READ ME:

**O que vale destacar no código:**

1: Organização dos solução, modularizando-no em

projetos menos a fim de satisfazer os estórias

A solução está divida em 5 partes:

*RestaurantApp.Voting.AcceptanceTest* : Testes de aceitação

*RestaurantApp.Voting.Business*: Regras de negócios

*RestaurantApp.Voting.Common*: Modelos de domínio

*RestaurantApp.Voting.DesktopApp*: Interface de usuário

*RestaurantApp.Voting.Repository*: Acesso de Persistência de dados.

2: As estruturas das implementações das unidades lógicas e de regra de négocio

de uma forma que facilita alem permitir os teste dos mesmo via a inversão controle

das unidades.

3: A utiliza de teste automatizados vinculados fortemente com as estórias especificada

via técnicas de BDD - Behaviour Driven Development em que as estórias

se tornam "executáveis" e verificáveis via automação.

4: A modelagem dos Objetos Domínio que representa com clareza o contexto do problema o qual o sistema pretende resolver.

**O que dá para melhorar no sistema:**

Bem feito mas não perfeito:

1: O sistema não trata o caso inevitável de emparte simples considera os mais votados como escolhidos do dia.

2: O (ainda) melhor seria ter um sistema central que faz o abertura

da votação tal que não haja necessidade para cada sistema de votação se-ecarregar de criar os candidatos do dia..

3: Deveria a possibilidade de cadastro de usuários e restaurantes via a interface de usuário.

4 3: pudia ter tido mais testes automatizados.

**Algo a mais a dizer:**

Resolver, o melhor poder propor uma solução para um problema real foi quão divertido quanto tanto gratificante. Aprendi muito desenvolvimento deste sistema.

Sempre procurando evitar a preocupação com o prázo e muito foco qualidade interna

e externa do software.

**Detalhes do software:**

1: Os usuários possuem senha para poder “*garantir”* a segurança da votação.

Os usuários "fakes" já cadastrados possuem senhas iguais ao seu nome para facilitar testes.

2: Depois terá seleção de restaurantes para votação do dia

3: O mais votado é atualizado em intervalos de 1(um) minuto até o horário do almoço e quando hover um voto, após do horário do almoço será exibido será exibido um botton par poder atualizar manualmente..

4: O horário de almoço é configurável no Arquivo AppSettings.config na pasta *..\RestaurantApp\RestaurantApp.Voting.DesktopApp\ConfigFiles\*. Isto como na Estória 3 especifica está definido para até o meio dia.

O connection string está definido arquivo ConnectionStrings.config na pasta *..\RestaurantApp\RestaurantApp.Voting.DesktopApp\ConfigFiles\* pastaentão o nome da base pode ser mudado, atualmente se usa (localdb)\v11.0 usa o seguinte SQl pode ser usado para limpar a base de dados. O modelo conceitual está no arquivo *...\RestaurantApp\RestaurantApp.Voting.Repository\RestaurantVoterDbContext.edmx*

DECLARE @sql NVARCHAR(MAX) = N'';  
   
SELECT @sql += N'  
ALTER TABLE ' + QUOTENAME(OBJECT\_SCHEMA\_NAME(parent\_object\_id))  
 + '.' + QUOTENAME(OBJECT\_NAME(parent\_object\_id)) +   
 ' DROP CONSTRAINT ' + QUOTENAME(name) + ';'  
FROM sys.foreign\_keys;  
   
SELECT @sql += N'  
DROP TABLE ' + QUOTENAME(OBJECT\_SCHEMA\_NAME(object\_id))  
 + '.' + QUOTENAME(OBJECT\_NAME(object\_id)) + ';'  
FROM sys.tables;  
   
PRINT @sql;  
EXEC sp\_executesql @sql;