API para gerenciamento de clientes

Documentação técnica

Analista Desenvolvedor: Moacir de Moraes

Sumário

Arquitetura Linguagem e Padrão utilizados 3
Organização das pastas e funções dentro do projeto 4
Regras de Negócio e seu funcionamento 5
Parâmetros configuráveis 6
Deploy da Aplicação 7
Utilização da aplicação 8

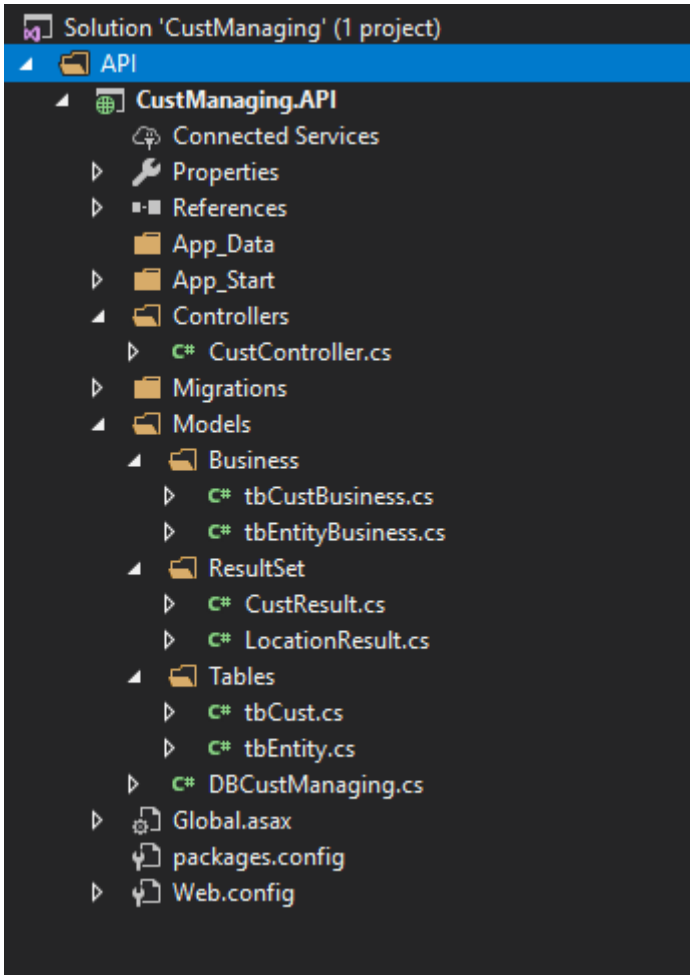
Arquitetura Linguagem e Padrão utilizados

O projeto foi desenvolvido dentro do Microsoft Visual Studio utilizando a linguagem C# no .Net framework. Foi utilizado o template de projeto de WEB API da ferramenta.

A ferramenta utilizada para persistência dos dados no SQL SERVER foi o Entity Framework, com a linha de utilização Code First que possibilita que as tabelas necessárias e suas estruturas sejam criadas e gerenciadas pela própria aplicação.

Organização das pastas e funções dentro do projeto

As pastas estão organizadas da seguinte forma:



- A pasta API contém todo o projeto da aplicação, caso seja necessário adicionar um novo projeto dentro da Solution, este ficará organizado dentro de outra pasta;
- As pastas App_Start, Controllers, e Migration contém os arquivos padrão de uma API e do Entity Framework: A rota de execução das funcionalidades, as funcionalidades e as configurações de gerenciamento das alterações estruturais feitas nos objetos tabela que devem ser replicados ao banco, respectivamente.
- A pasta Models contém as subpastas de Regra de negócio, ResultSet e Tabelas que são:
    - Business: Todas as regras de negócio de cada tabela e funções relacionadas
    - ResultSet: Todas os Modelos que são utilizados para tratativa de dados e retorno de informações porém não se tornam “fisicamente” tabelas.
    - Tables: Todas as tabelas que serão criadas e manipuladas pelo Entity Framework.

Regras de Negócio e seu funcionamento

As regras estão contidas dentro da pasta “Business” e são:

Regras da tbCustBusiness

```
namespace CustManaging.API.Models.Business
{
    public class tbCustBusiness
    {
        public string GoogleUrl = ConfigurationManager.AppSettings["GoogleURLLocalizar"].ToString();

        public tbCust InsertCust(tbCust cust)...

        public LocationResult getGeoLocation(string address)...

    }
}
```

InsertCust: Trata toda a criação de um novo cliente no banco de dados.

getGeoLocation: Utiliza a API do Google Maps para retornar as coordenadas a partir de um endereço enviado.

Regras da tbEntityBusiness

```
namespace CustManaging.API.Models.Business
{
    public class tbEntityBusiness
    {
        public void InsertEntity()...

        public string getClosestEntity(double custLat, double custLng)...

        public double compareDistance(double custLat, double custLng, double entLat, double entLng)...
    }
}
```

InsertEntity: Trata toda a criação das entidades legais a partir de um arquivo .Json configurável (Ver tópico de parâmetros configuráveis). Este método é utilizado apenas na primeira execução, criando a estrutura de tabelas e informações necessárias para as demais funcionalidades da aplicação.

getClosestEntity: Retorna a entidade mais próxima a partir do envio das coordenadas do cliente.

compareDistance: Compara a distância a partir de duas coordenadas enviadas.

Parâmetros configuráveis

Os parâmetros configuráveis são:

```
  <add key="GoogleURLLocalizar" value="https://maps.googleapis.com/maps/api/geocode/json?address="/>
  <add key="EntityJsonPath" value="C:\Users\Moacir\Downloads\code-challenge-master\seed\legalEntity.json"/>
</appSettings>
<system.web>...</system.web>
<system.webServer>...</system.webServer>
<runtime>...</runtime>
<system.codedom>...</system.codedom>
<entityFramework>...</entityFramework>
<connectionStrings>
  <add name="DBCustManaging" connectionString="data source=.\SQLEXPRESS;Integrated security=false;User ID=us
</connectionStrings>
```

Key GoogleURLLocalizar: Url de acesso a API do Google que retorna coordenadas a partir de um endereço enviado. A versão free utilizada é limitada a 2500 consultas por dia.

Key EntityPathJson: Caminho para o arquivo .Json que contém as entidades legais a serem cadastradas.

ConnectionStrings: String de conexão ao banco de dados.

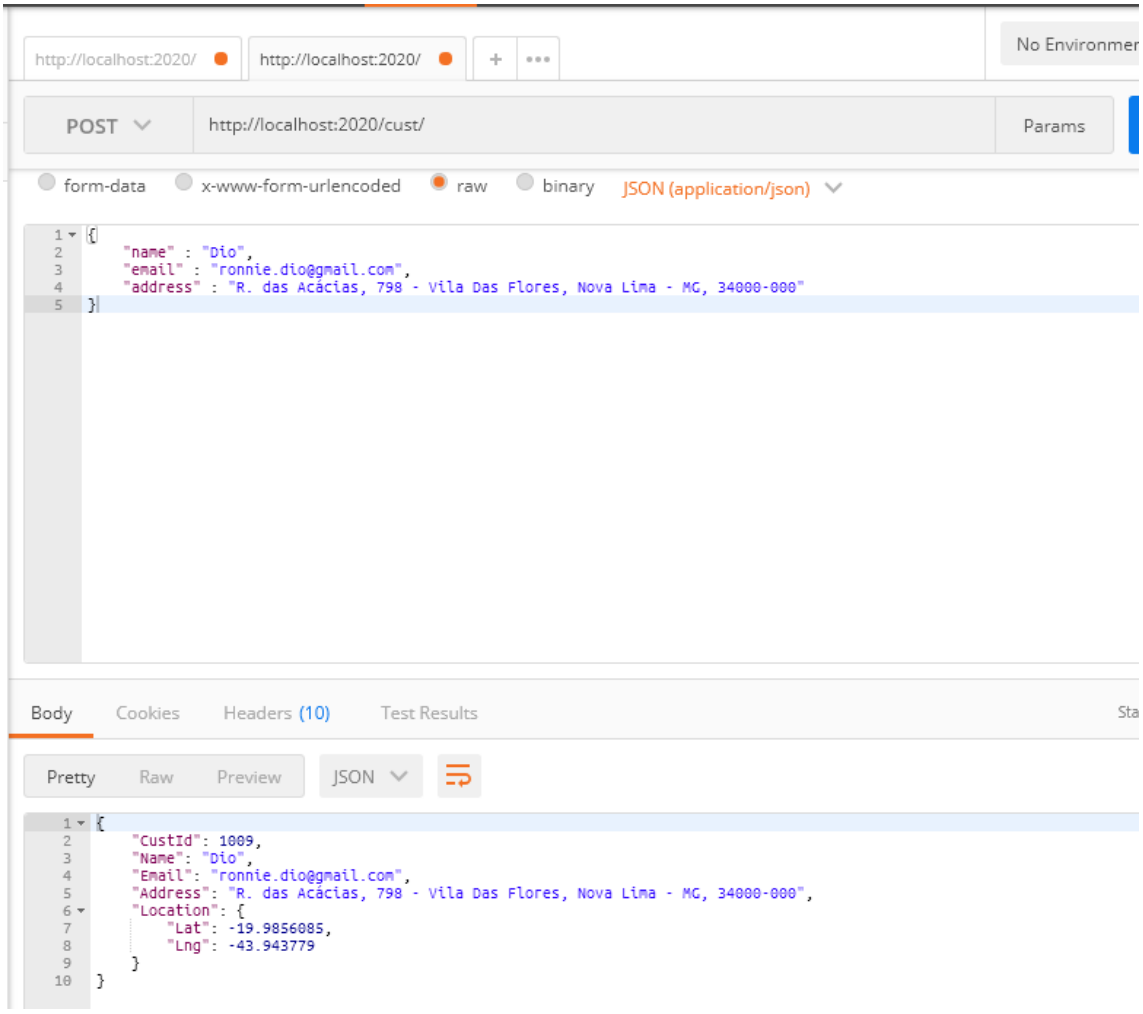
Deploy da Aplicação

Os passos para o deploy são:

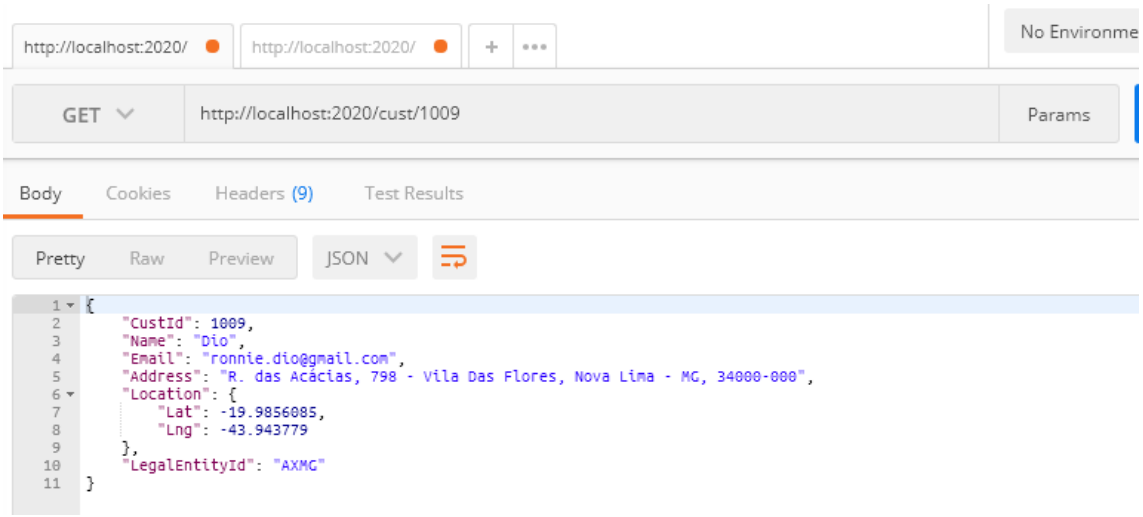
- Copiar os arquivos da pasta “Deploy” para um endereço acessível pelo IIS local;
- Alterar a key do caminho para o arquivo .Json com a lista de entidades a ser cadastradas;
- Alterar a connectionString (Importante manter as propriedades Name e Initial Catalog como “DBCustManaging”);
- Criar o banco “DbCustManaging” na instancia do SQL SERVER desejada;
- Criar um site no gerenciador do IIS mapeando a pasta “Deploy” copiada;
- Iniciar o site criado.

Utilização da aplicação

Para o Post é utilizada a seguinte estrutura de envio e retorno:



Para o Get a seguinte estrutura de envio e retorno:



(No exemplo foi utilizada a ferramenta Postman)