

Le monolithe est mort, vive le monolithe !

Sébastien Ballangé
Confoo 2024

Qui suis-je?

Développeur Staff chez Connect&GO

15+ années d'expérience en tant que développeur,
principalement PHP et Symfony



CONNECT&GO



Programme

- ➔ Mise en contexte rapide
- ➔ Du négatif
- ➔ Du positif
- ➔ Quelques pistes

Quelle architecture choisir
pour commencer un
nouveau projet ?

Posez-vous ces questions:

- Quelle est la taille de l'équipe?
 - Quand le projet doit-il être livré?
 - Quel budget?
 - Combien d'utilisateurs sont attendus au lancement?
 - Quelle expérience l'équipe a-t-elle?
-

-

Un monolithe !

Sans aucune hésitation,
dans la quasi-totalité des cas

Si vous ne savez pas exactement **pourquoi** vous envisagez une architecture en micro-services, **ne le faites pas.**



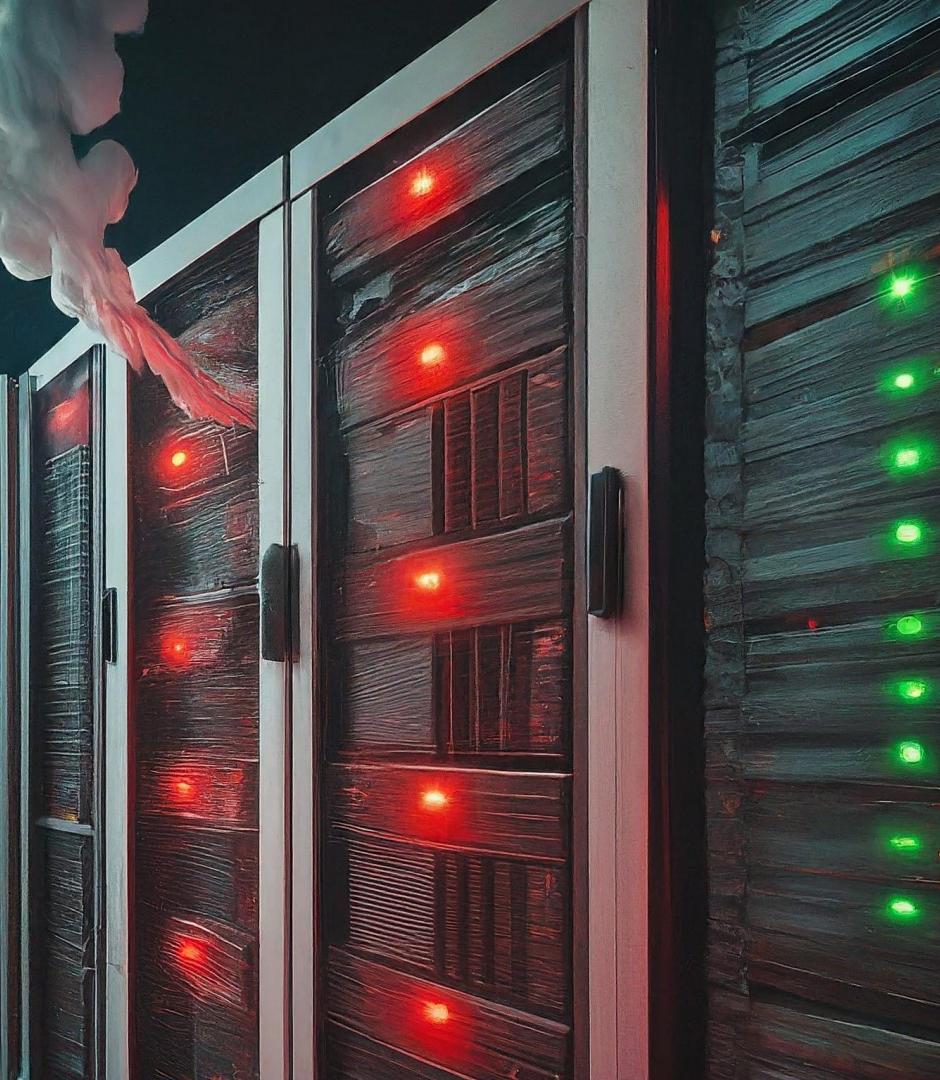


Les inconvénients classiques

Couplage étroit et complexité

- interdépendance entre les composants
- longs cycles de développement





Adaptation aux changements de trafic

- Impossible d'augmenter les ressources pour seulement certaines fonctionnalités
- Un ralentissement causé par une section peut se propager au reste

Obstacles à l'adoption de nouvelles technologies

- une seule stack donc difficile d'introduire des choses nouvelles pour des fonctionnalités spécifiques
- innovation plus difficile





Courbe d'apprentissage et tests

- volume élevé d'informations à absorber pour un nouveau membre de l'équipe
- large domaine à tester

Collaboration entre équipes

- Plus d'équipes, plus de dev, plus de problèmes de communication
- Risque accru de briser le travail des autres





Comprendre les avantages



Développement et déploiement simplifiés

- Base de code unique.
- Processus de développement plus simple.
- Il n'est pas nécessaire de gérer des communications entre services plus complexes.
- Le déploiement consiste à empaqueter et déployer une seule unité.

Cohérence améliorée entre les fonctionnalités

- Dans une architecture monolithique, tous les composants partagent le même modèle de données et la même base de données.
- Ceci assure une forte cohérence des données à travers les différentes fonctionnalités de l'application.
- Les transactions impliquant plusieurs fonctionnalités sont plus faciles à mettre en œuvre.





Débogage et tests facilités

- Les tests de bout en bout sont plus simples dans un monolithe en raison de sa nature unifiée.
- Les développeurs peuvent tracer les problèmes à travers l'ensemble du système sans barrières de communication.
- Les outils et processus adaptés aux architectures monolithiques ont tendance à être bien établis.

Performance

- Dans certains scénarios, les monolithes peuvent offrir des avantages en termes de performance.
- Beaucoup moins d'accès au réseau



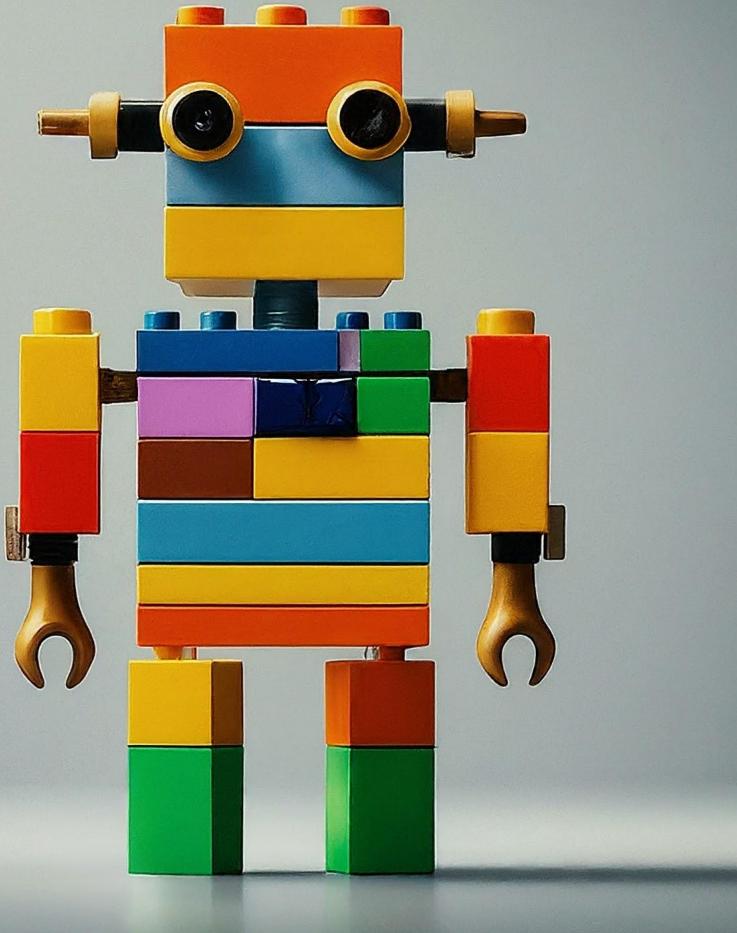


Cas d'utilisation idéaux pour les architectures monolithiques

- Applications à petite échelle : Pour les projets aux limites bien définies et limités en complexité.
- Applications ayant de forts besoins en cohérence des données : Applications où les transactions enjambant plusieurs fonctionnalités sont cruciales.
- Prototypage et MVP : Pour mettre rapidement des produits viables minimum (MVP) sur le marché, en privilégiant la vitesse à la flexibilité.

Oui mais!

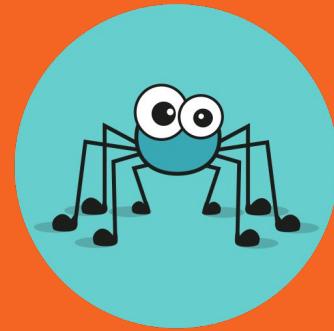




Dans micro-services, il y a services

- Regrouper la logique en services spécialisés et compartimentés
- Plus faciles à maintenir et tester

Raisonner en API



API PLATFORM



Et la performance dans
tout ça ?

Un monolithe, ou une citadelle?

- Un monolithe au centre et des avant-postes



<https://signalvnoise.com svn3/the-majestic-monolith-can-become-the-citadel/>



Conclusion

- **Ne pas choisir une architecture aveuglément**

Soyez pragmatiques et analysez **votre** situation

- **Votre pire ennemi: la complexité**

Pour citer Grug: “complexity very, very bad”

<https://grugbrain.dev/>



Merci !

Retrouvez toutes les présentations sur
<https://github.com/confooca/2024>

ConFoo.ca
DEVELOPER CONFERENCE

CONNECT&GO