RMC Brothers

Gerenciador de Aplicativos - SgunService

Versão 1.0

Sistema de Cadastro Único

Índice

Histórico de Alterações

Data	Versão	Descrição	Responsável
27/09/2013	1.0	Versão inicial	Elimara Lima

1. Informações gerais sobre o serviço

1.1. Contexto

O SGun foi criado para unificação de dados dos usuários da Sogesp.

Para permitir acesso de outras aplicações aos dados dos usuários da Sogesp, foi desenvolvido um web service com o propósito de prover acesso ao Sistema de Cadastro Único.

1.2. Objetivo

O objetivo principal é disponibilizar acesso para cadastro, alteração e consulta de dados dos sistemas integrados de forma simples e unificada.

1.3. Nome

Gerenciador de Aplicativos – SgunService.

1.4. Endereço do arquivo WSDL

Ambiente de Homologação: https://sgunw-teste.sogesp.com.br/sgunservice.svc

Ambiente de Produção: https://sgunw.sogesp.com.br/sgunservice.svc

1.5. Responsável técnico

Nome: Tiago Cavalli

E-mail: tiago.cavalli@rmcbrothers.com.br

Telefone: (11) 5071-7623

1.6. Requisitos e orientações para o acesso

É requisito do SGunService o envio de dados relacionados abaixo:

- Nome do aplicativo
- Endereço de Produção
- Endereço de Teste

Primeiramente, estará sendo liberado somente os dados de token de homologação.

Após efetuado os testes de implementação, será feita a validação do aplicativo desenvolvido que utilize o WebService.

Somente após esta validação que será enviado o token de produção.

Para validação, enviar e-mail para homologacao@rmcbrothers.com.br .

2. Informações detalhadas do serviço

2.1. Tabela de Mensagem de Retorno dos Serviços

Código	Descrição
-2	Erro
-1	Falha
0	Sucesso

Formato padrão abaixo representa os retornos das chamadas do WebService.

Código representa a tabela acima e Retorno representa um array de objetos que podem ser devolvidos.

```
{
resposta : {
              codigo: 0,
              retorno: []
           }
}
```

2.2. Considerações gerais sobre parâmetros de entrada e/ou saída

Para todas as solicitações feitas ao SgunService, é validada a credencial da aplicação (token, secretToken e url da aplicação solicitante). Após esta validação, é identificada a aplicação.

Todos os métodos necessitam de permissão de execução. Essa permissão é atribuída a aplicação conforme a necessidade do cliente. Caso não tenha este acesso, será retornado o erro de código 401 Unauthorized.

2.3. Operações

2.3.1. Login

Descrição

Efetua o login do usuário.

Informações do contrato

Método: POST/GET

Url de chamada do Método: Url do serviço + `/rest/login/json`

Formato da Resposta = Json

Formato da solicitação = Json

Parâmetro de Entrada

Tipo: LoginModel

Fornece informação necessária para efetuar o login.

Parâmetro de Saída

Tipo: string/Json

Retorna mensagem conforme validação dos parâmetros de entrada:

- Sucesso: retorna o guid, cookie e o objeto Usuario.
- Erro: retorna a mensagem “Login e/ou senha inválidos”.

2.3.2. Obter Usuário Logado

Descrição

Obtêm do banco de dados o usuário logado.

Informações do contrato

Método: POST

Url de chamada do Método: Url do serviço + `/rest/get-user-logged/json`

Formato da Resposta = Json

Formato da Solicitação = Json

Parâmetro de Entrada

Tipo: UserCookieModel

Informa o guid e cookie necessários para a busca do usuário pelo serviço.

Parâmetro de Saída

Tipo: UsuarioModel

2.3.3. Obter Usuário Por ID

Descrição

Obtêm os dados do usuário informando o código do usuário.

Informações do contrato

Método: POST

Url de chamada do Método: Url do serviço + `/rest/get-user-by-id/json`

Formato da Resposta = Json

Formato da Solicitação = Json

Parâmetro de Entrada

Tipo: int

Informa o código do usuário no formato Integer.

Parâmetro de Saída

Tipo: UsuarioModel

2.3.4. Obter Usuário Por Login

Descrição

Obtêm os dados do usuário informando o login do usuário, que deve ser somente o número do CPF ou CNPJ.

Informações do contrato

Método: POST

Url de chamada do Método: Url do serviço + `/rest/get-user-by-login/json`

Formato da Resposta = Json

Formato da Solicitação = Json

Parâmetro de Entrada

Tipo: string

Informa o login do usuário.

Parâmetro de Saída

Tipo: UsuarioModel

2.3.5. Obter Todos os Usuários

Descrição

Obtêm a lista de usuários cadastrados no sistema.

Informações do contrato

Método: POST

Url de chamada do Método: Url do serviço + `/rest/get-all-users/json`

Formato da Resposta = Json

Parâmetro de Entrada

Tipo: ListModel

Informa a parametrização para geração da listagem.

Parâmetro de Saída

Tipo: ListModel<UsuarioModel>

Retorna uma lista de usuários.

2.3.6. Salvar Usuário

Descrição

Efetua a gravação do novo usuário no banco de dados.

Informações do contrato

Método: POST

Url de chamada do Método: Url do serviço + `/rest/save-user/json`

Formato da Resposta = Json

Formato da Solicitação = Json

Parâmetro de Entrada

Tipo: AssociacaoModel

Recebe os dados de cadastro do novo usuário.

Parâmetro de Saída

Tipo: string

Retorna se foi ou não executado com sucesso a gravação dos dados.

2.3.7. Atualizar Usuário

Descrição

Efetua a atualização dos dados do usuário no banco de dados.

Informações do contrato

Método: POST

Url de chamada do Método: Url do serviço + `/rest/update-user/json`

Formato da Resposta = Json

Formato da Solicitação = Json

Parâmetro de Entrada

Tipo: UsuarioUpdateModel

Recebe os novos dados do usuário já existente.

Parâmetro de Saída

Tipo: string

Retorna a resposta da execução.

2.4. Objetos

2.4.1. AssociacaoModel

Descrição

Os objetos desta model são utilizados para criação dos usuários e seus respectivos tipos.

Propriedades

Nome do campo	Tipo	Requerido	Observações
`usuario`	`UsuarioModel`	Sim	Objeto que contém os dados do usuário
`relAssociado`	`RelAssociadoModel`	Não	Objeto que contém o tipo de associado do usuário
`relNaoAssociado`	`RelNaoAssociadoModel`	Não	Objeto que contém o tipo de não associado do usuário

2.4.2. ListModel

Descrição

Fornece parâmetros para popular a lista a ser exibida.

Propriedades

Nome do campo	Tipo	Requerido	Observações
`SortCol`	`int`	Não	Índice da coluna da ordenação
`SortDir`	`string`	Não	Informa a direção da ordenação
`Search`	`string`	Não	Texto a ser filtrado
`RowStart`	`int`	Não	Índice do registro que dá início a lista a ser exibida
`RowsDisplay`	`int`	Não	Número de linhas a serem exibidas na tabela

2.4.3. ListModel<T>

Descrição

Fornece a lista do objeto solicitado conforme parâmetros de construção da lista, informado pelo objeto ListModel.

Propriedades

Nome do campo	Tipo	Requerido	Observações
List	`List<T>`	Não	Lista do objeto solicitado
`SortCol`	`int`	Sim	Índice da coluna da ordenação
`SortDir`	`string`	Sim	Informa a direção da ordenação
`Search`	`string`	Não	Texto a ser filtrado
`RowStart`	`int`	Sim	Índice do registro que dá início a lista a ser exibida
`RowsDisplay`	`int`	Sim	Número de linhas a serem exibidas na tabela
`TotalRows`	`int`	Sim	Total de linhas obtidas conforme parâmetros informados no objeto ListModel.

2.4.4. LoginModel

Descrição

Fornece as informações do login do usuário.

Propriedades

Nome do campo	Tipo	Requerido	Observações
`UserName`	`string`	Sim	Nome do usuário
`Password`	`string`	Sim	Senha do usuário

2.4.5. UserCookieModel

Descrição

Fornece informações do usuário armazenado em cookie.

Propriedades

Nome do campo	Tipo	Requerido	Observações
guid	string	Sim	Identificador do cookie
us_cookie	string	Sim	Cookie do usuário

2.4.6. UsuarioModel

Descrição

Objeto que contém os dados do usuário.

Propriedades

Nome do campo	Tipo	Requerido	Observações
Id	int	Sim, se novo, Id = 0	Número identificador do usuário
Nome	string	Sim	Nome do usuário
Login	string	Sim	CPF / CNPJ do usuário, somente número
Senha	string	Sim	Senha do usuário
Guid	Guid	Não	Guid do usuário
Rg	string	Sim	RG do usuário
Crm	string	Sim, se for associado ou não associado com os tipos médico e residente	CRM do usuário
CrmUf	string	Sim, se for associado ou não associado com os tipos médico e residente	UF do CRM do usuário
Tego	string	Não	Número do Título de Especialização em Ginecologia e Obstetrícia
DataNascimento	DateTime	Sim	Data de nascimento do usuário
Sexo	string	Sim	Sexo do usuário – F ou M
NewsLetter	bool	Não	Informa se este usuário possui Newsletter

DataAlteracao	DateTime	Não	Data da última alteração do usuário
DataInclusao	DateTime	Não	Data da inclusão do usuário
Enderecos	List<EnderecoModel>	Sim	Lista de endereços do usuário
Emails	List<EmailModel>	Sim	Lista de e-mails do usuário
Telefones	List<TelefoneModel>	Sim	Lista de telefones do usuário
Formacoes	List<FormacaoModel>	Não	Lista de formações do usuário

2.4.7. UsuarioUpdateModel

Descrição

Objeto utilizado para enviar ao serviço SgunService as atualizações feitas no registro do usuário.

Propriedades

Nome do campo	Tipo	Requerido	Observações
Id	int	Sim	Número identificador do usuário
Nome	string	Sim	Nome do usuário
Rg	string	Sim	RG do usuário
Crm	string	Sim, se for associado ou não associado com os tipos médico e residente	CRM do usuário
CrmUf	string	Sim, se for associado ou não associado com os tipos médico e residente	UF do CRM do usuário
Tego	string	Não	Número do Título de Especialização em Ginecologia e Obstetrícia
DataNascimento	DateTime	Sim	Data de nascimento do usuário
Sexo	string	Sim	Sexo do usuário – F ou M

3. Exemplo de Chamada do Serviço

O exemplo abaixo refere-se ao login do usuário. Este exemplo foi baseado em .NET de nossas bibliotecas internas.

```
public class loginHandler : IHttpHandler
{
    public void ProcessRequest(HttpContext context)
    {
        //Obtem o login e senha digitados pelo usuário
        string login = context.Request["login"];
        string senha = context.Request["senha"];

        try
        {
            //Parâmetros de modelo personalizado para cada método
            LoginModel model = new LoginModel { UserName = login, Password = senha };

            //Modos de realizar a requisição do serviço
            string json = ConsumeService.Request<LoginModel, string> (
                            WebConfigurationManager.AppSettings["PathWebService"] +
                                "/rest/login/json",
                            Credential.credential,
                            model);

            CookieManager.createCookieWithJson(json);

            context.Response.ContentType = "application/json";
            context.Response.Write(json);

        }
        catch (Exception ex)
        {
            context.Response.ContentType = "text/plain";
            context.Response.Write(ex);
        }
    }
}
```

```
public static class ConsumeService
{
   public static int RequestInt(string url,
                                CredentialModel credentials,
                                object model,
                                string method = "POST",
                                string contentType = "application/json")
   {
      return (int)Request(url, credentials, model, typeof(int), method, contentType);
   }


   public static string Request(string url,
                                CredentialModel credentials,
                                object model,
                                string method = "POST",
                                string contentType = "application/json")
   {
      return Request(url, credentials, model, typeof(string), method, contentType).ToString();
   }


   public static object Request(string url,
                                CredentialModel credentials,
                                object model,
                                Type returnType,
                                string method = "POST",
                                string contentType = "application/json")
   {
      try
      {
         WebClient client = new WebClient();
         client.Headers["Content-type"] = contentType;
         client.Headers["Referer"] = HttpContext.Current.Request.Url.AbsoluteUri;
         client.Credentials = new NetworkCredential(credentials.Token, credentials.SecretToken);

         MemoryStream stream = new MemoryStream();
         DataContractJsonSerializer serializer = new DataContractJsonSerializer(
                                                    model.GetType());
         serializer.WriteObject(stream, model);

         byte[] data = client.UploadData(url, method, stream.ToArray());

         if (returnType == null)
             returnType = typeof(string);

         stream = new MemoryStream(data);
         serializer = new DataContractJsonSerializer(returnType);
         return Convert.ChangeType(serializer.ReadObject(stream), returnType);
      }
      catch (Exception ex)
      {
      //return null;
      throw;
      }
   }
```

```
    public static TOut Request<T, TOut>(string url,
                                        CredentialModel credentials,
                                        T model,
                                        string method = "POST",
                                        string contentType = "application/json")
    {
        try
        {
            WebClient client = new WebClient();
            client.Headers["Content-type"] = contentType;
            client.Headers["Referer"] = HttpContext.Current.Request.Url.AbsoluteUri;
            client.Credentials = new NetworkCredential(credentials.Token, credentials.SecretToken);

            MemoryStream stream = new MemoryStream();
            DataContractJsonSerializer serializer = new DataContractJsonSerializer(typeof (T));
            serializer.WriteObject(stream, model);

            byte[] data = client.UploadData(url, method, stream.ToArray());

            stream = new MemoryStream(data);
            serializer = new DataContractJsonSerializer(typeof (TOut));
            return (TOut) serializer.ReadObject(stream);
        }
        catch (Exception ex)
        {
            //return default(TOut);
            throw;
        }
    }

    public static TOut Request<TOut>(string url,
                                        CredentialModel credentials,
                                        string contentType = "application/json")
    {
        try
        {
            WebClient client = new WebClient();
            client.Headers["Content-type"] = contentType;
            client.Headers["Referer"] = HttpContext.Current.Request.Url.AbsoluteUri;
            client.Credentials = new NetworkCredential(credentials.Token, credentials.SecretToken);
            byte[] data = client.DownloadData(url);

            MemoryStream stream = new MemoryStream(data);
            DataContractJsonSerializer serializer = new DataContractJsonSerializer(typeof(TOut));
            return (TOut) serializer.ReadObject(stream);
        }
        catch (Exception ex)
        {
            //return default(TOut);
            throw;
        }
    }
}
```