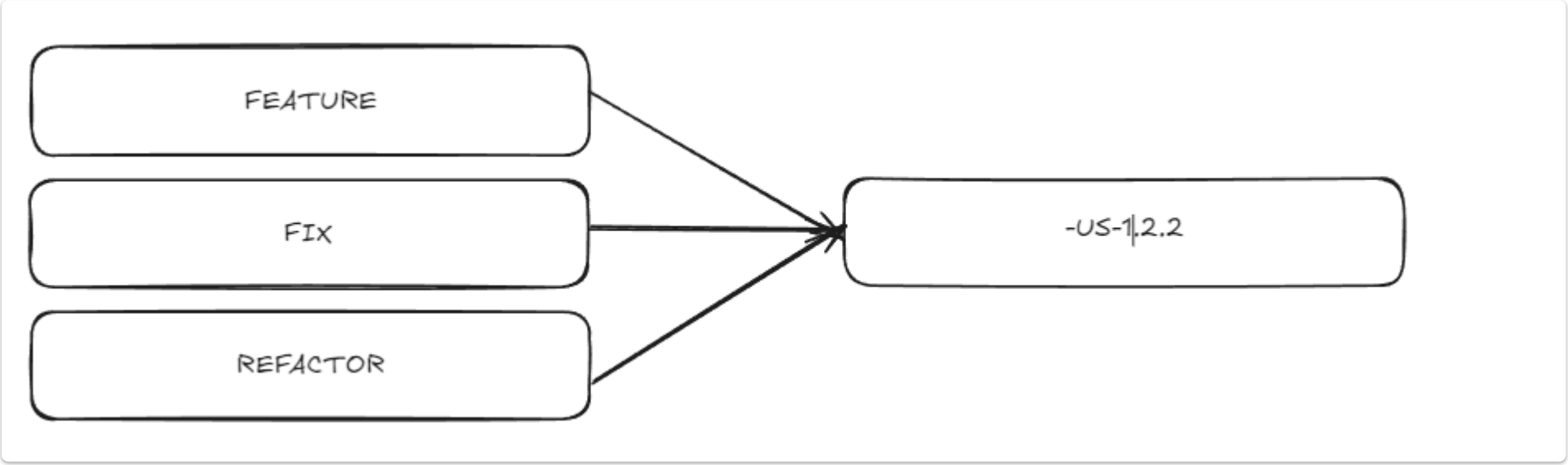
Las Mejores Prácticas de Desarrollo

Adoptar hábitos de trabajo y buenas practicas que garanticen la calidad del software y la eficiencia de todo el equipo durante el aprendizaje es muy importante ya que al momento de incorporarnos a un empleo tendremos mayor capacidad de adaptabilidad. Una comunicación constante y transparente resulta fundamental para alinear expectativas y detectar tempranamente cualquier problema, mantener un diálogo regular, tanto en reuniones cortas de sincronización como mediante herramientas de mensajería o wikis compartidos, facilita la resolución de dudas y el intercambio de aprendizajes.

En cuanto al control de versiones, utilizar Git de manera rigurosa se convirtió en una parte fundamental de nuestro trabajo. Definir ramas dedicadas a funcionalidades específicas y respetar un esquema claro de nombres, actualmente en el capstone utilizamos el siguiente



primero de que se trata la rama si es agregar nueva funcionalidad es un Feature, si es necesario arreglar un bug es un Fix y si es necesario hacer cambios a gran escala por algún motivo es un Refactor, esto contribuye a evitar confusiones y a mantener la estabilidad de la base de código. A su vez, los commits pequeños y descriptivos favorecen la trazabilidad de los cambios y simplifican la identificación de errores cuando surgen problemas.

La integración continua configura otro pilar esencial en el proceso de desarrollo. Cada vez que subimos código al repositorio, un pipeline automatizado ejecuta pruebas unitarias y de integración para asegurar que las nuevas contribuciones no rompan lo ya construido. Emplear entornos reproducibles con contenedores, por ejemplo Docker, reduce discrepancias entre el equipo de desarrollo, los servidores de integración y el entorno de producción, lo cual minimiza sorpresas indeseadas a la hora del despliegue.

La calidad del código y su mantenibilidad pasan también por la aplicación de estándares y la refactorización constante. Seguir guías de estilo ayuda a que todos los miembros del equipo lean y comprendan con rapidez lo que otros han escrito. Cuando detectamos fragmentos de código duplicados o excesivamente complejos, dedicamos tiempo a simplificarlos para evitar la acumulación de deuda técnica. Además, elegir nombres claros para variables y métodos disminuye la necesidad de excesivos comentarios, pues el propio código habla por sí mismo.

Mantener una documentación viva, actualizada y accesible a todos los miembros del equipo resulta igualmente importante. Un README detallado con instrucciones de instalación, configuración del entorno y ejecución de tests, junto con especificaciones de las APIs definidas a través de herramientas como OpenAPI, contribuye a que nuevas incorporaciones al proyecto se incorporen rápidamente y sin cuellos de botella. Desde las primeras etapas de diseño, procuramos gestionar secretos y credenciales a través de variables de entorno o servicios especializados, evitando su inclusión en el código. Finalmente, incorporar mecanismos de monitoreo y registro de eventos en los entornos de prueba y producción nos permite conocer de manera proactiva el comportamiento del sistema. Los logs centralizados y las alertas tempranas nos advierten sobre anomalías antes de que los usuarios finales se vean afectados, cerrando así un ciclo de retroalimentación continua que nutre el proceso de mejora del software. Como nombre antes adoptar estas prácticas desde nuestros primeros proyectos académicos establece un hábito profesional que, con el tiempo, se traducirá en software de mayor calidad, menor incidencia de errores y equipos más cohesionados y eficientes.