Rest APIs:

**1) Termologias**

- **Resource** é um objeto, ou a representção de alguma coisa que tem associação com dados e nele podem ser setados metodos para opera-los.

Ex.: entidades e CRUD.

- **Collections** é uma gama de **Resource**.

- **URL** (Uniforme **Resource** Locator) é o caminho através qual cada **resource**  pode ser localizada e algumas ações podem ser performadas.

**2) API endpoint**

Por exemplo APIs para companhias que tem empregados,

/getAllEmployess, é uma API que responde com uma lista de empregados, algumas outras APIS relacionadas com a companhia pode parecer com o seguinte:

/addNewEmployee

/updateEmployee

/deleteEmployee

/deleteAllEmployees

/promoteAllEmployees

/promoteAllEmployess

Muitos desses diversos endpoints que possam vir a ter irão ter muitas ações redundantes, consequentemente com o aumento de APIs a manutenção seria bastante onerosa.

**O que está errado?**  A **URL** deve somente conter apenas resources (substantivos), não verbos ou ações.

**O que seria o correto? /companies**,é um bom exemplo. Mas como poderiamos dizer ao servidor como performar ações na **resource companies**, como adicionar, deletar ou editar?

E ai que os metodos HTTP (GET, POST, DELETE, PUT) ou também chamados de verbos, entram em ação.

A **resource** sempre estará no plural no endpoint da API, e se quisermos acessar uma instância da **resource** podemos passar o **id** na **URL**.

- metodo **GET** caminho **/companies** deverá trazer uma lista de todas as companies

- metodo **GET** caminho **/companies/34** deverá trazer o datalhe da company 34

- metodo **DELETE** caminho **/companies/34** deverá deletar a company 34

Em alguns casos podemos ter **resource** dentro de **resource**, e.x.: Empregados da Companhia, alguns exemplos de endpoints desta API seria:

- **GET /companies/3/employees** traria uma lista de empregados da company 3.

- **POST**  /**companies** crairia uma nova company e retornaria os detalhes da nova company criada.

**3) HTTP metodos (verbos)**

A HTTP definiu alguns metodos que indicam o tipo de ação à ser performada nas **resources**.

GET: lista, POST: cria, PUT: edita ou cria caso não tenha, DELETE: remove

**4) HTTP codigos de resposta de status**

Quando um cliente faz uma requisição ao servior atravéz da API, o cliente deseja saber qual ser a resposta do servidor, dentre elas se deu falha, se foi passada ou se a requisição está incorreta.

**2xx (Success catgory)**

**200 ok,** para GET, PUT ou POST

**201 created.** POST

**204 No Content,** foi processada com sucesso porém não retornou nenhum conteudo. DELETE

**3xx (Redirection Category)**

**304** indica que o cliente já possui a resposta solicitada em cache. Consequentemente não necessita da transferência do mesmo dado de novo.

**4xx (Cliente Error Category)**

inidica que o cliente lançou uma falha de requisição.

**400 Bad Request** o servidor não conseguiu entender o que o cliente requisitou.

**401 Unauthorized** indica que o cliente não esta autorizado à acessar a **resource**, e que deverá reenviar a requisição com as credenciais requiridas.

**403 Forbidden** indica que a requisição é valida e o cliente está autenticado, porém o cliente não está autorizado à acessar o diretório do servidor.

**404 Not Found** indica que a **resource**  solicitada não está disponível no momento.

**410 Gone** indica que a **resource** solicitda não está mais disponível que quer dizer que foi intencionalmente movida.

**5xx (Server Error Category)**

**500 Inetrnal Server Error** indica que a requisição é valida porém o servidoe está totalmente confuso e o servidor é questionado para atender alguma condição inesperada.

**503 Service Unavailable** indica que o servidor não istá disponível para receber o processo de requisições. Na maioria dos casos quando o servidor está em manutenção.

**5) Field name casing convention**

Seguir a mesma conveção de use case durante todo projeto (maísculo e minúsculo).

**6) Searching, sorting, filtering and pagination**

Essas ações são implementadas na query do database. Não existe novas ações na API para lidar com esses casos. Vocẽ precisa pendurar esses parâmetros via o metodo GET da API.

**Sorting** nesse caso o endpoint deverá aceitar multiplos parâmetros de ordem na query. E.x.: **GET /companies?sort=rank\_asc** ordenaria as companhias pelo seu rank in ordem ascendente.

**Filtering** para filtrar o dado setado você pode passar várias opções pelos parametros da query.

e.x.: **GET /companies?category=banking&location=india**

**Searching** quando por exmplo pesquiasr o nome de uma companhia em uma lista de com companhia o edpoint da API deverá ser.

e.x.: **GET /companies?search=Digital Mckinsey**

**Pagination** quando o volume de dados buscados é muito grande, dividimos o dados em amostras menores, isto aprimora a performace e a maneira de lhe dar com a resposta

e.x.: **GET /companies?page=23**

Caso a requisição GET na URI seja muito longa o servidor responde com 414 - URI too long HTTP status, nesses casos podemos passar os parametros requisitados no body via metodo POST.

**7) Versioning**

Para previnir a quebra de APIs quando elas são atualizadas precisamos quebrar os produtos existentes em versões como no exemplo:

**http://api.yourservice.com/v1/companies/34/employees**