

Universidad de Sevilla

Escuela Técnica Superior de Ingeniería Informática

Pruebas Automáticas **(Automated Testing)**




Grado en Ingeniería Informática – Ingeniería del Software
Evolución y Gestión de la Configuración

Curso 2024 – 2025

Fecha	Versión
11/11/2024	1.0.

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

Equipo de trabajo
Aragón Sánchez, Alejandro
Chico Castellano, Álvaro
Guillén Fernández, David
Jiménez Osuna, Álvaro
Linares Barrera, Jaime
López Oliva, Ángela

	Pruebas Automáticas (Automated Testing)
	Control de Versiones

Control de Versiones

Fecha	Versión	Descripción
11/11/2024	1.0.	Creación, elaboración y firma del documento



Pruebas Automáticas (Automated Testing)

Índice de contenido

1. Introducción y objetivos	2
2. Objetivos de la Pruebas Automáticas	2
3. Tipos de Pruebas.....	3
4. Estrategia de pruebas.....	5
5. Resultados Esperados.....	6



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. Objetivos de la Pruebas Automáticas

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:

1. **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.

2. **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.

3. **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.

4. **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:

1. 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.

2. 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.



Pruebas Automáticas (Automated Testing)

- **Ejemplos:**

- Verificar que los usuarios puedan **cargar archivos** y generar diplomas sin errores.
- Probar que los formularios se validen correctamente antes de permitir la generación del diploma.
- Asegurar que los diplomas generados puedan visualizarse y descargarse sin problemas desde la plataforma.

3. 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:**

- **No se han implementado** en este proyecto, ya que la aplicación está diseñada para un número **limitado de usuarios administradores** y no requiere alta concurrencia.
- Dado el alcance actual del sistema, no se espera que el tráfico sea lo suficientemente alto como para justificar la implementación de este tipo de pruebas.



4. Estrategia de pruebas

La **estrategia de pruebas** en el proyecto **innosoft-diplomas-1** está diseñada para garantizar que el sistema sea probado exhaustivamente antes de su despliegue en producción. A continuación, se detalla el enfoque utilizado para la ejecución de pruebas y los criterios de éxito que deben cumplirse.

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.
- Antes de realizar un **commit**, los desarrolladores ejecutan **pruebas unitarias e integrales** localmente en su entorno. Esto asegura que el código que se sube al repositorio esté libre de errores evidentes y que las funcionalidades críticas no se vean afectadas por los cambios realizados.

2. Integración Continua con GitHub Actions

- Una vez que se realiza un **push** o se crea un **pull request** en el repositorio, se activa automáticamente un **pipeline de pruebas** en **GitHub Actions**.
- El pipeline incluye:
 - **Instalación de dependencias.**
 - Ejecución de **pruebas unitarias** y **pruebas de interfaz (UI)** utilizando **Selenium**.
- Este flujo asegura que cualquier cambio en el código pase por un **proceso automatizado de validación** antes de ser fusionado con la rama preproduction.

3. Despliegue Condicional

- Si todas las pruebas se ejecutan **correctamente y sin errores**, se permite el **despliegue automático** en el entorno preproduction.
- El despliegue condicional garantiza que solo el **código que ha pasado todas las validaciones** llegue a los entornos de pruebas o producción, reduciendo el riesgo de introducir errores en el sistema.

Criterios de Éxito

1. Cobertura de Pruebas Mínima

- Para asegurar que la mayor parte del código esté **probado y validado**, se ha establecido un objetivo de **al menos un 80% de cobertura** en las pruebas unitarias.
- Esta cobertura incluye pruebas para funciones críticas del backend, validación de datos, generación de diplomas y pruebas de interfaz de usuario.

2. Ejecución sin Errores

- Todas las pruebas deben **pasar exitosamente** para que un **pull request** sea **aprobado** y fusionado en la rama preproduction.
- Cualquier error detectado durante la ejecución de las pruebas en **GitHub Actions** debe ser **corregido de inmediato** antes de proceder con el despliegue.
- El cumplimiento de este criterio asegura que **solo código de calidad llegue a producción**, reduciendo significativamente los riesgos de fallos en el entorno en vivo.

5. Resultados Esperados

La implementación de **pruebas automáticas** en el proyecto **innosoft-diplomas-1** tiene como finalidad asegurar que el sistema funcione de manera **estable** y **confiable**. Los resultados esperados de este enfoque son los siguientes:

1. Mejora en la calidad del software

- Asegurar que el sistema cumpla con los **requisitos de calidad establecidos** mediante la detección temprana de errores en el ciclo de desarrollo.
- Prevenir la introducción de **regresiones** al validar que nuevas funcionalidades no afecten las ya existentes.

2. Reducción del riesgo en producción

- Minimizar los **errores en el entorno de producción** al automatizar la validación de todas las funcionalidades críticas antes del despliegue.
- Detectar problemas potenciales antes de que afecten a los usuarios finales, lo que se traduce en **menores tiempos de inactividad** y mayor satisfacción del cliente.

3. Mayor eficiencia en el ciclo de desarrollo

- **Acelerar el proceso de desarrollo** al reducir la necesidad de pruebas manuales repetitivas, lo que libera tiempo para que el equipo se concentre en desarrollar nuevas funcionalidades y mejoras.
- Facilitar la **integración continua (CI)** al permitir que los desarrolladores reciban retroalimentación rápida sobre los cambios realizados en el código.

4. Estabilidad y confiabilidad del sistema

- Asegurar que tanto las funciones internas del backend como la **interfaz de usuario** funcionen sin problemas, lo que garantiza una **experiencia fluida** para los usuarios.
- Validar los flujos de trabajo críticos, como la **generación y descarga de diplomas**, para que los usuarios puedan utilizar la plataforma sin interrupciones.

5. Mejora en la colaboración del equipo

- Fomentar un **entorno de trabajo colaborativo** mediante la automatización de las pruebas, lo que permite que el equipo se enfoque en tareas de mayor valor añadido.
- Asegurar que el código que se fusiona en la rama preproduction haya pasado todas las **validaciones automáticas**, aumentando la confianza en la estabilidad del proyecto.

6. Reducción de costos y tiempos de mantenimiento

- Detectar errores de forma temprana reduce los **costos asociados a la corrección** de problemas en etapas avanzadas del desarrollo.
- Facilitar el **mantenimiento a largo plazo** del sistema mediante pruebas automatizadas que aseguran que el código siga siendo robusto y fácil de modificar en el futuro.