

The Addison-Wesley Signature Series



PRINCIPLES OF WEB API DESIGN

DELIVERING VALUE WITH
APIs AND MICROSERVICES

JAMES HIGGINBOTHAM



Foreword by
MIKE AMUNDSEN

Praise for *Principles of Web API Design*

“I’ve had the good fortune to work alongside and learn from James over the past several years. His varied institutional knowledge, along with his depth of experience and eye for practical application, makes him unique among his peers. I am ecstatic that others now have the opportunity, in this book, to benefit from James’s compelling, pragmatic vision for how to make better APIs. *Principles of Web API Design* surveys the gamut of available techniques and sets forth a prescriptive, easy-to-follow approach. Teams that apply the guidance in this book will create APIs that better resonate with customers, deliver more business value in less time, and require fewer breaking changes. I cannot recommend *Principles of Web API Design* enough.”

—Matthew Reinbold, Director of API Ecosystems, Postman

“James is one of the preeminent experts on API design in the industry, and this comprehensive guide reflects that. Putting API design in the context of business outcomes and digital capabilities makes this a vital guide for any organization undergoing digital transformation.”

—Matt McLarty, Global Leader of API Strategy at MuleSoft,
a Salesforce company

“In modern software development, APIs end up being both the cause of and solution to many of the problems we face. James’s process for dissecting, analyzing, and designing APIs from concepts to caching creates a repeatable approach for teams to solve more problems than they create.”

—D. Keith Casey, Jr., API Problem Solver, CaseySoftware, LLC

“Following James’s clear and easy-to-follow guide, in one afternoon I was able to apply his process to current real-world use cases. I now have the practical guidance, techniques, and clear examples to help me take those next vital steps. Recommended reading for anyone connected to and working with APIs.”

—Joyce Stack, Architect, Elsevier

“*Principles of Web API Design* uncovers more than principles. In it, you’ll learn a process—a method to design APIs.”

—Arnaud Lauret, API Handyman

“This insightful playbook guides API teams through a structured process that fosters productive collaboration, valuable capability identification, and best-practice contract crafting. James distills years of experience into a pragmatic roadmap for defining and refining API products, and also provides a primer for API security, eventing, resiliency, and microservices alignment. A must-read for architects either new to the API discipline or responsible for onboarding new teams and instituting a structured API definition process.”

—Chris Haddad, Chief Architect, Karux LLC

Principles of Web API Design: Delivering Value with APIs and Microservices

Table of Contents

Cover
Half Title
Title Page
Copyright Page
Contents
Series Editor Foreword
Foreword
Preface
Acknowledgments
About the Author
Part I: Introduction to Web API Design
Chapter 1: The Principles of API Design
The Elements of Web API Design
Business Capabilities
Product Thinking
Developer Experience
API Design Is Communication
Reviewing the Principles of Software Design
Modularization
Encapsulation
High Cohesion and Loose Coupling
Resource-Based API Design
Resources Are Not Data Models

Table of Contents

Resources Are Not Object or Domain Models

Resource-Based APIs Exchange Messages

The Principles of Web API Design

Summary

Chapter 2: Collaborative API Design

Why an API Design Process?

API Design Process Antipatterns

The Leaky Abstraction Antipattern

The Next Release Design Fix Antipattern

The Heroic Design Effort Antipattern

The Unused API Antipattern

The API Design-First Approach

Remaining Agile with API Design-First

The Agile Manifesto Revisited

The Agility of API Design-First

The Align-Define-Design-Refine Process

The Role of DDD in API Design

API Design Involves Everyone

Applying the Process Effectively

Summary

Part II: Aligning on API Outcomes

Chapter 3: Identify Digital Capabilities

Ensuring Stakeholder Alignment

What Are Digital Capabilities?

Focusing on the Jobs to Be Done

What Are Job Stories?

The Components of a Job Story

Writing Job Stories for APIs

Method 1: When the Problem Is Known

Method 2: When the Desired Outcome Is Known

Table of Contents

Method 3: When the Digital Capability Has Been Identified

Overcoming Job Story Challenges

Challenge 1: Job Stories Are Too Detailed

Challenge 2: Job Stories Are Feature Centric

Challenge 3: Additional User Context Is Needed

Techniques for Capturing Job Stories

A Real-World API Design Project

Job Story Examples

Summary

Chapter 4: Capture Activities and Steps

Extending Job Stories into Activities and Steps

Identify the Activities for Each Job Story

Decompose Each Activity into Steps

What If Requirements Aren't Clear?

Using EventStorming for Collaborative Understanding

How EventStorming Works

Step 1: Identify Business Domain Events

Step 2: Create an Event Narrative

Step 3: Review the Narrative and Identify Gaps

Step 4: Expand Domain Understanding

Step 5: Review the Final Narrative

The Benefits of EventStorming

Who Should Be Involved?

Facilitating an EventStorming Session

Prepare: Gathering Necessary Supplies

Share: Communicating the EventStorming Session

Execute: Conducting the EventStorming Session

Wrap-up: Capture Activities and Activity Steps

Follow-up: Post-Session Recommendations

Customizing the Process

Summary

Part III: Defining Candidate APIs

Table of Contents

Chapter 5: Identifying API Boundaries

Avoiding API Boundary Antipatterns

The Mega All-in-One API Antipattern

The Overloaded API Antipattern

The Helper API Antipattern

Bounded Contexts, Subdomains, and APIs

Finding API Boundaries Using EventStorming

Finding API Boundaries through Activities

Naming and Scoping APIs

Summary

Chapter 6: API Modeling

What Is API Modeling?

The API Profile Structure

The API Modeling Process

Step 1: Capture API Profile Summary

Step 2: Identify the Resources

Step 3: Define the Resource Taxonomy

Step 4: Add Operation Events

Step 5: Expand Operation Details

Validating the API Model with Sequence Diagrams

Evaluating API Priority and Reuse

Summary

Part IV: Designing APIs

Chapter 7: REST-Based API Design

What Is a REST-Based API?

REST Is Client/Server

REST Is Resource-Centric

REST Is Message Based

REST Supports a Layered System

REST Supports Code on Demand

Hypermedia Controls

When to Choose REST

Table of Contents

REST API Design Process

- Step 1: Design Resource URL Paths
- Step 2: Map API Operations to HTTP Methods
- Step 3: Assign Response Codes
- Step 4: Documenting the REST API Design
- Step 5: Share and Gather Feedback

Selecting a Representation Format

- Resource Serialization
- Hypermedia Serialization
- Hypermedia Messaging
- Semantic Hypermedia Messaging

Common REST Design Patterns

- Create-Read-Update-Delete
- Extended Resource Lifecycle Support
- Singleton Resources
- Background (Queued) Jobs
- Long-Running Transaction Support in REST

Summary

Chapter 8: RPC and Query-Based API Design

What Is an RPC-Based API?

- The gRPC Protocol
- Factors When Considering RPC

RPC API Design Process

- Step 1: Identify RPC Operations
- Step 2: Detail RPC Operations
- Step 3: Document the API Design

What Is a Query-Based API?

- Understanding OData
- Exploring GraphQL

Query-Based API Design Process

- Step 1: Designing Resource and Graph Structures
- Step 2: Design Query and Mutation Operations
- Step 3: Document the API Design

Table of Contents

Summary

Chapter 9: Async APIs for Eventing and Streaming

The Problem with API Polling

Async APIs Create New Possibilities

A Review of Messaging Fundamentals

Messaging Styles and Locality

The Elements of a Message

Understanding Messaging Brokers

Point-to-Point Message Distribution (Queues)

Fanout Message Distribution (Topics)

Message Streaming Fundamentals

Async API Styles

Server Notification Using Webhooks

Server Push Using Server-Sent Events

Bidirectional Notification via WebSocket

gRPC Streaming

Selecting an Async API Style

Designing Async APIs

Command Messages

Event Notifications

Event-Carried State Transfer Events

Event Batching

Event Ordering

Documenting Async APIs

Summary

Part V: Refining the API Design

Chapter 10: From APIs to Microservices

What Are Microservices?

Microservices Reduce Coordination Costs

The Difference between APIs and Microservices

Weighing the Complexity of Microservices

Table of Contents

- Self-Service Infrastructure
- Independent Release Cycles
- Shift to Single-Team Ownership
- Organizational Structure and Cultural Impacts
- Shift in Data Ownership
- Distributed Data Management and Governance
- Distributed Systems Challenges
- Resiliency, Failover, and Distributed Transactions
- Refactoring and Code Sharing Challenges

Synchronous and Asynchronous Microservices

Microservice Architecture Styles

- Direct Service Communication
- API-Based Orchestration
- Cell-Based Architecture

Right-Sizing Microservices

Decomposing APIs into Microservices

- Step 1: Identify Candidate Microservices
- Step 2: Add Microservices into API Sequence Diagrams
- Step 3: Capture Using the Microservice Design Canvas
- Additional Microservice Design Considerations

Considerations When Transitioning to Microservices

Summary

Chapter 11: Improving the Developer Experience

Creating a Mock API Implementation

- Static API Mocking
- API Prototype Mocking
- README-Based Mocking

Providing Helper Libraries and SDKs

- Options for Offering Helper Libraries
- Versioning Helper Libraries
- Helper Library Documentation and Testing

Offering CLIs for APIs

Table of Contents

Summary

Chapter 12: API Testing Strategies

Acceptance Testing

Automated Security Testing

Operational Monitoring

API Contract Testing

Selecting Tools to Accelerate Testing

The Challenges of API Testing

Make API Testing Essential

Summary

Chapter 13: Document the API Design

The Importance of API Documentation

API Description Formats

OpenAPI Specification

API Blueprint

RAML

JSON Schema

API Profiles Using ALPS

Improving API Discovery Using APIs.json

Extending Docs with Code Examples

Write Getting Started Code Examples First

Expanding Documentation with Workflow Examples

Error Case and Production-Ready Examples

From Reference Docs to a Developer Portal

Increasing API Adoption through Developer Portals

Elements of a Great Developer Portal

Effective API Documentation

Question 1: How Does Your API Solve My Problems?

Question 2: What Problem Does Each API Operation Support?

Question 3: How Do I Get Started Using the API?

The Role of Technical Writer in API Docs

The Minimum Viable Portal

Table of Contents

Phase 1: Minimum Viable Portal

Phase 2: Improvement

Phase 3: Focusing on Growth

Tools and Frameworks for Developer Portals

Summary

Chapter 14: Designing for Change

The Impact of Change on Existing APIs

Perform an API Design Gap Analysis

Determine What Is Best for API Consumers

Change Management Is Built on Trust

Strategies for Change

API Versioning Strategies

Common Nonbreaking Changes

Incompatible Changes

API Versions and Revisions

API Versioning Methods

Business Considerations of API Versioning

Deprecating APIs

Establish a Deprecation Policy

Announcing a Deprecation

Establishing an API Stability Contract

Summary

Chapter 15: Protecting APIs

The Potential for API Mischief

Essential API Protection Practices

Components of API Protection

API Gateways

API Management

Service Meshes

Web Application Firewalls

Content Delivery Networks

Intelligent API Protection

Table of Contents

API Gateway Topologies

- API Management Hosting Options
- API Network Traffic Considerations
- Topology 1: API Gateway Direct to API Server
- Topology 2: API Gateway Routing to Services
- Topology 3: Multiple API Gateway Instances

Identity and Access Management

- Passwords and API Keys
- API Tokens
- Pass-by-Reference versus Pass-by-Value API Tokens
- OAuth 2.0 and OpenID Connect

Considerations before Building an In-House API Gateway

- Reason 1: API Security Is a Moving Target
- Reason 2: It Will Take Longer than Expected
- Reason 3: Expected Performance Takes Time
- What about Helper Libraries?

Summary

Chapter 16: Continuing the API Design Journey

Establishing an API Style Guide

- Methods for Encouraging Style Guide Adherence
- Selecting Style Guide Tone
- Tips for Getting Started with an API Style Guide
- Supporting Multiple API Styles

Conducting API Design Reviews

- Start with a Documentation Review
- Check for Standards and Design Consistency
- Review Automated Test Coverage
- Add Try It Out Support

Developing a Culture of Reuse

The Journey Has Only Begun

Appendix: HTTP Primer

Index

Table of Contents