

Desarrollo de Software 3

DoD API Java, Iniciando CIS API C#

Actividad	
Fecha	30/03/2025
Grupo	Dev Titans

Guido Mamani

Daniel Pérez Pérez

Agustín Deluca

Meyer Peña Pidiache

Sofia Beltrán Garcia


















Actividad #1 - Definition of Done - Capstone Project: CI/CD, Bug fixing.....	2
1. Completar los User Stories pendientes.....	2
2. Completar los unittest pendientes.	3
3. Bug Fixing: Arreglar todos los bugs reportados.	4
4. Implementacion del CI/CD.	4
5. Crear la documentación del Rest Api User utilizando Swagger.....	6
Actividad #2 - Implementación del CIS API	7
1. Desarrollo de Epics, Features y User Stories. Claros, y bien definidos. Los criterios de aceptación deben estar detallados y congruentes a cada elemento. Cada tarea desarrollada debe mostrar un avance en el ticket.	7
2. Resultados de Spikes o investigaciones: los resultados tienen que estar documentados al detalle en la Wiki del proyecto.	10
3. Se revisarán los mecanismo de avance de los sprints (Burndown y Burnup).	12
Burndown Chart.....	13
Burnup Chart	13
4. Durante las sesiones de laboratorio, se verificará que cada miembro del equipo haya participado. Es responsabilidad de cada uno estar al tanto de las actividades de los demás miembros del equipo.	13

Actividad #1 - Definition of Done - Capstone Project: CI/CD, Bug fixing

1. Completar los User Stories pendientes.

Se completaron todas las historias de usuarios que quedaron pendientes para finalizar la API de Java y avanzar con la API de C#.

Title	Weight	Assignees	Status
 [US 1.5.1] Crear pruebas unitarias para probar cada servicio #31	3	 Guido Rafael Mamani	Closed
 [US 1.3.5] Refactorización de seguridad #29	5	 Daniel Perez  Meyer Pidiache  Agustin De Luca	Closed
 Arreglar Pipeline #27	1	 Meyer Pidiache	Closed
 Arreglar el Readme #26	1	 Meyer Pidiache	Closed
 [US 1.4.2] Integración de Swagger en la construcción #13	1	 Meyer Pidiache	Closed
 [US 1.4.1] Documentación con Swagger #12	3	 Sofia Beltrán García  Meyer Pidiache	Closed

https://gitlab.com/jala-university1/cohort-3/oficial-es-desarrollo-de-software-3-cssd-232.ga.t1.25.m2/secci-n-c/devtitans/api-java-spring-boot/-/boards?milestone_title=Phase%201%20-%20Specifications

2. Completar los unittest pendientes.

Se realizaron los test unitarios correspondientes a logica del CRUD de usuarios, utilizando las siguientes tecnologías:

- **Spring Boot Starter Test:** Proporciona un entorno de pruebas unificado, incluyendo librerías como JUnit y Mockito.
- **Jacoco:** Configurado para generar reportes de cobertura de código (por defecto con `./gradlew test jacocoTestReport`).

A continuacion detallo los test unitarios realizados:

Controladores de Login

- **SignInControllerTest:** Se validan los escenarios de autenticación exitosa, credenciales inválidas (contraseña incorrecta) y usuario no encontrado (retornando 401).
- **SignUpControllerTest:** Cubre el registro de usuario con datos válidos (retorna 200), datos inválidos (400) y errores de servicio (500).

Controladores de Usuario

- **GetUserControllerTest:** Verifica que, dado un ID, se obtenga un usuario (200) o se retorne 404 si no existe.
- **DeleteUserControllerTest:** Simula la eliminación de un usuario por ID con confirmación "yes", el caso de usuario no encontrado y la solicitud inválida sin confirmación adecuada.
- **UpdateUserControllerTest:** Asegura que, si el usuario existe, se actualicen los campos y retorne 200; y que si no existe, retorne 404.

Resultado de los test:

```
com.devtitans.jalaucis.presentation.controller.user.DeleteUserControllerTest

Test deleteUserById_InvalidMessageConfirm_ReturnsBadRequest() PASSED
Test deleteUserById_ValidRequest_UserExists_ReturnsOk() PASSED
Test deleteUserById_ValidRequest_UserNotFound_ReturnsNotFound() PASSED

com.devtitans.jalaucis.presentation.controller.user.GetUserControllerTest

Test getUserById_UserNotFound_ReturnsNotFound() PASSED
Test getUserById_UserExists_ReturnsOk() PASSED

com.devtitans.jalaucis.presentation.controller.user.UpdateUserControllerTest

Test updateUser_UserNotFound_ReturnsNotFound() PASSED
Test updateUser_UserExists_ReturnsUpdatedUser() PASSED

SUCCESS: Executed 14 tests in 7.7s
```

Se configuro el estado de los test para que muestren un color verde y la palabra "PASSED" para indicar que se pasaron satisfactoriamente las pruebas.

3. Bug Fixing: Arreglar todos los bugs reportados.

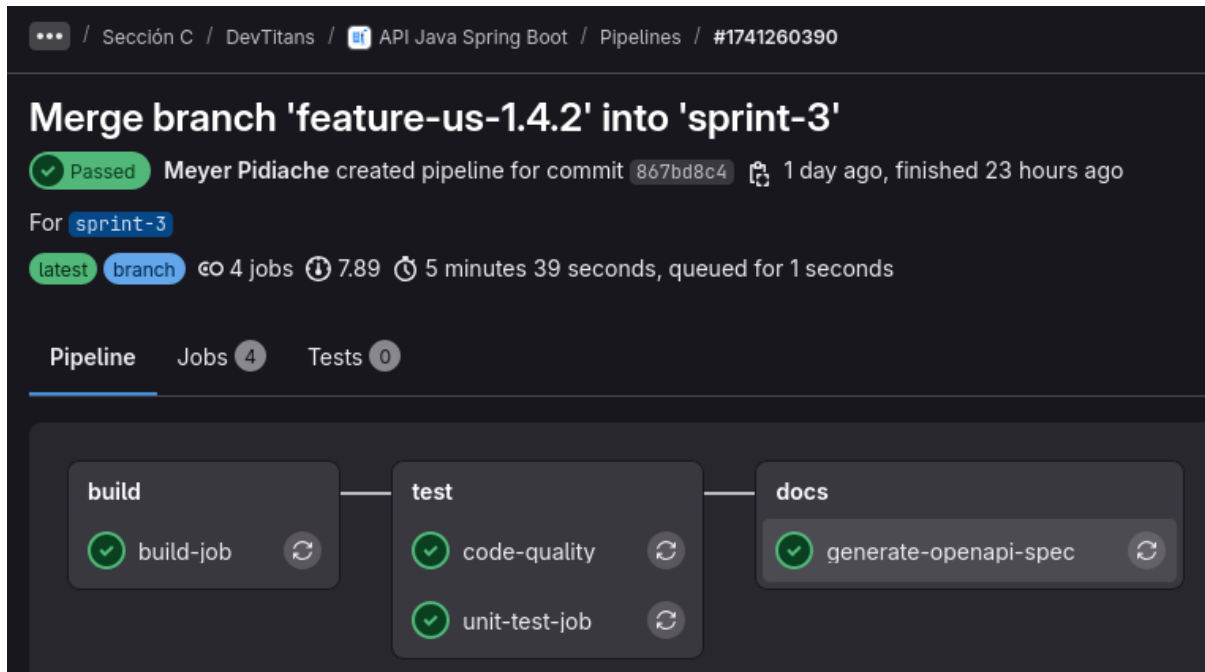
Solo se reportaron bugs correspondientes a:

1. Problemas con la integración con Swagger que se encargó Meyer Peña de resolver, dejo adjunta la historia de usuario. <https://gitlab.com/jala-university1/cohort-3/oficial-es-desarrollo-de-software-3-cssd-232.ga.t1.25.m2/secci-n-c/devtitans/api-java-spring-boot/-/issues/13>
2. Error con migraciones usando EntityFramework, esta tarea la soluciono Daniel Perez. Creando un nuevo proyecto y cambiando de paquete, dejo adjunta la historia de usuario. <https://gitlab.com/jala-university1/cohort-3/oficial-es-desarrollo-de-software-3-cssd-232.ga.t1.25.m2/secci-n-c/devtitans/jala-cis-csharp/-/issues/2>

4. Implementacion del CI/CD.

Para integrar el CI en el proyecto se creo un archivo **.gitlab-ci.yml**, donde se configura el pipeline para que cada vez que se haga un merge request o merge en la rama del sprint o main, se compile el código, se ejecuten los tests y se haga un análisis de código. Utilizando Gradle (con JDK 21) para compilar y correr los tests, y hasta se

configuró un servicio de MySQL para simular la base de datos durante las pruebas. Además, se guardan los reportes de los tests para poder ver rápidamente los resultados y detectar errores. También se incluye la documentación con swagger mediante el pipeline, donde el resultado se puede ver en Artifacts. Ej:



Con esta configuración, cada cambio pasa por un proceso de validación automático, lo que ayuda a mantener la calidad del código y detectar problemas antes de integrarlo en ramas principales como **main** o en las ramas de sprint. Con esto se simplifica mucho el mantenimiento y el despliegue del proyecto.

https://gitlab.com/jala-university1/cohort-3/oficial-es-desarrollo-de-software-3-cssd-232.ga.t1.25.m2/secci-n-c/devtitans/api-java-spring-boot/-/blob/main/.gitlab-ci.yml?ref_type=heads

Es importante aclarar que, en cuanto a CD, aún no se ha implementado. Por ahora, el foco está en asegurar que el código sea validado automáticamente antes de integrarlo en ramas críticas como **main** o las ramas de sprint. Esta configuración de CI ayuda a mantener la calidad del proyecto y sienta las bases para una futura implementación de CD cuando se requiera automatizar el despliegue.

5. Crear la documentación del Rest Api User utilizando Swagger.

Para finalizar el desarrollo y garantizar un uso óptimo de la API, implementamos Swagger. Con esta herramienta, podemos documentar y probar rápidamente todos los endpoints creados. Solo hace falta acceder a <http://localhost:8080/swagger-ui/index.html> para visualizar la documentación interactiva y ejecutar las pruebas correspondientes.

The screenshot displays the Swagger UI for the Jala UCIS API. At the top, the Swagger logo is visible alongside the text 'v1 OAS 3.1'. A search bar contains the path '/v3/api-docs', and an 'Explore' button is located to its right. Below this, the 'Servers' section shows a dropdown menu with the selected server 'http://localhost:8080 - Generated server url' and an 'Authorize' button. The main content area is organized into three expandable sections: 'Autenticación' (Endpoints para inicio de sesión y registro), 'Operaciones de Usuario' (Endpoints relacionados con la gestión de la cuenta del usuario autenticado), and 'Operaciones de Admin' (Endpoints exclusivos para administradores). Each section lists API endpoints with their respective HTTP methods (POST, GET, PUT, DELETE) and a lock icon indicating authentication requirements.

Method	Endpoint	Auth Required
POST	/api/auth/register	Yes
POST	/api/auth/login	Yes
GET	/api/users/{id}	Yes
PUT	/api/users/{id}	Yes
DELETE	/api/users/{id}	Yes
GET	/api/users	Yes
POST	/api/users	Yes
DELETE	/api/users	Yes

Actividad #2 - Implementación del CIS API

Especificaciones:

- Necesitamos persistir los topics e ideas de los usuarios. Por lo tanto, necesitamos extender la base de datos existente para incluir las tablas correspondientes.
- Queremos una API moderna de CIS que pueda manejar las operaciones CRUD, como crear temas, ideas y votar o quitar el voto a las ideas.
- Para usar esta API, necesitamos autenticar a un usuario a través de la API de Usuario.
- Queremos un cliente de API simple para operar y validar esta API de CIS (simular aleatoriamente “n” usuarios con temas, ideas, votar/quitar-voto para validar el sistema).
- Esta API debe estar definida antes de comenzar su implementación.

1. Desarrollo de Epics, Features y User Stories. Claros, y bien definidos. Los criterios de aceptación deben estar detallados y congruentes a cada elemento. Cada tarea desarrollada debe mostrar un avance en el ticket.

La estructuración del proyecto comienza con la definición de una [Epic](#) que a su vez se desglosa en features individuales. Cada funcionalidad se desarrolla a través de historias de usuario detalladas, por ejemplo, [Feature 2.2 Diseño e Implementación de la Base de Datos Actualizada](#).

Child items 14
22
0%
Add

[Feature 2.1] Desarrollo inicial y configuración de API CIS
#8
2
4
Mar 17 – Apr 13, 2025
Open

[Feature 2.2] Diseño e Implementación de la Base de Datos Actualizada
#9
6
18
Mar 17 – Apr 13, 2025
Open

[Feature 2.2] Gestión de Tópicos
jala-university1/cohort-3/oficial-es-desarrollo-de-software-3-cssd-232.ga.t1.25.m2/secci-n-c/devtitans/api-java-spring-boot#22
Open

[Feature 2.5] Conexion con API de usuarios
jala-university1/cohort-3/oficial-es-desarrollo-de-software-3-cssd-232.ga.t1.25.m2/secci-n-c/devtitans/api-java-spring-boot#28
Open

[Feature 2.4] Sistema de Votación de Ideas
jala-university1/cohort-3/oficial-es-desarrollo-de-software-3-cssd-232.ga.t1.25.m2/secci-n-c/devtitans/api-java-spring-boot#24
Open

[Feature 2.3] Gestión de Ideas
jala-university1/cohort-3/oficial-es-desarrollo-de-software-3-cssd-232.ga.t1.25.m2/secci-n-c/devtitans/api-java-spring-boot#23
Open

[Feature 2.2] Diseño e Implementación de la Base de Datos Actualizada
#9
6
18
Mar 17 – Apr 13, 2025
Open

[US 2.2.6] Crear estructura de pruebas unitarias
jala-university1/cohort-3/oficial-es-desarrollo-de-software-3-cssd-232.ga.t1.25.m2/secci-n-c/devtitans/jala-cis-csharp#3
Phase 2 - CIS A...
3
Mar 24 – 30, 2025
No start date – Mar 25, 2025
priority now status refine type story

[US 2.2.5] CRUD de votos
jala-university1/cohort-3/oficial-es-desarrollo-de-software-3-cssd-232.ga.t1.25.m2/secci-n-c/devtitans/jala-cis-csharp#4
Phase 2 - CIS A...
3
Mar 24 – 30, 2025
No start date – Mar 28, 2025
Sprint Backlog priority next

[US 2.2.2] Crear modelos a usar
jala-university1/cohort-3/oficial-es-desarrollo-de-software-3-cssd-232.ga.t1.25.m2/secci-n-c/devtitans/jala-cis-csharp#5
Phase 2 - CIS A...
3
Mar 24 – 30, 2025
Mar 24 – 25, 2025
priority now status in progress type story

[US 2.2.4] CRUD de ideas
jala-university1/cohort-3/oficial-es-desarrollo-de-software-3-cssd-232.ga.t1.25.m2/secci-n-c/devtitans/jala-cis-csharp#6
Phase 2 - CIS A...
3
Mar 24 – 30, 2025
No start date – Mar 28, 2025
priority next status in progress type story

[US 2.2.3] CRUD de tópicos
jala-university1/cohort-3/oficial-es-desarrollo-de-software-3-cssd-232.ga.t1.25.m2/secci-n-c/devtitans/jala-cis-csharp#7
Phase 2 - CIS A...
3
Mar 24 – 30, 2025
No start date – Mar 28, 2025
priority next status refine type story

[US 2.2.1] Configuración estructural del proyecto
jala-university1/cohort-3/oficial-es-desarrollo-de-software-3-cssd-232.ga.t1.25.m2/secci-n-c/devtitans/jala-cis-csharp#8
Phase 2 - CIS A...
3
Mar 24 – 30, 2025
priority now status refine type story

Durante la sesión de Sprint Planning, se lleva a cabo la creación de historias de usuario detalladas, alineadas con las funcionalidades (features) previamente definidas. Este proceso incluye la asignación de responsabilidades a los miembros del equipo y la identificación exhaustiva de las dependencias entre las historias. Se prioriza la claridad y la precisión en la definición de cada historia, asegurando que los criterios de

aceptación



sean

explícitos

y

medibles

[US 2.2.5] CRUD de votos

 Open  Issue created 3 days ago by Meyer Pidiache

User Story 2.2.5: Hacer metodos HTTP en API CIS

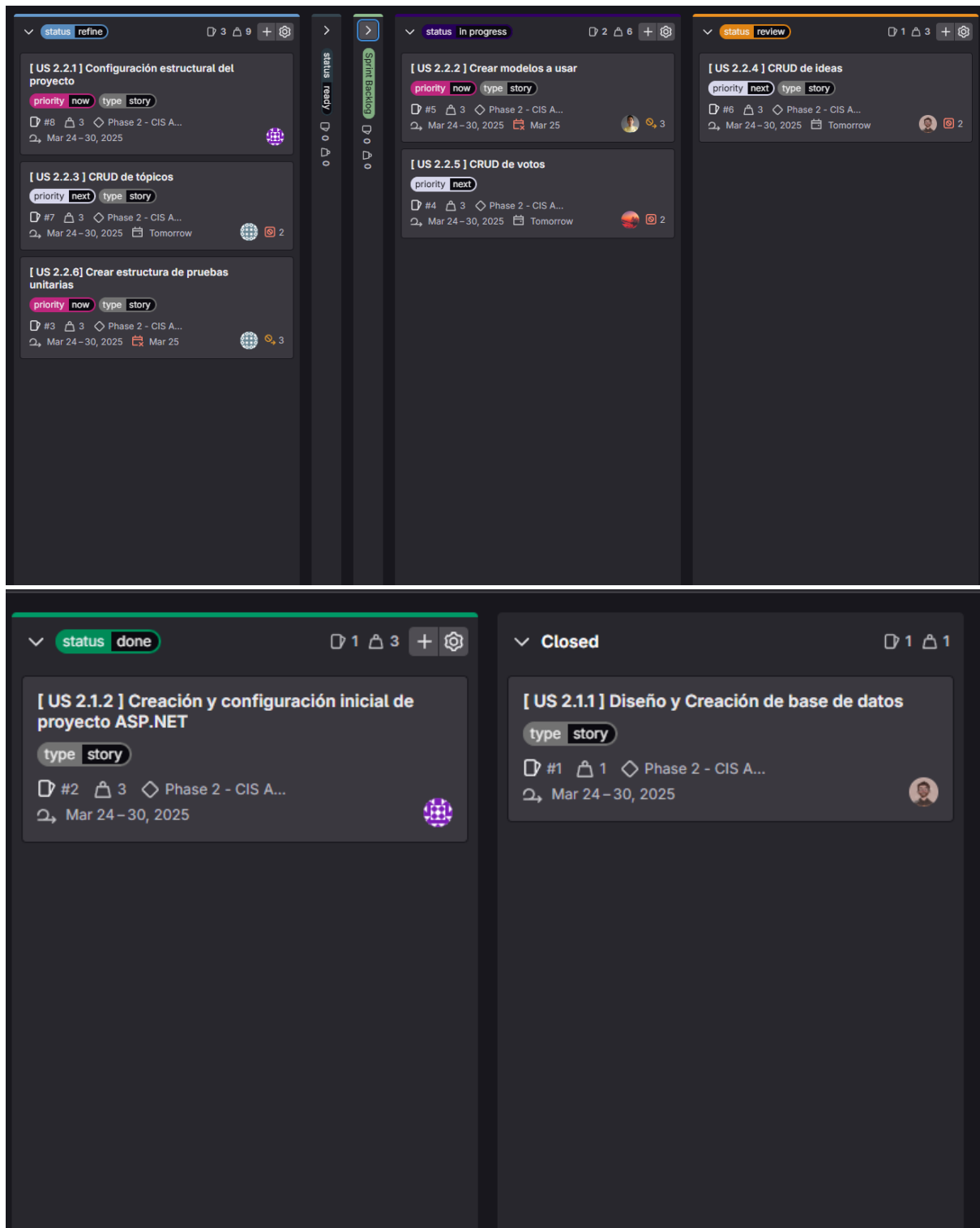
Como usuario final,

Quiero integrar los cuatros operaciones de una api rest,

Para garantizar que los usuarios puedan crear, leer, actualizar y puedan eliminar su voto con respecto a un toppic.

Criterios de Aceptación

- ☑ POST - Crear un voto, Endpoint: POST /api/topics/{topicId}/votes, Valida que el usuario no haya votado previamente en el tema, Respuesta exitosa: HTTP 201 Created + datos del voto en JSON. Conflictos: Si el usuario ya votó, retorna HTTP 409 Conflict.
- ☑ GET - Obtener voto específico, Endpoint: GET /api/votes/{voteId}, retorna HTTP 200 OK con los detalles del voto si existe. Si no existe, retorna HTTP 404 Not Found
- GET - Listar votos, listar por usuario: GET /api/users/{userId}/votes, listar por tema: GET /api/topics/{topicId}/votes, respuesta: HTTP 200 OK + array de votos en JSON.
- PUT - Actualizar voto, Endpoint: PUT /api/votes/{voteId}, Solo el usuario dueño del voto puede modificarlo. Valida el nuevo voteValue. Respuestas(Éxito: HTTP 200 OK + datos actualizados. No autorizado: HTTP 403 Forbidden. No encontrado: HTTP 404 Not Found).
- [] DELETE - Eliminar voto, endpoint: DELETE /api/votes/{voteId}, retorna HTTP 204 No Content tras eliminar. Maneja HTTP 403 (no autorizado) y HTTP 404 (no existe).



2. Resultados de Spikes o investigaciones: los resultados tienen que estar documentados al detalle en la Wiki del proyecto.

Los Spikes, como investigaciones o exploraciones técnicas, se documentan en la wiki del proyecto.

Spikes

Open Epic created 2 weeks ago by **Guido Rafael Mamani**

No description

0 0

Child items 4 6 67%

Add

[SPIKE 4] Investigar sobre el Framework a usar en Fase 2 +3 Open

jala-university1/cohort-3/oficial-es-desarrollo-de-software-3-cssd-232.ga.t1.25.m2/secci-n-c/devtitans/api-cis-csharp-asp.net#4

Phase 2 - CIS A... 1 Mar 24–30, 2025

type spike

[SPIKE 3] Investigar sobre Patrones de Diseño y Arquitectónicos +3 Open

jala-university1/cohort-3/oficial-es-desarrollo-de-software-3-cssd-232.ga.t1.25.m2/secci-n-c/devtitans/api-cis-csharp-asp.net#3

Phase 2 - CIS A... 1 Mar 24–30, 2025

type spike

[SPIKE 1] Investigar un framework de backend para Java +3 Closed

jala-university1/cohort-3/oficial-es-desarrollo-de-software-3-cssd-232.ga.t1.25.m2/secci-n-c/devtitans/api-java-spring-boot#16

Phase 1 - Speci... 1 Mar 10–16, 2025

type spike

[SPIKE 2] Investigar como aplicar seguridad con Springboot +3 Closed

jala-university1/cohort-3/oficial-es-desarrollo-de-software-3-cssd-232.ga.t1.25.m2/secci-n-c/devtitans/api-java-spring-boot#17

Phase 1 - Speci... 3 Mar 10–16, 2025

type spike

Los Spikes quedan documentados en la wiki

Spikes o investigaciones

Spike1-Investigar un fr...

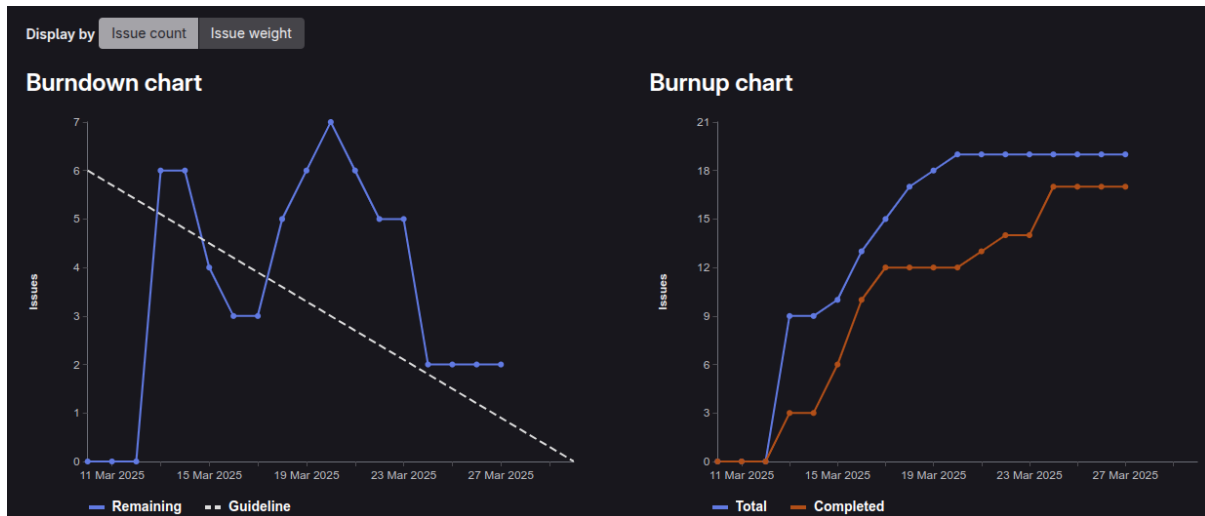
Spike3 Investigar patro...

Spikes2-Investigar co...

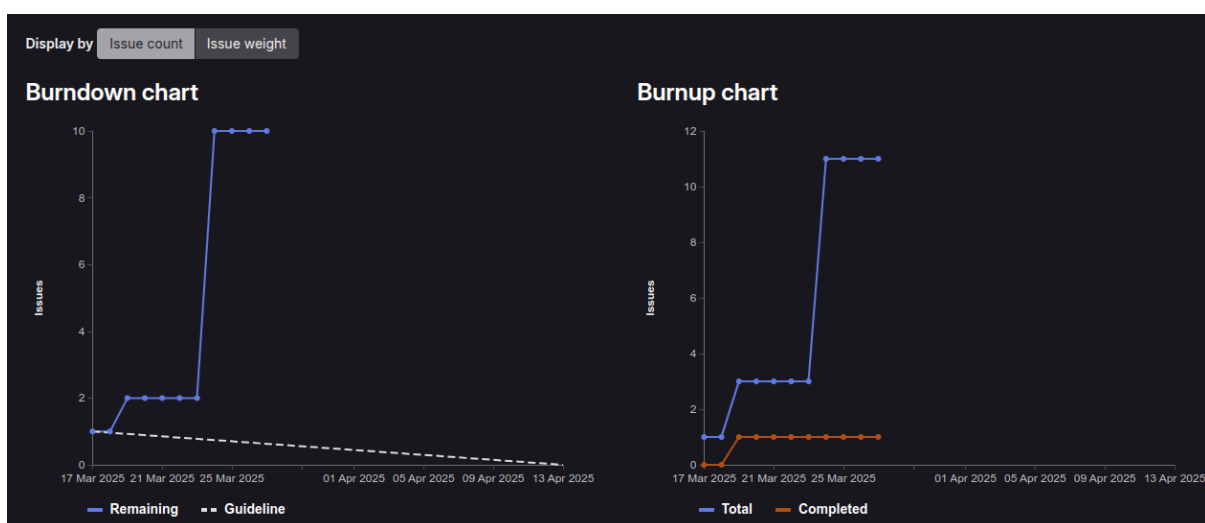
[View all pages](#)

<https://gitlab.com/groups/jala-university1/cohort-3/oficial-es-desarrollo-de-software-3-cssd-232.ga.t1.25.m2/secci-n-c/devtitans/-/wikis/Home/Spikes/Spike3-Investigar-patrones-de-diseño-y-arquitectura>

3. Se revisarán los mecanismo de avance de los sprints (Burndown y Burnup).



- **Cambios en el alcance:** Ambas gráficas reflejan que no se trató de un sprint (o período de trabajo) estático; hubo agregados de nuevos issues o reaperturas que alteraron la línea de “Remaining” y elevaron la línea de “Total”.
- **Recuperación y finalización:** A pesar de estos cambios, la mayor parte de los issues se completaron. Esto sugiere un buen ritmo de entrega o una priorización efectiva, permitiendo cerrar casi todo al final.
- **Planificación vs. Ejecución:** La burndown chart muestra que en ciertos momentos se estuvo por encima de la línea ideal (posiblemente por problemas o nuevos requerimientos), pero el cierre final es positivo porque la diferencia entre issues totales y completados es mínima.



Burndown Chart

- **Posible ingreso tardío de trabajo:** Es probable que se hayan añadido muchos issues de golpe hacia el 21 de marzo.
- **Falta de actualizaciones:** El hecho de que la línea quede plana podría significar que no se registró más progreso, o que el equipo efectivamente no avanzó en la resolución de los issues.
- **Dificultad para cumplir con la planificación:** La burndown chart revela que, tras agregar trabajo, no hubo reducción en el número de pendientes, lo que apunta a una planificación que no se ajustó o a bloqueos en la ejecución.

Burnup Chart

- **Aumento súbito de alcance:** El salto en la línea de “Total” confirma que se incorporaron nuevas tareas o requerimientos a mitad del período.
- **Baja finalización:** La línea de “Completed” casi no sube después de ese punto, por lo que los issues añadidos no se han completado (o no se ha actualizado su estado).
- **Riesgo de arrastre:** Con esa brecha sin moverse, el equipo lleva varios pendientes sin resolver, lo que puede trasladarse al siguiente sprint/iteración.

4. Durante las sesiones de laboratorio, se verificará que cada miembro del equipo haya participado. Es responsabilidad de cada uno estar al tanto de las actividades de los demás miembros del equipo.