Equipe
Anielle Cassiano
Julio Jacob
Evelyn Basso
Daniele Seixas
Gabriela Oliveira

Manual de Padronização

Dados da empresa

Nome da empresa: Linkedln Corporation

Descrição da empresa: O linkedln é uma rede social voltada para o ambiente corporativo e estudantil, nesta rede social é possível criar conexões profissionais, ter acesso a cursos, realizar campanhas de recrutamento e encontrar empregos, possibilitando dessa forma o desenvolvimento profissional e comercial de seus usuários.

Tipo de software

1. Plataforma web
2. Aplicativo mobile (compatíveis com Android e IOS)

Infraestrutura

As infraestruturas utilizadas, são:

- Plublic Cloud - Maior parte consumida pela aplicação, aplicada principalmente na parte da rede social da aplicação;
- On Premise - Menor parte utilizada pela aplicação, utilizada na parte que necessita de maior segurança.

Padrão de código

Alguns padrões a serem seguidos na escrita do código:

- Em caso de comentários de apenas uma linha utilizar o // (barra dupla)
- Comentários com mais de uma linha utilizar {}
- O código escrito deverá ser todo Indentação

- Em caso de palavras compostas deve ser utilizado o padrão CamelCase (onde a primeira letra da palavra deve ser Maiúsculas e não poderá ter espaço entre as palavras compostas)
- Palavras não compostas devem ter a primeira letra maiúsculas.

Repositório

- Git
- Link do repositório: https://github.com/juliojacob1979/treinamento-devops

Branch

Para a nomenclatura das branch utilizar o padrão abaixo citado:

- Type ou categoria da mesma. Ex: docs, feat, fix...;
- Descrição breve do que a mesma faz;
- Número da demanda que originou a criação da branch.

Exemplo: Para uma branch que cria a funcionalidade de cadastro de profissionais, a branch ficaria da seguinte forma:

- feat_CadastrarDadosProfissionais_#123

Issues

As issues devem ser criadas no git e devem ser vinculadas ao seu respectivo projeto.

Para a criação das issues devem ser adotados as seguintes regras:

- Issue de bug: deverá conter uma descrição do problema encontrado e evidências dos mesmos
- Issue de feat: deve conter a estórias de usuário e os critérios de aceite
- Issue de melhorias: deve conter uma descrição contendo, qual melhoria deve ser aplicada, local da aplicação, causa que originou a melhoria e evidencia (opcional)
- Issue de tarefas: deve conter uma descrição da tarefa a ser executado e evidencia (opcional)

Para a nomenclatura das issues deve ser utilizado o seguinte padrão:

- Nº da demanda\projeto - Descrição breve do assunto tratado

  Exemplo: Projeto 123456 - Erro ao cadastrar postagem com imagens

Identificação do tipo de issue aberta:

Todas a issue abertas devem conter Tag que especifique qual o seu tipo, as quais são:

- Bug - para erro, defeitos e falhas encontradas no produto;
- Feat - novas funcionalidades a serem desenvolvidas (estórias de usuário);
- Improvement - para melhorias do sistema (Performance, layout, ...);
- Task - para situações que não se enquadram nas listadas acima (criação\alteração de documentação e manuais do usuário).

Todas as issue devem conter a versão do sistema a qual elas pertencem.

Testes

Os testes aplicados são:

- Teste unitários: o sistema deverá ter no mínimo 60% do código coberto pelos testes unitários;
- Testes manuais: Os testes devem ser feitos de forma manual, seguindo o seu respectivo planejamento e devem conter evidências de teste (Em caso de sucesso ou falha);
- Teste automatizados: Após feito a validação manual deve ser automatizado as funcionalidades com maior relevância e prioridade para o sistema, fazendo com que essas novas funcionalidades façam parte do teste regressivo do sistema;
- Documentação dos testes: Devem ser construídas a documentação dos testes a serem executados, os casos de testes devem ser construídos utilizando o padrão Gherkin.

Quando executar os testes:

- Testes unitários: esses testes devem ser executados no momento da codificação e antes da funcionalidade ser enviada para teste;
- Teste manual: devem ser executados após a geração da versão para testes, as funcionalidades são testadas de maneira isolada;
- Teste automatizado: devem ser executados depois que uma versão final é criada, e antes que a mesma seja implantada em produção.

Responsável pela especificação e documentação de teste: Analista de Qualidade
Responsável pelos testes unitários: Desenvolvedor
Responsável pela execução testes: Todos da equipe

Deploy

Para realização do deploy iremos utilizar o CI\CD Delivery, o qual deverá passar pelas as seguintes etapas:

- Code review: essa prática deve ser realizada a cada mudança de código e visa aumentar a qualidade do código entregue. E deve seguir as seguintes fases:
    - Espera-se que o code review seja realizado de forma rotativa, onde todos os integrantes do time em algum momento assumam essa tarefa. Esse processo poderá contribuir bastante no amadurecimento do time em caminhar no mesmo "rumo", consequentemente isso refletirá em entregas melhores e com qualidade;
    - Incorporar automatização ao trabalho manual:essa incorporação pode ser feita com a definição de busca automática de algumas falhas já conhecidas ou de padrões que indiquem alguma vulnerabilidade;
    - Usar checklists: estas serão usadas para guiar a revisão do código.
- Deploy em homologação: etapa em que a funcionalidade já passou pelos testes locais de desenvolvimento e pelo code review, quando a funcionalidade estiver pronta para ser testada, deverá ser feito o deploy no ambiente de homologação, esse deploy será feito após sinalização do responsável pela execução dos testes manuais.
- Deploy em homologação (automatizado) - assim que a versão final é gerada, deve ser disparado de maneira automática o evento de execução da automação;
- Deploy em produção - assim que a versão for finalizada, passando pelas etapas citadas anteriormente, será realizado o deploy em produção, para que o mesmo ocorra deve ser verificado automaticamente alguns pontos:
    - A versão deverá ter sido aprovada nas etapas anteriores (Deploy em homologação (manual) e Automatizado)
    - Ter todas as revisões aprovadas
- Por regra do deploy em produção:
    - Deve ser feito ao final de cada sprint (a cada 15 dias);
    - Deve ser toda terça-feira
    - Deve ser no horário de 00:00 até as 05:00hrs da manhã

Disaster Recovery

Cenário 1

- Emergência: Desastres naturais

- Identificação: Alerta de desastres naturais foi emitido nos canais de informação da imprensa para a região onde a empresa fica localizada fisicamente.

- Mitigação: Backup das informações na nuvem em 2 servidores localizados em pontos geográficos diferentes.

Cenário 2

- Emergência: Incêndio

- Identificação: Instalação de chuveiros automáticos sprinklers, as bombas hidráulicas, extintores de incêndio, alarmes

- Mitigação: Backup das informações na nuvem em 2 servidores localizados em pontos geográficos diferentes.

Cenário 3

- Emergência: Roubo de dados

- Identificação: Receber um alerta de Ransomware contendo a mensagem solicitando pagamento para o resgate dos dados.

- Mitigação: Configurar permissão de acesso ao(s) servidor(es), reforçar a segurança da rede colocando validação em duas etapas, construindo uma estrutura de VPC/NAT/Privadas

Cenário 4

- Emergência: Exclusão de contas

- Identificação: Usuário tentou acessar sua conta e foi apresentado a mensagem usuario inválido

- Mitigação: Backup das informações na nuvem em 2 servidores localizados em pontos geográficos diferentes e realizar rollback primeiramente das contas Premium, e em seguida das contas Free.

Cenário 5

- Emergência: Meio de pagamento de cartão de crédito off para cadastro das contas Premium

- Identificação: Usuário clicou em “Assinar” e foi apresentado a mensagem “Não foi possível efetuar a assinatura”

- Mitigação: Sistema automaticamente deve ativar e ofertar as formas de pagamento secundários (Pix (QR Code) e boleto) para ativação da conta com assinatura mensal, após normalização do serviço de cartão de crédito enviar mensagem para o usuário ativado via formas de pagamento secundário solicitando a ativação da assinatura recorrente.

Diagrama Infraestrutura