

Desarrollo de Software 3

Planificación y definición de requerimientos.

Laboratorio	1
Fecha	08/03/2025
Grupo	Dev Titans

Guido Mamani

Daniel Pérez Pérez

Agustín Deluca

Meyer Peña Pidiache

Sofia Beltrán Garcia



Actividad #1 - Planificación y requerimientos del producto

Organización, implementación de la metodología de Scrum

La organización de cómo se va a implementar la metodología Scrum está en a Wiki del proyecto: <https://gitlab.com/jala-university1/cohort-3/oficial-es-desarrollo-de-software-3-cssd-232.ga.t1.25.m2/secci-n-c/dev-titans/-/wikis/home>

Se debe contar con una Definition of Ready y Definition of Done que esté documentada y accesible por todos los miembros del equipo

Aquí se organizó el DoR y el DoD: [https://gitlab.com/jala-university1/cohort-3/oficial-es-desarrollo-de-software-3-cssd-232.ga.t1.25.m2/secci-n-c/dev-titans/-/wikis/Definition-of-Ready-Definition-of-Done/Definicion-de-Done-\(DoD\)](https://gitlab.com/jala-university1/cohort-3/oficial-es-desarrollo-de-software-3-cssd-232.ga.t1.25.m2/secci-n-c/dev-titans/-/wikis/Definition-of-Ready-Definition-of-Done/Definicion-de-Done-(DoD))

Se debe contar con una definición de Roadmap y Milestones de acuerdo a las fases descritas para el proyecto

Aquí organizamos todo el RoadMap y los milestones donde implementamos de acuerdo con los requerimientos: https://gitlab.com/jala-university1/cohort-3/oficial-es-desarrollo-de-software-3-cssd-232.ga.t1.25.m2/secci-n-c/dev-titans/-/wikis/home/Roadmap_Milestones

Se debe realizar un análisis de requisitos que retorne una lista de Épicas, Features y Historias de usuario

Realizamos el análisis de requisitos para el Proyecto Capstone, basado en la información proporcionada en las tres diapositivas.

El proyecto se divide en tres fases progresivas que buscan modernizar un sistema Legacy, implementar nuevas funcionalidades y migrar la base de datos a una tecnología NoSQL.

Épicas

Épica 1: Modernización del Sistema de Usuarios

Desarrollo de una API moderna para gestionar usuarios que se integre con el sistema legacy existente, garantizando la compatibilidad de datos y la seguridad.

Épica 2: Implementación del Sistema de Gestión de Contenidos (CIS)

Desarrollo de un sistema de gestión de contenidos que se integre con el sistema legacy a través de la API de usuarios desarrollada previamente.

Épica 3: Migración a Base de Datos NoSQL

Migración de la arquitectura de almacenamiento de datos a MongoDB, adaptando las APIs existentes para mantener la compatibilidad y el rendimiento del sistema.

Features

Épica 1: Modernización del Sistema de Usuarios

Feature 1.1: API de Usuarios en Java

Desarrollo de una API RESTful en Java que proporcione operaciones CRUD para la gestión de usuarios, manteniendo la coherencia con el sistema heredado.

Feature 1.2: Sistema de Autenticación y Autorización

Implementación de mecanismos de seguridad para la API de usuarios, incluyendo inicio de sesión y gestión de contraseñas.

Épica 2: Implementación del Sistema de Gestión de Contenidos (CIS)

Feature 2.1: API del CIS en C#

Desarrollo de una API en C# que implemente la funcionalidad del sistema de gestión de contenidos (CIS).

Feature 2.2: Integración con Sistema Legacy

Integración de la API del CIS con el sistema legacy a través de la API de usuarios desarrollada en la fase 1.

Feature 2.3: Módulos de Gestión de Contenidos

Implementación de los módulos funcionales del CIS: Users, Topics, Ideas, y Votes.

Épica 3: Migración a Base de Datos NoSQL

Feature 3.1: Migración a MongoDB

Reemplazo de la base de datos existente por MongoDB, adaptando la estructura para mantener la compatibilidad.

Feature 3.2: Adaptación de APIs Existentes

Modificación de las APIs de usuarios y CIS para que funcionen con la nueva base de datos NoSQL.

Feature 3.3: Migración de Datos Existentes

Desarrollo de herramientas para migrar los datos existentes de la base de datos original a MongoDB con precisión.

Feature 3.4: Optimización de Rendimiento

Implementación de ajustes para mantener o mejorar el rendimiento del sistema después de la migración a MongoDB.

Historias de Usuario

Épica 1: Modernización del Sistema de Usuarios

Feature 1.1: API de Usuarios en Java

- **Historia 1.1.1:** Como administrador del sistema, quiero poder crear nuevos usuarios a través de la API para gestionar el acceso al sistema.
- **Historia 1.1.2:** Como administrador del sistema, quiero poder recuperar información de usuarios existentes para consultar sus datos.
- **Historia 1.1.3:** Como administrador del sistema, quiero poder actualizar los datos de los usuarios para mantener la información actualizada.
- **Historia 1.1.4:** Como administrador del sistema, quiero poder eliminar usuarios para revocar su acceso al sistema cuando sea necesario.
- **Historia 1.1.5:** Como desarrollador, quiero que la API sea coherente con el modelo de datos del sistema legacy para garantizar la integridad de la información.

Feature 1.2: Sistema de Autenticación y Autorización

- **Historia 1.2.1:** Como usuario, quiero poder iniciar sesión en el sistema con mi nombre de usuario y contraseña para acceder a mis recursos.
- **Historia 1.2.2:** Como administrador de seguridad, quiero que las contraseñas estén encriptadas en la base de datos para proteger la información sensible.
- **Historia 1.2.3:** Como administrador del sistema, quiero poder gestionar los permisos de los usuarios para controlar su acceso a diferentes funcionalidades.
- **Historia 1.2.4:** Como usuario, quiero poder cerrar sesión para garantizar la seguridad de mi cuenta cuando no estoy utilizando el sistema.

Épica 2: Implementación del Sistema de Gestión de Contenidos (CIS)

Feature 2.1: API del CIS en C#

- **Historia 2.1.1:** Como desarrollador, quiero implementar una API en C# que proporcione la funcionalidad del CIS siguiendo los estándares de diseño RESTful.
- **Historia 2.1.2:** Como desarrollador, quiero documentar los endpoints de la API del CIS para facilitar su uso por otros equipos.
- **Historia 2.1.3:** Como desarrollador, quiero implementar pruebas unitarias para la API del CIS para garantizar su calidad.

Feature 2.2: Integración con Sistema Legacy

- **Historia 2.2.1:** Como desarrollador, quiero que la API del CIS se comunice con el sistema legacy a través de la API de usuarios desarrollada en la fase 1.
- **Historia 2.2.2:** Como administrador del sistema, quiero que la integración mantenga la consistencia de los datos entre el CIS y el sistema legacy.
- **Historia 2.2.3:** Como usuario, quiero que mis credenciales de acceso sean las mismas para el sistema legacy y el nuevo CIS.

Feature 2.3: Módulos de Gestión de Contenidos

- **Historia 2.3.1:** Como usuario, quiero poder crear y gestionar temas (Topics) en el sistema de gestión de contenidos.
- **Historia 2.3.2:** Como usuario, quiero poder añadir ideas (Ideas) relacionadas con los temas existentes.
- **Historia 2.3.3:** Como usuario, quiero poder votar (Votes) las ideas para expresar mi opinión sobre ellas.
- **Historia 2.3.4:** Como administrador, quiero poder gestionar usuarios del CIS manteniendo la coherencia con el sistema legacy.

Épica 3: Migración a Base de Datos NoSQL

Feature 3.1: Migración a MongoDB

- **Historia 3.1.1:** Como arquitecto de datos, quiero diseñar un esquema en MongoDB que sea compatible con la estructura de datos existente.
- **Historia 3.1.2:** Como desarrollador de bases de datos, quiero implementar el nuevo esquema en MongoDB para reemplazar la base de datos existente.
- **Historia 3.1.3:** Como administrador del sistema, quiero probar el nuevo esquema con datos de prueba antes de la migración completa.

Feature 3.2: Adaptación de APIs Existentes

- **Historia 3.2.1:** Como desarrollador, quiero adaptar la API de usuarios en Java para que sea compatible con MongoDB.
- **Historia 3.2.2:** Como desarrollador, quiero adaptar la API del CIS en C# para que funcione con MongoDB.
- **Historia 3.2.3:** Como tester, quiero validar que todas las funcionalidades de las APIs sigan funcionando correctamente después de la migración.

Feature 3.3: Migración de Datos Existentes

- **Historia 3.3.1:** Como desarrollador, quiero crear un script de migración que transfiera los datos de la base de datos original a MongoDB con precisión.
- **Historia 3.3.2:** Como administrador de datos, quiero verificar la integridad de los datos migrados para asegurar que no se haya perdido información.
- **Historia 3.3.3:** Como administrador del sistema, quiero poder realizar una migración incremental para minimizar el tiempo de inactividad del sistema.

Feature 3.4: Optimización de Rendimiento

- **Historia 3.4.1:** Como desarrollador, quiero implementar índices en MongoDB para optimizar las consultas frecuentes.
- **Historia 3.4.2:** Como arquitecto de rendimiento, quiero realizar pruebas de carga para verificar que el rendimiento del sistema con MongoDB sea al menos igual al de la base de datos original.
- **Historia 3.4.3:** Como usuario, quiero que el sistema mantenga tiempos de respuesta rápidos después de la migración a MongoDB.
- **Historia 3.4.4:** Como administrador de sistemas, quiero monitorizar el rendimiento del sistema para identificar y resolver cuellos de botella.

Es necesario tener un mecanismo para medir el resultado de los sprints (Burndown y Burnup)

Gráficos Burndown y Burnup

Definición

- **Burndown Chart:**
 - Muestra la **cantidad de trabajo pendiente** (en horas, puntos de historia o tareas) durante el sprint.
 - La línea ideal desciende desde el total de trabajo planificado hasta cero al final del sprint.
- **Burnup Chart:**
 - Muestra la **cantidad de trabajo completado** frente al **total de trabajo planificado** (incluyendo cambios en el alcance).
 - La línea de "trabajo completado" sube, mientras que la de "alcance total" puede variar si hay ajustes.

Propósito

- **Burndown:**
 - Visualizar el progreso diario hacia la meta del sprint.
 - Identificar desviaciones (ej: si el equipo va más lento o rápido de lo esperado).
- **Burnup:**
 - Mostrar claramente cómo el trabajo completado avanza frente al alcance total.
 - Útil cuando hay cambios en el sprint (ej: nuevas tareas añadidas).

Elementos Clave

Burndown Chart:

1. **Eje X:** Días del sprint (ej: Día 1, Día 2... Día 14).
2. **Eje Y:** Trabajo restante (ej: puntos de historia).
3. **Línea Ideal:** Descendente desde el total planificado hasta cero.
4. **Línea Real:** Trabajo pendiente día a día.

Burnup Chart:

1. **Eje X:** Días del sprint.
2. **Eje Y:** Trabajo acumulado.

3. **Línea de Trabajo Completado:** Ascendente desde cero hasta el total entregado.
4. **Línea de Alcance Total:** Puede subir si se añaden tareas durante el sprint.

Creación

Burndown:

1. **Planificación:**
 - a. Al inicio del sprint, se sumará esfuerzo total planificado (100 puntos).
2. **Actualización diaria:**
 - a. Se registrará el trabajo restante cada día (ej: Día 1: 90 puntos, Día 2: 80 puntos).
3. **Ejemplo Visual: Trabajo Restante**



Burnup:

1. **Configuración:**
 - a. Se define el trabajo inicial (ej: 50 puntos) y el alcance total inicial.
2. **Actualización:**
 - a. Suma el trabajo completado cada día.
 - b. Si se añaden tareas, ajusta la línea de "alcance total".

Ejemplo Visual: Trabajo Acumulado



En ambos casos se utilizará una [plataforma propia](#).

Análisis

Burndown:

- **Escenario ideal:**
 - La línea real sigue o está por debajo de la línea ideal.
- **Problemas comunes:**
 - **Línea plana:** No se completó trabajo (bloqueos o falta de claridad).
 - **Línea por encima de lo ideal:** El equipo va más lento de lo previsto.
 - **Línea por debajo:** El equipo avanza más rápido (posible sobreestimación inicial).

Burnup:

- **Escenario ideal:**
 - La línea de "trabajo completado" converge con la de "alcance total" al final.
- **Problemas comunes:**
 - **Alcance aumentó:** Si la línea de alcance sube abruptamente, el sprint podría estar sobrecargado.

- **Brecha entre líneas:** Indica que el trabajo completado no alcanza el alcance total.

Creación de proyectos en GitLab y configuración de repositorios (Creación de proyectos, uso de Git, archivo Readme que indique de que trata el nuevo proyecto).

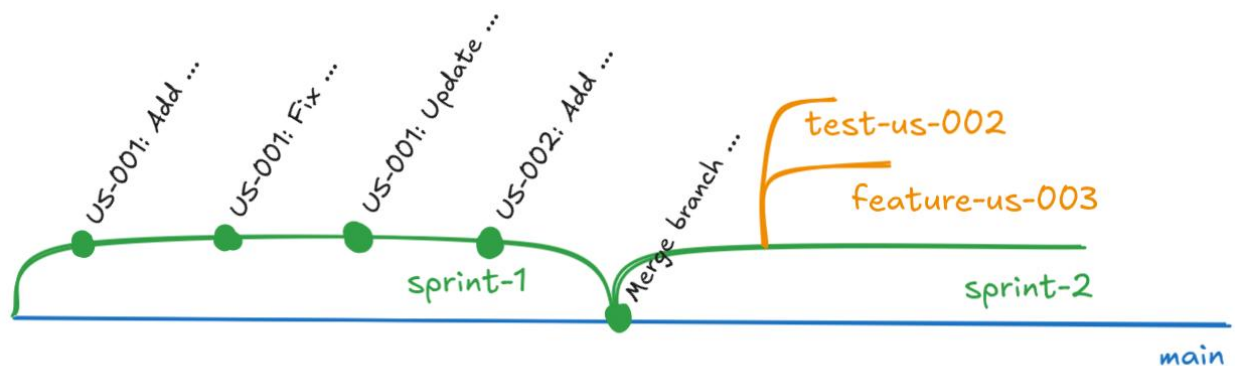
Se nos ha asignado un repositorio de GitLab, a cuyo Readme se ha agregado la información relacionada con el proyecto a trabajar. <https://gitlab.com/jala-university1/cohort-3/oficial-es-desarrollo-de-software-3-cssd-232.ga.t1.25.m2/secci-n-c/dev-titans>

Todas las decisiones técnicas realizadas e investigaciones idealmente deberían de estar documentadas en los Wikis

Solo tomamos dos decisiones técnicas muy importantes que fueron el GIT y la parte de las herramientas a utilizar: <https://gitlab.com/jala-university1/cohort-3/oficial-es-desarrollo-de-software-3-cssd-232.ga.t1.25.m2/secci-n-c/dev-titans/-/wikis/home/Decisiones-T%C3%A9cnicas>

Definición de Branching Model y un proceso de Code Reviews

El Branching Model que se usará es uno basado en rebase, donde se mantiene un historial limpio y lineal.



Para el uso de Git y GitLab como herramientas de control de versiones, se han definido las siguientes convenciones:

- **Ramas:** deben estar en minúscula y siguiendo el formato “feature-us-001”, donde feature puede ser test (para pruebas unitarias), fix (para corregir errores), refactor, docs, ej: docs-us-001. Para los sprints, se usa por ejemplo “sprint-1”-
- **Commits:**
 - En inglés.
 - Significativos.
 - Se va a usar la notación “US-001: Implement JWT token”, donde *US* es de *User Story*, 001 el número correspondiente al código del *PBI* y después de los dos puntos se inicia con un verbo para la descripción del cambio realizado.
 - El verbo de la descripción debe ser imperativo y preferiblemente que estén en esta lista: Implement, Add, Fix, Update, Refactor, Remove, Merge, Correct, Improve, Create, Enhance, Rename, Optimize y Resolve
- **Merge requests:**
 - El título debe seguir la notación “US-001: Implement login functionality”, donde la descripción debe ser el nombre de la historia de usuario (issue).
 - La descripción tiene que ser significativa y en inglés.

Configuraciones personalizadas de Git

Cada desarrollador debe configurar en su local el siguiente hook para los commits, que se agrega en /ruta/repo-local/.git/hooks/commit-msg . Luego tiene que darle permisos de ejecución (chmod +x).

```
#!/bin/bash

COMMIT_MSG_FILE=$1
ALLOWED_VERBS=("Implement" "Add" "Fix" "Update" "Refactor" "Remove" "Merge" "Correct"
"Improve" "Create" "Enhance" "Rename" "Optimize" "Resolve")

# Obtener la primera línea no comentada del mensaje
MSG_HEAD=$(grep -v -m 1 '^#' "$COMMIT_MSG_FILE")

# Validar formato base
if ! [[ "$MSG_HEAD" =~ ^US-[0-9]{3}:\ .+ ]]; then
    echo "❌ Formato inválido. Debe ser: 'US-XXX: Descripción'"
    echo "    Ejemplo: US-001: Implement login functionality"
    exit 1
fi

# Extraer la descripción después del ticket
DESCRIPTION=$(echo "$MSG_HEAD" | awk -F': ' '{print $2}')
FIRST_WORD=$(echo "$DESCRIPTION" | awk '{print $1}')

# Validar verbo inicial
if ! printf '%s\n' "${ALLOWED_VERBS[@]}" | grep -qx "$FIRST_WORD"; then
    echo "❌ Primer palabra debe ser un verbo en imperativo"
    echo "    Verbos permitidos: ${ALLOWED_VERBS[*]}"
    exit 1
fi

exit 0
```

Configuraciones personalizadas de GitLab

Para todos los merge requests, se habilita por defecto el squash de commits con el fin de evitar aglomeramiento de los mismos.

Squash commits when merging


Set the default behavior of this option in merge requests. Changes to this are also applied to existing merge requests. What is squashing?

- ☐ Do not allow
Squashing is never performed and the checkbox is hidden.
- ☐ Allow
Checkbox is visible and unselected by default.
- ☐ Encourage
Checkbox is visible and selected by default.
- ☒ Require
Squashing is always performed. Checkbox is visible and selected, and users cannot change it.

Todas las discusiones tienen que estar resueltas para poder hacer merge.

Merge checks

These checks must pass before merge requests can be merged.

- ☐ Pipelines must succeed
Merge requests can't be merged if the latest pipeline did not succeed or is still running.
 - ☐ Skipped pipelines are considered successful 
Introduces the risk of merging changes that do not pass the pipeline.
- ☒ All threads must be resolved
- ☐ Status checks must succeed
Merge requests can't be merged if the status checks did not succeed or are still running.

Los squash commits tendrán las descripciones de todos los commits de la merge requests.

Squash commit message template

The commit message used when squashing commits.

`%{title}`

`%{all_commits}`

















Leave empty to use default template. Maximum 500 characters. [What variables can I use?](#)

Code Review: Se necesita aprobación de 4 personas para tener aceptados los cambios a la rama principal, y 2 para los sprints. Adicionalmente se previene la edición de estas reglas durante el merge request. Cabe decir que los criterios de aceptación estarán basados en las DoD.

Approval rules 8 2

Add approval rule

Rule	Approvers	Target branch	Approvals required	Actions
Coverage-Check ? Requires approval for decreases in test coverage.				Enable
Minimum required approvals	Any eligible user ?	All branches	0	
Main Branch	    	main	4	 
Sprints	    	sprint-*	2	 

Security Approvals 0

Create more robust vulnerability rules and apply them to all your projects. Learn more.

Create security policy

Rule	Target branch	Approvals required
You don't have any security policies yet.		

Approval settings

Define how approval rules are applied to merge requests. Learn more.

☒ Prevent approval by author

☐ Prevent approvals by users who add commits

☒ Prevent editing approval rules in merge requests

☐ Require user re-authentication (password or SAML) to approve

When a commit is added:

☐ Keep approvals

☒ Remove all approvals

☐ Remove approvals by Code Owners if their files changed

Solo los *Maintainers* pueden hacer merge a main, y nadie puede hacer push directo. Los Developers y Maintainers puede hacer merge a las ramas sprint-*, pero nadie puede hacer push directo.

Branch rules 2

Add branch rule

main default protected

View details

- 1 approval rule
- Allowed to merge: Maintainers
- Allowed to push and merge: No one

sprint-* protected

View details

- 0 matching branches
- 1 approval rule
- Allowed to merge: Developers and Maintainers
- Allowed to push and merge: No one

El flujo de trabajo se configura para que las ramas sprint-* se les haga merge sobre main.

Merge request branch workflow

Branch target ⓘ 1
Add branch target
Create a merge request branch target. Learn more.

Branch name pattern	Target branch	Actions
sprint-*	main	🗑️

Para los push, se previenen commits que tengan archivos con secretos, también cuyos mensajes de commits no sigan la notación.

Push rules

Restrict push operations for this project. Learn more.

Select push rules

☐ Reject unverified users
Users can only push commits to this repository if the committer email is one of their own verified emails.

☐ Reject inconsistent user name
Users can only push commits to this repository if the commit author name is consistent with their GitLab account name.

☐ Reject unsigned commits
Only signed commits can be pushed to this repository.

☐ Reject commits that aren't DCO certified
Only commits that include a Signed-off-by: element can be pushed to this repository.

☐ Do not allow users to remove Git tags with git push
Users can still delete tags through the GitLab UI.

☐ Check whether the commit author is a GitLab user
Restrict commits to existing GitLab users.

☒ Prevent pushing secret files
Reject any files likely to contain secrets. What secret files are rejected?

Require expression in commit messages

All commit messages must match this regular expression. If empty, commit messages are not required to match any expression.

Reject expression in commit messages

Commit messages cannot match this regular expression. If empty, commit messages are not rejected based on any expression.

Branch name

All branch names must match this regular expression. If empty, any branch name is allowed.

Check Branch defaults > Branch name templates for potential conflicts.

Actividad 2 - Diseño y documentación

El Readme solicitado se puede encontrar en: https://gitlab.com/dev-titans/todo_activity2