



Team: NDY corp.

Progress Report 2

Date: March 05, 2023

		
Amangeldin Erasy	Amanzhol Nursultan	Shulanbay Darkhan
id:200103252	id:200103444	id:200103038

```
run test | debug test
func TestUser_Validate(t *testing.T) {
    testCases := []struct {
        name    string
        u        func() *model.User
        isValid bool
    }{
        {
            name: "valid",
            u: func() *model.User {
                return model.TestUser(t)
            },
            isValid: true,
        },
        {
            name: "empty email",
            u: func() *model.User {
                u := model.TestUser(t)
                u.Email = ""
                return u
            },
        },
    }

    // its our test cases
    for _, tc := range testCases {
        t.Run(tc.name, func(t *testing.T) {
            u := tc.u()
            if !tc.isValid {
                t.Errorf("User %s is not valid", u.Name)
            }
        })
    }
}
```

MS 2 OUTPUT DEBUG CONSOLE TERMINAL GITLENS: VISUAL FILE HISTORY COMMENTS

ng tool: C:\Program Files (x86)\Go\bin\go.exe test -timeout 30s -run ^TestUser_Validate\$ github.com/nd

```
run test | debug test
10 func TestUser_Validate(t *testing.T) {
11     testCases := []struct {
12         name    string
13         u        func() *model.User
14         isValid bool
15     }{
16     {
17         name: "valid",
18         u: func() *model.User {
19             return model.TestUser(t)
20         },
21         isValid: true,
22     },
23     {
24         name: "empty email",
25         u: func() *model.User {
26             u := model.TestUser(t)
27             u.Email = ""
28             return u
29         },
30     },
31 }
```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL GITLENS: VISUAL FILE HISTORY COMMENTS

Running tool: C:\Program Files (x86)\Go\bin\go.exe test -timeout 30s -run ^TestUser_Validate\$ github.com/ndy-corp/1.src/midterm-1/src-code/internal/app/model

ok github.com/ndy-corp/1.src/midterm-1/src-code/internal/app/model (cached)

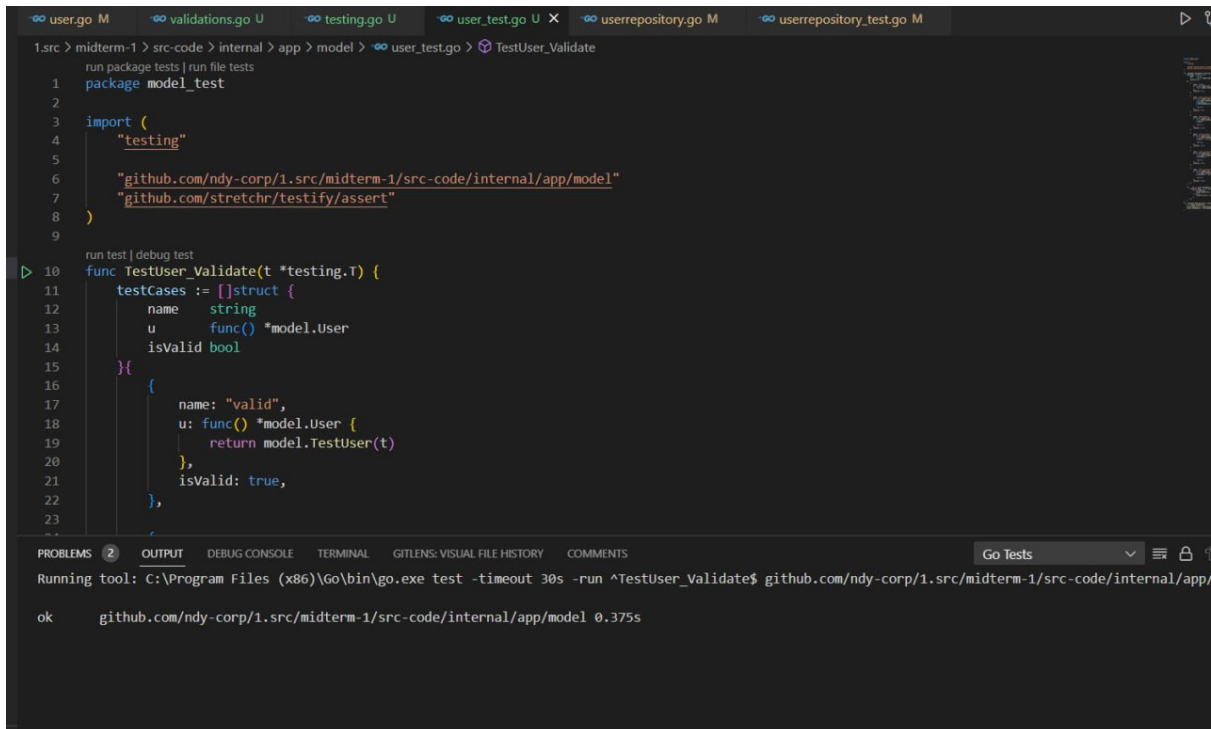
We're running a func "TestUser_Validate"

```
15 }
16
17 // Validate ...
18 func (u *User) Validate() error {
19     return validation.ValidateStruct(
20         u,
21         validation.Field(&u.Email, validation.Required, is.Email),
22         validation.Field(&u.Password, validation.By(requiredIf(u.EncryptedPassword == "")), validation.Length(6, 100)),
23     )
24 }
25
26 // BeforeCreate ...
27 func (u *User) BeforeCreate() error {
28     if len(u.Password) > 0 {
29         enc, err := encryptString(u.Password)
30         if err != nil {
31             return err
32         }
33         u.EncryptedPassword = enc
34     }
35     return nil
36 }
37 }
```

мы тут дали функцию пустого пароля

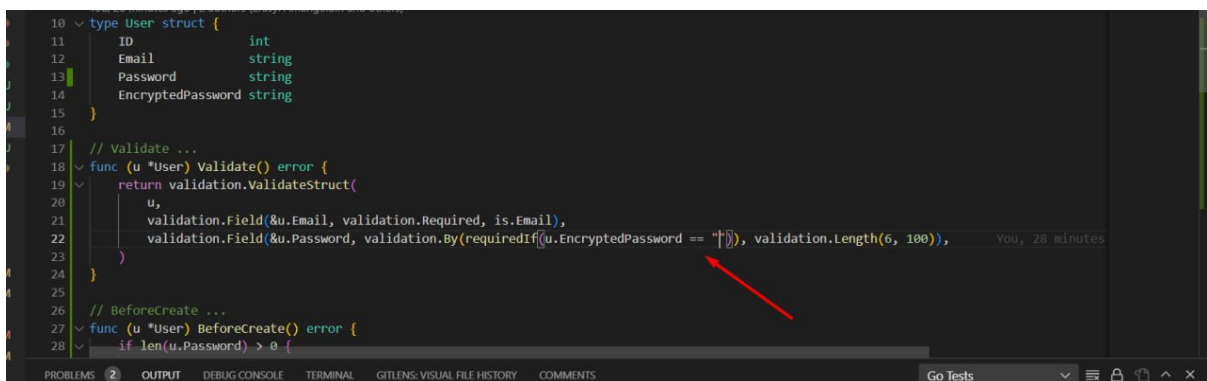
```
1.src > midterm-1 > src-code > internal > app > model > testing.go > TestUser
1  package model
2
3  import "testing"
4
5  // TestUser ...
6  func TestUser(t *testing.T) *User {
7      return &User{
8          Email: "user@example.org",
9          Password: "password",
10     }
11 }
12
```

We have created a method here that is called a **TestUser**, it will take that is, facts and return us a pointer to the user, it is just necessary in order not to write in tests every time to initialize a transaction from scratch there is an email such and such a password, we will just use this helper that will already return us a user with a valid a set of data.



```
1.src > midterm-1 > src-code > internal > app > model > user_test.go > TestUser_Validate
run package tests | run file tests
1 package model_test
2
3 import (
4     "testing"
5
6     "github.com/ndy-corp/1.src/midterm-1/src-code/internal/app/model"
7     "github.com/stretchr/testify/assert"
8 )
9
10 run test | debug test
11 func TestUser_Validate(t *testing.T) {
12     testCases := []struct {
13         name      string
14         u          func() *model.User
15         isValid   bool
16     }{
17         {
18             name: "valid",
19             u: func() *model.User {
20                 return model.TestUser(t)
21             },
22             isValid: true,
23         },
24     }
25
26     for _, tc := range testCases {
27         t.Run(tc.name, func(t *testing.T) {
28             u := tc.u()
29             assert.True(t, u.IsValid())
30         })
31     }
32 }
33
34 Running tool: C:\Program Files (x86)\Go\bin\go.exe test -timeout 30s -run ^TestUser_Validate$ github.com/ndy-corp/1.src/midterm-1/src-code/internal/app/
35
36 ok      github.com/ndy-corp/1.src/midterm-1/src-code/internal/app/model 0.375s
```

This week we reviewed the validations and covered all the necessary cases, we also added password encryption so as not to store it in plain text!



```
10 type User struct {
11     ID          int
12     Email       string
13     Password    string
14     EncryptedPassword string
15 }
16
17 // Validate ...
18 func (u *User) Validate() error {
19     return validation.ValidateStruct(
20         u,
21         validation.Field(&u.Email, validation.Required, is.Email),
22         validation.Field(&u.Password, validation.By(requiredIf(u.EncryptedPassword == "")), validation.Length(6, 100)),
23     )
24 }
25
26 // BeforeCreate ...
27 func (u *User) BeforeCreate() error {
28     if len(u.Password) > 0 {
29         u.EncryptedPassword = bcrypt.GenerateFromPassword([]byte(u.Password), bcrypt.DefaultCost)
30     }
31 }
```

Here, as an argument to the RequiredIf function for RequiredIf, we will pass such a check that the EncryptedPassword is equal to an empty string!

```
{
  name: "with encrypt password",
  u: func() *model.User {
    u := model.TestUser(t)
    u.Password = ""
    u.EncryptedPassword = "encryptedpassword"

    return u
  },
  isValid: true,
},
```

We have created a case here that password can be empty, but by default it will be filled in, that is, it will have some value EncryptedPassword!