W03 – Best practices for REST apis

You might have noticed, I am a big fan of topics and for this lesson, I was looking into tips and tricks that can make our REST apis closer to the insustry standard.

1. According to this article: <https://www.freecodecamp.org/news/rest-api-best-practices-rest-endpoint-design-examples/>, when designing APIs we should prefer JSON formats over XML or HTM. It is said to be preferred in the industry, because of how easily it allows navigation, parsing and implementation. For that it is important to set the Content-type to JSON on the client side.
2. Endpoints should not be verbs. Do not use something like <http://localhost/getUsers> or <http://localhost/returnUsers>. Prefer using simply <http://localhost/users>. This is because the HTTP methods are already verbs : POST, GET, DELETE, PUT, COPY, etc. That is preferred in order to avoid confusion
3. Use plural nouns for Collections. Prefer using <http://localhost/users> and <http://localhost/users/id>, when /users is acollection of records. ([http://localhost/user](http://localhost/users), can be confusing if it references a collection of records.)
4. Use the status codes in error handling. Just like we`ve seen and done in the lessons, use those, in order to help users and other programmers troubleshoot eventual problems.

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente com confiança média

400 bad request,

401 unauthorized

403 forbidden

404 not found

500 Internal server error

502 Bad Gateway

503 Service Unavailable

1. Avoid adding too many levels to your endpoints. The industry standard is to have a maximum of 3 levels.
2. Provide accurate API documentation, just as we learned this past week, it is important to keep a good documentation, in order to help the user to receive the data they want. Some good suggestions of what it should have are:
   1. Lists of error messages and respective cause
   2. Lists of possible endpoints the user can access
   3. Lists of examples in several different languages

Use Swagger, that is very popular amongst programmers.

To finish, a heads up.

As we work with a small database (right now it probably will have less than a couple of records). It is ok to return ALL records from the database. But as systems grow larger, we don`t want to allow that (imagine returning ALL facebook users all at once!). That would crash their server, which would put down services and a lot of money would be lost in revenue. Mr Zuckerberg would not be happy. In order to avoid that, remember allowing your system to filter what contacts will be brought up (like MY contacts). Another very handy way of dealing with that is pagination. This allows your system to go over a big amount of records but by bit.

<https://www.freecodecamp.org/news/rest-api-best-practices-rest-endpoint-design-examples/>

<https://stackoverflow.blog/2020/03/02/best-practices-for-rest-api-design/>