

Universidad de Sevilla

Escuela Técnica Superior de Ingeniería Informática

PRUEBAS AUTOMÁTICAS




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 Pruebas Automáticas
	Control de Versiones

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. Objetivo de las pruebas automatizadas.....	2
3. Tipos de pruebas	3
4. Estrategia de pruebas	5
5. Conclusión y resultados esperados	7

1. Introducción y objetivos

El presente documento tiene como finalidad establecer de manera clara y precisa los acuerdos y directrices que guiarán la implementación de **pruebas automáticas** en el desarrollo del proyecto **innosoft-diplomas-1**. Estas pruebas buscan asegurar la **calidad y estabilidad del sistema** antes de su despliegue, garantizando que todos los cambios en el código sean validados automáticamente para prevenir errores que puedan afectar la funcionalidad y la experiencia del usuario.

2. Objetivo de las pruebas automatizadas

Las pruebas automáticas en el proyecto **innosoft-diplomas-1** tienen como propósito asegurar la **calidad y estabilidad** del sistema a lo largo del ciclo de desarrollo. Los objetivos principales son los siguientes:

- **Detectar errores de forma temprana durante el ciclo de desarrollo** El uso de pruebas automatizadas permite identificar **fallos en el código** en las primeras etapas del desarrollo, evitando que los errores se acumulen y se conviertan en problemas más complejos y costosos de corregir en fases posteriores. Esto contribuye a una **entrega más rápida y segura** de nuevas funcionalidades, manteniendo la calidad del sistema.
- **Validar la funcionalidad y estabilidad del sistema después de cada cambio en el código** Cada vez que se realiza un cambio en el código, las pruebas automáticas verifican que las **funciones críticas del sistema** continúen funcionando correctamente. Esto incluye la generación de diplomas, la validación de formularios y la integración con otras partes del sistema, asegurando que las actualizaciones no introduzcan **regresiones** ni errores imprevistos.
- **Reducir la intervención manual en el proceso de pruebas para ahorrar tiempo y recursos** Al automatizar las pruebas, se reduce la necesidad de realizar **pruebas manuales repetitivas**, lo que ahorra tiempo y esfuerzo del equipo. Esto permite a los desarrolladores concentrarse en la **implementación de nuevas funcionalidades**, mejorando la eficiencia y agilidad del equipo.

Además, la automatización garantiza la **consistencia** de las pruebas y minimiza la posibilidad de errores humanos.

- **Asegurar que el proyecto cumpla con los requisitos de calidad establecidos** Las pruebas automáticas aseguran que el sistema cumpla con los **estándares de calidad** definidos por el equipo, tanto en términos de funcionalidad como de experiencia del usuario. Esto incluye validar que todas las características implementadas funcionen como se espera y que el sistema sea estable y confiable antes de ser desplegado en producción. De este modo, se mejora la **satisfacción del usuario final** y se reduce el riesgo de problemas en producción.

3. Tipos de pruebas

Se han implementado varios tipos de pruebas automáticas que validan tanto la lógica del backend como la experiencia del usuario en la interfaz web. A continuación, se detallan los tipos de pruebas utilizadas:

Pruebas Unitarias

- Descripción: Las pruebas unitarias tienen como objetivo **validar la correcta funcionalidad de las funciones críticas del backend**. Se centran en probar piezas individuales de código, como métodos y funciones, de forma aislada para asegurarse de que cada una de ellas se comporte como se espera.
- Implementación:
 - Se han desarrollado pruebas para verificar la **validación de datos**, la **generación de diplomas** y el **almacenamiento de archivos**.
 - Estas pruebas se ejecutan automáticamente mediante **GitHub Actions** en cada **push o pull request**, lo que permite detectar errores de forma temprana y evitar que lleguen a producción.
- Ejemplos:
 - Validar que los datos de entrada de los formularios sean correctos antes de proceder a la generación de un diploma.
 - Comprobar que los diplomas se generan y se guardan correctamente en el sistema de archivos.

- Asegurar que las funciones de almacenamiento y recuperación de datos no fallen bajo diferentes condiciones.

Pruebas de Interfaz (UI Testing)

- Descripción: Las pruebas de interfaz se enfocan en **validar la experiencia del usuario** en la plataforma. A través de estas pruebas, se asegura que los usuarios puedan **interactuar con el sistema** de manera fluida y sin problemas.
- Implementación: Utilizando **Selenium**, se automatizan flujos clave de la **interfaz de usuario** para simular las acciones que realizaría un usuario real. Estas pruebas validan la correcta **generación, visualización y descarga de diplomas**, así como la interacción con los formularios.
- Ejemplo: Asegurar que los diplomas generados puedan visualizarse y descargarse sin problemas desde la plataforma.

Pruebas de integración

- Descripción: Las pruebas de integración aseguran que diferentes componentes del sistema interactúen correctamente entre sí. En este caso, se prueba la interacción entre el backend, la base de datos y otros servicios externos, como los servicios de almacenamiento de archivos.
- Implementación:
 - Se validan las rutas de la API y las interacciones entre el backend y la base de datos, asegurando que los datos se procesen y se almacenen correctamente.
 - Estas pruebas también validan que la generación y recuperación de diplomas desde el backend funcione sin errores.
 - Los resultados de estas pruebas se integran en el flujo de **GitHub Actions**, lo que permite detectar fallos de integración rápidamente antes de que el código sea desplegado en producción.
- Ejemplos:
 - Verificar que los datos enviados por los usuarios se almacenen correctamente en la base de datos.

- Validar que los datos recuperados de la base de datos para la generación del diploma sean correctos y se reflejen adecuadamente en el archivo final.

Pruebas de carga

- Descripción: Las pruebas de carga tienen como finalidad evaluar el **comportamiento del sistema bajo condiciones de alta demanda**, asegurando que se mantenga estable y eficiente.
- Implementación: Prueba de cuantos diplomas pueden generarse y enviarse a la vez.

4. Estrategia de pruebas

La **estrategia de pruebas** en el proyecto **innosoft-diplomas-1** está diseñada para asegurar que el sistema se pruebe de manera exhaustiva, minimizando el riesgo de errores antes de su despliegue en producción. A continuación, se detalla el enfoque adoptado para la ejecución de pruebas y los criterios de éxito que deben cumplirse para garantizar la calidad del software a lo largo del ciclo de vida del desarrollo.

Flujo de Trabajo para la Ejecución de Pruebas

1. Ejecución Local por los Desarrolladores

- Objetivo: Detectar errores lo antes posible en el ciclo de desarrollo.
- Descripción: Antes de realizar un commit, los desarrolladores ejecutan pruebas unitarias e integrales de forma local en su entorno. Esto asegura que el código que se sube al repositorio esté libre de errores evidentes y que las funcionalidades críticas del sistema no se vean afectadas por los cambios realizados. Esta etapa de pruebas locales minimiza los riesgos de introducir fallos en el código antes de la integración con otras modificaciones.

2. Integración Continua con GitHub Actions

- **Objetivo:** Validar automáticamente el código antes de la integración.

- **Descripción:** Cada vez que se realiza un **push** o se crea un **pull request** en el repositorio, se activa un pipeline automatizado de pruebas en **GitHub Actions**.
- Este flujo garantiza que cualquier cambio en el código pase por un proceso automatizado de validación antes de ser fusionado con la rama de **preproducción**, asegurando que los errores sean detectados y corregidos a tiempo.

3. Despliegue Condicional

- **Objetivo:** Asegurar que solo código validado se despliegue en entornos de pruebas o producción.
- **Descripción:** Una vez que todas las pruebas se ejecutan correctamente y sin errores, se permite el despliegue automático en el entorno de producción. Este enfoque de despliegue condicional asegura que solo el código que ha pasado todas las pruebas y validaciones llegue a los entornos de pruebas o producción, lo que reduce significativamente el riesgo de introducir errores en el sistema.

4. Criterios de Éxito

- **Cobertura de Pruebas Mínima**

- **Objetivo:** Garantizar que la mayor parte del código esté probado y validado.
- **Descripción:** Se ha establecido un objetivo de cobertura mínima del 60% en las pruebas, lo que asegura que una gran parte del código esté cubierto por pruebas. Esta cobertura incluye pruebas críticas del backend, validación de datos y la generación de diplomas, así como pruebas de la interfaz de usuario para verificar la interacción con los usuarios finales. Este enfoque asegura que tanto las funcionalidades clave como la experiencia del usuario estén debidamente probadas.

- **Ejecución Sin Errores**

- **Objetivo:** Garantizar que todo el código que se despliega esté libre de errores (que se hayan podido detectar con estas pruebas).

- Descripción: Para que un **pull request** sea aprobado y fusionado con la rama de **preproducción**, todas las pruebas deben pasar exitosamente. Cualquier error detectado durante la ejecución de las pruebas en **GitHub Actions** debe ser corregido de inmediato antes de proceder con el despliegue. Este criterio asegura que únicamente el código de alta calidad, sin errores, sea integrado en el proyecto y desplegado en los entornos de producción, lo que mejora la estabilidad y fiabilidad del sistema.

5. Conclusión y resultados esperados

El enfoque de pruebas automáticas implementado en el proyecto **innosoft-diplomas-1** tiene como objetivo principal garantizar la calidad, estabilidad y fiabilidad del sistema a lo largo del ciclo de desarrollo. Las pruebas automatizadas permiten detectar errores de manera temprana, asegurando que cada cambio en el código no introduzca fallos que puedan afectar la funcionalidad o la experiencia del usuario. Este enfoque reduce la intervención manual, mejora la eficiencia del equipo y asegura que el código desplegado cumpla con los estándares de calidad definidos. La combinación de pruebas unitarias, de interfaz, de integración y de carga proporciona una cobertura completa, validando tanto el backend como la experiencia de usuario.

Resultados Esperados

- **Detección temprana de errores:** La automatización de pruebas permitirá identificar fallos en fases tempranas del desarrollo, evitando problemas más complejos en etapas posteriores.
- **Entrega constante de código funcional:** Cada cambio será validado automáticamente, asegurando que las funcionalidades críticas sigan funcionando correctamente sin introducir regresiones.
- **Reducción de errores en producción:** Solo se desplegará código que haya pasado todas las pruebas, lo que reducirá el riesgo de introducir errores en los entornos de producción.



- **Mayor eficiencia del equipo de desarrollo:** Al minimizar las pruebas manuales, los desarrolladores podrán enfocarse más en la implementación de nuevas funcionalidades y mejoras.
- **Cumplimiento de los estándares de calidad:** Las pruebas garantizan que el proyecto cumpla con los requisitos de calidad y rendimiento establecidos, mejorando la satisfacción del usuario final.