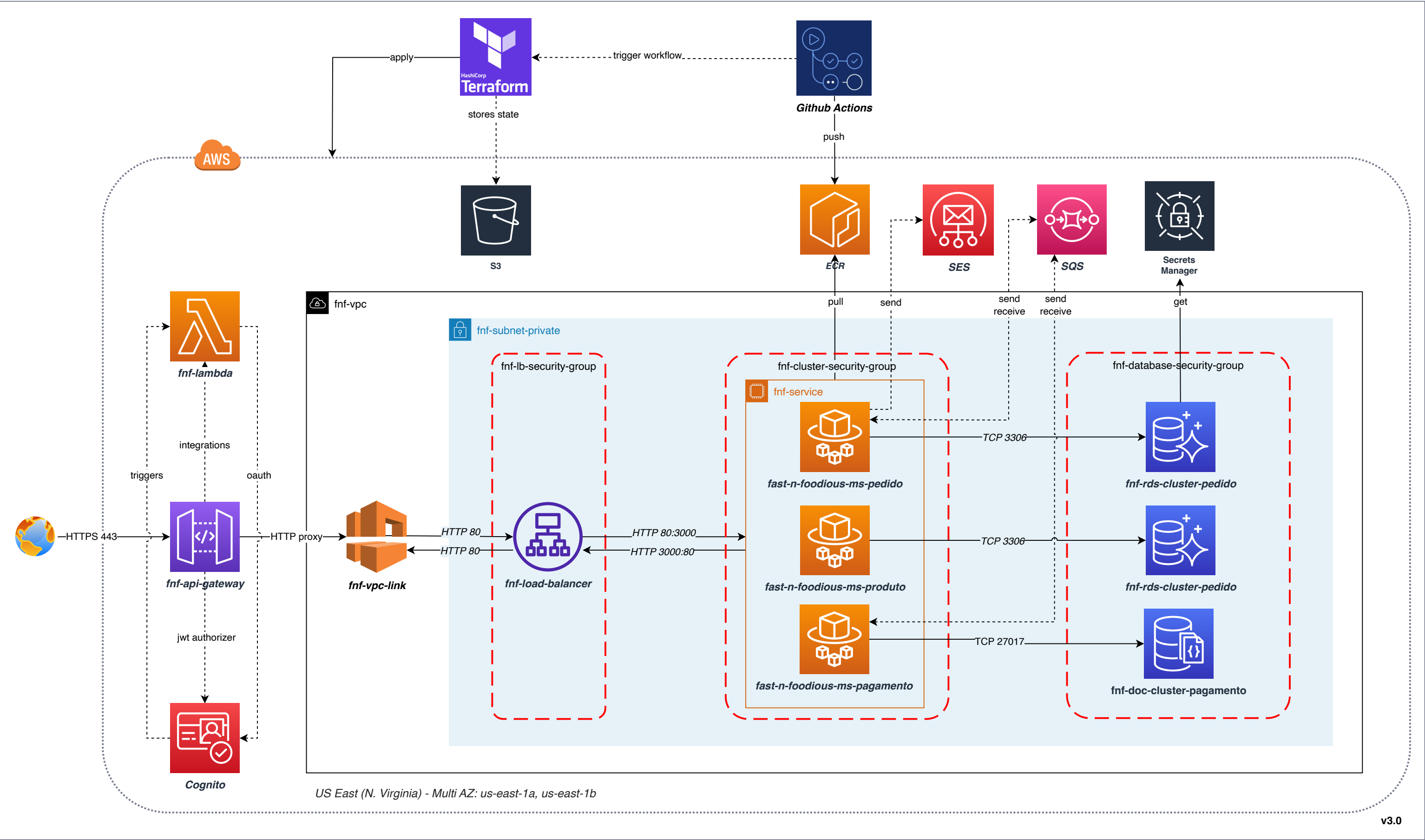
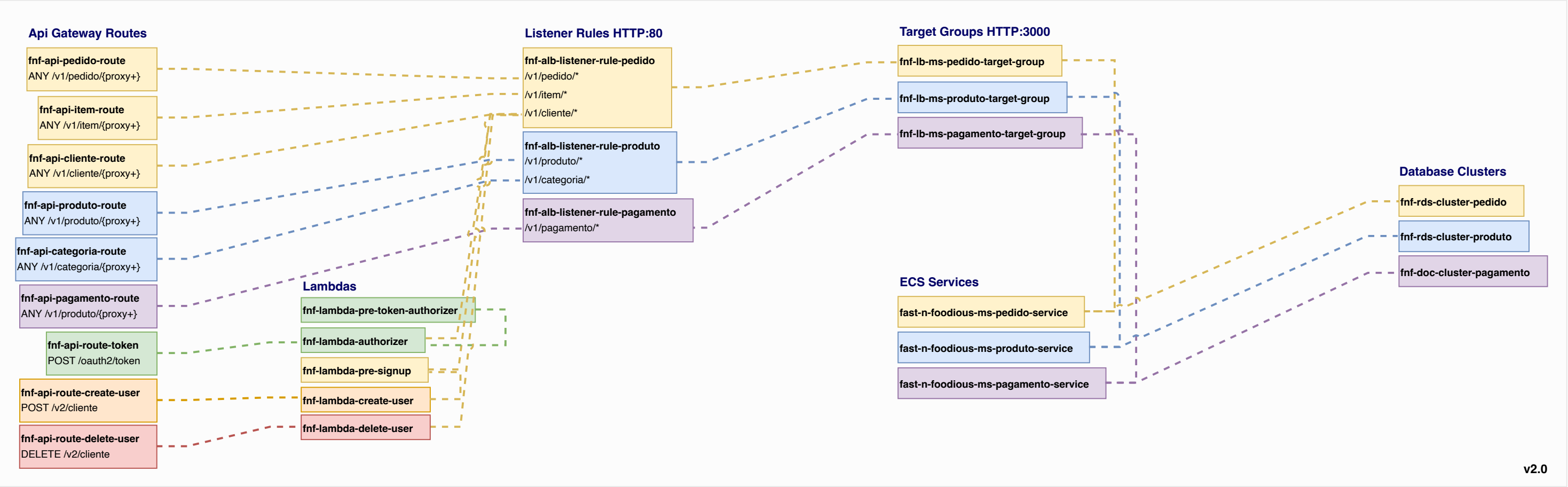


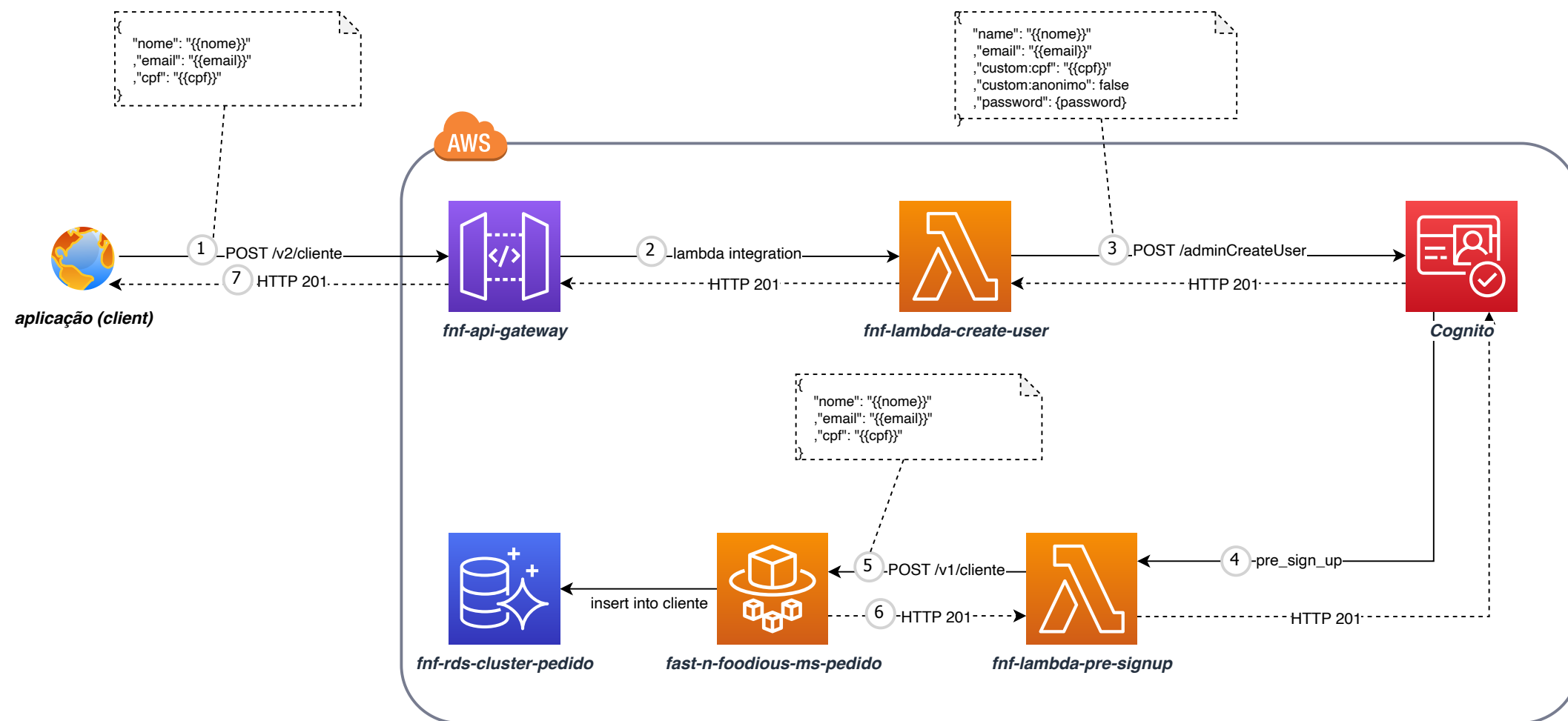
Fast n' Foodious: Arquitetura Cloud AWS



Fast n' Foodious: Resource Mapping

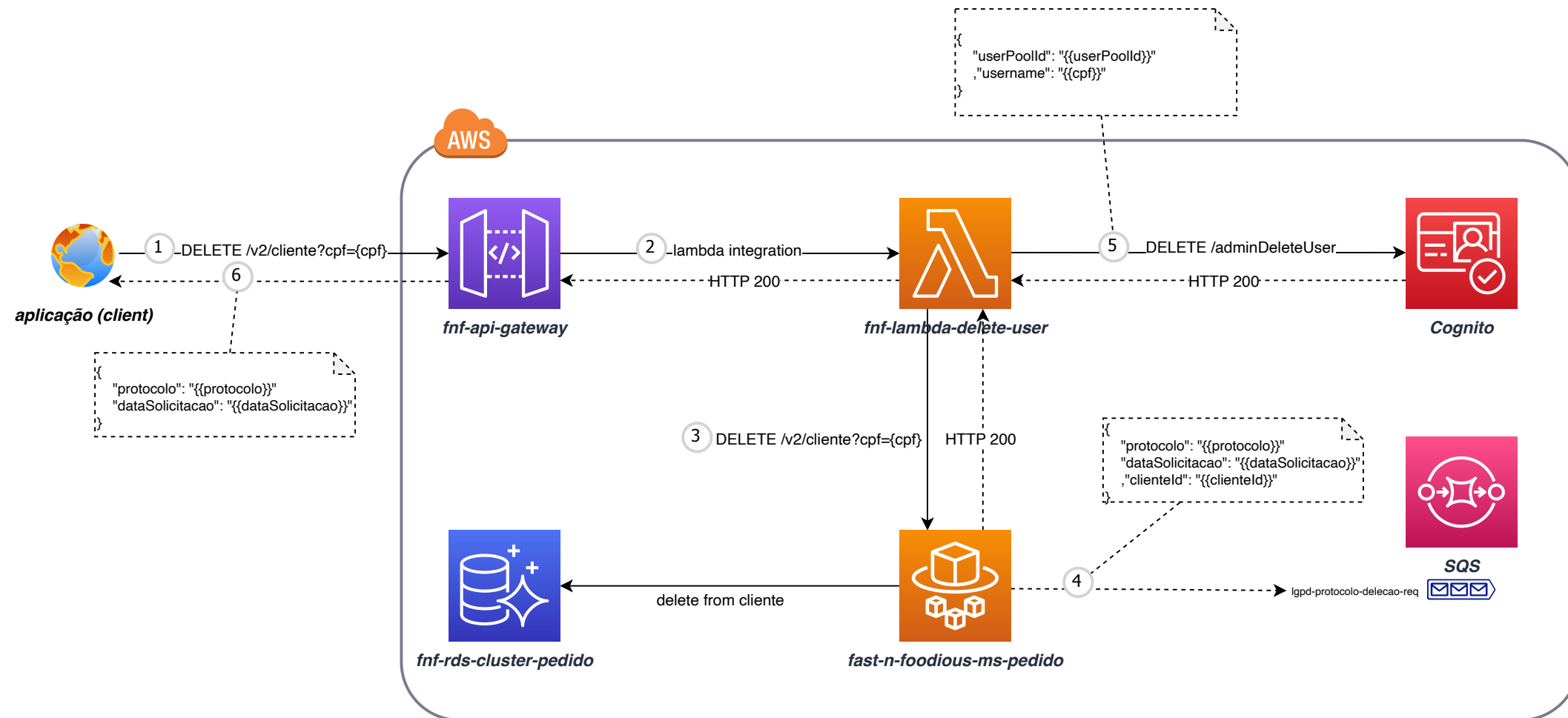


Fast n' Foodious: Cadastro de Clientes



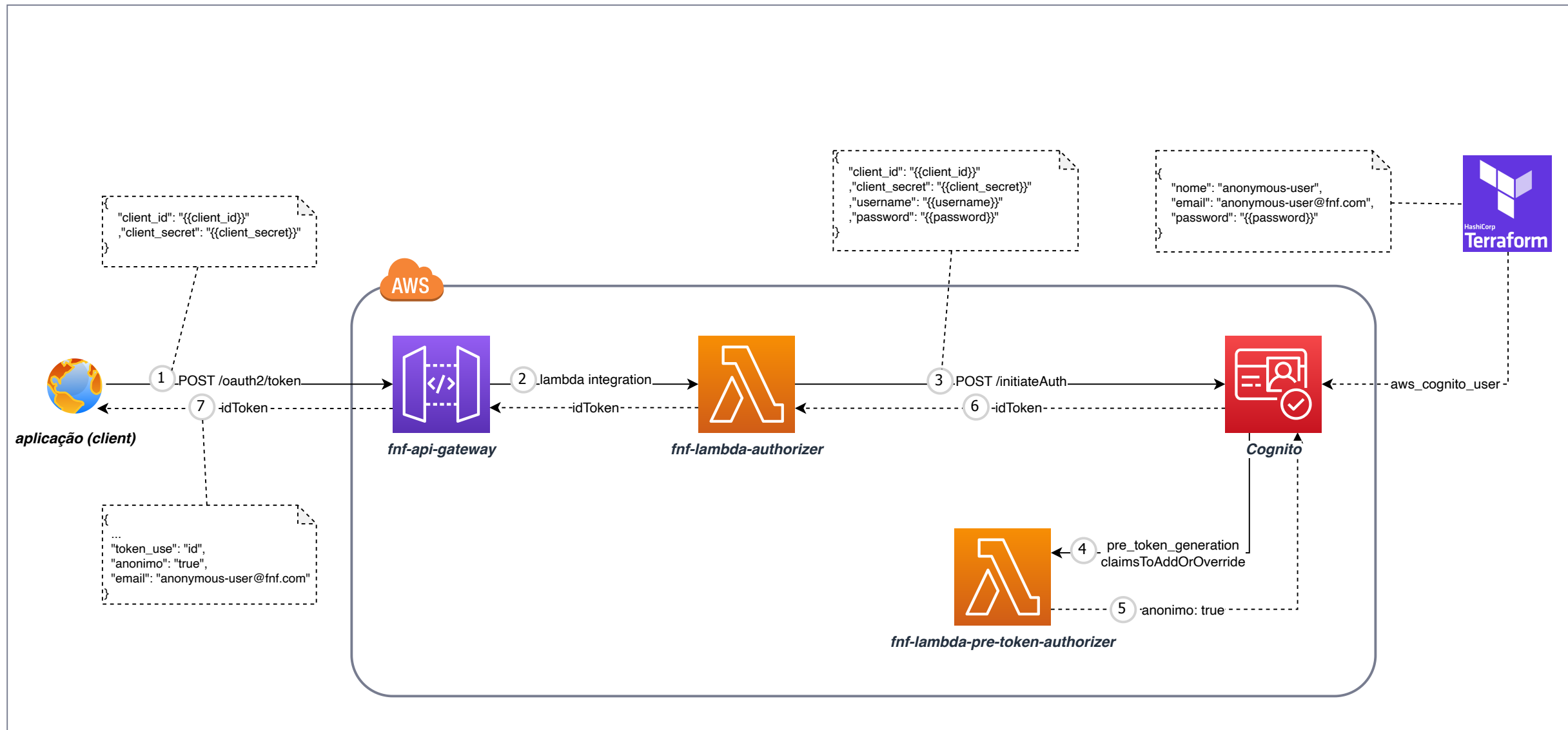
1. A aplicação realiza o cadastro de cliente, passando os dados de nome, email, cpf
2. O gateway realiza a integração com o fnf-lambda-create-user
3. O lambda inicia o cadastro de novo usuário no Cognito
4. O Cognito invoca o lambda fnf-lambda-pre-singup, para inclusão do cliente na aplicação (cliente fast-n-foodious)
5. O lambda fnf-lambda-pre-singup realiza o cadastro do cliente na aplicação (cliente fast-n-foodious)
6. O sistema fast-n-foodious retorna HTTP 201, confirmando a operação de cadastro
7. A aplicação recebe HTTP 201 como confirmação de cliente cadastrado

Fast n' Foodious: Deleção de Clientes (LGPD)



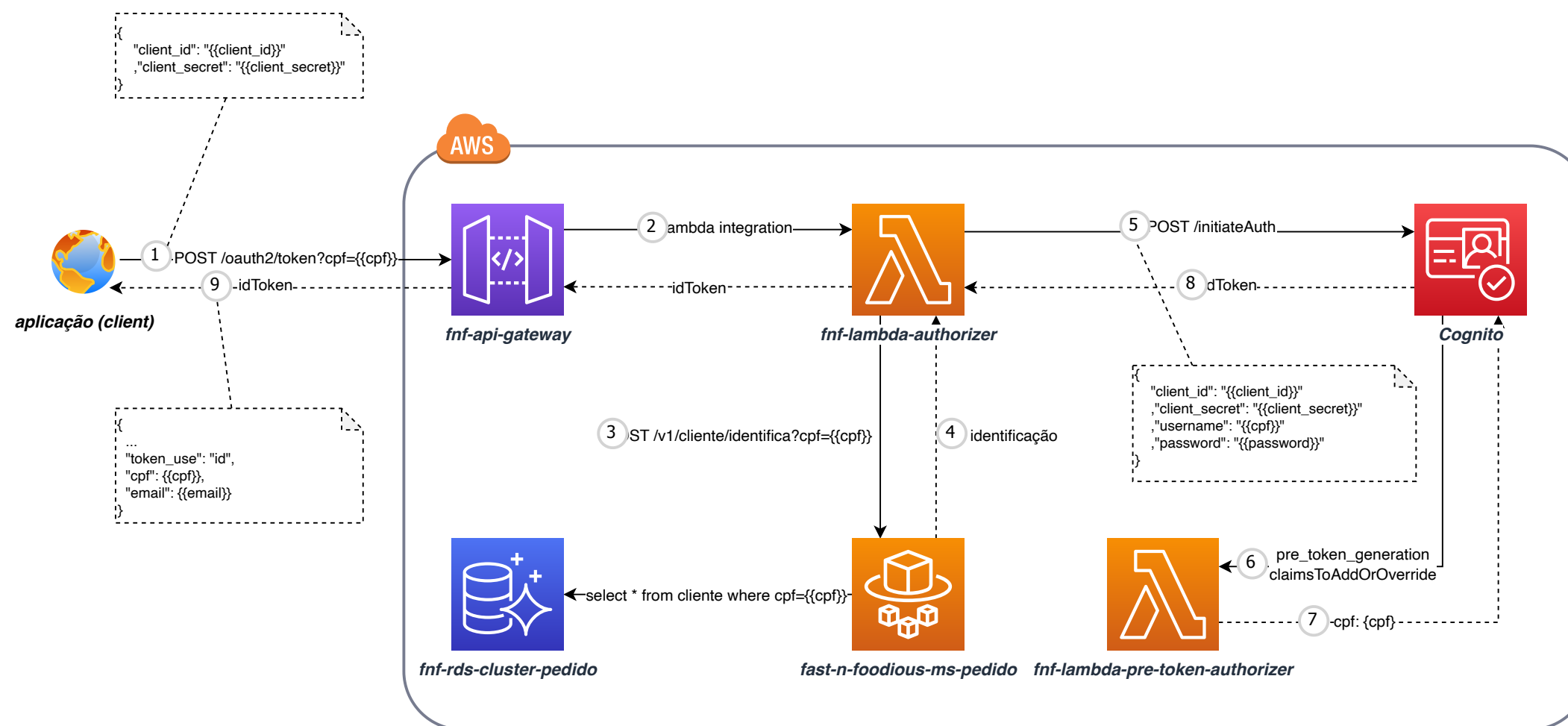
1. A aplicação realiza a deleção de cliente, passando os dados do cpf
2. O gateway realiza a integração com o fnf-lambda-delete-user
3. O lambda inicia a deleção do cliente na aplicação (cliente fast-n-foodious), recebendo HTTP 200
4. O sistema fast-n-foodious executa a deleção e publica o protocolo da fila para armazenamento posterior
5. O lambda inicia a deleção do usuário no Cognito, recebendo HTTP 200
6. A aplicação recebe a confirmação de cliente deletado com o protocolo da operação e a data de solicitação

Fast n' Foodious: Autenticação de Cliente Anônimo



1. A aplicação se autentica informando suas credenciais (client_id e client_secret)
2. O gateway realiza a integração com o fnf-lambda-authorizer
3. O lambda authorizer realiza a autenticação do cliente com o usuário anonymous-user no Cognito
4. O Cognito invoca o lambda fnf-lambda-pre-token-authorizer no evento pre_token_generation, para sobreposição do userAttributes
5. O Lambda fnf-lambda-pre-token-authorizer cria a claim anonimo e adiciona ao token
6. O Cognito gera um token de autorização assinado, contendo a claim anonimo
7. A aplicação recebe o idToken do cliente anônimo

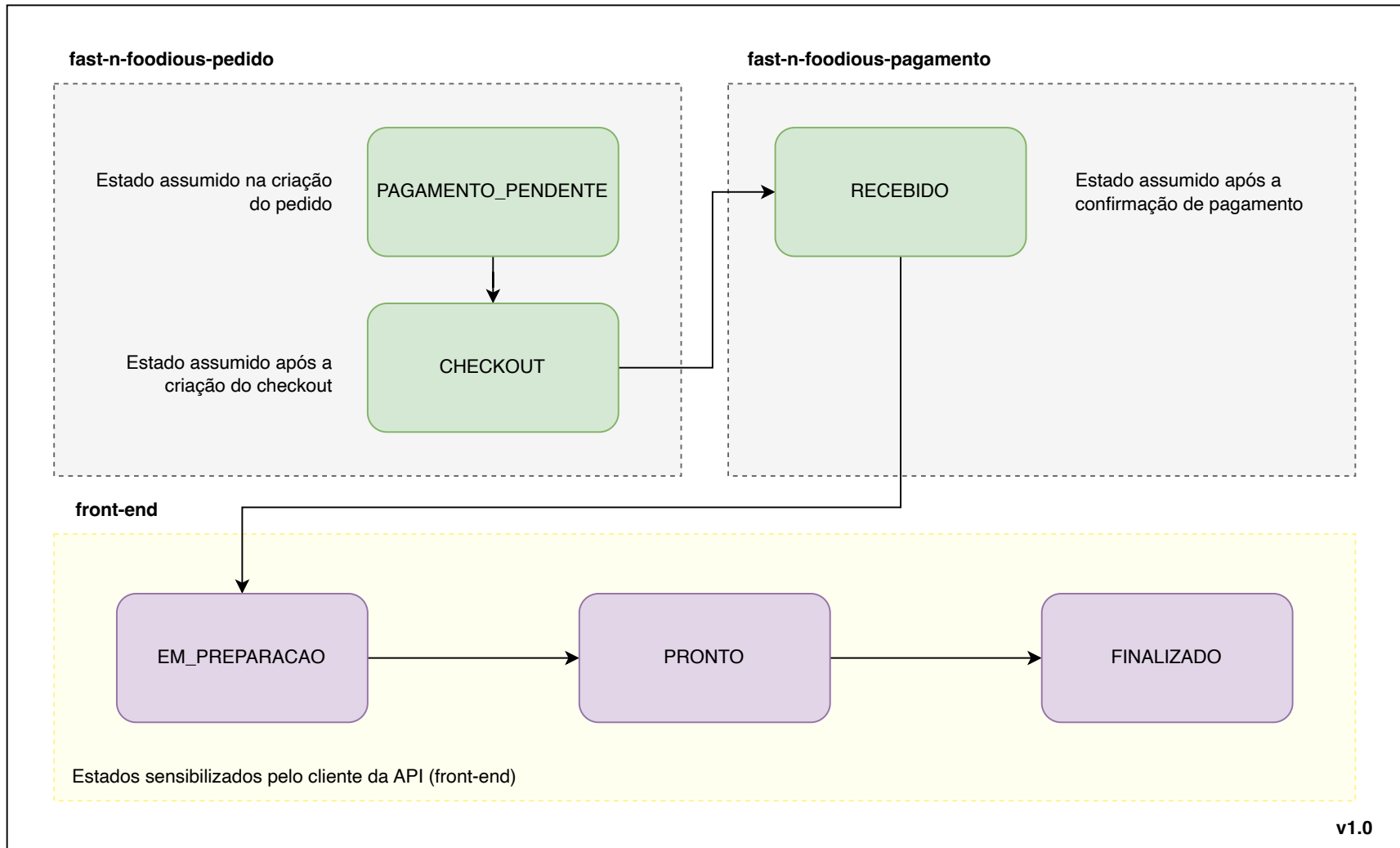
Fast n' Foodious: Autenticação de Cliente Identificado



1. A aplicação se autentica informando suas credenciais (`cliente_id`, `client_secret`)
2. O gateway realiza a integração com o `fnf-lambda-authorizer`
3. O lambda authorizer realiza a identificação do cliente (cliente `fast-n-foodious`)
4. O sistema `fast-n-foodious` retorna os dados do cliente identificado (nome, email e `cpf`)
5. O lambda authorizer realiza a autenticação do cliente identificado no Cognito
6. O Cognito invoca o lambda `fnf-lambda-pre-token-authorizer`, no evento `pre_token_generation`, para sobreposição do `userAttributes`
7. O lambda `fnf-lambda-pre-token-authorizer` cria a claim `cpf` e adiciona ao token
8. O Cognito gera um token de autorização assinado, contendo a claim `cpf`
9. O A aplicação recebe o `idToken` do cliente identificado

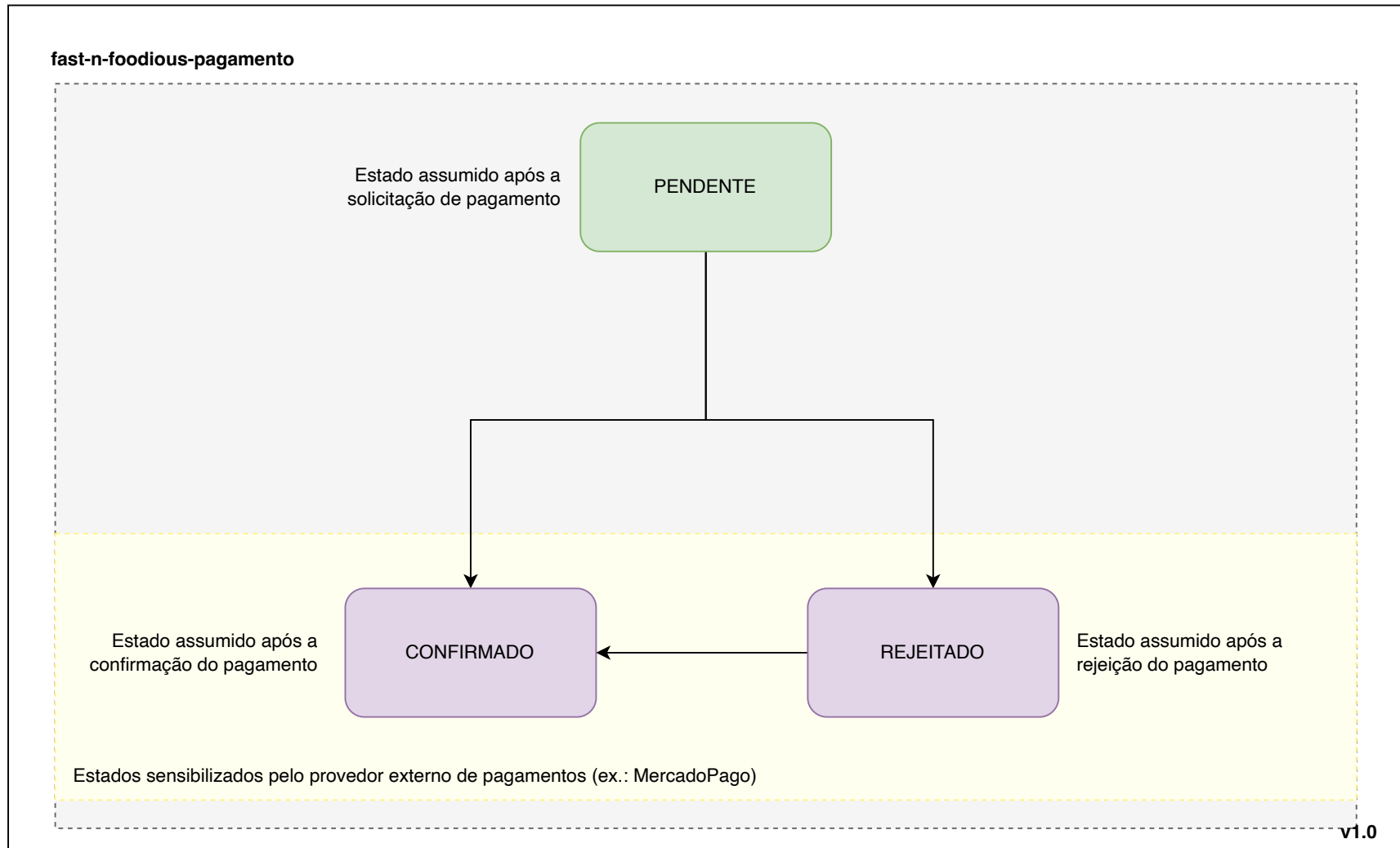
Transição de Estados do Pedido

Componentes que sensibilizam as mudanças de estados

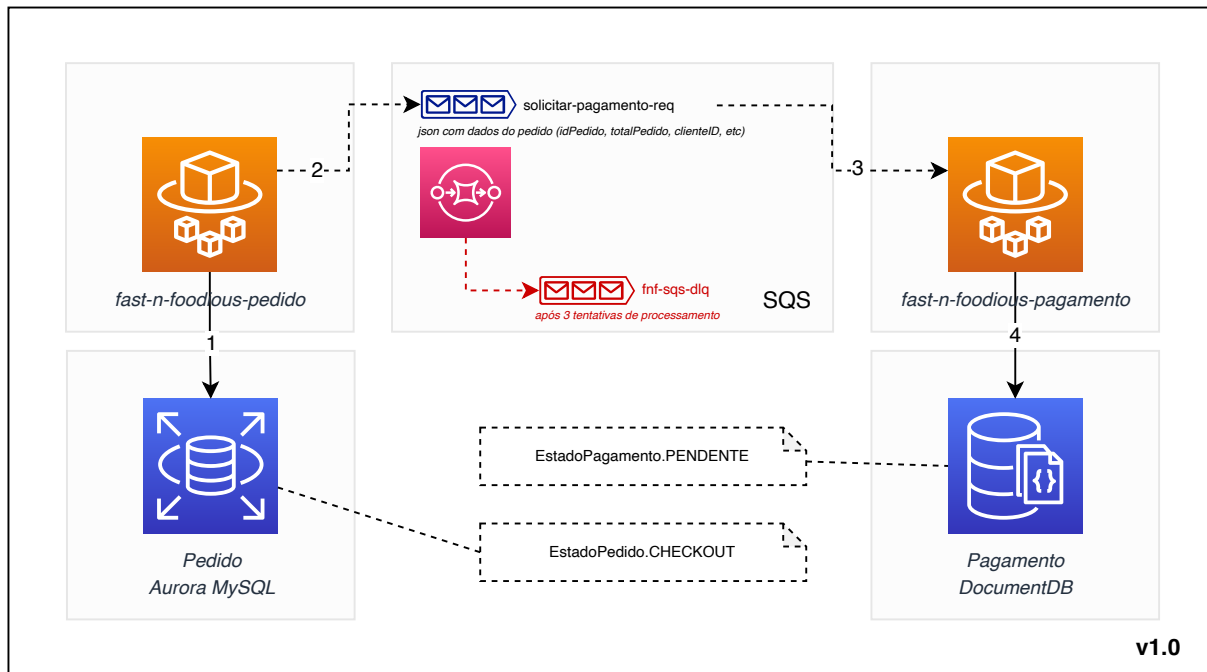


Transição de Estados do Pagamento

Componentes que sensibilizam as mudanças de estados

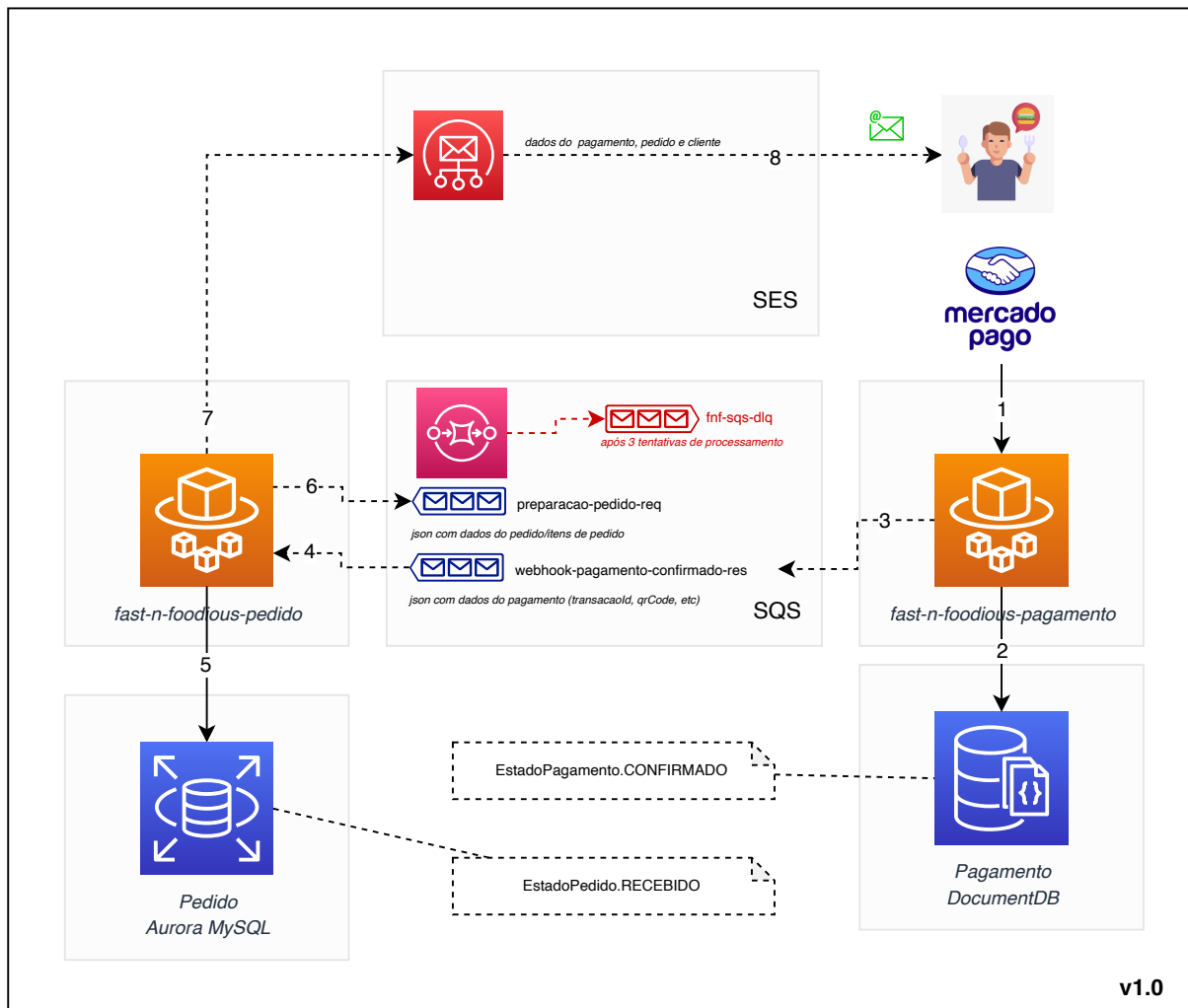


Checkout: Solicitação de Pagamento



Webhook: Atualização estado de Pagamento

Pagamento confirmado



Webhook: Atualização estado de Pagamento

Pagamento rejeitado

