**Idea de proyecto Luis González**

**Problemática a resolver**

En el ciclo actual de desarrollo de software de un ERP, el despliegue de cambios en ambientes compartidos es realizado manualmente, lo que genera diversos problemas operativos como conflictos entre desarrolladores, sobrescritura accidental de archivos y falta de coordinación en los procesos de reinicio (bounce) de servidores. Estos inconvenientes ocasionan retrasos, errores e ineficiencias en el flujo de trabajo del equipo.

Actualmente, el proceso manual incluye:

* Generación de archivos .jar con cambios por parte del desarrollador.
* Copia manual de los archivos .jar desde el ambiente local del desarrollador al servidor central en los servidores del cluster correspondiente (REST services, UI components, etc.).
* Notificación manual en el canal de Slack sobre el bounce planeado, indicando el tiempo para su inicio (5-15 minutos).
* Realización del bounce si no existe oposición por parte de otros desarrolladores, notificando posteriormente al canal.
* Gestión manual de pausas y reanudaciones del proceso si existe oposición o necesidad de coordinación adicional entre desarrolladores.

Otra problemática es que a veces las personas quieren probar cambios en el mismo .jar lo cual no es posible. Hacer el despliegue de .jars diferentes no afecta.

Los errores comunes que pasan es que:

1. Algunos desarrolladores hacen bounce de los servidores sin notificar en el canal de slack, afectando a otras personas que están trabajando en el ambiente.
2. Los desarrolladores deben estar al pendiente acerca de los bounces que hacen otras personas, revisando la hora y el tiempo en el que dijeron que harían el bounce, el cual no es exacto, a veces lo hacen un poco antes o después.
3. Cuando 2 personas hacen copia de sus .jars al mismo tiempo aproximadamente, solo los últimos cambios se conservan.
4. También alguien podría hacer la copia de sus jars justo antes de que se realice el bounce lanzado por otra persona y así sobreescribir los cambios que quería probar el desarrollador que lanzó el bounce inicialmente.

**Alcance específico del proyecto**

El objetivo es automatizar el proceso mediante un bot integrado en Slack, el cual deberá:

* Identificar automáticamente los archivos (.jar) que serán desplegados.
* Determinar el ambiente específico y los clusters afectados por el cambio.
* Establecer y comunicar tiempos exactos para la ejecución de bounces.
* Enviar avisos previos a la ejecución del bounce y gestionar pausas si hay objeciones del equipo.
* Permitir agregar archivos adicionales al despliegue mientras no existan conflictos.
* Gestionar la reanudación del proceso de bounce una vez alcanzado el acuerdo del equipo.
* Notificar al canal de Slack al completar el proceso de bounce.

Esta automatización garantizará la obligatoriedad de comunicar cambios en Slack, facilitará el seguimiento y coordinación, y eliminará múltiples pasos manuales.

**Tipos de datos involucrados**

* **Archivos de código (.jar)** generados por los desarrolladores.
* **Información del ambiente y clusters afectados** (REST services, UI components, etc.).
* **Mensajes de Slack** (solicitudes de bounce, confirmaciones, acuerdos).
* **Tiempos definidos y modificados** para ejecución de bounces.

**Origen de los datos**

* **Ambiente local del desarrollador**: Archivos .jar.
* **Servidor central (WebLogic)**: Configuración y estado actual de clusters.
* **Canal de Slack específico**: Comunicaciones y acuerdos entre desarrolladores.

**Áreas o departamentos involucrados**

1. Infraestructura para hacer la automatización.
2. Desarrollo para realizar las pruebas del chatbot.

### Modelos de IA recomendados

* **Un modelo gratuito de clasificación de texto:** Para identificar automáticamente qué .jars serán copiados, a qué ambiente se copiarán, qué clusters son afectados y dentro de cuánto tiempo iniciará el bounce basándose en la descripción de la solicitud de bounce.
* El modelo también debe de generar payloads en formato JSON para invocar las APIs requeridas para la automatización. Un buen ejemplo para intentar en primera instancia sería LLaMA 3

**Tecnologías y herramientas requeridas**

* **Slack API** para integración del bot.
* **Python** para desarrollo de scripts y automatización.
* **LangChain/LLM API** para uso del modelo LLM.
* **WebLogic Scripting Tool (WLST)** o APIs para despliegues y gestión del servidor.
* **Servidor o infraestructura Cloud** para alojar y ejecutar el bot.
* **Git** para control de versiones y manejo de archivos.

**Entregables del Proyecto**

1. **Slack Bot funcionando**:

* Capacidad de interpretar comandos en lenguaje natural.
* Automatización completa del proceso de despliegue y notificación

1. **Documentación clara y operativa**:

* Manual de uso para usuarios.

### Tiempos estimados

* Diseño y planificación inicial: 2 semanas (1 persona).
* Desarrollo del Slack Bot con capacidades de IA: 4 semanas (1-2 personas).
* Integración técnica con WebLogic y Slack: 3 semanas (1 persona).
* Fase de pruebas y ajustes finales: 2 semanas (1-2 personas).
* Creación de documentación y capacitación del equipo: 1 semana (1 persona).
* Lanzamiento y monitoreo inicial: 1 semana (1 persona).

Tiempo total estimado: 13 semanas.

### Costos aproximados

* Desarrollo e integración técnica: $5,000 – $10,000 USD.
* Infraestructura tecnológica (cloud hosting): $2,000 – $4,000 USD anual.
* Servicios y APIs de IA (OpenAI): $500 – $1,500 USD anual.
* Documentación, capacitación y soporte inicial: $1,000 – $2,000 USD.

Costo total estimado inicial: $8,500 – $17,500 USD.