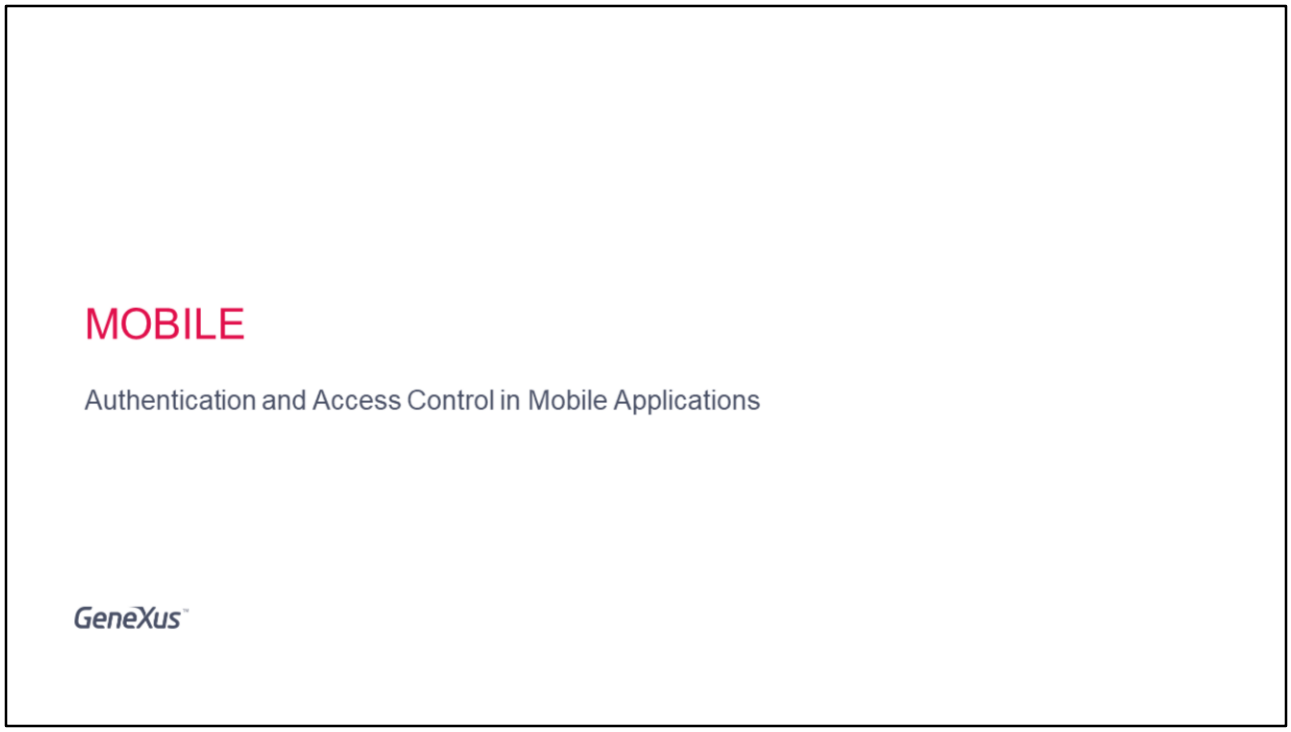

GeneXus™
by Globant

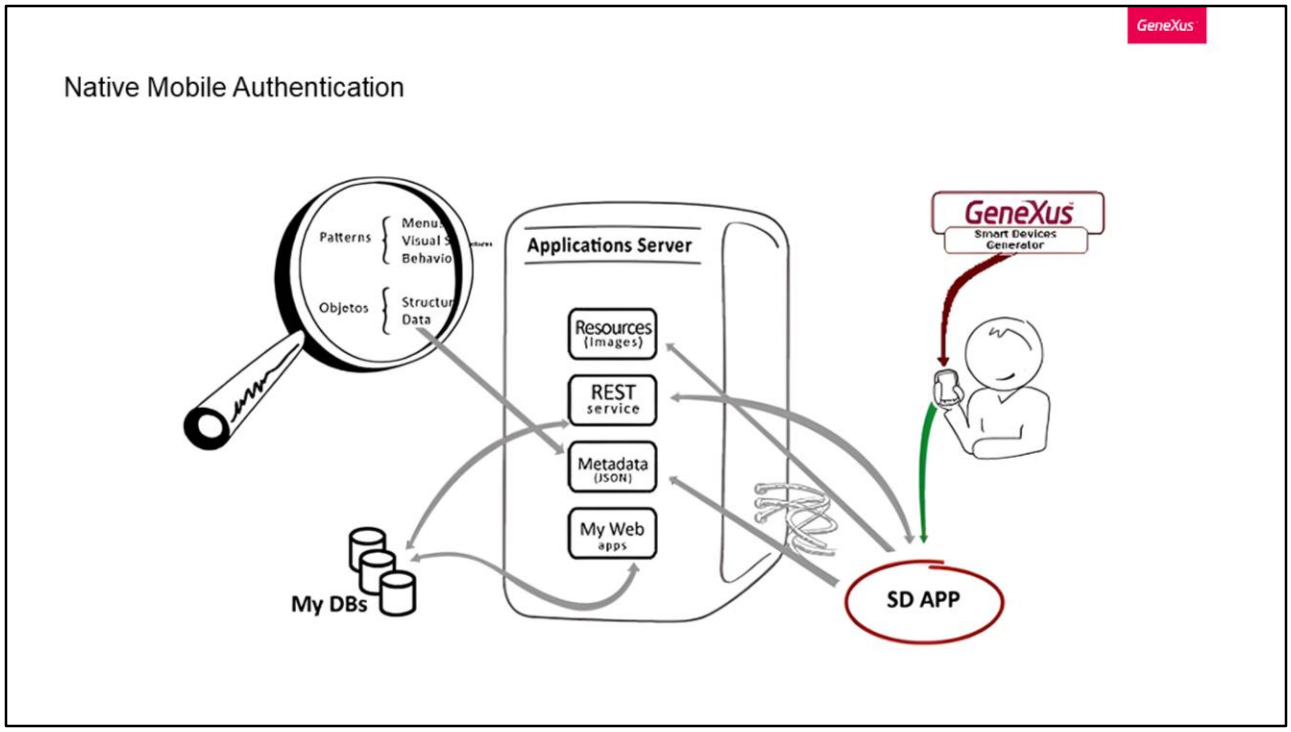
MOBILE

Nicolas Adrién

GeneXus



Neste vídeo falaremos sobre a autenticação e o controle de acesso nas aplicações móveis de GeneXus, usando GAM.

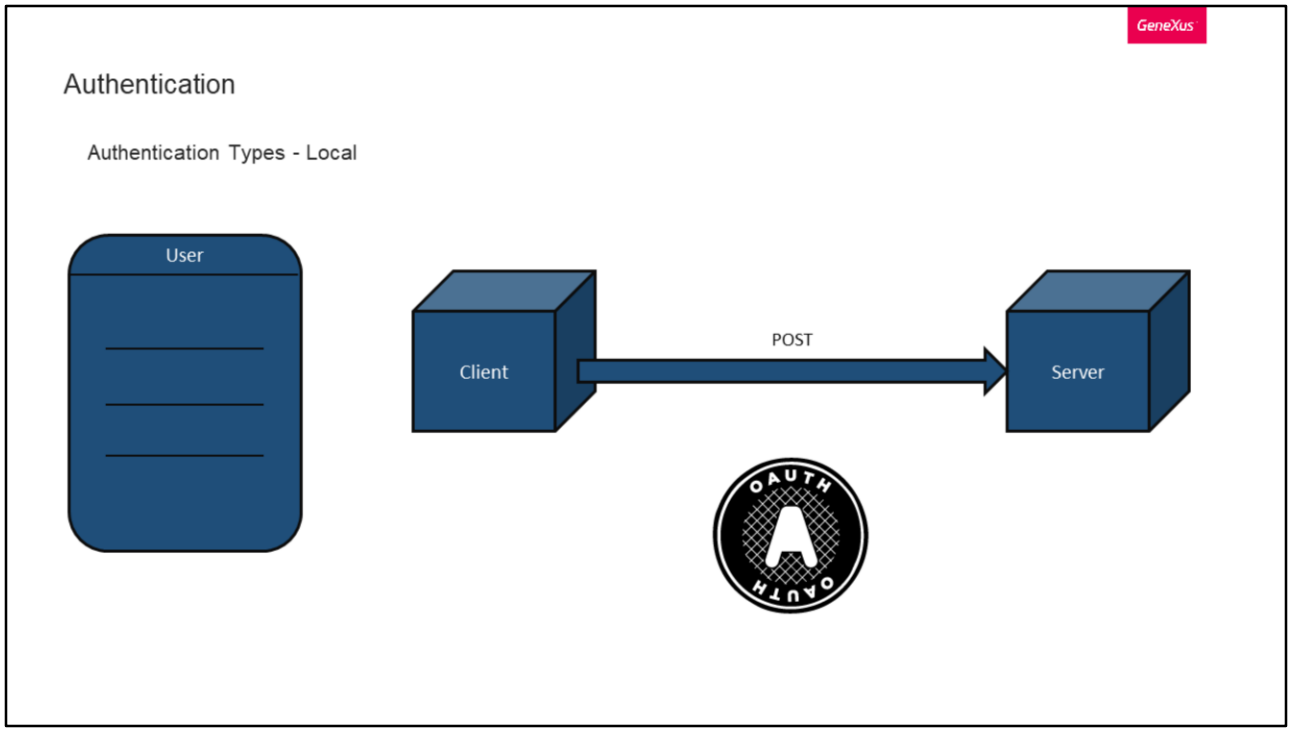


As aplicações móveis nativas instaladas nos dispositivos consomem serviços REST do servidor Web. Estas aplicações criam uma interface de usuário que também é baseada em recursos, além de metadados.

Dada esta arquitetura, é importante ter um mecanismo de segurança para evitar que estes tipos de ações sejam realizadas por usuários não autorizados.
A maneira de fazer isso, é claro, é usando o GAM.

Assim como nas aplicações web, por meio da configuração de Ativar a propriedade de segurança integrada, estamos adicionando segurança automaticamente à aplicação móvel. Esta segurança pode ser tanto por Autenticação quanto por Autorização.

Vejamos isto com mais profundidade.



Assim como nas aplicações web, nas aplicações móveis nativas, um cenário possível é que todos os objetos móveis nativos sejam privados.
Configurando nossa aplicação com o nível de segurança integrado em Autenticação, conseguiremos que todos os objetos exijam que o usuário se encontre com uma sessão ativa.

Vejamos agora como se comportam e funcionam os tipos de autenticação em aplicações móveis.
Vamos começar com o tipo Local.

Este tipo de autenticação funciona da mesma forma que as aplicações web, onde as credenciais dos usuários são armazenadas na tabela “User" do GAM.

Como já mencionamos, a senha não é armazenada. No seu lugar, é armazenado um hash dela utilizando uma chave única para cada usuário e o algoritmo SHA-512.

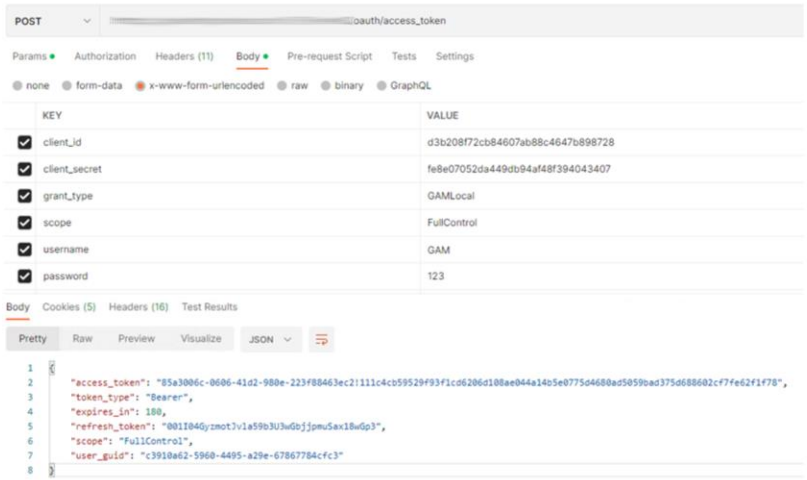
Em seguida, simplesmente é feita uma requisição POST a partir do cliente para o servidor, onde neste caso, o cliente corresponde ao dispositivo móvel.

Por trás, no que é baixo nível, este e todos os métodos de autenticação com GAM utilizam o protocolo OAuth 2.0.

GeneXus

Authentication

Authentication Types - Local



Aqui podemos ver um exemplo do que seria a requisição POST que mencionamos.

A requisição é realizada para a URL de nossa aplicação, seguida de /oauth/access_token.
Como parâmetros, são incluídos os típicos client_id e client_secret, o grant_type e scope, e finalmente, o nome de usuário e senha.

Se a requisição estiver correta, deveríamos receber um JSON como o seguinte, com todos os dados do acesso e identificador do usuário.

Isto pode ser útil para testar seus próprios tipos de autenticação, onde o teste pode ser realizado com qualquer ferramenta de testes de APIs.

GeneXus

Login

How to

```
Event 'BtnLogin'
    Composite
        GeneXus.Common.UI.Progress.ShowWithTitle("Connecting...")
        GeneXus.SD.Actions.Login(&UserName, &Password)
        GeneXus.Common.UI.Progress.Hide()
        Return
    EndComposite
EndEvent
```

&LoginExternalAdditionalParameters

Em temas anteriores, revisamos o mecanismo de Login para as aplicações web. Vejamos agora como isto funciona nas aplicações móveis.
Para ver isto, vamos nos concentrar no objeto GAMSDLogin, que está incluído nas implementações de referência que traz o GAM.

Neste caso, o início de sessão local é realizado através do External Object GeneXus.SD.Actions e seu método de início de sessão chamado Login.
Por trás, uma vez iniciada a sessão, a informação é armazenada no dispositivo, mas de forma criptografada, salvando um token de segurança utilizando o KeyStore (no caso de Android) ou Keychain (no caso de iOS).

Este método de Login pertencente a Actions está sobrecarregado, então permite incluir o parâmetro adicional &LoginExternalAdditionalParameters como vimos nos casos de Login externos há pouco.
Vejamos isto com outro exemplo e detalhe.

Esse parâmetro que mencionamos deve corresponder ao objeto chamado da mesma maneira, o qual se parece com isso.

Um objeto de utilização desta sobrecarga no método de Login SD pode ser o seguinte. O objeto LoginExternalAdditionalParameters permite estabelecer o GUID do Repositório ao qual queremos nos conectar. Isto é útil quando é uma aplicação MultiTenant.

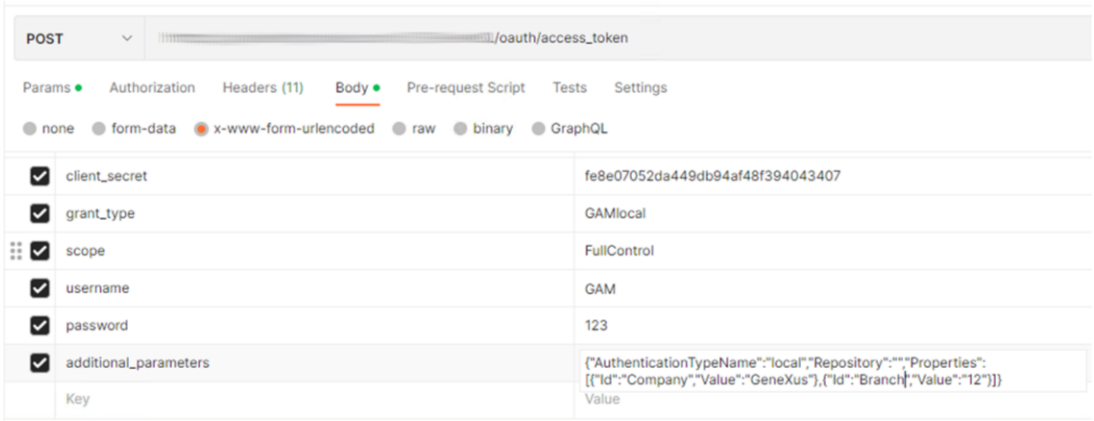
Quando damos um valor para a propriedade Repository do parâmetro &LoginExternalAdditionalParameters, pode ser estabelecida a conexão a esse Tenant (em particular para esse cliente), representada por um Repositório e seu GUID.

Além desta opção que utilizamos, temos outras que podem ser o Nome do tipo de autenticação, e uma lista de propriedades dinâmicas Id e Valor que poderia necessitar a aplicação.

GeneXus

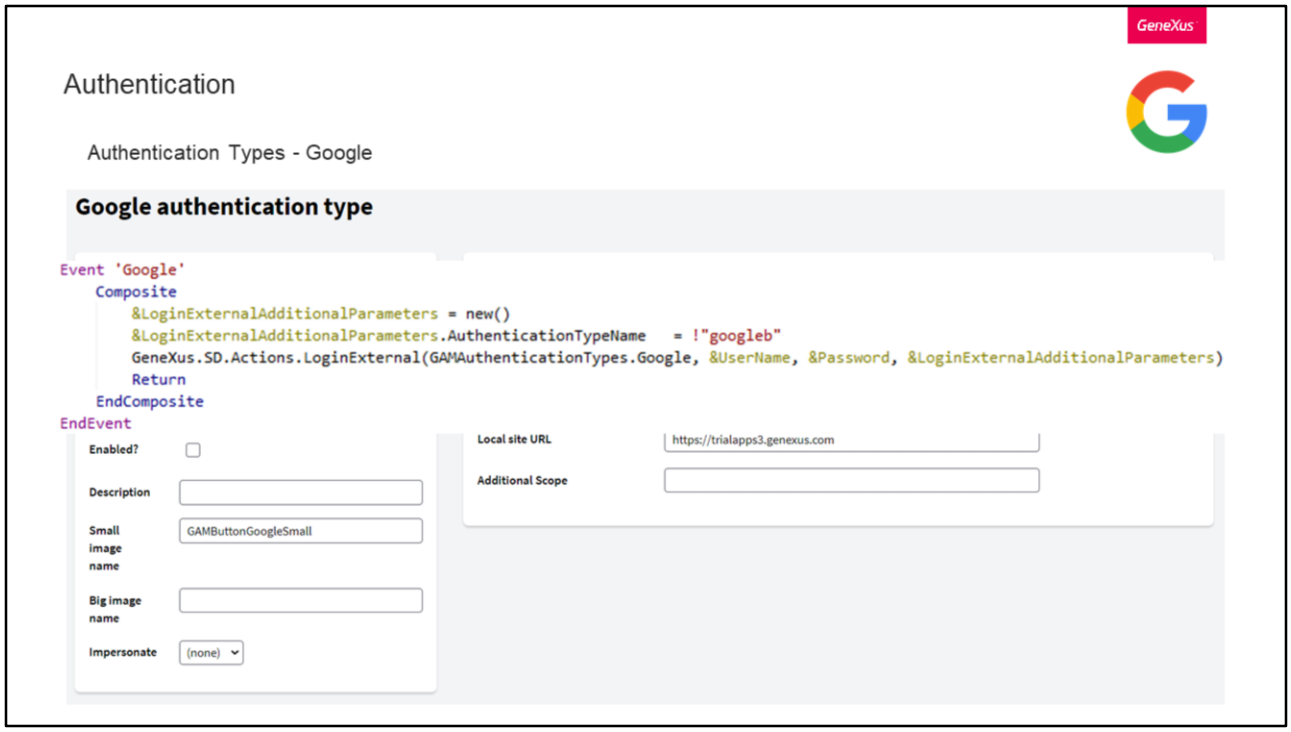
Login

How to



Voltando aos testes manuais, podemos replicar a requisição de login que vimos antes, mas desta vez adicionando estes parâmetros adicionais que comentávamos.
Para fazer isto, adicionamos à requisição o parâmetro additional_parameters, e nele, incluímos os dados que queremos adicionar à requisição.

Neste caso, vemos que é incluído o nome do tipo de autenticação que queremos utilizar, e duas das propriedades que mencionávamos anteriormente, como (Id, Valor), onde neste caso é o nome de uma empresa e sua filial.



Como já vimos no tema de Autenticação, podemos utilizar o Google como provedor de identidades para o início de sessão.
Vejamos uma configuração disto.

No site do Google devemos criar uma aplicação de cliente e obter a informação e segredo do cliente. A URL deste site é a seguinte.

Primeiro vamos para a seção API e Serviços e clicamos na seção Credenciais, selecionando "ID de cliente OAuth“.
Em seguida, selecionamos “Aplicação Web” no Tipo de aplicação.
Uma vez feito isto, devemos alterar as URI de redirecionamento. Lá podemos especificar a URI completa de nossa aplicação, incluindo /oauth/gam/signin, como é visto na imagem. Este último é importante e se aplica independentemente de nossa aplicação ser Java ou .NET.
Sempre deve ser especificada a URL completa da aplicação, incluindo o diretório virtual, seguido de /oauth/gam/signin.
Depois de confirmar isto, obteremos nossa informação do cliente e terminamos deste lado.

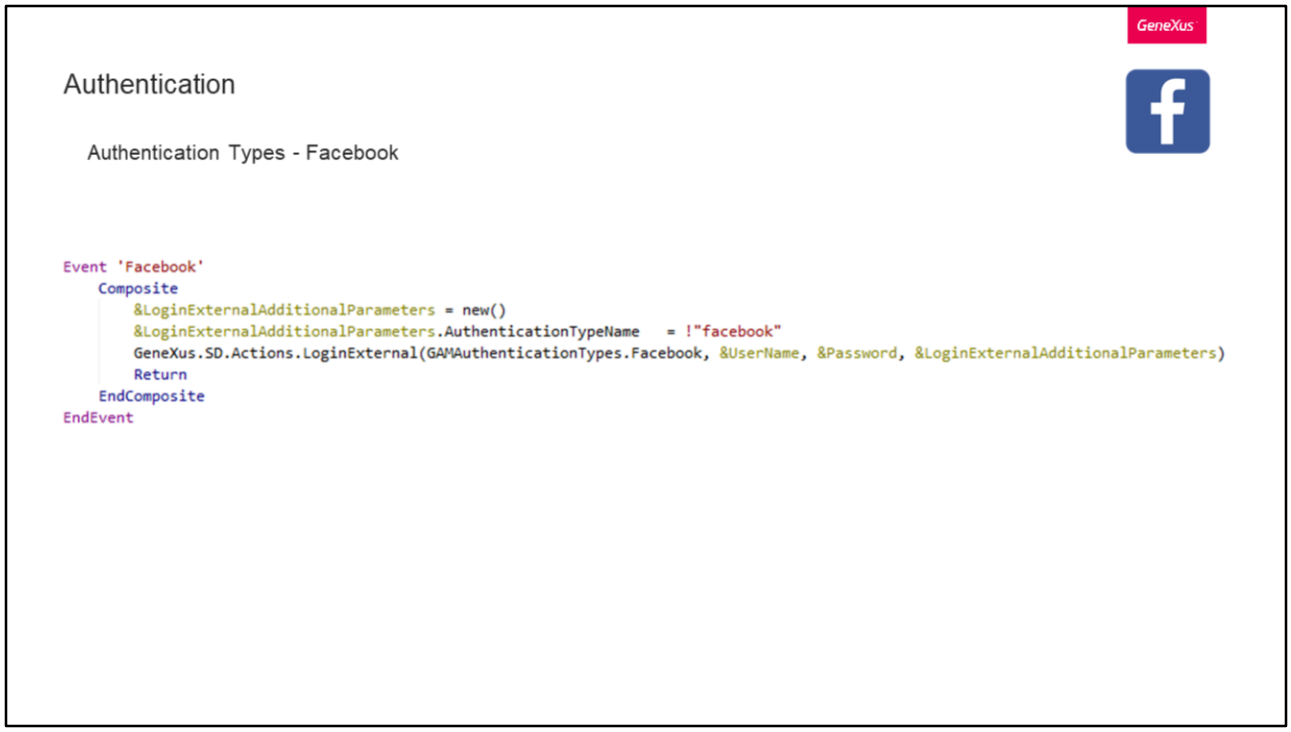
Agora vamos configurar nossa aplicação no backend do GAM.
Para isso criamos um novo tipo de autenticação do Google, e completamos a informação com a fornecida pelo Google.

Algo a destacar nesta etapa é que no campo Local site URL só é necessário inserir o domínio do servidor que executa a aplicação. Não é necessário inserir a URL completa do site, mas se o fizer, não deve ser incluído o "/servlet" em Java.

Neste momento já temos configurados o lado do Google e do Backend. Mas como estamos em uma aplicação móvel, precisamos de uma lógica diferente no Login de nossa aplicação, então devemos adicionar um evento em que nos autentique com Google.
Vemos que esta lógica agora utiliza o método LoginExternal do External Object Actions, e como primeiro parâmetro é indicado que é um tipo de autenticação do Google através do domínio GAMAuthenticationTypes.
Então o nome de usuário e senha serão ignorados e, por último, temos LoginExternalAdditionalParameters, com o qual podemos indicar informação adicional ao login que nos seja útil. Neste caso, definimos o nome do tipo de autenticação.
Isto é útil no caso em que tenhamos mais de um tipo de autenticação do Google no repositório e com este nome poderemos selecionar o que queremos. Caso contrário, podemos fazer a chamada de login sem utilizar este parâmetro.

Mais adiante veremos com mais detalhes este atributo.

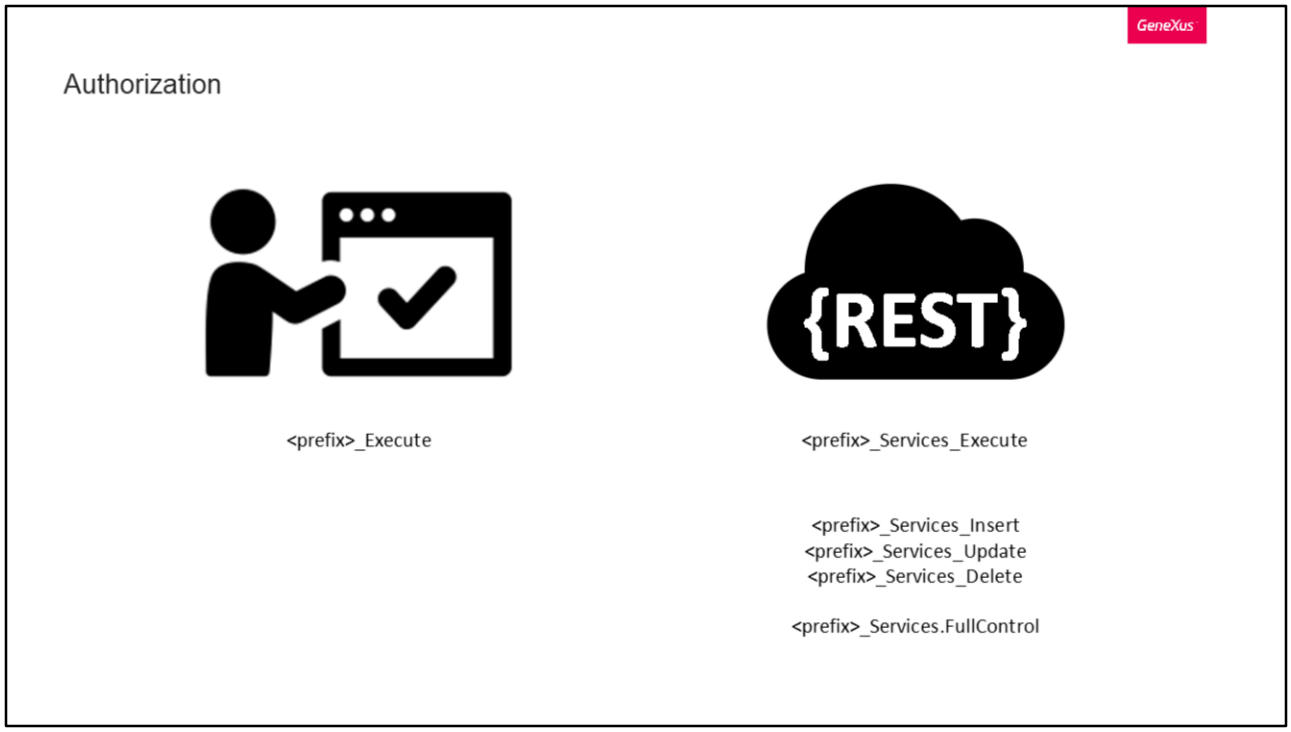
E com isto finalizamos o processo de autenticação com Google.



Vejamos agora o caso do Facebook.

No curso introdutório mostramos como fazer um tipo de autenticação com Facebook, então não vamos repetir isto. Você pode consultar como fazer isto no referido curso ou na Wiki de GeneXus.
Assim como no caso do Google, uma vez que tenhamos nosso cliente e o tipo de autenticação no backend do GAM criados e configurados, precisamos de uma lógica personalizada na aplicação.

Neste caso, novamente adicionamos um evento novo com o seguinte.
A explicação é a mesma que a do Google, com a diferença que agora no primeiro parâmetro do LoginExternal indicamos que o tipo é Facebook.



Autorização.
No caso dos painéis SD, o GAM verifica se o usuário tem permissão para executá-los. Para isso verifica se o usuário possui a permissão <prefix>_Execute, onde prefix é o Prefixo de Permissão definido para o objeto.
Caso seja detectado um erro de permissão, ele será exibido em tela ou será redirecionado para o objeto especificado como não autorizado para a propriedade SD.

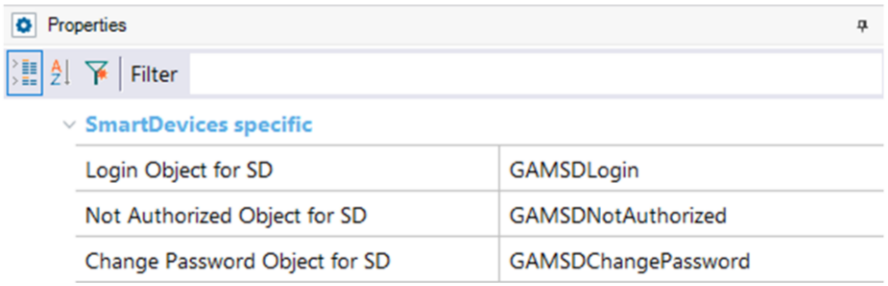
Nos painéis e WorkWith SD, a lógica de negócio é resolvida utilizando Web Services REST.
Quando são gerados estes objetos, também são gerados provedores de dados expostos como serviços web REST.
Estes serviços web tornam-se privados quando você declara a segurança dos objetos mencionados.

Um detalhe a considerar é que as ações nos painéis dos WorkWithSD estão relacionadas diretamente com o Business Component associado ao WorkWith. Isso significa que se for aplicado o padrão Trabalhar com a uma Transação, ela será salva automaticamente como Business Component exposto como um serviço web REST.
Para controlar as permissões sobre as ações de inserir, atualizar e excluir, você deve declarar as permissões no próprio Business Component.

Dado que a ideia é validar se o usuário possui direitos para executar um objeto dependendo do método com o qual é emitida a chamada, o método GET requer a permissão <prefix>_Services_Execute (onde neste caso prefix é a permissão definida para o Business Component), e esta é a permissão mínima exigida.
O restante das permissões são:
<prefix>_Services_Insert
<prefix>_Services_Update
<prefix>_Services_Delete
Aqui como sempre, também temos o FullControl que abrange todas as permissões.

GeneXus

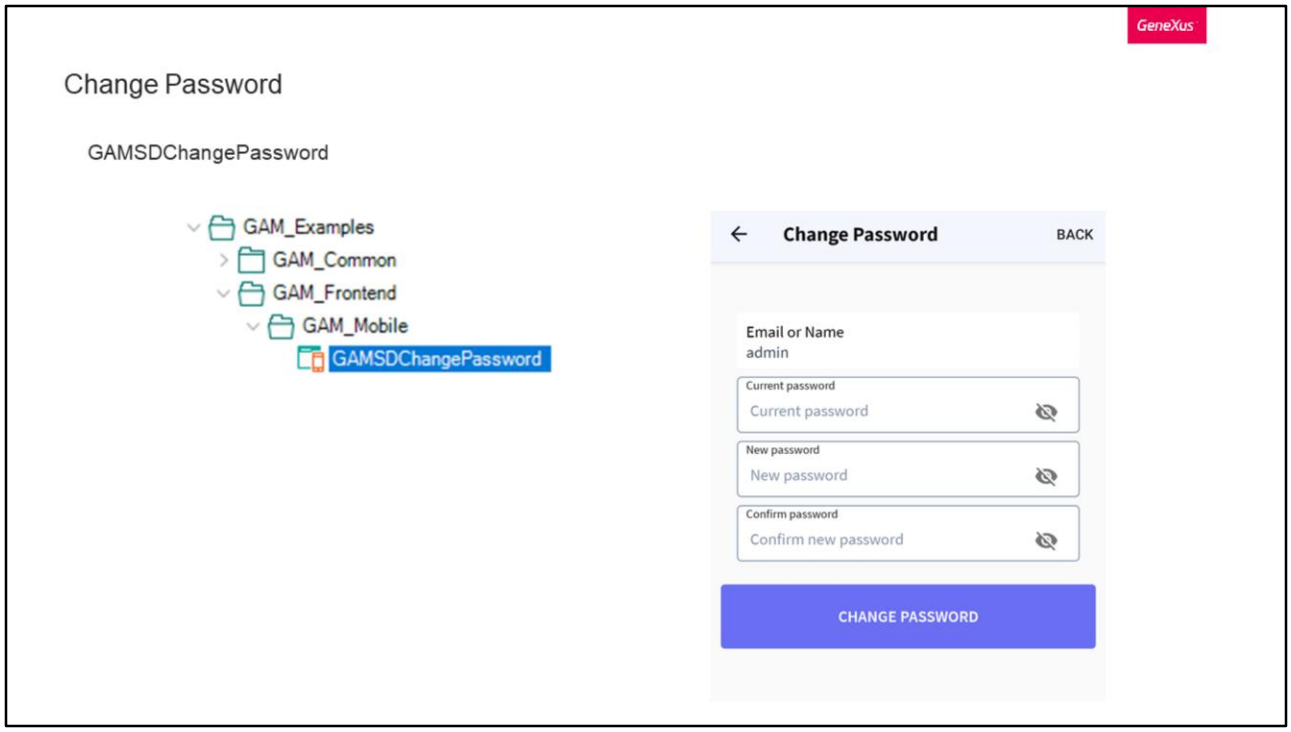
Object configuration



Anteriormente, mencionamos que o usuário poderia ser redirecionado para o objeto de não autorizado caso não tivesse permissões para acessar um recurso. Este objeto podemos configurar no nível de versão da KB, na seção SmartDevices specific.

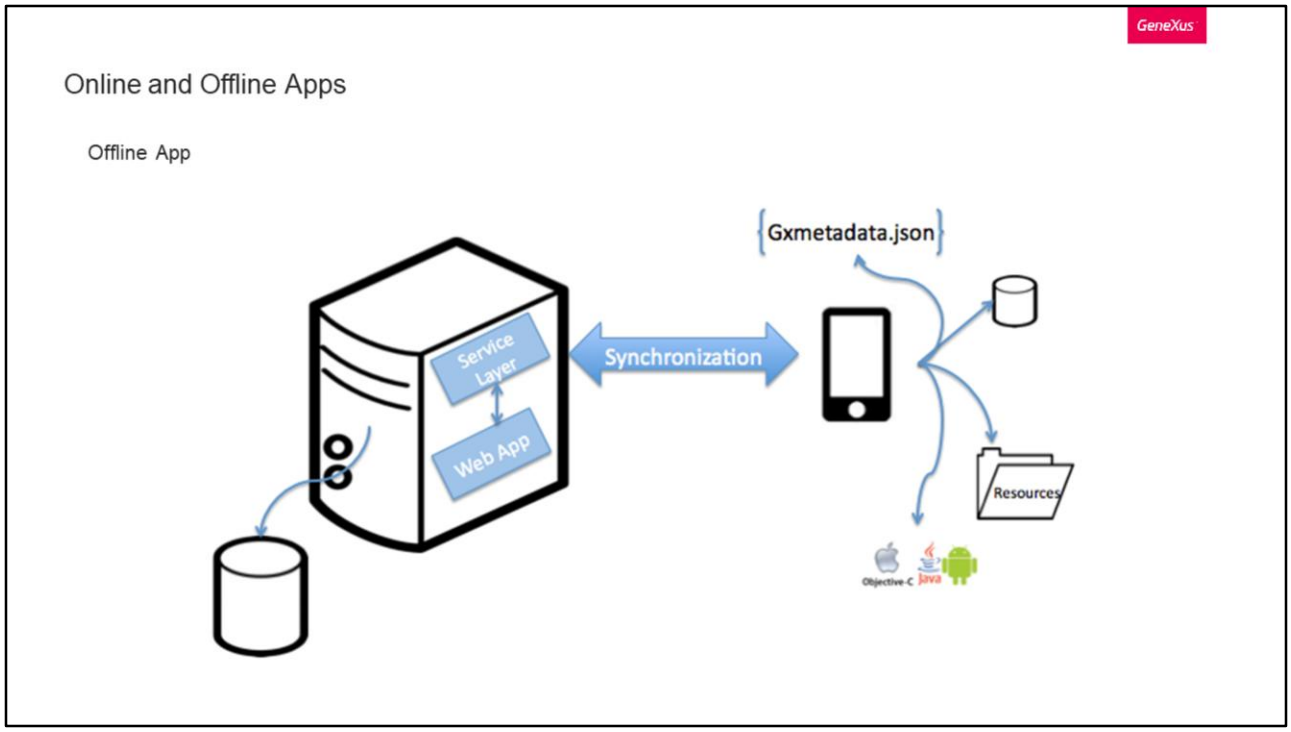
E não só podemos configurar isso, mas também é a partir dali que devemos definir quais objetos serão os associados ao login e à alteração de senha.

Vejamos uma particularidade deste último.



Da mesma forma que nas aplicações web, quando um usuário expira ou precisa alterar sua senha, ele é redirecionado automaticamente para o objeto de alteração de senha.
Em particular para Mobile, temos o objeto GAMSDChangePassword, dentro dos objetos de exemplo já incorporados pelo GAM, e é possível ser utilizado para este efeito.

Como dissemos antes, se ocorrerem as condições em que o usuário deve alterar sua senha, ele não poderá fazer mais nada enquanto não ocorra a alteração, pois as aplicações sempre o redirecionarão para o objeto que foi configurado para alterar a senha.



Com GeneXus temos a possibilidade de ter aplicações tanto Online como Offline. Vejamos as diferenças que elas têm, junto com seus objetos e métodos.

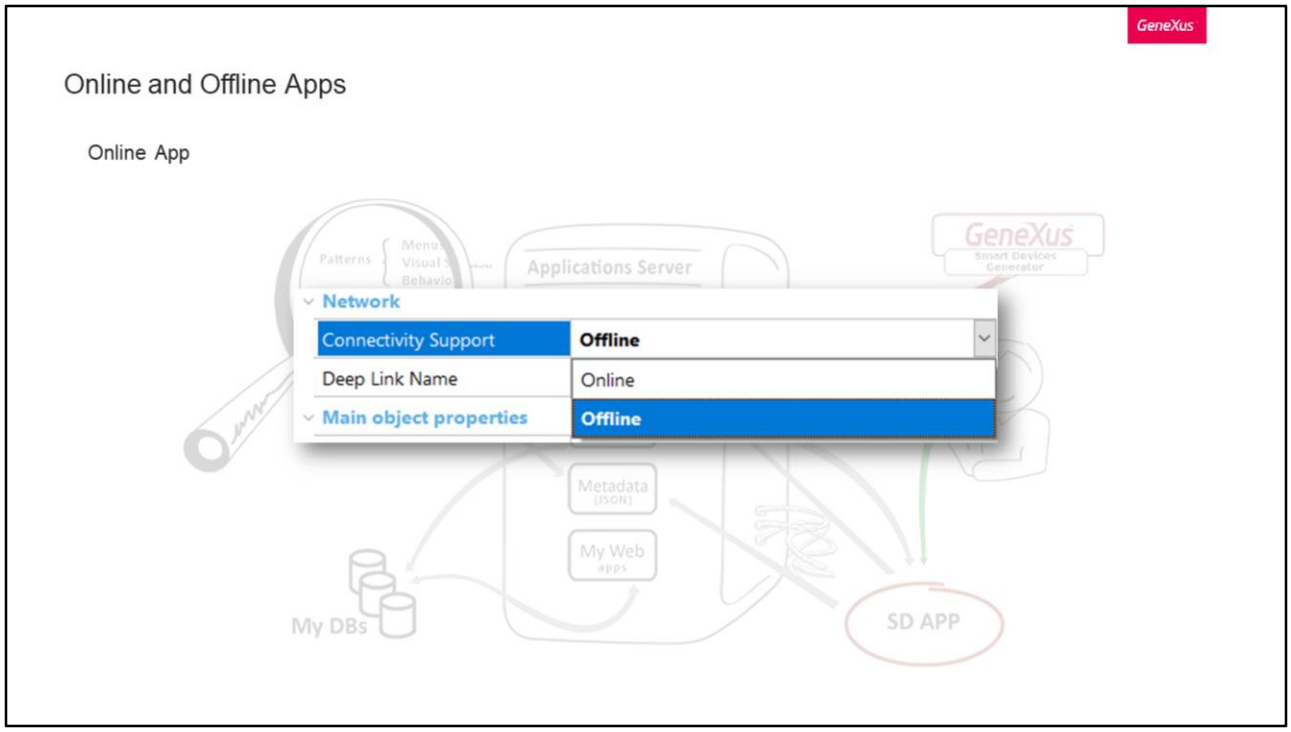
Vamos começar com as Offline.
A arquitetura destas aplicações móveis deve considerar duas situações em que estas aplicações devem funcionar: quando a aplicação está desconectada e quando tem uma conexão.
No primeiro caso, a aplicação deve ser capaz de processar dados e interagir com uma base de dados sem conexão com a rede.
No segundo caso, a aplicação pode sincronizar dados com o servidor chamando serviços REST.

Sem importar se está conectado ou não, o processamento da aplicação é realizado no dispositivo, atualizando sua base de dados local. Uma vez estabelecida uma conexão, a aplicação executará a sincronização de suas alterações que foram feitas enquanto estava desconectado.

Uma aplicação móvel dessas podemos dividir em dois componentes: um componente local e um componente de servidor.
O servidor pode ter um backend para a aplicação, a base de dados e uma camada de serviços para a sincronização de dados com os dispositivos.
Nos dispositivos, as aplicações também possuem uma base de dados com um

subconjunto dos dados do servidor e toda a lógica que deve ser executada localmente. Por sua vez, a aplicação também possui todos os metadados necessários para o desenho da interface de usuário e os eventos do usuário.

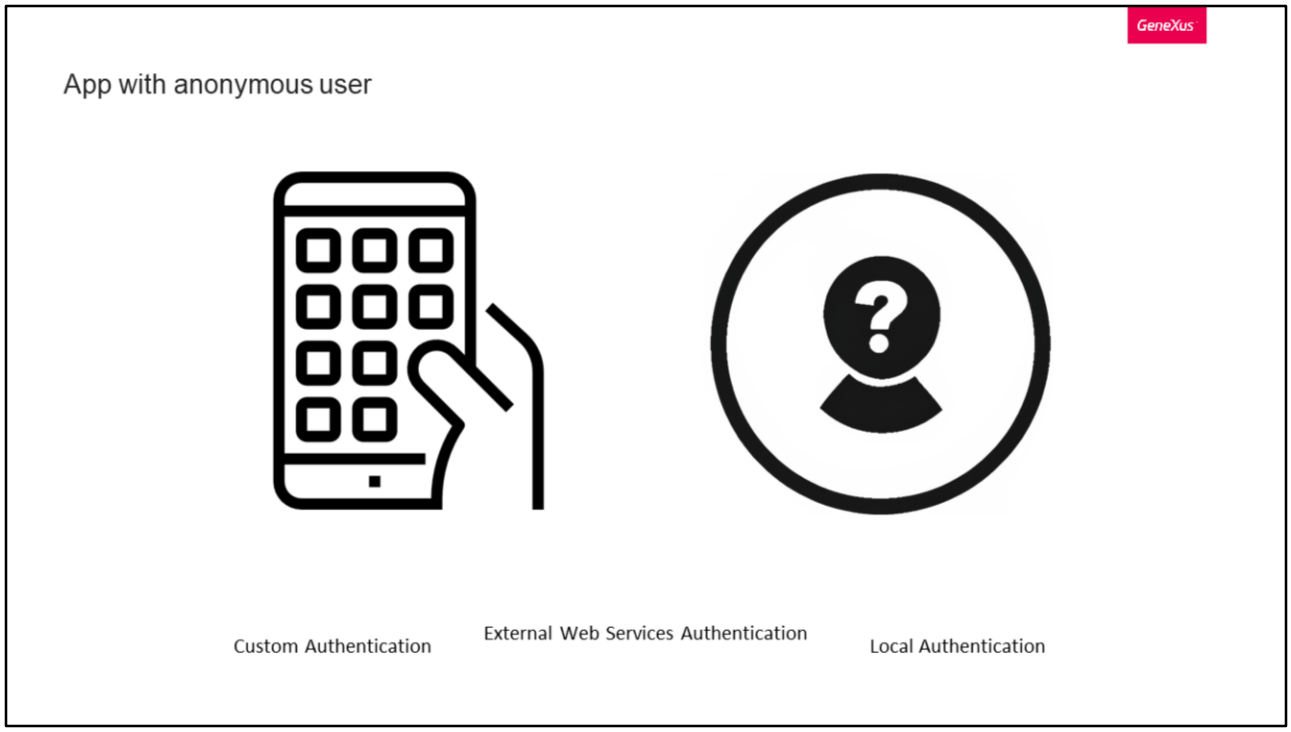
Ambos os componentes se comunicam por meio de serviços REST para realizar a sincronização necessária para completar a base de dados local e enviar as alterações realizadas no dispositivo para o servidor.



Vamos ver agora as aplicações com conexão, e para isso voltamos à ilustração que vimos no início.

Os dados requeridos pela aplicação que residem no dispositivo serão obtidos consumindo serviços REST, que acessarão a base de dados do usuário e retornarão a informação solicitada para a aplicação.
Para determinar se uma aplicação será Online ou Offline, você deve configurar a propriedade Connectivity Support no objeto Main, com o valor desejado. Como vemos, esta propriedade se encontra abaixo do item Network.

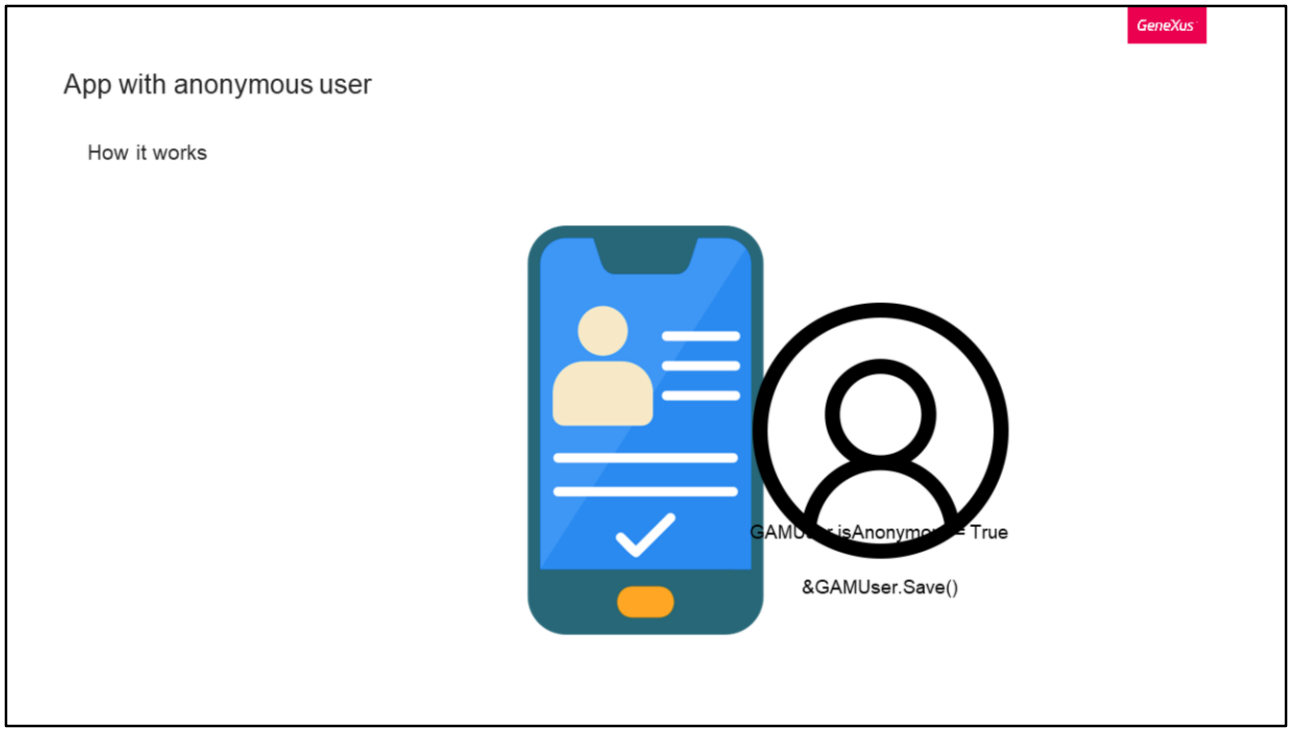
Se a aplicação é Offline, não é realizada uma verificação de permissões automáticas no dispositivo, isto só acontece nas aplicações Online, pois é acessado um serviço Rest no Servidor.



Aplicação com usuário anônimo, o que significa isto?
Este é um conceito que temos no desenvolvimento de aplicações móveis com segurança do GAM, que permite oferecer aos usuários a possibilidade de utilizar a aplicação com as mesmas ou semelhantes funcionalidades que possuem os usuários registrados.
Por trás, o que o GAM está fazendo é registrar automaticamente o usuário anônimo para cada dispositivo.

Para GeneXus 15 U10 ou versões anteriores, isto ocorre somente com os tipos de autenticação Custom, External Web Services e Local. A partir de GeneXus 15 U11 em diante, é válido para qualquer tipo de autenticação.

Vejamos como funciona.

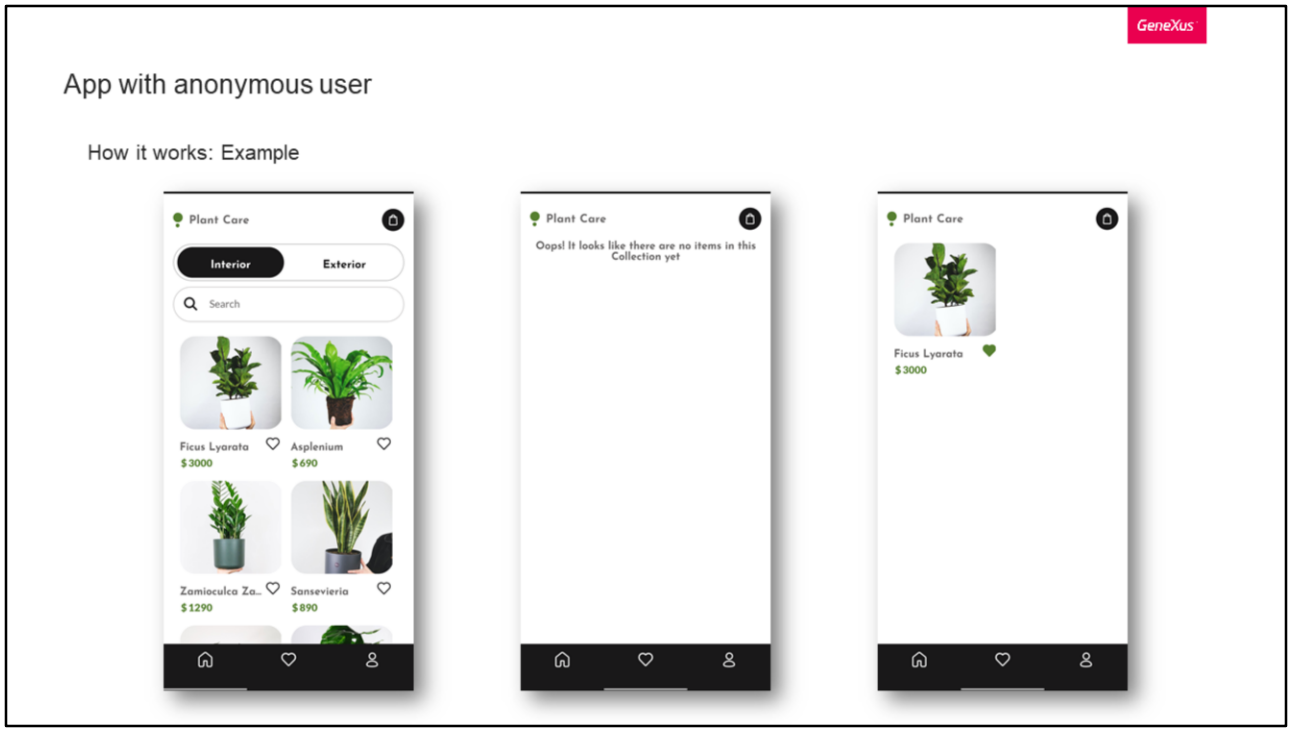


Cada dispositivo possui um identificador para ele, portanto, quando um usuário começa a utilizar a aplicação, automaticamente é criado um "usuário" do GAM usando esse identificador e solicita um token.

Este usuário criado possui as seguintes características:

- É visto pela aplicação como qualquer outro usuário, mas possui a propriedade GAMUser.isAnonymous = True
- Não são conhecidos, solicitados ou armazenados dados pessoais da pessoa que opera o dispositivo, portanto, permanece completamente anônimo
- Sua sessão nunca expira, sem importar a expiração do token de OAuth que tenha sido definido para as sessões

Se o usuário decide registrar-se na aplicação, ao utilizar o método &GAMUser.Save() será criado o novo usuário com seus dados cadastrais, e a ele será atribuído o mesmo identificador que tinha o usuário cadastrado automaticamente.
Isto permite que qualquer informação relacionada ao usuário registrado automaticamente que tenha gravado a aplicação, permaneça associada ao novo usuário registrado, sem ter que realizar alterações nas relações usuário-dados que tenham sido gravados anteriormente.



Vejamos um exemplo para entender isso em profundidade.

Suponhamos que temos uma aplicação de venda de plantas.
A referida aplicação tem a possibilidade de se registrar e fazer login com uma conta. Além disso, podemos marcar como favoritas as plantas que quisermos para depois, por exemplo, podermos comprá-las.

Agora, o ideal seria que, se ainda não iniciei sessão na aplicação e seleciono alguma planta como favorita, na hora do cadastro aquelas favoritas continuem sendo.

Isto é o que nos permite a funcionalidade de usuário anônimo.
Se a ativamos, a pessoa poderá marcar todos os favoritos que quiser sem se preocupar se já está logado na aplicação ou não, pois caso não esteja, ao se registrar, os favoritos serão passados automaticamente para sua conta.

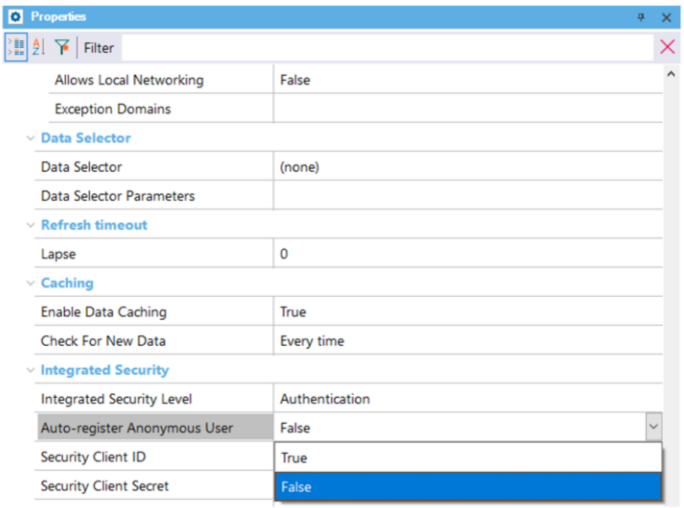
O mesmo poderia ser aplicado ao carrinho de compras, por exemplo, mantendo para a pessoa os itens selecionados no carrinho quando se registre para fazer o check-out.

Um detalhe a destacar é que se o usuário já estava registrado, os favoritos não são passados, pois o que ele faz é um Login e não um Usuário novo. Isto só funciona com o registro.

GeneXus

App with anonymous user

How to use it



Esta propriedade está disponível para os objetos Mobile que são Main (por exemplo o Panel), quando a propriedade de ativar a segurança integrada é definida como verdadeira.
O valor desta propriedade determina o comportamento quando um usuário anônimo tenta executar o primeiro objeto da aplicação que possui a propriedade Nível de segurança integrada em Autenticação ou Autorização.

Como vemos na imagem, os possíveis valores para a propriedade de autorregistro são Verdadeiro ou Falso.
Se selecionarmos Falso e tivermos definido o nível de segurança integrado como Autenticação ou Autorização, ao usuário anônimo será exibida uma caixa de diálogo indicando que ele precisa fazer login ou registrar-se.
Se selecionarmos Verdadeiro, em vez de solicitar login ou registro, a aplicação criará automaticamente um usuário com base na informação do dispositivo móvel do usuário.

GeneXus

App with anonymous user

How to identify auto-registered users

```
If &GAMSession.User.IsAutoRegisteredUser
    //Anonymous User
else
    //Registered User
EndIf
```

```
GAMUser.isAnonymous()
```

```
&GAMUser = GAMUser.Get()
&GAMUser.IsAutoRegisteredUser
```

Caso desejemos identificar estes usuários registrados automaticamente, temos as seguintes formas.
Uma opção é através do usuário da sessão GAM, perguntando diretamente se é um usuário autorregistrado.
Outra forma similar em termos de usuário é a seguinte, onde retornará um booleano indicando se o usuário atual é anônimo ou não.
E a última opção é a seguinte, onde primeiro se obtém o usuário do GAM e depois se consulta se é autorregistrado.

GeneXus™
by Globant

training.genexus.com
wiki.genexus.com