

Evidencia GA5-220501121-AA2-EV01

Informe Técnico Implementación de CI CD

Aprendiz: Juan Carlos Lopez Moreno

Instructor: Alvaro Esteban Betancourt Matoma

Área Técnica

Servicio Nacional de Aprendizaje – SENA

Técnico en Programación de Aplicaciones y Servicios Para la Nube

Código: 3186263

Diciembre 2025

1. Introducción

La modernización del ciclo de vida de desarrollo de software en el proyecto de la plataforma web de comercio electrónico Finca Miraflores ha culminado con la implementación exitosa de un pipeline de Integración Continua (CI) y Entrega Continua (CD). Este informe técnico detalla las labores realizadas para automatizar la construcción, prueba y despliegue de la aplicación, abandonando los procesos manuales propensos a errores en favor de un flujo de trabajo estandarizado y resiliente. La adopción de estas prácticas responde a la necesidad crítica de garantizar la calidad del código en un entorno de desarrollo ágil, permitiendo al equipo técnico detectar inconsistencias de manera temprana y entregar valor a los usuarios finales con mayor frecuencia y confianza.

El núcleo de esta implementación se basa en la orquestación de herramientas nativas de la nube que se alinean con la arquitectura serverless del proyecto. Se ha establecido una sinergia operativa entre el sistema de control de versiones y la plataforma de despliegue, creando un ecosistema donde cada cambio en el código fuente es sometido rigurosamente a validaciones automáticas antes de ser considerado apto para producción. Este documento expone cómo la integración de tecnologías como GitHub Actions y Vercel ha permitido establecer un estándar de calidad elevado, asegurando que la plataforma, construida sobre Next.js y Supabase, mantenga su integridad funcional y disponibilidad durante cada iteración del software.

2. Alcance

El alcance de este informe y de la implementación técnica descrita abarca la totalidad del flujo de trabajo de DevOps para la aplicación web de Finca Miraflores, desde el momento en que un desarrollador realiza una confirmación de cambios (commit) en el repositorio hasta que dichos cambios están disponibles para los usuarios finales en el entorno de producción. Se

detallan específicamente las configuraciones realizadas para la validación automática de la sintaxis del código, la verificación de tipos estáticos y la construcción de la aplicación, así como los mecanismos de despliegue automático y gestión de entornos.

Dentro de este marco operativo se incluyen las estrategias de protección de ramas principales, la gestión de variables de entorno seguras para el proceso de construcción y la implementación de despliegues de vista previa para la validación de características aisladas. Se excluyen del alcance de este documento las pruebas de carga masiva o la orquestación de infraestructura compleja mediante contenedores orquestados manualmente, dado que la arquitectura actual aprovecha servicios gestionados que abstraen dicha complejidad, permitiendo un enfoque centrado en la lógica de negocio y la eficiencia del despliegue.

3. Lista de Requerimientos y Herramientas

Para la materialización de las prácticas de CI/CD descritas se han requerido y configurado una serie de componentes de software y servicios en la nube que constituyen la infraestructura del pipeline. En primer lugar, se utiliza GitHub como repositorio central de código y plataforma de orquestación de flujos de trabajo mediante GitHub Actions, el cual requiere la definición de archivos de configuración en formato YAML dentro del directorio del proyecto. Para el análisis de calidad del código, se requieren las dependencias de desarrollo ESLint y Prettier, configuradas con las reglas de estilo y buenas prácticas específicas para el framework React y Next.js.

Asimismo, el proceso de compilación y verificación de tipos exige la presencia del compilador de TypeScript y el entorno de ejecución Node.js en su versión LTS, asegurando la compatibilidad con las dependencias del proyecto. Para la fase de entrega continua, es indispensable la cuenta en la plataforma Vercel vinculada al repositorio de GitHub, junto con la

configuración de los secretos y variables de entorno necesarios para la conexión con la base de datos Supabase y la pasarela de pagos Stripe durante la fase de construcción. Finalmente, se requiere una conexión a internet estable y permisos de administrador en el repositorio para configurar las reglas de protección de ramas que hacen cumplir el paso obligatorio de las pruebas automatizadas.

4. Implementación de la Práctica de Integración Continua (CI)

La labor de integración continua se ha consolidado mediante la creación de flujos de trabajo automatizados que se ejecutan inmediatamente después de cada interacción con el repositorio de código. Se ha configurado un archivo de definición de flujo de trabajo en GitHub Actions que instruye al servidor de integración para iniciar una máquina virtual limpia basada en Linux cada vez que se detecta un "push" a la rama principal o la apertura de una solicitud de extracción (Pull Request). Este entorno efímero clona el código más reciente, instala las dependencias del proyecto de manera determinista utilizando el archivo de bloqueo de versiones y procede a ejecutar una serie de scripts de validación secuenciales.

El primer nivel de validación implementado consiste en el análisis estático de código mediante ESLint. Esta herramienta escanea la totalidad de los archivos JavaScript y TypeScript en busca de patrones de código problemáticos, variables no utilizadas, errores de sintaxis y violaciones a las reglas de accesibilidad web predefinidas. Simultáneamente, se ejecuta el compilador de TypeScript en modo de verificación estricta, lo que garantiza que no existan discrepancias de tipos de datos que puedan causar errores en tiempo de ejecución. Esta fase es crítica para mantener la robustez de la aplicación, ya que impide que errores tipográficos o lógicos simples avancen hacia etapas posteriores del desarrollo.

Posteriormente, el pipeline ejecuta el comando de construcción de Next.js en el entorno de prueba. Este paso simula la compilación completa de la aplicación tal como ocurriría en el servidor de producción, verificando la integridad de las páginas estáticas y la correcta generación de las rutas del lado del servidor. Se ha establecido una política de protección en la rama principal del repositorio que impide fusionar cualquier cambio si alguno de estos pasos falla. De esta manera, se garantiza matemáticamente que la rama "main" siempre contiene código compilable, limpio y libre de errores sintácticos evidentes, cumpliendo con el objetivo fundamental de la integración continua de mantener la base de código en un estado siempre desplegable.

5. Implementación de la Práctica de Entrega Continua (CD)

La implementación de la entrega continua se ha realizado aprovechando la integración profunda entre el repositorio de GitHub y la plataforma Vercel, eliminando la necesidad de intervención humana para llevar el código a producción. Se ha configurado un sistema de despliegue atómico que se activa automáticamente tras la finalización exitosa de la fase de integración continua. Cuando un cambio es aprobado y fusionado en la rama principal, Vercel detecta el evento, descarga el nuevo código y comienza el proceso de construcción en su infraestructura global. Este proceso incluye la optimización de activos estáticos, la minificación de scripts y la configuración de las funciones serverless necesarias para la API y el renderizado dinámico.

Un aspecto destacado de la labor realizada es la implementación de entornos de vista previa o "Preview Deployments". Para cada solicitud de cambio propuesta por el equipo de desarrollo, el sistema genera automáticamente una URL única y temporal con una versión funcional de la aplicación que incluye los nuevos cambios. Esto permite realizar pruebas de

aceptación de usuario y validaciones visuales en un entorno idéntico a producción antes de realizar la fusión final del código. Esta capacidad ha transformado el flujo de revisión, permitiendo detectar problemas de diseño o funcionalidad que los tests automatizados no pueden capturar, asegurando que lo que se despliega es exactamente lo que se espera.

Finalmente, se ha asegurado la estabilidad del entorno productivo mediante mecanismos de reversión instantánea y despliegues inmutables. Cada nuevo despliegue en producción no sobrescribe los archivos existentes, sino que crea una nueva instancia de la aplicación. Solo cuando el sistema verifica que la nueva instancia está saludable y respondiendo correctamente, el enrutador global actualiza el tráfico para dirigir a los usuarios a la nueva versión. En caso de detectarse una anomalía crítica post-despliegue, se ha habilitado la capacidad de realizar un "rollback" inmediato a cualquier versión anterior estable con un solo clic, garantizando así la alta disponibilidad y la continuidad del negocio de Finca Miraflores frente a posibles incidentes.