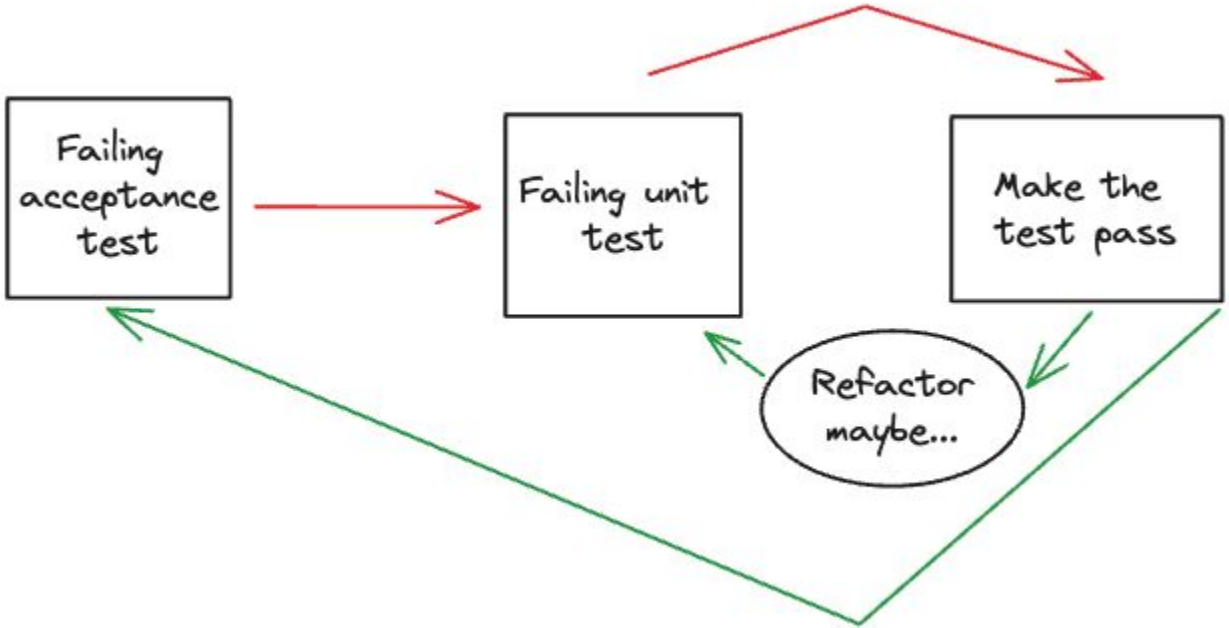
Outside-in ATDD

Enforce focus on business logic

“If the business fails to succeed, you will need a new job” (invented quote)

The most important rule of simple design is “**Passes the Tests**”

ATDD double development loop



Spoiler:

```
it('User first time login with email domain @davinci.care should see 3 free psychology consults', () => {
  iWorkForTenant({
    freePsychologyConsults: 3,
    name: 'Davinci',
  });
  iLoginWithEmail('mario.rossi@davinci.care');
  iCreateUserWithPersonalData({
    phoneNumber: '+393331234567',
    taxCode: 'RSSMRA80A01H501A',
    privacyConsent: true,
    surname: 'Rossi',
    name: 'Mario',
  });
  printFreePsychologyConsults();

  const expectedFreePsychologyConsults = 3;
  printedFreePsychologyConsultsShouldBe(expectedFreePsychologyConsults);
});
```

Stub example

Starting from the test it’s easier to create the repository.

Otherwise I can be tempted to write query directly in the service

First impressions

First impressions

Framework to know where to start (and when to stop **YAGNI**)

First impressions

The codebase grow naturally following Exagonal Architecture

- To test thing easily the code will be Loosely coupled

First impressions

Reduced Connascence with TDD

- Forced to think about **invalid state**
- Forced to think about good **names** and **types**

First impressions

For now it seems fun :)

Grazie

m.corradi@davinci.care

Domade?