****

**UNIVERSIDAD PRIVADA DE TACNA**

**FACULTAD DE INGENIERIA**

**Escuela Profesional de Ingeniería de Sistemas**

**Informe Final**

**“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*

| CONTROL DE VERSIONES | | | | | |
| --- | --- | --- | --- | --- | --- |
| Versión | Hecha por | Revisada por | Aprobada por | Fecha | Motivo |
| 1.0 | AHAO, DJAP, WESJ | PCQ | - | 21/09/25 | Versión 1.0 |

INDICE GENERAL

1. Antecedentes 1
2. Planteamiento del Problema 4
   1. Problema
   2. Justificación
   3. Alcance
3. Objetivos 6
4. Marco Teórico
5. Desarrollo de la Solución 9
   1. Análisis de Factibilidad (técnico, económica, operativa, social, legal, ambiental)
   2. Tecnología de Desarrollo
   3. Metodología de implementación

(Documento de VISION, SRS, SAD)

1. Presupuesto 8
2. Conclusiones 8

**Proyecto DevFlow**

**1. Antecedentes**

En el contexto actual del desarrollo de software, tanto en entornos académicos como profesionales, existe una marcada fragmentación entre las herramientas utilizadas en las diferentes fases del ciclo de vida del software. Las metodologías modernas —como Scrum, DevOps y marcos de calidad inspirados en ISO 9001 e ISO/IEC 27001— promueven la integración continua, la automatización y la trazabilidad; sin embargo, su aplicación práctica suele requerir configuraciones manuales y dispersas.

En el ámbito universitario, esta fragmentación genera una enseñanza parcial del proceso de desarrollo, limitando la comprensión integral del ciclo de vida. Por otro lado, en el entorno profesional, los equipos invierten tiempo y recursos configurando tableros, repositorios y pipelines, afectando la productividad y la estandarización de procesos.

DevFlow surge como respuesta a estas limitaciones, con el propósito de ofrecer un ecosistema unificado y modular que permita recorrer de manera ordenada todas las fases del desarrollo de software: desde la planificación hasta el monitoreo, integrando herramientas reales como Trello, PlantUML, Figma, GitHub Actions, Jenkins, Datadog y New Relic.

**2. Planteamiento del Problema**

**2.1 Problema Central**

El desarrollo de software moderno enfrenta una problemática constante: la **fragmentación de herramientas** y la **ausencia de un marco unificado** que permita ejecutar de manera coherente y trazable todas las fases del ciclo de vida.  
 Esta situación impacta negativamente tanto en el aprendizaje de los futuros ingenieros de software como en la eficiencia de los equipos profesionales.

* **En el ámbito académico**, los estudiantes y docentes no cuentan con un entorno integrado que permita visualizar y aplicar de manera práctica todas las fases del desarrollo.
* **En el ámbito profesional**, startups y empresas invierten semanas configurando entornos, pipelines y sistemas de monitoreo, lo que retrasa la entrega de valor y genera sobrecostos.

**2.2 Justificación**

DevFlow propone una solución integral que automatiza e integra las fases del ciclo de vida del software mediante un enfoque metodológico y práctico. Su justificación se fundamenta en los siguientes aspectos:

* **Educativo:** Promueve un aprendizaje basado en proyectos reales, conectando teoría y práctica.
* **Profesional:** Reduce tiempos de configuración y mejora la calidad de los procesos mediante flujos estandarizados.
* **Económico:** Disminuye los costos de implementación inicial y permite reutilizar configuraciones y artefactos.
* **Tecnológico:** Facilita la adopción de buenas prácticas de DevOps, CI/CD y control de calidad automatizado.
* **Social:** Fomenta la formación de profesionales más preparados para los desafíos reales del mercado laboral.

**2.3 Alcance**

El **MVP (Producto Mínimo Viable)** de DevFlow abarca la integración funcional de las fases esenciales del ciclo de vida del software:

* **Planificación:** Tablero en Trello con backlog y estados de avance.
* **Análisis:** Diagramas UML en formato .puml (clases, secuencia y componentes).
* **Diseño:** Wireframes y prototipos en Figma/xPRE con tokens reutilizables.
* **Codificación:** Monorepo con scaffolds para backend (Node.js), frontend (React), móvil (React Native) y desktop (Electron).
* **Pruebas:** Colecciones Postman y SonarQube para análisis de calidad.
* **Despliegue:** Dockerfile, docker-compose.yml, workflows de GitHub Actions y Jenkinsfile declarativo.
* **Monitoreo:** Dashboards básicos en Datadog y alertas en New Relic.

Quedan fuera del MVP (pero previstos para futuras fases):

* Escalado automático en la nube.
* Analítica predictiva de métricas de calidad.
* Reportes personalizados.
* Gestión multiusuario con roles.

3. Objetivos

3.1 Objetivo General

Desarrollar un ecosistema modular y escalable que integre todas las fases del ciclo de vida del software, facilitando su enseñanza, aprendizaje y aplicación profesional.

3.2 Objetivos Específicos

* Implementar un flujo de trabajo unificado con herramientas estándar de la industria.
* Demostrar la trazabilidad completa desde la planificación hasta el monitoreo.
* Reducir los tiempos de configuración inicial de proyectos.
* Fomentar la aplicación práctica de metodologías ágiles y DevOps.
* Ofrecer entregables exportables (diagramas, tableros, reportes y dashboards).
* Validar el MVP mediante indicadores de accesibilidad, usabilidad y automatización.

4. Marco Teórico

El proyecto se sustenta en fundamentos de **Ingeniería de Software**, **Metodologías Ágiles** y **DevOps**:

* **Ingeniería de Software:** Ciencia que estudia los métodos y herramientas para el desarrollo eficiente de sistemas de calidad.
* **Ciclo de vida del software:** Conjunto de fases interrelacionadas (planificación, análisis, diseño, codificación, pruebas, despliegue y mantenimiento).
* **Scrum/Kanban:** Marcos ágiles que promueven iteraciones cortas, mejora continua y colaboración.
* **DevOps:** Cultura y conjunto de prácticas que integran desarrollo (Dev) y operaciones (Ops) para lograr entregas continuas.
* **CI/CD (Integración y Despliegue Continuos):** Prácticas automatizadas que permiten construir, probar y desplegar software de manera confiable.
* **Calidad del Software:** Inspirada en ISO 9001 e ISO/IEC 27001, que promueven la estandarización y seguridad de la información.

**5. Desarrollo de la Solución**

**5.1 Análisis de Factibilidad**

| **Tipo** | **Descripción** |
| --- | --- |
| **Técnica** | Factible mediante tecnologías modernas (React, Node.js, Docker, Jenkins, Datadog). |
| **Económica** | Bajo costo inicial, ya que usa herramientas con planes gratuitos o académicos. |
| **Operativa** | Los usuarios pueden ejecutar el flujo completo con conocimientos básicos de desarrollo. |
| **Social** | Impacto positivo en formación profesional y en la productividad de equipos de desarrollo. |
| **Legal** | Cumple la Ley N° 29733 (Protección de Datos Personales) y N° 30096 (Delitos Informáticos). |
| **Ambiental** | No genera residuos físicos ni impacto ambiental directo, al ser una solución digital. |

**5.2 Tecnología de Desarrollo**

* **Frontend:** React / React Native / Electron
* **Backend:** Node.js + Express
* **Base de datos:** MySQL / SQL Server
* **Integración continua:** GitHub Actions / Jenkins
* **Monitoreo:** Datadog / New Relic
* **Diseño:** Figma / xPRE
* **Documentación:** PlantUML, Trello, SonarQube, Postman

**5.3 Metodología de Implementación**

Se aplicará una metodología **ágil incremental** con entregas en sprints semanales.  
 Los documentos base del proyecto serán:

* **Documento de Visión:** Define objetivos, alcance y justificación (actual documento).
* **SRS (Software Requirements Specification):** Especificaciones funcionales y no funcionales.
* **SAD (Software Architecture Document):** Diseño arquitectónico del sistema y sus componentes.

**6. Cronograma**

| **Fase** | **Actividades principales** | **Duración** |
| --- | --- | --- |
| Semana 1 | Planificación y análisis (tablero, UML) | 1 semana |
| Semana 2 | Diseño y codificación (monorepo, scaffolds) | 1 semana |
| Semana 3 | Pruebas e integración (Postman, Sonar, CI/CD) | 1 semana |
| Semana 4 | Monitoreo, demo y documentación final | 1 semana |

**7. Presupuesto**

| **Concepto** | **Costo (S/.)** |
| --- | --- |
| Costos generales (papelería, documentación) | 60.00 |
| Costos operativos (energía, hosting, pruebas) | 92.00 |
| Costos de personal (equipo de desarrollo) | 4,229.90 |
| **Total estimado** | **4,381.00** |

**8. Conclusiones**

* DevFlow representa una propuesta innovadora que unifica las fases del desarrollo de software en un solo flujo modular, aplicable tanto en contextos educativos como profesionales. Su enfoque metodológico y su integración con herramientas reales lo convierten en un puente entre la teoría académica y la práctica laboral.
* El MVP demuestra la viabilidad técnica, económica y operativa del proyecto, ofreciendo una experiencia práctica completa del ciclo de vida del software. Con ello, DevFlow se proyecta como una herramienta con potencial para convertirse en un estándar de referencia en la enseñanza y ejecución de proyectos tecnológicos.