Glossar

- ey, hier kann ich ja meine ganzen Fachbegriffe reinschreiben und es sollte super aussehen

Einleitung

"Monolithic architectures are not inherently bad, just as microservices are not inherently good. The key is knowing when to use each."

Einführung in das Thema

- Entstehung von Softwarearchitekturen
    - Wie/Was sind Mainframes
    - Wie/Was Monolithen
    - Wie/Was Modulare Architektur(spezifisch Microservices)

Problemstellung

- Welt wird immer schnelle -> Evolution immer wichtiger
        - Microservices ist moderne Architektur (hier ein Beispiel von einem häufig verwendeten Microservice)
        - Monolithen sind älter, aber ist das schlimm?
            - werden auch noch viel verwendet (hier dann noch ein Beispiel von einem häufig verwendeten Monolithen)
        - gibt vielleicht Vorteile bei den jeweilige Architekturen
            - welche Architektur ist denn besser/ hat mehr Vorteile
            - gibt es überhaupt eine bessere Architektur?

Forschungsfrage und Hypothesen

- Mit Forschungsfrage: “Welche Vorteile bietet der Einsatz von Microservices in Kombination mit Rest-APIs gegenüber Monolithen?” will ich die gegebene Problemstellung auf arbeiten, verarbeiten und zum Schluss beantworten
- Zudem sind aufgestellte Hypothesen
    - Microservices bieten weitaus mehr Vorteile gegenüber Monolithen
    - Microservices bieten kaum oder gar keine Vorteile gegenüber Monolithen
    - Microservices bieten mehr Nachteile als Vorteile gegenüber Monolithen
    - man kann einen Hybrid benutzen, welcher Vorteile von Monolithen und Microservices bietet -> möglichst wenig Nachteile -> Modulith (modularer Monolith)

Ziel der Arbeit

- Unterschied der Architekturen klar darstellen
- Zeigen, welches der Beiden besser ist
- besseres Verständnis für Beide bekommen
- Zukunft der Software Architekturen darstellen

Relevanz des Projekts für den Betrieb

- Planungstool für den erleichterten Netzausbau der Deutschen Telekom AG (DTAG)
- Plural ist so ein Planungssystem -> Kabelwege, Übertragungswege, Knoten, Geräte mit Ports und Steckplätzen (Hier vielleicht die Bilder von Ole, und eventuell alles so ein bisschen erklären) dokumentieren
- Plural wurde … entwickelt, mit dem Softwareentwicklungstool CA Gen (wird noch erklärt)
- Plural ist veralteter Monolith, kompliziert zu bedienen und verwendet CA Gen, welches von der jüngeren Generation schwerer zu bedienen
    - (die Kollegen, die das Planungstool entwickelt haben, werden durch den demographischen Wandel die telekom verlassen, danach gibt es keinen, welcher sich mit dem Programm gut auskennen und es wird um einiges schwerer das Planungstool zu erneuern)
- Plural soll von einem neuen System/Planungstool abgelöst werden
- da kommt PlanDok ins spiel
    - ist ein Programm, das auf dem Code von Plural basiert
    - wird mit Java und Quarkus-Framework programmiert, Microservice-orientierte Architektur, mit REST APIs
    - Bietet ein modernes und übersichtliches GUI (Web-Anwendung/Web-Interface oder so)
    - außerdem ist eine erleichterte erweiterung, durch Nutzung moderner entwicklungskomponenten gegeben

Methodik und Vorgehensweise

- benutzt eine konzeptionell deduktive Analyse
- außerdem Prototyping
- als erstes Wesentliche Merkmale beider architekturen heraussuchen, dann schauen welche bei beiden ähnlich sind -> diese als Kriterien benutzen
- danach praktische umsetzung des Prototyps, Mein eigener Restservice -> einzelne Dinge erklären so was wie die struktur der Microservices, einzelne Klassen erklären, vielleicht die pom, Tests erklären -> vllt auch einen zeigen, auch bisschen Datenbank
- wenn das fertig, dann Monolith
- Plural wird hier beispielhaft verwendet ->also nach den aufgestellten kriterien untersuchen -> die dann beantworten
- danach durch die konzeptionell deduktive analyse vergleichen

Grundlagen

- Hier wird festgelegt, was wichtig zu wissen ist, und was vielleicht eine Voraussetzung ist um die Arbeit zu verstehen

Wissenschaftliche Methode

- konzeptionell deduktive Analyse
    - Konzept/ in dem Fall einen Prototyp und eine beispielhafte verwendung, welche im endeffekt nach den gegebenen Kriterien Analysiert werden
- Prototyping
    - Prototyp entwicklung

Softwareentwicklung Technologien

- Java
    - Kurze zusammenfassung was es ist
    - außerdem woher es kommt
    - wann es entstanden ist, etc.
    - Warum wird es benutzt
- CA Gen
    - Ist ein System, zur entwicklung von Pseudocode, der zu C Code Generiert wird
    - Woher kommt es, warum wird es benutzt

Architekturen

- Microservices
    - Was ist das?
    - Woher kommt das?
    - Wieso wird es verwendet?
- Monolithen
    - Was ist das?
    - Woher kommt das?
    - Wieso wird es verwendet?

Software-mediär

- REST-APIs
    - Was ist das?
    - Woher kommt das?
    - Warum werden diese verwendet und nicht andere?

Frameworks

- Quarkus
    - Was ist das?
    - Woher kommt es?
    - Warum wird das benutzt? Gibt es Alternativen?

Bearbeitung beider Architekturen (Prototyping und beispielhafte Verwendung)

- Jetzt kommt bearbeitung beider Architekturen, zur aufstellung und Bearbeitung der Kriterien
- Recherche über Monolithen und Microservices hat Kriterien ergeben…
- Dafür wird eigener Restservice (Microservice) erstellt und Plural (Monolith) beispielhaft verwendet

Recherche/Kriterien

- hier kriterien die deswegen und deswegen wichtig sind
- vllt Modularität
- Verarbeitungsgeschwindigkeit
- Last verarbeitung oder so

Microservices

- hier kommt jetzt die Planung des Projekts und dann die Bearbeitung, alle nach den Kriterien

Implementierung

- Also, hab aufgeteilt in Planung, also alles was mit Vorbereitung zu tun hat, und Umsetzung aufgeteilt, weil, wichtig

Planung

- Mit dem Thema befasst, also hier jetzt über APIs informiert, über das Framework also Quarkus befasst
- Ich musste mir zu Beginn auch ein Thema aussuchen, zu dem ich einen Rest Service erstellen möchte. Ich habe mich für Speedruns entschieden. Dazu hatte ich mich auch informiert

Umsetzung

- Hab angefangen mit Projekterstellung und Speicherung in Gitlab, dann zuerst die yaml Datei (OpenApi First) -> bei dem Konzept openApi First wird eine Hülle für die API erstellt
- danach wurden die ganze Logik für den rest service erstellt -> jetzt das mit den Klassen reinschreiben, also entity, mapper, repository, Resource ordner oder Service ordner
- Zwischendrin immer mal wieder dependencies in der Pom angepasst -> Pom vielleicht auch nochmal erklären oder so
- Models werden erstellt wenn einmal durch maven generiert -> dann kann der mapper drauf zugreifen
- zum schluss noch unit test erstellt, also die sachen da testen um zu schauen ob alles funktioniert
- Für unit tests und ein paar klassen den Code reinschreiben -> vllt auch eine dependency oder so
- und die yaml

Monolithen

- also als Monolith wird PLURAL benutzt, hier werden beispielhaft einige sachen getestet -> Kriterien sollen damit erfüllt werden und gezeigt werden wie ein Monolith funktioniert

Beispielhafte verwendung

- hier muss ich nochmal mit Kathrin reden wie und was genau ich testen soll

Konzeptionell deduktive Analyse

- hier anhand gesammelter Kriterien eine Tabelle oder so erstellen und dann jede einzelne Spalte erklären oder so
- hier stellt sich dann raus was besser ist, oder ob es überhaupt besser ist

Zusammenfassung und Ausblick

- woah hier kann ich dann zusammenfassen was aus der Arbeit rauszunehmen ist
- zeigen warum alles was ich mache jetzt nicht useless war
- hier kann ich ja auch noch Modulithen als zukunft oder so zeigen

Anhang

KI-Verzeichnis