Stone

Desafio para Backend Developer NodeJS

Customer API

v1.0.0

Objetivos

Avaliar a qualidade técnica e comportamental do candidato para o papel de desenvolvedor NodeJS através do desenvolvimento de um mini projeto.
Este projeto deve ser versionado em um repositório público do Github e compartilhado com o recrutador.

Os seguintes quesitos serão avaliados:

Qualidade de código

O candidato deve construir o projeto de acordo com as stack definidas nas premissas e descrição do projeto. Serão avaliados os critérios:

- Boas práticas de código (DRY, KISS, YAGNI, SoC, Design Patterns)
- Legibilidade do código
- Performance e eficiência
- Aderência a stack proposta

Interpretação da documentação

Faz parte da avaliação o entendimento dos requisitos, premissas e restrições presentes nesta documentação. Todos os diagramas seguem o padrão C4 Model ou UML Sequence Diagrams.

Testes

Será avaliado a cobertura de testes unitários do código apresentado e a facilidade de serem adicionados novos códigos.

Sustentação

Será avaliada a facilidade de adicionar novas funcionalidades ao projeto.

Escopo

Será avaliado quanto do escopo proposto foi implementado, e se o candidato se manteve dentro do escopo proposto.

Premissas e Restrições

Stack de desenvolvimento

- NodeJS
- NestJS
- Axios
- Typescript (não obrigatório)

Testes

Os testes devem ser implementados utilizando o Jest.

Documentação

Forneça toda a documentação necessária no README.md descrevendo como construir, executar localmente e o processo de instalação em ambiente produtivo.
(*Devemos ser capazes de realizar todo o processo de avaliação do projeto sem a assistência do candidato*).

Observação: deve-se usar imagens docker para execução.

Infraestrutura

Para armazenamento dos dados, deve-se utilizar o Redis, e a conexão da aplicação com o Redis deve ser feita utilizando a biblioteca ioredis.

Para utilizar o Redis, recomendamos que o candidato utilize um container local em docker. Os comandos para fazer isso via terminal são:

- `docker pull redis`
- `docker run -d -p 6379:6379 redis`

Requisitos de negócio

Devido a um grande crescimento, apareceu uma necessidade de um cadastro performático e seguro para os clientes da empresa.

O escopo desse projeto é a implementação do backend desse sistema, que deve ser uma RESTful API que suporta as seguintes operações:

- Salvar um cliente novo
- Atualizar um cliente existente
- Buscar um cliente por ID

Para garantir a segurança dos dados cadastrados, a empresa disponibiliza uma API para autorização de acesso às operações disponíveis.

Este serviço já está desenvolvido e não entra no escopo deste projeto, as instruções de utilização podem ser encontradas neste documento.

Requisitos de técnicos

Implementar o componente Customer API. As chaves no cache deverão ser prefixadas com ‘customer:’. Ex: ‘`customer:814e93e6-c9c4-4a39-bc4a-518ab47d6278`’.

Diagrama de contexto

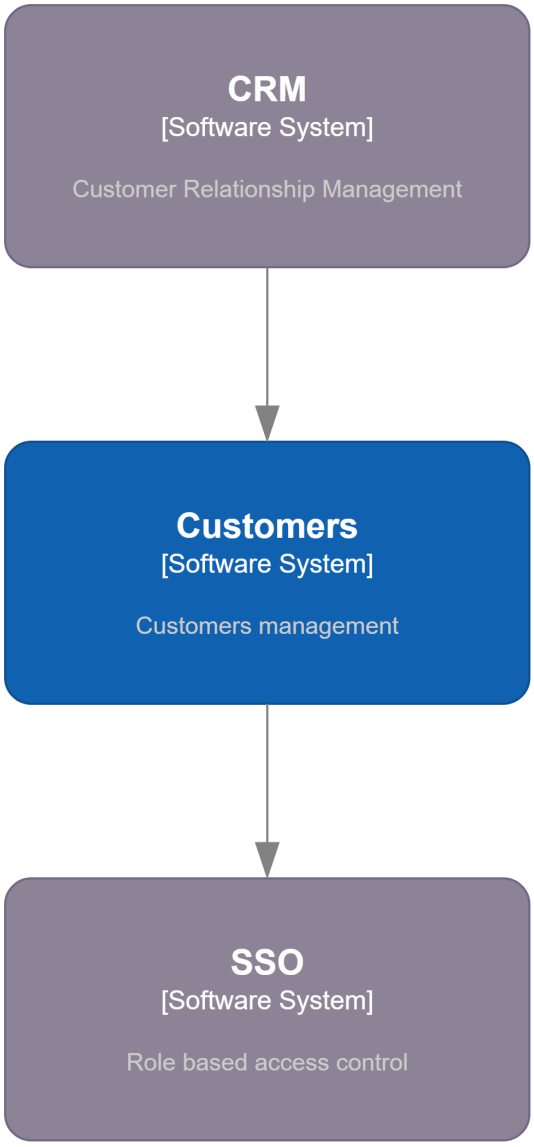


Diagrama de contêiner

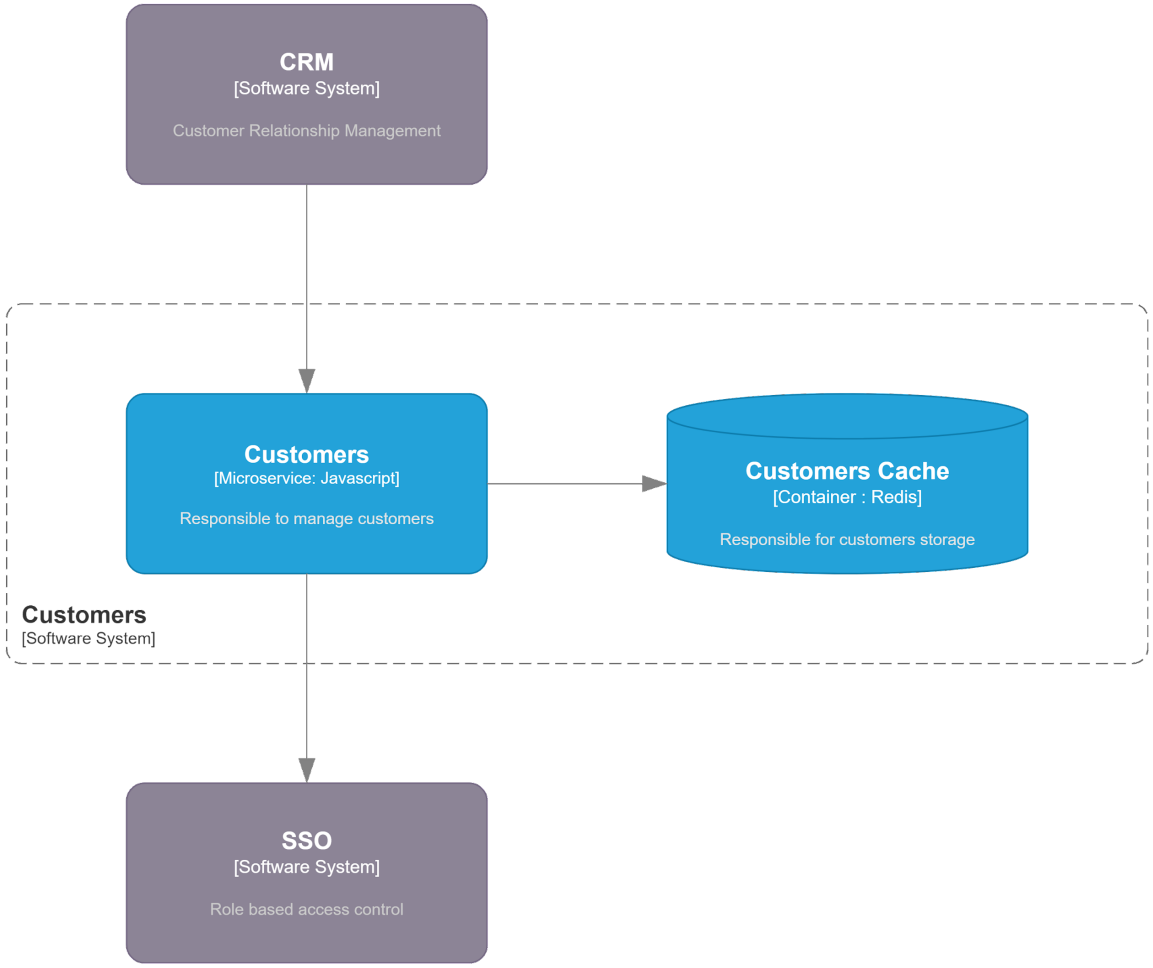


Diagrama de componente

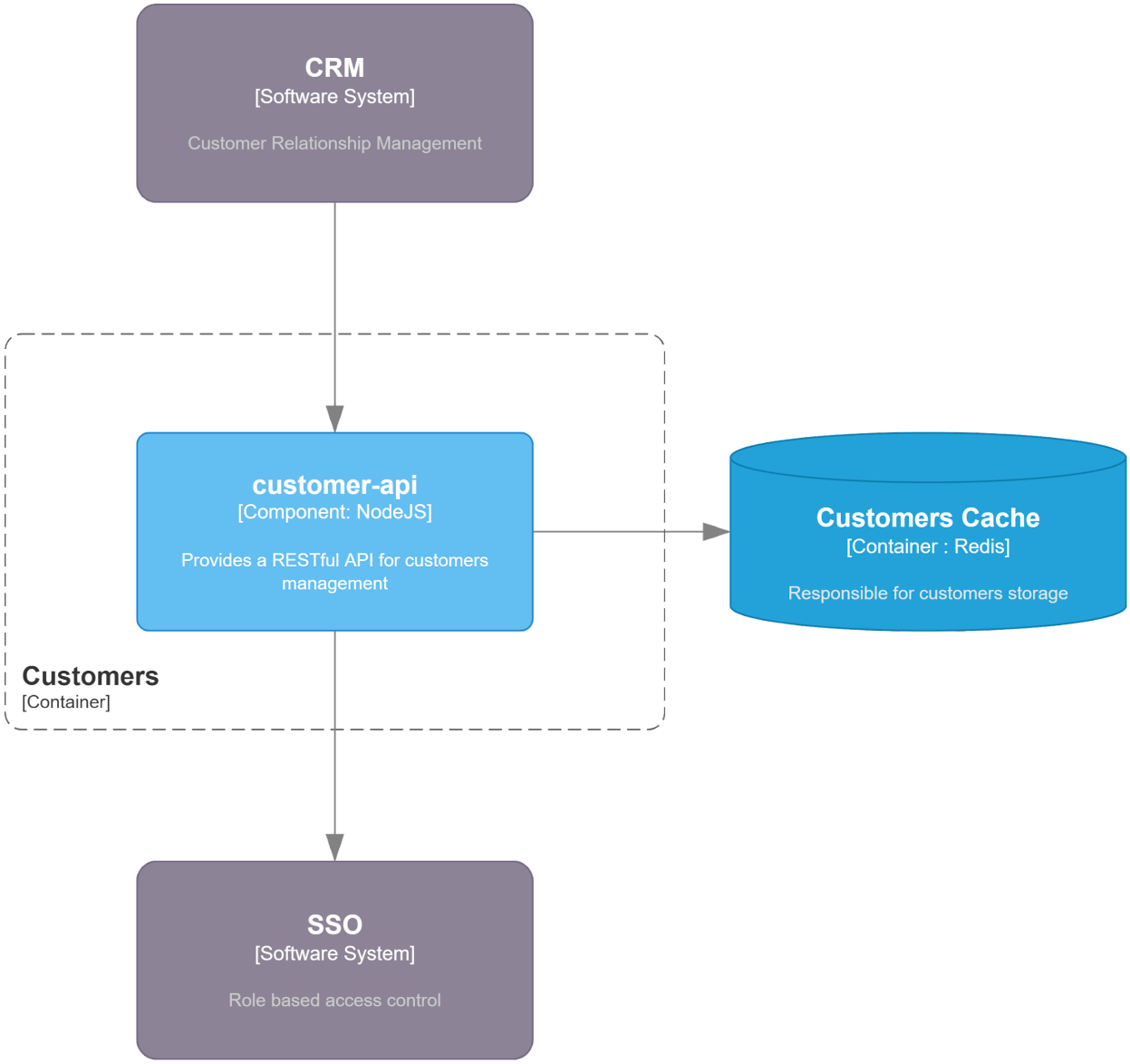


Diagrama de código

Faz parte da avaliação a capacidade do candidato organizar o código, portanto o diagrama de código não será fornecido.

Operações

Salvar um novo cliente

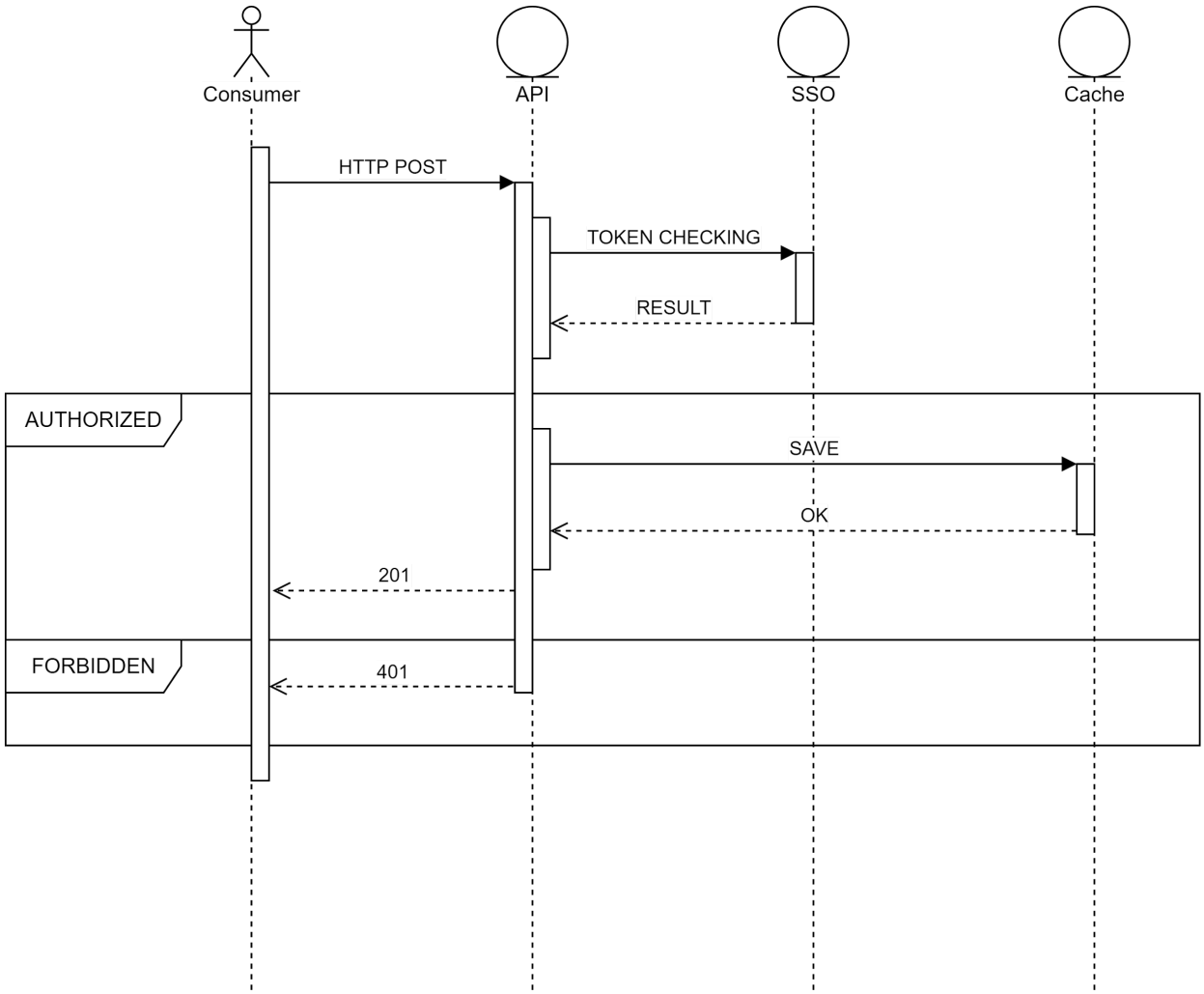
A criação de um novo cliente deve ter seu ID gerado pelo sistema no formato UUIDv4.

Endpoint	/customers
Method	POST
Headers	Content-Type: application/json Authorization: Bearer <token>

Request	{
	"document": "<number>",
	"name": "<string>"
	}

Response	● HTTP 201 ○ Customer ● HTTP 401 - não autorizado ● HTTP 400 - request inválida ● HTTP 502 - cache indisponível ● HTTP 502 - sso indisponível

Diagrama de sequência



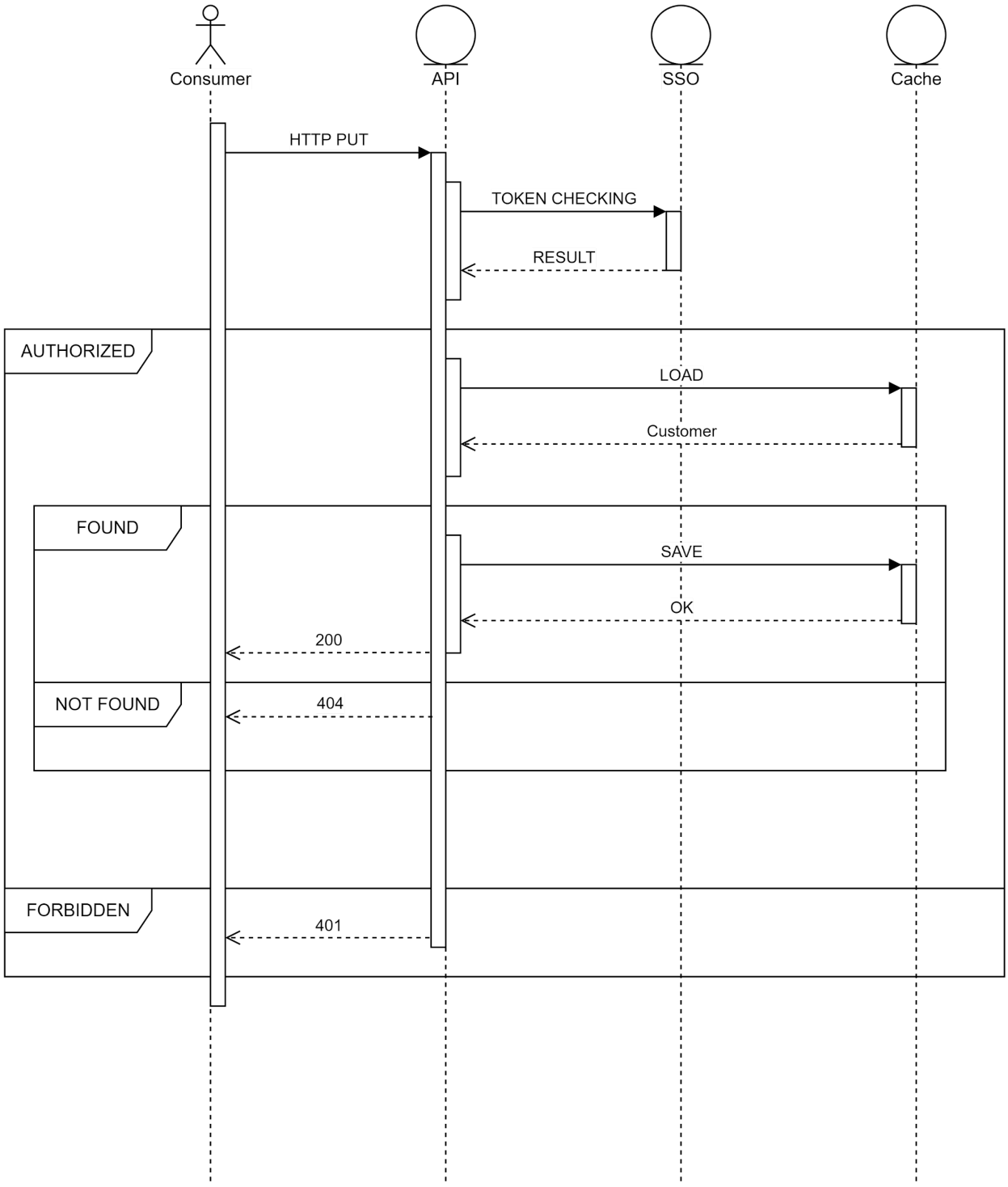
Atualizar um cliente

Endpoint	/customers/:id
Method	PUT
Header	Content-Type: application/json Authorization: Bearer <token>

Request	{
	"id": "<uuid v4>",
	"document": "<number>",
	"name": "<string>"
	}

Response	● HTTP 200 ○ Customer ● HTTP 401 - não autorizado ● HTTP 400 - request inválida ● HTTP 404 - cliente inexistente ● HTTP 409 - conflito de ID ● HTTP 502 - cache indisponível ● HTTP 502 - sso indisponível

Diagrama de sequência

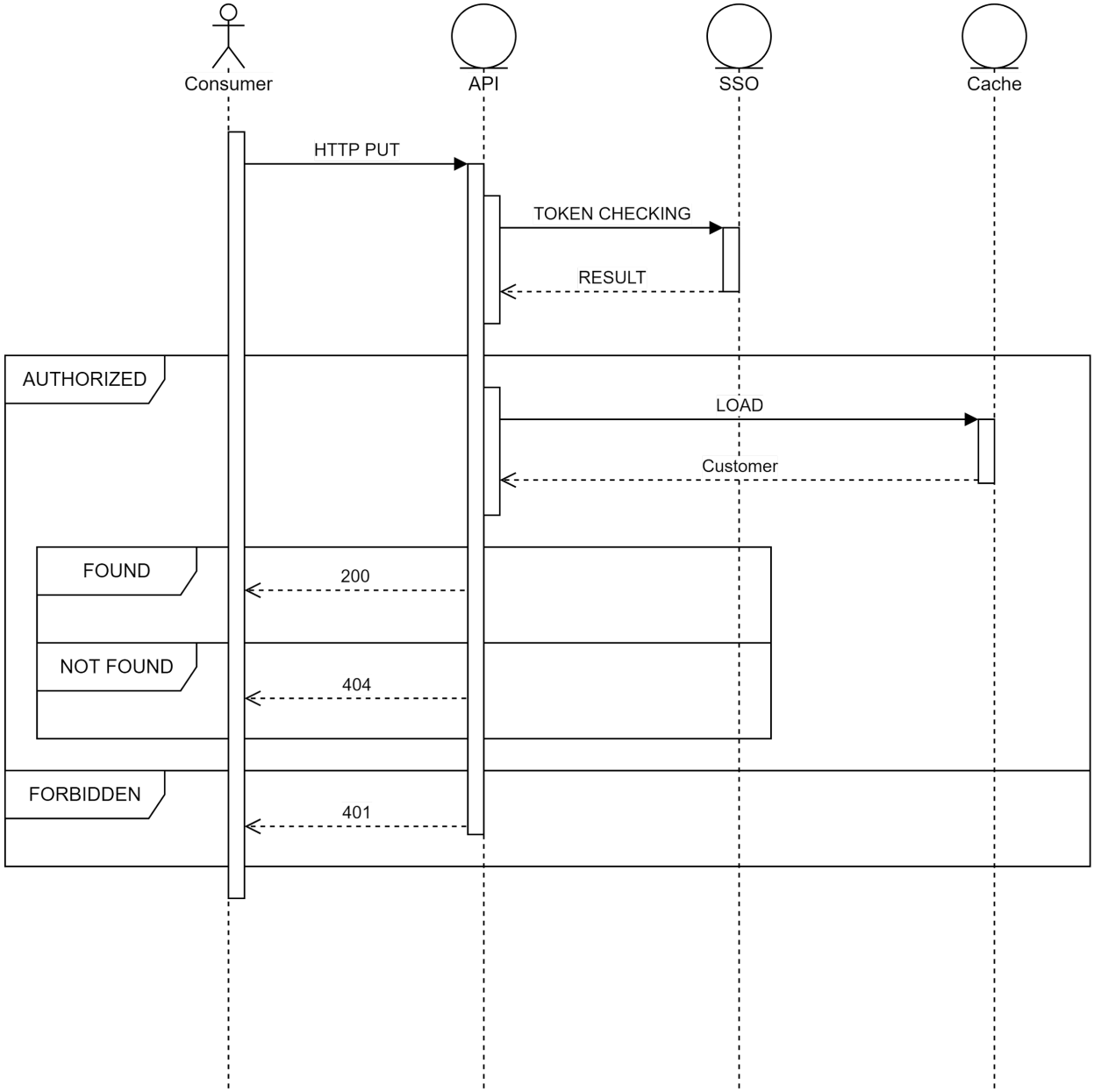


Buscar um cliente por ID

O sistema deve disponibilizar um endpoint para buscar um cliente cadastrado por ID.

Endpoint	/customers/:id
Method	GET
Header	Content-Type: application/json Authorization: Bearer <token>
Request	N/A
Response	● HTTP 200 ○ Customer ● HTTP 401 - não autorizado ● HTTP 404 - cliente inexistente ● HTTP 502 - cache indisponível ● HTTP 502 - sso indisponível

Diagrama de sequência



SSO

URL
https://accounts.seguros.vitta.com.br/

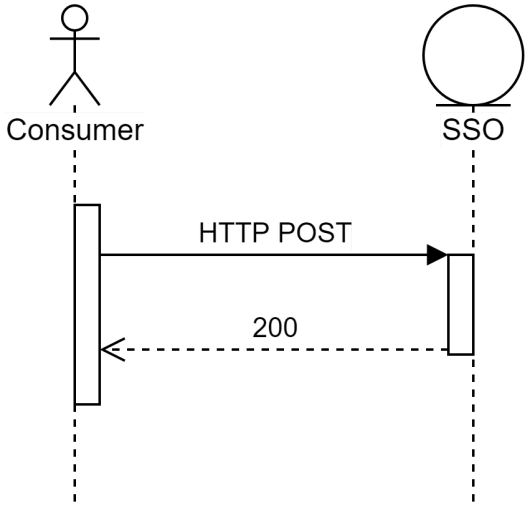
Autenticação

O token na resposta deve ser utilizado para autorização nas operações de clientes deste teste.

Endpoint	/auth/realms/careers/protocol/openid-connect/token
Method	POST
Header	Content-Type: application/x-www-form-urlencoded
Request	grant_type=client_credentials client_id=customers client_secret=453000f7-47a0-4489-bc47-891c742650e2 username=<seu_email> password=<base64_de_seu_email> scope=openid

Response	{
	"access_token": "<token>",
	"expires_in": 300,
	"refresh_expires_in": 0,
	"token_type": "Bearer",
	"not-before-policy": 0,
	"scope": "openid profile email",
	"Id_token": "..."
	}

Diagrama de sequência



Terminologia

API	Application Programming Interface
CRM	Customer Relationship Management
HTTP	Hypertext Transfer Protocol
HTTPS	HTTP over SSL
ID	Identification
RBAC	Role Based Access Control
REST	Representational State Transfer
RESTful	REST implementation
SSL	Secure Socket Layer
SSO	Single Sign On
UUID	Universally Unique ID
UUIDv4	UUID Version 4