Requisitos para o Desenvolvimento do MVP

Contexto:

Ao longo das aulas de Desenvolvimento Full Stack Básico a evolução da *web* foi nosso tema central. Vimos que desde a publicação da primeira página *web*, em 1991, muita coisa evoluiu e que seu crescimento exponencial logo nos primeiros anos foi a prova de seu sucesso, mas também um sinal de alerta. Como vimos, esse crescimento estava limitado às condições de infraestrutura da época e, além disso, não havia muita padronização na forma como as coisas eram feitas.

Como um caminho para garantir a continuidade da expansão da *web*, Roy Fielding propôs em 1993 que a escalabilidade da *web* fosse governada por um conjunto de *key constraints* (restrições/limitações-chave). Segundo ele, a continuidade da expansão da *web* estaria garantida se essas restrições fossem atendidas de forma igualitária. Essas restrições/limitações ainda são atuais, os conceitos ainda são válidos e elas devem ser observadas e atendidas no desenvolvimento de todo e qualquer sistema *web*.

É nesse contexto que vamos colocar em prática o conteúdo apresentado ao longo das aulas com o desenvolvimento de um *back-end* e de um *front-end*, explorando algumas dessas *key constraints*: (I) a separação de responsabilidades entre cliente e servidor; (II) a uniformidade de interfaces; (III) o desenvolvimento de sistemas em camadas; (IV) a ausência de estados; e (V) a execução de código sob demanda.

O seu objetivo é aplicar os conceitos aprendidos ao longo das aulas na criação de um MVP (Minimum Viable Product) de um sistema web inovador. Este sistema deverá abordar um problema real, oferecendo uma solução criativa que demonstre a compreensão das key constraints propostas por Roy Fielding. O desafio consiste em desenvolver tanto o backend quanto o frontend do sistema, você pode utilizar os códigos de exemplo apresentados em aula como base para desenvolver o seu próprio sistema. Todavia, não se limite a esses exemplos, explore novas possibilidades, seja criativo e inovador.

Observação 01: Antes de começar, certifique-se de ter o Python e todas as bibliotecas necessárias instaladas em seu computador. Você também precisará de um editor de código de sua preferência, como Visual Studio Code, Sublime Text ou Atom.

Requisitos e Composição da Nota

Pontuação	Requisito
	Back-end e Banco de Dados (4,0 pts)
1,5	A API deverá ser implementada em **Python e com Flask** com no mínimo 4 rotas (por exemplo, *"/cadastrar_usuario"*, *"/buscar_usuario", "/buscar_usuarios"* e *"/deletar_usuario"*). Pelo menos uma rota deve implementar o método POST (por exemplo, na rota de cadastro).
0,5	Fazer o uso do banco de dados **SQLite** com pelo menos uma tabela (por exemplo, tabela de usuários cadastrados).
1,0	Documentação da API com **Swagger** (OpenAPI), incluindo: ● Descrição clara de cada rota e método HTTP utilizado; ● Estrutura de requisição e resposta; ● Códigos de status esperados.
1,0	Criatividade e inovação: Será avaliado o que foi **implementado além do exemplo base**: novas rotas, funcionalidades extras, tratamento de datas, uso de relacionamentos e múltiplas tabelas. Demonstre iniciativa e aprofundamento técnico.
	Front-end (4,0 pts)
1,5	Desenvolvimento de uma **SPA (Single Page Application)** utilizando HTML, CSS e JavaScript.
1,0	Criatividade e Interatividade: será avaliada a originalidade da arquitetura visual da aplicação, considerando se a estrutura e organização da interface fogem do exemplo base fornecido. Isso inclui disposição dos elementos na tela, navegação, forma de interação com o usuário e quaisquer funcionalidades ou melhorias implementadas por iniciativa própria.
1,0	Deve-se exibir elementos em **lista ou cards**, como usuários, livros, etc.
0,5	Que, ao longo do código, seja feita a chamada a **todas as rotas** implementadas pela API.
	Observações: 1. Não utilize frameworks ou bibliotecas JavaScript baseados em SPA (Single Page Application) como Angular, Vue, ou React. O não cumprimento deste requisito resultará em uma penalização de 1,5 ponto na nota final. 2. É permitido o uso de frameworks de estilo, como Bootstrap. No entanto, é obrigatório incluir regras de estilização personalizadas em CSS. 3. O frontend deve ser executado corretamente ao abrir o arquivo index.html diretamente no navegador, sem a necessidade de extensões, servidores locais, configuração no navegador ou outras dependências adicionais. O não cumprimento deste requisito resultará em uma penalização de 2 pontos na nota final.

Organização dos Códigos (2,0 pts)	
1,0	Devem ser criados **dois projetos separados**: um para a API e outro para o front-end. Cada projeto deve estar em um repositório git próprio.
0,5	Em ambos os repositórios deve existir um arquivo README.md contendo as seguintes informações: ● Título e uma breve descrição do projeto; ● Instruções de Instalação, tais como: ○ descrever as etapas necessárias para que os usuários possam configurar o ambiente local, ○ instalar dependências, ○ comandos de inicialização, etc; ● Certifique-se de que o arquivo README.md seja formatado de forma clara e use cabeçalhos, listas e formatação de texto para tornar a documentação fácil de ler.
0,5	Qualidade da organização do código.

Atenção! *"Entregas que reutilizarem mais de 50% do código exemplo apresentado em aula serão penalizadas."*

Sobre a entrega:

Você deverá produzir um vídeo de **no máximo 4 minutos** que deve seguir o seguinte roteiro: Atenção: caso o tempo seja excedido, haverá desconto de 0,5 ponto na nota do vídeo.

Ordem	Descrição	Sugestão de tempo
1	**Apresentar o objetivo da aplicação desenvolvida.** *- Qual problema você está tentando resolver com a sua aplicação? Qual o propósito dela?*	De 20 a 60 segundos
2	**Apresentar a execução da API com você interagindo com as rotas (via Swagger)** *- Caso tenha implementado mais de 4 rotas você pode focar em apenas 4, aquelas que satisfazem ao primeiro requisito*	De 60 a 90 segundos
3	**Apresentar a execução do front-end com você interagindo com a aplicação mostrando em quais momentos cada rota da API é chamada**	De 60 a 90 segundos

Os três pontos apresentados fazem parte do roteiro e são igualmente importantes. Diante da ausência ou incompletude de um deles haverá uma penalização de até 2,0 pts à nota final (até 0,67 pts por ponto). **A não entrega do vídeo implicará na penalização em 2,0 pts à nota final.**

Você deverá também disponibilizar seus dois projetos em repositórios **públicos** do GitHub.

Sugestão: No momento da entrega publique o link completo **(com https://...)**, evitando o uso de textos clicáveis (hyperlinks).

Sugestão de mensagem de entrega:
Olá, segue os dados referentes à entrega do meu MVP. *Link para o vídeo: https://www.youtube.com....* *Link para o repositório do back-end: https://github.com/aluno123/back-end…* *Link para o repositório do front-end: https://github.com/aluno123/front-end…*

Se tiver dúvidas sobre como criar um repositório público no GitHub, consulte:https://docs.github.com/pt/repositories/creating-and-managing-repositories/creating-a-new-repository