Gerenciador de Web Services e Organizador de Requisições

Aluno: Matheus Heiden

Orientadora: Luciana Pereira de Araújo Kohler

FURB
UNIVERSIDADE DE BLUMENAU

Roteiro

- Introdução
- Objetivos
- Fundamentação Teórica
- Trabalhos Correlatos
- Requisitos
- Especificação
- Implementação
- Resultados
- Conclusões

Introdução

- Crescimento da implementação e utilização de Web Services;
- O que faz uma API:
  - expõe serviços ou dados à um consumidor;
  - é uma interface de software para software;
  - utiliza contratos durante a comunicação.

Introdução

- Com o maior interesse de empresas e clientes à integração de aplicações, cresce também a demanda aos servidores;
- Essa demanda pode causar lentidão ou até indisponibilidade da aplicação;

Objetivos

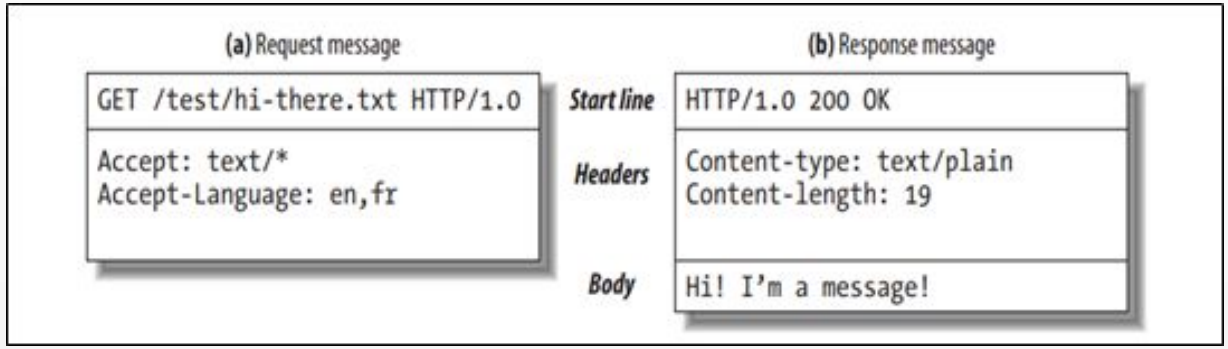
- Tem como objetivo operar entre o cliente e o Web Service, podendo receber requisições REST e SOAP, que as prioriza, valida, e encaminha ao Web Service

Objetivos

- Fornecer um ponto de entrada que recebe requisições;
- disponibilizar uma interface para o registro de Web Services;
- definição de prioridade e regras de utilização de requisições;
- algoritmo de priorização de requisições;
- algoritmo para definição, execução e limitação de requisições.

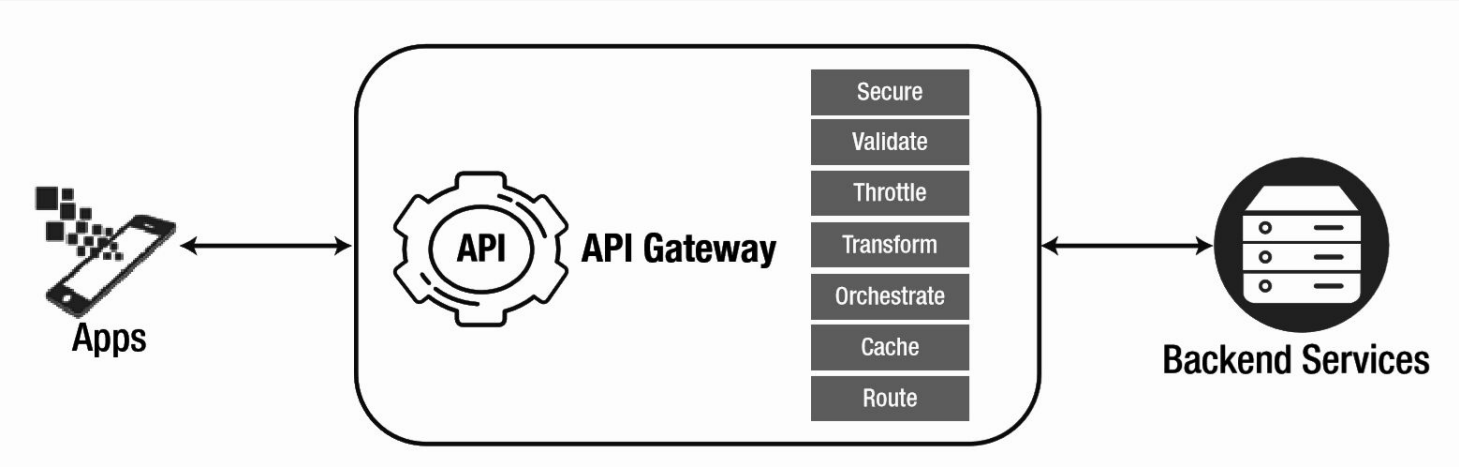
Fundamentação Teórica

- Servidores Web;
- Protocolo HTTP;
- Servidores de aplicação;
- Conteúdo dinâmico.
- Bancos de Servidores
- Carga de requisições



Fundamentação Teórica

- Gateway de API;
- Disponibiliza uma interface entre aplicação e Web Service
- Classificação de Requisições;
- Regulação e validação



Trabalhos Correlatos

Características	**Overeinder, Verkaik e Brazier (2008)**	**Mulesoft (2018)**	**Microsoft (2018)**
Controle de requisições	Sim	Sim	Sim
Gerenciamento de mais de um Web Service	Não	Sim	Sim
Gratuito	Sim	Não	Não
Priorização de recursos	Não	Não	Não
Requer implementação	Sim	Não	Não
Protocolos e arquiteturas suportadas	SOAP	SOAP, RESTful	SOAP, RESTful
Implementação modular	Não	Sim	Não

Requisitos

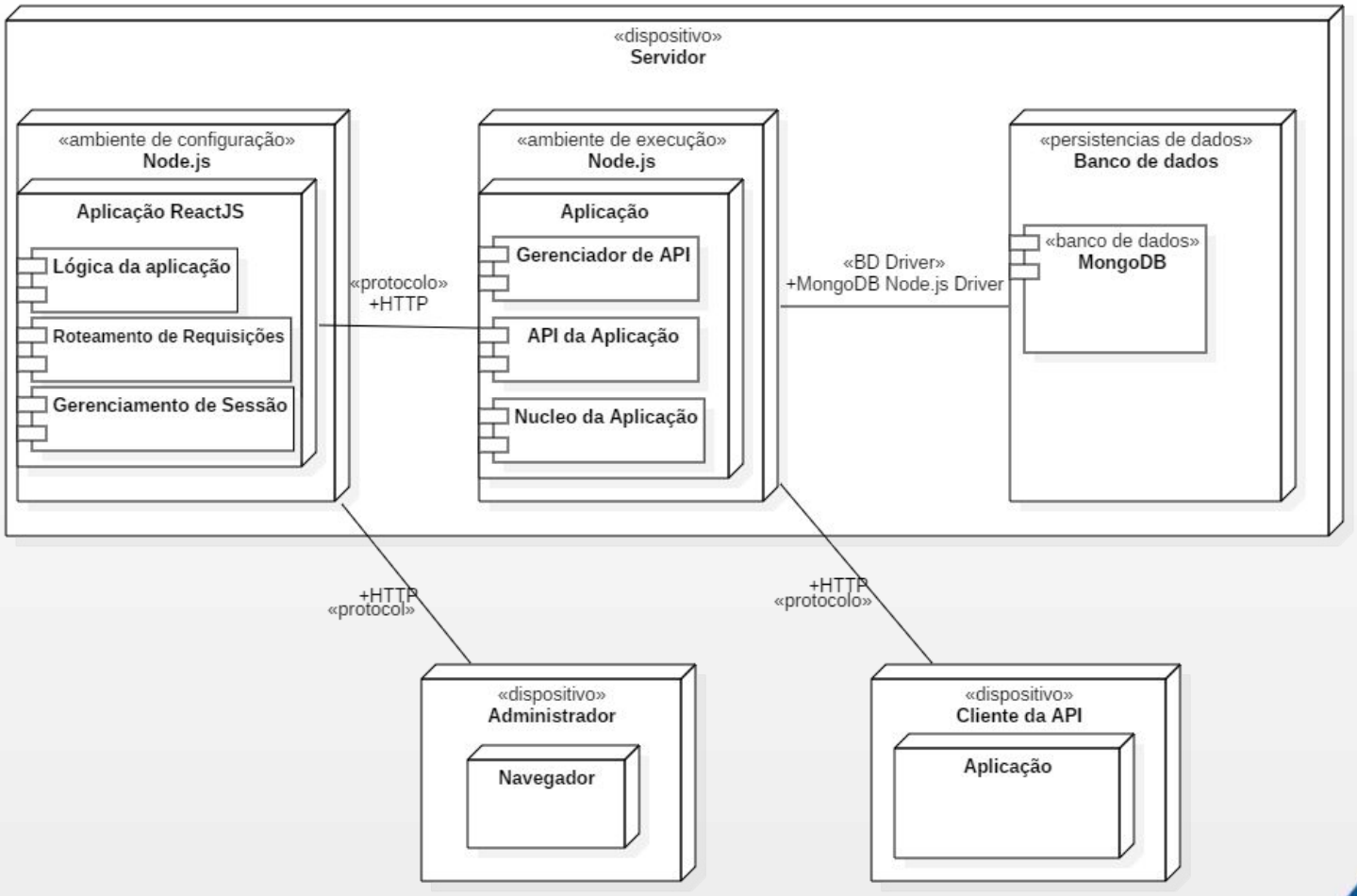
Requisitos Funcionais da Aplicação
RF01: possibilitar o registro de Web Services SOAP
RF02: possibilitar o registro de Web Services REST
RF03: possibilitar o registro de requisições
RF04: possibilitar o registro de classificações
RF05: possibilitar o registro de regras
RF06: classificar requisições recebidas pela aplicação (baixa prioridade, alta prioridade)
RF07: receber requisições feitas por um cliente
RF08: repassar requisição para a aplicação destino
RF09: mitigar o envio contínuo e ininterrupto de requisições, levando em consideração regras pré-definidas
RF10: permitir implementação modular possibilitando adição de funcionalidades ou remoção

Requisitos

Requisitos Não-Funcionais da Aplicação
RNF01: utilizar a linguagem de programação JavaScript dentro do ambiente Node.js
RNF02: implementar a persistência de dados em um banco não relacional MongoDB
RNF03: utilizar o ambiente de desenvolvimento Visual Studio Code

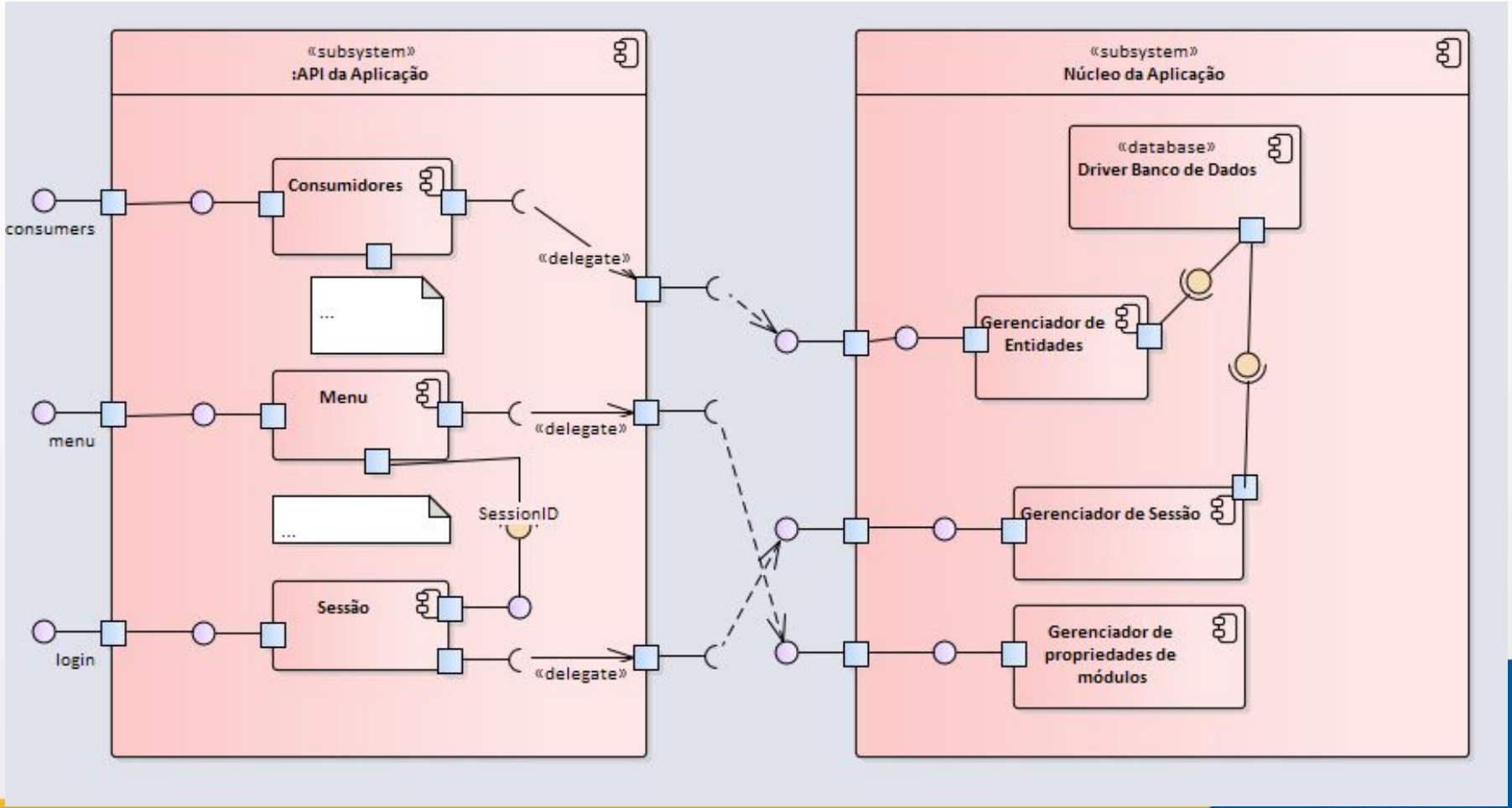
Especificação

- Diagrama de deployment



Especificação

- Diagrama de Componentes

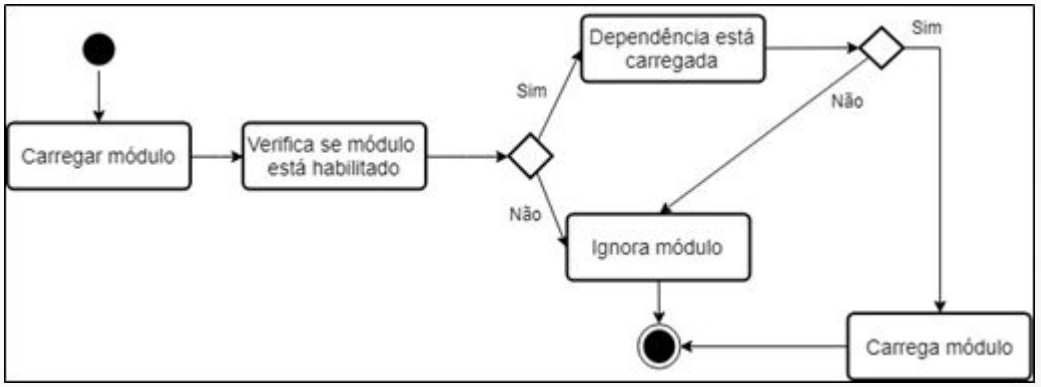


Implementação

- Aplicação base dividida em 6 módulos
- Implementada em JavaScript
- Utilizando a máquina virtual implementada pelo NodeJS
- Persistência de dados através do Banco de Dados MongoDB

Implementação

- Modularidade auxiliada pela implementação de factory e observer;
- O módulo base da aplicação é o módulo core
- Responsável pelo carregamento de módulos



Implementação

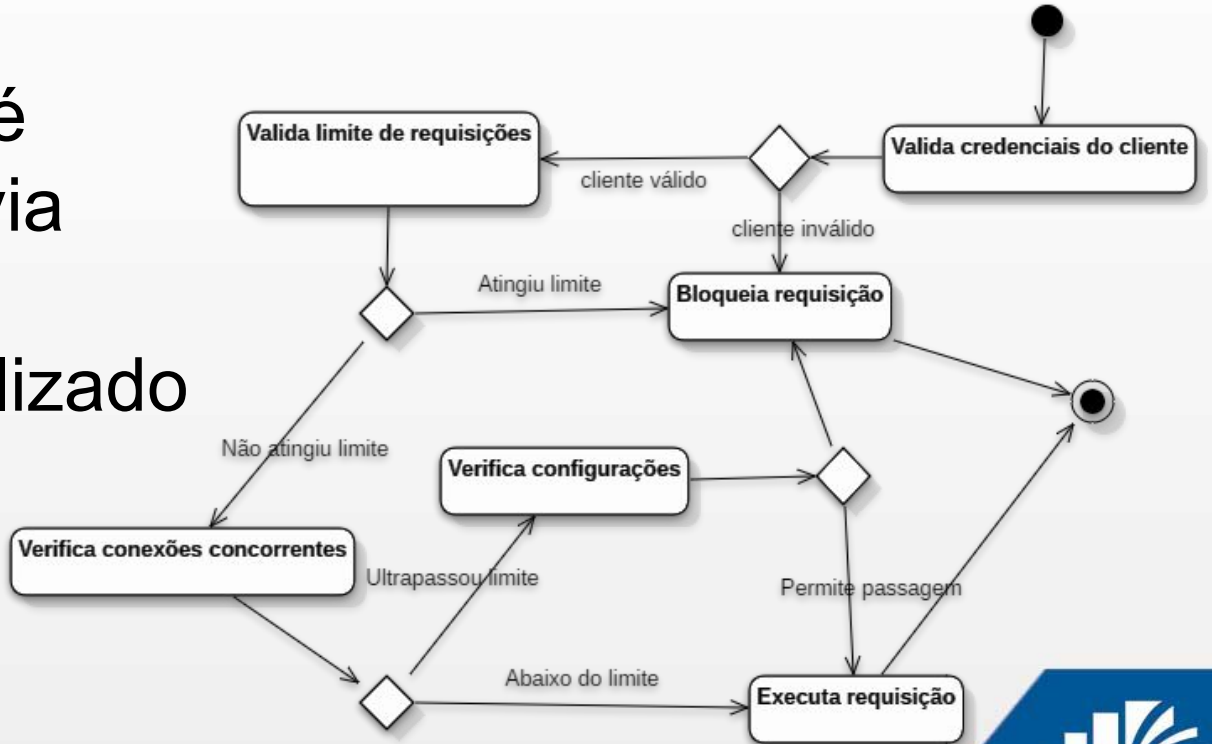
- Módulo api_manager
- Ativa o suporte a gerenciamento de API
- soap_api adiciona suporte ao protocolo SOAP
- rest_api adiciona suporte a RESTful

```
{
    "global" : {
        "api_manager" : [
            {
                "type" : "soap",
                "handler" : "request"
            }
        ]
    }
}
```

```
{
    "module":{
        "name": "web",
        "version":"0.1.0",
        "active": true
    }
}
```

Implementação

- A requisição precisa ter seu cliente definido
- A aplicação é descoberta via hostname
- Algoritmo utilizado para validar requisições



Implementação

```
1  Project.log('Current concurrent request limit: '+this.concurrent_limit_prevent)
2  if (this._verifyPendingActions() > this.concurrent_limit_prevent ) {
3      this._denyExec('Too many requests')
4      return x._postDispatch()
5  }
6
7  let delayCounter = 0
8  while (this.flags.includes(DELAY_EXECUTION) &&  //if this action is supposed to be delayed
9      this.application.shouldDelay() && //if the application has delay time configured
10     this._verifyPendingActions() > this.delay_with_qty
11 ) { // and if current actions breaks the limit of concurent action
12     if (delayCounter == this.delay_limit) {
13         if (!this.allow_passtrough) {
14             this.flags.push(PREVENT_EXECUTION)
15         }
16         break
17     }
18     this._delay()
19     x.wasDelayed = true
20     delayCounter++
21
22 }
```

Resultados e Discussões

- Testes foram efetuados em ambiente cloud
- Testes de Carga
- Testes de tempo de requisição
- Servidor de Aplicação do gerenciador de API
- Teste Comparativos

Núcleos de Processadores	4 núcleos
Processador	Intel Xeon E3-12xx
Frequência	2.5Ghz
Memória RAM	3.70GB
Distribuição	CentOS 7
Versão do Kernel	3.10.0

Resultados e Discussões

API SOAP

Com Gerenciador de API	Sem Gerenciador de API	Diferença
1332	837	495
1309	810	499
1156	927	229
937	945	-8
850	802	48
	Média	**161.2857143**
	Mediana	**123**
	Desvio Padrão	**269.3093791**

API REST

Com Gerenciador de API	Sem Gerenciador de API	Diferença
1044	1037	7
1100	1535	-435
1160	1004	156
1226	1027	199
958	1013	-55
1056	1736	-680
	Média	**17.34**
	Mediana	**124.5**
	Desvio Padrão	**414.4709246**

Resultados e Discussões

- Carga média do processador sobe com a quantidade requisições
- Consumo de memória se mantém estável

Requisições simultâneas	Carga média	Uso de memória
5 Clientes	0.2	630M
10 Clientes	0.4	635M
15 Clientes	0.59	648M
30 Clientes	0.71	655M

Resultados e Discussões

- Foram feitos benchmarks comparativos
- Configurações Agressivas
- Configurações conservadoras

Configuração	Agressivo	Conservador
Limite de requisições para Delay	20	30
Limite de Requisições Concorrentes	25	30
Limite de Delay de Requisições	5	30
Permitir Passagem de Requisições	Não	Sim
Delay de Requisições	10s	5s

Resultados e Discussões

- Processamento da aplicação destino:

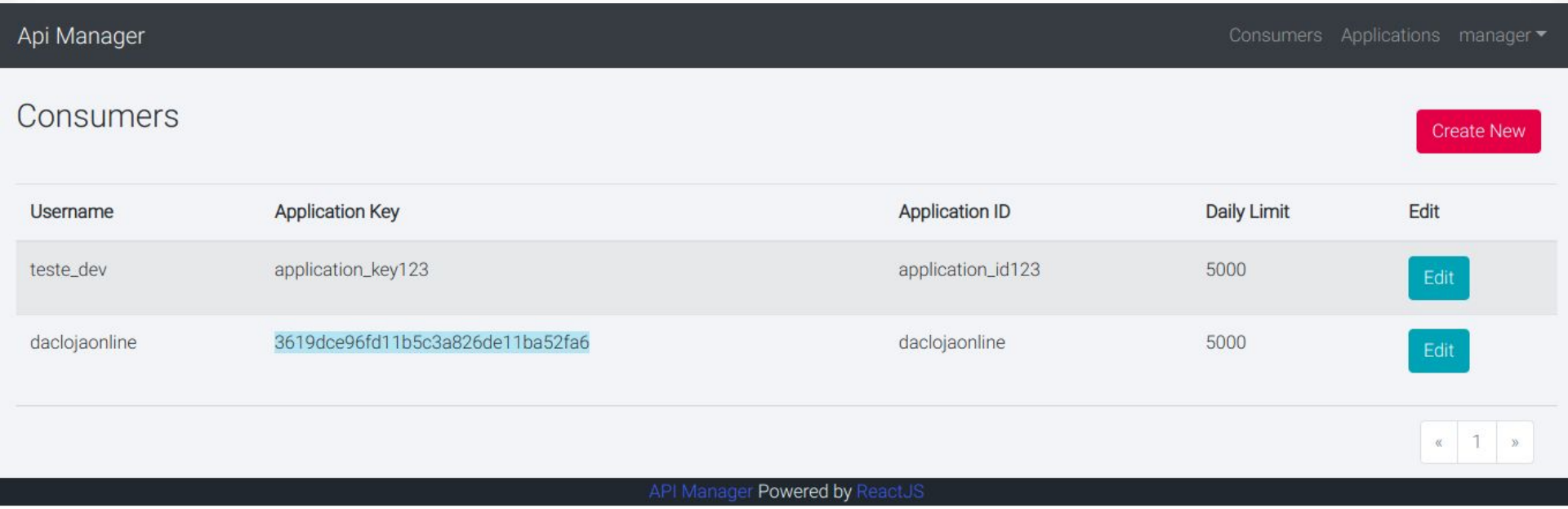
Requisições simultâneas	Sem gerenciador de API	Com gerenciador de API (conservador)	Com gerenciador de API (agressivo)
5 clientes	20%	18%	16%
10 clientes	35.65%	33.57%	29.55%
15 clientes	49.90%	41.43%	48.4%
30 clientes	74.40%	71.65%	37.2%

Resultados e Discussões

Características	Overeinder, Verkaik e Brazier (2008)	Mulesoft (2017)	Microsoft (2018)	Heiden (2018)
Controle de requisições	Sim	Sim	Sim	Sim
Gerenciamento de mais de um Web Service	Não	Sim	Sim	Sim
Gratuito	Sim	Não	Não	Sim
Priorização de recursos	Não	Não	Não	Sim
Requer implementação	Sim	Não	Não	Não
Protocolos e arquiteturas suportadas	SOAP	SOAP, RESTful	SOAP, RESTful	SOAP, RESTful
Implementação modular	Não	Sim	Não	Sim

Resultados e Discussões

Aplicação consumidora da API



Conclusões

- O objetivo principal e os específicos foram alcançados;
- Baixo overhead no tempo de requisição;
- Customizável e expansível ;
- Correlação entre a carga de um servidor e o uso do gerenciador de API.

Sugestões

- Incluir suporte a outros protocolos e arquiteturas
- Melhoria na implementação do roteamento de requisições suportar também caminhos;
- Suportar servidores com menos recursos;
- implementar utilização de cache de respostas.