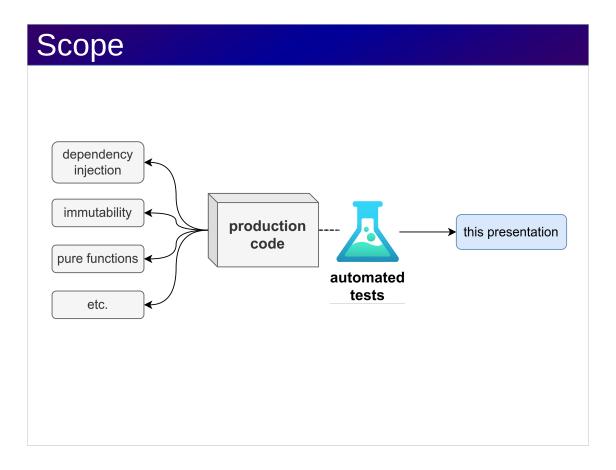
# Clean, beautiful tests

## Intro

- **Cristian Mocanu**
- 20 years experience as a software developer
- **-** DbSchema

https://dbschema.com/





- \* clean test code
- \* testable production code
  - \* if the production code is not easy to test, the tests will be bad
  - \* we won't discuss testable code principles
    - \* it's important, but we don't have time

## Outline

- Why are *good* tests important?
- Test code smells and fixes
- Takeaways

- Why are <u>good</u> tests important?
  Test code smells and fixes
- Takeaways

### Why are *good* tests important?



#### **Good tests**

- fast feedback (Cicu. verify results (mocks) fast feedback (clear test)
   lengthy debugging

#### **Bad tests**

- mock impl. details
- refactoring reluctance



fun

no fun

- \* fast feedback
  - \* reduced debugging time (clear test), compared to lengthy debugging (unclear test)
- \* functional documentation
  - \* always up-to-date
- \* refactor fearlessly
  - \* you have a safety net, that will tell you immediately if you broke anything (as opposed to having to fix a lot of tests when changing the smallest thing)
- \* it's fun
  - \* refactor without care / safety net

- Why are good tests important?Test code smells and fixes
- Takeaways

- Why are *good* tests important? Test code smells and fixes
- **Takeaways**

## Takeaways

- one test per functional requirement
  - the test name should explain the requirement
- setup, execute, verify
  - in this order
  - without interleaving, e.g. "setup, execute, verify, execute, verify" is not good
- no irrelevant detail
  - use test data builders
  - use custom assertions

# Main point

## Treat test code with the same care as your production code

- refactoring
- removing duplicationcreating abstractions
- etc.

# More info



https://github.com/cvmocanu/clean-tests-presentation/