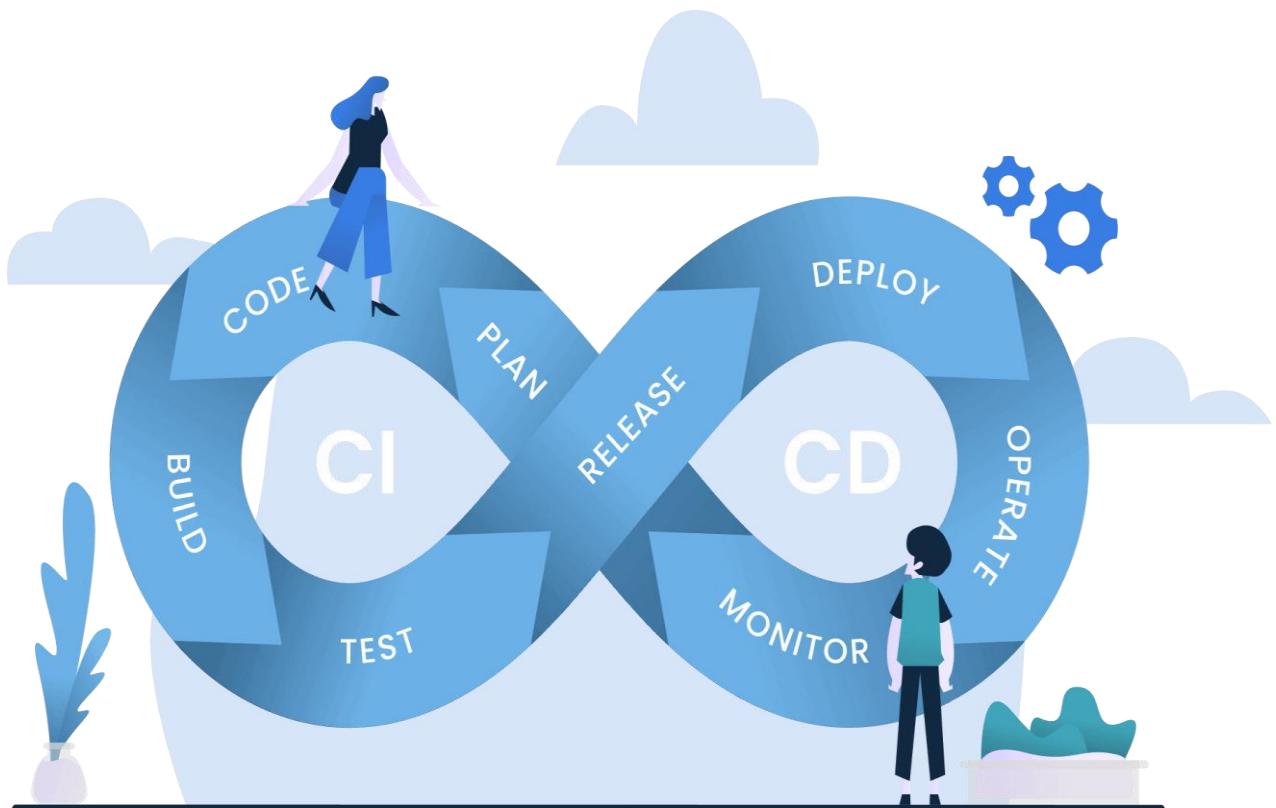


Práctica final: desarrollo ágil de una aplicación web con integración continua



Laura Gil Gómez

Integración Continua en el Desarrollo Ágil

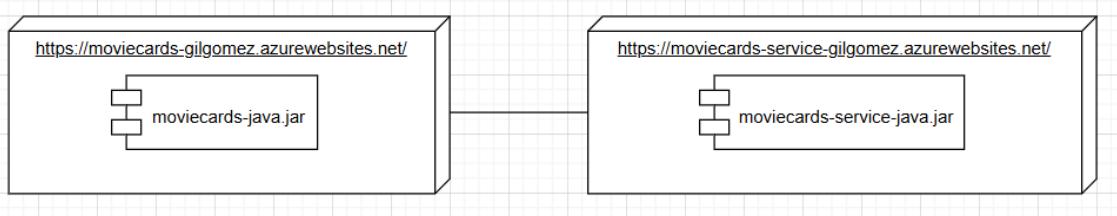
Máster Universitario en Desarrollo Ágil de Software para la Web

Índice

Introducción.....	3
Apartado 1.....	3
1.1. Instalar la infraestructura necesaria.....	3
1.2. Crear repositorio local y remoto.....	5
1.3. Diseño del flujo de trabajo para la integración continua	6
1.3.1. Trabajo build	7
1.3.2. Trabajo test	8
1.3.3. Trabajo qa	10
1.3.4. Trabajo deploy	13
1.4. Realizar una segunda versión de la aplicación	17
Apartado 2.....	23
Apartado 3.....	27
Apartado 4.....	30
Apartado 5.....	32
5.1. Añadir fase stage al workflow del proyecto	32
5.2. Modificar el código src.....	33
5.3. Modificar pruebas unitarias	36
5.4. Modificar pruebas de integración.....	36
5.5. Modificar pruebas funcionales.....	37
5.6. Garantía de calidad	38

Introducción

El objetivo de esta práctica es el desarrollo de un proyecto web llamado moviecards-gilgomez mediante integración continua y desarrollo ágil. Para ello, se van a realizar dos versiones a lo largo de cinco apartados: la primera versión consiste en una aplicación con un servicio web integrado (correspondiente al apartado 1), mientras que en la segunda versión se separa dicho servicio de la aplicación haciendo que haya independencia en el código (correspondiente a resto de apartados de la práctica). El proyecto quedará desplegado de la siguiente manera:



Apartado 1

1.1. Instalar la infraestructura necesaria

Se crean e instalan los siguientes elementos:

Cuenta gratuita en Github para tener las siguientes funcionalidades: servidor de integración continua, repositorio git para almacenar el código fuente del proyecto y tableros Kanban para la gestión ágil del proyecto. Tendrá como nombre de usuario lgilgomez.

Cuenta gratuita educativa en Azure mediante el propio correo de la universidad (en este caso laura.gilg@edu.uah.es).

Instalar Git, Visual Studio Code y Maven.

Descargar y ejecutar archivo con código de ejemplo disponible en el Aula Virtual llamado moviecards.zip. La aplicación queda funcionando en <http://localhost:8089>:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows

PS C:\moviecards> mvn spring-boot:run
[INFO] Scanning for projects...
[INFO]
[INFO] < com.lauracercas:moviecards >-
[INFO] Building moviecards 0.0.1-SNAPSHOT
[INFO]   from pom.xml
[INFO]   [ jar ]-
[WARNING] Parameter 'parameters' is unknown for plugin 'maven-compiler-plugin:3.1:compile (default-compile)'
[WARNING] Parameter 'parameters' is unknown for plugin 'maven-compiler-plugin:3.1:testCompile (default-testCompile)'
[INFO]
[INFO] >>> spring-boot:2.6.7:run (default-cli) > test-compile @ moviecards >>>
[WARNING] 1 problem was encountered while building the effective model for org.javassist:javassist:jar:3.20.0-GA during dependency collection step for project (use -X to see details)
[INFO]
[INFO] --- resources:3.2.0:resources (default-resources) @ moviecards ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
```



Ejecutar las pruebas de moviecards:

```

Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows

PS C:\moviecards> mvn clean verify
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.lauracercas:moviecards >-----
[INFO] Building moviecards 0.0.1-SNAPSHOT
[INFO]   from pom.xml
[INFO] -----[ jar ]-----
[WARNING] Parameter 'parameters' is unknown for plugin 'maven-compiler-plugin:3.1:compile (default-compile)'
[WARNING] Parameter 'parameters' is unknown for plugin 'maven-compiler-plugin:3.1:testCompile (default-testCompile)'
[WARNING] 1 problem was encountered while building the effective model for org.javassist:javassist:jar:3.20.0-GA during
dependency collection step for project (use -X to see details)
[INFO]
[INFO] --- clean:3.1.0:clean (default-clean) @ moviecards ---
[INFO] Deleting C:\moviecards\target
[INFO]
[INFO] --- resources:3.2.0:resources (default-resources) @ moviecards ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Using 'UTF-8' encoding for file filtering.

[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.034 s -- in com.lauracercas.moviecards.integrat
iontest.repositories.MovieJPAT
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] --- failsafe:3.2.5:verify (default) @ moviecards ---
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 24.394 s
[INFO] Finished at: 2024-12-30T10:12:50+01:00
[INFO] -----

```

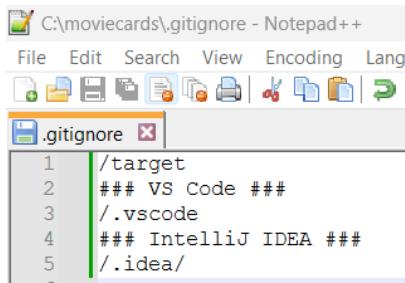
Instalar Docker y un contenedor con SonarQube. Para ello, se instala la imagen josehilera/Ubuntu-sonar junto a un contenedor cuyo nombre será asignado automáticamente por Docker: nifty_jemison. El servidor de SonarQube será accedido mediante la URL <http://localhost:9000> con nombre de usuario admin y contraseña 1234.

Containers [Give feedback](#)

Container CPU usage		Container memory usage		Show charts
No containers are running.				
<input type="text"/> Search		<input type="checkbox"/> Only show running containers		
□	Name	Container ID	Image	Port(s)
□	nifty_jemison	b83f3c75ff08	josehilera/ubuntu-sonar:latest	9000:9000

1.2. Crear repositorio local y remoto

Se convierte la carpeta moviecards en un repositorio local Git mediante el comando git init dentro de la propia carpeta, creándose automáticamente una carpeta llamada .git oculta. A continuación, se crea en la raíz del proyecto un fichero llamado .gitignore en el que se indicarán los ficheros y carpetas que no deben subirse al repositorio remoto.



C:\moviecards\.gitignore - Notepad++

File Edit Search View Encoding Lang

File Explorer View Status Bar

.gitignore

```
1 /target
2 ### VS Code ####
3 /.vscode
4 ### IntelliJ IDEA ####
5 /.idea/
```

Seguidamente se crea un repositorio remoto en Github de nombre moviecards con la siguiente configuración:

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner * Repository name *

 Igilgomez / moviecards
✓ moviecards is available.

Great repository names are short and memorable. Need inspiration? How about [super-duper-train](#) ?

Description (optional)

 Public
Anyone on the internet can see this repository. You choose who can commit.
  Private
You choose who can see and commit to this repository.

Initialize this repository with:

Add a README file
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: [None](#)

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: [None](#)

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

 You are creating a public repository in your personal account.

Create repository

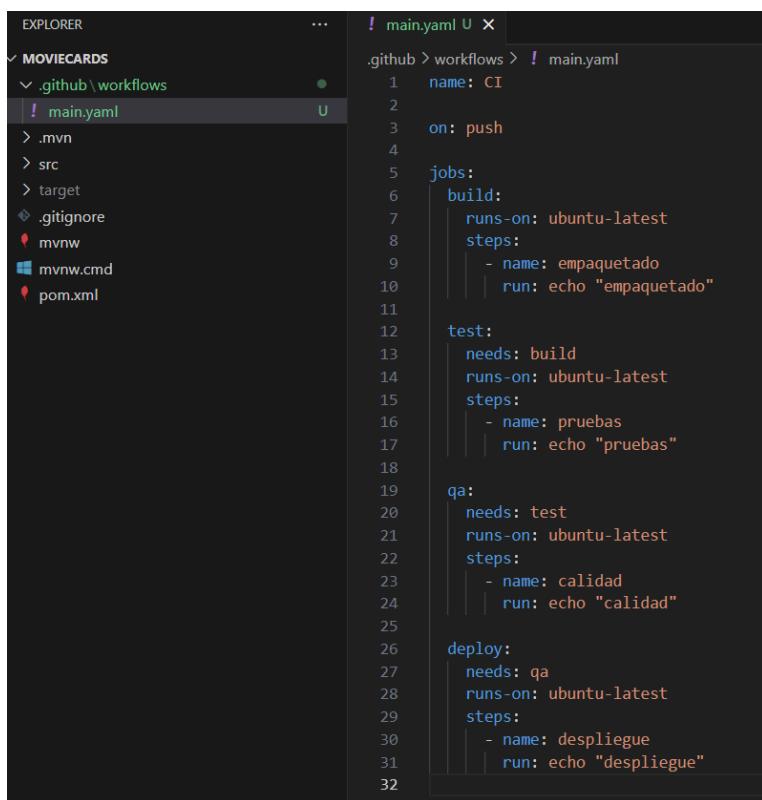
A la hora de subir el repositorio local existente al remoto se ejecutan los siguientes comandos:

```
C:\moviecards> git remote add origin https://github.com/lgilgomez/moviecards.git
C:\moviecards> git add .
C:\moviecards> git status
C:\moviecards> git commit -m "Commit inicial"
C:\moviecards> git push -u origin master
```

Finalmente queda una réplica del repositorio local en Github sin los archivos y carpetas indicadas en `.gitignore`.

1.3. Diseño del flujo de trabajo para la integración continua

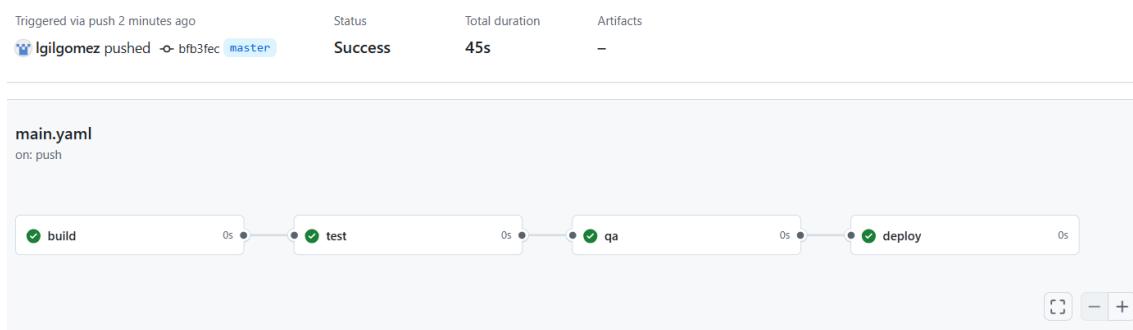
El flujo de trabajo (workflow) estará constituido en el fichero `main.yaml` de la siguiente manera:



```
EXPLORER ... main.yaml U X
MOVIECARDS .github > workflows > main.yaml
|.mvn
> src
> target
|.gitignore
mvnw
mvnw.cmd
pom.xml

main.yaml
1 name: CI
2
3 on: push
4
5 jobs:
6   build:
7     runs-on: ubuntu-latest
8     steps:
9       - name: empaquetado
10      run: echo "empaquetado"
11
12   test:
13     needs: build
14     runs-on: ubuntu-latest
15     steps:
16       - name: pruebas
17      run: echo "pruebas"
18
19   qa:
20     needs: test
21     runs-on: ubuntu-latest
22     steps:
23       - name: calidad
24      run: echo "calidad"
25
26   deploy:
27     needs: qa
28     runs-on: ubuntu-latest
29     steps:
30       - name: despliegue
31      run: echo "despliegue"
32
```

Una vez terminado de escribir el workflow, se hace push del proyecto y se observa la ejecución de dicho workflow en la sección Actions->All workflows->Nombre del commit creado con el push:



Triggered via push 2 minutes ago	Status	Total duration	Artifacts
lgilgomez pushed → bfb3fec master	Success	45s	-

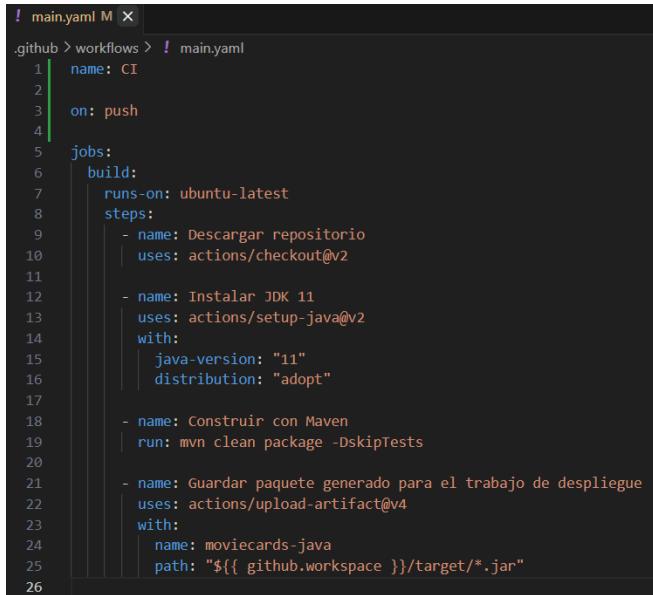
main.yaml
on: push

```

graph LR
    build --> test
    test --> qa
    qa --> deploy
    
```

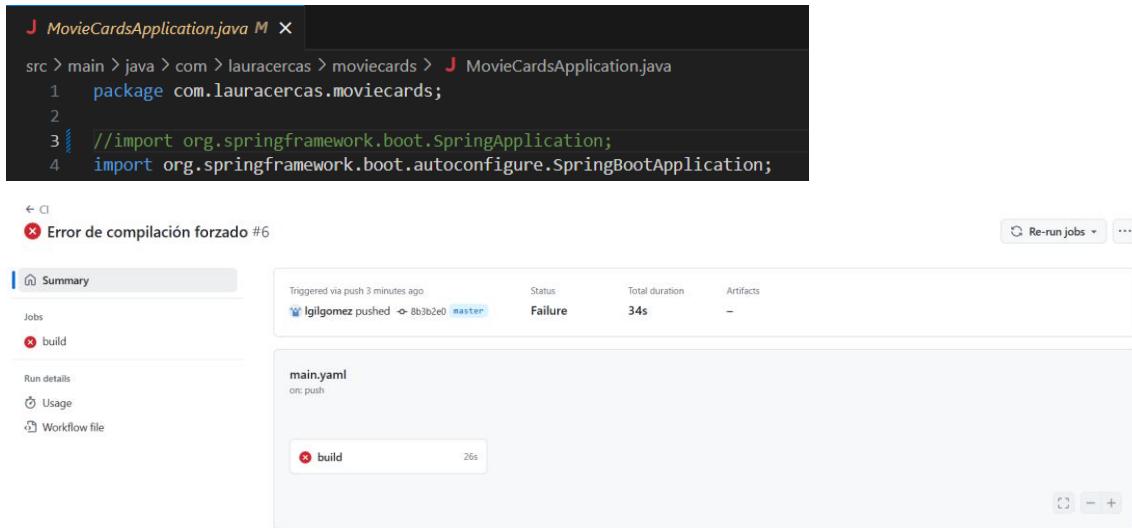
1.3.1. Trabajo build

Se realizan los siguientes cambios en el fichero main.yaml sin de momento hacer push contra el repositorio remoto:



```
! main.yaml M X
.github > workflows > ! main.yaml
1   name: CI
2
3   on: push
4
5   jobs:
6     build:
7       runs-on: ubuntu-latest
8       steps:
9         - name: Descargar repositorio
10        uses: actions/checkout@v2
11
12         - name: Instalar JDK 11
13        uses: actions/setup-java@v2
14        with:
15          java-version: "11"
16          distribution: "adopt"
17
18         - name: Construir con Maven
19        run: mvn clean package -DskipTests
20
21         - name: Guardar paquete generado para el trabajo de despliegue
22        uses: actions/upload-artifact@v4
23        with:
24          name: moviecards-jar
25          path: "${{ github.workspace }}}/target/*.jar"
26
```

Seguidamente se modifica el siguiente fichero con el objetivo de que se fuerce un error de compilación cuando se hace push:



```
J MovieCardsApplication.java M X
src > main > java > com > lauracercas > moviecards > J MovieCardsApplication.java
1 package com.lauracercas.moviecards;
2
3 //import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
```

← CI
✖ Error de compilación forzado #6

Summary	Status	Total duration	Artifacts
Jobs	Failure	34s	-
build			

Triggered via push 3 minutes ago

lgilgomez pushed 8b3b2e0 master

main.yaml
on: push

build 26s

Revisando la consola de trabajo de build se ve cuál ha sido el error:

```

build
failed 4 minutes ago in 26s
Search logs
16s

Construir con Maven
14238 [Error] /home/runner/work/moviecards/moviecards/src/main/java/com/lauracercas/moviecards/MovieCardsApplication.java:[15,9] cannot find symbol
14239     symbol: variable SpringApplication
14240     location: class com.lauracercas.moviecards.MovieCardsApplication
14241 [INFO] 1 error
14242 [INFO] -----
14243 [INFO] -----
14244 [INFO] BUILD FAILURE
14245 [INFO] -----
14246 [INFO] Total time: 15.705 s
14247 [INFO] Finished at: 2024-12-30T18:00:05Z
14248 [INFO] -----
14249 [Error] Failed to execute goal org.apache.maven.plugins:maven-compiler-plugin:3.1:compile (default:compile) on project moviecards: Compilation failure
14250 [Error] /home/runner/work/moviecards/moviecards/src/main/java/com/lauracercas/moviecards/MovieCardsApplication.java:[15,9] cannot find symbol
14251 [Error]     symbol: variable SpringApplication
14252 [Error]     location: class com.lauracercas.moviecards.MovieCardsApplication
14253 [Error] 
14254 [Error]     -> [Help 1]
14255 [Error] 
14256 [Error] To see the full stack trace of the errors, re-run Maven with the -e switch.
14257 [Error] Re-run Maven using the -X switch to enable full debug logging.
14258 [Error] 
14259 [Error] For more information about the errors and possible solutions, please read the following articles:
14260 [Error] [Help 1] http://cwiki.apache.org/confluence/display/Maven/MojoFailureException
14261 [Error] Process completed with exit code 1

```

Se corrige el error anterior dejando el fichero tal y como estaba, de tal forma que cuando se hace push se ejecuta el workflow correctamente. También se genera un snapshot con la aplicación jar moviecards-java generada:

```

MovieCardsApplication.java X
src > main > java > com > lauracercas > moviecards > MovieCardsApplication.java
1 package com.lauracercas.moviecards;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;

```

✓ Error de compilación corregido #7

Re-run all jobs ...

Summary		Status	Total duration	Artifacts
Jobs	Ilgomez pushed 8b26f00 master	Success	37s	1
Run details				
Usage				
Workflow file				

Annotations
1 warning

⚠ build
ubuntu-latest pipelines will use ubuntu-24.04 soon. For more details, see <https://github.com/actions/runner-images/issues/10636>

Artifacts
Produced during runtime

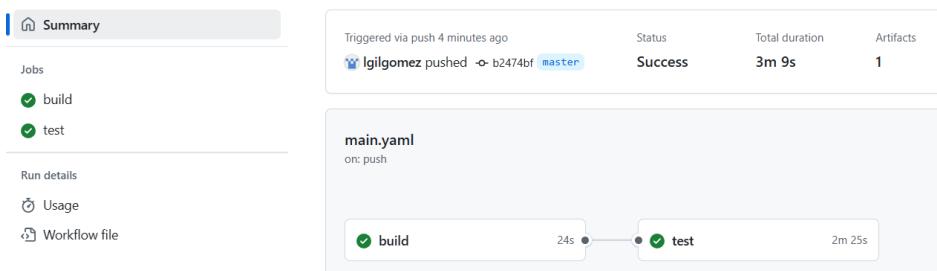
Name	Size
moviecards-java	38.4 MB

1.3.2. Trabajo test

Se añaden las siguientes modificaciones al archivo main.yaml para albergar la fase de testeo de la aplicación, de tal manera que al hacer push se comprueba que tanto la fase de build como de test en el workflow se han ejecutado correctamente:

```
! main.yaml M X
.github > workflows > ! main.yaml
  5   jobs:
  6     test:
  7       runs-on: ubuntu-latest
  8       steps:
  9         - name: Descargar repositorio
 10           uses: actions/checkout@v2
 11
 12         - name: Instalar JDK 11
 13           uses: actions/setup-java@v2
 14           with:
 15             java-version: "11"
 16             distribution: "adopt"
 17
 18         - name: Instalar Chrome y Chromedriver para pruebas end to end
 19           run: |
 20             wget https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb
 21
 22             sudo dpkg -i google-chrome-stable_current_amd64.deb
 23
 24             sudo apt --fix-broken install -y
 25
 26             CHROMEDRIVER_VERSION=$(curl -ss https://chromedriver.storage.googleapis.com/LATEST_RELEASE)
 27
 28             curl -L -o chromedriver.zip https://chromedriver.storage.googleapis.com/$CHROMEDRIVER_VERSION/chromedriver_linux64.zip
 29
 30             unzip chromedriver.zip
 31
 32             chmod +x chromedriver
 33
 34             sudo mv chromedriver /usr/local/bin/
 35
 36         - name: Ejecutar la aplicación para pruebas end to end
 37           run: mvn spring-boot:run & sleep 60
 38
 39         - name: Ejecutar las pruebas unitarias, de integración y end to end
 40           run: mvn clean verify
```

 Trabajo test creado #9

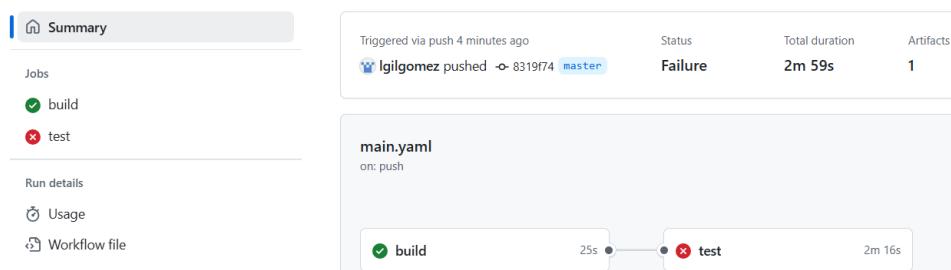


Se realiza una prueba fallida modificando el siguiente código, de tal manera que al hacer push el workflow falla en la fase de test:

```
J Actor.java M X
src > main > java > com > lauracercas > moviecards > model > J Actor.java
16 public class Actor {
47     public String getName() {
48         return "Hola";
49     }
}
```

← CI

✗ Fallo forzado en prueba unitaria #10

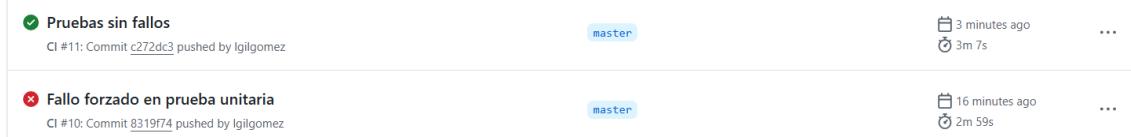
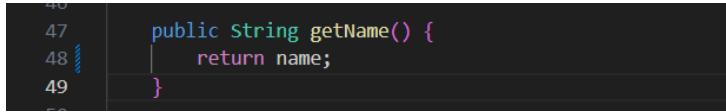


Puede comprobarse cuál ha sido el error en el archivo de logs:



```
C:\Users\Usuario\Downloads\logs_32540494350\test6.txt. Ejecutar las pruebas unitarias, de integración y end to end.txt - Notepad+
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
6. Ejecutar las pruebas unitarias, de integración y end to end.txt [3]
1865 2024-12-30T19:11:25.845343Z [INFO] Tests run: 11, Failures: 0, Skipped: 0, Time elapsed: 0 s - in com.lauracercas.moviecards.unittest.model.MovieTest
1869 2024-12-30T19:11:25.8453708Z [INFO] Running com.lauracercas.moviecards.unittest.model.ActorTest
1870 2024-12-30T19:11:25.8455184Z [ERROR] Tests run: 5, Failures: 1, Errors: 0, Skipped: 0, Time elapsed: 0.002 s <><> FAILURE! - in com.lauracercas.moviecards.unittest.model.ActorTest
1871 2024-12-30T19:11:25.845610E [ERROR] testSetGetName Time elapsed: 0 s <><> FAILURE!
1872 2024-12-30T19:11:25.8457974Z org.opentest4j.AssertionFailedError: expected: <Sample name> but was: <Hola>
1873 2024-12-30T19:11:25.8459380Z at com.lauracercas.moviecards.unittest.model.ActorTest.testSetGetName(ActorTest.java:29)
1874 2024-12-30T19:11:25.8460564Z
```

Se corrige el error para que no haya fallos al hacer push:

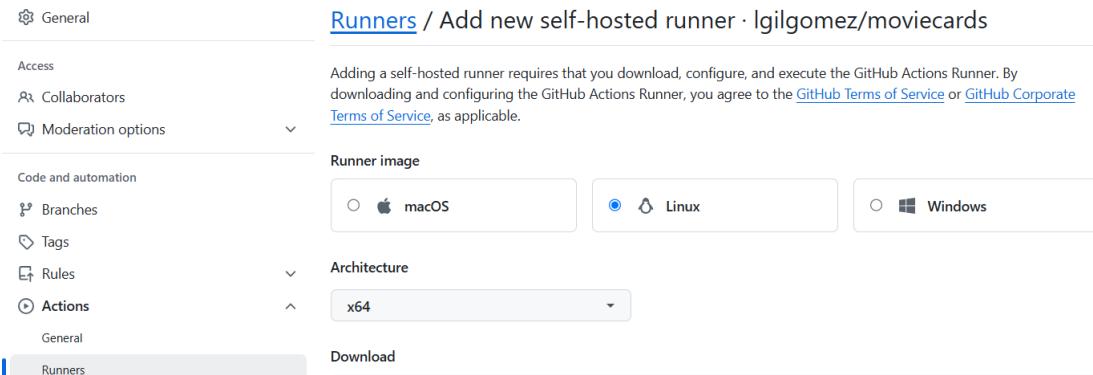


Commit	Author	Time Ago	Actions
c272dc3	Ilgilgomez	3 minutes ago	Pruebas sin fallos
8319f74	Ilgilgomez	16 minutes ago	Fallo forzado en prueba unitaria

1.3.3. Trabajo qa

Activar un runner propio en el contenedor ubuntu-sonar

Previamente se ha de hacer la siguiente configuración en Github para que el runner a crear quede alojado en Linux:

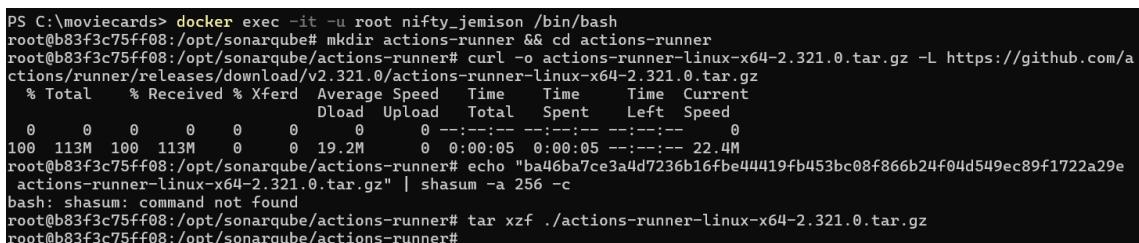


The screenshot shows the 'Runners / Add new self-hosted runner' page for the repository 'Ilgilgomez/moviecards'. The 'General' tab is selected. The 'Runner image' section has 'Linux' selected. The 'Architecture' dropdown shows 'x64'. The 'Download' button is visible at the bottom.

Se arranca el contenedor que aloja el servidor SonarQube de la siguiente manera:

```
docker exec -it -u root nifty_jemison /bin/bash
```

A continuación, se ejecutan los siguientes comandos para descargar e instalar en el contenedor anterior la última versión de creación de runners:



```
PS C:\moviecards> docker exec -it -u root nifty_jemison /bin/bash
root@b83f3c75ff08:/opt/sonarqube# mkdir actions-runner && cd actions-runner
root@b83f3c75ff08:/opt/sonarqube/actions-runner# curl -o actions-runner-linux-x64-2.321.0.tar.gz -L https://github.com/actions/runner/releases/download/v2.321.0/actions-runner-linux-x64-2.321.0.tar.gz
% Total    % Received % Xferd  Average Speed   Time   Time   Current
          Dload  Upload Total Spent   Left Speed
0       0      0      0      0      0      0      0      0      0      0      0      0
100  113M  100  113M  0      0  19.2M  0      0:00:05  0:00:05  0:00:05  22.4M
root@b83f3c75ff08:/opt/sonarqube/actions-runner# echo "ba46ba7ce3a4d7236b16fbe44419fb453bc08f866b24f04d549ec89f1722a29e" | shasum -a 256 -c
bash: shasum: command not found
root@b83f3c75ff08:/opt/sonarqube/actions-runner# tar xzf ./actions-runner-linux-x64-2.321.0.tar.gz
root@b83f3c75ff08:/opt/sonarqube/actions-runner#
```

Posteriormente se indican los comandos para configurar y arrancar un runner llamado mi-runner en el contenedor nifty_jemison:

```
root@b83f3c75ff08:/opt/sonarqube/actions-runner# export RUNNER_ALLOW_RUNASROOT="1"
root@b83f3c75ff08:/opt/sonarqube/actions-runner# ./config.sh --url https://github.com/lgilgomez/moviefcards --token BB6LN
4OPUKUFKS4FM3GRI03HOMCG2

[{"id": "runner-1", "label": "self-hosted", "os": "Linux", "arch": "X64"}, {"id": "runner-2", "label": "self-hosted", "os": "Linux", "arch": "X64"}]
  Self-hosted runner registration

# Authentication

✓ Connected to GitHub

# Runner Registration

Enter the name of the runner group to add this runner to: [press Enter for Default]

Enter the name of runner: [press Enter for b83f3c75ff08] mi-runner

This runner will have the following labels: 'self-hosted', 'Linux', 'X64'
Enter any additional labels (ex. label-1,label-2): [press Enter to skip]

✓ Runner successfully added
✓ Runner connection is good

# Runner settings

Enter name of work folder: [press Enter for _work]

✓ Settings Saved.

root@b83f3c75ff08:/opt/sonarqube/actions-runner# ./run.sh

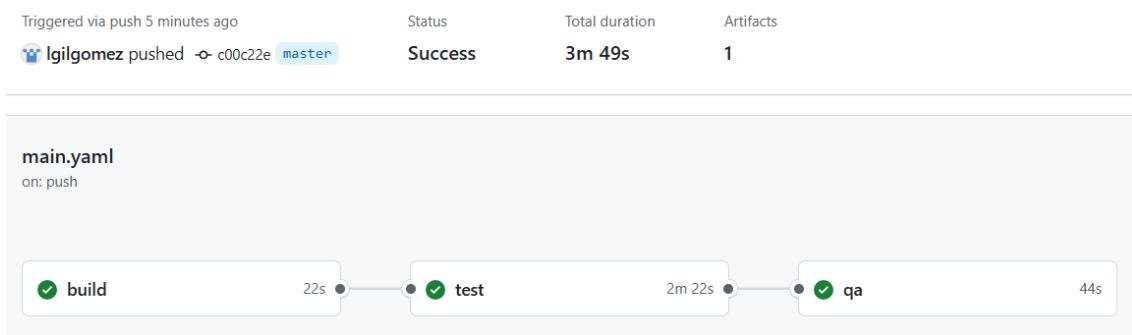
✓ Connected to GitHub

Current runner version: '2.321.0'
2024-12-30 19:51:51Z: Listening for Jobs
```

Definir el trabajo que

Se añade lo correspondiente a la fase qa en el fichero main.yaml de tal forma que cuando se hace un push se ejecuta el workflow correctamente:

```
! main.yaml M ×  
.github > workflows > ! main.yaml  
5   jobs:  
6     test:  
7       qa:  
8         needs: test  
9         runs-on: self-hosted  
10        continue-on-error: true  
11        steps:  
12          - name: Descargar repositorio  
13            uses: actions/checkout@v2  
14  
15          - name: Instalar JDK 11  
16            uses: actions/setup-java@v2  
17            with:  
18              java-version: "11"  
19              distribution: "adopt"  
20  
21          - name: Construir con Maven  
22            run: mvn clean package -DskipTests  
23  
24          - name: Revisar la calidad con Sonarqube  
25            run:  
26              mvn sonar:sonar -Dsonar.host.url=http://localhost:9000 -Dsonar.qualitygate.wait=true -Dsonar.login=admin -Dsonar.password=123456  
27  
28
```



2024-12-30 21:19:38Z: Running job: qa
2024-12-30 21:20:25Z: Job qa completed with result: Succeeded

Aumentar la exigencia de calidad

Si se inicia sesión en el servidor de SonarQube se puede observar que hay los siguientes problemas de calidad de código:

The screenshot shows the SonarQube interface for the moviecards project. The main view displays a list of 55 issues across various files, including ActorController.java and MovieController.java. The issues are categorized by type (Bug, Vulnerability, Code Smell) and severity (Blocker, Critical, Minor, Info, Major). A sidebar on the left provides filters for new code issues, type, and severity. The top navigation bar includes links for Projects, Issues, Rules, Quality Profiles, Quality Gates, and Administration.

Para que el control de calidad de la aplicación sea más exigente se define una regla en SonarQube llamada control calidad, de manera que la aplicación no apruebe la prueba de calidad si el número de fallos críticos es superior a 5. Se coloca después esa regla con la configuración Set as default para que se aplique a todos los proyectos.

The screenshot shows the SonarQube Quality Gates configuration. A new quality gate named "control calidad" is being created. The configuration specifies that the quality gate fails if there are more than 5 critical issues. The "On Overall Code" option is selected, and the operator is set to "is greater than" with a value of 5. The "Add Condition" button is visible at the bottom right.

Ejecutando de nuevo el workflow en Github se obtiene que ha fallado la fase de qa con 8 errores de calidad críticos:



Si se aumenta en 8 el número de fallos críticos permitidos en la regla control calidad para pasar la prueba de calidad en SonarQube y se refresca el workflow la fase de qa deja de fallar.

1.3.4. Trabajo deploy

Despliegue continuo

Consiste en hacer que la aplicación se despliegue a producción de manera automática, esto es, sin intervención de ninguna persona. Para ello se creará una aplicación web llamada moviecards-gilgomez en el portal de Azure con los siguientes parámetros:

Todos los servicios > App Services >

Crear aplicación web ...

Detalles del proyecto

Seleccione una suscripción para administrar los recursos implementados y los costos. Use los grupos de recursos como carpetas para organizar y administrar todos los recursos.

Suscripción *	Azure para estudiantes
Grupo de recursos *	(Nuevo) moviecards-gilgomez_group Crear nuevo

Detalles de instancia

Nombre: moviecards-gilgomez.azurewebsites.net

Pruebe un nombre de host predeterminado único (versión preliminar). [Más información sobre esta actualización](#)

Publicar *: Código

Pila del entorno en tiempo de ejecución *: Java 11

Pila de servidor web Java *: Java SE (Embedded Web Server)

Sistema operativo *: Linux

Región *: East US

¿No encuentra su plan de App Service? Pruebe otra región o seleccione su App Service Environment.

Planes de precios

El plan de tarifa de App Service determina la ubicación, las características, los costos y los recursos del proceso asociados a la aplicación. [Más información](#)

Plan de Linux (East US) *	(Nuevo) ASP-moviecardsgilgomezgroup-b6bd Crear nuevo
Plan de precios	Gratis F1 (Infraestructura compartida) Explorar planes de precios

Revisar y crear < Anterior Siguiente: Implementación >

A continuación, se configura la aplicación web creada de tal forma que hay que activar la opción “Credenciales de publicación de autenticación básica de SCM”, pinchando después en el botón Guardar:

Posteriormente se crea un nuevo fichero llamado master-moviecards-gilgomez.yaml que será generado de la siguiente manera a través de la opción Centro de implementación, finalizando con el botón Guardar:

Configuración de autenticación

Seleccione cómo desea que el flujo de trabajo de acción de GitHub se autentique en Azure. Si elige una identidad asignada por el usuario, la identidad seleccionada se federará con GitHub como cliente autorizado y se le concederán permisos de escritura en la aplicación. [Más información](#)

Tipo de autenticación*

Identidad asignada por el usuario
 Autenticación básica

Se pasa el fichero generado al repositorio local mediante pull y se copia su sección deploy al fichero main de la siguiente manera:

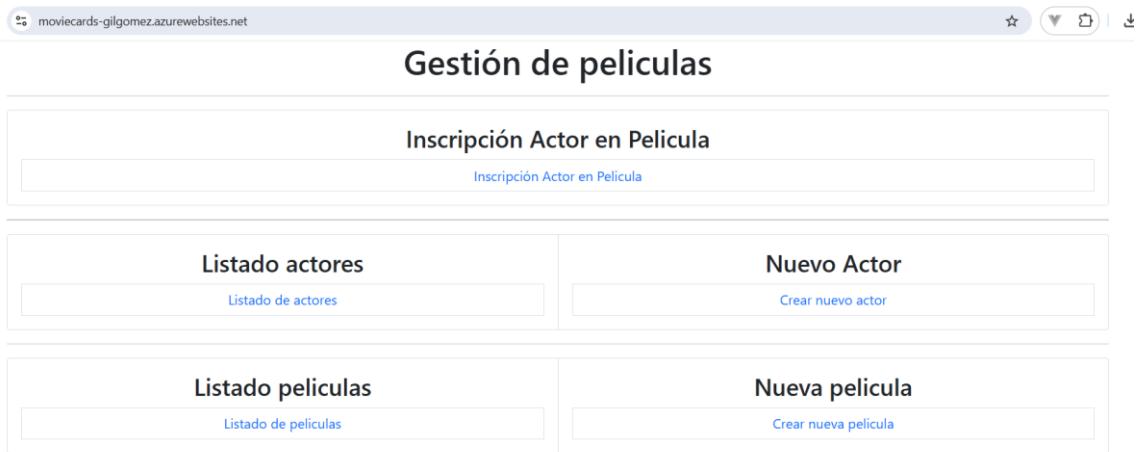
```

EXPLORER ... ! main.yaml M x
MOVIECARDS .github > workflows > ! main.yaml
  .github > workflows > ! main.yaml
    jobs:
      qa:
        steps:
          deploy:
            runs-on: ubuntu-latest
            needs: qa
            if: github.ref=='refs/heads/master'
            environment:
              name: 'Production'
              url: ${{ steps.deploy-to-webapp.outputs.webapp-url }}
          steps:
            - name: Download artifact from build job
              uses: actions/download-artifact@v4
              with:
                name: moviecards-java
            - name: Deploy to Azure Web App
              id: deploy-to-webapp
              uses: azure/webapps-deploy@v3
              with:
                app-name: 'moviecards-gilgomez'
                slot-name: 'Production'
                package: '*.jar'
                publish-profile: ${ secrets.AZUREAPPSERVICE_PUBLISHPROFILE_796042E71833499B93C0B99FD46FE4D1 }

```

Tras hacer push y ejecutarse el workflow correctamente se genera la siguiente URL como enlace a la aplicación desplegada en Azure:

<https://moviecards-gilgomez.azurewebsites.net/>



Entrega continua

Se modifica el fichero main.yaml de forma que para que se despliegue la aplicación debe ser aprobado por una persona de manera manual:

```

104   deploy:
105     runs-on: ubuntu-latest
106     needs: stage
107     if: github.ref=='refs/heads/master'
108     environment:
109       name: 'Production'
110       url: ${{ steps.deploy-to-webapp.outputs.webapp-url }}
111
112   steps:
113     - name: Aprobación manual
114       uses: trstringer/manual-approval@v1
115       with:
116         secret: ${{ secrets.TOKEN }}
117         approvers: lgilgomez
118
119     - name: Download artifact from build job
120       uses: actions/download-artifact@v4
121       with:
122         name: moviecards-java
123
124     - name: Deploy to Azure Web App
125       id: deploy-to-webapp
126       uses: azure/webapps-deploy@v3
127       with:
128         app-name: 'moviecards-gilgomez'
129         slot-name: 'Production'
130         package: '*.jar'
131         publish-profile: ${{ secrets.AZUREAPPSPROFILE_796042E71833499B93C0B99FD46FE4D1 }}
```

Previamente se establece una credencial (secreto) para habilitar el acceso a Github desde el repositorio local llamado TOKEN:

The screenshot shows the GitHub 'Personal access tokens' section. A new token named 'TOKEN' is being created. The token is described as 'classic' and has an expiration set to '30 days'. The note field contains 'TOKEN'.

Tras hacer push se observa que el workflow queda en estado de pausa hasta que no se aprueba el despliegue de forma manual en Github:

Manual approval required for workflow run 12552938000 #1

[Open](#) Igilgomez opened this issue 1 minute ago · 0 comments

A comment from Igilgomez states: "Workflow is pending manual review. URL: <https://api.github.com/lgilgomez/moviecards/actions/runs/12552938000>". It also lists "Required approvers: [lgilgomez]". Below this, Igilgomez self-assigned the issue. A follow-up comment says "yes".

1.4. Realizar una segunda versión de la aplicación

Crear un proyecto para el repositorio

Se va a crear un proyecto para la segunda versión de la aplicación, a la que se añadirán colores a la página principal que no había en la primera versión. Será un tablero de tipo Kanban.

The screenshot shows a GitHub project named 'moviecards'. At the top, there are navigation links for 'Ilgilgomez / Projects / moviecards'. A search bar is on the right. Below the title, there's a 'View 1' dropdown and a '+ New view' button. A 'Filter by keyword or by field' input field is also present. The main area displays a Kanban board with three columns: 'Todo' (0 items), 'In Progress' (0 items), and 'Done' (0 items). Each column has a descriptive text below it: 'This item hasn't been started' for Todo, 'This is actively being worked on' for In Progress, and 'This has been completed' for Done. There are also ellipsis and plus buttons for each column.

Se hace que el proyecto sea público de la siguiente forma:

Danger zone

This screenshot shows the 'Danger zone' section for a GitHub project. It includes three main sections: 'Visibility' (the project is currently public), 'Close project' (closing it will disable workflows and remove it from the list of open projects), and 'Delete project' (once deleted, there is no going back). Each section has a corresponding button: 'Public' (green checkmark), 'Close this project' (red button), and 'Delete this project' (blue button).

A continuación, se va a crear un sprint (milestone) llamado Aplicación con colores:

The screenshot shows the 'New milestone' creation form on GitHub. The title is 'Aplicación con colores'. The 'Due date (optional)' field is set to 'dd/mm/aaaa'. The 'Description' field contains the text: 'Se añadirán colores a la página principal de la aplicación index.html'. At the bottom, there is a 'Create milestone' button.

Al seleccionar el botón Create milestone, puede comprobarse que ahora aparece en la lista de milestones del repositorio:

The screenshot shows a GitHub milestones interface. At the top, there are tabs for 'Labels' and 'Milestones'. A green button labeled 'New milestone' is visible. Below the tabs, it says '1 Open' and '0 Closed'. The main card displays an issue titled 'Aplicación con colores' with a progress bar at 0% complete. The description reads: 'Se añadirán colores a la página principal de la aplicación index.html'. There are buttons for 'Edit', 'Close', and 'Delete'.

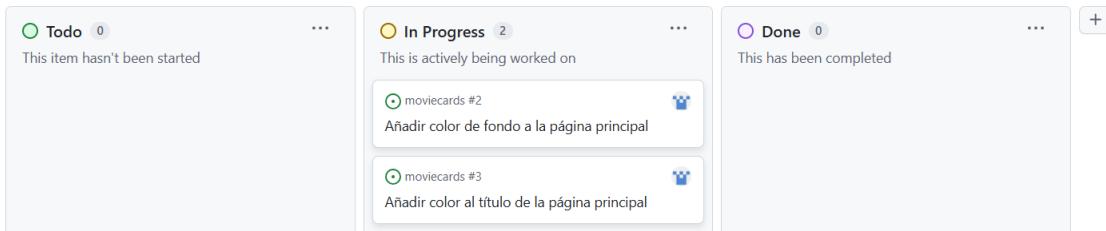
Seguidamente se crean dos historias de usuario (issues) para el milestone anterior:

This screenshot shows the process of creating a new GitHub issue. The title is 'Añadir color de fondo a la página principal'. The description area contains the text: 'Se trata de poner color de fondo gris en la página index.html.' The right sidebar shows assignees (Ilgomez), labels (None yet), projects (moviecards), and milestones (Aplicación con colores). A 'Submit new issue' button is at the bottom.

This screenshot shows the creation of a second GitHub issue. The title is 'Añadir color al título de la página principal'. The description area contains the text: 'Se trata de poner color de texto rojo y fondo beige en el título de la página index.html.' The right sidebar is identical to the first issue, showing the same assignee, labels, projects, and milestones. A 'Submit new issue' button is at the bottom.

This screenshot shows the GitHub project board for the 'moviecards' project. It has three columns: 'Todo' (2 items), 'In Progress' (0 items), and 'Done' (0 items). The 'Todo' column contains two items: 'moviecards #2' (status: Añadir color de fondo a la página principal) and 'moviecards #3' (status: Añadir color al título de la página principal). A search bar at the top right says 'Type / to search'.

El mismo desarrollador se encargará de los dos issues simultáneamente.



Para cada issue se va a crear una rama en git de la siguiente manera:

```
PS C:\moviecards> git branch añadir-color-fondo
PS C:\moviecards> git branch añadir-color-titulo
PS C:\moviecards> git branch
  añadir-color-fondo
  añadir-color-titulo
* master
PS C:\moviecards>
```

En la rama añadir-color-fondo se modifica el archivo index.html como se muestra a continuación:

```
<index.html M X>
src > main > resources > templates > <index.html> <html> <body>
7   <html xmlns:th="http://www.thymeleaf.org" lang="es">
12  </head>
13  <body style="background-color: lightgray">
14
```

Se suben los cambios a Github:

```
PS C:\moviecards> git checkout añadir-color-fondo
Switched to branch 'añadir-color-fondo'
PS C:\moviecards> git branch
* añadir-color-fondo
  añadir-color-titulo
  master
PS C:\moviecards> git add .
PS C:\moviecards> git status
On branch añadir-color-fondo
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   src/main/resources/templates/index.html

PS C:\moviecards> git commit -m "Añadido color de fondo en página principal"
[añadir-color-fondo efbee6a] Añadido color de fondo en página principal
  1 file changed, 1 insertion(+), 1 deletion(-)
PS C:\moviecards> git push origin añadir-color-fondo
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 8 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 694 bytes | 694.00 KiB/s, done.
Total 7 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
remote:
remote: Create a pull request for 'añadir-color-fondo' on GitHub by visiting:
remote:     https://github.com/lgilgomez/moviecards/pull/new/a%C3%B1adir-color-fondo
remote:
To https://github.com/lgilgomez/moviecards.git
 * [new branch]      añadir-color-fondo -> añadir-color-fondo
PS C:\moviecards>
```

Se hace merge de añadir-color-fondo con la rama master sin ningún tipo de conflicto, sin aprobar el despliegue manual:

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks. Learn more about diff comparisons here.

Añadido color de fondo en página principal

Add your description here...

Markdown is supported | Paste, drop, or click to add files

Create pull request

Reviewers: No reviews

Assignees: No one — assign yourself

Labels: None yet

Projects: moviecards

Milestone: Aplicación con colores

Development: Use [Closing keywords](#) in the description to automatically close issues

main.yaml

on: push

build 26s • test 2m 23s • qa 33s • deploy 2m 25s

Como ha finalizado el issue, en el tablero Kanban hay que arrastrarlo desde la columna In Progress a la columna Done:

Todo 0 This item hasn't been started

In Progress 1 This is actively being worked on

moviecards #3 Añadir color al título de la página principal

Done 2 This has been completed

moviecards #4 Añadido color de fondo en página principal

moviecards #2 Añadir color de fondo a la página principal

En la rama añadir-color-título se modifica el archivo index.html como se muestra a continuación:

```
index.html M X
src > main > resources > templates > index.html > html > body > div.container > hr
7   <html xmlns:th="http://www.thymeleaf.org" lang="es">
12  </head>
13  <body style="background-color: #beige">
14
15  <div class="container">
16    <div th:if="${message != null}" class='alert alert-success' th:text="${message}"
17      <h1 class="text-center" style="color: darkred">Gestión de películas</h1>
18    <hr>
```

Se suben los cambios a Github:

```
PS C:\moviecards> git checkout añadir-color-titulo
Switched to branch 'añadir-color-titulo'
PS C:\moviecards> git branch
  añadir-color-fondo
* añadir-color-titulo
  master
PS C:\moviecards> git add .
PS C:\moviecards> git status
On branch añadir-color-titulo
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   src/main/resources/templates/index.html

PS C:\moviecards> git commit -m "Añadido color en título página principal"
[añadir-color-titulo cf9d649] Añadido color en título página principal
  1 file changed, 2 insertions(+), 2 deletions(-)
PS C:\moviecards> git push origin añadir-color-titulo
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 8 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 707 bytes | 353.00 KiB/s, done.
Total 7 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
remote:
remote: Create a pull request for 'añadir-color-titulo' on GitHub by visiting:
remote:     https://github.com/lgilgomez/moviecards/pull/new/a%C3%Bladir-color-titulo
remote:
To https://github.com/lgilgomez/moviecards.git
 * [new branch]      añadir-color-titulo -> añadir-color-titulo
PS C:\moviecards>
```

Al tratar de hacer merge de añadir-color-titulo con la rama master se origina un conflicto debido a que se está tratando de modificar el color del fondo de body con dos valores distintos al mismo tiempo (mientras que en la primera rama body tiene asignado el color lightgray en la segunda rama se cambia esa misma línea de tal manera que ponga beige).

Añadido color en título página principal #6

Resolving conflicts between añadir-color-titulo and master and committing changes → añadir-color-titulo

The screenshot shows a conflict in the file `src/main/resources/templates/index.html`. The left pane shows the file structure with `index.html` being compared against `src/main/resources/templates/index.html`. The right pane shows the code with a conflict at line 16. The code is as follows:

```
12  </head>
13  <===== añadir-color-titulo
14  <body style="background-color: beige">
15  =====
16  <body style="background-color: lightgray">
17  >>>>> master
18
19  <div class="container">
```

A red bracket highlights the conflict area between line 13 and line 16. Line 13 is labeled "añadir-color-titulo" and line 16 is labeled "master".

Haciendo pull para actualizar el repositorio local se arreglarán los conflictos de manera manual en Visual Studio Code, seleccionando Aceptar cambio entrante para que quede el valor del fondo de body como lightgray:

```
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
13  <===== HEAD (Current Change)
14  <body style="background-color: beige">
15  =====
16  <body style="background-color: lightgray">
17  >>>>> 5621e0cde1b9a530c1b99cc117666637297ba214 (Incoming Change)
18
19  <div class="container">
20    <div th:if="${message != null}" class='alert alert-success' th:text="${message}" role='alert'></div>
```

Además, se añade manualmente en h1 un color de fondo beige:

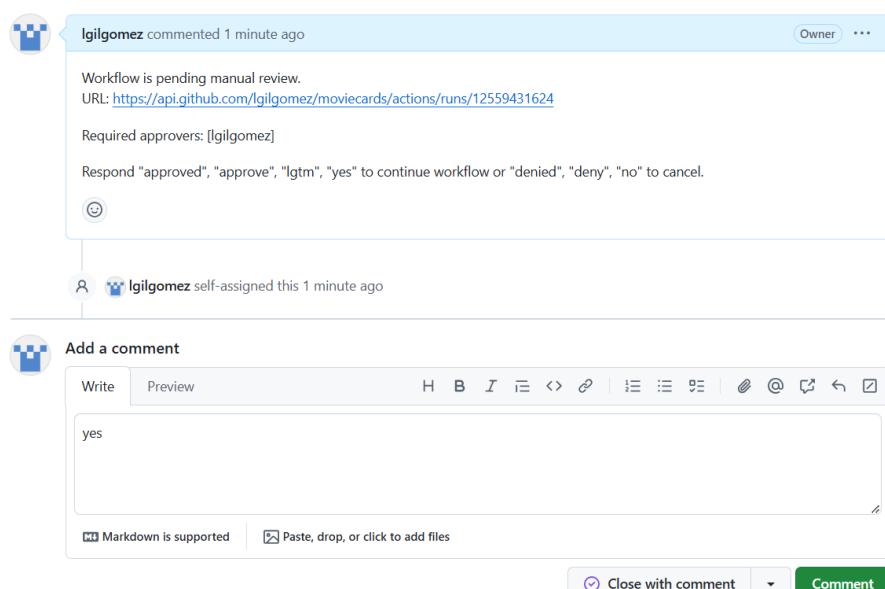
```
<body style="background-color: #lightgray">

<div class="container">
  <div th:if="${message != null}" class='alert alert-success' th:text="${message}" role='alert'></div>
  <h1 class="text-center" style="color: #darkred; background-color: #beige">Gestión de películas</h1>
  <hr>
```

Haciendo de nuevo push se hace merge de añadir-color-titulo con la rama master sin ningún tipo de conflicto, aprobando esta vez el despliegue manual de la aplicación:

Manual approval required for workflow run 12559431624 #7

 Open Igilgomez opened this issue 1 minute ago · 0 comments



<https://moviecards-gilgomez.azurewebsites.net/>



Como ha finalizado el issue, en el tablero Kanban hay que arrastrarlo desde la columna In Progress a la columna Done:

Todo 0
This item hasn't been started

In Progress 0
This is actively being worked on

Done 2
This has been completed

- moviecards #2
Añadir color de fondo a la página principal
- moviecards #3
Añadir color al título de la página principal

Por último, se cierra el milestone:

Aplicación con colores
Closed now (Last updated less than a minute ago)

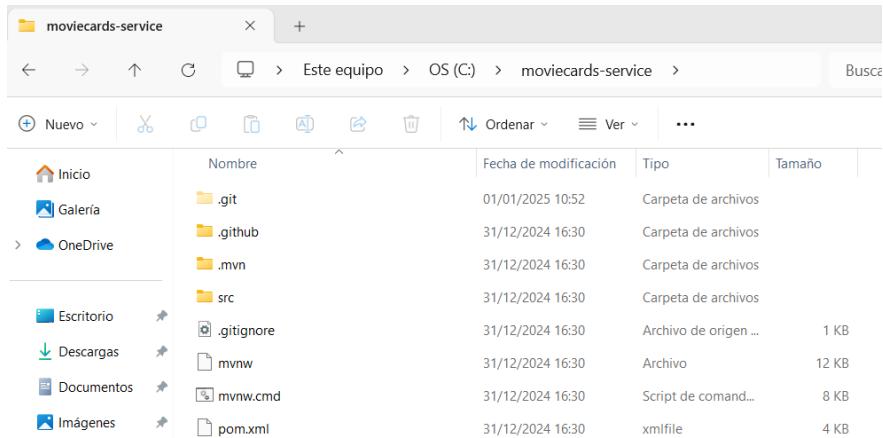
100% complete 0 open 3 closed
Edit Reopen Delete

Se añadirán colores a la página principal de la aplicación index.html

Apartado 2

El objetivo de este apartado es crear un nuevo repositorio llamado moviecards-service a partir del repositorio situado en la siguiente URL: <https://github.com/josehilar/moviecards-service>

Se descarga el código fuente, se descomprime y se crea un repositorio local mediante el comando git init.



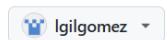
Seguidamente se crea el repositorio remoto y se sube el repositorio local con git remote y git push:

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Required fields are marked with an asterisk (*).

Owner * Repository name *



lgilgomez / moviecards-service

moviecards-service is available.

Great repository names are short and memorable. Need inspiration? How about [special-succotash](#) ?

Description (optional)

Public

Anyone on the internet can see this repository. You choose who can commit.

Private

You choose who can see and commit to this repository.

Initialize this repository with:

Add a README file

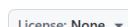
This is where you can write a long description for your project. [Learn more about READMEs](#).

Add .gitignore



Choose which files not to track from a list of templates. [Learn more about ignoring files](#).

Choose a license



A license tells others what they can and can't do with your code. [Learn more about licenses](#).

Se crea y configura una aplicación web en Azure para alojar este servicio creado exactamente de la misma forma que se hizo en el apartado 1 de la práctica:

Suscripción * ⓘ

Azure para estudiantes

└─ Grupo de recursos * ⓘ

(Nuevo) moviecards-service-gilgomez_group

[Crear nuevo](#)

Detalles de instancia

Nombre

moviecards-service-gilgomez

.azurewebsites.net

Se copia la parte de deploy del archivo generado por Azure en main.yaml con las siguientes modificaciones:

```
EXPLORER ... main.yaml x
MOVIECARDS-SERVICE
  .github\workflows\main.yaml
  .mvn
  src
  .gitignore
  mvnw
  mvnw.cmd
  pom.xml

.github > workflows > main.yaml
  jobs:
    deploy:
      environment:
        name: 'Production'
        url: ${{ steps.deploy-to-webapp.outputs.webapp-url }}
      steps:
        - name: Download artifact from build job
          uses: actions/download-artifact@v3
          with:
            name: moviecards-service-jar
        - name: Deploy to Azure Web App
          id: deploy-to-webapp
          uses: azure/webapps-deploy@v3
          with:
            app-name: 'moviecards-service-gilgomez'
            slot-name: 'Production'
            package: '*.jar'
            publish-profile: ${{ secrets.AZUREAPPSPROFILE_8D51462EAF8C427CAA789ED1F56DD7AE }}
```

Haciendo push se despliega automáticamente el proyecto en la siguiente URL y se prueban las siguientes operaciones del servicio en Postman: <https://moviecards-service-gilgomez.azurewebsites.net/>

Crear un actor

POST https://moviecards-service-gilgomez.azurewebsites.net/actors

Body (JSON)

```
1
2   "id": "",
3   "name": "Lady Gaga",
4   "birthDate": "1986-03-28",
5   "country": "EEUU",
6   "movies": []
```

Listar actores

GET https://moviecards-service-gilgomez.azurewebsites.net/actors

Body (Pretty)

```
1
2   {
3     "id": 1,
4     "name": "Lady Gaga",
5     "birthDate": "1986-03-28T00:00:00.000+00:00",
6     "country": "EEUU",
7     "movies": []
```

Consultar un actor por ID

GET https://moviecards-service-gilgomez.azurewebsites.net/actors/1

Body (Pretty)

```
1
2   {
3     "id": 1,
4     "name": "Lady Gaga",
5     "birthDate": "1986-03-28T00:00:00.000+00:00",
6     "country": "EEUU",
7     "movies": []
```

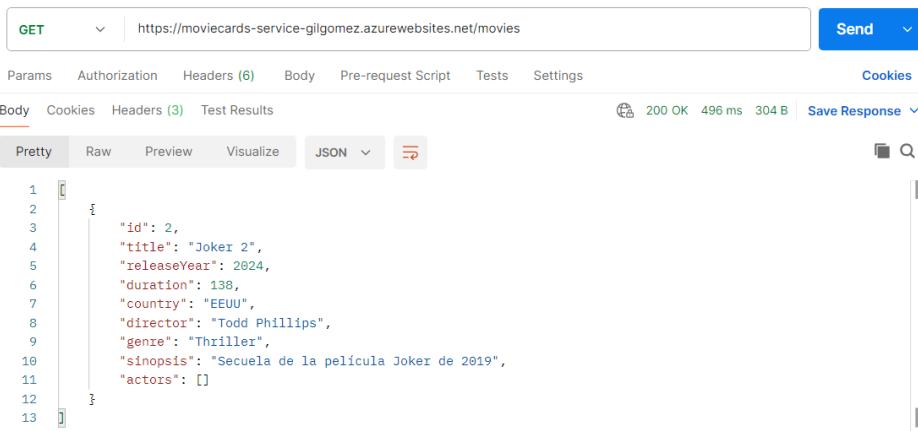
Crear una película

POST https://moviecards-service-gilgomez.azurewebsites.net/movies

Body (JSON)

```
1
2   {
3     "id": "",
4     "title": "Joker 2",
5     "releaseYear": 2024,
6     "duration": 130,
7     "country": "EEUU",
8     "director": "Todd Phillips",
9     "genre": "Thriller",
10    "sinopsis": "Secuela de la película Joker de 2019",
11    "actors": []
```

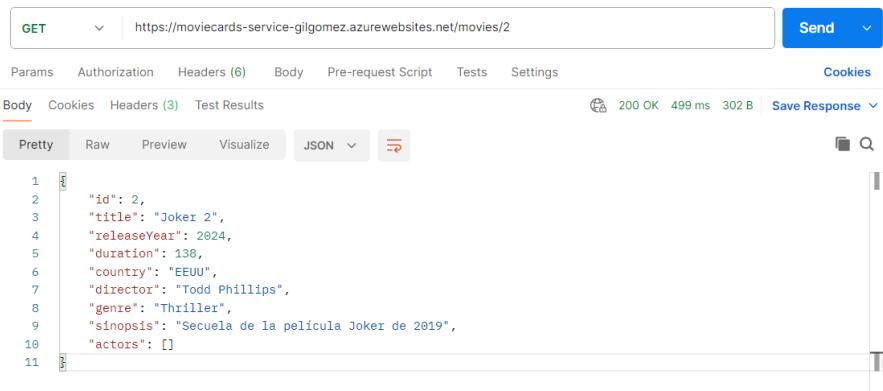
Listar películas



```
1
2
3
4
5
6
7
8
9
10
11
12
13
```

```
{  
    "id": 2,  
    "title": "Joker 2",  
    "releaseYear": 2024,  
    "duration": 138,  
    "country": "EEUU",  
    "director": "Todd Phillips",  
    "genre": "Thriller",  
    "sinopsis": "Secuela de la película Joker de 2019",  
    "actors": []  
}
```

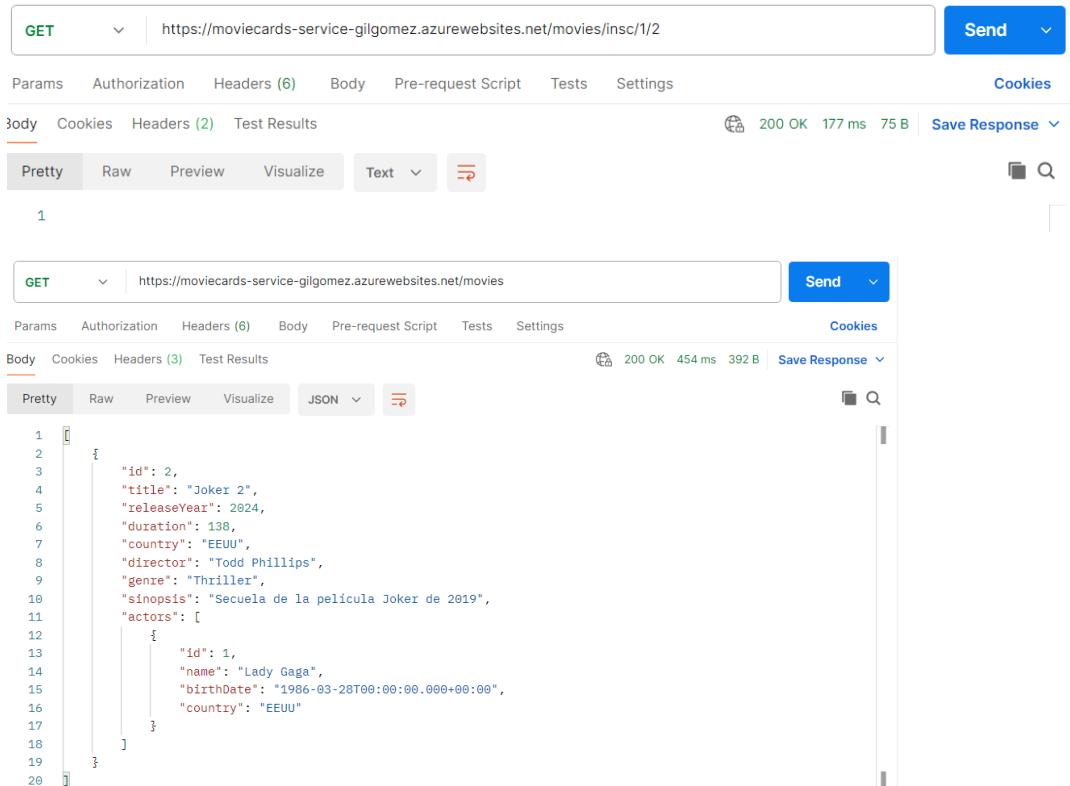
Consultar una película por ID



```
1
2
3
4
5
6
7
8
9
10
11
```

```
{  
    "id": 2,  
    "title": "Joker 2",  
    "releaseYear": 2024,  
    "duration": 138,  
    "country": "EEUU",  
    "director": "Todd Phillips",  
    "genre": "Thriller",  
    "sinopsis": "Secuela de la película Joker de 2019",  
    "actors": []  
}
```

Inscribir un actor en una película



```
1
```



```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
```

```
{  
    "id": 2,  
    "title": "Joker 2",  
    "releaseYear": 2024,  
    "duration": 138,  
    "country": "EEUU",  
    "director": "Todd Phillips",  
    "genre": "Thriller",  
    "sinopsis": "Secuela de la película Joker de 2019",  
    "actors": [  
        {  
            "id": 1,  
            "name": "Lady Gaga",  
            "birthDate": "1986-03-28T00:00:00.000+00:00",  
            "country": "EEUU"  
        }  
    ]  
}
```

Apartado 3

Se va a realizar una nueva versión de la aplicación moviecards, modificando el código del repositorio moviecards de la práctica del primer apartado para que utilice el servicio creado en el apartado anterior. Para ello, se va a generar un nuevo milestone llamado Modificación código apartado 3, que tenga a su vez dos issues para modificar el código de la aplicación en src/main y modificar el código de las pruebas en src/test:

The screenshot shows the GitHub interface for a repository named 'moviecards'. A new milestone is being created with the title 'Modificación código apartado 3'. The description field contains the following text:

```
Realizar una nueva versión de la aplicación moviecards, modificando el código del repositorio moviecards de la práctica del tema 5, para que utilice el servicio creado en el apartado anterior. Debe quedar funcionando en la misma URL de la versión anterior (moviecards:gilgomez)
```

Below the milestone creation form, a project board is displayed with three columns: 'Todo', 'In Progress', and 'Done'. Each column contains a list of tasks, some of which are highlighted in green or blue, indicating they have been modified.

Estado	Tareas
Todo	moviecards #8: Modificar el código de la aplicación en src/main moviecards #9: Modificar el código de las pruebas en src/test
In Progress	
Done	moviecards #2: Añadir color de fondo a la página principal moviecards #3: Añadir color al título de la página principal

Los cambios en src/main son los siguientes (líneas señaladas en verde y en azul):

```
J MovieCardsApplication.java M X
src > main > java > com > lauracercas > moviecards > J MovieCardsApplication.java
1 package com.lauracercas.moviecards;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.context.annotation.Bean;
6 import org.springframework.web.client.RestTemplate;
7
8 /**
9 * Autor: Laura Cercas Ramos
10 * Proyecto: TFM Integración Continua con GitHub Actions
11 * Fecha: 04/06/2024
12 */
13 @SpringBootApplication
14 public class MovieCardsApplication {
15
16     public static void main(String[] args) {
17         SpringApplication.run(MovieCardsApplication.class, args);
18     }
19
20     @Bean
21     public RestTemplate template() {
22         RestTemplate template = new RestTemplate();
23         return template;
24     }
25
26 }
```

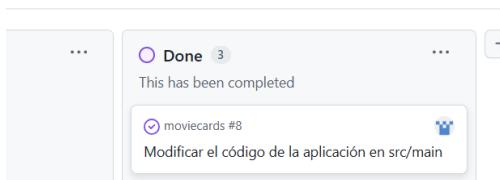
```
J CardController.java M X
src > main > java > com > lauracercas > moviecards > controller > J CardController.java
25 @Controller
26 public class CardController {
27
28
29     private final MovieService movieService;
30     @Autowired
31     ActorService actorService;
32     private final CardService cardService;
```

```
J ActorServiceImpl.java M X
src > main > java > com > lauracercas > moviecards > service > actor > J ActorServiceImpl.java
13 /**
14  * ...
15 */
16 @Service
17 public class ActorServiceImpl implements ActorService {
18
19     @Autowired
20     RestTemplate template;
21     String url = "https://moviecards-service-gilgomez.azurewebsites.net/actors";
22
23     @Override
24     public List<Actor> getAllActors() {
25         Actor[] actores = template.getForObject(url, Actor[].class);
26         List<Actor> actoresList = Arrays.asList(actores);
27         return actoresList;
28     }
29
30     @Override
31     public Actor save(Actor actor) {
32         if (actor.getId() != null && actor.getId() > 0) {
33             template.put(url, actor);
34         } else {
35             actor.setId(0);
36             template.postForObject(url, actor, String.class);
37         }
38         return actor;
39     }
40
41     @Override
42     public Actor getActorById(Integer actorId) {
43         Actor actor = template.getForObject(url+"/"+actorId, Actor.class);
44         return actor;
45     }
46 }
```

```
J MovieServiceImpl.java M X
src > main > java > com > lauracercas > moviecards > service > movie > J MovieServiceImpl.java
13 /**
14  * ...
15 */
16 @Service
17 public class MovieServiceImpl implements MovieService {
18
19     @Autowired
20     RestTemplate template;
21     String url = "https://moviecards-service-gilgomez.azurewebsites.net/movies";
22
23     @Override
24     public List<Movie> getAllMovies() {
25         Movie[] movies = template.getForObject(url, Movie[].class);
26         List<Movie> moviesList = Arrays.asList(movies);
27         return moviesList;
28     }
29
30     @Override
31     public Movie save(Movie movie) {
32         if (movie.getId() != null && movie.getId() > 0) {
33             template.put(url, movie);
34         } else {
35             movie.setId(0);
36             template.postForObject(url, movie, String.class);
37         }
38         return movie;
39     }
40
41     @Override
42     public Movie getMovieById(Integer movieId) {
43         Movie movie = template.getForObject(url+"/"+movieId, Movie.class);
44         return movie;
45     }
46 }
```

```
J CardServiceImpl.java M X
src > main > java > com > lauracercas > moviecards > service > card > J CardServiceImpl.java
13  /**
17   */
18  @Service
19  public class CardServiceImpl implements CardService {
20
21      @Autowired
22      ActorService actorService;
23
24      @Autowired
25      MovieService movieService;
26
```

Una vez se terminan los cambios se actualiza dicho issue a Done en el tablero Kanban:



Los cambios en src/test son los siguientes (líneas señaladas en verde y en azul):

```
J ActorServiceImplTest.java M X
src > test > java > com > lauracercas > moviecards > unittest > service > J ActorServiceImplTest.java
29  class ActorServiceImplTest {
30
31      @Mock
32      private RestTemplate template;
33      @InjectMocks
34      private ActorService sut = new ActorServiceImpl();
35      private AutoCloseable closeable;
36
37      @BeforeEach
38      void setUp() {
39          closeable = openMocks(this);
40      }
41
42      @AfterEach
43      void tearDown() throws Exception {
44          closeable.close();
45      }
46
47      @Test
48      public void shouldGetAllActors() {
49          Actor actors[] = new Actor[2];
50          actors[0] = new Actor();
51          actors[1] = new Actor();
52
53          when(template.getForObject(anyString(), any())).thenReturn(actors);
54
55          @Test
56          public void shouldGetActorById() {
57              Actor actor = new Actor();
58              actor.setId(1);
59              actor.setName("Sample Actor");
60
61              when(template.getForObject(anyString(), any())).thenReturn(actor);
62          }
63      }
64
65
66
67
```

```

J MovieServiceImplTest.java M X
src > test > java > com > laurarcas > moviecards > unittest > service > J MovieServiceImplTest.java
29 class MovieServiceImplTest {
30     @Mock
31     private RestTemplate template;
32     @InjectMocks
33     private MovieService sut = new MovieServiceImpl();
34     private AutoCloseable closeable;
35
36     @BeforeEach
37     public void setUp() {
38         closeable = openMocks(this);
39     }
40
41     @AfterEach
42     void tearDown() throws Exception {
43         closeable.close();
44     }
45
46     @Test
47     public void shouldGetAllMovies() {
48         Movie movies[] = new Movie[2];
49         movies[0] = new Movie();
50         movies[1] = new Movie();
51
52         when(template.getForObject(anyString(), any())).thenReturn(movies);
53
54         @Test
55         public void shouldGetMovieById() {
56             Movie movie = new Movie();
57             movie.setId(1);
58             movie.setTitle("Sample Movie");
59
60             when(template.getForObject(anyString(), any())).thenReturn(movie);
61
62         }
63
64     }
65
66

```

Una vez se terminan los cambios se actualiza dicho issue a Done en el tablero Kanban:



Tras hacer push la aplicación queda desplegada en la URL <https://moviecards-gilgomez.azurewebsites.net/>.

Apartado 4

En este apartado se modifica el código del repositorio del servicio moviecards-service para añadir un nuevo atributo de tipo fecha en la clase Actor llamado deadDate, que albergará la fecha de fallecimiento de un actor. Para ello, se hacen las siguientes modificaciones en el fichero Actor.java:

```

J Movie.java
> repositories
  < service
    < actor
      J ActorService.java
28
29
30
31
32

```

```

@DateTimeFormat(pattern = "yyyy-MM-dd")
private Date deadDate;
private String country;

```

```

66     public void setBirthDate(Date birthDate) {
67         this.birthDate = birthDate;
68     }
69
70     public Date getDeadDate() {
71         return deadDate;
72     }
73
74     public void setDeadDate(Date deadDate) {
75         this.deadDate = deadDate;
76     }
77
78     public String getCountry() {
79         return country;
80     }
81
82
83
84
85
86
87
88
89
90
91
92
93
94     @Override
95     public boolean equals(Object o) {
96         if (this == o)
97             return true;
98         if (o == null || getClass() != o.getClass())
99             return false;
100        Actor actor = (Actor) o;
101        return Objects.equals(id, actor.id) && Objects.equals(name, actor.name)
102            && Objects.equals(birthDate, actor.birthDate)
103            && Objects.equals(deadDate, actor.deadDate) && Objects.equals(country, actor.country);
104    }
105
106    @Override
107    public int hashCode() {
108        return Objects.hash(id, name, birthDate, deadDate, country);
109    }
110}

```

Haciendo push se despliega automáticamente el proyecto en la siguiente URL y se prueban las siguientes operaciones del servicio en Postman: <https://moviecards-service-gilgomez.azurewebsites.net/>

Crear un actor (con fecha de fallecimiento)

POST <https://moviecards-service-gilgomez.azurewebsites.net/actors> Send

Params Authorization Headers (8) Body **Pre-request Script** Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary **JSON** Beautify

```

1 {
2     "id": "",
3     "name": "Maggie Smith",
4     "birthDate": "1934-12-28",
5     "deadDate": "2024-09-27",
6     "country": "UK",
7     "movies": []
8 }
9

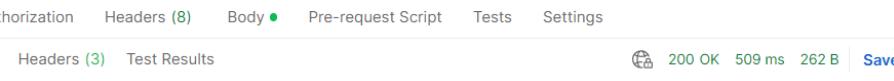
```

Body Cookies Headers (2) Test Results **200 OK 1612 ms 75 B Save Response**

Pretty Raw Preview Visualize Text

1

Listar actores



GET https://moviecards-service-gilgomez.azurewebsites.net/actors

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

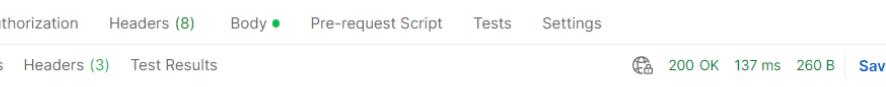
Body Cookies Headers (3) Test Results

200 OK 509 ms 262 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 1,
3   "name": "Maggie Smith",
4   "birthDate": "1934-12-28T00:00:00.000+00:00",
5   "deadDate": "2024-09-27T00:00:00.000+00:00",
6   "country": "UK",
7   "movies": []
8 }
```

Consultar un actor por ID



GET https://moviecards-service-gilgomez.azurewebsites.net/actors/1

Params Authorization Headers (8) Body ● Pre-request Script Tests Settings Cookies

Body Cookies Headers (3) Test Results 200 OK 137 ms 260 B Save Response

Pretty Raw Preview Visualize JSON

```
1
2     "id": 1,
3     "name": "Maggie Smith",
4     "birthDate": "1934-12-28T00:00:00.000+00:00",
5     "deadDate": "2024-09-27T00:00:00.000+00:00",
6     "country": "UK",
7     "movies": []
8
```

Apartado 5

En este apartado se incorporan todos los cambios relativos al atributo deadDate al proyecto moviecards, de tal forma que para crear un nuevo actor se pida al usuario también la fecha de fallecimiento y en la página de listado de actores aparezca una nueva columna con la fecha de fallecimiento. Para ello, se crea previamente un milestone llamado Apartado 5 práctica final ICDA, que tendrá una serie de issues correspondientes a los subapartados a desarrollar a continuación:

⌚ 1 Open ✓ 2 Closed

Sort ▾

Apartado 5 práctica final ICDA

No due date ⓘ Last updated less than a minute ago

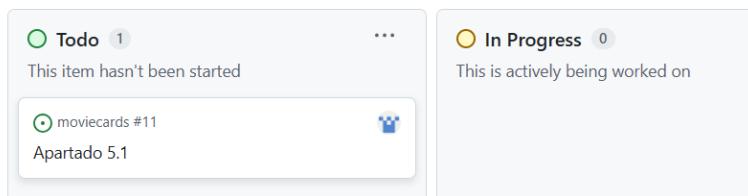
0% complete 0 open 0 closed

Edit Close Delete

Resolución de cada uno de los subapartados que comprenden el apartado 5 de la práctica final de ICDA.

5.1. Añadir fase stage al workflow del proyecto

Se añade al workflow del proyecto la fase stage para desplegar la aplicación en un entorno de pre-producción (situado entre “qa” y “deploy”). Esto queda indicado en el issue que es creado a continuación:



Se crea y configura una aplicación web en Azure para alojar este servicio creado exactamente de la misma forma que se hizo en los apartados 1 y 2 de la práctica.

Suscripción * ⓘ Azure para estudiantes

Grupo de recursos * ⓘ (Nuevo) moviecards-pre_group Crear nuevo

Detalles de instancia

Nombre moviecards-pre .azurewebsites.net

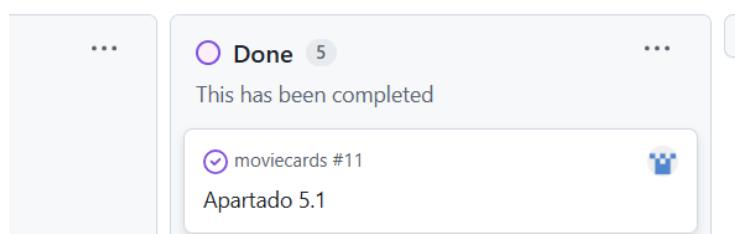
El fichero main.yaml es modificado de la siguiente manera:

```

81   mvn sonar:sonar -Dsonar.host.url=http://localhost:9000 -Dsonar.qualitygate.wait=true -Dsonar.log.level=INFO
82 stage:
83   runs-on: ubuntu-latest
84   needs: qa
85   if: github.ref=='refs/heads/master'
86   environment:
87     name: 'Pre-Production'
88     url: ${{ steps.deploy-to-webapp.outputs.webapp-url }}
89
90 steps:
91   - name: Download artifact from build job
92     uses: actions/download-artifact@v4
93     with:
94       name: moviecards-java
95
96   - name: Deploy to Azure Pre-Production Web App
97     id: deploy-to-preproduction
98     uses: azure/webapps-deploy@v3
99     with:
100       app-name: 'moviecards-pre'
101       slot-name: 'Production'
102       package: '*.jar'
103       publish-profile: ${{ secrets.AZUREAPPSPROFILE_BBB091937F15416EA711AC63710F0D5E }}
104 deploy:
105   runs-on: ubuntu-latest
106   needs: stage

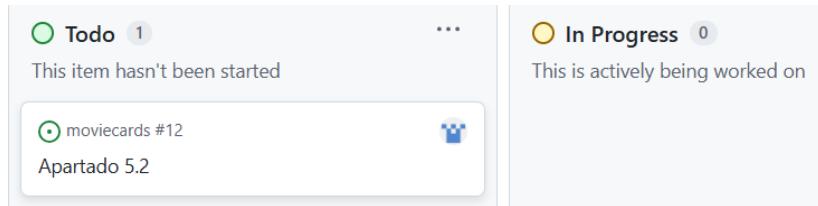
```

Se suben los cambios al repositorio remoto. Una vez se terminan los cambios se actualiza dicho issue a Done en el tablero Kanban:



5.2. Modificar el código src

Se modifica el código de la aplicación para manejar la fecha de fallecimiento de los actores. Esto queda indicado en el issue que es creado a continuación:



Se crea una nueva rama llamada añadir-fallecimiento-actores:

```
PS C:\moviecards> git branch añadir-fallecimiento-actores
PS C:\moviecards> git checkout añadir-fallecimiento-actores
M       .github/workflows/main.yaml
Switched to branch 'añadir-fallecimiento-actores'
PS C:\moviecards> git branch
  añadir-color-fondo
  añadir-color-titulo
* añadir-fallecimiento-actores
  master
PS C:\moviecards>
```

Modificar actors/list.html:

```
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
```

		<tr> <th scope="col">Identificador</th> <th scope="col">Nombre</th> <th scope="col">Fecha Nacimiento</th> <th scope="col">Fecha Fallecimiento</th> <th scope="col">País</th> <th scope="col">Editar</th> </tr> </thead> <tbody> <tr th:each="actor : \${actors}"> <td th:text="\${actor.id}"></td> <td th:text="\${actor.name}"></td> <td th:text="#{dates.format(actor.birthDate, 'dd-MM-yyyy')}"></td> <td th:text="#{dates.format(actor.deadDate, 'dd-MM-yyyy')}"></td> <td th:text="\${actor.country}"></td> <td> <a th:href="@{'/editActor/' + \${actor.id}}" class="btn btn-primary">Editar </td> </tr> </tbody>
--	--	--

Modificar actors/form.html:

```
65
66
67
68
69
70
71
72
73
74
75
76
77
78
```

<label for="deadDate" class="form-label">Fecha fallecimiento</label>
 <input
 type="date"
 class="form-control"
 th:field="*{deadDate}"
 id="deadDate"
 name="deadDate"
 placeholder="Escriba la fecha de fallecimiento"
 required="required"
 />

<label for="country" class="form-label">País</label>

Modificar model/Actor.java:

```
25
26
27
28
```

`@DateTimeFormat(pattern = "yyyy-MM-dd")
private Date deadDate;`

```

65     public Date getDeadDate() {
66         return deadDate;
67     }
68
69     public void setDeadDate(Date deadDate) {
70         this.deadDate = deadDate;
71     }
72
73     public String getCountry() {
74
75         @Override
76         public boolean equals(Object o) {
77             if (this == o) return true;
78             if (o == null || getClass() != o.getClass()) return false;
79             Actor actor = (Actor) o;
80             return Objects.equals(id, actor.id) && Objects.equals(name, actor.name)
81                 && Objects.equals(birthDate, actor.birthDate) && Objects.equals(deadDate, actor.deadDate)
82                 && Objects.equals(country, actor.country);
83         }
84
85         @Override
86         public int hashCode() {
87             return Objects.hash(id, name, birthDate, deadDate, country);
88         }
89     }
90 }
```

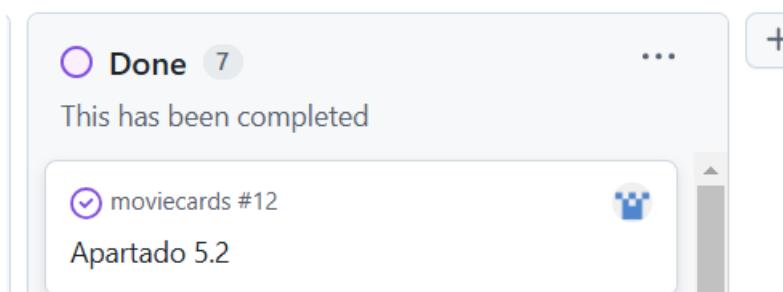
Hacer merge de añadir-fallecimiento-actores con la rama master. Al terminar volver a la rama master mediante git checkout master:

```

PS C:\moviecards> git add .
PS C:\moviecards> git status
On branch añadir-fallecimiento-actores
Changes to be committed:
(use "git restore --staged <file>..." to unstage)
    modified:   .github/workflows/main.yaml
    modified:   src/main/java/com/lauracercas/moviecards/model/Actor.java
    modified:   src/main/resources/templates/actors/form.html
    modified:   src/main/resources/templates/actors/list.html

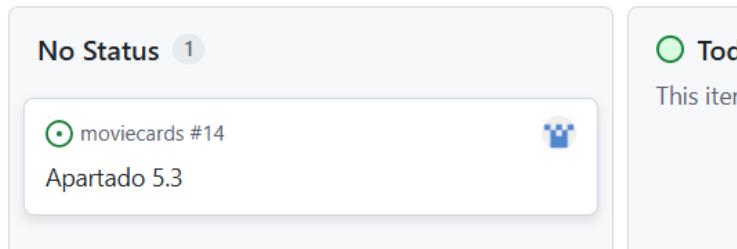
PS C:\moviecards> git commit -m "Modificación en src/main y templates para manejar fallecimiento de actores"
[añadir-fallecimiento-actores d1468c8] Modificación en src/main y templates para manejar fallecimiento de actores
 4 files changed, 51 insertions(+), 3 deletions(-)
PS C:\moviecards> git push origin añadir-fallecimiento-actores
Enumerating objects: 35, done.
Counting objects: 100% (35/35), done.
Delta compression using up to 8 threads
Compressing objects: 100% (13/13), done.
Writing objects: 100% (18/18), 1.56 KiB | 800.00 KiB/s, done.
Total 18 (delta 7), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (7/7), completed with 7 local objects.
remote:
remote: Create a pull request for 'añadir-fallecimiento-actores' on GitHub by visiting:
remote:     https://github.com/lgilgomez/moviecards/pull/new/a%C3%Bladir-fallecimiento-actores
remote:
To https://github.com/lgilgomez/moviecards.git
 * [new branch]      añadir-fallecimiento-actores -> añadir-fallecimiento-actores
PS C:\moviecards>
```

Una vez se terminan los cambios se actualiza dicho issue a Done en el tablero Kanban:



5.3. Modificar pruebas unitarias

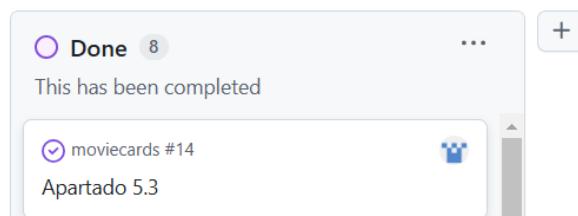
Modificar las pruebas unitarias para tener en cuenta la fecha de fallecimiento de los actores. Esto queda indicado en el issue que es creado a continuación:



Modificar unittest/model/ActorTest.java:

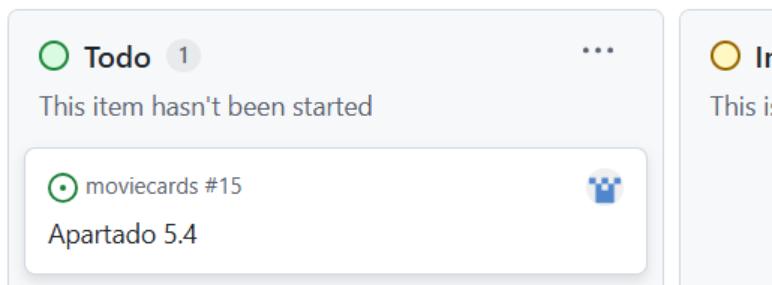
```
39 |     @Test
40 |     void testSetGetDeadDate() {
41 |         Date deadDateExample = new Date();
42 |         actor.setDeadDate(deadDateExample);
43 |         assertEquals(deadDateExample, actor.getDeadDate());
44 |
45 |     }
```

Se suben los cambios al repositorio remoto. Una vez se terminan los cambios se actualiza dicho issue a Done en el tablero Kanban:



5.4. Modificar pruebas de integración

Modificar las pruebas de integración para tener en cuenta la fecha de fallecimiento de los actores. Esto queda indicado en el issue que es creado a continuación:



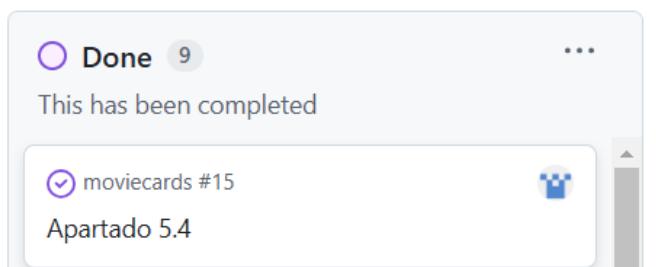
Modificar integrationtest/repositories/ActorJPAIT.java:

```

25 |     @Test
26 |     public void testSaveActor() {
27 |         Actor actor = new Actor();
28 |         actor.setName("actor");
29 |         actor.setBirthDate(new Date());
30 |         actor.setDeadDate(new Date());
31 |         actor.setCountry("spain");
32 |
33 |         Actor savedActor = actorJPA.save(actor);
34 |
35 |         assertNotNull(savedActor.getId());
36 |
37 |         Optional<Actor> foundActor = actorJPA.findById(savedActor.getId());
38 |
39 |         assertTrue(foundActor.isPresent());
40 |         assertEquals(savedActor, foundActor.get());
41 |
42 |
43 |     @Test
44 |     public void testFindById() {
45 |         Actor actor = new Actor();
46 |         actor.setName("actor");
47 |         actor.setBirthDate(new Date());
48 |         actor.setDeadDate(new Date());
49 |         Actor savedActor = actorJPA.save(actor);
50 |
51 |         Optional<Actor> foundActor = actorJPA.findById(savedActor.getId());
52 |
53 |         assertTrue(foundActor.isPresent());
54 |         assertEquals(savedActor, foundActor.get());
55 |
56 |     }

```

Se suben los cambios al repositorio remoto. Una vez se terminan los cambios se actualiza dicho issue a Done en el tablero Kanban:



5.5. Modificar pruebas funcionales

Modificar las pruebas funcionales (end to end) para tener en cuenta la fecha de fallecimiento de los actores. Esto queda indicado en el issue que es creado a continuación:

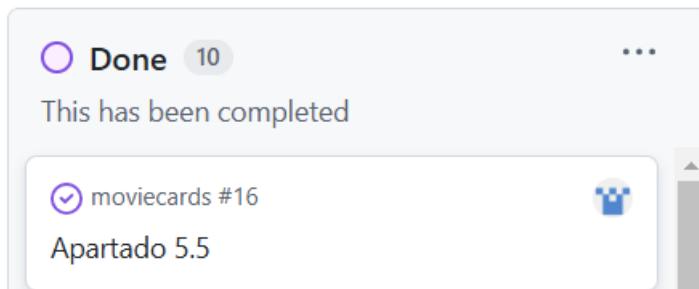
The image shows two adjacent Kanban cards. The left card is green and labeled 'Todo 1'. It has the text 'This item hasn't been started' and a sub-section with a green circle icon, 'moviecards #16', and 'Apartado 5.5'. The right card is yellow and labeled 'In Progress 0'. It has the text 'This is actively being worked on'.

Modificar endtoendtest/ActorE2ETest.java:

```

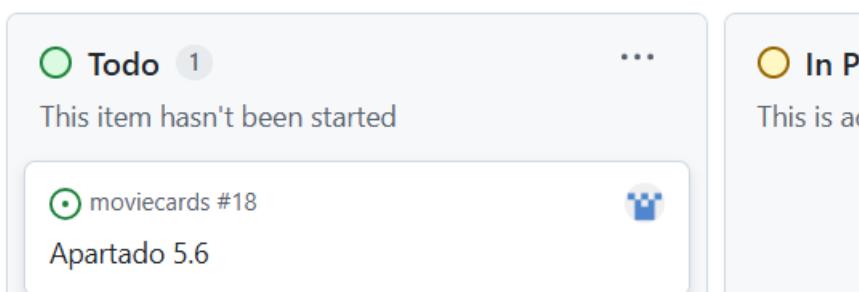
43     @Test
44     public void testPageLoad() {
45         driver.get("http://localhost:8089/actors/new");
46         assertEquals("FichasPelículasApp | Aplicación de gestión de fichas de películas", driver.getTitle());
47
48         assertTrue(driver.findElement(By.id("name")).isDisplayed());
49         assertTrue(driver.findElement(By.id("birthDate")).isDisplayed());
50         assertTrue(driver.findElement(By.id("deadDate")).isDisplayed());
51         assertTrue(driver.findElement(By.id("country")).isDisplayed());
52     }
53
54
55     @Test
56     public void testNewActorTitle() {
57         driver.get("http://localhost:8089/actors/new");
58         WebElement title = driver.findElement(By.className("title"));
59         assertEquals(NEW_ACTOR_TITLE, title.getText());
60     }
61
62     @Test
63     public void testListActors() {
64         driver.get("http://localhost:8089/actors");
65         WebElement title = driver.findElement(By.className("card-header"));
66         assertEquals("Listado Actores", title.getText());
67
68         WebElement table = driver.findElement(By.className("table-hover"));
69
70         WebElement thead = table.findElement(By.tagName("thead"));
71         assertTrue(thead.isDisplayed());
72
73         WebElement headerRow = thead.findElement(By.tagName("tr"));
74         assertEquals("Identificador", headerRow.findElements(By.tagName("th")).get(0).getText());
75         assertEquals("Nombre", headerRow.findElements(By.tagName("th")).get(1).getText());
76         assertEquals("Fecha Nacimiento", headerRow.findElements(By.tagName("th")).get(2).getText());
77         assertEquals("Fecha Fallecimiento", headerRow.findElements(By.tagName("th")).get(3).getText());
78         assertEquals("Pais", headerRow.findElements(By.tagName("th")).get(4).getText());
79         assertEquals("Editar", headerRow.findElements(By.tagName("th")).get(5).getText());
80     }
81 }
```

Se suben los cambios al repositorio remoto. Una vez se terminan los cambios se actualiza dicho issue a Done en el tablero Kanban:



5.6. Garantía de calidad

Hacer que no se despliegue la aplicación a producción si existen al menos 5 fallos críticos. Esto queda indicado en el issue que es creado a continuación:



Abriendo el servidor de SonarQube se observan cuáles son los fallos críticos que tiene la aplicación:

The screenshot shows the SonarQube interface for the 'moviecards' project. The 'Issues' tab is selected. On the left, there's a sidebar with filters for 'Type' (Bug, Vulnerability, Code Smell), 'Severity' (Blocker, Critical, Major, Minor, Info), and various scope and resolution filters. The main area lists issues across two files: 'src/.../moviecards/controller/ActorController.java' and 'src/.../moviecards/controller/MovieController.java'. Both files have four critical code smell issues. Each issue is detailed with its type (Code Smell), severity (Critical), status (Open, Not assigned), effort (e.g., 8min, 10min), and a comment. The interface includes a 'Bulk Change' button and a summary at the top right indicating 1 / 8 issues and 1h 12min effort.

Se va cambiando código según lo que indican los fallos críticos.

Modificaciones en ActorController.java:

```

23  @Controller
24  public class ActorController {
25
26      private final ActorService actorService;
27      private final String actorStr = "actor";
28      private final String titleStr = "title";
29      private final String actorsFormsStr = "actors/form";
30
31      @GetMapping("actors/new")
32      public String newActor(Model model) {
33          model.addAttribute(actorStr, new Actor());
34          model.addAttribute(titleStr, Messages.NEW_ACTOR_TITLE);
35          return actorsFormsStr;
36      }
37
38      @PostMapping("saveActor")
39      public String saveActor(@PathVariable Actor actor, BindingResult result, Model model) {
40          if (result.hasErrors()) {
41              return actorsFormsStr;
42          }
43          Actor actorSaved = actorService.save(actor);
44          if (actor.getId() != null) {
45              model.addAttribute("message", Messages.UPDATED_ACTOR_SUCCESS);
46          } else {
47              model.addAttribute("message", Messages.SAVED_ACTOR_SUCCESS);
48
49          model.addAttribute(actorStr, actorSaved);
50          model.addAttribute(titleStr, Messages.EDIT_ACTOR_TITLE);
51          return actorsFormsStr;
52      }
53
54      @GetMapping("editActor/{actorId}")
55      public String editActor(@PathVariable Integer actorId, Model model) {
56          Actor actor = actorService.getActorById(actorId);
57          List<Movie> movies = actor.getMovies();
58          model.addAttribute(actorStr, actor);
59          model.addAttribute("movies", movies);
60
61          model.addAttribute(titleStr, Messages.EDIT_ACTOR_TITLE);
62
63          return actorsFormsStr;
64      }
65
66      @GetMapping("editActor/{actorId}/")
67      public String editActor(@PathVariable Integer actorId, Model model) {
68          Actor actor = actorService.getActorById(actorId);
69          List<Movie> movies = actor.getMovies();
70          model.addAttribute(actorStr, actor);
71          model.addAttribute("movies", movies);
72
73          model.addAttribute(titleStr, Messages.EDIT_ACTOR_TITLE);
74
75          return actorsFormsStr;
76      }

```

Modificaciones en MovieController.java:

```
23  @Controller
24  public class MovieController {
25
26      private final MovieService movieService;
27      private final String movieStr = "movie";
28      private final String titleStr = "title";
29      private final String moviesFormsStr = "movies/form";
30
31
32
33
34
35
36
37
38
39
40
41      @GetMapping("movies/new")
42      public String newMovie(Model model) {
43          model.addAttribute(movieStr, new Movie());
44          model.addAttribute(titleStr, Messages.NEW_MOVIE_TITLE);
45          return moviesFormsStr;
46      }
47
48      @PostMapping("saveMovie")
49      public String saveMovie(@PathVariable Movie movie, BindingResult result, Model model) {
50          if (result.hasErrors()) {
51              return moviesFormsStr;
52          }
53          Movie movieSaved = movieService.save(movie);
54          if (movieSaved.getId() != null) {
55              model.addAttribute("message", Messages.UPDATED_MOVIE_SUCCESS);
56          } else {
57              model.addAttribute("message", Messages.SAVED_MOVIE_SUCCESS);
58          }
59
60          model.addAttribute(movieStr, movieSaved);
61          model.addAttribute(titleStr, Messages.EDIT_MOVIE_TITLE);
62          return moviesFormsStr;
63      }
64
65      @GetMapping("editMovie/{movieId}")
66      public String editMovie(@PathVariable Integer movieId, Model model) {
67          Movie movie = movieService.getMovieById(movieId);
68          List<Actor> actors = movie.getActors();
69          model.addAttribute(movieStr, movie);
70          model.addAttribute("actors", actors);
71
72          model.addAttribute(titleStr, Messages.EDIT_MOVIE_TITLE);
73
74          return moviesFormsStr;
75      }
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
```

Modificar regla de control calidad en Quality Gates de tal forma que si se detectan 5 o más problemas críticos significa que el proyecto no ha aprobado la calidad exigida de software:

Update Condition

Quality Gate fails when

Critical Issues

Operator

is greater than

Value

4

Update Condition Cancel

Modificar el archivo main.yaml de forma que si el proyecto suspende la calidad de software no debe continuar a las siguientes fases de despliegue:

```
62 qa:
63   needs: test
64   runs-on: self-hosted
65   continue-on-error: false
66   steps:
```

Se suben los cambios al repositorio remoto, pudiéndose observar que todas las fases del workflow se ejecutan correctamente, ya que han sido eliminados todos los fallos críticos de la aplicación.



En la siguiente URL se pueden ver los resultados finales de moviecards-gilgomez:

<https://moviecards-gilgomez.azurewebsites.net/>

The screenshot shows two browser windows. The top window displays the 'Listado Actores' page with columns for Identificador, Nombre, Fecha Nacimiento, Fecha Fallecimiento, País, and Editar. The bottom window shows the 'Nuevo Actor' form with fields for Nombre (with placeholder 'Escriba el nombre del actor'), Fecha nacimiento (dd/mm/aaaa), Fecha fallecimiento (dd/mm/aaaa), País (with placeholder 'Escriba el país del actor'), and a 'Guardar' button. Both windows have the URL https://moviecards-gilgomez.azurewebsites.net/.

Una vez se terminan los cambios se actualiza dicho issue a Done en el tablero Kanban, cerrando así el milestone llamado Apartado 5 práctica final ICDA:

