Normalización de procesos

PixelHub-2

Fecha	11 de noviembre de 2025
Tutor	José Ángel Galindo Duarte
Miembros	Ángel Postigo, Estrella del Carmen Carrasco Mkhazni, Ismael Cerdá Morales, Carlos Founoun Elaoud, Loubna Moraza Vergara, José Luis Terrón Hernández, Diego

Contenido

1. Gestión de cambios e incidencias

Los elementos de trabajo que se diferenciarán en el proyecto para los cambios y las incidencias, son:

Tipo	Descripción
Working Item (WI)	Representa una nueva funcionalidad o requisito a implementar.
Derivados	Engloba tareas de soporte, refactorización o ajustes que no estén directamente relacionados con una nueva funcionalidad.

Niveles de prioridad: un cambio o incidencia puede tener los siguientes niveles de prioridad: *Critical*, *High*, *Medium*, *Low*.

Nivel	Descripción
Critical	Requiere acción inmediata; afecta a la funcionalidad central y bloquea el uso
High	Debe abordarse pronto; afecta significativamente la experiencia del usuario o la calidad del código
Medium	Importante, pero puede planificarse para el próximo ciclo de desarrollo
Low	Sugerencia o mejora menor que puede abordarse cuando el tiempo lo permita

Los cambios e incidencias pueden pasar por los siguientes estados:

Estado	Definición
New	El elemento ha sido recién creado y está pendiente de ser revisado y evaluado por el equipo.
Accepted	El equipo ha revisado el elemento y se ha acordado que será trabajado e incorporado al *backlog*.
Rejected	El elemento ha sido revisado, pero se ha decidido que no se trabajará (ya sea por no ser relevante o estar fuera del alcance).
Started	Un miembro del equipo ha iniciado el desarrollo del cambio o la resolución de la incidencia.
Fixed	La implementación del cambio o la solución de la incidencia se ha completado en la rama de desarrollo.
Verified	Se ha comprobado y revisado la solución por un tercero (revisor o *tester*), confirmando su correcto funcionamiento y calidad.

Plantilla: Para abrir un nuevo cambio o incidencia, usaremos la siguiente plantilla.

- Resumen: Un resumen de una línea sobre el problema.
- Descripción:
  1. ¿Qué pasos reproducirán el problema?
  2. ¿Cuál es el resultado esperado? ¿Qué ves en su lugar?
  3. ¿Qué versión del producto estás usando?
  4. Por favor, proporciona cualquier información adicional a continuación.

2. Gestión del código fuente

Usamos Git para gestionar el código con el flujo denominado EGCFlow. Nuestra rama principal de desarrollo es trunk, y ahí vamos integrando directamente las *features* y los *bugs* con merges, sin usar *pull requests*, usando una rama por tarea. Luego, tenemos una rama

main separada que usamos solo para las releases, esta también se usa con integración continua, *mergeando trunk* en *main*.

Para los commits usaremos *Conventional Commits*, ejemplo:

- "*feat*: permitir la descarga de *datasets* en múltiples formatos"

3. Gestión de la construcción e integración continua

Una eficiente gestión de la construcción e integración continuas (CI/CD) es fundamental para garantizar una mayor calidad en un proyecto de desarrollo Software.

Para este proyecto se automatizarán estas tareas usando GitHub Actions, habiendo definido diversos *workflows* que aseguran que el código pase ciertas pruebas, se integre automáticamente en el repositorio central y se despliegue en Render.

Algunos de los *workflows* ya propuestos por la asignatura, como la revisión del *linting*, *commits* o pruebas se han dejado sin modificar.

Workflow	**Descripción**
CD_render.yml	Despliega automáticamente la aplicación en Render. Activado cuando el workflow *CI_pytest.yml* finaliza con éxito. Además, decide la rama que se despliegue (*main* o *trunk*), según se haga un *push* o una *release*.
CI_autoPR2.yml	Automatiza la integración del código entre el repositorio del equipo y el repositorio central. Activado tras un *push* a la rama *main*. Este *workflow* crea una *pull request* de la rama de equipo a la rama *main* del repositorio central, excluyendo los *workflows*.
CI_commits.yml	*Workflow* base. Fuerza el uso de un formato estandarizado para los mensajes de los *commits*, mediante *Conventional Commits*.
CI_lint.yml	*Workflow* base. Asegura que el código cumple un estilo de programación consistente y detecta errores sintácticos o de estilo, que no son atrapados por pruebas unitarias.
CI_pytest.yml	*Workflow* base. Ejecuta las pruebas unitarias y de integración para validar la calidad. Se activa con cada *push* o *pull request* en las ramas *main* o *trunk*.

4. Pruebas automáticas

La estrategia de pruebas automáticas sigue las pruebas base que vienen de *UVLHub*, extendiéndose para garantizar la calidad de los nuevos *WIs* que se desarrollen. Los tipos de prueba son los que se han visto en la asignatura:

Tipo de prueba	**Descripción**
Unitaria	Verifica la lógica de componentes individuales y de *Python* mediante *Pytest*.
Cobertura	Mide el porcentaje de código fuente que ha sido ejecutado por las pruebas unitarias.
Interfaz	Pruebas que validan la interacción del usuario con la aplicación, mediante *Selenium*.

Carga	Evaluación del rendimiento y la estabilidad del sistema en distintos volúmenes de tráfico.

- Cada nuevo requisito funcional implementado tendrá definidas pruebas.
- Pruebas funcionales unitarias para cada WI, garantizando el correcto comportamiento de la funcionalidad.
- Cada miembro del equipo desarrollará las pruebas de sus WIs.
- No se podrá fusionar una funcionalidad a main sin sus pruebas unitarias implementadas.

5. Automatización de la entrega y el despliegue

El proceso de despliegue del proyecto se ha estructurado para cubrir los tres criterios, de despliegue local, con contenedores y remoto.

1. Despliegue local: está implementado con el entorno de desarrollo virtual, utilizando el comando *flask run* directamente, con las opciones *--debug*, *--reload* y *--host=0.0.0.0*.
2. Despliegue con contenedores: está planificado para implementarse más adelante, facilitando la construcción en un entorno reproducible. Se utilizará *docker* y estará implementado para el tercer *milestone*.
3. Despliegue remoto: el despliegue continuo se realiza con Render. Queda automatizado con los *workflows* explicados en el tercer punto de este documento.