govtech stack

REST API Design Best Practice

govtech stack

What is REST?

REST (Representational State Transfer) is an architectural style proposed by Roy Fielding.

RESTful API

Roy defined 6 key constraints

- Client-Server:
  - This constraint operates on the concept that the client and the server should be separate from each other and allowed to evolve individually.

- Stateless:
  - REST APIs are stateless, meaning that calls can be made independently of one another, and each call contains all of the data necessary to complete itself successfully.

- Cache:
  - Because a stateless API can increase request overhead by handling large loads of incoming and outbound calls, a REST API should be designed to encourage the storage of cacheable data.

RESTful API continue...

Roy defined 6 key constraints

- Uniform Interface:
  - The key to the decoupling client from server is having a uniform interface that allows independent evolution of the application without having the application's services, or models and actions, tightly coupled to the API layer itself.

- Layered System:
  - REST APIs have different layers of their architecture working together to build a hierarchy that helps create a more scalable and modular application.

- Code on Demand:
  - Code on Demand allows for code or applets to be transmitted via the API for use within the application.

govtech stack

Topics

- API Security
- HATEOAS
- Resources Naming
- URI Design
- Actions Design
- Version Control
- Query Design
- HTTP Response Code

Basic API Security

- Accept connections over HTTPS only.
- Should not respond over plain HTTP request.
- Should not redirect from HTTP to HTTPS.

govtech stack

What is HATEOAS?

Hypermedia **A**s **T**he **E**ngine **O**f **A**pplication **S**tate

govtech stack

An Example of HATEOAS Implementation...

```
{
    "links" : {
        "hotels" : {
            "collection" : { "href" : "/hotels{?q&filter&sort}" },
            "item" : { "href" : "/hotels/{id}" }
        },
        "hotelsReviews" : {
            "collection" : { "href" : "/hotels/{id}/reviews{?q&filter&sort}" },
            "item" : { "href" : "/reviews/{id}" }
        },
        "hotelsPhotos" : {
            "collection" : { "href" : "/hotels/{id}/reviews{?q&filter&sort}" },
            "item" : { "href" : "/reviews/{id}" }
        },
        "users" : {
            "collection" : { "href" : "/users/{?q&filter&sort}" },
            "item" : { "href" : "/users/{id}" }
        }
    }
}
```

```
{
    "id" : 2231,
    "name" : "Omni Hotel San Francisco",
    "lat" : 123.456,
    "lng" : 654.321,

    "links" : {
        "self" : { "href" : "/hotels/2231" },
        "reviews" : { "href" : "/hotels/2231/reviews" },
        "manager" : { "href" : "users/8828" },
        "photos" : { "href" : "/hotels/2231/photos" }
    }
}
```

govtech stack

Should you implement **HATEOAS** for your REST API?

Up to YOU…

govtech stack

Resource Name Design

Should Resource Name be Singular or Plural?

- Keep the URL format consistent and always use a Plural.

- Avoid having to deal with odd pluralisation like person/ people, goose/geese

URI Designs

URI Designs Rules

- Forward slash separator (/) must be used to indicate a hierarchical relationship.
- A trailing forward slash (/) should not be included in URIs.
- Underscores (_) should not be used in URIs.
- Hyphens (-) should be used to improve the readability of URIs.
- Lowercase or camelCase letters should be preferred in URI paths.
- CRUD function names should not be used in URIs.

URI Template

- https://[API Service Host]/[Context Path]/[Version]/[Resources]?[Parameters]

- API Service Host: host.api.sg
- Context Path: agency/service
- Version: v1
- Resource: users

- https://host.api.sg/agency/service/v1/users?state=active

REST Operations Designs

govtech stack

REST Operations Designs

HTTP Verbs for Actions against Resource

- **GET /hotels** - retrieves a list of hotels
- **GET /hotels/12** - retrieves a specific hotel
- **POST /hotels** - creates a new hotel
- **PUT /hotels/12** - updates hotel #12
- **PATCH /hotels/12** - partially updates hotel #12
- **DELETE /hotels/12** - deletes hotel #12

API Versioning Designs

govtech stack

API Versioning Designs

API Version Control

Changes that don't require a New Version

- New resources.
- New HTTP methods on existing resources.
- New data formats.
- New attributes or elements on existing data types.

Change that require a New Version

- Removed or renamed URIs.
- Different data returned for same URI.
- Removal of support for HTTP methods on existing URIs.

How to Implement API Versioning?

- **Software Versions**

  Version Template: major.minor.patch.build

  Major – breaking changes

  Minor – non-breaking changes

- Only use major version in publish API.

  GET **v1**/resources?...

  GET **v2**/resources?...

URI Query Designs

govtech stack

URI Query Designs

Result Filtering, Sorting & Searching

- Filtering:
  - GET /tickets?state=open&priority=high
  - Retrieve tickets with state is **Open** and **Priority** set to high.
- Sorting:
  - GET /tickets?sort=-priority,createdDatetime
  - Sort by **descending** order on **Priority** and than by **CreatedDatetime** by ascending order.
- Searching:
  - i.e. /tickets?q=error
  - Retrieve the highest priority open tickets mentioning the word ’**error**’.

govtech stack

URI Query Designs

Limiting the return Fields

- API consumer doesn't always need the full representation of a resource.

- Allow API Consumer to control/manage return network payload and improve network latency.

- GET /tickets?fields=**id**,**subject**,**customerName**
    - Return id, subject and customerName only rather than all the available fields for the resource.

Pagination Support

- By default, collection always support pagination;
- GET /tickets?**page**=3&**perPage**=25
- Getting the 3rd page with each page 25 records.

HTTP Response Code Designs

200 Series HTTP Status Codes

200 OK — Response was successful.

201 Created — The entity submitted in the request body was created (synchronously).

202 Accepted — The entity submitted in the request body will be created (asynchronously).

204 No Content — No need to update the view.

205 Reset Content — Client should reset the view.

206 Partial Content —Partial content returned (e.g. ranged or paginated content).

govtech stack

400 Series HTTP Status Codes

- **400 Bad Request** — The request was malformed.
- **401 Unauthorized** — The client is not authenticated with the server.
- **403 Forbidden** — The client is authenticated with the server, but not authorised to perform the requested operation on the requested resource.
- **405 Method Not Allowed** — The HTTP method used is not allowed on the requested URL.
- **409 Conflict** — There was a conflict when performing the operation, for example, the request attempted to update a resource that had already changed.

govtech stack

500 Series HTTP Status Codes

- **500 Internal Server Error** — An error on the server occurred and was not handled.
- **501 Not Implemented** —The HTTP method is not currently implemented for the requested resource.
- **503 Service Unavailable** — The server or one of it's dependencies (such as a database) is unable to respond due to overload, outages, etc.

govtech stack

THANK YOU