Vorwort

Es gibt bereits sehr gute Bücher über Softwarearchitektur und Design. Warum dann noch dieses Buch über den Entwurf von Application Programming Interfaces, kurz APIs? Weil meiner Meinung nach der Bedarf dafür existiert! Denn API-Design unterscheidet sich von den klassischen objektorientierten Design-Heuristiken, die mit einem internen Datenmodell beginnen und versuchen, Applikationen im Hinblick auf Wartbarkeit und Flexibilität zu optimieren. Im Gegensatz dazu nimmt API-Design die Perspektive der Benutzer, d. h. anderer Entwickler, ein und versucht, die Benutzung von Komponenten oder Diensten durch gutes API-Design für diese Entwickler möglichst einfach zu machen. API-Design ist deswegen nicht nur wichtig für Open-Source-Frameworks, auch unternehmensinterne Softwarekomponenten können davon profitieren. Ein anderer Grund für dieses Buch ist die schnell wachsende Anzahl intern und extern eingesetzter Web-APIs. Deswegen finden Sie in diesem Buch Techniken und Best Practices für Java-, Web- und Messaging-APIs.

Warum ist API-Design wichtig?

APIs gibt es in allen Größen und Formen: Das Spektrum reicht von der Betriebssystem-API POSIX bis zur Web-API des Kurznachrichtendienstes Twitter und von der API der Java-Klassenbibliothek bis zur Web-API des Cloud-Speichers Amazon S3.

> APIs sind der Kleber, der unsere digitale Welt zusammenhält.

Als Softwareentwickler arbeiten wir ständig mit ihnen. Aber wir benutzen sie nicht nur, wir schreiben sie auch. Wenn Sie ein Softwareentwickler sind, dann sind Sie auch ein API-Designer. Warum ist das so? Beispielsweise ist eine gute Architektur in Module strukturiert. Jedes dieser Module braucht eine API, über die es aufgerufen werden

kann. Die APIs entscheiden darüber, wie einfach oder schwer die Module integriert werden können.

Nützliche Module werden wiederverwendet, sodass deren API nicht nach Belieben verändert werden sollte. Angenommen eine API wird von drei anderen Applikationen verwendet, dann gibt es drei gute Gründe, Kompatibilität bei Änderungen zu berücksichtigen.

> APIs können zu den wertvollsten Assets eines Unternehmens gehören.

Mithilfe von APIs können Kunden oder Partner einen Dienst integrieren. Häufig wird deswegen auch die API als das eigentliche Produkt angesehen. Kunden investieren viel Zeit und Geld in die Integration einer API. Die Kosten für einen Umstieg auf eine andere API sind meist sehr hoch. Deswegen werden Kunden durch APIs gebunden. Allein die Twitter-API hat Zehntausende registrierte Applikationen.

> APIs können zu den größten Verbindlichkeiten eines Unternehmens gehören.

APIs können nicht nur wertvoll für Unternehmen sein, sie können auch eine Last darstellen. Eine schlechte API oder eine schlechte Dokumentation kann unzählige Supportanfragen zur Folge haben. Aber eine schlechte API kann nicht mal eben verändert werden. Aus diesem Grund erfahren Sie in diesem Buch, was alles beim API-Design zu beachten ist.

Eine Frage der Perspektive

Zweifellos gibt es Softwareentwickler, die APIs korrekt entwickeln, ansonsten gäbe es nicht so viele gute Applikationen, Frameworks und Webservices. Doch es scheint so, als ob die Prinzipien des API-Designs häufig nur unbewusst durch Erfahrung erlernt werden. Softwareentwickler folgen Regeln, ohne sich dessen bewusst zu sein oder deren zugrunde liegende Motive zu kennen.

> Beim API-Design geht es um Kommunikation zwischen Entwicklern.

APIs werden nicht für Computer geschrieben, sondern für Menschen. Was bedeutet das? Wir schreiben Software nur in ganz wenigen Ausnahmen in Isolation. Vielmehr baut unsere Software auf existierenden Komponenten und Services auf, deren APIs wir auswählen und verste-

hen müssen. Weil diese Komponenten und Services von anderen Entwicklern geschrieben wurden, ergibt sich ein Kommunikationsproblem. Denn in den wenigsten Fällen können uns diese Entwickler persönlich erklären, wie die API funktioniert. Daher muss die API selbst der primäre Kommunikationskanal sein. Das kann nur funktionieren, wenn eine API klar, einfach und gut dokumentiert oder sogar selbsterklärend ist.

Können Sie voraussetzen, dass die Benutzer Ihrer API diese bis in letzte Detail verstehen? Das Gegenteil ist oftmals der Fall: Entwickler müssen ihren Job möglichst schnell erledigen. Da bleibt keine Zeit, alles über eine API in Erfahrung zu bringen, bevor man sie benutzt. Entwickler rufen eine Methode der API auf und schauen, was passiert. Wenn das gewünschte Verhalten eintritt, sind sie fertig, ansonsten probieren sie etwas anderes. Entwickler fangen häufig mit Erfahrung an und das Verständnis folgt später, in manchen Fällen nie [Tulach 2008]. Dieses Prinzip der Ahnungslosigkeit ist sogar gewünscht. Denn darum geht es ja gerade bei Modularisierung, Wiederverwendung und dem Geheimnisprinzip. Aus diesem Grund muss eine API intuitiv verständlich sein. Sie sollte keine Überraschungen bereithalten und es Entwicklern schwer machen, sie falsch zu benutzen.

> Die Perspektive ist beim API-Design entscheidend. APIs sollten aus der Perspektive ihrer potenziellen Benutzer entworfen werden.

Wer sind die Benutzer? Was wollen sie mit der API machen? Welche Technologien benutzen sie? Das sind Fragen, die API-Designer beantworten müssen, um eine erfolgreiche und gute API zu entwerfen.

Zielgruppe und Voraussetzungen

Dieses Buch richtet sich an Softwareentwickler und -architekten, die APIs für Frameworks, Bibliotheken oder andere Softwarekomponenten entwickeln. Aber prinzipiell ist das in diesem Buch vorgestellte API-Design für jeden Entwickler interessant, der Code schreibt, der von anderen Entwicklern wiederverwendet wird. Zu Beginn des Buches werden allgemeine Konzepte, Qualitätsmerkmale und Vorteile des API-Designs beschrieben. Dann folgen praktische Tipps und Best Practices für Java-Softwarekomponenten. Eine Übertragung auf andere Programmiersprachen ist durchaus möglich, muss aber durch den Leser erfolgen.

Das Buch richtet sich auch an Softwareentwickler und architekten, die Web-APIs entwickeln und dafür REST und HTTP einsetzen. Für mobile Applikationen, IoT-Szenarien, zur Integration von Microservices etc. eignen sich auch Messaging-APIs, die ebenfalls in diesem Buch betrachtet werden. Praktische Erfahrungen mit diesen Technologien sind sicherlich von Vorteil, aber keine zwingende Voraussetzung, denn alle Konzepte und Technologien werden erklärt.

Struktur des Buches

In diesem Buch werden Sie sowohl allgemeine Konzepte als auch konkrete Techniken und Best Practices für unterschiedliche APIs und Protokolle kennenlernen. Aus diesem Grund ist das Buch in vier Teile gegliedert, die wiederum aus mehreren Kapiteln bestehen.

Teil I: Grundlagen

Der erste Teil des Buches umfasst wichtige Grundlagen und besteht aus den Kapiteln 1 bis 3:

- **Kapitel 1** beginnt mit einem Überblick über die Geschichte der APIs. In diesem Einstieg werden Zweck, Funktion und Bedeutung von APIs beschrieben.
- **Kapitel 2** stellt die Qualitätsmerkmale vor, die beim API-Design berücksichtigt werden sollten. Diese Merkmale sind die Voraussetzung für alle weiteren Designtechniken in diesem Buch.
- **Kapitel 3** beschreibt das allgemeine Vorgehen beim Entwurf von APIs. Für den Entwurf werden Beispiele eingesetzt, die zeigen, wie die API von Clients in verschiedenen Szenarien benutzt werden soll.

Teil II: Java-APIs

Nach der allgemeinen Einführung geht es im zweiten Teil um objektorientierte Java-APIs:

- **Kapitel 4** beschreibt die vielfältigen Ausprägungen von objektorientierten APIs, die man in Bibliotheken, Frameworks und anderen Softwarekomponenten findet.
- **Kapitel 5** stellt grundlegende Techniken und Best Practices für Java-APIs vor. Hierzu zählen codenahe Themen wie Command/Query Separation, Design für Vererbung, Verwendung von Interfaces und Exception Handling.
- **Kapitel 6** betrachtet Techniken, die größeren Einfluss auf die Architektur der Anwendung haben, dennoch aber zum API-Design zählen.
- **Kapitel 7** führt in das Thema Kompatibiliät ein. In diesem Zusammenhang werden theoretische Grundlagen und praktische Techniken vorgestellt.

Teil III: Remote-APIs

An dieser Stelle verlassen wir die Welt der Java-APIs und kommen zu den Remote-APIs, die primär zur Integration unterschiedlicher Systeme eingesetzt werden. Konkret geht es im dritten Teil um RESTful HTTP, SOAP-Webservices und Messaging-APIs:

- **Kapitel 8** führt in den Architekturstil REST ein und beschreibt dessen Grundprinzipien anhand von RESTful HTTP.
- **Kapitel 9** stellt Techniken für Web-APIs vor und geht dabei u.a. auf Ressourcendesign, Medientypen und Fehlerbehandlung ein.
- **Kapitel 10** behandelt Designtechniken für SOAP-Webservices. Dabei werden Granularität, Message Exchange Patterns, Datentypen und Versionierung beschrieben.
- **Kapitel 11** stellt Messaging als weitere wichtige Alternative zur Integration und Aufrufverarbeitung vor. Neben einem umfassenden Anwendungsbeispiel werden verschiedene Protokolle und Produkte gezeigt.

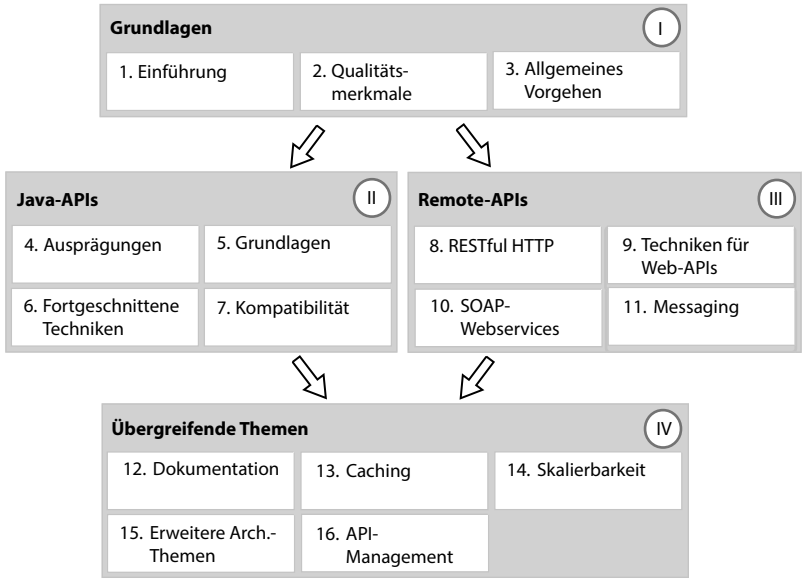
Teil IV: Übergreifende Themen

Der abschließende vierte Teil des Buches behandelt Querschnittsthemen wie Dokumentation, Skalierbarkeit und API-Management:

- **Kapitel 12** bietet Empfehlungen zur Dokumentation von APIs. Das Kapitel wird mit der Vorstellung hilfreicher Dokumentationswerkzeuge vervollständigt.
- **Kapitel 13** behandelt das Thema Caching, das insbesondere für die Performance von RESTful HTTP sehr wichtig sein kann.
- **Kapitel 14** stellt die Grundlagen skalierbarer Systeme vor. In diesem Zusammenhang werden das CAP-Theorem, statuslose Kommunikation, Load Balancing und verschiedene Architekturvarianten diskutiert.
- **Kapitel 15** diskutiert Consumer-Driven Contracts, »One size fits all«-APIs und andere Architekturthemen.
- **Kapitel 16** stellt zu guter Letzt das Thema API-Management vor. Architektur und Werkzeuge für das Veröffentlichen, Dokumentieren und Managen von APIs werden beschrieben.

Pfade durch das Buch

Falls Sie das Buch nicht von Anfang bis Ende lesen wollen, können Sie die Einteilung des Buches nutzen und verschiedene Pfade durch das Buch wählen. Wenn Sie beispielsweise hauptsächlich am Design von Web-APIs auf Basis von RESTful HTTP interessiert sind, können Sie nach den Kapiteln 1 bis 3 direkt mit den Kapiteln 8, 9 und 11–15 fortfahren. Falls Sie hingegen an Java-APIs interessiert sind, können Sie nach den Kapiteln 1 bis 7 zum Kapitel 12 springen.



Danksagung

Bei der Arbeit an diesem Buch konne ich von kritischen Diskussionen und wertvollen Kommentaren profitieren. Besonders möchte ich mich bei Eberhard Wolff, Stefan Tilkov, Dirk Ludwig, Ulf Fildebrandt, Ivo Walther und Stefanie Elste bedanken.

Ebenso geht ein Dankeschön an das Team vom dpunkt.verlag. Die Zusammenarbeit mit meinem Lektor René Schönfeldt war stets professionell und freundlich.

Der größte Dank gehört jedoch meiner Frau Ileana, die mich von Anfang an unterstützte und auf viele gemeinsame Abende und Wochenenden verzichten musste.

Bei der Arbeit an der 2. Auflage dieses Buches habe ich von den vielen Hinweisen und Verbesserungsvorschlägen von Anton Schönfeld, Prof. Dr. Dominik Gruntz und Matthias Müller profitiert.

Danke!