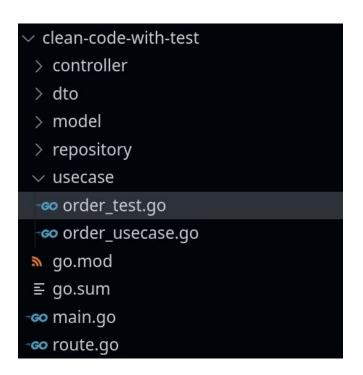




Write Unit Test



Structure Code



Kita akan memakai struktur kode yang sama pada materi clean code. Untuk kodenya bisa diakses <u>disini</u>



Write Unit Test

```
func TestCreateOrder(t *testing.T) {
   data := []dto.CreateOrderRequest{
        {ProductID: 1, Qty: 6},
   }
   ...
}
```

Pada file **order_test.go** buat sebuah fungsi testing, dan buat contoh dari payload atau data yang akan dikirim.



Write Mock Repository

```
• mock product repository.go
 package repository
 import (
     "github.com/born2ngopi/alterra/mvc/model"
    mock "github.com/stretchr/testify/mock"
 type mockProductReposory struct {
    mock . Mock
 func NewMockProductRepo() *mockProductReposory {
    return &mockProductReposory{}
 func (m *mockProductReposory) Find(productID uint)
  (model.Product, error) {
    ret := m.Called(productID)
    return ret.Get(0).(model.Product), ret.Error(1)
```

Buat mocking repository, disini kita menggunakan library **testify**.

Memberitahu bahwa mock atau object tiruan telah dipanggil.

ret.Get() akan membalikkan data sesuai indexnya, karena data bertipe interface{} maka kita perlu untuk mengcastingnya



Add Mock Data to testing file

```
usecase_test.go

...
mockProduct := model.Product{
    ID:    1,
    Name: "product-mock",
    Price: 4500,
    Stock: 10,
}

mockProductRepository := repository.NewMockProductRepo()
mockProductRepository.On("Find", uint(1)).Return(mockProduct,
nil)
...
```

Tambahkan data untuk mocking product

Find adalah nama method di repository yang akan dipanggil, dan kita set expected resultnya



Call Function You Want to Test

```
usecase_test.go

...
    service := NewOrderUsecase(mockOrderRepository,
    mockProductRepository)

if err := service.Create(data); err != nil {
    t.Errorf("Got Error %v", err)
}
...
```

Lalu panggil fungsi yang akan kita test

