Service Oriented Architecture (SOA)

SOMMAIRE

Auteurx : TRAORÉ Fayçal Samir Zégué

Identifiant INSA: ftraore

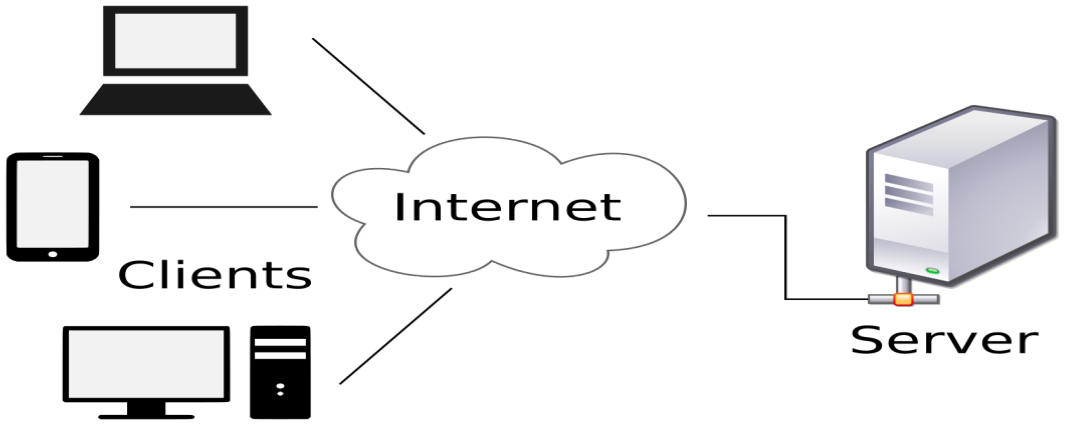
Email: ftraore@etud.insa-toulouse.fr

Introduction

Les architecture d'application indiquent la manières de déployer une application de sorte à la rendre accessible. Suivant l'architecture utilisée pour ce déploiement, les performances, la disponibilité, la généricité, etc sont beaucoup impactés

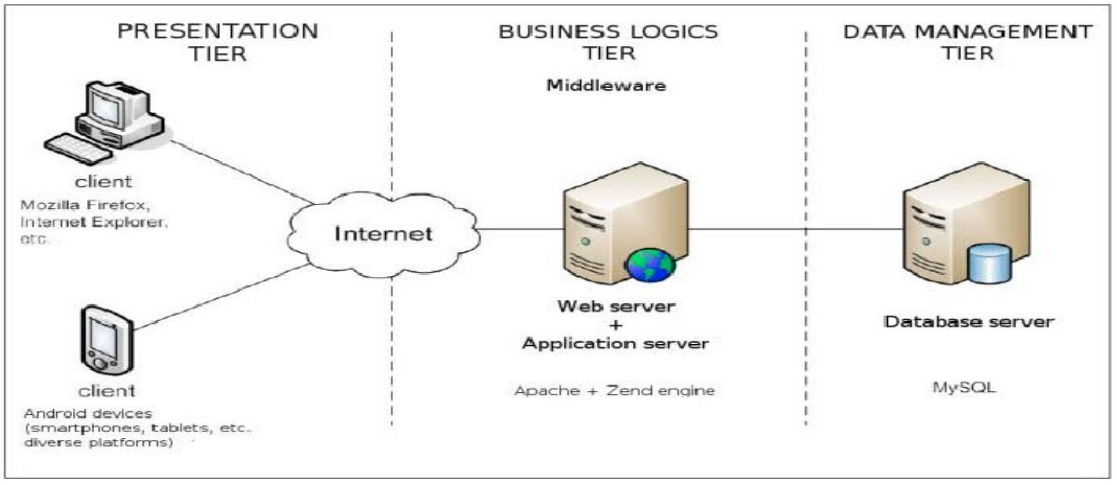
Les différents modèles d'architecture applicative

- La première consiste à exécuter une application en local, c'est à dire sur une même machine. C'est la plus basique et était utilisée avant l'apparition d'internet.
- L'architecture Client-Serveur, est une architecture composée en 2 partie: un client qui fait des requêtes et affiche les résultats renvoyés par le serveur qui répond aux requêtes. C'est une architecture 2-tier.



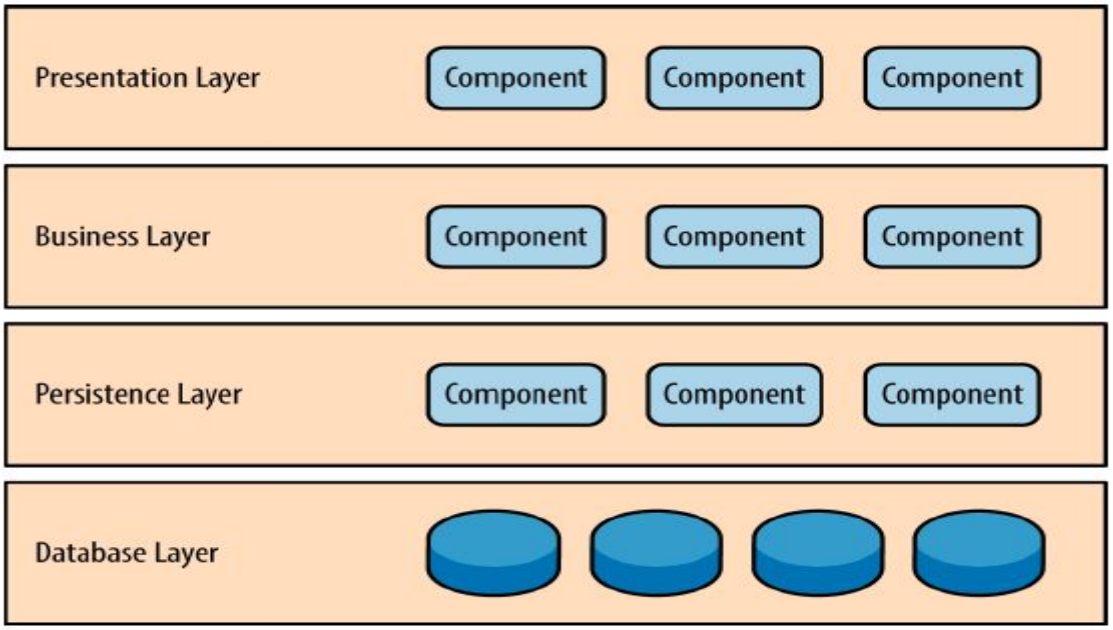
Architecture client-server (2-tier)

- L'architecture à 3 couches(niveaux) ou architecture 3-tier composée d'une couche de présentation, une couche de traitement et une dernière couche pour les données



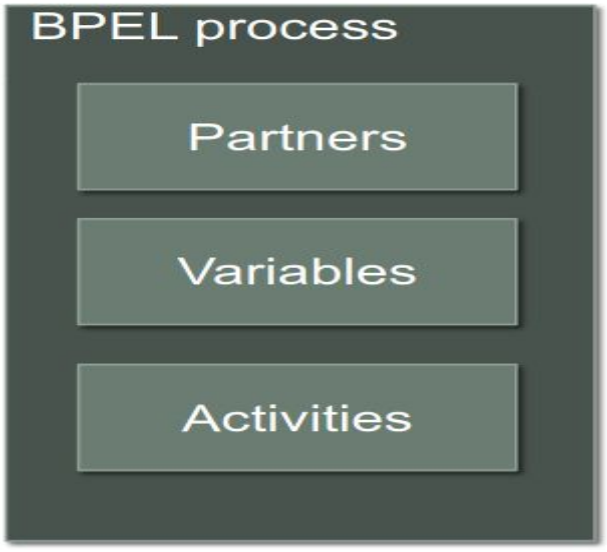
Architecture 3-tier

- Les application N-tier qui a toujours 3 couches comme l'architecture précédente mais permet de distribuer la couche applicative(couche de traitement) sur plusieurs serveurs



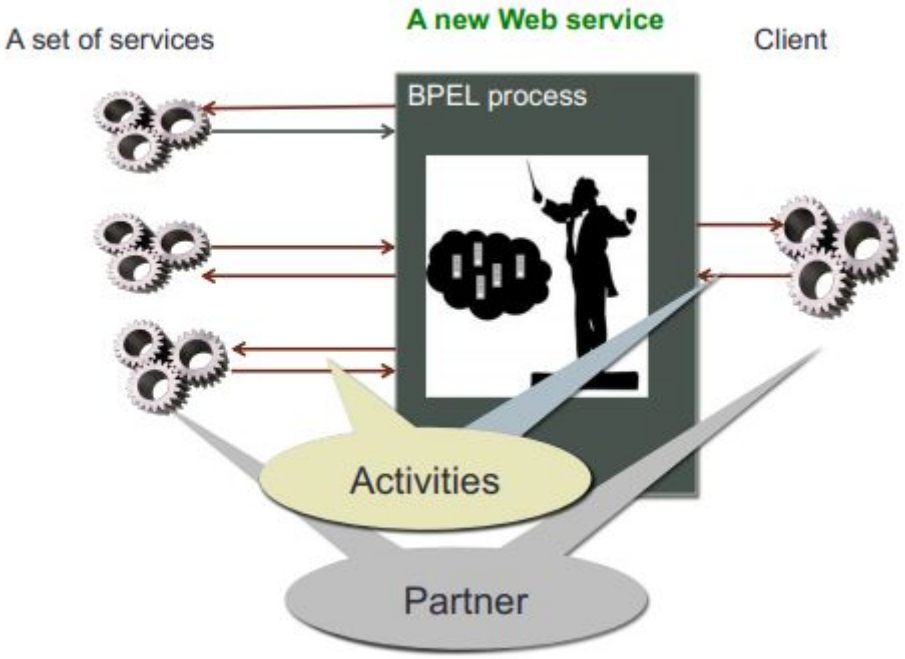
Architecture N-tier

Ce modèle d'architecture implique la communication et l'interaction de nombreux services. Pour permettre une bonne orchestration entre eux, des langages ont été mis au point. L'un d'entre eux, le BPEL (Business Process Execution Language) permet la spécification et l'exécution des compositions de services. Il permet aussi de gérer la programmation des activités et la gestion des erreurs. Typiquement, un processus décrit en BPEL se divise en 3 blocs :



- Partners : définit l'interaction entre le process BPEL et des web services,
- Variables : cette partie permet de stocker des données envoyées par les partners, et de contenir les données utilisées par le processus et utilisées par d'autres processus
- Activities : décrit les différentes étapes qui constituent le processus (chaque étape est appelée une activité, qu'elle soit simple ou structurée), et définit le flow de contrôle de ces étapes.

BLEP process (Moodle)

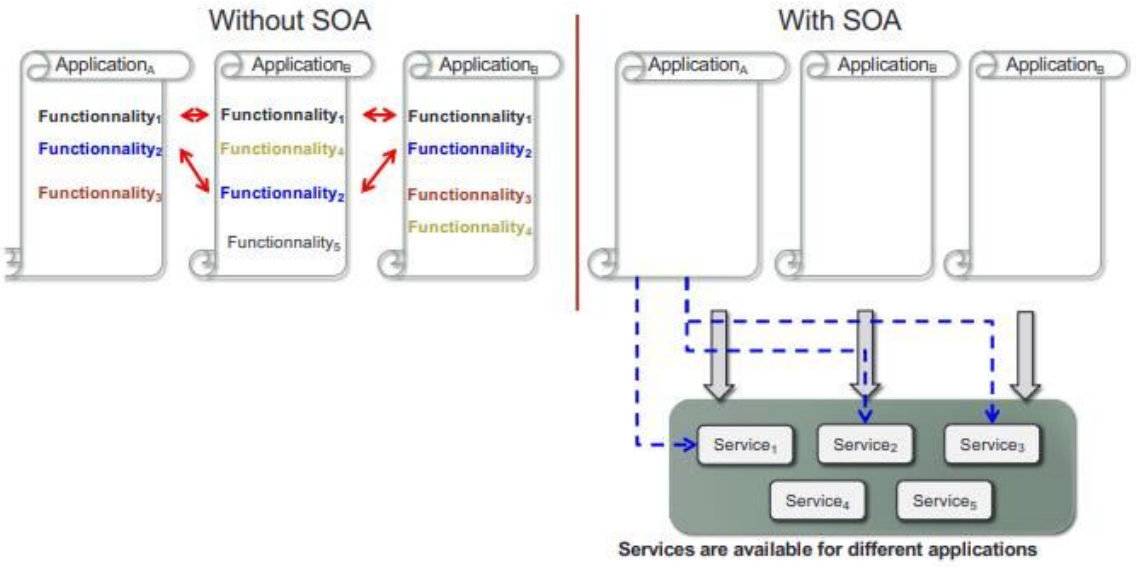


BPEL rôle

L’architecture orientée service et le concept de SOAP

L’architecture orientée services est apparue pour apporter des solutions au problème d’intégration d’applications. SOA consiste à encapsuler les applications sous forme de briques logicielles appelées services. Ainsi, les différentes fonctionnalités requises sont exposées sous forme d’un ou plusieurs services.

Par conséquent, en suivant cette architecture, les applications peuvent être construites en composant et en réutilisant des services, à l’image des Lego. Contrairement au développement classique des applications, SOA favorise la réutilisation, l’évolution et l’intégration des applications.

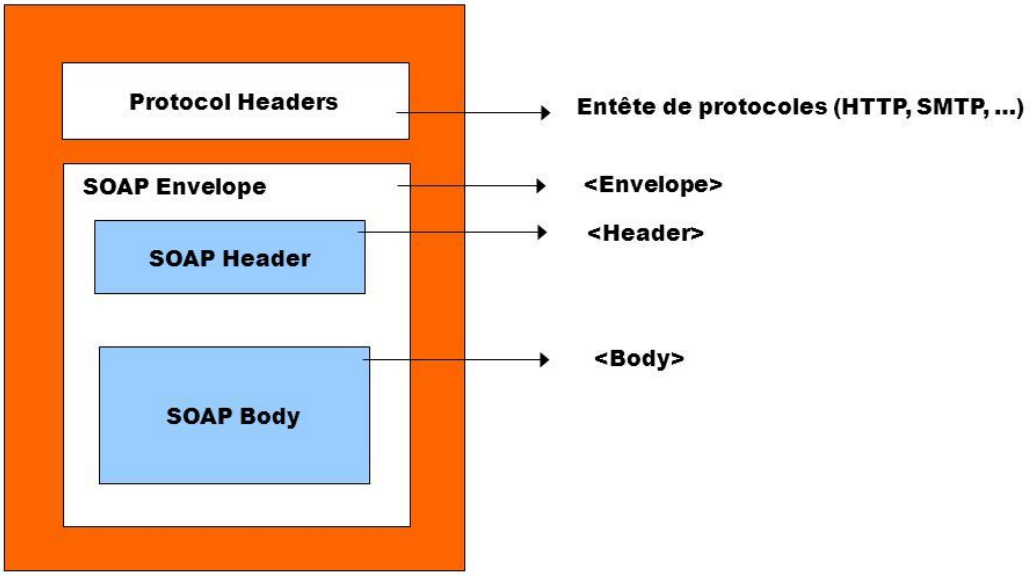


SOA vs non SOA

La technologie des services Web repose principalement sur trois standards :

- **WSDL** (Web service description language) ou, autrement dit, un langage de description qui est basé sur XML. Le but de ce langage est de décrire l'interface d'un service Web indépendamment de son implémentation. Cette description fournit des informations nécessaires à l'appel du service, comme les opérations qu'il offre, les types de données supportées, les protocoles utilisés et l'adresse du service. Elle peut être publiée dans des annuaires respectant le standard **UDDI** (Universal Description Discovery and Integration).
- Les services Web communiquent via l'échange de messages qui respectent un certain format. C'est justement ce que permet de faire **SOAP (Simple Object Access Protocol).** SOAP est un standard de communication basé sur XML.
- Les messages SOAP sont ensuite envoyés en utilisant un protocole de transport tel que HTTP.

SOAP



Structure d'un message SOAP

➔ L'enveloppe (une par message) contient les namespaces (entre autres : adresse du web service)

➔ Le header (optionnel), permet d'ajouter des informations si le message transite entre différents intermédiaires, ou si une authentification est nécessaire

➔ Le corps qui contient les informations pour le destinataire.

Trois acteurs interviennent lors de l'utilisation d'une application SOA :

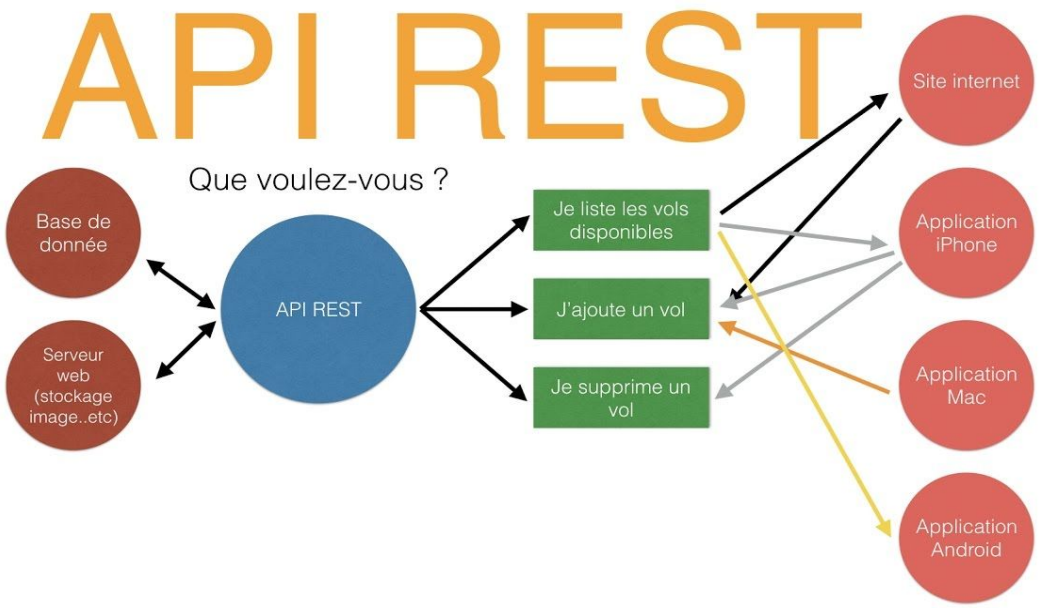
- Le provider : il développe, décrit et déploie le service. Il reçoit les requêtes du client et y répond.
- Le client : lance les requêtes au service de son choix, en respectant la description effectuée dans le directory.
- Le directory : propose une description du service aux clients.

Le recours à cette architecture implique donc l'implémentation d'une couchée supplémentaire de communication (tel que HTTP) et augmente l'utilisation de la bande passante engendrée par la verbosité de la bande passante.

L'architecture orientée ressource (REST)

Comme vu précédemment, dans le cadre d'une SOA, le SOAP offre de nombreux avantages : la standardisation des protocoles, un haut niveau de sécurité et de fiabilité, l'utilisation d'une couche de transport générique...

Mais les services SOAP présentent aussi des contraintes : verbosité et nécessité de faire appel à une couche protocolaire supplémentaire, complexité de la mise en place, entre autres. Une architecture alternative a donc été imaginée: l'architecture REST.



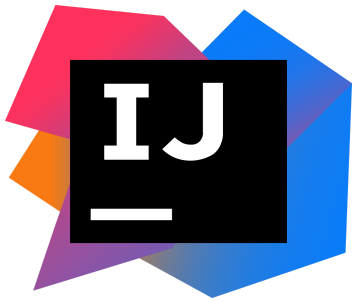
Architecture REST

REST (representational state transfer) est un style d'architecture logicielle définissant un ensemble de contraintes à utiliser pour créer des services web. Les services web conformes au style d'architecture REST, aussi appelés services web RESTful, établissent une interopérabilité entre les ordinateurs sur Internet. Les services web REST permettent aux systèmes effectuant des requêtes de manipuler des ressources web via leurs représentations textuelles à travers un ensemble d'opérations uniformes et prédéfinies sans état. Dans un service web REST, les requêtes effectuées sur l'URI d'une ressource produisent une réponse dont le corps est formaté en HTML, XML, JSON ou un autre format. La réponse peut confirmer que la ressource stockée a été altérée et elle peut fournir des liens hypertextes vers d'autres ressources ou collection de ressources liées. Lorsque le protocole HTTP est utilisé, comme c'est souvent le cas, les méthodes HTTP disponibles sont GET, HEAD, POST, PUT, PATCH, DELETE, CONNECT, OPTIONS et TRACE.

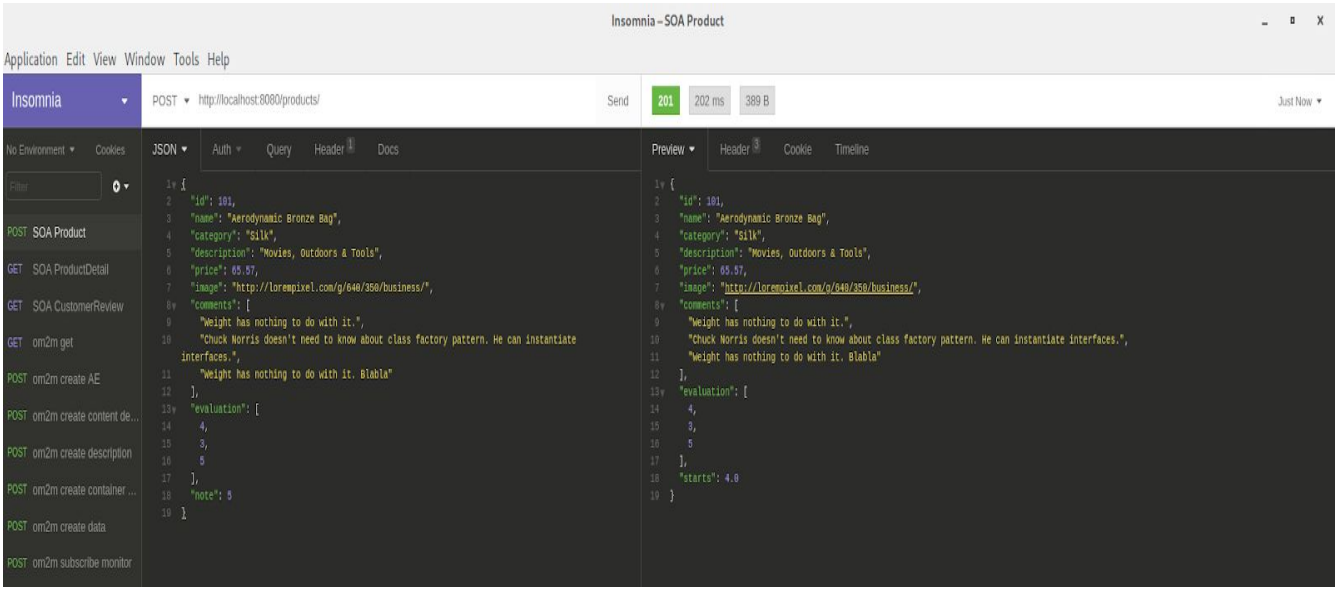
Mise en application d'une architecture avec implémentation de 3 micro-services

Pour la partie pratique de cet enseignement, nous avons développé une architecture composée de 3 microservices.
L'environnement de développement était composé:

- D’IntelliJ: l’IDE de développement sur lequel je programmais et exécutait le code
- Du framework Spring, pour la réalisation en JAVA des microservices



- Et d’insomnia comme client REST pour le débogage de l’API



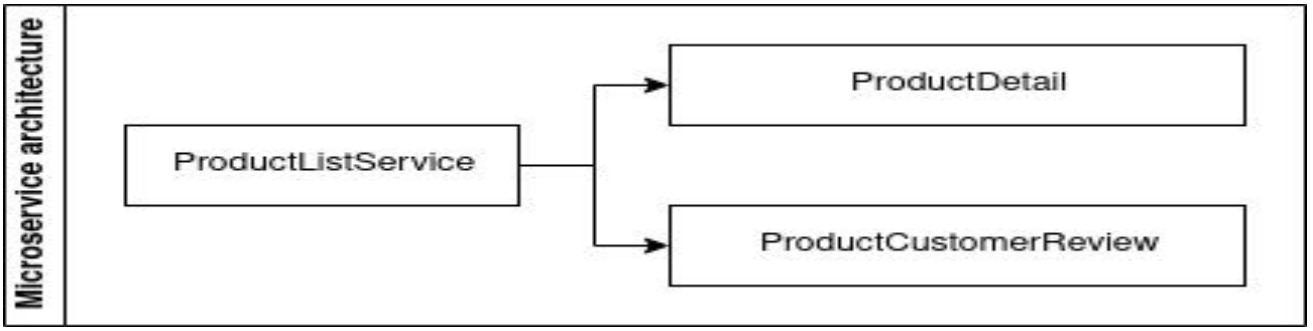
Capture d’écran Insomnia

L’application mise en place, ici est une API de E-COMMERCE permettant de gérer les produits et les détails qui lui sont liés comme son image, sa description, sa note ou les commentaires laissés par les clients.
L’architecture ce présente comme ci dessous:

- Un microservice **ProductListService** qui lui permet de récupérer toutes les informations sur un ou plusieurs produits. Il fait appel au deux microservices **ProductDetail** et **ProductCustomerReview** pour récupérer les informations complémentaires avant de les agréger et de les renvoyer en réponse. Hormis celui ci, les deux autres microservices sont indépendant et peuvent donc fonctions de façon séparé.
- Un microservice **ProductDetail** qui lui permet de gérer les informations relatives à un produit à savoir son nom, sa catégorie, sa description, son image et son prix

- Et un dernier microservice **ProductCustomerReview** qui lui permet de gérer les commentaires clients d'un produit spécifique.

Les différentes données entrées et visibles sur les captures d'écran sont générées au démarrage des différents microservices grâce à la classe **Faker** qui permet de générer des données aléatoires (catégories, description, images, nom, prénom, etc).



Product microservices architecture

Le microservice ProductListService

Il est lancé sur le port **8080** et répond via l'URL: **http://localhost:8080/products**

Il permet de:

- **GET** **/**: récupérer tous les produits
- **GET** **/ids:** récupérer tous les id des produits
- **GET** **/total:** récupérer le nombre total des produits
- **GET** **/one/{id}:** récupérer un produit spécifique
- **GET** **/all/{category}:** récupérer tous les produits d'une catégorie
- **POST** **/:** rajouter des produits
- **DELETE** **/{id}:** supprimer des produits

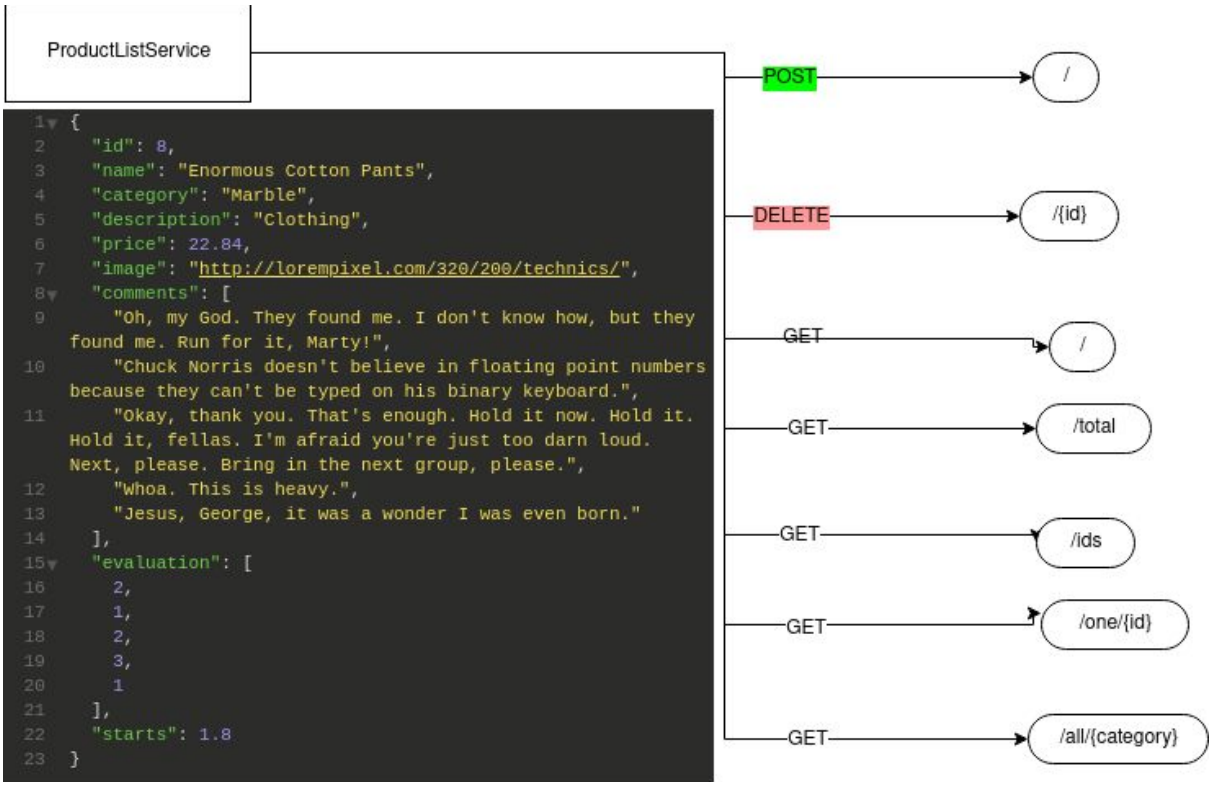


Schéma du mapping de l'API avec son modèle de données

Le microservice ProductDetail

Il est lancé sur le port **8081** et répond via l'URL: **http://localhost:8081/productDetail/**

Il permet de:

- **GET** **/{id}:** récupérer un détail de produit
- **GET** **/all/{category}:** récupérer tous les détails produits d'une catégorie
- **GET** **/categories:** récupérer la liste des catégories de produits disponible
- **POST** **/:** rajouter des détails produits
- **DELETE** **/{id}:** supprimer des détails produits

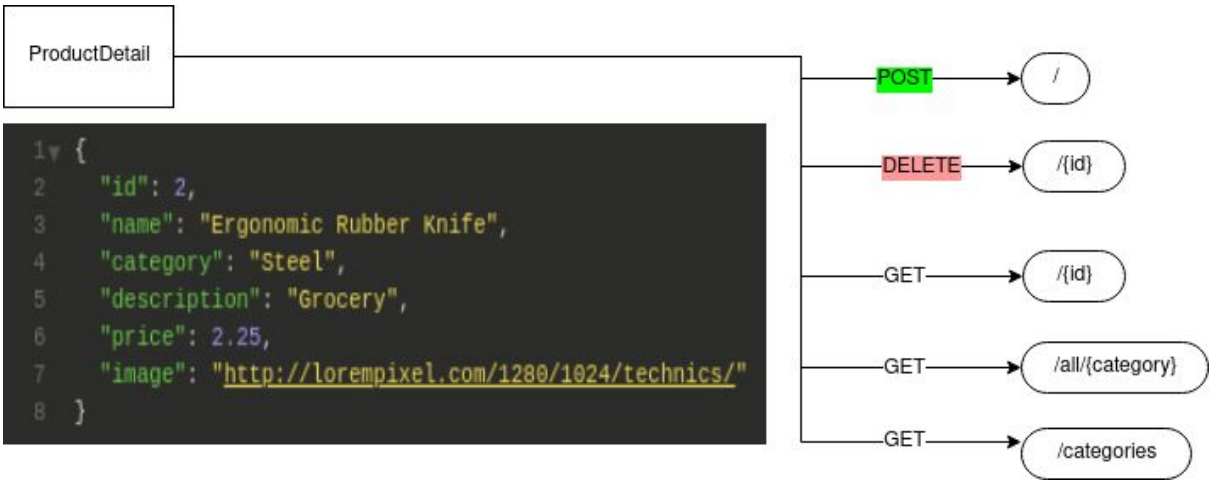


Schéma du mapping de l'API du service ProductDetail avec son modèle de données

Le microservice ProductCustomerReviewl

Il est lancé sur le port **8082** et répond via l'URL: **http://localhost:8082/reviews**

Il permet de:

- **GET** **/:** récupérer tous les commentaires de tous les produits
- **GET** **/{id}:** récupérer un commentaires
- **POST** **/:** rajouter plusieurs commentaires à la fois
- **POST** **/one:** rajouter un seul commentaire à un produit
- **DELETE** **/{id}:** supprimer un commentaire

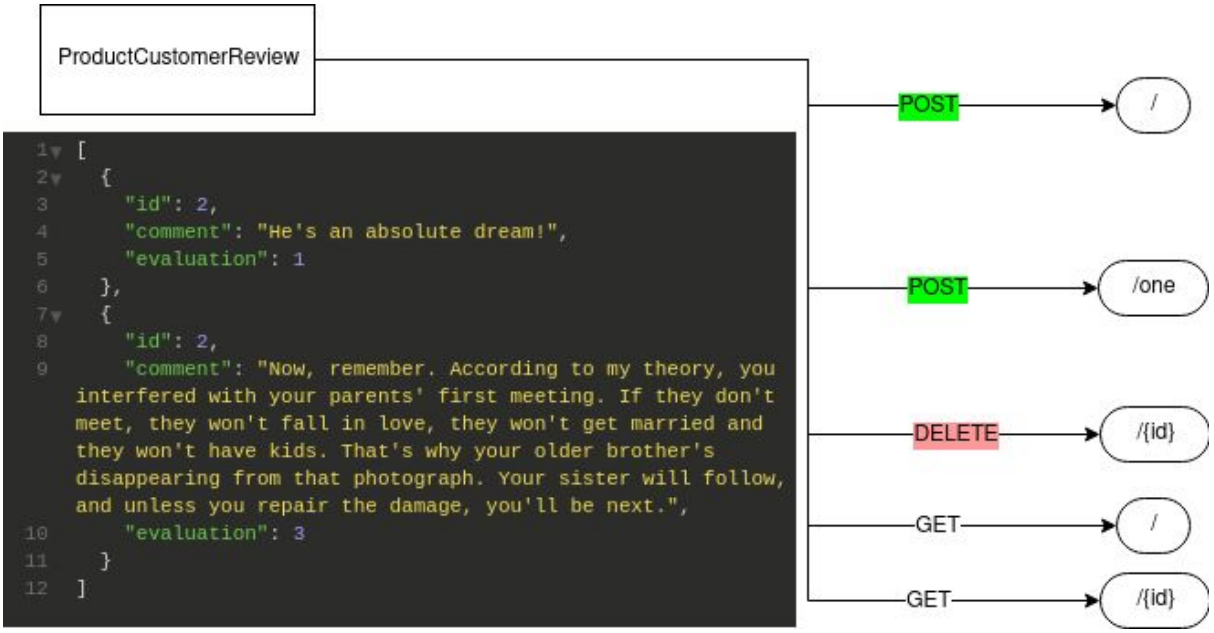


Schéma du mapping de l'API du service ProductCustomerReview avec son modèle de données

ANNEXE

Code Source Github

Les lien vers les dépôts github des différents microservices:

https://github.com/Faymir/INSA-microservice-ProductListService
https://github.com/Faymir/INSA-microservice-ProductDetail
https://github.com/Faymir/INSA-microservice-ProductReviews

Bibliographie

Moodle: I5ISSIL11/I5ISSIF11 - Architecture de service/Ingénierie Logicielle
Mettez en place une architecture pour objets connectés avec le standard oneM2M
Rest Wikipedia
Google image