

# Rapport Architecture micro service

## Membres de l'équipe :

Front : Guillaume GAUJAC , Damien CHAILLEY

Back : Lucas CORDIER Rayan RAHMANI-MERAITS

DOCKER : Romain RAMAMONJISOA

## **Sommaire :**

<b>Préambule</b>	<b>3</b>
<b>Frontend</b>	<b>4</b>
<b>Backend</b>	<b>5</b>
<b>Architecture</b>	<b>6</b>
<b>Résultats</b>	<b>7</b>
Vue d'ensemble des différentes parties de l'application	7
<b>Conclusion</b>	<b>9</b>
<b>Procédure de lancement</b>	<b>10</b>

# Préambule

Ce rapport regroupe, à la date du 28 février 2021, toutes les informations relatives au travail effectué dans le cadre du projet WonderMarket. Ce rendu s'inscrit dans le cadre du module d'Architecture et Micro-services (20\_E5FI\_5I\_IN6).

Nous avons été amené à élaborer un site d'e-commerce qui dispose de tous les services courants qui ont trait à l'achat en ligne (connexion, gestion du panier, recherche de la catégorie et du produit désiré, etc.), à l'instar des marketplaces en ligne telles qu'Amazon ou encore Cdiscount.

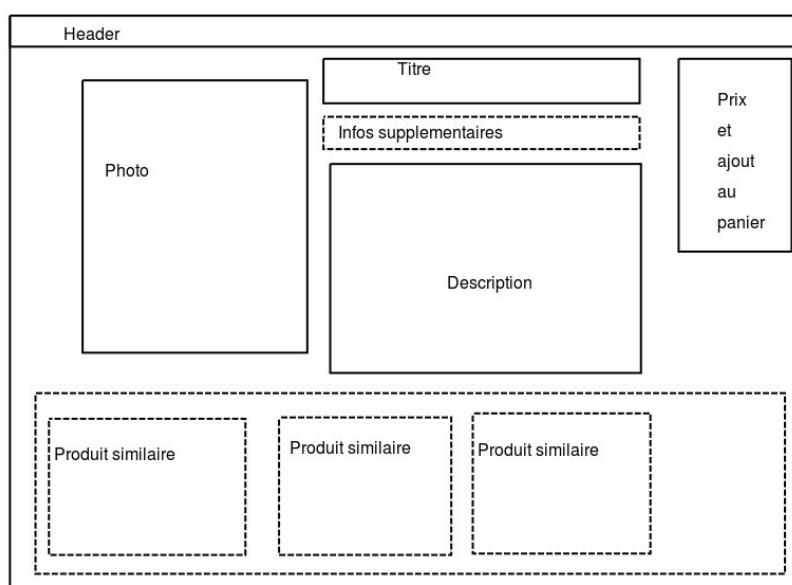
Notre objectif principal était que le site devait adopter une architecture qui se base sur différents micro-services, aussi bien sur sa partie visible (front-end) que sur sa partie cachée (backend).

Au cours de ce rapport, nous allons évoquer comment nous nous y sommes pris pour chacune de ces parties de l'application (frontend & backend).

# Frontend

Avant de commencer la phase de développement, les différents sous-groupes se sont réunis au préalable lors de la première séance afin de définir l'IHM ainsi que les services présents sur chacune des parties du site.

Concernant l'IHM du site, nous souhaitons que l'utilisateur ne soit pas dépaycé, de ce fait nous avons opté pour un design sobre en adéquation avec les autres marketplaces déjà présentes en ligne.



*L'IHM retenu pour lors de la visualisation des caractéristiques d'un produit*

Le diagramme montre un formulaire de connexion intitulé 'Connexion'. Il comprend deux champs de saisie : 'Email' (avec un soulignement rouge) et 'Mot de passe'. En dessous, un bouton bleu est étiqueté 'Se connecter'.

*L'IHM retenu pour la phase de connexion*

Pour mener à bien ce développement nous avons opté pour un framework populaire et simple d'utilisation en choisissant Angular. Les membres en charge de cette partie étaient monsieur Damien CHAILLEY ainsi que monsieur Guillaume GAUJAC.

Les micro-services présent au sein de cette partie sont au nombre de trois :

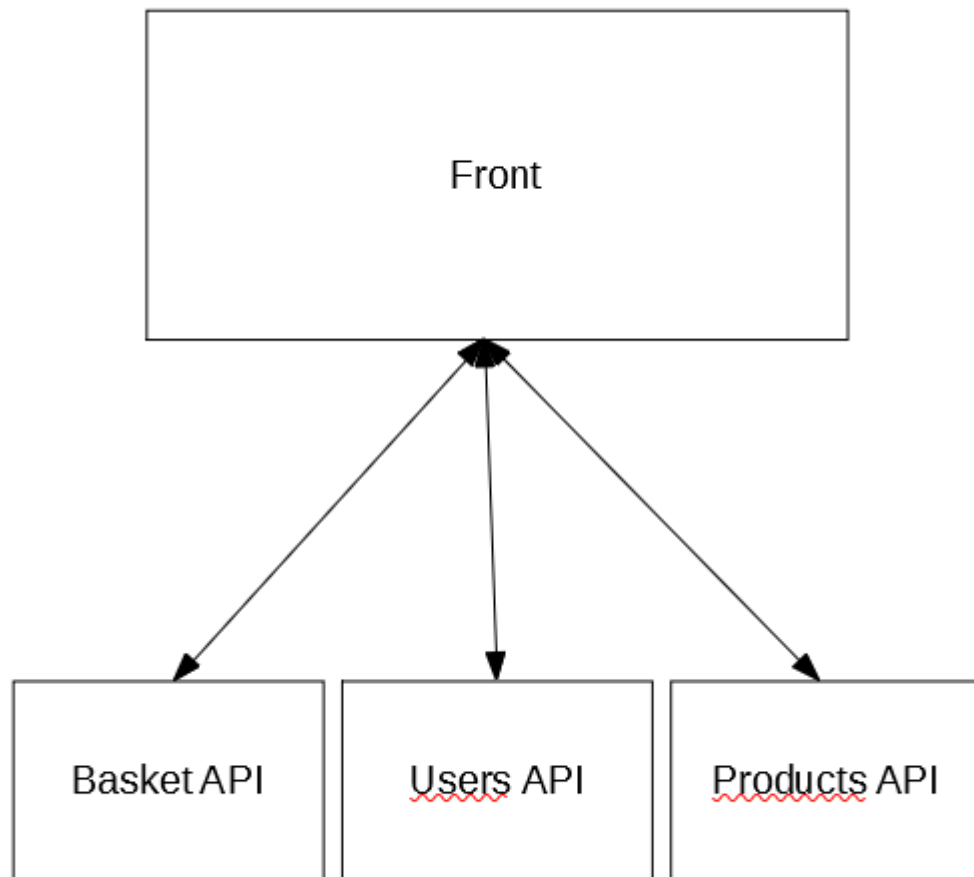
- le composant “Commerce” : ayant pour but de lister l’ensemble des produits. Elle fait guise de page principale.
- le composant “Inscription” : ayant pour but de disposer d’un service de connexion et d’enregistrement de compte (mot de passe, email, username).
- le composant “Profil” : ayant pour but de visualiser l’ensemble de ses informations personnelles (son mail, son historique, etc.)
- ainsi qu’une vue précise des produits.

## Backend

Concernant le backend, celui-ci met à disposition différentes API qui ont pour but d'interagir avec l'ensemble des actions qui permettent le bon fonctionnement d'une marketplace en ligne. Au nombre de 4, ces services sont :

- authentication : qui comme son nom l’indique regroupe l’ensemble des interactions permettant l’authentification d’un utilisateur au sein de l’application. Elle fournit la méthode *Authenticate(user, password)* qui retourne un token de connexion.
- user : qui permet de gérer la liste des utilisateurs au sein de l’application. Pour se faire elle peut retourner l’ensemble des utilisateurs grâce à la méthode *UserList()* ou encore un seul utilisateur avec *GetUser(id)*. De plus elle permet l’ajout et la suppression d’un utilisateur grâce, respectivement, aux méthodes *CreateUser(name, surname)* et *DeleteUser(userId)* qui retournent un booléen en fonction de la réussite de l’opération.
- basket : qui sert à retourner l’ensemble des articles présents au sein d’un panier ou alors à supprimer l’ensemble de ces produits selon l’utilisateur qui est donné en paramètre. Les méthodes mises à disposition sont : *GetBasket(userId)* qui retourne un objet *Basket* ainsi que *ClearBasket(userId)* qui retourne un booléen selon la réussite de cette opération (*false* si l’opération est impossible et *true* si celle-ci s’est fait sans encombre).
- products : qui gère l’ensemble des produits au sein de l’application. Elle permet notamment de lister l’ensemble des produits grâce aux méthodes *ProductList()* qui retourne l’ensemble des produits ou encore *GetProduct(productId)* qui en retourne le produit spécifié en paramètre. Nous pouvons également faire des recherches plus précises telles que *GetProductByName(productName)* qui permet de trier par le nom du produit ou encore *GetProductsByCategory(productCategory)* qui lui permet de trier par catégorie de produits.
- payment : qui valide la transaction grâce à la méthode *ValiderPanier(userId)* qui retourne un booléen en fonction de la réussite de l’opération.

# Architecture

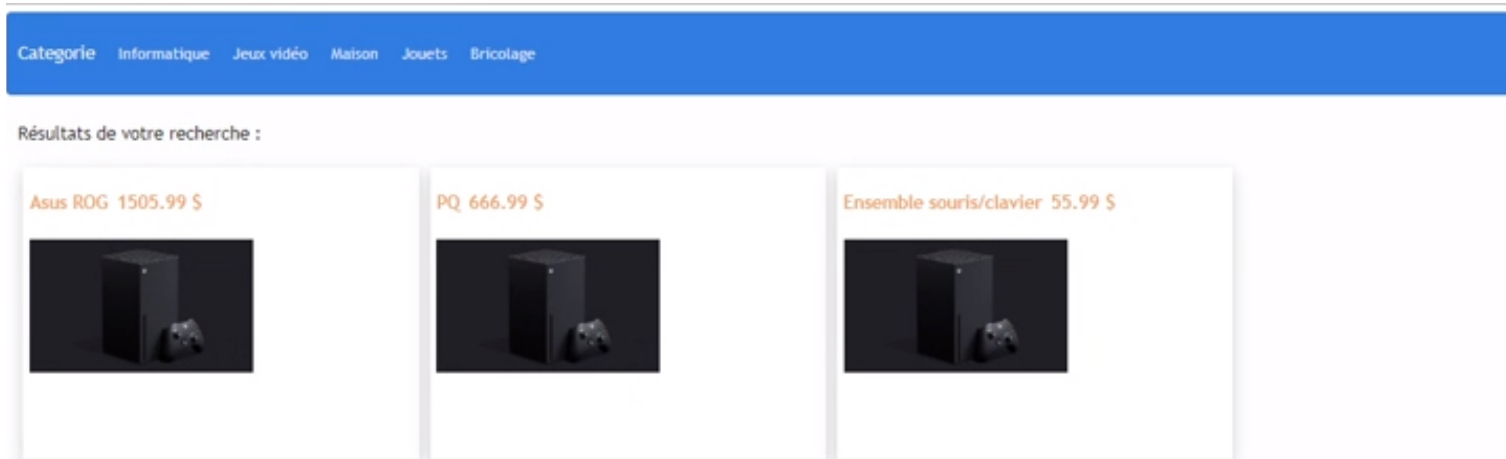


À l'heure actuelle, l'architecture se présente sous la forme suivante. Les modules implémentés sur le front font appel aux trois modules nécessaires au bon fonctionnement des modules implémentés sur le front.

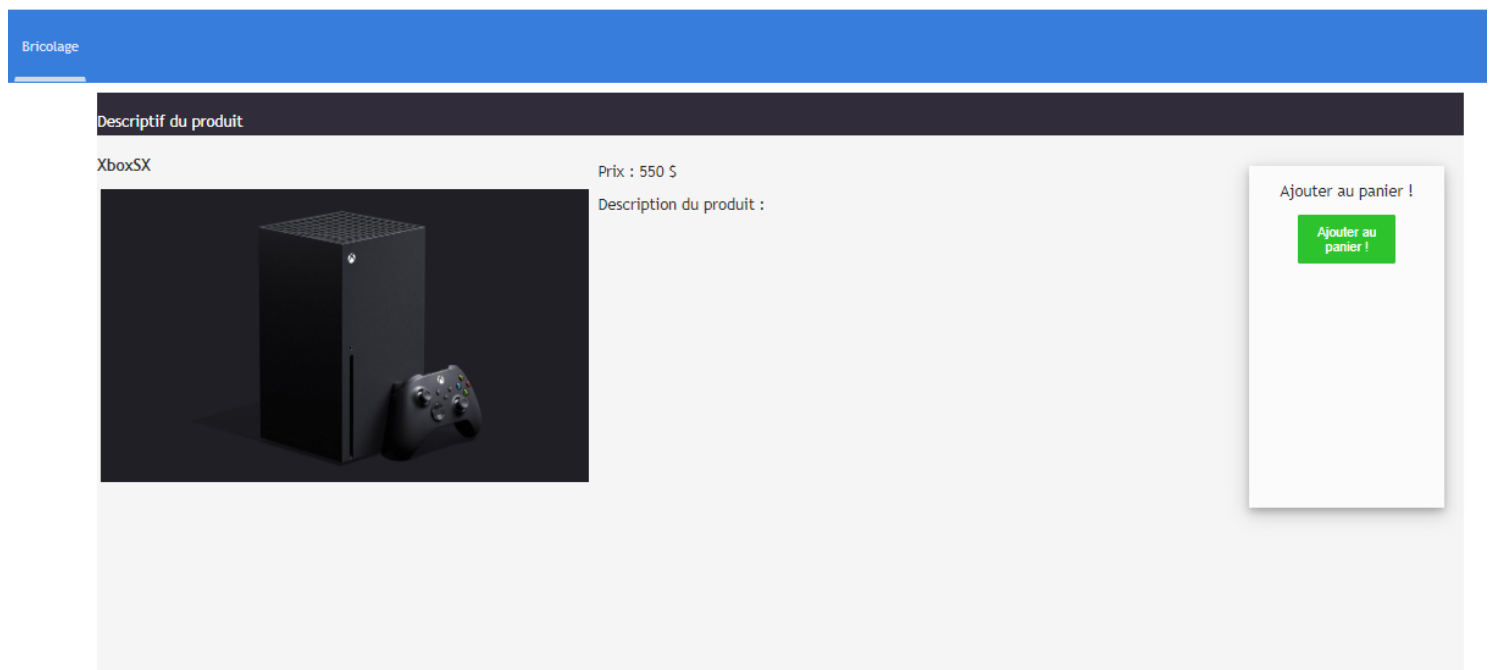
# Résultats

## Vue d'ensemble des différentes parties de l'application

**Composant commerce** : là se trouve tous les produits



**Composant description** hébergeant le composant **achat** :



**Composant compte** : contenant les informations de l'utilisateur

Categorie	Informatique	Jeux vidéo	Maison	Jouets	Bricolage
-----------	--------------	------------	--------	--------	-----------

Information du compte :

Prénom :Marcel

Nom :Pignoul

Email :Marcel.Pignoul@gmail.com

**La bar de navigation** toujours présente peu importe l'endroit où l'on se trouve.

Categorie	Informatique	Jeux vidéo	Maison	Jouets	Bricolage	Rechercher	<input type="text" value="xbox"/>		Panier	Compte
-----------	--------------	------------	--------	--------	-----------	------------	-----------------------------------	--	--------	--------


**Composant panier** :

Categorie	Informatique	Jeux vidéo	Maison	Jouets	Bricolage	Rechercher	<input type="text" value="xbox"/>		Panier	Compte
-----------	--------------	------------	--------	--------	-----------	------------	-----------------------------------	--	--------	--------

PQ

Prix : 666.99 \$

Description du produit : Pas trop mal




Supprimer l'article

Asus ROG

Prix : 1505.99 \$

Description du produit : c chere mais c pas mal



Supprimer l'article

Vous disposez de produits dans votre panier, n'attendez plus, passez la commande.

Passer la commande !



# Conclusion

Ce projet nous a permis de voir comment nous pouvions structurer un projet en plusieurs micro-services. L'avantage étant que chaque partie est distincte et modulaire.

Nous sommes parvenus à faire ce que nous souhaitions à savoir un amazon like avec un front utilisant le framework Angular (ts, html, css) et un back nous permettant de faire des requêtes flask à une api, le tout assemblé à l'aide de docker-compose.

Quelques remarques importantes :

Il manque quelques API à connecter côté front mais néanmoins le fonctionnement et le principe de micro-service est mis en place et les éléments du front (conteneur) communiquent bien avec le back (conteneur).

Nous avons rencontré quelques difficultés avec l'utilisation du framework Angular qui ne nous était pas familier et notamment sur les requêtes. Nous sommes néanmoins parvenus à un résultat assez satisfaisant selon nous.

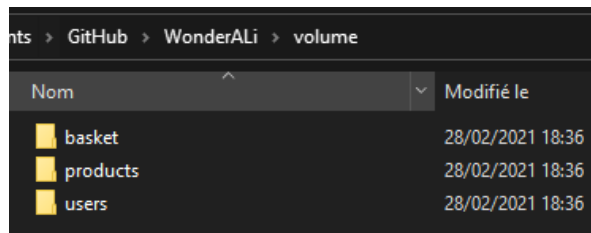
La partie connexion a également posé soucis à la personne qui était en charge de cette partie. En effet, nous étions parti dans l'optique d'utiliser des tokens JWT afin de s'améliorer dans cette compétence d'autant plus que cela nous aurait permis d'avoir quelque chose en accord avec les normes de sécurité actuelle. Néanmoins nous ne sommes pas parvenus au résultat escompté.




Nous n'avons pas gérer les images des produits comme vous avez pu le constater ou comme vous pourrez le constater. Il s'agit d'un axe d'amélioration à ce projet. Par ailleurs, il nous semble qu'il existe des conteneurs spécifiques pour stocker les images.

Dans le cas où vous rencontrez des difficultés, vous trouverez ci-joint un lien remontrant l'application en marche. <https://streamable.com/ifa2fd>

# Procédure de lancement

- 1) Récupérer l'archive du projet
- 2) Vérifier que le contenu des sous-répertoires de volume est bien vide.



nts > GitHub > WonderALi > volume	
Nom	Modifié le
 basket	28/02/2021 18:36
 products	28/02/2021 18:36
 users	28/02/2021 18:36

- 3) Utiliser la commande docker-compose up (c'est un peu long)
- 4) Attendre (longtemps, très longtemps)
- 5) Se rendre sur localhost (sans aucun port derrière, il y a une redirection)