TECNOLOGIAS

As tecnologias aqui escolhidas foram as seguintes: Node.js que é uma ferramenta que resolve a necessidade de subir um servidor web com um banco de dados, ao mesmo que é altamente escalável, possui uma comunidade muito grande e ativa, o que torna a mais fácil a procura por módulos e frameworks. Em ralação ao BD estou utilizando o MongoDB que é um banco de dados NOSQL e open source que no qual tem sua estrutura baseada em documentos JSON. Junto ao MongoDB utilizo uma biblioteca do Noje.js (Mongoose) que oferece uma solução em esquemas mapeando as collections do MongoDB. E por fim o Express.js que facilita muito a construção de uma API usando o Node, principalmente nas questões de rotas.

REQUISITOS

R1 Permanência de dados.

R2 Atender requisições.

R1 - Permanência de dados				Oculto [X]
Descrição: Ser capaz de realizar a permanência dos dados das vendas: merchantCnpj ,checkoutCode, cipheredCardNumber, amountInCents, installments, acquirerName, paymentMethod, cardBrandName,status, statusInfo, CreatedAt,AcquirerAuthorizationDateTime				
Requisitos Não-Funcionais:				
Nome	Restrição	Categoria	Desejável	Permanente
NF1.1 - Restrições no armazenamento do nº de cartão	Deve se criar uma cifra ou usar algum modo de criptografia para não se armazenar o número em si, apenas os últimos 4 dígitos.	Segurança	[ ]	[X]
NF1.2 – Restrições no armazenamento do CNPJ	O CNPJ da entrada deve ser um CNPJ válido	Validação	[ ]	[X]
NF1.3 – Restrições no armazenamento	O campo nunca deve possuir um valor	Validação	[ ]	[X]

da quantia	negativo.			
NF1.4 – Restrições no armazenamento do número de parcelas	O campo nunca deve possuir um valor negativo e deve estar condizente com o método de pagamento	Validação	[ ]	[X]
NF1.5 – Restrições no armazenamento dos nomes das Adquirentes	Apenas as seguintes são aceitas: Stone, Cielo, Rede, GetNet, Redecard, Bin e Elavon.	Validação e Modelo de negócios.	[ ]	[ ]
NF1.6 – Restrições no armazenamento dos métodos de pagamento	Apenas são aceitos os seguintes métodos de pagamento: Voucher, Débito à vista, Crédito à vista, Crédito Parcelado e Crédito Parcelado loja.	Validação e Modelo de negócios.	[ ]	[ ]
NF1.7 – Restrições no uso das Bandeiras	Apenas as seguintes bandeiras serão aceitas: Elo Débito, Sodexo Refeição, Sodexo, Elo, Visa, Maestro, Mastercard, Alelo Refeição, Electron, Alelo e Credz	Validação e Modelo de negócios.	[ ]	[ ]
NF1.8 – Status	Só existem dois status: Aprovado ou Reprovado	Validação e Modelo de negócios.	[ ]	[ ]
NF1.9 – checkoutCode	Só são aceitos valores dentro do intervalo 0-99999.	Validação e Modelo de negócios.	[ ]	[ ]

R2 - Atender requisições	Oculto [X]
Descrição: Capaz de retornar os dados em json, filtrados por qualquer um dos campos, para que possa ser consumida pelas automações comerciais.	

TESTES

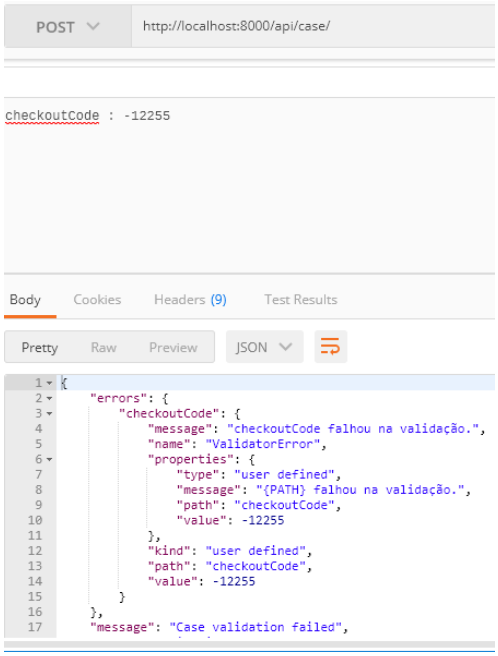
Requisito Verificado:	R1(NF1.1; NF1.2; NF1.3; NF1.4; NF1.5; NF1.6; NF1.7; NF1.8; NF1.9)
Identificação:	T1
PROCEDIMENTO	***VERIFICAÇÃO***
1. O cliente utiliza do método POST enviando os números do cartão.	A API guarda apenas os últimos quatro dígitos do número do cartão.
2. O cliente utiliza do método POST enviando os números do CNPJ.	A API deve guardar os números do CNPJ apenas se forem um número válido. Caso contrário retorna apenas um erro de validação.
3. O cliente utiliza do método POST enviando o valor da quantia em centavos.	A API verifica se a quantia não possui um valor negativo. Uma vez validado o valor é armazenado.
4. O cliente utiliza do método POST enviando o número de parcelas.	A API verifica se não possui um número de parcelas negativo ou nulo, posteriormente verifica-se se o número de parcelas é condizente com o método de pagamento(Ex: pagamento à vista implica em uma única parcela). Caso não condize é retornado um erro de validação, do contrario é armazenado o dado.
5. O cliente utiliza do método POST enviando o nome da adquirente.	A API verifica o adquirente , armazenando os dados caso seja uma entrada válida.
6. O cliente utiliza do método POST enviando o método de pagamento.	A API verifica se o método é um dos métodos válidos, armazenando os dados caso seja uma entrada válida.
7. O cliente utiliza do método POST enviando a bandeira utilizada.	A API verifica se a bandeira é uma das válidas, armazenando os dados caso seja válida.

8. O cliente utiliza do método POST enviando o status.	A API verifica se é um estado válido para o status(Aprovado ou reprovado). Caso válido os dados são armazenados.
9. O cliente utiliza do método POST enviando o checkoutCode.	A API verifica se o código se encontra dentro do intervalo de 0-99999. Caso não esteja nesse intervalo é retornado um erro de validação.

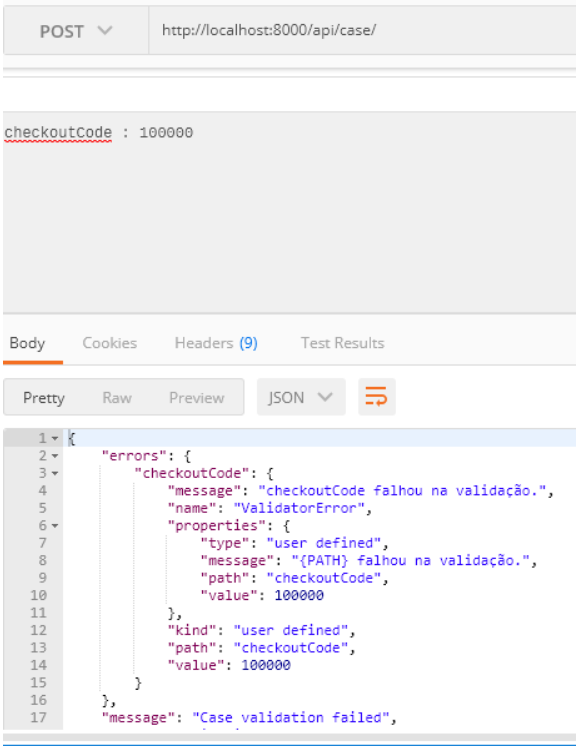
Requisito Verificado:	R2
Identificação:	T2
PROCEDIMENTO	***VERIFICAÇÃO***
1. O cliente utiliza do método GET utilizando qualquer filtro.	A API retorna os dados requisitados.

Para que tais testes fossem realizados foi utilizado uma ferramenta chamada postman que é uma extensão do Chrome que é muito útil para testar APIs. Abaixo segue os prints dos testes referentes aos requisitos NF1.9 e NF1.5, porém todos os testes foram realizados com a mesma ferramenta seguindo o mesmo método .

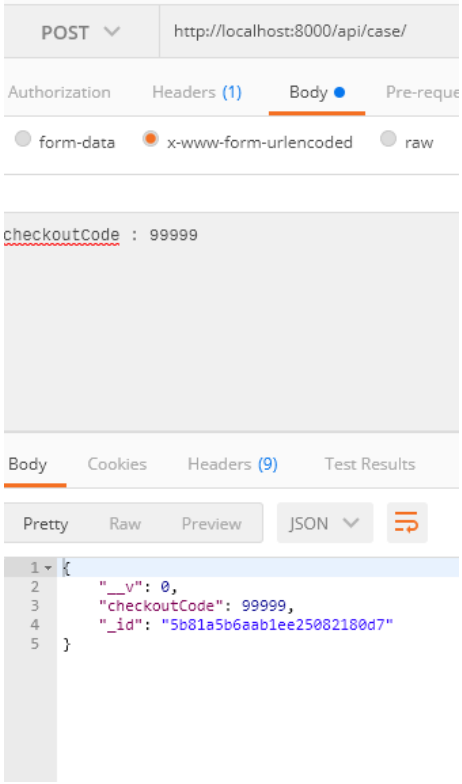
Teste do checkoutCode passando o valor ‘0’ e retornando que foi inserido com sucesso.



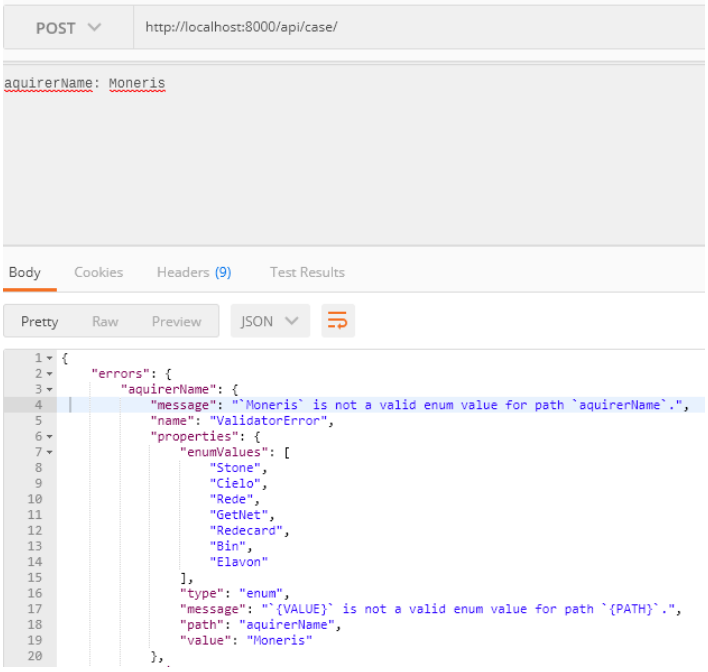
Teste do checkoutCode passando o valor ‘-12255’ e retornando que houve falha na validação.



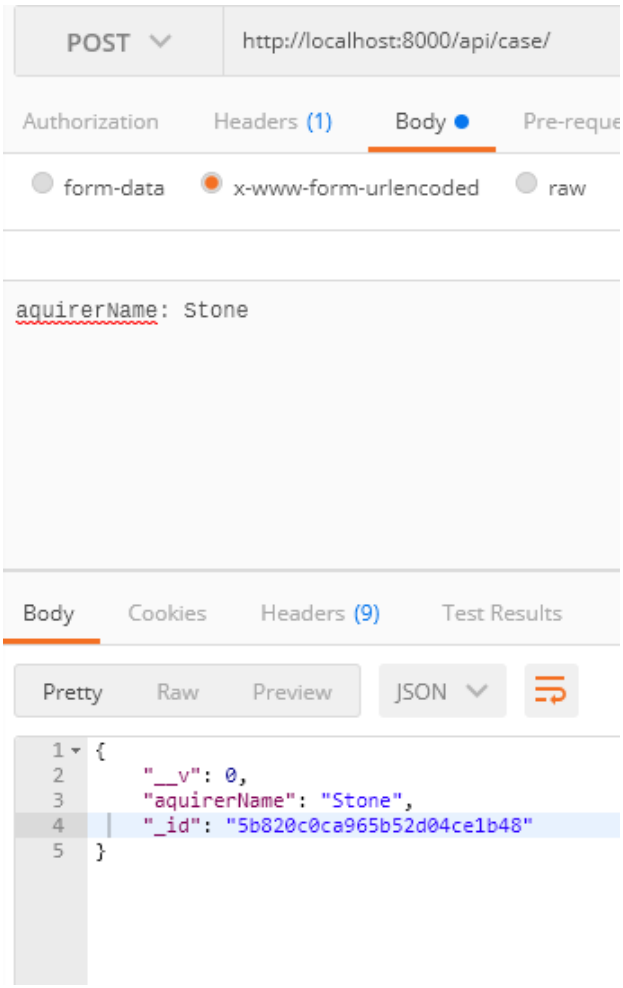
Teste do checkoutCode passando o valor ‘100000’ e retornando que houve falha na validação.



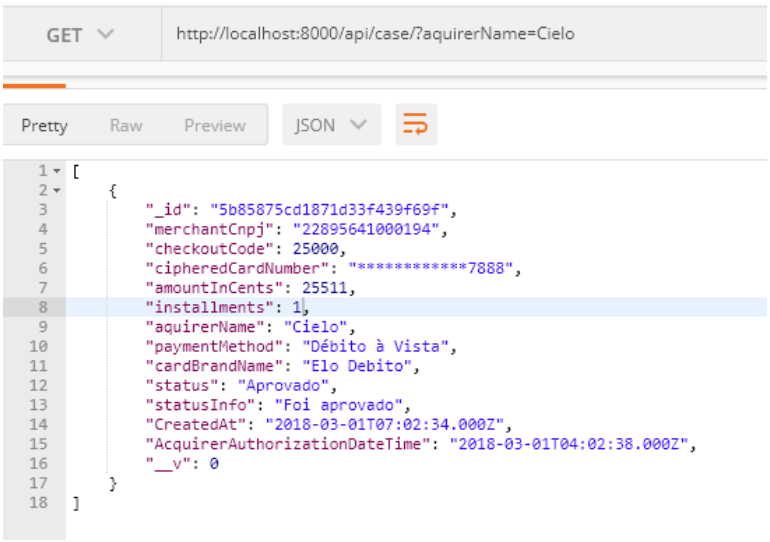
Teste do checkoutCode passando o valor ‘99999’ e retornando sucesso.



Teste do aquirerName passando o valor ‘Moneris’ (Que não está como uma adquirente válida) e retornando falha.



Teste do aquirerName passando o valor ‘Stone’ (Que está como uma adquirente válida) e retornando sucesso.



Teste realizando um requisição filtrando pelo “aquirerName” passando como parâmetro “Cielo”.