egSYS

Soluções Integradas Inteligentes

Golang Challenge

Versionamento

Versão	Motivo da Alteração	Data	Autor
1	Criação do documento	01/01/2022	Allan Ederich
2	Revisão do documento	20/01/2022	Allan Ederich

Vehicle Tracking System

Sumário

1. Sobre o Teste

O teste tem por objetivo avaliar o seu nível de conhecimento e sua senioridade nas principais tecnologias e padrões que adotamos na egSYS.

Neste teste, você irá desenvolver uma API básica para rastreamento de veículos onde iremos avaliar os seus conhecimentos em engenharia, arquitetura, infraestrutura e qualidade.

Reiteramos que não há a necessidade de ter conhecimento em todas as áreas citadas acima, porém o conhecimento e a conclusão das tarefas propostas nas áreas acima nos permite classificar você como júnior, pleno, sênior ou especialista, **consequentemente melhorando a sua faixa de remuneração**.

2. O que será avaliado?

O nosso modelo de avaliação é quantitativo, ou seja, você receberá uma nota e a partir dessa nota conseguiremos classificá-lo de acordo com o seu nível de senioridade técnica.

Temos um conjunto de testes automatizados para validar os contratos e as regras de negócios que propomos, então cuide das validações e tratamentos de erro da sua API.

2.1. Observações técnicas importantes

- Você terá que desenvolver a API usando a linguagem **Golang**.
- A solução deverá ser publicada no **Github** contendo um **ReadMe** com as instruções para a execução do projeto.

Critérios de avaliação:

- Manutenibilidade;
- Simplicidade;
- Testabilidade;
- Documentação.

Bônus:

- Gostamos de usar Docker nas nossas soluções;
- Fácil execução do projeto é um ponto favorável;
- Uma boa documentação do projeto e de suas APIs ajuda muita;
- Garantir a organização e uma boa cobertura de testes no seu projeto será visto de forma muito positiva;

2.2. Sistema de pontuação

Para cada item, você receberá uma avaliação que possui um fator de multiplicação de acordo com a sua relevância.

Avaliação	Fator Multiplicador	Observação
Não realizou	0	O participante não realizou o item.
Em desenvolvimento	0,3	O participante realizou o item porém não atendeu todos os requisitos solicitados e/ou possui muitos erros.
Desenvolvido	0,7	O participante realizou o item atendendo todos os requisitos solicitados sem erros.
Desenvolvido Plenamente	0,85	O participante realizou o item atendendo todos os requisitos solicitados sem erros, de forma organizada e legível.
Referência	1	O participante realizou o item atendendo todos os requisitos solicitados sem erros, de forma organizada e legível. Também foi constatado que o participante se preocupou com a qualidade do item, usando as melhores práticas do mercado (design patterns), tendo como base contexto, manutenibilidade, simplicidade, testabilidade e documentação.

2.3. Itens avaliados

Você será avaliado de acordo com cada item abaixo. Cada item possui um peso de acordo com a importância para a posição em que estamos procurando.

Área	Item	Peso	Observação
Engenharia	Golang	25	Será avaliado o uso correto dos recursos da linguagem, assim como bibliotecas e frameworks.
	Organização do projeto	10	Será avaliado o modo de como você organizou os seus pacotes por domínio, se eles se comunicam de forma clara e

			coesa. Será avaliado também a facilidade de entendimento de como o projeto está organizado. Dica: Use o ReadME para explicar a estrutura de pastas.
	Design Patterns	10	Será avaliado padrões de projeto de acordo com o contexto (não usar apenas por usar) e a implementação de forma correta. Dica: Alguns padrões podem ser úteis como DDD, TDD, Factory para criação de objetos, WorkerPool para gerenciamento de goroutines etc.
	Documentação do Projeto (ReadME)	5	Tente explicar o objetivo da sua API, assim como ela está organizada e como fazemos para testá-la.
Arquitetura	Desenho da solução	10	Realize o desenho da sua API. Podem ser utilizados diagramas de comunicação entre as API, bancos etc. Será avaliado um ou mais desenhos que consigam mostrar o comportamento da aplicação.
	Documentação das APIs	10	Queremos ver como você documenta as suas APIs. Pode usar ReadME, Swagger ou outra ferramenta. Usar libs que automatizem esse processo também é um plus.
Qualidade	Testes Unitários	10	Avaliaremos como você testa a sua aplicação e se o seu código é facilmente testável.
	Cobertura do código	10	Avaliaremos a cobertura e a mutabilidade que os seus testes

			procuram na forma de validar o seu código.
Infraestrutura	Docker / Docker Compose	10	Avaliaremos se você usou docker ou docker-compose para facilitar a criação de recursos com o banco de dados ou até mesmo a aplicação inteira para que possamos testar de forma automatizada.

2.4. Feedback

Ao finalizar o teste, você receberá uma matriz contendo a nota por item e a nota final com respectivas observações. Após essa etapa, você poderá contestar a sua nota e podemos reavaliar em conjunto.

Segue modelo:

Feedback Golang Developer

Olá xxx, a nota da sua avaliação técnica foi de 68,75 e você foi enquadrado técnicamente como **Pleno**.

Verifique cada observação técnica e entre em contato conosco caso discorde de alguma avaliação. Estamos dispostos a entender o seu lado e reavaliar se chegarmos em um consenso.

Área	Item	Avaliação	Peso	Fator	Nota Final	Observação do Avaliador
Engenharia	Golang	Desenvolvido Plenamente	25	0,85	21,25	x
	Organização do projeto	Desenvolvido	10	0,7	7	x
	Design Patterns	Desenvolvido	10	0,7	7	x
	Documentação do Projeto (ReadME)	Referência	5	1	5	x
Arquitetura	Desenho da solução	Não realizou	10	0	0	x
	Documentação das APIs	Em desenvolvimento	10	0,3	3	x

Qualidade	Testes Unitários	Desenvolvido Plenamente	10	0,85	8,5	x
	Cobertura do código	Desenvolvido	10	0,7	7	x
Infraestrutura	Docker / Docker Compose	Referência	10	1	10	x

3. O Projeto Vehicle Tracking System

De forma geral, o objetivo do projeto é receber a localização do veículo com a sua velocidade atual. Caso a velocidade atual esteja acima do cadastrado, você terá que enviar uma notificação para os sistemas cadastrados (você deverá fazer um POST em uma URL cadastrada).

No tópico Golang, serão avaliados também estruturas de programação assíncrona/concorrente, orientação a eventos, estratégias de retry e mensageria.

Abaixo, teremos dois tópicos, sendo o primeiro uma sugestão de estrutura de dados e a última o contrato mínimo das APIs.

3.1. Estrutura de Dados

Segue abaixo uma estrutura de dados **sugerida** ***(fique a vontade para criar o seu próprio modelo)***

Fleet (frota)

Cada frota tem um nome e uma velocidade máxima obrigatória.

Exemplo de dados:

Fleet_ID	**Name**	**Max_Speed (velocidade em m/s)**
1	Veículos de perseguição	30,55
2	Veículos de transporte de prisioneiros	25
3	Escolta armada	22,22

Fleet_Alert (alerta da frota)

Quando um veículo ultrapassa a velocidade especificada, devemos enviar os dados para todas as URL cadastradas daquela frota via POST.

Fleet_Alert_ID	Fleet_ID	WebHook
1	1	http://localhost:8081/fleet/alert

Vehicle (veículo)

Cada veículo pertence a uma frota e pode ter sua velocidade máxima alterada. Caso não tenha uma velocidade máxima customizada, usará como referência a velocidade máxima da frota.

Vehicle_ID	Fleet_ID	Name	Max_Speed (velocidade em m/s)
1	1	Honda Civic - Policia RS	50
2	2	Camburão	
3	3	Polícia Rodoviária	50

Vehicle_Position (posição do veículo)

Histórico da posição do veículo contendo data, latitude, longitude, velocidade atual e velocidade máxima (do veículo, se não houver, usar a velocidade da frota).

Vehicle_Position_ID	Vehicle_ID	Timestamp	Latitude	Longitude	Current_Speed	Max_Speed
1	1	ISO 8601	0	0	0	50
2	1	ISO 8601	0	0	0	50
3	1	ISO 8601	0	0	0	50

3.2. Endpoints

Desenvolva os endpoints abaixo considerando as regras de negócio mencionadas anteriormente.

Nos endpoints abaixo, você não poderá alterar o contrato, apenas adicionar novas informações caso seja necessário.

É fundamental que não haja alteração nos campos já existentes pois rodamos testes automatizados.

3.2.1. Reset

- **DELETE** /database *(limpa toda a base de dados)*

HTTP Code: 200

3.2.2. Fleets

- **GET** /fleets *(lista todas as frotas)*

HTTP Code: 200
Response Body:
[{ "id": 1, "name": "Veículos de perseguição", "max_speed": 30.55 }]

- **POST** /fleets *(Cria uma frota)*

HTTP Code: 201
Request Body:
{ "name": "Veículos de perseguição", "max_speed": 30.55 }

Response Body:
{ "id": 1 }

Regras de negócio:
Deve validar se o nome e a velocidade são válidos.
Se algum campo não for válido, deve retornar HTTP Code 400.
A velocidade deve ser maior que 0.

3.2.3. Fleet Alerts

- **GET** /fleets/{id}/alerts *(lista todas os alertas de uma frota)*

HTTP Code: 200
Response Body:
[{ "id": 1, "fleet_id": 1, "webhook": "http://localhost:8081/fleet/alert" }]

- **POST** /fleets/{id}/alerts *(Cria uma alerta para frota)*

HTTP Code: 201
Request Body:
{ "webhook": "http://localhost:8081/fleet/alert" }

Response Body:

{ "id": 1 }

Regras de negócio:
Deve validar se a url é válida. Se não for válida, deve retornar HTTP Code 400.

3.2.4. Vehicles

- **GET** /vehicles *(lista todos os veículos)*

HTTP Code: 200
Response Body:
[{ "id": 1, "fleet_id": 1, "name": "veículo 1", "max_speed": 50 }]

Regras de negócio:
Deve retornar o max_speed do veículo. Caso o veículo não tenha max_speed definido, deve retornar o max_speed da frota.

- **POST** /vehicles *(Cria uma veículo)*

HTTP Code: 201
Request Body:
{ "fleet_id": 1, "name": "veículo 1", "max_speed": 50 }

Response Body:
{ "id": 1 }

Regras de negócio:
Deve validar se os campos são válidos.
Se algum campo não for válido, deve retornar HTTP Code 400.
O campo max_speed é opcional (pode ser nulo na request e base).

3.2.5. Vehicles Positions

- **GET** /vehicles/{id}/positions *(lista todos as posições de um veículo)*

HTTP Code: 200
Response Body:
[{ "id": 1, "vehicle_id": 1, "timestamp": "ISO-8601", "latitude": 0, "longitude": 0, "current_speed": 0, "max_speed": 0 }]

- **POST** /vehicles/{id}/positions *(Salva a posição do veículo)*

HTTP Code: 201
Request Body:
{ "timestamp": "ISO-8601", "latitude": 0, "longitude": 0, "current_speed": 0 }

Response Body:
{ "id": 1 }

Regras de negócio:
Deve validar se os campos são válidos.
Se algum campo não for válido, deve retornar HTTP Code 400.

Emissão de Evento:
Após salvar a posição do veículo, você deverá verificar se a velocidade do veículo é maior do que a cadastrada (devemos verificar a velocidade cadastrada no veículo e se não tiver, usar a da frota).
Caso a velocidade seja maior, deverá enviar as seguintes informações para todos os webhooks cadastrados da frota:

- **POST** {url cadastrada}

{ "id": 1, "vehicle_id": 1, "timestamp": "ISO-8601", "latitude": 0, "longitude": 0, "current_speed": 0, "max_speed": 0 }

Ao enviar essa informação, você deve esperar que o serviço responda com um HTTP Status 200.
Se o serviço não responder com um HTTP 200, você deve tentar reenviar 3 vezes, sendo 1, 5 e 15 segundos após cada tentativa. Se todas as alternativas falharem, você pode cancelar a notificação.

Dica 1: Após salvar a informação, é uma boa prática que o evento seja processado de forma assíncrona (a requisição não precisa esperar o envio e as tentativas de envio de notificação)

Dica 2: Você usar uma estratégia de worker e/ou algum sistema de mensageria será visto com ótimos olhos.