**Universidade de Brasília - UnB**

**Relatório PC3 de Software**

**Orientadores:**

**Alex Reis, Guilherme Bestar, Rhander Viana,**

**Ricardo Chaim, Sébastien Rondineau**

**Brasília, DF**

**Abril de 2018**

# Integrantes de Software

Adailson Santos; Lucas Amoêdo; Leonardo Sagmeister; Marcelo Martins.

# Desenvolvimento arquitetural de software

Ao dar sequência à proposta da equipe em utilizar a arquitetura de microsserviços, utilizou-se o Docker para criar containers em cada serviço específico com dependências necessárias para a utilização do mesmo. Em cada serviço existe um Dockerfile que contém a descrição das tecnologias, volumes e portas previamente definidas. Existe ainda um arquivo chamado Docker-compose que irá gerenciar todos os containers e irá facilitar o controle e monitoramento de cada serviço.

## Serviços desenvolvidos com o Docker

* + 1. greenhouse

Esse é o principal serviço, ele trabalhar no gerenciamento dos containers, ele é responsável por criar uma rede de conexão entre todos os serviços, e por isso na sua raiz da aplicação encontra-se o arquivo docker-compose.yml em que nele há um algoritmo que instancia todos os serviços do projeto.

* + 1. webService

Nesse serviço se encontra a API do projeto, portanto é um serviço único e diferenciado que irá tratar de responsabilizar de receber os dados por meio de simuladores ou dados reais por meio do servidor rabbitmq instalado na Rasberry pi. Ao receber esses dados, eles serão tratados e realização o parser para o formato JSON para que seja possível enviar respostas em uma linguagem que possa ser escutada pelo webApp, o frontend da aplicação. Utilizou-se o framework Python e Flask, juntamente com algumas dependência cruciais para o desenvolvimento da aplicação.

* + 1. webApp

Esse serviço foi desenvolvido para o frontend da aplicação, dessa forma, foi instalado o node.js e npm que são dependências do ReactJS, o framework Javascript. Portanto, este escuta as requisições do servidor da API para o recebimento do JSON e o tratamento dos dados para componentes que sejam visíveis na interface da aplicação.

* + 1. postgres

Para o Banco de Dados do projeto, utilizou-se o Postgres, sendo assim, foi necessário um serviço que armazene todos os dados do projeto. Com o auxílio da dependência python SQLAlchemy utilizada em cada módulos de microsserviço, foi possível facilitar a criação da base de dados e a vinculação das models para a criação das tabelas e facilidade em realizar consultas de qualquer microsserviço por meio de simples chamadas de métodos disponíveis no SQLAlchemy.

* + 1. simulators

Para os testes da aplicação no início do projeto foi possível mocar os dados utilizando o serviço simulators, este gera dados randomizados e gera instâncias para os microsserviços responsáveis armazenar os dados no banco de dados, garantindo que toda a aplicação aconteça normalmente sem a necessidade previamente da RaspBerry pi.

* + 1. rabbit

Uma ótima maneira de trabalhar com microsserviços é utilizando o rabbitmq que utiliza o protocolo AMQP para o incremento de filas dos dados recebidos do microsserviços. Sendo assim, este serviço é responsável por gerenciar todas as filas e se conectar com todos os microsserviços, por isso é essencial que ele esteja ativo juntamente com o serviço de banco de dados antes que qualquer outro microsserviço se inicie.

* + 1. temperatureServer

Esse foi o primeiro microsserviço de integração desenvolvido, ele é responsável pelo recebimento da temperatura ambiente e pela conexão com o rabbit, esse relacionamento com o rabbit é para o incremento dos dados em uma fila.

* + 1. humidityServer

Esse microsserviço, da mesma forma que o anterior é responsável pelo recebimento da umidade relativa do ar interna à estufa. Ao receber os dados é realizado a inserção dos dados no banco de dados paralela à conexão do rabbit.

* + 1. iluminationServer

Para o monitoramento da luminosidade, foi desenvolvido o microsserviço de iluminação, este deve receber os dados recebidos pelos sensores e tratar em formato booleano (verdadeiro ou falso) para ilustrar a ausência ou não de luz dentro da estufa.

* + 1. phServer

O monitoramento do Ph também se torna necessário na aplicação, seguindo a lógica dos outros microsserviços de integração, teremos um serviço chamado phServer que irá receber da RaspBerry pi os dados em tempo reali e realizar o tratamento para o banco de dados.

* + 1. waterTemperatureServer

Esse serviço é bem similar ao serviço de temperatura ambiente, pois o recebimento e tratamento dos dados são realizados praticamente da mesma forma, sendo assim o serviço de temperatura da água irá conectar com o rabbit, realizar a inserção dos dados no banco.

* + 1. waterLevelServer

Esse serviço é responsável pelo troca de água automática da estufa. A integração necessita desse serviço para a coleta dos níveis de água e realizar o monitoramento para que, quando estiver em um nível de pH fora da faixa ideal, uma mensagem de notificação é enviada ao usuário e este tem a escolha de despejar a solução líquida contida no repositório líquido ou deixar conforme estar.

* + 1. drawerStatusServer

Para a integração de abertura/fechamento da gaveta é necessário esse serviço, ele recebe e envia sinais por meio do rabbit para a Raspberry pi. Para o recebimento de sinais, há um monitoramento do estado da gaveta e para o envio de sinais o algoritmo confere se a porta está fechada, caso esteja, a gaveta não abre, se a porta estiver aberta, a gaveta abre e fecha normalmente.

* + 1. nginx

Para trabalhar na configuração e carregamento de imagens foi necessário a utilização do Nginx e este cuida da configuração de pastas e portas das imagens recebidas pela integração da Raspberry pi e uma câmera que tira fotos em sequência e envia para o rabbitmq online na Amazon.

## Deployment

## Telas da aplicação

* + - 1. Login

O usuário deve estar autenticado para conseguir acessar nossa aplicação, sendo assim, após realizado o cadastro do usuário, previamente configurado no sistema, é possível fazer o login da aplicação.

* + - * 1. Interface Mobile
        2. Interface Web
      1. Dashboard

Para a página inicial, iniciamos com um Dashboard que realiza monitoramento em tempo real de todos os sensores da estufa.

* + - * 1. Interface Mobile
        2. Interface Web
      1. Relatórios

Para cada microsserviço de integração são gerados relatório para serem observados nos períodos de dia, mês e ano com suas respectivas médias.

* + - * 1. Interface Mobile
        2. Interface Web
      1. Gráficos

Os gráficos são gerados para fins de tornar ao usuário uma melhor visualização dos dados pertinentes no relatório, sendo assim, os gráficos ilustram a alteração dos valores de acordo com o período escolhido no sistema.

* + - * 1. Interface Mobile
        2. Interface Web
      1. Notificações

Sempre que os dados recebidos estiverem fora da faixa ideal definida para cada sensor, é enviado um alerta ao usuário para que ele possa corrigir.

* + - * 1. Interface Mobile
        2. Interface Web

# Utilização do Nameko na Raspberry pi

## 