****

**UNIVERSIDAD PRIVADA DE TACNA**

**FACULTAD DE INGENIERÍA**

**Escuela Profesional de Ingeniería de Sistemas**

**Propuesta del Proyecto**

**“Proyecto *DevFlow”***

**Curso:**

*Patrones de Software*

**Docente:**

*Mag. Patrick Cuadros Quiroga*

**Integrantes:**

*Akhtar Oviedo, Ahmed Hasan - (2022074261)*

*Anampa Pancca, David Jordan - (2022074268)*

*Salas Jimenez, Walter Emmanuel - (2022073896)*

**Tacna – Perú**

*2025*

Proyecto *DevFlow*

Informe de Factibilidad

Versión *1.0*

10/09/2025

| CONTROL DE VERSIONES | | | | | |
| --- | --- | --- | --- | --- | --- |
| Versión | Hecha por | Revisada por | Aprobada por | Fecha | Motivo |
| 1.0 | AAHV, DDAP, WESJ | PCQ |  | 11/09/2025 | Versión Original |

**Tabla de contenido**

Resumen Ejecutivo

I Propuesta narrativa

1. Planteamiento del Problema
2. Justificación del proyecto
3. Objetivo general
4. Beneficios
5. Alcance
6. Requerimientos del sistema
7. Restricciones
8. Supuestos
9. Resultados esperados
10. Metodología de implementación
11. Actores claves
12. Papel y responsabilidades del personal
13. Plan de monitoreo y evaluación
14. Cronograma del proyecto
15. Hitos de entregables

II Presupuesto

1. Planteamiento de aplicación del presupuesto
2. Presupuesto

3. Análisis de Factibilidad

4. Evaluación Financiera

**RESUMEN EJECUTIVO**

**Nombre del Proyecto:** DevFlow  
 **Tipo de Proyecto:** Plataforma tecnológica educativa y profesional  
 **Duración estimada:** 4 semanas  
 **Responsable:** Equipo de desarrollo académico / Facultad de Ingeniería de Sistemas  
 **Ubicación:** Implementación digital (entorno nube)  
 **Presupuesto estimado:** S/. 4,381.00

**Resumen:**DevFlow es una iniciativa tecnológica orientada a integrar de manera modular, práctica y escalable todas las fases del ciclo de vida del desarrollo de software (planificación, análisis, diseño, codificación, pruebas, despliegue y monitoreo). Su propósito es cerrar la brecha entre la enseñanza académica y la práctica profesional, ofreciendo un entorno unificado que permita recorrer el proceso completo de creación de software, utilizando herramientas reales de la industria.

En la actualidad, los desarrolladores —ya sean estudiantes, docentes o profesionales— trabajan en entornos fragmentados que dificultan la trazabilidad, la calidad y la automatización. DevFlow busca resolver este problema mediante un ecosistema de artefactos y plantillas configurables, con ejemplos prácticos que abarcan desde tableros de planificación en Trello hasta pipelines de integración y monitoreo en Datadog y New Relic.

Este enfoque innovador permite que tanto un estudiante universitario como una startup o empresa puedan aplicar metodologías ágiles y DevOps con una curva de aprendizaje reducida y con procesos estandarizados, generando ahorros de tiempo, mayor productividad y entregables de calidad.

**I. Propuesta Narrativa**

**1. Planteamiento del Problema**

El desarrollo de software se enfrenta a una problemática recurrente: la fragmentación de herramientas y la falta de un marco unificado que articule todas las fases del ciclo de vida del software.  
 En los entornos académicos, los cursos de ingeniería de software suelen enseñar fases aisladas —análisis, diseño o programación— sin un flujo integral que permita aplicar las metodologías ágiles de forma real. Esto genera que los estudiantes comprendan los conceptos, pero no logren ver cómo se conectan en un proceso completo.

Por otro lado, en los equipos profesionales, la preparación inicial de entornos, repositorios y pipelines consume un tiempo considerable. Cada proyecto requiere configurar desde cero tableros, repositorios, integraciones y dashboards, repitiendo tareas que podrían automatizarse o estandarizarse. Esta situación provoca ineficiencia, errores humanos y pérdida de trazabilidad, afectando la calidad del software y la velocidad de entrega.

En resumen, la falta de integración y estandarización en el ciclo de vida del desarrollo limita tanto el aprendizaje académico como la productividad profesional.

**2. Justificación del Proyecto**

El proyecto DevFlow se justifica en la necesidad de disponer de un entorno metodológico, práctico y escalable que permita a los usuarios recorrer de principio a fin el ciclo de vida del software con herramientas integradas, entregables concretos y trazabilidad en cada fase.

**Justificación educativa**

En el ámbito universitario, DevFlow servirá como herramienta pedagógica para enseñar metodologías ágiles, DevOps y marcos de calidad de manera aplicada, permitiendo a los estudiantes obtener evidencias tangibles de su aprendizaje. De esta forma, se transforma la enseñanza tradicional —basada en teoría— en una experiencia de “aprender haciendo”.

**Justificación profesional**

En entornos empresariales, DevFlow optimiza la configuración inicial de proyectos, reduce costos y acelera el tiempo de lanzamiento de productos (time-to-market). Además, estandariza buenas prácticas de ingeniería, garantizando consistencia, control de calidad y monitoreo continuo.

**Justificación tecnológica**

La solución se apoya en tecnologías modernas, de código abierto y multiplataforma, lo que la hace accesible y sostenible. Al integrar herramientas como Trello, PlantUML, Figma, GitHub Actions, Jenkins, Datadog y New Relic, se ofrece una experiencia completa de desarrollo sin necesidad de software propietario.

**Justificación social**

DevFlow contribuye al desarrollo del talento digital, promoviendo la empleabilidad de los futuros ingenieros y la profesionalización de equipos locales, impactando positivamente en la competitividad tecnológica del entorno académico y empresarial.

**3. Objetivo General**

Desarrollar e implementar una plataforma modular (DevFlow) que integre de manera unificada y práctica todas las fases del ciclo de vida del desarrollo de software, aplicando principios de agilidad, automatización y trazabilidad para optimizar tanto la enseñanza como la ejecución profesional de proyectos tecnológicos.

**4. Beneficios**

1. **Académicos:**
   * Fomenta el aprendizaje activo mediante la práctica de un flujo completo.
   * Mejora la comprensión de metodologías ágiles y DevOps.
   * Facilita la evaluación docente con entregables medibles.
2. **Profesionales:**
   * Reduce el tiempo de configuración inicial de proyectos.
   * Aumenta la productividad y la calidad del código.
   * Mejora la trazabilidad y reduce los errores humanos.
3. **Institucionales:**
   * Estandariza las prácticas de enseñanza en ingeniería de software.
   * Permite a universidades ofrecer proyectos integrales y modernos.
4. **Sociales:**
   * Contribuye a la formación de profesionales digitales competentes.
   * Fomenta la cultura tecnológica y la innovación local.

**5. Alcance**

El alcance del MVP (Producto Mínimo Viable) se centra en demostrar la integración funcional de todas las fases del ciclo de vida del software mediante un flujo básico completamente operativo:

* **Planificación:** Tablero Trello con flujo ágil (backlog, To Do, Doing, Review, Done).
* **Análisis:** Diagramas UML (.puml) para clases, componentes y secuencias.
* **Diseño:** Wireframes y prototipos en Figma/xPRE con tokens reutilizables.
* **Codificación:** Monorepo con ejemplos para backend (Node.js), frontend (React), móvil (React Native) y escritorio (Electron).
* **Pruebas:** Colecciones Postman (mínimo 3 endpoints) y SonarQube para calidad de código.
* **Despliegue:** Archivos Dockerfile, docker-compose.yml, workflows de GitHub Actions y Jenkinsfile.
* **Monitoreo:** Dashboards básicos en Datadog y alertas configuradas en New Relic.

**Fuera del alcance inicial:**

* Escalado automático en la nube.
* Analítica predictiva y reportes avanzados.
* Módulos multiusuario con roles diferenciados.
* Sincronización en tiempo real.

**6. Requerimientos del Sistema**

**Hardware mínimo:**

* Laptop o PC con 8 GB RAM y procesador de al menos 4 núcleos.

**Software necesario:**

* Node.js, Docker, Git, Visual Studio Code, navegador moderno (Chrome/Edge).

**Conectividad:**

* Internet estable (mínimo 10 Mbps).

**Compatibilidad:**

* Compatible con Windows, macOS y Linux.

**7. Restricciones**

* Dependencia total de conexión a internet.
* Login básico, no institucional.
* No se almacenarán datos personales sensibles.
* Escalabilidad limitada a datasets medianos (<50,000 registros).
* Sin soporte multiusuario ni analítica avanzada en la fase MVP.

**8. Supuestos**

* Los usuarios cuentan con conocimientos básicos de desarrollo de software.
* Los entornos de prueba disponen de recursos técnicos adecuados.
* Se contará con el apoyo de docentes y validadores externos.
* Las herramientas integradas mantienen sus versiones y APIs activas durante la prueba.

**9. Resultados Esperados**

* MVP completamente funcional que integre al menos siete herramientas.
* Creación de proyectos iniciales en menos de 30 minutos.
* Satisfacción del usuario ≥ 85% en pruebas piloto.
* Éxito ≥ 90% en pipelines CI/CD.
* Manual de usuario y documentación técnica entregados.
* Validación académica y demostración ante comité o público objetivo.

**10. Metodología de Implementación**

El proyecto adoptará una metodología ágil incremental (Scrum), con entregas parciales semanales que permitan verificar avances funcionales.

**Fases de desarrollo:**

1. **Sprint 1:** Planificación y análisis del sistema (1 semana).
2. **Sprint 2:** Diseño de interfaces y creación del monorepo base.
3. **Sprint 3:** Pruebas funcionales, integración y configuración de CI/CD.
4. **Sprint 4:** Monitoreo, validación final y presentación del MVP.

**Documentos clave:**

* **Documento de Visión** (definición del proyecto).
* **SRS (Software Requirements Specification)**.
* **SAD (Software Architecture Document)**.

**11. Actores Claves**

* **Estudiantes:** Usuarios principales que aplican y prueban la metodología completa.
* **Docentes:** Supervisores del proceso y validadores de resultados académicos.
* **Equipo de Desarrollo:** Encargado de la implementación técnica, integración y documentación.
* **Startups / Empresas piloto:** Colaboradores externos que prueban la aplicabilidad real.
* **Profesionales DevOps:** Especialistas que configuran pipelines y monitoreo.

**12. Papeles y Responsabilidades**

| **Actor** | **Rol** | **Responsabilidad Principal** |
| --- | --- | --- |
| **Docente** | Aprobador | Validar entregables y cumplimiento de objetivos. |
| **Equipo Dev** | Responsable | Implementar artefactos técnicos y documentar. |
| **Estudiantes** | Ejecutores | Aplicar la metodología, realizar pruebas y reportes. |
| **Empresa Piloto** | Consultado | Brindar retroalimentación práctica y técnica. |
| **Coordinador académico** | Informado | Supervisar la correcta gestión del proyecto. |

**13. Plan de Monitoreo y Evaluación**

| **Indicador** | **Meta** | **Instrumento / Evidencia** |
| --- | --- | --- |
| Accesibilidad | Proyecto inicial en < 30 min | Cronometría de pruebas piloto |
| Usabilidad | ≥ 85% satisfacción | Encuesta de usuario |
| Automatización | ≥ 90% éxito en CI/CD | Logs de Jenkins/Actions |
| Cobertura | ≥ 7 herramientas integradas | Checklist técnico |
| Confiabilidad | 0 fallos críticos en demo | Registro de pruebas |
| Impacto académico | Adopción por docentes y estudiantes | Encuestas y reportes |

**14. Cronograma del Proyecto (MVP – 4 Semanas)**

| **Semana** | **Actividad** | **Entregables** |
| --- | --- | --- |
| 1 | Planificación, análisis y diseño UML | Documento de visión, diagramas UML |
| 2 | Diseño visual y codificación base | Prototipos Figma, monorepo inicial |
| 3 | Pruebas, CI/CD y control de calidad | Pipelines, SonarQube y reportes |
| 4 | Monitoreo y demo final | Dashboard Datadog, documentación y demo |

**15. Hitos de Entregables**

1. Documento de visión y SRS aprobados.
2. Monorepo funcional con integración backend/frontend.
3. Pipelines CI/CD operativos y pruebas automatizadas.
4. Dashboards y alertas configurados.
5. Presentación final del MVP y manual de usuario.

**II. Presupuesto**

**1. Planteamiento de Aplicación del Presupuesto**

El presupuesto está orientado a cubrir los costos operativos y humanos necesarios para el desarrollo, pruebas y documentación del MVP. Se prioriza el uso de software libre, licencias educativas y recursos propios, garantizando un costo mínimo y una alta rentabilidad académica y técnica.

**2. Presupuesto Detallado**

| **Concepto** | **Descripción** | **Costo (S/.)** |
| --- | --- | --- |
| **Costos generales** | Materiales de apoyo, documentación, manuales impresos | 60.00 |
| **Costos operativos** | Consumo eléctrico, pruebas, hosting temporal, herramientas auxiliares | 92.00 |
| **Costos de personal** | Trabajo técnico del equipo de desarrollo (estimado 160 horas) | 4,229.90 |
| **Total estimado** |  | **4,381.00** |

**3. Análisis de Factibilidad**

| **Tipo** | **Descripción** | **Resultado** |
| --- | --- | --- |
| **Técnica** | Factible gracias al uso de herramientas modernas (Node.js, React, Docker). | Viable |
| **Económica** | Bajo costo operativo y uso de licencias educativas gratuitas. | Rentable |
| **Operativa** | Implementación viable en entornos académicos y profesionales. | Funcional |
| **Social** | Contribuye al aprendizaje y desarrollo de competencias digitales. | Positivo |
| **Legal** | Cumple la Ley N° 29733 y N° 30096; no maneja datos personales. | Conforme |
| **Ambiental** | Sin impacto ambiental directo; proyecto 100% digital. | Sostenible |

**4. Evaluación Financiera**

El costo total de S/. 4,381.00 se considera bajo frente al valor generado por el proyecto.  
 Beneficios medibles:

* Ahorro de entre 2 a 3 semanas en configuración inicial de proyectos.
* Incremento estimado de 30% en productividad en equipos de desarrollo.
* Alta replicabilidad en contextos educativos y empresariales.
* Retorno académico inmediato mediante la adopción de metodologías integradas.

El impacto financiero y educativo conjunto posiciona a DevFlow como una inversión estratégica de bajo costo y alto valor agregado.