CODECOOL

REST API

REST API

CODECOOL

- Very common in the commercial world
- Meets following criteria:
  - **Uniform interface**
    - All API requests for the same resource should look the same
  - **Client-server decoupling**
    - client and server applications must be completely independent of each other
  - **Statelessness**
    - Server applications aren’t allowed to store any data related to a client request.
  - **Cacheability**
    - When possible, resources should be cacheable on the client or server side
  - **Layered system architecture**
    - REST APIs need to be designed so that neither the client nor the server can tell whether it communicates with the end application or an intermediary.

CODECOOL

Examples

- https://swapi.dev/
- https://petstore.swagger.io/
- https://any-api.com/
  - https://api.github.com/
- https://public-apis.io/
  - https://random.dog/woof.json?ref=apilist.fun
  - https://randomfox.ca/

JSON

CODECOOL

REST APIs should accept JSON for request payload and also send responses to JSON.

```
{
    hey: "guy",
    anumber: 243,
  - anobject: {
        whoa: "nuts",
      - anarray: [
            1,
            2,
            "thr<h1>ee"
        ],
        more: "stuff"
    },
    awesome: true,
    bogus: false,
    meaning: null,
    japanese: "明日がある。",
    link: http://jsonview.com,
    notLink: "http://jsonview.com is great"
}
```

```
{
  "id": 1,
  "name": "Foo",
  "price": 123,
  "tags": [
    "Bar",
    "Eek"
  ],
  "stock": {
    "warehouse": 300,
    "retail": 20
  }
}
```

URL - PATH

CODECOOL

Use nouns instead of verbs in endpoint paths

The action should be indicated by the HTTP request method that we're making. The most common methods include GET, POST, PUT, and DELETE.

GET /articles/ for getting news articles. Likewise, **POST /articles/** is for adding a new article , **PUT /articles/{id}** is for updating the article with the given id. **DELETE /articles/{id}** is for deleting an existing article with the given ID.

Use logical nesting on endpoints

CODECOOL

If we want an endpoint to get the comments for a news article, we should append the /comments path to the end of the /articles path.

In the code above, we can use the GET method on the path **/articles/{articleId}/comments**.

Filtering, sorting, and pagination

CODECOOL

/employees?lastName=Smith&age=30


```
[
   {
      "firstName": "John",
      "lastName": "Smith",
      "age": 30
   }
]
```


What are advantages of using REST API?

CODECOOL

- Frontend is totally separated from backend
- You can substitute frontends in the future, no coupling like in MVC
- Client doesn’t need to download any packages, just make sure API is up and running and it can be used
- Easy to expose API to many clients [Fronted/Mobile]

What are DISADVANTAGES of Rest API?

CODECOOL

- **Lack of state**
  - The client bears the task of maintaining the state, making the client application heavy and difficult to maintain.
- **Give me more endpoints!**
  - No matter what, there will always be a need for new endpoint, that fetches similar data, but other client needs extra this and that
  - You can either add multiple similar endpoints obscuring API or make one with overhead props that are not necessarily needed by all


CODECOOL