Testing with Go

DevFest Sevilla 2025

Nicolás Palumbo

About me

Nicolás Palumbo

Senior Software Engineer | New Relic

Agenda

No alarms and no surprises.

Agenda

- ¿Why?
- Test Pyramid
- Kata
- Go Test!
- Test Doubles
- More dependencies!
- Integration Tests
- Conclusion
- Thanks, Ask!
- Q / A

Why?

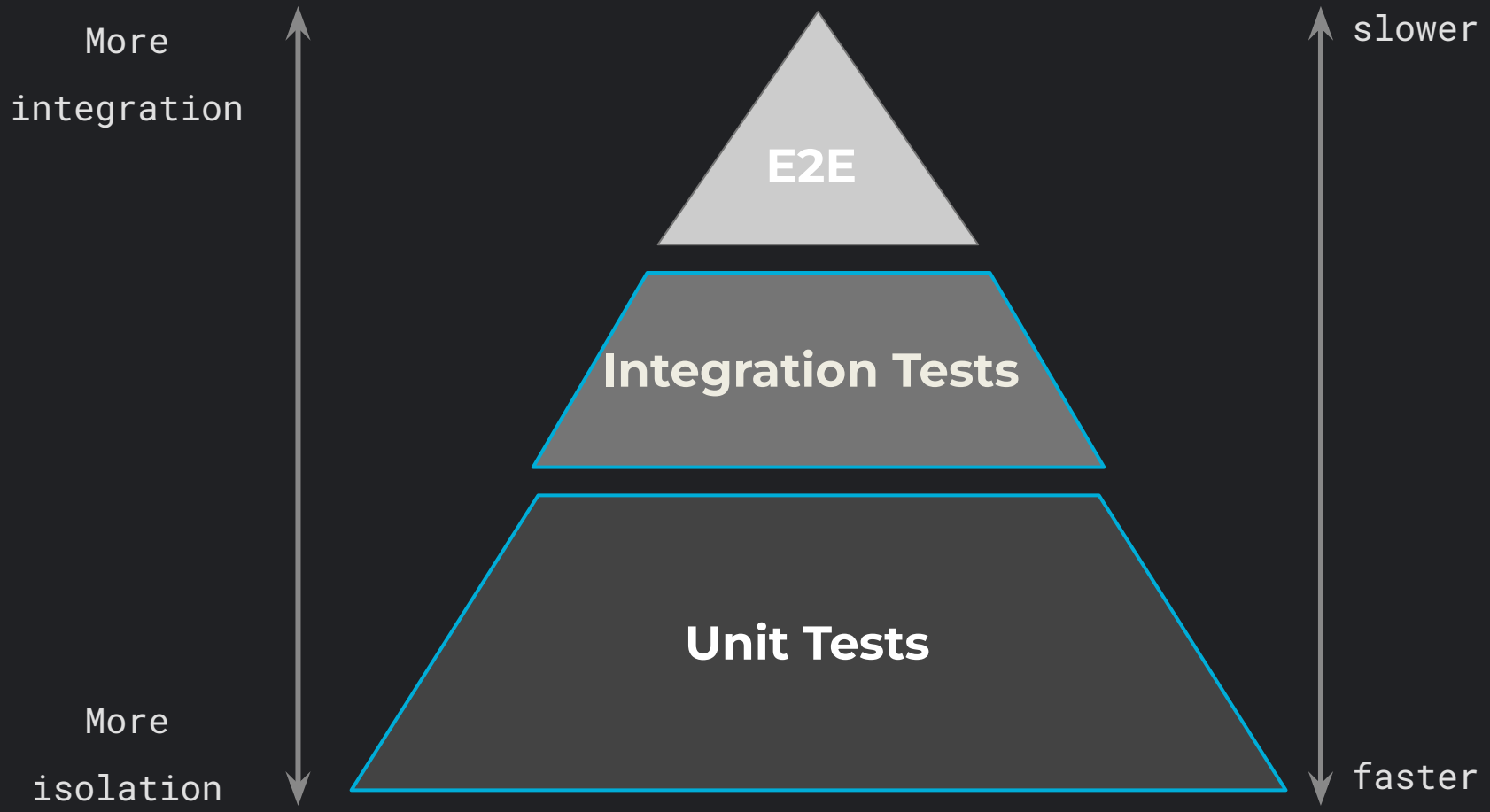
Why Oh Why?

Test Pyramid

$$V = \frac{na^2h}{12} \cot\left(\frac{\pi}{n}\right)$$

Test Pyramid

Test Pyramid



Kata

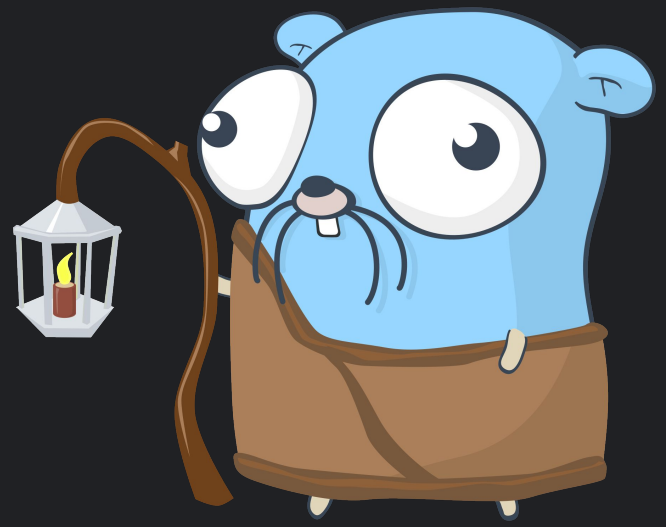
(型 or 形)

Kata

第一 Health & Level

All Characters, when created, have:

- **Health**, starting at 1000
- **Level**, starting at 1
- May be **Alive** or **Dead**, starting Alive



Kata

第二 Deal Damage

Characters can Deal Damage to Characters..

- **Damage** is subtracted from Health.
- When damage received exceeds current Health, Health becomes 0 and the character dies.

Go Test!

1, 2, 3… Probando

Go Test!

Benchmark Testing

To find performance issues.

```
package charlakata_test

import (
     "charlakata"
     "testing"
)

func BenchmarkPlayerRPG_Attack(b *testing.B) {
   for b.Loop() {

        // Fill this in!
   }
}
```

Go Test!

Fuzzy Testing

To find edge cases.

```
package charlakata_test

import (
     "charlakata"
     "testing"
)
// Fuzz test
func Fuzz_Damage_Boundaries(f *testing.F) {
   f.Add(1000, 1) // Seed corpus

   // Fuzz target
   f.Fuzz(func(t *testing.T, initialHealth, amount int) {

       if targetPlayer.GetHealth() < 0 {
            t.Errorf("got %d for %d", target.GetHealth(), amount)
       }
   })
}
```

Go Test!

Coverage

```
➜  go test -cover ./... -coverpkg=charlakata -coverprofile=coverage.out
ok      charlakata      0.209s  coverage: 100.0% of statements in charlakata

➜  go tool cover -func=coverage.out
charlakata/player.go:16:        TakeDamage      100.0%
charlakata/player.go:20:        DealDamage      100.0%
charlakata/player.go:30:        GetHealth       100.0%
charlakata/player.go:34:        GetLevel        100.0%
charlakata/player.go:38:        IsAlive         100.0%
charlakata/player.go:42:        NewRpgPlayer    100.0%
total:                          (statements)    100.0%

➜ go tool cover -html=coverage.out
```

Test Doubles

doppelgänger

Test Doubles

Stubs

“**Stubs** provide canned answers to calls made during the test.”

```
type StubPlayer struct{}

func (s *StubPlayer) GetLevel() int {
     return 1
}

func (s *StubPlayer) IsAlive() bool {
     return true
}
```

Test Doubles

Mocks

"**Mocks** are pre-programmed with expectations which form a specification of the calls they are expected to receive."

```
// Given
targetPlayer.EXPECT().IsAlive().Return
(expected.alive).Times(1)

// When
// ...

// Then
targetPlayer.AssertExpectations(t)
```

Test Doubles

Spies

"**Spies** are stubs that also record some information based on how they were called."

```
type SpyPlayer struct{}


func (s *SpyPlayer) TakeDamage(health int) {
     s.TakeDamageCallCount++
}
```

More dependencies!

Fatto trenta, facciamo trentuno.

More dependencies!

Testify

- Error messages are more friendly and readable.
- Assertions are more readable.
- Assertions can be annotated with a message.

```
assert.Equal(t, 1000, player.GetHealth())

assert.Equal(t, 1, player.GetLevel())

assert.True(t, player.IsAlive(), "Player should be alive")
```

More dependencies!

Ginkgo & Gomega

- Ginkgo: Testing Framework.
- Gomega: Matchers.
- Combined, they provide a DSL for writing tests.

```
It("should start with the correct initial parameters", func() {
    player := charlakata.NewRpgPlayer(1000, 1)

    Expect(player.GetHealth()).To(Equal(1000))
    Expect(player.GetLevel()).To(Equal(1))
    Expect(player.IsAlive()).To(BeTrue())
})
```

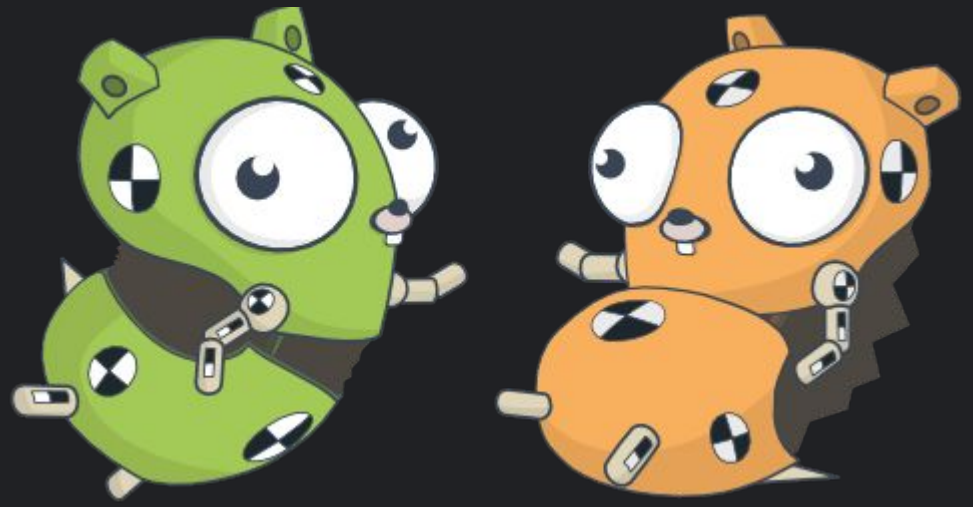
Integration Tests

$$(\int_0^b f(x)dx).$$

Integration Tests

Integration Tests

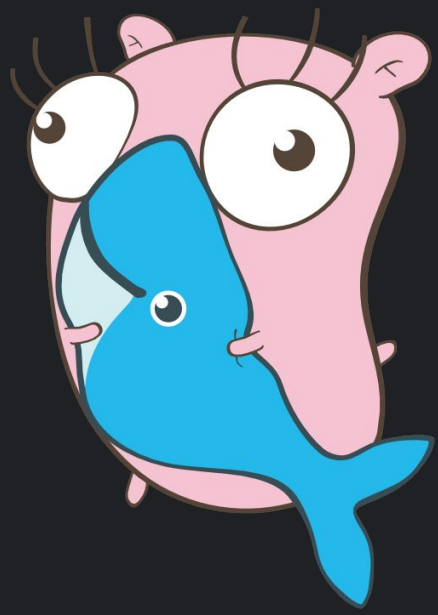
To find issues that come when testing the interaction between two or more modules.



Integration Tests

Testcontainers

- Uses real services that run in docker containers.
- Makes it possible to write tests that rely on the same tech that we use in production, without mocking or using in memory services.



Conclusion

C'est fini

Thanks, Ask!

Gratias vobis ago

Gracias, Pedir!

Come to the /dev/null talks event

devnull
talks

Q / A

Въпрос / Отговор

Bonus

Μπόνους

Bonus

Golang Sevilla:

https://www.meetup.com/golang-sevilla/

dev/null talks:

https://www.meetup.com/dev-null/

LinkedIn:

https://www.linkedin.com/in/nicol%C3%A1s-palumbo-9372615/

Gophers:

https://github.com/egonelbre/gophers

Original Slides:

https://docs.google.com/presentation/d/1sXTPhZUY5gDld7kpLMlvnPxa2Sps0ksvNzrrk58Uw9A/view

Martin Fowler's Test Pyramid & Test Doubles content:

https://martinfowler.com/(bliki/TestDouble.html|articles/practical-test-pyramid.html)