P.PORTO

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

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.

2.1.1.1 Express

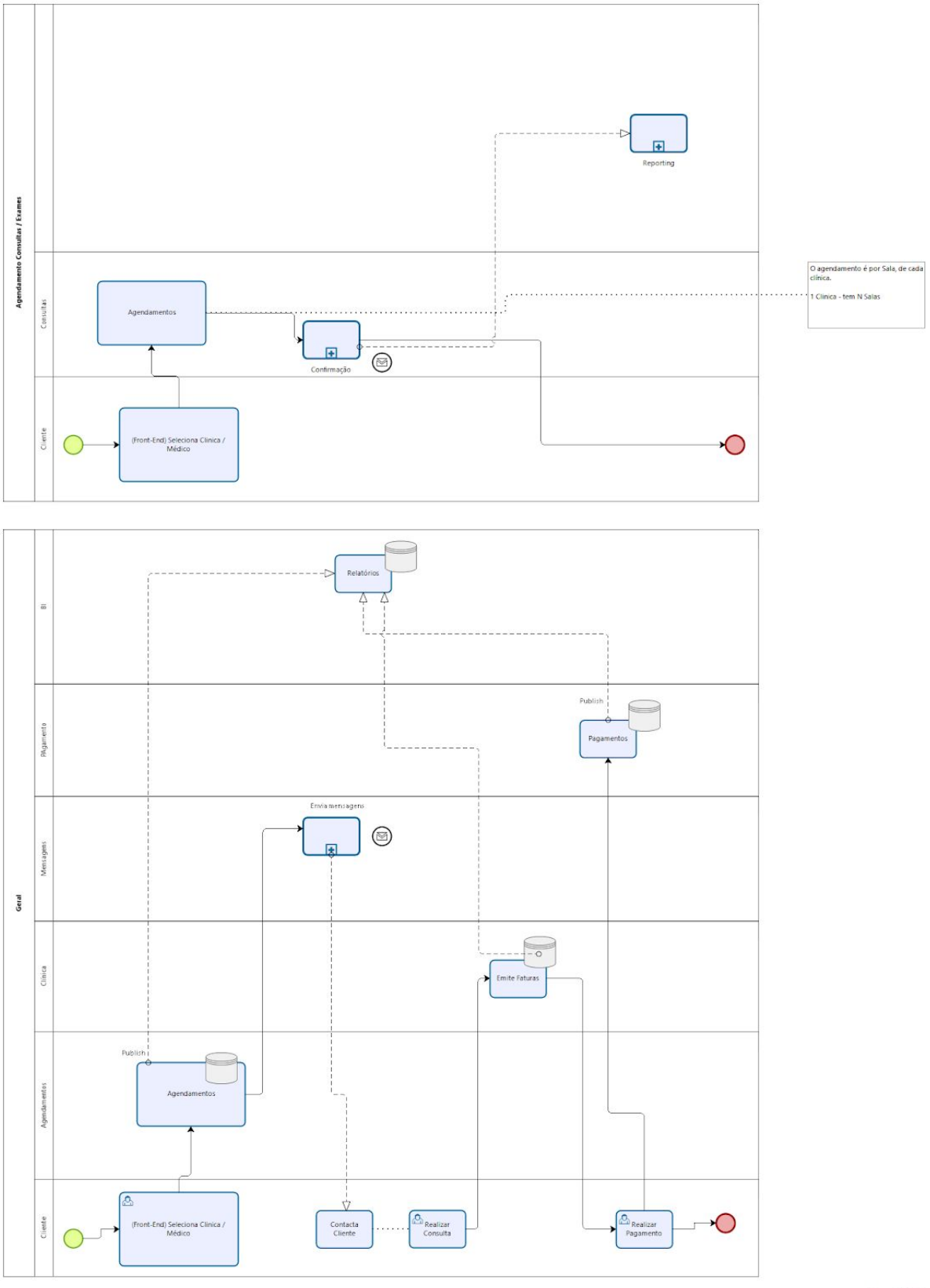
Framework para facilitar requisições HTTP: POST, GET, DELETE, UPDATE de forma mais fácil e rápida, deixa o código mais legível.
É o framework mais utilizado para desenvolvimento back-end para APIs em node.js.
Muito material para estudo.
https://expressjs.com/pt-br/guide/database-integration.html

2.1.2. Redis

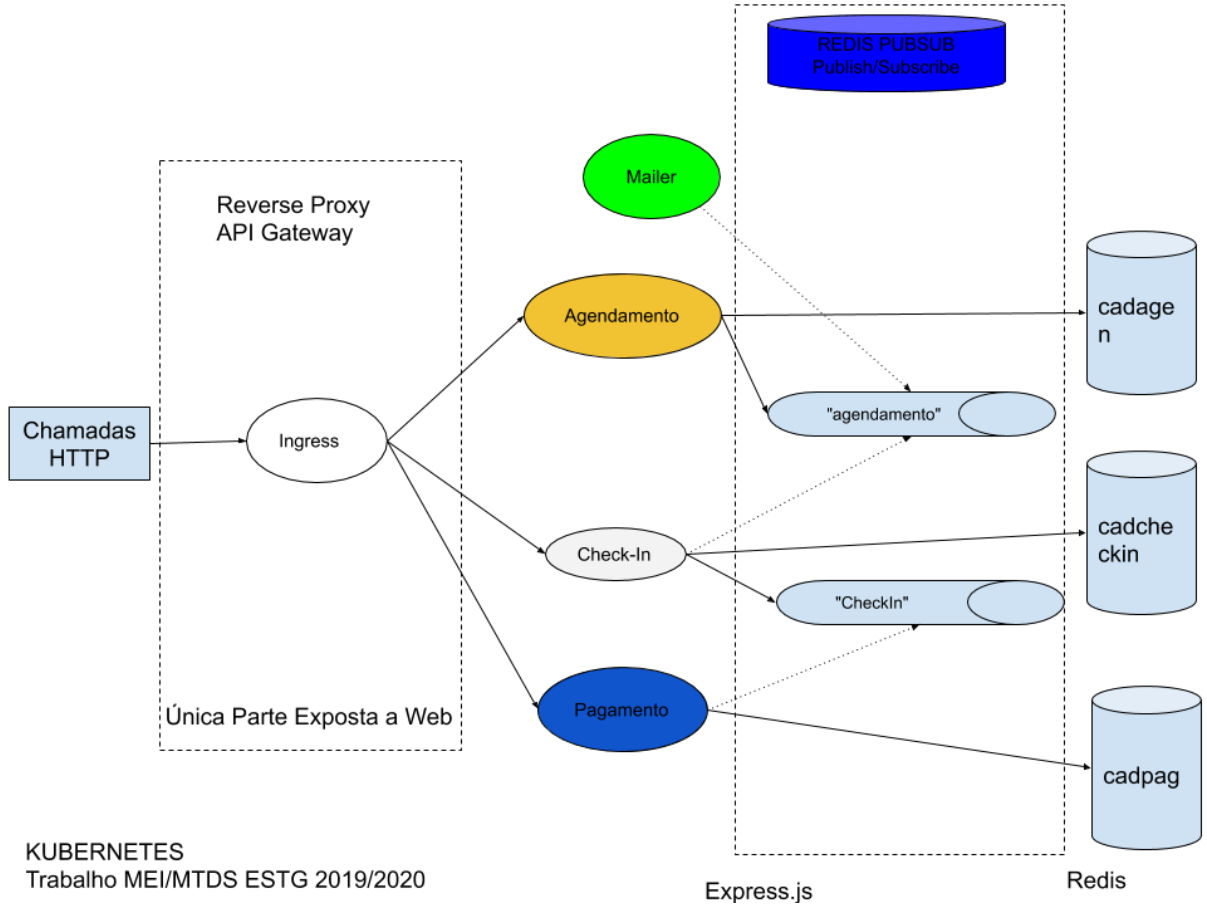
Banco de dados em memória, ele tem várias funções interessantes, neste trabalho usarei como banco de dados comum e principalmente a sua função PUBLISHER/SUBSCRIBER que cria canais de comunicação semelhante o Kafka ou rabbitmq.

2.2. Arquitetura

2.2.1 Modelo Inicial



2.2.1 Arquitetura Final



2.3. Microserviços

Resumidamente os microserviços foram separados por distritos 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, não está totalmente finalizado não tive tempo de terminá-lo pois estava sob responsabilidade do outro elemento do grupo que basicamente desistiu do trabalho.

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.

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 oque 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 (Anexo A)

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 (Anexo C)

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

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. **(FEITO)**
Finalizar a ligação front-end com a api

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