API Design Guidelines

Mike Kistler & Dan Hudlow
IBM Cloud Developer Experience

Why you need API Design Guidelines

- Consistency in your API design
  - will benefit your users
  - will benefit your service/library/tool developers

Guiding Principles

- Usefulness
- Adherence to HTTP semantics
- Ease of use and low barrier to entry
- Defensiveness/Compatibility
- Security
- Longevity

References

- Microsoft API Guidelines

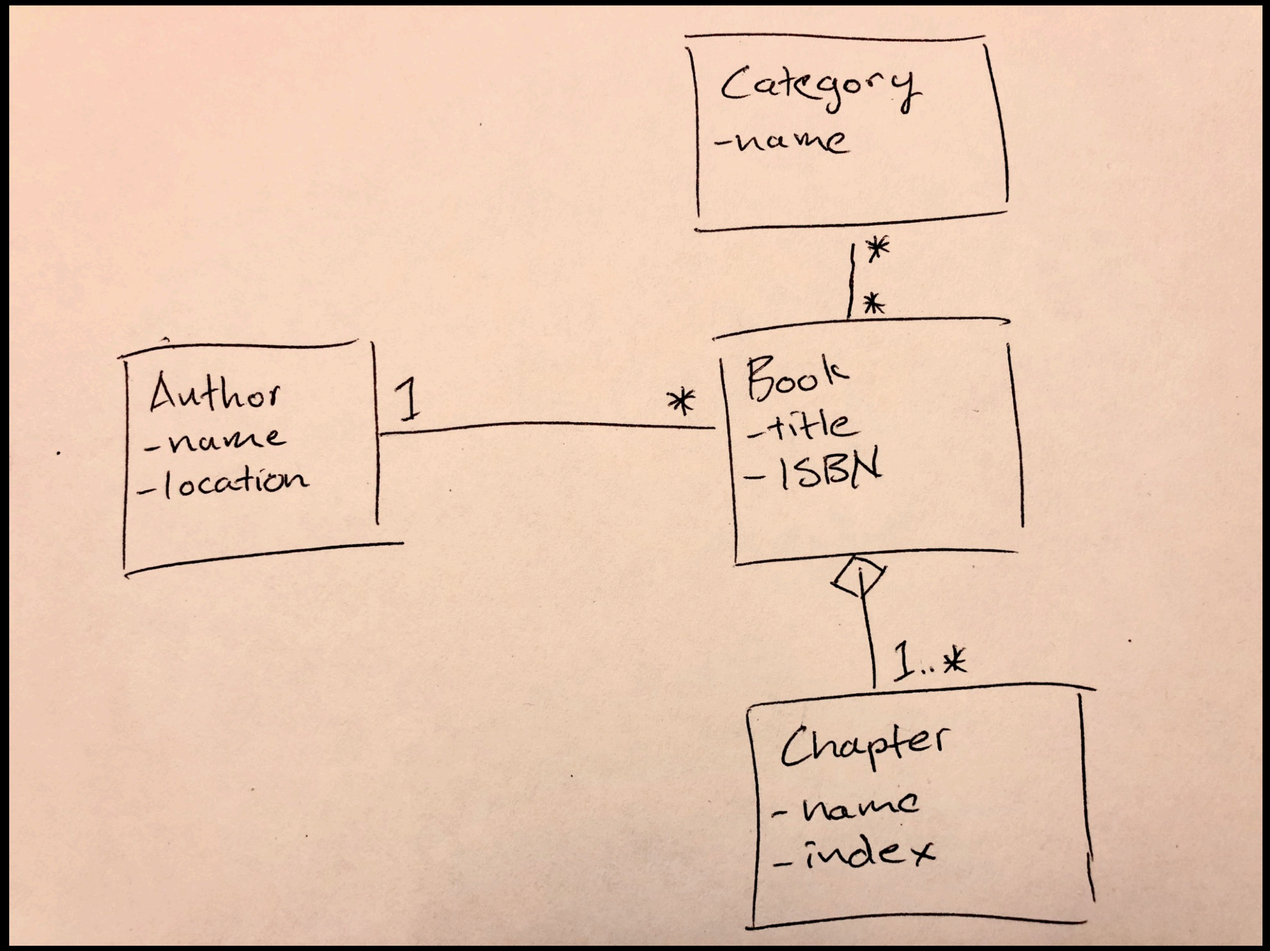
  https://github.com/microsoft/api-guidelines/blob/vNext/Guidelines.md
- Google API Guidelines

  https://cloud.google.com/apis/design/
- API Stylebook

  http://apistylebook.com/design/guidelines/

Design First

- User stories
- ERDs
- and UML



Specification Format

- OpenAPI

  http://spec.openapis.org/oas/v3.0.2

  or

  https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.2.md
- Linux Foundation project (Open)
- Specifically OpenAPI v3.x
  - published July 2017

Resource-oriented API design

- API consists of nouns — resources — and verbs — operations
- Final static segment in API path is the resource name
  - Always plural
- Resource instances have a unique ID
- Resource has a well defined schema of its contents

Example Resource

/authors/18345


```
{
  “id”: 18345,
  “first_name”: “Scott”,
  “last_name”: “Thompson”,
  “city”: “Dallas”,
  “region”: “Texas”,
  “country”: “United States”,
  “tags”: [“Objective-C”, “Swift”, “Ruby”],
  “href”: “https://api.hudlow.org/authors/18345”
}
```

Resource Format

- JSON
  - JSON is *unordered*
  - JSON names are *case-sensitive*
- But not *any* JSON
  - Use well-defined types
  - Don’t mix types within an array
  - Beware of "null"
- Create clear guidelines about what is/is not allowed

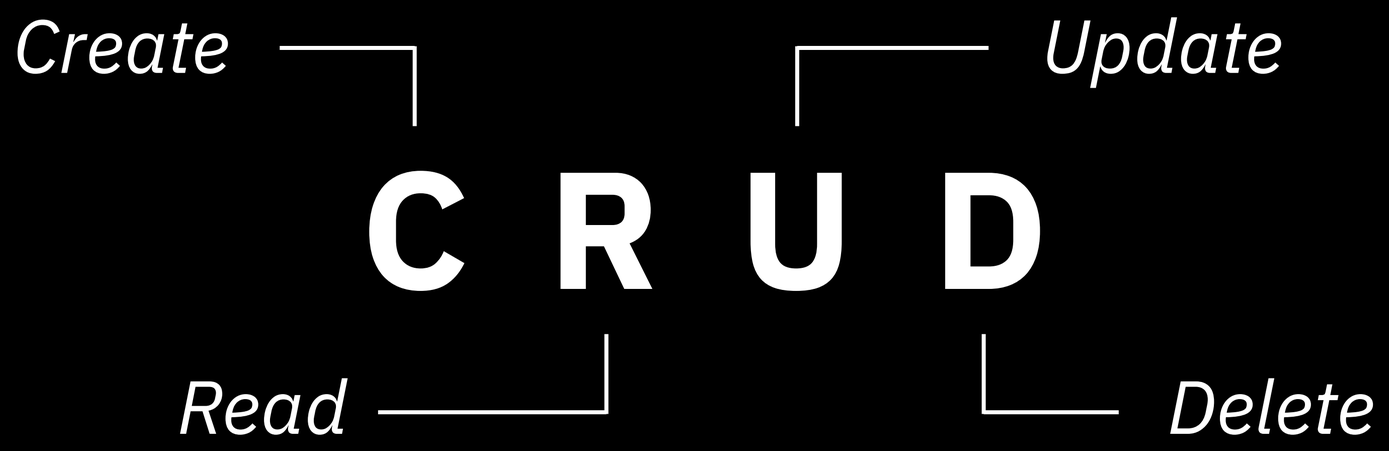
Naming Conventions

- `snake_case`
- `camelCase`
- `UpperCamelCase`
- `Kebab-case`

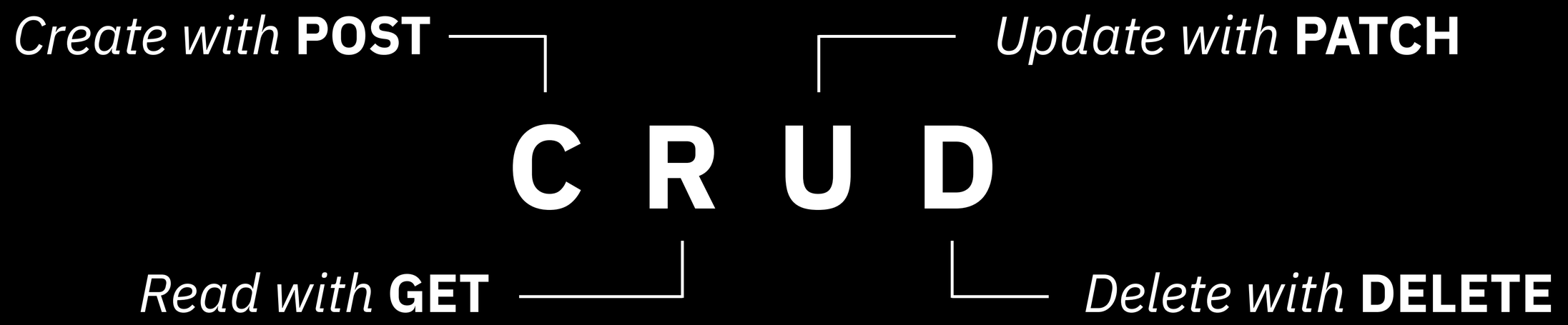
Naming Conventions

- `snake_case`
  - `Our choice for parameter and property names`
- `camelCase`
  - `Our choice for operation ids`
- `UpperCamelCase`
  - `Our choice for schema names`
- `kebab-case`

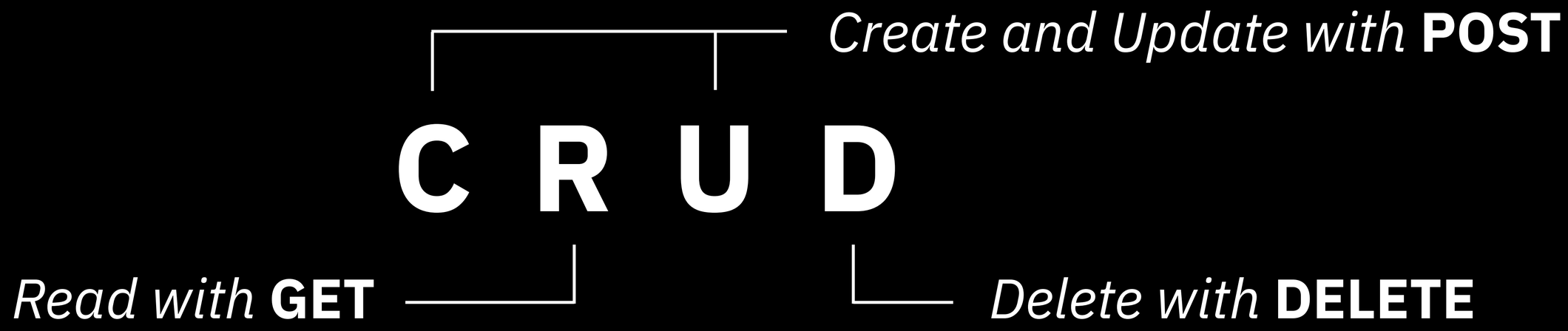
Operations



Operations


Create with **POST**
Update with **PATCH**
C R U D
Read with **GET**
Delete with **DELETE**

Operations


Create and Update with POST
C R U D
Read with GET
Delete with DELETE

HTTP Methods

GET & HEAD

- Safe
- Idempotent
- Ignore request bodies

POST

- Unsafe
- Non-idempotent
- Used for creation of subordinate resources:

  `POST /books → /books/38573`

- May also be used for modifying an existing resource:

  `POST /books/38573`

PATCH

- Unsafe
- Non-idempotent
- Modify an existing resource

PUT

- Unsafe
- Idempotent
- Used for creating or replacing a resource at a known URL

Standard Error Model

```
400 Bad Request

{
  “code”: “missing_field”,
  “message”: “The `first_name` field is needed to create an author.”,
  “target”: {
    “type”: “field”,
    “name”: “first_name”
  }
}
```

Programmatic Information in Errors

```
400 Bad Request

{
  “code”: “missing_field”,
  “message”: “The `first_name` field is needed to create an author.”,
  “target”: {
    “type”: “field”,
    “name”: “first_name”
  }
}
```

Collections

```
GET /authors

{
  “authors”: [
    {
      “id”: 18345,
      “first_name”: “Scott”,
      “last_name”: “Thompson”,
      “href”: “https://api.hudlow.org/authors/18345”
    },
    {
      “id”: 63840,
      “first_name”: “David”,
      “last_name”: “Gelphman”,
      “href”: “https://api.hudlow.org/authors/63840”
    }
  ]
}
```

Offset & Limit Pagination

GET /authors?***offset=4***&***limit=2***

```
{
  “total_count”: 12
  “authors”: [
    {
      “id”: 18345,
      “first_name”: “Scott”,
      “last_name”: “Thompson”,
      “href”: “https://api.hudlow.org/authors/18345”
    },
    {
      “id”: 63840,
      “first_name”: “David”,
      “last_name”: “Gelphman”,
      “href”: “https://api.hudlow.org/authors/63840”
    }
  ]
}
```

Token-based Pagination

```
GET /authors

{
  “total_count”: 12
  “next”: {
    “token”: “d537748fe4”
  },
  “authors”: [
    {
      “id”: 18345,
      “first_name”: “Scott”,
      “last_name”: “Thompson”,
      “href”: “https://api.hudlow.org/authors/18345”
    },
    {
      “id”: 63840,
      “first_name”: “David”,
      “last_name”: “Gelphman”,
      “href”: “https://api.hudlow.org/authors/63840”
    }
  ]
}
```

Pagination

- Offset and limit pagination
  - Stateless
  - Imprecise
- Token-based pagination
  - Robust
  - More difficult and demanding to implement
  - Stateful vs stateless considerations

Versioning

Compatibility

- **Usually compatible**
  - Adding new fields to models
  - Adding new types of resources at new URLs
- **Usually incompatible**
  - Removing resource types
  - Removing fields from resources
  - Adding new required fields to templates
  - Making previously valid field values invalid

Specifying Versions

- Custom header

  ```
  API-Version: 3
  ```

- Query parameter

  ```
  /books?api_version=3
  ```

- Content type

  ```
  Accept: application/vnd.github.v3+json
  ```

- Root path

  ```
  /v3/books
  ```

Go Forth

- Define your principles and priorities
- Create purposeful guidelines
- Write them down
- Start designing better web APIs