THE MAKING OF A GREAT API

ROB ALLEN
NINETEEN FEET LTD

SLIDES: HTTPS://AKRABAT.COM/6424

@akrabat
FIT FOR PURPOSE
FEATURES OF A GOOD API

• Malleability
• Correctness
• Error handling
• Documentation
• Security
A GOOD API IS MALLEABLE
DECouple Your Representation

- Database
- Entity
- Application logic
- Resource
  - Properties
  - Links
  - Forms
  - Queries
- Representation

{ ... }
HYPERMEDIA

- An explorable API
- Rename end points at will
- Re-home end points on different servers
HYPERMEDIA

{
    "_links": {
        "self": { "href": "https://api.example.com/orders/523" },
        "customer": { "href": "https://api.example.com/customers/32" },
        "invoice": { "href": "https://api.example.com/invoices/873" },
        "shippingNote": {
            "href": "https://cdn.example.com/ad8wd4.pdf",
            "type": "application/pdf"
        }
    },
    "currency": "GBP",
    "status": "shipped",
    "total": 123.45
}

@akrabat
A GOOD API IS CORRECT
EMBRACE HTTP VERBS

<table>
<thead>
<tr>
<th>METHOD</th>
<th>USED FOR</th>
<th>IDEMPOTENT?</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>Retrieve data</td>
<td>Yes</td>
</tr>
<tr>
<td>PUT</td>
<td>Change data</td>
<td>Yes</td>
</tr>
<tr>
<td>DELETE</td>
<td>Delete data</td>
<td>Yes</td>
</tr>
<tr>
<td>PATCH</td>
<td>Update data</td>
<td>No</td>
</tr>
<tr>
<td>POST</td>
<td>Change data</td>
<td>No</td>
</tr>
</tbody>
</table>
STATUS CODES MATTER

1xx    Informational
2xx    Success
3xx    Redirection
4xx    Client error
5xx    Server error
MEDIA TYPES

- Read **Content-Type** header to decode incoming data
- Honour the **Accept** header when sending data
HANDLING CHANGES

• Avoid major new versions
• Make changes backwards-compatible
• Think about forwards-compatibility
A NEW VERSION IS A NEW API

- Separate code
- Domain or URL path is fine
  - api.example.com/v2/user
  - api2.example.com/user
- Use Server header for minor and patch info
A GOOD API HAS GREAT ERRORS
GREAT ERROR HANDLING

• Error representations are first class citizens
• Code for computers; messages for humans
• Pretty print for the humans!
HTTP PROBLEM (RFC 7807)

HTTP/1.1 503 Service Unavailable
Content-Type: application/problem+json
Content-Language: en

{
    "status": 503,
    "type": "https://example.com/service-unavailable",
    "title": "Could not authorise user due to an internal problem.",
    "detail": "The authentication service is down for maintenance.",
    "instance": "https://example.com/maintenance-schedule/2017-06",
    "error_code": "AUTHSERVICE_UNAVAILBLE"
}
A GOOD API IS DOCUMENTED
PROFILE LINKS (RFC 6906)

Header:

   Link: <https://www.example.com/docs>;rel="profile"

Body:

   {
      "_links": {
         "profile": { "href": "https://www.example.com/docs/" }
      }
   }
HUMAN DOCUMENTATION

- Tutorials
- Reference
OPENAPI SPECIFICATION

• Spec-First API Design
• Tooling: https://openapi.tools
A GOOD API IS SECURE
OAUTH 2

- Application identification
- User identification
- User’s trust relationship is with the API
RATE LIMIT

HTTP/1.1 429 Too Many Requests
Content-Type: application/problem+json
X-RateLimit-Limit: 5000
X-RateLimit-Remaining: 0
X-RateLimit-Reset: 1471549573

{
  "status": 429,
  "type": "https://dev.example.com/rate_limits",
  "title": "API rate limited exceeded.",
  "error_code": "RATE_LIMIT_EXCEEDED"
}
TO SUM UP
THANK YOU

HTTPS://JOIND.IN/TALK/4CFB9