Clean Architecture - Camadas

1. Presentation Layer (Camada de Apresentação)

- **Responsabilidade**: Entrada e saída de dados (REST APIs).
- **Tecnologias**: Spring Web (`@RestController`)
- **Componentes**:
  - `CepController`
  - Validação de entrada
  - Conversão de DTOs

2. Application Layer (Camada de Aplicação)

- **Responsabilidade**: Orquestração de casos de uso da aplicação, lógica de negócios sem dependência de frameworks.
- **Componentes**:
  - `CepService`
  - Casos de uso (`ConsultarCepUseCase`)
  - Interfaces (`CepGateway`, `CepRepository`)

3. Domain Model (Modelo de Domínio)

- **Responsabilidade**: Entidades de negócio e regras do domínio puras.
- **Componentes**:
  - `Cep`
  - Validações internas
  - Objetos de valor (ex: `Endereco`)

4. Infrastructure Layer (Infraestrutura)

- **Responsabilidade**: Integrações externas e persistência.
- **Tecnologias**:
  - Spring Data JPA
  - WireMock
  - Docker
  - PostgreSQL
  - AWS (RDS, ECS)
- **Componentes**:
  - `CepRepositoryImpl` (implementação JPA)
  - `ExternalCepClient` (chamada à API externa via `RestTemplate` ou `WebClient`)
  - `LogRepository` para armazenar logs no banco

5. Test Layer (TDD e QA)

- **Responsabilidade**: Garantir cobertura e qualidade do código.
- **Tecnologias**: JUnit 5, Mockito, Testcontainers, WireMock
- **Testes**:

- Unitários (serviços, casos de uso)
- Integração (chamada à API externa mockada)
- Contratos (testes com dados simulados)

Fluxo de chamada

1. `CepController` recebe uma requisição REST.
2. Chama `ConsultarCepUseCase` na camada de aplicação.
3. O use case delega para `ExternalCepClient` (mockada com WireMock) e salva log com `CepRepository`.
4. O resultado é retornado para o controller e enviado ao usuário.