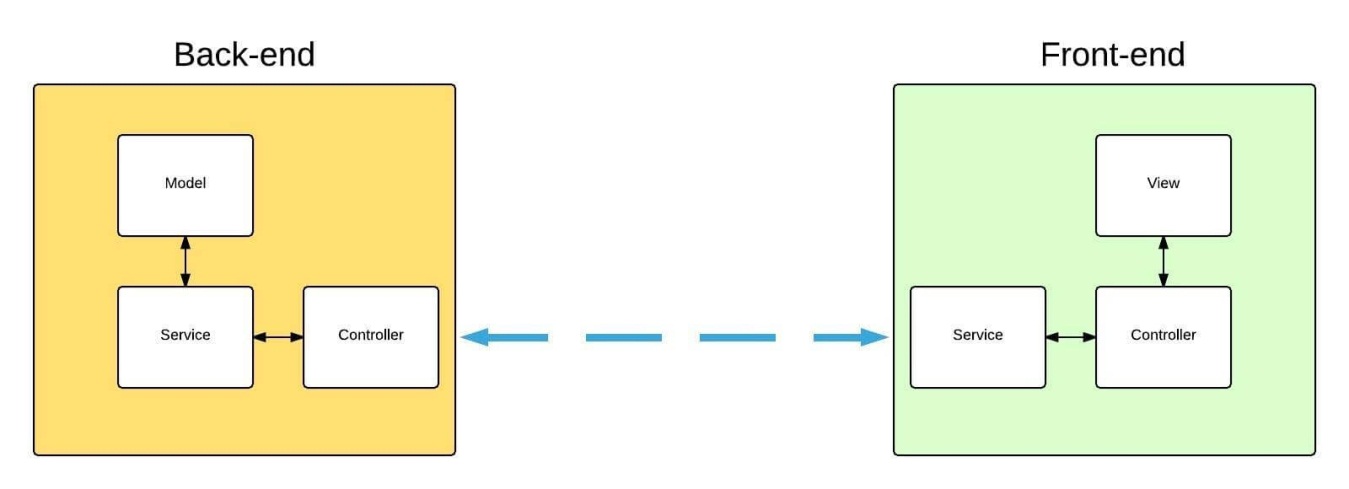
# Architecture Utilisée:



## Le Modèle

Le modèle détermine quelles données l'application doit contenir. Si l'état de ces données change, alors le modèle va généralement avertir la vue (donc l'affichage peut changer au besoin) et parfois le contrôleur (dans le cas où une logique différente est nécessaire pour contrôler la modification de la vue).

## La Vue

La vue détermine comment les données de l'application doivent être affichées.

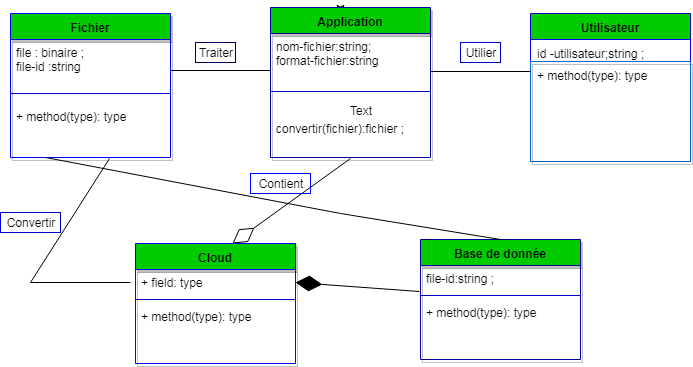
## Le contrôleur

Le contrôleur contient toute la logique de mise à jour du modèle et/ou la vue en réponse aux entrées (actions) de l'utilisateur sur l'application.

Dans notre application Le modèle contiendra le code permettant d’accéder aux SGBD. Toutes requêtes SQL ou équivalent doit donc être dans le modèle.  
La vue contiendra les pages qui seront affichées à l’utilisateur. C’est-à-dire la page au notre application également afficher ,Le contrôleur sert d’aiguillage. Il reçoit la demande de l’utilisateur, appelle le modèle si nécessaire et appelle la vue à transmettre au client.

Quand on réalise une application avec l’architecture MVC, il faut que toutes demandes de l’utilisateur passent par le contrôleur. Aucun affichage ne sera généré par le contrôleur ou le modèle. Dans notre application le contrôleur est le service CLOUD .

# Architecture générale d’une application de base de données :



# Stratégies de gestion des clients :

Deux stratégies se présentent à nous :

* La gestion des clients par threads,
* La gestion des clients par file d’attente,

## Threads :

Un thread, ou fil d’exécution est similaire à un processus car tous deux représentent l’exécution d’un ensemble d’instructions du langage machine d’un processeur. Du point de vue de l’utilisateur, ces exécutions semblent se dérouler en parallèle. Toutefois, la ou chaque processus possède sa propre mémoire virtuelle, les threads d’un même processus se partagent sa mémoire virtuelle.

La programmation utilisant des threads est toutefois plus rigoureuse que la programmation séquentielle, et l'accès à certaines ressources partagées doit être restreint par le programme lui-même, pour éviter que l'état d'un processus ne devienne temporairement incohérent, tandis qu'un autre thread va avoir besoin de consulter cette portion de l'état du processus. La complexité des programmes utilisant des threads est aussi nettement plus grande.

## File d’Attente :

En informatique, une file d'attente est une structure de données basée sur le principe du premier entré, premier sorti ou FIFO (« first in, first out ») ce qui veut dire que les premiers éléments ajoutés à la file seront les premiers à en être retirés.

Les queues servent à organiser le traitement séquentiel des blocs de données d'origines diverses.

Dans un logiciel informatique, l'avantage de cette politique d'ordonnancement réside dans sa relative simplicité, cependant elle pénalise les processus à temps bref d'exécution.

# Les technologies d’implémentation :

Pour notre projet, nous nous basons sur trois technologies principales :

* + MySQL : MySQL est un serveur de bases de données relationnelles SQL 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 multithread et multi-utilisateur.
  + NodeJS : Moteur d’exécution de code JavaScript pour la création d’applications. Comparable à la JVM permettant au langage Java de s’exécuter sur les machines compatibles. Il nous servira à créer un serveur pour exécuter certaines parties de notre application, codées en JavaScript.
  + VueJS : Framework JavaScript permettant la création d’interface graphiques ergonomiques et modernes pour les applications Web. Il utilise la technologie NodeJS pour s’exécuter.