Best Practices for designing REST APIs

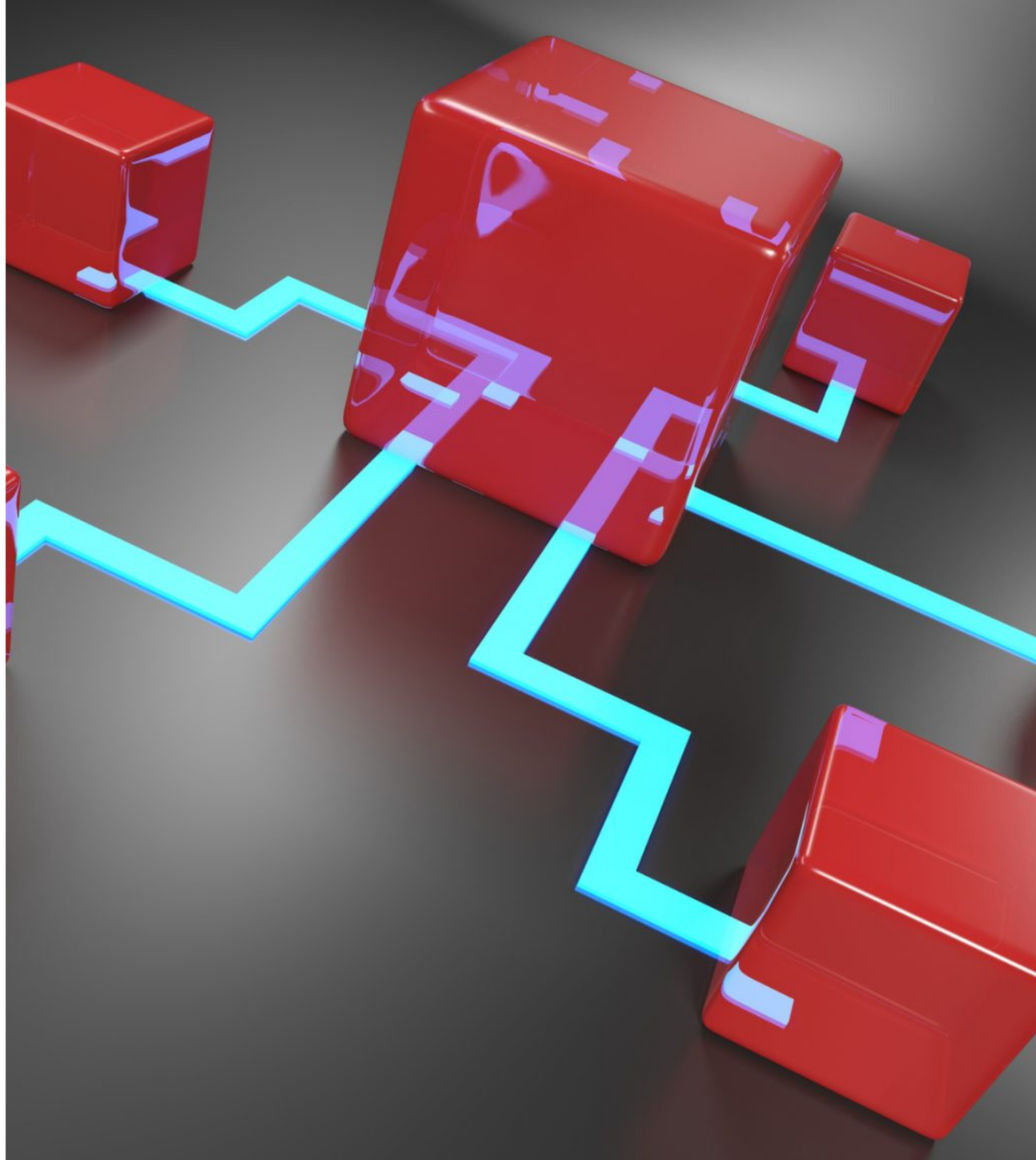
Let’s explore and see them in action too!

AB Satyaprakash - 03 June 2022

Follow @ABSatyaprakash

Best practices

1. Accept and respond with JSON
2. Use nouns instead of verbs in endpoint paths
3. Name collections with plural nouns
4. Nesting resources for hierarchical objects
5. Handle errors gracefully and return standard error codes
6. Allow filtering, sorting, and pagination
7. Maintain Good Security Practices
8. Cache data to improve performance
9. Versioning our APIs



1. Accept and respond with JSON

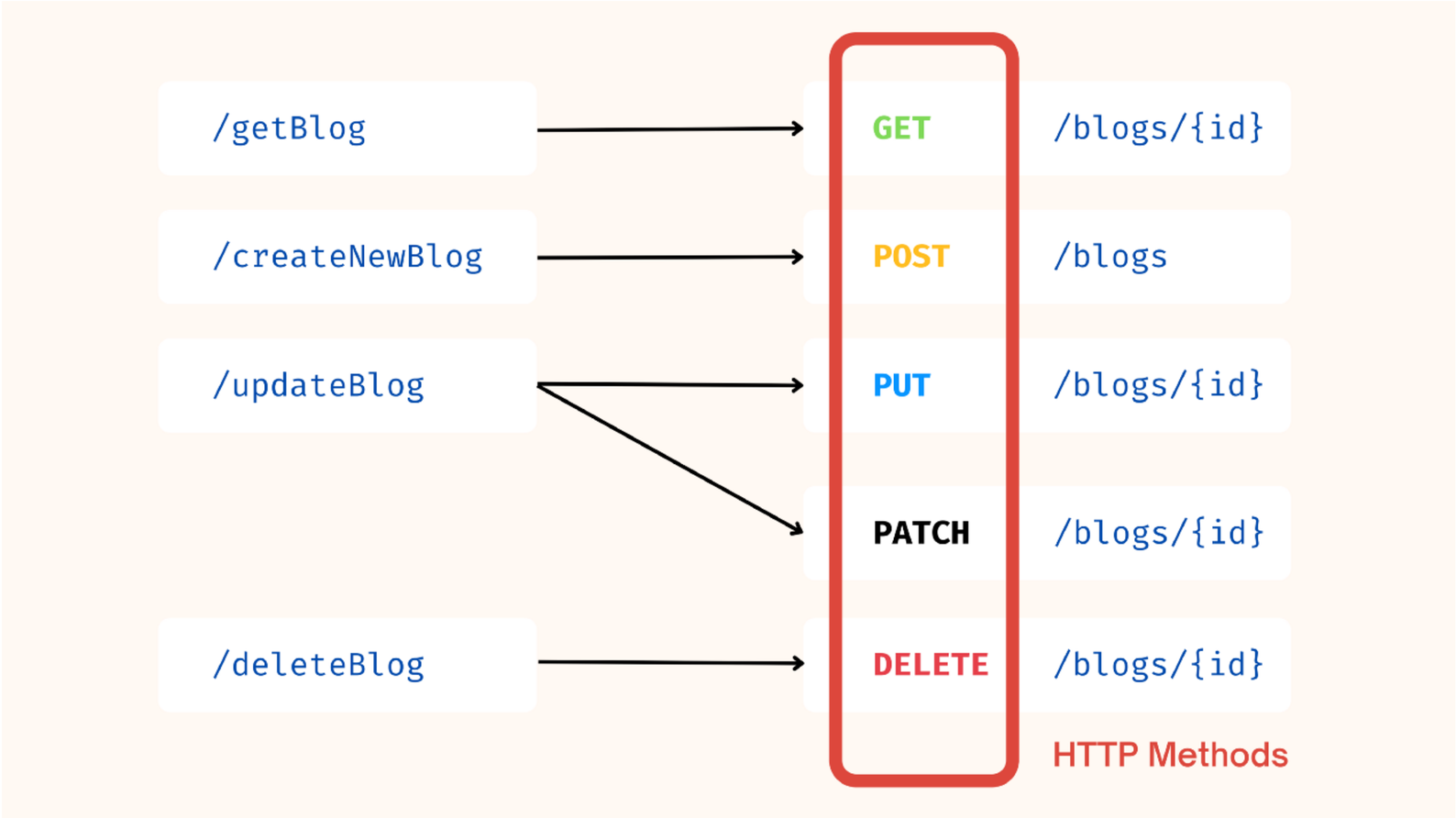
- JSON (JavaScript Object Notation) is a lightweight format for storing and transporting data.
- REST APIs should accept JSON for request payload and also send responses to JSON.
- XML isn’t widely supported by frameworks. Form data is good to send and receive files.

2. Use nouns instead of verbs in endpoint paths

- Adding verbs to endpoints make them unnecessarily long.
- Based on developers whims, the GET request might look like ‘retrieve’ or ‘fetch’ - no consistency and not recommended.
- Our HTTP request methods already have the verbs.
- GET retrieves resources. POST submits new data to the server. PUT updates existing data. DELETE removes data. These form the CRUD.

3. Name collections with plural nouns

- It is customary to name the collections (or resources) with plural nouns.
- Along with the last rule, this is how endpoints should look!



4. Nesting resources for hierarchical objects

- When designing endpoints, it makes sense to group those that contain associated information.
- For getting comments of a particular article, we need to use -
  - GET */articles/:articleId/comments*

```
[
    {
        "id": 01,
        "title": "REST API - Best Practices",
        "author": "AB Satyaprakash",
        "comments":[
          {
                "id": 01,
                "author": "Amit",
                "content": "Such a nice article. Wow!"
          },
          {
                "id": 02,
                "author": "Jacob",
                "content": "Maybe try modifying point 4 to somehting more reasonable!"
          }
        ]
    },
    {
        "id": 02,
        "title": "10 Best Apps for MacOS",
        "author": "AB Satyaprakash",
        "comments":[
          {
                "id": 02,
                "author": "Jeremy",
                "content": "I never heard about PQR application! Thanks"
          }
        ]
    },
]
```

5. Handle errors gracefully & return standard error codes

- HTTP response codes - eliminate API user confusion, give maintainers information.
- Codes should be accompanied with error messages.
- Standard HTTP status codes are - 400 Bad Request, 401 Unauthorised, 403 Forbidden, 404 Not Found, 500 Internal server error, 502 Bad Gateway, 503 Service Unavailable.

6. Allow filtering, sorting, and pagination

- Filter, pagination, and sorting - improve performance since databases behind a REST API can get very large.
- Filter - all data not needed at once.
- Paginate - return few results at a time.
- Sort - reorder data as per requirement.

7. Maintain Good Security Practices

- SSL/TLS for security is a must. SSL certificate - free or nominal cost.
- Client should receive `only` request information and nothing more.
- Authorisation needed for accessing private resources.
- Granular roles for each user to access a set of features. Enforce principle of least privilege.
- Admin should have control over data and users (roles, access, etc.).

8. Cache data to improve performance

- Query local memory cache instead of DB to improve performance.
- Handle outdated data - causes problem otherwise.
- Caching solutions - Redis, in-memory caching, etc. Express has `apicache` middleware to add caching to our app without much configuration.

9. Versioning our APIs

- We should have different versions of API if we're making any changes to them that may break clients.
- Versioning is usually done with /v1/, /v2/ etc. added at the start of the API path.
- The `v1` endpoint - works for people who don't want to move on yet.
- The `v2` endpoint - works for new users with new features.

Thank you!

Follow @ABSatyaprakash