# **Manual de Planeación de Pruebas de Software**

# **Sistema: GestorPlus**

# 1. Introducción

Este documento describe el plan de pruebas para el sistema GestorPlus, cuyo objetivo es asegurar la calidad, funcionalidad, seguridad y rendimiento del software antes de su despliegue en producción.

## 2. Alcance

Pruebas sobre los módulos principales: usuarios, ausencias, publicaciones, notificaciones, autenticación, etc.

Pruebas de integración entre módulos.

Pruebas de seguridad y rendimiento.

Pruebas de despliegue e implantación.

### 3. Estrategia de Pruebas

3.1. Tipos de Pruebas

Pruebas Unitarias:  
Verifican el correcto funcionamiento de funciones y métodos individuales.  
Herramienta: PHPUnit  
Ubicación: [models](vscode-file://vscode-app/opt/visual-studio-code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html), [controllers](vscode-file://vscode-app/opt/visual-studio-code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)

Pruebas de Integración:  
Validan la interacción entre módulos y componentes.  
Herramienta: PHPUnit, scripts de integración  
Ubicación sugerida: /backend/test/integration/

Pruebas Funcionales:  
Aseguran que los requerimientos del usuario se cumplen.  
Herramienta: Postman, Selenium, pruebas manuales  
Ubicación de reportes: [html](vscode-file://vscode-app/opt/visual-studio-code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)

Pruebas de Seguridad:  
Detectan vulnerabilidades como inyección SQL, XSS, CSRF, etc.  
Herramienta: OWASP ZAP, scripts propios  
Ubicación: [SecurityTesting](vscode-file://vscode-app/opt/visual-studio-code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)

Pruebas de Rendimiento y Estrés:  
Evalúan el comportamiento bajo carga y la estabilidad.  
Herramienta: JMeter, Artillery  
Ubicación: [StressTesting](vscode-file://vscode-app/opt/visual-studio-code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)

Pruebas de Implantación/Despliegue:  
Validan la correcta instalación y configuración en los entornos de destino.  
Herramienta: Scripts de despliegue, Docker  
Ubicación: [deployment](vscode-file://vscode-app/opt/visual-studio-code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)

#### 4. Organización y Responsabilidades

|  |  |
| --- | --- |
| Tipo de Prueba | Responsable |
| asd | Desarrolladores |
| Integración | Desarrolladores |
| Funcionales | QA/Testers |
| Seguridad | QA/Security |
| Rendimiento/Estrés | QA/DevOps |
| Implantación | DevOps |

##### 5. Criterios de Aceptación y Salida

Todas las pruebas unitarias y de integración deben pasar sin errores.

Las funcionalidades críticas deben estar cubiertas por pruebas funcionales.

No deben existir vulnerabilidades críticas abiertas.

El sistema debe soportar la carga esperada sin degradación significativa.

El despliegue debe ser reproducible y sin errores.

###### 6. Planificación y Cronograma

|  |  |  |  |
| --- | --- | --- | --- |
| Fase | Fecha Inicio | Fecha Fin | Responsable |
| Pruebas Unitarias | 2025-05-26 | 2025-05-28 | Dev Team |
| Pruebas Integración | 2025-05-29 | 2025-05-30 | Dev Team |
| Pruebas Funcionales | 2025-05-31 | 2025-06-02 | QA |
| Pruebas Seguridad | 2025-06-03 | 2025-06-04 | QA/Security |
| Pruebas Rendimiento | 2025-06-05 | 2025-06-06 | QA/DevOps |
| Pruebas Implantación | 2025-06-07 | 2025-06-08 | DevOps |

(Fechas de ejemplo)

7. Documentación y Reportes

Resultados de pruebas unitarias: Reportes HTML en [html](vscode-file://vscode-app/opt/visual-studio-code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html).

Casos de prueba funcionales: Documentados en Markdown, Excel o herramientas de gestión.

Incidentes y bugs: Registrados en el sistema de seguimiento (Jira, GitHub Issues, etc.).

Evidencias de pruebas de seguridad y estrés: Guardadas en sus respectivas carpetas.

8. Riesgos y Contingencias

|  |  |
| --- | --- |
| Riesgo | Mitigación |
| Cambios de requerimientos tardíos | Revisión continua y comunicación constante |
| Falta de cobertura en pruebas | Revisar y actualizar casos de prueba |
| Errores en datos de prueba | Validar y versionar datos de prueba |
| Problemas de integración continua | Automatizar pruebas y despliegues |

9. Ejecución de Pruebas

Unitarias:

docker exec -it gestorplus-php vendor/bin/phpunit --testdox

Reporte HTML:

docker exec -it gestorplus-php vendor/bin/phpunit --testdox-html=test/html/report.html

Funcionales:  
Ejecutar casos de prueba en Postman o Selenium.

Seguridad:  
Ejecutar OWASP ZAP o scripts de auditoría.

Rendimiento:  
Ejecutar JMeter o Artillery con los escenarios definidos.

10. Revisión y Actualización

El plan de pruebas debe revisarse y actualizarse en cada ciclo de desarrollo o cuando cambien los requerimientos del sistema.

11. Anexos

Ejemplos de casos de prueba.

Matriz de trazabilidad de requerimientos.

Evidencias de ejecución.