Architectures Logiciels modernes : *Concevoir des logiciels agiles, sécurisés et évolutifs*

Régis ATEMENGUE

Software Engineer | Technical Instructor.

@regis_ate | www.regisatemengue.com

Agenda

- Scenarios de motivation
- Architecture Logicielle
- Styles Architecturaux
- Microservices
- Avantages et Challenges
- Cas d'etude Cameroun.

Scenario 01: Reseau Social

Scenario 01 : Plateforme de médias sociaux

Contexte :

Vous faites partie d'une équipe de développement d'une plateforme de réseau social populair en pleine croissance en tant qu'**architecte logiciel / developpeur.** L'architecture actuelle, principalement basée sur un **modèle traditionnel** de **client-serveur** a besoin d'aide pour gérer le ***nombre croissant d'utilisateurs***, de **publications et d'interactions**. La plateforme doit évoluer pour garantir son évolutivité et des mises à jour en **temps réel.** Après cette évolution, nous devrons gérer la complexité, maintenir la **propreté du code** et offrir une expérience **fluide** aux utilisateurs.

Scenario : Plateforme de médias sociaux

La plateforme de réseau social comprend des fonctionnalités telles que les **profils utilisateur**, les **publications**, les **mentions** « J'aime », les **commentaires**, les **demandes d'ajout à la liste d'amis**, les **flux d'actualités**, le **partage** et les **notifications**.

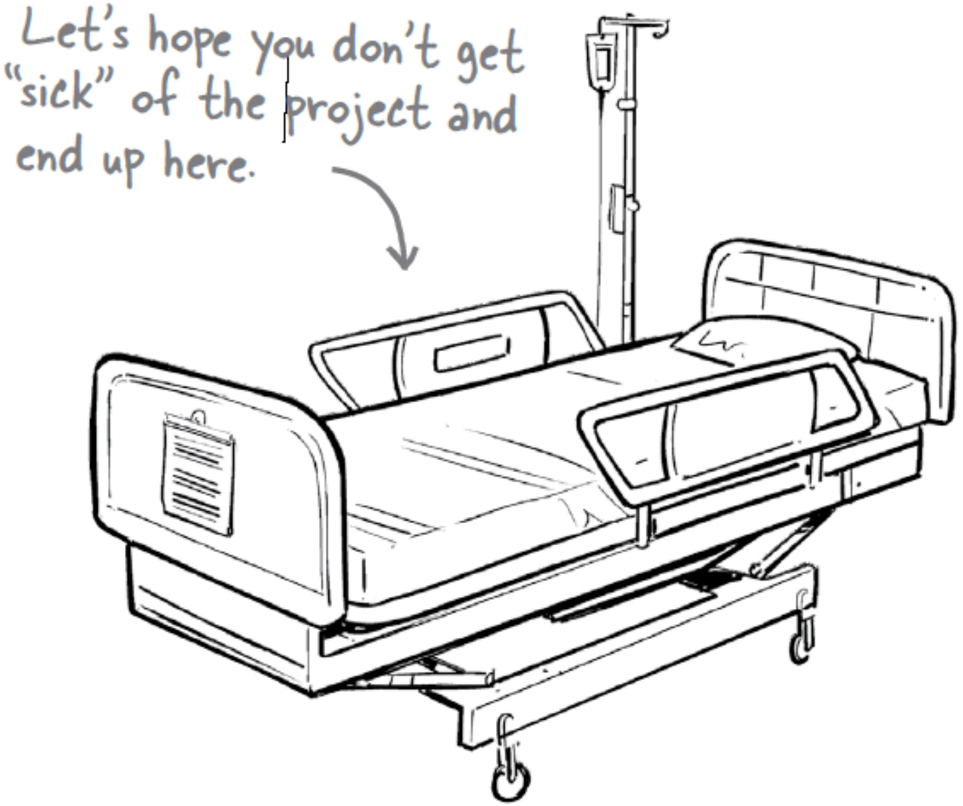
Les utilisateurs ont signalé des retards occasionnels dans la réception des mises à jour et des perturbations pendant les heures de pointe, ainsi que des retards dans l'affichage des mises à jour et des notifications en raison de l'architecture **monolithique**

Scenario 02: StayHealthy, Inc.

Scenario 02 : StayHealthy, Inc.

StayHealthy, Inc. est une entreprise spécialisée dans les systèmes de surveillance médicale des patients hospitalisés. Grâce à ses systèmes, les médecins et les infirmières peuvent surveiller la **fréquence cardiaque**, le taux d'oxygène, **la température corporelle**, le **taux de glycémie** et bien d'autres paramètres des patients, et même déterminer si ceux-ci sont **endormis** ou **éveillés**. En cas de problème, un médecin ou une infirmière est immédiatement averti.

Les progrès récents de la médecine ont fait naître de nouveaux besoins en matière de surveillance médicale. StayHealthy prévoit donc de tirer parti des nouvelles technologies pour remplacer son logiciel actuel de surveillance médicale des patients par un nouveau système appelé **MonitorMe**.



Besoins : MonitoMe, Inc.

Fonction principale : Surveiller les signes vitaux des patients et alerter les professionnels de santé en cas de dépassement de seuils prédéfinis.

Fonctionnalités détaillées :

1. **Collecte et Affichage :** Lit les données de 8 signes vitaux (fréquence cardiaque, pression artérielle, etc.) et les affiche sur un écran de monitoring unique.
2. **Alerte Personnalisable :** Notifie un professionnel de santé (via téléphone ou poste central) en cas d'anomalie détectée.
3. **Historique des Données :** Enregistre et permet de consulter l'historique de toutes les mesures des 5 dernières minutes.
4. **Multi-patients :** Capable de surveiller jusqu'à 500 patients simultanément.
5. **Architecture Redondante :** Le système continue de fonctionner même si une source de données tombe en panne.
6. **Déploiement par Site :** Chaque hôpital (site physique) dispose de sa propre instance indépendante du système et de ses données.

Signes vitaux surveillés : Fréquence cardiaque, pression artérielle, taux d'oxygène, glycémie, fréquence respiratoire, électrocardiogramme (ECG), température corporelle et état de sommeil.

Questions ?

Quel est le meilleur choix : un programme unique qui fait tout, ou plusieurs petits programmes qui communiquent entre eux ?

Est-il préférable de tout regrouper pour plus de simplicité, ou de séparer les éléments pour éviter qu'une seule panne ne paralyse l'ensemble du système ?

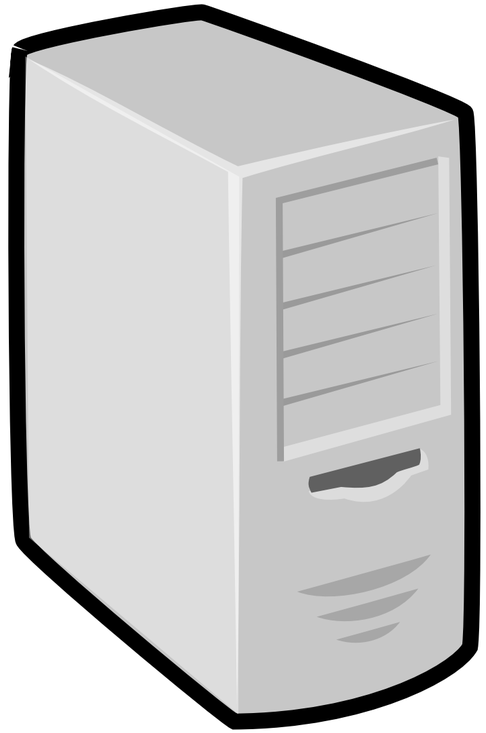
Comment intégrer les exigences de sécurité (confidentialité des données des utilisateurs, audit des accès etc....) dans des systèmes avec plusieurs fonctionnalites ?

Évolution de la Conception Logicielle : du Passé au Présent

Le passé

Le Passé : Rigidité et Contraintes Manuelles

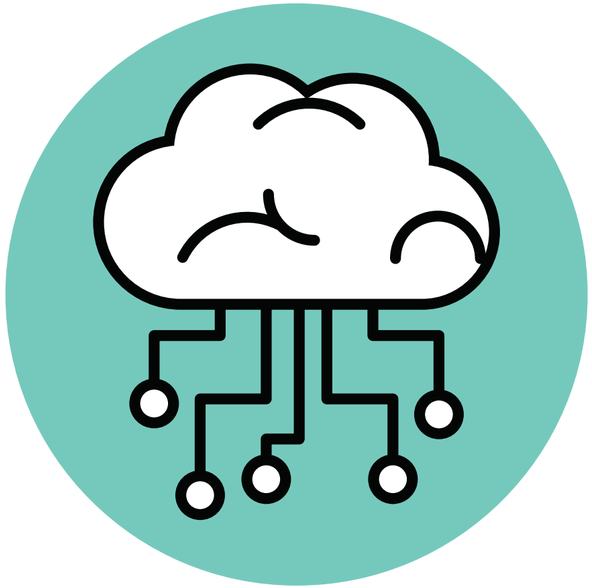
- **Processus rigide :** Domination du modèle **Waterfall** (Cascade), une approche séquentielle et linéaire.
- **Limites technologiques :** Dépendance à des ressources physiques encombrantes (ordinateurs centraux) et à une puissance de calcul très faible.
- **Tâches manuelles :** Méthodes de programmation laborieuses, comme l'utilisation de **cartes perforées**.
- **Résultat :** Des cycles de développement extrêmement longs et peu flexibles.



Le Présent/Futur

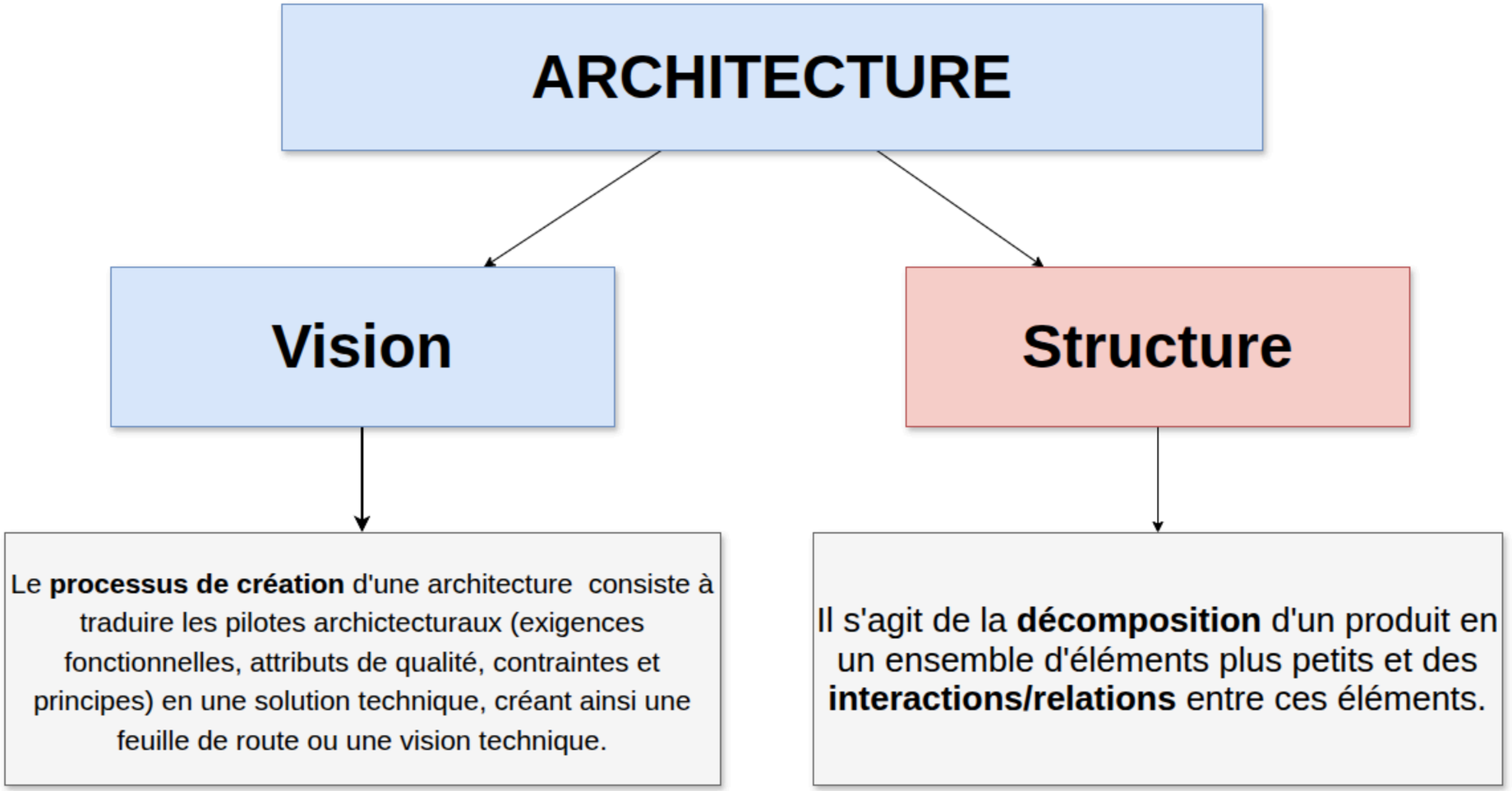
Le Présent/Futur : Agilité et Automatisation Intelligente

- **Abondance des ressources** : Accès facile à une puissance de calcul quasi-illimitée et évolutive grâce au cloud computing.
- **Automatisation intelligente :** Intégration d'outils basés sur l'IA (ex: analyse automatisée du code) pour optimiser les workflows.
- **Efficacité et agilité :** Adoption de méthodes permettant des itérations rapides et une grande réactivité.
- **Résultat :** Amélioration significative de la qualité des logiciels et de la vitesse de livraison.

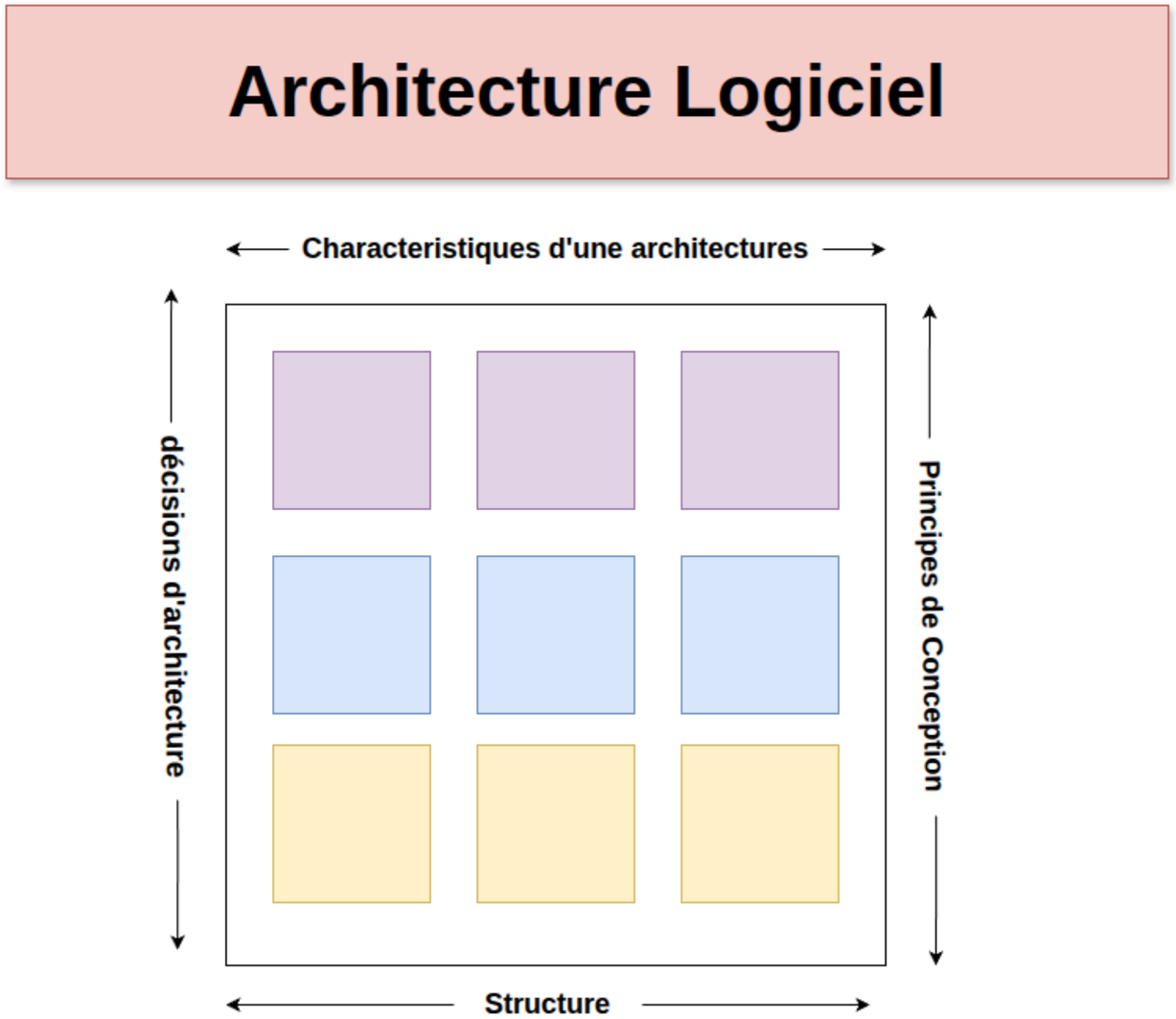


Qu'est-ce que l'architecture logicielle ?

Architecture



Architecture logicielle

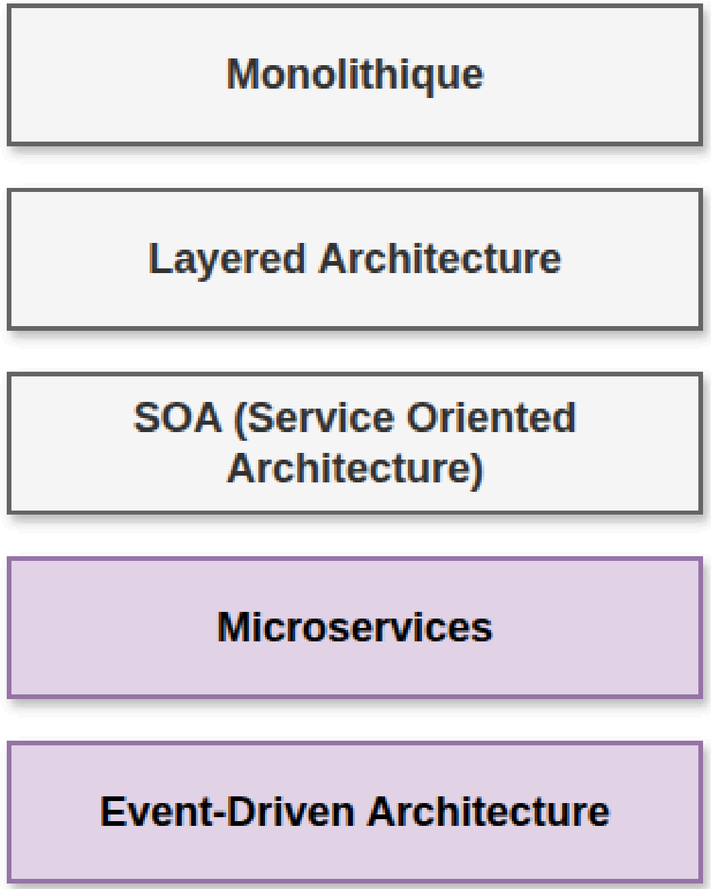


Architecture logicielle

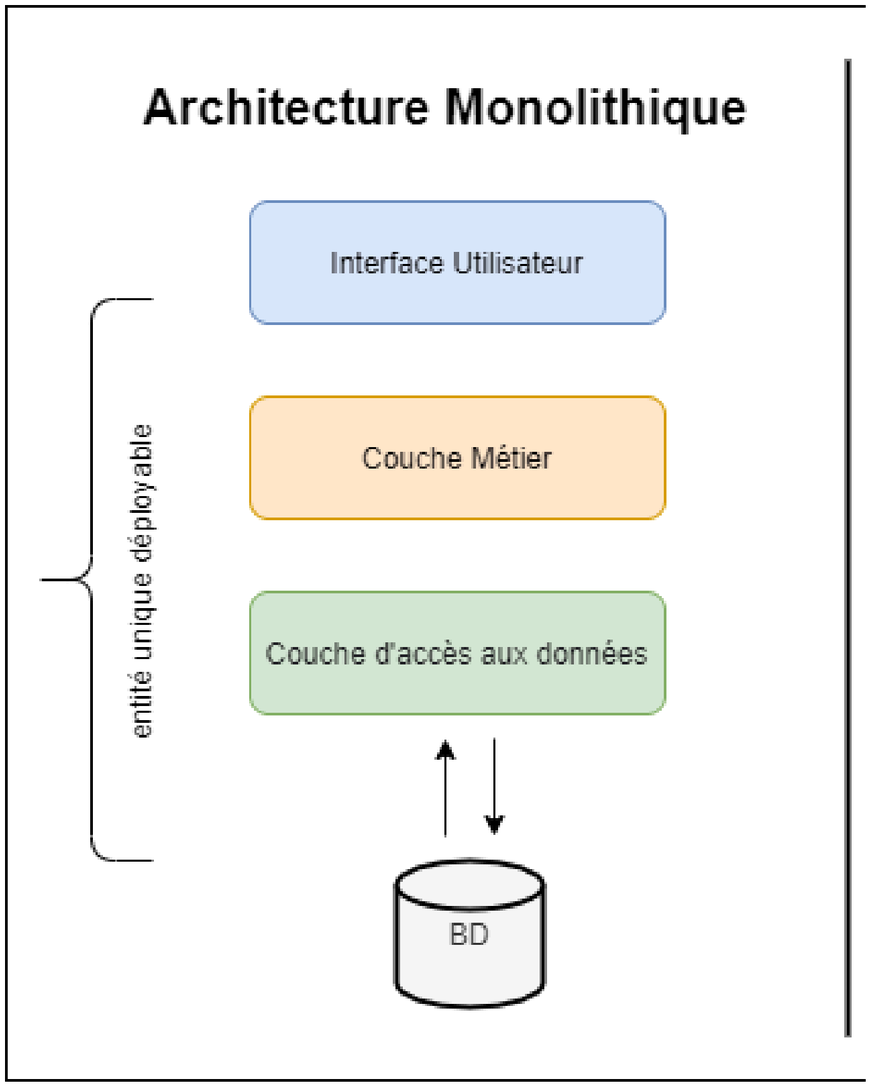
En d'autres termes, cela englobe tout ce qui touche à la **conception d'un système logiciel,** depuis la **structure** du **code** et la compréhension du fonctionnement global du système logiciel à un niveau élevé jusqu'à la manière dont ce système logiciel est déployé sur l'infrastructure.

Styles Architectaux / Structure

Styles Architectaux / Structure



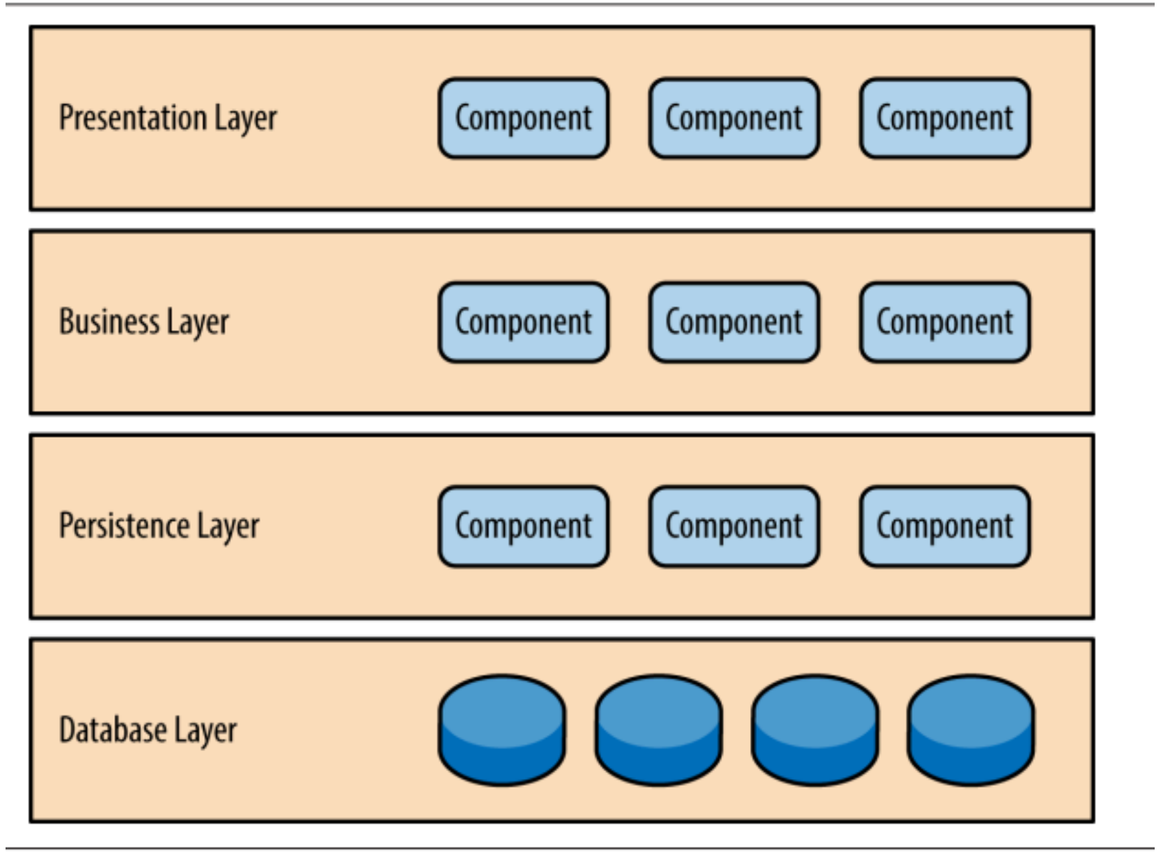
Monolithique



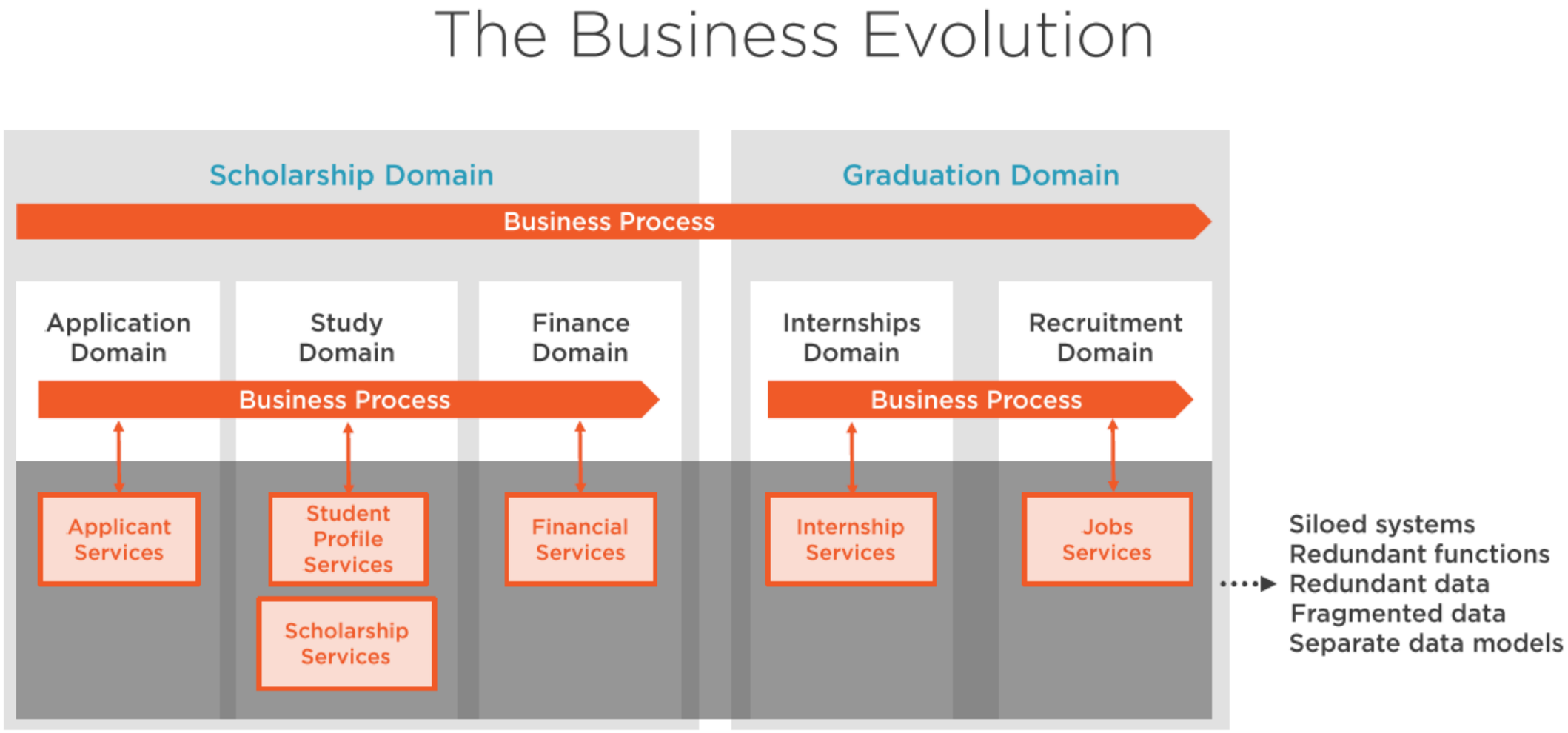
Un monolithe est une application complète qui s'exécute dans un seul processus.

La mise à jour du code d'un monolithe est une opération risquée. C'est tout ou rien. Lorsque vous appliquez une modification de code qui perturbe le monolithe, l'ensemble de l'application cesse de fonctionner, vos clients se retrouvent dans l'embarras et votre entreprise perd de l'argent.

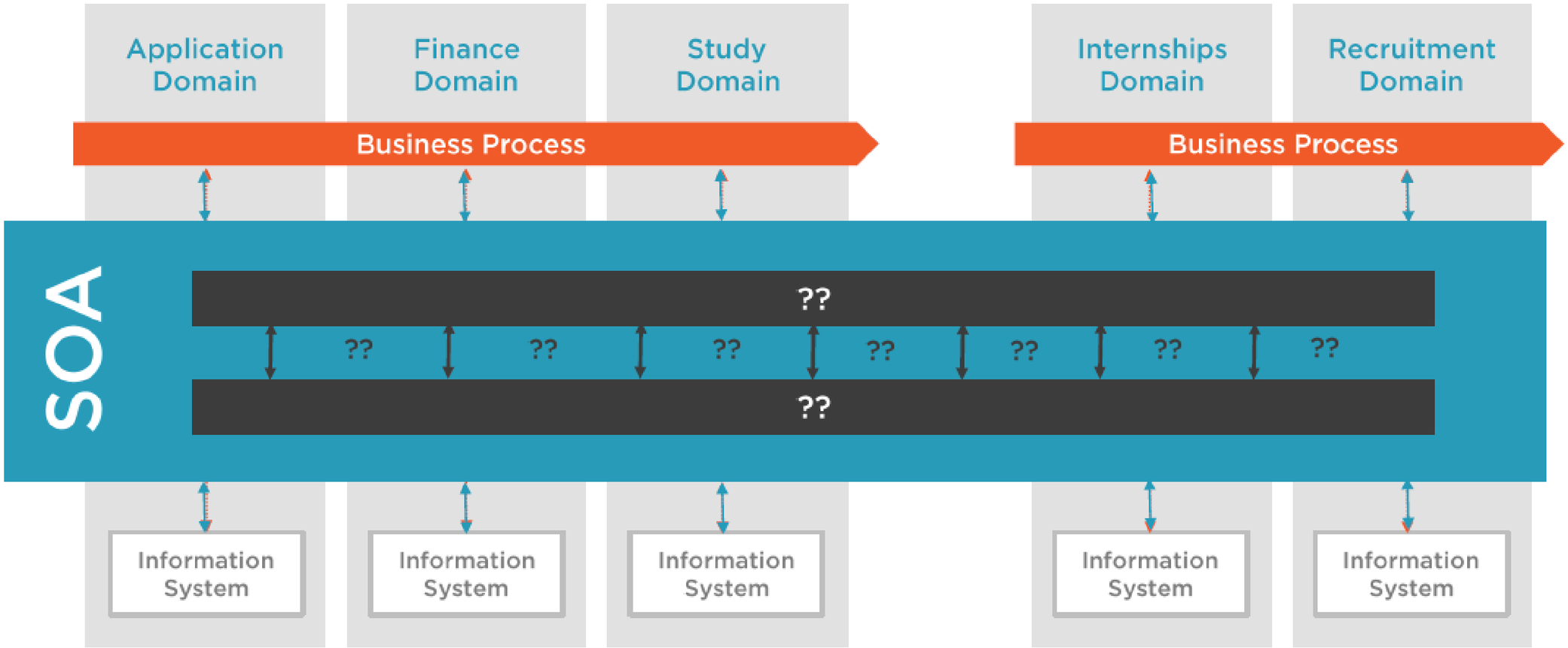
Layered Architecture



SOA


The Business Evolution
Scholarship Domain
Graduation Domain
Business Process
Application Domain
Study Domain
Finance Domain
Internships Domain
Recruitment Domain
Business Process
Business Process
Applicant Services
Student Profile Services
Financial Services
Internship Services
Jobs Services
Scholarship Services
Siloed systems
Redundant functions
Redundant data
Fragmented data
Separate data models

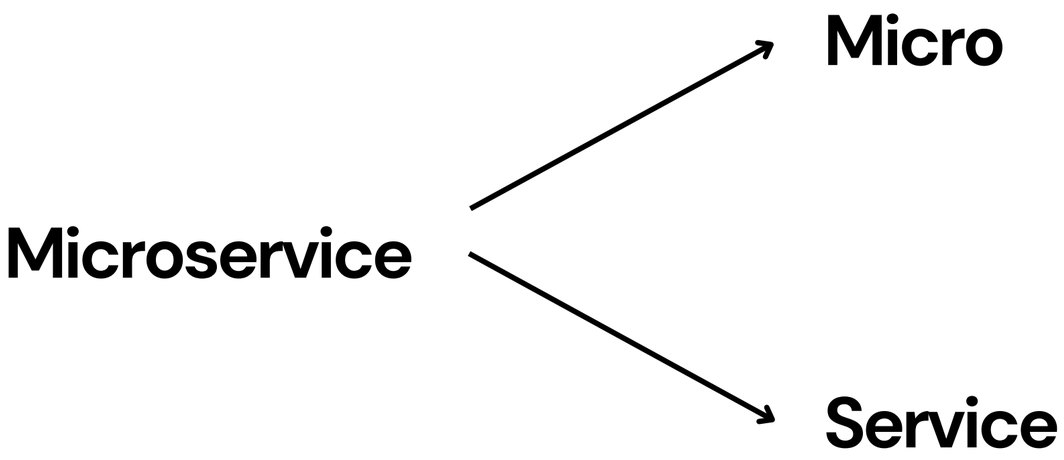
SOA


Application Domain
Finance Domain
Study Domain
Internships Domain
Recruitment Domain
Business Process
Business Process
SOA
??
??
??
??
??
??
??
??
??
??
Information System
Information System
Information System
Information System
Information System

Architecture Microservice

Architecture Microservice

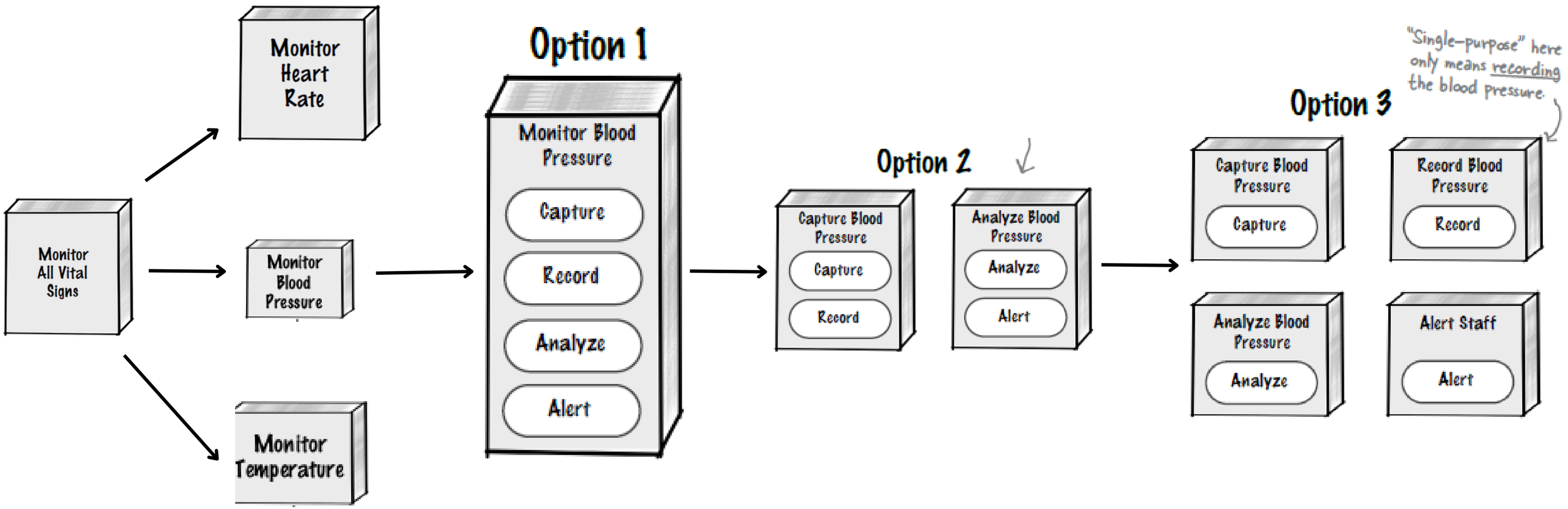
Qu'est ce qu'un microservice



Un **microservice** est un processus **logiciel minuscule** et indépendant qui s'exécute selon son propre calendrier de déploiement et peut être mis à jour indépendamment.

Microservice MonitorMe

Qu'est ce qu'un microservice



Benefits of Microservices

Small Services

Can be owned by a team

Easier to understand

Can be rewritten

Technology Choice

Adopt new technology

Use the right tool

Standardize where it makes sense

Individual Deployment

Lower risk

Minimize downtime

Frequent updates

Scaling

Scale services individually

Cost-effective

Agility

Adapt rapidly

Easier reuse

Challenges of Microservices

Developer Productivity

How can we make it easy for developers to be productive working on the system?

Complex Interactions

Take care to avoid inefficient, chatty communications between microservices

Deployment

You will need to automate the process

Monitoring

We need a centralized place to check logs and monitor for problems

Cas de d’etude au Cameroun

Cas de d’etude au Cameroun

RH - Gestion des ressources humaines

Courrier: Gestion du Courrier

Autres architectures

- Architecture Orientée Événements (Event-Driven Architecture - **EDA**)
- Architecture Sans Serveur (Serverless)
- Architecture en Conteneurs (Container-Based)

MERCI

regis_ate

Regis Atemengue

Regis Atemengue

www.regisatemengue.com

Régis ATEMENGUE

Software Engineer | Technical Instructor.

@regis_ate | www.regisatemengue.com