**Guia rápido: como funciona a aplicação e seus requisitos para rodar:**

**1 - Tecnologias utilizadas:**

C#, CSS, Asp.NET, Razor, WCF

IDE – Visual Studio 2013

Banco de dados: SQL Server

ORM: Entity Framework

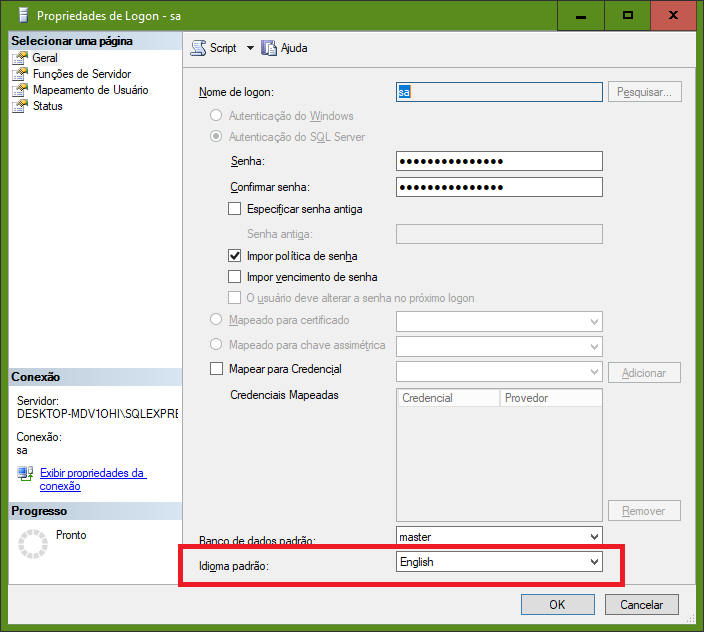
**2 – Configuração do Banco de Dados:**

Nome do servidor: .\SQLEXPRESS

Usuario: sa

Senha: 123456

*Obs: É necessário que o usuário sa esteja configurado com o Idioma padrão English. Caso contrário poderá dar problema nos campos data (Job, será explicado mais adiante).*



**3 – A aplicação:**

Dentro da pasta WebServiceGameOperation será encontrado quatro arquivos, são eles:

**GameOperation:** Projeto no qual faz as requisições ao serviço.

**GameResultService\_1:** Serviço com os dois endpoints.

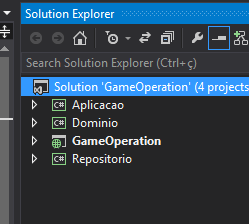
**GameOperationInsertScoreJob:** Job para rodar os inserts da tabela temporária na tabela oficial.

*Obs.: Não foi possível utilizar o SQLServerAgent (Job do banco) porque a versão que foi utilizada do Sql Server é a express, e o SQLServerAgent se encontra em versões pagas do banco, a solução encontrada para rodar um job foi criar um aplicativo com o mesmo conceito (GameOperationInsertScoreJob).*

**ScriptBanco.sql:** Script contendo todas as querys para a criação do banco.

**4 - Algumas considerações:**

A solution Explorer do projeto game operation está estruturada da seguinte forma:



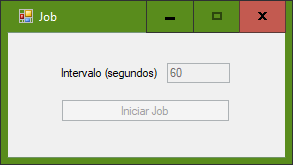
**Aplicação** – Referente ao acesso do banco de dados para acessar os usuários (classe Player), e os Jogos (Classe Game) – via injeção de dependência.

**Dominio** – As Classes Game e Player são definidas aqui, classes que fazem representação do banco de dados de modo geral são criadas aqui.

**GameOperation** – Todos os arquivos asp, css, javascript estão aqui, ou seja toda estrutura mvc, dentro da pasta Service References é onde está o conector com o serviço dos endpoints.

**Repositorio** – Lógica para os inserts e updates listas e todos as operações crud. Nota: nem todas elas são utilizadas mas foram criadas para manter o padrão crud.

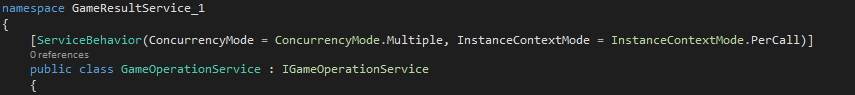
*Obs.: Para utilizar a aplicação que fará a requisição do serviço é necessário que além do próprio serviço o aplicativo* ***GameOperationInsertScoreJob*** *esteja rodando.*



**Como foi tratada a múltipla concorrência:**

Foi utilizada duas camadas:

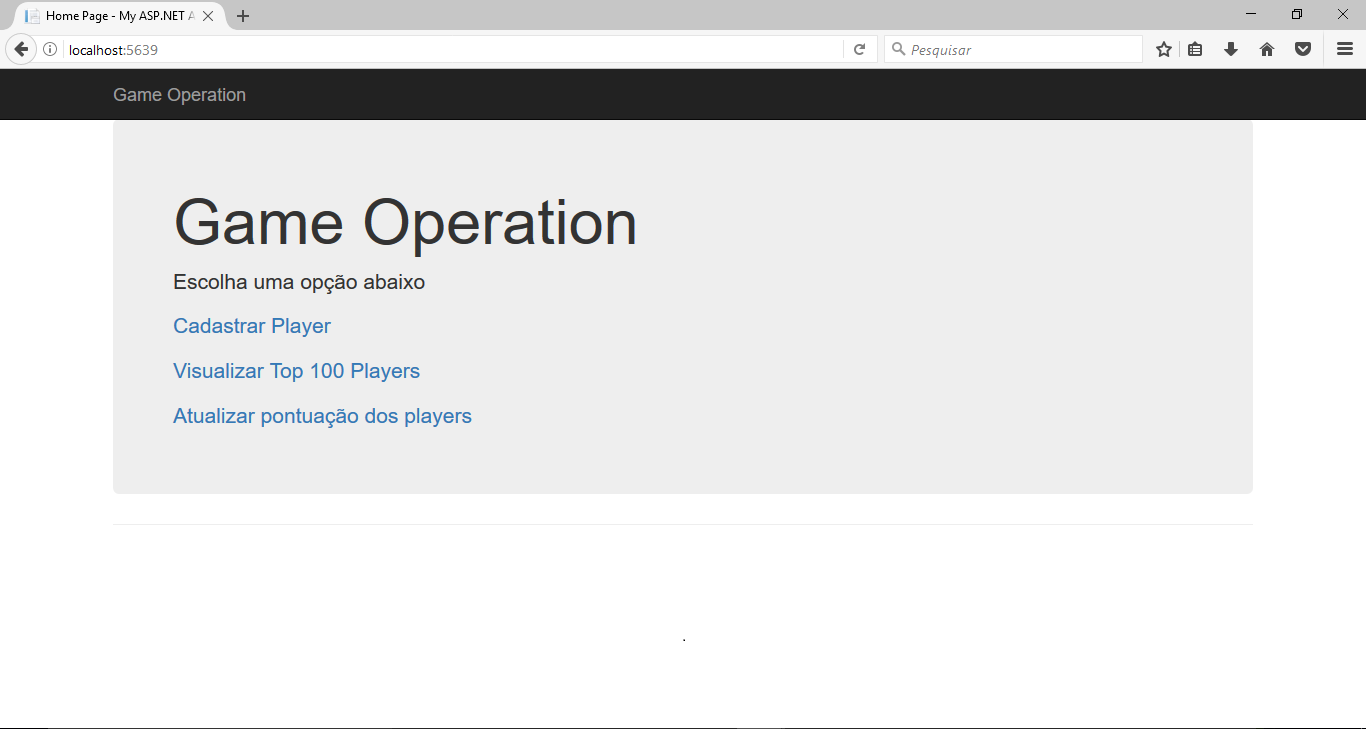
A primeira no próprio serviço no projeto GameResultService\_1.



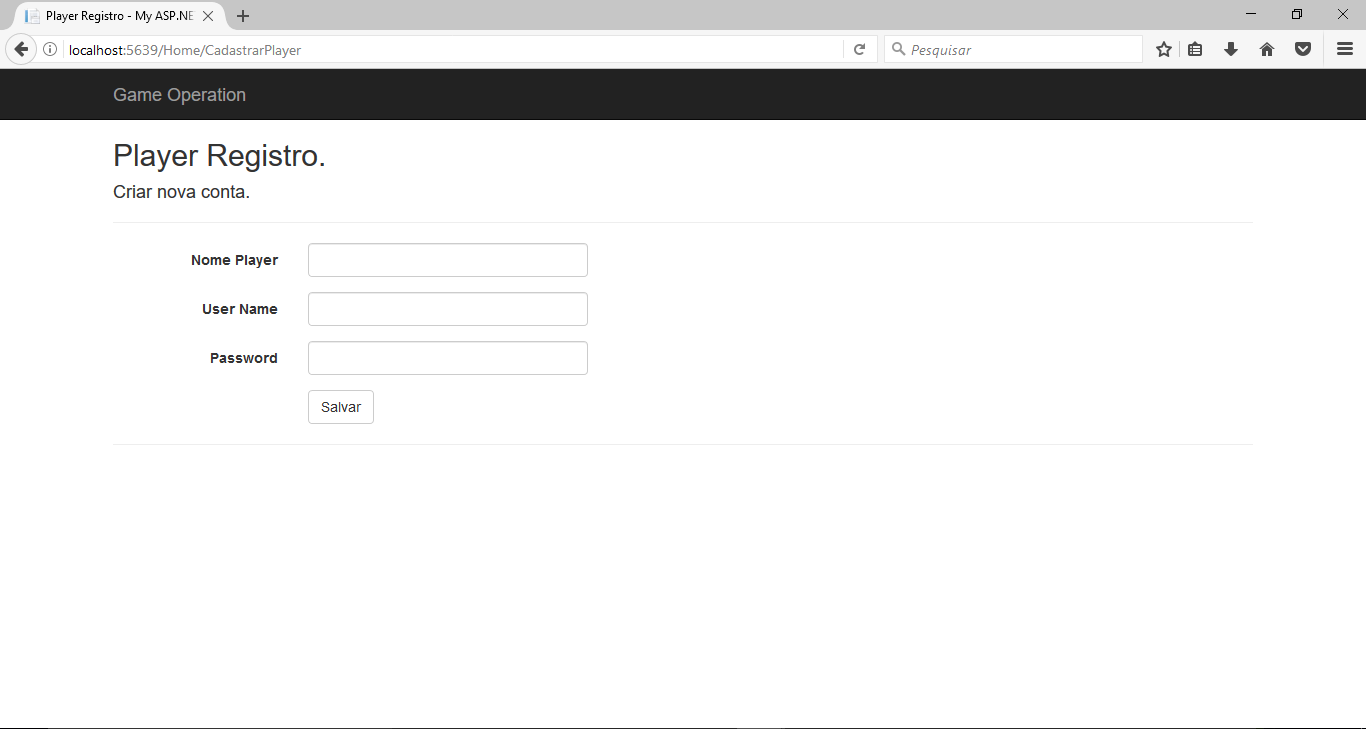
A configuração do ServiceBehavior para o muliple mode, PerCall do próprio WCF garante que este modo haverá sempre uma instância servindo cada requisição. Mas mesmo assim quando é salvo no banco após passar pelo serviço os dados são colocados na tabela temporária GameResultService\_1, com o Job Rodando (aplicativo GameOperationInsertScoreJob) é possível informar o número de segundos que os inserts na tabela oficial acontecerão. Essa verificação é feita pela data em ambas as tabelas (temporária e tabela oficial GameResult) e por isso é importante aqui o idioma do usuário sa estar em inglês, tive alguns problemas com isso e utilizando o idioma inglês resolveu.

**LeaderBoard:** Para mostrar o ranking de jogadores por pontos foi criada a View de mesmo nome – Leaderbord.

Tela inicial



Tela para cadastrar os players caso não exista:



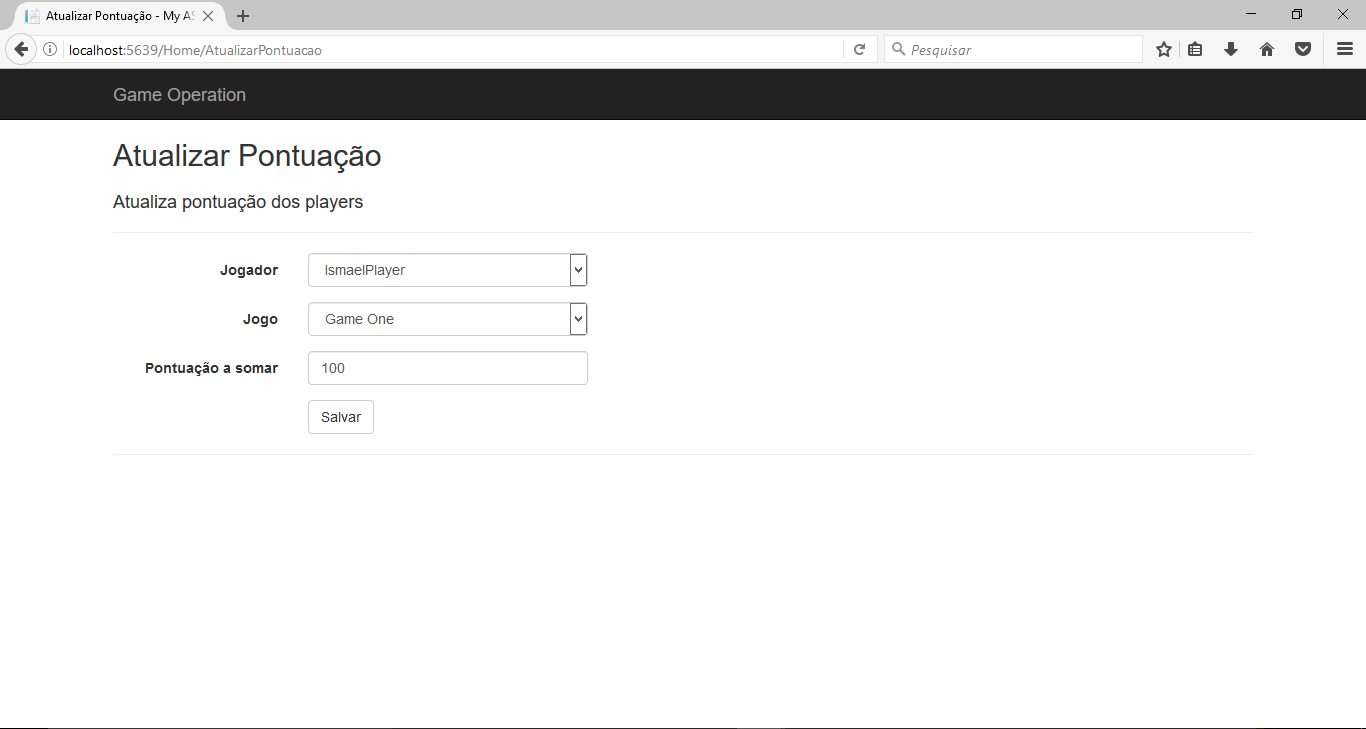
*Obs.: Não foi adicionada edição e deleção de players na view, mas ambas os métodos de acesso para um projeto crud completo foram criados, e serão implementados posteriormente.*

Visualizar top 100



*Obs.: Para simular a requisição de tempos em tempos a tela é atualizada automaticamente a cada 5 segundos.*

Tela para atualizar a pontuação dos players.



Serviço do projeto GameOperationService

