Backend with Go

~APIサーバーの開発で私が考えていること~

自己紹介

- 名前: 鈴木 進也
  - yanyanと呼ばれています
- 株式会社fluct アドプラットフォーム事業本部 開発本部所属
  - GoでGraphQLサーバーを書いたり、データエンジニアリングをしています
- 趣味
  - valorant
  - FF14 (最近始めました)
  - キーボードで散財

お題目

- バックエンドアプリケーションのアーキテクチャの話
- API設計について
- テストの話
- 思想を言語化する

アーキテクチャの話

ここでいうアーキテクチャとは

- アプリケーションの実装をレイヤーごとに分けて整理する
- レイヤーに分けることによって以下のことが達成できる
  - 関心事の分離
  - 依存関係の整理

よく目にするアーキテクチャたち

- レイヤードアーキテクチャ
- ヘキサゴナルアーキテクチャ
- オニオンアーキテクチャ
- クリーンアーキテクチャ
- etc...

彼らは銀の弾丸ではない

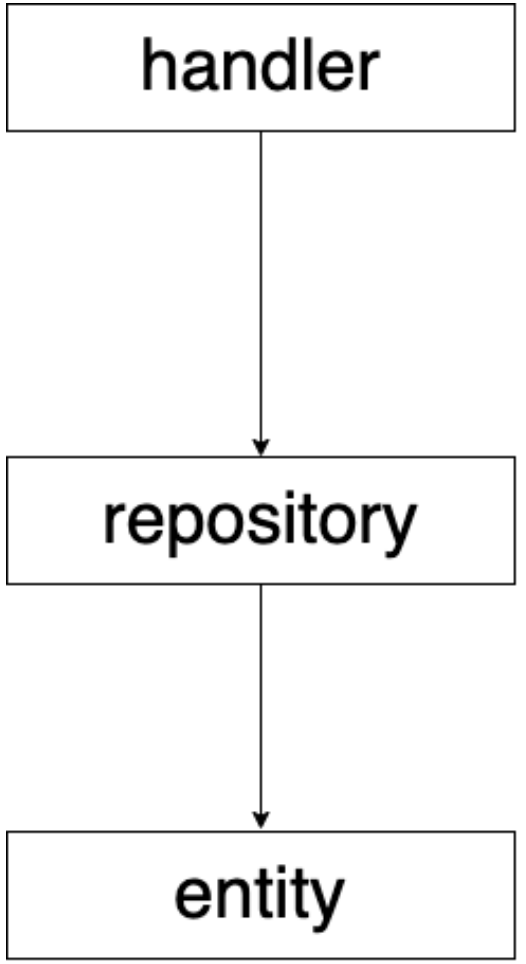
いかなるアプリケーションでも、このアーキテクチャを適用しとけばよいというわけではない

必要なときに増築する

- 良いアーキテクチャは、開発が進むにつれて変わっていくもの
- アプリケーションの規模が小さい段階から、壮大なアーキテクチャにしようとすると色々めんどくさい
  - ほとんどなにもしてないレイヤーが生まれる
  - 意味のない抽象化
  - なぜそのレイヤーが存在しているのかわからない = 認知負荷が高い
- 大事な考え方を守りつつ、必要に応じて層を足したり抽象化をすればよい

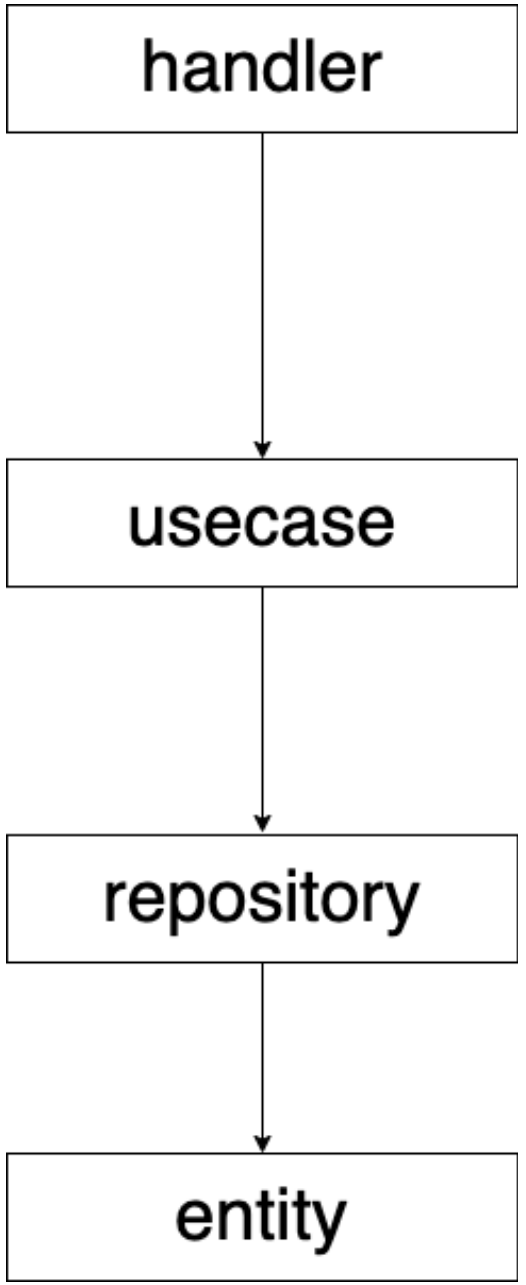
例えば

- 単に来たリクエストに応じてCRUDするだけならこれくらい素朴でもいい
- 開発したいことに応じてアーキテクチャも変化させていく



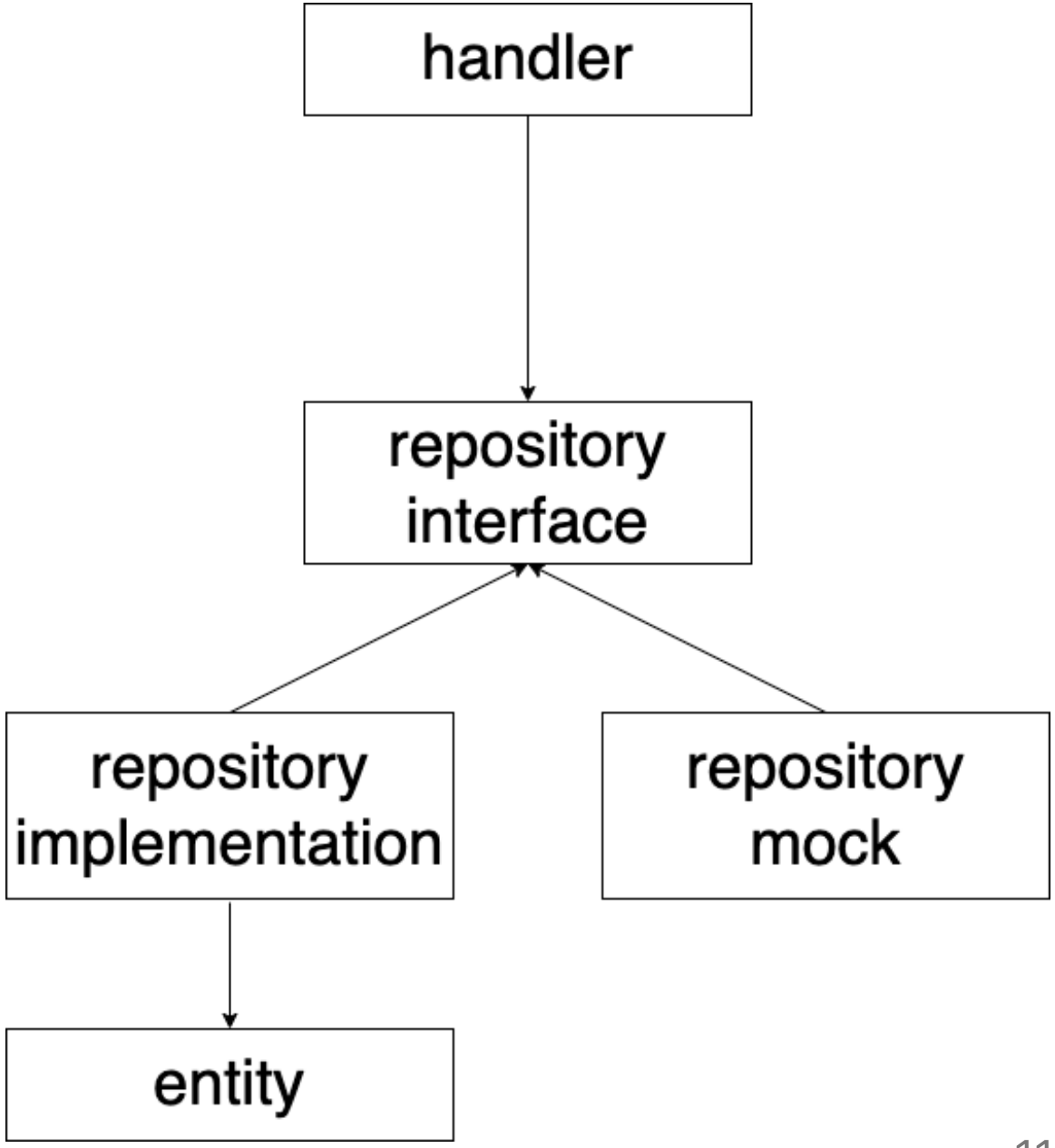
扱う関心事が増えた

- ビジネスロジックを書く層がほしい！
    - あとから足せば良い



抽象化したい

- repositoryに依存する層のユニットテストをしたい！
  - repositoryの部分はフェイクに差し替えたい
- インターフェースに依存する形にする
  - 具象が１個だけなら抽象化する必要もない



大事な考え

私がアーキテクチャの構造を考えるときに守りたいこと

1. 関心事の分離
2. 依存の流れを1方向にする

これらを守りながら、その時々でベストな設計を模索する

関心事の分離

- 関心事とは
  - 働きかける対象
  - e.g.) DBとのやりとり、HTTP req/resについてetc...

関心事の分離

- まずは存在する関心事を言語化することが大事
- 1レイヤーが複数の関心事を扱わないようにする
    - e.g.) ファットコントローラー

```
func BanbutsuHandler(w http.ResponseWriter, r *http.Request) {
    // http requestのあれこれ

    // ビジネスロジック

    // DBとのやりとり

    // http responseのあれこれ
}
```

各層が１つの関心事しか扱わないとどう嬉しい？

- 認知負荷が低い
  - 触りたい実装がどこにあるかが把握しやすい
  - e.g.) DB周りはrepository層をみればおｋ
- 変更しやすい
  - 変更するためにいじらなければならない箇所が明確になる
- 壊れたときに直しやすい
  - 壊れた原因が特定しやすい

依存関係

- レイヤー構造を成すので、レイヤー間に依存関係が生まれる
- 依存とは
  - 依存される側の知識が依存する側に漏れ出ている状態
  - メソッドの呼び出しに必要な引数とか
- 依存される側に変更が入ると、する側も影響を受ける
  - あるモジュールが依存したりされたりしまくっている (密結合) と辛い

依存の流れを1方向にする

- 依存の流れを交通整理する
- 具体的な関心事をもつレイヤー -> 抽象的な関心事を持つレイヤーという依存の流れを守る

memo: 円環構造の図を載せる

API設計について

- memo
  - rest, graphql, gRPC といったAPIスタイルに選択肢がある
  - 自分はいまgraphqlを使っているよ