



# BUENAS PRÁCTICAS DE GESTIÓN Y DESARROLLO DE SOFTWARE

DRA. IREN LORENZO FONSECA

# Estructuración de los temas

## Metodología Constructiva (Bottom-Up )

2



# Estructuración de los temas

Metodología Constructiva (Bottom-Up )

3

**Metodologías de Gestión del  
Desarrollo de SW**

Buenas Prácticas de desarrollo y gestión  
de software

**Desarrollador**

**Dev, DevOps, Q&A**

Metodologías y Tecnologías para la  
gestión y el desarrollo de software

# Algo de contexto

Tipo de empresas implicadas



ISO 9001



ISO 27001

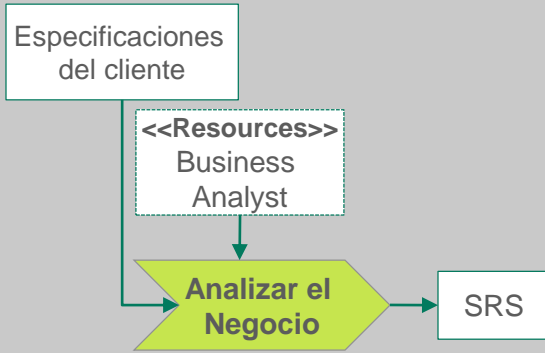


GGTI

Primeras etapas del desarrollo  
de software

# Workflow

6



# Elaboración de SRS

## Ingeniería de Requisitos



### Técnicas



Entrevistas



Observación



Estudio de documentación



Cuestionarios

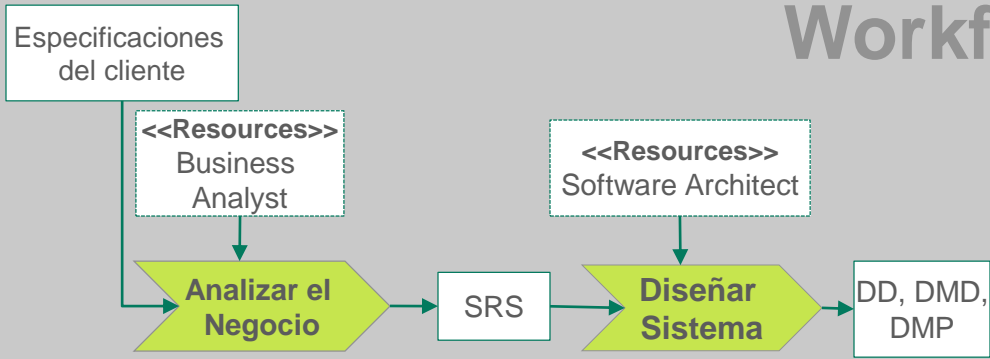


Reuniones



Desarrollo de Prototipos

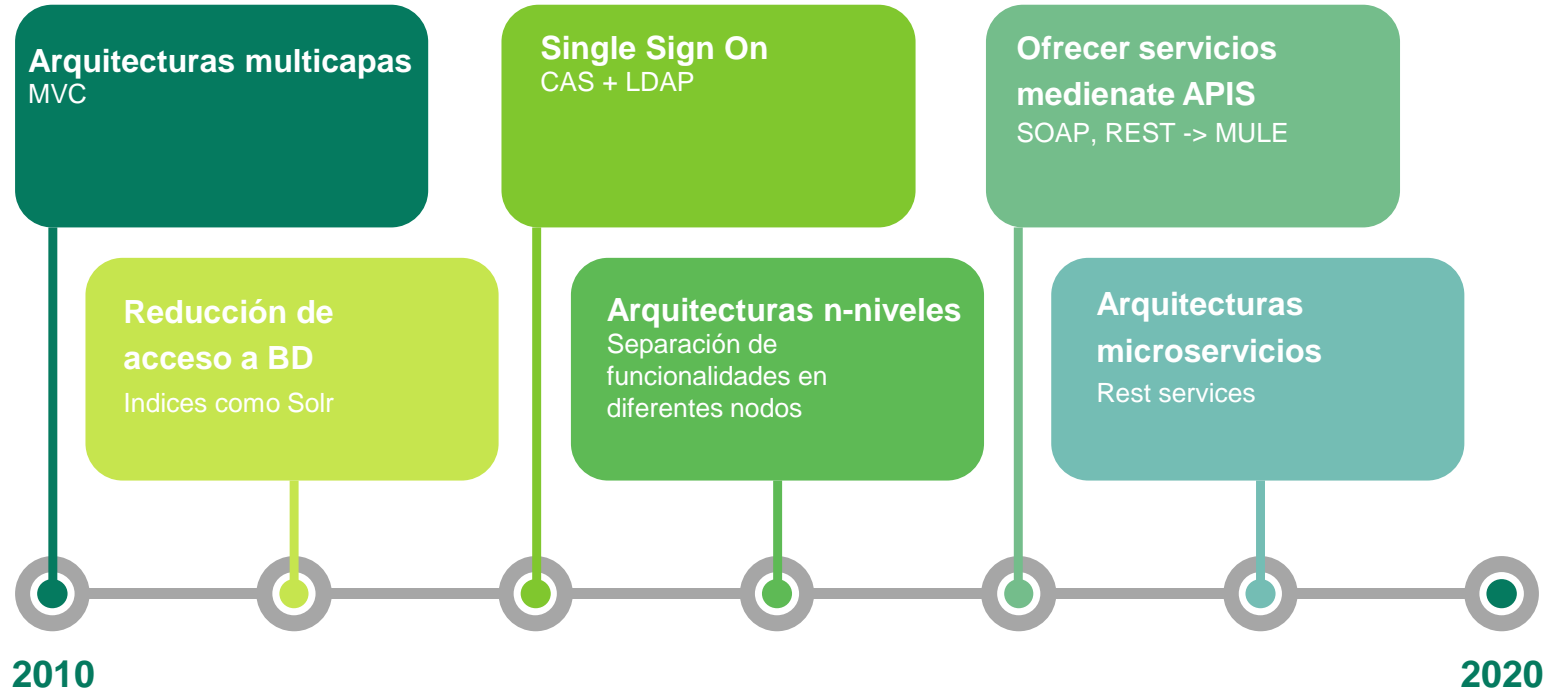
# Workflow





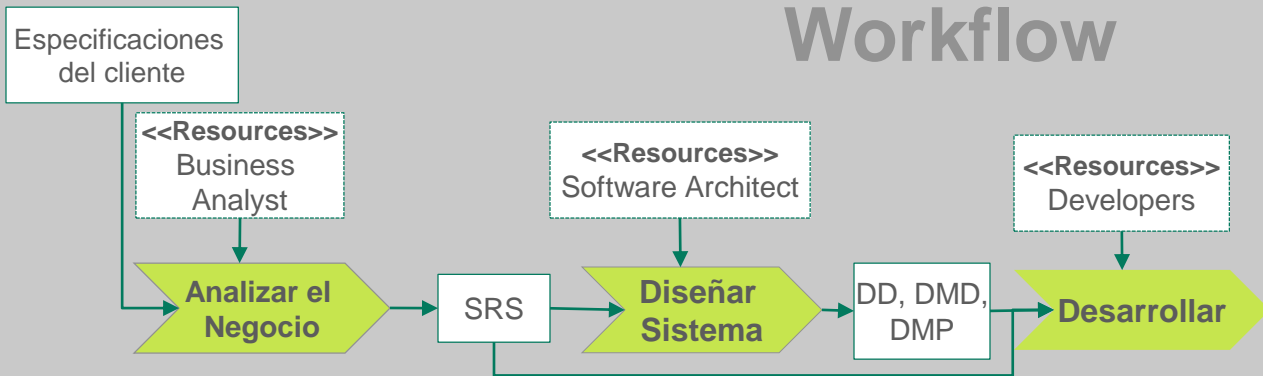
# Diseño del sistema

## Arquitectura



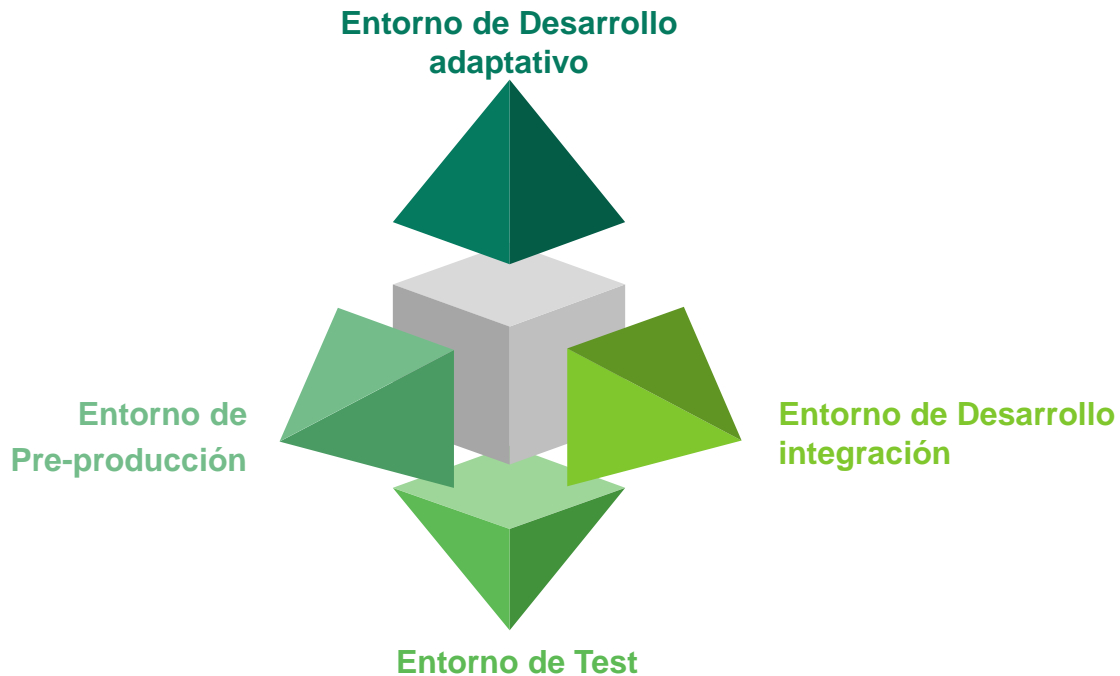
# Workflow

10

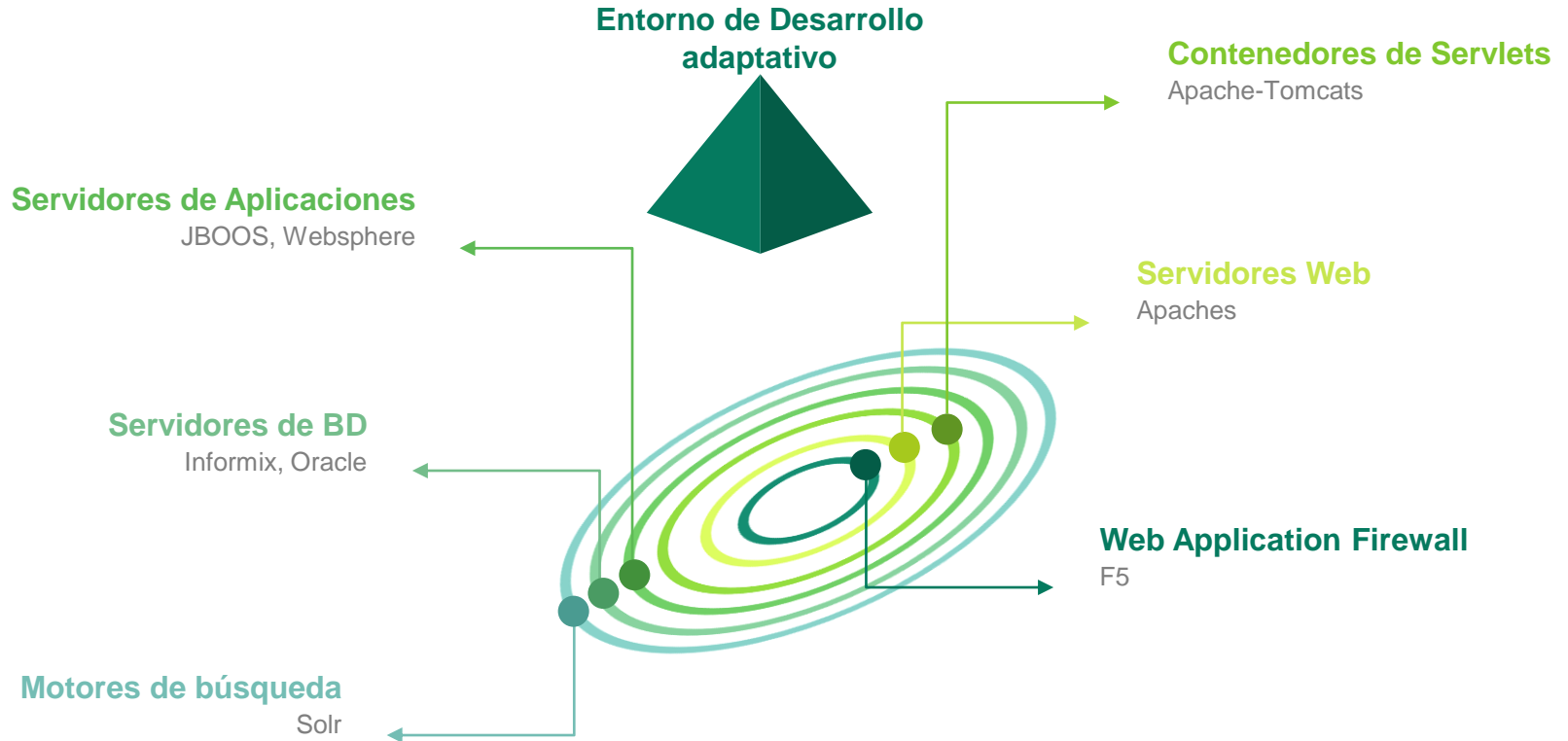


# Desarrollo

## Entornos de trabajo



# Entornos de Trabajo



# Metodologías

## Gestión y Desarrollo

### eXtreme Programing

Generalmente para proyectos de mediano alcance

### Pair Programming

Grandes beneficios para los tiempos de resolución de problemas

### SCRUM

Depende del Proyecto pero la media es de 2-4 semana de Spring

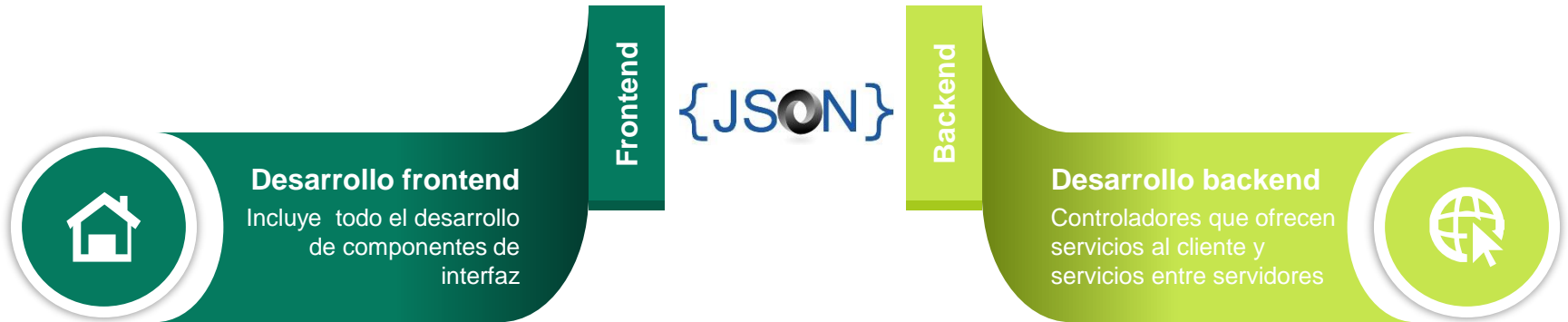
### Test-driven development

Eleva los indicios de calidad del proyecto



# Separación backend-frontend

Facilita la separación del equipo



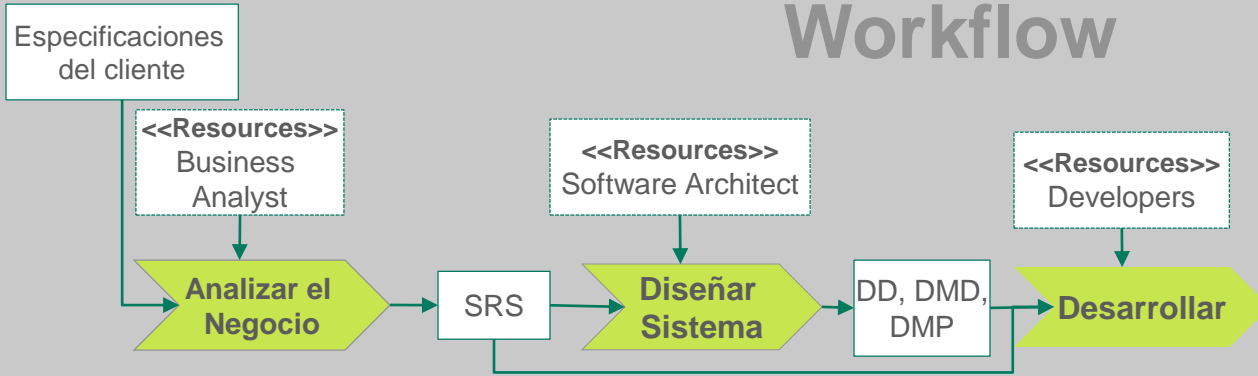


GGTI

Herramientas de soporte  
al desarrollo

# Workflow

16





# Herramientas

## Soporte al desarrollo



# Herramientas

## Soporte al desarrollo



# Herramientas

Soporte al desarrollo



## Sistemas de control de versiones

Guardar, etiquetar y recuperar versiones.

Gestión de trabajo a través de ramas

## Control de versiones

## Herramientas CI



### Plan Compile

Detecta fallos de compilación y de tests



### Plan Delivery

Sube la versión al Nexus



### Plan Quality

Interactúa con la plataforma de calidad para generar estadísticas

# Herramientas

Soporte al desarrollo



## Sistemas de control de versiones

Guardar, etiquetar y recuperar versiones.

Gestión de trabajo a través de ramas

## Control de versiones

## Herramientas CI



### Plan Compile

Detecta fallos de compilación y de tests



### Plan Delivery

Sube la versión al Nexus

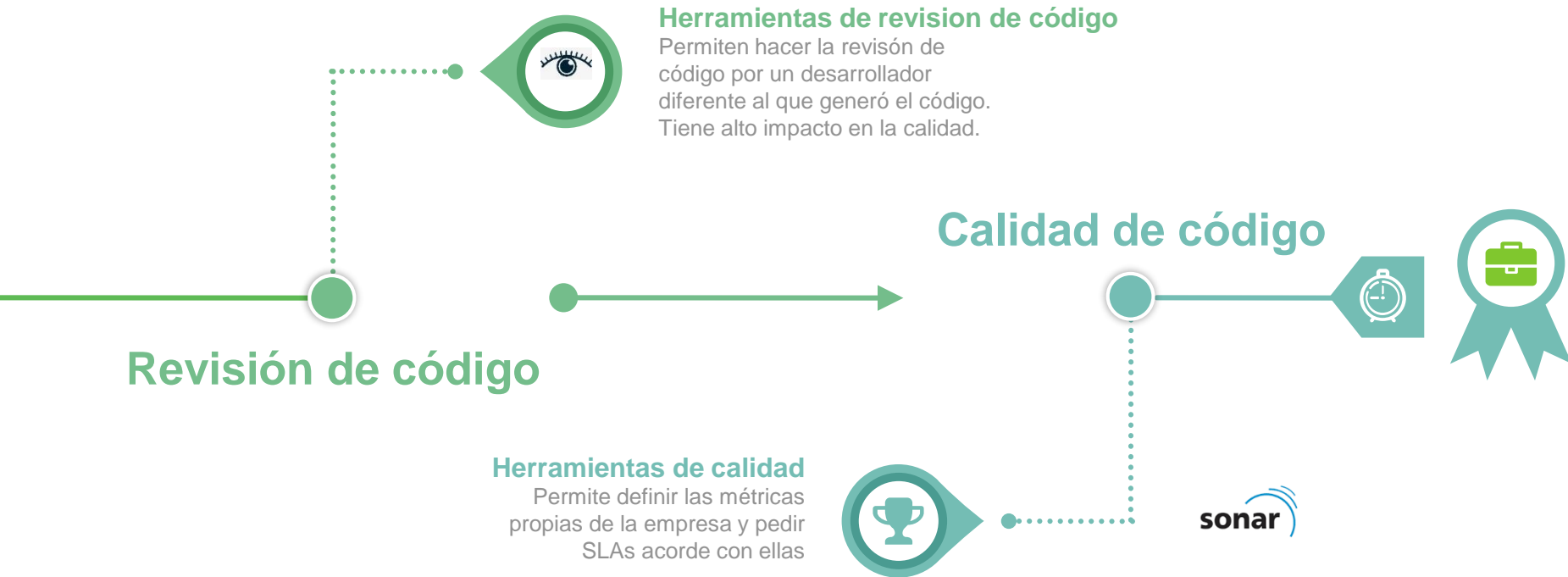


### Plan Quality

Interactúa con la plataforma de calidad para generar estadísticas

# Herramientas

## Soporte al desarrollo



# Entregables

## SLAs



### Unit Testing

Un mínimo de 70% de cobertura



### Test Automáticos

Un mínimo de 50% de cobertura de Test Cases



### Calidad de código

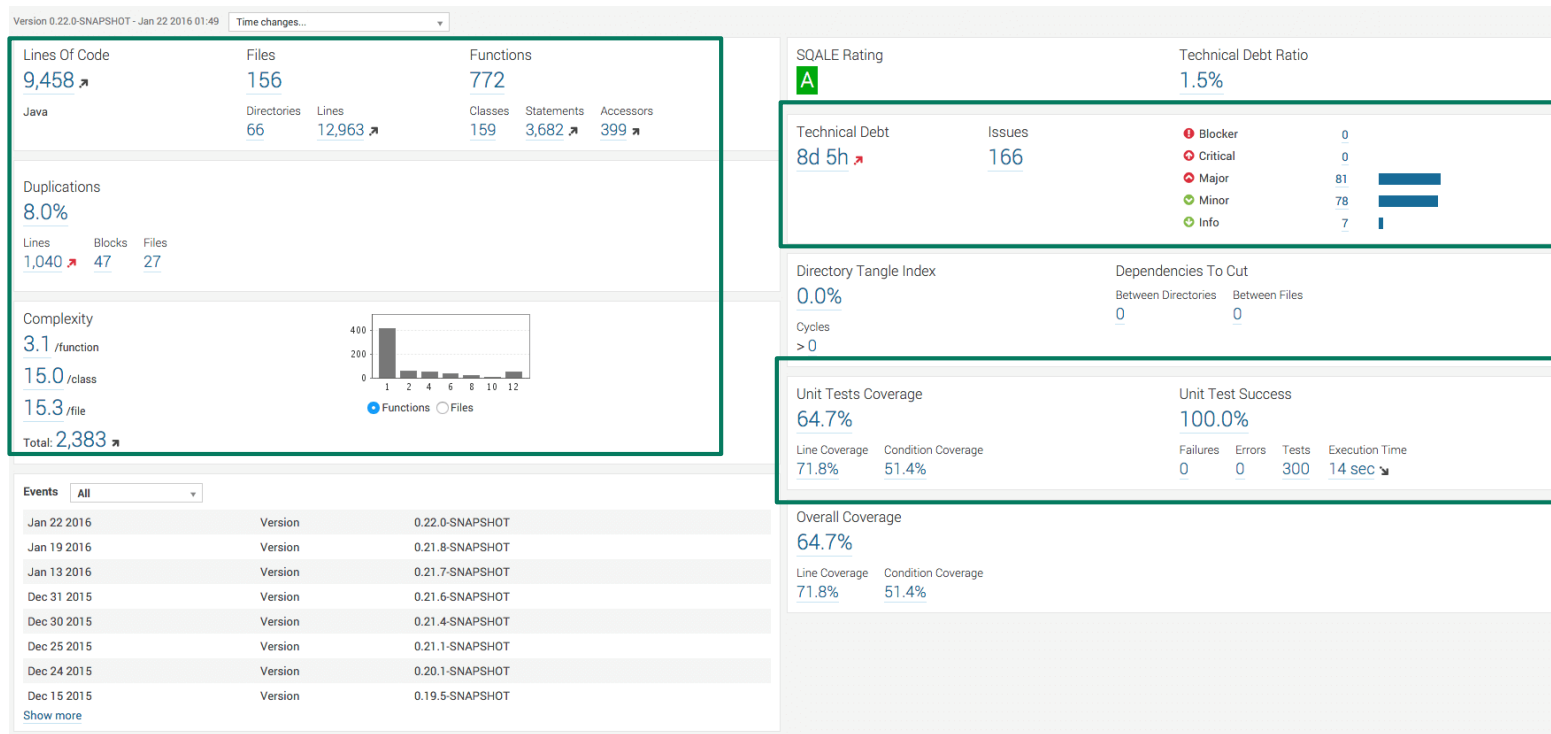
No se permiten blockers

El resto de las categorías deben estar igual o mejor que la entrega anterior:

- Critical
- Major
- Minor
- Info
- Cobertura

# Sonar

## Métricas de calidad



# Entregables

## SLAs



### Unit Testing

Un mínimo de 70% de cobertura



### Test Automáticos

Un mínimo de 50% de cobertura de Test Cases



### Calidad de código

No se permiten blockers

El resto de las categorías deben estar igual o mejor que la entrega anterior:

- Critical
- Major
- Minor
- Info
- Cobertura



# Herramientas

## Soporte al desarrollo (Resumen)

### Gestión de tareas



### Gestión de dependencias



### Sistemas de control de versiones



### Sistemas de revision de código



### Herramientas de CI



### Herramientas de Calidad





**GGTI**

Instalación y Pruebas

# Componentes de una entrega

## Entrega a DevOps



### Launch Plan incremental

Recetario de instalación del Sistema sobre la version existente. Incluye instrucciones de rollback.

### Launch Plan (LPN)

Recetario de la instalación del Sistema desde cero

### Release Note (RLN)

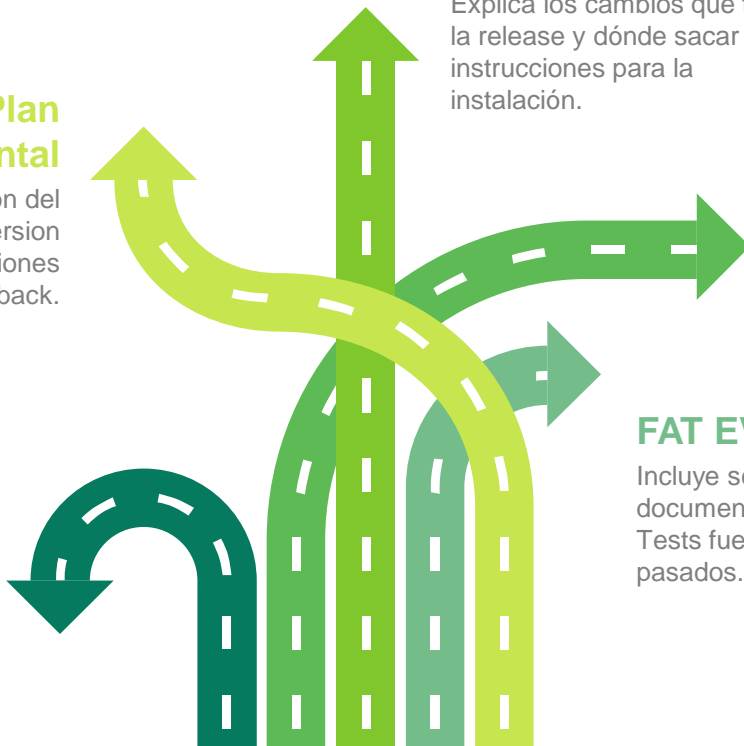
Explica los cambios que tiene la release y dónde sacar las instrucciones para la instalación.

### Factory Acceptance Test (FAT)

Incluye los resultados de las pruebas y los datos de calidad de la aplicación en comparación con la version anterior.

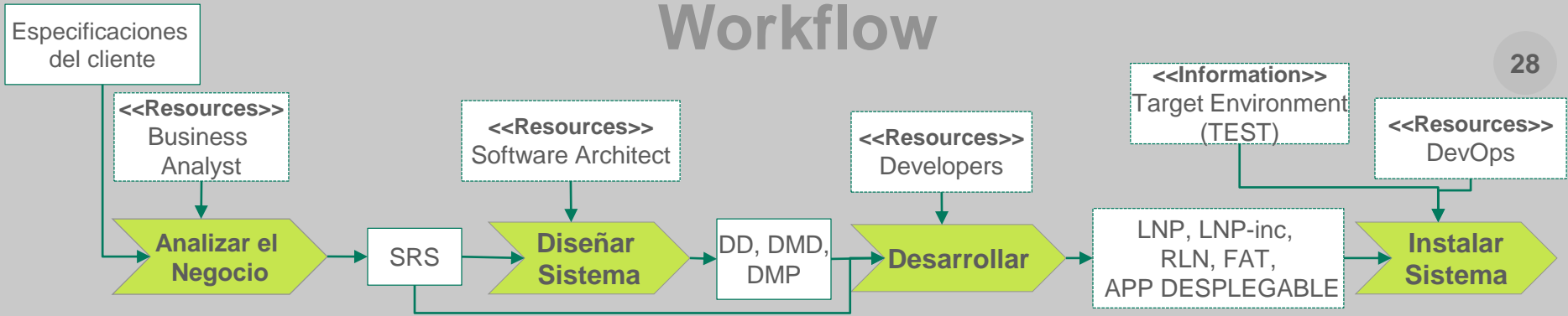
### FAT EVIDENCES

Incluye screenshots y documentos que avalen que los Tests fueron efectivamente pasados.



# Workflow

28





## Aplicación compilada

La entrega incluye la aplicación compilada y LPN explica como desplegarla



## Enlace repositorio

Enlace al repositorio de versiones y LPN con instrucciones de los tokens a cambiar y el profile MAVEN a ejecutar



## Puppet

Permite la gestión de la configuración del entorno de modo que la ejecución de tareas de despliegue se facilita



## Ansible

Automatiza el aprovisionamiento de software, la gestión de configuraciones y el despliegue de aplicaciones

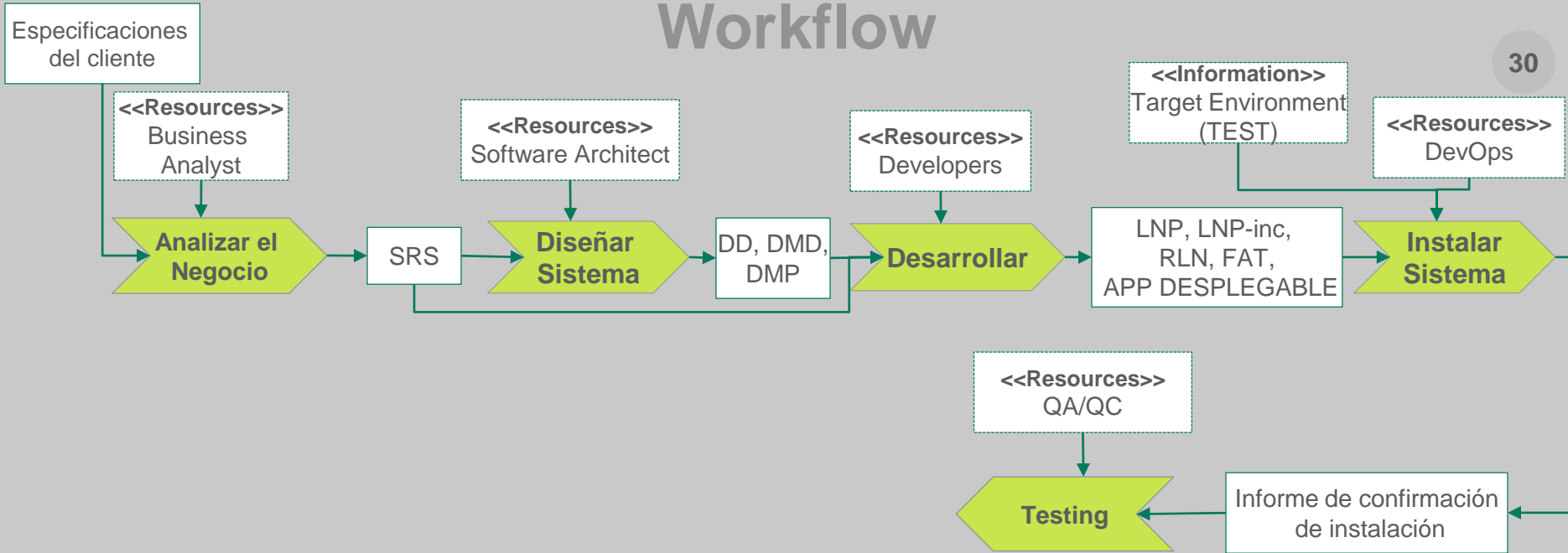


## Kubernetes

Automatización del despliegue, ajuste de escala y manejo de aplicaciones en contenedores

# Workflow

30



# Calidad

## Tests aplicados al producto

Testing

31

### Test automáticos

Comprobar que los tests automáticos entregados funcionan correctamente



### Tests Funcional

Los test cases que han sido creados a la par que el SRS, se ejecutan para ver si la aplicación cumple con los requisitos



### Chequeo de documentación

Comprobar que la documentación de calidad entrega es correcta. Que efectivamente se ejecutó un FAT y que los datos de calidad de código son correctos



### Tests de Intrusion

Comprobar con la guía OWASP la seguridad de la aplicación entregada



### Test de estrés

A través de tests de performance se comprueba la capacidad de la aplicación para soportar un número alto de usuarios concurrentes



### Generar Informe de SAT

Se genera el informe de System Acceptance Tests

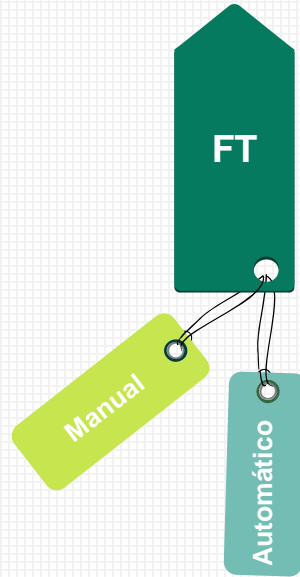


# QA & Testing Services

¿Qué se espera de ellos?

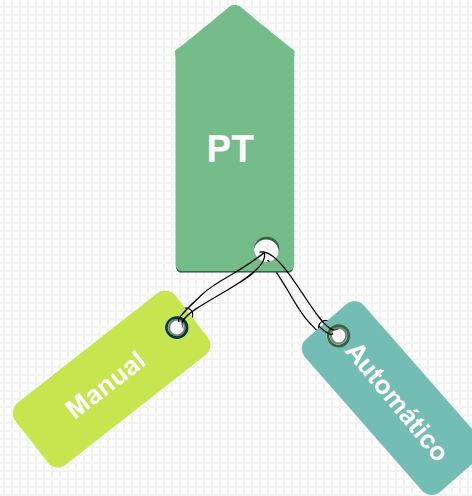
Testing

32



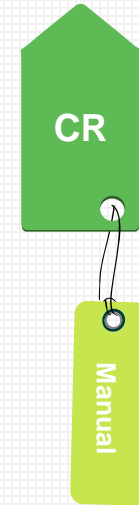
## Functional Test

Orientado al Test Case. Prueba las funcionalidades del sistema



## Performance Test

Prueba el Sistema bajo diferentes niveles de carga



## Code Review

Revisiones de código



# Tipos de Tests Funcionales

Testing

33

## System Acceptance Test (SAT)

Test llevabos a cabo por QA, siguiendo un plan de Test basado en TCs

## Smoke Test

Prueba rápida sobre sistemas inestables. Prueba los TCs más críticos para saber si vale la pena hacer tests más profundos

## Factory Acceptance Test (FAT)

Proceso que evalúa si el paquete compilado cumple con el diseño especificado

## Sanity Test

Es sobre sistemas más estables. No es necesario seguir los TCs pero si es necesario saber de las funcionalidades del sistema

## Retest

Ejecutar el mismo plan de test sobre las mismas funcionalidades pero en diferentes versiones del producto

## Regression Test

Comprobar que funcionalidades anteriores siguen funcionando correctamente.

## User Acceptance Test (UAT)

Pruebas funcionales llevas a cabo por los usuarios finales



D

Test

### E2E

Pruebas de Sistema completo. Más orientado a los casos de pruebas. Más difíciles de mantener por su complejidad



### Integration Test

Prueba integración entre diferentes capas del sistema. Ej. conexión con la BD o integración entre servicios. Son más complejos y más difíciles de mantener



### Unit Test

Práctica de probar trozos pequeños de código. Si el código está mal diseñado es difícil de testear.



MOCK

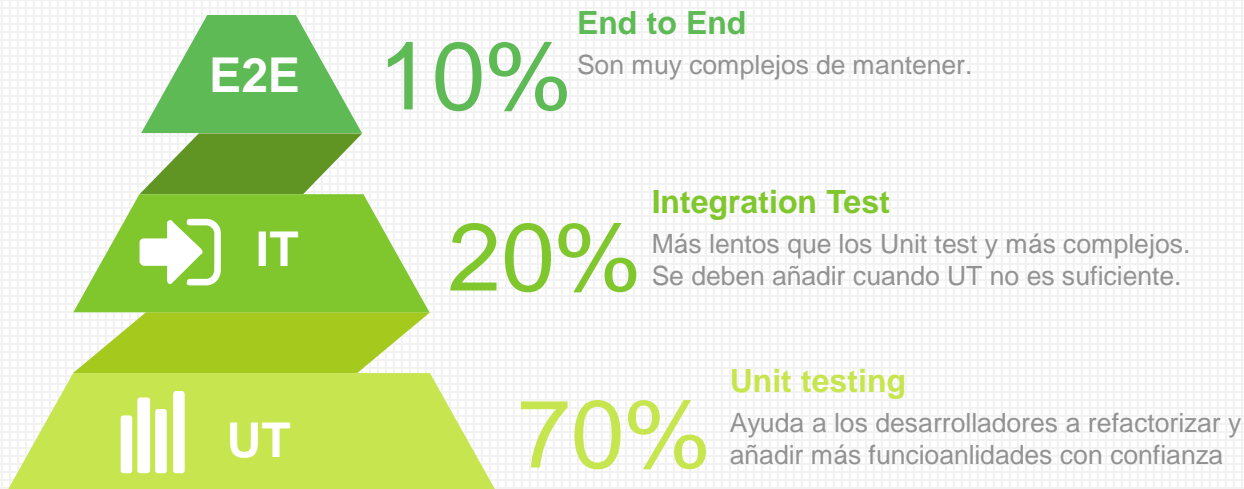
Mockito

SINON.JS

# Google Testing Blog

## Propuesta

35



# Calidad

## Tests aplicados al producto

Testing

36

### Test automáticos

Comprobar que los tests automáticos entregados funcionan correctamente



### Tests Funcional

Los test cases que han sido creados a la par que el SRS, se ejecutan para ver si la aplicación cumple con los requisitos



### Chequeo de documentación

Comprobar que la documentación de calidad entrega es correcta. Que efectivamente se ejecutó un FAT y que los datos de calidad de código son correctos



### Tests de Intrusion

Comprobar con la guía OWASP la seguridad de la aplicación entregada



### Test de estrés

A través de tests de performance se comprueba la capacidad de la aplicación para soportar un número alto de usuarios concurrentes

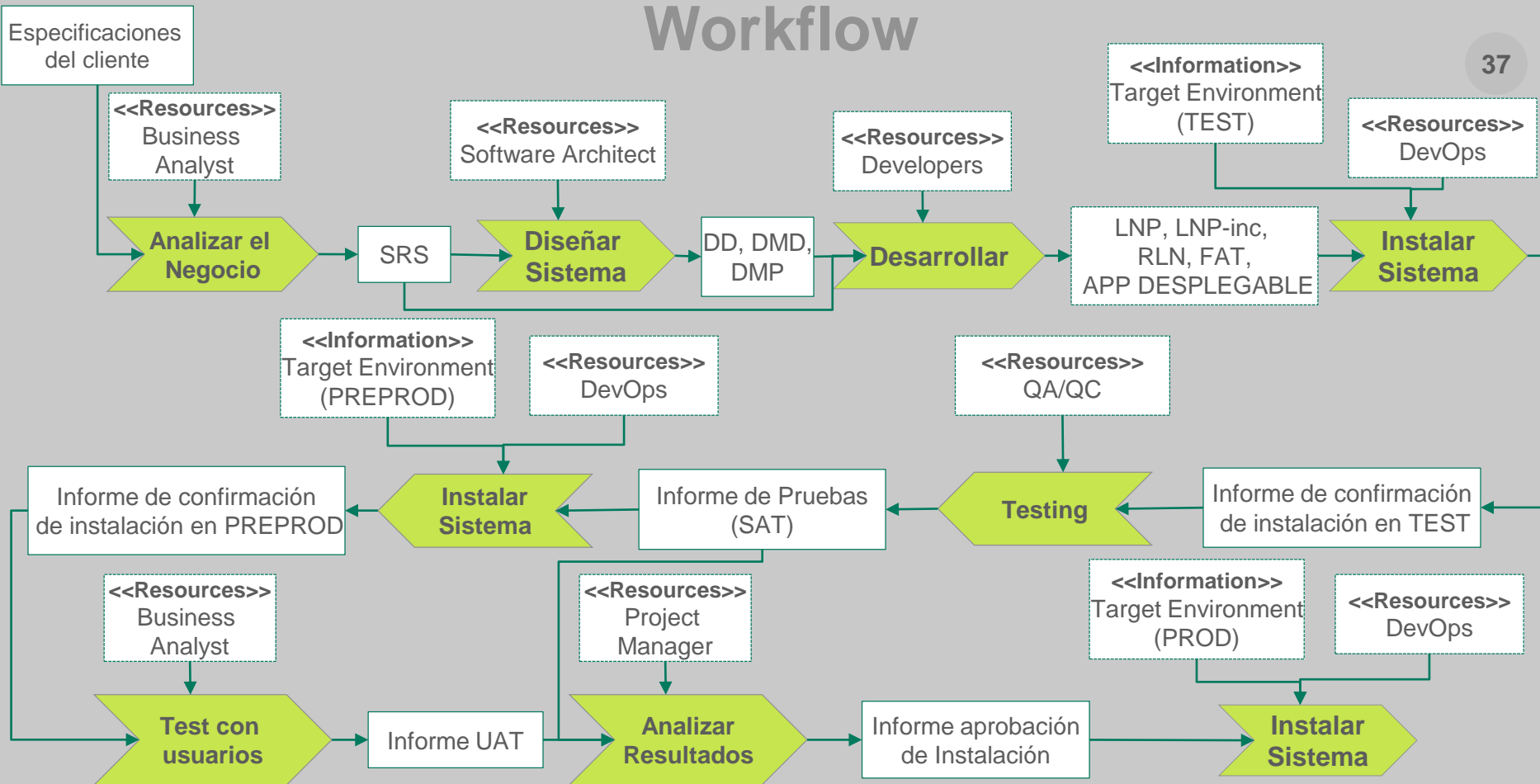


### Generar Informe de SAT

Se genera el informe de System Acceptance Tests



## 37



# Estimación proyectos Fix-Price

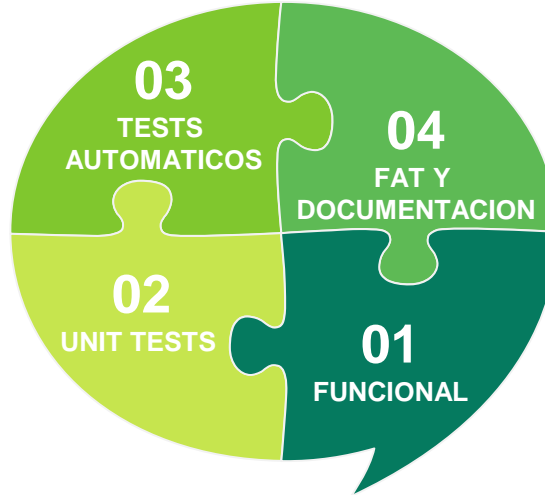
proceso como un todo

## Tests Automáticos

Se tiene en cuenta también los mds requeridos para la implementación de los Tests automáticos

## Tests Unitarios

Se añade a la estimación el tiempo necesitado por los desarrolladores para implementar los unit tests.



## FAT y Documentación

Por ultimo el Factory Acceptance Tests y toda la documentación requerida también se añade a la estimación temporal que se ofrece a los Project Managers

## Funcionalidad

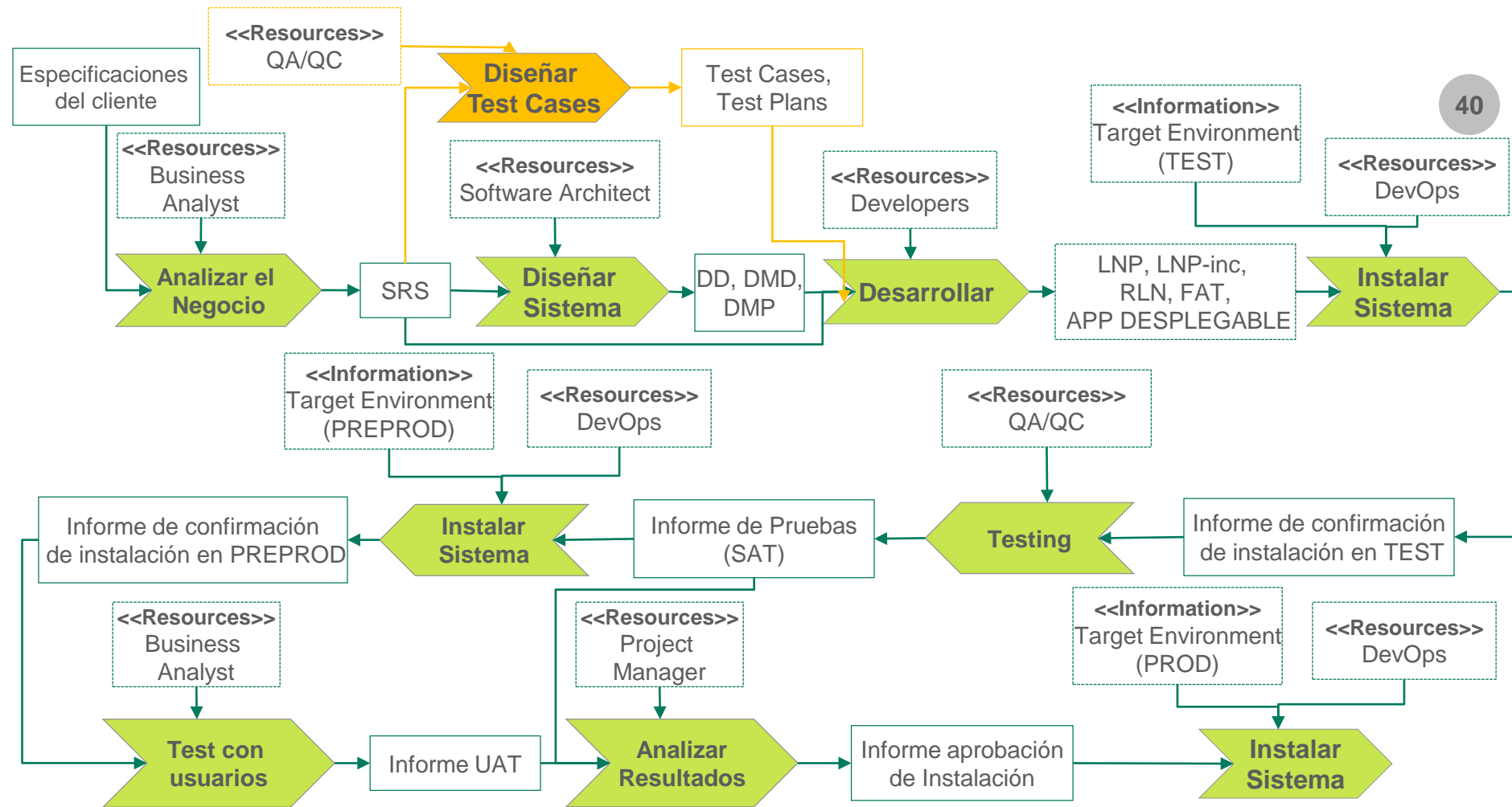
Se estima el tiempo que tardarían los desarrolladores en implementar las funcionalidades pedidas



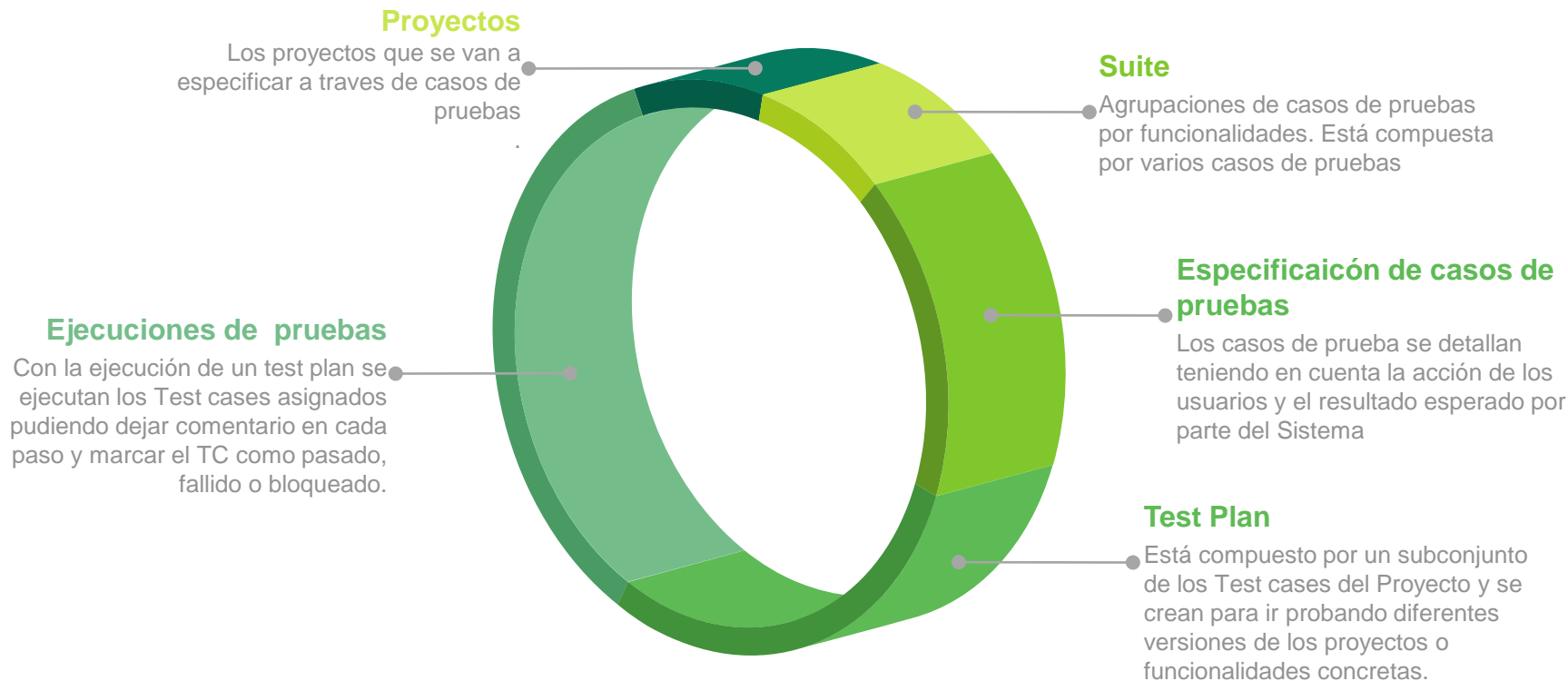


**GGTI**

Testing





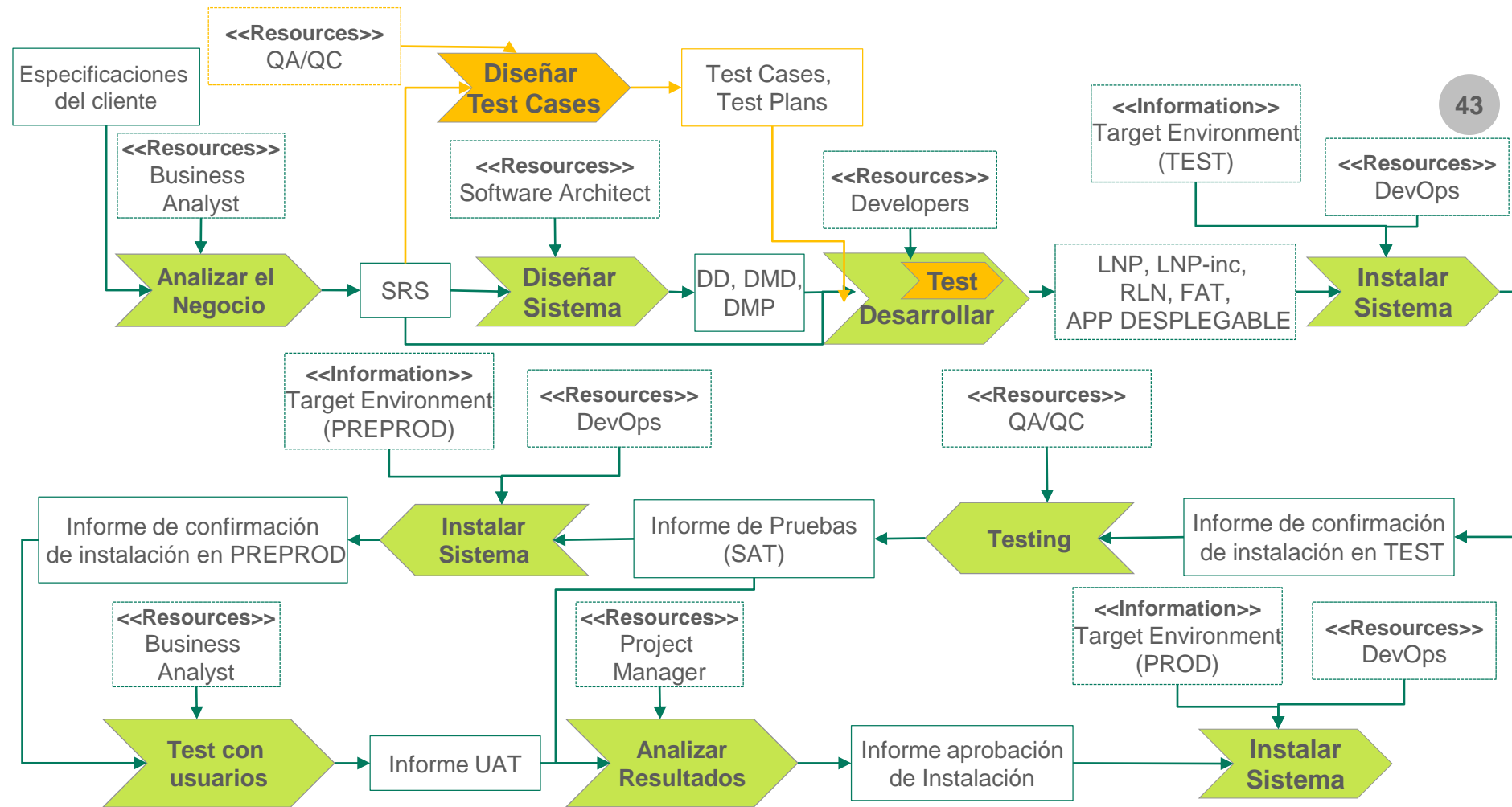


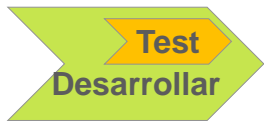
# Tareas de definición de Test Cases

Testlink

42



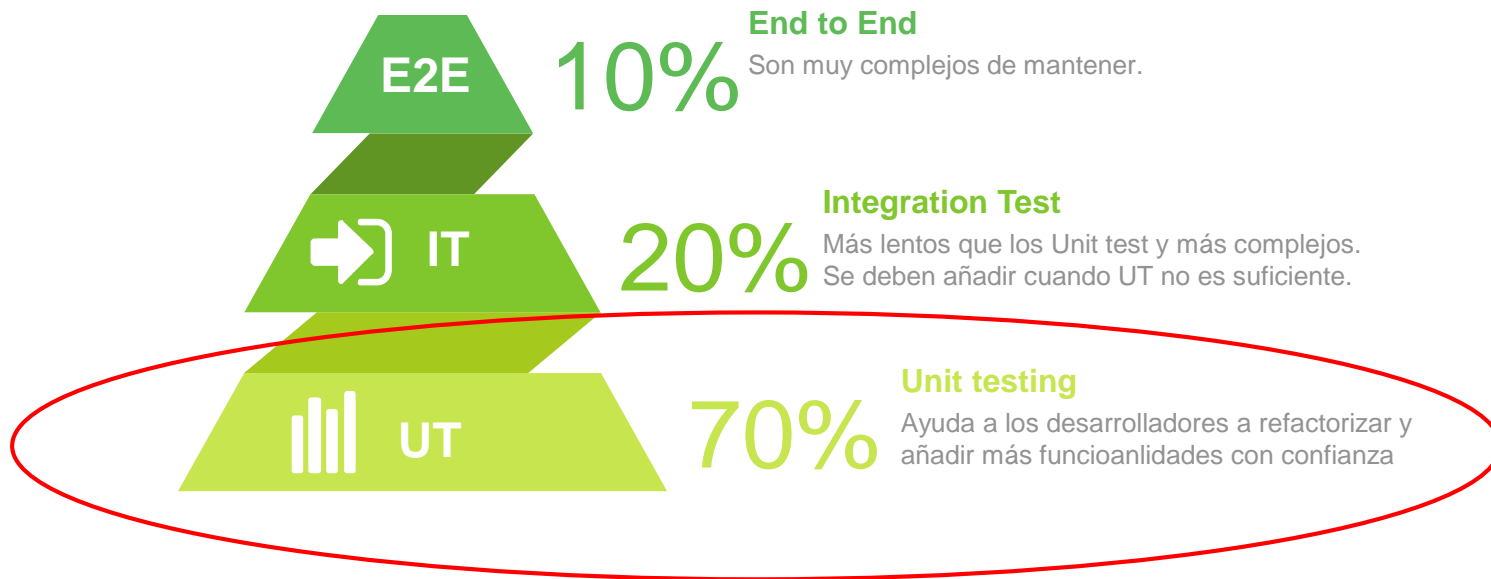




# Pirámide de Pruebas

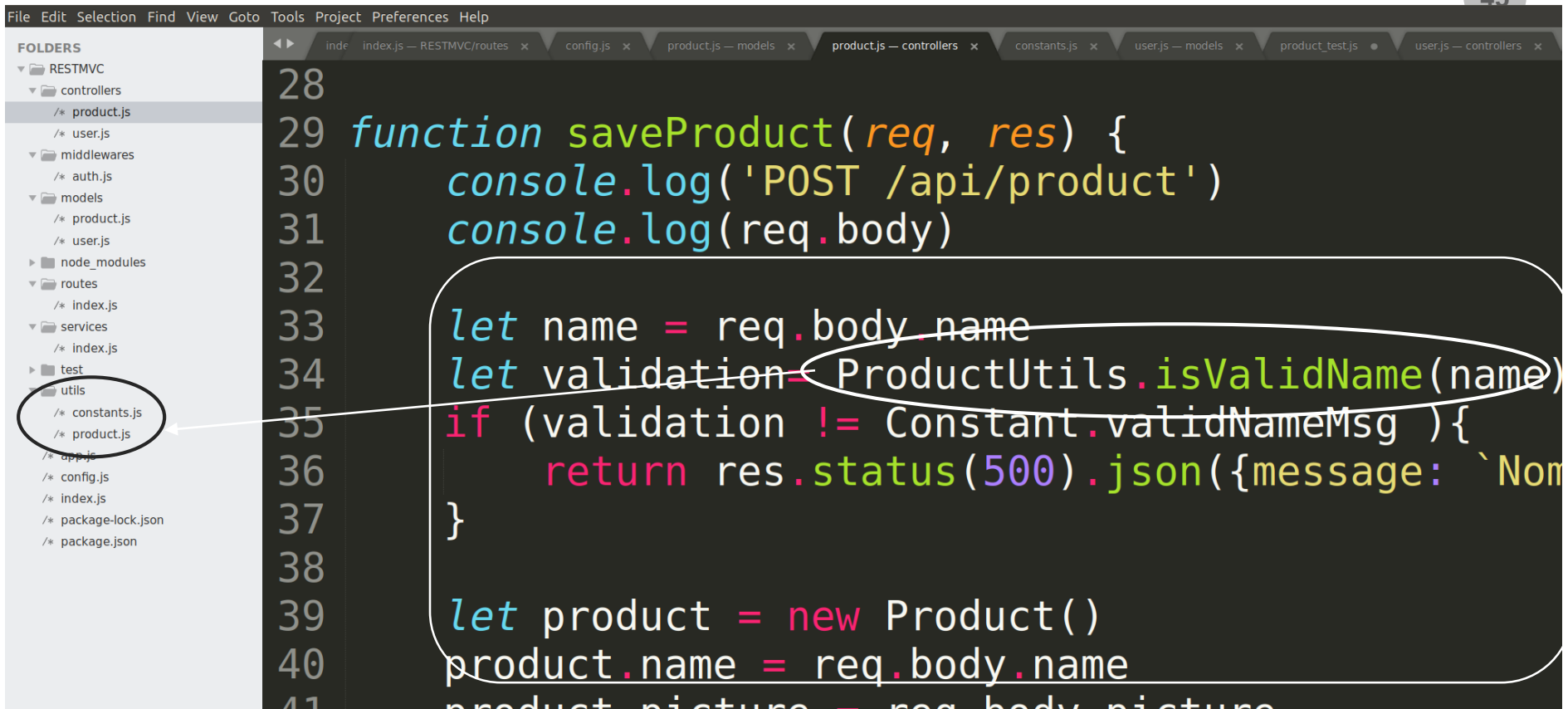
Propuesta de google

44



# Ejemplo para nodejs-express

45



```
28
29 function saveProduct(req, res) {
30   console.log('POST /api/product')
31   console.log(req.body)
32
33   let name = req.body.name
34   let validation = ProductUtils.isValidName(name)
35   if (validation !== Constant.validNameMsg) {
36     return res.status(500).json({message: `Nom
37   }
38
39   let product = new Product()
40   product.name = req.body.name
41   product.picture = req.body.picture
```

# Ejemplo para nodejs-express

46

```
'use strict'

const Constant = require('./constants')

function isValidName(name){
  let result

  //Checking lenght
  result = (name.length > Constant.minLength)
    && (name.length < Constant.maxLength)
  if (!result) return Constant.invalidNameLengthMsg

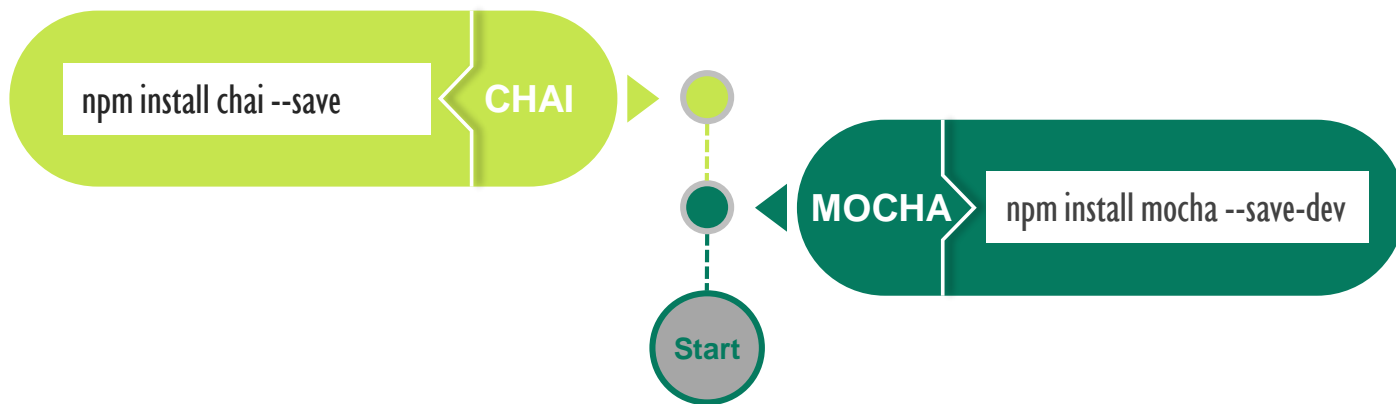
  //Checking prefix
  result = name.startsWith(Constant.productPrefix)
  if (!result) return Constant.invalidNamePrefixMsg

  return Constant.validNameMsg
}

module.exports = {isValidName}
```

# Qué se necesita para hacer unit testing nodejs

47



# Ejemplo para nodejs-express

package.json

48

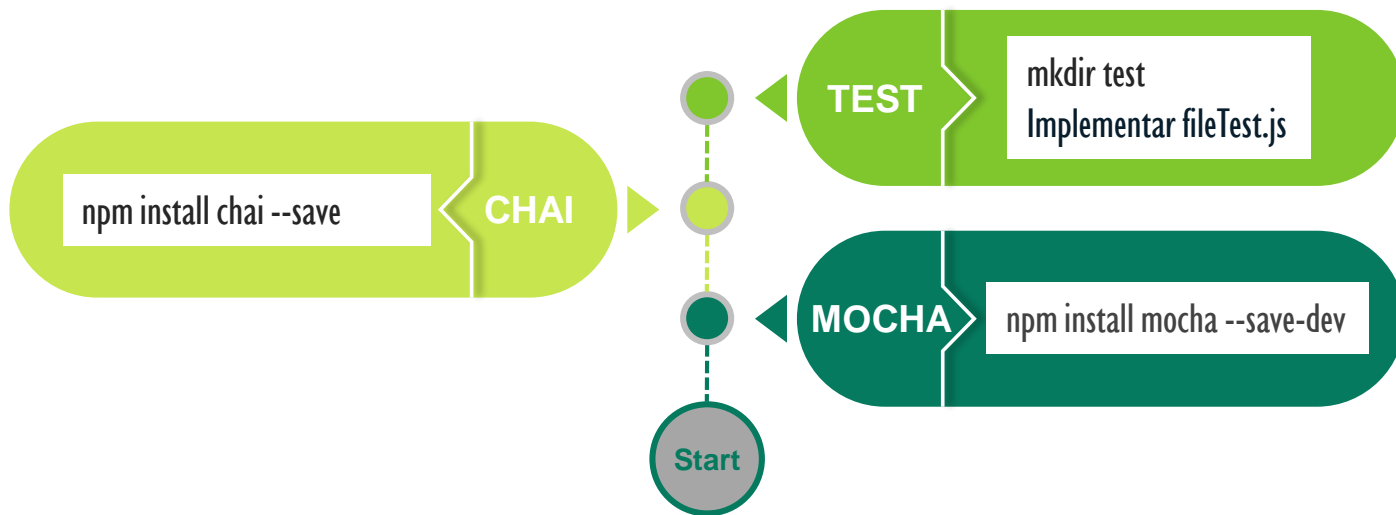
```
"dependencies": {  
  "chai": "^4.2.0",
```

```
"devDependencies": {  
  "mocha": "^6.0.2",
```



# Qué se necesita para hacer unit testing nodejs

49



# Ejemplo para nodejs-express

Product\_test.js

50

▼ test

/\* product\_test.js

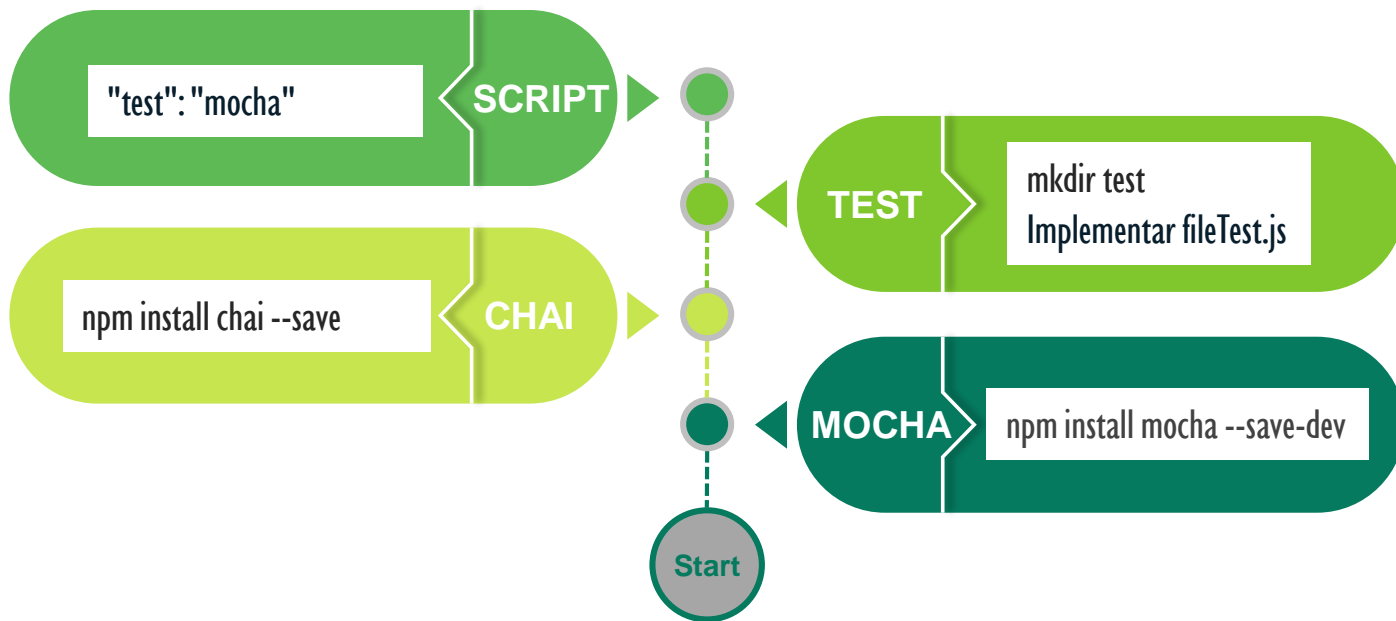
```
const expect = require("chai").expect;
const assert = require("chai").assert;
const should = require("chai").should();

const productValidator = require("../utils/product");

describe("Product validation tests using different assert from CHAI module: ",
  function() {
    describe("Check isValidName Function: ", function() {
      it("Check a valid product name using expect: ", function() {
        result = productValidator.isValidName('P44FR_PROD1');
        expect(result).to.equal('Valid name');
        //other things that can be checked
        expect(result).to.be.a('string');
        expect(result).to.have.lengthOf(10);
      });
    });
  });
```

# Qué se necesita para hacer unit testing nodejs

51



# Ejemplo para nodejs-express

52

```
"scripts": {  
  "test": "mocha test/*_test.js",
```

package.json

```
ilorenzo@ubuntu:~/SD/node/RESTMVC$ npm test
```

```
> restmvc@1.0.0 test /home/ilorenzo/SD/node/RESTMVC  
> mocha test/*_test.js
```

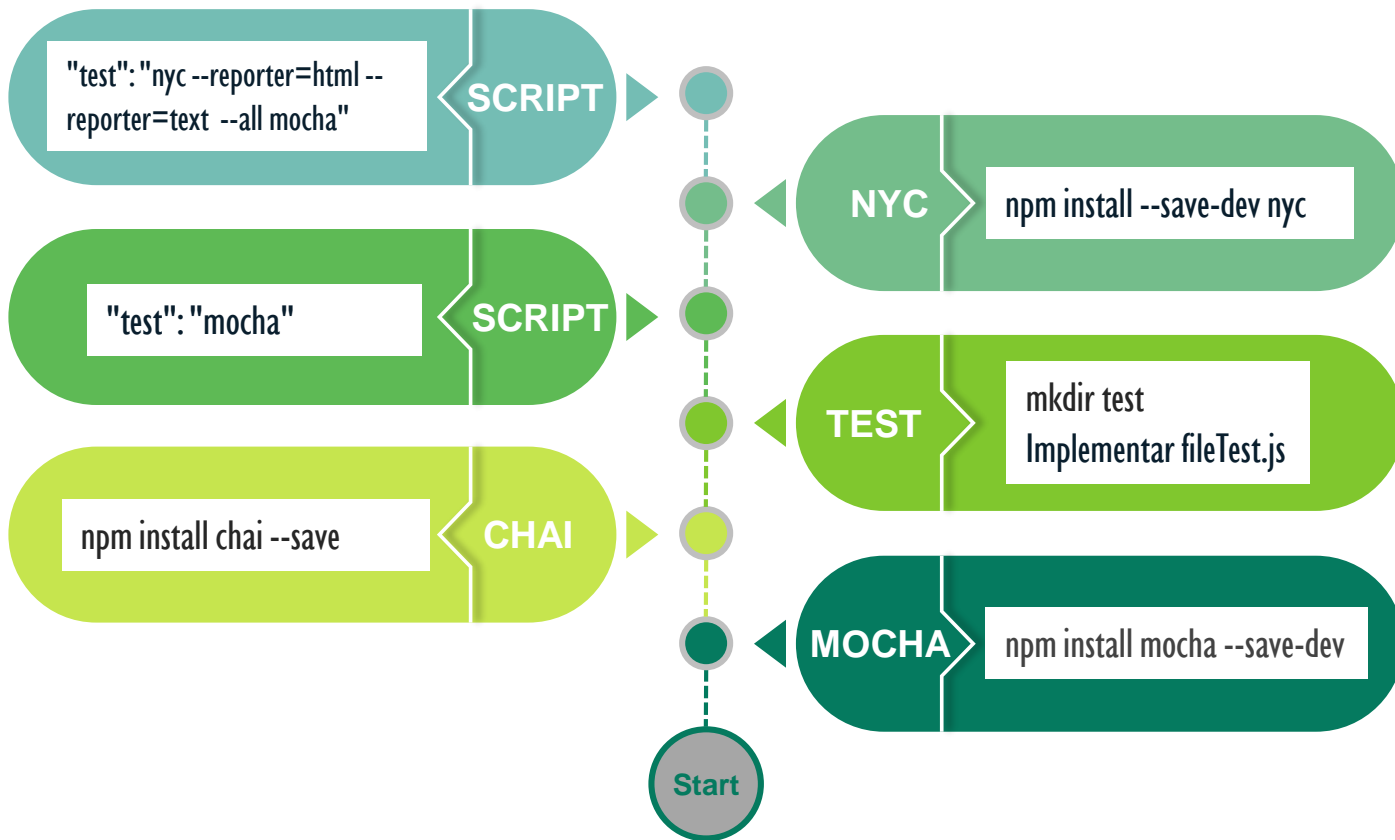
Product validation tests using different assert from CHAI module:  
Check isValidName Function:

- ✓ Check a valid product name using expect:
- ✓ Check min lenght of the product name using assert:
- ✓ Check max lenght of the product name using should:
- ✓ Check the prefix of the product name using assert:

```
4 passing (25ms)
```

# Qué se necesita para hacer unit testing nodejs

53



# Ejemplo para nodejs-express

## package.json

54

```
"scripts": {  
  "test": "nyc --reporter=html --reporter=text --all mocha",  
}
```

```
ilorenzo@ubuntu:~/SD/node/RESTMVC$ npm test
```

coverage

- RESTMVC
  - /\* base.css
  - /\* block-navigation.js
  - <> index.html
  - /\* prettify.css
  - /\* prettify.js
  - sort-arrow-sprite.png
  - /\* sorter.js

```
> restmvc@1.0.0 test /home/ilorenzo/SD/node/RESTMVC  
> nyc --reporter=html --reporter=text --all mocha
```

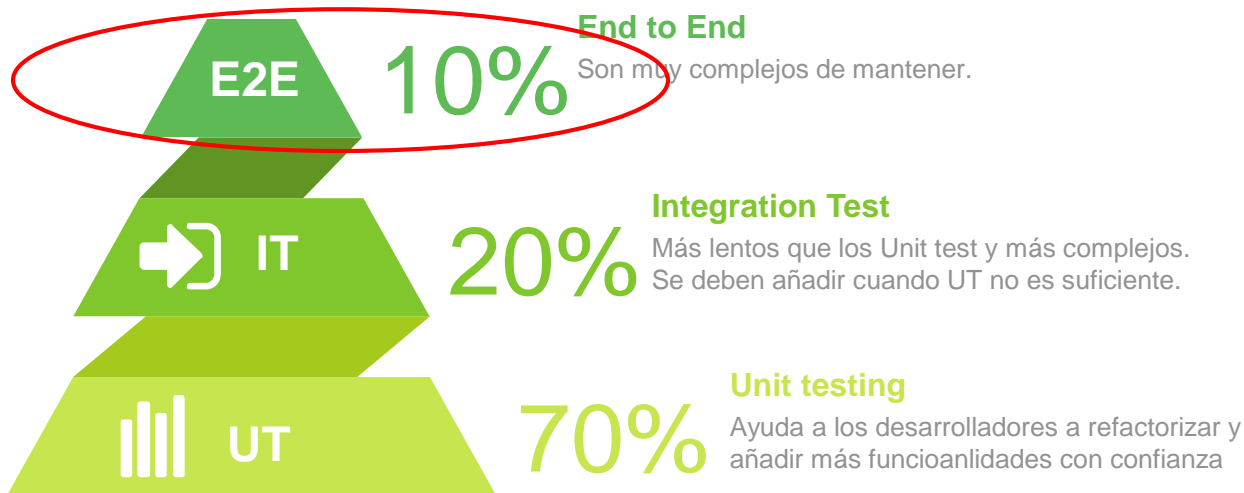
Product validation tests using different assert from CHAI module:

Check isValidName Function:

- ✓ Check a valid product name using expect:
- ✓ Check min lenght of the product name using assert:
- ✓ Check max lenght of the product name using should:
- ✓ Check the prefix of the product name using assert:

4 passing (83ms)

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	10.19	13.04	3.57	9.93	
RESTMVC	0	0	0	0	
app.js	0	100	100	0	... ,7,10,11,12,14
config.js	0	0	100	0	1
index.js	0	0	0	0	... 7,8,9,11,13,14
RESTMVC/controllers	0	0	0	0	
product.js	0	0	0	0	... 71,73,74,75,80
user.js	0	0	0	0	... 23,24,26,28,35
RESTMVC/middlewares	0	0	0	0	
auth.js	0	0	0	0	... 12,15,16,19,23
RESTMVC/models	0	0	0	0	
product.js	0	100	100	0	3,4,6,15
user.js	0	0	0	0	... 22,23,25,26,31
RESTMVC/routes	0	100	0	0	
index.js	0	100	0	0	... 15,16,17,18,21
RESTMVC/services	0	0	0	0	
index.js	0	0	0	0	... 23,28,30,37,40
RESTMVC/utlis	100	100	100	100	
constants.js	100	100	100	100	
product.js	100	100	100	100	



# End to End Test

## Selenium

56

```
"dependencies": {  
  "selenium-webdriver": "^4.0.0-alpha.1",  
  "chromedriver": "^2.46.0",  
}
```

`npm install -S selenium-webdriver`

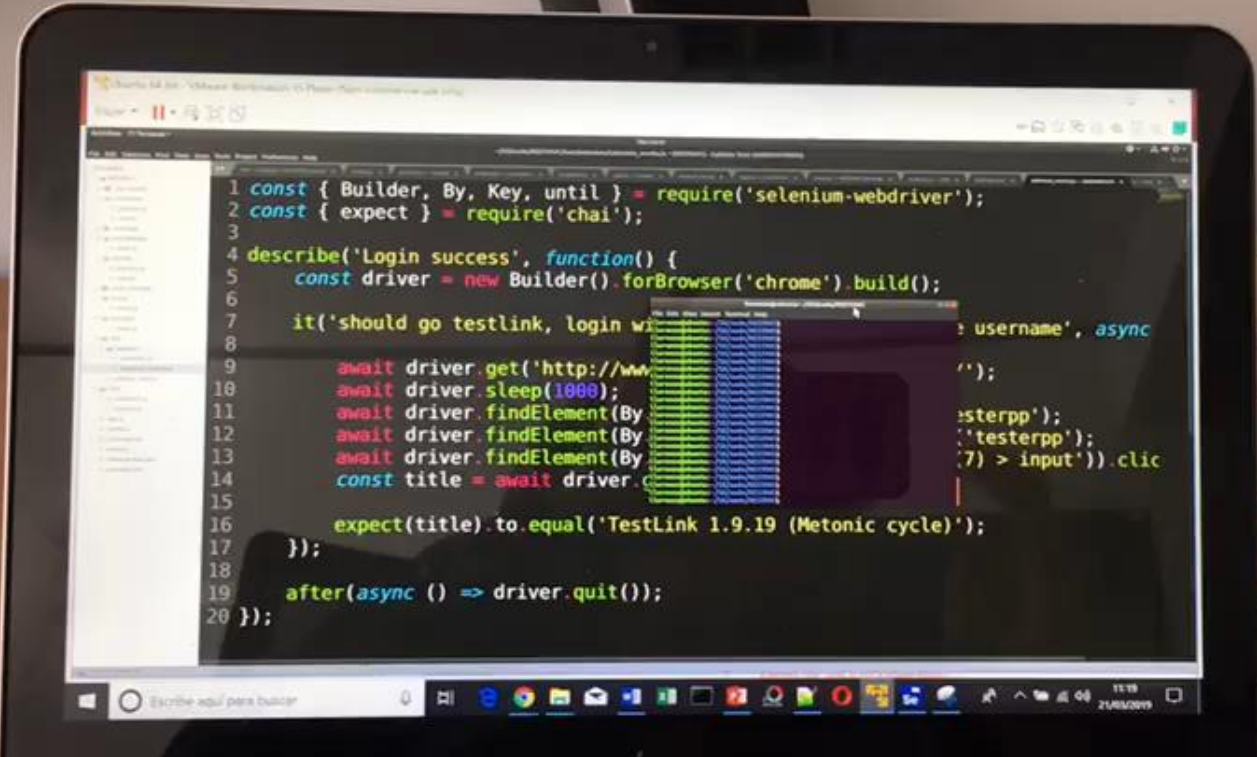
`npm install --save chromedriver`

```
export  
PATH="/home/ilorenzo/SD/node/RESTMVC/node_modules/chromedriver/bin:$PATH"
```



# End to End Test

57



```
1 const { Builder, By, Key, until } = require('selenium-webdriver');
2 const { expect } = require('chai');
3
4 describe('Login success', function() {
5   const driver = new Builder().forBrowser('chrome').build();
6
7   it('should go testlink, login with username', async function() {
8
9     await driver.get('http://www.testlink.com');
10    await driver.sleep(1000);
11    await driver.findElement(By.id('username')).sendKeys('testerpp');
12    await driver.findElement(By.id('password')).sendKeys('testerpp');
13    await driver.findElement(By.id('login')).click();
14    const title = await driver.getTitle();
15
16    expect(title).to.equal('TestLink 1.9.19 (Metonic cycle)');
17  });
18
19  after(async () => driver.quit());
20 });
```

# Implementación test selenium

mocha

58

```
const { Builder, By, Key, until } = require('selenium-webdriver');
const { expect } = require('chai');

describe('Login success', function() {
  const driver = new Builder().forBrowser('chrome').build();

  it('should go testlink, login with testpp user and check the username',
    async () => {

      await driver.get('http://www.multimedia-sma.es/testlink/');
      await driver.sleep(1000);
      await driver.findElement(By.id('tl_login')).sendKeys('testerpp');
      await driver.findElement(By.id('tl_password')).sendKeys('testerpp');
      await driver.findElement(By.css('.form__field:nth-child(7) > input')).click();
      const title = await driver.getTitle();

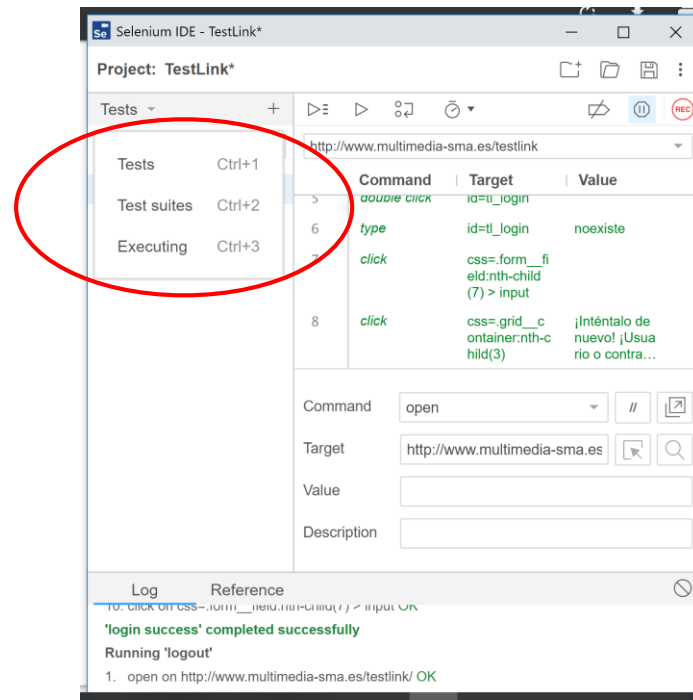
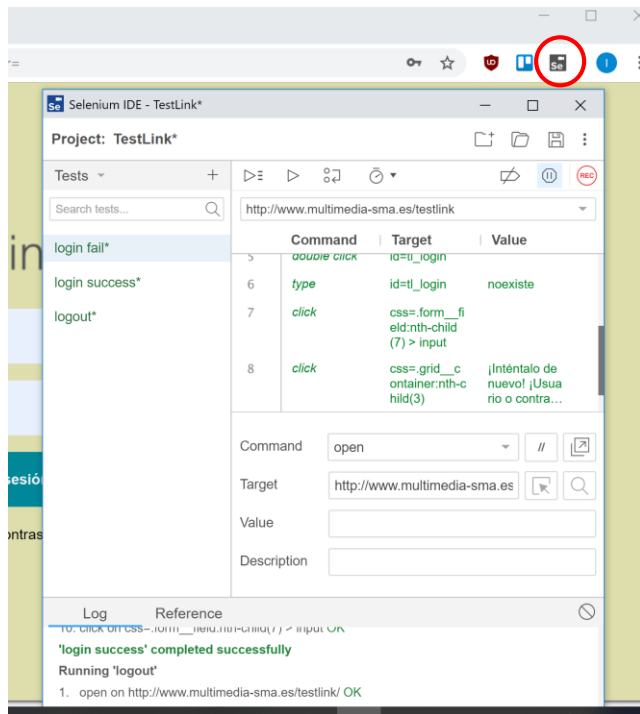
      expect(title).to.equal('TestLink 1.9.19 (Metonic cycle)');
    });

  after(async () => driver.quit());
});
```

# Selenium plugin

## Chrome

59





# Questions Answers

PREGUNTAS

.....