**Nome**: Nícolas Nishiyama Soares
**Matrícula**: 2023006813

1. DevOps é o conceito de engenharia de software que prevê a integração entre desenvolvedores e operacionais de TI, com objetivo de evitar conflitos e integrar os dois grupos.

2. Concordo, pois um engenheiro DevOps deve ter conhecimento de todo o processo de produção de um software, para que possa integrar os dois grupos citados acima. É importante que ele tenha esse domínio, pois caso surja alguma dúvida, ou considerando a sua participação no desenvolvimento, ele deverá ser capaz de executar sua função.

3. O DVCS é um sistema de controle de versão de código que armazena o código temporariamente no servidor local para depois ser enviado ao global (diferentemente do VCS que apenas armazena no servidor global). Dessa forma, aumenta a eficiência no **tempo dos commits** e o **trabalho offline** auxilia na flexibilidade da codificação.

4. O uso de mono-repositórios prejudica a eficiência no uso do código quando se considera o seu uso. Os mono-repositórios deixam o algoritmo todo em um diretório impedindo a instalação de partes e dificultando o uso da CI.

5. A integração contínua (*continuous integration*) é a ideia de dividir o problema e solucioná-lo em partes por meio de uma série de commits. A entrega contínua (continuous delivery) é uma mudança que, no momento de implementação, necessita de aprovação humana para ser posta em produção. Já o deployment contínuo (continuous deployment) é a implementação de uma funcionalidade sem intervenção humana.

6. Eles são conceitos importantes, pois são abordados tanto pelos desenvolvedores como pelos operacionais de TI desde os commits até a implementação do sistema.

7. O teatro da integração contínua é o uso da CI de forma incorreta, dando a ilusão de uma agilidade e eficiência na produção, entretanto falhas no uso desta prática ágil impedem sua boa execução. A não automatização de testes ou ocultação de bugs nas compilações ou até a baixa frequência de commits prejudicam o sistema e o uso da CI.

Baseado no artigo "*Continuous integration theater.*" de FELIDRÉ et al. (2019).

8. Eu iria optar pelo delivery contínuo, pois por meio dele haveria a necessidade de uma confirmação humana no momento de implementação dos commits. Dessa

forma, o processo não estaria completamente automatizado evitando a passada de bugs e erros.

9. Atrasa a visualização do projeto como um todo, pois o uso da feature flag desabilita trechos do código para mudanças até que sejam corrigidas e habilitadas novamente.

10. No caso do *ifdef* e do *endif* eles necessitam da declaração de uma macro para caso ela possua um valor seja realizado o bloco de código que estiver entre os indicadores. Já na feature flag, é declarada uma variável com valor binário e por meio dela é feita uma estrutura de condição para habilitar ou não o código.

11. A business flags, pois a sua aplicação é para um grupo específico, podendo permanecer no código, enquanto a release flags são destinada a apenas um momento do processo de produção de software.

12. As *branches* armazenam o histórico dos códigos em momentos específicos da empresa, quando vai fazer alguma atualização, por exemplo.

13. a) A atualização feita para uma população específica deve ter utilizado o conceito de business flags chamando de conditional features.
b) Feature Flags

**Artigo utilizado na 7° questão**

FELIDRÉ, Wagner et al. Continuous integration theater. In: **2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)**. IEEE, 2019. p. 1-10.