O pesadelo de qualquer dev: Vamos falar sobre ABEN Ds

Salve jovem padawan, sextou eca estamos iniciando um novo mês e adentrando no último trimestre de 2021, a pandemia continua lá fora, menos tenebrosa, mas assustadoramente real e nos enlouquecendo pouco a pouco.

No artigo de hoje irei falar sobre o pior pesadelo de todo programador, estou falando sobre os ABENDs, términos anormais de nossos programas em produção, momentos de terror e pânico, que dá calafrios na espinha e nos provoca enjoos e tremores. A triste notícia é que um dev sendo dev, ao longo da carreira irá acumular historias tenebrosos, que com o passar dos anos viram cômicas e chegamos a rir um pouco.

O que é ABEND?

Se não leu os artigos anteriores, vamos recapitular um pouco, esta palavra é um anacronimo, que surgiu da união de duas palavras inglesas: Abnormal END, termino anormal de um programa.

Da categoria de problemas em software este é o mais grave, pois derruba o programa e enquanto não for solucionado o usuário estará off-line. Causando muita dor, gerando dor de cabeça sem fim, onde toda a cadeia hierárquica será acionada e a depender da magnitude, literalmente cabeças irão rolar.

Mas antes de aprofundarmos nossa história vamos falar sobre Testes e a necessidade de utiliza-lo para garantir a qualidade do software, após isso irei comentar sobre os tipos de anomalia e o grau de gravidade das mesma.

Tipos de Testes

Existem inúmeros tipos de teste e nomenclaturas que não acabam mais, vou me ater nas mais tradicionais e que realmente importam para nos programadores:

1. Teste Unitário

A unidade mais básica de teste, não depende de ninguém, um trabalho solitário entre o DEV e o programa, garante que o software foi alterado e está aderente a especificação. Uma tarefa essencial para a conclusão do trabalho, pois garante a funcionalidade e que não existe rabichos no software. Se bem conduzido é capaz de capturar 90% das anomalias.

1. Teste Integrado de Sistemas

Existem funcionalidades que somente conseguem serem testados, quando o software é integrado ao SISTEMA, pois aqui encontraremos os caminhos críticos e as dependências do aplicativo, a sua condução perfeita demanda muito esforço, pois existem outras equipes e outros sistemas envolvidos, porém ao ser bem conduzido garantira 97% da confiabilidade do software.

1. Teste de Aceite

Nesta etapa dos testes, tecnicamente o analista/programador tem pouco influencia, sendo um agente passivo, pois o programa estará blindado, alocado dentro da cadeia normal do Sistema, a massa de teste é fidedigna, quase um ambiente de produção, com a sua conclusão 99% dos problemas serão resolvidos,

1. Teste em Produção

A depender do produto, alguns testes deverão ocorrer em ambiente de produção, principalmente devido a criticidade das informações, dificuldade em reproduzir dados entre os ambientes, por isso em alguns sistemas. A funcionalidade piloto é liberada apenas para alguns grupos de usuários e áreas funcionais, que irão trabalhar e debugar o software real-time com uma massa de testes bem robusta.

Poxa vida o que é Massa de Teste?

Desculpe o tiozão, o veio renegado as vezes se empolga e começa a falar sem parar, assumindo que somos todos DEVs, esquecendo os jovens padawans, que ainda não foram a recruta e não conhecem o BASECAMP.

Massa de teste foi uma palavra inventada nos primórdios da computação brasileira, inspirada na massa de pizza, companheira fiel dos programadores que avançavam noite a dentro codificando.

O seu significado é o conjunto de dados necessários para executar um teste de software, protegendo a identidade dos clientes, através de mascaramento de dados e sempre respeitando a LGPD, para evitar grandes confusões.

Recordo-me quando estava trabalhando em Portugal, na Direção Geral de Impostos, quando uma squad de outro sistema, fizeram alterações no IVA e o mock foi apresentado em rede nacional, para indicar os devedores de impostos. Até aí normal, nada excepcional, se na apresentação indicava que Herman Jose, um grande ator/apresentador lusitano fosse indicado como grande devedor, imagem o BO, gerou processo e pagamento de indenizações arcadas pela consultora que desenvolveu o aplicativo.

Agora vamos falar sobre anomalias, erros e abends.

Às vezes é confuso ler sobre temas, chovem termos técnicos, palavras cabalísticas em tons herméticos, que o jovem padawan fica perdido e não entende nada, por isso vamos definir algumas ideias.

Dentro do ambiente de desenvolvimento existem dois tipos principais de palavras, conceitos para definir anomalias e paradas na execução de um software: WARNING e ERROR.

O que é um warning?

No decorrer da execução de um software, acontecem algumas paradas inesperadas, porem que não causam danos e rapidamente resolvidos, para essa categoria de parada foi criado o conceito de Aviso/Warning.

Um warning nada mais é que um aviso ao usuário, ou operador do sistema dizendo que algo não está bem e que deve ser solucionado para avançar na execução do aplicativo. Os mais comuns são a falta de preenchimento de um campo num formulário, um hardware desligado ou inoperante, tipo impressora off-line ou sem papel.

A boa pratica recomenda o programador prever todas essas pequenas paradas e preparar o programa para informar o usuário e indicar maneiras de auto solucionar-se, de maneira a continuar o processamento até o fim.

O que é um ERROR?

Aqui entramos na parte crítica, pois o programa chegando aqui, algo muito ruim está acontecendo e que necessitara de intervenção urgente para corrigi-lo ou contorna-lo, tenha atenção que existem diversos subcategorias de acordo com a origem da anomalia.

A depender do autor e da linguagem de programação podemos dividir Error em 5 subcategorias de anomalias, umas mais graves outra nem por isso:

Engano

Erro

Defeito

Falha

Exceção

O que é um engano?

Ao programarmos devemos sempre nos atentar que Informática é uma ciência exata na fronteira do conhecimento, sendo largamente influenciada pelas ciências humanas. Ou seja, lidamos com seres humanos o tempo todo, então nosso programa deve estar preparado para a interface existente entre o teclado e a cadeira.

Quanto mais interação humana houver nos inputs, mais validações devemos ter na informação, bloqueando caminhos alternativos e possibilidades de usar portas dos fundos

Mesma assim ocorreram enganos, que é justamente um erro causado por usuários imputando dados ou manipulando front-ends, alguns inofensivos outros perigosamente danosos, caso chegando a fraudes e no pior dos casos sabotagem.

Atenção a este tipo de ação, que vai além dos testes, pois o sabotador tenta deliberadamente danificar e conduzir ações que produzem resultados danosos e incorretos.

O que é um erro?

Erro é o acontecimento inesperado que causa resultados imprevisíveis na execução do software, em casos extremos podem gerar um defeito visível, outras vezes não acontece nada, ficando adormecido até que muitos meses após a implantação, um evento desencadeia sua execução, causando surpresas nada agradáveis e surgindo um defeito no software, algumas vezes bem difíceis de identificar.

O que é um defeito?

O defeito é quando o erro se manifesta, podendo ser intermitente ou frequentes, que acarretam em falhas, algumas vezes problemas na compilação, erros em hardwares e muito, mas muito frequente problemas na codificação. Erros de simpatia em nome de funções e ou variáveis, problemas de validação de campos e pasmem nos processos batch, provocados por uma troca de arquivo nos inputs ou cadastro de aplicativos nos servidores.

Acarretando falhas que muitas vezes são silenciosas e apenas quando ganha proporções gigantescas e que o erro explodira e como uma avalanche destruirá tudo a frente encadeando resultados assombrados.

O que é uma falha?

Esse aqui é irônico, senão fosse trágico, uma falha ocorre quando um software não atende aquilo para que foi projetado, algumas vezes intencionalmente por algum requisito faltante, outras vezes por incompetência do Desenvolvedor. Vou exemplificar um projeto que participei em 2008.

Imagine faltando 50 dias para entrar em produção, uma analista nas reuniões falava que estava tudo dentro dos conformes, estando nos testes unitários, pronto para liberar para o SIT. Conforme o gerente do projeto começou a pressionar a pessoa para ver o código.

Simplesmente ela meteu baixa psicológica e o quando fomos ver o código, muitas rotinas de validação estavam comentarizadas e o arquivo gerado era incompleto, não davam erro, porém não funcionam. Por sorte está história teve um final feliz, pois assumi a codificação, suei a camisa trabalhando 12 horas por dia durante 3 semanas e entreguei o código antes do dead-end.

O que é uma exceção.

Explicar exceção é um pouco mais complicado, pois não é um erro em si, ou defeito, mas é algo inesperado que suspende ou bloqueia a execução de um software.

A maior parte das vezes são eventos externos ao software ou Sistema, difíceis de reproduzir e costumam consumir muito tempo para a sua solução. Como exemplo imagine um programa externo que invade a memória do computador e afeta o seu processamento.

Imagine um banco de dados bloqueados por um backup, restauro ou registro bloqueado por outro aplicativos, um servidor que caiu naquele momento e por ai a fora. Prepare-se para lutar contra um inimigo poderoso.

Performance

Para evitar problemas em software, procure codificar de maneira legível, cuidado com os modismos e grafias herméticas de software, como tudo na vida, menos é mais.

Muitos ifs e funções e chamadas e bla bla blas é sinal de código mal feito, logica pouco performática e com certeza evoluirá para o Spaghetti Code. Produzir um Código limpo, fácil e rápido devem ser sua máxima.

Cuidado com querys muito grande e com muitas condições e joiners , pense se é necessária? Não existe uma maneira alternativa para codifica-la? O perigo de querys enormes são ambientais, imagina que no ambiente de produção os registros chegam a milhões enquanto no desenvolvimento são apenas centenas.

Os servidores de produção atendem a todos os Sistemas da Instituição, enquanto nosso software em desenvolvimento usa localmente nosso server e as rotas são únicas para ele.

Pense grande sempre, seu software hoje trata poucos registros, mas amanhã ou depois a quantidade de registros crescera exponencialmente e um if sem utilidade é um milissegundo a mais por registro, multiplique por horas, dias, meses e veja o custo desta geringonça.

Antes de partirmos muita atenção ao comentarizar trechos de código para efetuar testes, antes de passar para a próxima etapa, compare o software gerado com o antigo, existem inúmeros aplicativos que fazem isso, inclusive o Github tem essa funcionalidade, saiba que é um dos erros mais comuns no ambiente de desenvolvimento e inclusive em minha carreira encontrei inúmeros programas em produção, que outras squads efetuaram as guardas e assim ficaram.

O caderninho de boas praticas

Nem conto para você a quantidade de erros, que já produzi em software, faz parte e é inerente a nossa atividade, por isso devemos testar, pois nessa hora capturamos o erro e corrigimos ante de provocar danos.

SE algum dia trabalhar em equipe de sustentação, como boa pratica tenha um caderno para anotar essas descobertas, muita coisa sobre a ferramenta, a framework, a linguagem de programação somente se descobre usando, codificando e testando. Anote isso ajudara em projetos futuros.

Na programação capture as exceções e trate-as com atenção, rotinas de try-catch são fundamentais para proteger seu código, se seu processamento encontrar alguma anomalia, verifique este registro, tente entender a sua origem.

Lembre-se do paradigma principal de nosso trabalho: INPUT => Processamento => OUTPUT, então se algo vai mal no processamento a culpa esta no Input, veja a origem, o que faltou, como chegou até aqui.

Muitas vezes alterações em outros sistemas ou programas externos a seu habitat natural, que provocam isso, conheça os seus fornecedores, descubra se está acontecendo alguma alteração ou nova implementação. Muitas vezes outras equipes esquecem-se de informar os outros de guardas relevantes.

Conclusão

Bom padawan neste artigo executamos um overview sobre erros e acontecimentos inesperados no desenvolvimento do software, tenha cuidado ao codificar e doble ou triplique a atenção ao conduzir os testes.

Nossa atividade necessita de muita atenção para a boa conclusão, procure prever caminhos críticos e coloque armadilhas de erro, para poder capturar a excençao antes de provocar danos e danificar o ouput.

Lembre-se sempre que o usuário tem o magnifico dom de encontrar ou produzir erros, por mais que teste sempre sobrara algo não testado e com certeza o líder do squad, ira le comer o toco, não desanime faz parte da experiência.