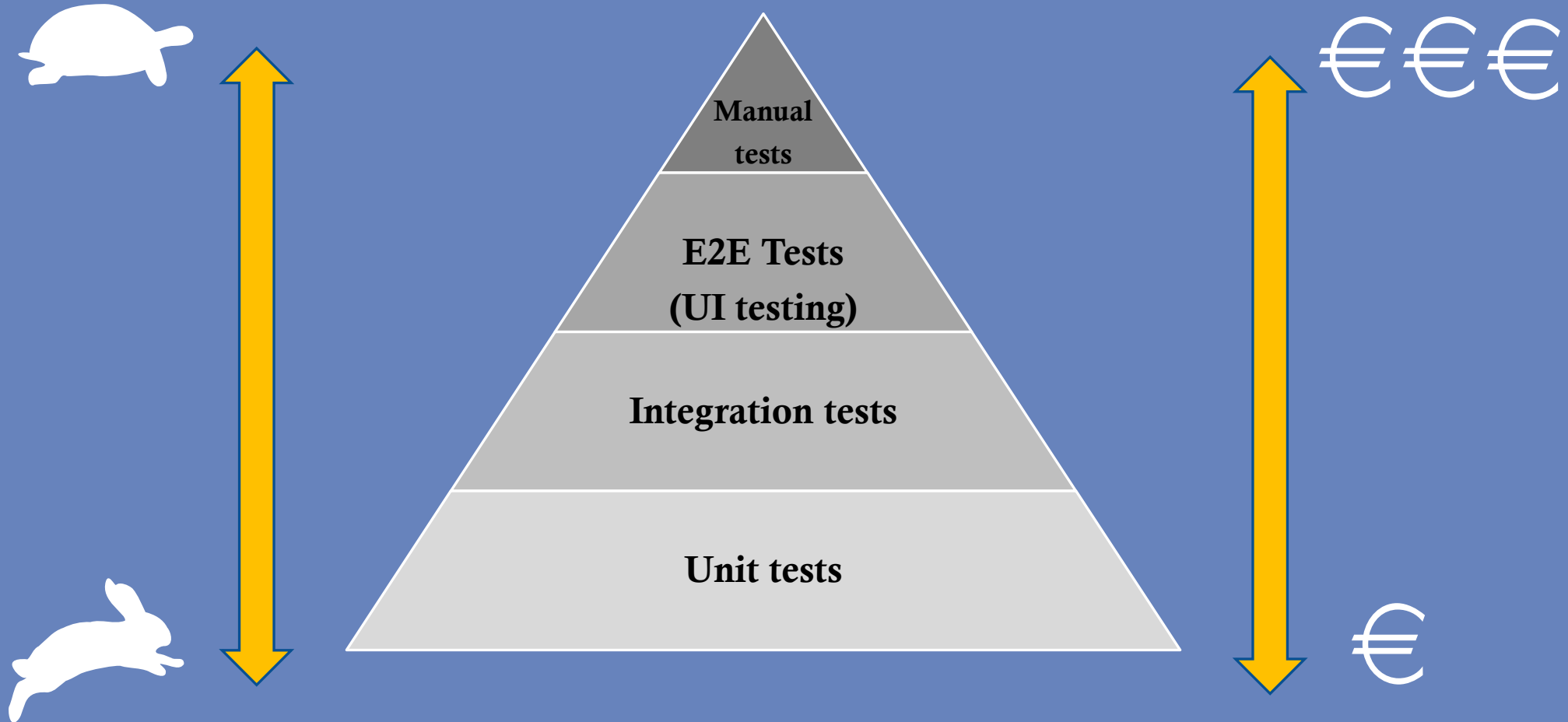


# QA – Quality assurance

By: Szigeccsán Dávid

# A TESZTELÉS ALAPJAI



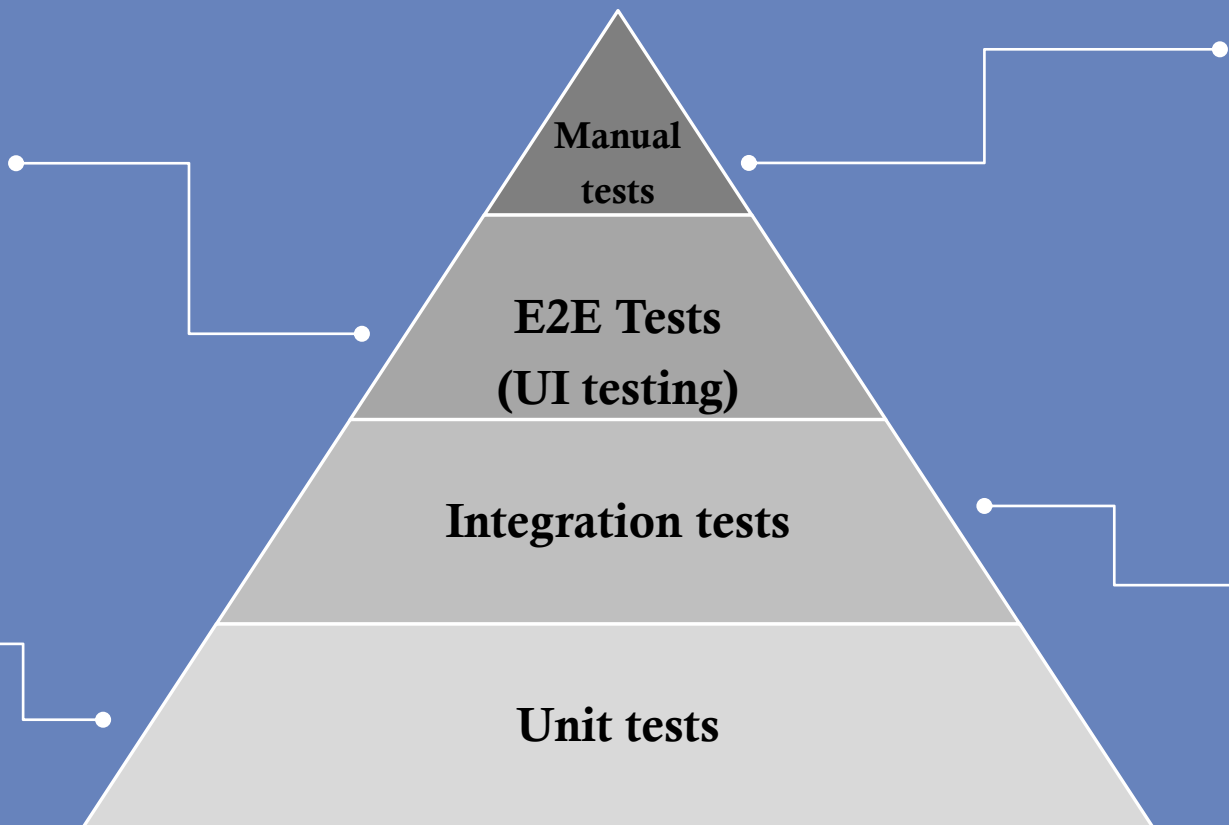
# A TESZTELÉS ALAPJAI

## E2E (End to End) tesztek

Lényegében a teljes rendszer működését ellenőrzi abból a szempontból, hogy egy adott végpont megfelelően működik-e. Azzal nem foglalkozik, hogy a végponton belül mely egységek hogyan kommunikálnak egymással, csak azt, hogy a folyamat végig működik-e.

## Unit tesztek

Általában ebből a típusú tesztből van a legtöbb, mert minden apró egységhez külön-külön készül, hogy a teljes rendszer legapróbb működési elemei le legyenek tesztelve.



## Manuális tesztek

A manuális tesztelés azokat a részeket érinti, amire nincsenek automatizált tesztek. Lehetőség szerint az ilyen típusú tesztek számát a lehető legalacsonyabbra kell csökkenteni.

## Integrációs tesztek

Ezekből kevesebb van. Csak az apró egységek összekapcsolását ellenőrzi, azok együttes működését, nem pedig külön az apró részleteket.

A phpUnit elsősorban egy unit tesztelésre kifejlesztett eszköz, ami az xUnit architektúrán alapszik. Működése nagyon egyszerű. Egy phar (php archive) fájlba csomagolva letölthető a teljes framework és egy paranccsal elindítható.

```
$ wget -O phpunit https://phar.phpunit.de/phpunit-8.phar  
$ chmod +x phpunit  
$ ./phpunit --version  
PHPUnit 8.0.0 by Sebastian Bergmann and contributors.
```

Ha már működik a keretrendszer, akkor létrehozhatjuk az első tesztünket.

# PHPUnit

Az első tesztünk létrehozására többféleképpen is van lehetőségünk.

1. Közvetlenül a programunk mellé vegyesen a production kóddal.
2. Egy production kódtól elválasztott külön mappába

Javasolt a tesztek és a production kód elválasztása, amit tipikusan az **src** és **test(s)** mappákkal szokás megoldani.

Az első teszt létrehozása után a PHPUnit-nak meg tudjuk adni, hogy az újonnan létrehozott tesztet futtassa. Például ha a teszt osztályunkat **EmailTest**-nek neveztük el, akkor az alábbi utasítás segítségével futtathatjuk a tesztek amiket ebbe az osztályba írtunk meg.

```
./phpunit tests/EmailTest
```

Feltétele a futtatásnak az is, hogy a PHPUnit **PHPUnit\Framework\TestCase** osztályából származtassuk le a teszt osztályunkat.

Az osztályba több lehetőségünk van teszteseteket megírni. A PHPUnit alapértelmezett működése szerint minden olyan publikus metódus, ami **test** kulcsszóval kezdődik az tesztesetnek számít.

Ha ettől eltérünk, akkor a **/\*\* @test \*/** annotációval jelezhetjük egy publikus metódusról, hogy az egy teszteset.

```
public function testIsValidEmailFormatShouldReturnTrueWhenTheParameterIsValidEmailAddress()  
{  
    $this->assertTrue(  
        Email::IsValidEmailFormat('user@example.com')  
    );  
}
```

F.I.R.S.T

F.I.R.S.T

**F**AST  
**I**SOLATED/INDEPENDENT  
**R**EPEATABLE  
**S**ELF-VALIDATING  
**T**HOROUGH/TIMELY



# MUTATION TESTING

## Mi az a Mutation Testing?

A Mutation Testing egy hiba alapú tesztelési technika. Ezzel a technikával a tesztek hatékonyságát lehet vizsgálni.

## Hogyan működik?

A tesztelés során a kódbázisból apró módosításokkal generál új változatokat (ezek a mutánsok). Minden változat csak egy helyen tér el az eredeti kódtól. Ezeken a módosított verziókon futtatja a teszt eseteket.

## Teszt eltörik vagy lefut?

Ha a teszt eltörik, az azt jelenti, hogy a mutáció detektálva lett (megöltük a mutánst).




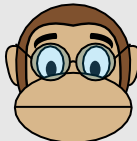

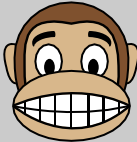

Ha viszont minden teszt lefut, az azt jelenti, hogy nincs tökéletesen letesztelve az adott kódrészlet (túlélte a mutáns).

## Futtatás eredménye

A futtatás eredménye tartalmazza a módosításokat, amiket nem sikerült detektálni, hogy arra új teszt esetet tudjunk készíteni.

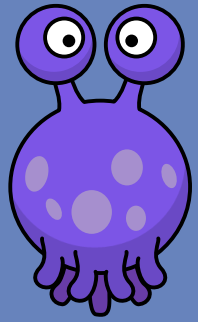
7

# MUTATION TESTING

Program	Tesztesetek	Tesztek eredménye
		
		
		



Infection



Wiki





Thanks!

