Continuous Delivery

Assignment 02

Harald Pinsker (S1910455003)

1 Go REST API

Task: Follow this tutorial https://semaphoreci.com/community/tutorials/building-and-testing-a-rest-api-in-go-with-gorilla-mux-and-postgresql and document the progress.

1.1 Setup

I installed Go and Postgres (incl. the required environment variables) and started the Postgres database server.

1.2 Git

I created a new repo on GitHub: https://github.com/HPin/go-rest-api
Then I cloned it and initialized the Go modules as described in the tutorial with

```
go mod init
```

1.3 Write the code

1.3.1 Test 1: Without any test cases

```
Haralds-MacBook-Air:go-rest-api hp$ go test -v
testing: warning: no tests to run
PASS
ok      github.com/HPin/go-rest-api     0.649s
```

1.3.2 Add test case for empty tables

1.3.3 Test 2: TestEmptyTable

```
Haralds-MacBook-Air:go-rest-api hp$ go test -v
=== RUN   TestEmptyTable
    TestEmptyTable: main_test.go:47: Expected response code 200. Got 404
    TestEmptyTable: main_test.go:68: Expected an empty array. Got 404 page not found
--- FAIL: TestEmptyTable (0.01s)
FAIL
exit status 1
FAIL    github.com/HPin/go-rest-api     0.542s
```

1.3.4 Add test cases for all other api operations

1.3.5 Test 3: Test all api operations

```
Haralds-MacBook-Air:go-rest-api hp$ go test -v
=== RUN   TestEmptyTable
    TestEmptyTable: main_test.go:47: Expected response code 200. Got 404
    TestEmptyTable: main_test.go:68: Expected an empty array. Got 404 page not found
--- FAIL: TestEmptyTable (0.01s)
FAIL
exit status 1
FAIL    github.com/HPin/go-rest-api     0.542s
Haralds-MacBook-Air:go-rest-api hp$ go test -v
=== RUN   TestEmptyTable
    TestEmptyTable: main_test.go:50: Expected response code 200. Got 404
    TestEmptyTable: main_test.go:71: Expected an empty array. Got 404 page not found
--- FAIL: TestEmptyTable (0.01s)
=== RUN   TestGetNonExistentProduct
    TestGetNonExistentProduct: main_test.go:86: Expected the 'error' key of the response to be set to 'Product not found'. Got ''
--- FAIL: TestGetNonExistentProduct (0.00s)
=== RUN   TestCreateProduct
    TestCreateProduct: main_test.go:50: Expected response code 201. Got 404
    TestCreateProduct: main_test.go:105: Expected product name to be 'test product'. Got '<nil>'
    TestCreateProduct: main_test.go:109: Expected product price to be '11.22'. Got '<nil>'
    TestCreateProduct: main_test.go:115: Expected product ID to be '1'. Got '<nil>'
--- FAIL: TestCreateProduct (0.00s)
=== RUN   TestGetProduct
    TestGetProduct: main_test.go:50: Expected response code 200. Got 404
--- FAIL: TestGetProduct (0.01s)
=== RUN   TestUpdateProduct
    TestUpdateProduct: main_test.go:50: Expected response code 200. Got 404
    TestUpdateProduct: main_test.go:165: Expected the name to change from '<nil>' to '<nil>'. Got '<nil>'
    TestUpdateProduct: main_test.go:169: Expected the price to change from '<nil>' to '<nil>'. Got '<nil>'
--- FAIL: TestUpdateProduct (0.04s)
=== RUN   TestDeleteProduct
    TestDeleteProduct: main_test.go:50: Expected response code 200. Got 404
    TestDeleteProduct: main_test.go:50: Expected response code 200. Got 404
--- FAIL: TestDeleteProduct (0.01s)
FAIL
exit status 1
FAIL    github.com/HPin/go-rest-api     0.594s
```

1.3.6 Add the code with the actual app functionality

1.3.7 Create routes and route handlers

Note: the tuturial code had one flaw in the Run function of the App struct where the port parameter was not used and the port was hardcoded instead, so I changed this:

```
func (a *App) Run(addr string) {
    log.Fatal(http.ListenAndServe(":8010", a.Router))
}
```

to this:

```
func (a *App) Run(addr string) {
    log.Fatal(http.ListenAndServe(addr, a.Router))
}
```

1.3.8 Test 4: Test of the fully functioning app

```
Haralds-MacBook-Air:go-rest-api hp$ go test -v
=== RUN   TestEmptyTable
--- PASS: TestEmptyTable (0.02s)
=== RUN   TestGetNonExistentProduct
--- PASS: TestGetNonExistentProduct (0.00s)
=== RUN   TestCreateProduct
--- PASS: TestCreateProduct (0.00s)
=== RUN   TestGetProduct
--- PASS: TestGetProduct (0.01s)
=== RUN   TestUpdateProduct
--- PASS: TestUpdateProduct (0.01s)
=== RUN   TestDeleteProduct
--- PASS: TestDeleteProduct (0.01s)
PASS
ok      github.com/HPin/go-rest-api     0.527s
```

All tests passed.

1.4 Add Functionality

Task: Add 2-3 small features

I implemented three different new functions/endpoints:

1. Get a random product
2. Get the number of products in the database
3. Get the products in a simple HTML table (<table><tr><th></th></tr></table>)

```
a.Router.HandleFunc("/product/random", a.getRandomProduct).Methods("GET")
a.Router.HandleFunc("/products/number", a.getNumberOfProducts).Methods("GET")
a.Router.HandleFunc("/products/htmltable", a.getHTMLTable).Methods("GET")
```

Test again:

```
Haralds-MacBook-Air:go-rest-api hp$ go test -v
=== RUN   TestEmptyTable
--- PASS: TestEmptyTable (0.01s)
=== RUN   TestGetNonExistentProduct
--- PASS: TestGetNonExistentProduct (0.01s)
=== RUN   TestCreateProduct
--- PASS: TestCreateProduct (0.00s)
=== RUN   TestGetProduct
--- PASS: TestGetProduct (0.01s)
=== RUN   TestUpdateProduct
--- PASS: TestUpdateProduct (0.01s)
=== RUN   TestDeleteProduct
--- PASS: TestDeleteProduct (0.01s)
=== RUN   TestGetRandomProduct
--- PASS: TestGetRandomProduct (0.02s)
=== RUN   TestGetNumberOfProducts
--- PASS: TestGetNumberOfProducts (0.01s)
=== RUN   TestGetHTMLTable
--- PASS: TestGetHTMLTable (0.01s)
PASS
ok      github.com/HPin/go-rest-api     0.677s
```

All tests passed.