APÊNDICE A – CÓDIGO DO SISTEMA DE VISTORIAS

A Figura 3.20 mostra a disposição dos projetos da solução ControleShopping2.0. O resto do código fonte que não está no Apêndice A são estruturas próprias do próprio Visual Studio com o Xamarin para trabalhar com Android, Azure, Web Aplication, SQL Server 2012 e API.

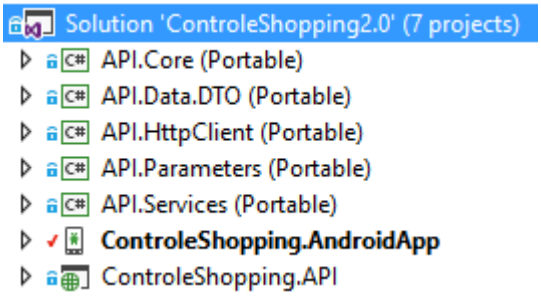

Figura 3.20 – Projetos do código
Fonte: Autor

- Projeto API.Core:
    - Classe ConfigurationManager;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace API.Core
{
   public static class ConfigurationManager
   {
      public static string MaxRequestLenght { get; set; }
      public static class AppSettings
      {
         public static string AccountName { get; set; }
         public static string TokenAccess { get; set; }
         public static string BusinessRulesPath { get { return "/App_Data/"; } }
         public static string ApiUrl {get { return "http://controleshopping.azurewebsites.net"; }
// variavel global que não pode ser alterada. no set
            }
      }
   }
}
```

- Projeto API.Core:
    - Classe ConvertExtensions;

```
using System;
using System.Collections;
```

```
using System.Collections.Generic;
using System.IO;
using System.Linq;

namespace API.Core
{
    public static class ConvertExtensions
    {
        public static double ToDouble(this object value)
        {
            if (value == null)
                return 0.0;
            Double res;
            Double.TryParse(value.ToString(), out res);
            return res;
        }

        public static decimal ToDecimal(this object value)
        {
            if (value == null)
                return 0;
            Decimal res;
            Decimal.TryParse(value.ToString(), out res);
            return res;
        }

        public static byte ToByte(this object value)
        {
            byte res = 0;
            if (value != null)
                byte.TryParse(value.ToString(), out res);
            return res;
        }

        public static int ToInt32(this object value)
        {
            int res = 0;
            if (value != null)
                int.TryParse(value.ToString(), out res);
            return res;
        }

        public static long ToInt64(this object value)
        {
            long res = 0;
            if (value != null)
                long.TryParse(value.ToString(), out res);
            return res;
        }

        public static DateTime? ToDate(this object value)
        {
            try
            {
                return DateTime.Parse(value.ToString());
            }
            catch
            {
                return null;
            }
        }

        public static bool ToBoolean(this object value)
        {
```

```
            bool result = false;
            if (value != null)
            {
                bool.TryParse(value.ToString(), out result);
                return result;
            }
            return false;

        }

        public static IEnumerable<IEnumerable<T>> Split<T>(this T[] array, int size)
        {
            for (var i = 0; i < (float)array.Length / size; i++)
            {
                yield return array.Skip(i * size).Take(size);
            }
        }
    }
}
```

- Projeto API.Core:
    - Classe DataTimeConvert;

```
using Newtonsoft.Json;
using Newtonsoft.Json.Converters;
using System;
using System.Globalization;
using System.Net;

namespace API.Core
{
    public class DateTimeConverter : DateTimeConverterBase
    {
        public override object ReadJson(JsonReader reader, Type objectType, object
existingValue, JsonSerializer serializer)
        {
            string value = reader.Value.ToString();

            /* Verifica se a data foi passada com Html Encode e valida o formato.
             * Exemplo: 21%2F09%2F1977
             */
            if (value.IndexOf(@"%2F") > -1)
            {
                value = WebUtility.UrlDecode(value);
            }

            DateTime date;

            try
            {
                //DateTime.TryParse(value, CultureInfo.InvariantCulture,
DateTimeStyles.AssumeLocal, out date);
                date = DateTime.Parse(value,CultureInfo.CurrentCulture,
DateTimeStyles.AssumeLocal);
            }
            catch (Exception)
            {
                date = new DateTime();
            }
```

```
            return date;
        }

        public override void WriteJson(JsonWriter writer, object value, JsonSerializer
serializer)
        {
            writer.WriteValue(((DateTime)value).ToString("dd/MM/yyyy HH:mm:ss",
CultureInfo.InvariantCulture));
        }
    }

}
```

- Projeto API.Core:
    - Classe FileSet;

```
using System.Collections.Generic;
using System.Dynamic;
using Newtonsoft.Json;

namespace API.Core
{
    public class FileSet
    {
        private dynamic _parameters = new ExpandoObject();

        public string Name { get; set; }

        public dynamic Parameters
        {
            get { return _parameters; }
            set { _parameters = value; }
        }

        [JsonIgnore]
        public IList<File> Files { get; set; }
    }

    public class File
    {
        public string Name { get; set; }

        public string MimeType { get; set; }

        public byte[] Content { get; set; }
    }
}
```

- Projeto API.Core:
    - Classe FormatExtensions;

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace API.Core
{
    public static class FormatExtensions
    {
        /// <summary>
        /// Format date in dd/MM/yyyy (pt-BR)
        /// </summary>
        /// <param name="value">Date</param>
        /// <returns>Formated date</returns>
        public static string FormatPT(this DateTime value)
        {
            return ((DateTime)value).ToString("dd/MM/yyyy");
        }

        public static DateTime ParseDateTime(this string date)
        {
            string dayOfWeek = date.Substring(0, 3).Trim();
            string month = date.Substring(4, 3).Trim();
            string dayInMonth = date.Substring(8, 2).Trim();
            string time = date.Substring(11, 9).Trim();
            string offset = date.Substring(20, 5).Trim();
            string year = date.Substring(25, 5).Trim();
            string dateTime = string.Format("{0}-{1}-{2} {3}", dayInMonth, month, year, time);
            DateTime ret = DateTime.Parse(dateTime);
            return ret;
        }

        public static string Cut(this string value, int maxLenght)
        {
            if (value == null)
                return null;

            if (value.Length > maxLenght)
                return value.Substring(0, maxLenght - 1);
            else
                return value;
        }

        public static string StripTagsCharArray(this string source)
        {
            char[] array = new char[source.Length];
            int arrayIndex = 0;
            bool inside = false;

            for (int i = 0; i < source.Length; i++)
            {
                char let = source[i];
                if (let == '<')
                {
                    inside = true;
                    continue;
                }
                if (let == '>')
                {
                    inside = false;
                    continue;
                }
                if (!inside)
                {
                    array[arrayIndex] = let;
```

```
                arrayIndex++;
            }
        }
        return new string(array, 0, arrayIndex);
    }
  }
}
```

- Projeto API.Core:
    - Classe JsonSettings;

```
using Newtonsoft.Json;

namespace API.Core
{
  public class JsonSettings
  {
    public static JsonSerializerSettings GetSerializerSettings()
    {
      var settings = new JsonSerializerSettings();

      settings.Formatting = Formatting.Indented;
      settings.DateTimeZoneHandling = DateTimeZoneHandling.Utc;

      settings.Converters.Add(new DateTimeConverter());

      return settings;
    }
  }
}
```

- Projeto API.Core:
    - Classe ParameterException;

```
using System;

namespace API.Core
{
  public class ParameterException : Exception
  {
    public ParameterException(string parameterName, object parameterValue,
TypeParameterException type)
      : base(FormatMessage(parameterName, parameterValue, type))
    {
    }

    public ParameterException(string parameterName, object parameterValue)
      : base(FormatMessage(parameterName, parameterValue,
TypeParameterException.Invalid))
    {
    }
```

```
        public ParameterException(string parameterName, TypeParameterException type)
            : base(FormatMessage(parameterName, null, type))
        {
        }

        public ParameterException(string message)
            : base(message)
        {
        }

        private static string FormatMessage(string parameterName, object parameterValue,
TypeParameterException type)
        {
            switch (type)
            {
                case TypeParameterException.IsNullOrEmpty:
                    return string.Concat("O parâmetro \"{0}\" não pode ser {1}.", parameterName,
parameterValue == null ? "nulo" : "vazio");
                case TypeParameterException.IsNull:
                    return string.Concat("O parâmetro \"{0}\" não pode ser nulo.",
parameterName);
                case TypeParameterException.Invalid:
                    return string.Format("O parâmetro \"{0}\" com valor \"{1}\" é inválido.",
parameterName, parameterValue);
                default:
                    throw new Exception("Invalid type of parameter exception.");
            }
        }
    }
}
```

- Projeto API.Core:
    - Declaração de Enumeração TypeParameterException;

```
namespace API.Core
{
    public enum TypeParameterException
    {
        IsNullOrEmpty,
        Invalid,
        IsNull
    }
}
```

- Projeto API.Data.DTO:
    - Classe GrupoUsuario;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace API.Data.DTO
{
   public class GrupoUsuario
   {
      public GrupoUsuario()
      {
         this.Verificacao = new HashSet<Verificacao>();
         this.Usuario = new HashSet<Usuario>();
      }

      public int idGrupoUsuario { get; set; }
      public string deGrupoUsuario { get; set; }

      public virtual ICollection<Verificacao> Verificacao { get; set; }
      public virtual ICollection<Usuario> Usuario { get; set; }
   }
}
```

- Projeto API.Data.DTO:
    - Classe HistoricoVerificacao;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace API.Data.DTO
{
   public class HistoricoVerificacao
   {
      public int idHistoricoVirificacao { get; set; }
      public int idVerificacao { get; set; }
      public int idUsuario { get; set; }
      public System.DateTime dtVerificacao { get; set; }

      public virtual Usuario Usuario { get; set; }
      public virtual Verificacao Verificacao { get; set; }
   }
}
```

- Projeto API.Data.DTO:
    - Classe ProblemaVerificacao;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace API.Data.DTO
```

```
{
   public class ProblemaVerificacao
   {
      public int idProblemaVerificacao { get; set; }
      public int idVerificacao { get; set; }
      public string deProblemaVerificacao { get; set; }
      public byte[] imgProblema { get; set; }
      public System.DateTime dtProblemaVerificacao { get; set; }
      public virtual Verificacao Verificacao { get; set; }
   }
}
```

- Projeto API.Data.DTO:
    - Classe Produto;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace API.Data.DTO
{
   public class Produto
   {
      public Produto()
      {
         this.Verificacao = new HashSet<Verificacao>();
      }

      public int idProduto { get; set; }
      public int idSetor { get; set; }
      public string deProduto { get; set; }
      public string deLocalizacao { get; set; }
      public byte[] coQrCode { get; set; }

      public virtual Setor Setor { get; set; }
      public virtual ICollection<Verificacao> Verificacao { get; set; }
   }
}
```

- Projeto API.Data.DTO:
    - Classe Setor;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace API.Data.DTO
{
   public class Setor
   {
```

```
        public Setor()
        {
            this.Produto = new HashSet<Produto>();
        }

        public int idSetor { get; set; }
        public string deSetor { get; set; }

        public virtual ICollection<Produto> Produto { get; set; }
    }
}
```

- Projeto API.Data.DTO:
    - Classe TipoVerificacao;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace API.Data.DTO
{
    public class TipoVerificacao
    {
        public TipoVerificacao()
        {
            this.Verificacao = new HashSet<Verificacao>();
        }

        public int idTipoVerificacao { get; set; }
        public string deTipoVerificacao { get; set; }

        public virtual ICollection<Verificacao> Verificacao { get; set; }
    }
}
```

- Projeto API.Data.DTO:
    - Classe Usuario;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace API.Data.DTO
{
    public class Usuario
    {
        public int idUsuario { get; set; }
        public string nmUsuario { get; set; }
        public string coSenha { get; set; }
        public bool icAdministrador { get; set; }
```

```
        public string edEmail { get; set; }

        public virtual ICollection<HistoricoVerificacao> HistoricoVerificacao { get; set; }

        public virtual ICollection<GrupoUsuario> GrupoUsuario { get; set; }
    }
}
```

- Projeto API.Data.DTO:
    - Classe Verificacao;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace API.Data.DTO
{
    public class Verificacao
    {
        public int idVerificacao { get; set; }
        public int idGrupoUsuario { get; set; }
        public int idProduto { get; set; }
        public int idTipoVerificacao { get; set; }
        public string deVerificacao { get; set; }

        public virtual GrupoUsuario GrupoUsuario { get; set; }
        //public virtual ICollection<HistoricoVirificacao> HistoricoVirificacao { get; set; }
        public virtual Produto Produto { get; set; }
        public virtual TipoVerificacao TipoVerificacao { get; set; }
    }
}
```

- Projeto API.HttpClient:
    - Classe ApiException;

```
using API.HttpClient.Proxy;
using System;

namespace API.HttpClient
{
    public class ApiException : Exception
    {
        private string _id;
        private string _description;

        public string ID {
            get { return _description; }
        }

        public string Description
        {
            get { return _description; }
```

```
        }

        public ApiException(ResponseMessageProxy message)
            : base(message.description)
        {
            _id = message.id;
            _description = message.description;
        }
    }
}
```

- Projeto API.HttpClient:
    - Classe Headers;

```
using System;
using System.Globalization;
using System.Resources;
using System.Text;
using System.Reflection;

namespace API.HttpClient
{
    public sealed class Headers
    {
        public static string GetAuthenticationString(string accountName, string tokenAccess)
        {
            return string.Concat("Basic ",
Convert.ToBase64String(Encoding.UTF8.GetBytes(string.Format("{0}:{1}", accountName,
tokenAccess))));
        }

        public static string GetTimestamp()
        {
            return DateTime.UtcNow.ToString("U", CultureInfo.InvariantCulture);
        }

        public static string DecodeAuthenticationString(string
authenticateUsingHttpBasicAuthentication)
        {
            authenticateUsingHttpBasicAuthentication =
authenticateUsingHttpBasicAuthentication.Replace("Basic ", "");

            byte[] encodedDataAsBytes =
System.Convert.FromBase64String(authenticateUsingHttpBasicAuthentication);

            string returnValue =
System.Text.Encoding.UTF8.GetString(encodedDataAsBytes,0,encodedDataAsBytes.Len
gth);

            return returnValue;
        }
    }
}
```

- Projeto API.HttpClient:

- Classe HttpClientBase;

```
using API.Core;
using API.HttpClient.Proxy;
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Dynamic;
using System.Linq;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Text;
using System.Threading.Tasks;
using System.Reflection;
using System.Net;
using static API.Core.ConfigurationManager;
using Android.Widget;

namespace API.HttpClient
{
    public abstract class HttpClientBase
    {
        private System.Net.Http.HttpClient _httpClient = new System.Net.Http.HttpClient();

        private dynamic _parameters = new ExpandoObject();

        protected System.Net.Http.HttpClient Client { get { return _httpClient; } }

        protected string RequestUri { get; set; }

        protected dynamic Parameters { get { return _parameters; } }

        protected HttpClientBase(string requestUri) //sempre deve ter a string de requisição
        {
            RequestUri = string.Format("api/{0}", requestUri); // para onde ele vai aopontar na
api
            Client.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue("application/json"));
        }

        protected async Task<ResponseProxy<TResult>> ExecuteRequest<T1, TResult>(T1
param, HttpMethod method = HttpMethod.Get) //Classe assincrona task, que retorna
reponse proxy
            where TResult : class
            where T1 : class
        {
            try
            {
                var parameters = new StringContent("");
                string queryString = "";
                switch (method)
                {
                    case HttpMethod.Get:
                        queryString = GetQueryString(param);
                        return await GetRequest<TResult>(queryString);

                    case HttpMethod.Delete:
                        queryString = GetQueryString(param);
                        return await DeleteRequest<TResult>(queryString);

                    case HttpMethod.Post:
                        parameters = GetStringContent(param);
                        return await PostRequest<TResult>(parameters);
```

```
                case HttpMethod.Put:
                    parameters = GetStringContent(param);
                    return await PutRequest<TResult>(parameters);

                default:
                    return new ResponseProxy<TResult>();
            }
        }
        catch (Exception ex)
        {
            var result = new ResponseProxy<TResult>();
            result.message.description = ex.Message;
            return result;
        }

    }

    protected ResponseProxy<T> UploadResponse<T>(FileSet fileSet, Dictionary<string,
object> queryStringParameters = null)
        where T : class
    {
        var multipartContent = new MultipartFormDataContent();

        var postedFilesJson = JsonConvert.SerializeObject(fileSet,
JsonSettings.GetSerializerSettings());

        multipartContent.Add(new StringContent(postedFilesJson, Encoding.UTF8,
"application/json"), "fileset");

        int counter = 0;

        foreach (var file in fileSet.Files)
        {
            var fileContent = new ByteArrayContent(file.Content);
            fileContent.Headers.ContentType = new
MediaTypeHeaderValue(file.MimeType);
            multipartContent.Add(fileContent, "file" + counter++, file.Name);
        }

        var httpClient = new System.Net.Http.HttpClient();


        string queryString = "";

        if (!string.IsNullOrEmpty(queryString) && queryStringParameters.Count > 0)
        {
            queryString = string.Format("?{0}", GetQueryString(queryStringParameters));
        }

var uri = new Uri(string.Format("{0}/{1}{2}", AppSettings.ApiUrl, RequestUri, queryString));

        var responseMessage = httpClient.PostAsync(uri.ToString(),
multipartContent).Result;

        var response = responseMessage.Content.ReadAsStringAsync().Result;

        return JsonConvert.DeserializeObject<ResponseProxy<T>>(response,
JsonSettings.GetSerializerSettings());
    }

    /// <summary>
    /// Método utilizado para enviar requisições POST para o barramento de serviço.
```

```
        /// </summary>
        /// <typeparam name="T1"></typeparam>
        /// <typeparam name="TResult"></typeparam>
        /// <param name="parameter"></param>
        /// <returns></returns>
        protected async Task<ResponseProxy<TResult>>
PostRequest<TResult>(StringContent stringContent) // PostRequest tipo de retorno T1 e
TResult.
            where TResult : class
        {
            using (var client = new System.Net.Http.HttpClient())
            {


                var responseString = string.Empty; // Usar string vazio de inicio

                client.BaseAddress = new Uri(AppSettings.ApiUrl);  // vai no appsettings pegar a
ApiURl e setar como uma nova url, atribuindo a client.base Address


                client.DefaultRequestHeaders.Accept.Clear(); // formatos que o client aceita.
Primeiro limpa
                client.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue("application/json")); // adiciona um novo de tal forma

                HttpRequestMessage request = new
HttpRequestMessage(System.Net.Http.HttpMethod.Post,RequestUri);
                request.Content = stringContent;
                var response = await client.SendAsync(request).ConfigureAwait(false);

                if (response.IsSuccessStatusCode)
                {
                    responseString = await response.Content.ReadAsStringAsync(); // retorno dos
dados em string
                }

                var responseProxy = await Task.Factory.StartNew(() =>
JsonConvert.DeserializeObject<ResponseProxy<TResult>>(responseString,
JsonSettings.GetSerializerSettings())); // explicar o funcionamento, que está
deseraliazando um objeto

                return responseProxy; // retorno já no formato necessário. objeto TResult.
retorna do tipo responseProxy, nele tem o data, que retornara tresult no caso
            }
        }

        /// <summary>
        /// Método utilizado para enviar requisições PUT para o barramento de serviço.
        /// </summary>
        /// <typeparam name="T1">TIpo do parametro</typeparam>
        /// <typeparam name="TResult">Tipo do retorno</typeparam>
        /// <param name="parameter">objeto parameter utilizado para </param>
        /// <returns></returns>
        protected async Task<ResponseProxy<TResult>>
PutRequest<TResult>(StringContent stringContent)
            where TResult : class
        {
            using (var client = new System.Net.Http.HttpClient())
            {
                try
                {

                    var responseString = string.Empty;
```

```
                client.BaseAddress = new Uri(AppSettings.ApiUrl);


                client.DefaultRequestHeaders.Accept.Clear();
                client.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue("application/json"));

                HttpRequestMessage request = new
HttpRequestMessage(System.Net.Http.HttpMethod.Post, RequestUri);
                request.Content = stringContent;
                var response = await client.SendAsync(request).ConfigureAwait(false);
                if (response.IsSuccessStatusCode)
                {
                    responseString = await response.Content.ReadAsStringAsync();
                }

                var responseProxy = await Task.Factory.StartNew(() =>
JsonConvert.DeserializeObject<ResponseProxy<TResult>>(responseString,
JsonSettings.GetSerializerSettings()));

                return responseProxy;
            }
            catch (HttpRequestException EX)
            {
                throw new ApiException(new ResponseMessageProxy
                {
                    description = EX.Message,
                    id = EX.Source.ToString()
                });
            }
        }
    }

    /// <summary>
    /// Método utilizado para enviar requisições GET para o barramento de serviço.
    /// </summary>
    /// <typeparam name="TResult">Tipo do retorno</typeparam>
    /// <param name="parameter">objeto parameter utilizado para </param>
    /// <returns></returns>
    protected async Task<ResponseProxy<TResult>> GetRequest<TResult>(string
QueryString)
        where TResult : class
    {
        var queryString = string.Format("?{0}", QueryString);
        var responseString = "";
        using (var client = new System.Net.Http.HttpClient())
        {
            client.BaseAddress = new Uri(string.Format("{0}/{1}{2}", AppSettings.ApiUrl,
RequestUri, queryString));

            client.DefaultRequestHeaders.Accept.Clear();
            client.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue("application/json"));

            var response = await client.GetAsync(client.BaseAddress);
            if (response.IsSuccessStatusCode)
            {
                responseString = await response.Content.ReadAsStringAsync();
            }
            var responseProxy = new ResponseProxy<TResult>();
            try
            {
```

```
                responseProxy =
JsonConvert.DeserializeObject<ResponseProxy<TResult>>(responseString,
JsonSettings.GetSerializerSettings());
            }catch(JsonException Jex)
            {
                throw Jex;
            }
            return responseProxy;
        }
    }

    /// <summary>
    /// Método utilizado para enviar requisições DELETE para o barramento de serviço.
    /// </summary>
    /// <typeparam name="TResult">Tipo do retorno</typeparam>
    /// <param name="parameter">objeto parameter utilizado para </param>
    /// <returns></returns>
    protected async Task<ResponseProxy<TResult>> DeleteRequest<TResult>(string
parameter)
        where TResult : class
    {
        var queryString = string.Format("?{0}", parameter);
        var responseString = "";
        using (var client = new System.Net.Http.HttpClient())
        {
            client.BaseAddress = new Uri(string.Format("{0}/{1}{2}", AppSettings.ApiUrl,
RequestUri, queryString));

            client.DefaultRequestHeaders.Accept.Clear();
            client.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue("application/json"));

            var response = await
client.DeleteAsync(client.BaseAddress).ConfigureAwait(false);
            if (response.IsSuccessStatusCode)
            {
                responseString = await response.Content.ReadAsStringAsync();
            }

            var responseProxy = await Task.Factory.StartNew(() =>
JsonConvert.DeserializeObject<ResponseProxy<TResult>>(responseString,
JsonSettings.GetSerializerSettings()));

            return responseProxy;
        }
    }
    #region Construtores
    private StringContent GetStringContent(object obj)
    {
        var jsonObject = JsonConvert.SerializeObject(obj);
        StringContent result = new StringContent(jsonObject);
        return result;
    }

    public string GetQueryString(object obj)
    {
        try
        {
            var properties = from p in obj.GetType().GetRuntimeProperties()
                             where p.GetValue(obj, null) != null
                             select p.Name + "=" + WebUtility.UrlEncode(p.GetValue(obj,
null).ToString());

            return String.Join("&", properties.ToArray());
```

```
            }catch(Exception e)
            { throw e; }
        }
        #endregion
    }


}
```

- Projeto API.Parameters:
    - Classe GrupoUsuarioParameter;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace API.Parameters
{
    public class GrupoUsuarioParameter : ParameterBase // Sempre é pasado o parametro
dessa forma para a API.
    {
        public int idGrupoUsuario { get; set; } // Objeto que pode tanto atribuir quanto setar o
valor
        public int idUsuario { get; set; }
        public string GrupoUsuarioJSON { get; set; }

        public enum Operacao //tipos de operações que pode ter
        {
            Get = 1 ,
            GetList = 2,
            GetListByUsuario = 3,
            Create  = 15
        }
    }
}
```

- Projeto API.Parameters:
    - Classe HistoricoVerificacaoParameter;

```
using API.Data.DTO;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace API.Parameters
{
    public class HistoricoVerificacaoParameter : ParameterBase
    {
        public int idHistoricoVerificacao { get; set; }
        public string HistoricoVerificacaoJSON { get; set; }

        public enum Operacao
```

```
        {
            Get = 1,
            GetList = 2,
            GetListByUsuario = 3,
            Create = 15
        }
    }
}
```

- Projeto API.Parameters:
    - Classe ParameterBase;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace API.Parameters
{
    public class ParameterBase
    {
        public string includeProperties { get; set; }
        public int nuPagina { get; set; }
        public int nuResultados { get; set; }
        public int idOperacao { get; set; }
    }
}
```

- Projeto API.Parameters:
    - Classe ProblemaVerificacaoParameter;

```
using API.Data.DTO;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace API.Parameters
{
    public class ProblemaVerificacaoParameter : ParameterBase
    {
        public int idProblemaVerificacao { get; set; }
        public ProblemaVerificacao ProblemaVerificacao { get; set; }
        public string ProblemaVerificacaoJSON { get; set; }

        public enum Operacao
        {
            Get = 1,
            GetList = 2,
            GetListByVerificacao = 3,
            Create = 15,
```

```
    }
  }
}
```

- Projeto API.Parameters:
    - Classe ProdutoParameter;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace API.Parameters
{
  public class ProdutoParameter:ParameterBase
  {
    public int idProduto { get; set; }
    public string ProdutoJSON { get; set; }

    public enum Operacao
    {
      Get = 1,
      GetList =2,
      Create = 15
    }
  }
}
```

- Projeto API.Parameters:
    - Classe SetorParameter;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace API.Parameters
{
  public class SetorParameter : ParameterBase
  {
    public int idSetor { get; set; }
    public string SetorJSON { get; set; }
    public enum Operacao
    {
      Get = 1,
      GetList = 2,
      Create = 15
    }
  }
}
```

- Projeto API.Parameters:
    - Classe TipoVerificacaoParameter;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace API.Parameters
{
   public class TipoVerificacaoParameter : ParameterBase
   {
      public int idTipoVerificacao { get; set; }
      public string TipoVerificacaoJSON { get; set; }
      public enum Operacao
      {
         Get = 1,
         GetList = 2,
         Create = 15
      }
   }
}
```

- Projeto API.Parameters:
    - Classe UsuarioParameter;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;


namespace API.Parameters
{
   public class UsuarioParameter : ParameterBase
   {
      public int idUsuario { get; set; }
      public string edEmail { get; set; }
      public string nmUsuario { get; set; }
      public string coSenha { get; set; }
      public string UsuarioJSON { get; set; }
      public enum Operacao
      {
         Get = 1,
         GetByEmailSenha = 2,
         GetList = 3,
         IncluirUsuario = 20,
         UpdateUsuario = 21
      }
   }
}
```

- Projeto API.Parameters:
    - Classe VerificacaoParameter;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace API.Parameters
{
   public class VerificacaoParameter : ParameterBase
   {
      public int idVerificacao { get; set; }
      public int idGrupoUsuario { get; set; }
      public string VerificacaoJSON { get; set; }
      public enum Operacao
      {
         Get = 1,
         GetList = 2,
         GetListByGrupoUsuario = 3,
         Create = 20,
         Update = 21
      }
   }
}
```

- Projeto API.Services:
    - Classe GrupoUsuarioAPIServices;

```
using API.Data.DTO;
using API.HttpClient;
using API.HttpClient.Proxy;
using API.Parameters;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using static API.Parameters.GrupoUsuarioParameter;

namespace API.Services
{
   public class GrupoUsuarioAPIServices : HttpClientBase,
IAPIServicesRepository<GrupoUsuario, GrupoUsuarioParameter> // A requisição do
GrupoUsuario herda de HttpclientBase, utilizando a interface IAPIServiceRepository,
utilizando os dois tipos de retorno.
   {
      public GrupoUsuarioAPIServices
         (
            Operacao operacao, //lista de nome que equivalem um número. deixa o
aplicativo mais legivel. Para não ter que trabalhar com vairos tipos de nomes de
operações.
            int? idGrupoUsuario = null, //colocar interrogação nos valores primitivos para
atribuir valores nulos.
```

```
            int? idUsuario = null, //colocar interrogação nos valores primitivos para atribuir
valores nulos.
            string includeProperties = ""
        ) : base("GrupoUsuario") // atribuir GrupoUsuarioAPIServices a base
grupoUguario. Se quiser herdar qlqr coisa utilizar o BASe
    {
        Parameters.idOperacao = (int)operacao;
        Parameters.idGrupoUsuario = idGrupoUsuario;
        Parameters.idUsuario = idUsuario;
        Parameters.includeProperties = includeProperties;
    }


    //Sempre necessita ter todas essas funções abaixo quando está utilizando uma
interface.
    public bool Delete()
    {
        throw new NotImplementedException();
    }

    public bool Delete(out object data)
    {
        throw new NotImplementedException();
    }

    public async Task<GrupoUsuario> Get(GrupoUsuarioParameter parameter)
    {
        var result = await ExecuteRequest<GrupoUsuarioParameter,
GrupoUsuario>(parameter, HttpMethod.Get);
        if (result.status == ResponseStatus.Success)
        {
            return result.data;
        }
        else
        {
            throw new Exception(result.message.description);
        }
    }

    public async Task<IList<GrupoUsuario>> GetList(GrupoUsuarioParameter
parameter)
    {
        var result = await ExecuteRequest<GrupoUsuarioParameter,
List<GrupoUsuario>>(parameter,HttpMethod.Get);
        if (result.status == ResponseStatus.Success)
        {
            return result.data;
        }
        else
        {
            throw new Exception(result.message.description);
        }
    }

    public async Task<GrupoUsuario> Post(GrupoUsuarioParameter parameter)
    {
        var result = await ExecuteRequest<GrupoUsuarioParameter,
GrupoUsuario>(parameter,HttpMethod.Post);
        if (result.status == ResponseStatus.Success)
            return result.data;
        else
            throw new Exception(result.message.description);
    }
```

```
        public async Task<GrupoUsuario> Update(GrupoUsuarioParameter parameter)
        {
            var result = await ExecuteRequest<GrupoUsuarioParameter,
GrupoUsuario>(parameter, HttpMethod.Put);
            if (result.status == ResponseStatus.Success)
                return result.data;
            else
                throw new Exception(result.message.description);
        }
    }
}
```

- Projeto API.Services:
    - Classe HistoricoVerificacaoAPIServices;

```
using API.Data.DTO;
using API.HttpClient;
using API.HttpClient.Proxy;
using API.Parameters;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using static API.Parameters.HistoricoVerificacaoParameter;

namespace API.Services
{
    public class HistoricoVerificacaoAPIServices : HttpClientBase,
IAPIServicesRepository<HistoricoVerificacao, HistoricoVerificacaoParameter>
    {
        public HistoricoVerificacaoAPIServices
            (
                Operacao operacao,
                int? idHistoricoVerificacao = null
            ) : base("HistoricoVerificacao")
        {
            Parameters.idOperacao = (int)operacao;

            Parameters.idHistoricoVerificacao = idHistoricoVerificacao;
        }

        public bool Delete()
        {
            throw new NotImplementedException();
        }

        public bool Delete(out object data)
        {
            throw new NotImplementedException();
        }

        public async Task<HistoricoVerificacao> Get(HistoricoVerificacaoParameter
parameters)
        {
            var result = await ExecuteRequest<HistoricoVerificacaoParameter,
HistoricoVerificacao>(parameters,HttpMethod.Get);
```

```
            if (result.status == ResponseStatus.Success)
            {
                return result.data;
            }
            else
            {
                throw new Exception(result.message.description);
            }
        }

        public async Task<IList<HistoricoVerificacao>>
GetList(HistoricoVerificacaoParameter parameters)
        {
            var result = await ExecuteRequest<HistoricoVerificacaoParameter,
List<HistoricoVerificacao>>(parameters, HttpMethod.Get);
            if (result.status == ResponseStatus.Success)
            {
                return result.data;
            }
            else
            {
                throw new Exception(result.message.description);
            }
        }

        public async Task<HistoricoVerificacao> Post(HistoricoVerificacaoParameter
parameter)
        {
            var result = await ExecuteRequest<HistoricoVerificacaoParameter,
HistoricoVerificacao>(parameter,HttpMethod.Post);
            if (result.status == ResponseStatus.Success)
                return result.data;
            else
                throw new Exception(result.message.description);
        }

        public async Task<HistoricoVerificacao> Update(HistoricoVerificacaoParameter
parameter)
        {
            var result = await ExecuteRequest<HistoricoVerificacaoParameter,
HistoricoVerificacao>(parameter,HttpMethod.Put);
            if (result.status == ResponseStatus.Success)
                return result.data;
            else
throw new Exception(result.message.description);
}
}
}
```

- Projeto API.Services:
    - Classe IAPIServicesRepository;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace API.Services
```

```
{
   public interface IAPIServicesRepository<TResult, TParameter>
      where TResult : class
       where TParameter : class
   {
       Task<TResult> Get(TParameter parameter);
       Task<IList<TResult>> GetList(TParameter parameter);
       Task<TResult> Post(TParameter parameter);
       Task<TResult> Update(TParameter parameter);
       bool Delete();
       bool Delete(out object data);
   }
}
```

- Projeto API.Services:
    - Classe ProblemaVerificacaoAPIServices;

```
using API.Data.DTO;
using API.HttpClient;
using API.HttpClient.Proxy;
using API.Parameters;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using static API.Parameters.ProblemaVerificacaoParameter;

namespace API.Services
{
   public class ProblemaVerificacaoAPIServices : HttpClientBase,
IAPIServicesRepository<ProblemaVerificacao, ProblemaVerificacaoParameter>
   {
       public ProblemaVerificacaoAPIServices
          (
             Operacao operacao,
             int? idProblemaVerificacao = null
          ) : base("ProblemaVerificacao")
       {

       }

       public bool Delete()
       {
          throw new NotImplementedException();
       }

       public bool Delete(out object data)
       {
          throw new NotImplementedException();
       }

       public async Task<ProblemaVerificacao> Get(ProblemaVerificacaoParameter
parameters)
       {
          var result = await
ExecuteRequest<ProblemaVerificacaoParameter,ProblemaVerificacao>(parameters,HttpM
ethod.Get);
          if (result.status == ResponseStatus.Success)
```

```
        {
            return result.data;
        }
        else
        {
            throw new Exception(result.message.description);
        }
    }

    public async Task<IList<ProblemaVerificacao>>
GetList(ProblemaVerificacaoParameter parameters)
    {
        var result = await
ExecuteRequest<ProblemaVerificacaoParameter,List<ProblemaVerificacao>>(parameters
,HttpMethod.Get);
        if (result.status == ResponseStatus.Success)
        {
            return result.data;
        }
        else
        {
            throw new Exception(result.message.description);
        }
    }

    public async Task<ProblemaVerificacao> Post(ProblemaVerificacaoParameter
parameter)
    {
        var result = await ExecuteRequest<ProblemaVerificacaoParameter,
ProblemaVerificacao>(parameter,HttpMethod.Post);
        if (result.status == ResponseStatus.Success)
            return result.data;
        else
            throw new Exception(result.message.description);
    }

    public async Task<ProblemaVerificacao> Update(ProblemaVerificacaoParameter
parameter)
    {
        var result = await ExecuteRequest<ProblemaVerificacaoParameter,
ProblemaVerificacao>(parameter, HttpMethod.Put);
        if (result.status == ResponseStatus.Success)
            return result.data;
        else
            throw new Exception(result.message.description);
    }
  }
}
```

- Projeto API.Services:
    - Classe ProdutoAPIServices;

```
using API.Data.DTO;
using API.HttpClient;
using API.HttpClient.Proxy;
using API.Parameters;
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Text;
using System.Threading.Tasks;
using static API.Parameters.ProdutoParameter;

namespace API.Services
{
   public class ProdutoAPIServices : HttpClientBase, IAPIServicesRepository<Produto,
ProdutoParameter>
   {
      public ProdutoAPIServices
         (
            Operacao operacao,
            int? idProduto = null,
            string includeProperties = ""
         ) : base("Produto")
      {
         Parameters.idOperacao = (int)operacao;
         Parameters.idProduto = idProduto;
         Parameters.includeProperties = includeProperties;
      }

      public bool Delete()
      {
         throw new NotImplementedException();
      }

      public bool Delete(out object data)
      {
         throw new NotImplementedException();
      }

      public async Task<Produto> Get(ProdutoParameter parameters)
      {
         var result = await ExecuteRequest<ProdutoParameter,
Produto>(parameters,HttpMethod.Get);
         if (result.status == ResponseStatus.Success)
         {
            return result.data;
         }
         else
         {
            throw new Exception(result.message.description);
         }
      }

      public async Task<IList<Produto>> GetList(ProdutoParameter parameters)
      {
         var result = await
ExecuteRequest<ProdutoParameter,List<Produto>>(parameters,HttpMethod.Get);
         if (result.status == ResponseStatus.Success)
         {
            return result.data;
         }
         else
         {
            throw new Exception(result.message.description);
         }
      }

      public async Task<Produto> Post(ProdutoParameter parameter)
      {
         var result = await ExecuteRequest<ProdutoParameter,
Produto>(parameter,HttpMethod.Post);
         if (result.status == ResponseStatus.Success)
```

```
            return result.data;
        else
            throw new Exception(result.message.description);
    }

    public async Task<Produto> Update(ProdutoParameter parameter)
    {
        var result = await ExecuteRequest<ProdutoParameter,
Produto>(parameter,HttpMethod.Put);
        if (result.status == ResponseStatus.Success)
            return result.data;
        else
            throw new Exception(result.message.description);
    }
  }
}
```

- Projeto API.Services:
    - Classe SetorAPIServices;

```
using API.Data.DTO;
using API.HttpClient;
using API.HttpClient.Proxy;
using API.Parameters;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using static API.Parameters.SetorParameter;

namespace API.Services
{
  public class SetorAPIServices : HttpClientBase, IAPIServicesRepository<Setor,
SetorParameter>
  {
    public SetorAPIServices
      (
        Operacao operacao,
        int?  idSetor = null,
        string includeProperties = ""
      ) : base("Setor")
    {
      Parameters.idOperacao = (int)operacao;
      Parameters.idSetor = idSetor;
      Parameters.includeProperties = includeProperties;
    }

    public bool Delete()
    {
      throw new NotImplementedException();
    }

    public bool Delete(out object data)
    {
      throw new NotImplementedException();
    }

    public async Task<Setor> Get(SetorParameter parameters)
```

```
        {
            var result = await
ExecuteRequest<SetorParameter,Setor>(parameters,HttpMethod.Get);
            if (result.status == ResponseStatus.Success)
            {
                return result.data;
            }
            else
            {
                throw new Exception(result.message.description);
            }
        }

        public async Task<IList<Setor>> GetList(SetorParameter parameters)
        {
            var result = await ExecuteRequest<SetorParameter,
List<Setor>>(parameters,HttpMethod.Get);
            if (result.status == ResponseStatus.Success)
            {
                return result.data;
            }
            else
            {
                throw new Exception(result.message.description);
            }
        }

        public async Task<Setor> Post(SetorParameter parameter)
        {
            var result = await ExecuteRequest<SetorParameter,
Setor>(parameter,HttpMethod.Post);
            if (result.status == ResponseStatus.Success)
                return result.data;
            else
                throw new Exception(result.message.description);
        }

        public async Task<Setor> Update(SetorParameter parameter)
        {
            var result = await ExecuteRequest<SetorParameter,
Setor>(parameter,HttpMethod.Post);
            if (result.status == ResponseStatus.Success)
                return result.data;
            else
                throw new Exception(result.message.description);
        }
    }
}
```

- Projeto API.Services:
    - Classe TipoVerificacaoAPIServices;

```
using API.Data.DTO;
using API.HttpClient;
using API.HttpClient.Proxy;
using API.Parameters;
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Text;
using System.Threading.Tasks;
using static API.Parameters.TipoVerificacaoParameter;

namespace API.Services
{
   public class TipoVerificacaoAPIServices : HttpClientBase,
IAPIServicesRepository<TipoVerificacao, TipoVerificacaoParameter>
   {
      public TipoVerificacaoAPIServices
         (
            Operacao operacao,
            int? idTipoVerificacao = null,
            string includeProperties = ""
         ) : base("TipoVerificacao")
      {
         Parameters.idOperacao = (int)operacao;
         Parameters.idTipoVerificacao = idTipoVerificacao;
         Parameters.includeProperties = includeProperties;
      }

      public bool Delete()
      {
         throw new NotImplementedException();
      }

      public bool Delete(out object data)
      {
         throw new NotImplementedException();
      }

      public async Task<TipoVerificacao> Get(TipoVerificacaoParameter parameters)
      {
         var result = await
ExecuteRequest<TipoVerificacaoParameter,TipoVerificacao>(parameters,HttpMethod.Get
);
         if (result.status == ResponseStatus.Success)
         {
            return result.data;
         }
         else
         {
            throw new Exception(result.message.description);
         }
      }

      public async Task<IList<TipoVerificacao>> GetList(TipoVerificacaoParameter
parameters)
      {
         var result = await
ExecuteRequest<TipoVerificacaoParameter,List<TipoVerificacao>>(parameters,HttpMetho
d.Get);
         if (result.status == ResponseStatus.Success)
         {
            return result.data;
         }
         else
         {
            throw new Exception(result.message.description);
         }
      }

      public async Task<TipoVerificacao> Post(TipoVerificacaoParameter parameter)
      {
```

```
        var result = await ExecuteRequest<TipoVerificacaoParameter,
TipoVerificacao>(parameter);
        if (result.status == ResponseStatus.Success)
            return result.data;
        else
            throw new Exception(result.message.description);
    }

    public async Task<TipoVerificacao> Update(TipoVerificacaoParameter parameter)
    {
        var result = await ExecuteRequest<TipoVerificacaoParameter,
TipoVerificacao>(parameter);
        if (result.status == ResponseStatus.Success)
            return result.data;
        else
            throw new Exception(result.message.description);
    }
  }
}
```

- Projeto API.Services:
    - Classe UsuarioAPIServices;

```
using API.Data.DTO;
using API.HttpClient;
using API.HttpClient.Proxy;
using API.Parameters;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using static API.Parameters.UsuarioParameter;

namespace API.Services
{
  public class UsuarioAPIServices : HttpClientBase, IAPIServicesRepository<Usuario,
UsuarioParameter>
  {

    public UsuarioAPIServices
      (
        Operacao idOperacao,
        int? idUsuario = null,
        string edEmail = "",
        string coSenha = "",
        string includeProperties = ""
      ) : base("Usuario")
    {
      Parameters.edEmail = edEmail;
      Parameters.coSenha = coSenha;
      Parameters.includeProperties = includeProperties;
      Parameters.idUsuario = idUsuario;

      Parameters.idOperacao = (int)idOperacao;
    }

    public bool Delete()
    {
```

```
            throw new NotImplementedException();
        }

        public bool Delete(out object data)
        {
            throw new NotImplementedException();
        }

        public async Task<Usuario> Get(UsuarioParameter parameters)
        {
            var result = await
ExecuteRequest<UsuarioParameter,Usuario>(parameters,HttpMethod.Get);
            if(result.status == ResponseStatus.Success)
            {
                return result.data;
            }
            else
            {
                throw new Exception(result.message.description);
            }
        }

        public async Task<IList<Usuario>> GetList(UsuarioParameter parameters)
        {
            var result = await
ExecuteRequest<UsuarioParameter,List<Usuario>>(parameters,HttpMethod.Get);
            if (result.status == ResponseStatus.Success)
            {
                return result.data;
            }
            else
            {
                throw new Exception(result.message.description);
            }
        }

        public async Task<Usuario> Post(UsuarioParameter parameter)
        {
            var result = await ExecuteRequest<UsuarioParameter,
Usuario>(parameter,HttpMethod.Post);
            if (result.status == ResponseStatus.Success)
                return result.data;
            else
                throw new Exception(result.message.description);
        }

        public async Task<Usuario> Update(UsuarioParameter parameter)
        {
            var result = await ExecuteRequest<UsuarioParameter,
Usuario>(parameter,HttpMethod.Put);
            if (result.status == ResponseStatus.Success)
                return result.data;
            else
                throw new Exception(result.message.description);
        }
    }
}
```

- Projeto API.Services:

- Classe VerificacaoAPIServices;

```
using API.Data.DTO;
using API.HttpClient;
using API.HttpClient.Proxy;
using API.Parameters;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using static API.Parameters.VerificacaoParameter;

namespace API.Services
{
   public class VerificacaoAPIServices : HttpClientBase,
IAPIServicesRepository<Verificacao, VerificacaoParameter>
   {
      public VerificacaoAPIServices
         (
            Operacao operacao,
            int? idAtividade = null,
            int? idGrupoUsuario = null,
            string includeProperties = ""
         )
         :base("Verificacao")
      {
         Parameters.idOperacao = (int)operacao;
         Parameters.idAtividade = idAtividade;
         Parameters.idGrupoUsuario = idGrupoUsuario;

         Parameters.includeProperties = includeProperties;
      }

      public bool Delete()
      {
         throw new NotImplementedException();
      }

      public bool Delete(out object data)
      {
         throw new NotImplementedException();
      }

      public async Task<Verificacao> Get(VerificacaoParameter parameters)
      {
         var result = await
ExecuteRequest<VerificacaoParameter,Verificacao>(parameters,HttpMethod.Get);
         if (result.status == ResponseStatus.Success)
         {
            return result.data;
         }
         else
         {
            throw new Exception(result.message.description);
         }
      }

      public async Task<IList<Verificacao>> GetList(VerificacaoParameter parameters)
      {
         var result = await ExecuteRequest<VerificacaoParameter,
List<Verificacao>>(parameters,HttpMethod.Get);
         if (result.status == ResponseStatus.Success)
```

```
            {
                return result.data;
            }
            else
            {
                throw new Exception(result.message.description);
            }
        }

        public async Task<Verificacao> Post(VerificacaoParameter parameter)
        {
            var result = await ExecuteRequest<VerificacaoParameter,
Verificacao>(parameter,HttpMethod.Post);
            if (result.status == ResponseStatus.Success)
                return result.data;
            else
                throw new Exception(result.message.description);
        }

        public async Task<Verificacao> Update(VerificacaoParameter parameter)
        {
            var result = await ExecuteRequest<VerificacaoParameter,
Verificacao>(parameter,HttpMethod.Put);
            if (result.status == ResponseStatus.Success)
                return result.data;
            else
                throw new Exception(result.message.description);
        }
    }
}
```

- Projeto API.Services:
    - Classe VerificacaoAPIServices;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using System.Reflection;
using System.Net;
using System.Net.Http;
using Newtonsoft.Json;
using API.Parameters;
using static API.Core.ConfigurationManager;
using API.HttpClient.Proxy;
using API.Data.DTO;

namespace App06Teste
{
    [Activity(Label = "CriarVerificacaoActivity")]
    public class CriarVerificacaoActivity : Activity
    {
```

```
        Spinner spnProduto;
        Spinner spnTipoVerificacao;
        Spinner spnGrupoUsuario;
        EditText txtdeVerificacao;
        Button btnSalvarVerificacao;
        List<GrupoUsuario> _listaGrupoUsuario;
        List<TipoVerificacao> _listaTipoVerificacao;
        List<Produto> _listaProduto;
        Verificacao _verificacao;
        protected override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);
            SetContentView(Resource.Layout.CriarVerificacao);
            // Create your application here
            spnProduto = FindViewById<Spinner>(Resource.Id.spnProduto);
            spnTipoVerificacao = FindViewById<Spinner>(Resource.Id.spnTipoVerificacao);
            spnGrupoUsuario = FindViewById<Spinner>(Resource.Id.spnGrupoUsuario);
            txtdeVerificacao = FindViewById<EditText>(Resource.Id.txtdeVerificacao);

            btnSalvarVerificacao = FindViewById<Button>(Resource.Id.btnSalvarVerificacao);

            btnSalvarVerificacao.Click += BtnSalvarVerificacao_Click;
        }

        protected async override void OnResume()
        {
            base.OnResume();

            try
            {
                using (HttpClient Client = new HttpClient())
                {
                    Client.DefaultRequestHeaders.Accept.Add(new
System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json"));
                    Client.MaxResponseContentBufferSize = 256000;
                    GrupoUsuarioParameter parameter = new GrupoUsuarioParameter()
                    {
                        idOperacao = (int)GrupoUsuarioParameter.Operacao.GetList
                    };
                    var uri = new Uri(string.Format("{0}/api/GrupoUsuario/Get?{1}",
AppSettings.ApiUrl, GetQueryString(parameter)));

                    var requestMessage = new HttpRequestMessage()
                    {
                        RequestUri = uri,
                        Method = HttpMethod.Get
                    };
                    var response = new HttpResponseMessage();
                    try
                    {
                        response = await Client.GetAsync(uri).ConfigureAwait(true);
                    }
                    catch (HttpRequestException HRex)
                    {
                        SetAlerta(HRex.Message + HRex.StackTrace);
                    }
                    catch (Exception ex)
                    {
                        SetAlerta(ex.Message + ex.StackTrace);
                    }
                    if (response.IsSuccessStatusCode)
                    {
                        string stringResult = "";
                        try
```

```
                    {
                        await response.Content.ReadAsStringAsync().ContinueWith((ta =>
                        {
                            if (ta.Exception == null)
                            {
                                stringResult = ta.Result;
                            }
                            else
                            {
                                SetAlerta(ta.Exception.Message);
                            };
                        })); ;
                    } catch (HttpRequestException HRex)

                    {
                        SetAlerta(HRex.Message + HRex.StackTrace);
                    }
                    catch (Exception ex)
                    {
                        SetAlerta(ex.Message + ex.StackTrace);
                    }
                    var proxyUsuario =
JsonConvert.DeserializeObject<ResponseProxy<List<GrupoUsuario>>>(stringResult);
                    if (proxyUsuario.status == ResponseStatus.Success)
                        _listaGrupoUsuario = proxyUsuario.data;
                    else
                        SetAlerta(proxyUsuario.message.description);
                }
            }
        }
        catch (Exception ex) { SetAlerta(ex.Message); }

        try
        {
            using (HttpClient Client = new HttpClient())
            {
                Client.DefaultRequestHeaders.Accept.Add(new
System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json"));
                Client.MaxResponseContentBufferSize = 256000;
                TipoVerificacaoParameter parameter = new TipoVerificacaoParameter()
                {
                    idOperacao = (int)TipoVerificacaoParameter.Operacao.GetList
                };
                var uri = new Uri(string.Format("{0}/api/TipoVerificacao/Get?{1}",
AppSettings.ApiUrl, GetQueryString(parameter)));

                var requestMessage = new HttpRequestMessage()
                {
                    RequestUri = uri,
                    Method = HttpMethod.Get
                };
                var response = new HttpResponseMessage();
                try
                {
                    response = await Client.GetAsync(uri).ConfigureAwait(true);
                }
                catch (HttpRequestException HRex)
                {
                    SetAlerta(HRex.Message + HRex.StackTrace);
                }
                catch (Exception ex)
                {
                    SetAlerta(ex.Message + ex.StackTrace);
```

```
                }
                if (response.IsSuccessStatusCode)
                {
                    string stringResult = "";
                    try
                    {
                        await response.Content.ReadAsStringAsync().ContinueWith((ta =>
                        {
                            if (ta.Exception == null)
                            {
                                stringResult = ta.Result;
                            }
                            else
                            {
                                SetAlerta(ta.Exception.Message);
                            };
                        })); ;
                    }
                    catch (HttpRequestException HRex)
                    {
                        SetAlerta(HRex.Message + HRex.StackTrace);
                    }
                    catch (Exception ex)
                    {
                        SetAlerta(ex.Message + ex.StackTrace);
                    }
                    var proxyUsuario =
JsonConvert.DeserializeObject<ResponseProxy<List<TipoVerificacao>>>(stringResult);
                    if (proxyUsuario.status == ResponseStatus.Success)
                        _listaTipoVerificacao = proxyUsuario.data;
                    else
                        SetAlerta(proxyUsuario.message.description);
                }
            }
        }
        catch (Exception ex) { SetAlerta(ex.Message); }

        try
        {
            using (HttpClient Client = new HttpClient())
            {
                Client.DefaultRequestHeaders.Accept.Add(new
System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json"));
                Client.MaxResponseContentBufferSize = 256000;
                ProdutoParameter parameter = new ProdutoParameter()
                {
                    idOperacao = (int)VerificacaoParameter.Operacao.GetList
                };
                var uri = new Uri(string.Format("{0}/api/Produto/Get?{1}", AppSettings.ApiUrl,
GetQueryString(parameter)));

                var requestMessage = new HttpRequestMessage()
                {
                    RequestUri = uri,
                    Method = HttpMethod.Get
                };
                var response = new HttpResponseMessage();
                try
                {
                    response = await Client.GetAsync(uri).ConfigureAwait(true);
                }
                catch (HttpRequestException HRex)
                {
                    SetAlerta(HRex.Message + HRex.StackTrace);
```

```
                }
                catch (Exception ex)
                {
                    SetAlerta(ex.Message + ex.StackTrace);
                }
                if (response.IsSuccessStatusCode)
                {
                    string stringResult = "";
                    try
                    {
                        await response.Content.ReadAsStringAsync().ContinueWith((ta =>
                        {
                            if (ta.Exception == null)
                            {
                                stringResult = ta.Result;
                            }
                            else
                            {
                                SetAlerta(ta.Exception.Message);
                            };
                        })); ;
                    }
                    catch (HttpRequestException HRex)
                    {
                        SetAlerta(HRex.Message + HRex.StackTrace);
                    }
                    catch (Exception ex)
                    {
                        SetAlerta(ex.Message + ex.StackTrace);
                    }
                    var proxyUsuario =
JsonConvert.DeserializeObject<ResponseProxy<List<Produto>>>(stringResult);
                    if (proxyUsuario.status == ResponseStatus.Success)
                        _listaProduto = proxyUsuario.data;
                    else
                        SetAlerta(proxyUsuario.message.description);
                }
            }
        }
        catch (Exception ex) { SetAlerta(ex.Message); }

        var adapter = new ArrayAdapter(this, Resource.Layout.spinnerTextFile);
        foreach(var grupoUsu in _listaGrupoUsuario)
    {

adapter.Add(grupoUsu.deGrupoUsuario);
        }
        spnGrupoUsuario.Adapter = adapter;

        var adapterVeri = new ArrayAdapter(this, Resource.Layout.spinnerText2);
        foreach(var tipoVerificacao in _listaTipoVerificacao)
        {
            adapterVeri.Add(tipoVerificacao.deTipoVerificacao);
        }
        spnTipoVerificacao.Adapter = adapterVeri;

        var adapterProduto = new ArrayAdapter(this, Resource.Layout.spinnerText3);
        foreach( var prod in _listaProduto.Select(prod => new { prod.deProduto
}).Distinct())
        {
            adapterProduto.Add(prod.deProduto);
        }
        spnProduto.Adapter = adapterProduto;
```

```
    }

    private async void BtnSalvarVerificacao_Click(object sender, EventArgs e)
    {
        try
        {
            using (HttpClient Client = new HttpClient())
            {
                Client.DefaultRequestHeaders.Accept.Add(new
System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json"));
                Client.MaxResponseContentBufferSize = 256000;
                VerificacaoParameter parameter = new VerificacaoParameter()
                {
                    idOperacao = (int)VerificacaoParameter.Operacao.Create,
                    VerificacaoJSON = JsonConvert.SerializeObject(new Verificacao()
                    {
                        deVerificacao = txtdeVerificacao.Text,
                        idTipoVerificacao = _listaTipoVerificacao.Where(tipoVerif =>
tipoVerif.deTipoVerificacao ==
spnTipoVerificacao.SelectedItem.ToString()).FirstOrDefault().idTipoVerificacao,
                        idProduto = _listaProduto.Where(prod => prod.deProduto ==
spnProduto.SelectedItem.ToString()).FirstOrDefault().idProduto,
                        idGrupoUsuario = _listaGrupoUsuario.Where(grupoUsu =>
grupoUsu.deGrupoUsuario ==
spnGrupoUsuario.SelectedItem.ToString()).FirstOrDefault().idGrupoUsuario
                    })
                };
                var uri = new Uri(string.Format("{0}/api/Verificacao/Post",
AppSettings.ApiUrl));

                var requestMessage = new HttpRequestMessage()
                {
                    RequestUri = uri,
                    Method = HttpMethod.Post
                };
                var response = new HttpResponseMessage();
                try
                {
                    HttpRequestMessage request = new
HttpRequestMessage(System.Net.Http.HttpMethod.Post, uri);
                    request.Content = GetStringContent(parameter);
                    request.Headers.Accept.Add(new
System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json"));
                    response = await Client.SendAsync(request).ConfigureAwait(true);
                }
                catch (HttpRequestException HRex)
                {
                    SetAlerta(HRex.Message + HRex.StackTrace);
                }
                catch (Exception ex)
                {
                    SetAlerta(ex.Message + ex.StackTrace);
                }
                if (response.IsSuccessStatusCode)
                {
                    SetAlerta("Sucesso");
                    string stringResult = "";
                    try
                    {
                        await response.Content.ReadAsStringAsync().ContinueWith((ta =>
                        {
                            if (ta.Exception == null)
                            {
                                stringResult = ta.Result;
```

```
                    }
                    else
                    {
                        SetAlerta(ta.Exception.Message);
                    };
                })); ;
            }
            catch (HttpRequestException HRex)
            {
                SetAlerta(HRex.Message + HRex.StackTrace);
            }
            catch (Exception ex)
            {
                SetAlerta(ex.Message + ex.StackTrace);
            }
            var proxyUsuario =
JsonConvert.DeserializeObject<ResponseProxy<Verificacao>>(stringResult);
            if (proxyUsuario.status == ResponseStatus.Success)
                _verificacao = proxyUsuario.data;
            else
                SetAlerta(proxyUsuario.message.description);
            if (_verificacao.idVerificacao > 0)
                StartActivity(typeof(MenuAdministracaoActivity));
            else
                SetAlerta("Ocorreu um erro.");
        }
        else
            SetAlerta(response.StatusCode.ToString());
      }
    }
    catch (Exception ex) { SetAlerta(ex.Message); }
  }

  private void SetAlerta(string v)
  {
    Toast toast = Toast.MakeText(this, v, ToastLength.Short);
    toast.Show();
  }

  public string GetQueryString(object obj)
  {
    var properties = from p in obj.GetType().GetRuntimeProperties()
                     where p.GetValue(obj, null) != null
                     select p.Name + "=" + WebUtility.UrlEncode(p.GetValue(obj,
null).ToString());

    return String.Join("&", properties.ToArray());
  }
  private StringContent GetStringContent(object obj)
  {
    var jsonObject = JsonConvert.SerializeObject(obj);
    StringContent result = new StringContent(jsonObject, Encoding.UTF8,
"application/json");
    return result;
  }
 }
}
```

- Projeto App06Teste (Representa o ControleShopping.AndroidApp)::

- Classe CriarVerificacaoActivity;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using System.Reflection;
using System.Net;
using System.Net.Http;
using Newtonsoft.Json;
using API.Parameters;
using static API.Core.ConfigurationManager;
using API.HttpClient.Proxy;
using API.Data.DTO;

namespace App06Teste
{
    [Activity(Label = "CriarVerificacaoActivity")]
    public class CriarVerificacaoActivity : Activity
    {
        Spinner spnProduto;
        Spinner spnTipoVerificacao;
        Spinner spnGrupoUsuario;
        EditText txtdeVerificacao;
        Button btnSalvarVerificacao;
        List<GrupoUsuario> _listaGrupoUsuario;
        List<TipoVerificacao> _listaTipoVerificacao;
        List<Produto> _listaProduto;
        Verificacao _verificacao;
        protected override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);
            SetContentView(Resource.Layout.CriarVerificacao);
            // Create your application here
            spnProduto = FindViewById<Spinner>(Resource.Id.spnProduto);
            spnTipoVerificacao =
FindViewById<Spinner>(Resource.Id.spnTipoVerificacao);
            spnGrupoUsuario =
FindViewById<Spinner>(Resource.Id.spnGrupoUsuario);
            txtdeVerificacao =
FindViewById<EditText>(Resource.Id.txtdeVerificacao);

            btnSalvarVerificacao =
FindViewById<Button>(Resource.Id.btnSalvarVerificacao);

            btnSalvarVerificacao.Click += BtnSalvarVerificacao_Click;
        }

        protected async override void OnResume()
        {
            base.OnResume();

            try
            {
                using (HttpClient Client = new HttpClient())
```

```
                {
                    Client.DefaultRequestHeaders.Accept.Add(new
System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json
"));
                    Client.MaxResponseContentBufferSize = 256000;
                    GrupoUsuarioParameter parameter = new
GrupoUsuarioParameter()
                    {
                        idOperacao =
(int)GrupoUsuarioParameter.Operacao.GetList
                    };
                    var uri = new
Uri(string.Format("{0}/api/GrupoUsuario/Get?{1}", AppSettings.ApiUrl,
GetQueryString(parameter)));

                    var requestMessage = new HttpRequestMessage()
                    {
                        RequestUri = uri,
                        Method = HttpMethod.Get
                    };
                    var response = new HttpResponseMessage();
                    try
                    {
                        response = await
Client.GetAsync(uri).ConfigureAwait(true);
                    }
                    catch (HttpRequestException HRex)
                    {
                        SetAlerta(HRex.Message + HRex.StackTrace);
                    }
                    catch (Exception ex)
                    {
                        SetAlerta(ex.Message + ex.StackTrace);
                    }
                    if (response.IsSuccessStatusCode)
                    {
                        string stringResult = "";
                        try
                        {
                            await
response.Content.ReadAsStringAsync().ContinueWith((ta =>
                            {
                                if (ta.Exception == null)
                                {
                                    stringResult = ta.Result;
                                }
                                else
                                {
                                    SetAlerta(ta.Exception.Message);
                                };
                            })); ;
                        }
                        catch (HttpRequestException HRex)
                        {
                            SetAlerta(HRex.Message + HRex.StackTrace);
                        }
                        catch (Exception ex)
                        {
                            SetAlerta(ex.Message + ex.StackTrace);
                        }
```

```
                        var proxyUsuario =
JsonConvert.DeserializeObject<ResponseProxy<List<GrupoUsuario>>>(stringRe
sult);
                        if (proxyUsuario.status ==
ResponseStatus.Success)
                            _listaGrupoUsuario = proxyUsuario.data;
                        else
                            SetAlerta(proxyUsuario.message.description);
                    }
                }
            }
            catch (Exception ex) { SetAlerta(ex.Message); }

            try
            {
                using (HttpClient Client = new HttpClient())
                {
                    Client.DefaultRequestHeaders.Accept.Add(new
System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json
"));
                    Client.MaxResponseContentBufferSize = 256000;
                    TipoVerificacaoParameter parameter = new
TipoVerificacaoParameter()
                    {
                        idOperacao =
(int)TipoVerificacaoParameter.Operacao.GetList
                    };
                    var uri = new
Uri(string.Format("{0}/api/TipoVerificacao/Get?{1}", AppSettings.ApiUrl,
GetQueryString(parameter)));

                    var requestMessage = new HttpRequestMessage()
                    {
                        RequestUri = uri,
                        Method = HttpMethod.Get
                    };
                    var response = new HttpResponseMessage();
                    try
                    {
                        response = await
Client.GetAsync(uri).ConfigureAwait(true);
                    }
                    catch (HttpRequestException HRex)
                    {
                        SetAlerta(HRex.Message + HRex.StackTrace);
                    }
                    catch (Exception ex)
                    {
                        SetAlerta(ex.Message + ex.StackTrace);
                    }
                    if (response.IsSuccessStatusCode)
                    {
                        string stringResult = "";
                        try
                        {
                            await
response.Content.ReadAsStringAsync().ContinueWith((ta =>
                            {
                                if (ta.Exception == null)
                                {
                                    stringResult = ta.Result;
                                }
```

```
                                else
                                {
                                    SetAlerta(ta.Exception.Message);
                                };
                            })); ;
                        }
                        catch (HttpRequestException HRex)
                        {
                            SetAlerta(HRex.Message + HRex.StackTrace);
                        }
                        catch (Exception ex)
                        {
                            SetAlerta(ex.Message + ex.StackTrace);
                        }
                        var proxyUsuario =
JsonConvert.DeserializeObject<ResponseProxy<List<TipoVerificacao>>>(stringResult)
;
                        if (proxyUsuario.status == ResponseStatus.Success)
                            _listaTipoVerificacao = proxyUsuario.data;
                        else
                            SetAlerta(proxyUsuario.message.description);
                    }
                }
            }
            catch (Exception ex) { SetAlerta(ex.Message); }

            try
            {
                using (HttpClient Client = new HttpClient())
                {
                    Client.DefaultRequestHeaders.Accept.Add(new
System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json"));
                    Client.MaxResponseContentBufferSize = 256000;
                    ProdutoParameter parameter = new ProdutoParameter()
                    {
                        idOperacao = (int)VerificacaoParameter.Operacao.GetList
                    };
                    var uri = new Uri(string.Format("{0}/api/Produto/Get?{1}",
AppSettings.ApiUrl, GetQueryString(parameter)));

                    var requestMessage = new HttpRequestMessage()
                    {
                        RequestUri = uri,
                        Method = HttpMethod.Get
                    };
                    var response = new HttpResponseMessage();
                    try
                    {
                        response = await
Client.GetAsync(uri).ConfigureAwait(true);
                    }
                    catch (HttpRequestException HRex)
                    {
                        SetAlerta(HRex.Message + HRex.StackTrace);
                    }
                    catch (Exception ex)
                    {
                        SetAlerta(ex.Message + ex.StackTrace);
                    }
                    if (response.IsSuccessStatusCode)
                    {
                        string stringResult = "";
```

```
                        try
                        {
                            await
response.Content.ReadAsStringAsync().ContinueWith((ta =>
                            {
                                if (ta.Exception == null)
                                {
                                    stringResult = ta.Result;
                                }
                                else
                                {
                                    SetAlerta(ta.Exception.Message);
                                };
                            })); ;
                        }
                        catch (HttpRequestException HRex)
                        {
                            SetAlerta(HRex.Message + HRex.StackTrace);
                        }
                        catch (Exception ex)
                        {
                            SetAlerta(ex.Message + ex.StackTrace);
                        }
                        var proxyUsuario =
JsonConvert.DeserializeObject<ResponseProxy<List<Produto>>>(stringResult);
                        if (proxyUsuario.status == ResponseStatus.Success)
                            _listaProduto = proxyUsuario.data;
                        else
                            SetAlerta(proxyUsuario.message.description);
                    }
                }
            }
            catch (Exception ex) { SetAlerta(ex.Message); }

            var adapter = new ArrayAdapter(this,
Resource.Layout.spinnerTextFile);
            foreach(var grupoUsu in _listaGrupoUsuario)
            {
                adapter.Add(grupoUsu.deGrupoUsuario);
            }
            spnGrupoUsuario.Adapter = adapter;

            var adapterVeri = new ArrayAdapter(this,
Resource.Layout.spinnerText2);
            foreach(var tipoVerificacao in _listaTipoVerificacao)
            {
                adapterVeri.Add(tipoVerificacao.deTipoVerificacao);
            }
            spnTipoVerificacao.Adapter = adapterVeri;

            var adapterProduto = new ArrayAdapter(this,
Resource.Layout.spinnerText3);
            foreach( var prod in _listaProduto.Select(prod => new {
prod.deProduto }).Distinct())
            {
                adapterProduto.Add(prod.deProduto);
            }
            spnProduto.Adapter = adapterProduto;
                }

private async void BtnSalvarVerificacao_Click(object sender, EventArgs e)
        {
```

```
            try
            {
                using (HttpClient Client = new HttpClient())
                {
                    Client.DefaultRequestHeaders.Accept.Add(new
System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json"));
                    Client.MaxResponseContentBufferSize = 256000;
                    VerificacaoParameter parameter = new VerificacaoParameter()
                    {
                        idOperacao = (int)VerificacaoParameter.Operacao.Create,
                        VerificacaoJSON = JsonConvert.SerializeObject(new
Verificacao()
                        {
                            deVerificacao = txtdeVerificacao.Text,
                            idTipoVerificacao =
_listaTipoVerificacao.Where(tipoVerif => tipoVerif.deTipoVerificacao ==
spnTipoVerificacao.SelectedItem.ToString()).FirstOrDefault().idTipoVerificacao,
                            idProduto = _listaProduto.Where(prod =>
prod.deProduto == spnProduto.SelectedItem.ToString()).FirstOrDefault().idProduto,
                            idGrupoUsuario = _listaGrupoUsuario.Where(grupoUsu =>
grupoUsu.deGrupoUsuario ==
spnGrupoUsuario.SelectedItem.ToString()).FirstOrDefault().idGrupoUsuario
                        })
                    };
                    var uri = new Uri(string.Format("{0}/api/Verificacao/Post",
AppSettings.ApiUrl));

                    var requestMessage = new HttpRequestMessage()
                    {
                        RequestUri = uri,
                        Method = HttpMethod.Post
                    };
                    var response = new HttpResponseMessage();
                    try
                    {
                        HttpRequestMessage request = new
HttpRequestMessage(System.Net.Http.HttpMethod.Post, uri);
                        request.Content = GetStringContent(parameter);
                        request.Headers.Accept.Add(new
System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json"));
                        response = await
Client.SendAsync(request).ConfigureAwait(true);
                    }
                    catch (HttpRequestException HRex)
                    {
                        SetAlerta(HRex.Message + HRex.StackTrace);
                    }
                    catch (Exception ex)
                    {
                        SetAlerta(ex.Message + ex.StackTrace);
                    }
                    if (response.IsSuccessStatusCode)
                    {
                        SetAlerta("Sucesso");
                        string stringResult = "";
                        try
                        {
                            await
response.Content.ReadAsStringAsync().ContinueWith((ta =>
                            {
                                if (ta.Exception == null)
                                {
```

```
                                stringResult = ta.Result;
                            }
                            else
                            {
                                SetAlerta(ta.Exception.Message);
                            };
                        })); ;
                    }
                    catch (HttpRequestException HRex)
                    {
                        SetAlerta(HRex.Message + HRex.StackTrace);
                    }
                    catch (Exception ex)
                    {
                        SetAlerta(ex.Message + ex.StackTrace);
                    }
                    var proxyUsuario =
JsonConvert.DeserializeObject<ResponseProxy<Verificacao>>(stringResult);
                    if (proxyUsuario.status == ResponseStatus.Success)
                        _verificacao = proxyUsuario.data;
                    else
                        SetAlerta(proxyUsuario.message.description);
                    if (_verificacao.idVerificacao > 0)
                        StartActivity(typeof(MenuAdministracaoActivity));
                    else
                        SetAlerta("Ocorreu um erro.");
                }
                else
                    SetAlerta(response.StatusCode.ToString());
            }
        }
        catch (Exception ex) { SetAlerta(ex.Message); }
    }

    private void SetAlerta(string v)
    {
        Toast toast = Toast.MakeText(this, v, ToastLength.Short);
        toast.Show();
    }

    public string GetQueryString(object obj)
    {
        var properties = from p in obj.GetType().GetRuntimeProperties()
                         where p.GetValue(obj, null) != null
                         select p.Name + "=" +
WebUtility.UrlEncode(p.GetValue(obj, null).ToString());

        return String.Join("&", properties.ToArray());
    }
    private StringContent GetStringContent(object obj)
    {
        var jsonObject = JsonConvert.SerializeObject(obj);
        StringContent result = new StringContent(jsonObject, Encoding.UTF8,
"application/json");
        return result;
    }
}
```

- Projeto API.Teste06:
    - Classe ListaAtividadesActivity;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Android.App;
using Android.Widget;
using Android.OS;
using System.Linq;
using System.Net.Http;
using System;
using static API.Core.ConfigurationManager;
using API.Data.DTO;
using System.Threading.Tasks;
using Newtonsoft.Json;
using API.HttpClient.Proxy;
using System.Collections.Generic;
using System.Reflection;
using API.Parameters;
using System.Net;
using Android.Content;
using Android.Graphics;

namespace App06Teste
{
    [Activity(Label = "Lista de Atividades")]
    class ListaAtividadesActivity : Activity
    {
        LinearLayout linearLayout;
        Button btnCarregarLista;
        List<Verificacao> listaVerificacao;
        List<Produto> listaProduto;
        protected async override void OnCreate(Bundle bundle)
        {
            base.OnCreate(bundle);
            SetContentView(Resource.Layout.ListaAtividades);
            linearLayout =
FindViewById<LinearLayout>(Resource.Id.linearLayout1);


        }

        protected async override void OnResume()
        {
            base.OnResume();

            int idProduto = 0;
            int.TryParse(Intent.GetStringExtra("idProduto"), out
idProduto);


            try
            {
                using (HttpClient Client = new HttpClient())
                {
                    Client.DefaultRequestHeaders.Accept.Add(new
System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json
"));
                    Client.MaxResponseContentBufferSize = 256000;
                    VerificacaoParameter parameter = new
VerificacaoParameter()
                    {
```

```
                        idOperacao =
(int)VerificacaoParameter.Operacao.GetListByGrupoUsuario,
                        idGrupoUsuario =
Session.Usuario.GrupoUsuario.FirstOrDefault().idGrupoUsuario,
                        includeProperties = "Produto"
                    };
                    var uri = new
Uri(string.Format("{0}/api/Verificacao/Get?{1}", AppSettings.ApiUrl,
GetQueryString(parameter)));

                    var requestMessage = new HttpRequestMessage()
                    {
                        RequestUri = uri,
                        Method = HttpMethod.Get
                    };
                    var response = new HttpResponseMessage();
                    try
                    {
                        response = await Client.GetAsync(uri);
                    }
                    catch (HttpRequestException HRex)
                    {
                        SetAlerta(HRex.Message + HRex.StackTrace);
                    }
                    catch (Exception ex)
                    {
                        SetAlerta(ex.Message + ex.StackTrace);
                    }
                    if (response.IsSuccessStatusCode)
                    {
                        string stringResult = "";
                        try
                        {
                            stringResult = await
response.Content.ReadAsStringAsync();
                        }
                        catch (HttpRequestException HRex)
                        {
                            SetAlerta(HRex.Message + HRex.StackTrace);
                        }
                        catch (Exception ex)
                        {
                            SetAlerta(ex.Message + ex.StackTrace);
                        }
                        var proxyUsuario =
JsonConvert.DeserializeObject<ResponseProxy<List<Verificacao>>>(stringRes
ult);
                        if (proxyUsuario.status ==
ResponseStatus.Success)
                            listaVerificacao = proxyUsuario.data;
                        else
                            SetAlerta(proxyUsuario.message.description);
                    }
                }
                foreach (var verificao in listaVerificacao)
                {
                    if (idProduto > 0)
                    {
                        if (verificao.Produto.idProduto == idProduto)
                        {
                            Button btnNew = new
Button(base.ApplicationContext);
```

```
                            btnNew.Text = verificao.deVerificacao;
                            btnNew.Id = verificao.idVerificacao;
                            btnNew.TextAlignment =
Android.Views.TextAlignment.Center;
                            btnNew.TextSize = 35;
                            btnNew.SetTextColor(Color.White);

                            btnNew.Click += (senders, EventArgs) => {
btnNew_Click(senders, EventArgs, verificao.idVerificacao); };
                            linearLayout.AddView(btnNew);
                        }
                    }
                    else
                    {
                        Button btnNew = new
Button(base.ApplicationContext);
                        btnNew.Text = verificao.deVerificacao;
                        btnNew.Id = verificao.idVerificacao;
                        btnNew.TextAlignment =
Android.Views.TextAlignment.Center;
                        btnNew.TextSize = 35;
                        btnNew.SetTextColor(Color.White);

                        btnNew.Click += (senders, EventArgs) => {
btnNew_Click(senders, EventArgs, verificao.idVerificacao); };
                        linearLayout.AddView(btnNew);
                    }
                }
            }
            catch (Exception ex) { SetAlerta(ex.Message); }
        }

        protected void btnNew_Click(object sender, EventArgs e, int
idVerificacao)
        {
            Intent intent = new Intent(this,
typeof(VerificacaoActivity));
            intent.PutExtra("idVerificacao", idVerificacao);
            StartActivity(intent);
            //SetAlerta(idVerificacao.ToString());
        }

        private void SetAlerta(string v)
        {
            Toast toast = Toast.MakeText(this, v, ToastLength.Short);
            toast.Show();
        }

        public string GetQueryString(object obj)
        {
            var properties = from p in
obj.GetType().GetRuntimeProperties()
                             where p.GetValue(obj, null) != null
                             select p.Name + "=" +
WebUtility.UrlEncode(p.GetValue(obj, null).ToString());

            return String.Join("&", properties.ToArray());
        }
    }
}
```

- Projeto App06Teste (Representa o ControleShopping.AndroidApp):
    - Classe ListaProdutosActivity;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using System.Net.Http;
using API.Parameters;
using static API.Core.ConfigurationManager;
using Newtonsoft.Json;
using API.HttpClient.Proxy;
using API.Data.DTO;
using System.Net;
using System.Reflection;
using Android.Graphics;

namespace App06Teste
{
    [Activity(Label = "Lista de Produtos")]
    public class ListaProdutosActivity : Activity
    {
        LinearLayout linearLayoutListaProdutos;
        List<Verificacao> _listaVerificacao;
        List<Produto> _listaProduto;
        protected override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);
            SetContentView(Resource.Layout.ListaProdutos);

            linearLayoutListaProdutos =
FindViewById<LinearLayout>(Resource.Id.linearLayoutListaProdutos);
            // Create your application here
        }

        protected async override void OnResume()
        {
            base.OnResume();

            try
            {
                using (HttpClient Client = new HttpClient())
                {
                    Client.DefaultRequestHeaders.Accept.Add(new
System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json"));
                    Client.MaxResponseContentBufferSize = 256000;
                    VerificacaoParameter parameter = new VerificacaoParameter()
                    {
                        idOperacao =
(int)VerificacaoParameter.Operacao.GetListByGrupoUsuario,
```

```
                        idGrupoUsuario =
Session.Usuario.GrupoUsuario.FirstOrDefault().idGrupoUsuario,
                        includeProperties = "Produto"
                    };
                    var uri = new
Uri(string.Format("{0}/api/Verificacao/Get?{1}", AppSettings.ApiUrl,
GetQueryString(parameter)));

                    var requestMessage = new HttpRequestMessage()
                    {
                        RequestUri = uri,
                        Method = HttpMethod.Get
                    };
                    var response = new HttpResponseMessage();
                    try
                    {
                        response = await
Client.GetAsync(uri).ConfigureAwait(true);
                    }
                    catch (HttpRequestException HRex)
                    {
                        SetAlerta(HRex.Message + HRex.StackTrace);
                    }
                    catch (Exception ex)
                    {
                        SetAlerta(ex.Message + ex.StackTrace);
                    }
                    if (response.IsSuccessStatusCode)
                    {
                        string stringResult = "";
                        try
                        {
                            await
response.Content.ReadAsStringAsync().ContinueWith((ta =>
                            {
                                if (ta.Exception == null)
                                {
                                    stringResult = ta.Result;
                                }
                                else
                                {
                                    SetAlerta(ta.Exception.Message);
                                }
                            })); ;
                        }
                        catch (HttpRequestException HRex)
                        {
                            SetAlerta(HRex.Message + HRex.StackTrace);
                        }
                        catch (Exception ex)
                        {
                            SetAlerta(ex.Message + ex.StackTrace);
                        }
                        var proxyUsuario =
JsonConvert.DeserializeObject<ResponseProxy<List<Verificacao>>>(stringResult);
                        if (proxyUsuario.status == ResponseStatus.Success)
                            _listaVerificacao = proxyUsuario.data;
                        else
                            SetAlerta(proxyUsuario.message.description);
                    }
                }
                if (_listaProduto == null)
```

```
                    _listaProduto = new List<Produto>();
                foreach (var verificao in _listaVerificacao)
                {
                    _listaProduto.Add(verificao.Produto);
                }
                foreach (var produto in _listaProduto.Select(prod => new {
prod.idProduto, prod.deProduto}).Distinct())
                {
                    Button btnNew = new Button(base.ApplicationContext);
                    btnNew.Text = produto.deProduto;
                    btnNew.Id = produto.idProduto;
                    btnNew.TextAlignment = Android.Views.TextAlignment.Center;
                    btnNew.TextSize = 35;
                    btnNew.SetTextColor(Color.White);

                    btnNew.Click += (senders, EventArgs) => {
btnNew_Click(senders, EventArgs, produto.idProduto); };
                    linearLayoutListaProdutos.AddView(btnNew);
                }
            }
            catch (Exception ex) { SetAlerta(ex.Message); }
        }

        private void btnNew_Click(object senders, EventArgs eventArgs, int
idProduto)
        {
            Intent intent = new Intent(this, typeof(ListaAtividadesActivity));
            intent.PutExtra("idProduto", idProduto.ToString());
            StartActivity(intent);
        }

        private void SetAlerta(string v)
        {
            Toast toast = Toast.MakeText(this, v, ToastLength.Short);
            toast.Show();
        }

        public string GetQueryString(object obj)
        {
            var properties = from p in obj.GetType().GetRuntimeProperties()
                             where p.GetValue(obj, null) != null
                             select p.Name + "=" +
WebUtility.UrlEncode(p.GetValue(obj, null).ToString());

            return String.Join("&", properties.ToArray());
        }
    }
        }
```

- Projeto App06Teste (Representa o ControleShopping.AndroidApp):
    - Classe LoginActivity;

```
        using Android.App;
        using Android.Widget;
        using Android.OS;
        using System.Linq;
        using System.Net.Http;
        using System;
```

```
using static API.Core.ConfigurationManager;
using API.Data.DTO;
using System.Threading.Tasks;
using Newtonsoft.Json;
using API.HttpClient.Proxy;
using System.Collections.Generic;
using System.Reflection;
using API.Parameters;
using System.Net;
using Android.Content;

namespace App06Teste
{
    [Activity(Label = "Login", MainLauncher = true, Icon =
"@drawable/icon")]
    public class LoginActivity : Activity
    {
        EditText txtedEmail;
        EditText txtcoSenha;
        Button btnLogin;
        TextView txtnmUsuario;
        TextView txtdeGrupo;
        Usuario usuario;
        Intent intent;
        protected override void OnCreate(Bundle bundle)
        {
            base.OnCreate(bundle);
            SetContentView(Resource.Layout.Login);
            btnLogin = FindViewById<Button>(Resource.Id.btnLogin);
            btnLogin.Click += BtnLogin_Click;

        }

        private async void BtnLogin_Click(object sender, System.EventArgs
e)
        {
            txtedEmail = FindViewById<EditText>(Resource.Id.txtedEmail);
            txtcoSenha = FindViewById<EditText>(Resource.Id.txtcoSenha);
            txtnmUsuario =
FindViewById<TextView>(Resource.Id.txtnmUsuario);
            txtdeGrupo =
FindViewById<TextView>(Resource.Id.txtdeGrupoUsuario);
            if (string.IsNullOrEmpty(txtedEmail.Text))
            {
                SetAlert("É necessário preencher o e-mail.");
                return;
            }
            if (string.IsNullOrEmpty(txtcoSenha.Text))
            {
                SetAlert("É necessário preencher a senha.");
            }

            var usuario = await new
API.Services.UsuarioAPIServices(API.Parameters.UsuarioParameter.Operacao.
GetByEmailSenha, null, txtedEmail.Text, txtcoSenha.Text,
"GrupoUsuario").Get(new API.Parameters.UsuarioParameter()
            {
                idOperacao =
(int)API.Parameters.UsuarioParameter.Operacao.GetByEmailSenha,
                edEmail = txtedEmail.Text,
                coSenha = txtcoSenha.Text,
                includeProperties = "GrupoUsuario"
```

```
            });

            if (usuario != new Usuario())
            {
                if (usuario.GrupoUsuario.Any())
                {
                    Session.Usuario = usuario;
                    txtdeGrupo.Text =
usuario.GrupoUsuario.FirstOrDefault().deGrupoUsuario;
                    if (usuario.icAdministrador)
                    {
                        intent = new Intent(this,
typeof(MenuAdministracaoActivity));
                    }
                    else
                    {
                        intent = new Intent(this,
typeof(ListaAtividadesActivity));
                    }
                    StartActivity(intent);
                }
                else
                    txtdeGrupo.Text = "Não está em nenhum grupo";
                txtnmUsuario.Text = usuario.nmUsuario;
            }
        }

        protected void SetAlert(string alerta)
        {
            Toast toast = Toast.MakeText(this, alerta, ToastLength.Long);
            toast.Show();
            return;
        }
    }
}
```

- Projeto App06Teste (Representa o ControleShopping.AndroidApp):
    - Classe MainViewModel;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using XLabs.Platform.Device;

namespace App06Teste
{
    public class MainViewModel : XLabs.Forms.Mvvm.ViewModel
    {
        private readonly IDevice _device;
        private string _message;
```

```
        public MainViewModel(IDevice device)
        {
            _device = device;
            Message = String.Format("Hello Xamarin Forms Labs MVVM
Basics!! How is your {0} device", device.Manufacturer);
        }

        public string Message
        {
            get { return _message; }
            set { SetProperty(ref _message, value); }
        }
    }
}
```

- Projeto App06Teste (Representa o ControleShopping.AndroidApp):
    - Classe Media;

```
using System;
using XLabs.Platform.Services.Media;
using XLabs.Ioc;
using XLabs.Platform.Device;
using Xamarin.Forms;

namespace App06Teste
{
    public static class Media
    {
        static IMediaPicker mediaPicker = null;
        public static IMediaPicker MediaPicker
        {
            get
            {
                if (mediaPicker == null)
                {
                    var device = Resolver.Resolve<IDevice>();
                    mediaPicker = DependencyService.Get<IMediaPicker>()
?? device.MediaPicker;
                    if (mediaPicker == null) throw new
NullReferenceException("MediaPicker DependencyService.Get error");
                }

                return mediaPicker;
            }
        }
    }
}
```

- Projeto App06Teste (Representa o ControleShopping.AndroidApp):
    - Classe MenuAdministracaoActivity;

```
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;

namespace App06Teste
{
    [Activity(Label = "MenuAdministracaoActivity")]
    public class MenuAdministracaoActivity : Activity
    {
        Button btnGerenciarUsuario;
        Button btnCriarProduto;
        Button btnCriarSetor;
        Button btnCriarVerificacao;
        Button btnListaAtividades;
        protected override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);
            SetContentView(Resource.Layout.MenuAdministracao);

            // Create your application here
            btnGerenciarUsuario =
FindViewById<Button>(Resource.Id.btnGerenciarUsuario);
            btnCriarProduto =
FindViewById<Button>(Resource.Id.btnCriarProduto);
            btnCriarSetor =
FindViewById<Button>(Resource.Id.btnCriarSetor);
            btnCriarVerificacao =
FindViewById<Button>(Resource.Id.btnCriarVerificacao);
            btnListaAtividades =
FindViewById<Button>(Resource.Id.btnListaAtividades);

            btnGerenciarUsuario.Click += BtnGerenciarUsuario_Click;
            btnCriarProduto.Click += BtnCriarProduto_Click;
            btnCriarSetor.Click += BtnCriarSetor_Click;
            btnCriarVerificacao.Click += BtnCriarVerificacao_Click;
            btnListaAtividades.Click += BtnListaAtividades_Click;
        }

        private void BtnCriarVerificacao_Click(object sender, EventArgs
e)
        {
            StartActivity(typeof(CriarVerificacaoActivity));
        }

        private void BtnCriarSetor_Click(object sender, EventArgs e)
        {
            StartActivity(typeof(CriarSetorActivity));
        }

        private void BtnCriarProduto_Click(object sender, EventArgs e)
        {
            StartActivity(typeof(CriarProdutoActivity));
        }

        private void BtnGerenciarUsuario_Click(object sender, EventArgs
e)
```

```
        {
            StartActivity(typeof(GerenciarListaUsuarioActivity));
        }

        private void BtnListaAtividades_Click(object sender, EventArgs e)
        {
            StartActivity(typeof(ListaProdutosActivity));
        }
    }
}
```

- Projeto App06Teste (Representa o ControleShopping.AndroidApp):
    - Classe VerificacaoActivity;

```
using System;
using System.Collections.Generic;
using System.Linq;
using Android.App;
using Android.Widget;
using Android.OS;
using System.Net.Http;
using static API.Core.ConfigurationManager;
using API.Data.DTO;
using System.Threading.Tasks;
using Newtonsoft.Json;
using API.HttpClient.Proxy;
using System.Reflection;
using API.Parameters;
using System.Net;
using Android.Content;
using Android.Graphics;
using Android.Views;
using Android.Provider;
using Android.Content.PM;
using Java.IO;
using System.Globalization;

namespace App06Teste
{
    [Activity(Label = "VerificacaoActivity")]
    public class VerificacaoActivity : Activity
    {
        ImageView imgView;
        Button btnScan;
        Button btnTirarFoto;
        Button btnSalvar;
        Switch switchProblema;
        private static Verificacao _verificacao;
        EditText _txtProblema;
        private static ProblemaVerificacao _problemaVerificacao;

        protected async override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);

            SetContentView(Resource.Layout.Verificacao);
            _verificacao = new Verificacao();
            // Create your application here
```

```
            try
            {

ZXing.Mobile.MobileBarcodeScanner.Initialize(Application);
                //Create your application here


                btnScan = FindViewById<Button>(Resource.Id.btnScan);
                btnTirarFoto = FindViewById<Button>(Resource.Id.btnFoto);
                btnSalvar = FindViewById<Button>(Resource.Id.btnSalvar);
                switchProblema =
FindViewById<Switch>(Resource.Id.switchProblema);
                imgView =
FindViewById<ImageView>(Resource.Id.imgQRVerificacao);
                _txtProblema =
FindViewById<EditText>(Resource.Id.txtProblema);

                _txtProblema.TextChanged += _txtProblema_TextChanged;

                btnScan.Click += btnScan_Click;
                btnTirarFoto.Click += btnTirarFoto_Click;
                btnSalvar.Click += btnSalvar_Click;
                switchProblema.CheckedChange +=
switchProblema_CheckedChange;
            }
            catch (Exception e) { SetAlerta(e.Message); }
            if (IsThereAnAppToTakePictures())
            {
                CreateDirectoryForPictures();
            }
        }

        private void _txtProblema_TextChanged(object sender,
Android.Text.TextChangedEventArgs e)
        {
            btnSalvar.Enabled = true;
        }

        protected async override void OnResume()
        {
            base.OnResume();
            int idVerificacao = Intent.GetIntExtra("idVerificacao", 0);

            if (idVerificacao == 0)
            {
                base.OnBackPressed();
            }
            if (_verificacao == null)
                _verificacao = new Verificacao();
            _verificacao.idVerificacao = idVerificacao;
            if (_problemaVerificacao == null)
                _problemaVerificacao = new ProblemaVerificacao();
            _problemaVerificacao.idVerificacao = idVerificacao;
            await GetVerificacao(idVerificacao);
        }

        private void switchProblema_CheckedChange(object sender,
CompoundButton.CheckedChangeEventArgs e)
        {
            switchProblema = (Switch)sender;
```

```
            if (switchProblema.Checked)
            {
                btnSalvar.Enabled = false;
                _txtProblema.Visibility = ViewStates.Visible;
                btnTirarFoto.Visibility = ViewStates.Visible;
            }
            else
            {
                btnSalvar.Enabled = true;
                _txtProblema.Visibility = ViewStates.Invisible;
                btnTirarFoto.Visibility = ViewStates.Invisible;
            }
        }

        private async void btnSalvar_Click(object sender, EventArgs e)
        {
            try
            {
                using (HttpClient Client = new HttpClient())
                {
                    Client.DefaultRequestHeaders.Accept.Add(new
System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json
"));
                    Client.MaxResponseContentBufferSize = 256000;
                    HistoricoVerificacaoParameter parameter = new
HistoricoVerificacaoParameter()
                    {
                        idOperacao =
(int)HistoricoVerificacaoParameter.Operacao.Create,
                        HistoricoVerificacaoJSON =
JsonConvert.SerializeObject(new HistoricoVerificacao()
                        {
                            dtVerificacao = DateTime.Now,
                            idUsuario = Session.Usuario.idUsuario,
                            idVerificacao =
_problemaVerificacao.idVerificacao
                        })
                    };
                    var uri = new
Uri(string.Format("{0}/api/HistoricoVerificacao/Post",
AppSettings.ApiUrl));

                    var requestMessage = new HttpRequestMessage()
                    {
                        RequestUri = uri,
                        Method = HttpMethod.Post
                    };
                    var response = new HttpResponseMessage();
                    try
                    {
                        HttpRequestMessage request = new
HttpRequestMessage(System.Net.Http.HttpMethod.Post, uri);
                        request.Content = GetStringContent(parameter);
                        request.Headers.Accept.Add(new
System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json
"));
                        response = await
Client.SendAsync(request).ConfigureAwait(true);
                    }
                    catch (HttpRequestException HRex)
                    {
                        SetAlerta(HRex.Message + HRex.StackTrace);
```

```
                    } catch (Exception ex)

                    {
                        SetAlerta(ex.Message + ex.StackTrace);
                    }
                    if (response.IsSuccessStatusCode)
                    {
                        SetAlerta("Sucesso");
                        string stringResult = "";
                        try
                        {
                            await
response.Content.ReadAsStringAsync().ContinueWith((ta =>
                            {
                                if (ta.Exception == null)
                                {
                                    stringResult = ta.Result;
                                }
                                else
                                {
                                    SetAlerta(ta.Exception.Message +
ta.Exception.StackTrace);
                                };
                            })); ;
                        }
                        catch (HttpRequestException HRex)
                        {
                            SetAlerta(HRex.Message + HRex.StackTrace);
                        }
                        catch (Exception ex)
                        {
                            SetAlerta(ex.Message + ex.StackTrace);
                        }
                        var proxyUsuario =
JsonConvert.DeserializeObject<ResponseProxy<Verificacao>>(stringResult);
                        if (proxyUsuario.status ==
ResponseStatus.Success)
                            _verificacao = proxyUsuario.data;
                        else
                            SetAlerta(proxyUsuario.message.description);
                        if (_verificacao.idVerificacao > 0 &&
!switchProblema.Selected)

StartActivity(typeof(MenuAdministracaoActivity));
                    }
                    else
                        SetAlerta(response.StatusCode.ToString());
                }
            }
            catch (Exception ex) { SetAlerta(ex.Message); }

            if (switchProblema.Selected)
            {
                SetAlerta("Entrou");
                try
                {
                    using (var client = new HttpClient())
                    {
                       using (var content =
                            new MultipartFormDataContent())
                        {
```

```
                            content.Add(new StreamContent(new
System.IO.MemoryStream(_problemaVerificacao.imgProblema)), "bilddatei",
"upload.jpg");
                            ProblemaVerificacaoParameter parameter = new
ProblemaVerificacaoParameter()
                            {
                                idOperacao =
(int)ProblemaVerificacaoParameter.Operacao.Create,
                                ProblemaVerificacaoJSON =
JsonConvert.SerializeObject(new ProblemaVerificacao()
                                {
                                    deProblemaVerificacao =
_problemaVerificacao.deProblemaVerificacao,
                                    idVerificacao =
_problemaVerificacao.idVerificacao
                                })
                            };

                            content.Add(GetStringContent(parameter));

                            using (
                        var message =
                            await
client.PostAsync(string.Format("{0}/api/ProblemaVerificacao/Post",
AppSettings.ApiUrl), content))
                            {
                                var input = await
message.Content.ReadAsStringAsync();
                                var _proble =
JsonConvert.DeserializeObject<ResponseProxy<ProblemaVerificacao>>(input);
                                if (_proble.status ==
ResponseStatus.Success)
                                    _problemaVerificacao = _proble.data;
                                else

SetAlerta(_proble.message.description);
                                if
(_problemaVerificacao.idProblemaVerificacao > 0)

StartActivity(typeof(MenuAdministracaoActivity));
                            }
                        }
                    }
                }
                catch (Exception ex) { throw ex; }
            }
        }

        private void btnTirarFoto_Click(object sender, EventArgs e)
        {
            if (_problemaVerificacao == null)
                _problemaVerificacao = new ProblemaVerificacao();
            _problemaVerificacao.deProblemaVerificacao =
_txtProblema.Text;
            try
            {

                Intent intent = new
Intent(MediaStore.ActionImageCapture);
                App._file = new File(App._dir,
string.Format("{1}/_verificacao.${2}_{0}.jpg",
```

```
DateTime.Now.ToString("dd/MM/yy:hhmmss"), _verificacao.deVerificacao,
_verificacao.idVerificacao));
                intent.PutExtra(MediaStore.ExtraOutput,
Android.Net.Uri.FromFile(App._file));
                StartActivityForResult(intent, 0);
            }
            catch (TaskCanceledException)
            {
                SetAlerta("TakePhoto cancelled");
            }
            catch (Exception ex)
            {
                SetAlerta(ex.Message + "\n" + ex.StackTrace);
            }
        }

        protected async override void OnActivityResult(int requestCode,
Result resultCode, Intent data)
        {
            base.OnActivityResult(requestCode, resultCode, data);

            if (resultCode == Result.Ok)
            {
                Android.Net.Uri selectedImage =
Android.Net.Uri.FromFile(App._file);
                Bitmap bitmap;

                try
                {
                    int height = Resources.DisplayMetrics.HeightPixels;
                    int width = imgView.Height;
                    BitmapFactory.Options options = new
BitmapFactory.Options { InJustDecodeBounds = true };


                    int outHeight = options.OutHeight;
                    int outWidth = options.OutWidth;
                    int inSampleSize = 1;

                    if (outHeight > height || outWidth > width)
                    {
                        inSampleSize = outWidth > outHeight
                                        ? outHeight / height
                                        : outWidth / width;
                    }


                    options.InSampleSize = inSampleSize;
                    options.InJustDecodeBounds = false;
                    Bitmap resizedBitmap =
BitmapFactory.DecodeFile(App._file.Path, options);

                    if (resizedBitmap == null || resizedBitmap.ByteCount
== 0)
                        SetAlerta("Bitmap Lixo");
                    else
                        SetAlerta(resizedBitmap.ByteCount.ToString());
                    byte[] bitmapData;
                    using (var stream = new System.IO.MemoryStream())
                    {

resizedBitmap.Compress(Bitmap.CompressFormat.Jpeg, 100, stream);
```

```
                        bitmapData = stream.ToArray();
                    }
                    if (_problemaVerificacao == null)
                        _problemaVerificacao = new ProblemaVerificacao();

                    _problemaVerificacao.imgProblema = bitmapData;

                    SetAlerta("Imagem Salva com sucesso");

SetAlerta(_problemaVerificacao.idVerificacao.ToString());

                    switchProblema.Visibility = ViewStates.Visible;
                    switchProblema.Selected = true;
                    btnSalvar.Enabled = true;

                    Toast.MakeText(this, selectedImage.ToString(),
                            ToastLength.Long).Show();
                }
                catch (Exception e)
                {
                    Toast.MakeText(this, e.Message, ToastLength.Long)
                            .Show();
                }
                GC.Collect();
            }
        }

        private async void btnScan_Click(object sender, EventArgs e)
        {
            try
            {

var scanner = new ZXing.Mobile.MobileBarcodeScanner();
                var result = await scanner.Scan();

                if (result != null)
                {
                    SetAlerta("Scanned Barcode: " + result.Text);
                    int idProdutoBarCode = 0;
                    int.TryParse(result.Text, out idProdutoBarCode);
                    if (_verificacao.idProduto != idProdutoBarCode)
                        base.OnBackPressed();
                    switchProblema.Visibility = ViewStates.Visible;
                    btnSalvar.Enabled = true;
                }
            }
            catch (Exception ex) { SetAlerta(ex.Message); }
        }

        protected async Task GetVerificacao(int idVerificacao)
        {
            try
            {
                using (HttpClient Client = new HttpClient())
                {
                    Client.DefaultRequestHeaders.Accept.Add(new
System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json
"));
                    Client.MaxResponseContentBufferSize = 256000;
                    VerificacaoParameter parameter = new
VerificacaoParameter()
                    {
```

```
                        idOperacao =
(int)VerificacaoParameter.Operacao.Get,
                        idVerificacao = idVerificacao
                    };
                    var uri = new
Uri(string.Format("{0}/api/Verificacao/Get?{1}", AppSettings.ApiUrl,
GetQueryString(parameter)));

                    var requestMessage = new HttpRequestMessage()
                    {
                        RequestUri = uri,
                        Method = HttpMethod.Get
                    };
                    var response = new HttpResponseMessage();
                    try
                    {
                        response = await Client.GetAsync(uri);
                    }
                    catch (HttpRequestException HRex)
                    {
                        SetAlerta(HRex.Message + HRex.StackTrace);
                    }
                    catch (Exception ex)
                    {
                        SetAlerta(ex.Message + ex.StackTrace);
                    }
                    if (response.IsSuccessStatusCode)
                    {
                        string stringResult = "";
                        try
                        {
                            stringResult = await
response.Content.ReadAsStringAsync();
                        }
                        catch (HttpRequestException HRex)
                        {
                            SetAlerta(HRex.Message + HRex.StackTrace);
                        }
                        catch (Exception ex)
                        {
                            SetAlerta(ex.Message + ex.StackTrace);
                        }
                        var proxyUsuario =
JsonConvert.DeserializeObject<ResponseProxy<Verificacao>>(stringResult);
                        if (proxyUsuario.status ==
ResponseStatus.Success)
                            _verificacao = proxyUsuario.data;
                        else
                            SetAlerta(proxyUsuario.message.description);
                    }
                }
            }
            catch (Exception ex) { SetAlerta(ex.Message); }
        }

        private void SetAlerta(string message)
        {
            Toast.MakeText(this, message, ToastLength.Long).Show();
        }

        private object GetQueryString(VerificacaoParameter parameter)
        {
```

```
            var properties = from p in
parameter.GetType().GetRuntimeProperties()
                             where p.GetValue(parameter, null) != null
                             select p.Name + "=" +
WebUtility.UrlEncode(p.GetValue(parameter, null).ToString());

            return String.Join("&", properties.ToArray());
        }
        private void CreateDirectoryForPictures()
        {
            App._dir = new File(
                Android.OS.Environment.GetExternalStoragePublicDirectory(
                    Android.OS.Environment.DirectoryPictures),
"CameraAppDemo");
            if (!App._dir.Exists())
            {
                App._dir.Mkdirs();
            }
        }

        private bool IsThereAnAppToTakePictures()
        {
            Intent intent = new Intent(MediaStore.ActionImageCapture);
            IList<ResolveInfo> availableActivities =
                PackageManager.QueryIntentActivities(intent,
PackageInfoFlags.MatchDefaultOnly);
            return availableActivities != null &&
availableActivities.Count > 0;
        }

        public static class App
        {
            public static File _file;
            public static File _dir;
            public static Bitmap bitmap;
        }

        private StringContent GetStringContent(object obj)
        {
            var jsonObject = JsonConvert.SerializeObject(obj);
            StringContent result = new StringContent(jsonObject,
System.Text.Encoding.UTF8, "application/json");
            return result;
        }
    }
}
```

- Projeto ControleShopping.API:
    - Classe VerificacaoActivity;

```
using System;
using System.Web;
using System.Data.Entity.Infrastructure;
using System.Data.Entity.Validation;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Reflection;
```

```
using System.Web.Http;
using System.IO;
using API.HttpClient.Proxy;
using API.Core;
using log4net;
using ControleShopping.API.Infra;
using System.Web.Http.OData;

namespace ControleShopping.API
{
    public class ApiControllerBase : ApiController
    {
        protected object responseResult { get; set; }
        protected int nuQtdRegistros { get; set; }

        protected HttpResponseMessage GetBadRequestResponse()
        {
            return Request.CreateResponse(HttpStatusCode.BadRequest,
new ResponseProxy<ResponseMessageProxy>
            {
                status = ResponseStatus.Error,
                message = new ResponseMessageProxy
                {
                    id = "-2",
                    description = "Solicitação Inválida."
                }
            });
        }

        protected HttpResponseMessage GetNotFoundResponse()
        {
            return Request.CreateResponse(HttpStatusCode.NotFound,
new ResponseProxy<ResponseMessageProxy>
            {
                status = ResponseStatus.NotFound,
                message = new ResponseMessageProxy
                {
                    id = "-5",
                    description = "Não encontrado."
                }
            });
        }

        protected HttpResponseMessage GetOkResponse()
        {
            return Request.CreateResponse(HttpStatusCode.OK, new
ResponseProxy<NullObject>
            {
                status = ResponseStatus.Success,
                data = null,
                total = 0
            });
        }

        protected HttpResponseMessage GetNotFoundResponse<T>(T
dataProxy) where T : class
        {
            return Request.CreateResponse(HttpStatusCode.OK, new
ResponseProxy<T>
            {
                status = ResponseStatus.NotFound,
                data = dataProxy,
```

```
                total = 0,
            });
        }

        protected HttpResponseMessage GetOkResponse<T>(T dataProxy,
int total = 0) where T : class
        {
            return Request.CreateResponse(HttpStatusCode.OK, new
ResponseProxy<T>
            {
                status = ResponseStatus.Success,
                data = dataProxy,
                total = total,
            });
        }

        protected HttpResponseMessage GetDeleteResponse()
        {
            return Request.CreateResponse(HttpStatusCode.OK, new
ResponseProxy<object>
            {
                status = ResponseStatus.Success
            });
        }

        protected string GetEventoLog(object logArguments)
        {
            var properties =
                logArguments.GetType()
                    .GetProperties(BindingFlags.Instance |
BindingFlags.Public | BindingFlags.InvokeMethod);

            var propertiesToJoin =
                properties.Select(
                    property =>
                        string.Format("{0}:{1}", property.Name,

logArguments.GetType().GetProperty(property.Name).GetValue(logArgumen
ts, null)))
                    .ToList();

            return string.Join(Environment.NewLine,
propertiesToJoin);
        }

        protected void LogEvento(string evento, Exception excessao)
        {
            ILog log = LogManager.GetLogger("LogApi");
            if (log.IsErrorEnabled)
            {
                var parametosRequisicao = new
                {
                    SESSION_ID =
HttpContext.Current.Request.Headers.Get("Cookie") +
Environment.NewLine,
                    URL = HttpContext.Current.Request.Url +
Environment.NewLine,
                    QUERY_STRING =
HttpContext.Current.Request.QueryString + Environment.NewLine,
                    FILE_PATH = HttpContext.Current.Request.FilePath
+ Environment.NewLine,
```

```
                    HOST_NAME =
   HttpContext.Current.Server.MachineName + Environment.NewLine,
                    USER_AGENT =
   HttpContext.Current.Request.UserAgent + Environment.NewLine,
                    EXCESSAO_SERVIDOR =
   HttpContext.Current.Server.GetLastError() + Environment.NewLine,
                    EXCESSAO_APLICACAO = excessao +
   Environment.NewLine,
                    PARAMETROS_PESQUIA = evento +
   Environment.NewLine,
                };
                log.Error(parametosRequisicao);
            }
       }
```

- Projeto ControleShopping.API:
    - Classe JobRelatorio;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Threading;
using System.Net.Mail;
using System.Net;
using ControleShopping.API.Models;
using System.Text;

namespace ControleShopping.API
{
    public class JobRelatorio
    {
        private ControleShoppingEntities db = new
ControleShoppingEntities();
        private System.Threading.Timer timer;
        private bool icJaFoi;
        public void SetUpTimer(TimeSpan alertTime)
        {
            DateTime current = DateTime.Now;
            TimeSpan timeToGo = alertTime - current.TimeOfDay;
            if (timeToGo < TimeSpan.Zero)
            {
                icJaFoi = false;
                return;//time already passed
            }
            if (!icJaFoi)
            {
                this.timer = new System.Threading.Timer(x =>
                {
                    this.EnviarRelatorio();
                    icJaFoi = true;
                }, null, timeToGo, Timeout.InfiniteTimeSpan);
            }
        }

        private void EnviarRelatorio()
        {
            var client = new SmtpClient("smtp.gmail.com", 587)
```

```
            {
                Credentials = new
NetworkCredential("Fillipemoura2@gmail.com", "Fillipe@)20"),
                EnableSsl = true
            };
            client.Send("Fillipemoura2@gmail.com",
"Fillipemoura2@gmail.com", "Relatório do dia " +
DateTime.Now.ToString("dd/MM/yy"), GetTextRelatorio());

        }

        private string GetTextRelatorio()
        {
            var historico = db.HistoricoVerificacao.Where(histo =>
histo.dtVerificacao <= DateTime.Now.AddDays(-1));
            var texto = new StringBuilder();
            texto.Append("<table>");
            texto.Append("<tr>");
            texto.Append("<th>Verificação</th>");
            texto.Append("<th>Data da Verificação</th>");
            texto.Append("<th>Existe problema</th>");
            texto.Append("</tr>");
            foreach (var verificacao in historico)
            {
                texto.Append("<tr>");
                texto.Append("<td>" +
verificacao.Verificacao.deVerificacao + "</td>");
                texto.Append("<td>" + verificacao.dtVerificacao +
"</td>");
                texto.Append("<td>" +
verificacao.Verificacao.ProblemaVerificacao.Any(problema =>
problema.dtProblemaVerificacao >= DateTime.Now.AddDays(-1)) + "</td>");
                texto.Append("</tr>");
            }
            texto.Append("</table>");
            return texto.ToString();
        }
    }
  }
```

- Projeto ControleShopping.API:
    - Classe AccountController;

```
using System;
using System.Collections.Generic;
using System.Net.Http;
using System.Security.Claims;
using System.Security.Cryptography;
using System.Threading.Tasks;
using System.Web;
using System.Web.Http;
using System.Web.Http.ModelBinding;
using Microsoft.AspNet.Identity;
using Microsoft.AspNet.Identity.EntityFramework;
using Microsoft.AspNet.Identity.Owin;
using Microsoft.Owin.Security;
using Microsoft.Owin.Security.Cookies;
using Microsoft.Owin.Security.OAuth;
```

```
using ControleShopping.API.Models;
using ControleShopping.API.Providers;
using ControleShopping.API.Results;

namespace ControleShopping.API.Controllers
{
    [Authorize]
    [RoutePrefix("api/Account")]
    public class AccountController : ApiController
    {
        private const string LocalLoginProvider = "Local";
        private ApplicationUserManager _userManager;

        public AccountController()
        {
        }

        public AccountController(ApplicationUserManager userManager,
            ISecureDataFormat<AuthenticationTicket> accessTokenFormat)
        {
            UserManager = userManager;
            AccessTokenFormat = accessTokenFormat;
        }

        public ApplicationUserManager UserManager
        {
            get
            {
                return _userManager ??
Request.GetOwinContext().GetUserManager<ApplicationUserManager>();
            }
            private set
            {
                _userManager = value;
            }
        }

        public ISecureDataFormat<AuthenticationTicket> AccessTokenFormat
{ get; private set; }

        // GET api/Account/UserInfo
        [HostAuthentication(DefaultAuthenticationTypes.ExternalBearer)]
        [Route("UserInfo")]
        public UserInfoViewModel GetUserInfo()
        {
            ExternalLoginData externalLogin =
ExternalLoginData.FromIdentity(User.Identity as ClaimsIdentity);

            return new UserInfoViewModel
            {
                Email = User.Identity.GetUserName(),
                HasRegistered = externalLogin == null,
                LoginProvider = externalLogin != null ?
externalLogin.LoginProvider : null
            };
        }

        // POST api/Account/Logout
        [Route("Logout")]
        public IHttpActionResult Logout()
        {
```

```
Authentication.SignOut(CookieAuthenticationDefaults.AuthenticationType);
            return Ok();
        }

        // GET api/Account/ManageInfo?returnUrl=%2F&generateState=true
        [Route("ManageInfo")]
        public async Task<ManageInfoViewModel> GetManageInfo(string
returnUrl, bool generateState = false)
        {
            IdentityUser user = await
UserManager.FindByIdAsync(User.Identity.GetUserId());

            if (user == null)
            {
                return null;
            }

            List<UserLoginInfoViewModel> logins = new
List<UserLoginInfoViewModel>();

            foreach (IdentityUserLogin linkedAccount in user.Logins)
            {
                logins.Add(new UserLoginInfoViewModel
                {
                    LoginProvider = linkedAccount.LoginProvider,
                    ProviderKey = linkedAccount.ProviderKey
                });
            }

            if (user.PasswordHash != null)
            {
                logins.Add(new UserLoginInfoViewModel
                {
                    LoginProvider = LocalLoginProvider,
                    ProviderKey = user.UserName,
                });
            }

            return new ManageInfoViewModel
            {
                LocalLoginProvider = LocalLoginProvider,
                Email = user.UserName,
                Logins = logins,
                ExternalLoginProviders = GetExternalLogins(returnUrl,
generateState)
            };
        }

        // POST api/Account/ChangePassword
        [Route("ChangePassword")]
        public async Task<IHttpActionResult>
ChangePassword(ChangePasswordBindingModel model)
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }

            IdentityResult result = await
UserManager.ChangePasswordAsync(User.Identity.GetUserId(),
model.OldPassword,
```

```
                model.NewPassword);

            if (!result.Succeeded)
            {
                return GetErrorResult(result);
            }

            return Ok();
        }

        // POST api/Account/SetPassword
        [Route("SetPassword")]
        public async Task<IHttpActionResult>
SetPassword(SetPasswordBindingModel model)
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }

            IdentityResult result = await
UserManager.AddPasswordAsync(User.Identity.GetUserId(),
model.NewPassword);

            if (!result.Succeeded)
            {
                return GetErrorResult(result);

        }

            return Ok();
        }

        // POST api/Account/AddExternalLogin
        [Route("AddExternalLogin")]
        public async Task<IHttpActionResult>
AddExternalLogin(AddExternalLoginBindingModel model)
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }


Authentication.SignOut(DefaultAuthenticationTypes.ExternalCookie);

            AuthenticationTicket ticket =
AccessTokenFormat.Unprotect(model.ExternalAccessToken);

            if (ticket == null || ticket.Identity == null ||
(ticket.Properties != null
                && ticket.Properties.ExpiresUtc.HasValue
                && ticket.Properties.ExpiresUtc.Value <
DateTimeOffset.UtcNow))
            {
                return BadRequest("External login failure.");
            }

            ExternalLoginData externalData =
ExternalLoginData.FromIdentity(ticket.Identity);

            if (externalData == null)
```

```
            {
                return BadRequest("The external login is already
associated with an account.");
            }

            IdentityResult result = await
UserManager.AddLoginAsync(User.Identity.GetUserId(),
                new UserLoginInfo(externalData.LoginProvider,
externalData.ProviderKey));

            if (!result.Succeeded)
            {
                return GetErrorResult(result);
            }

            return Ok();
        }

        // POST api/Account/RemoveLogin
        [Route("RemoveLogin")]
        public async Task<IHttpActionResult>
RemoveLogin(RemoveLoginBindingModel model)
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }

            IdentityResult result;

            if (model.LoginProvider == LocalLoginProvider)
            {
                result = await
UserManager.RemovePasswordAsync(User.Identity.GetUserId());
            }
            else
            {
                result = await
UserManager.RemoveLoginAsync(User.Identity.GetUserId(),
                    new UserLoginInfo(model.LoginProvider,
model.ProviderKey));
            }

            if (!result.Succeeded)
            {
                return GetErrorResult(result);
            }

            return Ok();
        }

        // GET api/Account/ExternalLogin
        [OverrideAuthentication]
        [HostAuthentication(DefaultAuthenticationTypes.ExternalCookie)]
        [AllowAnonymous]
        [Route("ExternalLogin", Name = "ExternalLogin")]
        public async Task<IHttpActionResult> GetExternalLogin(string
provider, string error = null)
        {
            if (error != null)
            {
```

```
                return Redirect(Url.Content("~/") + "#error=" +
Uri.EscapeDataString(error));
            }

            if (!User.Identity.IsAuthenticated)
            {
                return new ChallengeResult(provider, this);
            }

            ExternalLoginData externalLogin =
ExternalLoginData.FromIdentity(User.Identity as ClaimsIdentity);

            if (externalLogin == null)
            {
                return InternalServerError();
            }

            if (externalLogin.LoginProvider != provider)
            {

Authentication.SignOut(DefaultAuthenticationTypes.ExternalCookie);
                return new ChallengeResult(provider, this);
            }

            ApplicationUser user = await UserManager.FindAsync(new
UserLoginInfo(externalLogin.LoginProvider,
                externalLogin.ProviderKey));

            bool hasRegistered = user != null;

            if (hasRegistered)
            {

Authentication.SignOut(DefaultAuthenticationTypes.ExternalCookie);

                 ClaimsIdentity oAuthIdentity = await
user.GenerateUserIdentityAsync(UserManager,
                    OAuthDefaults.AuthenticationType);
                ClaimsIdentity cookieIdentity = await
user.GenerateUserIdentityAsync(UserManager,
                    CookieAuthenticationDefaults.AuthenticationType);

                AuthenticationProperties properties =
ApplicationOAuthProvider.CreateProperties(user.UserName);
                Authentication.SignIn(properties, oAuthIdentity,
cookieIdentity);
            }
            else
            {
                IEnumerable<Claim> claims = externalLogin.GetClaims();
                ClaimsIdentity identity = new ClaimsIdentity(claims,
OAuthDefaults.AuthenticationType);
                Authentication.SignIn(identity);
            }

            return Ok();
        }

        // GET
api/Account/ExternalLogins?returnUrl=%2F&generateState=true
        [AllowAnonymous]
        [Route("ExternalLogins")]
```

```
        public IEnumerable<ExternalLoginViewModel>
GetExternalLogins(string returnUrl, bool generateState = false)
        {
            IEnumerable<AuthenticationDescription> descriptions =
Authentication.GetExternalAuthenticationTypes();
            List<ExternalLoginViewModel> logins = new
List<ExternalLoginViewModel>();

            string state;

            if (generateState)
            {
                const int strengthInBits = 256;
                state =
RandomOAuthStateGenerator.Generate(strengthInBits);
            }
            else
            {
                state = null;
            }

            foreach (AuthenticationDescription description in
descriptions)
            {
                ExternalLoginViewModel login = new ExternalLoginViewModel
                {
                    Name = description.Caption,
                    Url = Url.Route("ExternalLogin", new
                    {
                        provider = description.AuthenticationType,
                        response_type = "token",
                        client_id = Startup.PublicClientId,
                        redirect_uri = new Uri(Request.RequestUri,
returnUrl).AbsoluteUri,
                        state = state
                    }),
                    State = state
                };
                logins.Add(login);
            }

            return logins;
        }

        // POST api/Account/Register
        [AllowAnonymous]
        [Route("Register")]
        public async Task<IHttpActionResult>
Register(RegisterBindingModel model)
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }

            var user = new ApplicationUser() { UserName = model.Email,
Email = model.Email };

            IdentityResult result = await UserManager.CreateAsync(user,
model.Password);

            if (!result.Succeeded)
```

```
            {
                return GetErrorResult(result);
            }

            return Ok();
        }

        // POST api/Account/RegisterExternal
        [OverrideAuthentication]
        [HostAuthentication(DefaultAuthenticationTypes.ExternalBearer)]
        [Route("RegisterExternal")]
        public async Task<IHttpActionResult>
RegisterExternal(RegisterExternalBindingModel model)
        {
            if (!ModelState.IsValid)
            {

        return BadRequest(ModelState);
    }

    var info = await Authentication.GetExternalLoginInfoAsync();
    if (info == null)
    {
        return InternalServerError();
    }

            var user = new ApplicationUser() { UserName = model.Email,
Email = model.Email };

            IdentityResult result = await UserManager.CreateAsync(user);
            if (!result.Succeeded)
            {
                return GetErrorResult(result);
            }

            result = await UserManager.AddLoginAsync(user.Id,
info.Login);
            if (!result.Succeeded)
            {
                return GetErrorResult(result);
            }
            return Ok();
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing && _userManager != null)
            {
                _userManager.Dispose();
                _userManager = null;
            }

            base.Dispose(disposing);
        }

        #region Helpers

        private IAuthenticationManager Authentication
        {
            get { return Request.GetOwinContext().Authentication; }
        }
```

```
        private IHttpActionResult GetErrorResult(IdentityResult result)
        {
            if (result == null)
            {
                return InternalServerError();
            }

            if (!result.Succeeded)
            {
                if (result.Errors != null)
                {
                    foreach (string error in result.Errors)
                    {
                        ModelState.AddModelError("", error);
                    }
                }

                if (ModelState.IsValid)
                {
                    // No ModelState errors are available to send, so
just return an empty BadRequest.
                    return BadRequest();
                }

                return BadRequest(ModelState);
            }

            return null;
        }

        private class ExternalLoginData
        {
            public string LoginProvider { get; set; }
            public string ProviderKey { get; set; }
            public string UserName { get; set; }

            public IList<Claim> GetClaims()
            {
                IList<Claim> claims = new List<Claim>();
                claims.Add(new Claim(ClaimTypes.NameIdentifier,
ProviderKey, null, LoginProvider));

                if (UserName != null)
                {
                    claims.Add(new Claim(ClaimTypes.Name, UserName, null,
LoginProvider));
                }

                return claims;
            }

            public static ExternalLoginData FromIdentity(ClaimsIdentity
identity)
            {
                if (identity == null)
                {
                    return null;
                }

                Claim providerKeyClaim =
identity.FindFirst(ClaimTypes.NameIdentifier);
```

```
                if (providerKeyClaim == null ||
String.IsNullOrEmpty(providerKeyClaim.Issuer)
                    || String.IsNullOrEmpty(providerKeyClaim.Value))
                {
                    return null;
                }

                if (providerKeyClaim.Issuer ==
ClaimsIdentity.DefaultIssuer)
                {
                    return null;
                }

                return new ExternalLoginData
                {
                    LoginProvider = providerKeyClaim.Issuer,
                    ProviderKey = providerKeyClaim.Value,
                    UserName = identity.FindFirstValue(ClaimTypes.Name)
                };
            }
        }

        private static class RandomOAuthStateGenerator
        {
            private static RandomNumberGenerator _random = new
RNGCryptoServiceProvider();

            public static string Generate(int strengthInBits)
            {
                const int bitsPerByte = 8;

                if (strengthInBits % bitsPerByte != 0)
                {
                    throw new ArgumentException("strengthInBits must be
evenly divisible by 8.", "strengthInBits");
                }

                int strengthInBytes = strengthInBits / bitsPerByte;

                byte[] data = new byte[strengthInBytes];
                _random.GetBytes(data);
                return HttpServerUtility.UrlTokenEncode(data);
            }
        }

        #endregion
    }
}
```