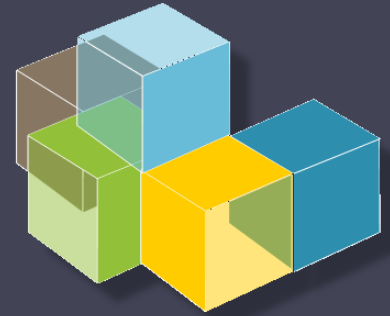


Universidad de Oviedo



Escuela de
Ingeniería
Informática



ARQUITECTURA
DEL SOFTWARE

Arquitectura del Software

Lab. 06

TDD: Desarrollo basado en pruebas

Cobertura de código (Codecov)

Integración continua (Travis)

Herramientas estáticas de chequeo de código (Codacy)

2019-20

Jose Emilio Labra Gayo

Pablo González

Irene Cid

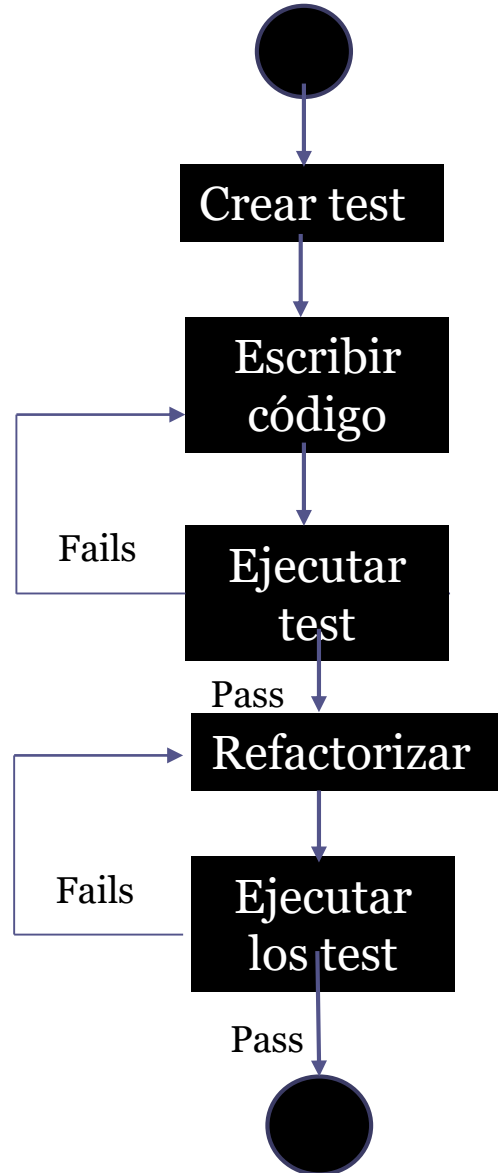
Hugo Lebrede

TDD

- Proceso de desarrollo de software en el que los requisitos son convertidos en casos de prueba específicos.
- Opuesto al desarrollo del software que permite que se añada software no probado.
- Técnica creada por Kent Beck

TDD

- Fases:



Codecov

- Herramienta de cobertura de código.
- Cobertura código: Medida para expresar que líneas de código han sido ejecutado por un test suite.
- Codecov utiliza los siguientes términos por cada línea:
 - Hit: Indica que la línea fue ejecutado
 - Partial: Indica que no fue ejecutado al completo. Por ejemplo condiciones
 - Miss: No fue ejecutado

Codecov

- Ratio de cobertura lo calcula con la siguiente formula:

$$\text{hits} / (\text{hits} + \text{misses} + \text{partials})$$

Tras los test genera un fichero que le permite realizar un análisis

- <https://codecov.io/gh/arquisoft/viade> ???

TDD - Fase 1 Creamos un test

```
test('check email button activation', async () => {  
  const correctValues = { email: "irene@gmail.com", remail: "irene@gmail.com" };  
  
  //mostramos el formulario  
  const { getByLabelText, getByTestId, getByText, container } = render(<EmailForm />);  
  
  //Recuperamos los inputs  
  const input_email = getByLabelText('email-input');  
  const input_remail = getByLabelText('remail-input');  
  
  //Ejeuctamos un evento de cambio en el input email  
  fireEvent.change(input_email, { target: { value: correctValues.email } });  
  expect(getByText(/Submit/i).closest('button')).toHaveAttribute('disabled');  
  
  //Ejeuctamos un evento de cambio en el input remail  
  fireEvent.change(input_remail, { target: { value: correctValues.remail } });  
  expect(getByText(/Submit/i).closest('button')).not.toHaveAttribute('disabled');  
});
```

EmailForm.test.js

Vamos a crear un formulario en el que el usuario debe introducir su email y confirmarlo en otro campo (remail). Solo cuando ambos emails tengan el formato correcto y sean iguales se activa el botón Submit

TDD - Fase 2 - Escribimos código

```
export default class EmailForm extends Component {
  constructor(props) {
    super(props);
    this.state = { email: '', remail: '', submitDisable: true };
    this.handleChange = this.handleChange.bind(this);
  }
  handleChange(event) {
    let finalState = Object.assign(this.state, { [event.target.name]: event.target.value });
    this.setState({ [event.target.name]: event.target.value });
    if (finalState.email === finalState.remail) {
      if (finalState.email.match(/^(?![\w.%+-]+)([a-zA-Z0-9_-]+)([a-zA-Z0-9_-]{2,})$/i)) {
        this.setState({ submitDisable: false });
      }
    }
    else {
      this.setState({ submitDisable: true });
    }
  }
  else {
    this.setState({ submitDisable: true });
  }
}

render() {
  return (
    <Form>
      <Form.Group controlId="formGroupEmail">
        <Form.Label>Email address</Form.Label>
        <Form.Control type="text" name="email" aria-label="email-input" placeholder="Enter email" onChange={this.handleChange}>
      </Form.Group>
      <Form.Group controlId="formGroupRemail">
        <Form.Label>Retype Email address</Form.Label>
        <Form.Control aria-label="remail-input" type="text" name="remail" placeholder="Retype mail" onChange={this.handleChange}>
      </Form.Group>
      <Button aria-label="boton-enviar" disabled={this.state.submitDisable}>Submit</Button>
    </Form>
  );
}
```

BigFoot

Un lugar para crear nuestras propias rutas

Learn more

Mis rutas

Accede y compartes tus rutas.

A través de fichero

A través de interface

Email address

irene@gmail.com

Retype Email address

irene@gmail.com

Submit

TDD - Fase 3 - Ejecutamos nuestro test

- Desde terminal ejecutamos
> `npm test -p EmailForm.test.js`
- Esto lanza
> `react-scripts test --coverage`

```
PASS src/EmailForm.test.js
  ✓ check email submit activation (73ms)
  ✓ check mal formado mail (21ms)
```

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	100	100	100	100	
EmailForm.js	100	100	100	100	

```
Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        4.737s
Ran all test suites matching /EmailForm\.test\.js/i.
```


TDD - Fase 5 - Ejecutamos todos los test

- Desde terminal ejecutamos
`> npm test -a`

- Esto lanza

```
PASS src/App.test.js
PASS src/EmailForm.test.js
```

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	100	100	100	100	
EmailForm.js	100	100	100	100	

```
Test Suites: 2 passed, 2 total
Tests:       3 passed, 3 total
Snapshots:  0 total
Time:        4.798s
Ran all test suites.
```

- Referencia de librería :<https://github.com/testing-library/react-testing-library>

Integración continua

- Módulo informático que consiste en hacer integraciones automáticas de un proyecto.
- Todas las tareas para construir el software se ejecutan con cada nueva incorporación de código.

Integración continua

- Ejemplos:
 - Jenkins
 - Pipeline
 - Hudson
 - Apache Continuum
 - Travis

Integración continua

- Usos comunes:
 - Mantenimiento del código en un repositorio
 - Automatizar la construcción
 - Cada commit es un build
 - Construcción rápida
 - Ejecutar casos de prueba en clon de entorno de producción
 - Hacer visibles los resultados del último build.

Travis

- Servicio de integración continua para proyectos almacenados en GitHub
- Gratis de usar para proyectos de código abierto
- Se configura a través de un fichero YAML llamado *.travis.yml* que se coloca en la raíz del proyecto.

Travis

- .travis.yml:
 - Especifica el lenguaje de programación
 - El entorno de construcción
 - El entorno de ejecución de test
 - Otros parámetros

```
language: node_js
node_js:
  - 12.14.0
cache:
  directories:
    - node_modules
before_install:
  - sudo apt-get update
  - sudo apt-get -y install ruby openjdk-8-jre
  - sudo gem install asciidoctor asciidoctor-diagram
script:
  - npm install -g codecov
  - npm test && codecov
  - npm run build
  - npm run docs
deploy:
  provider: pages
  skip_cleanup: true
  github_token: $github_token
  local_dir: build
on:
  branch: master
```

Travis

- dist: Distro usada para Linux. En macOS y Windows tiene la etiqueta os
- sudo: If sudo is activated or not
- language: Programming language
- node_js: Version of nodejs.
- addons: Extensiones usadas
- Directories: Donde se encuentran las dependencias
- Install: Comando para desplegar el proyecto

Travis

- Script: Comandos ejecutados tras desplegarse el proyecto.
- after_success: Si no ha ocurrido ningún error se ejecutarán los comandos establecidos.

Análisis estático de código

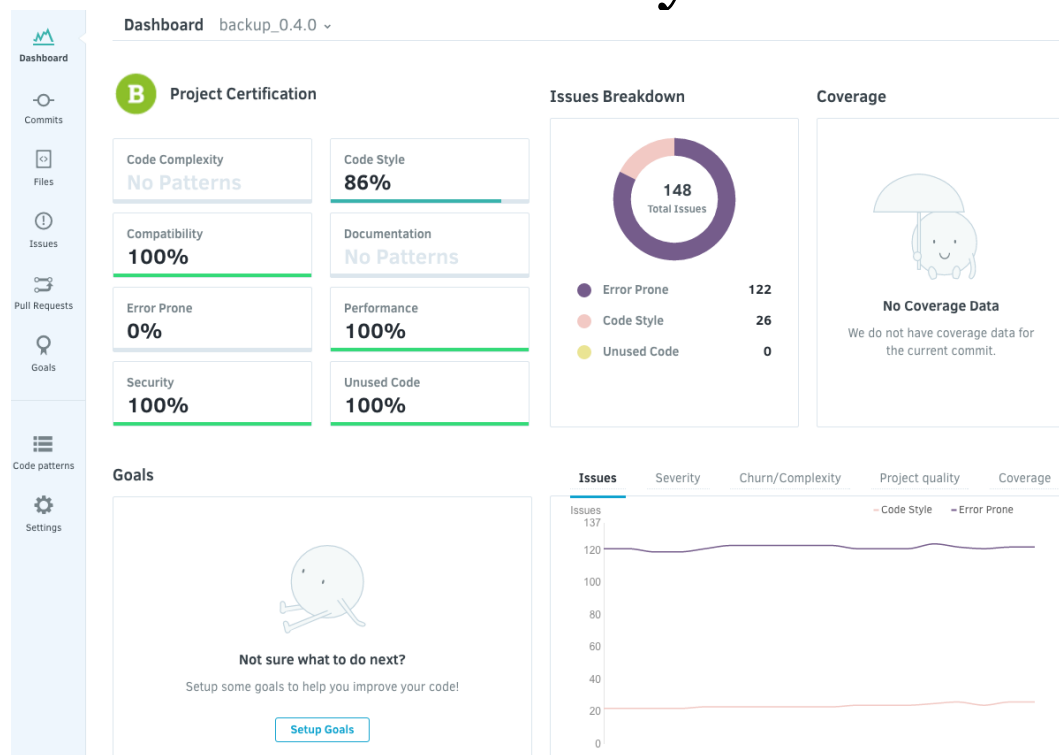
- Analizan el código sin compilarlo.
- Detectan bugs, code smells, vulnerabilidades del sistema, etc
- Útiles para controlar la calidad del código.
- Si el código no cumple la calidad requerida se puede bloquear el commit.

Codacy

- Herramienta de análisis estático del código.
- Se necesita:
 - Servidor git como GitHub
 - Acceso al repositorio
 - Lenguaje aceptado.
- El proyecto se importa a Codacy para que pueda ser analizado

Codacy

- Tras el análisis inicial Codacy envía una notificación cuando haya finalizado



Codacy

- En el Project Dashboard se pueden encontrar dos secciones principales: la vista del Proyecto de una rama en concreto o la rama principal
- Para cada rama hay las siguientes secciones:
 - **Quality evolution**
 - **Issues breakdown**
 - **Coverage status**
 - **Hotspots**
 - **Logs**
 - **Pull requests status**

Codacy: Project certification and Quality evolution

B Project certification

Quality evolution

Last 7 days

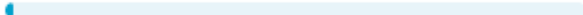



Last 31 days



Codacy: Issues breakdown

Issues breakdown

7017 total issues

Category		Total
Security		104
Error Prone		616
Code Style		6297
Compatibility		0
Unused Code		0
Performance		0

[See all issues](#)

Codacy: Coverage status

Coverage

95% LoC covered



■ Quality settings: 60% coverage

Files without coverage ⓘ 13

Files not up to standards ⓘ 0

Files up to standards ⓘ 5

[See all files](#)

Codacy

- **Security:** problemas de seguridad, vulnerabilidades potenciales, dependencias no seguras.
- **Error Prone:** malas practices / patrones que causas fallos en el Código o son propensos a ello
- **Code Style:** relacionado con el estilo del código, longitud de la linea, espacios, tabulados, etc
- **Compatibility:** Identifica código que tiene problemas con sistemas antiguos o problema con varias plataformas
- **Unused Code:** Código que no se usa
- **Performance:** Código ineficiente

Codacy: Files

Files master ▾

Search file

GRADE ▲	FILENAME ▲	ISSUES ▾	DUPLICATION ▲	COMPLEXITY ▲	COVERAGE ▲
A	tests/Codacy/Coverage/Parser/CloverParserTest.php	1	0	4	-
A	src/Codacy/Coverage/Parser/CloverParser.php	1	0	16	94%
B	src/Codacy/Coverage/Application.php	0	0	1	0%
A	tests/Codacy/Coverage/Parser/ParserTest.php	0	0	1	-
A	tests/Codacy/Coverage/Util/GitClientTest.php	0	0	1	-
A	tests/Codacy/Coverage/Parser/PhpUnitXmlParserTest.php	0	0	2	-
B	src/Codacy/Coverage/Command/Phpunit.php	0	0	3	0%
A	src/Codacy/Coverage/Util/GitClient.php	0	0	3	67%
A	src/Codacy/Coverage/Util/CodacyApiClient.php	0	0	4	-

Codacy: File detail

E

squbs-unicomplex/src/main/scala/org/squbs/unicomplex/streaming/ServiceRegistry.scala

Ignore File

TIME TO FIX: 1 hour

[View on GitHub](#)

Size		Structure		Complexity		Duplication	
Lines of code:	273	Number of Classes:	8	Complexity:	26	Number of Clones:	13
Source lines of code:	194	sLoC / Class: ⓘ	24.25	Complexity / Class:	3.25	Duplicated lines of code:	134
Commented lines of code:	26	Number of Methods:	31	Complexity / Method:	0.84		
		sLoC / Method: ⓘ	6.26	Churn:	19		

Fin