Resposta do Exercício - Arquitetura Três Camadas (Three-Tier)

1. Descrição das Três Camadas do Sistema de Matrícula

Camada de Apresentação (View):

- Interface web acessada pelo aluno via navegador.
- Exibe lista de disciplinas disponíveis.
- Permite selecionar disciplinas e confirmar matrícula.
- Mostra mensagens de erro (ex: 'choque de horário') ou sucesso.

Camada de Aplicação (Lógica de Negócio / Controller):

- Responsável por:
  - Validar login do aluno.
  - Verificar regras de matrícula (pré-requisitos, carga horária máxima, choque de horário).
  - Enviar comandos para inserir ou consultar dados.
- Geralmente construída com frameworks como Django (Python), Spring (Java), Laravel (PHP), Express (Node.js).

Camada de Dados (Model / Banco de Dados):

- Armazena:
  - Tabela alunos (matricula, nome, curso)
  - Tabela disciplinas (codigo, nome, horario)
  - Tabela matriculas (aluno_id, disciplina_id)
- SGBD: PostgreSQL, MySQL ou outro relacional.

2. Exemplo de Interação do Usuário (Fluxo)

Cenário: O aluno 'João Silva' deseja se matricular na disciplina 'Banco de Dados II'.

Passo a passo:

1. João acessa o sistema pelo navegador e faz login. (Apresentação)
2. A interface exibe todas as disciplinas disponíveis para o curso dele. (Apresentação + Aplicação)
3. João clica em 'Matricular-se' na disciplina 'Banco de Dados II'. (Apresentação)
4. O sistema envia a requisição para o servidor. (Aplicação)
5. A aplicação verifica:
   - Se João já está matriculado em outra disciplina no mesmo horário.
   - Se ele atende os pré-requisitos.

- Se a carga horária total não será excedida.

6. Tudo ok? O sistema grava a matrícula na tabela matriculas. (Dados)

7. A aplicação envia a resposta de sucesso para a interface. (Aplicação -> Apresentação)

8. João vê a mensagem: 'Matrícula realizada com sucesso!'

3. Importância da Separação em Camadas

- Organização e clareza: cada camada tem uma responsabilidade clara, facilitando o desenvolvimento em equipe.

- Segurança: o banco de dados não é acessado diretamente pelo usuário, reduzindo vulnerabilidades.

- Manutenção facilitada: é possível atualizar a interface sem alterar a lógica de negócio ou o banco.

- Escalabilidade: se o número de alunos crescer, a aplicação pode ser distribuída em servidores diferentes.

- Reutilização: a mesma camada de aplicação pode servir tanto um app web quanto um aplicativo mobile.