Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

placeholder

1.2 Zielsetzung

placeholder

Kapitel 2

Grundlagen

2.1 Erklärung API

Der Begriff API steht für Application Programming Interface (auf Deutsch Anwendungs-Programmier-Schnittstelle). Die Grundlagen heutiger APIs wurden 1952 von David Wheeler, einem Informatikprofessor an der Universität Cambrigde, in einem Leitfaden verfasst. Dieser Leitfaden beschreibt das Extrahieren einer Sub-Routinen-Bibliothek mit einheitlichem dokumentierten Zugriff. [4] Ira Cotton und Frank Greatorex erwähnten den Begriff erstmalig auf der Fall Joint Computer Con.[2] Dabei erweiterten sie den Leitfaden von David Wheeler um einen wesentlichen Punkt: Der konzeptionellen Trennung der Schnittstelle und Implementierung der Sub-Routinen-Bibliothek. Somit kann die Implementierung auf die die Schnittstelle zugreift ohne Einfluss auf die Benutzer ausgetauscht werden.[3]

> APIs sind wie User Interfaces - nur mit anderen Nutzern im Fokus. David Berlind [1]

APIs sind also wie UIs für die Interaktion mit Benutzern gedacht. Der wesentliche Unterschied zwischen UI-Schnittstellen und einer API liegt aber an der Art der Nutzer die auf das Interface zugreifen. Bei UIs spricht man von einem *human-readable-interface*, das bedeutet das ein Menschlicher User mit dem System interagiert. Bei einer API spricht man von einem *machine-readable-interface*, also von einer Schnittstelle die für die Kommunikation zwischen Maschinen gedacht ist.

2.1.1 Abstraktes Beispiel API

Ein abstraktes Beispiel für eine API wäre beispielsweise die Post. Angenommen eine Person will einen Brief an eine andere Person senden, um diese Person zum Essen einzuladen. Was passiert ist dann folgendes:

- Person 1 verfasst eine Nachricht und packt diesen in einen Briefumschlag
- Der Briefumschlag wird mit einer Briefmarke und der Adresse des Empfängers und des Absenders versehen
- Der Brief wird nun an die Post übergeben
- Die Post kümmert sich um die Zustellung des Briefes

- Person 2 erhält den Brief

Damit dieser Zugriff der unterschiedlichen Systeme (hier Menschen) auf die Post funktioniert muss eine genaue Definition des Service vorliegen. Die genaue Spezifikation des Service ist dabei folgende:

- Das zu versendende Objekt (Brief, Paket, etc.) muss an einem Sammelposten der Post abgegeben werden
- Das Objekt ist dabei mit einer Empfängeradresse und einer Absenderadresse zu versehen, zusätzlich muss eine Briefmarke gekauft werden

Das stellt die Art des Services dar und legt somit fest was über die API versendet wird. Zudem wird die Repräsentation des der API definiert, also in welcher Form der Service im System des Servicenutzers integiert wird. Der Briefkasten / Sammelposten ist dabei die eigentliche Schnittstelle - also die API. Die Zustellung ist dabei Implementierungsdetail, der Weg vom Sammelposten der Post über die Verteilerzentren und mit dem Briefträger zum Empfänger. Dieses Implementierungsdetail kann beliebig angepasst werden, die Zustellung kann beispielsweise über den Land- oder Luftweg erfolgen. Der Benutzer welcher den Brief versendet hat ist davon nicht betroffen.

2.2 Erklärung REST-API

REST steht für Representational State Transfer [4]. REST ist dabei aber keine konkrete Technologie oder ein Standard. REST beschreibt einen Architekturstil welcher im Jahr 2000 von Roy Fielding konzipiert wurde. Bei REST werden Daten als Ressourcen gesehen und in einem spezifischen Format übertragen. Ursprünglich wurde von Fielding dabei XML verwendet. XML wird aber in den letzten Jahren verstärkt durch JSON abgelöst.

Deswegen ist JSON besser als XML für den Datenaustausch mittels einer API geeignet:

- JSON wurde speziell für den leichtgewichtigen Datenaustausch konzipiert.
- JSON ist schneller als XML
- JSON hat weniger Overhead als XML da XML deklarativ ist und somit wesentlich mehr Daten als JSON beinhaltet

Zum Vergleich hier ein Buch welches in JSON und XML repräsentiert wird:

2.3 Erklärung GraphQL

placeholder

2.4 Unterschiede zwischen GraphQL und REST

placeholder

placeholder

Kapitel 3

GraphQL

3.1 Interfaces

3.2 Input Typen

placeholder

3.3 Parameter

3.4 Skalare

placeholder

3.5 Querys

3.5.1 Designempfehlungen

placeholder

3.5.2 Verschachtelte Querys

3.5.3 Variablen

placeholder

3.5.4 Fragmentierung

3.5.5 Aliase

placeholder

3.6 Mutationen

3.6.1 Designempfehlungen

placeholder

3.7 Bekannte Probleme

3.7.1 Authentifizierung, Autorisierung und Rollenmanagement

placeholder

3.7.2 1 + n Problem

3.7.3 Subscriptions

placeholder

placeholder

3.7.4 Fehlermanagement

placeholder

3.7.5 Pagination

3.7.6 Caching

placeholder

Kapitel 4

Entwicklung

4.1 Anwendungsszenario

4.2 Architektur

placeholder

placeholder

4.3 Entwurf Schema

placeholder

4.4 Umsetzung GraphQL mit .NET

4.4.1 Verwendete Bibliotheken

placeholder

4.4.2 Entity Framework

placeholder

placeholder

4.4.3 Umsetzung Authentifizierung, Autorisierung, Rollenmanagement

placeholder

4.4.4 Umsetzung Subscriptions

placeholder

4.4.5 Umsetzung 1 + n Problem

placeholder

4.4.6 Umsetzung Pagination

Kapitel 5

Conclusio

5.1 Fazit

5.2 Ausblick

Literatur

[1] David Berlind. *APIs Are Like User Interfaces–Just With Different Users In Mind*. 2017 (siehe S. 5).

[2] Ira W Cotton und Frank S Greatorex Jr. „Data structures and techniques for remote computer graphics“. In: *Proceedings of the December 9-11, 1968, fall joint computer conference, part I*. 1968, S. 533–544 (siehe S. 5).

[3] Dominik Kress. *GraphQL: Eine Einführung in APIs mit GraphQL*. dpunkt. verlag, 2020 (siehe S. 5).

[4] David J Wheeler. „The use of sub-routines in programmes“. In: *Proceedings of the 1952 ACM national meeting (Pittsburgh)*. 1952, S. 235–236 (siehe S. 5, 6).