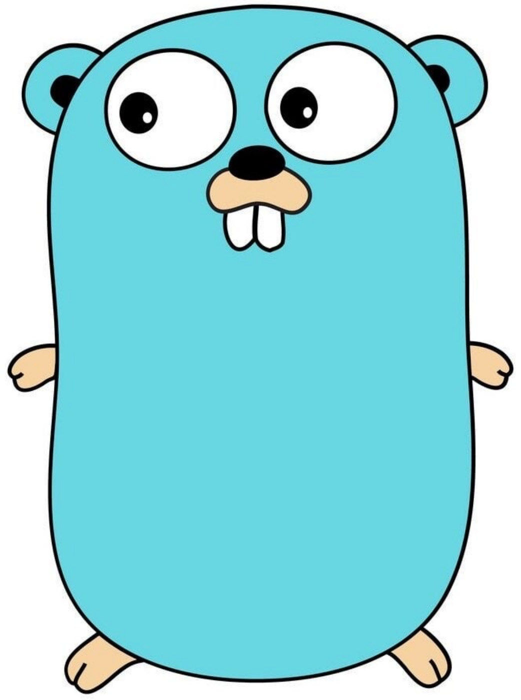
Тестирование

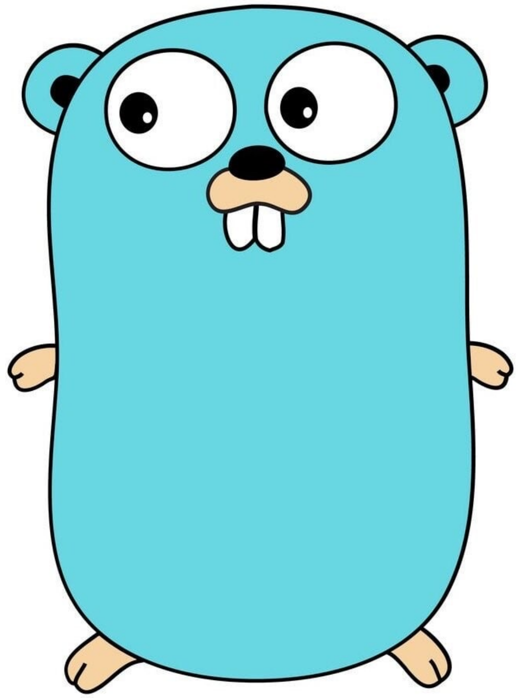
Это процесс проверки ПО на соответствие между реальным и ожидаемым поведением



Зачем

- Описание ожиданий (TDD)
- Проверка соответствия ожиданиям
- Проверка регрессии

TDD – test-driven development



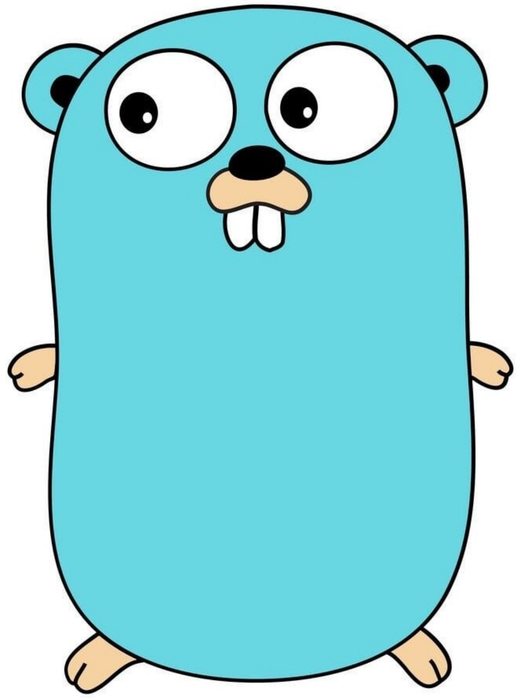
Запуск

Команда запуска:

go test [build/test flags] [packages] [build/test flags & test binary flags]

Документация:

go help test



Флаги

Флаг	Описание
-v	Подробный вывод логов прохождения тестов
-timeout	Ограничение времени тестирования
-cover	Анализ покрытия
-cpu 1, 2, 4	GOMAXPROCS
-failfast	Если один тест упал, другие не выполняются
-parallel	Кол-во параллельно запускаемых тестов, GOMAXPROCS
-bench	Запуск benchmark тестов
-fuzz	Запуск fuzz тестов
-race	Выявление гонки данных

Документация

go help testflag

План

1. Подходы
2. Автоматизация
3. Мокирование
4. Пирамида тестирования

ТИНЬКОФФ

Подходы

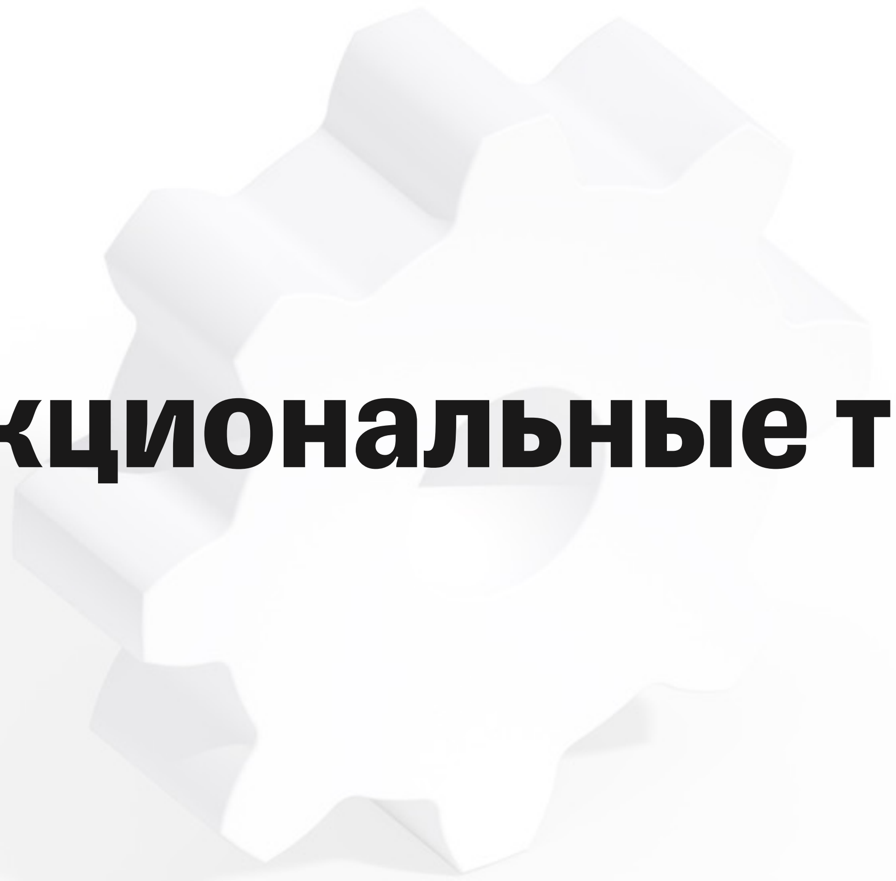
Какие тесты бывают

- Функциональные
  - **Модульные (unit)**
  - **Интеграционные**
  - Приемочные и UI

- Нефункциональные
  - **Производительности**
  - Надежности (отказоустойчивости)
  - Удобство пользования, ...

- Связанные с изменениями
  - Регрессионные

- **Автоматические**
- Ручные



Функциональные тесты

Модульные тесты

Для тестирования отдельных “модулей” кода: **отдельных функций** и их композиции

```go
func Int2Str(val int) string {
    return fmt.Sprint(val)
}

func TestInt2Str() {
    if got := Int2Str(7); got != "7" {
    // AAAaaa!!!
    }
}
```

Модульные тесты

Для тестирования отдельных “модулей” кода: отдельных функций и **их композиции**

```go
func Int2Str(val int) string {
    return fmt.Sprint(val)
}

func Str2Int(val string) (res int) {
    _, _ = fmt.Sscan(val, &res)
    return
}

func TestInt2StrAndStr2Int() {
    const in = 7
    if got := Str2Int(Int2Str(in)); in != got {
    // AAAaaa!!!
    }
}
```

Модульные тесты – из «коробки»

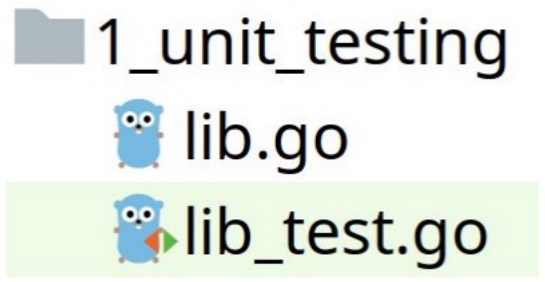
lib.go

```go
package lib

import "fmt"

func Int2Str(val int) string {
    return fmt.Sprint(val)
}
```

lib_test.go

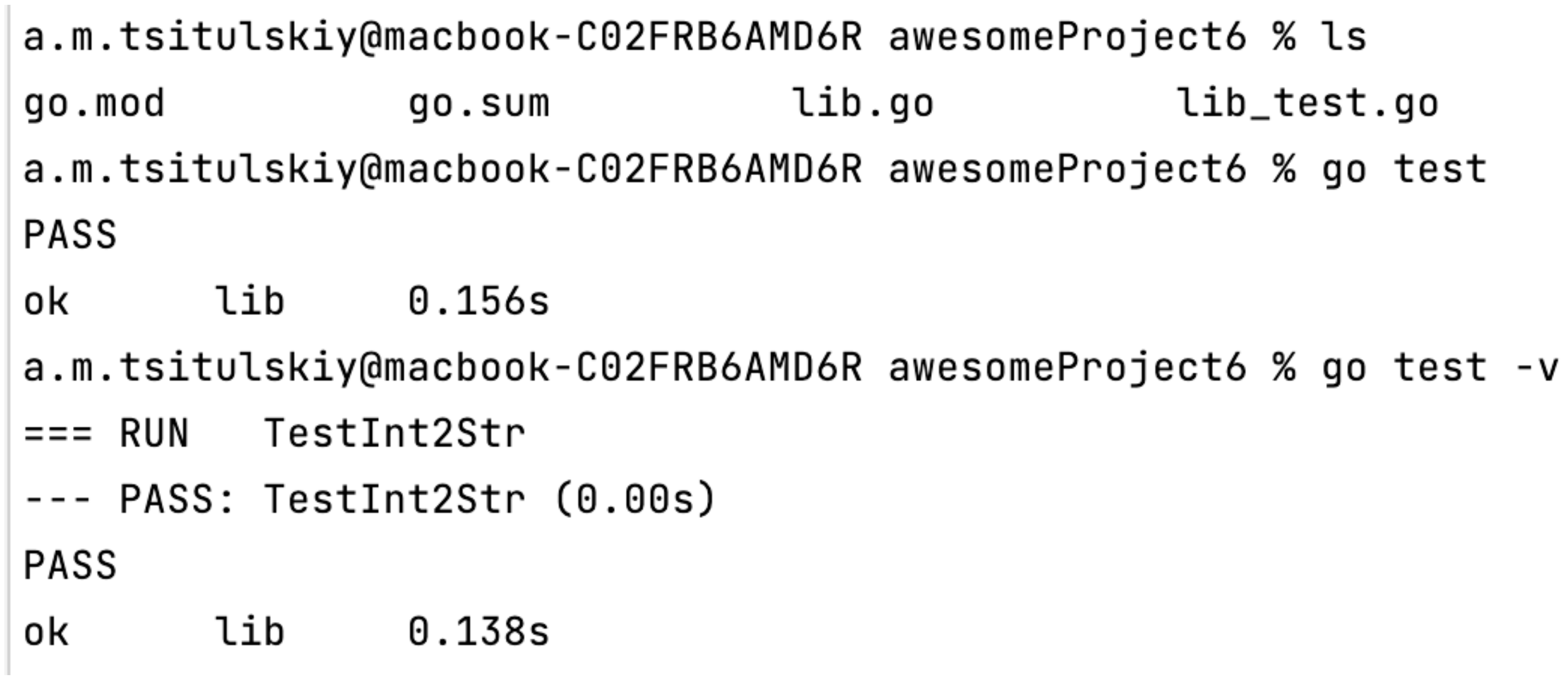
```go
package lib_test

import (
    "lib"
    "testing"
)

func TestInt2Str(t *testing.T) {
    const expect = "7"
    if got := lib.Int2Str(7); got != expect {
        t.Errorf(`Expect %v got %v`, expect, got)
    }
}
```

Модульные тесты – из «коробки»


1_unit_testing
lib.go
lib_test.go

- Общий* пакет
- *_test.go в имени файла
- Test* в именах функций
- import "testing"
- *testing.T в сигнатуре функций

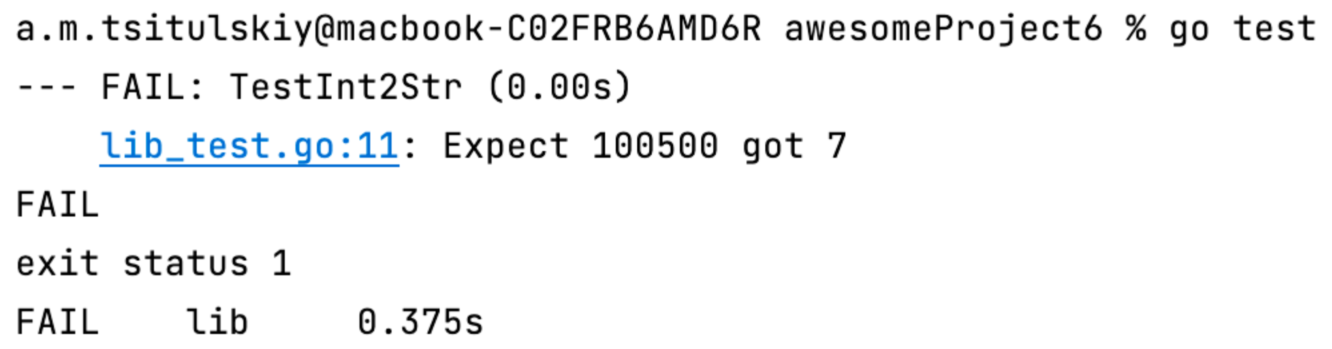
Модульные тесты – из «коробки»


```
a.m.tsitulskiy@macbook-C02FRB6AMD6R awesomeProject6 % ls
go.mod          go.sum          lib.go          lib_test.go
a.m.tsitulskiy@macbook-C02FRB6AMD6R awesomeProject6 % go test
PASS
ok      lib     0.156s
a.m.tsitulskiy@macbook-C02FRB6AMD6R awesomeProject6 % go test -v
=== RUN   TestInt2Str
--- PASS: TestInt2Str (0.00s)
PASS
ok      lib     0.138s
```

Модульные тесты – из «коробки»

```go
package lib_test

import (
    "lib"
    "testing"
)

func TestInt2Str(t *testing.T) {
    const expect = "100500"
    if got := lib.Int2Str(7); got != expect {
        t.Errorf(`Expect %v got %v`, expect, got)
    }
}
```

Модульные тесты – из «коробки»



```
a.m.tsitulskiy@macbook-C02FRB6AMD6R awesomeProject6 % go test
--- FAIL: TestInt2Str (0.00s)
    lib_test.go:11: Expect 100500 got 7
FAIL
exit status 1
FAIL    lib     0.375s
```

Модульные тесты – примеры

```go
import (
    "fmt"
)

func ExampleInt2Str() {
    fmt.Println(Int2Str(7))
    // Output: 7
}
```

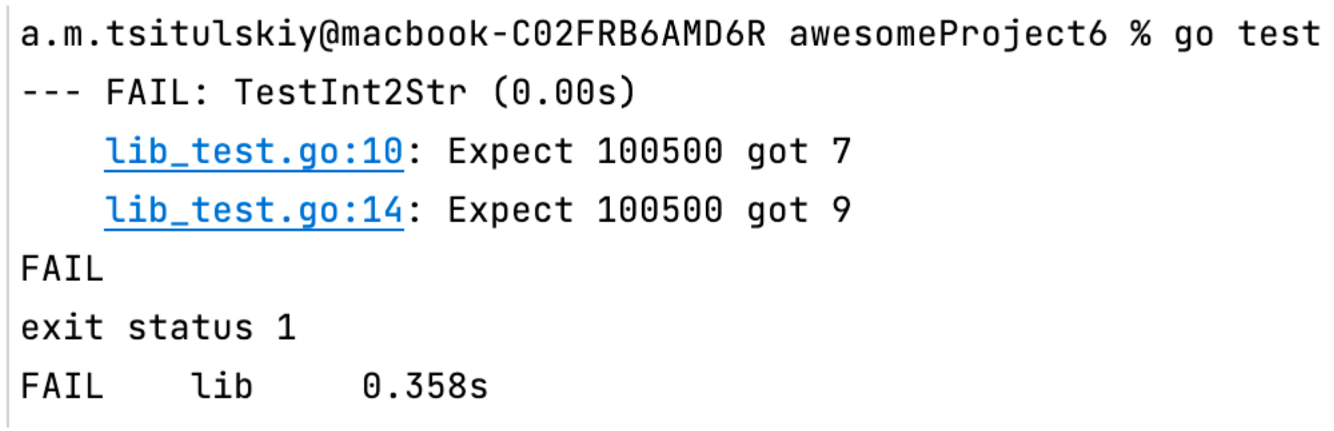
Гарантированно правильные примеры

Например для разработки cli

Модульные тесты – из «коробки»

```
package lib_test

import (
    "lib"
    "testing"
)

func TestInt2Str(t *testing.T) {
    if expect, got := "100500", lib.Int2Str(7); got != expect {
        t.Errorf(` Expect %v got %v ` , expect, got)
    }

    if expect, got := "100500", lib.Int2Str(9); got != expect {
        t.Errorf(` Expect %v got %v ` , expect, got)
    }
}
```

Модульные тесты – из «коробки»


```
a.m.tsitulskiy@macbook-C02FRB6AMD6R awesomeProject6 % go test
--- FAIL: TestInt2Str (0.00s)
    lib_test.go:10: Expect 100500 got 7
    lib_test.go:14: Expect 100500 got 9
FAIL
exit status 1
FAIL    lib     0.358s
```

Модульные тесты – из «коробки»

Методы	Что происходит, кроме вывода сообщения
Log	Вывести сообщение, только если тест упал или с -v
Error	Отметить тест упавшим, но продолжить его
Fatal	Отметить упавшим и прервать его
Skip	Отметить пропущенным и прервать его
panic()	Отметить упавшим, вывести стек

Модульные тесты – из «коробки»

```
func TestParallel_1(t *testing.T) {
    t.Parallel()
    t.Log(`parallel 1:`, t.TempDir())
}

func TestParallel_2(t *testing.T) {
    t.Parallel()
    t.Log(`parallel 2:`, t.TempDir())
}

func TestSubtests(t *testing.T) {
    t.Run(`sub1`, TestParallel_1)
    t.Run(`sub2`, TestParallel_2)
}
```

Модульные тесты – табличные тесты

```
if expect, got := "7", Int2Str(7); got != expect {
    t.Errorf( ` Expect %v got %v ` , expect, got)
}

if expect, got := "0", Int2Str(0); got != expect {
    t.Errorf( ` Expect %v got %v ` , expect, got)
}
```

Модульные тесты – табличные тесты

```go
type Test struct {
    In int
    Expect string
}

tests := [...]Test{
    {7, "7"},
    {0, "0"},
   //...
}

for idx, test := range tests {
    got := Int2Str(test.In)
    if got != test.Expect {
        t.Fatalf(`test %d: expect %v got %v`, idx, test.Expect, got)
    }
}
```

Модульные тесты – табличные тесты

```go
type Test struct {
    Name string
    In int
    Expect string
}

tests := [...]Test{
    {"Non zero", 7, "7"},
    {"Zero", 0, "0"},
    {"Negative", -1, "1"}, // bug!
}

for _, test := range tests {
    got := Int2Str(test.In)
    if got != test.Expect {
        t.Fatalf(`test %q: expect %v got %v`, test.Name, test.Expect, got)
    }
}
```

Модульные тесты – табличные тесты

```
for _, test := range tests {
    t.Run(test.Name, func(t *testing.T) {
      t.Parallel()

      got := Int2Str(test.In)
      if got != test.Expect {
        t.Fatalf(`test %q: expect %v got %v`, test.Name, test.Expect, got)
      }
    })
  }
}
```

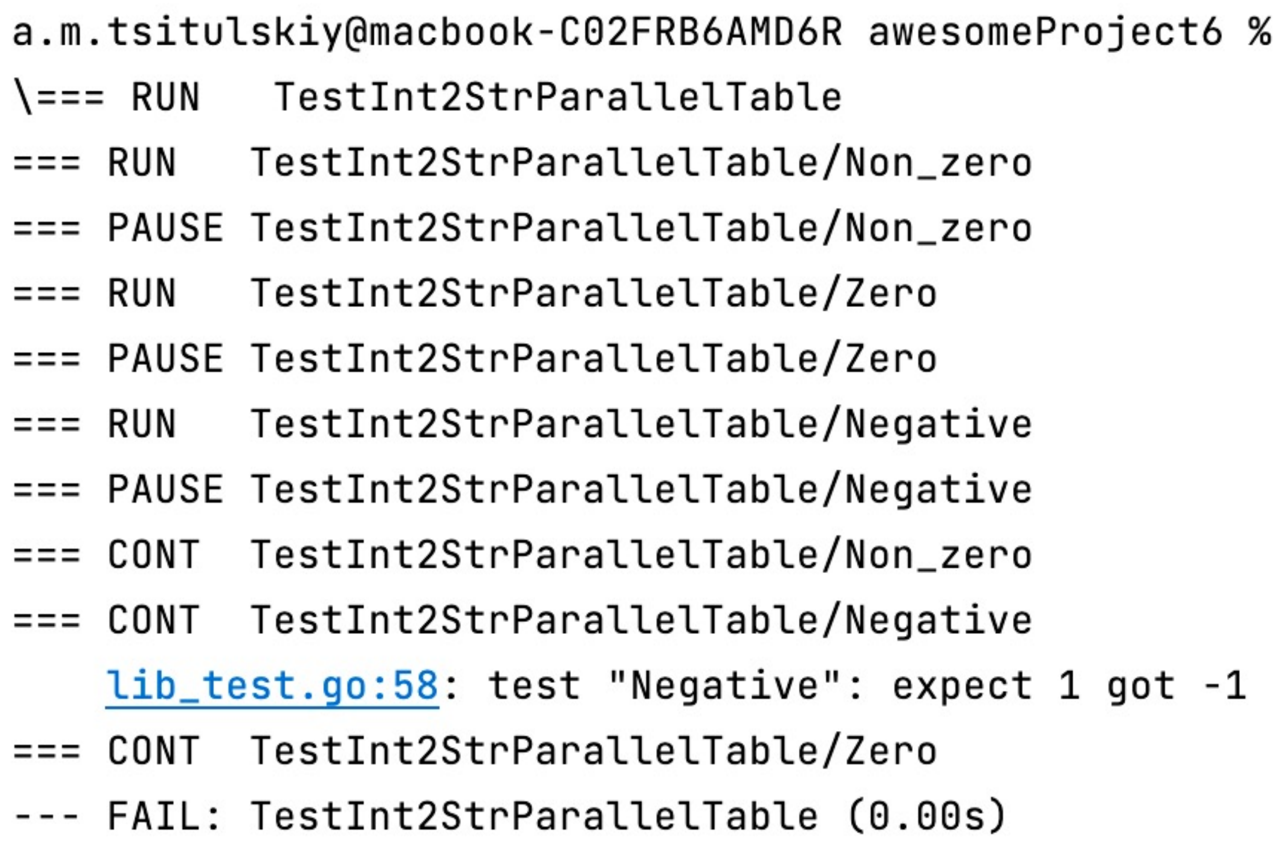
Модульные тесты – табличные тесты

```
=== RUN   TestInt2StrParallelTable/Negative
=== PAUSE TestInt2StrParallelTable/Negative
=== CONT  TestInt2StrParallelTable/Non_zero
    lib_test.go:57: test "Negative": expect 1 got -1
=== CONT  TestInt2StrParallelTable/Zero
=== CONT  TestInt2StrParallelTable/Negative
=== CONT  TestInt2StrParallelTable/Zero
    lib_test.go:57: test "Negative": expect 1 got -1
=== CONT  TestInt2StrParallelTable/Negative
    lib_test.go:57: test "Negative": expect 1 got -1
--- FAIL: TestInt2StrParallelTable (0.00s)
    --- FAIL: TestInt2StrParallelTable/Non_zero (0.00s)
    --- FAIL: TestInt2StrParallelTable/Zero (0.00s)
    --- FAIL: TestInt2StrParallelTable/Negative (0.00s)
FAIL
```

Модульные тесты – табличные тесты

```go
for _, test := range tests {
  test := test

  t.Run(test.Name, func(t *testing.T) {
    t.Parallel()

    got := Int2Str(test.In)
    if got != test.Expect {
      t.Fatalf(`test %q: expect %v got %v`, test.Name, test.Expect, got)
    }
  })
}
```

Модульные тесты – табличные тесты



```
a.m.tsitulskiy@macbook-C02FRB6AMD6R awesomeProject6 % go test -v .
\=== RUN   TestInt2StrParallelTable
=== RUN   TestInt2StrParallelTable/Non_zero
=== PAUSE TestInt2StrParallelTable/Non_zero
=== RUN   TestInt2StrParallelTable/Zero
=== PAUSE TestInt2StrParallelTable/Zero
=== RUN   TestInt2StrParallelTable/Negative
=== PAUSE TestInt2StrParallelTable/Negative
=== CONT  TestInt2StrParallelTable/Non_zero
=== CONT  TestInt2StrParallelTable/Negative
    lib_test.go:58: test "Negative": expect 1 got -1
=== CONT  TestInt2StrParallelTable/Zero
--- FAIL: TestInt2StrParallelTable (0.00s)
```

Модульные тесты – табличные тесты

```
import (
    "reflect"
    "testing"
)

a := map[int]int{1: 2, 4: 2}
b := map[int]int{4: 2, 1: 2}
c := map[int]int{4: 2, 1: 4}

if !reflect.DeepEqual(a, b) {
    t.Fatal("a is not equal to b")
}

if reflect.DeepEqual(a, c) {
    t.Fatal("a is equal to c")
}
```

Модульные тесты – setup & teardown

```go
func TestMain(m *testing.M) {
    fmt.Println("Before all tests")
    code := m.Run()
    fmt.Println("After all tests")
    os.Exit(code)
}
```

Модульные тесты – testify

```go
import (
    "math/rand"
    "testing"
    "github.com/stretchr/testify/assert"
)

func TestInt2Str_Testify(t *testing.T) {
    assert.Equal(t, "7", Int2Str(7))
    assert.Equal(t, "10", Int2Str(0), "zero value")
    assert.ElementsMatch(t, []int{1, 2, 3}, []int{2, 3, 1})
    assert.InDelta(t, 7, 5+rand.Intn(4), 3)
}
```

https://github.com/stretchr/testify

Модульные тесты – setup & teardown

```
type MyFirstTestSuite struct {
    suite.Suite
    VarSome int
}

func (suite *MyFirstTestSuite) SetupTest() {
    suite.VarSome = 5
}

func (suite *MyFirstTestSuite) TestExample() {
    suite.Equal(5, suite.VarSome)
}

func TestExampleTestSuite(t *testing.T) {
    suite.Run(t, new(MyFirstTestSuite))
}
```

Модульные тесты – rapid

```
type Test struct {
    In int
    Expect string
}

tests := [...]Test{
    {7, "7"},
    {0, "0"},
    // ...
}

for idx, test := range tests {
    got := Int2Str(test.In)
    if got != test.Expect {
        t.Fatalf( ` test%d: expect %v got %v ` , idx, test.Expect, got)
    }
}
```

Модульные тесты – rapid

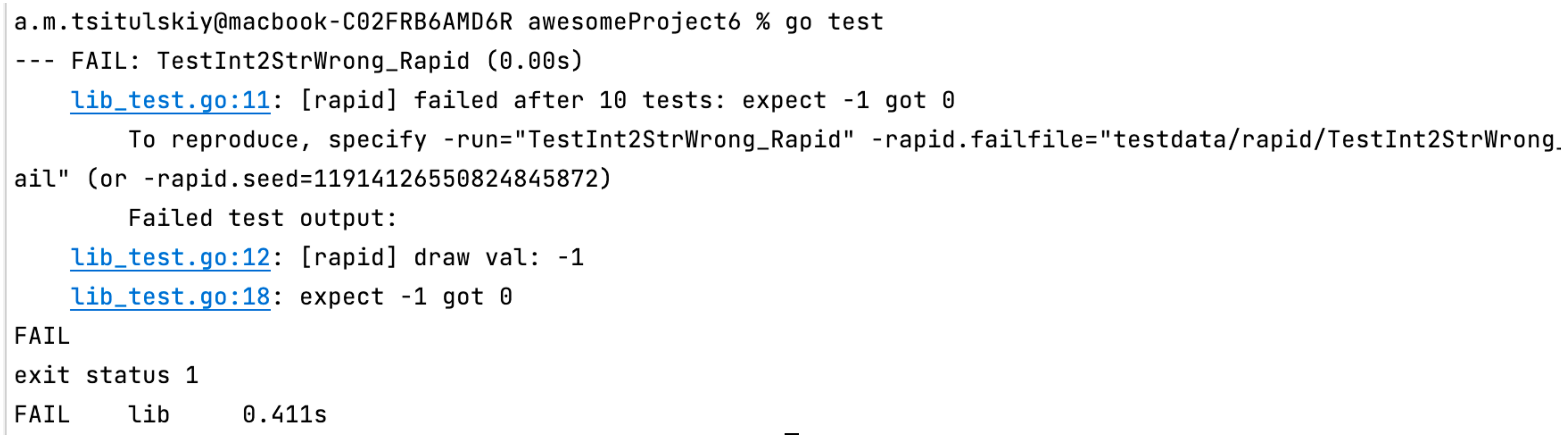
```go
func Int2StrWrong(val int) string {
    if val == -1 || val == math.MaxInt16 {
      return "0"
    }

    return fmt.Sprint(val)
}
```

Модульные тесты – rapid

```
import (
  "fmt"
  "lib"
  "pgregory.net/rapid"
  "testing"
)

func TestInt2StrWrong_Rapid(t *testing.T) {
    rapid.Check(t, func(t *rapid.T) {
        val := rapid.Int32().Draw(t, "val")

        got := lib.Int2StrWrong(int(val))
        expect := fmt.Sprint(val)

        if got != expect {
            t.Fatalf("expect %v got %v", expect, got)
        }
    })
}
```

Автоматическая генерация testcase'ов

Модульные тесты – rapid



```
a.m.tsitulskiy@macbook-C02FRB6AMD6R awesomeProject6 % go test
--- FAIL: TestInt2StrWrong_Rapid (0.00s)
    lib_test.go:11: [rapid] failed after 10 tests: expect -1 got 0
        To reproduce, specify -run="TestInt2StrWrong_Rapid" -rapid.failfile="testdata/rapid/TestInt2StrWrong_
ail" (or -rapid.seed=11914126550824845872)
        Failed test output:
    lib_test.go:12: [rapid] draw val: -1
    lib_test.go:18: expect -1 got 0
FAIL
exit status 1
FAIL    lib     0.411s
```

go test -rapid.checks=1000

Модульные тесты – go-fuzz

Фаззинг — это техника тестирования программного обеспечения, часто автоматическая или полуавтоматическая, заключающаяся в передаче приложению на вход неправильных, неожиданных или случайных данных.

Go 1.18+

Модульные тесты – go-fuzz

```go
func Int2StrWrong(val int) string {
  if val == -1 || val == math.MaxInt16 {
     return `0`
  }

  return fmt.Sprint(val)
}
```

Модульные тесты – go-fuzz

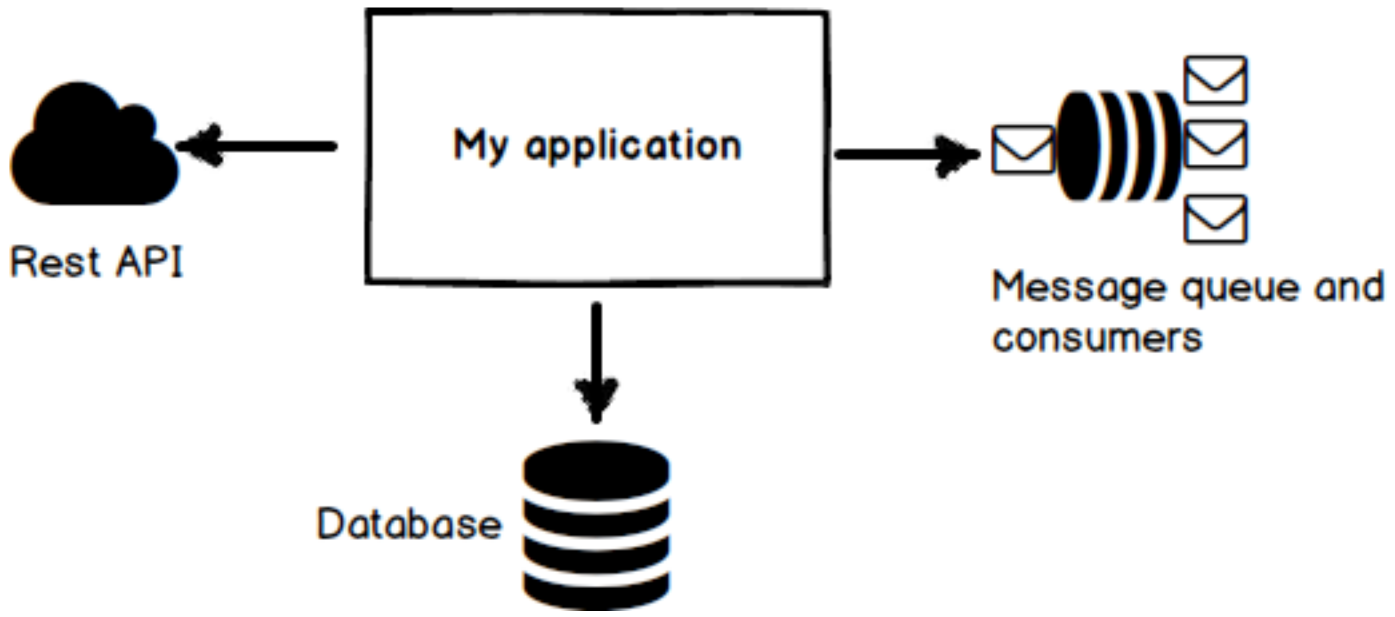
```go
func FuzzInt2StrWrong_Fuzz(f *testing.F) {
  testcases := []int{90, 1000}

  for _, tc := range testcases {
    f.Add(tc)
  }

  f.Fuzz(func(t *testing.T, s int) {
    got := Int2StrWrong(s)
    expect := fmt.Sprintf("%d", s)

    if got != expect {
      t.Errorf("For (%d) Expect: %s, but got: %s", s, expect, got)
    }
  })
}
```

Модульные тесты – go-fuzz

```
a.m.tsitulskiy@macbook-C02FRB6AMD6R awesomeProject6 % go test -v -fuzz .
=== FUZZ  FuzzInt2StrWrong_Fuzz
fuzz: elapsed: 0s, gathering baseline coverage: 0/2 completed
fuzz: elapsed: 0s, gathering baseline coverage: 2/2 completed, now fuzzing with 12 workers
fuzz: elapsed: 0s, execs: 81 (692/sec), new interesting: 1 (total: 3)
--- FAIL: FuzzInt2StrWrong_Fuzz (0.12s)
    --- FAIL: FuzzInt2StrWrong_Fuzz (0.00s)
        lib_test.go:34: For (-1) Expect: -1, but got: 0

    Failing input written to testdata/fuzz/FuzzInt2StrWrong_Fuzz/f7e676cb066ab312f957210335a21e05724a292a231be84518e34e1f3b6699c7
    To re-run:
    go test -run=FuzzInt2StrWrong_Fuzz/f7e676cb066ab312f957210335a21e05724a292a231be84518e34e1f3b6699c7
FAIL
exit status 1
FAIL    lib     0.303s
```

Интеграционные тесты

Для тестирования взаимодействия модулей и сервисов



Интеграционные тесты

lib.go

```
func HTTPReq(addr string) (string, error) {
  var body []byte

  resp, err := http.DefaultClient.Get(addr)
  if err != nil {
    return "", err
  }

  defer func() { _ = resp.Body.Close() }()

  _, err = resp.Body.Read(body)
  if err != nil {
    return "", err
  }

  return string(body), nil
}
```

Интеграционные тесты

lib_test.go

```go
type server struct{}

func (s *server) ServeHTTP(resp http.ResponseWriter, req *http.Request) {
    fmt.Printf("HTTP handler: %q\n", req.RequestURI)
    _, _ = resp.Write([]byte(req.RequestURI))
}
```

Интеграционные тесты

lib_test.go

```
func setup(ipAddr string, t *testing.T) (int, func() error) {
    ipAddr += ":0"
    server := &http.Server{Addr: ipAddr, Handler: &server{}}

    ln, err := net.Listen("tcp", ipAddr)
    if err != nil {
        t.Fatalf("Could not listen port: %s", err)
    }

    go server.Serve(ln)

    port := ln.Addr().(*net.TCPAddr).Port

    return port, server.Close
}
```

Интеграционные тесты

lib_test.go

```
func TestHttpReq(t *testing.T) {
    const ipAddr = "127.0.0.1"

    port, closer := setup(ipAddr, t)
    defer closer()

    addrWithPort := net.JoinHostPort(ipAddr, strconv.Itoa(port))

    const expect = "/hello_world"

    got, _ := HttpReq("http://" + addrWithPort + expect)
    if got != expect {
        t.Fatalf("Expect %v got %v", expect, got)
    }
}
```

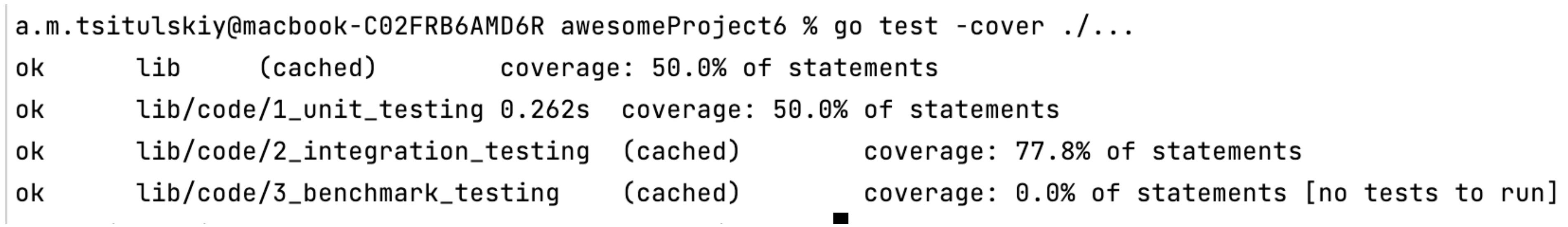
Интеграционные тесты

Интеграционные тесты

lib_test.go

```
func TestHttpReq(t *testing.T) {
    server := httptest.NewServer(http.HandlerFunc(func(resp http.ResponseWriter, req *http.Request) {
      fmt.Printf("HTTP handler: %q\n", req.RequestURI)
      _, _ = resp.Write([]byte(req.RequestURI))
    }))

    defer func() { server.Close() }()

    const expect = "/hello_world”

    got, err := lib.HttpReq(server.URL + expect)

    assert.NoError(t, err)
    assert.Equal(t, expect, got)
}
```

Покрытие тестами



```
a.m.tsitulskiy@macbook-C02FRB6AMD6R awesomeProject6 % go test -cover ./...
ok      lib     (cached)        coverage: 50.0% of statements
ok      lib/code/1_unit_testing 0.262s  coverage: 50.0% of statements
ok      lib/code/2_integration_testing  (cached)        coverage: 77.8% of statements
ok      lib/code/3_benchmark_testing    (cached)        coverage: 0.0% of statements [no tests to run]
```

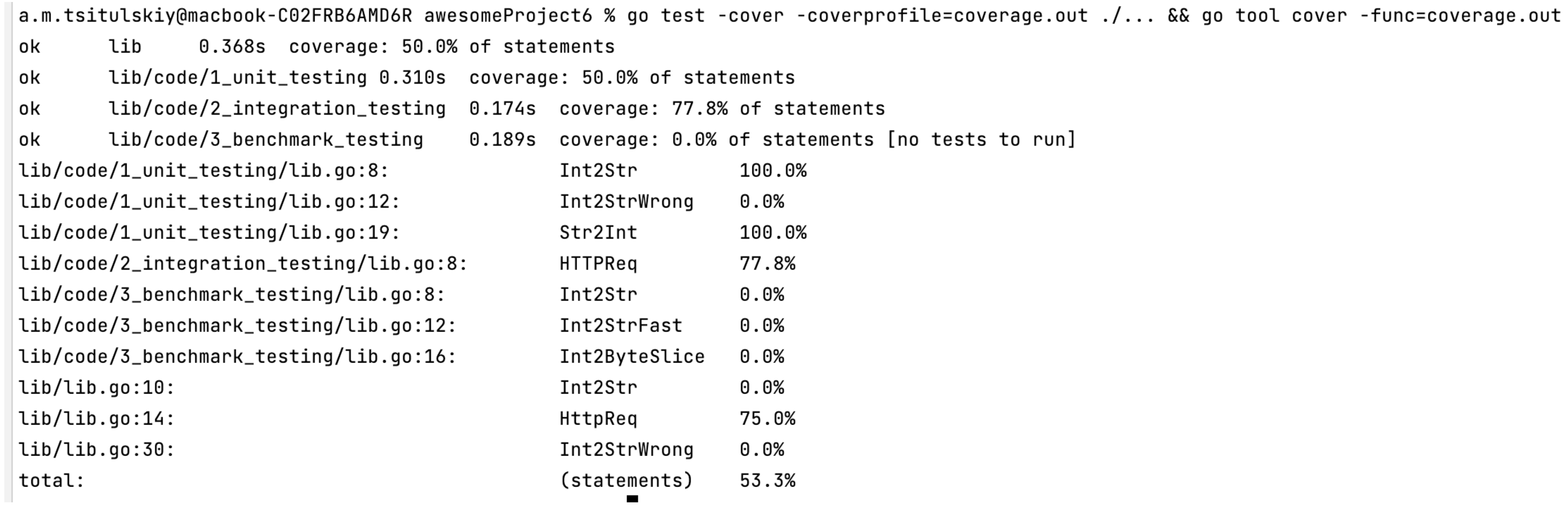
Покрытие тестами

```go
func Int2StrWrong(val int) string {
  if val == -1 || val == math.MaxInt16 {
    return `0`
  }

  return fmt.Sprint(val)
}
```

→

```go
func Int2StrWrong(val int) string {
  GoCover.Count[1] = 1

  if val == -1 || val == math.MaxInt16 {
    GoCover.Count[2] = 1
    return `0`
  }

  GoCover.Count[3] = 1
  return fmt.Sprint(val)
}
```

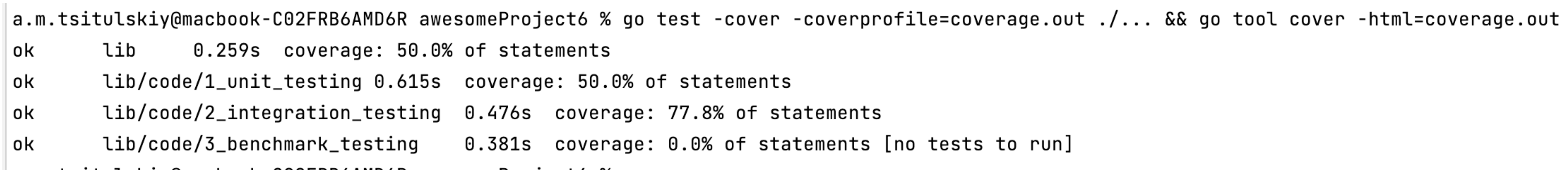
Покрытие тестами



```
a.m.tsitulskiy@macbook-C02FRB6AMD6R awesomeProject6 % go test -cover -coverprofile=coverage.out ./... && go tool cover -func=coverage.out
ok      lib     0.368s  coverage: 50.0% of statements
ok      lib/code/1_unit_testing 0.310s  coverage: 50.0% of statements
ok      lib/code/2_integration_testing  0.174s  coverage: 77.8% of statements
ok      lib/code/3_benchmark_testing    0.189s  coverage: 0.0% of statements [no tests to run]
lib/code/1_unit_testing/lib.go:8:               Int2Str         100.0%
lib/code/1_unit_testing/lib.go:12:              Int2StrWrong    0.0%
lib/code/1_unit_testing/lib.go:19:              Str2Int         100.0%
lib/code/2_integration_testing/lib.go:8:        HTTPReq         77.8%
lib/code/3_benchmark_testing/lib.go:8:          Int2Str         0.0%
lib/code/3_benchmark_testing/lib.go:12:         Int2StrFast     0.0%
lib/code/3_benchmark_testing/lib.go:16:         Int2ByteSlice   0.0%
lib/lib.go:10:                                  Int2Str         0.0%
lib/lib.go:14:                                  HttpReq         75.0%
lib/lib.go:30:                                  Int2StrWrong    0.0%
total:                                          (statements)    53.3%
```

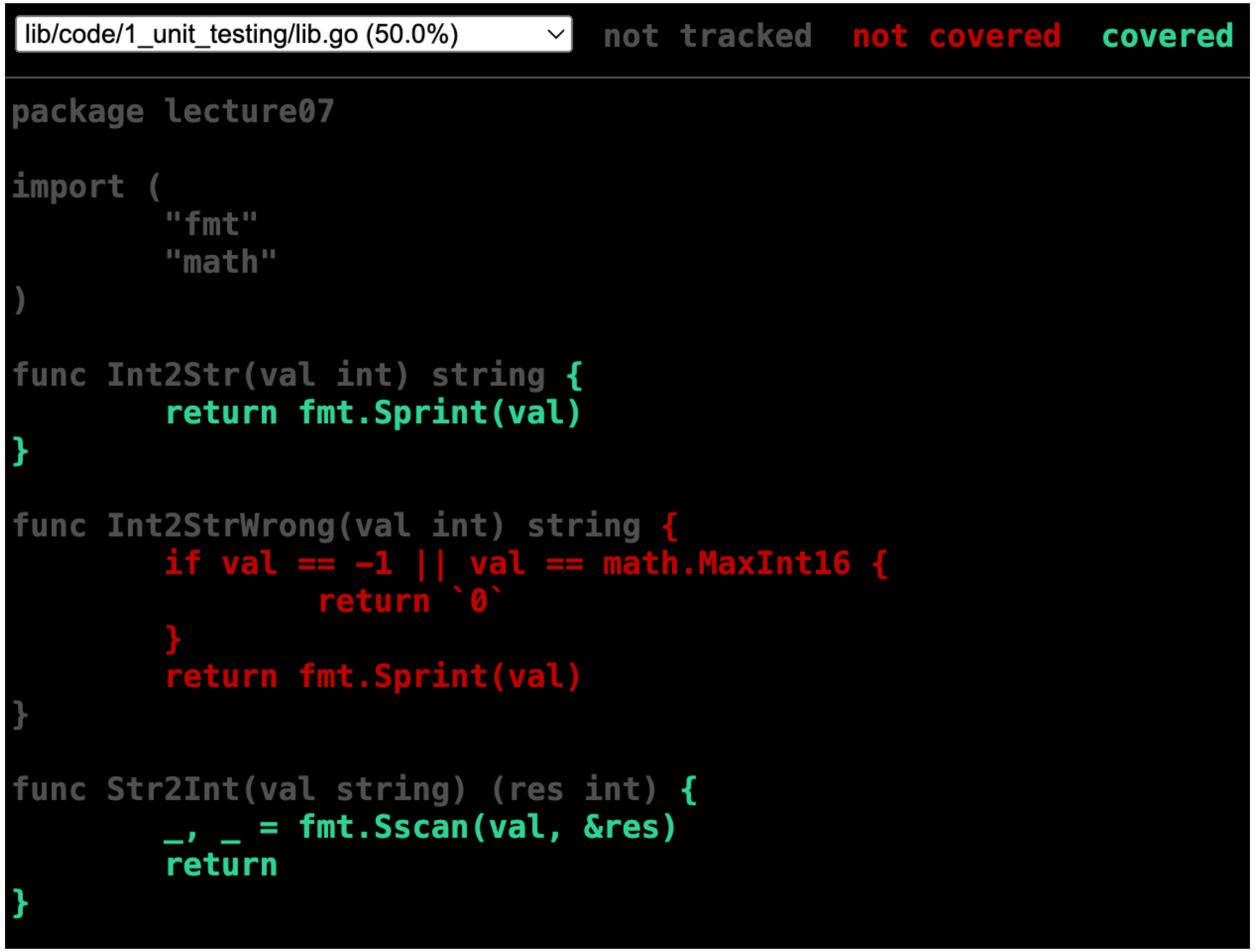
Покрытие тестами

Вместо –func используем -html



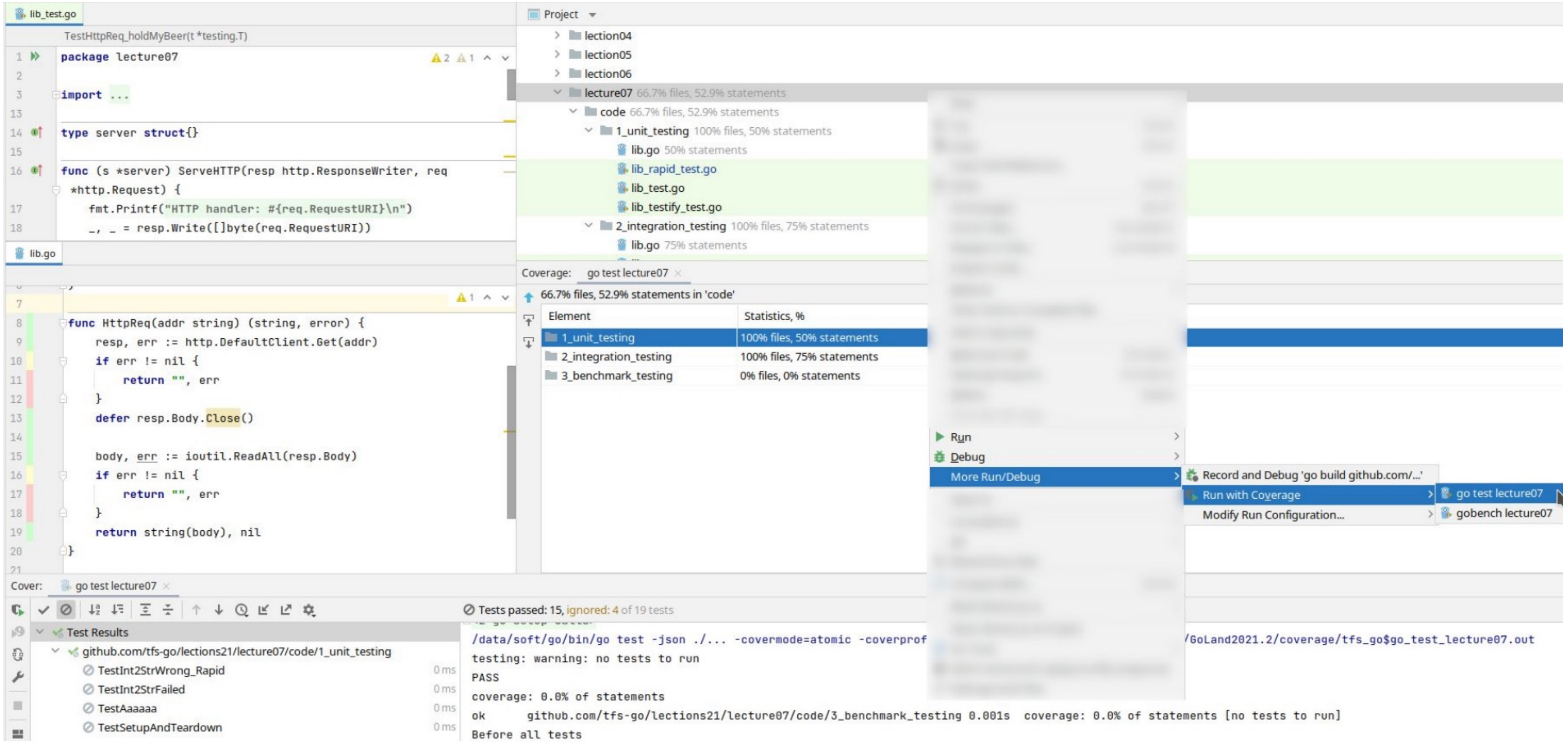
```
a.m.tsitulskiy@macbook-C02FRB6AMD6R awesomeProject6 % go test -cover -coverprofile=coverage.out ./... && go tool cover -html=coverage.out
ok      lib     0.259s  coverage: 50.0% of statements
ok      lib/code/1_unit_testing 0.615s  coverage: 50.0% of statements
ok      lib/code/2_integration_testing  0.476s  coverage: 77.8% of statements
ok      lib/code/3_benchmark_testing    0.381s  coverage: 0.0% of statements [no tests to run]
```

Покрытие тестами



```
package lecture07

import (
        "fmt"
        "math"
)

func Int2Str(val int) string {
        return fmt.Sprint(val)
}

func Int2StrWrong(val int) string {
        if val == -1 || val == math.MaxInt16 {
                return `0`
        }
        return fmt.Sprint(val)
}

func Str2Int(val string) (res int) {
        _, _ = fmt.Sscan(val, &res)
        return
}
```

Покрытие тестами



lib_test.go

TestHttpReq_holdMyBeer(t *testing.T)

```go
package lecture07

import ...

type server struct{}

func (s *server) ServeHTTP(resp http.ResponseWriter, req
 *http.Request) {
    fmt.Printf("HTTP handler: #{req.RequestURI}\n")
    _, _ = resp.Write([]byte(req.RequestURI))
```

lib.go

```go
func HttpReq(addr string) (string, error) {
    resp, err := http.DefaultClient.Get(addr)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }
    return string(body), nil
}
```

```
Project
  lection04
  lection05
  lection06
  lecture07 66.7% files, 52.9% statements
    code 66.7% files, 52.9% statements
      1_unit_testing 100% files, 50% statements
        lib.go 50% statements
        lib_rapid_test.go
        lib_test.go
        lib_testify_test.go
      2_integration_testing 100% files, 75% statements
        lib.go 75% statements
```

Coverage: go test lecture07

66.7% files, 52.9% statements in 'code'

Element	Statistics, %
1_unit_testing	100% files, 50% statements
2_integration_testing	100% files, 75% statements
3_benchmark_testing	0% files, 0% statements

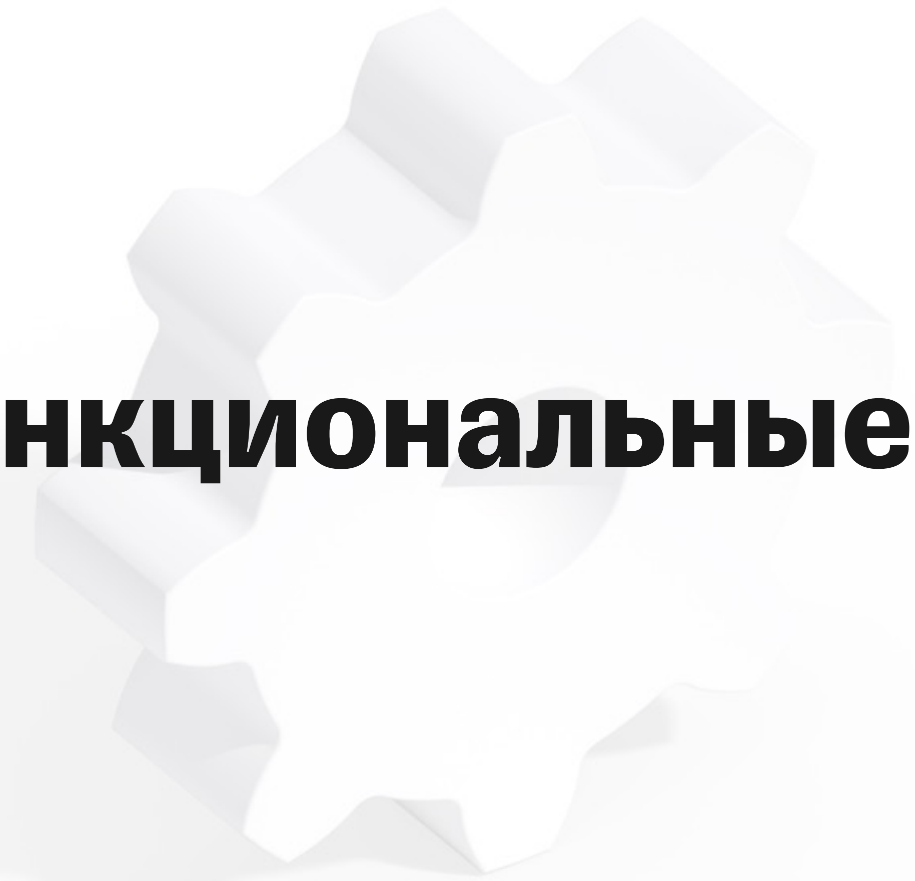
Cover: go test lecture07

Tests passed: 15, ignored: 4 of 19 tests

```
Test Results
  github.com/tfs-go/lections21/lecture07/code/1_unit_testing
    TestInt2StrWrong_Rapid   0 ms
    TestInt2StrFailed        0 ms
    TestAaaaaa               0 ms
    TestSetupAndTeardown     0 ms
```

```
/data/soft/go/bin/go test -json ./... -covermode=atomic -coverprof                    /GoLand2021.2/coverage/tfs_go$go_test_lecture07.out
testing: warning: no tests to run
PASS
coverage: 0.0% of statements
ok      github.com/tfs-go/lections21/lecture07/code/3_benchmark_testing 0.001s  coverage: 0.0% of statements [no tests to run]
Before all tests
```

Нефункциональные тесты



Тесты производительности

lib.go

```go
func Int2Str(val int) string {
    return fmt.Sprint(val)
}

func Int2StrFast(val int) string {
    return strconv.Itoa(val)
}
```

Тесты производительности

lib.go

```go
func Int2Str(val int) string {
    return fmt.Sprint(val)
}

func Int2StrFast(val int) string {
    return strconv.Itoa(val)
}
```

lib_test.go

```go
func BenchmarkInt2Str(b *testing.B) {
  for i := 0; i < b.N; i++ {
    _ = Int2Str(i)
  }
}
```

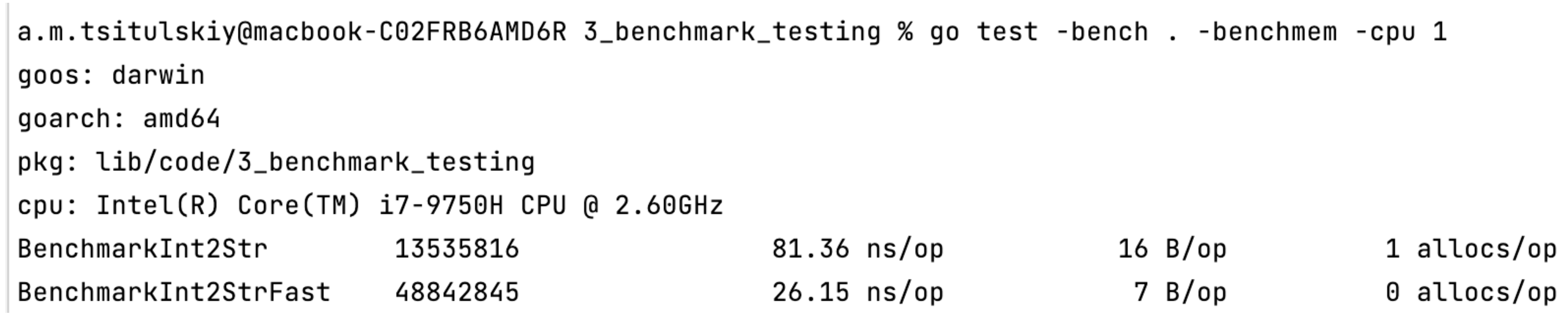
Отличия:

- Benchmark* в именах функций
- *testing.B в сигнатуре функций
- Нужно учитывать b.N

Тесты производительности

```
a.m.tsitulskiy@macbook-C02FRB6AMD6R 3_benchmark_testing % go test -bench . -cpu 1
goos: darwin
goarch: amd64
pkg: lib/code/3_benchmark_testing
cpu: Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz
BenchmarkInt2Str        12212322                83.50 ns/op
BenchmarkInt2StrFast    49165014                26.81 ns/op
```

Тесты производительности



```
a.m.tsitulskiy@macbook-C02FRB6AMD6R 3_benchmark_testing % go test -bench . -benchmem -cpu 1
goos: darwin
goarch: amd64
pkg: lib/code/3_benchmark_testing
cpu: Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz
BenchmarkInt2Str        13535816                81.36 ns/op           16 B/op          1 allocs/op
BenchmarkInt2StrFast    48842845                26.15 ns/op            7 B/op          0 allocs/op
```

Тесты производительности

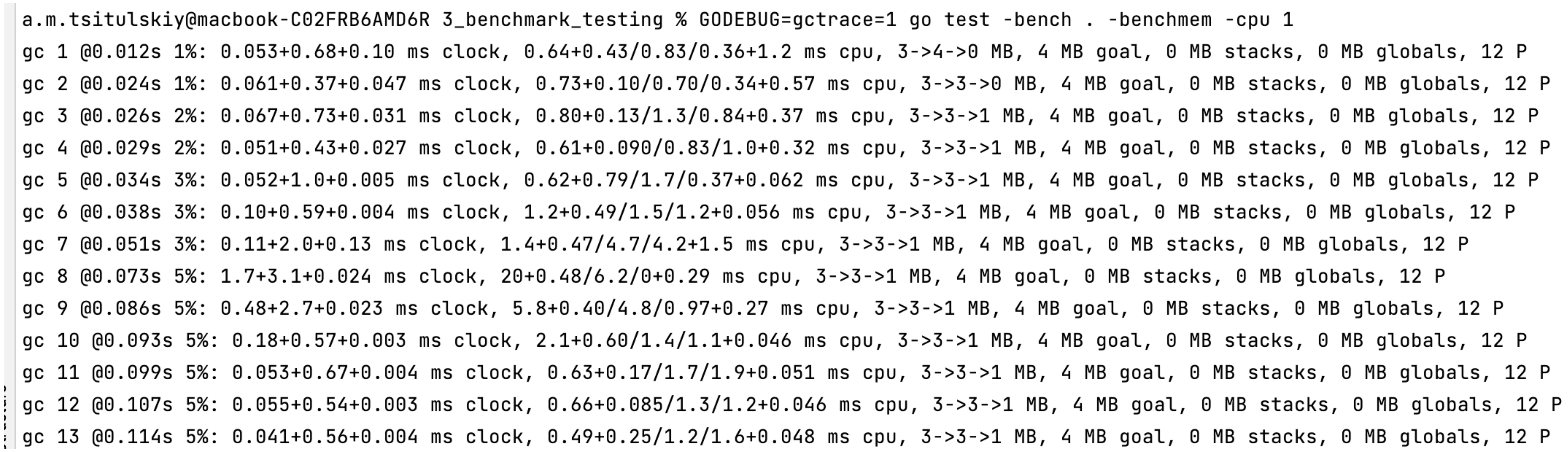
lib.go

```go
func Int2Str(val int) string {
    return fmt.Sprint(val)
}

func Int2StrFast(val int) string {
    return strconv.Itoa(val)
}

func Int2ByteSlice(val int, dst []byte) []byte {
    return strconv.AppendInt(dst, int64(val), 10)
}
```

Тесты производительности

```
a.m.tsitulskiy@macbook-C02FRB6AMD6R 3_benchmark_testing % go test -bench . -benchmem -cpu 1
goos: darwin
goarch: amd64
pkg: lib/code/3_benchmark_testing
cpu: Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz
BenchmarkInt2Str        13118210                82.58 ns/op          16 B/op          1 allocs/op
BenchmarkInt2StrFast    47741358                27.62 ns/op           7 B/op          0 allocs/op
BenchmarkInt2ByteSlice  88969916                15.94 ns/op           0 B/op          0 allocs/op
PASS
ok      lib/code/3_benchmark_testing    4.224s
```

Тесты производительности



```
a.m.tsitulskiy@macbook-C02FRB6AMD6R 3_benchmark_testing % GODEBUG=gctrace=1 go test -bench . -benchmem -cpu 1
gc 1 @0.012s 1%: 0.053+0.68+0.10 ms clock, 0.64+0.43/0.83/0.36+1.2 ms cpu, 3->4->0 MB, 4 MB goal, 0 MB stacks, 0 MB globals, 12 P
gc 2 @0.024s 1%: 0.061+0.37+0.047 ms clock, 0.73+0.10/0.70/0.34+0.57 ms cpu, 3->3->0 MB, 4 MB goal, 0 MB stacks, 0 MB globals, 12 P
gc 3 @0.026s 2%: 0.067+0.73+0.031 ms clock, 0.80+0.13/1.3/0.84+0.37 ms cpu, 3->3->1 MB, 4 MB goal, 0 MB stacks, 0 MB globals, 12 P
gc 4 @0.029s 2%: 0.051+0.43+0.027 ms clock, 0.61+0.090/0.83/1.0+0.32 ms cpu, 3->3->1 MB, 4 MB goal, 0 MB stacks, 0 MB globals, 12 P
gc 5 @0.034s 3%: 0.052+1.0+0.005 ms clock, 0.62+0.79/1.7/0.37+0.062 ms cpu, 3->3->1 MB, 4 MB goal, 0 MB stacks, 0 MB globals, 12 P
gc 6 @0.038s 3%: 0.10+0.59+0.004 ms clock, 1.2+0.49/1.5/1.2+0.056 ms cpu, 3->3->1 MB, 4 MB goal, 0 MB stacks, 0 MB globals, 12 P
gc 7 @0.051s 3%: 0.11+2.0+0.13 ms clock, 1.4+0.47/4.7/4.2+1.5 ms cpu, 3->3->1 MB, 4 MB goal, 0 MB stacks, 0 MB globals, 12 P
gc 8 @0.073s 5%: 1.7+3.1+0.024 ms clock, 20+0.48/6.2/0+0.29 ms cpu, 3->3->1 MB, 4 MB goal, 0 MB stacks, 0 MB globals, 12 P
gc 9 @0.086s 5%: 0.48+2.7+0.023 ms clock, 5.8+0.40/4.8/0.97+0.27 ms cpu, 3->3->1 MB, 4 MB goal, 0 MB stacks, 0 MB globals, 12 P
gc 10 @0.093s 5%: 0.18+0.57+0.003 ms clock, 2.1+0.60/1.4/1.1+0.046 ms cpu, 3->3->1 MB, 4 MB goal, 0 MB stacks, 0 MB globals, 12 P
gc 11 @0.099s 5%: 0.053+0.67+0.004 ms clock, 0.63+0.17/1.7/1.9+0.051 ms cpu, 3->3->1 MB, 4 MB goal, 0 MB stacks, 0 MB globals, 12 P
gc 12 @0.107s 5%: 0.055+0.54+0.003 ms clock, 0.66+0.085/1.3/1.2+0.046 ms cpu, 3->3->1 MB, 4 MB goal, 0 MB stacks, 0 MB globals, 12 P
gc 13 @0.114s 5%: 0.041+0.56+0.004 ms clock, 0.49+0.25/1.2/1.6+0.048 ms cpu, 3->3->1 MB, 4 MB goal, 0 MB stacks, 0 MB globals, 12 P
```

https://www.ardanlabs.com/blog/2019/05/garbage-collection-in-go-part2-gctraces.html

Тесты производительности

https://dave.cheney.net/2013/06/30/how-to-write-benchmarks-in-go

Профилирование

“Профилирование и оптимизация программ на Go”

https://habr.com/ru/company/badoo/blog/301990/

ТИНЬКОФФ

Автоматизация

Автоматизация

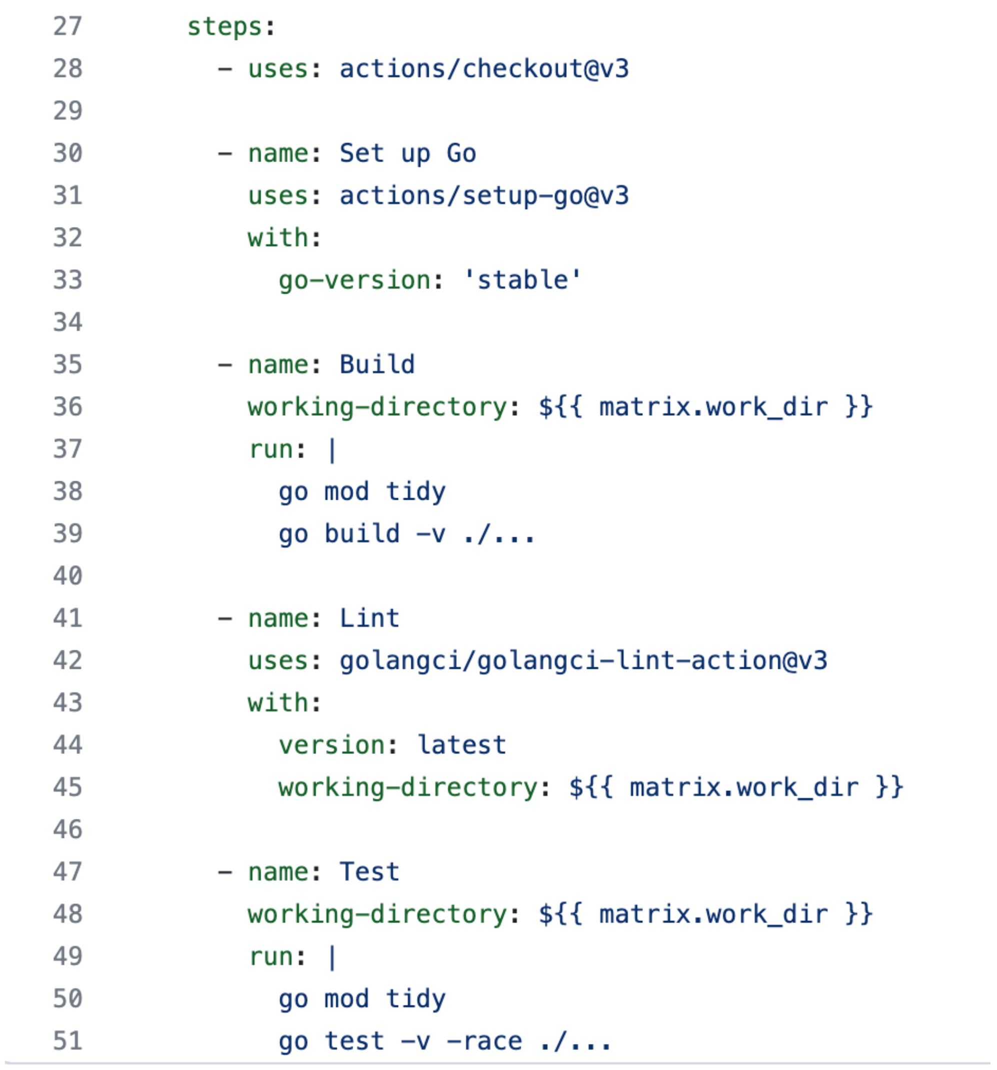
push кода в репозиторий ⇒ запуск автоматических проверок

- Статический анализ кода
- Тесты и покрытие
- Проверка сборки

Автоматизация – Github Actions

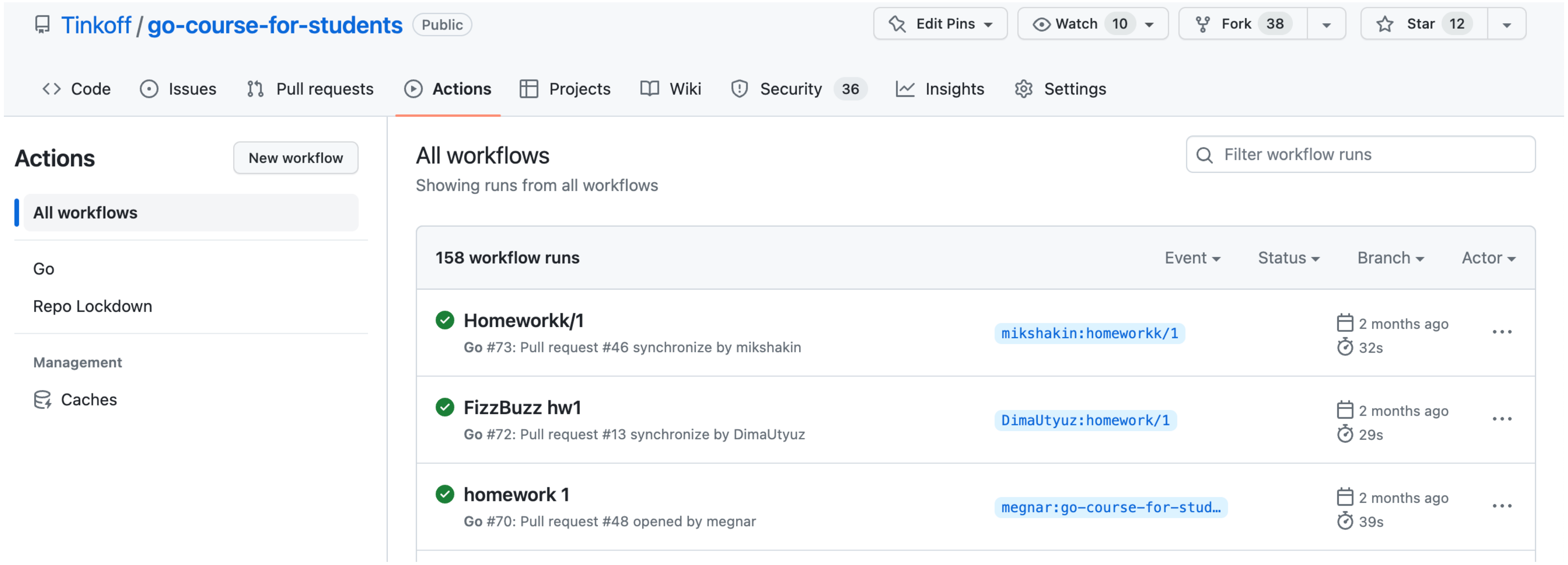
https://github.com/Tinkoff/go-course-for-students/blob/main/.github/workflows/go.yaml

Автоматизация – Github Actions



```yaml
    steps:
      - uses: actions/checkout@v3

      - name: Set up Go
        uses: actions/setup-go@v3
        with:
          go-version: 'stable'

      - name: Build
        working-directory: ${{ matrix.work_dir }}
        run: |
          go mod tidy
          go build -v ./...

      - name: Lint
        uses: golangci/golangci-lint-action@v3
        with:
          version: latest
          working-directory: ${{ matrix.work_dir }}

      - name: Test
        working-directory: ${{ matrix.work_dir }}
        run: |
          go mod tidy
          go test -v -race ./...
```

Автоматизация – Github Actions



All workflows

Showing runs from all workflows

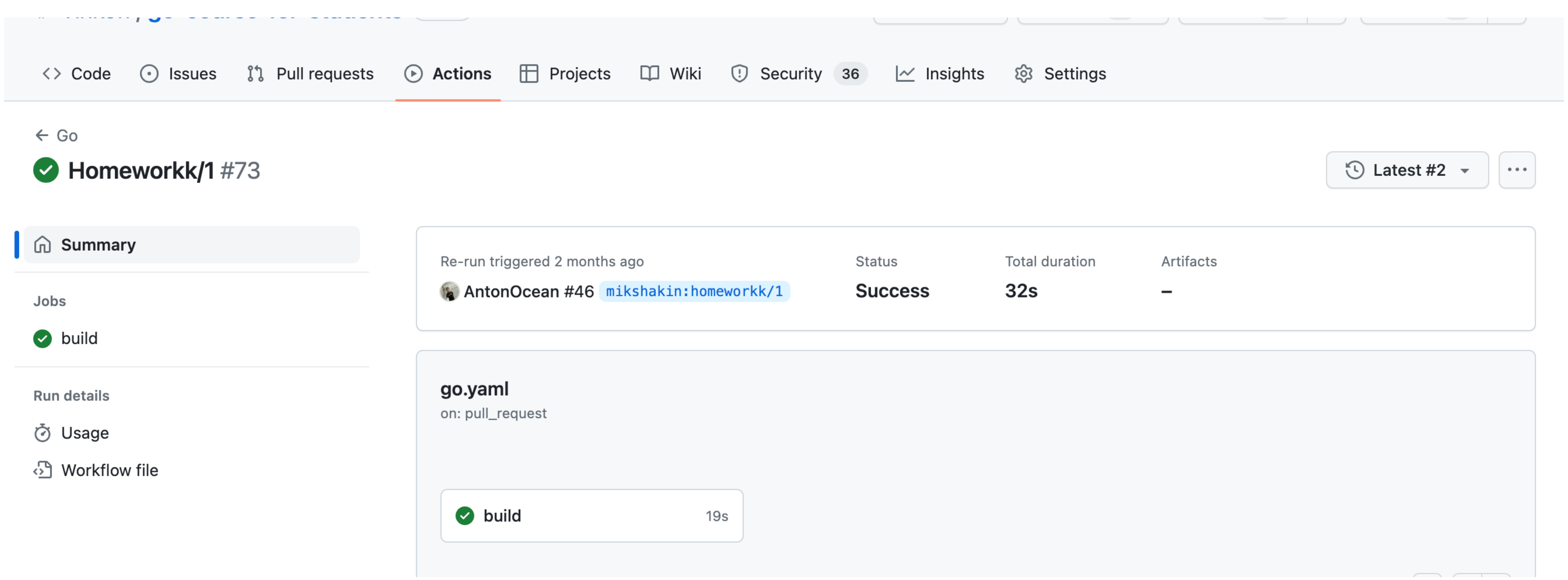
158 workflow runs

Homeworkk/1
Go #73: Pull request #46 synchronize by mikshakin
`mikshakin:homeworkk/1`
2 months ago
32s

FizzBuzz hw1
Go #72: Pull request #13 synchronize by DimaUtyuz
`DimaUtyuz:homework/1`
2 months ago
29s

homework 1
Go #70: Pull request #48 opened by megnar
`megnar:go-course-for-stud…`
2 months ago
39s

Автоматизация – Github Actions



Homeworkk/1 #73

Re-run triggered 2 months ago

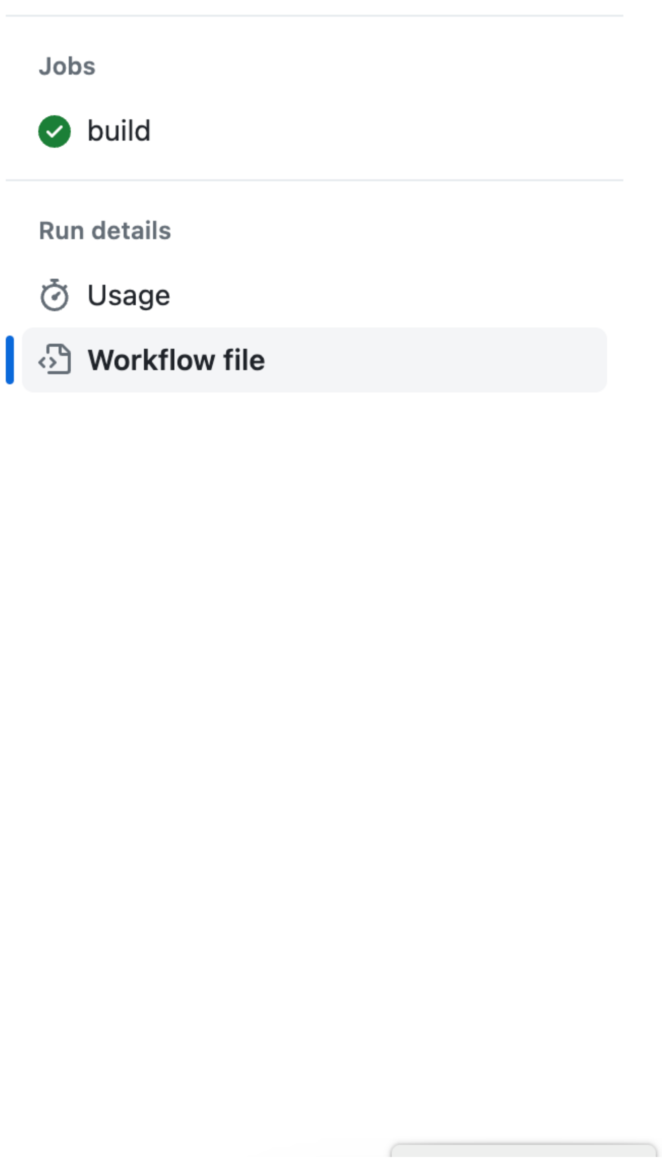
AntonOcean #46 `mikshakin:homeworkk/1`

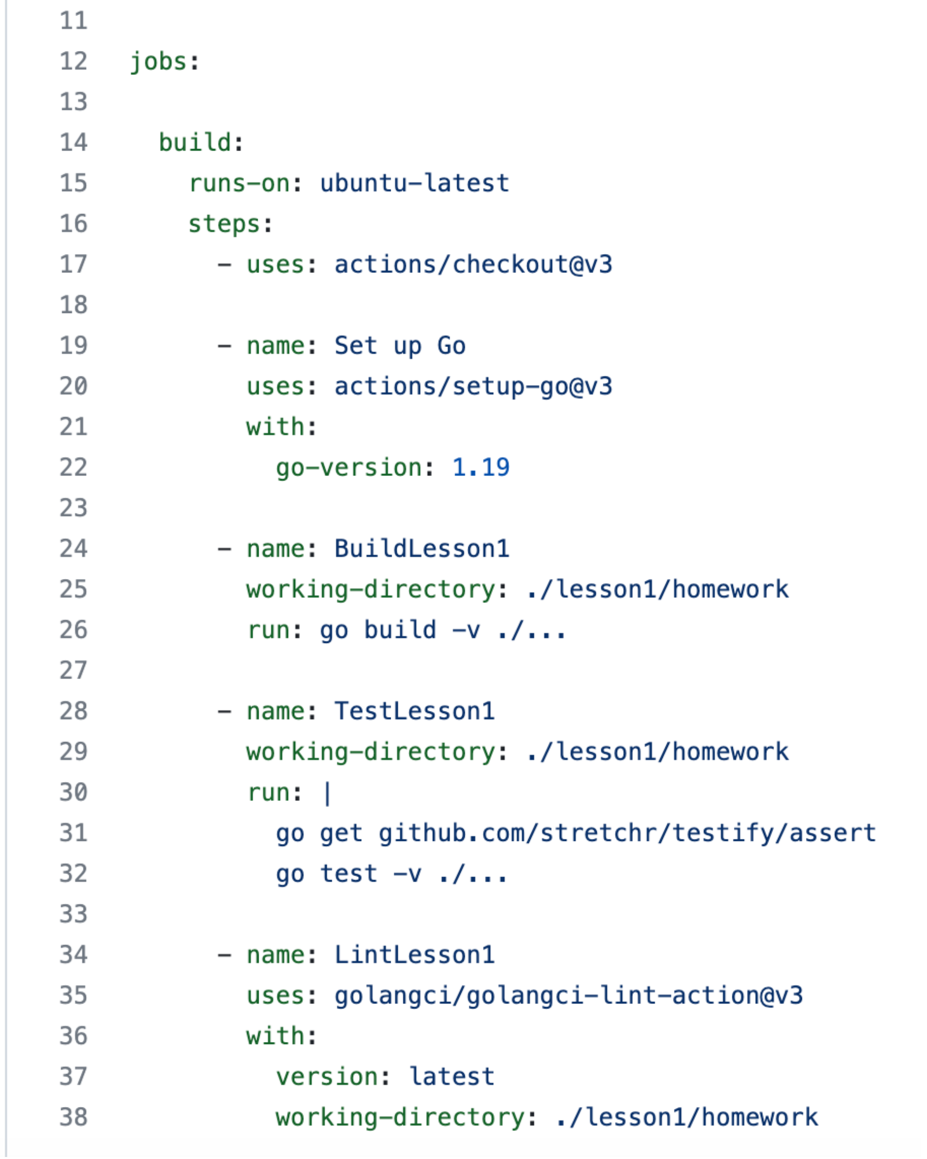
Status	Total duration	Artifacts
Success	32s	–

go.yaml
on: pull_request

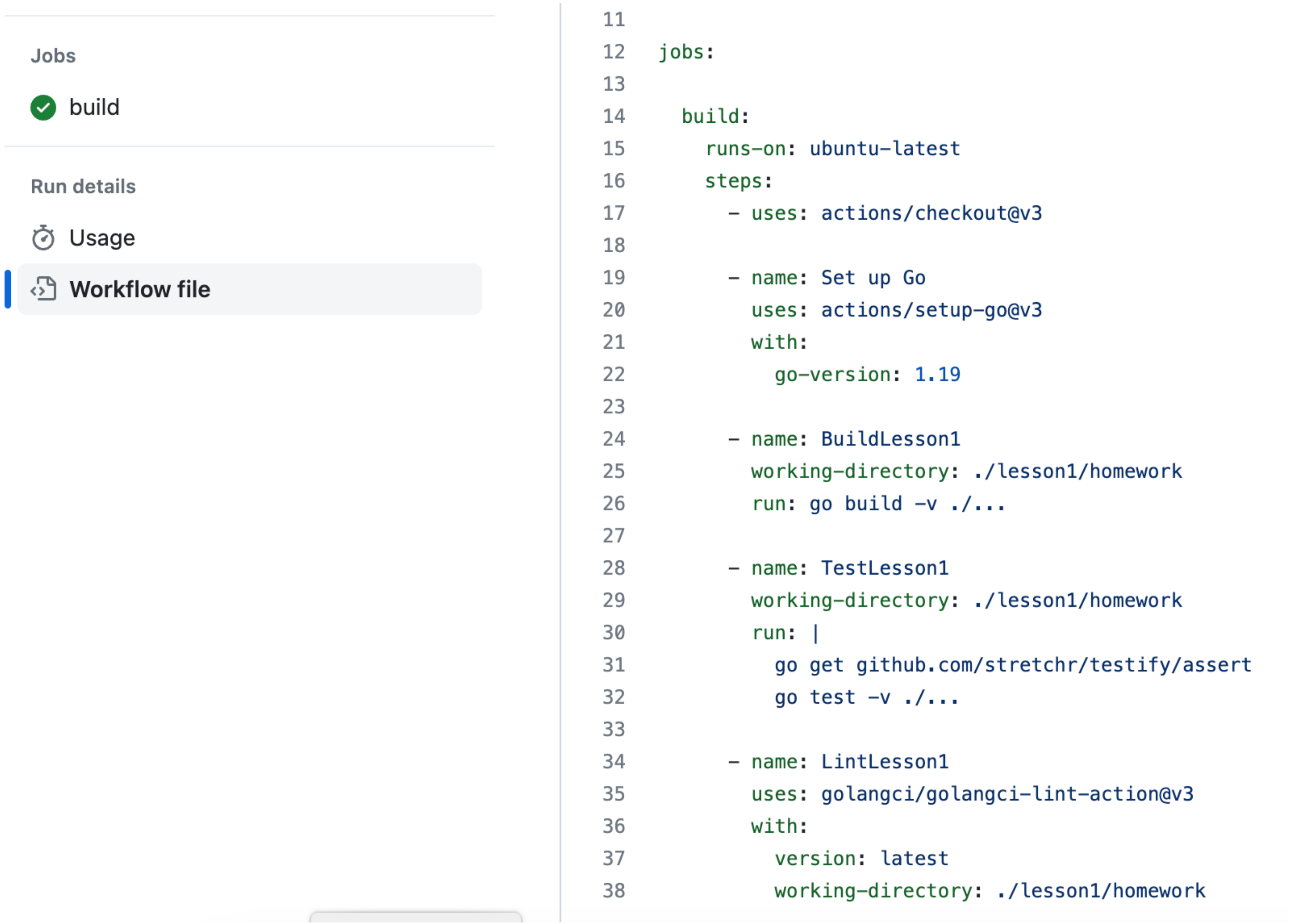
build 19s

Автоматизация – Github Actions




```yaml

jobs:

  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3

      - name: Set up Go
        uses: actions/setup-go@v3
        with:
          go-version: 1.19

      - name: BuildLesson1
        working-directory: ./lesson1/homework
        run: go build -v ./...

      - name: TestLesson1
        working-directory: ./lesson1/homework
        run: |
          go get github.com/stretchr/testify/assert
          go test -v ./...

      - name: LintLesson1
        uses: golangci/golangci-lint-action@v3
        with:
          version: latest
          working-directory: ./lesson1/homework
```

Автоматизация – Github Actions



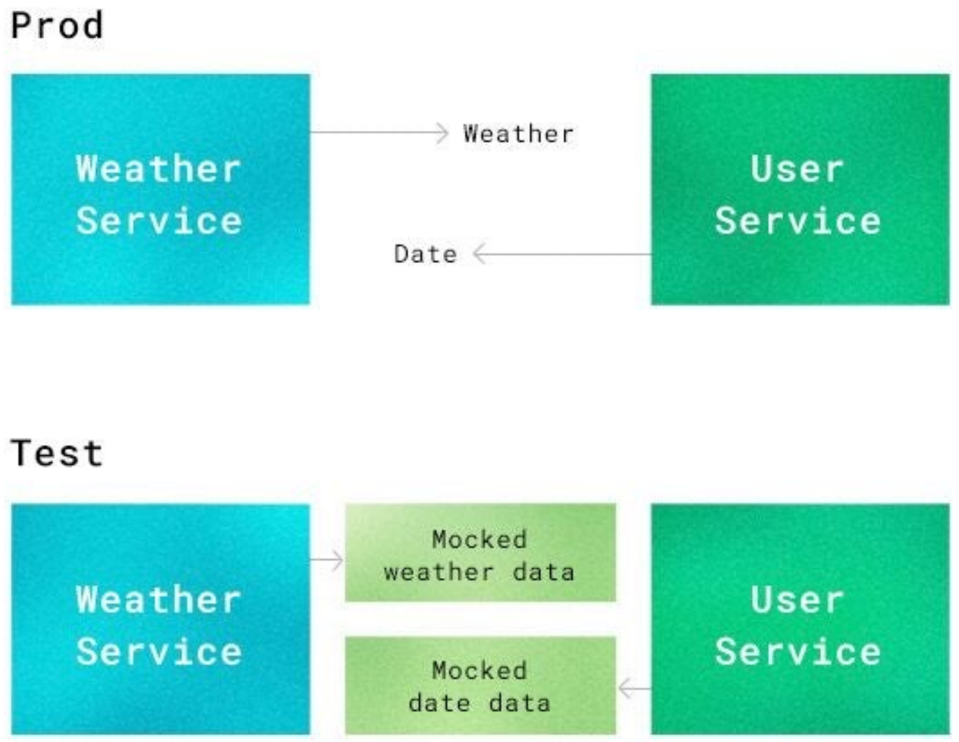
```yaml

jobs:

  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3

      - name: Set up Go
        uses: actions/setup-go@v3
        with:
          go-version: 1.19

      - name: BuildLesson1
        working-directory: ./lesson1/homework
        run: go build -v ./...

      - name: TestLesson1
        working-directory: ./lesson1/homework
        run: |
          go get github.com/stretchr/testify/assert
          go test -v ./...

      - name: LintLesson1
        uses: golangci/golangci-lint-action@v3
        with:
          version: latest
          working-directory: ./lesson1/homework
```

ТИНЬКОФФ

Мокирование

Мокирование

Mock — объекты, которые заменяют реальный объект в условиях теста и позволяют проверять вызовы своих методов.



Репозиторий

```
type Product struct {
  ID string
  Name string
  Price int
}

type ProductRepository struct {
  data  map[string]Product
  mutex sync.Mutex
}

func NewProductRepository() ProductRepository {
  return ProductRepository{
    data:  make(map[string]Product),
    mutex: sync.Mutex{},
  }
}
```

Интерфейс

```
type ProductRepositoryInterface interface {
  Add(product Product) error
}

func (r *ProductRepository) Add(product Product) error {
  r.mutex.Lock()
  defer r.mutex.Unlock()
  r.data[product.ID] = product
  return nil
}
```

Сервис

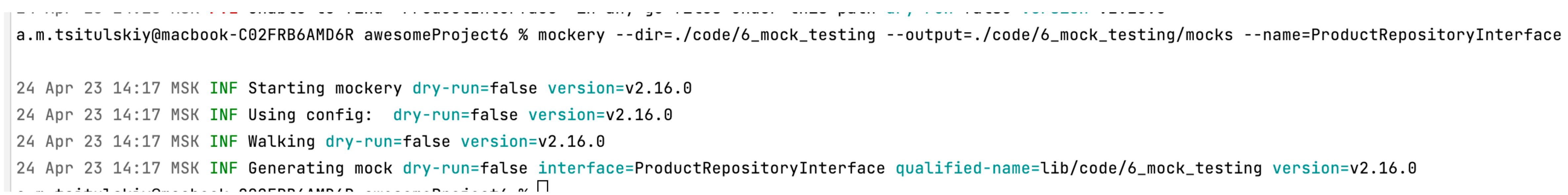
```
type ProductService struct {
  repo ProductRepositoryInterface
}

func NewProductService(repo ProductRepositoryInterface) ProductService {
  return ProductService{
    repo: repo,
  }
}
```

Сервис

```go
func (s ProductService) Insert(productID string, product Product) error {
  if len(productID) == 0 {
    return errors.New("productID can not be null")
  }

  err := s.repo.Add(Product{
    ID:    productID,
    Name:  product.Name,
    Price: product.Price,
  })
  if err != nil {
    return err
  }

  return nil
}
```

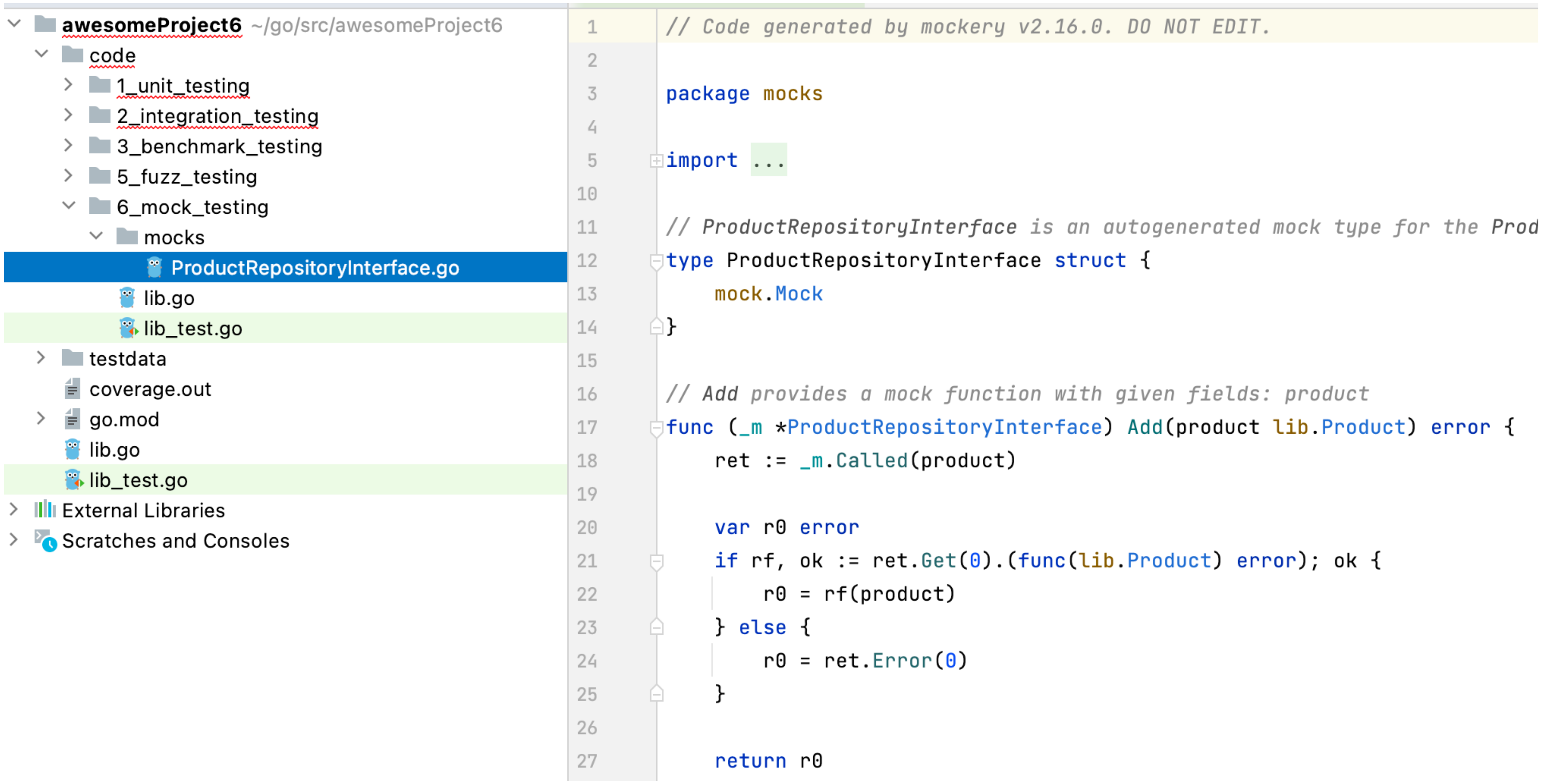
Mockery

go get github.com/vektra/mockery



```
[illegible]
a.m.tsitulskiy@macbook-C02FRB6AMD6R awesomeProject6 % mockery --dir=./code/6_mock_testing --output=./code/6_mock_testing/mocks --name=ProductRepositoryInterface

24 Apr 23 14:17 MSK INF Starting mockery dry-run=false version=v2.16.0
24 Apr 23 14:17 MSK INF Using config:  dry-run=false version=v2.16.0
24 Apr 23 14:17 MSK INF Walking dry-run=false version=v2.16.0
24 Apr 23 14:17 MSK INF Generating mock dry-run=false interface=ProductRepositoryInterface qualified-name=lib/code/6_mock_testing version=v2.16.0
[illegible]
```

Mockery



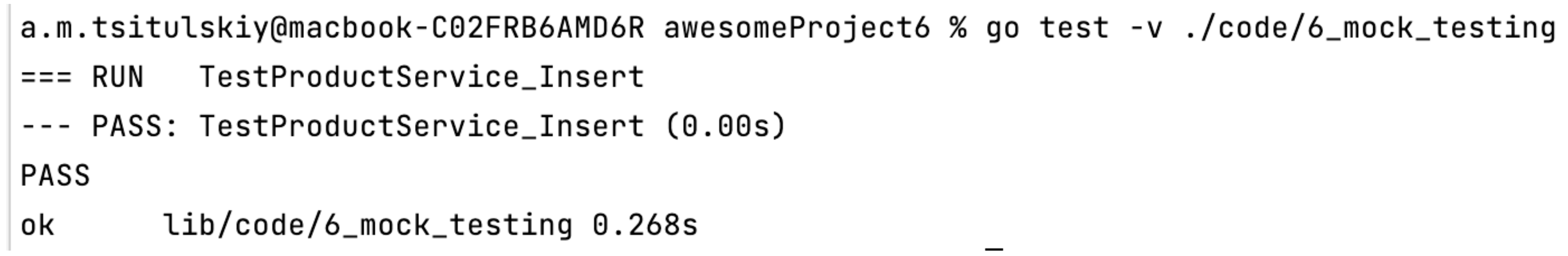
```
awesomeProject6 ~/go/src/awesomeProject6
  code
    1_unit_testing
    2_integration_testing
    3_benchmark_testing
    5_fuzz_testing
    6_mock_testing
      mocks
        ProductRepositoryInterface.go
      lib.go
      lib_test.go
  testdata
  coverage.out
  go.mod
  lib.go
  lib_test.go
External Libraries
Scratches and Consoles
```

```go
// Code generated by mockery v2.16.0. DO NOT EDIT.

package mocks

import ...

// ProductRepositoryInterface is an autogenerated mock type for the Prod
type ProductRepositoryInterface struct {
	mock.Mock
}

// Add provides a mock function with given fields: product
func (_m *ProductRepositoryInterface) Add(product lib.Product) error {
	ret := _m.Called(product)

	var r0 error
	if rf, ok := ret.Get(0).(func(lib.Product) error); ok {
		r0 = rf(product)
	} else {
		r0 = ret.Error(0)
	}

	return r0
```

Mockery

```
func TestProductService_Insert(t *testing.T) {
  repo := &mocks.ProductRepositoryInterface{}
  repo.On("Add", mock.AnythingOfType("lib.Product")).
    Return(nil).
    Once()

  service := lib.NewProductService(repo)

  err := service.Insert("2f1afe98-63c4-4f59-bcaf-1df835602bdb", lib.Product{
    Name:  "Macbook",
    Price: 20500,
  })

  assert.Nil(t, err)
}
```

Mockery

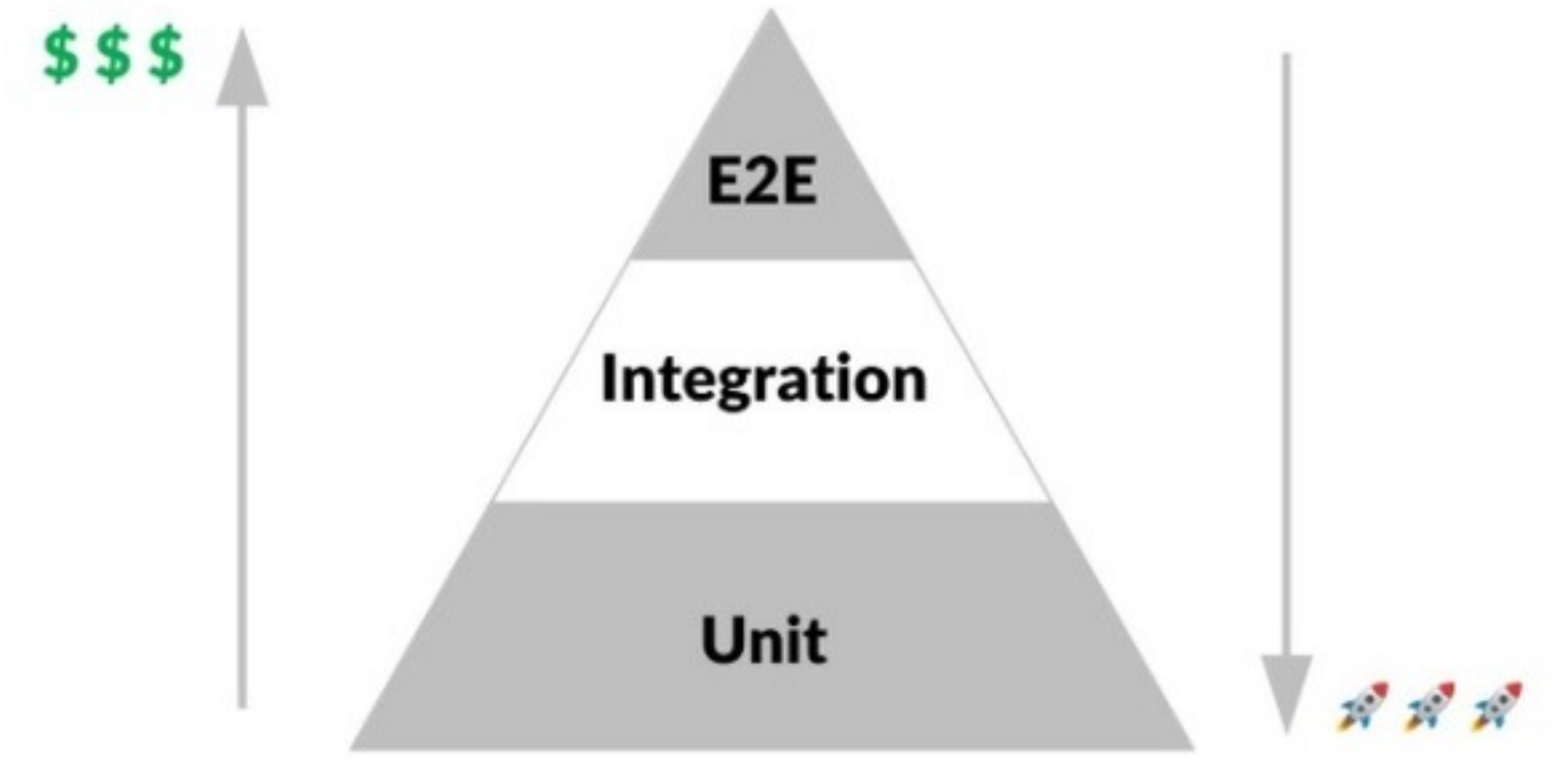


```
a.m.tsitulskiy@macbook-C02FRB6AMD6R awesomeProject6 % go test -v ./code/6_mock_testing
=== RUN   TestProductService_Insert
--- PASS: TestProductService_Insert (0.00s)
PASS
ok      lib/code/6_mock_testing 0.268s
```

ТИНЬКОФФ

Пирамида тестировани

Пирамида тестирования



Материалы

1. https://habr.com/ru/company/badoo/blog/301990/ - Профилирование и оптимизация программ на Go
2. https://www.ardanlabs.com/blog/2019/05/garbage-collection-in-go-part2-gctraces.html
3. https://dave.cheney.net/2013/06/30/how-to-write-benchmarks-in-go
4. https://www.youtube.com/watch?v=EJVp13f_aIs - Фаззинг
5. https://github.com/avelino/awesome-go#testing – Еще либы для тестирования
6. https://habr.com/ru/companies/oleg-bunin/articles/709248/ - Фаззинг