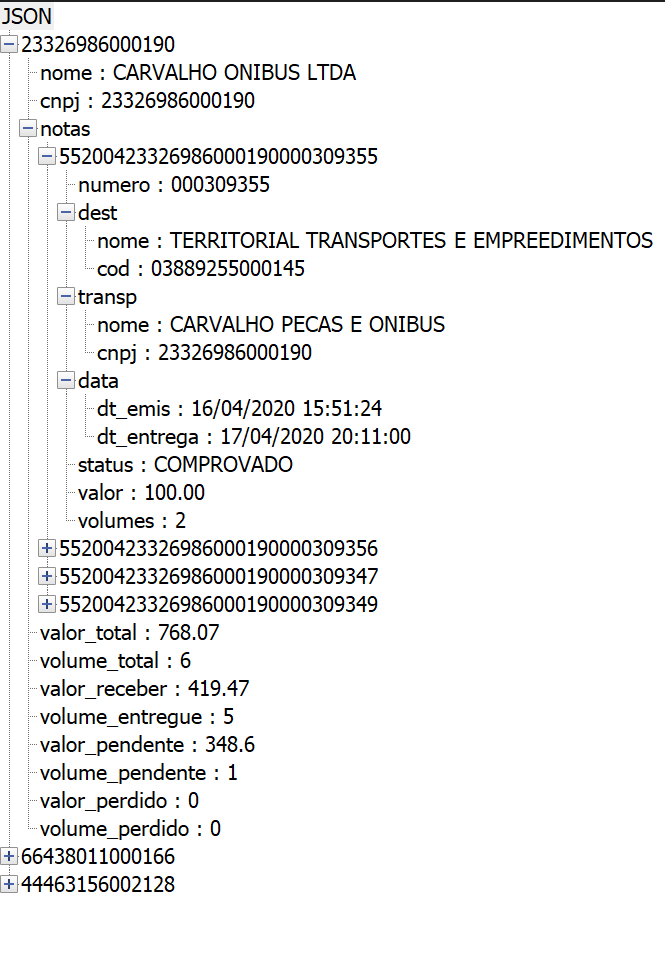
Nome: Matheus Thiago de Souza Ferreira

Link do Repositório:

**Documentação:**

Estrutura da API:

Para atender ao que foi proposto a estrutura da API foi construída da seguinte forma:



A estrutura foi construída de forma a ter como **Chave** o CNPJ do Remetente. Dentro da sua estrutura temos as informações referentes a esse remetente, sendo: nome, cnpj, notas, valor total das notas, volume total de mercadorias, valor a receber pelas entregas, volumes entregues, valor pendente (aguardando a entrega), volumes pendentes, valor perdido (quanto deixou de receber por atraso na entrega) e volume perdido.

Dentro da estrutura das notas temos as chaves dos serviços e dentro delas temos todos os dados referentes ao serviço. Sendo: numero do pedido, informações do destinatário (nome e código), informa da transportadora (nome e cnpj), data (data de emissão e entrega), status (comprovado ou em aberto), valor do serviço e quantidade de itens transportados.

Funções PHP:

Ao todo o nosso PHP é composto por 9 funções, sendo elas:

* function printJson($url) – Recebe o endereço da API base e exibe a nova API.

Exemplo:

Recebe:

url: “<http://homologacao3.azapfy.com.br/api/ps/notas>”

Executa:

Chama a função returnJson.

Retorna:

Exibe a nova estrutura da API, conforme mostrado na imagem anterior.

* function saveJsonFile($filename, $url) – Recebe o nome do arquivo e o endereço da API base, e salva a nova API em um .json com o nome informado.

Exemplo:

Recebe:

nome do arquivo: “notas”,

url: “<http://homologacao3.azapfy.com.br/api/ps/notas>”

Executa:

Chama a função returnJson.

Retorna:

Salva a nova estrutura da API em um arquivo de nome “notas.json”.

* function returnJson($url) – Recebe o endereço da API base e retorna a nova API.

Exemplo:

Recebe:

url: “<http://homologacao3.azapfy.com.br/api/ps/notas>”

Executa:

Decodifica a API base presente na URL e passa para a função createArray

Retorna:

Retorna a estrutura da nova API

* function createArray($data) – Recebe a API base decodificada e passa seus objetos para createNota e updateNota para gerar a nova estrutura dentro de um array.

Exemplo:

Recebe:

Data: API Decodificada

Executa:

Seleciona cada objeto dentro de $data e indica qual função irá trabalhar com ele (createNota ou updateNota).

Retorna:

Retorna um array com a estrutura da nova API.

* function createNota($info) – Recebe um objeto dado por createArray e gera toda a estrutura do remetente.

Exemplo:

Recebe:

Info: Objeto vindo de $data

Executa:

Cria a estrutura do remetente e chama as funções auxiliares.

Retorna:

Retorna a estrutura do remetente já finalizada.

* function updateNota($info , $nota) – Recebe um objeto dado por createArray e adiciona uma nova nota que pertence a um remetente que já existe.

Exemplo:

Recebe:

Info: Objeto vindo de $data.

Executa:

Adiciona uma nova nota fiscal dentro da estrutura do remetente e chama as funções auxiliares.

Retorna:

Retorna a estrutura do remetente atualizada.

function defineData($info) – Recebe um objeto vindo do defineNotaFiscal e verifica se deve retornar só a data de emissão ou a data de emissão e a de entrega.

Exemplo:

Recebe:

Info: Objeto vindo de $data.

Executa:

Verifica se a nota fiscal está em “Aberto” ou “Comprovado.

Retorna:

Retorna a estrutura das datas que se encaixam na nota fiscal.

* function defineNotaFiscal($info) – Recebe um objeto vido de createNota e updateNota e gera a estrutura da nota fiscal.

Exemplo:

Recebe:

Info: Objeto vindo de $data.

Executa:

Gera a estrutura da nota fiscal.

Retorna:

Retorna a estrutura da nota fiscal.

* function calculeValores($info, $nota) – Recebe o objeto e o remetente e calcula os valores (total , entregue, pendente e perdido) e os volumes.

Exemplo:

Recebe:

Info: Objeto vindo de $data e nota: estrutura do remetente ao qual serão realizados os cálculos.

Executa:

Verifica em qual atributo os valores se encaixam e realiza a soma do seu valor com o do objeto.

Retorna:

Retorna a estrutura do remetente.

Ao final obtemos a seguinte estrutura:

