Testing

How to model readable system tests

O mnie

- Kamil Gajowy
- backend developer
- boomer 👴
- boardgames fan

Every story begins with Nest

// show how e2e/unit tests usually starts with

```
Test.createTestingModule({ ... })
```

Setup hell

// show a bit of setups, seeds, cqrs maybe

```
describe('...', () => {
    let result: unknown
    beforeEach(async () => {

    })

    test('it should....', async () => {
        // ...
    })

    test('it should also...')
})
```

Nest-ed Evolution

```
describe('...', () => {
    let result: unknown
    beforeEach(async () => {

    })
    describe('...', () => {
        let result: unknown
        beforeEach(async () => {

        })
        test('it should also...')
        test('it should also...')

        describe('...', () => {
            let result: unknown
            beforeEach(async () => {

            })
            test('it should also...')
            test('it should also...')
        })
    })
})
```

Un(expected)

```
// some code to show except( ... )
// which hardly tells us what to do
```

How we read?

// show some big legacy test file // and try to read with people what it tests

No-ise - GWT / AAA

AAA - Asset, Act, Assert GWT - Given, When, Then

// few words there

No-ise - setup chore

// show how to encapsulate setups

No-ise - Asset/Given (beforeEach)

// show how to encapsulate seeds, mock, stubs…

No-ise - Act/When (beforeEach)

// show how to encapsulate actions

No-ise - Asset/Then (it, test)

// show how to encapsulate expectations

Clean’em up 🕸️

// show how clean up artifacts, like created entities

Putting pieces together

// show how tests looks now in spec

Let’s compare readability

// not sure about this slide yet

Takeaways

- We read code much more often than we write
- Be the "good guy" - for others and yourself
- Express your intentions in enjoyable way
- Write tests by staring from spec, filling fixtures later