



**ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO**

AVALIAÇÃO CONTÍNUA:
Trabalho Prático Microserviços
Agendamento para Clínicas

Alonso Lima Machado
(Nº 8190006)

Trabalho Prático apresentado no âmbito da unidade curricular de Métodos e Técnicas de Suporte ao Desenvolvimento de Software, 1º ano do Mestrado em Engenharia Informática

Docente: Prof. Doutor Ricardo Santos
(rjs@estg.ipp.pt)

2019-2020

ÍNDICE GERAL

| | |
|---|-----------|
| 1.INTRODUÇÃO | 4 |
| 2. O SISTEMA | 4 |
| 2.1. Stack tecnológica | 4 |
| 2.1.1. Node.js | 4 |
| 2.1.1.1 Express | 4 |
| 2.1.2. Redis | 5 |
| 2.2. Arquitetura | 6 |
| 2.2.1 Modelo Inicial | 6 |
| 2.2.1 Arquitetura Final | 7 |
| 2.3. Microserviços | 7 |
| 2.3.1 Agendamento | 7 |
| 2.3.2 CheckIn | 7 |
| 2.3.3 Pagamento | 8 |
| 2.3.4 Front End | 8 |
| 2.3.4.1 Agendamento | 8 |
| 2.3.4.2 CheckIn | 10 |
| 2.3.4.3 Pagamentos | 11 |
| 2.3.5 API Gateway - Ingress | 12 |
| 2.3.6 Mailer - Não finalizado | 12 |
| 3. CONCLUSÃO | 12 |
| 3.1 Considerações sobre o Trabalho Prático | 12 |
| 3.2 Passo a Passo | 12 |
| 3.2.1. Starte seu minikube com no minimo 4GB de RAM. | 12 |
| 3.2.2. Execute o criar.sh | 12 |
| 3.2.3. minikube ip | 12 |
| 3.2.4. Entre no seu navegador no ip do minikube ip | 12 |
| 3.2.5. Observações Importantes | 12 |
| 3.2.6. Para deletar execute o deletar.sh | 12 |
| 3.2.7. Para completar faltou o microserviço de email para contactar o cliente antes | 13 |
| REFERÊNCIAS | 14 |
| ANEXOS | 15 |
| Anexo A. Avaliação Época Normal | 16 |
| Anexo B. Minikube DashBoard | 17 |

1.INTRODUÇÃO

Este trabalho prático insere-se no âmbito da Unidade Curricular de Métodos e Técnicas de Suporte ao Desenvolvimento de Software, e o tema que o escolhi foi sistema de agendamento de consultas online.

A ideia deste sistema surgiu da necessidade de se marcar consultas por telefone e/ou presencialmente, mas a ideia evoluiu para a parte da gestão administrativa dos agendamentos e pagamentos dessas consultas.

Para a concretização desta ideia propus uma arquitetura de micro serviços, que foi decomposta pelas funções do negócio tentando seguir SOA/DDD (Agendar, Relizar e Pagar), e para qual foi utilizado Node.js com o framework Express para implementar os micro serviços e redis para a comunicação como publisher/subscriber.

2. O SISTEMA

2.1. Stack tecnológica

2.1.1. Node.js

Node.js é uma framework de programação baseado em Javascript, criada para ser altamente escalável e adaptável, é baseada no V8 engine javascript do Google Chrome e tem uma comunidade gigantesca mantenedora, ele tem uma arquitetura baseada em Eventos, usando uma técnica chamada loop de eventos que é totalmente assíncrono ele interpreta as requisições de forma assíncrona e não permite bloqueios, mas contudo ele é single-thread até a sua versão 12 onde está sendo testado uma forma de multithreading experimental no node.js.

Perfeito para criar APIs que é basicamente oque é um microserviço, várias APIs que tratam de forma separada seu domínio.

Maturidade: Netflix, Walmart, Paypal, LinkedIn, Mozilla, Yahoo e outras companhias usam bastante Node.js em sua stack.

2.1.2 Express

Framework para facilitar requisições HTTP: POST, GET, DELETE, UPDATE de forma mais fácil e rápida, deixa o código mais legível.

É o framework mais utilizado para desenvolvimento back-end para APIs em node.js.

Muito material para estudo.

<https://expressjs.com/pt-br/guide/database-integration.html>

2.1.3. Redis

Banco de dados em memória, ele tem várias funções interessantes, neste trabalho usarei como banco de dados comum e principalmente a sua função PUBLISHER/SUBSCRIBER que cria canais de comunicação semelhante o Kafka ou rabbitmq.

2.1.4. Front End - Express HandleBars

Existem várias opções como PUG, Mustache, Handlebars e EJS.

Framework Front-End Handlebars, especialmente feito para Express, pois Express não tem uma view Engine definida.

Muito material para estudo.

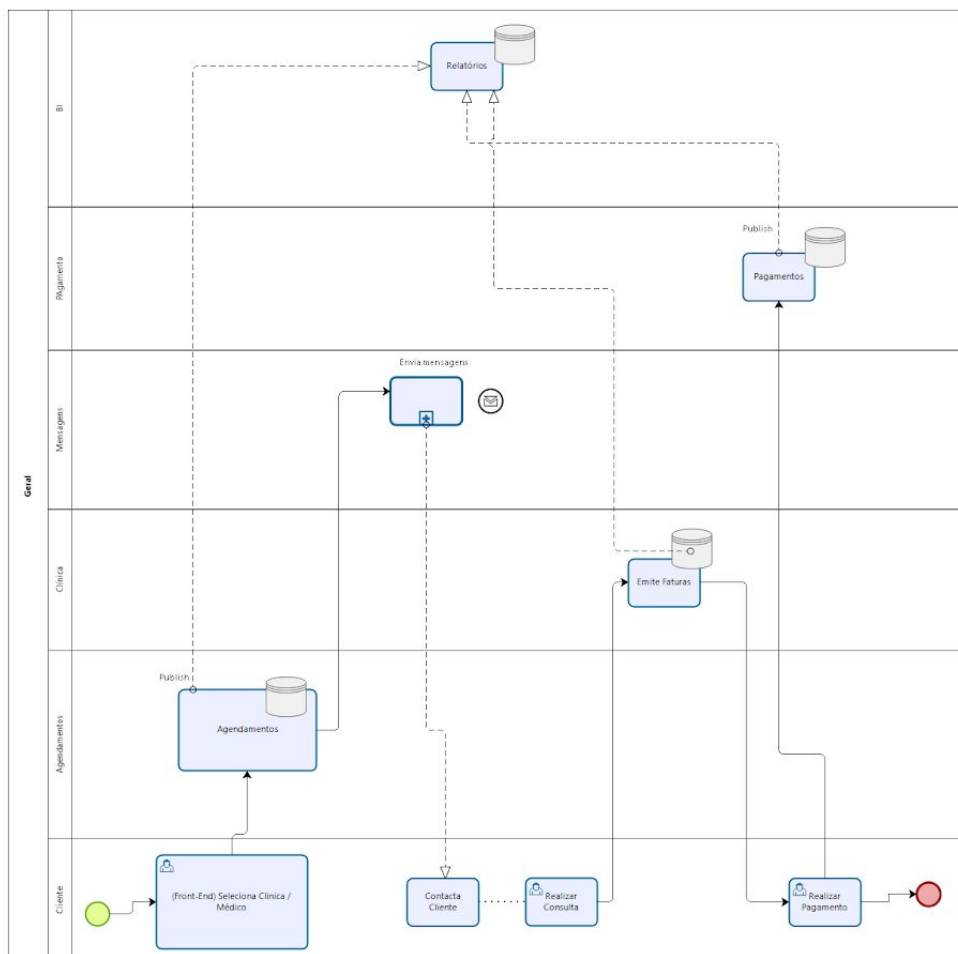
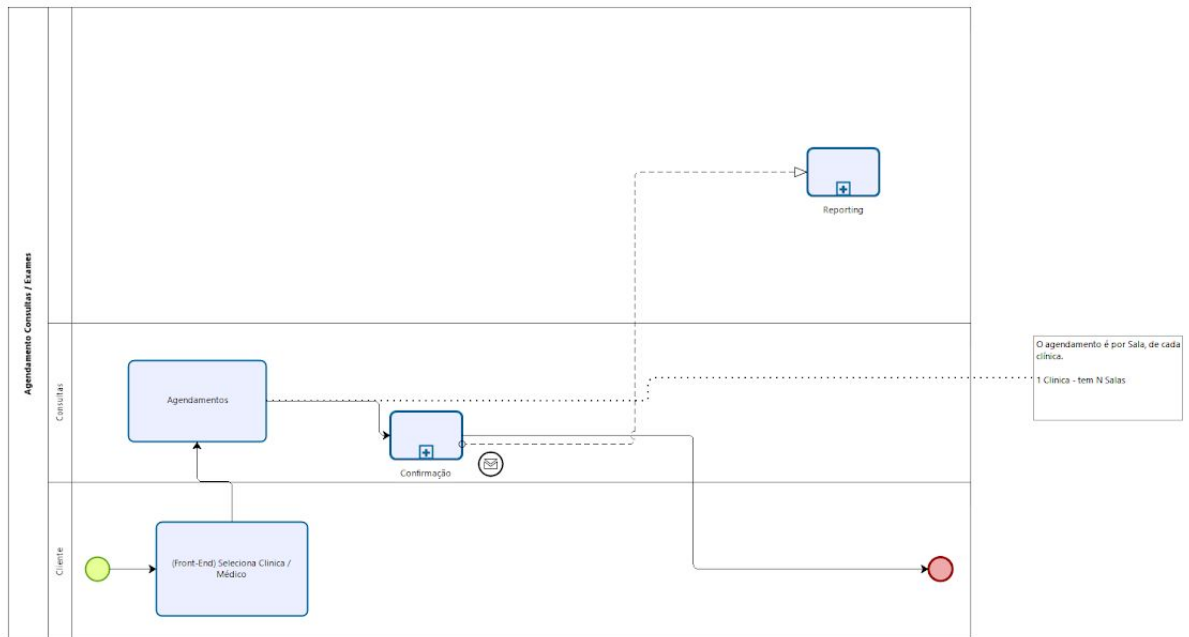
<https://handlebarsjs.com/>

<https://www.npmjs.com/package/express-handlebars>

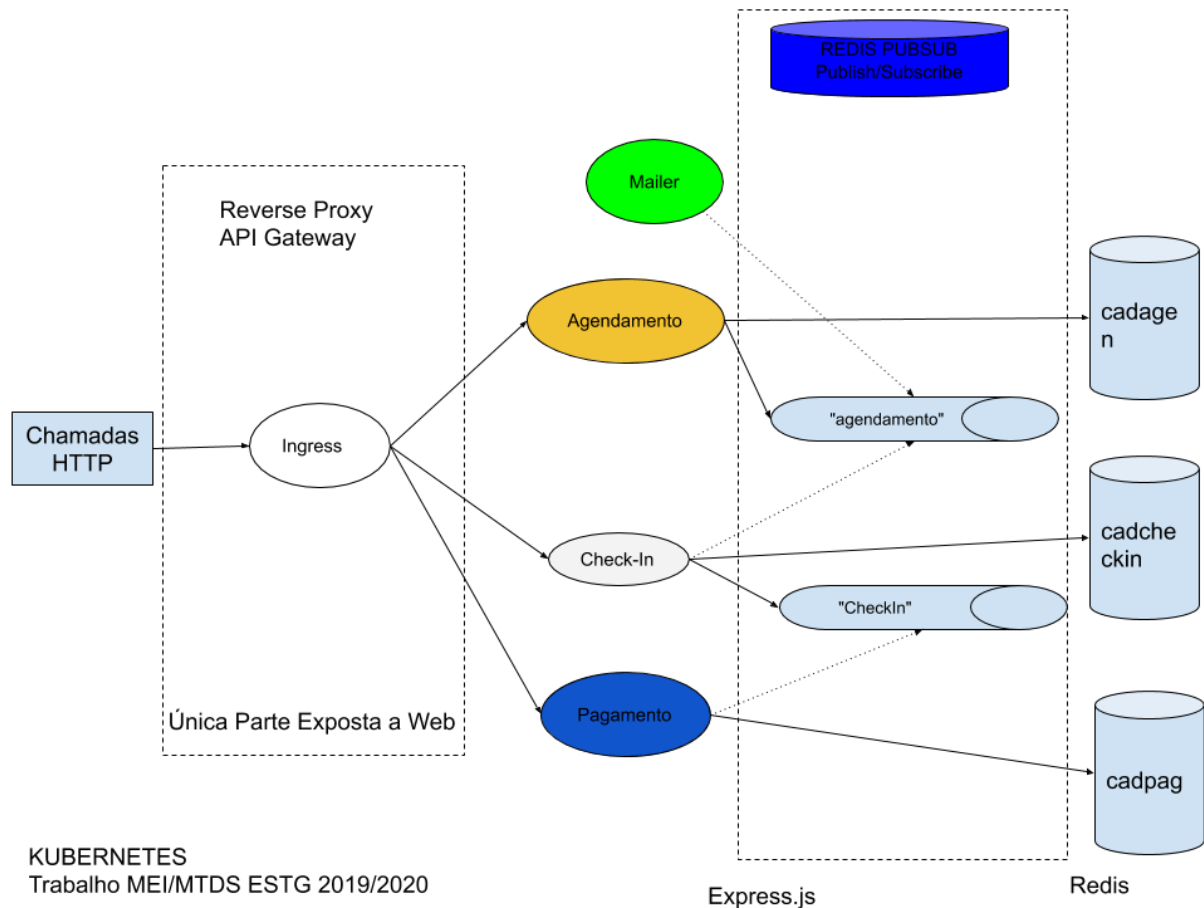
<https://expressjs.com/en/guide/using-template-engines.html>

2.2. Arquitetura

2.2.1 Modelo Inicial



2.2.1 Arquitetura Final



2.3. Microserviços

Resumidamente os microserviços foram separados por funcionalidade para podermos garantir a independência do sistema caso existir uma falha.

2.3.1 Agendamento

Microserviço que lida somente com agendamentos, tem alguns endpoints rest:

- POST Salva o agendamento novo na lista interna cadagen e publica no redis PUBSUB na lista "agendamento";
- GET ID Pega o agendamento com aquele id
- GET Lista Retorna todos os agendamentos da lista

2.3.2 CheckIn

Microserviço que lida somente com realizações de consulta, ele escuta as listas do redis PUBSUB em "agendamento:braga" ou "agendamento:porto" e tem alguns endpoints rest:

- POST Salva o realização nova na lista e publica no redis PUBSUB na lista "checkIn", dependendo de qual for o seu endpoint.
- GET ID Pega a realização com aquele id
- GET Lista Retorna todos os checkin da lista

2.3.3 Pagamento

Microserviço que lida somente com pagamentos de consulta, ele escuta as listas do redis PUBSUB em "CheckIn" e tem alguns endpoints rest:

- POST Salva o pagamento nova na lista interna e depois no redis dele cadpag.
- GET ID Pega a realização com aquele id
- GET Lista Retorna todos os pagamentos da lista

2.3.4 Front End

Microserviço que lida com as requisições do usuário e mostra os formulários e listas.

- Funcionalidade LOGIN (/logar e /logout), está implementada como variável Global no back-end, deverá ser trocada em implementação futura por COOKIE de SESSÃO.

2.3.4.1 Agendamento

- /agendamentos/todos - Mostra em formato de Lista no Front-end todos os agendamentos gravados pelo microserviço Agendamento.

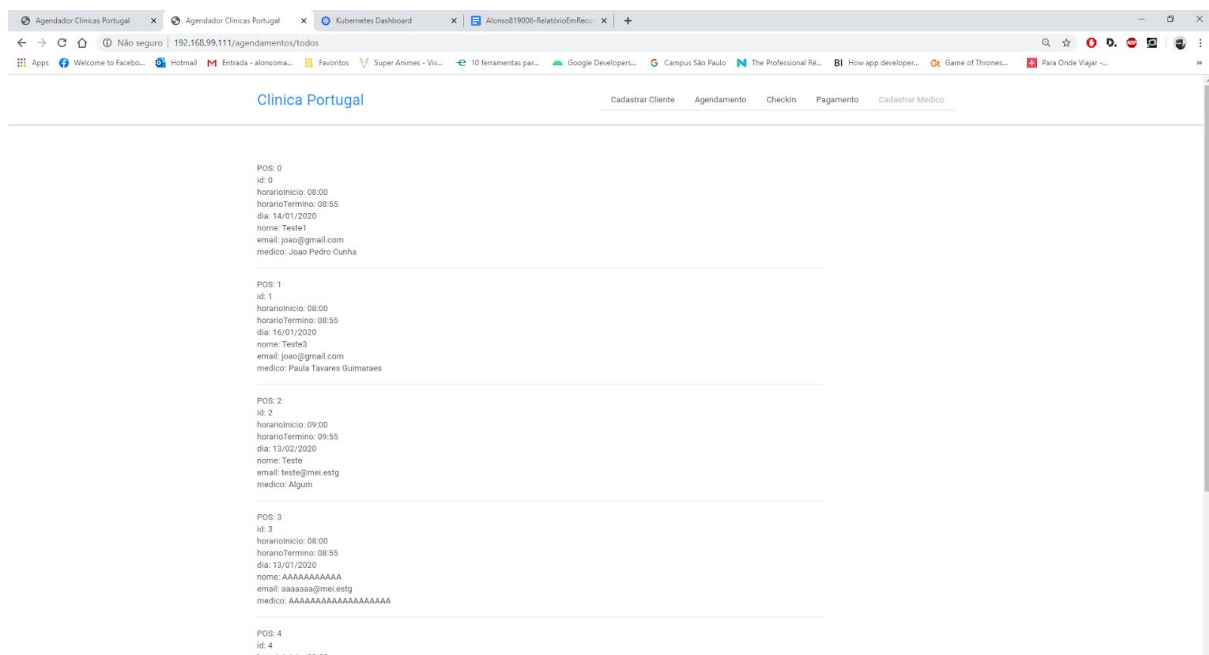


Figura Lista de Agendamentos

- /agendamento/ - Página para adicionar um Agendamento. (Não é necessário estar logado)

The screenshot shows a web browser window with the URL 192.168.99.111/agendamento/. The page title is "Agendamento". The form contains the following fields:

- Escolha a Cidade: Porto
- Escolha o seu Médico: João Marques
- Escolha o Horário: 08:45 as 09:30
- Horário de Início: 08:45
- Horário de Término: 09:30
- Dia: 15/03/2020
- Nome Completo: Pedro Antonio ESTG
- Email: teste@teste.rmel.stg

A blue "SALVAR" button is at the bottom of the form. The footer text reads: "Trabalho Microservices - MEI ESTG 2019/2020 - Alonso Machado - ©Copyright 2020".

Figura Agendamento

This screenshot shows the "Dia" field with the value "01/01/2020". Below the field, a red error message box states: "O valor deve ser 18/02/2020 ou posterior." The "Nome Completo" field below it contains the text "Pedro Antonio ESTG".

Figura Validação Data do Agendamento não pode ser inferior a HOJE

The screenshot shows the "Dia" field with a date picker open. The date picker is set to "Fevereiro de 2020" and displays a calendar grid. The date "18" is highlighted in blue. The "Nome Completo" field contains "Pedro Antonio ESTG". A blue "SALVAR" button is visible at the bottom.

Figura Agendamento Data - Datepicker

2.3.4.2 CheckIn

- /checkin/todos - Mostra em formato de Lista no Front-end todos os checkin gravados pelo microserviço Checkin.

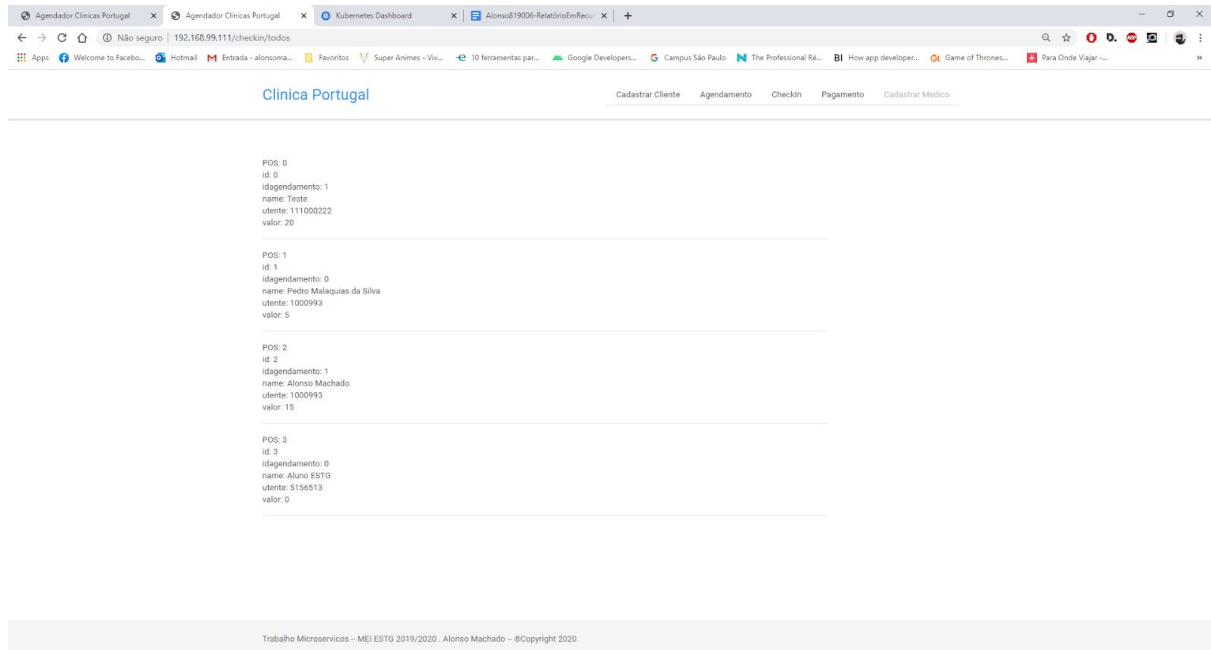


Figura Lista de Todos os CheckIn realizados

- /checkin/ - Página para adicionar um Agendamento. (É necessário estar logado)

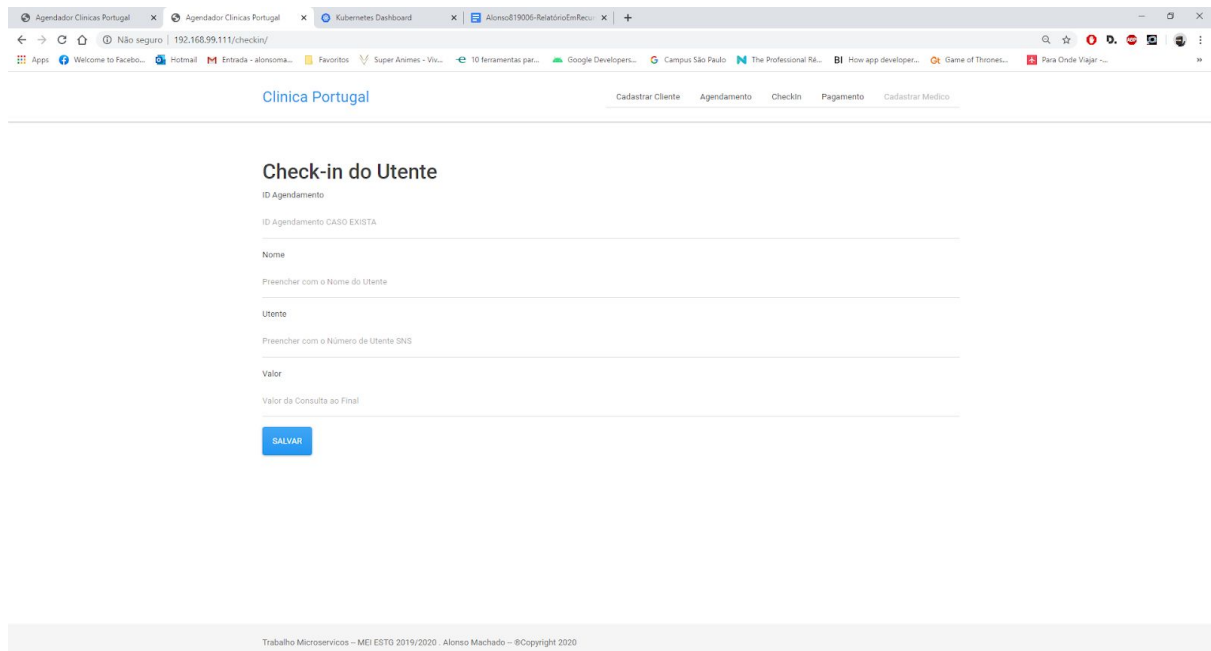


Figura CheckIn

2.3.4.3 Pagamentos

- /pagamentos/todos - Mostra em formato de Lista no Front-end todos os pagamentos gravados pelo microserviço Pagamento.

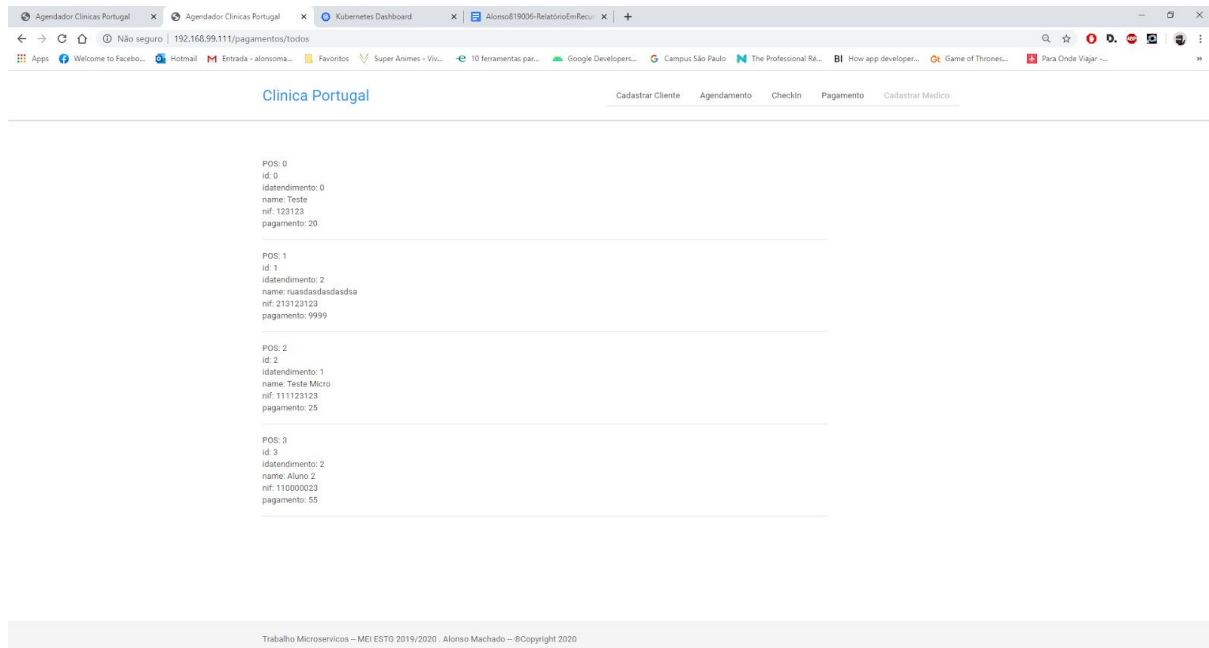


Figura Lista de Todos os Pagamentos realizados

- /pagamento/ - Página para adicionar um Agendamento. (É necessário estar logado)

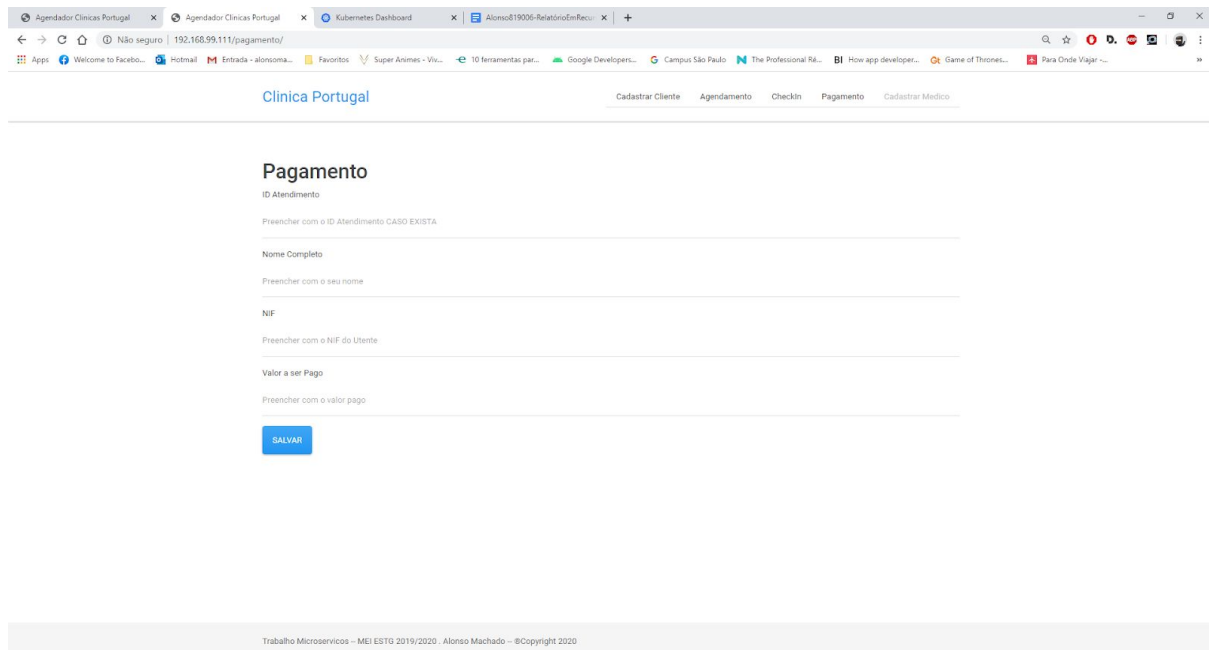


Figura CheckIn

2.3.5 API Gateway - Ingress

Neste trabalho usei o Ingress padrão do minikube, que é o Ngix, se pode ativar usando o comando `minikube addons enable ingress`, e o arquivo de configuração das rotas está no anexo B.

2.3.6 Mailer - Não finalizado

Microserviço que lida com envio de email de confirmação de agendamentos de consultas para o email cadastrado no agendamento.

3. CONCLUSÃO

3.1 Considerações sobre o Trabalho Prático

Realmente este trabalho foi bem mais complicado do que eu esperava, pois já tinha feito o trabalho prático da disciplina de Computação em Nuvem em JAVA RMI e usado Docker, mas neste sofri algumas complicações quanto a linguagem de programação pois foi a primeira vez que usei node.js e também seus frameworks como express e front-end framework express-handlebars, no início do projeto escolhi spring boot mas senti falta de material de pesquisa e o framework com as anotações estava ficando pesado e não estava entendendo bem o que se passava então resolvi trocar para node.js uma escolha acertada.

3.2 Passo a Passo

Clone o GITHUB <https://github.com/alonsomachado/microservicosclinicas>

3.2.1. Starte seu minikube com no minimo 4GB de RAM.

3.2.2. Execute o criar.sh

Este script vai criar as imagens pelos dockerfiles e depois subirá nos containers no minikube.

3.2.3. minikube ip

3.2.4. Entre no seu navegador no ip do minikube ip

3.2.5. Observações Importantes

Algumas funcionalidades não foram implementadas no front-end ainda pois não tive muito tempo para codifica-lo pois seria responsabilidade de outra pessoa do grupo, mas funcionam pelo POSTMAN.

3.2.6. Para deletar execute o deletar.sh

3.2.7. Para completar faltou o microserviço de email para contactar o cliente antes

REFERÊNCIAS

<https://expressjs.com/pt-br/>
<https://expressjs.com/pt-br/guide/database-integration.html>
<https://github.com/alonsomachado/microservicosclinicas>
<https://cloudnweb.dev/2019/08/implementing-redis-pub-sub-in-node-js-application/>
<https://hackernoon.com/scale-your-microservices-with-an-easy-message-queue-on-redis-e92n2gk3>
https://www.youtube.com/watch?time_continue=1364&v=9S-mphgE5fA&feature=emb_logo
(FRONT com Express Handlebars)
<https://handyman.dulare.com/passing-variables-through-express-middleware/>
https://www.nginx.com/products/nginx/kubernetes-ingress-controller/?utm_campaign=kubernetes&utm_medium=products&utm_source=youtube&utm_content=kic
https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/routes
<https://expressjs.com/en/guide/routing.html>
<https://kubernetes.io/docs/concepts/services-networking/ingress/>
<https://kubernetes.io/docs/tasks/access-application-cluster/ingress-minikube/>
<https://github.com/kubernetes/ingress-nginx/issues/1120>

Express Handlebars Tutorial, disponível em:

<https://www.youtube.com/watch?v=JbrqxPcuYVc&list=PLurIMwd6GdCi3ssXNAcjZ2l5mYaTfYPhf&index=6>, Acessado em 10/02/2020

ANEXOS

Anexo A. Avaliação Época Normal

O professor apontou como melhorias para o sistema:

Principalmente acabar com a limitação de distritos, fazer um sistema de microserviços único e totalmente escalável e não limita-lo por distrito.

Trocar o nome de realiza para Check-in pois para ele faz mais sentido;

Refazer desenho da arquitetura contendo as listas do pubsub separadas para melhor visualização e tirar o services deixar subentendidos.

Finalizar a ligação front-end com a api

Gostaria também de acrescentar:

Microserviço MAILER:

Criar o microserviço de email ou SMS que avisaria o cliente 24 ou 48h antes da consulta para lembrá-lo e não perder a consulta, ele estava planejado mas era dispensável;

Anexo B. Minikube DashBoard

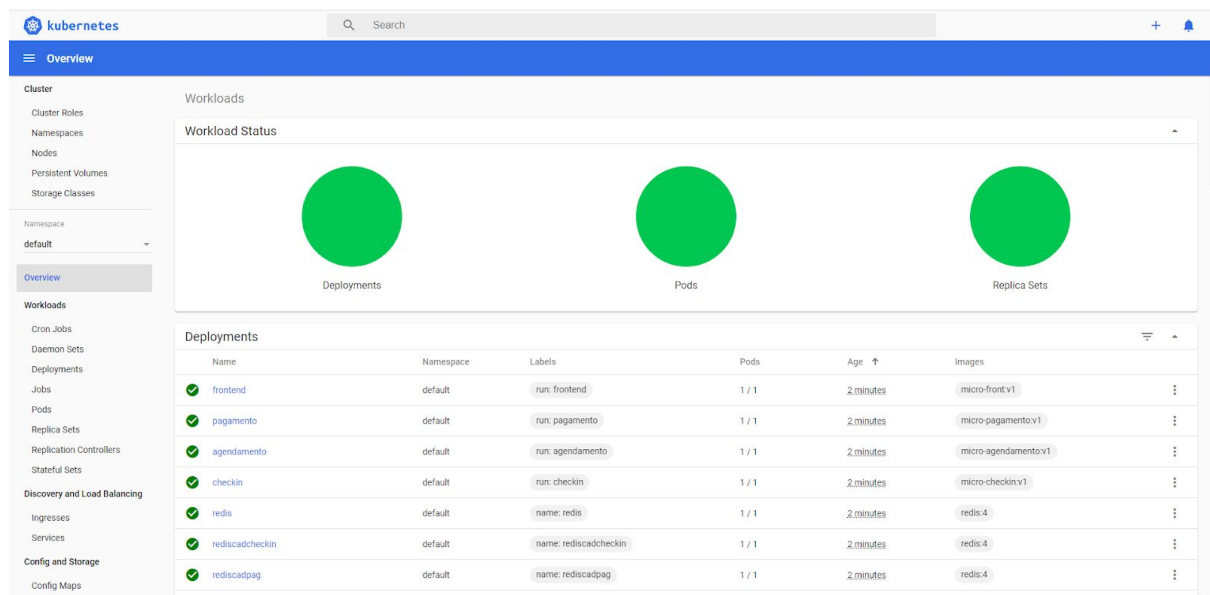


Figura 1: Overview com Deployments

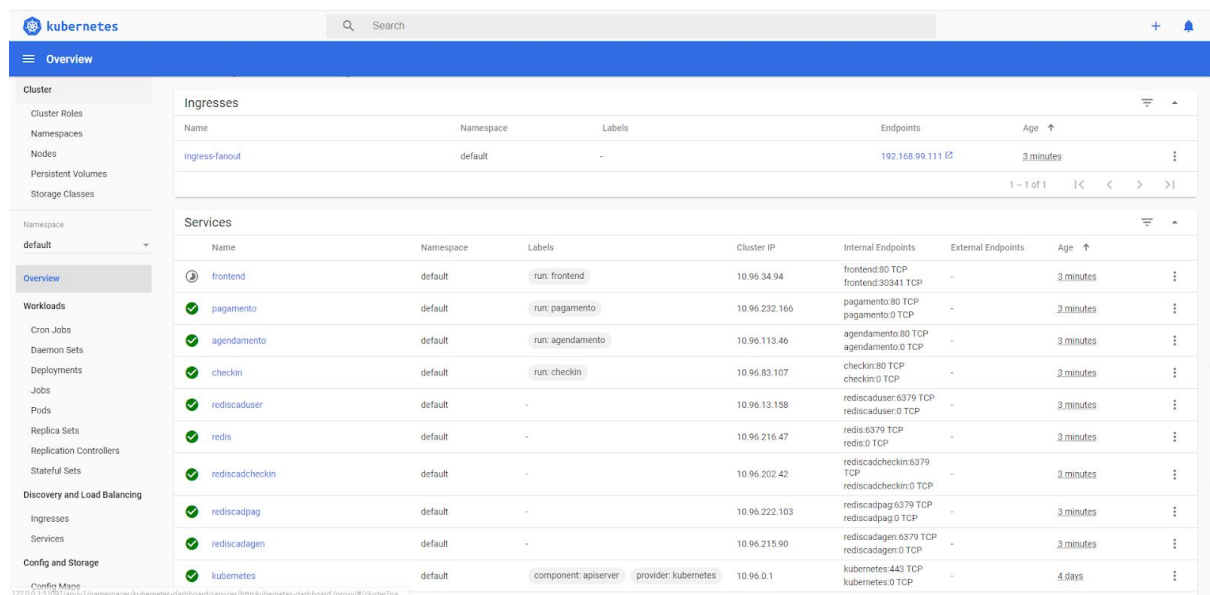


Figura 2: Ingress e Services