Ficha-Resumo - Padrões de projeto utilizados para desenvolvimento de API REST com Node JS.

Hoje API's são a base para qualquer negócio informatizado, portanto são a base para qualquer empresa, por isso vamos abordar abaixo conceitos básicos sobre o padrão Rest, sobre o protocolo HTTP e uma série de boas práticas visando a construção de uma API Rest de alto nível.

Rest: Representational State Transfer (Transferencia de Estado Representacional, representa uma série de princípios definidos pela World Wide Web, visando a padronização de rotas, requisições e comunicações sem estado, o próprio protocolo HTTP, foi baseado nestas regras e se propõe a resolver todos estes problemas.

O protocolo HTTP é a base de tudo, as requisições chegam e saem através dele, os padrões são baseados em sua estrutura, e seus códigos de resposta, por esse motivo, todas as nossas API's devem respeitá-lo e aplicar suas melhores práticas. Existem 9 diferentes verbos do protocolo HTTP conforme demonstrado abaixo:

Verbos Protocolo HTTP	
Verbo	Objetivo
CONNECT	Utilizado para abrir uma comunicação bidirecional com determinado recurso
DELETE	Utilizado para deletar determinado domínio
GET	Utilizado para recuperar os dados de um determinado dominio
HEAD	Basicamente faz o mesmo que o verbo GET, porem na sua resposta só é retornado o cabeçalho da requisição
OPTIONS	Utilizado para recuperar as possiveis opções de requisições sobre um determinado domínio
PATCH / MERGE	Utilizado para atualizar parcialmente os dados de um determinado domínio
POST	Utilizado para criar dados de um determinado dominio ou realizar operações lócigas (Por exemplo um calculo)
PUT	Utilizado para atualizar dados de um determinado dominio
TRACE	Utilizado para enviar mensagens por todo o caminho entre a origem e o destino de uma requisição, provendo um grande mecanismo de debug

Padronização das rotas

O ideal é que seja definido um padrão único para ser aplicado em todas as Api's de uma empresa, abaixo um exemplo prático cobrindo os principais verbos do protocolo HTTP:

Roteamento		
Requisição	Path	Objetivo
GET	/api/products	Consultar todos produtos
GET	/api/products/4	Consultar produto específico
POST	/api/products	Criar um produto
PUT	/api/products/7	Atualizar um produto específico
DELETE	/api/products/2012	Deletar um produto específico

O correto é que as rotas sejam apenas substantivos que representem o domínio em si e utilizem os verbos do protocolo HTTP, conforme o exemplo abaixo:

Roteamento		
Requisição	Path	Objetivo
GET	/api/products/	Consultar todos produtos
GET	/api/people/	Consultar todas as pessoas
GET	/api/campaigns/	Consultar todas as campanhas

Padronização dos códigos de resposta

O protocolo HTTP é consistido por uma infinidade de códigos, implementá-los à risca se torna algo extremamente impossível, é um árduo trabalho tanto para quem constrói as Api's quanto para quem as consome.

Para resolver este problema, é recomendado eleger um grupo de códigos HTTP para serem utilizados como base no desenvolvimento das API's, seguindo esta linha, veja as recomendações abaixo:

HTTP 200 - OK

Geralmente esse é um código de retorno é utilizado como coringa, onde independente do tipo de requisição, se o processamento é realizado com sucesso, esse é o código de retorno default.

HTTP 201- Created

Geralmente utilizado em requisições POST, onde algum dado foi criado em algum repositório.

HTTP 202 - Accepted

Geralmente utilizado em requisições POST, indica que algum dado foi recebido e seu processamento será realizado de forma assíncrona, por exemplo a API posta o dado em uma fila MQ e um worker tratará o dado em um segundo momento.

HTTP 204 - No Content

Geralmente utilizado em requisições PUT ou DELETE, indica que o processamento foi finalizado e a API não tem nenhum dado para devolver para seu cliente.

HTTP 400 - Bad Request

Geralmente utilizado em todos os tipos de requisições, ocorre quando o cliente preencheu algum dado incorreto na requisição.

HTTP 401 - Unauthorized

Geralmente é utilizado para qualquer tipo de requisição, ocorre sempre que um usuário não esta autenticado para uso da API.

HTTP 404 - Not Found

Esse código de resposta pode ser utilizado quando a aplicação cliente procura por uma rota/recurso e ela não existe ou em uma requisição GET ou PUT, quando a aplicação client tenta consultar ou alterar um domínio que não existe.

HTTP 408 - Timeout

Geralmente é utilizado para qualquer tipo de requisição, ocorre sempre que uma API gera timeout em uma das suas operações.

HTTP 409 - Conflict

Geralmente é utilizado em requisições do tipo POST, ocorre quando o cliente tenta criar um dado que já existe.

HTTP 500 - Internal Server Error

Geralmente é utilizado para qualquer tipo de requisição, ocorre para qualquer tipo de erro de processamento da API.

HTTP 502 - Bad Gateway

Geralmente é utilizado para qualquer tipo de requisição, ocorrerá sempre que uma dependência externa a API, apresentar algum tipo de comportamento inesperado.

Versionamento

Existem diferentes modelos de versionamento de API, não existe nenhuma regra definida no protocolo HTTP, também não existe nenhum tipo de padrão que seja unanimidade pela comunidade, seguindo esta linha, a sugestão é que sua utilização esteja bem explicita para os clientes, com a versão no final do Path, por exemplo:

Roteamento		
Requisição	Path	Objetivo
GET	/api/products/v1	Consultar todos produtos
GET	/api/products/v2	Consultar todos produtos
GET	/api/products/4/v1	Consultar produto específico
GET	/api/products/4/v2	Consultar produto específico
POST	/api/products/v1	Criar um produto
POST	/api/products/v2	Criar um produto

Uma prática importante é garantir que o versionamento da API não fique diretamente em seu código fonte, deixe o código fonte limpo, essa responsabilidade deve estar com um Gateway de API.

Autenticação

Garantir a segurança dos dados também faz parte do padrão de desenvolvimento de uma API, todas devem ter no mínimo uma autenticação do tipo basic, onde geralmente é criado um token em base64 a partir de um usuário e senha.

Stateless

Garantir que a API não guarde nenhum tipo de estado referente a comunicação, toda requisição deve ser inteligente o suficiente para se resolver, sem armazenar quaisquer dados de seção.

Negociação de conteúdo

Apesar do ideal ser trafegar apenas JSON, existem APIs que trafeguam outros tipos de dados, por exemplo XML, com isso deve ser usado as chaves Accept e Content-Type do protocolo HTTP para detalhar todas as possibilidades

Paginação

Em requisições GET que retornem listas, de ser devolvido somente o necessário, o recomendado é paginar a requisição, desta forma todos ganham, pois será trafegado somente o necessário, melhorando a experiencia do usuário final.

Ordenação

Otimizar as requisições facilita ainda mais a vida dos clientes, permitindo que a API ordene sua resposta de tal forma o cliente pode não precisar realizar nenhum tipo de tratamento no dado.

Respostas Parciais

Otimizar as requisições GET, permitindo que os clientes recuperarem somente os campos que realmente necessitam, desta forma de ponta a ponta serão trafegado apenas os dados necessários.

Hateoas

Representa o termo Hypermedia as the engine of application State (Hipermídia como o mecanismo do estado do aplicativo), é um padrão que sempre inclui links uteis junto as respostas de uma requisição.

Open Documentation

Além de seguir as melhores práticas, é recomendado a utilização de frameworks como o Swagger, que é um framework open source que apoia o desenvolvedor no desenho, documentação e especificação de suas Api's.