Anderson Schieck Lopes

6 - Reflexões sobre as Decisões Arquiteturais:

Uso de record para DTOs (CreateUserDTO, CreatePhoneDTO, UserDTO, PhoneDTO):

Justificativa: Os record são ideais para DTOs porque são imutáveis, reduzem boilerplate (getters, setters, etc.) e são mais legíveis. A imutabilidade aumenta a segurança ao evitar modificações acidentais dos dados.

Separação de DTOs de Entrada e Saída:

Decisão: Usamos CreateUserDTO para entrada e UserDTO para saída.

Justificativa: Essa separação segue o princípio de responsabilidade única. O CreateUserDTO não inclui o id (gerado pelo banco), enquanto o UserDTO inclui o id e os dados completos, o que é útil para o cliente.

Uso de Mappers (UserMapper e PhoneMapper):

Justificativa: Separar a lógica de mapeamento em classes dedicadas torna o código mais modular, facilita testes e permite reutilização em diferentes partes da aplicação.

Configuração da Relação @OneToMany com cascade = CascadeType.ALL:

Justificativa: O cascade = CascadeType.ALL simplifica o salvamento dos telefones ao salvar o usuário, e o orphanRemoval = true garante que telefones removidos da lista sejam deletados do banco, mantendo a consistência.

Uso do UserService para Cadastro:

Justificativa: Delegar a lógica de cadastro ao UserService segue o padrão de camadas (Controller -> Service -> Repository), mantendo a lógica de negócio fora do controller.

Endpoint RESTful:

Justificativa: O uso de POST em /users segue as convenções REST, e retornar o status HTTP 201 (Created) com o UserDTO fornece feedback claro ao cliente.