UNIVERSIDADE DE FORTALEZA - UNIFOR
Disciplina: [T200] Projeto de Arquitetura de Sistemas
Prof. Doutorando Nathalino Pachêco

Mini Projeto

1. Objetivo

Desenvolver uma aplicação monolítica que opere como fachada/API Gateway para três microsserviços acadêmicos pré-existentes (Discente, Disciplina e Biblioteca) agregando e apresentando informações a usuários finais. O sistema não altera dados nos serviços externos, portanto, as operações de escrita são simulações locais voltadas a fins didáticos.

2. Escopo Funcional

2.1 Funcionalidades de Consulta (Leitura)

- **Consultar dados do discente** (id, nome, curso, modalidade e status da situação acadêmica [ativo ou trancado]).
- **Listar disciplinas** oferecidas, pelos respectivos cursos (id, curso, nome e vagas).
- **Listar livros** do acervo com status de disponibilidade (id, titulo, autor, ano e status da situação do livro [disponível ou indisponível]).

2.2 Funcionalidades de Simulação (Escrita local)

Os itens abaixo **não são persistidos** nos microsserviços; o estado vive apenas enquanto a aplicação estiver em execução.

- **Simular matrícula** e **cancelamento** de disciplina.
- **Simular reserva** e **cancelamento** de livro.

Essas simulações devem ocorrer por código de identificação único, por exemplo, a matrícula do discente é única e poderá ser utilizado para várias ações (reserva de livro e matrícula).

3. Arquitetura

- **Monolito interno** responsável por toda lógica de apresentação, negócio e persistência volátil;
- **Serviços externos simulados**: consumidos apenas por chamadas (leitura), representando microsserviços de diferentes domínios (Discente, Disciplina e Biblioteca); e
- **Repositório em memória**.

Justificativa da não-persistência permanente

O mini projeto tem duração máxima de ~34 dias e foco em integração de APIs e implementação com boas práticas, portanto, exigir um SGDB acrescentaria sobrecarga não alinhada ao objetivo pedagógico da disciplina. Existindo o interesse individual do discente, será incorporado como pontuação bônus.

4. Regras de Negócio e Restrições

1. Um discente pode simular matrícula em, no máximo, 5 disciplinas simultâneas, desde que sua situação acadêmica não esteja trancada e as disciplinas não pertençam a outros cursos.
2. Não é permitido matricular-se em disciplina sem vagas (informação fornecida pelo microsserviço: disciplina).
3. Um livro só pode ser reservado se marcado como disponível pelo microsserviço biblioteca.
4. O sistema não grava mudanças nos microsserviços, logo todos os estados simulados são descartados ao encerrar a aplicação individual do discente.

5. Microsserviços Externos

Para fins didáticos, o sistema integra três serviços simulados (*mock services/endpoints*) disponibilizados por meio do *AWS API Gateway*, que representam microsserviços acadêmicos independentes.

Cada serviço fornece dados estáticos em formato *JSON*, simulando o comportamento de um microsserviço real, sem regras de negócio, persistência ou implantação autônoma. O propósito é ensinar integração entre domínios

distintos e exercitar padrões arquiteturais (*API Gateway*, Objeto de Transferência de Dados, tratamento de falhas e isolamento de contexto).

Serviço	Endpoint Base	Descrição
Discente	`https://rmi6vdpsq8.execute-api.us-east-2.amazonaws.com/msAluno`	Dados dos discentes
Disciplina	`https://sswfuybfs8.execute-api.us-east-2.amazonaws.com/disciplinaServico/msDisciplina`	Oferta de disciplinas e o quantitativo de vagas disponíveis
Biblioteca	`https://qiiw8bgxka.execute-api.us-east-2.amazonaws.com/acervo/biblioteca`	Acervo e disponibilidade dos livros

Observação: estes *endpoints* não implementam todas as características de microsserviços reais (autonomia de deploy, banco de dados próprio, versionamento de API, observabilidade, etc.), mas seguem o modelo conceitual de microsserviços e permitem praticar a integração e a aplicação de padrões arquiteturais modernos.

6. Tecnologias e Padrões de Projeto

- O **MVC** é o padrão estrutural sugerido. Caso seja escolhida outra arquitetura, é dever do discente demonstrar, no projeto prático, a aplicação da arquitetura e a responsabilidade de suas camadas.
- Aplicar no **mínimo de 2 princípios SOLID** (recomendado: SRP e DIP) e (3) **GRASP** (sugerido: Low Coupling, High Cohesion, Controller e Polimorfismo).
- **Padrões GoF são opcionais.** Caso sejam utilizados, o discente deve justificar explicitamente no código-fonte o padrão aplicado.

- Linguagem sugerida: **Java (Conforme disciplina de POO na Universidade)**
    - Posso utilizar outra linguagem de programação orientada a objetos? Sim, desde que o(a) discente assuma as responsabilidades pela escolha, sendo plenamente possível oferecer apoio durante o processo.

7. Requisitos Não Funcionais

Categoria	Requisito
Usabilidade	Utilizar as 10 Heurísticas de Nielsen como parâmetros. (UI e UX)
Desempenho/ Eficiência	As requisições aos serviços externos devem ter tempo de resposta ≤ 3 segundos sob condições normais. Ao passar do parâmetro deve ser registrado em logs (opcional: AOP).
Tolerância a Falhas / Degradação	Em caso de falha em um serviço, a aplicação deve degradar graciosamente (mensagem amigável ou fallback).
Manutenibilidade	O código deve seguir padrões de nomenclatura, separação de pacotes e ausência de duplicação significativa.
Documentação	Incluir *Readme*.md com descrição da arquitetura, padrões adotados, decisões de design e instruções de execução.

8. Formação / Composição

Individual.

9. Entrega do Material

O projeto do sistema desenvolvido deverá ser compactado e submetido no AVA. Não será permitido links em repositórios de código por motivos óbvios: indisponibilidade do projeto posteriormente, inviabilizando reavaliações.

10. Critérios de Avaliação

1. **Integração correta com os três serviços externos simulados** (“microsserviços” acadêmicos).
2. Implementação **do padrão MVC** (ou definição clara de outra arquitetura utilizada), com aplicação de pelo menos duas regras **SOLID** e três padrões **GRASP**.
3. **Consistência terminológica**, em conformidade com os microsserviços.
4. **Manipulação correta dos dados**, com especificação da estrutura utilizada.
5. **Persistência dos dados em memória** e, caso haja armazenamento em um banco de dados, será concedida bonificação.
6. **Interface do usuário (UI) compatível com as tecnologias utilizadas e suas limitações** — bônus será concedido em casos de interfaces com recursos adicionais.
7. **Documentação completa e robusta** de todo o projeto, detalhando principalmente a arquitetura.
8. **Em caso de atraso ou ausência na entrega**, será considerado um **decréscimo** de **0,2** ponto na nota final.
9. **Outros critérios poderão ser utilizados de acordo com a especificação deste documento**, ficando evidente na descrição da nota final. Portanto, será considerada também a apresentação, domínio e alterações sobre o projeto.

11. Glossário

Termo	Definição
Discente	Pessoa regularmente matriculada na instituição.
Simulação	Operação que altera apenas o estado em memória, sem afetar microsserviços.
Fachada/API Gateway	Camada que unifica chamadas aos serviços externos e provê API própria.

UX	*User Experience*
UI	*User Interface*

12. Referências

Gamma, E. et al. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

Aniche, Mauricio. Orientação a Objetos e SOLID para Ninjas: Projetando classes flexíveis. Editora Casa do Código, 2015.

Larman, Craig. Utilizando UML e Padrões-Uma Introdução à Análise e ao projeto Orientado a Objetos e ao desenvolvimento Interativo; 3ª. Bookman, 2007.

RICHARDS, Mark; FORD, Neal. Fundamentos da arquitetura de software: uma abordagem de engenharia. Rio de Janeiro: Alta Books, 2024. 1 recurso online. ISBN 9788550819754. Disponível em: https://integrada.minhabiblioteca.com.br/books/9788550819754. Acesso em: 05 jan 2024.

SOMMERVILLE, Ian. Engenharia de software. 9. ed. São Paulo: Pearson Prentice Hall, 2011.