

O GAM

Relatos de um Nobre utilizando Genexus Access Manager

Sumário Maroto

Introdução	2
O Começo	4
A Arquitetura	9
Como eu utilizo	10
Usando o GAM	15
Roles, Permissions e Usuários	15
GAM Menu	17
GAM User	23
Alguns Outros Macetes	27
Timeout do repositório	27
Obter Token para acessar procedures externamente (rest, http, soap)	27
GAM Deploy Tool	29

Introdução

eu posso dizer que não encontrei melhor analogia para descrever o gam e o que ele faz como a comparação com uma casa

uma casa sem segurança, não tem muros, nem porta e qualquer um pode adentrar nela e fazer o que bem entender.

uma casa com segurança, possui portas que separam os cômodos uns dos outros, estas portas possuem fechaduras cada qual com sua chave específica

nem todos podem frequentar os cômodos mais íntimos de uma casa com segurança, como o quarto do casal ou das crianças.

As chaves das portas geralmente sempre ficam em posse dos donos da casa, para que estes tenham o controle de quem entra ou sai da casa

Alem disso os donos da casa também determinam o que pode ser feito dentro de um comodo da casa, assistir televisão, sentar-se no sofa e assim por diante.

Nessa abstração inicio dizendo que o gam garante a segurança da sua casa... ou do seu sistema.

Existem muitas formas de se implementar o gam, sendo o inicio primitivo estas duas:

Authentication - Se tem a chave do portão principal, tem acesso a casa toda

Authorization - para cada lugar e cada coisa que acessar ou fizer na casa, terá que ter a permissão concedida pelo dono da casa.

Porem o gam tem uma arquitetura flexivel o suficiente para que hajam n combinações de segurança de acordo com seu sistema ou negócio(de acordo com sua casa hehe), que abordarei no decorrer deste livro, deixando minha opinião sobre qual eu prefiro ou tenho mais afinidade.

OK, temos nossa casa, nossas portas e fechaduras, mas e nosso querido genexus?

Podemos dizer que ele é o severino da historia, pedreiro, arquiteto, chaveiro, marceneiro, o pau pra toda obra!

O cara que vai erguer os muros da casa e torná-la aconchegante para que desfrutemos como ninguém o que ela nos proporciona e que também a torna bela para visitaç o.

O Começo

Eu comecei com o gam pelas formas basicas, que citei ali na introdução.

Mas por algum bloqueio mental ou por falta de ligação da documentação do wiki mesmo, achava que o gam não nos poupava trabalho no genexus,

Muito pelo contrário, achava que com ele tínhamos o mesmo trabalho que sem ele, até com mais dificuldade por não haver uma documentação completa sobre sua api, fora o peso que acarretava na kb.

Minha postura era a de sempre atacar o gam, desencorajar seu uso, por não domina-lo, achava que era um produto que não passava confiança.

Até que um dia fui apresentado ao video do Mario Nantes, e as peças começaram a se encaixar um pouco pra mim:

[Curso Mário Nantes](#)

achei o vídeo fantástico, nele o mario mostra como funciona realmente o authentication, authorization e integração com o facebook.

o que mais gostei do video dele é que ele não teve que codificar nada no severi... genexus para que o sistema começasse a barrar os acesso indevidos.

Depois disso resolvi que não podia mais ficar sendo preconceituoso com o gam, minha marra tinha simplesmente caído com o vídeo o mario.

Coincidentemente em um dos meus clientes meu objetivo era o de amarrar as pontas de segurança do erp que estava sendo refeito na web, alguns módulos ja estavam na reta final para serem disponibilizados,

A modelagem de segurança artesanal do cliente não mostrava a flexibilidade necessária para a proporção e complexidade que o sistema pedia.

Percebi que aquele era o melhor momento para partir para uma abordagem diferente da tradicional.

O ERP do cliente é composto de 5 grandes sistemas, dividido cada qual com varios sub módulos, sendo que alguns determinados clientes utilizavam módulos especificos...

A princípio minha ideia para a montagem desse cenário foi:

Pensei em montar todo o ambiente de carga utilizando a api do gam, porém acabou tornando-se um trabalho pesado de programação e a api do gam não tem uma documentação clara, o que me forçou a utilizar o método de tentativa e erro

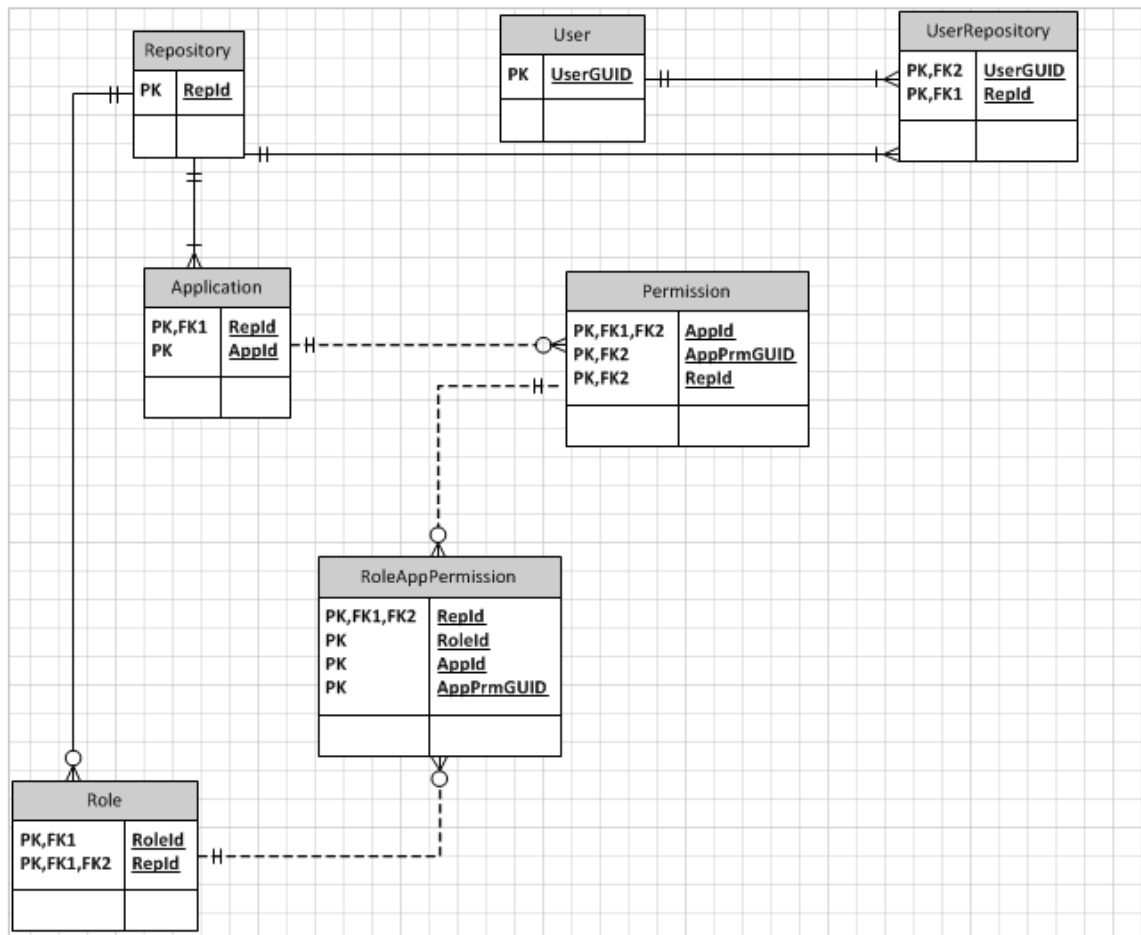
A KB tem 5 grandes sistemas dentro dela, cada qual com n módulos..., certamente é um dos cenários mais complexos que peguei até hoje e particularmente eu preferia que cada sistema

fosse uma kb diferente, tendo uma kb geral que ligasse as funcionalidades em comum entre as kbs...

Logo minha linha de planejamento com base no que este projeto representa foi:

- Sistema1 = Repositório 1
 - Módulo 1 = Role
 - Módulo 2 = Role
 - Módulo 3 = Role
 - Módulo 4 = Role
- Sistema2 = Repositório 2
 - Módulo 1 = Role
 - Módulo 2 = Role
 - Módulo 3 = Role
 - Módulo 4 = Role

A ideia parecia coerente tendo em vista a Modelagem do GAM:



Até cogitei criar cada sistema como uma application:

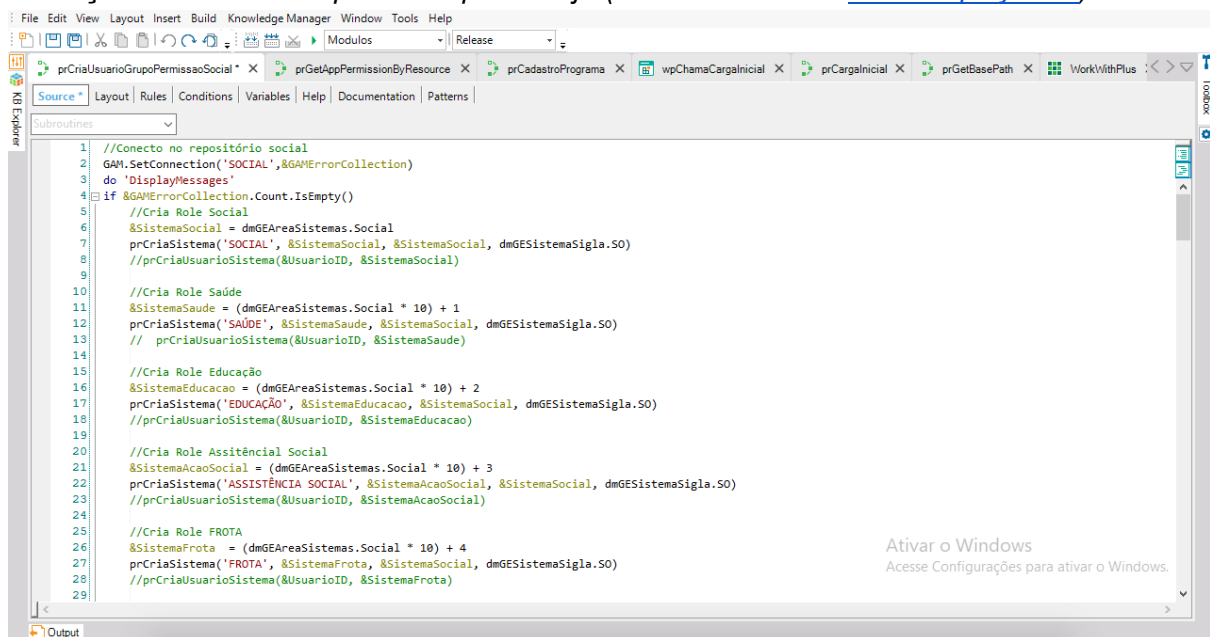
- Sistema1 = Application 1
 - Módulo 1 = Role
 - Módulo 2 = Role
 - Módulo 3 = Role
 - Módulo 4 = Role
- Sistema2 = Application 2
 - Módulo 1 = Role
 - Módulo 2 = Role
 - Módulo 3 = Role
 - Módulo 4 = Role

Mas, como disse lá trás, todos os "Sistemas" estavam dentro de uma kb, o que tornava inviável, pois cada kb debaixo do gam cria uma application dela mesma.

Logo seria meio que uma gambiarra tentar fazer desse jeito, então realmente optei pela primeira forma, pelo fato das roles não se misturarem entre os sistemas envolvidos e de ter a liberdade para criar quantos repositórios eu quisesse através do [GAM Deploy Tool](#)

No fim das contas as duas semanas iniciais de trabalho que tive resultaram nessas procedures abaixo

Começo escolhendo o repositório que desejo (criado através do [GAM Deploy Tool](#))



```
1 //Conecto no repositório social
2 GAM.SetConnection('SOCIAL', &GAMErrorCollection)
3 do 'DisplayMessages'
4 if &GAMErrorCollection.Count.IsEmpty()
5 //Cria Role Social
6 &SistemaSocial = dmGEAreaSistemas.Social
7 prCriaSistema('SOCIAL', &SistemaSocial, &SistemaSocial, dmGESistemaSigla.SO)
8 //prCriaUsuarioSistema(&UsuarioID, &SistemaSocial)
9
10 //Cria Role Saúde
11 &SistemaSaude = (dmGEAreaSistemas.Social * 10) + 1
12 prCriaSistema('SAÚDE', &SistemaSaude, &SistemaSocial, dmGESistemaSigla.SO)
13 // prCriaUsuarioSistema(&UsuarioID, &SistemaSaude)
14
15 //Cria Role Educação
16 &SistemaEducacao = (dmGEAreaSistemas.Social * 10) + 2
17 prCriaSistema('EDUCAÇÃO', &SistemaEducacao, &SistemaSocial, dmGESistemaSigla.SO)
18 //prCriaUsuarioSistema(&UsuarioID, &SistemaEducacao)
19
20 //Cria Role Assitencial Social
21 &SistemaAcaoSocial = (dmGEAreaSistemas.Social * 10) + 3
22 prCriaSistema('ASSISTÊNCIA SOCIAL', &SistemaAcaoSocial, &SistemaSocial, dmGESistemaSigla.SO)
23 //prCriaUsuarioSistema(&UsuarioID, &SistemaAcaoSocial)
24
25 //Cria Role FROTA
26 &SistemaFrota = (dmGEAreaSistemas.Social * 10) + 4
27 prCriaSistema('FROTA', &SistemaFrota, &SistemaSocial, dmGESistemaSigla.SO)
28 //prCriaUsuarioSistema(&UsuarioID, &SistemaFrota)
29
```

Procedure para criar uma role nova de maneira programática

```
&GamRole.Load(&ID)
&GamRole.Name = &Description
&GamRole.Description = &Description
&GamRole.Save()

If &GamRole.Success()
    Commit
Else
    &GamErrorCollection = &GamRole.GetErrors()
    do 'DisplayMessages'
Endif

Sub 'DisplayMessages'
    For &GamError in &GamErrorCollection
        prjavadebug(Format("!%1 (GAM%2)", &GamError.Message, &GamError.Code), &Pgmmname)
    EndFor
EndSub
```

·Após

a criação crio os menus e options:

```
// Cria Menu para Sistemas
&MenuSaudeId = 250//prCriaMenu('SAÚDE', 1, &SistemaSaude)
&MenuEducacaoId = 251//prCriaMenu('EDUCAÇÃO', 2, &SistemaEducacao)
&MenuAcaoSocialId = 252//prCriaMenu('ASSISTÊNCIA SOCIAL', 3, &SistemaAcaoSocial)
&MenuFrotaId = 253//prCriaMenu('FROTA', 4, &SistemaFrota)
&MenuSocialAdminId = 254//prCriaMenu('ADMINISTRAÇÃO', 5, &SistemaSocial)
&MenuSocialParamId = 255//prCriaMenu('PARAMETRIZAÇÕES', 6, &SistemaSocial)
&MenuSocialConfigId = 256//prCriaMenu('CONFIGURAÇÕES', 7, &SistemaSocial)
```

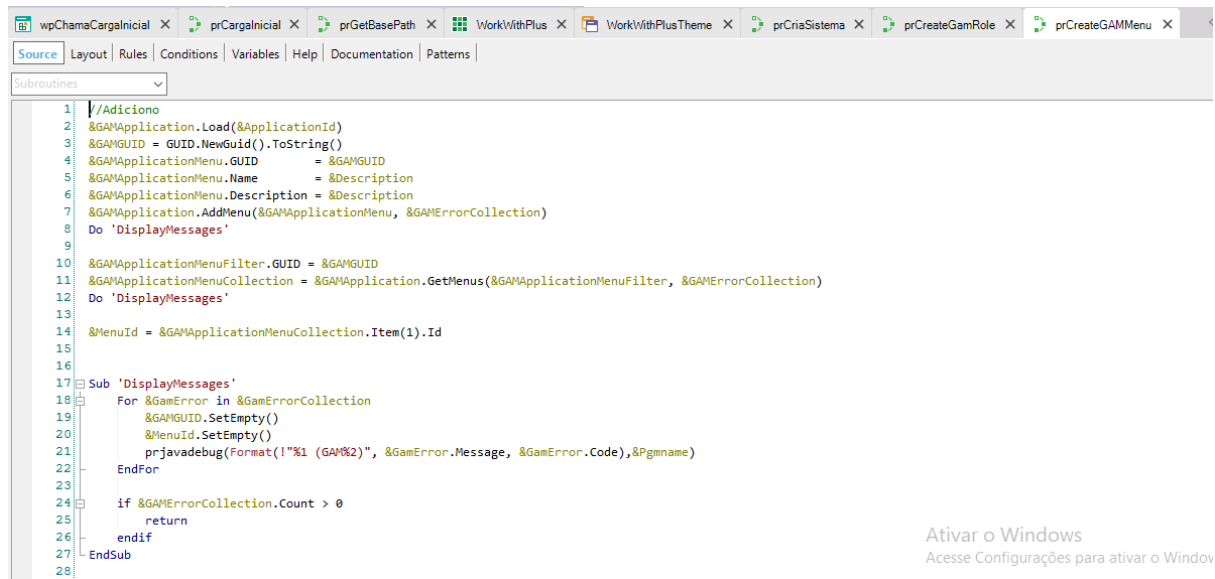
```
//////////Cadastro Geral De Programas da ListMPPPrograms
// prCadastroPrograma.Call(dmGESistemaSigla.SO, dmGEAreaSistemas.Social)
```

```
// Remove menus nao utilizados ou que mudaram de menu pai
//Do 'RemoveMenu'
```

```
// Menu Configuracoes para todos os sistemas
//Do 'CriaItensMenuConfig'
```

```
&MenuId=257//prCreateGAMMenu('Localidades', prgetApplicationId())
&MenuOptionId = prCreateGAMMenuOption(&MenuSocialParamId, &MenuId, 'Localidades', GAMMenuOptionType.Menu, prGetAppPermissionByResource('so.abastecimento'))
```

Criação de menus:



Criação de menu options:

A Arquitetura

Antes de prosseguirmos pro que eu considero o modelo ideal, gosto de explicar as principais que conheço com o GAM

•Simple

- um repositório com uma aplicação

É o gam aplicado na própria KB, com api e backend

•UM:N

- Um repositório com muitas aplicações

É quando um repositório atende apenas um cliente e este possui diversos sistemas que o utilizam o mesmo backend do gam

•N:N

- Muitos repositórios com muitas aplicações

É quando um backend possui diversos repositórios criados, um para cada cliente, da mesma maneira citada acima, cada repositório pode conter diversas aplicações

•Oauth (Multitenant)

- Um Gam conectando em outro gam

Neste cenário temos um backend servidor de gam, que replica-se para um ou mais backend's clientes, o backend cliente herda tudo que é do backend server.

Para maiores informações sobre esta configuração consulte o post do genexando:

<http://www.genexando.com/2015/11/single-sign-on.html>

Como eu utilizo

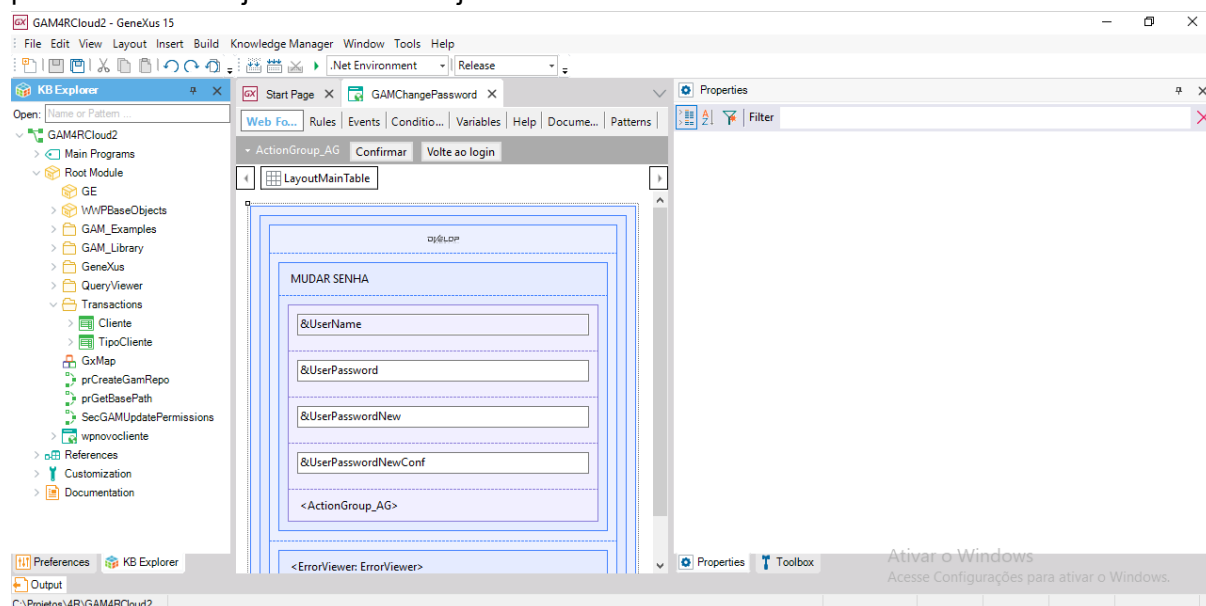
Agora que passei os conceitos base sobre o GAM nos capítulos anteriores, gostaria de compartilhar como eu utilizo o mesmo.

Como disse no capítulo anterior e em algumas conversas com conhecidos de genexus, gosto de ter uma KB somente para o GAM, fazendo um comparativo com o capítulo anterior, diria que uso as arquiteturas:

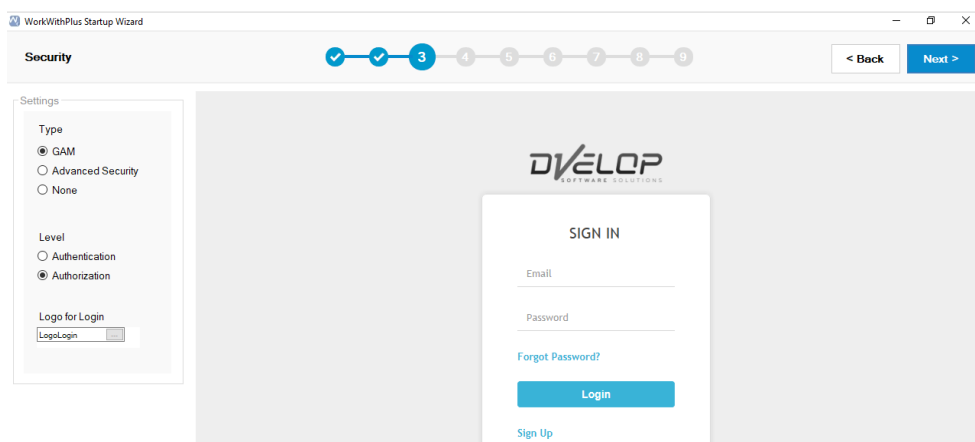
UM:N (Um repositório com muitas aplicações)

N:N (Muitos repositórios com muitas aplicações)

com backend todo configurado pelo pattern WorkWithPlus tendo em vista que por questões de produtividade ele já deixa tudo no jeito

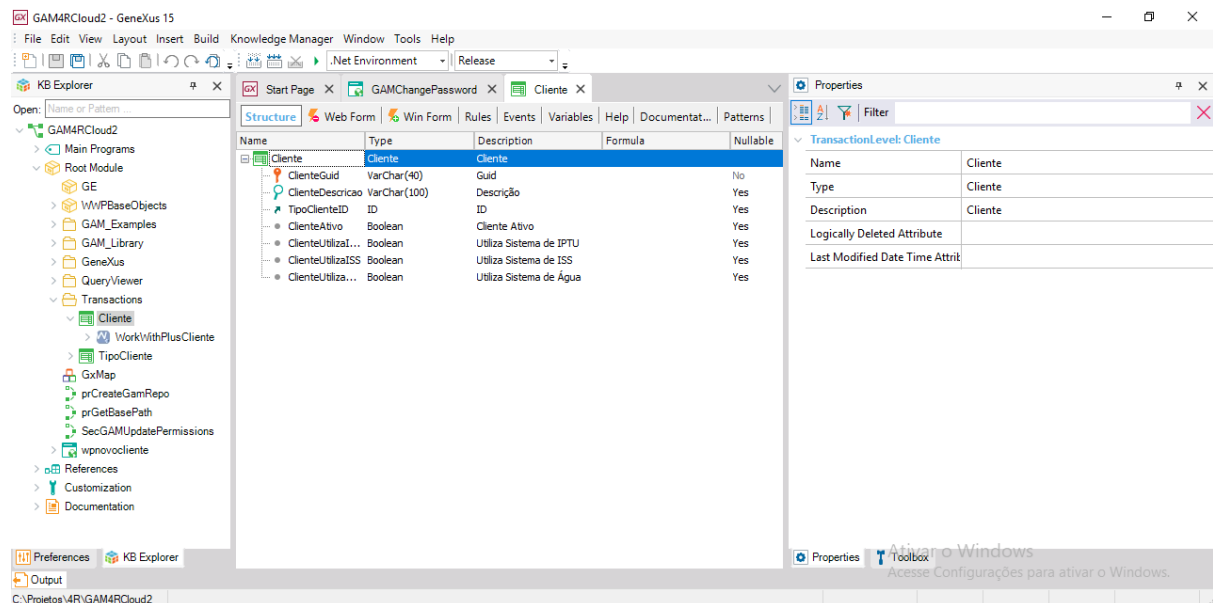


Repare na transação de Cliente na figura acima



Utilizando o backend de GAM com Authorization

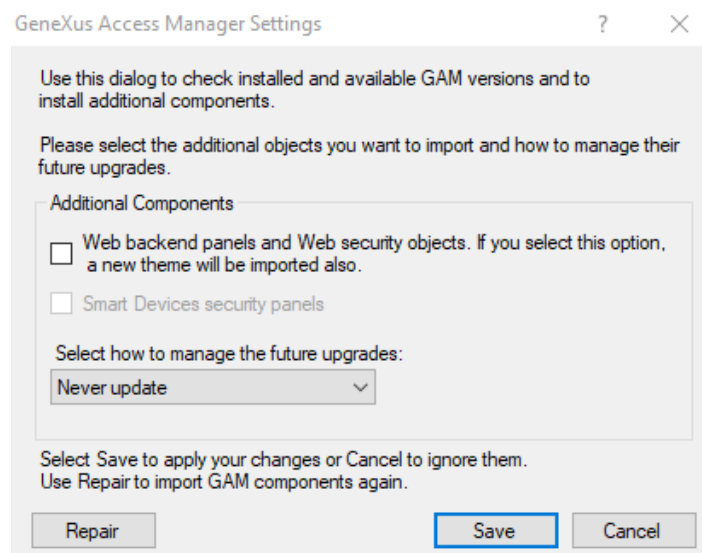
Conforme visto nas figuras acima, uso o gam com Authorization para ter mais controle sobre a segurança das aplicações que usam minha KB e tenho uma transação de cliente que mantenho nesta KB, esta transação contém em sua primary key o mesmo GUID do repositório do GAM.



Ou seja, **cada novo cliente é um novo repositório**, que pode ter um ou vários sistemas embaixo dele

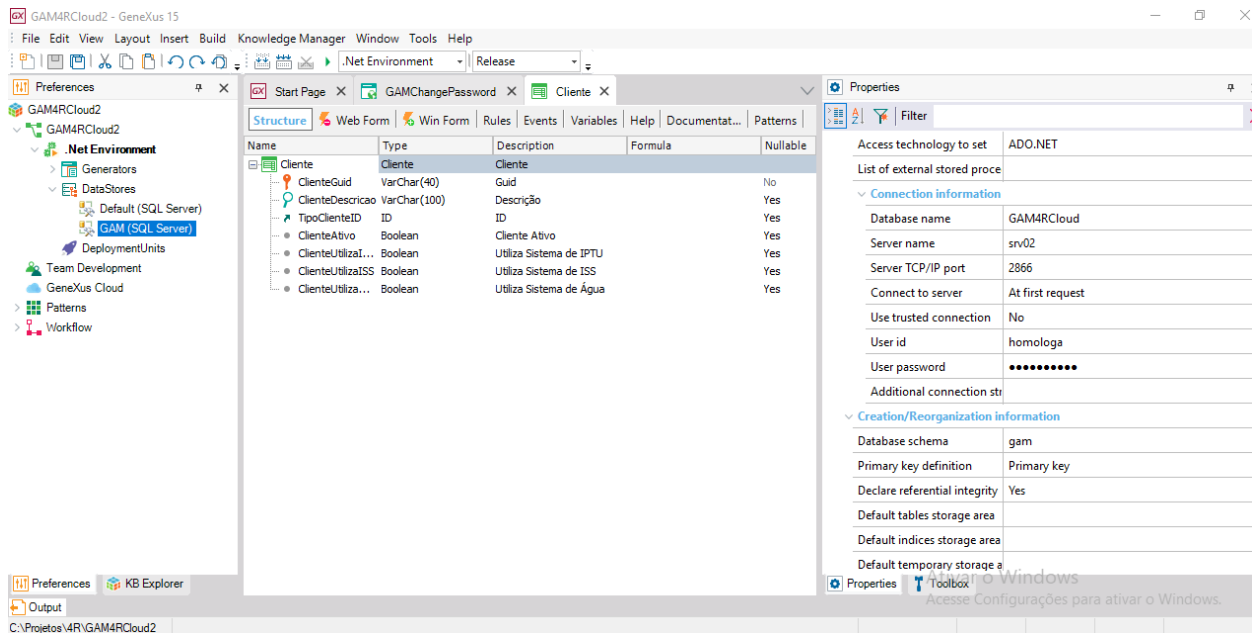
OK, mas ainda não mostrei como ligar outras KB's genexus a minha KB do GAM, o que não é nenhum bixo de sete cabeças, posso dizer que o processo consiste em 4 simples passos:

1. Aplicar somente a api do gam a KB Cliente:

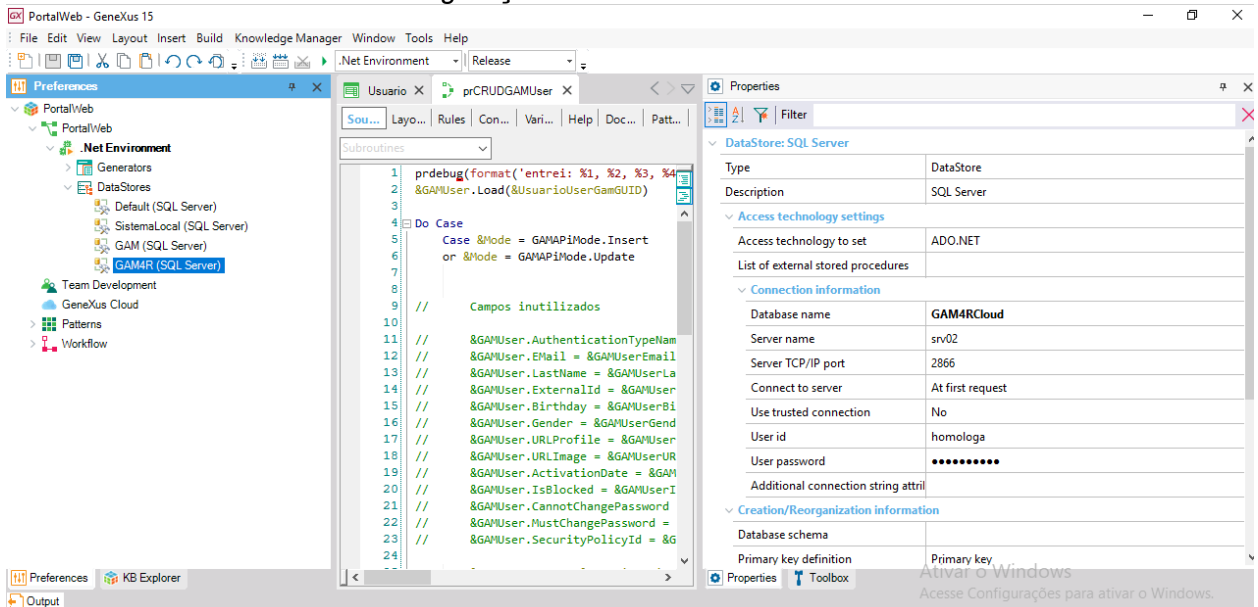


Se estiver rodando o startup wizard do WorkWithPlus na KB, pode desabilitar o GAM

2. Configurar o data store de GAM da KB Cliente para que acesse o banco de GAM da KB de GAM

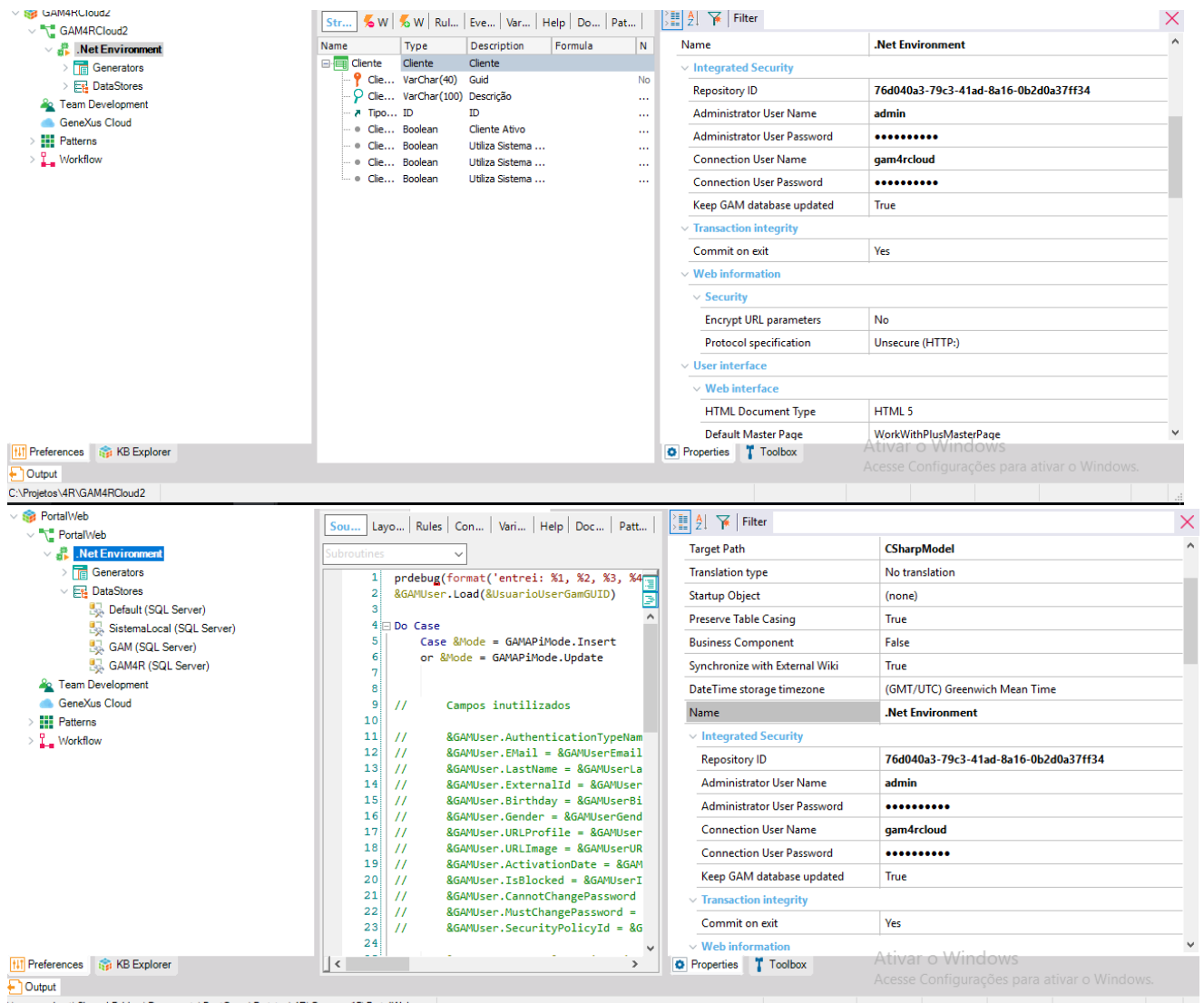


Configuração de Datastore na KB GAM



Configuração de Datastore na KB Cliente do GAM

3. Configurar a conexão ao Repositório na KB Cliente, para que o mesmo seja identico ao da KB do GAM:



Segue aqui uma breve descrição das propriedades modificadas:

- Repository ID
 - É o GUID do repositório principal, ele identifica em qual repositório sua KB irá se conectar
- Administrator User Name / Password
 - Diz respeito ao usuário administrador do backend do GAM, geralmente vem como admin e senha adminh234, mas pode ser mudado.
- Connection User Name / Password

- Diz respeito ao usuário que acessa o repositório de GAM, é o usuário que o gam usa internamente para se conectar a um repositório
- Por default este usuário leva o nome do projeto, sendo sua senha o mesmo seguido de 123
- Exemplo:
 - Usuário: gamcloud
 - Senha: gamcloud123
- O repository id, usuário e senha do repositório ficam armazenados no arquivo connection.gam gerado no ato de build da KB ou através do gamdeploytool

4. Rebuild All

Se tudo foi feito corretamente, a KB cliente irá se registrar como uma application no repositório de gam central e todos os objetos com acesso via browser da mesma serão criados como permissions embaixo desta aplicação ao término do rebuild all.

Usando o GAM

Roles, Permissions e Usuários

A primeira coisa a se fazer em um backend de gam é configurar os perfis de acesso, ou seja, definir quais os perfis de usuários irão acessar sua aplicação e o que os mesmos poderão fazer dentro dela

Fazemos isso através das **Roles**:

Permissions of Role: Administrator






BackInsertSave changes

ApplicationRuBot

Namecliente

Type(Nenhum)

Inherited?All

Permission Name	Description	Access Type	Inherited
 cliente_Delete	Cadastro Cliente Delete	Allow	<input checked="" type="checkbox"/>
 cliente_Execute	Cadastro Cliente	Allow	<input checked="" type="checkbox"/>
 cliente_FullControl	Cadastro Cliente FullControl	Allow	<input type="checkbox"/>
 cliente_Insert	Cadastro Cliente Insert	Allow	<input checked="" type="checkbox"/>
 cliente_Update	Cadastro Cliente Update	Allow	<input checked="" type="checkbox"/>

No caso da Role Administrator o genexus automaticamente já concede permissão para os objetos criados e alterados no genexus a cada build, porém para as roles novas é necessário ter em conta para que servem as permissions

As permissions existem inicialmente dentro da application, sendo possível associá-las nas roles. Para as transações são criadas permissions equivalentes aos modos que a transação possui, conforme mostrado na figura acima:

- TRNNAME_Insert
 - Permite que a transação seja acessada em modo insert
- TRNNAME_Update
 - Permite que a transação seja acessada em modo update
- TRNNAME_Delete
 - Permite que a transação seja acessada em modo delete
- TRNNAME_Execute
 - Permite que a transação seja acessada em modo display
- TRNNAME_FullControl
 - Concede acesso a todos os outros modos citados acima

O Access Type informa se é para permitir ou negar explicitamente a uma permissão. Se a permissão não está dentro de nenhuma das roles do usuário então o mesmo não tem permissão para acessar a opção correspondente no sistema.

As permissões para os outros tipos de objeto (procedure http, rest, data provider) são somente do tipo `_Execute`

Para associar uma ou mais roles ao usuário também é bem simples:

Users

Search
Role

	Roles	Password	Name	First name	Last name
<input type="button" value="✕"/>	<input type="button" value="✕"/>	<input type="button" value="✕"/>	<input type="button" value="✕"/>	admin	Administrator User
<input type="button" value="✕"/>	<input type="button" value="✕"/>	<input type="button" value="✕"/>	<input type="button" value="✕"/>	userrubotservice	Rubot Service

Selection de Usuários no GAM, clique no botão role da linha de um usuário

Roles of User: Rubot Service

Display inherit roles ☐

Role

Página 1 de 1

Clique em insert

Role

☐ Unknown

☐ Administrator

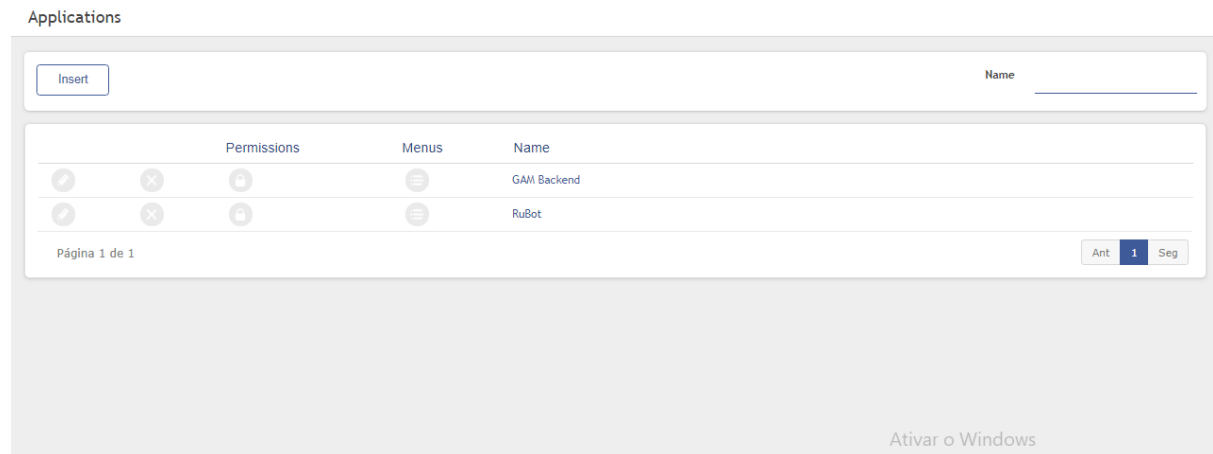
Página 1 de 1

Escolha as roles que deseja associar e saia cantando.

GAM Menu

O Menu foi uma das últimas features novas a entrarem no GAM, passando a ser oferecido a partir do Genexus 15, sendo possível utilizá-lo por aplicação dentro do GAM

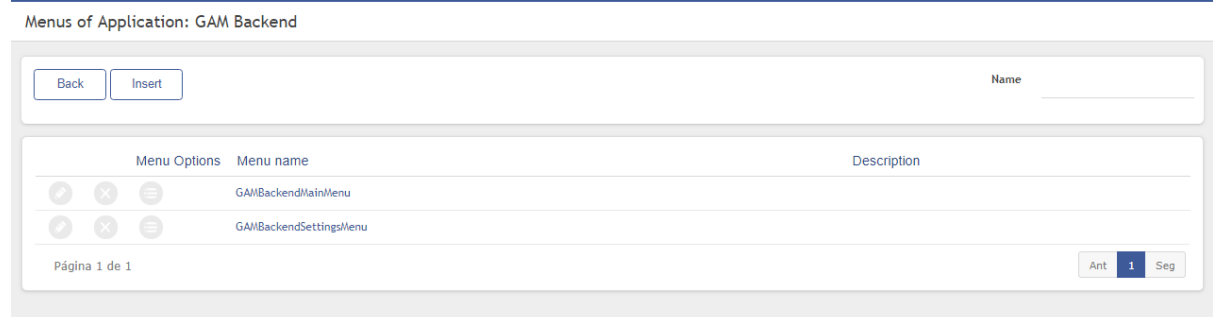
Não existe uma forma explícita para se criar um menu no gam, nem uma forma clara de associá-lo a um usuário (como uma role por exemplo), todo controle feito é pela lógica do desenvolvedor do sistema.



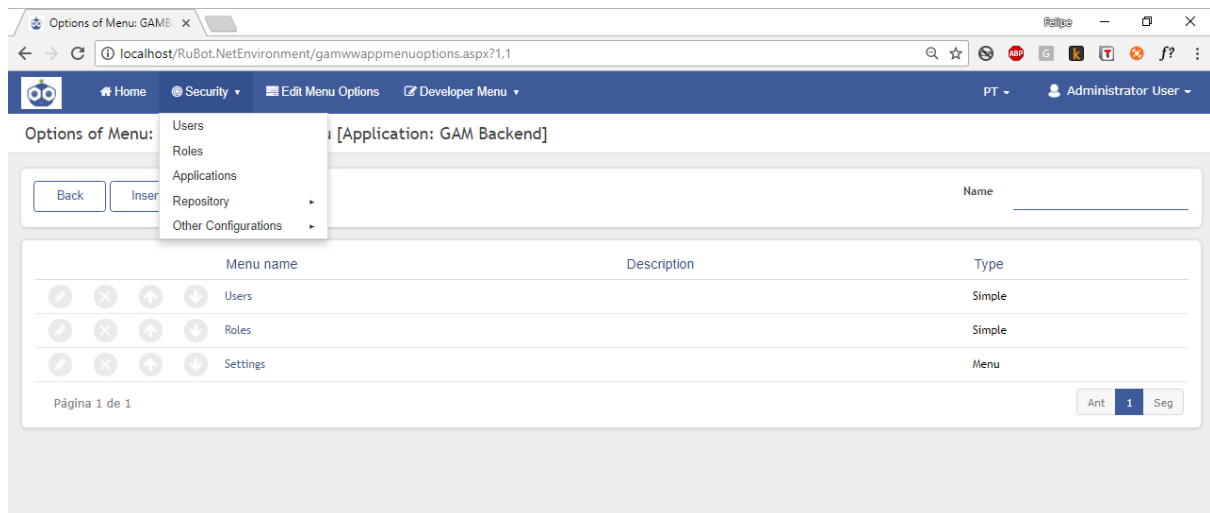
Acima os Menus do GAM Backend e da Aplicação RuBot, clique no ícone para visualizar os menus de cada uma.

A grande sacada do Menu é que você pode referenciar as permissions nele, estas já te dão o objeto que um menu irá abrir, se a permission esta negada ou não existe na role que o usuário está associado, a opção de menu simplesmente não é listada.

Além disso um menu pode ser do tipo menu, usando as opções de outro menu simples como subopções.



Aqui os dois menus principais utilizados no backend do GAM



Aqui as opções do menu GAM Backend

Abaixo as opções de um menu do tipo simples, basta selecionar a permission para que o gam carregue o objeto correspondente em Resource, além disso é possível passar parâmetros para este objeto através de Resource Parameters, usando na notação do genexus é claro.

O WorkWithPlus traz alguns mimos como poder utilizar um font icon (Font Awesome) na opção de menu e definir o target que o menu abrirá (Blank).

Aqui a visão de uma opção do menu que representa um submenu, lembra do menu mostrado na pagina 16? olha ele sendo usado nesse opção de menu da figura abaixo:

GAM App Menu Option Entry

The screenshot shows a form titled "Menu Option" with the following fields:

- Application:** GAM Backend
- Menu:** GAMBackendMainMenu
- GUID:** 8d9934db-05db-4d64-adba-5e0466gam005
- Name:** Settings
- Description:** (empty)
- Type:** Menu
- SubMenu:** GAMBackendSettingsMenu

E aqui as opções que este menu possui:

Options of Menu: GAMBackendSettingsMenu [Application: GAM Backend]

Back Insert

Menu name	Description	Type
Security Policies		Simple
Applications		Simple
Repository Configuration		Simple
Repository Connections		Simple
Authentication Types		Simple
Change Password		Simple
Change Working Repository		Simple

Página 1 de 1

Ant 1 Seg

Esta forma de cadastrar menus e submenus aparentemente confusa (realmente é confuso pra entender no início), permite que possamos carregar os menus da aplicação de maneira recursiva, com um data provider, desta forma na procedure GAMSidebarMenuOptionsData:

```

parm(&GAMMenuOptionListCollection, &MenuId);

DVEExtUC__SDTSidebarMenuOptionsData
{
    Item input &GAMMenuOptionList in &GAMMenuOptionListCollection
    {
        id      = &GAMMenuOptionList.Name
        link    = &GAMMenuOptionList.Link
        iconClass = prGetGamApplicationMenuOptionPropertyValue(&MenuId,
&GAMMenuOptionList.Id, 'IconClass')
        caption    = &GAMMenuOptionList.Description
        subitems   = GAMSidebarMenuOptionsData(&GAMMenuOptionList.Nodes, &MenuId)
    }
}

```

Procedure para pegar o icone da aplicação:

```

&GAMApplication = GamApplication.Get()
&GAMApplicationMenu = &GAMApplication.GetMenu(&MenuId, &GAMEErrorCollection)
&GAMApplicationMenuOption =
&GAMApplicationMenu.GetMenuOptionById(&GAMApplication.Id,&MenuIOptionId, &GAMEErrorCollection)

&PropertyValue = prgetGamPropertyByName(&GAMApplicationMenuOption.Properties, &PropertyName)

```

Procedura para pegar uma propriedade estendida do gam de maneira genérica
prgetGamPropertyByName:

```

For &i = 1 to &GAMPropertyCollection.Count
    if &GAMPropertyCollection.Item(&i).Id = &PropertyName
        &PropertyValue = &GAMPropertyCollection.Item(&i).Value
        return
    endif
Endfor

```

Como eu disse no começo da explicação; não existe o modo mais correto sobre como associar um menu a um usuário, existem algumas abordagens que podem ser utilizadas:

1. Fixo

Se alguma condição ocorrer, carrega-se um menu correspondente

```

if &usuario = "nobre" //condição fixa
    &GAMMenuOptionList= GamRepository.GetApplicationMenu(GamApplication.GetGUID(), "guid-do-menu-a-ser-carregado", &GAMEErrorCollection)
    &SidebarMenuOptionsData = GAMSidebarMenuOptionsData(&GAMMenuOptionList.Nodes, &GAMMenuOptionList.Id)
else
    &GAMMenuOptionList = GamRepository.GetApplicationMenu(GamApplication.GetGUID(), "guid-do-outromenu-a-ser-carregado", &GAMEErrorCollection)
    &SidebarMenuOptionsData = GAMSidebarMenuOptionsData(&GAMMenuOptionList.Nodes, &GAMMenuOptionList.Id)
endif

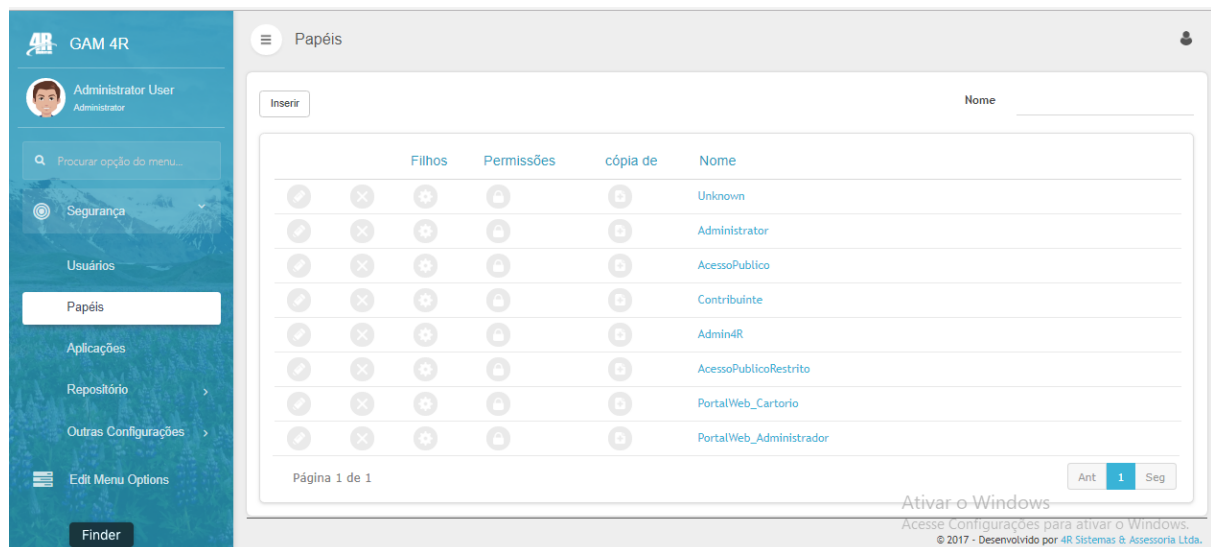
```

No caso do exemplo acima, basta trocar o "guid-do-menu-a-ser-carregado", pelo código guid do menu que deseja-se carregar

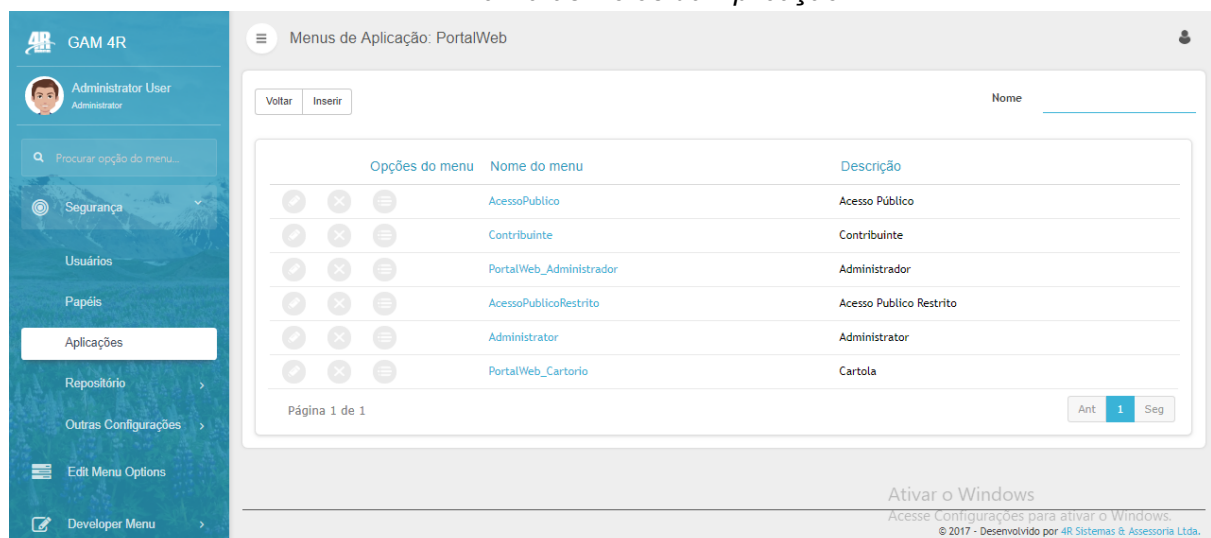
The screenshot shows a web application window titled "GAM App Menu Entry". Inside, there is a "Menu" button in the top left. Below it is a form with the following fields: "Application" with the value "GAM Backend", "GUID" with the value "6d9934db-05d5-4d54-a8ba-5e0466gam001", "Description" (empty), and "Name" with the value "GAMBackendMainMenu". At the bottom of the form are two buttons: "CONFIRMAR" and "FECHAR".

2. Menu = Role

Esta é a abordagem que utilizo, crío menus com os mesmos nomes que as minhas roles e pego a role principal do usuário que está acessando o sistema, com isto consigo deixar o carregamento do menu dinâmico de acordo com a role que o usuário possui:



Acima as Roles da Aplicação



Acima os Menus da Aplicação

Código para implementar desta forma:

```
&GamUser = GamUser.Get() //pego user que está logado
&GamRoleCollection= &GamUser.GetRoles(&GAMEErrorCollection)

if not &GamRoleCollection.Count.IsEmpty()
    &GamApplication = GamApplication.Get() //pego aplicação que estou utilizando
    &GAMApplicationMenuFilter.Name = &GamRoleCollection.Item(1).Name //pego role do user
    &GAMApplicationMenuCollection =
&GamApplication.GetMenus(&GAMApplicationMenuFilter,&GAMEErrorCollection) // pego menu
    if not &GAMApplicationMenuCollection.Count.IsEmpty()
        &GAMMenuOptionList = GamRepository.GetApplicationMenu(GamApplication.GetGUID(),
&GAMApplicationMenuCollection.Item(1).GUID, &GAMEErrorCollection) //Pego Option List do Menu
        &SidebarMenuOptionsData = GAMSidebarMenuOptionsData(&GAMMenuOptionList.Nodes,
&GAMMenuOptionList.Id) //data provider do menu que mostrei na página 18
    endif
endif
```

3. Menuzão

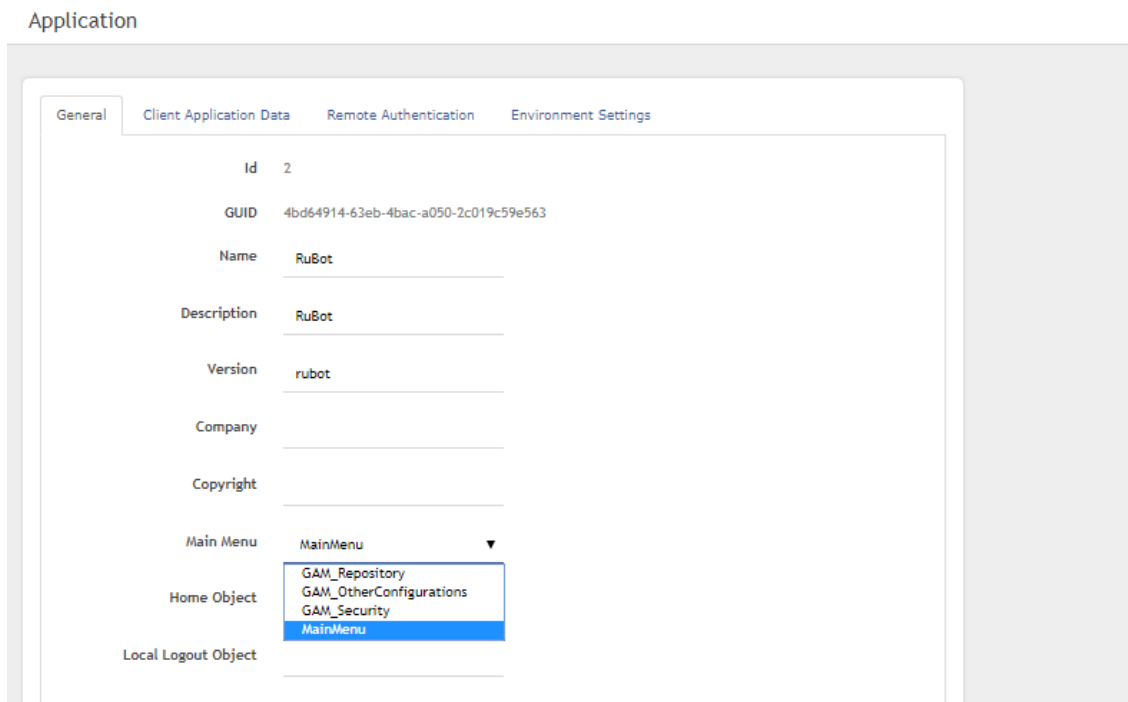
A considerar o modo de funcionamento do gam, não é necessário criar vários menus no mesmo, apenas um menu principal da aplicação e defini-lo como o menu a ser mostrado pela sua aplicação.

Digo isto pelo fato do gam não carregar as opções que o usuário não possui permissão de acesso, logo no fim das contas o menu irá exibir só o que o usuário final tem acesso, sendo que um usuário do tipo administrador tem acesso a tudo.

Bastando apenas um comando para recuperar o menu principal da application:

```
&GAMApplication.GetUserMainMenu(&GAMMenuAdditionalParameters, &Errors)
```

Application



No

cadastro de application é possível definir qual o menu principal da mesma

GAM User

Em um dado momento do meu projeto, me foi passado que era necessário que novos usuários pudessem ser cadastrados por fora do backend do gam....

Então tentei da primeira forma:

Repliquei a tela GamUserEntry para minha kb cliente e adicionei algumas propriedades na mesma que eram respectivos ao cadastro de usuário que estava fazendo na época.

Confesso que deu muito trabalho adicionar as propriedades estendidas, além disso ficava ruim para controlar somente os usuários que estavam fora do backend e que foram criados pela tela de usuários da aplicação e não pela tela do GAM e ainda por cima o cadastro de usuário não tinha a mesma dinâmica que uma transação.

Como diz um conhecido, **o molho estava ficando mais caro que o frango.**
Ou como sempre digo, **se esta difícil é porque está errado**

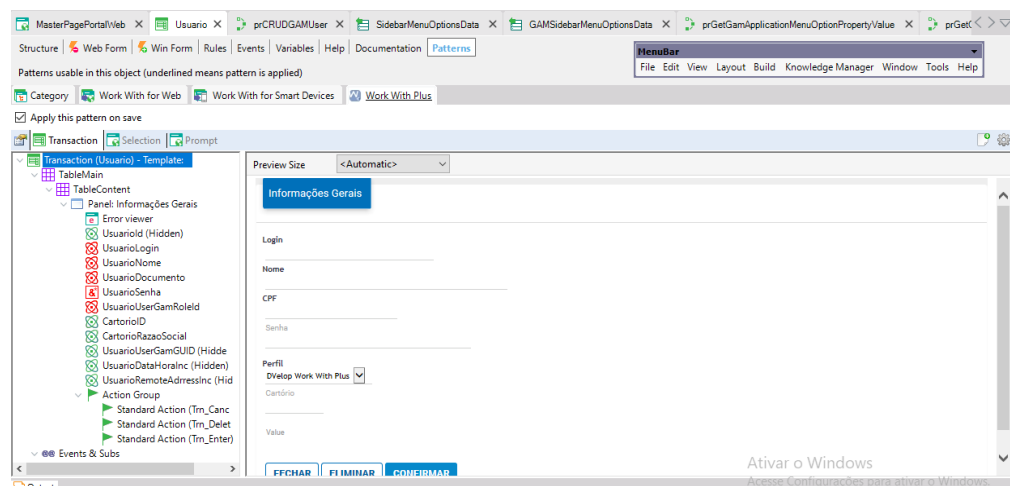
Então conversando sobre este requisito com o Luchini, ele me disse:

Por que você não cria uma tabela de usuários sua e cria um usuário correspondente no backend do gam para cada usuário que criar na sua tabela?

E fazia muito sentido este questionamento, porque essa abordagem cobria todos os defeitos da primeira abordagem:

- Mecanismo transparente para mostrar só os usuários criados pela aplicação
- Poder das transações
- Era fácil criar um usuário no backend do gam por procedure.

Enfim, consegui cumprir meu requisito no prazo que havia dado conseguindo mudar a estratégia de desenvolvimento... o cadastro de usuário através da aplicação ficou assim:



Alguns detalhes importantes sobre a figura anterior:

- Não armazeno a senha do usuário em nenhum momento na minha tabela de usuários da aplicação, esta vai direto para o user do gam.
- Associo o GUID do usuário criado ao atributo UsuarioUserGamGUID
- No combo Perfil, consigo carregar as roles do gam através de um filtro personalizado no evento start da TRN (Roles PortalWeb_, mostradas na pagina 20)

```
UsuarioUserGamRoleId.Clear()
&GamRoleFilter          = prGetGamRoleFilter(!"PortalWeb_") //pego todas roles que começam com
"PortalWeb_"
&GamRoleCollection = GamRepository.GetRoles(&GamRoleFilter, &GamErrorCollection)
UsuarioUserGamRoleId.AddItem("", "Informe um Perfil")
For &GamRole in &GAMRoleCollection
    UsuarioUserGamRoleId.AddItem(&GamRole.GUID, &GamRole.Description)
Endfor
```

- Ao término do cadastro chamo minha procedure que replica os dados no backend do gam

```
prCRUDGAMUser(TrnMode.Convert(&Mode), UsuarioLogin, UsuarioUserGamRoleId, UsuarioNome,
&UsuarioSenha, UsuarioUserGamGUID) on AfterValidate;
```

Se vier com "mimimi" de; "podia ter feito no AfterComplete", prefiro no AfterValidate pois trabalho somente com o commit da transação, abaixo o código da mesma:

```
&GAMUser.Load(&UsuarioUserGamGUID)

Do Case
    Case &Mode = GAMAPiMode.Insert
    or &Mode = GAMAPiMode.Update

//          Campos inutilizados

//          &GAMUser.AuthenticationTypeName = &GAMUserAuthenticationTypeName
//          &GAMUser.Email = &GAMUserEmail //Desativado no repositório
//          &GAMUser.LastName = &GAMUserLastName
//          &GAMUser.ExternalId = &GAMUserExternalId
//          &GAMUser.Birthday = &GAMUserBirthday
//          &GAMUser.Gender = &GAMUserGender
//          &GAMUser.URLProfile = &GAMUserURLProfile
//          &GAMUser.URLImage = &GAMUserURLImage
//          &GAMUser.ActivationDate = &GAMUserActivationDate
//          &GAMUser.IsBlocked = &GAMUserIsBlocked
//          &GAMUser.CannotChangePassword = &GAMUserCannotChangePassword
//          &GAMUser.MustChangePassword = &GAMUserMustChangePassword
//          &GAMUser.SecurityPolicyId = &GAMUserSecurityPolicyId

    &GAMUser.Name = &UsuarioLogin
    &GAMUser.Email = format('%1@4rsistemas.com.br', &UsuarioLogin)
    &GAMUser.FirstName = &UsuarioNome
    &GAMUser.IsActive = true
    &GAMUser.IsEnabledInRepository = true
    &GAMUser.DontReceiveInformation = true
    &GAMUser.PasswordNeverExpires = true

    if not &UsuarioSenha.IsEmpty()
        &GAMUser.Password = &UsuarioSenha
    endif
```

```

        &GAMUser.Save()
        &UsuarioUserGamGUID = &GAMUser.GUID //pega guid do usuário criado.
        &GamRoleCollection = &GAMUser.GetRoles(&GamErrorCollection)
        do 'GamError'

        For &GamRole in &GamRoleCollection
            &GAMUser.DeleteRole(&GamRole, &GamErrorCollection)
            do 'GamError'
        Endfor

        &GAMUser.AddRole(GAMRepository.GetRoleByGUID(&UsuarioUserGamRoleId,
        &GamErrorCollection), &GamErrorCollection)
        do 'GamError'

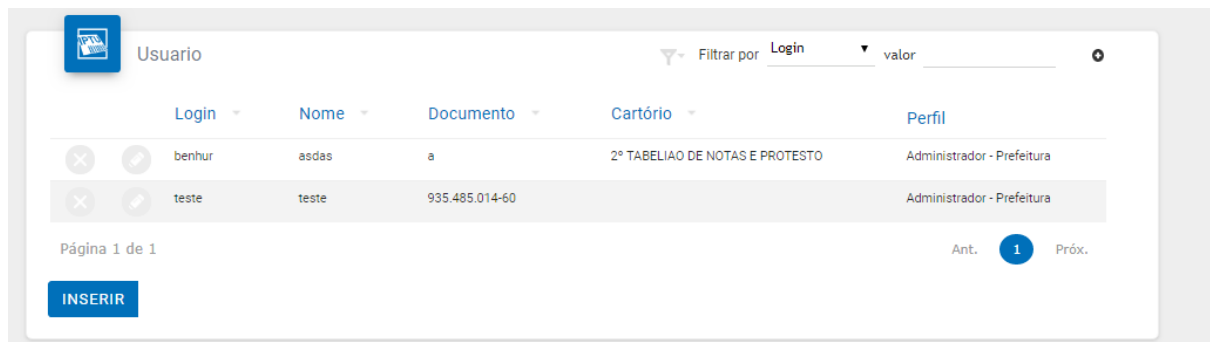
//            Endif
        Case &Mode = GAMAPiMode.Delete
            &GAMUser.Delete()
    EndCase

    If not &GAMUser.Success()
        &GamErrorCollection = &GAMUser.GetErrors()
        do 'GamError'
    Endif

    sub 'GamError'
        For &GamError in &GamErrorCollection
            PrDebug(Format("!%1 (GAM%2) %3", &GamError.Message, &GamError.Code, &Pgmname))
        EndFor
    endsub

```

Commit on exit = false



Selection da Tela de Usuário

The screenshot shows a web form for user registration. At the top left is a blue header with the text 'Informações Gerais'. Below this, there are five labeled input fields: 'Login' with the value 'Nobre', 'Nome' with the value 'Nobre GAM', 'CPF' with the value '32401096035', 'Senha' with masked characters '.....', and 'Perfil' with a dropdown menu showing 'Administrador - Prefeitura'. Below the dropdown is a small list with three options: 'Informe um Perfil', 'Cartório', and 'Administrador - Prefeitura', with the last one highlighted in blue. At the bottom left are two buttons: 'FECHAR' and 'CONFIRMAR'. At the bottom right, there is a watermark that says 'Ativar o Windows' and a smaller line of text 'Acesse Configurações para ativar o Windows'.

Transação de usuário

Não digo que a primeira abordagem que disse anteriormente é descartável, digo que ela é válida quando você só precisa criar um usuário no GAM por fora do Backend dele.

Porém se é necessário a figura de um usuário para o contexto de negócio do sistema, minha recomendação é ter este usuário como uma tabela do sistema e usar a retaguarda do gam para cuidar da segurança do mesmo.

Aplicação

Geral Dados do aplicativo do cliente Autenticação remota Configurações de ambiente

ID do Cliente 3da8628e03bb42a4823e0276fc

Segredo do cliente 0f8c1ebf1c094a019953b9acfad

☐ Acesso de usuário único?

Revogou // 00:00 REVOGAR

CONFIRMAR FECHAR

Obtenha o client id que será enviado na solicitação através do backend do gam

Utilize-o para obter o access_token

```
//First get the access_token through an HTTP POST
```

```
&addstring
```

```
= 'client_id=be47d883307446b4b93fea47f9264f88&grant_type=password&scope=FullControl&username=test&password=test'
```

```
&httpClient.Host= &server
```

```
&httpClient.Port = &port
```

```
&httpClient.BaseUrl = &urlbase + '/oauth/'
```

```
&httpClient.AddHeader("Content-Type", "application/x-www-form-urlencoded")
```

```
&httpClient.AddString(&addstring)
```

```
&httpClient.Execute('POST', 'access_token')
```

```
&httpstatus = &httpClient.StatusCode
```

```
msg('Http status: ' + &httpstatus, status)
```

```
&result = &httpClient.ToString()
```

```
&AccessTokenSDT.FromJson(&result) // Load the AccessToken in a SDT which has this structure (*)
```

```
//call DPProduct web service
```

```
&httpClient.BaseUrl = &urlbase + '/rest/'
```

```
&httpClient.AddHeader("Content-Type", "application/json")
```

```
&httpClient.AddHeader('Authorization', 'OAuth ' + &AccessTokenSDT.access_token)
```

```
&httpClient.AddHeader("GENEXUS-AGENT", "SmartDevice Application")
```

```
&httpClient.Execute('GET', 'DPProduct')
```

GAM Deploy Tool

Certo dia perguntaram no grupo, como obtenho cláusulas sql do que o gam faz? como consigo gerar uma query para serem rodadas em um servidor que não possuo acesso?

A resposta é curta é: você não faz =(

A resposta longa é: você não faz porque não precisa fazer =)

Agora falando sério, é de suma importância que o gam deploy tool tenha acesso ao banco de produção para fazer o trabalho dele e que se você desenvolvedor não tem acesso a nenhum recurso do servidor de produção onde esta hospedado o seu sistema, entregue este capítulo nas mãos dos responsáveis para que o mesmo possa utiliza-lo, do contrário, nenhum conceito que passei até agora valerá a pena

O Gam Deploy Tool controla todo o deploy do GAM para você, tanto a exportação, quanto importação dos dados do backend do gam.

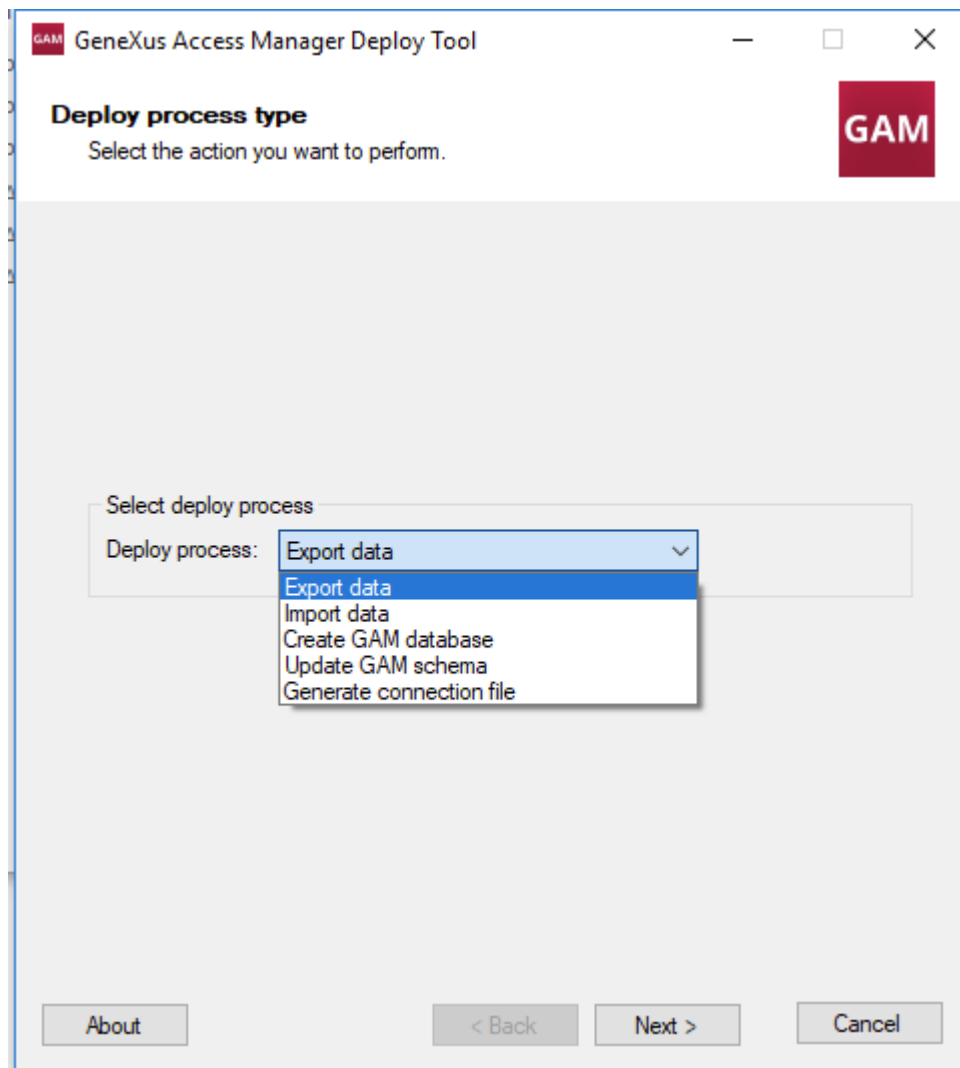
Podemos dizer que o se no backend do gam usamos o usuário admin, com o gam deploy, trabalhamos com um usuário que possui privilégios ainda mais elevados... o usuário:



Sim se o GAM é a doença, este cara é a cura.

Baixe o gam deploy tool do site da genexus, instale-o e execute em modo administrador (se não executar em modo administrador ele não vai funcionar)

A princípio o GDT dispõe das opções abaixo:



Uma breve explicação do que cada uma delas faz:

- Export data
 - Exporta todos os dados do gam ou de maneira customizada, permitindo escolher se quer exportar repositórios, roles, applications, permissions e users
 - ao fim do processo o GDT gera um arquivo empacotado com extensão .gpkg
- Import data
 - faz o processo reverso da exportação, importando os dados do pacote para um novo backend do gam.
- Create GAM Database
 - Cria a base dados do backend do gam e ja da a carga dos metadados iniciais por fora do Genexus
- Update GAM Database
 - Atualiza a estrutura do banco de dados do GAM

- Utilizado geralmente quando a genexus solta uma atualização estrutural do GAM
- Generate connection file
 - Gera um arquivo connection.gam com todos os repositórios solicitados
 - Utilizo muito esta opção para poder trabalhar com diversos repositórios, inclusive com o repositório secreto **gam-manager**, que é o repositório dos repositórios

para utilizar qualquer uma das opções acima precisamos do nosso super usuário:

usuário: **gamadmin**

senha: **gamadmin123** (é a senha default)

O link abaixo explica detalhadamente todas elas:

<https://wiki.genexus.com/commwiki/servlet/wiki?18608,GAM+Deploy+Tool>,

As opções que mais utilizo são

Export

Import

Generate connection file

Dentro da arquitetura que comentei no início do livro, trabalho com toda a minha prototipação do sistema e exporto o meu repositório como um "template" a ser utilizado pelos novos repositórios.

Assim posso criar novos repositórios com todos os dados que foram carregados neste template através da opção Import do GDT.

Depois que importar o novo repositório é importante atualizar o arquivo connection.gam, para que os outros repositórios possuam visibilidade perante a aplicação utilizada e para o próprio backend.

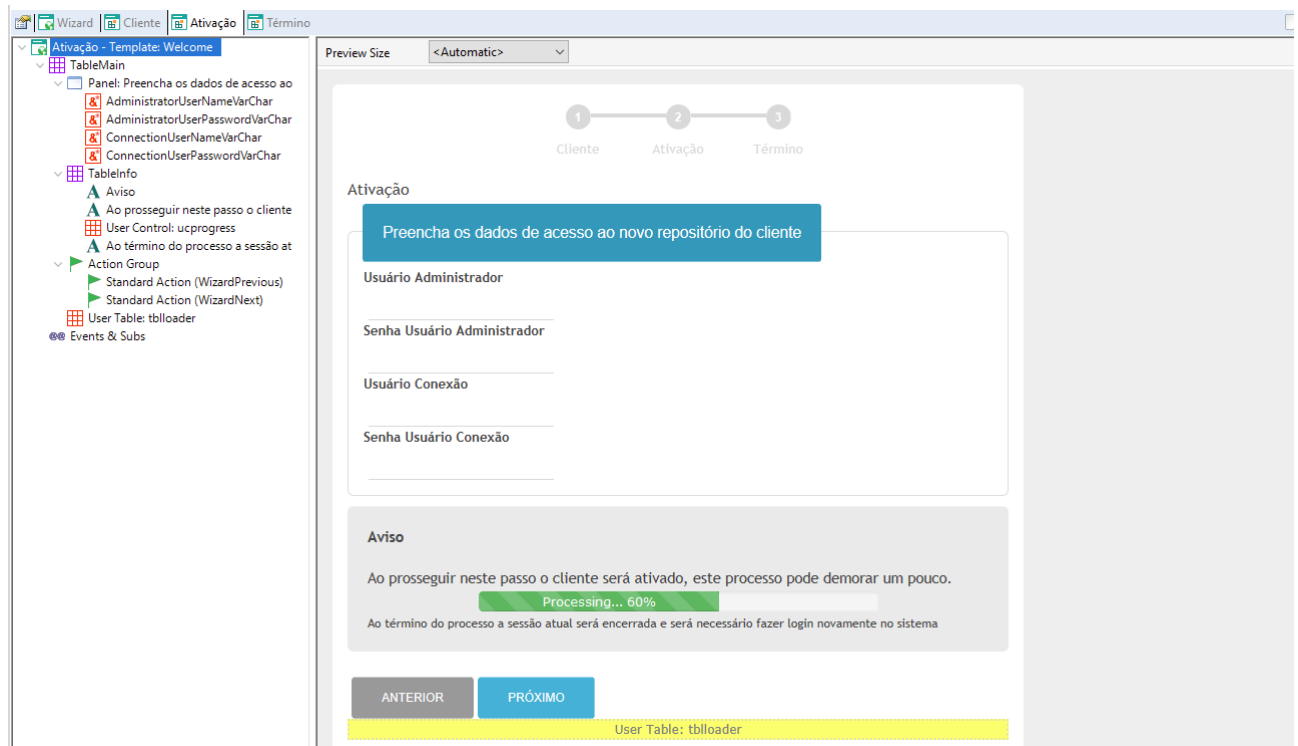
Tudo que citei acima seria a maneira mais manual de se fazer o trabalho, atualmente disponho de uma solução que automatiza os dois últimos passos.

Esta solução envolve o uso da API Repository do Gam, que possui um método responsável por criar novo repositório a partir de um arquivo .gpkg do gam.

Na próxima página segue com exclusividade o código que faz isso

Existem alguns pré requisitos para este código funcionar adequadamente:

- É necessário ter o repositório gam-manager no connection.gam
- É necessário descompactar a pasta do gpkg exportado anteriormente e referenciá-la para que a api gam repository consiga importar o pacote para um novo repositório
- É possível definir nome e senha do usuário admin e de conexão do novo repositório
- Esta procedure deve ser chamada como um command line ou através de submit
- No meu caso chamo com submit e demonstro o progresso para o usuário final através de webnotifications e progressbar



```
//wpNovoClienteAtiva
```

```
Sub 'CheckRequiredFields'
```

```
/* Generated by DVelop Work With Plus Pattern [Start] - Do not change */
```

```
&CheckRequiredFieldsResult = True
If &AdministratorUserNameVarChar.IsEmpty()
    Msg('Usuário Administrador é obrigatório.')
    &CheckRequiredFieldsResult = False
EndIf
If &AdministratorUserPasswordVarChar.IsEmpty()
    Msg('Senha Usuário Administrador é obrigatório.')
    &CheckRequiredFieldsResult = False
EndIf
If &ConnectionUserNameVarChar.IsEmpty()
    Msg('Usuário Conexão é obrigatório.')
    &CheckRequiredFieldsResult = False
EndIf
If &ConnectionUserPasswordVarChar.IsEmpty()
    Msg('Senha Usuário Conexão é obrigatório.')
    &CheckRequiredFieldsResult = False
EndIf
```

```
/* Generated by DVelop Work With Plus Pattern [End] - Do not change */
```

```

        if &CheckRequiredFieldsResult
            Do 'SaveVariablesToWizardData'
            Do 'CreateGamRepo'
            &HasValidationErrors = true
        endif

EndSub

Event onmessage(&NotificationInfo)

    if &NotificationInfo.Message.IndexOf('sucesso') > 0
        wpnovocliente(!'Ativa', !'Final', false)
    else
        //msg(&NotificationInfo.Message)
        Explanation.Caption = &NotificationInfo.Message
    endif

EndEvent

Sub 'CreateGamRepo'
    &WizardData.Cliente.ClienteGuid = guid.NewGuid().ToString()
    prCreateGamRepo.Submit('', &WizardData, &GAMErrorCollection)
EndSub

```

```

//prCreateGAMREPO
//Parm(&wpnovoclienteData, &Errors);

// SetConnection to the GAM Manager Repository
&isOK = GAM.SetConnection('GAM-Manager', &Errors)

&ProgressIndicator.Type = ProgressIndicatorType.Determinate
&ProgressIndicator.MaxValue = 100
&ProgressIndicator.ShowWithTitle("Ativando Cliente...")
&ProgressIndicator.Value = 1

if not &isOK
do "Process Errors"
else

    // Login to the New Repository (*) if needed
    &User="gamadmin"
    &password="gamadmin123"

    &isOK = GAMRepository.Login(&User, &Password, &AdditionalParameter,

```

```

&Errors)//verificar esta linha
    msg('linha 15' + &isOK.ToString(), status)
    if not &isOK
        do "Process Errors"
    else

        // New Repository Data
        &GAMRepositoryCreate.GUID = GUID.NewGuid().ToString() //&GAMGUID
is GAMGUID data type
        //sobre o GUID da linha de cima -> inventei um novo GUID
        &GAMRepositoryCreate.NameSpace = 'Gam4RCloud'
        &GAMRepositoryCreate.CanRegisterUsers = true
        &GAMRepositoryCreate.GiveAnonymousSession = true
        &GAMRepositoryCreate.Name =
&wpnovoclienteData.Cliente.ClienteGuid
        &GAMRepositoryCreate.Description =
&wpnovoclienteData.Cliente.ClienteDescricao//'Test Nobre Description'
        //Administrator user which is going to be created in the
repository
        &GAMRepositoryCreate.AdministratorUserName =
&wpnovoclienteData.Ativa.AdministratorUserNameVarChar//'admin'
        &GAMRepositoryCreate.AdministratorUserPassword =
&wpnovoclienteData.Ativa.AdministratorUserPasswordVarChar//'admin123'
        //Connection user which is going to be created in the new
repository
        &GAMRepositoryCreate.ConnectionUserName =
&wpnovoclienteData.Ativa.ConnectionUserNameVarChar
        &GAMRepositoryCreate.ConnectionUserPassword =
&wpnovoclienteData.Ativa.ConnectionUserPasswordVarChar
        &GAMRepositoryCreate.AllowOAuthAccess = true

        &PackageDirectoryPathVarChar = prGetBasePath() + &file.Separator +
!'GAM_package_GAMPackage' + &file.Separator + !'Data'
        msg('linha 39' + &PackageDirectoryPathVarChar, status)
        &directory.Source = &PackageDirectoryPathVarChar
        if &directory.Exists()

            &ProgressIndicator.Value = 10
            // New Repository Configuration
            &GAMImportRepositoryConfiguration.PackageDirectoryPath
=
&PackageDirectoryPathVarChar// "C:\Projetos\4R\Genexus15\PortalWeb\CSharpModel\
Web\GAM_package_GAMPackage\Data" // for example: 'C:\GAM\Data'
            // sobre a linha de cima, fiz um export Full do package
desta KB

```

```

        &GAMImportRepositoryConfiguration.RepositoryCreate
    = &GAMRepositoryCreate
        &GAMImportRepositoryConfiguration.UpdateConnectionFile
    = true //do you want to update the connection.gam file online?
        &GAMImportRepositoryConfiguration.AdministratorRole =
'56c5e448-220b-4de8-a695-cc0e02acdee3' //GAM GUID of "Administrator" Role in
the package
        //sobre o GUID da linha de cima, olhei pelo sistema qual era
o GUID do usuário administrator.

        // New Repository creation
        &isOK =
GAM.CreateRepositoryFromPackage(&GAMImportRepositoryConfiguration,
&Errors)//&Errors is collection of GAMError
        if not &isOK
            do "Process Errors"
            msg('linha 58' + &isOK.ToString(), status)
        else
            &ProgressIndicator.Value = 90

            do "Process Errors"
            If &isOK
                For each
                    where ClienteGuid =
&wpnovoclienteData.Cliente.ClienteGuid
                        ClienteAtivo = &isOK
                    When None
                        New
                            ClienteGuid =
&wpnovoclienteData.Cliente.ClienteGuid
                            ClienteAtivo = &isOK
                            ClienteDescricao =
&wpnovoclienteData.Cliente.ClienteDescricao
                            ClienteUtilizaAGUA =
&wpnovoclienteData.Cliente.ClienteUtilizaAGUA
                            ClienteUtilizaIPTU =
&wpnovoclienteData.Cliente.ClienteUtilizaIPTU
                            ClienteUtilizaISS =
&wpnovoclienteData.Cliente.ClienteUtilizaISS
                            TipoClienteID =
&wpnovoclienteData.Cliente.TipoClienteID
                        Endnew
                endfor
            EndIf

```

```

        &ProgressIndicator.Value = 99

        &ProgressIndicator.Hide()

        &notificationinfo.Message='sucesso'
        &webnotification.Notify(&notificationinfo)
    endif

    else
        &Error = new()
        &Error.Code = GAMErrorMessages.RepositoryNotFound
        &Error.Message = 'Não foi possível encontrar o diretório com
os dados do novo repositório'
        &Errors.Add(&Error)
        Do "Process Errors"
    endif

endif

endif

endif

////If needed, SetConnection to the New Repository
//&isOK = GAM.SetConnection('TestNobre', &Errors)
//msg('linha 49' + &isOK.ToString(), status)
//do "Process Errors"
//
//// Login to the New Repository (*) if needed
//&User="admin"
//&password="admin123"
//
//&isOK = GAMRepository.Login(&User, &Password, &AdditionalParameter,
&Errors)//verificar esta linha
//msg('linha 57' + &isOK.ToString(), status)
//do "Process Errors"

Sub "Process Errors"
    &ErrorVarChar.SetEmpty()
    &ErrorVarChar = "Houve um ou mais erros no processo de ativação: "
    For &Error in &Errors
        Msg(Format(!"%1 (GAM%2)", &Error.Message, &Error.Code), status)

        &ErrorVarChar += Format(!"%1 (GAM%2)", &Error.Message,
&Error.Code)
    EndFor

    if &Errors.Count > 0

```

```
&notificationinfo      = new()  
&notificationinfo.Id    = "Error"  
&notificationinfo.Message=&ErrorVarChar//&Errors.ToJson()  
&ProgressIndicator.Value = 100  
&ProgressIndicator.Hide()  
&Ret = &webnotification.Notify(&notificationinfo)  
msg('Status do comando: ' + &Ret.ToString(),status)  
//return  
endif  
  
EndSub
```

Enfim, acredito que toda minha experiência com gam se resumem a estas quase quarenta páginas do livro, ainda faltam assuntos de destaque como:

- autenticação pelo facebook,
- single sign on

Porém estes são temas que ainda não tive a oportunidade de desbravar com o gam e que para efeito geral ainda não foram solicitados pelos meus clientes.

Tenho em mente dar continuidade a este material com o que venho aprendendo e oferecer um curso onde demonstro tudo que ensinei aqui na prática.

Agradeço muito pela sua paciência por ter lido este material e torço pelo seu sucesso por este que é um tema polêmico neste mar de genexus, porem espero que este livro mostre que ele em sua maior parte não é nenhum bixo de sete cabeças.

