Introducción a las pruebas automáticas

Esteban Manchado Velázquez <emanchado@demiurgo.org>

¿Qué es una «prueba automática»?

Programa que comprueba que otro programa funcione correctamente

Resultado repetible

(entorno controlado, NO producción)

Monitorización NO es una prueba automática

Razones para escribir pruebas

Ayudan a refactorizar

Explican el comportamiento esperado

Evitan cambios accidentales

Ahorran tiempo de prueba

Nos dan confianza

Clasificaciones

Muchas taxonomías

Unitarias
Funcionales

White box

Black box

Prueba unitaria

```
var RUList = require("../RUList").RUList;

describe("Recently-Used List", function() {
    it("add last items at the beginning", function() {
        var list = new RUList();

        var item1 = "~/.ssh/config",
            item2 = "~/.bashrc";
        list.add(item1);
        list.add(item2);
        expect(list.get(0)).toEqual(item2);
        expect(list.get(1)).toEqual(item1);
    });
});
```

Prueba funcional

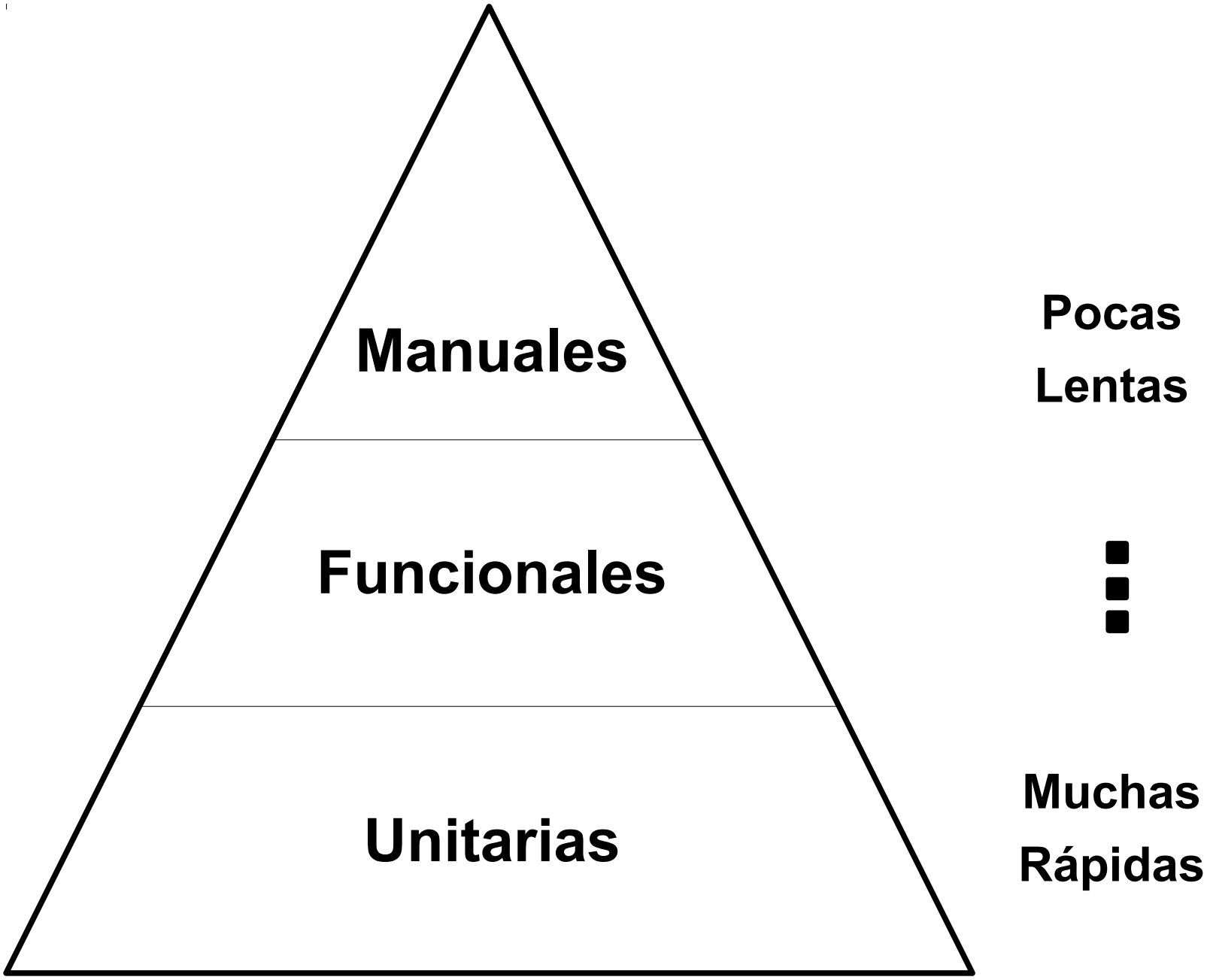
```
var url = "http://localhost:8000";

describe("Widget API", function() {
    it("can save simple widgets", function() {
        var props = { name: "Widget 1", type: "basic" };

        Request.post(url + "/widgets", props).then(function (res) {
            var data = JSON.parse(res);

            return Request.get(url + "/widgets/" + data.id);
        }).then(function (getRes) {
            var data = JSON.parse(getRes);

            expect(data.name).toEqual(props.name);
            expect(data.type).toEqual(props.type);
        });
    });
});
```


Manuales
Funcionales
Unitarias
Pocas
Lentas
Muchas
Rápidas

Anatomía de una prueba

Prueba unitaria

```
var RUList = require("../RUList").RUList;

describe("Recently-Used List", function() {
    it("add last items at the beginning", function() {
        var list = new RUList();

        var item1 = "~/.ssh/config",
            item2 = "~/.bashrc";
        list.add(item1);
        list.add(item2);
        expect(list.get(0)).toEqual(item2);
        expect(list.get(1)).toEqual(item1);
    });
});
```

Preparación

Afirmaciones

Prueba funcional

```
var url = "http://localhost:8000";

describe("Widget API", function() {
    it("can save simple widgets", function() {
        var props = { name: "Widget 1", type: "basic" };

        Request.post(url + "/widgets", props).then(function (res) {
            var data = JSON.parse(res);

            return Request.get(url + "/widgets/" + data.id);
        }).then(function (getRes) {
            var data = JSON.parse(getRes);

            expect(data.name).toEqual(props.name);
            expect(data.type).toEqual(props.type);
        });
    });
});
```

P

A

Prueba unitaria con «setup» (*before*)

```
var RUList = require("../RUList").RUList;

describe("Recently-Used List", function() {
    before(function() {
        this.list = new RUList();
    });

P   it("add last items at the beginning", function() {
        var item1 = "~/.ssh/config",
            item2 = "~/.bashrc";
        this.list.add(item1);
        this.list.add(item2);
A       expect(this.list.get(0)).toEqual(item2);
        expect(this.list.get(1)).toEqual(item1);
    });
```

Cómo debe ser una prueba

No debe fallar cuando no haya problemas

Debe fallar cuando haya problemas

Pruebas pequeñas y triviales que cubran un solo caso

Pruebas independientes y sólo con los datos estrictamente necesarios

Ejercicio

Dada una clase que implementa una «lista de uso reciente»

Escribir pruebas que cubran la funcionalidad importante

Consejos

Intentar demostrar que el código *no funciona*,
no confirmar que sí funciona

Pruebas que no intentan fallar

```
test_assert(strcmpend("ab", "abb") < 0);
// Apesta porque lo siguiente también pasa:
// test_assert(strcmp("ab", "abb") < 0);

test_assert(strcasecmpend("abcDEf", "deg") < 0);
// Apesta porque lo siguiente también pasa:
// test_assert(strcmpend("abcDEf", "deg") < 0);
```

Los casos límite son
una fuente muy buena de pruebas

Pruebas comunes
(no muy útiles)

```
split_values = "1, 2, 3".split(", ")
assertEqual("1", split_values[0])
```

Pruebas adicionales (usando los límites)

```
split_values = "1, 2, 3, ".split(", ")
assertEqual("", split_values[3])

split_values = ", 1, 2, 3".split(", ")
assertEqual("", split_values[0])

split_values = "1,,2".split(",")
assertEqual("", split_values[1])

assertRaise(lambda x: "123".split(""))
```

Más pruebas comunes
(no muy útiles)

```
wrap_string(sl, "wrapping functionality: w00t", 10);
/* … */
test_streq(cp, "wrapping\nfunctional\nity: w00t\n");
```

Más pruebas adicionales (usando los límites)

```
wrap_string(sl, "wrapping functionality: w00t", 5);
/* … */
test_streq(cp, "wrapp\ning\nfunct\nional\nity:\nw00t\n");
```

Ejecutar las pruebas frecuentemente

Controlar el entorno

No reusar datos de prueba

Facilitar el proceso de pruebas

Extras del DVD

Mocks

Evitan preparaciones tediosas

Permiten probar interacciones con sistemas externos

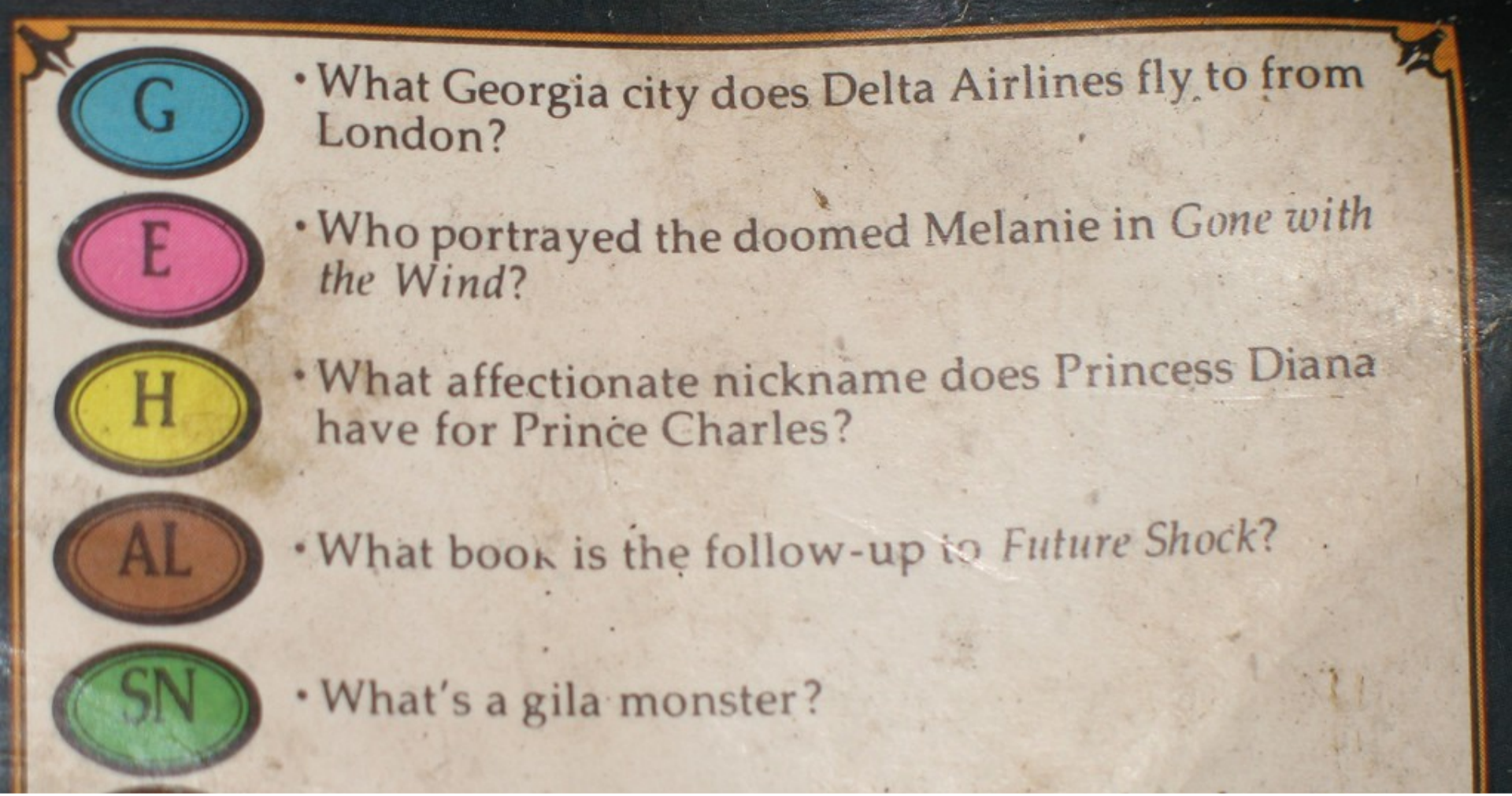
Permiten/ayudan a reproducir ciertas interacciones

Pero no sean abusadores

Ejemplo de uso de mocks

```
it("serve files from default mountPath = /", function(done) {
    var fileContents    = "file contents",
        dirFileContents = "dir file contents";
    var head = new RoboHydraHeadFilesystem({
        documentRoot: '/var/www',
        fs: fakeFs({'/var/www/file.txt':     fileContents,
                    '/var/www/dir/file.txt': dirFileContents})
    });

    checkRouting(head, [
        ['/file.txt', fileContents],
        ['/dir/file.txt', dirFileContents],
        ['/dir/non-existentfile.txt', {statusCode: 404}]
    ], done);
});
```


G
• What Georgia city does Delta Airlines fly to from London?
E
• Who portrayed the doomed Melanie in *Gone with the Wind*?
H
• What affectionate nickname does Princess Diana have for Prince Charles?
AL
• What book is the follow-up to *Future Shock*?
SN
• What's a gila monster?

¿Preguntas?

Créditos de las fotografías

- Michael Verhoef:
  https://www.flickr.com/photos/nettsu/5197360443
- Biodiversity Heritage Library:
  https://www.flickr.com/photos/biodivlibrary/10575423436/
- Hiné Mizushima:
  https://www.flickr.com/photos/sheishine/17440749922/
- Helgi Halldórsson:
  https://www.flickr.com/photos/8058853@N06/6972773870/
- Duncan Hull:
  https://www.flickr.com/photos/dullhunk/202872717/
- cornflakegirl_:
  https://www.flickr.com/photos/marycunningham/379760083/
- IvanClow:
  https://www.flickr.com/photos/ivanclow/4329024374
- Laura:
  https://www.flickr.com/photos/amalthya/66109250/
- Andy Bullock:
  https://www.flickr.com/photos/andybullock77/3239860312

http://demiurgo.org/workshops/intro-pruebas/
https://emanchado.github.io/camino-mejor-programador/

http://robohydra.org