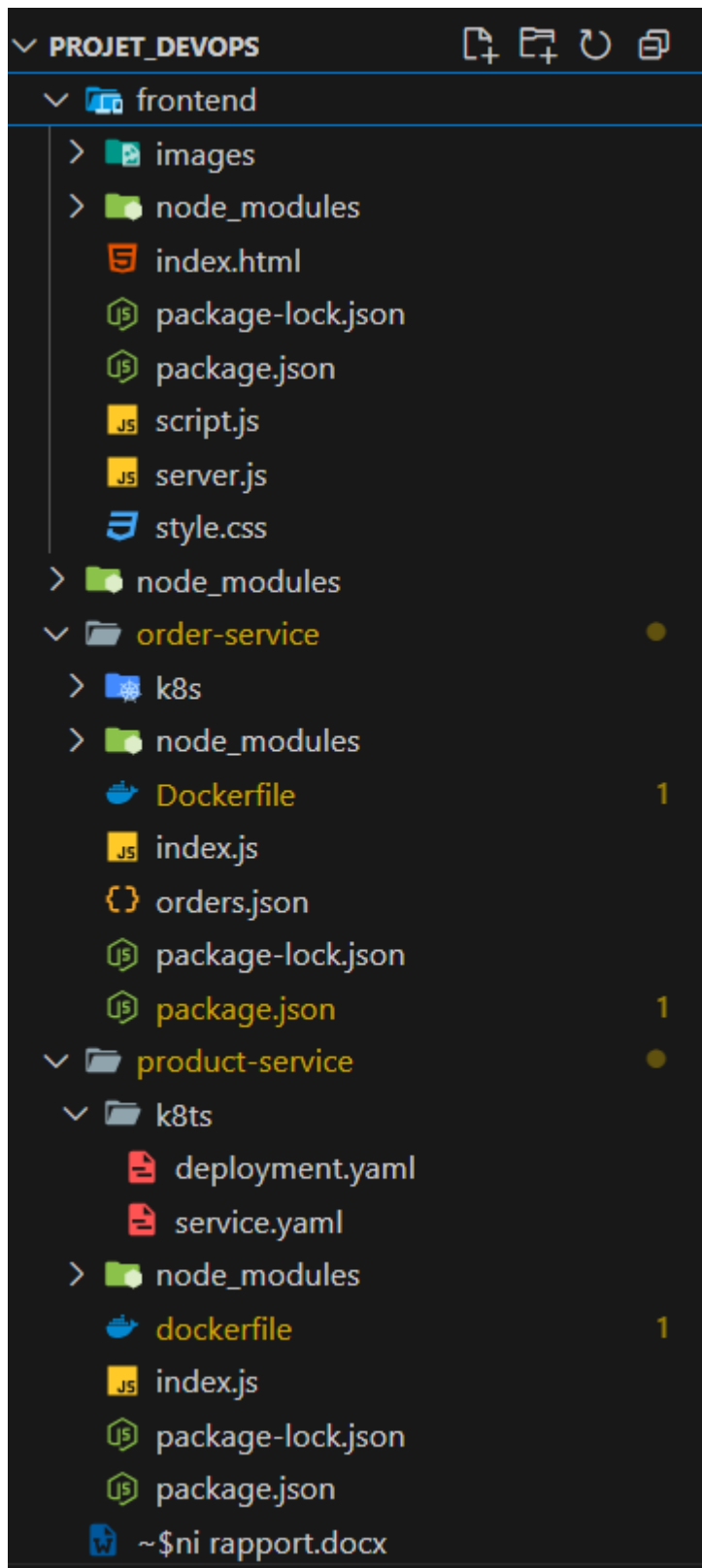
## 1. Présentation du projet

**Nom du projet :** Projet\_Devops

Développement d'un site e-commerce de commandes de produits avec architecture microservices, conteneurisation (Docker), orchestration (Kubernetes), et intégration continue (CI).

**Fonctionnalités réalisées :**

- Page d'accueil avec commandes existantes et onglets pour la navigation.
- Affichage et recherche de produits.
- Création de nouvelles commandes.
- Interface web de démonstration.
- Conteneurisation avec Docker pour chaque service.
- Déploiement via Kubernetes avec des fichiers YAML pour les services et le déploiement.
- CI avec GitLab CI/CD pour l'intégration continue et le déploiement automatisé.



## 2. Architecture technique

### **Microservices :**

- product-service : Node.js – expose une API REST de produits
- order-service : Node.js – gère la création et récupération de commandes
- frontend : HTML/CSS/JS – simple interface qui appelle les deux APIs

## 3. Stack technique

Outil/Technologie	Utilisation
Node.js + Express	Développement backend (2 services)
HTML/CSS/JS	Frontend
Docker	Conteneurisation
Kubernetes	Orchestration
GitLab CI	Intégration continue
URL	Tests manuels des endpoints

## 4.Installation dépendances et démarrage de service

### **Product-service**

(Bash)      `cd product-service`  
  
              `npm install`  
  
              `node index.js`

Cela démarre l'API produits sur <http://localhost:3001/products>

←

→

↻

localhost:3001/products

🔥 Débuter avec Firefox

📁 Acer

🌐 Express VPN

🌐 McAfee Security

🖨️ LastPa

JSON

Données brutes

En-têtes

Enregistrer

Copier

Tout réduire

Tout développer

🔍 Filtrer le JSON

▼ 0:

name:

"Téléphone"

description:

"Téléphone tactile 10 pouces "

imageUrl:

"../images/smartphone.png"

price:

1000.99

▼ 1:

name:

"Ordinateur portable"

description:

"Ordinateur portable 15 pouces"

imageUrl:

"../images/ordi.png"

price:

799.99

▼ 2:

name:

"Casque"

description:

"Casque sans fil Bluetooth"

imageUrl:

"../images/casque.png"

price:

100.99

▼ 3:

name:

"Montre"

description:

"Montre connectée noire"

imageUrl:

"../images/montre.png"

price:

300.99

▼ 4:

name:

"Tablette"

description:

"Tablette tactile dernière génération"

imageUrl:

"../images/tablette.png"

price:

459.99

▼ 5:

name:

"Souris"

description:

"Souris san fil"

imageUrl:

"../images/casque.png"

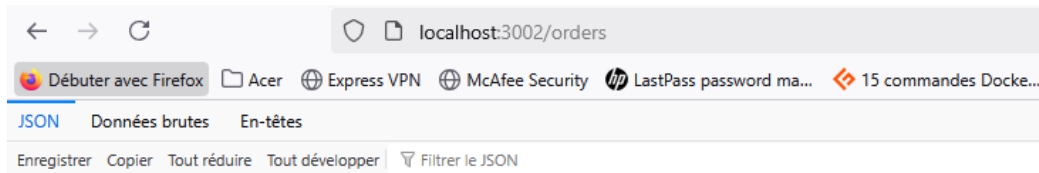
price:

30.9

## Order-service

(Bash) `cd order-service`  
`npm install`  
`node index.js`

Cela démarre l'API commandes sur <http://localhost:3002/orders>



## Frontend

(bash) `cd frontend`  
`npm init -y`  
`npm install express`  
`node server.js`

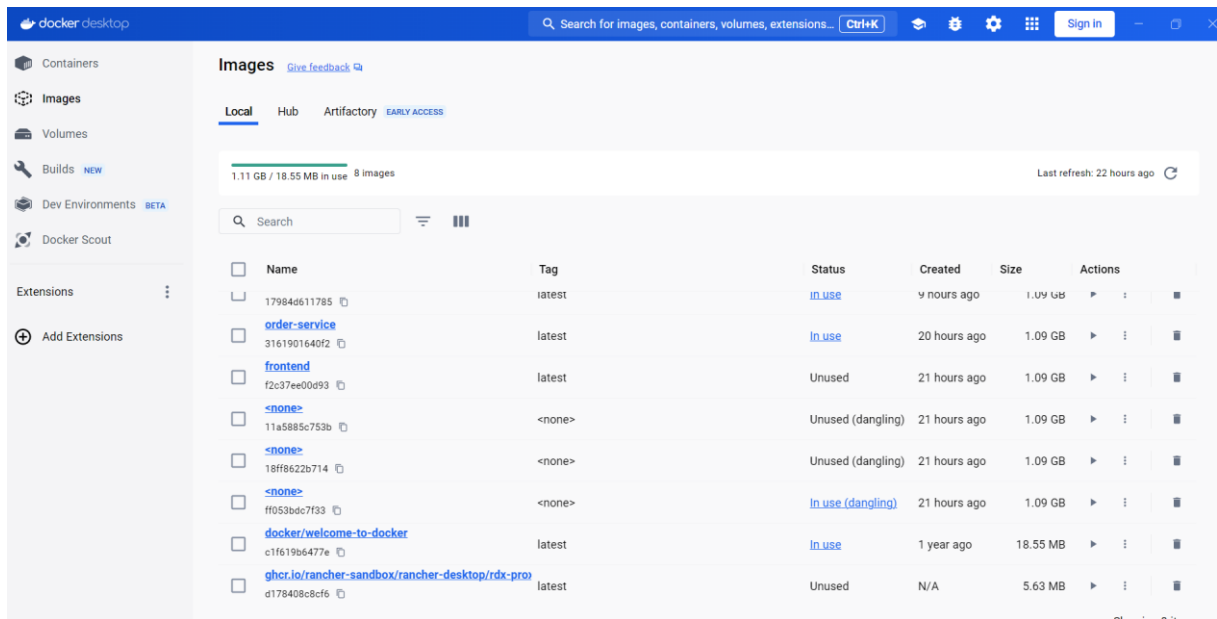
Le site sera accessible sur <http://localhost:8080>

## 4. Docker

Chaque microservice est conteneurisé avec Docker :

- Un **Dockerfile** est créé pour chaque service.
- Chaque service est exposé sur le port approprié :
  - **product-service** : Port 3001
  - **order-service** : Port 3002

## Démarrer **Docker Desktop** dans windows



### **Commandes pour démarrer Docker localement :**

(bash) `cd product-service`

`docker build -t product-service .`

`docker run -p 3001:3001 product-service`

`cd order-service`

`docker build -t order-service .`

`docker run -p 3002:3002 order-service`

## 5. Kubernetes

Chaque service possède :

- un Deployment YAML

Pour visualiser les containers Docker ou pods Kubernetes, utiliser Docker Desktop ou Rancher Desktop.

- **Builder les images Docker**
- Installer Rancher desktop
- Démarrer **Rancher Desktop** dans windows
- Puis dans le terminal, lancer les commandes :

- `kubectl apply -f product-service/k8s/`
- `kubectl apply -f order-service/k8s/`
- `kubectl apply -f frontend/k8s/`

```
nebal@Kelyan MINGW64 /c/Kelyan/projet_DEVOPS
$ kubectl apply -f product-service/k8s/
kubectl apply -f order-service/k8s/
kubectl apply -f frontend/k8s/
deployment.apps/product-service unchanged
service/product-service unchanged
deployment.apps/order-service unchanged
service/order-service unchanged
deployment.apps/frontend unchanged
service/frontend unchanged
```

Vérifier que les pods et services sont créés correctement :

`kubectl get pods`

`kubectl get services`

```
$ kubectl get pods
● kubectl get services
```

NAME	READY	STATUS	RESTARTS	AGE
frontend-69cd78d98d-tz2td	0/1	ErrImageNeverPull	0	19h
order-service-654b9588bf-l2jcs	0/1	ErrImageNeverPull	0	19h
product-service-6dd5b6b4ff-7p2rq	0/1	ErrImageNeverPull	0	19h

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
frontend	NodePort	10.43.186.155	<none>	80:30743/TCP	19h
kubernetes	ClusterIP	10.43.0.1	<none>	443/TCP	19h
order-service	ClusterIP	10.43.49.234	<none>	3002/TCP	19h
product-service	ClusterIP	10.43.63.134	<none>	3001/TCP	19h

## 6. CI/CD

Un fichier `.gitlab-ci.yml` est utilisé pour gérer les étapes de CI/CD

## 7. Tests et démo

### Produits (GET /products)

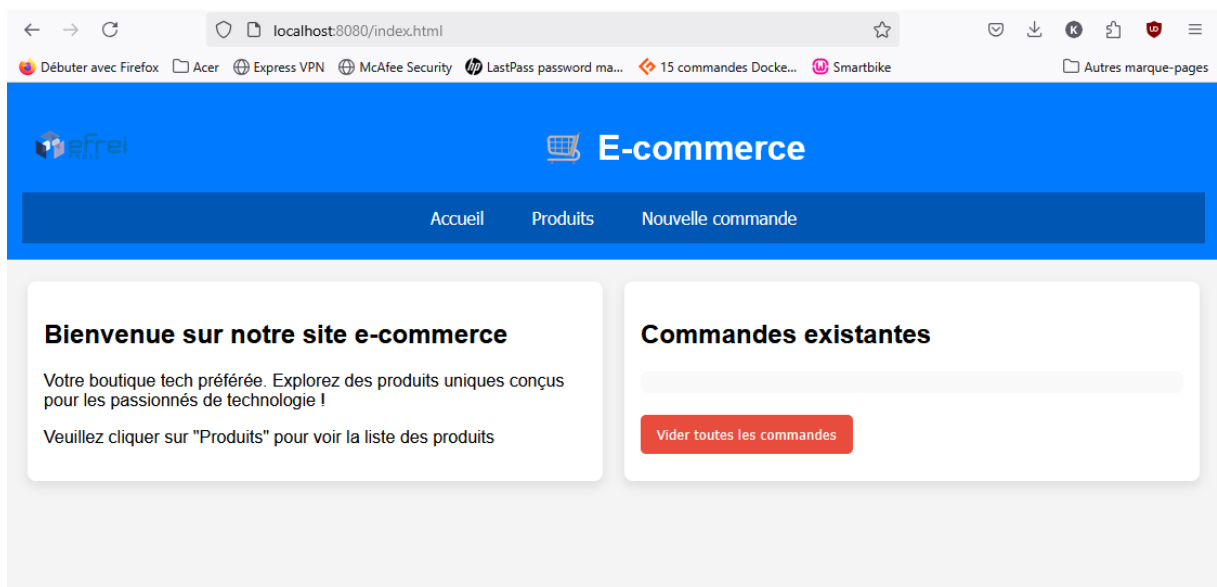
GET <http://localhost:3001/products>

### Commandes (POST /orders)

POST <http://localhost:3002/orders>

```
{  
  "productName": "Téléphone",  
  "quantity": 1,  
  "clientEmail": "client@example.com"  
}
```

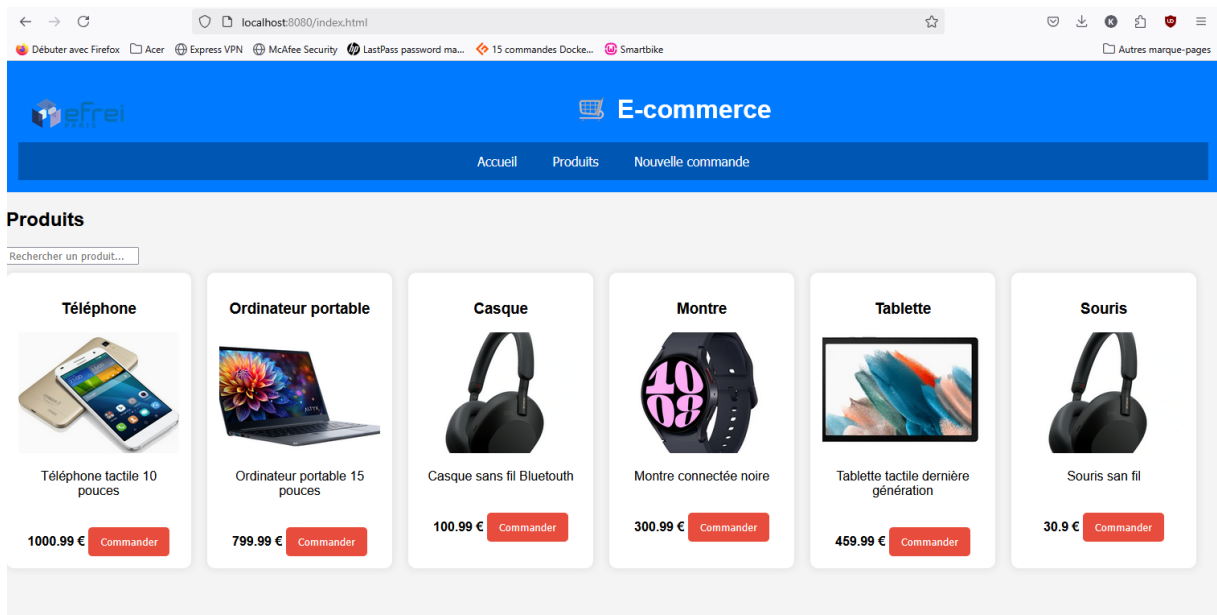
### **Page d'accueil :**



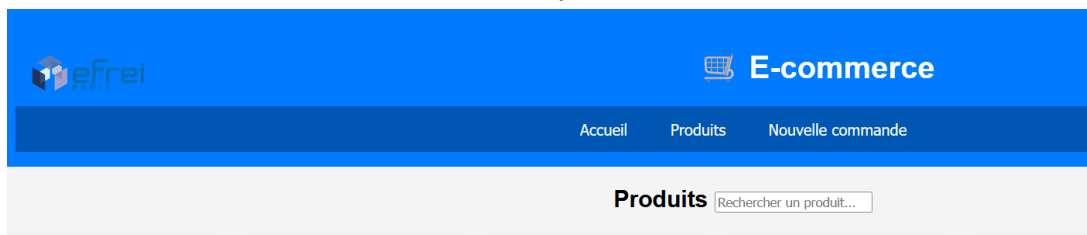
En cliquant sur le logo EFREI et sur l'onglet « Accueil », cela permet de revenir sur la page d'accueil.



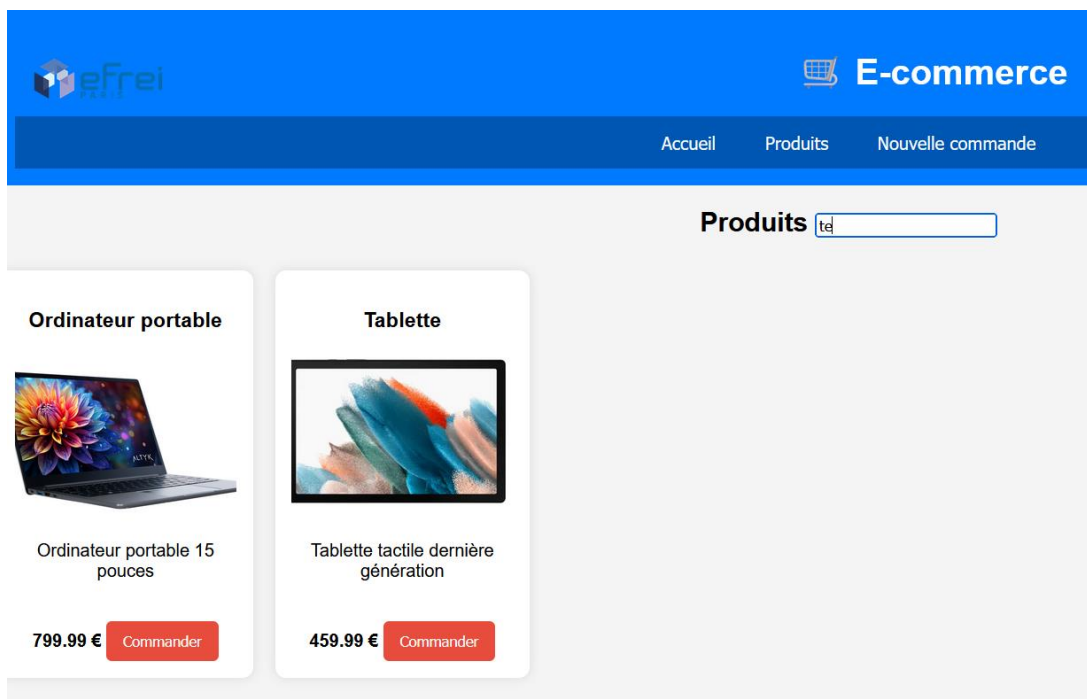
## Onglet → Produits



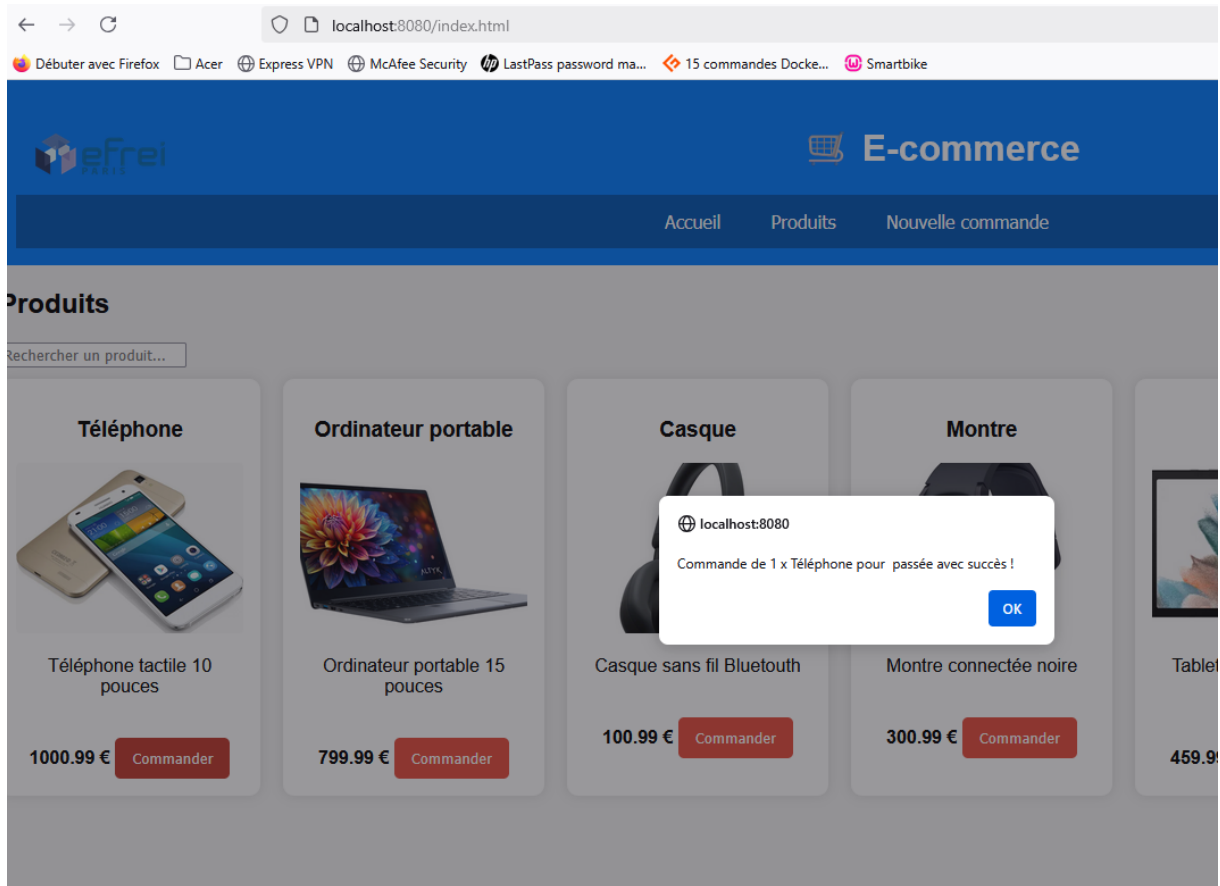
Possibilité d'effectuer une recherche de produits :



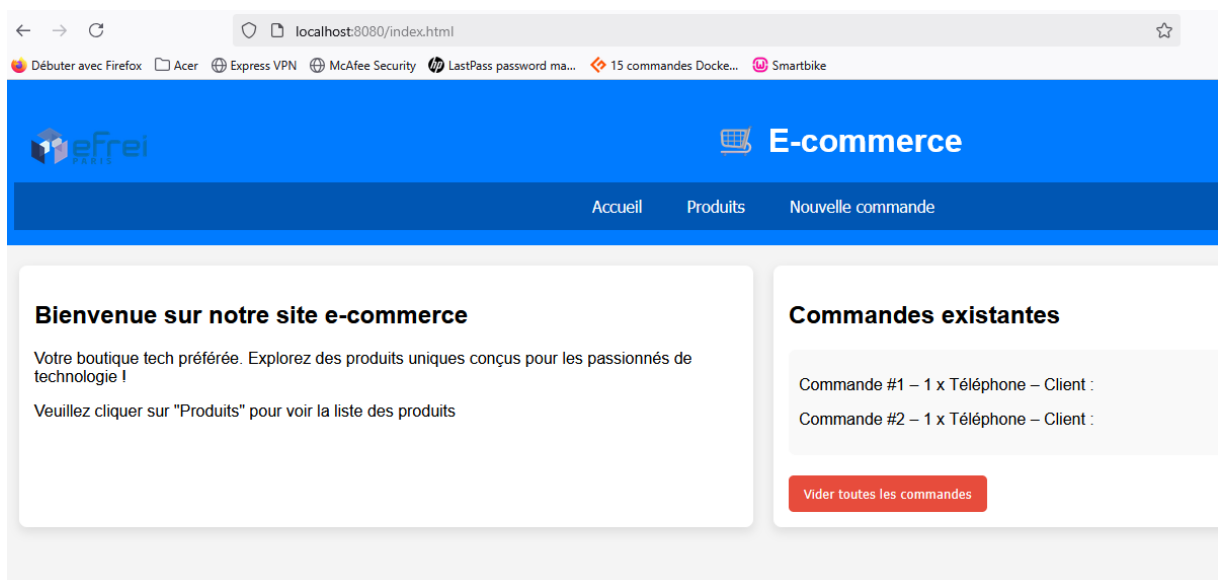
Résultat obtenu après une recherche



Le bouton « Commander » permet de passer une commande sur le produit sélectionné :



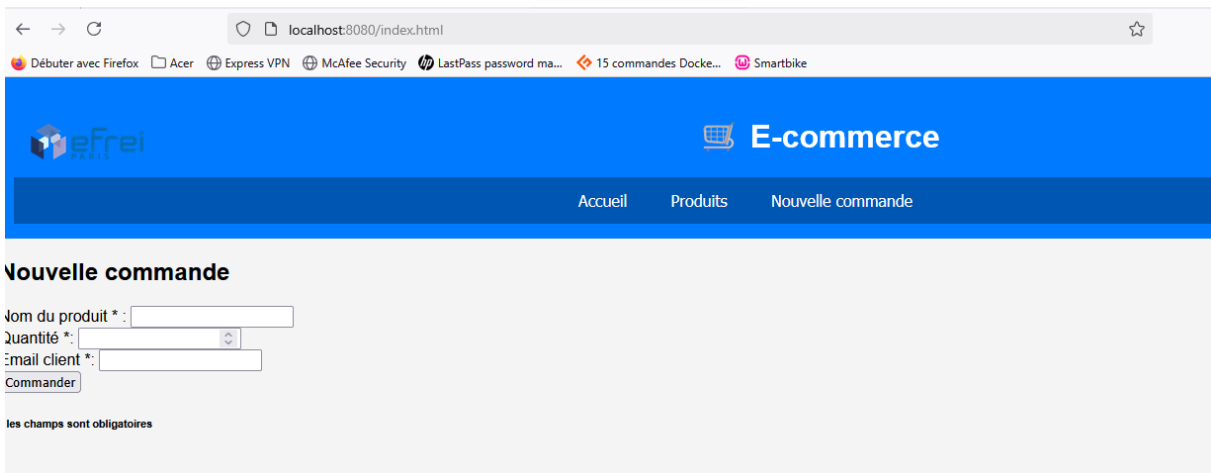
La commande est visible dans la page d'accueil dans la partie de droite « Commandes existantes »



On a la possibilité de vider toutes les commandes en cliquant sur le bouton « **Vider toutes les commandes** ».

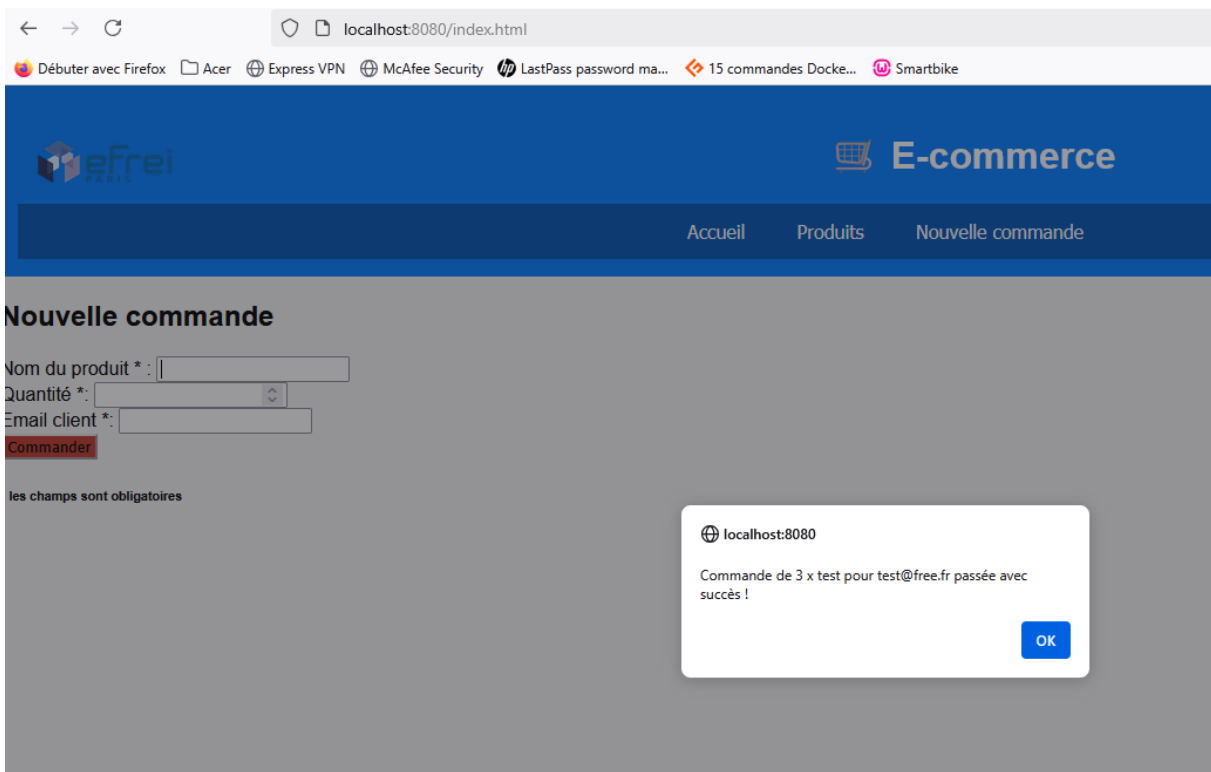
### Onglet → Nouvelle commande

Dans l'onglet « Nouvelle commande », on peut créer une nouvelle commande (tous les champs sont obligatoires) et la commande sera visible dans la page d'accueil.

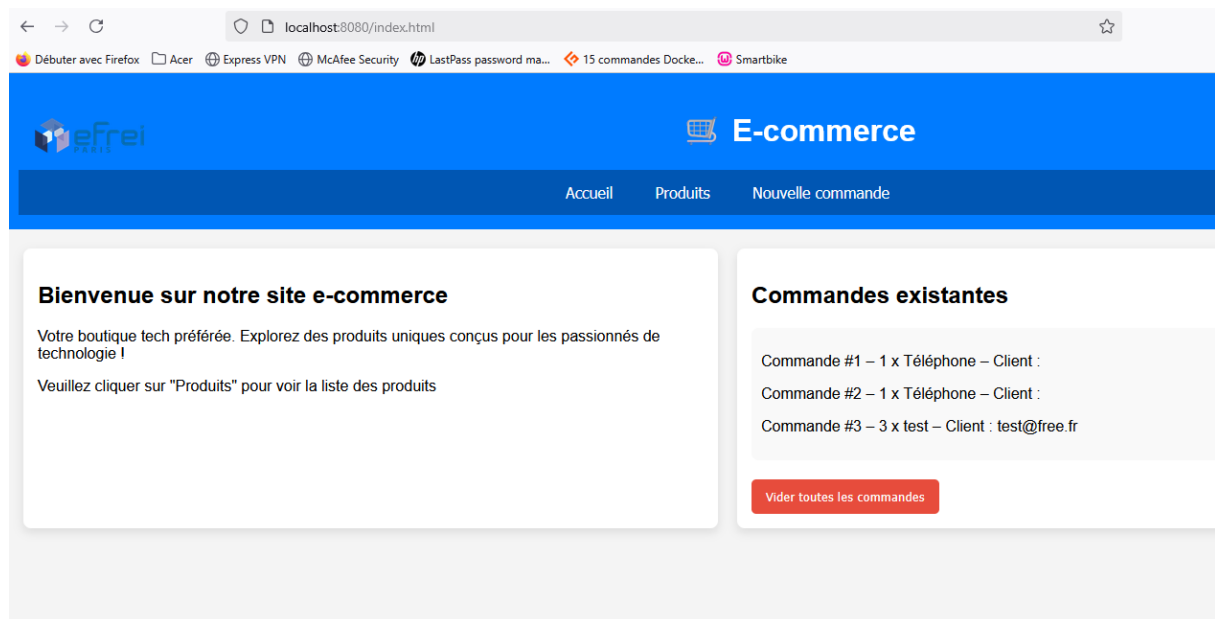


The screenshot shows a web browser window with the address bar displaying 'localhost:8080/index.html'. The browser's toolbar includes various extensions like 'Débuter avec Firefox', 'Acer', 'Express VPN', 'McAfee Security', 'LastPass password ma...', '15 commandes Docke...', and 'Smartbike'. The website's header is blue with the 'efrei' logo on the left and 'E-commerce' with a shopping cart icon on the right. Below the header is a navigation bar with three links: 'Accueil', 'Produits', and 'Nouvelle commande'. The main content area is titled 'Nouvelle commande' and contains a form with three input fields: 'Nom du produit \*:', 'Quantité \*:', and 'Email client \*:'. Each field has an asterisk indicating it is required. Below the fields is a 'Commander' button. At the bottom of the form, a note states 'les champs sont obligatoires'.

Exemple:



This screenshot shows the same 'Nouvelle commande' form as the previous one, but with a success message overlay. The form fields are now filled with example data: 'Nom du produit \*:' is 'test', 'Quantité \*:' is '3', and 'Email client \*:' is 'test@free.fr'. The 'Commander' button is highlighted in red. A white modal box with a blue border is centered on the screen, displaying the message: 'localhost:8080' followed by 'Commande de 3 x test pour test@free.fr passée avec succès !'. There is an 'OK' button in the bottom right corner of the modal box. The background of the page is dimmed to show the modal box.



## 8. Lien vers le dépôt Git

git init

git add README.md

git commit -m "first commit"

git branch -M main

git remote add origin https://github.com/Kelyan1305/Projet\_Devops\_docker.git

git push -u origin main

[https://github.com/Kelyan1305/Projet\\_Devops\\_docker.git](https://github.com/Kelyan1305/Projet_Devops_docker.git)