



**Escuela Superior
de Ingeniería y Tecnología**
Universidad de La Laguna

Sistemas y Tecnologías Web

Proyecto: Iteración #4

Continuous Integration / GitHub Actions

Jacobo Labrador González

[\(alu0101119663@ull.edu.es\)](mailto:alu0101119663@ull.edu.es)

Vlatko Jesús Marchán Sekulic

[\(alu0101321141@ull.edu.es\)](mailto:alu0101321141@ull.edu.es)

Tanausú Falcón Casanova

[\(alu0101320878@ull.edu.es\)](mailto:alu0101320878@ull.edu.es)



Índice

Introducción	2
Integración Continua	2
Despliegue/Delivery Continuo	2
Tareas finalizadas.	3
1. Setup del sistema de integración continua.	4
2. Configurar .github/workflows	5
3. Github Actions	7
Tareas a realizar en la iteración 5	10
Referencias	10



Introducción

En esta iteración se van a asignar las tareas para realizar la integración continua (CI) de los tests previamente hechos en iteraciones anteriores.

- Test con Jest/Supertest y Jest/React Testing Library
- Configurar `.github/workflows`.
- Setup del sistema de integración continua.

Se ha de mencionar que la utilización de Travis CI ha quedado descartada por recomendación del profesorado.

Integración Continua

La integración continua consiste en la práctica en ingeniería del software de hacer integraciones automáticas de un proyecto lo más a menudo posible, para así encontrar y detectar fallos en un menor intervalo de tiempo. Se entiende por integración continua la compilación y ejecución de las pruebas de todo un proyecto.

Para la realización de Integración Continua se implementa las herramientas proporcionas a través de Github Actions.

Despliegue/Delivery Continuo

En cuanto a CD (Continuous Delivery/Deployment) se entiende como dos acrónimos diferentes.

- Entrega continua

La entrega continua es una extensión de la integración continua, ya que implementa automáticamente todos los cambios de código en un entorno de pruebas/producción tras la fase de compilación.

Esto significa que, además de las pruebas automatizadas, cuenta con un proceso de publicación automatizado y se puede implementar la aplicación en cualquier momento.



- Despliegue continuo

La implementación continua va un paso más allá que la entrega continua. Mediante esta práctica, los cambios que pasan por todas las fases de la canalización de producción se publican para los clientes. No existe una intervención humana y sólo una prueba sin éxito evitará implementar un nuevo cambio en la producción.

Se debe remarcar que el CD de la aplicación se realizará en una posterior iteración, junto al despliegue del backend en servicios como Heroku y el frontend.

Tareas finalizadas.

★ =	Setup del sistema de integración continua (AL, AL, VL)	Deliver	<input type="checkbox"/>
★ ≡	Test con Travis CI (AL, AL, VL)	Deliver	<input type="checkbox"/>
★ =	Configurar .travis.yml / .github/workflows / config.yml (AL, AL, VL)	Deliver	<input type="checkbox"/>

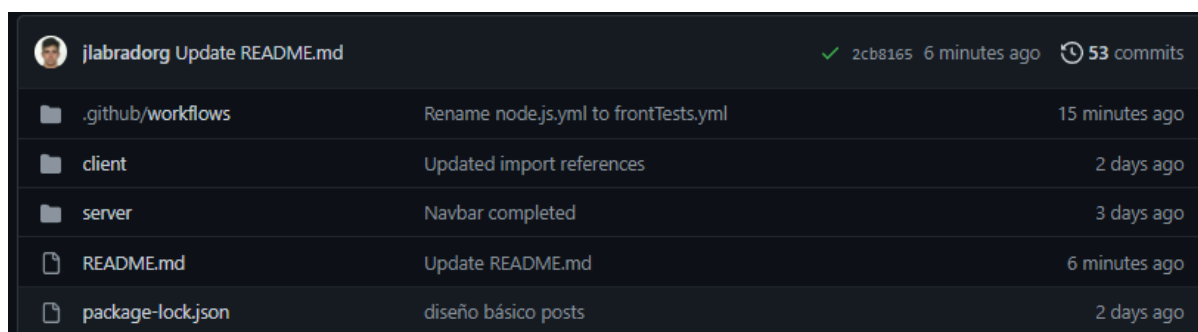
Figura 1: Tareas finalizadas en Pivotal Tracker.



1. Setup del sistema de integración continua.

Para la creación del setup del sistema de integración continua, se ha de autorizar ese CI como aplicación externa que pueda clonar los repositorios y configurar un hook para que con cada push en GitHub se ejecuten los tests. Para ello, se ha utilizado las propias GitHub Actions.

Dentro de la carpeta `.github/workflows` se aloja el fichero de setup para los test tanto del frontend como del backend. Estos workflows deben de estar en la raíz del proyecto. Por tanto, la estructura del repositorio quedaría de la siguiente manera:



jlabradorg Update README.md		✓ 2cb8165 6 minutes ago 53 commits
📁 .github/workflows	Rename node.js.yml to frontTests.yml	15 minutes ago
📁 client	Updated import references	2 days ago
📁 server	Navbar completed	3 days ago
📄 README.md	Update README.md	6 minutes ago
📄 package-lock.json	diseño básico posts	2 days ago

Figura 2: Estado del repositorio actual.



2. Configurar .github/workflows

El primer fichero para la configuración del sistema de integración continua es el fichero para los tests del frontend. En éste, se define el nombre de la acción, sobre qué ramas se ejecutará, en qué versión de node se establecerán el entorno y los tests, y por último, se definirán los pasos a seguir para la instalación de dependencias y ejecución de los propios tests.

```
38 lines (30 sloc) | 981 Bytes
1 # This workflow will do a clean installation of node dependencies, cache/restore them, build the source code and run tests across different versions of node
2 # For more information see: https://docs.github.com/en/actions/automating-builds-and-tests/building-and-testing-nodejs
3
4 name: React Tests
5
6 on:
7   push:
8     branches: [ "main" ]
9   pull_request:
10    branches: [ "main" ]
11
12 jobs:
13   build:
14
15     runs-on: ubuntu-latest
16
17     strategy:
18       matrix:
19         node-version: [14.x, 16.x, 18.x, 19.x]
20         # See supported Node.js release schedule at https://nodejs.org/en/about/releases/
21
22     defaults:
23       run:
24         working-directory: ./client
25
26     env:
27       CI: false
28
29     steps:
30     - uses: actions/checkout@v3
31     - name: Use Node.js ${{ matrix.node-version }}
32       uses: actions/setup-node@v3
33       with:
34         node-version: ${{ matrix.node-version }}
35         cache: 'npm'
36     - run: npm ci
37     - run: npm run build --if-present
38     - run: npm test
```

Figura 3: Contenido del archivo frontTests.yml.



El segundo fichero tiene una estructura similar, con el añadido de la declaración de la variable de entorno para la database (sería recomendable establecer dicha variable en los secrets del repositorio). En este caso, se comprobarán los test de integración realizados tanto para el backend como la database.

```
38 lines (30 sloc) | 1.07 KB
1 # This workflow will do a clean installation of node dependencies, cache/restore them, build the source code and run tests across different versions of node
2 # For more information see: https://docs.github.com/en/actions/automating-builds-and-tests/building-and-testing-nodejs
3
4 name: React Tests BackEnd
5
6 on:
7   push:
8     branches: [ "main" ]
9   pull_request:
10     branches: [ "main" ]
11
12 jobs:
13   build:
14
15     runs-on: ubuntu-latest
16
17     strategy:
18       matrix:
19         node-version: [14.x, 16.x, 18.x, 19.x]
20         # See supported Node.js release schedule at https://nodejs.org/en/about/releases/
21
22     defaults:
23       run:
24         working-directory: ./server
25
26     env:
27       MONGO_DB_URI_TEST: mongodb+srv://SyTW:sytw123@cluster0.vrubeal.mongodb.net/test-app?retryWrites=true&w=majority
28
29     steps:
30       - uses: actions/checkout@v3
31       - name: Use Node.js ${{ matrix.node-version }}
32         uses: actions/setup-node@v3
33         with:
34           node-version: ${{ matrix.node-version }}
35           cache: 'npm'
36       - run: npm install
37       - run: npm run build --if-present
38       - run: npm test
```

Figura 4: Contenido del archivo backTest.yml.



3. Github Actions

Como se comentó anteriormente, debido a las recomendaciones del profesorado, se ha descartado el uso de Travis CI y se ha optado por la utilización de las GitHub Actions.

Dentro de la sección de GitHub Actions, se podrán comprobar los tests, que se ejecutarán por cada push o pull request que se realice a la rama main.

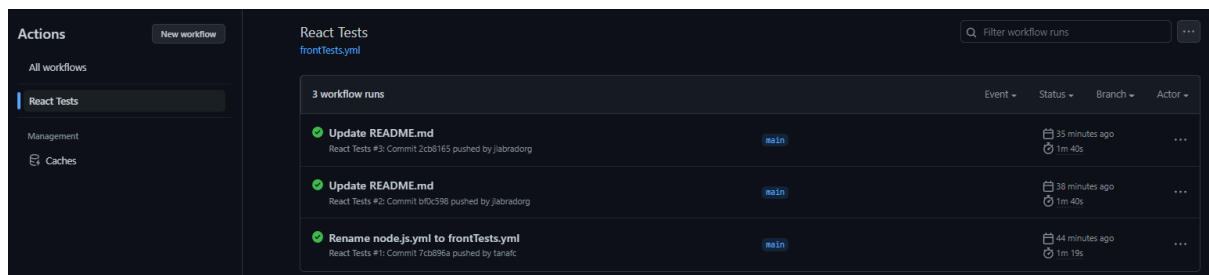


Figura 5: Resultado de la ejecución de los test del Frontend.

Si se hace accede a un trabajo en particular, se encuentran los tests completados por las diferentes builds.

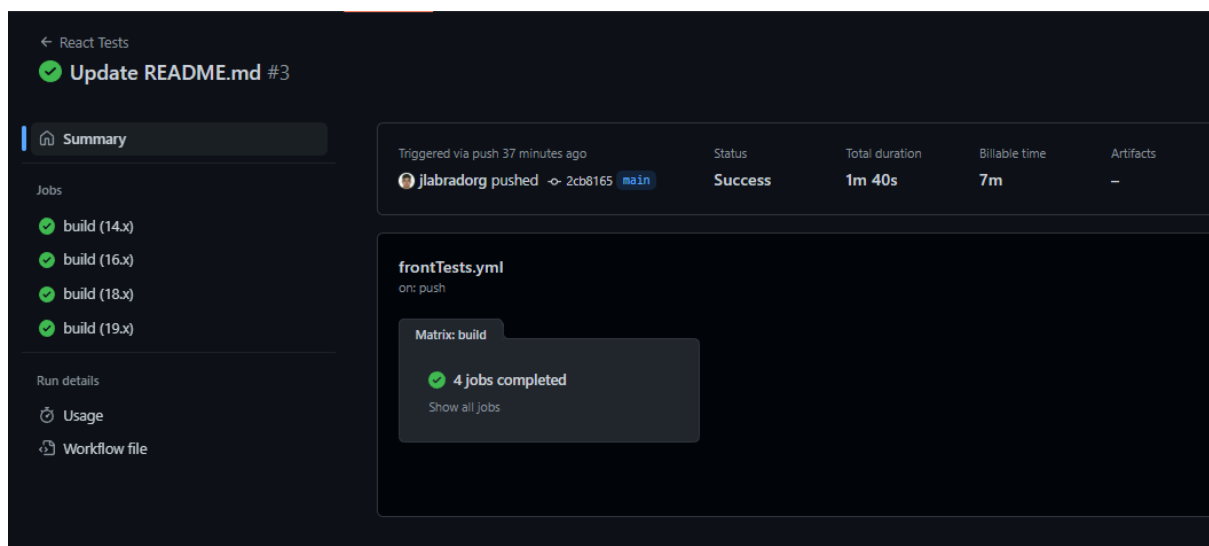


Figura 6: Resultado de la ejecución de los tests del Frontend para las diferentes builds.



Dentro de cada build se pueden apreciar las diferentes acciones que se ejecutan. Si accedemos a una en particular, se muestra el resultado que ha mostrado la consola, así como los tests que se han ejecutado.

The screenshot shows a GitHub Actions workflow run for 'build (19.x)' that succeeded 37 minutes ago. The log lists several steps: 'Set up job', 'Run actions/checkout@v3', 'Use Node.js 19.x', 'Run npm ci', 'Run npm run build --if-present', and 'Run npm test'. The 'Run npm test' step is expanded, showing the following console output:

```
1  ▶ Run npm test
2
3  > utopia-react@0.1.0 test
4  > react-scripts test
5
6  PASS src/_tests_/Register.test.js
7  PASS src/_tests_/Login.test.js
8  PASS src/_tests_/Navbar.test.js
9
10 Test Suites: 3 passed, 3 total
11 Tests:      14 passed, 14 total
12 Snapshots:  0 total
13 Time:        0.548 s
14 Ran all test suites.
```

Figura 7: Resultado de cada uno de los pasos.

Al igual que se ha visto para los tests del frontend, se realiza de forma similar para el backend.

The screenshot shows the 'React Tests BackEnd' workflow in GitHub Actions. The left sidebar lists 'All workflows', 'React Tests BackEnd', 'React Tests FrontEnd', 'Management', and 'Caches'. The main area shows a table of workflow runs for 'backTest.yml'.

	Event	Status	Branch	Actor
3 workflow runs				
change names	React Tests BackEnd #3: Commit 7365e2a pushed by alu0101321141	main	3 hours ago	...
arreglado	React Tests BackEnd #2: Commit 23045a5 pushed by alu0101321141	main	3 hours ago	...
workflow Server	React Tests BackEnd #1: Commit e40bd14 pushed by alu0101321141	main	3 hours ago	...

Figura 8: Resultado de la ejecución de los tests del Backend.

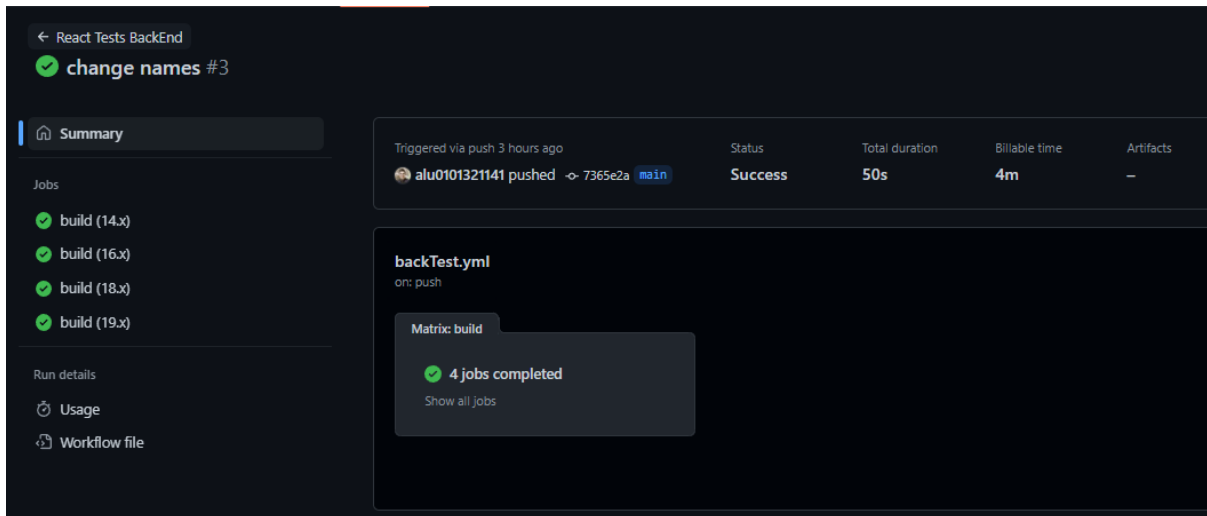


Figura 9: Resultado de los tests del Backend para las diferentes builds.



Figura 10: Resultado de cada uno de los pasos.



Tareas a realizar en la iteración 5

Como se comentó anteriormente, el despliegue de la aplicación utilizando CD se realizará en posteriores iteraciones.

Además de la refactorización del código, se plantean las siguientes tareas a completar en la siguiente iteración:

▼	4 points	3 • 26 Dec - 1 Jan 2023 •	100%
★	=	Setup SonarCloud/Codecov/Coveralls (AL , AL, VL)	Start <input type="checkbox"/>
★	=	Configuración CI (AL, AL, VL)	Start <input type="checkbox"/>

Figura 11 : Tareas a realizar en la próxima iteración

Referencias

- [1] Herramienta de Pivotal Tracker: <https://www.pivotaltracker.com/>
- [2] Github Actions: <https://github.com/features/actions>
- [3] Repositorio del proyecto: <https://github.com/SyTW2223/E08.git>