

Inhaltsverzeichnis

Vorwort	v
1 API-Grundlagen	1
1.1 Was ist ein API?	1
1.2 Vorteile eines API	3
1.2.1 Flexibilität für Anbieter und Konsument	3
1.2.2 Einheitliches Design und Funktionen	4
1.2.3 Neue Geschäftsfelder	4
1.2.4 Innovationstreiber API	4
1.3 API: Die Definition	5
1.3.1 API-Vertrag	5
1.3.2 Die Akteure eines API	6
1.3.3 Release-Arten von APIs	7
1.4 Mögliche API-Technologien und -Spezifikationen	8
1.4.1 Geschichte der Remote Execution	8
1.4.2 RESTful HTTP	10
1.4.3 JSON:API	12
1.4.4 gRPC	15
1.4.5 GraphQL	18
1.4.6 Die Technologien im Vergleich	20
2 Von der Idee zur Umsetzung	21
2.1 API Value Chain	22
2.1.1 Geschäftsmodelle für private und öffentliche APIs ...	24
2.2 Release-Arten	27
2.2.1 Private APIs	27
2.2.2 Öffentliche APIs	28
2.3 Erste Schritte: Allgemeines Vorgehen	29
2.3.1 Use Cases identifizieren	29
2.3.2 Funktionale Anforderungen	31
2.3.3 Nicht-funktionale Anforderungen	32
2.3.4 Die gemeinsame Sprache	32
2.3.5 Gemeinsames Vokabular durch Schema.org erzeugen	33

3	Grundlagen der GraphQL-API.....	37
3.1	Das Graphen-Modell erzeugen	38
3.2	Abfragen mit GraphQL	40
3.2.1	Grundlegende Querys	40
3.2.2	Querys generell.....	41
3.2.3	Verschachtelte Querys	42
3.2.4	Parameter in Querys	43
3.2.5	Variablen in Querys	44
3.2.6	Aliases in Querys.....	46
3.2.7	Fragmentierte Querys	47
3.2.8	Direktiven in Querys	49
3.2.9	Inline-Fragmente in Querys	51
3.2.10	Metafelder in Querys.....	52
3.2.11	Mutationen: Datenmanipulation mit GraphQL.....	54
3.2.12	Subscriptions: GraphQL Message Streaming.....	55
3.3	Das GraphQL-Typ-System: Schemadefinition	57
3.3.1	Grundlegende Schemas.....	58
3.3.2	Skalar-Typen.....	60
3.3.3	Enumerations-Typen	60
3.3.4	Typ-Modifikatoren: Listen und Non-Null.....	61
3.3.5	Parameter	62
3.3.6	Input-Typen	63
3.3.7	Interfaces	65
3.3.8	Union-Typen	66
4	API-Design I: Rund ums Schema des API	67
4.1	Qualitätsmerkmale	67
4.2	Designempfehlungen.....	71
4.2.1	Schemadesign-Empfehlungen.....	72
4.2.2	Mutation-Designempfehlungen	77
4.3	HTTP: Netzwerk-Design	81
4.3.1	POST-Requests	81
4.3.2	GET-Requests.....	82
4.3.3	Responses.....	83
4.4	Pagination	84
4.4.1	Splicing	84
4.4.2	Offset-basierte Pagination.....	85
4.4.3	Cursor-basierte Pagination	85
4.4.4	Edges und Connections	86
4.5	Fehlermanagement	88
4.5.1	Application Errors.....	89
4.5.2	Type und Validation Errors.....	89
4.5.3	Fehler mit partiellen Ergebnissen	90
4.5.4	Fehler ohne Teilergebnisse	91

5	API-Design II: Die Landschaft um das API	93
5.1	Autorisierung	93
5.1.1	GraphQLs Probleme mit Autorisierung	93
5.1.2	Autorisierung auf Ebene der Geschäftslogik	94
5.2	Dokumentation	95
5.2.1	Statische Dokumentation	96
5.2.2	Dynamische Dokumentation	98
5.3	Versionierung	100
5.3.1	GraphQLs Evolution im Beispiel	101
5.4	Monitoring und Instrumentation	102
5.4.1	Feingranulares Monitoring	102
5.4.2	Verstehen, wie das API genutzt wird	103
5.5	Performanzoptimierung: Caching und Batching	103
5.5.1	Das 1+n-Problem	104
5.5.2	DataLoader	105
5.5.3	CDN-Caching	106
5.5.4	Clientseitiges Caching	107
6	Implementierung mit Node I: Das erste Schema	109
6.1	Use Case	109
6.2	Initiales Aufsetzen des Projekts	110
6.2.1	Das Node.js-Projekt aufsetzen	110
6.2.2	Den GraphQL-Server mit Apollo aufsetzen	111
6.3	Das initiale Schema aufsetzen	113
6.3.1	Parameter und erste Resolver-Logik	114
6.3.2	Feld-Level-Resolver und Interfaces	116
6.3.3	Interfaces und Filter für IDs	118
6.3.4	Typrelationen	122
7	Implementierung mit Node II: Erweitertes Schema und Mutationen	127
7.1	Schema-Modularisierung	127
7.1.1	Technische Separation	127
7.1.2	Domain-Separation	131
7.1.3	Resolver Map aufteilen und Models durch Context verteilen	133
7.2	Mutationen	136
7.2.1	Ein Produkt erstellen	136
7.2.2	Produkt löschen	139
7.2.3	Wunschliste mit Input-Typen erstellen	141
7.2.4	Wunschlisten kaskadierend löschen	145

8	Implementierung mit Java I: Das erste Schema	149
8.1	Use Case	150
8.2	Das Projekt aufsetzen	150
8.2.1	Den GraphQL-Server aufsetzen	151
8.3	Das initiale Schema aufsetzen	154
8.3.1	Objekte im Schema auflösen	154
8.3.2	Feld-Resolver	156
8.3.3	Das Node-Muster	157
8.3.4	Ergebnisse filtern durch Parameter	159
8.3.5	Objekt-Relationen	160
8.3.6	ID-Referenz-basierte Objekt-Relationen	162
9	Implementierung mit Java II: Erweitertes Schema und Mutationen	167
9.1	Selbstdefinierte Skalar- und Geschäftslogik-Felder	167
9.1.1	Skalar-Typ in Schema und POJO definieren	168
9.1.2	Die GraphQLScalarType-Implementierung	169
9.1.3	Geschäftslogik-Felder und -Parameter	172
9.2	Mutationen erstellen und Schemamodularisierung	175
9.2.1	Kunden registrieren	175
9.2.2	Adressen löschen	178
9.2.3	Adressen erstellen: Input-Typen	180
9.2.4	Bestellung erstellen: verschachtelte Input-Typen	182
	Literaturverzeichnis	187
	Index	191