**Choix architectural**

**L’architecture monolithique**

Les applications monolithiques ont été conçues pour traiter de multiples tâches connexes. Il s'agit généralement d'applications complexes qui englobent plusieurs fonctions étroitement couplées. Parmi les avantages de l’architecture monolithique on trouve la simplicité de développement, de déploiement, de maintenance et des tests en raison de leur structure centralisée. Par contre les monolithes peuvent être difficiles à faire évoluer pour répondre à une demande croissante, ainsi que le déploiement d’une nouvelle fonctionnalité nécessite souvent le déploiement de l’ensemble du programme monolithique, ce qui peut être risqué et lent. Finalement, un monolithe peut être difficile de mettre à jour des composants spécifiques sans perturber l’ensemble du système. Cela peut rendre la maintenance à long terme coûteuse et compliquée.

**L’architecture microservices**

Les microservices sont une approche architecturale dans laquelle une application est décomposée en un ensemble de petits services autonomes et indépendants, chacun gérant une fonctionnalité spécifique. Parmi les avantages de l’architecture microservices on trouve l’évolutivité puisque les microservices permettent de faire évoluer des parties spécifiques de l’application sans toucher au reste, ce qui facilite l’évolutivité, le déploiement indépendants car chaque service peut être déployé indépendamment, ce qui accélère les mises à jour et réduit les risques. De plus une meilleure gestion des équipes puisqu’ils peuvent être organisées autour des services, ce qui facilite la gestion et la collaboration. Par contre l’architecture microservices présente une complexité de gestion d’un grand nombre de microservices en termes de surveillance, de déploiement et de mise à l’échelle alors la mise en place initiale d’une architecture de microservices peut être plus coûteuse en termes de temps et de ressources.

**La refactorisation d’un monolithe en microservices**

La refonte d'une application monolithique en une architecture de microservices est un processus complexe mais souvent nécessaire pour répondre aux besoins évolutifs et aux demandes changeantes du monde de la technologie. Cette migration se fait par découper notre application en plusieurs microservices et concevoir chaque microservice autour d’un domaine métier spécifique passant par trois étapes l’identifications des domaines, la création des microservices et l’établissement de la communication entres ces microservices. La refonte d’un monolithe en une architecture de microservices peut apporter de nombreux avantages en termes d’évolutivité, de déploiement et de gestion des équipes.

**Choix :**

L’architecture monolithique et l’architecture en microservices sont totalement opposées même en termes d’avantages et d’inconvénients ce qui rend la refactorisation d’un monolithe en microservice très intéressante pour se bénéficier à la fois des avantages d’un monolithe au début du projet et les avantages des microservices à la fin et à long terme.

|  |  |  |
| --- | --- | --- |
|  | Architecture monolithique | Architecture en microservices |
| Implémentation | * Simple | * Un peu complexe |
| Déploiement | * Le déploiement initial Simple. * Le déploiement d’une nouvelle fonctionnalité nécessite souvent le déploiement de l’ensemble du programme monolithique. | * Complexité de déploiement initiale de chaque microservices indépendamment des autres. * Le déploiement des nouvelles fonctionnalités est plus simple car on ne va pas toucher à un système complexe. |
| Mise à jour | * Complexe car ça nécessite de mettre à jours de l’ensemble du programme monolithique. | * Facile et rapide car la mise à jour est réalisée individuellement pour chaque microservices. |
| Évolutivité | * Il est difficile de maintenir et de gérer les nouveaux changements car tout le système doit être redéployé | * Chaque élément peut être mis à l'échelle indépendamment sans temps d'arrêt. |
| Cout | * Réduit, ne nécessite pas beaucoup de ressources car tout est regroupés dans un même monolithe0. | * Couteuse en termes de temps et de ressources. |
| Maintenance | * Maintenance à long terme coûteuse et compliquée. | * Maintenance indépendante pour chaque microservice. |
| Agilité | * Pas flexible et impossible d'adopter de nouvelles technologies, langages ou Framework. | * On peut intégrer des nouvelles technologies pour répondre aux différents objectifs. |
| Résilience | * Un beug ou un problème peut affecter l’ensemble de système. | * Une panne dans un microservice n’affecte pas les autres microservices. |
| Structure | * Centralisée | * Constituée par des microservice, chacun a son propre domaine métier et répond à ces fonctionnalités. |

Source :

<https://www.openlegacy.com/blog/monolithic-application> : **tableau de comparaison**

https://loud-technology.com/blog/refactoriser-un-monolithe-en-microservices/#:~:text=Pourquoi%20refactoriser%20une%20application%20monolithique%20en%20microservices%3F&text=La%20refonte%20d%27une%20application,du%20monde%20de%20la%20technologie. : **Pourquoi refactoriser une application monolithique en microservices?**