

PRÁCTICA FINAL



**ESCUELA POLITÉCNICA SUPERIOR
MÁSTER EN DESARROLLO ÁGIL PARA LA WEB**

2025-2026

Asignatura: Integración Continua en el Desarrollo Ágil

Alumno: Juan Higuero López

Fecha: 22 de febrero de 2026

Índice:

1. Introducción.....	1
2. Desarrollo de la Práctica (Tema 5)	1
2.1. Implementación de Build y Test	1
2.2. Implementación de QA.....	2
2.3. Despliegue en Microsoft Azure	3
2.4. Aprobación Manual	4
2.5. Planificación Ágil con GitHub Projects	6
2.6. Gestión de Ramas y Resolución de Conflictos.....	7
3. Creación de moviecards-service	11
4. Refactorización de la Aplicación Principal	12
5. Implementación del Atributo deadDate.....	13
6. Entorno de Pre-producción y Ciclo de Calidad	13

1. Introducción

El presente proyecto se enmarca en la evolución de la plataforma Moviecards, una aplicación web desarrollada con Spring Boot para la gestión de fichas de actores y películas. El objetivo principal de esta fase del desarrollo ha sido la modernización de su arquitectura y la automatización total de su ciclo de vida mediante técnicas de Integración Continua (CI) y Despliegue Continuo (CD) utilizando GitHub Actions.

Tradicionalmente, la aplicación operaba bajo una estructura monolítica con una base de datos local. En esta actualización, se ha procedido a la extracción de la lógica de negocio hacia un servicio externo denominado moviecards-service, desplegado de forma independiente en Azure Web Apps. Este cambio ha permitido que la aplicación principal actúe como un cliente REST, consumiendo datos a través de una API externa, lo que mejora la escalabilidad y el desacoplamiento del sistema.

Además de la reestructuración arquitectónica, se ha implementado un flujo de trabajo automatizado que garantiza la calidad del software en cada cambio realizado. Este flujo incluye:

- **Gestión Ágil:** Organización del trabajo mediante Milestones e Issues en GitHub para el seguimiento de tareas.
- **Validación Técnica:** Ejecución de baterías de pruebas unitarias, de integración y funcionales (End-to-End) para validar nuevas funcionalidades como la fecha de fallecimiento de los actores.
- **Control de Calidad:** Integración con SonarQube para monitorizar la deuda técnica y asegurar que ningún código con problemas críticos llegue a producción.
- **Estrategia de Despliegue:** Uso de entornos de pre-producción (Stage) y despliegue final en producción supeditado a aprobaciones manuales.

2. Desarrollo de la Práctica (Tema 5)

En este bloque de la asignatura se procedió a la preparación de toda la infraestructura necesaria para el ciclo de vida del software. El trabajo comenzó con la creación de cuentas en GitHub y Azure, y la instalación de los recursos tecnológicos necesarios: Visual Studio, Maven, GitHub Desktop y Docker.

2.1. Implementación de Build y Test

Se desplegó un contenedor de SonarQube mediante Docker para el análisis estático de código. Tras crear el repositorio en GitHub, se inició la construcción del archivo main.yaml, siguiendo un proceso de validación iterativo en cada uno de sus componentes:

- **Apartado Build:** Se configuró la etapa de compilación. Para verificar la robustez del pipeline, se provocó un error forzado en la construcción,

confirmando que el flujo se detenía correctamente, y se procedió a su posterior corrección.

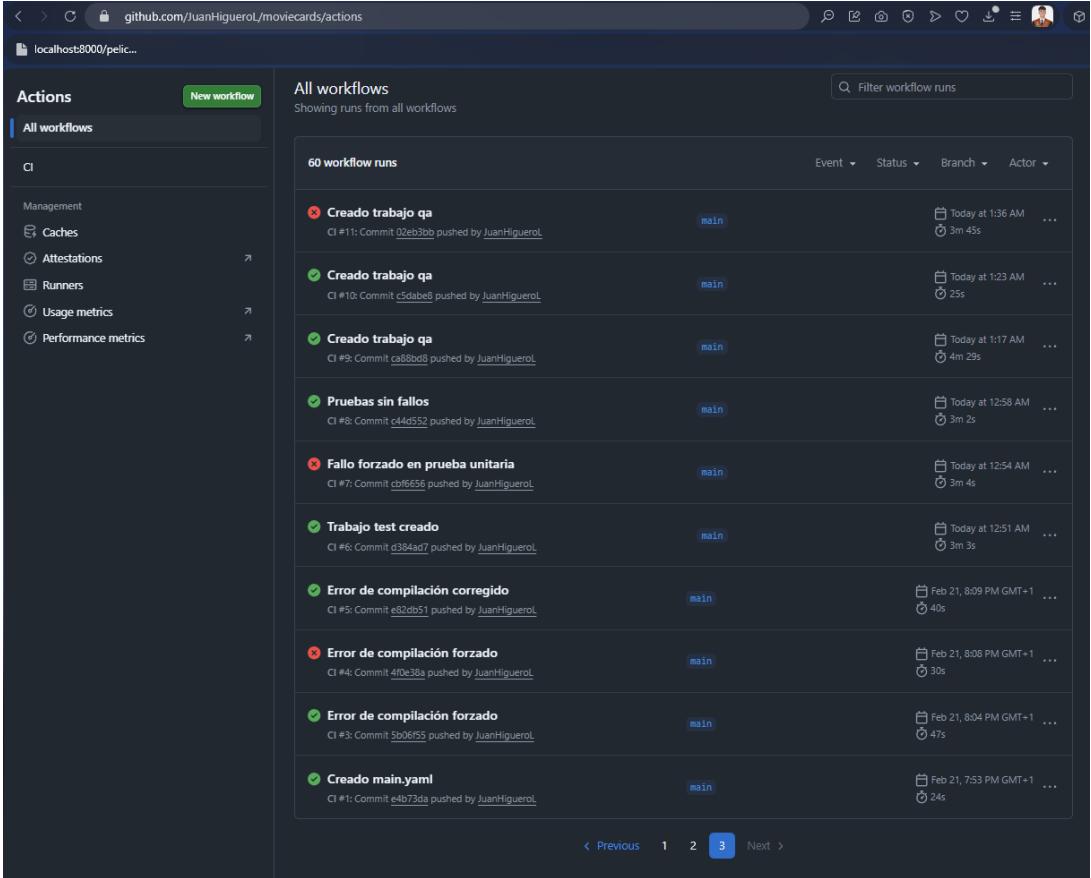
- **Apartado Test:** Se implementó la fase de pruebas automáticas siguiendo las directrices de la práctica. Al igual que en el paso anterior, se forzó un error para validar que el sistema de integración continua detectaba fallos en los tests antes de restablecer el código a su estado funcional.

2.2. Implementación de QA

Para la ejecución del análisis de calidad (QA), se optó por una configuración basada en un Self-hosted Runner:

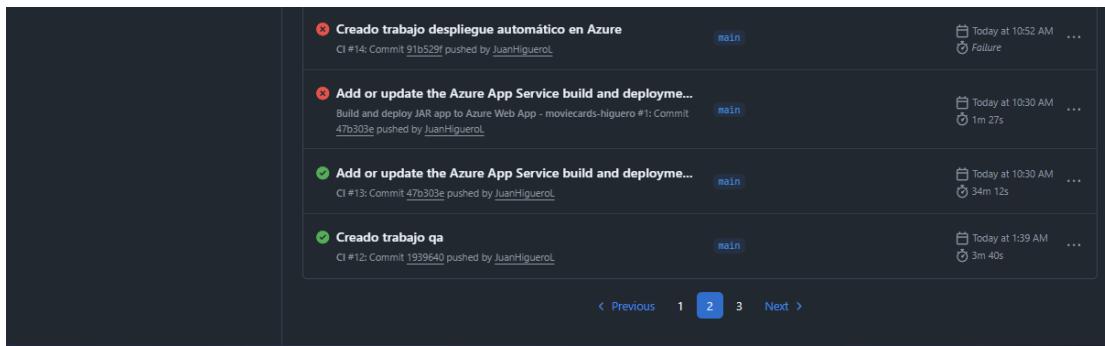
- **Configuración del Runner:** Se hizo uso de actions-runner para crear una instancia propia denominada "mi-runner", la cual se ejecutó y gestionó desde PowerShell.
- **Integración de SonarQube:** Se modificó el main.yaml integrando los parámetros de acceso y la contraseña configurada en SonarQube.
- **Ajuste de Umbrales:** Durante las pruebas de calidad, se aplicó inicialmente una restricción máxima de 5 errores críticos. Dado que el proyecto no superaba este filtro en su estado inicial, se aumentó el límite a 8 errores para permitir la continuidad del flujo y el avance hacia las siguientes fases.

En las siguientes imágenes se muestran los procesos mencionados en Github.



The screenshot shows the GitHub Actions interface for a repository. On the left, there's a sidebar with 'Actions' selected, showing 'All workflows' and a 'New workflow' button. Below that is a 'CI' section with 'Management' and several collapsed sections: 'Caches', 'Attestations', 'Runners', 'Usage metrics', and 'Performance metrics'. The main area is titled 'All workflows' and shows '60 workflow runs'. A search bar at the top right says 'Filter workflow runs'. The table lists the runs, each with a status icon (green checkmark for success, red X for failure), the name of the workflow, the commit hash, the branch (all are 'main'), the run time (e.g., 'Today at 1:36 AM'), and a duration (e.g., '3m 45s'). Some rows have three dots at the end. At the bottom, there are navigation buttons for 'Previous', page numbers (1, 2, 3, Next), and a total count of '245'.

Workflow Run	Status	Commit	Branch	Time	Duration
Creado trabajo qa	Success	CI #11: Commit 02eb3bb pushed by JuanHiguerol	main	Today at 1:36 AM	3m 45s
Creado trabajo qa	Success	CI #10: Commit c5dabe8 pushed by JuanHiguerol	main	Today at 1:23 AM	25s
Creado trabajo qa	Success	CI #9: Commit ca88bd8 pushed by JuanHiguerol	main	Today at 1:17 AM	4m 29s
Pruebas sin fallos	Success	CI #8: Commit c44d552 pushed by JuanHiguerol	main	Today at 12:58 AM	3m 2s
Fallo forzado en prueba unitaria	Failure	CI #7: Commit cbf6656 pushed by JuanHiguerol	main	Today at 12:54 AM	3m 4s
Trabajo test creado	Success	CI #6: Commit d384d7 pushed by JuanHiguerol	main	Today at 12:51 AM	3m 3s
Error de compilación corregido	Success	CI #5: Commit e82db51 pushed by JuanHiguerol	main	Feb 21, 8:09 PM GMT+1	40s
Error de compilación forzado	Failure	CI #4: Commit 4f0e36a pushed by JuanHiguerol	main	Feb 21, 8:08 PM GMT+1	30s
Error de compilación forzado	Failure	CI #3: Commit 5b06f55 pushed by JuanHiguerol	main	Feb 21, 8:04 PM GMT+1	47s
Creado main.yaml	Success	CI #1: Commit e4b73da pushed by JuanHiguerol	main	Feb 21, 7:53 PM GMT+1	24s



2.3. Despliegue en Microsoft Azure

Una vez validada la estructura del pipeline, se procedió a la puesta en marcha de la aplicación en la nube. Debido a limitaciones técnicas y errores de disponibilidad en otras regiones, se optó por crear la Web App en Azure bajo la región Italy North.

Para integrar el despliegue en el flujo de trabajo, se siguieron los pasos de configuración para conectar Azure con GitHub. Este proceso generó automáticamente un archivo YAML de despliegue específico de Azure; de este archivo se extrajo el secreto de publicación, que fue configurado en los secretos del repositorio de GitHub. Una vez transferida esta configuración al main.yaml principal para unificar el flujo, el archivo generado por Azure fue eliminado para mantener la limpieza del repositorio.

Tras realizar el commit y el push de estos cambios, la aplicación ya puede visualizarse en Azure.

The screenshot shows the GitHub Actions CI details for a specific job (#17). The job is titled "Creado trabajo despliegue automático en Azure".

Summary: Triggered via push 6 minutes ago by JuanHigueroL pushed to c424831 main. Status: Success. Total duration: 3m 49s. Artifacts: 1.

Workflow: main.yaml (on: push)

```

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

Artifacts: Produced during runtime.

Name	Size	Digest
moviecards-java	38.4 MB	sha256:755ad8715cd2d2f216...

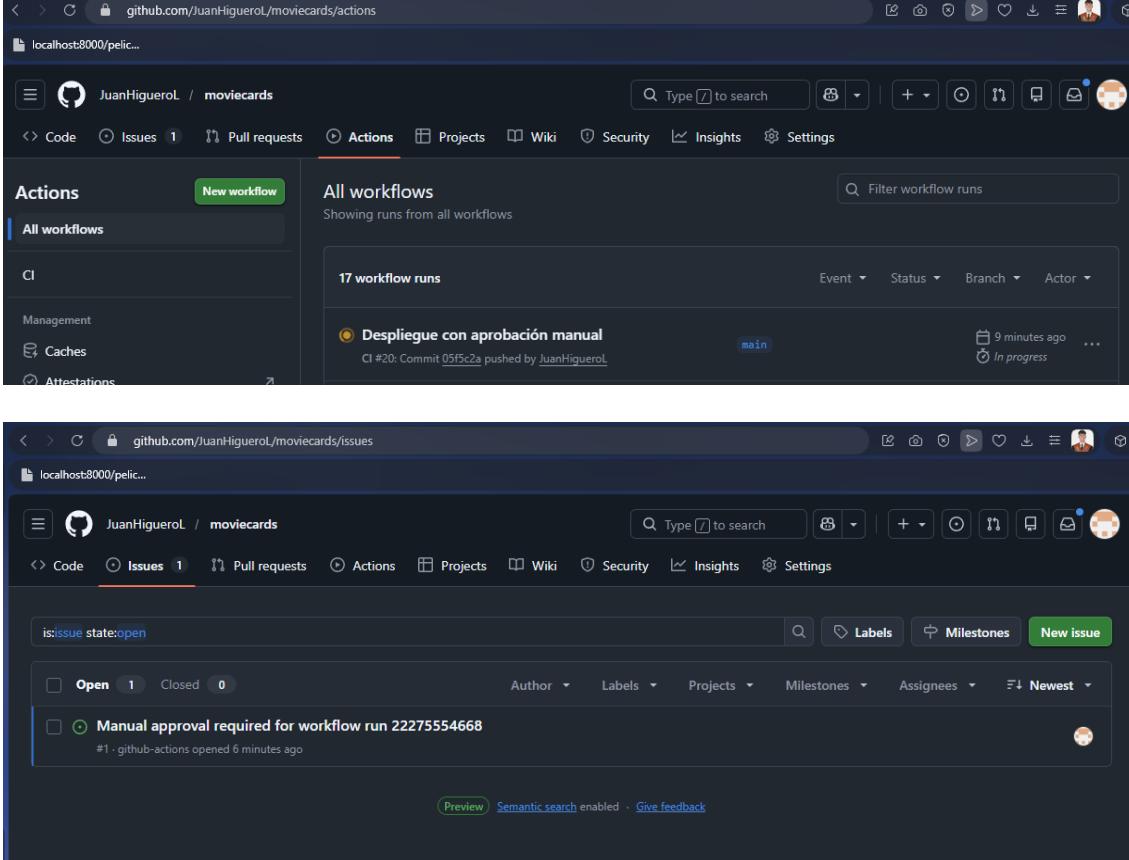


The screenshot shows a web interface for movie management. At the top, there's a header with the title "Gestión de películas". Below it, there are four main functional areas arranged in a grid:

- Inscripción Actor en Película:** A form with a single input field labeled "Inscripción Actor en Película".
- Listado actores:** A section with a button "Listado de actores".
- Nuevo Actor:** A section with a button "Crear nuevo actor".
- Listado películas:** A section with a button "Listado de películas".
- Nueva película:** A section with a button "Crear nueva película".

2.4. Aprobación Manual

Siguiendo las directrices de seguridad de la práctica, se implementó un paso de Aprobación Manual antes del despliegue definitivo en producción. Este mecanismo asegura que un responsable deba validar el estado del proyecto antes de que los cambios se apliquen en el entorno real. En el flujo se observa cómo el workflow entra en estado de espera hasta que el usuario autorizado introduce un "yes".



The screenshots illustrate the implementation of manual approval for a GitHub Actions workflow.

GitHub Actions Page:

- The URL is github.com/JuanHigueroL/moviecards/actions.
- The sidebar shows the "Actions" tab is selected.
- The main area displays "All workflows" and "17 workflow runs".
- A specific workflow run is highlighted: "Despliegue con aprobación manual" (Event: CI #20: Commit 95f5c2a pushed by JuanHigueroL). It shows the status as "In progress" and was run 9 minutes ago.

GitHub Issues Page:

- The URL is github.com/JuanHigueroL/moviecards/issues.
- The sidebar shows the "Issues" tab is selected.
- The search bar contains the query "is:issue state:open".
- The results show one open issue: "#1 · Manual approval required for workflow run 22275554668" (Author: JuanHigueroL, Created 6 minutes ago).
- The issue details page shows the message: "Manual approval required for workflow run 22275554668" and "#1 · github-actions opened 6 minutes ago".

github.com/juanHigueroL/moviecards/issues/1

localhost8000/pelic...

Manual approval required for workflow run 22275554668 #1

[Edit](#) [New issue](#) [...](#)

[Open](#)

github-actions bot opened 8 minutes ago [...](#)

Note
Workflow is pending manual review.
URL: <https://github.com/JuanHigueroL/moviecards/actions/runs/22275554668>

Important
Required approvers:
• [@JuanHigueroL](#)

Tip
Respond "approved", "approve", "lgtm", "yes" to continue workflow or "denied", "deny", "no" to cancel.

[Create sub-issue](#) [...](#)

github-actions assigned [JuanHigueroL](#) 8 minutes ago

Add a comment

Write Preview [H](#) [B](#) [I](#) [E](#) [<>](#) [Ø](#) [≡](#) [≡≡](#) [%](#) [@](#) [🔗](#) [☒](#) [↶](#)

yes

github.com/juanHigueroL/moviecards/actions/runs/22275554668

localhost8000/pelic...

JuanHigueroL / moviecards

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

Type to search

[Re-run all jobs](#) [...](#)

Despliegue con aprobación manual #20

Summary

All jobs [...](#)

build **test** **qa** **deploy**

Triggered via push 13 minutes ago **Status Success** Total duration **13m 16s** Artifacts **1**

main.yaml
on: push



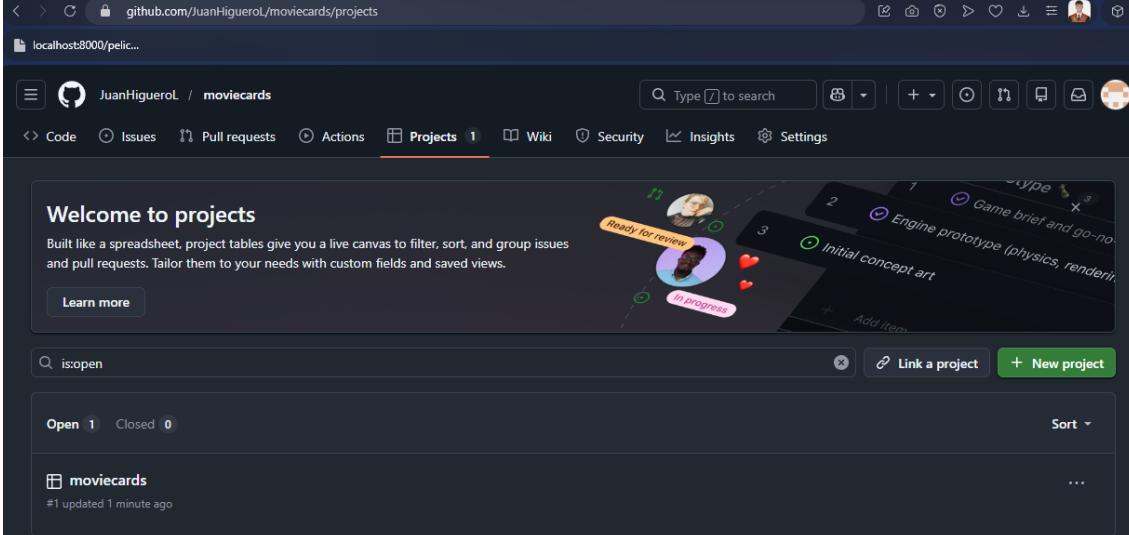
Artifacts
Produced during runtime

Name	Size	Digest
moviecards-java	38.4 MB	sha256:c1a4202f074ffde4f...

2.5. Planificación Ágil con GitHub Projects

Para la gestión de tareas se hace uso de la metodología ágil con las herramientas nativas de GitHub:

- **Configuración del Proyecto:** Se creó un proyecto denominado "moviecards" para centralizar la gestión de tareas.
- **Milestones y User Stories:** La planificación se organizó mediante Milestones (Sprints), dentro de los cuales se definieron las Issues que representan las historias de usuario.
- **Tablero Kanban:** Se configuró un tablero visual para monitorizar el flujo de trabajo, permitiendo mover las tareas entre los estados Todo, In Progress y Done.



Welcome to projects

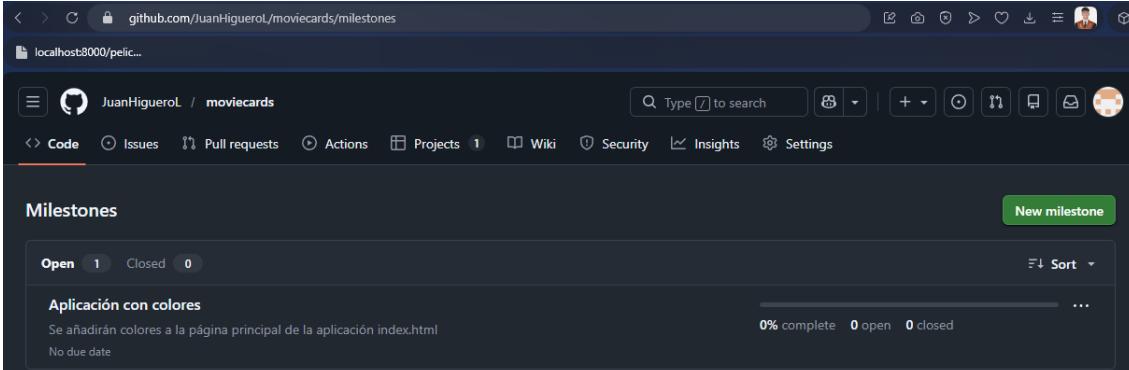
Built like a spreadsheet, project tables give you a live canvas to filter, sort, and group issues and pull requests. Tailor them to your needs with custom fields and saved views.

Learn more

is:open

Open 1 Closed 0 Sort ▾

moviecards #1 updated 1 minute ago



Milestones

New milestone

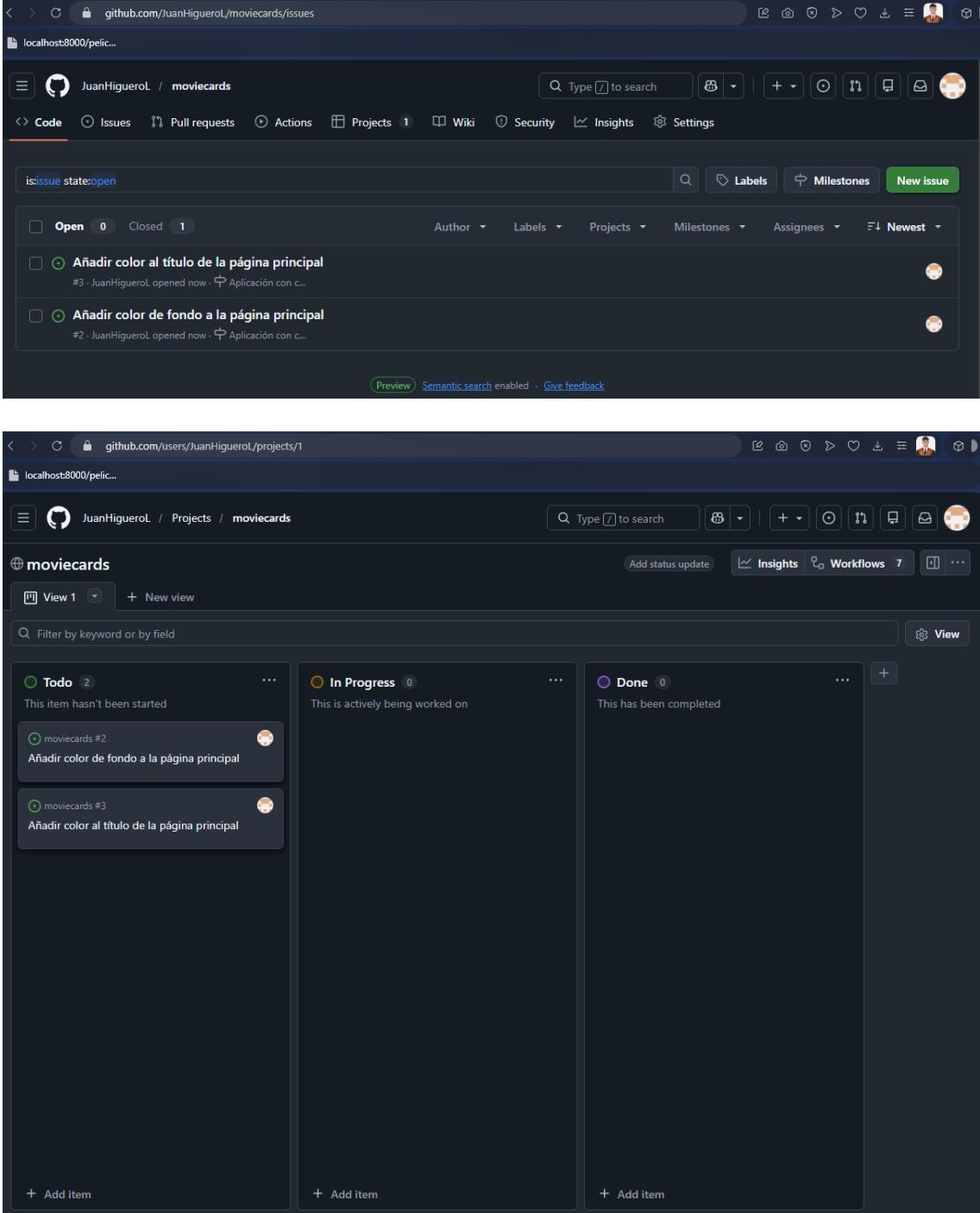
Aplicación con colores

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

No due date

0% complete 0 open 0 closed

Sort ▾



The image shows two screenshots of a GitHub repository named 'moviecards'.

Screenshot 1: GitHub Issues

- The URL is github.com/JuanHigueroL/moviecards/issues.
- The search bar contains the query: `is:issue state:open`.
- There are two open issues listed:
 - #3: Añadir color al título de la página principal (Author: JuanHigueroL, opened now)
 - #2: Añadir color de fondo a la página principal (Author: JuanHigueroL, opened now)
- Filters at the top include: Open (0), Closed (1), Author, Labels, Projects, Milestones, Assignees, and Newest.

Screenshot 2: GitHub Projects

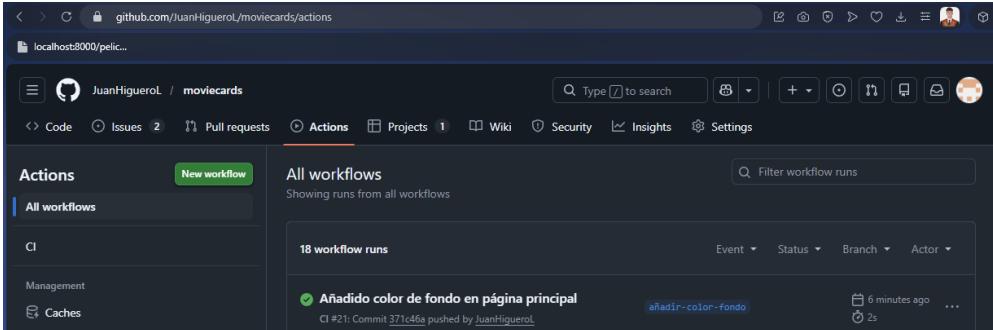
- The URL is github.com/users/juanhigueroL/projects/1.
- The board is titled 'moviecards'.
- It has three columns: Todo, In Progress, and Done.
- Todo:** Contains two items:
 - moviecards #2: Añadir color de fondo a la página principal
 - moviecards #3: Añadir color al título de la página principal
- In Progress:** Contains one item: moviecards #2 (Añadir color de fondo a la página principal).
- Done:** Contains one item: moviecards #3 (Añadir color al título de la página principal).
- Each column has a '+ Add item' button at the bottom.

2.6. Gestión de Ramas y Resolución de Conflictos

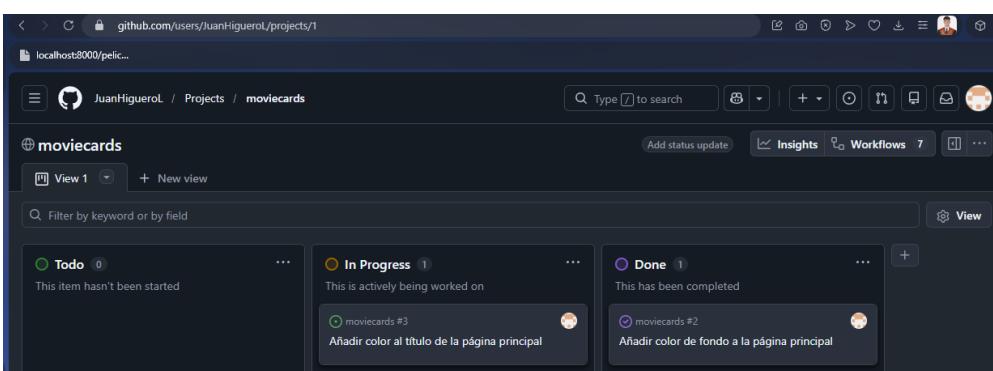
Se trabajó con un modelo de ramas para implementar cambios estéticos en la interfaz:

- Desarrollo en Ramas:** Se crearon dos ramas específicas, una para modificar el color del fondo y otra para el color del título.
- Validación y Merge:** Tras realizar los commits y verificar que las pruebas automatizadas en el pipeline no reportaban errores, se procedió a mover las tareas en el tablero Kanban y realizar el merge hacia la rama `main`.

3. Gestión de Conflictos: Al trabajar en la segunda rama, surgieron errores de compatibilidad debido a que ambas ramas modificaban las mismas líneas de código. Se procedió a la resolución manual de los conflictos.

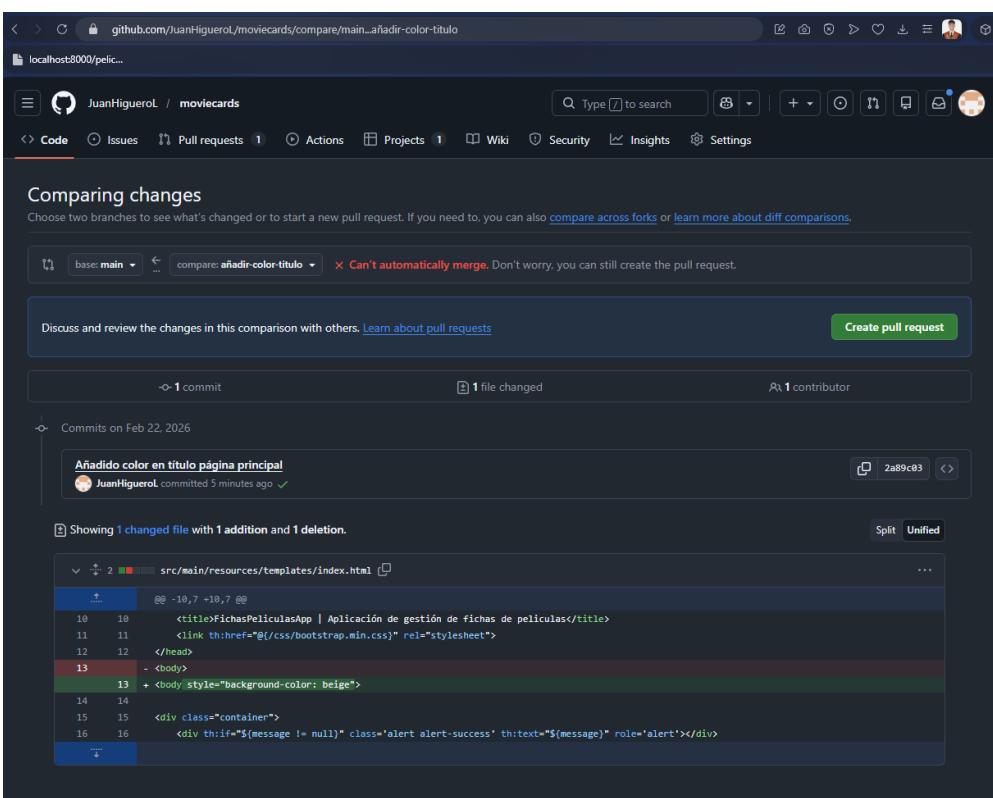


The screenshot shows the GitHub Actions interface for the 'moviecards' repository. On the left, there's a sidebar with 'Actions' selected, showing 'All workflows'. The main area displays 'All workflows' with a search bar and a filter for 'Workflow runs'. A single workflow run is listed under '18 workflow runs': 'Añadido color de fondo en página principal' (CI #21). It includes details like the commit hash (371c46a), author (JuanHigueroL), event (añadir-color-fondo), status (6 minutes ago), and duration (2s).

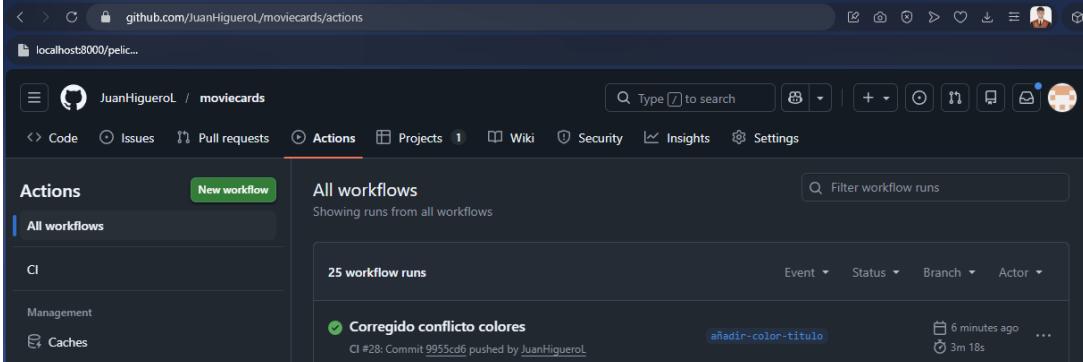


The screenshot shows the GitHub Projects interface for the 'moviecards' repository. It features a board with three columns: 'Todo' (0 items), 'In Progress' (1 item: 'Añadir color al título de la página principal'), and 'Done' (1 item: 'Añadir color de fondo a la página principal'). Each card has a small profile picture and a timestamp.

Aviso de conflicto de fusión.

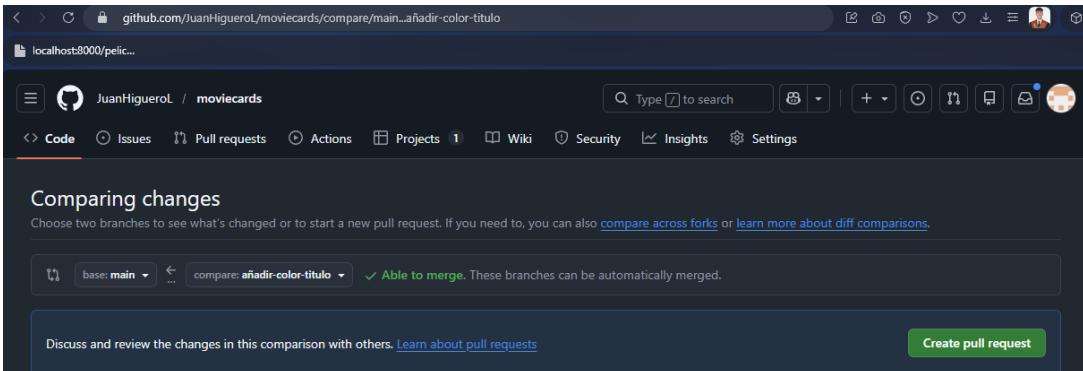


The screenshot shows the GitHub 'Comparing changes' page for the 'moviecards' repository, comparing 'main' and 'añadir-color-título'. It indicates that the merge failed ('Can't automatically merge'). The comparison summary shows 1 commit, 1 file changed, and 1 contributor. The commit details show a merge from 'main' to 'añadir-color-título' at 2a89c83. The diff view shows a conflict in the file 'src/main/resources/templates/index.html' at line 13, where the code was modified from '' to '+ <body style="background-color: beige">'.

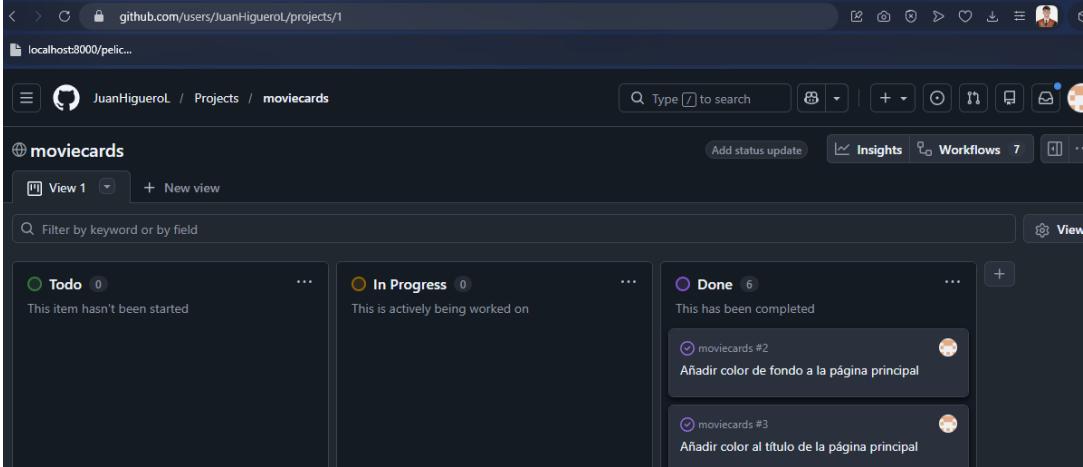


The screenshot shows the GitHub Actions interface for the repository 'JuanHiguerol/moviecards'. The 'Actions' tab is selected, displaying 'All workflows'. A single workflow run is listed under '25 workflow runs', titled 'Corregido conflicto colores'. The run was triggered by a commit (CI #28) pushed by JuanHiguerol. It was completed 6 minutes ago and took 3m 18s. A green checkmark icon indicates success.

Tras resolver el conflicto de fusión, el merge con la rama main se completa correctamente.

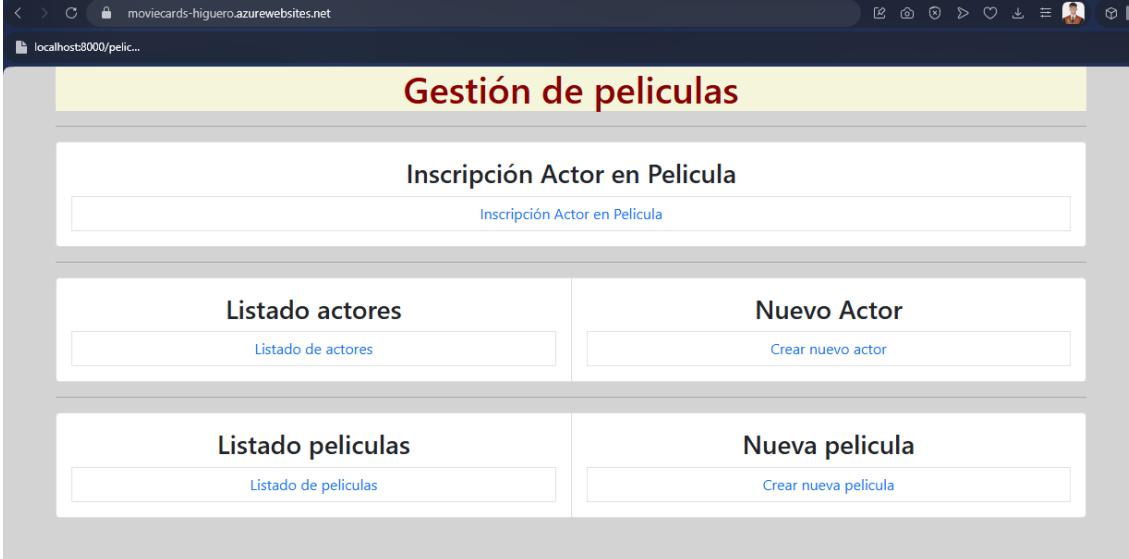


The screenshot shows the GitHub 'Compare changes' interface. It displays a comparison between the 'main' branch and the 'añadir-color-título' branch. A green checkmark indicates that the branches are 'Able to merge'. The interface includes a 'Create pull request' button and a note encouraging users to discuss changes with others.



The screenshot shows the GitHub Projects interface for the 'moviecards' project. It features three columns: 'Todo' (0 items), 'In Progress' (0 items), and 'Done' (6 items). The 'Done' column contains two completed tasks: 'moviecards #2: Añadir color de fondo a la página principal' and 'moviecards #3: Añadir color al título de la página principal'. The 'View' button at the top right allows switching between different project views.

Una vez unido todo en la rama main se pueden visualizar los cambios finales en la página web de Azure.



The screenshot shows a web browser window with the URL `localhost:8000/pelic...`. The page title is **Gestión de películas**. The main content area is titled **Inscripción Actor en Película**, which contains a single input field labeled **Inscripción Actor en Película**.

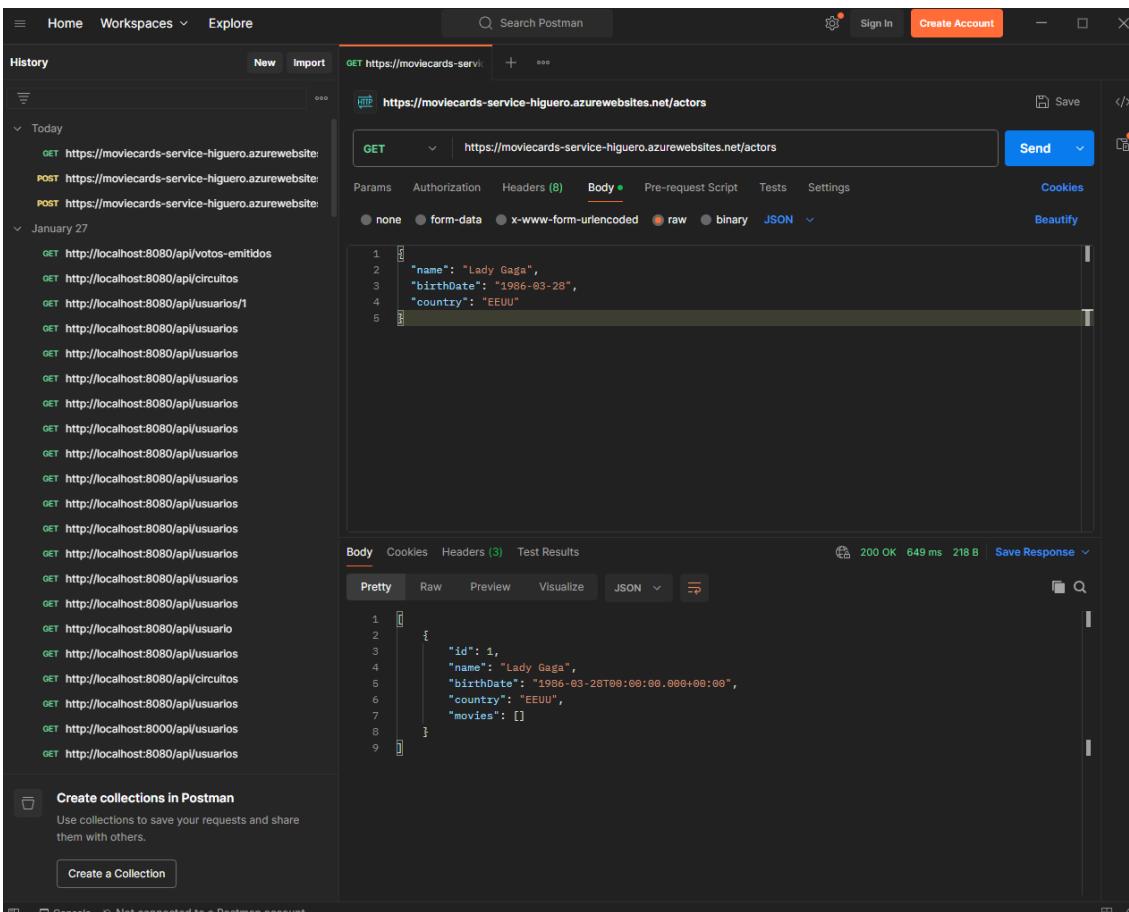
The page is divided into four main sections:

- Listado actores**: Contains a button labeled **Listado de actores**.
- Nuevo Actor**: Contains a button labeled **Crear nuevo actor**.
- Listado películas**: Contains a button labeled **Listado de películas**.
- Nueva película**: Contains a button labeled **Crear nueva película**.

3. Creación de moviecards-service

Siguiendo los requerimientos del proyecto, se ha procedido a la creación de un nuevo repositorio independiente denominado moviecards-service. Este repositorio contiene la lógica de negocio y el acceso a datos, siendo una API REST para la aplicación web principal.

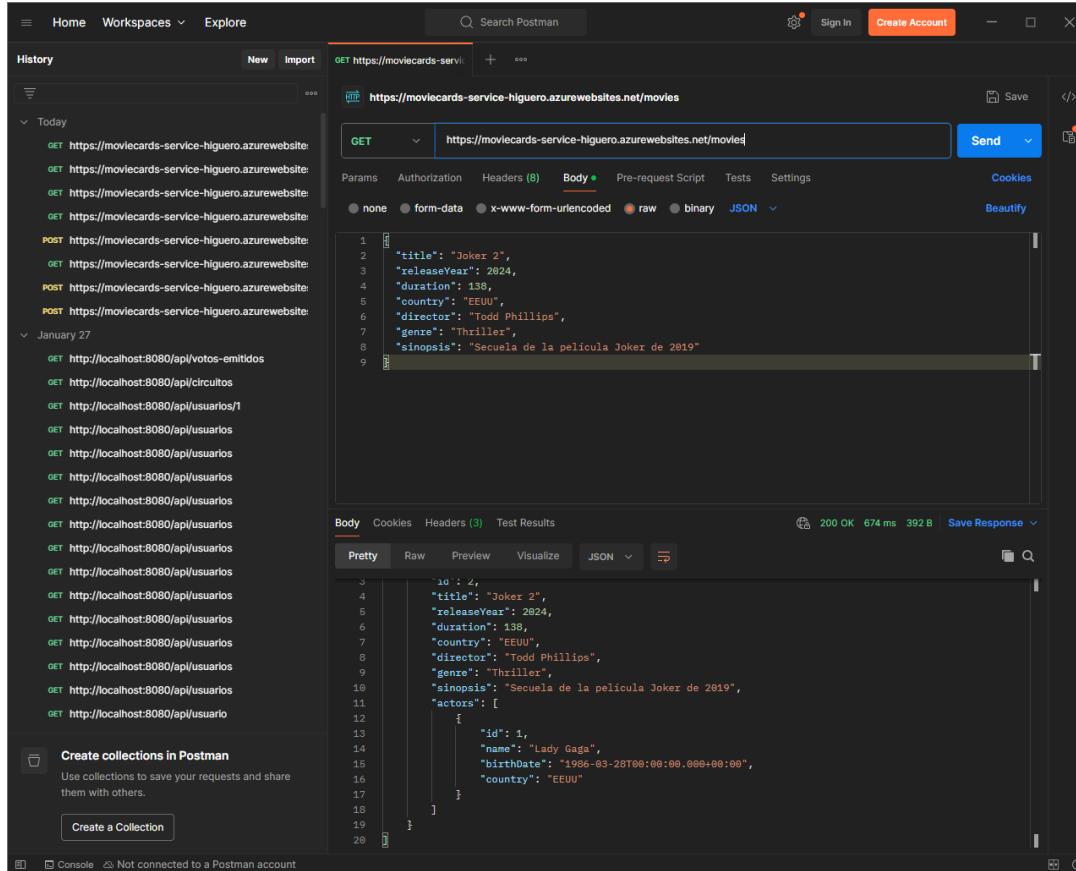
Para su puesta en marcha, se creó una nueva Web App en Azure específica para el servicio. El proceso de despliegue automatizado se replicó siguiendo la metodología establecida en la práctica inicial, asegurando una entrega continua en la URL de producción del servicio. La validación del sistema se realizó mediante Postman, confirmando el correcto funcionamiento de los métodos GET y POST tanto para actores como para películas, así como la funcionalidad de asociación entre ambos.



The screenshot shows the Postman application interface. On the left, there's a sidebar with a history of requests. In the main area, a request is being viewed for the URL `https://moviecards-service-higuero.azurewebsites.net/actors`. The method is set to `GET`. The `Body` tab is selected, showing a JSON response for an actor named "Lady Gaga". The response body is:

```
1
2   {
3     "id": 1,
4     "name": "Lady Gaga",
5     "birthdate": "1986-03-28T00:00:00.000+00:00",
6     "country": "EEUU",
7     "movies": []
8   }
```

Below the Body tab, the status bar indicates a `200 OK` response with `649 ms` and `218 B` size.



4. Refactorización de la Aplicación Principal

Con el servicio externo operativo, se inició la versión 2.0 de la aplicación moviecards. Para gestionar este cambio, se creó un nuevo Sprint en GitHub, dividiendo la tarea en dos historias de usuario principales: la modificación del código fuente en src/main para sustituir el acceso a base de datos por llamadas mediante RestTemplate, y la actualización de los tests en src/test para reflejar esta nueva arquitectura. A continuación, se muestra el historial de commits que demuestran su realización.

Commit	Author	Branch	Time Ago
CI #40: Commit 41f2259	JuanHigueroL	main	14 minutes ago
CI #39: Commit b324111	JuanHigueroL	main	20 minutes ago
CI #38: Commit b9a38b7	JuanHigueroL	main	30 minutes ago
CI #37: Commit 4e83f01	JuanHigueroL	main	39 minutes ago
CI #36: Commit e88d037	JuanHigueroL	main	45 minutes ago
CI #35: Commit 1167129	JuanHigueroL	main	51 minutes ago

5. Implementación del Atributo `deadDate`

Para dar soporte a la fecha de fallecimiento, se modificó el código del repositorio moviecards-service. Se integró el atributo `deadDate` en la clase Actor, replicando la lógica existente para `birthDate` tanto en la declaración de variables como en sus métodos de acceso (getters y setters).

6. Entorno de Pre-producción y Ciclo de Calidad

La fase final del proyecto se centró en la integración visual de la nueva funcionalidad y la robustez del pipeline:

- **Entorno de Stage:** Se ha escalado la infraestructura en Azure creando una nueva aplicación denominada moviecards-pre-higuero. En el archivo `main.yaml`, se ha insertado un nuevo trabajo llamado `stage` entre las fases de "qa" y "deploy". Este trabajo utiliza un secreto específico de la nueva instancia de Azure para permitir la validación en un entorno de pre-producción antes de la salida a producción definitiva.
- **Integración de `deadDate`:** Se ha actualizado la interfaz de usuario para solicitar la fecha de fallecimiento en el formulario de creación y mostrarla en el listado general de actores. Este desarrollo se realizó en una rama independiente antes de su integración final en la rama `main`.
- **Batería de Pruebas:** Se han modificado los tests unitarios, de integración y funcionales (End-to-End) para validar que el sistema maneja correctamente el atributo `deadDate` en todas sus capas.
- **Garantía de Calidad y SonarQube:** Inicialmente, el proyecto presentaba 8 errores críticos; tras una refactorización de los controladores `ActorController` y `MovieController` se ha logrado reducir la cifra a solo 2 problemas críticos, superando con éxito el umbral de calidad exigido.

localhost:9000/project/issues

localhost8000/pelic...

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

moviecards main

Last analysis of this Branch had 2 warnings February 22, 2026 at 8:35 PM Version 0.0.1-SNAPSHOT

Overview Issues Security Hotspots Measures Code Activity Project Settings Project Information 1 / 54 issues 2h 11min effort

Filters

Issues in new code

Type

- Bug 0
- Vulnerability 2
- Code Smell 52

Severity

- Blocker 2
- Critical 2
- Major 1
- Minor 7
- Info 42

Scope

Resolution

Status

Security Category

Creation Date

Language

Rule

Tag

Directory

File

Assignee

Author

Bulk Change

src.../com/lauracercas/moviecards/MovieCardsApplication.java

Immediately return this expression instead of assigning it to the temporary variable "template". ✖ 4 hours ago L22 % clumsy

Code Smell Minor Open Not assigned 2min effort Comment

src.../moviecards/controller/ActorController.java

Refactor this method to not always return the same value. 39 seconds ago L49 % No tags

Code Smell Blocker Open Not assigned 4min effort Comment

Replace this persistent entity with a simple POJO or DTO object. 19 hours ago L49 % clumsy

Vulnerability Critical Open Not assigned 10min effort Comment cwe, owasp-a5, spring

src.../moviecards/controller/MovieController.java

Refactor this method to not always return the same value. 39 seconds ago L49 % No tags

Code Smell Blocker Open Not assigned 4min effort Comment

Replace this persistent entity with a simple POJO or DTO object. 19 hours ago L49 % clumsy

Vulnerability Critical Open Not assigned 10min effort Comment cwe, owasp-a5, spring

src.../moviecards/service/actor/ActorServiceImpl.java

Remove this unused import 'com.lauracercas.moviecards.repositories.ActorJPA'. 4 hours ago L5 % unused

Code Smell Minor Open Not assigned 2min effort Comment clumsy

Immediately return this expression instead of assigning it to the temporary variable "actoresList". 4 hours ago L30 % clumsy

Code Smell Minor Open Not assigned 2min effort Comment clumsy

Immediately return this expression instead of assigning it to the temporary variable "actor". 10 minutes ago L47 % clumsy

Code Smell Minor Open Not assigned 2min effort Comment clumsy

src.../moviecards/service/movie/MovieServiceImpl.java

Remove this unused import 'com.lauracercas.moviecards.repositories.MovieJPA'. 4 hours ago L4 % clumsy