Universidad de Sevilla

Escuela Técnica Superior de Ingeniería Informática

Grado en Ingeniería Informática – Ingeniería del Software

Evolución y Gestión de la Configuración

Curso 2024 – 2025

Políticas de salmorejo-hub

Grupo salmorejo-hub

salmorejo-hub-1

salmorejo-hub-2

1. Índice

---

2. Política de ***commits***

Esta política sigue el estándar de *Conventional Commits* para asegurar que los mensajes de commit sean claros, consistentes y permitan el rastreo eficiente de cambios y su relación con issues.

2.1. Estructura general de un commit

Cada mensaje de commit debe seguir la siguiente estructura:

<tipo>: <mensaje del commit> [Ref #X | Closes #X]

- **`<tipo>`**: Describe el tipo de cambio realizado. Este debe ser uno de los tipos permitidos (ver más abajo).
- **`<mensaje del commit>`**: Un resumen breve y claro del cambio realizado, en inglés.
- **`[Ref #X | Closes #X]`**: (Solo si hay una issue en GitHub asociada a la tarea) Referencia a un issue de GitHub. `Ref #X` indica una referencia sin cerrar el issue, mientras que `Closes #X` indica que el commit cierra el issue al hacer el *merge* del PR.

2.2. Tipos de Commits Permitidos

1. **`feat:`** Para añadir nuevas funcionalidades.
2. **`fix:`** Para solucionar errores.
3. **`docs:`** Para añadir o modificar documentación.
4. **test:** Para añadir o modificar test
5. **ci:** Para los cambios que afecten a la integración continua

2.3. Ejemplos

- feat: Implement user authentication Ref #15
- fix: Resolve image loading issue in profile Closes #18
- docs: Add usage examples in configuration section Ref #22

2.4. Notas adicionales

- **Lenguaje**: Todos los mensajes deben estar en inglés.
- **Longitud**: Limitar el mensaje del commit a 72 caracteres cuando sea posible, para facilitar su lectura en consolas.
- **Atomicidad**: los commits deben ser atómicos, es decir, deben contener cambios que afecten únicamente a una issue y si es posible que se hagan commits de funciones o vistas en cuanto estén finalizadas.

---

3. Política de ***ramas***

3.1. Estructura general de una rama

El nombre de cada rama debe seguir la siguiente estructura:

<tipo>/<WI[1-6]>-<nombre-del-WI>

- **`<tipo>`**: Describe el tipo de rama. Este debe ser uno de los tipos permitidos (ver más abajo).
- **<WI[1-6]>**: representa el número del work item estipulado. No se hace distinción explícita entre los WI de cada grupo.
- **<nombre-del-WI>**: nombre del work item, separado por guiones.

3.2. Tipos de ramas permitidos

a. **`feat:`** Para añadir nuevas funcionalidades.
b. **`fix:`** Para solucionar errores.
c. **`docs:`** Para añadir o modificar documentación.

3.3. Ejemplo

- feat/WI1-remember-my-password
- fix/WI2-generate-api-token
- docs/WI6-fakenodo

---

4. Política de ***Pull Request***

4.1. Estructura general:

La estructura general de las Pull Request deberá seguir la siguiente estructura:

> **<tipo>/WI[1-2.1-6] [breve descripción] [Ref #X | Closes #X]**

- **`<tipo>`**: Describe el tipo de merge. Este debe coincidir con el de la rama origen o ser **Integration** si pretende añadir funcionalidades de un repositorio a otro de la organización.
- **<WI[1-6]>**: representa el número del work item estipulado, si procede. No se hace distinción explícita entre los WI de cada grupo.
- **<Breve descripción>**: breve descripción de los cambios a integrar.
- **`[Ref #X | Closes #X]`**: Describe si la Pull Request añade funcionalidad y además cierra la issue.

---

5. Política de ***Issues***

5.1. Estructura general:

Las estructura general de las Issues deberá seguir la siguiente estructura:

> **WI-[1-2].[#].[#]: [nombre de la issue]**

- Donde:
  - [1-2]: indica que la issue ha sido creada por algún integrante de salmorejo-hub-1 o, en cuyo caso, de salmorejo-hub-2.
  - [#]: indica el número del WI asociado al integrante en particular.
  - [#]: subnúmero que tiene cierto significado para un WI. Normalmente se usará para crear una sub-issue dentro de dicho WI. Opcional.
- Añadir la label de **incidencia** si ha habido algún problema que sea necesario reproducir. Indicar en la descripción de esta los pasos a seguir para repetir el error sucedido.
  - Estados (mediante un tablero kanban):
    - ToDo: Tarea programada pero aún no empezada.

        - In Progress: tarea en desarrollo.
        - In review: tarea en proceso de prueba.
        - Closed: tarea finalizada e issue cerrada.
    - Prioridades (usar labels de #priority: x):
        - Crítica: Resolver en menos de 3 días. (#priority:critical)
        - Alta: Resolver en menos de una semana. (#priority:medium)
        - Baja: Intentar resolver.. (#priority:low)
    - Tipos (usar labels de #x):
        - Incidencia: El sistema se ha roto o falla una funcionalidad. (#incidence)
        - Documentación: Hace falta documentación de alguna parte (#documentation)
        - Material: Necesidad de alguna vista o funcionalidad para algún trabajo distinto al que se pide. (#material)
        - Tarea: Work item a realizar por el alumnado o alguna funcionalidad a añadir (#enhancement)
        - Despliegue: Tarea relacionada con el despliegue, ya sea local, docker, vagrant (#deployment)
        - Integración: Tarea relacionada con la integración continua (#integration)
        - Otros: Tareas que no pertenecen a ninguna de las categorías anteriores (#others)
- Ejemplos:
    - WI-1: Set up initial project environment
    - WI-2: Create data model for reservations
    - WI-3: Implement user registration for residents

5.2. Definición de hecho:

Una tarea se considerará acabada cuando el código haya sido completado y testeado por completo. Además, se tiene que comprobar que los cambios no afecten negativamente al resto del sistema.

5.3. Plantilla de issues:

En las issues catalogadas como incidencias se deberá seguir la siguiente plantilla:

Description: Briefly explain the problem. Including the context in which the system was when the incident happened.
Steps to reproduce
1. Describe the first step.
2. Describe the second step, if necessary.
3. Describe the third step, if necessary.

Expected result: Explains what should happen.
Current result: Explains what is currently happening.
Additional information (Optional): Provides screenshots, logs or additional context.

5.4. Plantilla de tareas:

En las issues catalogadas como tareas o material se deberá seguir la siguiente plantilla:

Descripción: Explica brevemente la tarea y los objetivos que se quieren alcanzar con ella.
Pasos (actualizar constantemente):

1. Describe el primer paso.
2. Describe el segundo paso, si es necesario.
3. Describe el tercer paso, si es necesario.

Módulos modificados: listar los módulos que han sido modificados con una breve descripción de los cambios.
Información adicional (Opcional): Proporciona capturas de pantalla, logs o contexto adicional.

Ejemplo de plantilla de tareas en inglés:

Docker Release and Deployment

Description: Prepare and execute the Docker release and deployment process to streamline the application's deployment. This process should ensure a smooth and reliable transition from development to production.

Process followed (update constantly):

1. Configure and update the Dockerfile with the necessary dependencies and settings.
2. Build Docker images and test them in a local environment.
3. Push the Docker images to the appropriate registry (e.g., Docker Hub or private registry).
4. Set up CI/CD scripts or pipelines to deploy the images to staging and production environments.
5. Validate the deployment in the target environment, ensuring the system works as expected.

Modified modules:

- **Dockerfile:** Updates to include environment-specific configurations and required dependencies.
- **CI/CD Pipelines:** Configuration or adjustments in the pipelines to support release and deployment.
- **Documentation:** Update technical guides or documentation to reflect the deployment process.

Additional information (Optional):

- Screenshots of build and deployment logs.
- Links to the Docker images uploaded to the registry.
- Additional context on specific configurations or issues encountered during the process.

6. Política de ***Integración***

6.1. Periodicidad y objetivos

Se establece que la política de integración que se llevará a cabo entre salmorejo-hub-1 y salmorejo-hub-2 será 1 vez semanal como mínimo, en la que se integrarán funcionalidades terminadas funcionalmente y testeados correctamente, incluyéndose en el repositorio padre y desplegándose. Se establece también que se realizará entre el lunes y el martes de cada semana, previo a cada sesión de seguimiento y defensa.
(A develop)

6.2. Versionado y release

Se establece un versionado en el repositorio padre con forma A.B.C:

A. El dígito aumenta cada vez que se completa un WI.
B. El dígito se aumenta cada vez que se hace un PR con nueva funcionalidad pero que no completa un WI. Si se completara algún WI junto con funcionalidad de otro WI incompleto, se aumentaría el dígito A y el resto se ponen a 0.
C. El dígito aumenta cada vez que se hagan PR de corrección de errores.

El versionado solo se tiene en cuenta con los cambios que llegan a la rama Main.

Se hará una release cada vez que se complete un WI o cada vez que se solucione un bug crítico. Será la release cuando se despliegue el código.