

Integración Continua en la Gestión de Transporte en Chile

TransChile desarrolla su software de manera artesanal y desorganizada. No utilizan control de versiones, y los desarrolladores trabajan de forma aislada. El código se integra manualmente una vez al mes y se sube al servidor por FTP, sin validación ni pruebas automatizadas. Esto resulta en un entorno de desarrollo frágil, propenso a errores en producción y a vulnerabilidades de seguridad.

Análisis del Estado Actual

Problemas Detectados

- ❌ Falta de control de versiones: sin repositorio central, los desarrolladores trabajan en archivos locales y sincronizan el código manualmente una vez al mes.
- ❌ Sin CI/CD: no utilizan Jenkins, GitHub Actions ni GitLab CI/CD; no se ejecutan pruebas automáticas.
- ❌ Problemas de seguridad: uso de paquetes inseguros sin auditoría ni análisis estático.
- ❌ Código de baja calidad: sin herramientas como SonarQube ni revisiones formales.
- ❌ Flujo de trabajo ineficiente: cada desarrollador sube código sin revisión, sin metodología como Git Flow o Trunk-based Development.





Estado actual de TransChile (flujo desorganizado)

Imagen1

Problemas más críticos

- Sin repositorio central ni versiones trazables.
- Sin pruebas ni integración continua.
- Vulnerabilidades por paquetes no auditados.
- Código de baja calidad y errores frecuentes en producción.

Impacto de la situación actual

-  Demoras operativas y errores logísticos.
-  Pérdida de tiempo en resolución de errores evitables.
-  Mayor riesgo de ataques y fallos en producción.
-  Falta de confianza en la entrega continua del software.

Propuesta de implementación de Git y control de versiones

Comparación de flujos de trabajo

Flujo	Ventajas	Desventajas
Git Flow	Ideal para versiones planificadas, controlado	Complejo para equipos pequeños
GitHub Flow	Ligero, perfecto para entregas continuas	Requiere buena gestión de PRs
Trunk-based Development	Flujo simple, fomenta CI diaria	Requiere commits pequeños frecuentes

Imagen2

Se recomienda implementar Git con GitHub Flow, ideal para equipos que hacen entregas frecuentes y necesitan fluidez.

Justificación

- 🤝 Facilita la colaboración y revisión.
- 🔗 Compatible con integración continua.
- 📄 Proporciona trazabilidad y control.

Implementación de integración continua

Herramienta propuesta: GitHub Actions

- 🔗 Integración nativa con GitHub
- 💰 Gratis para proyectos públicos
- ⚙️ YAML flexible para definir flujos

Pipeline de CI/CD propuesto

```
name: CI Pipeline

on:
  push:
    branches: [main]
  pull_request:
    branches: [main]

jobs:
  build-test-deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: Configurar JDK
        uses: actions/setup-java@v3
        with:
          java-version: '17'
      - name: Compilar proyecto
        run: mvn clean compile
      - name: Ejecutar pruebas
        run: mvn test
      - name: Analizar calidad con SonarQube
        run: mvn sonar:sonar
      - name: Despliegue (simulado)
        run: echo "Desplegando a servidor de pruebas..."
```

Estrategias de calidad




- ✓ Pruebas automáticas obligatorias antes del merge
- 🎯 Cobertura mínima del 80% para aprobación
- 🔒 Integración con SonarQube y Dependabot para análisis continuo y actualizaciones seguras

🔒 Seguridad y análisis estático




SonarQube

- 🔍 Detección de vulnerabilidades, errores y código duplicado
- 🧠 Reportes automáticos en cada `push` a la rama `main`
- 🛑 Posibilidad de bloquear despliegues si hay fallas críticas

Infraestructura segura





-  Gestión de roles y permisos en GitHub
-  Uso de claves SSH en procesos de despliegue
-  Auditorías de cambios y trazabilidad del código

Prevención de paquetes inseguros

-  Dependabot activado para monitoreo de dependencias
-  Validación previa de nuevas bibliotecas y versiones
-  Archivo `SECURITY.md` para documentar políticas de seguridad y fuentes confiables





Conclusiones

La empresa TransChile debe modernizar su flujo de desarrollo adoptando herramientas y metodologías clave:

-  Git y GitHub Flow para control de versiones
-  GitHub Actions para automatización y CI/CD
-  SonarQube para control de calidad y seguridad
-  Herramientas de prevención de errores y ciberataques

Esto asegurará un software más seguro, confiable y ágil, mejorando sustancialmente toda la operación logística.

Beneficios Esperados

-  Mayor calidad del código
-  Reducción de errores en entorno productivo
-  Flujo de trabajo colaborativo, trazable y profesional
-  Procesos seguros, auditables y escalables