FIAP

Challenge Plusoft - Finder
Sistema de geolocalização e alerta em edificações

Aline Triñanes Machado - RM 84449 - alinetrim@hotmail.com

Alysson Gustavo Rodrigues Maciel - RM 86484 - alymaciel8@gmail.com

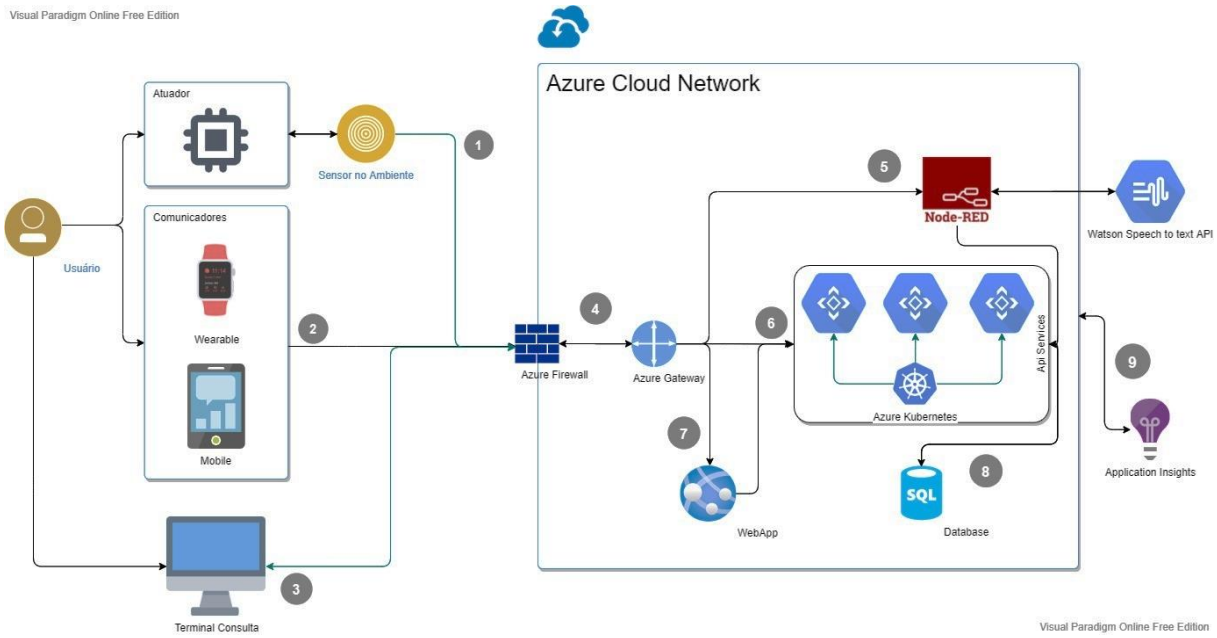
Gabriel Franham - RM 80483 - gabrielfranham@gmail.com

Gabriel Garcia Pereira - RM 86288 - gabriel.garp@outlook.com

Helouíse Cristina de Almeida Itokazo - RM 85110 - helouise.almeida93@gmail.com

Jonas Muniz de Souza - RM 84575 - jonasmzsouza@gmail.com

1. Definição da arquitetura de solução



1.1 - Definição

Optamos por utilizar uma arquitetura REST e MICROSERVICES. Utilizaremos a arquitetura REST porque esta arquitetura nos permite padronizar a interface do back-end que será consumida pelos dispositivos e suas aplicações, facilitando a comunicação, integração e transferência de dados entre eles. Utilizando a arquitetura de microsserviços, separamos as responsabilidades de cada API REST, facilitando sua manutenção, direcionando melhor as requisições de cada interface ou dispositivo e possibilitando uma escalabilidade mais eficiente.

1.2 – Explicação

1 – O sensor de ambiente é responsável por captar o sinal de um RFID que o funcionário carrega consigo. Esse RFID carrega a informação do funcionário como usuário no sistema, sendo o papel do sensor de apenas transmitir esses dados para a Azure Function responsável por atualizar a sua localização atual na base de dados.

2 – Os dispositivos wearable e mobile são responsáveis por realizar e receber solicitações de chamados, através do comando de voz do wearable ou interface mobile, fazendo requisições para os microsserviços responsáveis por esse processamento.

3 – A aplicação desktop tem como papel servir como um terminal de consultas. Suas requisições serão para os microsserviços que buscam históricos de chamados e suas informações.

4 – Através dos serviços Azure Firewall e Gateway de aplicativo de Azure, as requisições serão processadas e enviadas até a sua API de destino.

5 – O node-red será responsável por receber solicitações por voz do wearable e se comunicar com a API Watson Speech to Text, realizando o processamento do pedido por voz e então enviando ao microsserviço que inicia a solicitação de um chamado.

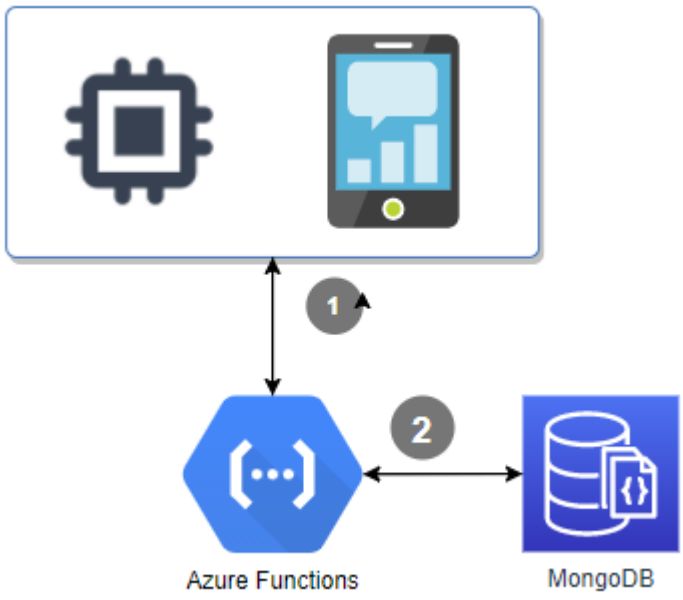
6 – Os serviços(apis) essenciais serão empacotados em contêineres e orquestrados pelo cluster Azure Kubernetes Services.

7 – A aplicação desktop ficará hospedada no serviço Azure WebApp, que fará requisições aos microsserviços de consulta.

8 – Utilizaremos o Azure SQL, serviço que será utilizado pelos microsserviços para a permanência e consulta dos dados da solução.

9 – Através do serviço Application Insights da Azure, será possível monitorar o desempenho de todo o fluxo da aplicação, facilitando a manutenção caso haja necessidade da otimização dos processos.

2 – Definição da arquitetura do mecanismo de localização



2.1 – Definição

Optamos por utilizar o recurso Azure Functions disponibilizado pela Azure para realizar os procedimentos de busca e procura por oferecer escala flexível e execuções rápidas baseadas em eventos. Como tanto a atualização da localização de uma pessoa e também a busca devem ser realizadas de forma rápida, optamos por utilizar o MongoDB que ficará responsável por guardar e buscar somente essa informação.

2.2 – Explicação

1 – O sensor de RFID, wearable e o dispositivo mobile serão responsáveis por disparar os eventos que acionarão as Azure Functions de atualização de localização e busca por usuário, respectivamente.

2 – As solicitações e dados recebidos de localização serão armazenados e consultados em um banco de dados NoSQL MongoDB, que constantemente será atualizado com novas localizações e receberá requisições de consulta.

Principais apis

Api de atualização do local do usuário (Azure Function)

Trigger: /atualizarLocalService

Método	Endpoint	Verbo	Ação
atualizarLocalUsuario	/AtualizarLocal/{idUsuario}	PUT	Atualiza na base o ambiente em que o usuário está no momento

Api de localização de funcionários solicitados em um chamado (Azure Function)

Trigger: /chamadoService

Método	Endpoint	Verbo	Ação
chamadoPorNome	/chamadoPorNome	POST	Inclui um chamado que busca por um usuário informado como parâmetro no corpo da requisição
chamadoPorFuncao	/chamadoPorFuncao	POST	Inclui um chamado que busca por usuários de determinada função, que estejam próximos do usuário que fez a solicitação.

Api de gerenciamento de cadastro de ambientes

Path: /ambiente

Método	Endpoint	Verbo	Ação
incluir	/ambiente	POST	Inclui um novo ambiente na base
alterar	/ambiente/{id}	PUT	Altera um ambiente existente
consultar	/ambiente/{id}	GET	Consulta um ambiente existente
excluir	/ambiente/{id}	DELETE	Exclui um ambiente cadastrado
buscarTodos	/ambiente	GET	Retorna todos os ambientes da base

Api de gerenciamento de cadastro de setores

Path: /setor

Método	Endpoint	Verbo	Ação
incluir	/setor	POST	Inclui um novo setor na base
alterar	/setor /{id}	PUT	Altera um setor existente
consultar	/setor /{id}	GET	Consulta um setor existente
excluir	/setor /{id}	DELETE	Exclui um setor cadastrado
buscarTodos	/setor	GET	Retorna todos os setores da base

Path: /cargo

Método	Endpoint	Verbo	Ação
incluir	/cargo	POST	Inclui um novo cargo na base
alterar	/cargo /{id}	PUT	Altera um cargo existente
consultar	/cargo /{id}	GET	Consulta um cargo existente
excluir	/cargo /{id}	DELETE	Exclui um cargo cadastrado
buscarTodos	/cargo	GET	Retorna todos os cargos da base

Api de gerenciamento de usuários
Path: /usuario

Método	Endpoint	Verbo	Ação
incluir	/usuario	POST	Inclui um novo usuário na base
alterar	/usuario /{id}	PUT	Altera um usuário existente
consultar	/usuario /{id}	GET	Consulta um usuário existente
desativar	/usuario /{id}	DELETE	Exclui um usuário cadastrado
buscarPorNome	/usuário/nome/{nome}	GET	Consulta um usuário pelo nome

Repositório do github: https://github.com/GarciaGP/PluFinderApi

Link Azure: https://plufinderapi.azurewebsites.net/swagger-ui/

Link Heroku: https://plufinderapi.herokuapp.com/swagger-ui/