Chapitre 4 : Conception

Introduction :

Le présent chapitre sera consacré à la présentation de l’étape de conception de notre système. La conception est un processus créatif qui permet de décrire la manière non ambigüe du fonctionnement désiré du système afin d’en faciliter la réalisation et la maintenance. A cet effet, nous présentons dans une première partie une conception générale de notre application ensuite dans la deuxième partie nous détaillons la conception de l’application et de la base de données utilisée.

1. Conception Générale :
   1. Conception architecturale :

L’architecture de la plateforme est « Microservices » qui utilise le model MVC pour chaque service

* + 1. L’architecture microservices :

1. Définition :

les microservices sont une technique de [développement logiciel](https://fr.wikipedia.org/wiki/D%C3%A9veloppement_logiciel) — une variante du [style architectural](https://fr.wikipedia.org/wiki/Patron_d%27architecture) de l'[architecture orientée services](https://fr.wikipedia.org/wiki/Architecture_orient%C3%A9e_services) (SOA) — qui structure une application comme un ensemble de services [faiblement couplés](https://fr.wikipedia.org/wiki/Couplage_(informatique)). Les microservices indépendants communiquent les uns avec les autres en utilisant des [API](https://fr.wikipedia.org/wiki/Interface_de_programmation) indépendantes du langage de programmation.

Des API [REST](https://fr.wikipedia.org/wiki/Representational_State_Transfer) sont souvent employées pour relier chaque microservice aux autres. Un avantage avancé est que lors d'un besoin critique de mise à jour d'une ressource, seul le microservice contenant cette ressource sera mis à jour, l'ensemble de l'application restant compatible avec la modification, contrairement à la totalité de l'application dans une architecture classique, par exemple une [architecture trois tiers](https://fr.wikipedia.org/wiki/Architecture_trois_tiers). Cependant, le coût de mise en place, en raison des compétences requises, est parfois plus élevé.

1. Les microservices :

Chaque micro-service peut être:

* Développé indépendamment
* Avoir une base de données indépendante
* testé indépendamment
* déployé indépendamment

(Technologie différente des autres)

Les applications de microservices permettent aux développeurs de décomposer plus facilement leur travail en petites équipes indépendantes et d’intégrer ce travail au fur et à mesure.

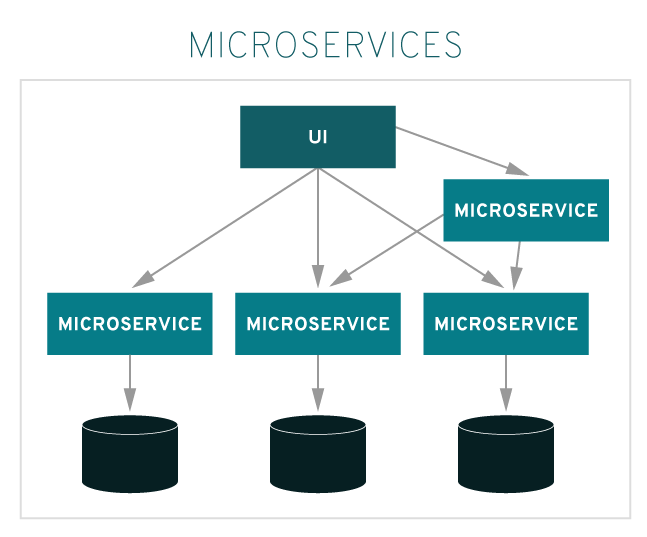


Figure 1: Architecture Microservices

2.le modèle MVC :

1. L’architecture MVC :
   1. Définition :

L'architecture *Modèle/Vue/Contrôleur* (MVC) est une façon d'organiser une interface graphique d'un programme. Elle consiste à distinguer trois entités distinctes qui sont, le *modèle*, la *vue* et le *contrôleur* ayant chacun un rôle précis dans l'interface.

modèle : données (accès et mise à jour)

vue : interface utilisateur (entrées et sorties)

contrôleur : gestion des événements et synchronisation

1. Model (modèle) :

Le modèle contient les données manipulées par le programme. Il assure la gestion de ces données et garantit leur intégrité. Dans le cas typique d'une base de données, c'est le modèle qui la contient.

Le modèle offre des méthodes pour mettre à jour ces données (insertion suppression, changement de valeur).

1. View (vue) :

La vue fait l'interface avec l'utilisateur. Sa première tâche est d'afficher les données qu'elle a récupérées auprès du modèle. Sa seconde tâche est de recevoir toutes les actions de l'utilisateur (clic de souris, sélection d’une entrée, boutons, …). Ces différents évènements sont envoyés au contrôleur.

1. Contrôleur :

Le contrôleur est chargé de la synchronisation du modèle et de la vue. Il reçoit tous les événements de l'utilisateur et enclenche les actions à effectuer. Si une action nécessite un changement des données, le contrôleur demande la modification des données au modèle et ensuite avertit la vue que les données ont changé pour que celle-ci se mette à jour. Certains évènements de l'utilisateur ne concernent pas les données mais la vue. Dans ce cas, le contrôleur demande à la vue de se modifier.

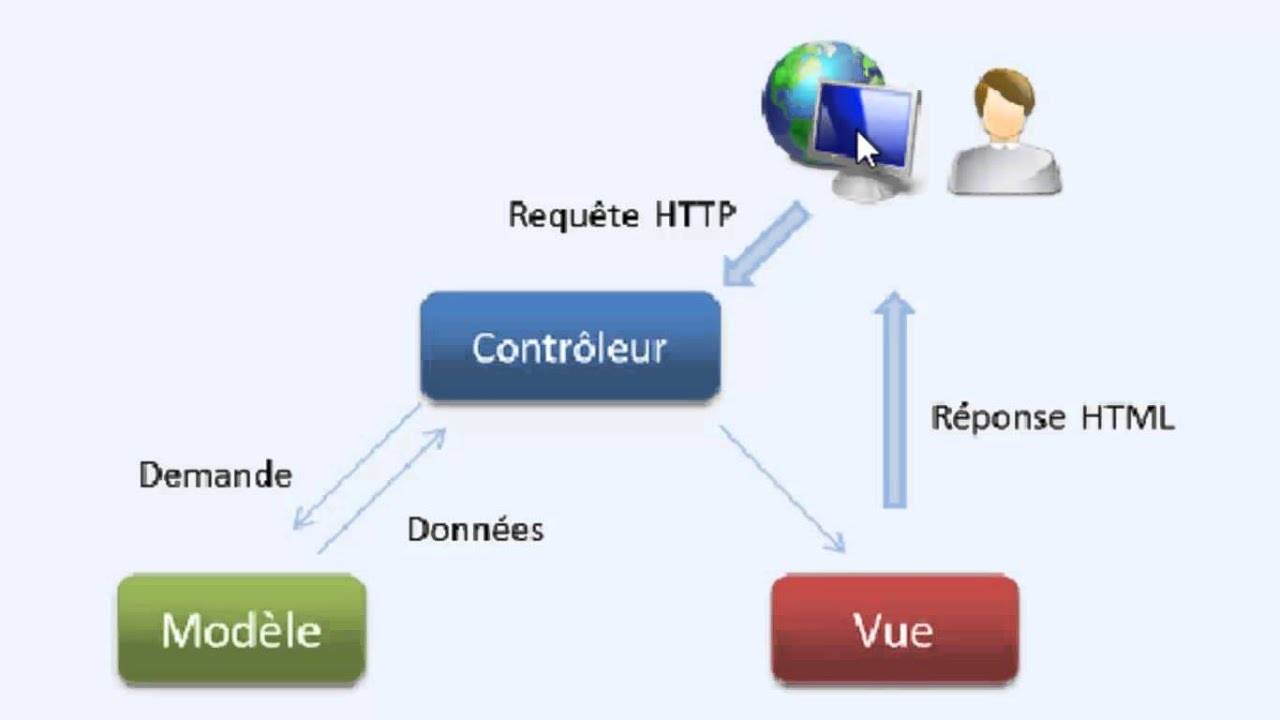


Figure 2: Architecture MVC

* + 1. Architecture retenue :

Contrairement à l'architecture monolithique traditionnelle, l'architecture de microservices est une approche dans laquelle l'application est structurée via plusieurs petits services autonomes qui fonctionnent ensemble.

Cette architecture a plusieurs avantages puisqu’elle :

* Rend possible un déploiement continu et automatisé.
* Permet aux développeurs de prendre des décisions appropriées et spécifiques au service.
* Chaque microservice possède sa propre base de données.
* Permet aux entreprises d'optimiser les ressources pour le développement et les applications.
* Optimise le dimensionnement et s'intègre facilement aux services tiers.

L’architecture microservices utilise le modèle MVC dans chaque microservice car le modèle MVC a une :

* Maintenance facile et ajout de fonctionnalités.
* Réduit le temps de développement du projet.
* Aide à créer des logiciels fiables avec des architectures testées.
* Facilite la réutilisation du code.
* Meilleure intégration de l'équipe .
* Réduction de la complexité du code.
  1. Conception détaillée :

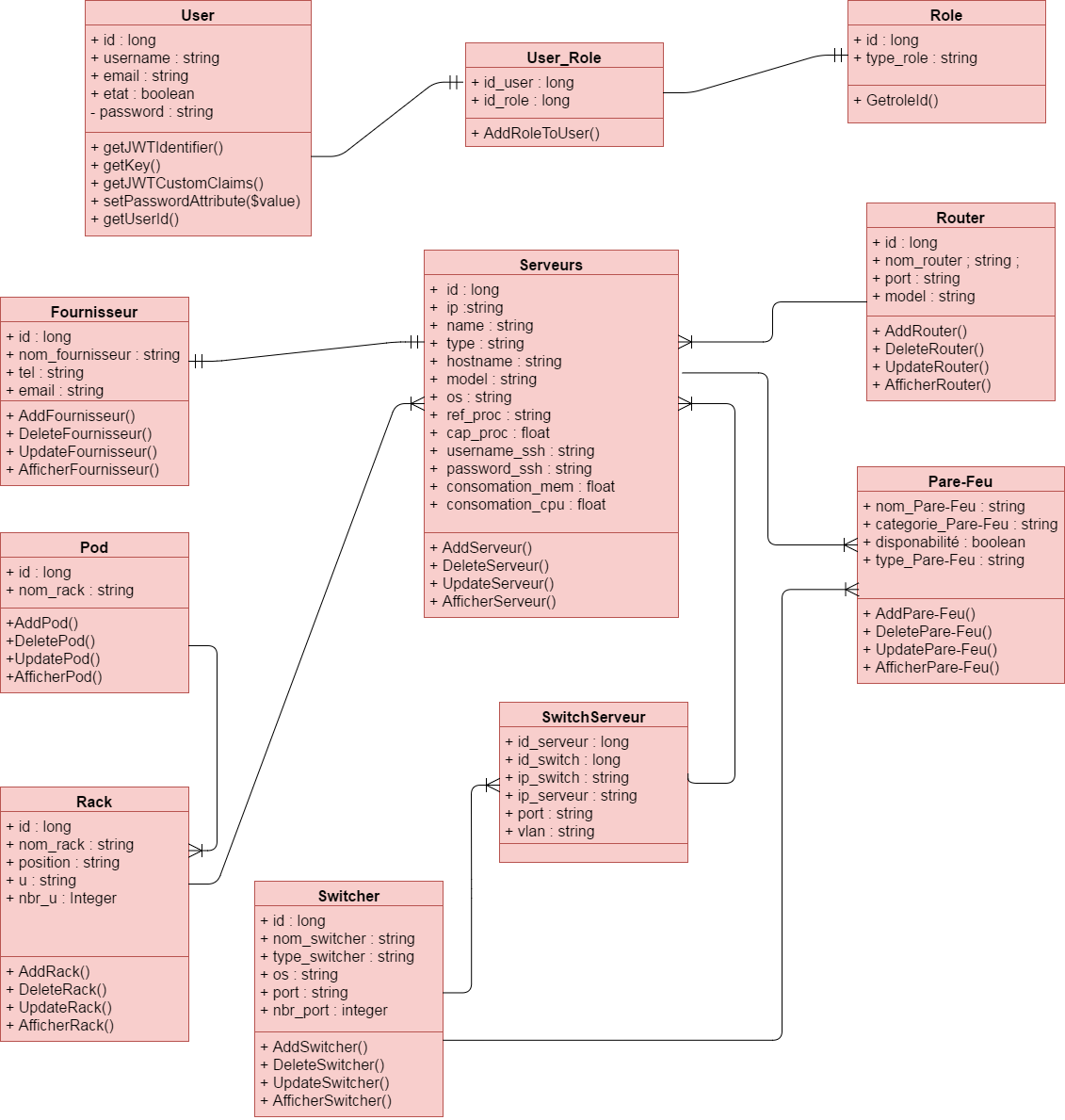
Cette section sera consacrée à la conception détaillée de notre application, nous débutons avec les diagrammes de classes

1.2.1 Diagramme de classes :

Un diagramme de classes fournit une vue globale d'un système en présentant ses classes, interfaces et collaborations, et les relations entre elles. Les diagrammes de classes sont statiques : ils affichent ce qui interagit mais pas ce qui se passe pendant l'interaction.

1.2.1.1 Diagramme de classe générale :

Le diagramme de classes générale représente la liaison entre les différents classe de l’application et les attributs de chaque classe avec ses opérations



1.2.2 : conception des bases de données :

Cette partie est consacrée à la conception des bases de données utilisées…

-SQL :

Pour La gestion du premier microservice (user) j’ai utilisé la base de données SQL avec le logiciel MySQL

SQL :

Le langage **SQL** (Structured Query Language) est un langage informatique utilisé pour exploiter des bases de données. Il permet de façon générale la définition, la manipulation et le contrôle de sécurité de données.

                Les instructions SQL couvrent 4 domaines :

* [Langage de définition de données](https://fr.wikipedia.org/wiki/Langage_de_d%C3%A9finition_de_donn%C3%A9es),
* [Langage de manipulation de données](https://fr.wikipedia.org/wiki/Langage_de_manipulation_de_donn%C3%A9es),
* [Langage de contrôle de données](https://fr.wikipedia.org/wiki/Langage_de_contr%C3%B4le_de_donn%C3%A9es),
* [Langage de contrôle des transactions](https://fr.wikipedia.org/wiki/Langage_de_contr%C3%B4le_des_transactions).

My SQL :



Figure 3: Logiciel de base de donnée MySQL

MySQL est un serveur de [bases de données relationnelles](https://fr.wikipedia.org/wiki/Base_de_donn%C3%A9es_relationnelle) [SQL](https://fr.wikipedia.org/wiki/Structured_Query_Language) développé dans un souci de performances élevées en lecture, ce qui signifie qu'il est davantage orienté vers le service de données déjà en place que vers celui de mises à jour fréquentes et fortement sécurisées. Il est [multi-thread](https://fr.wikipedia.org/wiki/Processus_l%C3%A9ger" \o "Processus léger) et multi-utilisateur.

-NoSQL :

NoSQL correspond à « **not only SQL** » et c’est en effet ce que ce modèle de base de données veut être : non pas une contrepartie, mais bien un enrichissement et **complément utile des bases de données SQL relationnelles traditionnelles**. Ce faisant, les bases de données NoSQL dépassent les limites des systèmes relationnels et exploitent un modèle de base de données alternatif. Cela ne veut toutefois pas dire qu’aucun système SQL n’est utilisé. Il existe de nombreuses variantes combinées au sein desquelles les deux solutions peuvent être utilisées et qui restent toutefois englobées sous l’étiquette NoSQL.

* Organisent les gros volumes de données au moyen de techniques flexibles,.

* Dans la mesure où les systèmes NoSQL exploitent un cluster matériel et un serveur Cloud, les **capacités**sont réparties de manière **uniforme** et la base de données fonctionne avec fluidité.
* Contrairement aux bases de données relationnelles qui perdent rapidement de la puissance à mesure que le volume de données augmente, les bases de données NoSQL s’imposent comme une solution performante, flexible et évolutive pour le stockage de grosses quantités de données.
* L’une des particularités des systèmes NoSQL est, en outre, leur capacité de **dimensionnement horizontal.** Les bases de données SQL relationnelles sont dimensionnées verticalement.
* Leur entière performance repose sur un seul et même serveur.

 Les solutions NoSQL répartissent généralement leurs données sur plusieurs serveurs. En cas d’augmentation du volume de données, de **nouveaux serveurs** peuvent alors facilement être **ajoutés**.

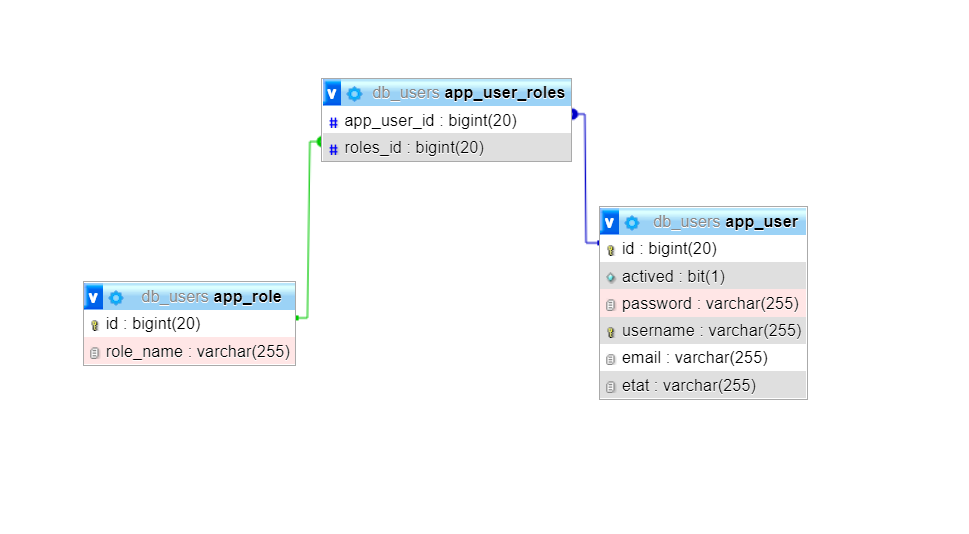
-MongoDB :



MongoDB est un [système de gestion de base de données](https://fr.wikipedia.org/wiki/Syst%C3%A8me_de_gestion_de_base_de_donn%C3%A9es) [orienté documents](https://fr.wikipedia.org/wiki/Base_de_donn%C3%A9es_orient%C3%A9e_documents), [répartissable sur un nombre quelconque d'ordinateurs](https://fr.wikipedia.org/wiki/Scalability) et ne nécessitant pas de schéma prédéfini de données. Il est écrit en [C++](https://fr.wikipedia.org/wiki/C%2B%2B). Le serveur et les outils sont distribués sous [licence SSPL](https://fr.wikipedia.org/wiki/Server_Side_Public_License), les pilotes sous [licence Apache](https://fr.wikipedia.org/wiki/Licence_Apache) et la documentation sous [licence Creative Commons](https://fr.wikipedia.org/wiki/Licence_Creative_Commons)[4](https://fr.wikipedia.org/wiki/MongoDB#cite_note-licensing-4). Il fait partie de la mouvance [NoSQL](https://fr.wikipedia.org/wiki/NoSQL).

1.2.2.1 : Base de donnée User :

Concepteur de la base de données SQL :



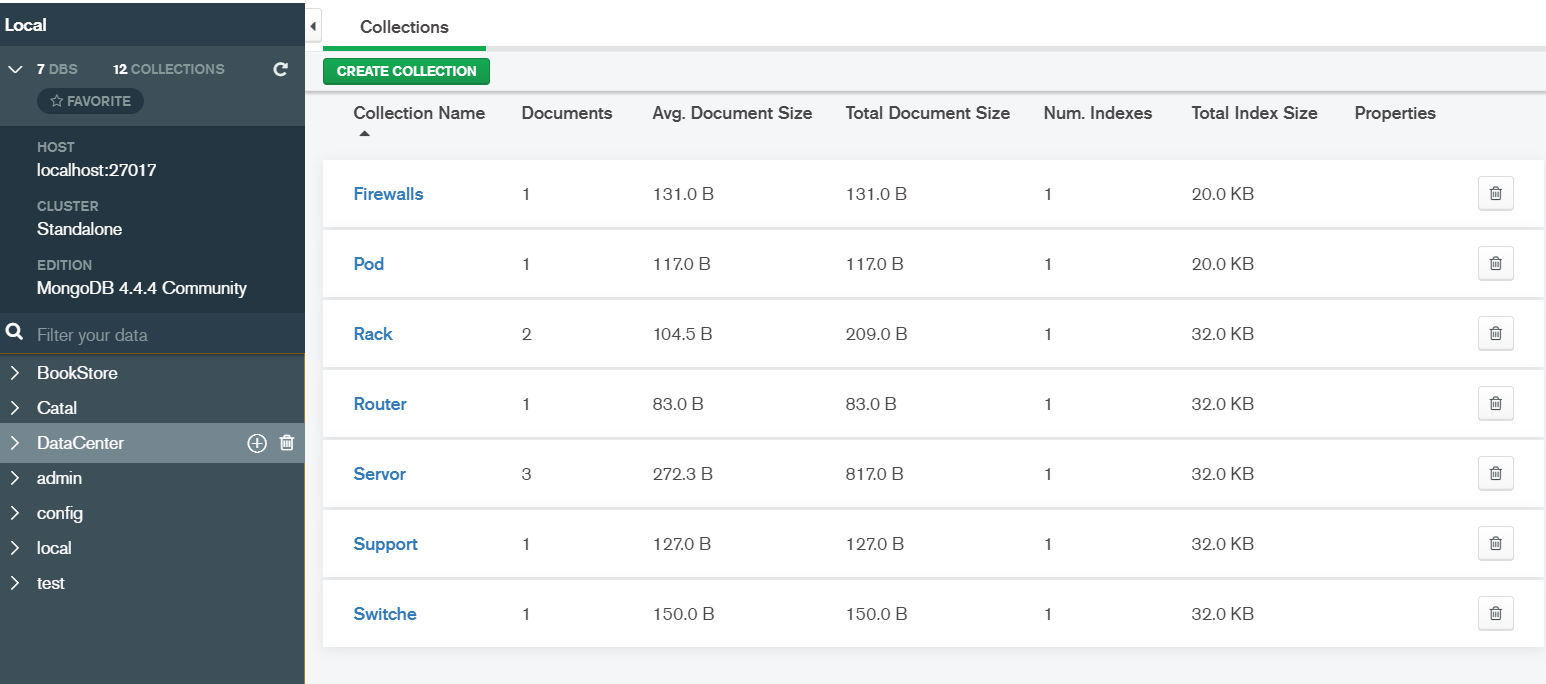
Vue globale des tables SQL :



1.2.2.2 : Base de données DATACENTER :

Pour La gestion du deuxième microservice (DATACENTER) j’ai utilisé la base de données NoSQL avec le logiciel MongoDB .

Concepteur base de données DATACENTER NoSQL :



## Conclusion

Tout au long de ce chapitre, nous avons présenté une conception détaillée de notre application. J’ai commencé par la description de l’architecture globale de l’application, puis j’ai présenté une vue statique de l’application moyennant les diagrammes de classes. Ensuite, j’ai cité les différentes méthodes de bases de données que j’ai choisies. Enfin nous avons pu construire les bases de données de l’application avec ses différentes tables et collections.

A présent, nous sommes capables d’entamer la partie réalisation.