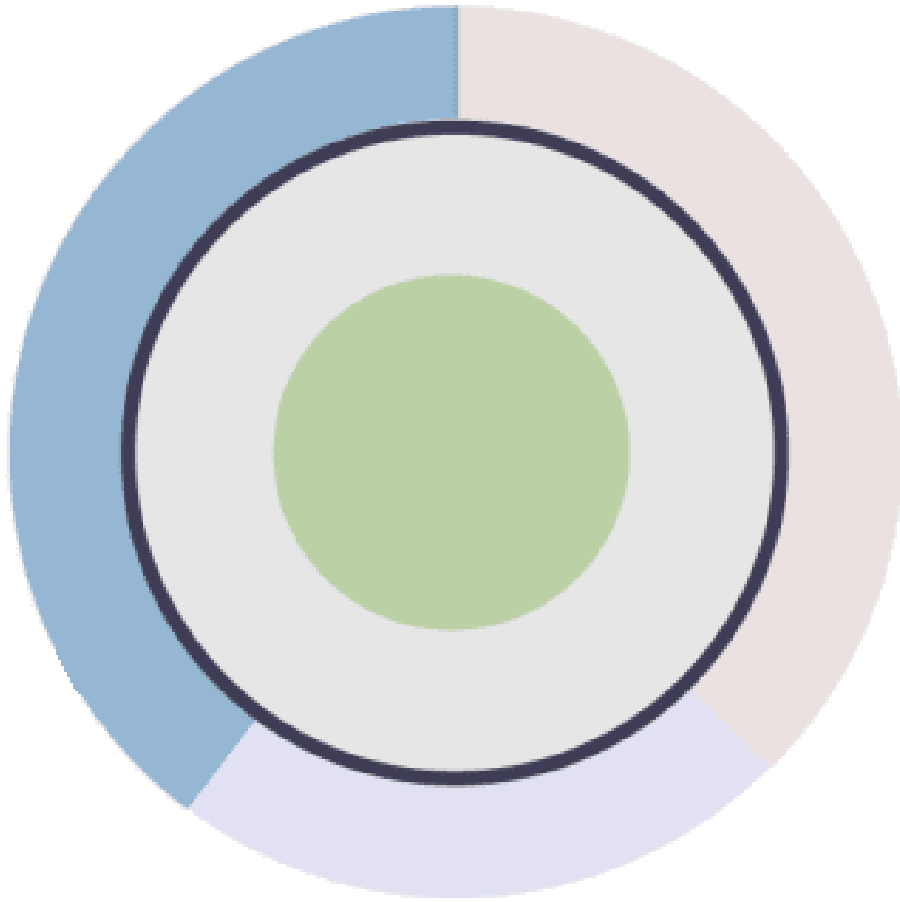


DDD - Domain Driven Design



Diogo Antonio Sperandio Xavier

SUMÁRIO

[Introdução](#)

[Prática](#)

[Camadas](#)

[Projeto rascunho:](#)

[Leitura adicional:](#)

Introdução

O objetivo do material em questão é apresentar um overview sobre **DDD (Domain driven design)**. Não são necessários conhecimentos em programação para compreender os conceitos e a ideia, mas caso possua algum, fica mais fácil o aprofundamento no tema.

Espero que esse livro possa colaborar com seu desenvolvimento de alguma forma, seja com conhecimento ou com um passo-a-passo que melhore sua venda e seus negócios. Caso isso aconteça, já estou de antemão muito feliz por isso.

Prática

Vale ressaltar que não importa qual a linguagem de programação vai ser utilizada, sistema de mensageria ou banco, a estrutura se mantém. Abaixo um Exemplo:

```

  ▾ MODELO DDD
    ▾ app
      ▾ router
        -go router.go
      ▾ utils
        -go env-variables.go
      -go main.go
    ▾ domain
      ▾ database
        -go business-entity.go
      ▾ kafka
        -go business-kafka.go
      ▾ logs
        -go log.go
    ▾ infra
      ▾ database
        -go db.go
      ▾ kafka
        ▾ config
          -go kafka.go
        ▾ consumer
          -go kafka-consumer.go
      ▾ repository
        -go business-repo-impl.go
        -go business-repo.go
    ▾ usecases
      -go business-usecase-impl.go
      -go business-usecase.go
```

Camadas

- Infra: Camada responsável pelas conexões e configurações. Tudo relacionado diretamente a infra, no caso do exemplo, banco de dados, repository (interface e implementação) e kafka, deve ficar nessa cama.

```
▼ infra
  ▼ database
    db.go
  ▼ kafka
    ▼ config
      kafka.go
    ▼ consumer
      kafka-consumer.go
  ▼ repository
    business-repo-impl.go
    business-repo.go
```

- Domain: Camada responsável pelas entidades. Aqui ficam os modelos, dtos e padrões do código.

```
▼ domain
  ▼ database
    business-entity.go
  ▼ kafka
    business-kafka.go
  ▼ logs
    log.go
```

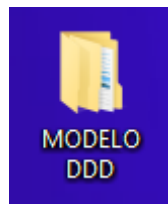
- Usecases: Camada responsável pelas regras do seu sistema onde se encontram as interfaces e implementações relacionadas a serviço. Pode ser chamada de camada de serviços (Service).

```
▼ usecases
  business-usecase-impl.go
  business-usecase.go
```

- App: Camada de exposição da API. Aqui também ficam os controllers, caso seja necessário, além de routers, utils e o main do projeto.

```
▼ app
  ▼ router
    router.go
  ▼ utils
    env-variables.go
  main.go
```

Projeto rascunho:



Leitura adicional:

- <https://medium.com/beelabacademy/domain-driven-design-vs-arquitetura-em-compartamentos-d01455698ec5>
- <https://www.devmedia.com.br/java-e-domain-driven-design-na-pratica-java-magazine-87/19019>