dito

Mocks

Um idioma genérico para testar pacotes.

1

Quem sou eu?

Prazer, Matheus Andrade .

Faço parte do time de infraestrutura da Dito

- Dificuldade inicial em testar
- Solução
- Casos de uso
- Outras opções de ferramentas

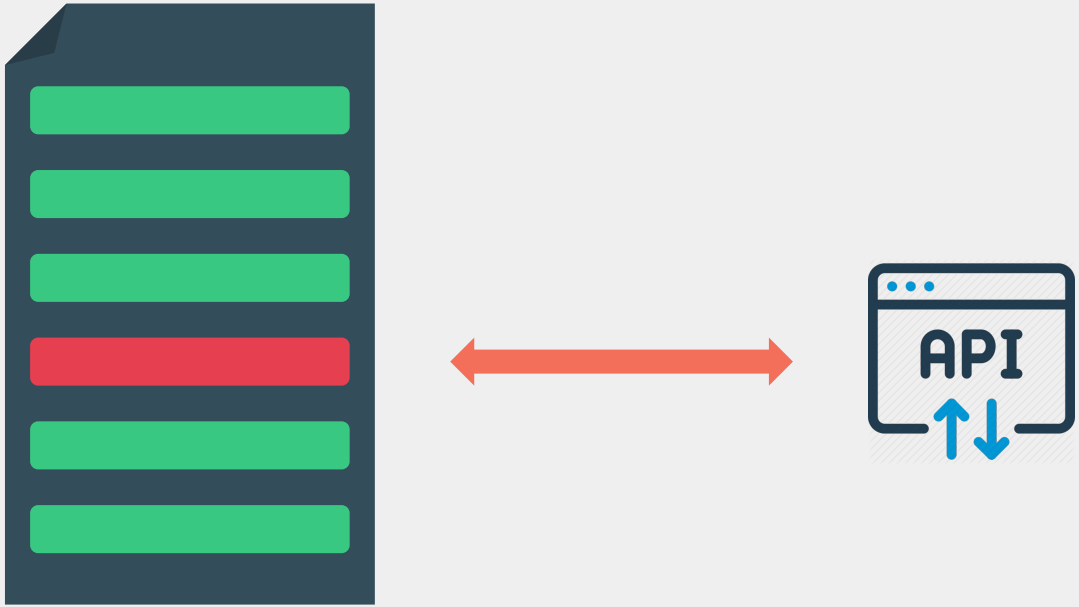
2

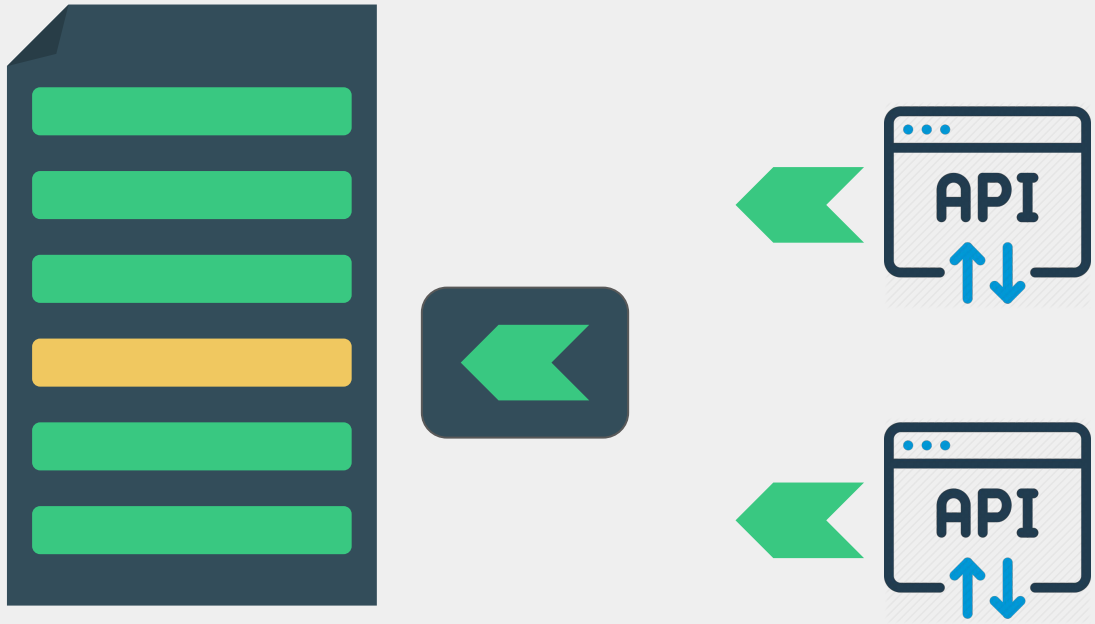
A necessidade

Imagine que você precisa desenvolver um serviço de envio de emails.

Sua equipe opta por utilizar um provedor que possui uma determinada API para envios

```go
type Email struct {
    Message string   `json:"message"`
    Subject string   `json:"subject"`
    From    string   `json:"from"`
    To      []string `json:"to"`
}

func SendEmail(email Email) error {
    emailPayload, _ := json.Marshal(email)

    request, _ := http.NewRequest(
        "POST",
        "https://api.greatmail.com/send",
        ioutil.NopCloser(bytes.NewReader(emailPayload)),
    )

    _, err := http.DefaultClient.Do(request)
    if err != nil {
        return err
    }

    return nil
}
```


API


API
API

```
type httpProvider interface {
    Do(*http.Request) (*http.Response, error)
}

type Client struct {
    http httpProvider
}
```

```go
func NewClient() Client {
  return Client{
    http: &http.Client{
      Timeout: 10 * time.Second,
    },
  }
}

func (c Client) SendEmail(email Email) error {
  /* previous code */
  _, err := c.http.Do(request)
  if err != nil {
    return err
  }

  return nil
}
```

3

Vamos testar nossas requisições

```go
type httpProviderMock struct {}


func (m httpProviderMock) Do(request *http.Request) (*http.Response, error) {

  return &http.Response{
    StatusCode: 200,
  }, nil
}
```

```go
type httpProviderMock struct {
  DoFn func(*http.Request) (*http.Response, error)
}


func (m httpProviderMock) Do(request *http.Request) (*http.Response, error) {
  return m.DoFn(request)
}
```

```go
func TestSendEmail(t *testing.T) {

    t.Run("should return an error when fail to send email", func(t *testing.T) {

        client := Client{
            http: httpProviderMock{
                DoFn: func(*http.Request) (*http.Response, error) {
                    return nil, errors.New("fail to send email")
                },
            },
        }

        err := client.SendEmail(Email{})
        if err == nil {
            t.Error("invalid error received, expeted a non-nil error")
        }
    })
}
```

4

Explorando alguns casos de teste

```
t.Run("should send an email with a correct body", func(t *testing.T) {
  var receivedRequest *http.Request

  client := Client{
    http: httpProviderMock{
      DoFn: func(request *http.Request) (*http.Response, error) {
        receivedRequest = request
        return nil, nil
      },
    },
  }

  msg, _ := json.Marshal(createFakeEmail())
  expectedBody := ioutil.NopCloser(bytes.NewReader(msg))
  client.SendEmail(createFakeEmail())
  assert.Equal(t, expectedBody, receivedRequest.Body, "wrong email sent")
})
```

```go
t.Run("should fail in the second sent", func(t *testing.T) {
  count := 0
  errs := []error{nil, errors.New("failed")}

  client := Client{
    http: httpProviderMock{
      DoFn: func(request *http.Request) (*http.Response, error) {
        count++
        return &http.Response{}, errs[count-1]
      },
    },
  }

  expectedErrs := []error{nil, errors.New("failed")}

  outputErr := client.SendEmail(createFakeEmail())
  assert.Equal(t, expectedErrs[0], outputErr, "wrong error returned")

  outputErr := client.SendEmail(createFakeEmail())
  assert.Equal(t, expectedErrs[1], outputErr, "wrong error returned")
})
```

5

Quais são as outras formas existentes ?

GoMock

- Framework de mocks
- Gera arquivos automaticamente
- Constrói as nossas instâncias de mock

```go
//go:generate  mockgen -destination=mock_greatmail.go -package=greatmail main/greatmail HTTPProvider

// HTTPProvider ...
type HTTPProvider interface {
    Do(*http.Request) (*http.Response, error)
}
```

```
t.Run("should send an email correctly", func(t *testing.T) {
    ctrl := gomock.NewController(t)
    defer ctrl.Finish()

    mockProvider := NewMockHTTPProvider(ctrl)
    sendTest := Client{http: mockProvider}

    msg, _ := json.Marshal(createFakeEmail())
    request, _ := http.NewRequest(
      "POST",
      "https://api.greatmail.com/send",
      ioutil.NopCloser(bytes.NewReader(msg)),
    )

    mockProvider.EXPECT().Do(request).Times(1)
    sendTest.SendEmail(createFakeEmail())
})
```

Http Test

- Nativo do http padrão
- Sube um pequeno servidor local
- Podemos fazer requisições utilizando a url da estrutura criada

```
func NewClient() Client {
  return Client{
    http:    &http.Client{},
    baseURL: "https://api.greatmail.com",
  }
}
```

```
func (c Client) SendEmail(email Email) error {
  /* previous code */
  request, _ := http.NewRequest(
    "POST",
    c.baseURL+"/send",
    ioutil.NopCloser(bytes.NewReader(emailPayload)),
  )
  /* previous code */
}
```

```
t.Run("should send an email correctly", func(t *testing.T) {
  server := httptest.NewServer(http.HandlerFunc(func(rw http.ResponseWriter, req *http.Request) {
    expedtedBody, _ := json.Marshal(createFakeEmail())
    body, _ := ioutil.ReadAll(req.Body)

    assert.Equal(t, expedtedBody, body, "wrong email received")
    rw.Write([]byte(`ok`))
  }))
  defer server.Close()

  client := Client{
    http:    &http.Client{},
    baseURL: server.URL,
  }

  client.SendEmail(createFakeEmail())
})
```

Escolha sua lib
Nada é uma regra fixa

- Testify
- GoMock
- GinkMoc
- Go

dito

OBRIGADO.

Matheus Andrade

Back-end engineer
matheus.andrade@dito.com.br

linkedin.com/in/jonicreide/
github.com/jonicreide

dito.com.br

github.com/jonicreide/tdc-go-examples/