ESCOLA SUPERIOR DE TECNOLOGIA IPCA

3º Ano

ESI

Engenharia de Sistemas Informáticos

2020/2021

Integração de Sistemas de Informação

Criação e Uso de Serviços REST

Pedro Barros 14363

Marco Henriques 16961

Data:10 de janeiro de 2021

Conteúdo

Introdução

O presente relatório é baseado no trabalho prático da unidade curricular de “Integração de Sistemas de Informação”, do curso Engenharia de Sistemas Informáticos do 3º ano.

O relatório tem como objetivo de servir de documentação do trabalho prático elaborado com o tema de “Criação e Uso de Serviços REST”.

Este Trabalho prático consiste na implementação de operações sobre uma base de dados alojada na cloud (Azure). Esta base de dados armazena dados relativamente a feiras locais.

O objetivo principal deste trabalho é a implementação de Serviços REST de forma a serem acessíveis por qualquer um que precisar de lhes aceder.

Enquadramento

Num processo de desenvolvimento de software muitas vezes é preciso recorrer ao desenvolvimento REST, pois posteriormente uma aplicação a desenvolver em Web ou Windows tem um conjunto de serviços muito interessantes ao seu dispor.

Para este desenvolvimento vamos então precisar de uma Base de dados, sobre a qual iremos realizar operações.

Esta Base de dados estará hospedada na Cloud da Azure (servidor:isi2020.database.windows.net, username: isi2020, password: qwerty123.). Esta base de dados contém tabelas referentes aos feirantes e a feiras, desenvolvida no âmbito do trabalho prático da unidade curricular de Integração de Sistemas de Informação.

Os serviços REST desenvolvidos foram ainda publicados também na Cloud da Azure, onde podem ser acedidos no seguinte link: https://tp2isi.azurewebsites.net

Serviços REST

Com o objetivo de ajudar a efetuar operações sobre uma base de dados em SQL, alojada na Cloud da Azure, foram desenvolvidos serviços para duas diferentes tabelas: Feirante e Feiras.

Para além dos serviços supracitados, foi usado um serviço externo através de uma API que permite obter condições e previsões meteorológicas do concelho de Barcelos.

HttpPost

Foram desenvolvidos serviços que permitem a inserção de dados.

Antes de desenvolver o serviço, é importante defini-lo. Para o exemplo que se segue, em REST, define-se:

```
40        [HttpPost("AdicionarFeirante")]
```

Este serviço faz um INSERT de um Feirante na base de dados, utilizando como parametros de entrada os varios estados de Feirante. Retorna true se foi bem sucedido e false se não foi bem sucedido.

Neste caso, queremos adicionar novos valores à base de dados, por isso selecionamos a tabela Feirante inserindo os respetivos parâmetros. Assim que o comando esteja terminado, ExecuteNonQuery() deverá retornar um valor inteiro que é o numero de linhas afetadas, se retornar 0 significa que nada aconteceu, se retornar 1 ou maior significa que algo foi alterado:

```
        try
        {

            // criar query
            string query = "INSERT INTO Feirante ( Cod_id, Nome, Cc_number, Contacto, Localidade, Username, Pass, Email) VALUES (@Cod_id, @Nome, @Cc_number, @Contacto, @Localidade, @Username, @Pass, @Email)";

            SqlCommand cmd = new SqlCommand(query, connection);

            // Instancia parametros
            cmd.Parameters.AddWithValue("@cod_id", x.Cod_id);
            cmd.Parameters.AddWithValue("@Nome", x.Nome);
            cmd.Parameters.AddWithValue("@Cc_number", x.Cc_number);
            cmd.Parameters.AddWithValue("@Contacto", x.Contacto);
            cmd.Parameters.AddWithValue("@Localidade", x.Localidade);
            cmd.Parameters.AddWithValue("@Username", x.Username);
            cmd.Parameters.AddWithValue("@Pass", x.Pass);
            cmd.Parameters.AddWithValue("@Email", x.Email);

            // executa a query e retorna o numero de linhas afetadas
            int cnt = cmd.ExecuteNonQuery();

            // fecha conexão
            connection.Close();


            if (cnt > 0)
            {
                return true;    // Foi adicionado
            }
            else
            {
                return false;  // Não foi adicionado
            }


        }
        catch (Exception ErroInserirValues)
        {
            throw new Exception(ErroInserirValues.Message);
        }
    }
```

HttpDelete

Foram também implementados serviços para a eliminação de registos na base de dados.

```
[HttpDelete("RemoveFeirante/{cod_id}")]
```

Para a exemplificação deste serviço usamos a tabela Feirante. Para a eliminação de um registo desta tabela usamos como parâmetro de entrada um ID, e através desse ID é eliminado o Feirante da tabela.

```
try
{

    // abre conexão
    connection.Open();

    // cria query
    string query = "DELETE FROM Feirante WHERE  @cod_id = cod_id ";
    SqlCommand cmd = new SqlCommand(query, connection);
    int feirantes;

    // instancia parametro
    cmd.Parameters.AddWithValue("@cod_id", cod_id);

    // executa query e retorna numero de linhas afetadas
    feirantes = cmd.ExecuteNonQuery();


    connection.Close();

    if (feirantes >= 1)
    {
        return true;
    }
    else
    {
        return false;
    }


}
catch
{
    return false;
}
```

Este Serviço faz uso de uma query com o STATEMENT DELETE, para apagar um registo da tabela Feirante usando o ID do mesmo.Retorna true se eliminou e false se não eliminou.

HttpGet

Implementamos também alguns serviços para retornar dados da base de dados.

```
[HttpGet("VerFeirantes")]
```

Este serviço utiliza o SELECT para retornar todos os campos da tabela Feirante. Em caso de sucesso retorna TRUE e em caso de insucesso retorna FALSE.

```
try
{

    // cria query (Feirante)
    string query = "Select * FROM Feirante";

    SqlDataAdapter cmd = new SqlDataAdapter(query, connection);

    DataTable dt = new DataTable();

    // preenche dataset com o resultado da querry
    cmd.Fill(dt);

    //Cria uma lista com feirantes
    List<Feirante> h = new List<Feirante>();

    // Percorre a datatable e adiciona á lista
    foreach (DataRow line in dt.Rows)
    {

        Feirante novoFeirante = new Feirante(
            Convert.ToInt32(line["Cod_id"]),
            line["nome"].ToString(),
            Convert.ToInt32(line["cc_number"]),
            Convert.ToInt32(line["contacto"]),
            line["localidade"].ToString(),
            line["username"].ToString(),
            line["pass"].ToString(),
            line["email"].ToString()
            );

        h.Add(novoFeirante);
    }

    // serializa para json
    string jsonString = JsonConvert.SerializeObject(dt);



    return jsonString;
}
//
catch (Exception Erro)
{
    // fecha conexão
    connection.Close();
    throw new Exception("Erro" + Erro.Message);
}
```

Serviço Externo - Meteorologia

Para o desenvolvimento deste serviço foi necessário escolher uma API que possa fornecer um serviço de meteorologia de uma determinada zona do país.

Para aceder a esta API, é nos concedido um *URI* modelo em que temos de adicionar nome da cidade e APIkey disponibilizada no api.openweathermap.org

É efetuado um requeste para ligar ao endereço através de *HttpWebRequest.* Se foi possível ligar com dados meteorológicos sobre um distrito. No nosso caso a escolha foi *Openweathermap.*
Irá ser retornada uma *string* sob formato JSON que deverá ser tratado e armazenado num objeto. As classes que descrevem esse objeto foram geradas automaticamente.

```
HttpWebRequest request;
#endregion
try
{
    url = "http://api.openweathermap.org/data/2.5/forecast?q={cityname}&appid={APIkey}";

    uri = new StringBuilder();
    uri.Append(url);
    uri.Replace("{cityname}", HttpUtility.UrlEncode("Barcelos")); // 2742416 code Barcelos
    uri.Replace("{APIkey}", apiKey);



    #region PreparaPedido
    // Prepara e envia pedido
    request = WebRequest.Create(uri.ToString()) as HttpWebRequest;
    #endregion

    #region EnviaPedidoeAnaliseResposta
    // Analise resposta
    using (HttpWebResponse response = request.GetResponse() as HttpWebResponse)
    {
        if (response.StatusCode != HttpStatusCode.OK)
        {
            string message = String.Format("GET falhou. Recebido HTTP {0}", response.StatusCode);
            throw new ApplicationException(message);
        }

        // Guarda conteudo num stream de memoria
        var copyStream = new MemoryStream();
        response.GetResponseStream().CopyTo(copyStream);

        // Serealiza de json para objeto
        DataContractJsonSerializer jsonSerializer = new DataContractJsonSerializer(typeof(Root));
        copyStream.Position = 0L;   // Inicio do stream
        Root myClass = (Root)jsonSerializer.ReadObject(copyStream);
```

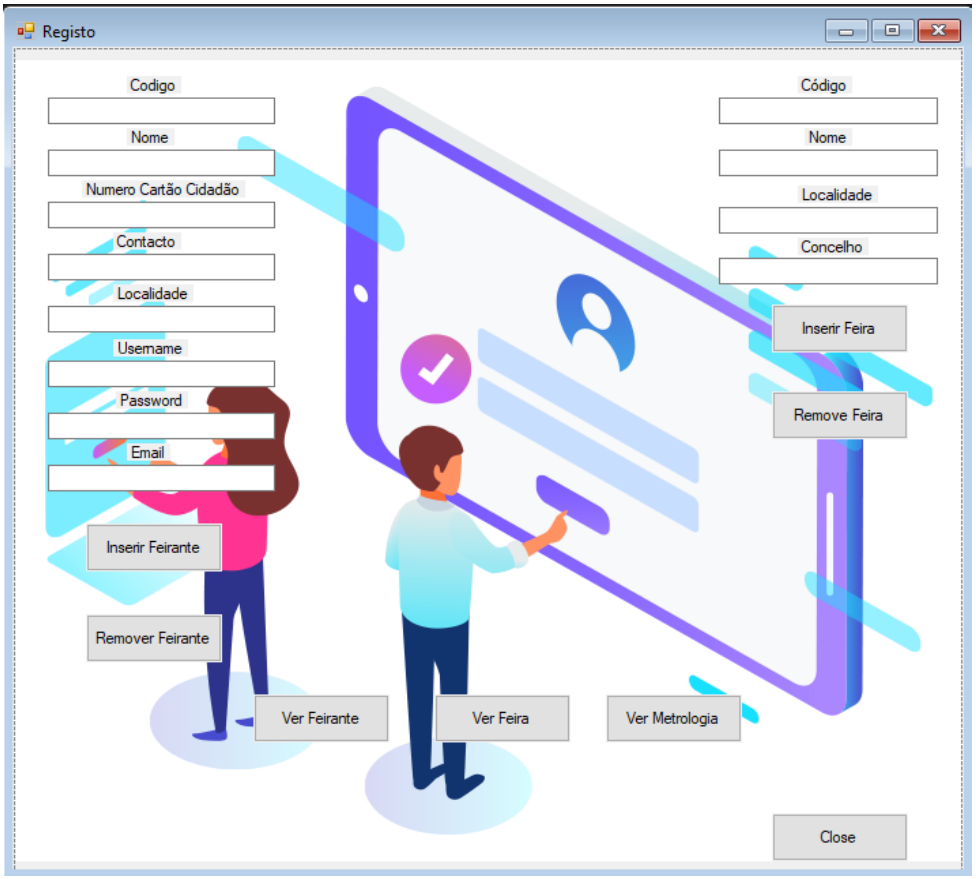
Clientes

No âmbito da realização de testes aos serviços desenvolvidos foi desenvolvida 1 aplicação cliente, para serviços REST.

Cliente REST

No cliente que utiliza serviços REST, implementamos apenas para 7 serviços, que achamos ser os mais importantes e interessantes em mostrar:

- Inserir Feirante
- Remover Feirante
- Inserir Feira
- Remover Feira
- Ver Feirantes
- Ver Feira
- Ver Meteorologia



Problemas e adversidades

Ao longo do desenvolvimento do trabalho foram surgindo algumas adversidades que nos fizeram perder tempo crucial no desenvolvimento do projeto.

Na publicação da Base de dados escolhemos como host a Azure por ser uma parceira com o IPCA e por isso, tirarmos proveito dessa parceria em nosso favor. Nesta fase ocorreram alguns contratempos, como a primeira base de dados que criamos expirou o saldo disponível para alunos e tivemos que criar outra base de dados com outro email institucional o que nos fez perder algum tempo. Perante a inserção de dados e de consulta dos mesmos, a base de dados por vezes demorava tempo a responder, o que muitas das vezes nos levaria em erro.

Na implementação de Serviços REST iam surgindo erros de compilação que não conseguia-mos resolver, consumindo-nos também algum tempo. Nomeadamente na inserção de Feiras no Swagger em que a nossa API desligava sem retornar qualquer erro.

O consumo de tempo durante o desenvolvimento do trabalho fez com que não conseguíssemos implementar a parte de *Web API Security*.

Conclusão

A realização deste Trabalho prático permitiu-nos perceber melhor como funciona o mundo do desenvolvimento de serviços REST, e o que implica aceder a bases de dados hospedadas na Cloud.

Com este trabalho ficamos também mais cientes em relação ás dificuldades e adversidades encontradas na implementação dos serviços, e quais as melhores formas de ultrapassar essas dificuldades. Ficamos com noção das normas a seguir para a publicação de serviços na Cloud, e o que implica essa publicação. No futuro teremos alguma experiência e noção do funcionamento desta implementação.

Durante a realização deste trabalho fomos assimilando conceitos lecionados nas aulas de Engenharia de Integração e Sistemas, e usando essa aprendizagem, fomos ultrapassando dificuldades que nos foram aparecendo.

Por fim, podemos concluir que o trabalho foi uma mais valia no enriquecimento do nosso conhecimento sobre outras metodologias, e foi possível aumentar a nossa capacidade de ultrapassar e contornar adversidades.