Recherche: Monolithic vs Microservice Architecture

1. Einführung

Microservices sind ein Architekturansatz, bei dem Anwendungen in kleine, unabhängige Dienste aufgeteilt werden, die jeweils eine spezifische Funktion erfüllen. Diese Dienste kommunizieren über leichtgewichtige Protokolle wie REST oder gRPC. [Quelle: https://martinfowler.com/articles/microservices.html]

---

3. Fragestellungen

Was versteht man unter Microservices?

Microservices sind eine Architektur, bei der Anwendungen in unabhängige, lose gekoppelte Dienste aufgeteilt werden. Jeder Dienst ist eigenständig und kann unabhängig entwickelt, bereitgestellt und skaliert werden. [Quelle: https://aws.amazon.com/microservices/]

Beispiel für den Einsatz von Microservices

Netflix nutzt Microservices, um verschiedene Funktionen wie Benutzerverwaltung, Streaming und Empfehlungen zu trennen. Dies ermöglicht eine unabhängige Skalierung und schnelle Fehlerbehebung. [Quelle: https://netflixtechblog.com/tagged/microservices]

Vor- und Nachteile einer monolithischen Architektur

Vorteile:

1. **Einfachheit:** Leicht zu entwickeln und zu testen. [Quelle: https://martinfowler.com/articles/microservices.html]
2. **Performance:** Direkte Kommunikation zwischen Komponenten. [Quelle: https://spring.io/blog/]
3. **Kosteneffizienz:** Weniger Infrastruktur erforderlich. [Quelle: https://aws.amazon.com/monolithic/]

Nachteile:

1. **Skalierbarkeit:** Schwieriger zu skalieren. [Quelle: https://spring.io/blog/]

2. **Fehleranfälligkeit:** Ein Fehler kann das gesamte System beeinträchtigen. [Quelle: https://aws.amazon.com/monolithic/]
3. **Wartung:** Änderungen sind komplex und zeitaufwendig. [Quelle: https://martinfowler.com/articles/microservices.html]

Vor- und Nachteile von Microservices

Vorteile:

1. **Skalierbarkeit:** Dienste können unabhängig skaliert werden. [Quelle: https://aws.amazon.com/microservices/]
2. **Flexibilität:** Verschiedene Technologien können verwendet werden. [Quelle: https://martinfowler.com/articles/microservices.html]
3. **Fehlerisolierung:** Ein Fehler in einem Dienst beeinträchtigt nicht das gesamte System. [Quelle: https://spring.io/blog/]

Nachteile:

1. **Komplexität:** Erhöhte Komplexität bei der Verwaltung. [Quelle: https://martinfowler.com/articles/microservices.html]
2. **Kosten:** Höhere Infrastrukturkosten. [Quelle: https://aws.amazon.com/microservices/]
3. **Kommunikation:** Erfordert robuste Kommunikationsmechanismen. [Quelle: https://spring.io/blog/]

Was ist ein Service Mesh?

Ein Service Mesh ist eine Infrastruktur, die die Kommunikation zwischen Microservices verwaltet. Es bietet Funktionen wie Lastverteilung, Sicherheit und Überwachung. [Quelle: https://istio.io/latest/docs/concepts/what-is-istio/]

Vergleich von Service Mesh Produkten

1. **Istio:** Umfangreiche Funktionen, aber komplex. [Quelle: https://istio.io/]
2. **Linkerd:** Einfacher zu implementieren, weniger Funktionen. [Quelle: https://linkerd.io/]
3. **Consul:** Gute Integration mit verschiedenen Plattformen. [Quelle: https://www.consul.io/]

Microservice-Architektur mit Java

Mit **Spring Boot** und **Spring Cloud** können Microservices in Java entwickelt werden. Diese Frameworks bieten Tools für REST-APIs, Service Discovery und Konfigurationsmanagement. [Quelle: https://spring.io/projects/spring-boot]

Microservice-Architektur mit Python

Python-Frameworks wie **Flask** und **FastAPI** eignen sich für die Entwicklung von Microservices. Docker und Kubernetes können für die Bereitstellung verwendet werden. [Quelle: https://fastapi.tiangolo.com/]

---

4. Konzept für ein eCommerce-System

Architektur:

- **Microservices:** Für Benutzerverwaltung, Produktkatalog, Bestellabwicklung und Zahlungsabwicklung.
- **Datenbanken:** Separate Datenbanken für jeden Dienst.
- **Technologien:** Spring Boot (Java), Flask (Python), Docker, Kubernetes.

Merkmale:

- Synchronisation der Kundendatenbank. [Quelle: https://aws.amazon.com/microservices/]
- Unterstützung für Rechnungen und Gutschriften. [Quelle: https://spring.io/projects/spring-boot]
- Integration von Zahlungsoptionen wie Kreditkarte und PayPal. [Quelle: https://stripe.com/docs]

Begründung für Microservices:
Microservices ermöglichen eine flexible Skalierung und vereinfachen die Integration neuer Funktionen. Dies ist besonders wichtig für ein wachsendes eCommerce-System. [Quelle: https://martinfowler.com/articles/microservices.html]