DESAFIO CAPGEMINI

Roteiro do Desafio

*** FASE 1 - API Rest ***

Desenvolvimento da API com os serviços propostos abaixo.

*** FASE 2 - Frontend em Vue JS ***

Criar uma tela que seja possível realizar as operações (saldo, depósito e saque).

O frontend deve consumir a API criada na FASE 1.

Informações sobre implementação (opcional)

Favor informar os seguintes pontos sobre sua implementação:

1) Faça uma breve descrição sobre a arquitetura utilizada no problema proposto;

A apiRest foi feita com o Laravel versão 5.4.36, foi utilizado o banco mysql para desenvolvimento. Como o tempo é corrido abstrai validação de token do usuário assim como tela de login e os demais cadastros deixando a api pronta para fazer essas operações e focar no Vue que não havia tanto conhecimento.

2) Em sua análise, houve algum problema relevante que gostaria de destacar?

Aprender o Vue nesse pequeno espaço de tempo, ja que não tinha tanta experiência com o Vue-cli e decidi usa-lo para ter uma estrutura e conhecimento mais específico do framework.

3) Houve alguma solução ou padrão arquitetural que tenha aplicado e gostaria de ressaltar?

Usei o componente v-money para formatar o campo do input para moeda, o Axios para as requisições HTTP e para otimizar o tempo utilizei o tamplate Vue Argon Design e pena que não consegui usar o componente de modal dele .

A base da url da api para alterar está no diretorio src/plugins/axios.js entro da pasta vue-argon

DESAFIO CAPGEMINI

OBJETIVO DO DESAFIO

O Banco Capgemini necessita criar uma Web API Rest para expor os seguintes serviços para seus clientes:

1) Serviço para verificar o saldo de uma conta corrente;

2) Serviço para realizar um depósito em uma determinada conta corrente;

3) Serviço para realizar um saque de uma determinada conta corrente.

REQUISITOS

1) A API Rest deve ser desenvolvida no framework Laravel (versão 5.2 ou superior);

2) O frontend deve ser desenvolvido utilizando Vue JS;

3) É necessário persistir algum dado, o candidato fica livre para propor a utilização de algum banco de dados. Por ter uma melhor portabilidade, sugerimos a utilização do banco em memória SqlLite em memória;

4) Deve ser utilizado Migrations;

5) A entrega deverá ser realizada via GitHub. O candidato deverá criar um repositório no GitHub para realizar a entrega do projeto (enviar o link ao final do desenvolvimento).

ANÁLISE

Serão avaliados os seguintes requisitos:

1) Solução do problema proposto;

2) Arquitetura da aplicação e sua distribuição em pacotes;

3) Legibilidade do código;

4) Utilização de orientação a objetos;

5) Melhores práticas de desenvolvimento;