

Curso: Análise e Desenvolvimento de Sistemas – ADS

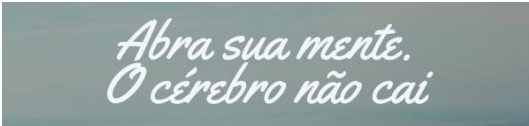
Ano: 2023/1

Orientação Técnica (OT) – 27

Inativação de registros

Introdução

Lhe damos boas vindas à OT final desta trilha! Nela você adquirirá um conhecimento bastante relevante durante sua jornada. E não é só mais um código, mas sim uma nova maneira de pensar! Então...



*Abra sua mente.
O cérebro não cai*

Inativação de registros

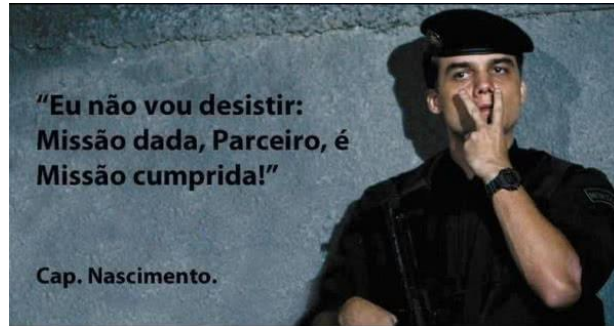
A inativação de um registro pode ser usada como uma **alternativa ou complemento à exclusão**. Com ela, transformamos um registro em algo que **não pode ser mais utilizado** em ações futuras, **porém não o excluimos** do sistema, evitando possíveis problemas de integridade referencial que vimos anteriormente. Além disso, ele possibilita que o **registro seja reativado**, fazendo com que não seja necessário realizar o mesmo cadastro novamente caso se necessite novamente daquele registro.

Como ponto negativo, ter apenas a inativação faz com que, mesmo que desnecessários, **os dados não possam ser excluídos** pelo usuário, gerando um maior número de dados no BD do que o necessário. Porém em alguns casos, isso não se torna um problema pela quantidade de dados no BD.

No nosso caso, a usaremos no **registro de marcas**, como uma opção extra para o usuário. Ou seja, **adicionaremos a possibilidade de inativação** de uma marca **mas manteremos a exclusão**. Além disso, faremos com que, por padrão, as **novas marcas estejam ativadas**, evitando assim que façamos alterações na inserção de marcas.

Para que tudo isso dê certo, uma série de mudanças serão feitas, e não apenas na programação...

ATENÇÃO: Essa OT pode ser um pouco difícil. Como motivação, segue uma célebre frase do cinema nacional:



Alteração do banco para suportar a inativação

Para darmos suporte para a inativação/reactivação, precisamos **alterar a tabela de marcas**. Atualmente, ela possui apenas 2 colunas: 'id' e 'nome'. Precisamos então **adicionar a coluna 'status'**, que será valorada da seguinte forma:

- **0** para marcas inativas;
- **1** para marcas ativas.

Para isso, no Workbench, execute o seguinte **comando SQL ALTER TABLE**, conforme imagem abaixo:

```
ALTER TABLE marcas ADD status TINYINT(1) NOT NULL DEFAULT 1;
```

Destrinchando:

- **ALTER TABLE marcas**: Indicamos que queremos alterar a tabela 'marcas';
- **ADD status TINYINT(1) NOT NULL**: Indicamos que estamos adicionando uma coluna chamada 'status', do tipo *TINYINT(1)* cujo *preenchimento é obrigatório*;
- **DEFAULT 1**: indicamos assim que essa coluna tem o *valor padrão 1*, ou seja, se nenhum valor for passado ao se inserir um novo registro na tabela, esta coluna assume o valor 1 para o registro.

Verifique agora, ainda no Workbench, todos os registros da tabela, e perceberá que a **coluna 'status' foi adicionada**, e que *todos os registros existentes na tabela já receberam o valor 1*, graças ao "**DEFAULT 1**".




Controlando o status pelo CRUD de marcas

Essa é simples de entender: se temos agora como ativar/inativar as marcas, precisamos fazer algo em nosso CRUD para permitir que o usuário faça isso, ou então as alterações anteriores de nada servirão. Para isso, faremos um processo bem similar ao de exclusão, mas com uma pitada de alteração.

Criando o botão de ativação/inativação

Como primeira ação, na tabela que exibe as marcas registradas, você adicionará uma nova coluna à esquerda da que contém os botões de alterar e excluir, com um *slider*.



Calma, Sérgio... O *slider* não vive, e também não se alimenta. Ele é um tipo de botão. Com certeza você o conhece, talvez não por esse nome... Olha ele aí: Porém, ele não é nativo do HTML. Precisamos fazê-lo com efeitos CSS, normalmente usando um checkbox HTML. Veja os links abaixo, eles demonstram algumas alternativas de construção de um *slider* com CSS: 

- https://www.w3schools.com/howto/howto_css_switch.asp
- <https://willianjusten.com.br/criando-um-switch-button-com-css/>
- <https://www.youtube.com/watch?v=QM-UPLdy2kw>
- <https://www.youtube.com/watch?v=BQSNBa3gZIU>
- <https://proto.io/freebies/onoff/>

Se ainda não criou a **nova coluna** faça isso agora, e dentro dela faça com que apareça um **slider**, seguindo algum dos exemplos.

ATENÇÃO: É relevante para as ações posteriores que ele seja baseado em um **checkbox**, então escolha um exemplo que seja assim. Ele também não deve conter os textos “On/Off” como na maioria dos exemplos, ok?

Buscar o status da marca no servidor

Nesse momento, nossa intenção é fazer com que o lado cliente receba também o status da marca. Para realizar essa parte da tarefa, você deve alterar algumas coisas no backend, adicionando a elas o novo campo do BD:

- A classe modelo;
- O método que consulta as marcas registradas no BD e as retorna em uma lista para nossa classe Rest.

Fazendo isso de maneira correta, você será capaz de visualizar os status das marcas registradas se der um **console.log** no **success** do **Ajax** do método **COLDIGO.marcas.buscar**. Então, nosso próximo passo é...

ATENÇÃO: Não dá pra seguir em frente sem isso pronto...

Exibir o status no botão switch

Primeiro, vamos analisar o que temos. Já possuímos um **checkbox**, visualmente alterado para um **switch**, que vai nos **mostrar o status de cada marca**, e também já **buscamos o status** em si no BD e o **retornamos ao lado cliente**. Assim, precisamos dar um jeito de fazer com que esse **checkbox** **fique marcado quando o status for 1** (ativo) e **desmarcado quando o status for 0** (inativo).

Para deixarmos um checkbox marcado, devemos fazer **basicamente a mesma coisa** que fizemos no formulário de edição de produtos para **deixar a marca daquele produto selecionada** (encontre essa parte de seu código para usar como referência): **se o status da marca que estamos verificando for 1, colocamos o atributo “checked” no checkbox**.

Após essa etapa, quando acessar o CRUD de marcas, os botões *switch* aparecerão mostrando corretamente se a marca está ativa ou inativa. É claro que para testar isso, você deve ter algumas marcas em cada status, né...

ATENÇÃO: Não dá pra seguir em frente sem isso pronto... Se precisar de ajuda, por favor nos chame!

Chamar a função de ativação/inativação

Agora, criaremos “mais uma letra” em nosso CRUD. Se até agora ele cria, consulta, altera e exclui, agora ele deve ativar/inativar. Para isso, você precisará fazer as mesmas coisas que fez para todos os outros:

- criar uma função JavaScript, que:
 - será acionada com o clique do botão;
 - receberá como parâmetro o id da marca (como as funções JS de alteração e exclusão);
 - se comunicará via Ajax com o servidor para alterar o status da marca;
 - exibirá um aviso de sucesso ou erro dependendo do que acontecer no servidor.
- criar um novo método na classe *MarcaRest* para receber essa requisição e retornar a resposta ao cliente;
- criar outro método na interface *MarcaDAO*;
- criar outro método na classe *JDBCMarcaDAO* que realize a ativação/inativação da marca através de um SQL UPDATE.

Outros detalhes podem ser necessários, mas essas coisas supracitadas não devem ser feitas de um jeito diferente sem que fale conosco antes justificando. Acreditamos que, depois de ter feito quase 10 vezes esses processos, já deva ter o conhecimento suficiente para fazer o que solicitamos sem grandes problemas, então vá em frente!

ATENÇÃO: Se precisar de ajuda, por favor nos chame, mas não dá pra seguir em frente sem isso pronto...

Finalizando o serviço

Mesmo com tudo o que fizemos, **ainda pode ocorrer um problema**: se tentarmos alterarmos o status mas isso não acontecer lá no servidor, o **checkbox vai ficar “invertido”**, ou seja, se tentarmos ativar uma marca e der erro no backend, o checkbox não vai voltar para inativo.

Assim, faça com que a função **error** do **Ajax** altere o **checkbox** para **voltá-lo para o status anterior**. Para isso, uma **dica muito importante**: além do id da marca, receba como parâmetro da função ativada pelo botão switch o próprio elemento clicado, usando a palavra **“this”**. Assim, você poderá facilmente manipular o checkbox que foi clicado para ativá-lo e inativá-lo. Feito isso, é só realizar a seguinte linha em JS no local onde quiser voltar o checkbox ao seu estado anterior:

```
checkbox.checked=!checkbox.checked;
```

Para entender essa linha, considere “checkbox” como a variável em que recebeu o **“this”** na função, e lembre-se que a exclamação serve para negar ou inverter um valor. Assim, podemos dizer que o **atributo checked de “checkbox” (que é a variável que contém o checkbox clicado) recebe o valor inverso do que ele possui atualmente (assim, o *true* vira *false* e vice-versa)**.

Ajustes gerais finais

Esta fase de nosso projeto está chegando ao fim. Você já possui 2 CRUDs em funcionamento, mas nem tudo está acontecendo da melhor maneira possível. Assim, solicitamos aqui que faça algumas correções/melhorias finais em seu projeto:

Mostrar mensagens do BuildErrorResponse

Ao enviar uma Response usando o método BuildErrorResponse, enviamos uma mensagem para o usuário, certo? Mas você reparou que a mensagem que enviamos não é a que mostramos a ele? Corrija isso em **TODOS os error dos Ajax** que seu projeto possui. Deixando bem claro, é só neles que precisam ser feitas alterações, OK? Se não sabe de onde tirar a mensagem que enviamos, dê um **console.log** do que foi recebido do servidor, e tente descobrir onde ela está...

Erros informados ao cliente via BuildErrorResponse

Verifique se em seu sistema TODAS as mensagens de erro geradas nas classes MarcaRest e ProdutoRest são enviadas para o lado cliente via BuildErrorResponse. Se alguma é enviada pelo BuildResponse, altere, afinal esse método é para quando as coisas dão certo...

Cadastro/edição de produtos

Agora que nossas marcas podem ser desativadas, altere o registro de produtos para que apareçam apenas as marcas ativas nos formulários.

Um é bom, dois é demais!

Já reparou que podemos cadastrar duas marcas com o mesmo nome? E que também podemos registrar dois produtos exatamente iguais? Não permita que isso aconteça!

ATENÇÃO: no caso de produto, seja coerente: O que tornaria um produto igual ao outro? Todos os dados precisam ser iguais para que eles sejam considerados repetidos?

Conclusão

Com o fim desta OT, chegamos ao fim da trilha backend, e consequentemente das duas trilhas de programação web! Nelas, você usou pelo menos 6 linguagens diferentes para:

- criar um layout básico de um site web;
- deixar nossas páginas interativas;
- comunicar seu sistema com um banco de dados;
- fazer verificações lógicas...;
- entre outras coisas básicas no desenvolvimento de um sistema web.

Porém, nosso trabalho para a Coldigo Geladeiras não pára por aqui! Em momentos oportunos, ainda a utilizaremos para adquirir novos conhecimentos, através do Módulos que realizaremos para ela, que abordarão, entre outros:

- relatórios;
- formulários mestre-detalle;
- e autenticação.

Assim, podemos dizer que essa conclusão conclusiva conclui a OT que conclui a trilha que conclui as trilhas pré Projeto Individual!

Após finalizar a OT, crie um novo commit no Git com o nome da OT e comunique um orientador para novas instruções.