



---

## TRABAJO FINAL. INTEGRACIÓN CONTINUA

---



20 DE FEBRERO DE 2026  
Eva Huertas Morera

## **Contenido**

1. Introducción .....	2
1.1. Objetivo del trabajo .....	2
1.2. Alcance del proyecto .....	2
2. Realizar la práctica del tema 5 .....	3
3. Configuración de moviecards-service .....	6
3.1. Creación del repositorio .....	6
3.1. Despliegue en Azure App Service .....	7
3.3. Verificación de funcionamiento (Postman) .....	8
4. Modificaciones en la aplicación principal (moviecards) .....	9
4.1. Reconfiguración de la arquitectura.....	9
4.2. Implementación de la comunicación REST.....	9
4.3. Gestión del proyecto y Metodología Ágil .....	9
5. Evolución del Servicio: Gestión de la Fecha de Fallecimiento.....	10
5.1. Modificación del Modelo de Datos .....	10
5.2. Despliegue y persistencia .....	11
5.3. Validación de la API con Postman .....	11
6. Implementación de funcionalidades en moviecards fecha de fallecimiento .....	12
6.1. Configuración del entorno de Pre-producción (Stage) .....	12
6.2. Actualización de la interfaz y lógica de negocio.....	13
6.3. Pruebas y calidad de código.....	14

## 1. Introducción

En el siguiente documento se detalla el proceso de integración continua de la aplicación web moviecards. El proyecto se centra en la transición de una arquitectura con un frontend situado en el proyecto moviecard hacia una arquitectura de servicios desacoplados, aplicando flujos de trabajo de integración y despliegue continuo (CI/CD) en Github.

### 1.1. Objetivo del trabajo

El objetivo principal es separar la lógica de negocio en un servicio independiente denominado moviecards-service. Esta nueva iteración busca ofrecer una API autónoma que pueda ser consumida por diversos clientes.

### 1.2. Alcance del proyecto

La evolución del sistema contempla los siguientes hitos clave:

- **Desacoplamiento:** Instalación del servicio en una URL independiente (<https://moviecards-service-huertas.azurewebsites.net>) y la aplicación cliente en la URL original.
- **Nuevas Funcionalidades:** Incorporación del atributo deadDate (fecha de fallecimiento) en la entidad de actores, afectando tanto al servicio como a la interfaz de usuario de la aplicación web.
- **Calidad de Software:** Implementación de un entorno de pre-producción (*staging*) y la integración de umbrales de calidad en SonarQube para restringir el despliegue si se detectan 5 o más problemas críticos.
- **Automatización:** Actualización del flujo de trabajo en GitHub Actions para evaluar las pruebas unitarias, de integración y funcionales (E2E) adaptadas a la nueva arquitectura.

**Para ello se deberá seguir los apartados mencionados en el enunciado.**

1) Realizar la práctica del tema 5 de la asignatura para la creación de la aplicación **moviecards**. La aplicación debe quedar funcionando en <https://moviecards-apellido.azurewebsites.net>, donde “apellido” es el apellido del alumno.

2) Crear un nuevo repositorio llamado **moviecards-service** con el código del servicio y desplegarlo en Azure. Debe quedar funcionando en <https://moviecards-service-apellido.azurewebsites.net>, donde “apellido” es el apellido del alumno. Para este repositorio no es necesario crear ningún proyecto asociado al repositorio, ni tampoco milestones ni issues. NOTA: El profesor probará el funcionamiento del servicio usando Postman.

3) 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 <https://moviecards-apellido.azurewebsites.net>. Para ello, hay que crear en el proyecto un nuevo sprint (milestone) con los siguientes issues:

3.1) Modificar el código de la aplicación en src/main.

3.2) Modificar el código de las pruebas en src/test.

**Apartados para realizar una nueva versión de la aplicación para manejar la fecha de fallecimiento de los actores:**

4) Modificar el código del repositorio **moviecards-service** para añadir un nuevo atributo en la clase **Actor**, llamado **deadDate**, que contenga la fecha de fallecimiento del actor, y desplegar en Azure, en la misma URL del servicio: <https://moviecards-service-apellido.azurewebsites.net>. Para este repositorio no es necesario crear ningún proyecto asociado al repositorio, ni tampoco milestones, issues, ramas ni nuevas pruebas. NOTA: El profesor probará el funcionamiento del servicio usando Postman para crear actores con fecha de fallecimiento.

5) Realizar una nueva versión de la aplicación **moviecards**, para que en la página web, 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. Debe quedar funcionando en la misma URL de la versión anterior <https://moviecards-apellido.azurewebsites.net>. Para ello, hay que crear en el proyecto un nuevo sprint (milestone) con los siguientes issues: 5.1) Añadir al workflow del proyecto un nuevo trabajo, entre los trabajos “qa” y “deploy”, llamado “stage” para desplegar la aplicación en un entorno de pre-producción, también en Azure, pero usando otro nombre para la aplicación, por ejemplo, moviecards-pre.

5.2) Modificar el código de la aplicación para manejar la fecha de fallecimiento de los actores. NOTA: Debe crearse una nueva rama en el repositorio para los cambios y después integrarlos en la rama máster.

5.3) Modificar las pruebas unitarias para tener en cuenta la fecha de fallecimiento de los actores.

5.4) Modificar las pruebas de integración para tener en cuenta la fecha de fallecimiento de los actores.

5.5) Modificar las pruebas funcionales (end to end) para tener en cuenta la fecha de fallecimiento de los actores.

5.6) Como garantía de calidad, hay que modificar el código fuente de la aplicación hasta conseguir tener menos de 5 problemas críticos, de tal forma que, si SonarQube detecta que tiene 5 o más, no debe poderse desplegar la aplicación a producción.

## **2. Realizar la práctica del tema 5**

Se realizó la practica 5 guiada, que se empezó en clase, en la que se realizaron todos los pasos para la configuración del main.yaml, que cubrirán las tareas de los actions y por otra parte la ejecución tanto del Docker como del sonar, con la imagen de josehilera.

The screenshot shows the SonarQube interface at [localhost:9000/projects](http://localhost:9000/projects). A message at the top indicates that the version is no longer active. The main area displays two projects: 'moviecards' (Passed) and 'MovieCards'. The 'moviecards' project has 0 bugs, 0 vulnerabilities, 0 hotspots reviewed, and 45 code smells. A note says 'Project's Main Branch is not analyzed yet.' and a 'Configure analysis' button is available.

The screenshot shows the Docker Container interface with a search bar and a filter for running containers. It lists several containers:

Name	Container ID	Image	Port	Actions
wonderful_haibt	3f972b769392	docker/des	-	<span>⋮</span> <span>▶</span> <span>⋮</span> <span>trash</span>
ubuntu-sonar	5abba05da199	josehilera/i_9000	-	<span>⋮</span> <span>▶</span> <span>⋮</span> <span>trash</span>
thirsty_maxwell	eaf67f146ddc	propamap-i_3000	-	<span>⋮</span> <span>▶</span> <span>⋮</span> <span>trash</span>
serene_ramanujan	f4b065d3a31b	docker/wel_8080	-	<span>⋮</span> <span>▶</span> <span>⋮</span> <span>trash</span>
part1	-	-	-	<span>⋮</span> <span>▶</span> <span>⋮</span> <span>trash</span>
getting-started-todo-app	-	-	-	<span>⋮</span> <span>▶</span> <span>⋮</span> <span>trash</span>

Una vez realizado todo eso se pudo visualizar la página del proyecto lanzado en la siguiente imagen. También se hicieron los cambios en cuanto al formato de la página por eso se ve de color gris y amarillo.

The screenshot shows the application interface at <https://moviecards-huertas.azurewebsites.net>. The main title is 'Gestión de películas'. Below it, there are four main sections: 'Inscripción Actor en Película', 'Listado actores', 'Nuevo Actor', 'Listado películas', and 'Nueva película'. Each section contains a link to its respective page.

Las restricciones que se pusieron a sonarqube inicialmente, fueron las que se ven en la imagen, con 5 errores críticos al principio y que mas tarde en otra parte de la práctica guiada se cambió a 8, para que no diesen error en los actions.

The screenshot shows the SonarQube interface for managing quality gates. A message at the top indicates that the active version is no longer supported and suggests upgrading. The navigation bar includes 'Projects', 'Issues', 'Rules', 'Quality Profiles', 'Quality Gates', and 'Administration'. The 'Quality Gates' tab is selected. A sub-menu for 'control calidad' is shown, with tabs for 'Sonar way' (selected) and 'BUILT-IN' (disabled). The 'DEFAULT' tab is also present. The main area displays a table of quality rules:

Metric	Operator	Value
Duplicated Lines (%)	is greater than	3.0%
Maintainability Rating	is worse than	A (Technical debt ratio is less than 5.0%)
Reliability Rating	is worse than	A (No bugs)
Security Hotspots Reviewed	is less than	100%
Security Rating	is worse than	A (No vulnerabilities)

Below the table, there's a section for 'Conditions on Overall Code' with a single row: 'Critical Issues is greater than 5'. A note below says: 'You may click unlock to edit this quality gate. Adding extra conditions to a compliant quality gate can result in drawbacks. Are you reconsidering? Clean as You Code? We strongly recommend this methodology to achieve a Clean Code status.' A 'Unlock editing' button is available. To the right, sections for 'Projects' and 'Permissions' are visible.

A continuación, se muestran todos los github actions de la práctica guiada.

The screenshot shows the GitHub Actions page for the repository 'moviecards'. The 'Actions' tab is selected, and the 'All workflows' tab is chosen. A green button for 'New workflow' is visible. The left sidebar lists CI, Management, Caches, Attestations, Runners, Usage metrics, and Performance metrics. The main area shows a table of workflow runs:

Event	Status	Branch	Actor
Feb 9, 12:16 PM GMT+1	Success	master	...
Feb 9, 12:11 PM GMT+1	Success	master	...

Below the table, there are buttons for 'Previous', 'Next', and a page number '2'.

This screenshot shows another view of the GitHub Actions page for the 'moviecards' repository, specifically the 'All workflows' section. The left sidebar is identical to the previous screenshot. The main area shows a table of workflow runs:

Event	Status	Branch	Actor
Feb 9, 1:50 PM GMT+1	Success	master	...
Feb 9, 1:33 PM GMT+1	Success	master	...
Feb 9, 1:19 PM GMT+1	Success	master	...
Feb 9, 1:19 PM GMT+1	Success	master	...
Feb 9, 1:03 PM GMT+1	Success	master	...
Feb 9, 12:36 PM GMT+1	Success	master	...
Feb 9, 12:32 PM GMT+1	Success	master	...
Feb 9, 12:27 PM GMT+1	Success	master	...
Feb 9, 12:19 PM GMT+1	Success	master	...

Below the table, there are buttons for 'Previous', 'Next', and a page number '2'.

Usage metrics	master	Feb 9, 3:39 PM GMT+1 4m 51s
Performance metrics	cambios-moviecards-service	Feb 9, 3:33 PM GMT+1 4m 37s
	master	Feb 9, 3:02 PM GMT+1 5m 22s
	ahadir-color-titulo	Feb 9, 2:57 PM GMT+1 4m 4s
	ahadir-color-titulo	Feb 9, 2:46 PM GMT+1 4m 4s
	master	Feb 9, 2:36 PM GMT+1 4m 45s
	ahadir-color-fondo	Feb 9, 2:28 PM GMT+1 4m 35s

Además de los actions se generaron los milestone y los issues pertinentes a esos cambios, que se muestran a continuación.

Milestones / Aplicación con colores

### Aplicación con colores

Closed No due date Closed 2 weeks ago 100% complete

Open 0 Closed 4

- Añadir color a la página principal #2 - by evalh02 was closed 2 weeks ago
- Añadir color al título de la página principal #3 - by evalh02 was closed 2 weeks ago
- Añadido color de fondo en página principal evalh02/movecards#4 - by evalh02 was closed 2 weeks ago ✓ 3/4
- Añadido color en título página principal evalh02/movecards#6 - by evalh02 was closed 2 weeks ago ✓ 3/4

## 3. Configuración de moviecards-service

### 3.1. Creación del repositorio

Siguiendo las recomendaciones, se ha procedido de la siguiente manera:

- Se ha utilizado el código base proporcionado por el profesor desde el repositorio oficial.

Platform Solutions Resources Open Source Enterprise Pricing

josehilera / moviecards-service Public

Code Issues Pull requests Actions Projects Security Insights

master 1 Branch 0 Tags

Go to file Code Clone

HTTPS GitHub CLI

https://github.com/josehilera/moviecards-service

Clone using the web URL.

Open with GitHub Desktop Download ZIP

About

No description, website, or topics provided.

Activity 0 stars 1 watching 4 forks Report repository

Releases

No releases published

Packages

No packages published

Languages

- Se creó un nuevo repositorio local mediante el comando git init y se vinculó a un nuevo repositorio remoto en GitHub denominado moviecards-service.

The screenshot shows the GitHub repository page for 'moviecards-service'. The repository was created by 'evalh02' and has 1 branch and 0 tags. It contains 5 commits from Eva Huertas, including changes to '.github/workflows' and 'mvnw' files. There is a 'README' section with a placeholder 'Add a README' button. The repository has 0 stars, 0 forks, and no releases or packages published. It also has 2 deployments to 'Production'.

- No se asociaron proyectos, milestones ni issues a este repositorio específico, ya que el enfoque principal fue su despliegue y disponibilidad funcional.

### 3.1. Despliegue en Azure App Service

El servicio se ha desplegado en la plataforma Azure, garantizando que sea accesible de forma independiente.

- **URL de acceso:** El servicio está operativo en <https://moviecards-service-huertado.azurewebsites.net> (sustituyendo "apellido" por el correspondiente).

The screenshot shows the Azure portal configuration for the 'moviecards-service-huertas' application. It includes details such as the resource group ('moviecards-service-huertas\_group'), location ('Poland Central'), and GitHub project link ('https://github.com/evalh02/moviecards-service'). The application is currently running ('En ejecución') and is associated with an 'Azure for Students' subscription.

- **Configuración de CI/CD:** Se configuró un flujo de trabajo mediante GitHub Actions (.github/workflows/main.yaml), actualizando el secreto AZUREAPPSERVICE\_PUBLISHPROFILE generado por Azure para permitir despliegues automáticos cada vez que se detectan cambios en el código.

### 3.3. Verificación de funcionamiento (Postman)

Para asegurar la integridad del servicio antes de conectarlo con la aplicación cliente, se realizaron pruebas manuales utilizando Postman. Se validaron los siguientes puntos finales (endpoints):

- Actores:** Pruebas de creación (POST /actors) y listado (GET /actors).
- Películas:** Verificación de creación (POST /movies) y consulta de detalles (GET /movies/idM).
- Relaciones:** Se comprobó la funcionalidad de inscribir actores en películas mediante el endpoint /movies/insc/idA/idM.

En la siguiente imagen se indica uno de ellos, en este caso el listado de los actores.

```

1  [
2   {
3     "id": 1,
4     "name": "Lady Gaga",
5     "birthDate": "1986-03-28T00:00:00.000+00:00",
6     "deadDate": "1986-03-04T00:00:00.000+00:00",
7     "country": "EEUU",
8     "movies": []
9   },
10  {
11    "id": 3,
12    "name": "Leonardo Dicaprio",
13    "birthDate": "1986-03-28T00:00:00.000+00:00",
14    "deadDate": "1986-03-04T00:00:00.000+00:00",
15    "country": "EEUU",
16    "movies": []
17  }
18 ]
  
```

#### **4. Modificaciones en la aplicación principal (moviecards)**

En esta fase se actualizó la aplicación original de la práctica del Tema 5 con lo especificado en el enunciado de la práctica.

##### **4.1. Reconfiguración de la arquitectura**

Se realizaron cambios estructurales en el código fuente (src/main) para integrar la comunicación entre servicios:

- **Inyección de dependencias:** Se añadió un Bean de RestTemplate en la clase principal MovieCardsApplication.java para habilitar las peticiones HTTP.
- **Controladores:** Se modificó CardController.java para utilizar @Autowired en la vinculación del servicio de actores.
- **Servicios de negocio:** Se actualizaron ActorServiceImpl.java y MovieServiceImpl.java para sustituir las interfaces JPA por llamadas a la API externa mediante RestTemplate.

##### **4.2. Implementación de la comunicación REST**

Para conectar con el servicio desplegado en Azure, se aplicaron las siguientes modificaciones técnicas:

- **Endpoints:** Se configuraron las URLs base apuntando a <https://moviecards-service-huertas.azurewebsites.net/> tanto para actores como para películas.
- **Operaciones CRUD:**
  - El método findAll() ahora utiliza template.getForObject para recuperar listas de objetos desde el servicio.
  - El método save() se reestructuró para diferenciar entre inserciones (POST) y actualizaciones (PUT) dependiendo de la existencia de un ID previo.
  - El método getById() consulta directamente al endpoint específico del recurso (ej. /actors/{id}).

##### **4.3. Gestión del proyecto y Metodología Ágil**

Siguiendo los principios de desarrollo ágil, estos cambios se organizaron de la siguiente manera:

- Se creó un nuevo Sprint (Milestone) específico en el proyecto de GitHub para agrupar las tareas de migración.
- Se vincularon Issues para el seguimiento de la modificación del código de la aplicación y la actualización de las pruebas.

The screenshot shows a GitHub milestones page for the repository 'evah02/moviecards'. A milestone titled 'Modificaciones para moviecards...' is displayed, which is currently closed. The status bar indicates '100% complete'. Below the title, there is a note: 'Modificaciones en los archivos de src/main y src/test, para la unión con moviecards-service y cambios recomendados de la práctica.' Under the note, there are three items listed: 'Modificar el código de la aplicación en src/main.', 'Modificar el código de las pruebas en src/test.', and 'Cambios src/main y src/test'. Each item has a status of 'Closed' and was completed 2 weeks ago.

## 5. Evolución del Servicio: Gestión de la Fecha de Fallecimiento

Este apartado describe la actualización del modelo de datos en el servicio para permitir el manejo de información más completa de los actores.

### 5.1. Modificación del Modelo de Datos

Se ha actualizado el código fuente del repositorio moviecards-service para integrar el nuevo requisito:

- Atributo deadDate:** Se ha añadido un nuevo atributo llamado deadDate en la clase Actor para almacenar la fecha de fallecimiento.

The screenshot shows the GitHub code editor for the file 'Actor.java' located at 'src/main/java/com/lauracercas/moviecards/model'. The code defines a class 'Actor' with several fields and annotations. The newly added field 'deadDate' is highlighted in blue. The code includes imports for 'com.fasterxml.jackson.annotation.JsonIgnoreProperties' and 'java.time.format.DateTimeFormat'. The 'deadDate' field is annotated with '@DateTimeFormat(pattern = "yyyy-MM-dd")'.

```

21 import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
22
23 @Entity
24 public class Actor {
25     @Id
26     @GeneratedValue(strategy = GenerationType.AUTO)
27     private Integer id;
28
29     private String name;
30
31     @DateTimeFormat(pattern = "yyyy-MM-dd")
32     private Date birthDate;
33
34     @DateTimeFormat(pattern = "yyyy-MM-dd")
35     private Date deadDate;
36
37     private String country;
38
39     @ManyToOne(mappedBy = "actors")
40     @JsonIgnoreProperties("actors") // Añadido
41     private List<Movie> movies;
42
43     public Actor() {
44     }
45
46     public Actor(Integer id, String name) {
47         this.id = id;
48         this.name = name;
49     }
50
51     public Integer getId() {
52         return id;
53     }
54 }

```

- Actualización del POJO:** Se han incluido los métodos *getter* y *setter* correspondientes para asegurar que el nuevo campo sea serializable y pueda ser procesado por la API REST.

```

25  public class Actor {
26      public String getName() {
27          return name;
28      }
29
30      public void setName(String name) {
31          this.name = name;
32      }
33
34      public Date getBirthDate() {
35          return birthDate;
36      }
37
38      public void setBirthDate(Date birthDate) {
39          this.birthDate = birthDate;
40      }
41
42      public Date getDeadDate() {
43          return deadDate;
44      }
45
46      public void setDeadDate(Date deadDate) {
47          this.deadDate = deadDate;
48      }
49
50      public String getCountry() {
51          return country;
52      }
53
54      public void setCountry(String country) {
55          this.country = country;
56      }
57
58  }

```

## 5.2. Despliegue y persistencia

- Actualización en Azure:** Los cambios se han desplegado en la URL existente del servicio: <https://moviecards-service-huertas.azurewebsites.net>.
- Integración:** No ha sido necesario crear nuevos proyectos, milestones o ramas específicas para este cambio en el repositorio del servicio, siguiendo las instrucciones de simplificación para este componente.

## 5.3. Validación de la API con Postman

Para garantizar que el servicio procesa correctamente el nuevo campo, se han realizado las siguientes pruebas:

- Creación de actores con fallecimiento:** Se ejecutaron peticiones POST enviando un cuerpo JSON que incluye el campo "deadDate": "YYYY-MM-DD".

We're updating our plans and pricing on March 1. See [our blog](#) for more details.

New Import

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

Body raw JSON

```

1 {
2     "id": "",
3     "name": "Leonardo Dicaprio",
4     "birthDate": "1986-03-28",
5     "deadDate": "1986-03-04",
6     "country": "EEUU",
7     "movies": []
8 }

```

- Consulta de datos:** Se verificó mediante peticiones GET que el campo deadDate se devuelve correctamente en la respuesta JSON junto con el nombre y la fecha de nacimiento.

```

1 [          ]
2 {           }
3   "id": 1,
4   "name": "Lady Gaga",
5   "birthDate": "1986-03-28T00:00:00.000+00:00",
6   "deadDate": "1986-03-04T00:00:00.000+00:00",
7   "country": "EEUU",
8   "movies": []
9 }
10 {
11   "id": 3,
12   "name": "Leonardo Dicaprio",
13   "birthDate": "1986-03-28T00:00:00.000+00:00",
14   "deadDate": "1986-03-04T00:00:00.000+00:00",
15   "country": "EEUU",
16   "movies": []
17 }
18 ]

```

## 6. Implementación de funcionalidades en moviecards fecha de fallecimiento

En esta fase final, se transformó la aplicación **moviecards** para consumir la nueva información del servicio y se blindó el proceso de despliegue mediante un pipeline de integración continua más robusto.

### 6.1. Configuración del entorno de Pre-producción (Stage)

Se ha modificado el flujo de trabajo de GitHub Actions para añadir una capa de seguridad antes del despliegue final:

- Nuevo Job "stage":** Se insertó un trabajo llamado stage en el workflow, ubicado estratégicamente entre las fases de "qa" (análisis de calidad) y "deploy" (producción).

```

  - name: Revisar la calidad con Sonarqube
    run: mvn sonar:sonar -Dsonar.host.url=http://localhost:9000 -Dsonar.qualitygate.wait=true -Dsonar.login=admin -Dsonar.password=admin2
  - stage: stage
    runs-on: ubuntu-latest
    needs: qa
  - steps:
    - name: Download artifact from build job
      uses: actions/download-artifact@v4
      with:
        name: moviecards-jar
    - name: Deploy to Azure Web App
      id: deploy-to-webapp
      uses: azure/webapps-deploy@v3
      with:
        app-name: 'moviecards-pre-huertas'
        slot-name: 'Production'
        package: '*.jar'
        publish-profile: ${{ secrets.AZUREAPPSERVICE_PUBLISHPROFILE_548C94F0C017442EF944348920EBC21 }}
```

- Entorno Aislado:** La aplicación se despliega automáticamente en un entorno de pre-producción en Azure (moviecards-pre-huertas.azurewebsites.net) para realizar validaciones finales en un entorno idéntico al real.

## 6.2. Actualización de la interfaz y lógica de negocio

Se adaptó la aplicación cliente para manejar el campo `deadDate`:

- Interfaz de usuario:** Se modificó el formulario de creación de actores para solicitar la fecha de fallecimiento y se añadió una nueva columna en la tabla de listado de actores para visualizar este dato.

```

48      name="name"
49      placeholder="Escriba el nombre del actor"
50      required="required"
51    />
52  </div>
53  <div class="mb-3">
54    <label for="birthDate" class="form-label">Fecha nacimiento</label>
55    <input
56      type="date"
57      class="form-control"
58      th:field="*{birthDate}"
59      id="birthDate"
60      name="birthDate"
61      placeholder="Escriba la fecha de nacimiento"
62      required="required"
63    />
64  </div>
65  <div class="mb-3">
66    <label for="deadDate" class="form-label">Fecha fallecimiento</label>
67    <input
68      type="date"
69      class="form-control"
70      th:field="*{deadDate}"
71      id="deadDate"
72      name="deadDate"
73      placeholder="Escriba la fecha de fallecimiento"
74      required="required"
75    />
76  </div>

```

The screenshot shows two GitHub repository pages side-by-side.

**Left Repository:** moviecards / master / src / main / resources / templates / actors / list.html

- File Structure:** .github/workflows, .mvn, src (main, resources, static, templates/actors/list.html), test/java/com/lauracercas/movieca..., .gitignore, mvnw.
- list.html Content:**

```

10     <title>FichasPeliculasApp | Aplicación de gestión de fichas de películas</title>
11     <link th:href="@{/css/bootstrap.min.css}" rel="stylesheet">
12   </head>
13   <body>
14     <div class="container mb-5">
15
16       <div class="card">
17         <h2 th:text="Listado Actores" class="card-header"></h2>
18         <div class="card-body">
19           <table class="table table-hover">
20             <thead class="thead-light">
21               <tr>
22                 <th scope="col">Identificador</th>
23                 <th scope="col">Nombre</th>
24                 <th scope="col">Fecha Nacimiento</th>
25                 <th scope="col">Fecha Fallecimiento</th>
26                 <th scope="col">País</th>
27                 <th scope="col">Editar</th>
28             </tr>
29           </thead>
30           <tbody>
31             <tr th:each="actor : ${actors}">
32               <td th:text="${actor.id}"></td>
33               <td th:text="${actor.name}"></td>
34               <td th:text="#{#dates.format(actor.birthDate, 'dd-MM-yyyy')}"></td>
35               <td th:text="#{#dates.format(actor.deadDate, 'dd-MM-yyyy')}"></td>
36               <td th:text="${actor.country}"></td>
37               <td>
38                 <a th:href="@{'/editActor' + ${actor.id}}" class="btn btn-primary">Editar</a>
39               </td>
40             </tr>
        
```

**Right Repository:** https://github.com/evah02/moviecards/blob/master/src/main/java/com/lauracercas/moviecards/model/Actor.java

- File Structure:** .github/workflows, .mvn, src (main, java/com/lauracercas/movieca..., controller, model/Actor.java, model/Card.java, model/Movie.java), repositories, service, util.
- Actor.java Content:**

```

14
15 /**
16  * Autor: Laura Cercas Ramos
17  * Proyecto: TFM Integración Continua con GitHub Actions
18  * Fecha: 04/06/2024
19  */
20 @Entity
21 public class Actor {
22     @Id
23     @GeneratedValue(strategy = GenerationType.AUTO)
24     private Integer id;
25
26     private String name;
27
28     @DateTimeFormat(pattern = "yyyy-MM-dd")
29     private Date birthDate;
30     @DateTimeFormat(pattern = "yyyy-MM-dd")
31     private Date deadDate;
32
33     private String country;
34
35     @ManyToMany(mappedBy = "actors")
36     private List movies;
    }
  
```

- Gestión de ramas:** Siguiendo metodologías ágiles, los cambios se desarrollaron en una rama secundaria y, tras verificar su funcionamiento, se integraron en la rama master.

### 6.3. Pruebas y calidad de código

Para asegurar que los cambios no introdujeran errores, se actualizaron todos los niveles de pruebas:

- Pruebas unitarias e integración:** Se ajustaron los tests para validar que el objeto Actor procesa correctamente el nuevo atributo y que la comunicación con el servicio REST es estable.

```

moviecards / src / test / java / com / lauracercas / moviecards / unittest / model / ActorTest.java
Code Blame 61 lines (47 loc) · 1.48 KB
14 public class ActorTest {
15     void testSetName() {
16         ...
17     }
18
19     @Test
20     void testSetGetBirthDate() {
21         Date birthDateExample = new Date();
22         actor.setBirthDate(birthDateExample);
23         assertEquals(birthDateExample, actor.getBirthDate());
24     }
25
26     @Test
27     void testSetGetDeadDate() {
28         Date deadDateExample = new Date();
29         actor.setDeadDate(deadDateExample);
30         assertEquals(deadDateExample, actor.getDeadDate());
31     }
32
33     @Test
34     void testSetGetCountry() {
35         String countryExample = "Sample country";
36         actor.setCountry(countryExample);
37         assertEquals(countryExample, actor.getCountry());
38     }
39
40     @Test
41     void testSetGetMovies() {
42         List<Movie> moviesExample = new ArrayList<Movie>();
43         actor.setMovies(moviesExample);
44         assertEquals(moviesExample, actor.getMovies());
45     }
46
47     @Test
48     void testSaveActor() {
49         Actor actor = new Actor();
50         actor.setName("actor");
51         actor.setBirthDate(new Date());
52         actor.setDeadDate(new Date());
53         actor.setCountry("spain");
54
55         Actor savedActor = actorJPA.save(actor);
56
57         assertNotNull(savedActor.getId());
58
59         Optional<Actor> foundActor = actorJPA.findById(savedActor.getId());
60
61         assertTrue(foundActor.isPresent());
62         assertEquals(savedActor, foundActor.get());
63     }
64
65     @Test
66     void testFindById() {
67         Actor actor = new Actor();
68         actor.setName("actor");
69         actor.setBirthDate(new Date());
70         actor.setDeadDate(new Date());
71         Actor savedActor = actorJPA.save(actor);
72     }

```

  

```

moviecards / src / test / java / com / lauracercas / moviecards / integrationtest / repositories / ActorJPAT.java
Code Blame 56 lines (43 loc) · 1.53 KB
20 public class ActorJPAT {
21     ...
22
23     @Test
24     public void testSaveActor() {
25         Actor actor = new Actor();
26         actor.setName("actor");
27         actor.setBirthDate(new Date());
28         actor.setDeadDate(new Date());
29         actor.setCountry("spain");
30
31         Actor savedActor = actorJPA.save(actor);
32
33         assertNotNull(savedActor.getId());
34
35         Optional<Actor> foundActor = actorJPA.findById(savedActor.getId());
36
37         assertTrue(foundActor.isPresent());
38         assertEquals(savedActor, foundActor.get());
39     }
40
41     @Test
42     public void testFindById() {
43         Actor actor = new Actor();
44         actor.setName("actor");
45         actor.setBirthDate(new Date());
46         actor.setDeadDate(new Date());
47         Actor savedActor = actorJPA.save(actor);
48     }
49

```

- Pruebas funcionales (E2E):** Se modificaron las pruebas de extremo a extremo para simular el flujo completo del usuario, desde la inserción de la fecha de fallecimiento hasta su visualización en el listado.

```

https://github.com/evah02/moviecards/blob/master/src/test/java/com/lauracercas/moviecards/endtoendtest/ActorE2ETest.java

Files
master
Go to file t

.moviecards/.github/workflows
.moviecards/.mvn
src/main/test/java/com/lauracercas/moviecards/endtoendtest
  ActorE2ETest.java
  IndexE2ETest.java
  MovieE2ETest.java
integrationtest/repositories
unittest
.gitignore
mvnw
mvnw.cmd
pom.xml

Code Blame 83 lines (65 loc) · 2.94 KB
22 public class ActorE2ETest {
23     public void testPageLoad() {
24
25         assertTrue(driver.findElement(By.id("name")).isDisplayed());
26         assertTrue(driver.findElement(By.id("birthdate")).isDisplayed());
27         assertTrue(driver.findElement(By.id("deadDate")).isDisplayed());
28         assertTrue(driver.findElement(By.id("country")).isDisplayed());
29     }
30
31     @Test
32     public void testNewActorTitle() {
33         driver.get("http://localhost:8089/actors/new");
34         WebElement title = driver.findElement(By.className("title"));
35         assertEquals(NEW_ACTOR_TITLE, title.getText());
36     }
37
38     @Test
39     public void testListActors() {
40         driver.get("http://localhost:8089/actors");
41         WebElement title = driver.findElement(By.className("card-header"));
42         assertEquals("Listado Actores", title.getText());
43
44         WebElement table = driver.findElement(By.className("table-hover"));
45
46         WebElement thead = table.findElement(By.tagName("thead"));
47         assertTrue(thead.isDisplayed());
48
49         WebElement headerRow = thead.findElement(By.tagName("tr"));
50         assertEquals("Identificador", headerRow.findElements(By.tagName("th")).get(0).getText());
51         assertEquals("Nombre", headerRow.findElements(By.tagName("th")).get(1).getText());
52         assertEquals("Fecha Nacimiento", headerRow.findElements(By.tagName("th")).get(2).getText());
53         assertEquals("Fecha Fallecimiento", headerRow.findElements(By.tagName("th")).get(3).getText());
54         assertEquals("País", headerRow.findElements(By.tagName("th")).get(4).getText());
55         assertEquals("Editar", headerRow.findElements(By.tagName("th")).get(5).getText());
56
57         WebElement headerRow = thead.findElement(By.tagName("tr"));
58         assertEquals("Identificador", headerRow.findElements(By.tagName("th")).get(0).getText());
59         assertEquals("Nombre", headerRow.findElements(By.tagName("th")).get(1).getText());
60         assertEquals("Fecha Nacimiento", headerRow.findElements(By.tagName("th")).get(2).getText());
61         assertEquals("Fecha Fallecimiento", headerRow.findElements(By.tagName("th")).get(3).getText());
62         assertEquals("País", headerRow.findElements(By.tagName("th")).get(4).getText());
63         assertEquals("Editar", headerRow.findElements(By.tagName("th")).get(5).getText());
64
65         WebElement headerRow = thead.findElement(By.tagName("tr"));
66         assertEquals("Identificador", headerRow.findElements(By.tagName("th")).get(0).getText());
67         assertEquals("Nombre", headerRow.findElements(By.tagName("th")).get(1).getText());
68         assertEquals("Fecha Nacimiento", headerRow.findElements(By.tagName("th")).get(2).getText());
69         assertEquals("Fecha Fallecimiento", headerRow.findElements(By.tagName("th")).get(3).getText());
70         assertEquals("País", headerRow.findElements(By.tagName("th")).get(4).getText());
71         assertEquals("Editar", headerRow.findElements(By.tagName("th")).get(5).getText());
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("País", headerRow.findElements(By.tagName("th")).get(4).getText());
79         assertEquals("Editar", headerRow.findElements(By.tagName("th")).get(5).getText());

```

- Control de Calidad con SonarQube:** Se estableció un umbral crítico de calidad; si el análisis detecta **5 o más problemas críticos**, el pipeline detiene automáticamente el proceso, impidiendo el despliegue a producción.

localhost:9000/projects

You're running a version of SonarQube that is no longer active. Please upgrade to an active version immediately. [Learn More](#)

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

My Favorites All

Filters

Quality Gate: Passed (1), Failed (0)

Bugs: A rating (1), B rating (0), C rating (0), D rating (0), E rating (0)

Reliability: Bugs: A rating (1), B rating (0), C rating (0), D rating (0), E rating (0)

2 project(s)

Perspective

moviecards Passed

Bugs: 0 A, Vulnerabilities: 0 A, Hotspots Reviewed: 0 A, Code Smells: 45 A

MovieCards

Project's Main Branch is not analyzed yet. [Configure analysis](#)

localhost:9000/issues?resolved=false

You're running a version of SonarQube that is no longer active. Please upgrade to an active version immediately.

sonarqube Projects Issues Rules

My Issues All

Filters

Type: Bug (0), Vulnerability (0), Code Smell (45)

Severity: Blocker (2), Critical (0), Minor (1), Info (42), Major (0)

Los acciones resultantes de todas las fases mencionadas después de completar la parte de la práctica 5 guiada son los que se muestran a continuación, en los cuales se han ido realizando los cambios requeridos por la práctica.

https://github.com/evah02/moviecards/actions

Actions New workflow

All workflows

CI

Management

Caches

Attestations

Runners

Usage metrics

Performance metrics

Workflow stage

Add or update the Azure App Service build and deployment workflow config

Add or update the Azure App Service build and deployment workflow config

Merge pull request #10 from evah02/cambios-moviecards-service

Cambios src/main y src/test

Chat

6m 1s

Feb 9, 4:17 PM GMT+1 1m 34s

Feb 9, 4:00 PM GMT+1 1m 26s

Feb 9, 4:00 PM GMT+1 7m 48s

Feb 9, 3:39 PM GMT+1 4m 51s

Feb 9, 3:33 PM GMT+1 4m 37s

The screenshot shows the GitHub Actions page for the repository `moviecards`. The left sidebar has sections for CI, Management (Caches, Attestations, Runners, Usage metrics, Performance metrics), and a New workflow button. The main area is titled "All workflows" and shows "27 workflow runs". A search bar at the top right says "Filter workflow runs". Below the search bar is a "Help us improve GitHub Actions" card with a "Give feedback" button. The workflow runs table lists several entries:

Event	Status	Branch	Actor	Date	Duration	More
Merge pull request #24 from evah02/cambios-criticos	master			Feb 12, 10:22 PM GMT+1	1m 33s	...
Problemas criticos resueltos	cambios-criticos			Feb 12, 8:21 PM GMT+1	4m 59s	...
Merge pull request #21 from evah02/añadir-fecha-fallecimiento	master			Feb 12, 6:05 PM GMT+1	5m 37s	...
Cambio return	añadir-fecha-fallecimiento			Feb 12, 5:47 PM GMT+1	16m 13s	...
Añadir campo fecha fallecimiento	añadir-fecha-fallecimiento			Feb 9, 5:19 PM GMT+1	3s	...
Workflow stage orden correcto	master			Feb 9, 4:19 PM GMT+1	6m 1s	...

Finalmente, la aplicación quedaría de la siguiente forma aplicando todos los cambios anteriores.

The screenshot shows the "Gestión de películas" application. The main menu has items like "Inscripción Actor en Película", "Nuevo Actor", "Listado actores", "Listado películas", and "Nueva película". The "Inscripción Actor en Película" section is active, showing a form with fields for "Inscripción Actor en Película" and a "Crear nuevo actor" button. Below it are two cards: "Listado actores" and "Nuevo Actor". The "Nuevo Actor" card has a "Crear nuevo actor" button. Further down are two more cards: "Listado películas" and "Nueva película". The "Nueva película" card has a "Crear nueva película" button.

The screenshot shows the "Listado Actores" section. It displays a table with columns: Identificador, Nombre, Fecha Nacimiento, Fecha Fallecimiento, País, and Editar. Two rows are shown: one for Lady Gaga (Identificador 1) and one for Leonardo DiCaprio (Identificador 3). Both rows have an "Editar" button. At the bottom of the table is a "Volver a la Página de Inicio" button.

Identificador	Nombre	Fecha Nacimiento	Fecha Fallecimiento	País	Editar
1	Lady Gaga	28-03-1986	04-03-1986	EEUU	<button>Editar</button>
3	Leonardo DiCaprio	28-03-1986	04-03-1986	EEUU	<button>Editar</button>

Nuevo Actor

Nombre  
Escriba el nombre del actor

Fecha nacimiento  
dd/mm/aaaa

Fecha fallecimiento  
dd/mm/aaaa

País  
Escriba el país del actor

**Guardar**

[Volver a la Página de Inicio](#)

Editar Actor

Nombre  
Lady Gaga

Fecha nacimiento  
28/03/1986

Fecha fallecimiento  
04/03/1986

País  
EEUU

**Guardar**

**Películas en las que ha participado Lady Gaga**  
No hay ninguna película registrada en la que haya participado este actor

[Volver a la Página de Inicio](#) [Volver al listado de actores](#)