BERRADA Amine - CORNILLON Jean - GEHIN Sylvain - PERREYON Thomas

ATELIER II

Tableau comparatif de différentes architectures logicielles

	REST	MVC	SOA	Micro services
Modularité	Potentiellement réutilisable	Potentiellement réutilisable	Grande modularité, on peut facilement remplacer un service par un autre, et chacun d'eux est réutilisable	Services réutilisables par essence même et indépendant des autres
Coût	Dans la moyenne des coûts classiques	Dans la moyenne des coûts classiques	Coût élevé à la fois financièrement et humainement pour une entreprise, car nécessité de former une équipe d'experts pour la conception ainsi que des équipes de développement	Coût de mise en place élevé du fait des compétences requises
Mise à jour	Possibilité de mettre à jour un composant sans pour autant affecter le reste du système, même si celui-ci est actif	Difficile car les éléments sont interdépendants	Normale	Seul le service contenant la ressource est mis à jour si nécessaire, le reste de l'application restant compatible avec la modification
Développement	Relativement simple, mais impose un principe de développement à suivre	Relativement compliquée mais développement en parallèle envisageable	Relativement simple car application composée de services simples.	Simple car travail en parallèle possible du fait de l'indépendance de chacun des services

Conception	Complexe car doit respecter les contraintes inhérentes à l'architecture	Complexe car l'architecture impose des contraintes strictes et relativement complexes.	Nécessite une réflexion quant à la responsabilité de chacun des services. De plus, architecture utilisée pour développement d'applications à long terme donc conception très importante	Nécessite une réflexion quant à la responsabilité de chacun des services. De plus, apparition de nouvelles contraintes et difficultés pour permettre un faible couplage des microservices.
Interdépendance des éléments	Le code côté client est totalement indépendant du code côté serveur	Le modèle, la vue et le contrôleur ont chacun besoin les uns des autres pour assurer le fonctionnement du service	Même si les services sont indépendants, ils partagent tous les mêmes unités de stockage de données et restent donc un minimum interdépendants	Chaque microservice est indépendant
Déploiement	Déploiement relativement simple	Déploiement compliqué car il doit être réalisé sur chacun des points d'utilisation	Déploiement relativement simple malgré le fait que les services sont un minimum interdépendants	Déploiement rapide de chacun des microservices indépendamment des autres. Cependant, plus complexe lorsqu'il s'agit de déployer l'ensemble de l'application.

Sources :

http://adslbox.free.fr/rapports/rapport-gl-service-oriented-architecture.pdf

https://fr.wikipedia.org/wiki/Microservices

https://en.wikipedia.org/wiki/Representational_state_transfer

https://www.bluesoft-group.com/soa-microservices-deux-architectures-si/

https://web2.cegepat.qc.ca/~claudeboutet/index.php/2016/04/14/le-pattern-architectural-mvc/

https://docs.microsoft.com/fr-fr/dotnet/architecture/microservices/multi-container-microservice-net-applications/microservice-application-design

https://www.mulesoft.com/fr/resources/api/what-are-microservices

https://www.vigicorp.fr/blog/architecture-microservices/

https://www.talend.com/fr/resources/microservices-vs-soa/

https://www.redhat.com/fr/topics/integration/whats-the-difference-between-soap-rest