20210228

Sunday, February 28, 2021 8:12 AM

verificar como sera o procedimento num cenario com multiplos desenvolvedores, pq como se dara o versionamento por incremento de processo se varias pessoas estiverem trabalhando ao mesmo tempo num mesmo plano de execucao? trabalhar no mesmo processo teria que ser impossivel por conta do tamanho que ele deve ter, mas se o processo contiver um plano de execucao dentro como seria essa alteracao? a organizacao das estorias nao permitira tal cenario, mesmo com falha , entao como nao ha dependencia entre processos ou acusara um erro ou executara normalmente acho improvavel acontecer erro, seria por ma implentacao e nao seguimento das diretrizes, mas tudo eh possivel (contradicao)

a passagem de dados com a interface deve ser tranquila pois ha esse contrato, mas see houver traducao entre os contratos de comunicacao, pode ser dificil trabalhar num fluxo de antes e depois (pipeline) com muitas pessoas. temos que levantar um cenario desses. nesse caso para agregador dica bom mas tem que" aceitar" todos patterns dentro do cenario de composicao

dado que:

```
planoExecX ->
        processoA
        processoB ->
                *impl processo...
                planoExecY ->
                        processoA
                        processoB
        processoC ->
                *impl processo...
                planoExecP ->
                        processoA
                planoExecO ->
                        processoA
```

~~nao tratar plano de execucao como processo. criar contrato/interface iplanoexecucao~~

*tudo eh processo

utilizar ponteiros para os planos?? acho uma boa

pensar numa solucao para o reuso, pois se houver reutilizacao de um plano de execucao em mais de um lugar e esses lugares necessitarem de comportamentos distintos de um mesmo processo? como versionar?

pensar na nomenclatura de namespace/package, versionamento, plano de ececucao, processo

ideia 1:

```
planosExecs
        uteis
                notificadores
                        notificadorA
                        notificadorB
                        notificacoes
```

notificacaoA
notificacaoB
notificacaoC
notificacaoD
publicadoresMensagens
ouvintesMensagens

planoExecX
planoExecX
processos
processoA
procpoliticaK
procregraJ
processoB
processoC
proctraducaoA
planoExecY
planoExecY
processos
processoA
processoB
planoExecP
planoExecP
processos
processoA
planoExecO
planoExecO
processoA

observar que a imutabilidade tem 2 cenarios em tempo de execucao e construcao:
dados : sao imutaveis para o mundo externo (ex agregados)
comportamento: imutaveis para tudo e todos , inclusive implementacao(processo, testes (pensar bem nesse cenario p ver se eh pratico))

o ~~que eu faco se existir um processo com 30 testes e eu alterar o comportamento do processo?~~
- ~~replico os 30 cenarios de teste adicionando o comportamento?~~
- ~~altero o comportamento do processo e seus testes e versiono no git de forma especifica?~~

nao corremos esse risco acima por conta dos tipos de atualizacao, entao sera necessario apenas adicionar o teste respectivo ao novo comportamento

as estorias podem ser representadas dentro de features ou epicos que sao tratados como plano de execucao:
epico titulo nome plano execucao
bug - igual estoria
estoria titulo nome processo
task titulo blablabla

epico titulo nome plano execucao
feature titulo nome processo
estoria titulo blablabla
task titulo blablabla

bug - igual estoria

feature titulo nome plano execucao
estoria titulo blabla - tag nome processo
task titulo blablabla
bug - igual estoria

feature titulo nome plano execucao
estoria titulo nome processo
bug titulo blablabla
task - igual bug

estoria titulo plano execucao
task titulo blabla - tag nome processo
bug - igual task

comunicacao entre modulos e contextos deve ser feita por pub/sub com contrato sera o unico acoplamento e ainda sera baixo, se nao atribuir contrato a comunicacao pode nao acontecer,, dabdo muita liberdade e quebrar cenaarios sem saber.
utilizar isso mesmo para cenarios de validacao de hash (ah informacao no maximo pode ser verificada diretamente, mas nunca obtida por completo) entre os comunicantes. teremos uma pprta de entrada e uma de saida para isso com selecionador interno por processo

diferenca entre politica, regra e especificacao

- especificacao é versionavel por si mesma e nao apenas pelo processo de acordo com esta modelagem, seja por versao semver ou por vigencia , publico-alvo, por nome de lei, validacao de dados traduzidos de acordo com versao ou leis etc, nao pode ser anulavel ela eh simplesmente substituida ou utilizada de acordo com a versao mais nova, validacaonpara um chain ou factory ou builder etc, estatico no codigo, valida entrada e saida de pub/sub
- politica é mais dinamica ela sera versionavel por processo, porem ela pode ter uma vida util de vigencia mas que é alteravel de acordo com suas regras. ela tem um alvo e agrupa as regras para atender esse objetivo. entrada de dados de um plano de execucao e processo, saida de dados, regra de negocio , regra de aplicacao etc contratos de emtrada e saida de dados, (alterna o fluxo???). ela permite ser anulavel completamente ou em partes por uma politica mais nova. pode vir de banco de dados
- regra é versionavel por processo , é tambem mais dinamica e tem a ver com dados e condicionais para validacao dessas estruturas de dados: cliente tem de existir, data de vencimento nao extrapolada , contrato de entrada e saida de dados. pode ser anulavel por regras de sua politica mais nova, todas as regras independente do motivo de alteracao sao executadas desde a primeira sequencialmente, ppde vir de banco de dados.

*os tres sao processos na modelagem.
*politicas agrupam regras
*especificacao funciona sozinha. objetiva. rigida. nao pode ser anulavel, nao vem de banco de dados

para processos e planos de execucao: comportamento eu agrego (composicao), seja por adicao, exclusao ou alteracao. sempre imutavel.

exemplo:

- para um plano de execucao X num processo A eu quero remover uma regra exercida numa politica para ele: eu crio uma nova regra para A indicando o objetivo, anulando a regra que preciso e insiro ela depois da regra anterior na composicao da politica. se essa politica for utilizada em mais de um lugar que nao deva obedecer essa alteracao entao temos que montar uma nova politica baseada na anterior e incluir a anulacao.

?motivos de mudanca de regras: anulacao, inremento, correcao de bug

?motivos de mudanca de plano de execucao: incremento de processo, remocao de processo, correcao de bug, se existir alteracao de codigoque existe apenas nesse plano e ele passa por versionamento

?motivos de mudanca de processo: alteracao ou remocao de plano de execucao interno, correcao de bug (se tiver alteracao de codigo que existe apenas nesse processo ai ele passa por versionamento

fazer um teste com source generator do c# para ele consumir codigo salvo no redis (mesmo que distribuido):

inicia a compilacao
executa script para obter os codigos no redis ou scylladb e desconecta dele
escreve esses codigos para leitura do compilador
compilador valida esses codigos
segue o fluxo de compilacao

se der merda saberemos em tempo de compilacao...

como eu encontro um problema de alteracao de regra implementada erroneamente? procuro o plano de execucao atraves de seu codigo universal -> procuro nos processos de regras e politicas. como eh ~~sequencial~~ e rastreavel, analisando a regra que deu errado ,consigo encontra-la facilmente para corrigir .

?corro o risco de ter um loop infinito entre plano de execucao e processos?

plano de execucao chama um processo que chama esse mesmo plano de execucao . n deve ter sentido essa implementacao.

~~se nao for um processo , plano de execucao entao pode alterar codigo normalmente versionando pelo git mesmo~~

as questoes que o versionamento de processos e planos de execucao devem atacar é o uso de varios modelos de implementacao ao mesmo tempo e nao deixar com que um fluxo atrapalhe o outro. responsabilidade unica tem que estar na ponta da lingua. é uma das colunas desse modelo. aliando a tdd fica improvavel essa colisao isso ocorrer.

uma coisa boa para entendimento do incremento dos processos eh uma visao clara de pertencimento expandida na tela, isso e possivel pegando cenarios executados mas por enquanto depende de ferramenta de classe da ide por exemplo. montar essa ferramenta? pensei numa ferramenta que pode ate ser util para validacao do modelo, mas meu medo em deixar eele muito ridido mesmo trabalhando por composicao:

~~independente da linguagem :~~

1. ~~a ferramenta escaneia os arquivos e coloca como chave em um dicionario o nome dos arquivos~~
2. ~~monta as relacoes entre esses itens do dicionario, para iss...~~

utilizar reflection para pegar o fluxo de execucao 👍👍👍?? pode ter problema de performance, tratar com benchmarks.

1. crio um plano de execucao q obtem esses planos e processos organizando de acordo com suas chamadas (assincronia atrapalharia? deveria ser um fluxo a parte para nao depender do desenvolvedor atualizar. como faremos uma especie de controlador de execucao atraves de um plano de execucao vindo de uma chamada, entao podemos definir nesse cara um processo de ferramentas que possuem cenarios em que devem ou nao ser executados ou um liga desliga. pode ser util para auditoria de performance entre outros

observar que esse modelo vai de encontro ao que chamamos de middleware por exemplo, que seria a execucao em pilha, contudo temos que validar a execucao de tudo. multi threads seria muito foda, acho que em go nao teriamos esse problema (analisar) mas em outras linguagens temos que ficar atento por conta da ordem de execucao, logo se houver dependencia teremos problema, we for agregacao nao teremos desde que as validacoes sejam independentes.

apos algumas pocs percebi que com essa modelagem eu consigo 100% de cobertura de teste no que importa que sao entrada e saida dos processos e o processamento em si. de forma alguma havera a possibilidade de validar comportamento de linguagem no teste. se eles altereram temos o incremento, entao o codigo anterior garante o que ja existe enquanto o atual garante o novo e assim sucessivamente

~~para sustentar a ideia de incremento no plano de execucao, eu tenho que criar um metodo que permita a adicao de um processo especificamente num lugar da sequencia. entao a linha de raciocionio é eu crio um novo plano alterando utilizando o anterior e substituindo apra o novo o que fpr necessario.~~

~~verificar o seguinte: nao ha motivo para se ter um plano de execucao dentro de um processo~~

toda execucao de processo é feita por um plano de execucao e deve-se ter a liberdade para posicionar esse plano dentro do processo

todo processo e plano devem ser assincronos e os planos de entrada devem ser tambem multi-thread, entao teremos:

plano de execucao que é chamado de "fora", independente do meio, executa em multi-thread): grpc, soap, http, fila etc

lembrando que api consultada diretamente serve apenas para validacao de dados por hash
o resto tudo deve ser feito via mensageria

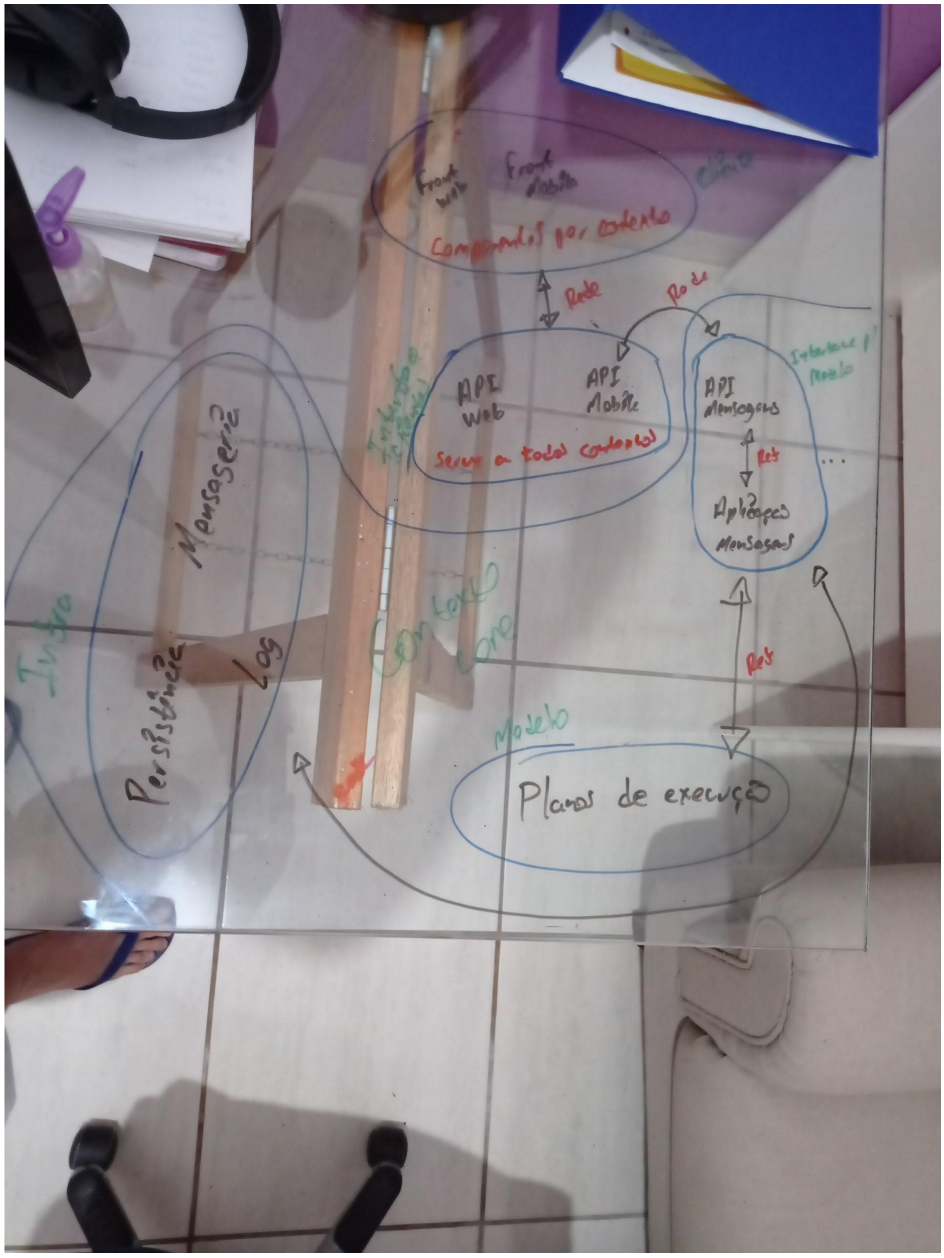
diagrama de comunicacao 1:

load balanced em todas comunicacoes por rede

permite escalar horizontal qualquer componente

infra por enquanto para testes (mas no geral independe de tech)

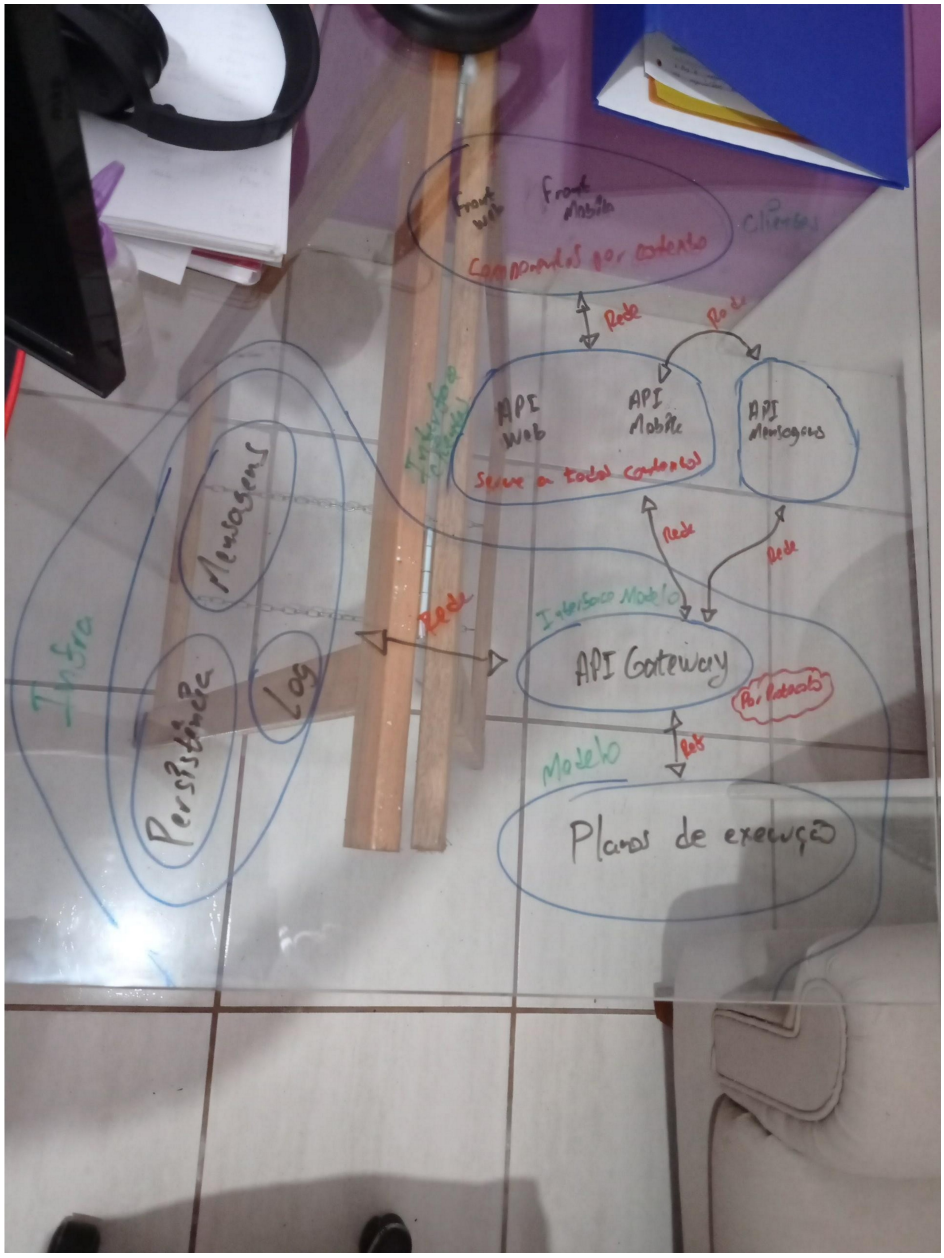
redis para mensagens (pub/sub)
redis para persistencia de modelo



importante: tem que ser passado metadados de cliente consumidor para as apis , seja sistema ou operador

deve existir processos para trabalhar autorizacao

diagrama de comunicacao 2:



*API gateway por protocolo e por contexto. cada sub dominio tera uma fila interna e a comunicacao entre todos sera analogo a correio (mensageria) apenas sdk e shared kernel podem ser referenciados.

vou dar um exemplo de comportamento de persistencia. essa persistencia armazenara os dados por versao de plano de execucao e processo. entao vamos imaginar que eu tenha plano de execucao por tabela (relacional mesmo) esses dados jamais serao alterados , mesmo sendo do proprio contexto , logo , como teremos versionamento do "cliente" ate o servidor caso esses dados tenham que ser recuperados em tela , eles poderao, pois a versao anterior do componente e do processo ainda esta la. essa eh uma questao forte desse modelo tb