RESTful APIs guidelines

A Zalando showcase

Lampros Papadimitriou

December 21, 2018

Check24 Baufinanzierung GmbH

The soft part

About Zalando

- Split in verticals
- Teams serving other teams
- Agile project structures
- Strong appearance in Github
- Techblogs

Principles

The Robustness Principle

Be liberal in what you accept, be conservative in what you send

- Treat you API as a product
- Be customer-oriented
- Act as a product owner
- Actively improve and maintain API consistency over long term
- Simple, comprehensive and usable API

API first!

- Define API first!
- Follow standards
- Early review and feedback
- Free yourself from *HOW* concerns

Example

- PO receives requirement
- Team designs API draft
- Lint it (internal tool)
- OP communicates draft with client(s)
- Review (internal tool), repeat
- Teams proceed with implementation
- Publish in API Portal (internal)

OpenAPI

- *Must* provide API specification in YAML
- *Should* provide API user manual online
- *Must* provide metainformation (title, description, team, audience)
- *Must* use semantic version

Security

- *Must* secure endpoints with OAuth 2.0
- *Must* design and assign permissions

Versioning

- *Must* never break backwards compatibility
- *Should* avoid versioning (MUST avoid URL versioning)
- *Should* prefer compatible extensions
- Clients *should* not crash on compatible API extensions
- *Must* use media type versioning
- *Must* obtain approval of clients (e.g. deprecation)

Example

Accept: application/x.zalando.cart+json;version=2

The hard part

API design

- *Must* Use functional name schema
- *Must* always return JSON objects as top-level
- *Must* property names be ASCII *snake_case* (and never camelCase)
- *Should* use additionalProperties
- *Should* use enumeration values as String
- *Should* use standards for Date, duration, currency, country, language

API naming

- *Must* Use lowercase separate words with hyphens for Path Segments
- *Must* Use snake_case for query parameters
- *Must* Pluralize resource names
- *Must* avoid trailing slashes
- *Must* stick to conventional query params
- *Must* avoid actions - think about resources (lvl 2)
- *Must* keep URLs verb-free
- *Must* identify resources and sub via path
- *Must* booleans not be null
- *May* consider not nested URLs

HTTP requests

- GET for read
- PUT to update entire resources (replace)
- POST to create single resources
- PATCH update parts of single resources
- DELETE to delete resources
- HEAD to retrieve header infomation (Etag?)
- Prefer POST over PUT
- Use ETag & If-(None)-Match

Http status codes and errors

- Use standard HTTP codes
- 201 CREATED (sync) vs 202 ACCEPTED (async)
- 207 (multi-status)
- 409 CONFLICT (concurrency problem)
- 412 PRECONDITION FAILED (If-Match mismatch)
- 423 LOCKED Pessimistic locking
- 429 Too many requests

Performance tips

- *Should* Use gzip compression
- *Should* reduce bandwidth needs and improve responsiveness
- *Should* support filtering fields
- *Should* support pagination
- *Should* allow optional embedding
- MUST Document caching, if supported (default: Cache-Control: no-cache)

Common field names

- id
- xyz_id
- created
- modified
- type
- etag

Proprietary Headers

- X-Flow-ID (troubleshooting)
- X-Frontend-Type (mobile-app / browser)
- X-Device-Type (tablet, desktop)
- X-Device-OS (IOS, Android)

Useful tools

- swagger online editor
- connexion
- problem / problem-spring-web
- jackson-datatype-money
- intellij-swagger

Questions?