**Análisis Sugerido para el kata de líder Tecnico**

**1. Necesidades Identificadas**

Las principales necesidades del proyecto en Sofka giran en torno a la falta de experiencia en plataformas de streaming, desalineamientos en el equipo y la ausencia de procesos consolidados, lo que impacta en la entrega del MVP.

* **Falta de experiencia en el dominio**: Es esencial que el equipo, tanto del cliente como los Sofkianos, adquieran conocimientos sobre la industria de plataformas de streaming.
* **Inmadurez en procesos**: La falta de pipelines de CI/CD y prácticas DevOps eficaces retrasa el ciclo de desarrollo.
* **Desalineamiento en los equipos**: Hay conflictos de visión y una falta de colaboración fluida entre las células.
* **Presión de tiempos**: El cliente ha fijado un plazo ajustado para el MVP, generando urgencia en las entregas.
* **Falta de liderazgo técnico**: No todas las células cuentan con un líder técnico asignado.
* **Automatización de pruebas limitada**: Las pruebas manuales incrementan el tiempo necesario para regresiones, afectando la calidad y los tiempos de entrega.

**2. Soluciones Propuestas e Implementación**

El objetivo es resolver los problemas de procesos, desalineamientos y acelerar la adopción de nuevas prácticas, todo mientras se reduce la curva de aprendizaje sobre el dominio.

**Soluciones para la Falta de Experiencia y Alineación del Equipo**

* **Workshops intensivos de dominio**: Organizar talleres con expertos en plataformas de streaming para capacitar tanto al equipo interno como al cliente, asegurando que todos los participantes comprendan las necesidades del negocio.
* **Mentoría técnica**: Asignar mentores senior para guiar a los miembros menos experimentados y ayudarles a asimilar rápidamente tanto el dominio como las mejores prácticas técnicas.
* **Comunicación abierta**: Fomentar una colaboración cercana entre células mediante herramientas como Slack, Jira y Confluence. Esto incluye la implementación de reuniones de sincronización y planificación para asegurar que todos los equipos sigan la misma visión.

**Establecimiento de Prácticas DevOps**

* **Pipelines de CI/CD**: Implementar gradualmente pipelines utilizando herramientas como Jenkins, GitLab CI/CD o GitHub Actions, AzureDevOps. Esto garantizará un flujo de desarrollo más ágil y confiable.
* **Automatización de pruebas**: Incluir frameworks como Selenium o Cypress para acelerar las pruebas de regresión y QA. Además, implementar pruebas unitarias y de integración desde las primeras fases del desarrollo.
* **Monitorización continua**: Utilizar herramientas como Prometheus o Grafana para monitorizar tanto el rendimiento como la estabilidad del sistema en tiempo real, lo que permitirá resolver problemas de manera proactiva.

**Facilitar la Colaboración y Resolver Conflictos**

* **Definir roles y responsabilidades claras**: Cada célula debe tener bien definidas sus responsabilidades, y los líderes técnicos deben encargarse de tomar decisiones rápidas.
* **Rotación de roles**: Permitir que los miembros del equipo roten entre diferentes roles para mejorar la colaboración interdepartamental y aumentar la empatía entre equipos.
* **Mediación y resolución de conflictos**: Implementar sesiones de mediación para resolver conflictos de visión y liderazgo dentro de las células.

**3. Planificación de Tiempo**

Se propone un plazo de 2-3 meses para implementar las soluciones mencionadas, dado que involucran tanto cambios técnicos (CI/CD, automatización de pruebas) como de cultura organizacional (mentorías, talleres, alineación de equipos). Sin embargo, es necesario seguir un enfoque iterativo que permita entregar valor continuamente mientras las prácticas se consolidan.

**4. Matriz de Riesgos**

| **Riesgo** | **Probabilidad** | **Impacto** | **Acción de Mitigación** |
| --- | --- | --- | --- |
| Falta de experiencia en el dominio | Alta | Alto | Implementar workshops especializados y mentoría técnica para acelerar la curva de aprendizaje del equipo. |
| Retraso en la implementación de QA | Media | Alto | Introducir herramientas de automatización como Selenium o Cypress y capacitar al equipo de QA. |
| Falta de pipelines de CI/CD | Alta | Alto | Establecer pipelines CI/CD utilizando Jenkins, GitLab CI, o GitHub Actions y fomentar prácticas DevOps. |
| Conflictos en los equipos | Media | Medio | Facilitar actividades de team building, mejorar la comunicación efectiva, y mediar en conflictos internos. |
| Selección de proveedor de nube | Media | Medio | Evaluar las opciones con el cliente, considerando factores técnicos y financieros para seleccionar el mejor proveedor. |
| Escasez de recursos humanos | Media | Alto | Priorizar tareas críticas, redistribuir responsabilidades y considerar contrataciones temporales si es necesario. |
| Retrasos en la entrega del MVP | Media | Alto | Establecer hitos claros con monitoreo constante, ajustes al alcance del proyecto y gestión ágil del backlog. |
| Problemas de calidad del software | Media | Alto | Implementar pruebas exhaustivas, revisiones de código, análisis estático, y seguimiento de bugs en tiempo real. |
| Desalineación de expectativas del cliente | Media | Alto | Establecer canales de comunicación claros, gestionar las expectativas con entregas incrementales y reportes frecuentes. |
| Falta de adopción de nuevas prácticas | Media | Medio | Proporcionar capacitación continua, recompensar el éxito en la adopción de nuevas prácticas y reforzar su importancia. |

**5. Plan de Aprendizaje y Crecimiento**

* **Certificaciones y cursos online**: Proporcionar acceso a plataformas como Coursera, Udemy y Pluralsight, incentivando la obtención de certificaciones en tecnologías relevantes.
* **Mentoría continua**: Asegurar que todos los desarrolladores tengan acceso a un mentor experimentado que les ayude a superar obstáculos técnicos y de negocio.
* **Rotación de roles**: Animar a los miembros del equipo a experimentar con diferentes roles para ampliar sus habilidades y mejorar la colaboración.

**6. Enfoque DevOps**

* **Implementar CI/CD**: Configuración de pipelines que incluyan construcción, pruebas y despliegue automatizados.
* **Monitorización en la nube**: Migrar la aplicación a un entorno cloud y utilizar herramientas de monitoreo como Prometheus o Grafana para identificar problemas tempranamente.

**7. Gestión del Backlog y Productividad**

* **Priorización colaborativa**: Trabajar con el cliente para definir y priorizar las funcionalidades clave del MVP. Esto garantiza que los recursos se destinen a las características que agregan mayor valor.
* **KPIs y retrospectivas**: Definir KPIs claros que midan el rendimiento del equipo, calidad del software y velocidad de entrega. Llevar a cabo retrospectivas periódicas para identificar mejoras continuas.

**8. Comunicación con el Cliente**

* **Transparencia**: Mantener una comunicación continua y abierta con el cliente, ajustando expectativas cuando sea necesario. Informar sobre el progreso de manera regular, estableciendo una relación de confianza.

**Conclusión**

Este plan proporciona una guía práctica y eficiente para abordar las principales áreas problemáticas del proyecto, con soluciones bien alineadas con las necesidades identificadas. La clave es la adopción iterativa de prácticas DevOps, una mejor alineación del equipo mediante mentoría y workshops, y una comunicación clara con el cliente para gestionar expectativas.

**Ing. Mildred. M. Moreno L.**