Objetivo do Projeto - Teste para novos DEVs

Desenvolver um sistema de administração de clientes com funcionalidades de CRUD (Create, Read, Update, Delete). O projeto deve priorizar a experiência do usuário, garantindo uma interface visualmente atraente e responsiva, adaptada tanto para dispositivos móveis quanto para desktops.

Requisitos Gerais

1. **Formulários e Validações:**
    - Utilize as validações nativas do HTML 5 (como required, min, max, maxlength, type, etc.) para garantir a integridade dos dados conforme as regras especificadas abaixo.
    - Você tem a liberdade de usar frameworks SPA com JavaScript ou TypeScript, como React, Angular, PHP Laravel ou Vue.js, para a construção do frontend.
    - O uso de bibliotecas de terceiros, como jQuery e Bootstrap, é permitido para aprimorar o comportamento e o estilo do sistema;
    - Embora o aspecto visual seja um critério importante na avaliação e a forma como o programador estrutura seu código, apresentar seu trabalho e documentar também vai ser um requisito na avaliação.
2. **Integração com Banco de Dados:**
    - Utilize um banco de dados relacional de sua escolha (SQL Server ou MySQL). Caso você esteja utilizando um sistema operacional diferente do Windows, como macOS ou Linux, pode-se optar pelo uso de imagens Docker para o banco de dados.
    - A aplicação deve ser capaz de ser transformada em imagens Docker e deve ser acompanhado de um docker-compose com a documentação necessária para a execução.
3. **Expectativas do Projeto:**
    - Implementação de um frontend que consuma uma API Restful.
    - Desenvolvimento de um backend com uma WebAPI, preferencialmente utilizando um design pattern adequado na linguagem de sua escolha (por exemplo, .NET com C# e controllers).
    - O banco de dados deve ser configurado por meio de migrations ou scripts SQL disponibilizados pelo desenvolvedor.
    - Fornecimento de uma documentação clara e objetiva, que oriente sobre como iniciar o projeto em diferentes sistemas operacionais.
4. Caracteristica importante sobre o projeto:
    - O frontend deve se comunicar com seu backend através da WebAPI, e as regras de negócio, validações como CPF repetido no cadastramento, número de telefones inválidos entre outras regras que citaremos abaixo, pode ficar no seu Backend.
    - O backend deve se comunicar com o banco de dados relacional de sua escolha, para administrar seu sistema em CRUD.
    - O fluxo de dados, deve ser o seguinte:

1. Front-end consumo WebAPI -> Comunicação com o Backend -> Banco de dados <- Request
2. Banco de dados -> Response Backend -> Tratativas Backend -> Response Frontend.

Funcionalidades Específicas para a construção do sistema:

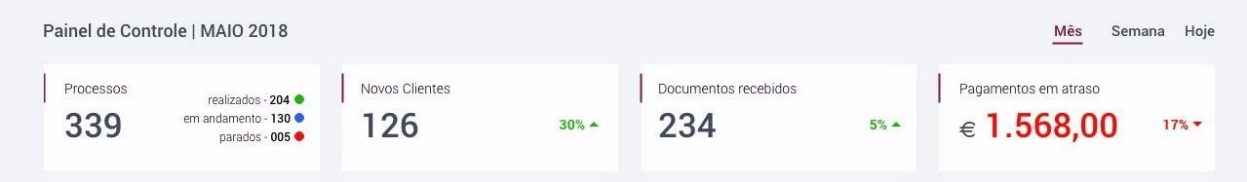
1. **Cadastro de Clientes:**
   - O sistema deve permitir que o usuário liste, insira, edite e exclua clientes.
   - Campos obrigatórios:
     - **Nome:** Texto de até 150 caracteres.
     - **CPF:** Texto de até 11 caracteres, somente dígitos, com validação de CPF, sem permitir duplicidades.
     - **Data de Nascimento:** Campo de data, obrigatório, não permitindo datas futuras.
     - **Data de Cadastro:** Campo de data de leitura única, preenchido automaticamente com a data atual na criação do cliente.
     - **Renda Familiar:** Campo decimal opcional, com valor mínimo de 0 (zero).
2. **Listagem de Clientes:**
   - Deve ser possível pesquisar clientes pelo nome.
   - Exibir uma lista com os seguintes campos: Nome e Renda Familiar.
   - A renda deve ser exibida dentro de um badge, que deve ser desenvolvido sem o uso de bibliotecas externas. A cor do badge varia conforme a classe econômica do cliente: 

     - **Classe A:** Renda até R$ 980,00, fundo vermelho.
     - **Classe B:** Renda entre R$ 980,01 e R$ 2.500,00, fundo amarelo.
     - **Classe C:** Renda acima de R$ 2.500,00, fundo verde.
3. **Relatórios:**
   - Criar uma página que exiba os dados em cards:
     - Quantidade de clientes maiores de 18 anos com renda familiar acima da média.
     - Quantidade de clientes nas classes A, B e C.
     - Filtro para visualizar os dados de acordo com o período: hoje, esta semana, ou este mês.

Considerações Finais

- O layout deve ser funcional, intuitivo e visualmente agradável, considerando boas práticas de UI/UX.
- A documentação deve ser clara, permitindo a execução do projeto em diferentes ambientes de desenvolvimento.

Segue um exemplo de um relatório em formato de card (o layout não precisa ficar exatamente como está na imagem):



A comunicação do desenvolvedor é crucial para essa avaliação. A capacidade de apresentar seu trabalho de forma técnica e acessível, sendo didático e compreensível, é fundamental no nosso mercado de desenvolvimento. Nesse caso de uso acima, o desenvolvedor quem vai passar o prazo de quanto tempo precisa para esse desenvolvimento, esse fator também será levado em conta.

O desenvolvedor deve subir o código no GitHub ou em uma ferramenta de gerenciamento de código e liberar os acessos para andre.bury@wittel.com e tiago.farias@wittel.com.

Caso o DEV tenha criado imagens docker do seu desenvolvimento, nos passar por e-mail a documentação e o docker-compose, além de acesso ao Docker Hub para download das imagens.

Lembrando que a análise vai ser feita sobre a apresentação e da estruturação do código.