

Universidad de Sevilla

Escuela Técnica Superior de Ingeniería Informática

## **INTEGRACIÓN CONTINUA**




Grado en Ingeniería Informática – Ingeniería del Software

Evolución y Gestión de la Configuración

Curso 2024 – 2025

Proyecto: [innosoft-diplomas-1](#)

Autores
Aragón Sánchez, Álvaro
Chico Castellano, Álvaro
Guillén Fernández, David
Jiménez Osuna, Álvaro
Linares Barrera, Jaime
López Oliva, Ángela

	Evaluación y Gestión de la Configuración Integración Continua
	<b>Control de Versiones</b>

### **Control de versiones**

Fecha	Versión	Descripción
11/11/24	1.0.	Creación y elaboración del documento
07/12/24	1.1.	Última corrección del documento



## Índice de contenido

1. Introducción y objetivos.....	2
2. Herramientas utilizadas.....	2
3. Implementación .....	3
4. Buenas prácticas .....	4
5. Conclusiones y resultados esperados .....	5

## 1. Introducción y objetivos

El presente documento tiene como finalidad definir de manera clara y precisa los acuerdos y directrices que regirán la implementación de la Integración Continua (CI) en el desarrollo del proyecto **innosoft-diplomas-1**. Dichos acuerdos buscan establecer un flujo de trabajo automatizado que garantice la calidad del código y el correcto funcionamiento del sistema a lo largo del ciclo de vida del desarrollo.

## 2. Herramientas utilizadas

Como herramientas para la integración continua, hemos utilizado un **repositorio de GitHub**, que actúa como el centro de control donde se gestiona todo el código fuente del proyecto. En este repositorio, los desarrolladores realizan cambios mediante pushes y pull requests, lo que permite un control centralizado del código y facilita la colaboración entre los miembros del equipo.

Para automatizar el proceso de validación y despliegue, hemos configurado **GitHub Actions**, que se integra directamente con el repositorio y ejecuta diversas tareas de forma automática cada vez que se realizan cambios. Esto asegura que cualquier modificación realizada en el código pase por un conjunto de pruebas antes de ser fusionada a las ramas principales, lo que mejora la calidad del software y reduce el riesgo de introducir errores en producción.

La configuración de GitHub Actions se realiza mediante varios **ficheros YAML** almacenados en el repositorio, específicamente en el directorio `.github/workflows`. Estos ficheros definen los diferentes pasos del pipeline de integración continua, como la instalación de dependencias, la ejecución de pruebas unitarias y de integración, y la validación de la calidad del código mediante herramientas de análisis estático. Además, estos ficheros permiten la automatización del proceso de despliegue, asegurando que, si todas las pruebas pasan correctamente, el código se despliegue automáticamente en el entorno de preproducción o en plataformas como Render. Gracias a esta integración, el proceso de validación y despliegue se realiza de manera eficiente y sin intervención manual, garantizando que siempre se esté trabajando con código validado y listo para ser lanzado.

### 3. Implementación

En la implementación de la integración continua, hemos utilizado varios scripts que automatizan distintos aspectos del flujo de trabajo, asegurando que el código se valide y despliegue correctamente. Los scripts utilizados son los siguientes:

- **commits:** Este script se asegura de que todos los commits realizados en el repositorio sigan las convenciones de **Commits Convencionales**, lo que facilita la lectura y gestión del historial de cambios. Al aplicar esta convención, se mejora la trazabilidad y la automatización de tareas, como la generación de changelogs.
- **deployment-dockerhub:** Este script es responsable de construir una nueva imagen de **Docker** cada vez que se actualiza el código en la rama **main** y se pasan las pruebas. La imagen generada es publicada automáticamente en **Docker Hub**, lo que facilita el despliegue y distribución de la aplicación en diferentes entornos de producción y pruebas.
- **lint:** Este script se encarga de medir la calidad del código mediante análisis estático con **flake8**, una herramienta que evalúa el estilo y las mejores prácticas de Python. El objetivo es asegurarse de que el código sigue un estándar consistente y libre de errores comunes de sintaxis y formato.
- **test:** El script de **test** lanza las pruebas unitarias y de integración previamente configuradas, asegurando que el código se comporta de la manera esperada y no introduce errores. Solo si todas las pruebas pasan con éxito, el pipeline continúa hacia el siguiente paso, lo que asegura que no se introduzcan fallos en el sistema.
- **codacy:** Este script ejecuta **Codacy**, una herramienta que analiza la calidad general del código, proporcionando métricas y recomendaciones sobre cómo mejorar la base de código. Codacy permite detectar posibles vulnerabilidades, problemas de mantenibilidad y otros aspectos clave de calidad.
- **render:** Finalmente, el script **render** se encarga de desplegar la aplicación en la plataforma de **Render**. Este proceso solo ocurre cuando el código de la rama **main** ha pasado correctamente todas las pruebas y validaciones. El despliegue en **Render** asegura que la aplicación esté disponible en un entorno remoto que

simula el entorno de producción, facilitando la validación final antes de un lanzamiento completo.

Cada uno de estos scripts forma parte de un pipeline automatizado que no solo mejora la calidad y seguridad del código, sino que también agiliza el proceso de desarrollo y despliegue, reduciendo el riesgo de errores y garantizando un flujo de trabajo eficiente.

#### 4. Buenas prácticas

En el proyecto, seguimos las buenas prácticas de desarrollo, como realizar **commits diarios a la rama main**, lo que asegura que los cambios se integren de manera constante y fluida. Esta práctica ayuda a evitar conflictos de integración y facilita la gestión del historial del proyecto. Además, se valida cada cambio realizado mediante pruebas automáticas, lo que garantiza que el código no introduzca errores o fallos. Las pruebas se ejecutan en los ficheros de código para verificar que las funcionalidades sean correctas y que no se rompa el sistema al introducir nuevas modificaciones.

Además de las pruebas, se cumple con las buenas prácticas de codificación, como la implementación de convenciones de commits convencionales y el uso de herramientas como flake8 para asegurar que el estilo del código sea consistente. También se realiza una verificación continua de la calidad del código a través de Codacy y se garantiza que el código pase todas las validaciones antes de ser desplegado en producción, lo que minimiza los riesgos y mejora la mantenibilidad del sistema. Estas prácticas no solo mejoran la calidad del código, sino que también optimizan el flujo de trabajo y facilitan la colaboración entre los miembros del equipo.

## 5. Conclusiones y resultados esperados

El proyecto **innosoft-diplomas-1** ha logrado implementar un sistema robusto y eficiente de integración continua, utilizando una serie de herramientas y scripts que automatizan y optimizan el proceso de desarrollo, validación y despliegue. A través de la integración de **GitHub Actions**, se ha logrado automatizar la ejecución de pruebas, la validación del código y el despliegue continuo en plataformas como **Render**. Los scripts que hemos implementado, como los de **commits convencionales**, **Docker**, **Codacy** y **linting**, aseguran que el código no solo sea funcional y libre de errores, sino que también cumpla con los estándares de calidad y buenas prácticas establecidas. Gracias a estas herramientas, todo el flujo de trabajo se realiza de manera controlada, eficiente y sin intervención manual, lo que minimiza el riesgo de errores en producción y acelera el proceso de entrega de nuevas funcionalidades.

### Resultados esperados

- **Mejora en la Calidad del Código:** La implementación de herramientas como **flake8** y **Codacy** asegura que el código se adhiera a los estándares de calidad, identificando problemas potenciales en el estilo, la seguridad y la mantenibilidad. Esto contribuye a una base de código más limpia, fácil de mantener y menos propensa a errores.
- **Entrega Continua de Funcionalidades:** La automatización del proceso de despliegue en **Render** y la ejecución de pruebas en cada **commit** garantizan que las nuevas funcionalidades y correcciones se entreguen rápidamente, sin comprometer la estabilidad de la aplicación. Cada cambio es validado automáticamente, lo que permite un despliegue continuo y sin interrupciones.
- **Reducción de Errores en Producción:** Al automatizar la ejecución de pruebas y la validación del código en cada paso del proceso, se espera una significativa reducción de errores en producción. Solo el código que pasa todas las pruebas y validaciones se fusiona a la rama principal y se despliega en el entorno de producción.
- **Flujo de Trabajo Ágil y Controlado:** Las buenas prácticas de desarrollo, como la realización de **commits diarios** y el uso de **commits convencionales**, aseguran un flujo de trabajo ordenado y colaborativo. Esto mejora la trazabilidad del proyecto y facilita la gestión del código a lo largo del ciclo de vida del desarrollo.



## Evaluación y Gestión de la Configuración Integración Continua

En resumen, la implementación de la integración continua en este proyecto mejora la calidad del código, acelera el proceso de despliegue y asegura que los cambios se integren de manera fluida y controlada, lo que facilita el desarrollo y mantenimiento a largo plazo del sistema.