FACULDADE DE TECNOLOGIA DE SÃO JOSÉ DOS CAMPOS

FATEC PROFESSOR Jessen Vidal

JOSÉ FRANCISCO FORNEIRO JUNIOR

AMBIENTE DE SUPORTE AO BANCO DE DADOS

Orientador: Juliana Forin Pasquini Martinez

São José dos Campos

2022

SUMÁRIO

[1 Introdução 3](#_Toc104804156)

[1.1 Definição do problema 3](#_Toc104804157)

[1.2 Objetivo 3](#_Toc104804158)

[2 Desenvolvimento 4](#_Toc104804159)

[2.1 Arquitetura 4](#_Toc104804160)

[2.2 Detalhes 5](#_Toc104804161)

[3 Resultados e Discussão 9](#_Toc104804162)

# Introdução

A empresa Brasil317 é uma empresa de licitações com sistema ainda em construção. Sendo assim ela ainda não possui um sistema de suportes, tendo como consequência que todas as alterações dos pregões precisam ser feitas diretamente no banco de dados, aumentando a possibilidade de erros e um enorme prejuízo financeiro a empresa.

## Definição do problema

Conforme o problema citado na introdução, foi proposto o desafio de criação de um sistema de suporte para diminuir as alterações feitas diretamente no Banco de Dados. Dessa forma, somente o líder do setor de suportes terá a permissão de fazer alterações diretamente no Banco de Dados. Para os demais integrantes da empresa, como operações e estagiários, somente será permitida a alteração através do sistema.

## Objetivo

O objetivo deste trabalho consiste em construir uma API REST e posteriormente um Fron end para executar suportes diminuindo assim a necessidade de alterações diretamente no Banco de Dados.

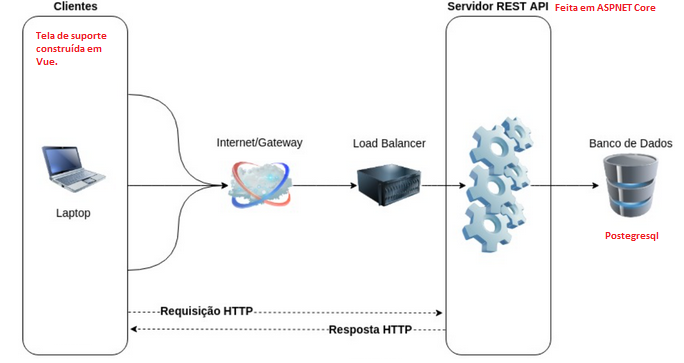
# Desenvolvimento

Para o desenvolvimento desse trabalho está sendo utilizado ASP NET CORE 3.1, API REST e programação Front end por VUE. O banco de dados é Postgresql.

## Arquitetura

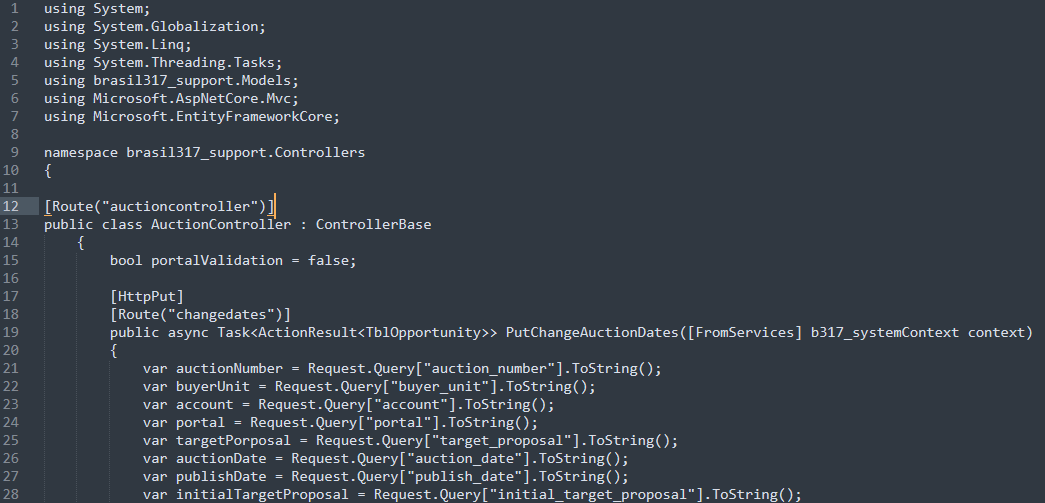
A arquitetura REST possui uma série de regras e princípios que devem ser seguidas:

* **Cliente-Servidor:** Trata de separar as preocupações de interface do usuário (User Interface) do banco de dados, abstraindo a dependência entre os lados clientes/servidor e permitindo a evolução desses componentes sem impacto e quebra de contrato.
* **Interface Uniforme:** É a interoperabilidade entre os componentes cliente e servidor. Como o cliente e servidor compartilham da mesma interface, é necessário estabelecer um contrato para a comunicação entre essas partes.
* **Stateless:** Cada requisição acionada entre a comunicação cliente-servidor deve possuir toda a informação necessária e compreensível para realizar a origem da requisição, não sendo de responsabilidade do servidor armazenar qualquer tipo de contexto. Isso pode gerar alto tráfego de dados e impacto na performance da aplicação, porém pode-se utilizar recursos de cache nesses casos.
* **Cache:** É utilizado para melhorar a performance de comunicação entre aplicações, otimizando o tempo de resposta na comunicação entre cliente-servidor.
* **Camadas:** Nesse modelo, o cliente não deve conectar-se diretamente ao servidor da aplicação, porém uma camada de balanceamento de carga deverá ser acionada para essa responsabilidade.

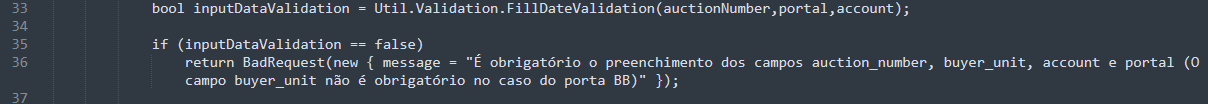


## Detalhes

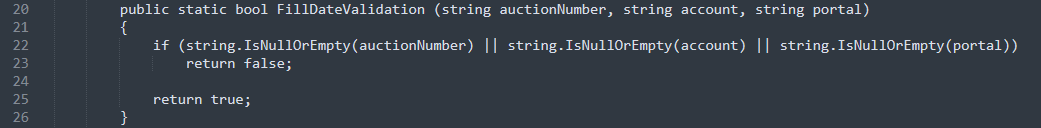
Declaração da Rota e variáveis a serem inseridas no Postman:



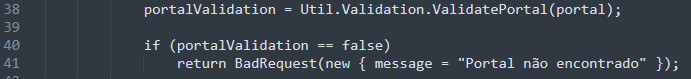
Envia para a classe FillDateValidation(auctionNumber,portal,account) para verificar se o auctionNumber, portal e account estão preenchidos:



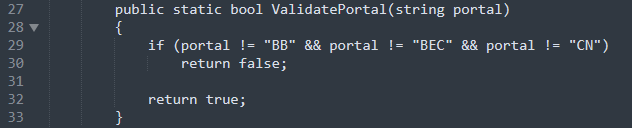
Classe FillDateValidation:



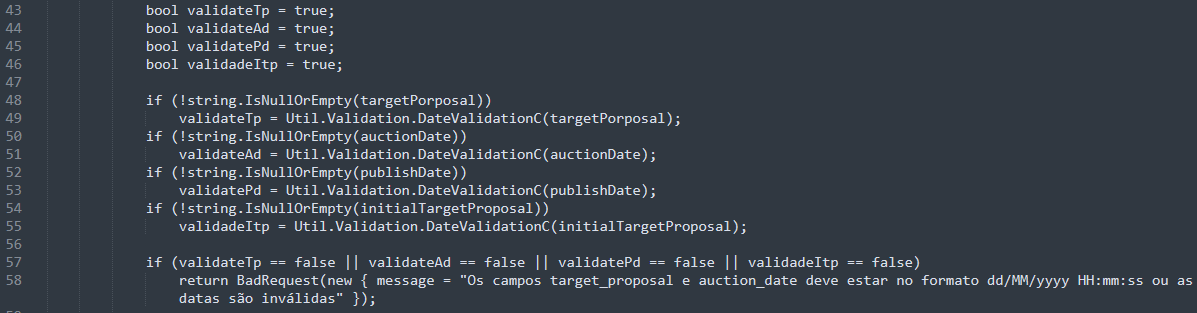
Envia para a classe ValidatePortal(portal) para verificar se o portal é válido.



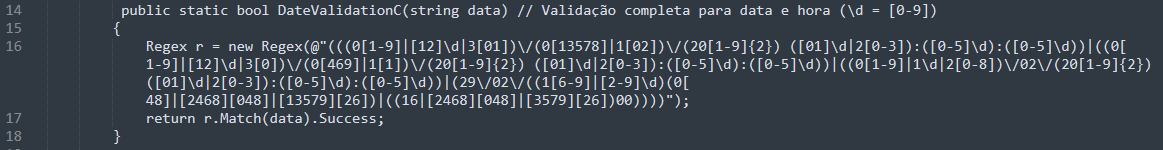
Classe ValidatePortal:



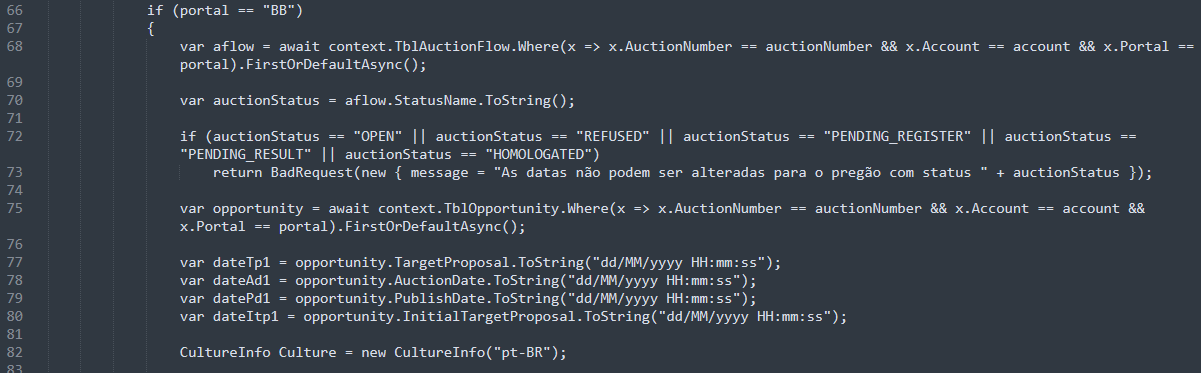
Envia para a classe DateValidationC para verificação se a data é Válida:



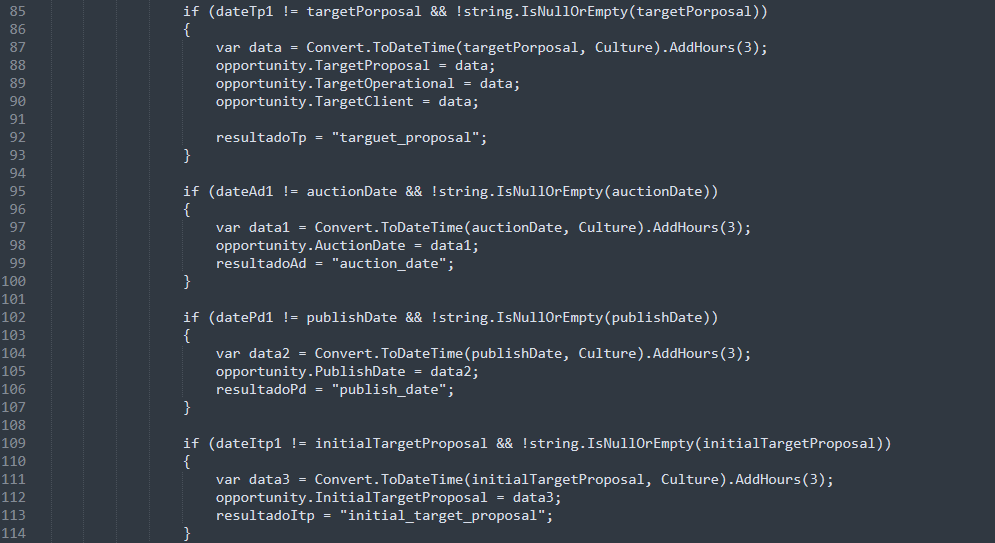
Classe DateValidationC: (Feita com regex)



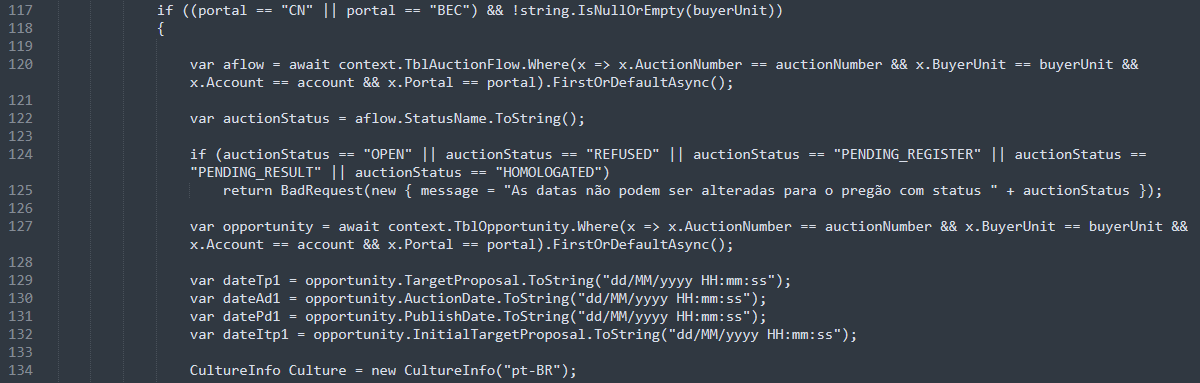
Se o pregão for do BB, pega o status do pregão solicitado, verifica o status do pregão e pega as datas do pregão:



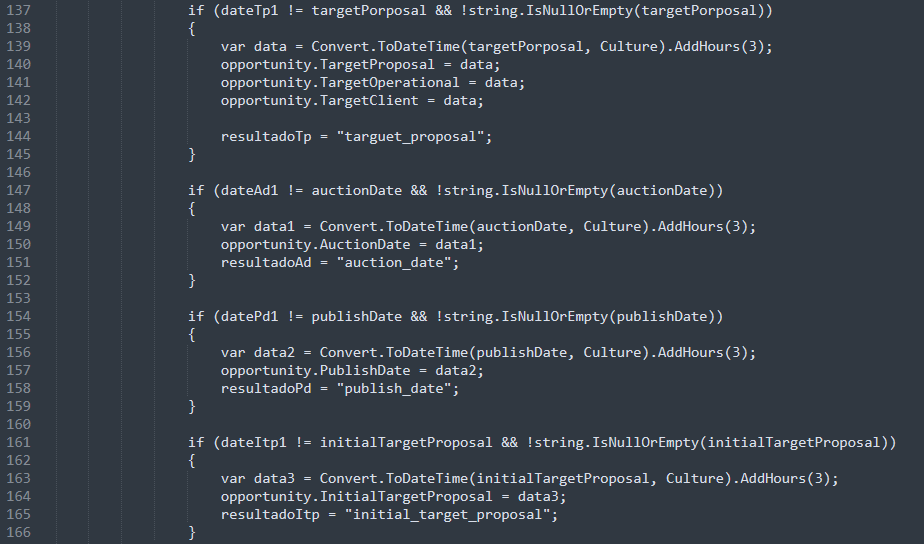
Faz a comparação das datas do pregão que estão no banco com as novas datas. Se as novas datas forem iguais do banco ou nulas, passa para a próxima data:



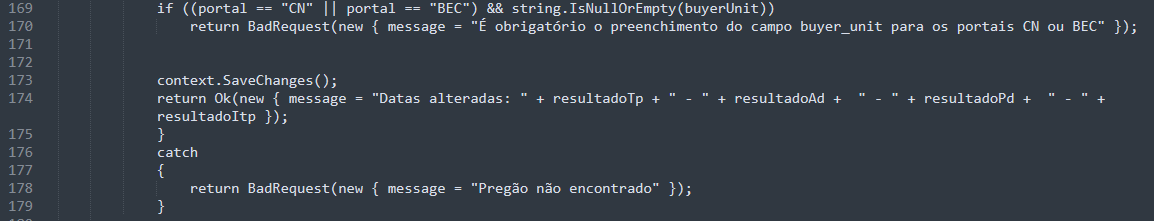
Se o pregão for do CN ou BEC, verifica se o buyerUnit está preenchido, pega o status do pregão solicitado, verifica o status do pregão e pega as datas do pregão:



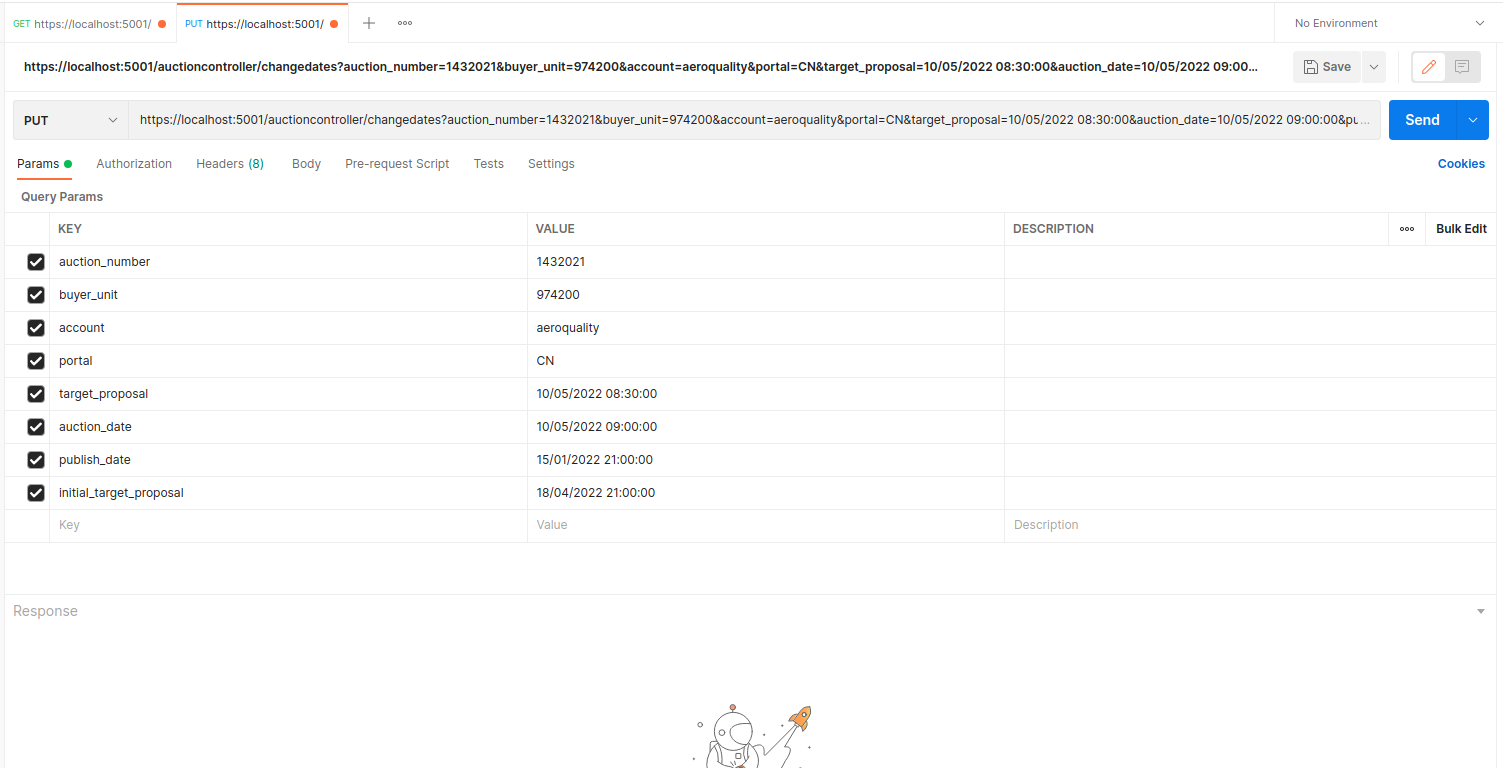
Faz a comparação das datas do pregão que estão no banco com as novas datas. Se as novas datas forem iguais do banco ou nulas, passa para a próxima data:



Caso o portal seja do CN ou BEC e o buyerUnit não esteja preenchido:



Tela do Postman para alterar as datas do pregão:



# Resultados e Discussão

As tecnologias utilizadas no desenvolvimento deste projeto são as mesmas da Brasil317 que é ASP.NET Core no Visual Studio Code, Postman e o banco de dados Postegres. Futuramente será utilizado o Vue no front end. Essas ferramentas, são as utilizadas na Brasil317 que é a empresa que trabalho.

No desenvolvimento desse código foram encontrados algumas dificuldades como regras de validação de datas, haja vista que não encontrei nenhum pacote para isso em c#. Sendo assim, a solução que encontrei foi utilizar regex para esse fim.

Também tive dificuldades nas demais validações e tratamentos de erros, devido a grande quantidade de regras para bloqueios que transações que não podem ocorrer.

O projeto está sendo mostrado ao meu cliente (Brasil317) a qual está satisfeita com o que está sendo apresentado.