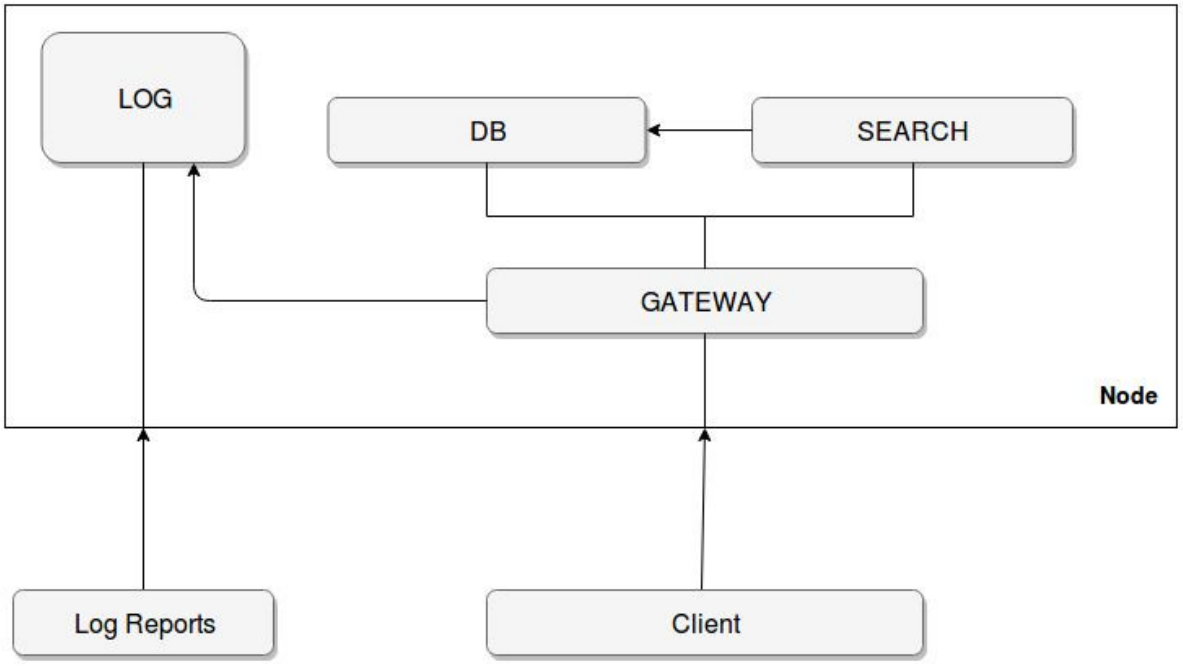
Catalog Series E Dramas

Responsáveis:
Arthur Pires <ahlp@cin.ufpe.br>
Giovanni Barros <gaabs@cin.ufpe.br>
Maria Cecilia Hunka <mchma@cin.ufpe.br>
Renato Deyvson da Silva <rdms@cin.ufpe.br>

Arquitetura



O sistema consiste de Basicamente 4 micro serviços, Serviço de Log, Serviço de DB, Serviço de Search e o Serviço de Gateway. Cada um desses serviços possuem vários pods e consequentemente vários containers.

- DB: Serviço responsável pelo CRUD dos modelos, espera sempre receber um ID no cabeçalho do request, ID o qual é setado pelo Gateway. Usa um banco relacional para armazenamento dos modelos. Todos seus endpoints são restritos a usuários logados.
- Gateway: Serviço responsável por ser o entrypoint e fazer a autenticação de toda aplicação. Tem um banco para gerenciar os registros e gera um Token no qual posteriormente consegue recuperar o ID do usuário. Esse ID recuperado é repassado pro serviço que foi solicitado.
- Search: Serviço responsável por guardar informações do DB que são públicas em memória para consulta mais rápida. Todos os seus endpoints são públicos.
- Log: Serviço composto pelo Kibana, Elasticsearch e Logstash para gerenciamento de logs da aplicação.

Notas de Release

Funcionalidades

- É possível se registrar e gerar um Token que garante acesso a todos os endpoints não públicos do sistema
- É possível consultar, criar, editar e deletar Series, Episodios e Profiles.

Problemas conhecidos

- No serviço de search os valores não são atualizados, request para o serviço de Database precisa ser refeito
- Logs não são apresentados de forma agradável
- No serviço de Gateway é necessário mexer no código para adicionar novas rotas
- Há algum problema com o volume utilizado no Gateway, isso ocasiona a geração de um token para cada sessão criada
- No Serviço de Search, o Haskell não deveria ter um gateway
- Não é possível atualizar seu login e senha
- Senha é guardada em texto aberto

Tecnologias

O deploy tem como base as tecnologias de Docker e Kubernetes, sendo necessários as versões 1.10 do Kubernetes e o Docker 17.03, que é a versão do Docker suportada pelo Kubernetes.

Quanto aos serviços, suas tecnologias e dependências são:

- DB: Ruby, Rails, Gem e o banco relacional Postgre
- Gateway: Golang e o banco de dados não relacional Mongodb
- Search: Haskell, Cabal, GHC, Node.js, npm
- Log: Logstash, Elasticsearch e Kibana

Quanto as libs de uso em cada tecnologia, no README de cada repositório há links dos mesmo.

Deploy

O deploy pelo kubernetes consiste de apenas 3 passos:

1. Tenha o Kubernetes e o Docker configurados
2. Clone https://github.com/ahlp/catalog-service
3. Entre em deploy-k8s/ e execute **$make deploy**

Para deletar o deploy só dar **$make deleteDeploy**

Esse deploy referenciam as imagens presentes no repositório do DockerHub que possuem começam o nome com “csd-”. Caso queira modificar e gerar novas imagens, dentro dos repositório de Gateway e Search há um makefile para fazer esse build e no Database há instruções no README. O Log se utiliza de imagens dos próprios serviços, então não há dockerfile para esse serviço no nosso repositório.

Caso seja queira utilizar outras imagens para o deploy, se faz necessário alterar a referência das imagens nos seguintes arquivos:

- Para alterar a imagem do DB:
  - deploy-k8s/deploy-crud/deployment-crud.yaml
- Para alterar a imagem do Gateway:
  - deploy-k8s/deploy-gateway/deployment-gateway.yaml
- Para alterar a imagem do Search:
  - deploy-k8s/deploy-search/deployment-search-gateway.yaml
  - deploy-k8s/deploy-search/deployment-search.yaml

Referências

- Projeto: https://github.com/ahlp/catalog-service
- https://kubernetes.io/
- https://www.docker.com/
- https://www.ruby-lang.org/
- https://www.haskell.org/ghc/
- https://golang.org/
- https://nodejs.org/en/
- https://www.mongodb.com/
- https://www.postgresql.org/
- https://www.elastic.co/