Contexto

A componentização desempenha um papel fundamental na arquitetura de um sistemas web modernos, permitindo a construção eficiente, escalável e modular de aplicações online. Ao dividir um sistema em componentes independentes e reutilizáveis, a complexidade do desenvolvimento é reduzida, facilitando a manutenção, atualizações e correções.

Seguindo essa abordagem, cada componente pode ser projetado para cumprir uma tarefa ou entregar um serviço específico, como autenticação de usuários, gerenciamento de banco de dados, interface do usuário, entre outras. Essa modularização facilita o desenvolvimento paralelo por equipes distintas e oferece uma visão clara das interações entre as diferentes partes do sistema.

Para padrões de arquitetura como microsserviços, a componentização é fundamental. Cada microsserviço é uma componente separada que pode ser desenvolvida, implantada e escalada de maneira independente. Para esse contexto, vale reforçar que cada microsserviço deve ser uma unidade autônoma que desempenha uma função específica da aplicação.

É nesse contexto que vamos colocar em prática o conteúdo apresentado ao longo das aulas explorando a ideia de componentes e serviços. **Neste MVP queremos ver você entregando o seu melhor no desenvolvimento de um sistema web em que seus diferentes componentes atuem como serviços autônomos.**

Objetivo

O seu objetivo é desenvolver um sistema composto por no mínimo três componentes que se comunicam, seguindo o padrão REST e/ou GraphQL . Você não irá desenvolver todos eles, pelo menos um deles deve ser um componente externo (serviço externo como por exemplo o ViaCEP e FakeStore). A persistência de dados deve existir e ser feita utilizando o SQLite, MySQL ou PostgreSQL. Além disso, cada

componente desenvolvido deve ter o seu próprio repositório e, na raiz do repositório, deve existir um Dockerfile com as instruções que possam garantir a sua execução utilizando containers. Muita informação? **Na figura abaixo temos um esquemático de uma proposta de sistema mínimo.**

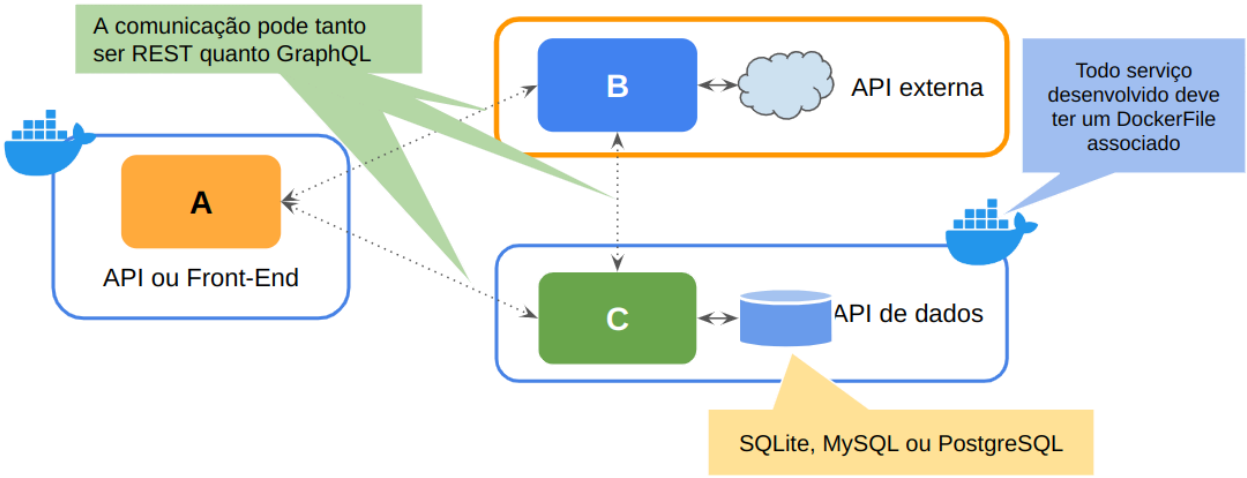

Figura 1. Exemplo de arquitetura que atende aos requisitos mínimos.

Na Figura 1, os componentes “A”, “B” e “C” ilustram um possível sistema que poderá ser desenvolvido **atendendo aos requisitos mínimos**. Os serviços que serão entregues por cada um desses componentes será definido por você. Observe que o componente “A” é o componente principal e pode ser um ***front-end*** ou uma ***API***. Além disso, cada componente deve se comunicar, pelo menos, com um outro (ou seja, não necessariamente o “A” deveria se comunicar com o “B” e o “C”, mas tem que se comunicar com pelo menos um deles)

Exemplos de cenários

Cenário 1: A componente “A” poderia ser o *front-end* de um sistema de compras online que consulta dados de um produto utilizando um serviço externo (componente “B”) e que tem um módulo de compras (componente “C”) para efetuar a compra.

Cenário 2: Definir a componente “A” como uma API que faz consulta a outros serviços. Para entender melhor esse cenário, podemos facilmente fazer a analogia com um serviço de acompanhamento de entregas, que consulta a distância entre dois

endereços via API externa (componente “B”) e que verifica a atualização do status da entrega pela componente “C”. Nesse cenário, o sistema representado pela componente “A” serve como um agregador de informações.

Antes de Tudo

Antes de começar, certifique-se de instalar o Docker, o Python e todas as demais bibliotecas necessárias em seu computador. Você também precisará de um editor de código de sua preferência, como Visual Studio Code ou outros. Para aqueles que optarem por desenvolver uma *interface* para usuários e se for fazer uso de biblioteca de componentes (React, Next, Vue, etc) certifiquem-se de instalar o Node.js e as dependências necessárias para o desenvolvimento dessa *interface*.

Link para instalação do Docker nos sistemas:

- Windows: https://docs.docker.com/desktop/install/windows-install/
- Ubuntu: https://docs.docker.com/engine/install/ubuntu/
- Mac OS: https://docs.docker.com/desktop/install/mac-install/

Requisitos e composição da nota

Pontuação	Requisito
Para o componente Principal (5,0 pontos)	
2,0	**[Caso seja uma API]** Implementar uma API REST em **Python** utilizando **Flask** com pelo menos 5 rotas. Deve existir pelo menos uma para cada um dos métodos: POST, PUT, DELETE, GET.
	Observação : A ausência de um dos métodos implicará na penalização de 0,5 pts.
	[Caso seja um Front-end] Desenvolver uma interface do usuário, utilizando **HTML**, **CSS** e **JavaScript**: ● Será permitido a utilização de bibliotecas ou frameworks baseadas em Javascript, como o React, Next, e outras; ● Será permitido também o uso de bibliotecas de componentes, como o Material UI, Bootstrap, e outras; ● A interface do usuário deve fazer chamadas a pelo menos 5 rotas diferentes.
	Observação : caso chame menos de 5 rotas, haverá uma penalização de 0,5 pts.
1,0	**[Caso seja uma API ou Front-end]** o seu README.md deve conter, no mínimo, as seguintes informações: ● Título e uma breve descrição do projeto; ● Instruções de instalação, tais como: descrever as etapas necessárias para que os desenvolvedores 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. ● É fortemente recomendado produzir uma imagem (fluxograma) ilustrando a arquitetura da aplicação desenvolvida. Use como exemplo a Figura 1.
1,0	*Dockerfile* para cada componente desenvolvido com todo o processo de implementação da solução em um *container docker*.
	Observação : a não execução correta ou não entrega do DockerFile implicará na penalização de 1,0 pts
1,0	Criatividade e Inovação
Para o segundo componente desenvolvido (3,0 pontos)	
2,0	Implementação de uma API REST ou GraphQL.
0,5	O seu README.md deve conter as seguintes informações: ● Título e uma breve descrição do projeto; ● Instruções de Instalação, tais como: descrever as etapas necessárias para que os desenvolvedores 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	*Dockerfile* para cada componente desenvolvido com todo o processo de implementação da solução em um *container docker*.
	Observação : a não execução correta ou não entrega do DockerFile implicará na penalização de 0,5 pts

Para a componente externa (2,0 pontos)	
1,0	Uso de uma **API externa pública** e que ofereça um serviço não pago.
b1,0	Apresentar na documentação a componente principal a API externa que será utilizada, deixando claro informações como: licença de uso (se aplicável), cadastro (se necessário) e rotas que foram utilizados.
Caso você queira inserir novos componentes (Opcional)	
0,5 extra	Para cada nova componente externa utilizada
1,5 extra	Para cada novo componente desenvolvido
	Atenção: Lembrar de que cada componente deve estar em um projeto/repositório diferente e não esquecer do Dockerfile.

Observação! *A nota poderá ser muito maior que 10, nós vamos te informar a nota após a avaliação, mas a nota final do MVP (a nota que vai aparecer no sistema) será limitada a 10 pontos.*

Atenção! *Caso sejam utilizados como base os exemplos apresentados em aula, serão penalizadas as entregas que não apresentarem pelo menos 50% de código novo.*

Sobre a entrega:

Você deverá produzir um vídeo de **no máximo 6 minutos** que deve seguir o seguinte roteiro:

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	***Apresentação da arquitetura e das estratégias de comunicação implementada***	De 30 a 60 segundos
3	**Apresentação da componente externa**	De 30 a 60 segundos
4	**Apresentação da segunda componente implementada.** *- Execução da API (via docker) com você interagindo com todas as rotas implementadas*	De 60 a 90 segundos
5	**Apresentação da componente principal:** *- **[caso seja uma API]** execução da API (via docker) com você interagindo com todas as rotas implementadas no Swagger.* *- **[caso seja um Front]** execução do front-end (via docker) com você interagindo com a aplicação mostrando em quais momentos são feitas as chamadas aos outros componentes.*	De 60 a 90 segundos

Os três pontos apresentados fazem parte do roteiro e **é obrigatório seguir essa ordem**. Os cinco pontos são igualmente importantes e diante da ausência ou incompletude de um deles haverá uma penalização de até 2,0 pts à nota final (até 0,6 pts por ponto). **A não entrega do vídeo implicará na penalização em 2,0 pts à nota final.**

Você deverá também disponibilizar seus componentes **em repositórios separados e públicos no GitHub**.

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 da componente principal: https://github.com/aluno123/minha_loja_front…* *Link para o repositório da segunda componente: https://github.com/aluno123/minha_loja_api…* *Foi utilizada a API 123, acessível pelo link: : https://api123.com/api*

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