CI/CD SisGlobal – Manifiesto

Definiciones:

En términos simples el CI/CD se define como el mecanismo para que las aplicaciones puedan llegar al usuario de manera automatizada y con la menor participación posible del ser humano. Para lograr esto, debe determinarse un flujo de pasos que el equipo de desarrollo debe seguir.

SisGlobal es un sistema que tiene como objetivo automatizar la operatividad de la empresa Acero Global C.A y sus asociadas. Utiliza tecnologías WEB y es soportado completamente por personal interno de la empresa. La filosofía de dicho personal es colocar al sistema SisGobal dentro del flujo de trabajo CI\CD bajo la cultura DEVOPS.

Infraestructura:

Lo que inserta la aplicación de negocios Sisglobal a una cultura DEVOPS es el sistema GITLAB (basado en GIT) y los contenedores con DOCKER.

Sisglobal además, usa una arquitectura de microservicios y una filosofía de desarrollo web backend – frontend, lo que convierte a GITLAB en un recipiente de los repositorios de los proyectos que dan respuestas a los requerimientos de la empresa y sus usuarios.

Cada repositorio de cada proyecto contará con al menos 3 ramas:

- "Master": rama que contiene el código que van a producción y por lo tanto es la versión de la aplicación que se está utilizando en la empresa. Cuenta con la permisología necesaria para ignorar cualquier push directamente a esta rama que no esté autorizado. Su fuente de datos es la base de datos principal de la empresa.

- "QOS": esta versión de la aplicación es utilizada por los usuarios finales para el testeo de las funcionalidades nuevas del sistema y tienen un tiempo de vigencia en las que se realizarán las observaciones pertinentes. Cuenta con la permisología necesaria para ignorar cualquier push directamente a esta rama que no esté autorizado. Su fuente de datos es la base de datos principal de la empresa.

- "Desarrollo": versión o rama del repositorio en donde se integran todas las funcionalidades desarrolladas por los programadores cuyo objetivo principal es hacer las pruebas en campo de la integración de toda la aplicación con el módulo o funcionalidad específica. Su fuente de datos en una copia en estructura de la base de datos principal pero con datos sin relevancia.

- Otras ramas: el desarrollador no está limitado a la cantidad de ramas creadas en su repositorio local y puede "subir" algunas de estas ramas al

repositorio remoto con la intención de compartir con los otros miembros cualquier innovación que considere importante.

Las ramas de los repositorios tienen una estructura de archivos y carpetas descritas de la siguiente forma:

- "app/": contiene todo el código que el desarrollador y el equipo de trabajo utiliza
- "logs/": todos los logs propios del contenedor en donde corre la aplicación.
- "Dockerfile": descripción de las imágenes y comandos necesarios para que el contenedor exista.
- "docker-compose.yml": alberga la descripción de la ejecución de todos los contenedores y su relación, necesarios para el funcionamiento del servicio o aplicación. Este es el archivo a ejecutar con *docker-compose – d up.*
- .gitignore: en este archivo se colocan todos los archivos y carpetas a ignorar por el sistema GIT.
- "Readme.md": archivo de texto en formato marckdown que describe la documentación de la aplicación y otras informaciones de importancia.

Proceso de Trabajo:

1. El desarrollador debe partir de una copia local del repositorio(s) que contengan el proyecto(s).
2. Teniendo una copia del repositorio, creará una nueva rama local con un nombre descriptivo a la funcionalidad o módulo requerido y en donde desarrollará dichas funciones y pruebas requeridas teniendo como fuente de datos la base de datos designada para pruebas.
3. Después de culminar el desarrollo, cambios y pruebas locales, el desarrollador pasa a la rama "desarrollo" dentro de su repositorio local y hace *git merge* con la rama de la *feature* desarrollada. Luego hace push desde rama desarrollo al repositorio remoto.
4. El servidor CI/CD realizará la las pruebas, build e integración automáticamente. El desarrollador debe informar de las condiciones necesarias adicionales (scritps, tablas, etc…) para que lo desarrollado pase las pruebas necesarias y se logre la integración con el resto de la aplicación.
5. Una vez superadas las pruebas unitarias y de integración el líder técnico realizará pruebas humanas del módulo y hará una revisión del código, para dar su aprobación a la integración del módulo o funcionalidades a la

rama QOS. Si el líder técnico tiene observaciones no procede la integración a QOS y hace las indicaciones al desarrollador. El desarrollador vuelve al paso 3.

6. Las funcionalidades de QOS aprobadas por el usuario final son responsabilidad del líder técnico integrarlas a la rama master. Si el usuario informa de una observación de la funcionalidad o funcionalidades en el periodo de pruebas, el líder técnico informará al desarrollador y este volverá al paso 3.

Reglas y Consideraciones Importante:

1. El frontend principal está desarrollado en Angular y un paquete de componentes UI llamado PrimeNG. Priorizar el desarrollo en este framework.

2. El backend está desarrollado con las tecnologías RESTFUL. Priorizar el uso de esta tecnología.

3. El uso de variables de entorno, POO, JSON y JWT son de carácter obligatorio para el desarrollo de cualquier servicio o aplicación.

4. Priorizar el uso de tecnologías programación reactiva y funcional.

5. Priorizar el uso de patrones de diseño y arquitectura como MVC, CQRS, DDD, etc.

6. El desarrollado en lenguajes, framework o plataformas no utilizados en la actualidad por el equipo de desarrollo puede ser aplicado siempre que cumpla con lo siguiente:

    a. La notificación y el consentimiento de los demás integrantes del equipo para garantizar la escalabilidad y mantenibilidad de las aplicaciones.

    b. Estándares internacionales como los emanados por W3C, ECMA International, entre otros.

    c. Sean bajo la filosofía Backend – Frontend.