**Architecture REST**

**En quoi consiste l’architecture REST ?**

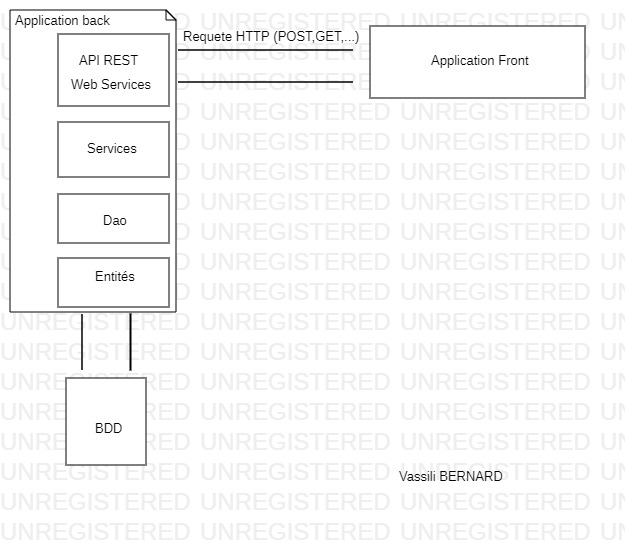
# L’architecture REST (Representational state transfer) est un type d’architecture en web services. Ce type d’architecture a pour objectif de séparer le partie front, de la partie back. La partie Front va venir communiquer sur une API back permettant le traitement de la donnée.

**Comment ça fonctionne ?**

Le fonctionnement d’une architecture REST se fait par l’interaction entre le Front et le Back par une API. Le front va envoyer ou récupérer de la données en requêtant par des requêtes HTTP (GET,POST,PUT,DELETE,…) sur l’API. Le back lui va traiter la donnée et communiquer avec la base de données.

L’objectif est réellement de séparer la partie Front du Back afin d’avoir deux applications. Ce genre d’architecture permettrais notamment de séparer les différentes responsabilités. Notre back étant indépendant et hors contexte du front, nous pouvons créer des web services qui répondront à nos différentes demandes et responsabilités.

L’illustration ci-dessous est un exemple d’architecture Rest que l’on peut retrouver.



**Quels sont les avantages ?**

L’avantage d’une telle architecture logicielle est déjà que ce genre d’architecture est en couche et donc plus ergonomique pour le développeur et les différents aspect (communication avec la base de données, partie métier, web serives) sont bien séparé et défini.

De plus une API REST n’est pas compliquée à mettre en œuvre, beaucoup de framework embarque déjà les composants nécessaires à celui-ci. Prenez Spring et Java, en seulement quelque annotation vos web services sont créer, ils ne restent plus qu’à lancer votre serveur et vous pouvez requêter sur vos web services. La montée en compétences pour une équipe de développeur sera alors plus facile et rapide.

**Quels sont les inconvénients ?**

Les inconvénients de cette architecture sont le fait que lorsque on a une modification partie front comme le requêtage sur l’api avec une nouvelle URL, il ne faut pas oublier de l’implémenter partie back aussi. Donc il faut qu’il y est une cohérence entre le Back et le Front.

De plus lorsqu'il y a une monté de version il faut vérifier et faire attention à cette cohérence.

**Conclusion**

Dans le cadre d’une architecture REST, j’utiliserais Java 8 et Spring ainsi que le framework Hibernate pour la partie Back. Partie Front j’utiliserais Angular. Mes choix se portent du fait que ce sont des technologies que je connais et qui sont adapté pour une montée en compétence rapidement. Java bien que parfois critiqué, avec Spring permet de mettre les web services rapidement et simplement. Quand à Angular sont efficacité suffit à requêter sur l’API REST. De plus ces deux technologies possèdent une communauté étendue et complète.

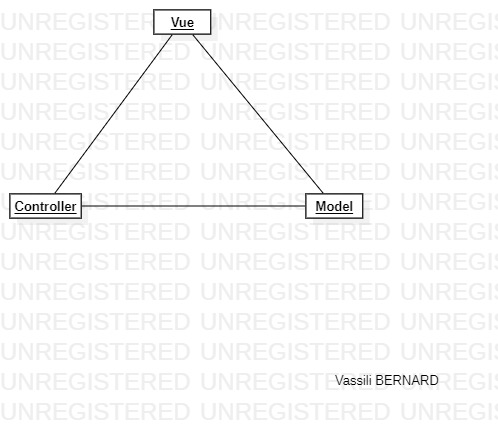
**Architecture MVC**

**En quoi consiste une architecture MVC ?**

L’architecture MVC (Modèle,Vue,Controller) est une architecture qui consiste à séparer notre application en modules. On se retrouve donc avec trois modules qui auront chacun leur rôles :

* Le modèle : permet la communication avec la base de données
* La vue : il s’agit ici plus du front
* Controller : “le maître d’orchestre” qui va charger de rediriger l’information va les différents traitements et peut être la renvoyer.

**Comment fonctionne une architecture MVC ?**



Les actions de l’utilisateur dans la Vue seront envoyées au Controller. Le Controller se chargera alors de transmettre la donnée au Model. Le model communiquera avec la base de données et renverra les données demandées s'il faut.

La fonctionnement et l’architecture est plutôt basique et simple de fonctionnement.

**Quels sont les avantages ?**

En termes d’architecture la maintenance et l’évolutivité est plutôt simple. Le mise en place d’une telle architecture n’est pas bien compliqué. De plus on gagne en souplesse sur cette architecture au vu du nombre de couches.

**Quels sont les inconvénients ?**

L’architecture possède peu de couche et on se retrouve avec de nombreux fichiers par modules posant la confusion dans le développement. Pour les plus petites projets le MVC n’est pas adapté car on se retrouve avec de nombreux fichiers pour un projet simple.

**Conclusion**

Afin de mettre en place ce type d’architecture, je ferais recours évidement à des Framework dédié à cet effet. On peut faire recours à des développeur Full stack, vue que le front se trouve dans la même “application” contrairement à une architecture en REST. Ce type d’architecture peut être simple en maintenabilité mais attention au développement des différents modules car bien souvent dans ce genre d’architecture le développeur à tendance à mettre le traitement de la donnée dans les modules qui ne sont pas prévu à cet effet. Il faut bien définir les normes de nommage et le métier avant de se lancer dans cette architecture (notamment on peut faire recours à la méthode DDD)