

RESTful API Design

APIs your consumers will love

Matthias Biehl

RESTful API Design
Copyright © 2016 by Matthias Biehl
All rights reserved, including the right to reproduce
this book or portions thereof in any form whatsoever.

First edition: August 2016

Biehl, Matthias

API-University Press

Volume 3 of the API-University Series.

Includes illustrations, bibliographical references and index.

ISBN-13: 978-1514735169

ISBN-10: 1514735164



API-University Press

<http://www.api-university.com>

info@api-university.com

Contents

1. Introduction	17
1.1. What is an API?	17
1.2. Why APIs?	18
1.3. How are APIs used?	20
1.4. What is API Design?	21
1.5. What is the difference between API Design and API Architecture?	22
1.6. Why is API Design Important?	24
1.7. Why should I build RESTful APIs?	25
1.8. Why do I need OpenAPI, Swagger or RAML? . .	26
1.9. How to put API Design into Practice?	26
2. Consumer-Oriented API Design: APIs are Products	29
2.1. Why should APIs be Consumer-Oriented?	29
2.2. What is a Consumer-Oriented API?	30
2.3. How to build Consumer-Oriented APIs?	31
2.3.1. Identify the API Consumers	31
2.3.2. Engage with API Consumers	31
2.3.3. Learn about the Solution Architecture of Consumers	32
3. API Design and Development Approach	33
3.1. Foundations	33
3.1.1. Consumer-Oriented Design Approach . .	34
3.1.1.1. Inside-out Approach	34
3.1.1.2. Outside-in Approach	34
3.1.2. Contract First Design Approach	35

3.1.3.	Agile Design Approach	36
3.1.4.	Simulation-based Design	36
3.1.4.1.	Simulation of Backends	37
3.1.4.2.	Simulation of the API	37
3.1.5.	Conclusion	38
3.2.	Design Approach	39
3.2.1.	Overview	39
3.2.2.	Phase 1: Domain Analysis	40
3.2.2.1.	Verification of Phase 1: Simula- tion & Demo App	42
3.2.3.	Phase 2: Architectural Design	43
3.2.3.1.	Verification of Phase 2: Simula- tion & Demo App	44
3.2.4.	Phase 3: Prototyping	45
3.2.4.1.	Validation of Phase 3: Accep- tance Tests with Pilot Consumers	47
3.2.5.	Phase 4: Implementation for Production .	48
3.2.5.1.	Verification of Phase 4: Accep- tance Tests with Pilot Consumers	49
3.2.6.	Phase 5: Publish	49
3.2.6.1.	Verification of Phase 5: Study Metrics, Reports and Logs	50
3.2.7.	Maintenance	52
3.3.	Discussion	52
3.3.1.	Hand-over Points	52
3.3.2.	Pre-Work vs. Actual Work	53
3.4.	Summary	54
4.	API Design with API Description Languages	55
4.1.	What are API Description Languages?	56
4.2.	Usage	57
4.2.1.	Communication and Documentation . . .	57
4.2.2.	Design Repository	59
4.2.3.	Contract Negotiation	59

4.2.4.	API Implementation	60
4.2.5.	Client Implementation	61
4.2.6.	Discovery	61
4.2.7.	Simulation	62
4.3.	Language Features	63
4.4.	Limitations	65
4.5.	Summary	65
5.	API Architectural Design Decisions	67
5.1.	Requirements for APIs	68
5.1.1.	Responsibilities of APIs	68
5.1.1.1.	Gathering Data	68
5.1.1.2.	Structuring and Formatting Data	68
5.1.1.3.	Delivering Data	69
5.1.1.4.	Securing and Protecting	69
5.1.2.	Desirable Properties of APIs	69
5.1.2.1.	Consumer-Centric	70
5.1.2.2.	Self-Explanatory, Intuitive and Pre- dictable	70
5.1.2.3.	Explorable and Discoverable	70
5.1.2.4.	Well-Documented	71
5.1.2.5.	Atomic	71
5.1.2.6.	Forgiving and Forward Compatible	71
5.1.2.7.	Secure and Compliant	71
5.1.2.8.	Performant, Scalable and Available	71
5.1.2.9.	Conforming to Standards	72
5.1.2.10.	Interoperable	72
5.1.2.11.	Reusable	72
5.1.2.12.	Backward Compatible	72
5.1.2.13.	Summary	73
5.2.	Architectural Patterns	73
5.2.1.	Client Server Patterns	73
5.2.1.1.	Stateful Server Pattern	74
5.2.1.2.	Stateless Server Pattern	74

5.2.2.	Facade Pattern	75
5.2.2.1.	Proxy Pattern	76
5.3.	Architectural Styles	76
5.3.1.	REST Style	77
5.3.2.	HATEOAS Style	78
5.3.3.	RPC Style	78
5.3.3.1.	How does RPC work?	78
5.3.3.2.	JSON-RPC	79
5.3.3.3.	XML-RPC	79
5.3.4.	SOAP Style	79
5.3.5.	Streaming Style	80
5.4.	Architectural Trade-offs	81
5.4.1.	RPC in Comparison to REST	81
5.4.2.	HATEOAS in Comparison to REST	82
5.4.3.	SOAP in Comparison to REST	83
6.	Introduction to REST	85
6.1.	HTTP	85
6.2.	REST Concepts	86
6.2.1.	Resource	87
6.2.2.	API	87
6.2.3.	Representation	88
6.2.4.	Uniform Resource Interface	88
6.3.	REST Constraints	89
6.4.	State in REST	90
6.4.1.	Application State	90
6.4.2.	Resource State	91
6.5.	Advantages of REST	91
6.6.	HATEOAS Style	93
6.6.1.	HATEOAS Concepts	93
6.6.2.	HATEOAS Constraints	94
6.6.3.	Advantages of HATEOAS	94

7. API Frontend Design Decisions	95
7.1. Resources	95
7.1.1. What is a Resource?	96
7.1.2. Instance Resources	97
7.1.3. Collection Resources	97
7.1.4. Controller Resources	98
7.1.5. Resource Ordering	99
7.1.5.1. Root Resource	99
7.1.5.2. Sub Resource	100
7.1.6. Resource Granularity	100
7.1.7. Resource Relations	101
7.1.7.1. Option 1: Resource Ordering	102
7.1.7.2. Option 2: IDs and Separate Root Resources	102
7.1.7.3. Option 3: Links and Separate Root Resources	103
7.1.7.4. Option 4: Embedded Resources	104
7.1.8. Links	105
7.1.8.1. Relative URIs vs. Absolute URIs	106
7.1.8.2. Expressing URI Parameters	106
7.1.9. Best Practices for Resource Design	107
7.1.9.1. No Redundancy	107
7.1.9.2. No Internal Data	107
7.1.9.3. No Composite Resources	107
7.2. URI Design	108
7.2.1. Introduction	108
7.2.2. Design	109
7.2.3. Recommendations	109
7.2.3.1. URI Template	110
7.2.3.2. Stable URIs	111
7.2.3.3. Nesting Depth	111
7.2.3.4. Maximum Length of URIs	112
7.2.3.5. URLs of Collections Resources	112

7.3.	Representations	112
7.3.1.	Content Negotiation	114
7.3.1.1.	API-side Content Negotiation Mechanism	115
7.3.1.2.	Client-side Content Negotiation Mechanism	116
7.3.1.3.	Negotiating Media-Types	117
7.3.1.4.	Negotiating Language	117
7.3.1.5.	Negotiating Character Encoding	118
7.3.1.6.	Negotiating Content Encoding	118
7.3.2.	Standard Data Formats	118
7.3.2.1.	Date and Time	118
7.3.2.2.	UUID	119
7.3.2.3.	Binary Data	119
7.3.3.	JSON	120
7.3.3.1.	JSON Schema: Defining the Structure of your JSON Objects	121
7.3.3.2.	Conversion between JSON and XML	121
7.3.3.3.	Common JSON Anti Patterns	121
7.3.3.4.	JSONP	122
7.4.	Parameters	124
7.4.1.	Use of Parameter Types	124
7.4.1.1.	Filter- and Sorting	124
7.4.1.2.	Locators	125
7.4.1.3.	Projections	126
7.4.1.4.	Projection on Collection Resources	127
7.4.1.5.	Projection on Instance Resources	127
7.4.1.6.	Metadata	128
7.4.2.	Parameter Types	128
7.4.2.1.	Path Parameters	128
7.4.2.2.	Query Parameters	128
7.4.2.3.	Form Parameters	129
7.4.2.4.	Header Parameters	129

7.5.	Methods	132
7.5.1.	Use of HTTP Methods	133
7.5.1.1.	Retrieve a Resource	133
7.5.1.2.	Create a new Resource	134
7.5.1.3.	Update a Resource	135
7.5.1.4.	Delete a Resource	135
7.5.1.5.	Check Existence of a Resource	136
7.5.1.6.	Determine the Supported Methods	136
7.5.1.7.	Test the Request	136
7.5.2.	Meaning of HTTP Methods	136
7.5.2.1.	GET	136
7.5.2.2.	POST	137
7.5.2.3.	PUT	137
7.5.2.4.	DELETE	138
7.5.2.5.	PATCH	139
7.5.2.6.	HEAD	139
7.5.2.7.	OPTIONS	140
7.5.2.8.	TRACE	140
7.5.2.9.	CONNECT	140
7.5.2.10.	Non-Standard HTTP Methods	141
7.5.3.	Properties of HTTP Methods	141
7.5.3.1.	Safe	141
7.5.3.2.	Idempotent	142
7.6.	Status Codes	142
7.6.1.	Overview of HTTP Status Codes	143
7.6.2.	Redirection	144
7.6.3.	Error Handling	145
7.6.3.1.	Client Errors	145
7.6.3.2.	Server Errors	147
7.6.3.3.	Error Message	148
7.7.	Input and Output Validation	148
7.7.1.	Input Validation	149
7.7.2.	Output Validation	150
7.8.	Consistent Names	150

7.9.	Integratation	152
7.9.1.	Cross-Origin Resource Sharing (CORS)	153
7.9.2.	Browser Explorability	154
7.9.3.	Robustness	154
8.	OpenAPI/Swagger for API Frontend Design	155
8.1.	Introduction	155
8.2.	Root Element	158
8.3.	Resources	159
8.4.	Schema	161
8.5.	Parameters	163
8.6.	Reusable Elements	164
8.7.	Security	165
8.7.1.	Security Definition	165
8.7.2.	Security Binding	167
9.	RAML for API Frontend Design	169
9.1.	Introduction	169
9.2.	Root Element	171
9.3.	Schema	174
9.4.	Parameters	174
9.4.1.	Path Parameters	175
9.4.2.	Query Parameters	176
9.4.3.	Form Parameters	176
9.4.4.	Header Parameters	176
9.5.	Reusable Elements	177
9.5.1.	External Elements: Inclusion of Files	177
9.5.2.	Internal Elements: Definition of Resource Types and Traits	177
9.5.3.	Internal Elements: Usage of Resource Types and Traits	178
9.6.	Security	178

10. API Backend Design Decisions	181
10.1. Backends	182
10.2. Transformations	183
10.2.1. Transformation Source and Target	183
10.2.1.1. Request Transformation	183
10.2.1.2. Response Transformation	184
10.2.2. Transformation Tasks	184
10.2.2.1. Data Structure Transformation	184
10.2.2.2. Representation Transformation	185
10.2.2.3. Conversion between JSON and XML	185
10.2.2.4. Security Mediation	187
10.2.3. Transformation Tools	187
10.3. Dealing with Backend Errors	188
10.3.1. Logging	189
10.3.2. Require a Transaction ID / Request ID / Tracking ID	189
11. Non-Functional Properties of APIs	191
11.1. Security	191
11.1.1. The Appropriate Level of Security	191
11.1.2. Security Concerns	192
11.1.2.1. Authentication	192
11.1.2.2. Authorization	193
11.1.2.3. Delegation	193
11.1.2.4. Identity	193
11.1.2.5. Attacks	194
11.1.2.6. Integrity of API Input and Output	194
11.1.3. Security Mechanisms	194
11.1.3.1. API Keys	194
11.1.3.2. HTTP Basic	195
11.1.3.3. HTTP Digest	196
11.1.3.4. OAuth	198
11.1.3.5. OpenID Connect and JWT	199

11.1.3.6.	Access Restrictions by IP, Location and Time	200
11.1.3.7.	X.509 Transport Layer Security (TLS)	200
11.1.3.8.	Visibility Levels	201
11.1.3.9.	Validation	201
11.1.4.	Best Practice	202
11.1.4.1.	Ensuring Confidentiality and Integrity of Information in URIs	202
11.1.4.2.	Treat All APIs as Public APIs	203
11.1.4.3.	Known Vulnerabilities and Known Attack Patterns	203
11.1.4.4.	Protect All APIs with OAuth by Default	204
11.1.4.5.	CORS	204
11.2.	Performance	204
11.2.1.	Caching	204
11.2.2.	API-Side Caching	205
11.2.3.	Client-Side HTTP Caching	205
11.2.3.1.	Reading Cached Data with Conditional GET	205
11.2.3.2.	Writing Cached Data with Conditional POST, PUT and DELETE	207
11.2.3.3.	Writing Cached Data with Conditional PUT and DELETE	207
11.2.3.4.	Writing Cached Data with Conditional POST	208
11.2.3.5.	Cache Expiration	208
11.2.3.6.	Cache Control	209
11.2.3.7.	Header Parameters for Cache Control	209
11.2.3.8.	Cacheable Responses	210
11.2.4.	Pagination	211
11.2.5.	Use of Traffic Shaping	212

11.2.6.	Enable Content Compression	212
11.2.7.	Remove Whitespace from Responses . . .	212
11.3.	Availability	213
11.3.1.	Traffic Shaping	213
11.3.1.1.	Use Case: Protect API Platform	213
11.3.1.2.	Use Case: Protect Backends . .	214
11.3.1.3.	Use Case: Limit User Access . .	214
11.3.2.	Rate Limitation	214
11.3.2.1.	Implement Rate Limitation . .	214
11.3.2.2.	Optimistic Rate Limitation . . .	215
11.3.3.	Spike Limitation and Spike Smoothing . .	216
11.3.4.	Quota	217
11.3.5.	Caching	217
11.4.	Evolution and Versioning	218
11.4.1.	The Evolution Challenge	218
11.4.2.	Analysis of Evolution	219
11.4.2.1.	Backward Compatible Changes .	219
11.4.2.2.	Forward Compatible Changes . .	220
11.4.2.3.	Incompatible Changes	220
11.4.2.4.	Conclusion of the Analysis . . .	221
11.4.3.	Anticipating and Avoiding Evolution . . .	222
11.4.4.	Evolution from a Methodological Perspec- tive	223
11.4.5.	Coping with Evolution - Versioning . . .	223
11.4.5.1.	HATEOAS Versioning via Links	224
11.4.5.2.	Realize API Versioning in Accept Header	224
11.4.5.3.	Realize API Versioning as URI Path Parameter	225
11.4.5.4.	Realize API Versioning in a Cus- tom HTTP Header	225
11.4.5.5.	Realize API Versioning as Query Parameter	225

11.4.5.6. Realize API Versioning as a new Subdomain	226
11.4.6. Supporting Multiple Versions Simultaneously	226
11.4.7. Anti-Patterns	226
11.4.7.1. Using Token Attributes to Store Application State	226
11.4.7.2. Using Resources to Store Application State	227
12. API Client Design	229
12.1. Designing the Solution	229
12.1.1. Functionality in the Client or in the API?	230
12.1.2. Use an existing API or build a new API?	230
12.1.3. How to choose a third party API?	231
12.1.3.1. Step 1: Find the API	231
12.1.3.2. Step 2: Learn about the API	231
12.1.3.3. Step 3: Test the API	232
12.1.3.4. Step 4: Use the API	233
12.2. Discovering APIs	233
12.2.1. Consumer Discovery	233
12.2.2. Automatic Discovery	233
12.3. Calling APIs	234
12.3.1. Implementing the API Call	234
12.3.2. Content Negotiation	235
12.3.3. Permissive Processing	235
12.3.4. Dealing with Errors	235
A. Appendix	239
A.1. Feedback	239
A.2. About the Author	239
A.3. Other Products by the Author	240
A.3.1. Online Course on RESTful API Design	240
A.3.2. Book on API Architecture	241

A.3.3. Online Course on API Security with OAuth	
2.0	242
A.3.4. Book on API Security with OAuth 2.0 . . .	242

B. HTTP Methods	245
------------------------	------------

C. HTTP Headers	247
------------------------	------------

D. HTTP Status Codes	253
-----------------------------	------------