

Architecture et Micro-Services : WonderMarket



Groupe : WonderMarket
Enseignant : Wallerand DELEVACQ

Membres de l'équipe et rôles



**Damien CHAILLEY et Guillaume
GAUJAC :**

Front-End



Romain RAMAMONJISOA :

Docker



**Lucas CORDIER et Rayan
RAHMANI-MERAITS :**

Back-End



Notre projet : WonderMarket

- Créer une plateforme d'achat en ligne pour toute sorte de produits, mêmes ceux mondialement en rupture (oui, nous avons la PS4)
- Nous voulions un aspect e-commerce familial (type Amazon, Instant-Gaming)
- Le client doit disposer des services courants, mettre des produits dans son panier, en enlever, rechercher des produits, etc...

Technologies

- Backend : Python, flask, requests



- IDE : Visual Studio Code, Sublime Text 3



- Logiciels BDD : PostgreSQL, PGAdmin



- Docker



Technologies

- Côté front

ANGULAR JS



CSS



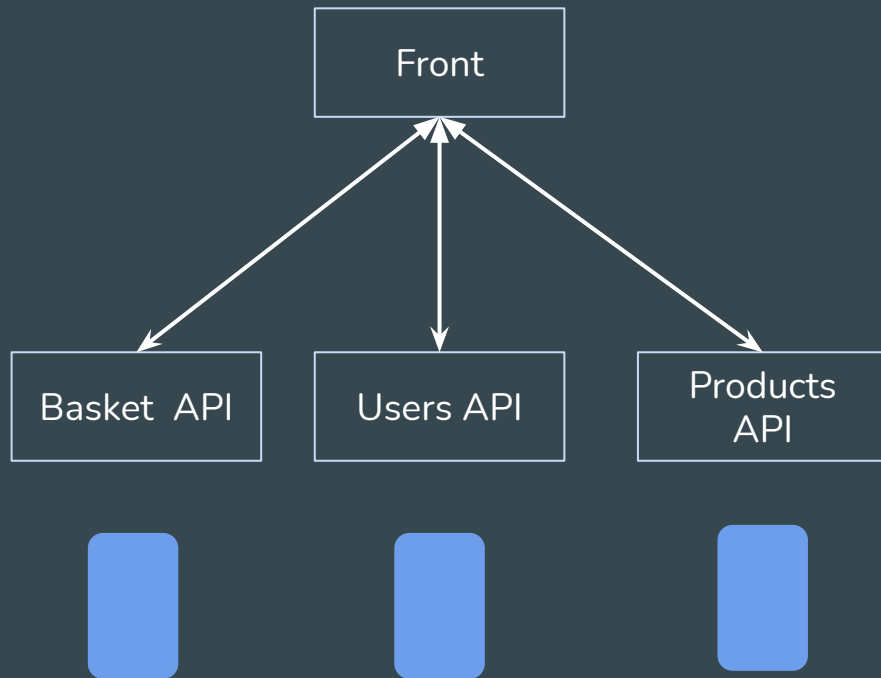
Typescript



HTML



Architecture / Plan



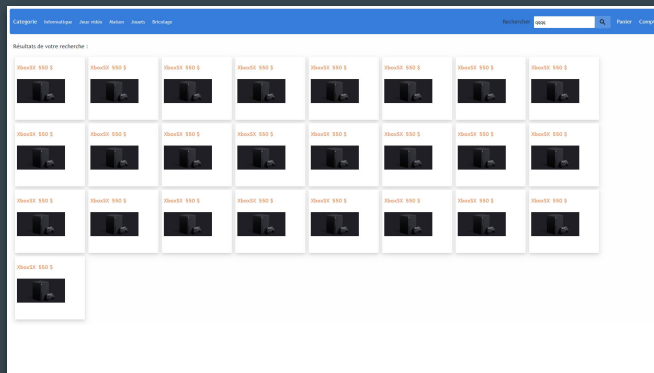
Front-End

3 composants / services :

- **Commerce** : page principale qui permet de lister l'ensemble des produits.
- **Inscription** : Disposer d'un service de connexion et d'enregistrement de compte (mot de passe, email, username)
- **Profil** : Pouvoir visualiser l'ensemble de ses informations personnelles (son mails, son historique, etc.)
- Vue précise des produits

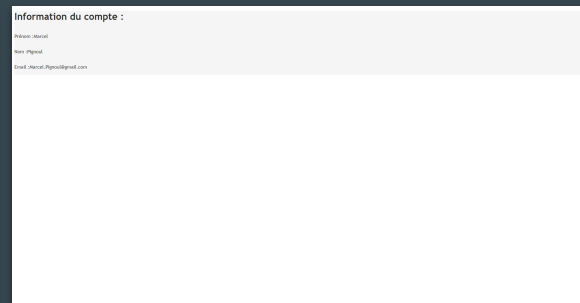
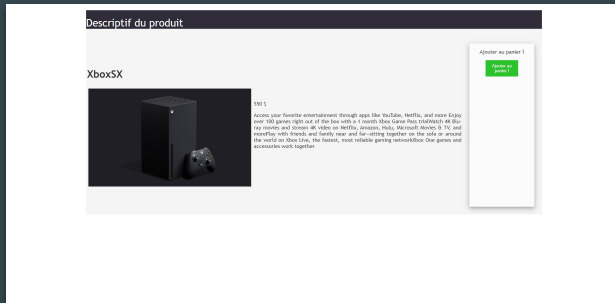
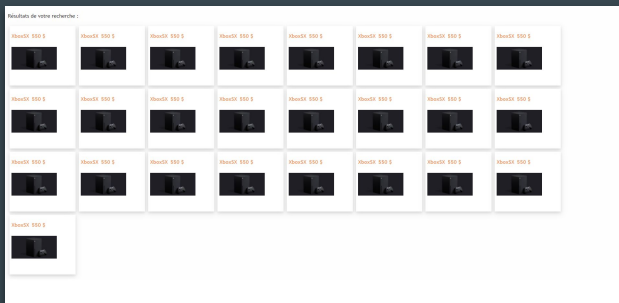
Front-End

Mécanique du front end



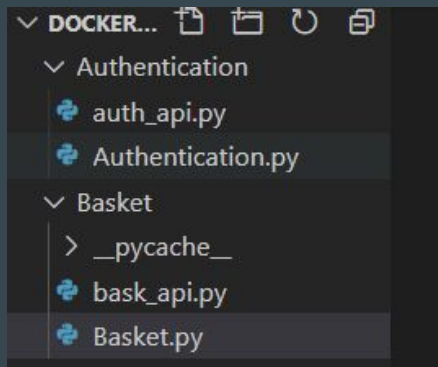
App-component

host



Back-End : Plan principal

- Séparer les différents micro-services
- Chaque mirco-service contient :
 - Un programme .py, contenant les fonctions principales propres au micro-service
 - Un programme .py, représentant l'API du micro-service
- Voici un exemple :



Back-End : Liste des micro-services et fichiers fonctions

AUTHENTICATION :

- `Authenticate(user, password) : token`

USER :

- `UserList() : Users[]` - `GetUser(id) : User`
- `CreateUser(name, surname) : bool`
- `DeleteUser(userId) : bool`

BASKET :

- `GetBasket(userId) : Basket`
- `ClearBasket(userId) : bool`

PAYEMENT

- `ValiderPanier(userId) : bool`

PRODUCTS :

- `ProductList() : Products[]`
- `GetProduct(productId) : Product`
- `GetProductsByName(productName) : Products`
- `GetProductsByCategory(productCategory) : Products`
- `CreateProduct(name) : bool`
- `DeleteProduct(productId) : bool`

BASKET :

- `GetBasket(userId) : Basket`
- `ClearBasket(userId) : bool`

Docker

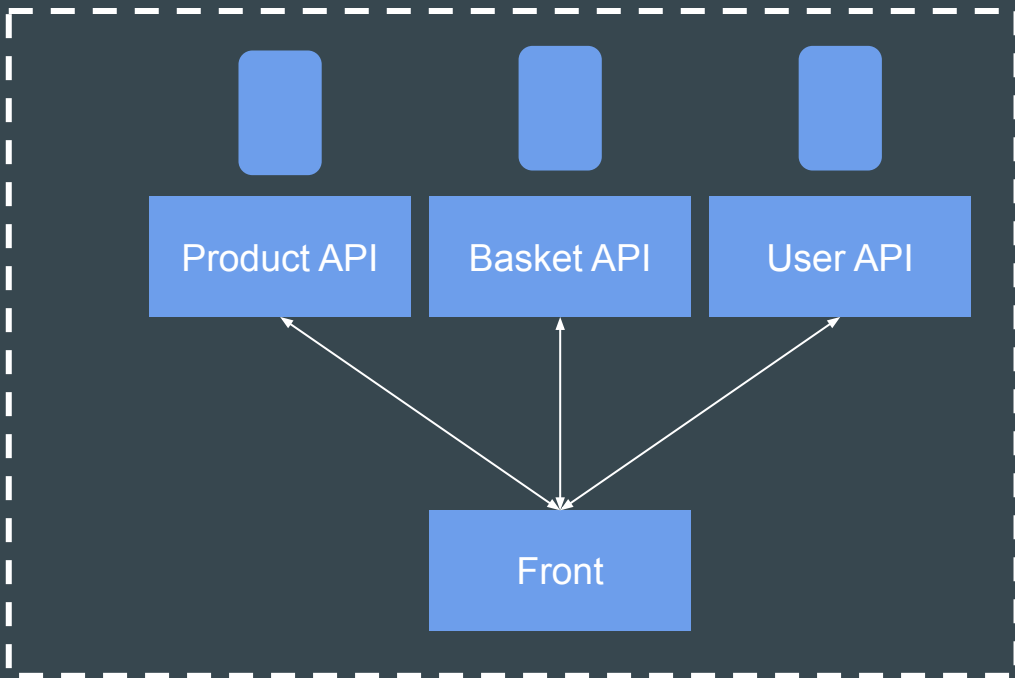
Docker compose



Créer les databases

Créer les volumes

Connecter les conteneurs entre eux



Résultat et conclusion

Bon avancement

Projet très modulaire et évolutif

Quelques difficultés mais plus évolutif

Merci pour votre attention !