TEC | Tecnológico de Costa Rica

Área Académica de Ingeniería en Computadores

SOA41D: Arquitectura Orientada a Sistemas Emergentes

Investigación I

Branching strategies - Git flow

Estudiantes:

Luis Daniel Prieto Sibaja

Profesor:

Alejandra Bolaños Murillo

Fecha de Entrega:

1/3/2022

I Semestre – 2022

Investigación

En la actualidad, cuando se trata de la realización de proyectos de desarrollo, mantenimiento y actualización de software, se debe tomar en cuenta el uso de una estrategia de control de versiones para tener un buen manejo del flujo de trabajo del proyecto, así como buenos resultados. Al trabajar con la herramienta Github, hay *branching strategies* que se pueden utilizar para manejar el desarrollo, control e integración del código. En Git, las ramas son simplemente punteros los cuales Git puede usar para llevar registro de los cambios que se realicen, una rama es simplemente otro nodo en el grafo del proyecto, por lo que aplicar una branching strategy es sencillo. Uno de los modelos más exitosos es el Git Flow. Este se basa esencialmente en utilizar la rama principal como un historial de todos los cambios, y la rama del desarrollador como rama de integración para todos los desarrolladores.

Otros métodos utilizados para modular las etapas de un proyecto son los conceptos del poly-repo y mono-repo. Un mono-repo es un repositorio simple que alberga todo el código de la aplicación, por lo que librerías y microservicios coexisten como archivos externos al proyecto principal. En el caso de un poly-repo, cada microservicio, interfaz de usuario, librería externa y demás utiliza su propio repositorio. No hay una decisión definitiva sobre cuál esquema de repositorio es mejor, si bien el mono-repo centraliza el proyecto y el poly-repo lo modula, depende del proyecto en cuestión que se quiera trabajar. Similar a los poly-repos, continuando la idea de repositorios externos, existen los *git submodules*, los cuales son proyectos externos a uno principal, los cuales se mantienen separados del mismo (librerías externas de terceros por ejemplo), mientras que al mismo tiempo se tratan como un subdirectorio del mismo repositorio. De esta forma, un submodule es esencialmente un “proyecto dentro de un proyecto”.

Bajo esa misma línea, podemos explorar dos conceptos relacionados al de submodule. Estos conceptos son conocidos como *upstream* y *remote*. El primero se refiere al repositorio o rama “padre” de un submodule, o de un repositorio forkeado, siendo este del que derivan los demás. El segundo concepto es similar al primero, puesto que se refiere a la instancia de un repositorio ubicada en la plataforma del control de versiones, paralela a la copia en la máquina local que fue clonada por el desarrollador. En el caso de los upstreams y los remotes, se introduce el concepto del *pull request*, el cual es simplemente un *push* al repositorio remoto si este pertenece a otro autor u otra compañía.

Referencias

1) LaunchDarkly. (2014). Git Branching Strategies vs Trunk-Based Development. Recuperado de: https://launchdarkly.com/blog/git-branching-strategies-vs-trunk-based-development/
2) Visolyaputra, N. (2021). How to choose bewteen mono-repo and poly-repo. Recuperado de: https://www.accenture.com/us-en/blogs/software-engineering-blog/how-to-choose-between-mono-repo-and-poly-repo
3) Git Tools - Submodules. Recuperado de: https://git-scm.com/book/en/v2/Git-Tools-Submodules