Teste técnico para vaga de desenvolvedor

Para ter uma maior facilidade em rodar o programa de teste solicito que seja feito um repositório no GitHub, e cada exercício separado em sua respectiva pasta, explicando dentro de um readme.md como deverá ser executado e com as descrições de como deverá ser utilizado.
Esse repositório deverá ser compartilhado de forma pública ou privada(Caso escolha essa opção deverá ser compartilhado no email wesleykowalski10@gmail.com).
Antes de começar a fazer os exercícios gostaria que fosse estipulado um prazo total para conclusão dos exercícios, que deverá ser compartilhado no contato abaixo.
Diante de dúvidas poderá solicitar apoio no WhatsApp: (47) 99199-0037.

Exercício 1 - Programa de leitura e gravação em arquivo .json

O exercício 1 é construído para que seja avaliado como o candidato trabalha com regras e processos definidos, junto com leitura e gravação de arquivos.
Elabore um programa que consiga através de 2 passos base, modificar ou ler as informações de um arquivo no formato json, o nome do arquivo deverá ser dado como "config.json".
Caso o arquivo não exista, deverá ser criado o mesmo na pasta que o programa está sendo executado.
Ao executar, o programa deverá perguntar para o usuário o ele deseja fazer, seguindo as opções abaixo:

1 - Read configuration - (Escrever no arquivo)
2 - Write configuration - (Ler arquivo)

Caso o usuário escolha a opção 1, deverá ser apresentado na tela o json que está salvo dentro do arquivo,
Se o arquivo estiver vazio, então deverá mostrar na tela uma mensagem informando que o arquivo não contém informações.
Caso o usuário escolha a opção 2, então deverá ser feita 3 perguntas, e guardada as suas respectivas respostas.

1 - Informe o nome do servidor:
2 - Informe o IP do servidor:
3 - Informe a senha do servidor:

As perguntas devem ser feitas nesta mesma sequência, e após isso deverá ser salvo as respostas no arquivo json,
Após isso deverá ser retornada uma mensagem que foi salvo com sucesso as informações no arquivo, e também apresentar o json salvo no mesmo.

O arquivo json deverá conter os seguintes itens: "server_name", "server_ip", "server_password"

"server_name" : Representando o nome do servidor;
"server_ip" : Representando o IP do servidor;
"server_password" : representando a senha do servidor.

Exercício 2 - Elabore um programa de cadastro e operação de estacionamentos simples usando API REST

Esse exercício tem o intuito de avaliar como é o seu desempenho referente a demandas de criação de novas aplicações, entendimento de requisitos técnicos e conceitos de aplicação Python/Django e também AngularJS ou VueJS.
Agora falando um pouco da aplicação, ela será responsável por fazer os cadastros e operação com veículos de um estacionamento.
Deverá ser utilizado Python, Django como linguagens de Back-End.
Banco de dados deverá ser utilizado Sqlite3.
Para o Front-End deverá ser utilizado VueJS ou AngularJS.
Você poderá fazer com a arquitetura de classes e modelos da forma que preferir, o importante é entregar o resultado de cadastro, consulta e operação.

A aplicação deverá conter 7 tabelas, são elas:

CUSTOMER			
Name	**Type**	**Required**	
ID	INT	NOT NULL	PK
NAME	VARCHAR(50)	NOT NULL	
CARD_ID	VARCHAR(10)	NULL	

VEHICLE			
Name	**Type**	**Required**	
ID	INT	NOT NULL	PK
PLATE	VARCHAR(10)	NOT NULL	
MODEL	VARCHAR(30)	NULL	
DESCRIPTION	VARCHAR(50)	NULL	
CUSTOMER_ID	INT	NULL	FK

PLAN			
Name	**Type**	**Required**	
ID	INT	NOT NULL	PK
DESCRIPTION	VARCHAR(50)	NOT NULL	
VALUE	FLOAT	NOT NULL	

CUSTOMER_PLAN			
Name	**Type**	**Required**	
ID	INT	NOT NULL	PK
CUSTOMER_ID	INT	NOT NULL	FK
PLAN_ID	INT	NOT NULL	FK
DUE_DATE	DATETIME	NULL	

CONTRACT			
Name	**Type**	**Required**	
ID	INT	NOT NULL	PK
DESCRIPTION	VARCHAR(50)	NOT NULL	
MAX_VALUE	FLOAT	NULL	

CONTRACT_RULE			
Name	**Type**	**Required**	
ID	INT	NOT NULL	PK
CONTRACT_ID	INT	NOT NULL	FK
UNTIL	INT	NOT NULL	
VALUE	FLOAT	NOT NULL	

PARKMOVEMENT			
Name	**Type**	**Required**	
ID	INT	NOT NULL	PK
ENTRY_DATE	DATETIME	NOT NULL	FK
EXIT_DATE	DATETIME	NULL	
VEHICLE_ID	ID		FK
VALUE	FLOAT	NULL	

Regras de projeto

Deverá conter uma tela inicial onde será a tela de operação, as demais telas deverão ser adicionadas no menu.

Deverá ter as seguintes telas de cadastro:

- Veículo
- Cliente
- Plano
- Contrato

Na tela de operação deverá conter os seguintes componentes:

- Placa
- Valor a cobrar(Desabilitado inicialmente)
- Uma grid com os veículos no pátio contendo data de entrada, placa, cartão.
- Botão de entrada
- Botão de saída(Desabilitado inicialmente)

Deverá conter os EndPoints abaixo, são eles:

- api/v1/customer - Para cadastro e edição de clientes
- api/v1/plan - Para cadastro e edição de planos
- api/v1/vehicle - Para cadastro e edição de veículos
- api/v1/contract - Para cadastro e edição de contratos
- api/v1/customervehicle - Para cadastro de veículos
- api/v1/customerplan - Para efetuar o vínculo do plano com o cliente
- api/v1/parkmovement - Para dar entrada/saída em movimentos do pátio

Regras de negócio a serem seguidas:

- Mensalista são clientes com planos vinculados, rotativos são clientes que não possuem plano, somente contrato para cálculo.
- Ao dar entrada ou saída em um veículo poderá ser utilizado os campos PLATE ou CARD_ID que estão nas tabelas.
- Caso o veículo não esteja cadastrado na base de dados deverá cadastrar o mesmo sem vínculo com cliente, pois ele será um veículo rotativo.
- Antes de cadastrar um veículo novo deverá ser validado se o mesmo está na base de dados, caso esteja sem vínculo com cliente deverá utilizar esse mesmo veículo e não cadastrar um novo, se a placa que estiver tentando cadastrar já é de um cliente o mesmo não poderá ser cadastrado.
- Não poderá dar entrada com CARD_ID que não está na base.
- Entradas somente são permitidas com CARD_ID cadastrados e com PLATES.
- Caso seja feita uma tentativa de entrada um cartão ou placa que já está no pátio, o mesmo não deverá ser possível, e deverá informar que já está no pátio e não pode ser finalizado a entrada
- Entrada e saída de mensalistas não são cobradas.

- Entrada de rotativos utilizem o contrato de cálculo rotativo, de momento deverá conter apenas 1 que será o contrato de teste.

- **Contrato**
    - Deverá ser cadastrado 1 único contrato, o mesmo terá regras baseadas em minutos, quanto mais minutos ficar no pátio mais deverá ser cobrado do veículo rotativo.
    - O contrato de cálculo possui um valor máximo estipulado, nossos clientes chama de diária, caso o valor do cálculo extrapole esse valor, ou seja seja maior que o valor máximo, então deverá ser cobrado o valor máximo, esse valor está presente no campo MAX_VALUE
    - O contrato deverá ser cadastrado e editado através de um único endPoint, onde as regras devem entrar como itens de lista dentro de um dos campos do json.

Faço a seguinte sugestão, foque em resolver a regra de negócio do estabelecimento em questão, então para efetuar isso cadastre os models e gere a base através do migrate do django, após isso faça o cadastro manual das informações na base e deixe a tela de operação operacional, após isso foque nas telas de cadastros.
A parte do front-end deixo em suas mãos.

Qualquer dúvida fico a disposição para tirar qualquer dúvida, mesmo no final de semana.
Agradeço de momento, e desejo um ótimo teste, que seja produtivo e que ele seja capaz de extrair o que há de melhor em você.
Peço que disponibilize em um diretório git, o mesmo peço que seja incluído como membro o e-mail wesley@cloudpark.com.br.