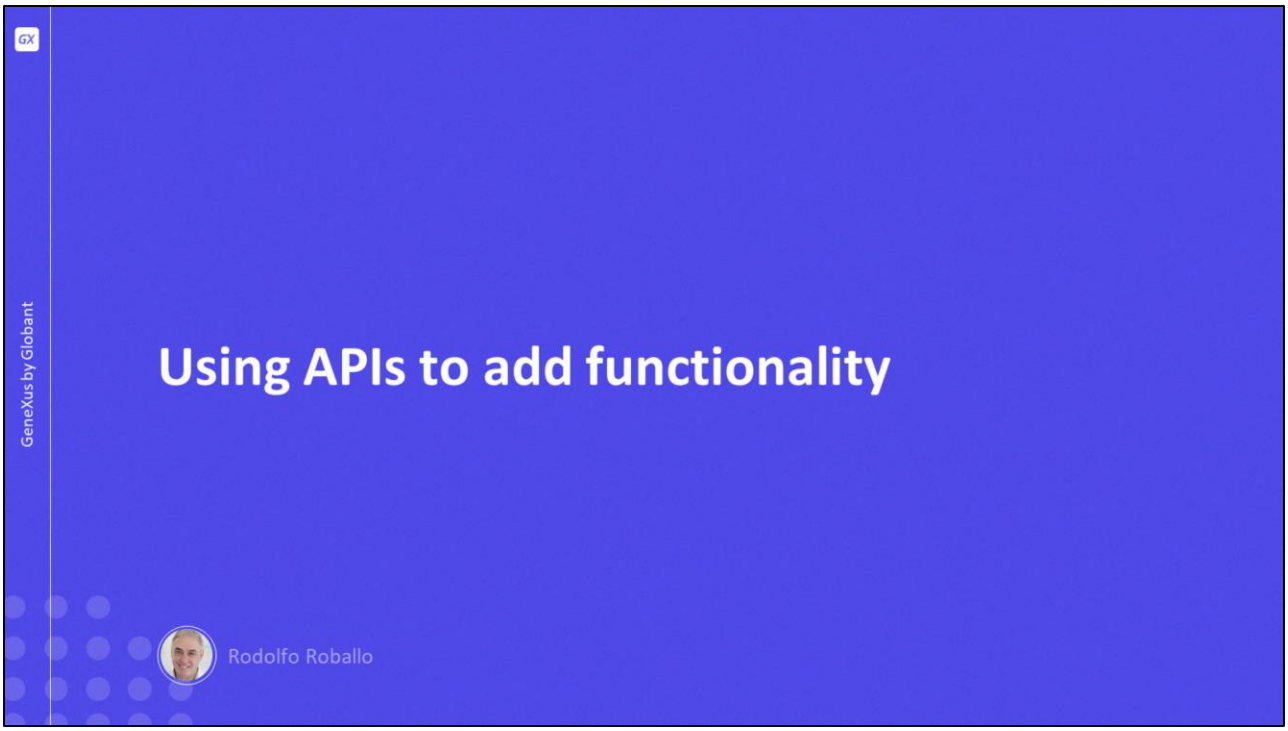
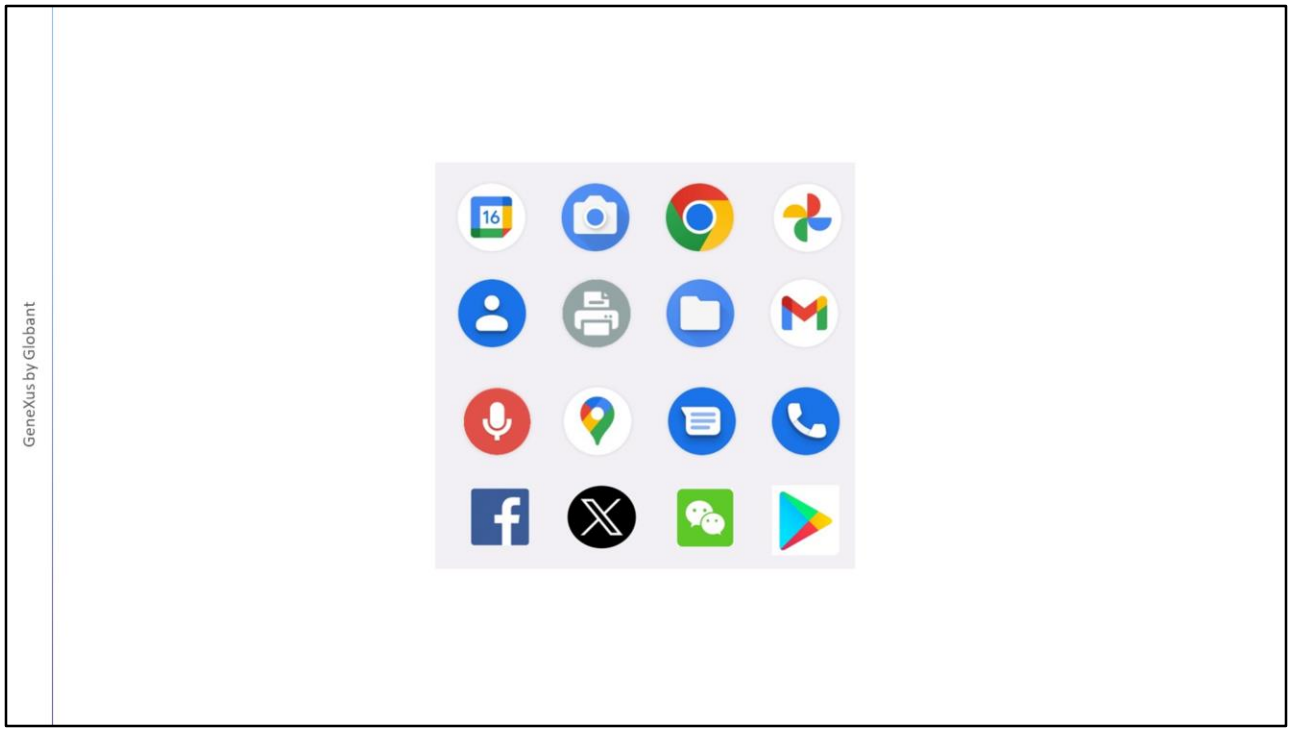


Neste vídeo veremos como adicionar à nossa aplicação, funcionalidades que nos permitem a integração com recursos do dispositivo, tanto físicos quanto lógicos, bem como facilitar a comunicação com aplicações externas.

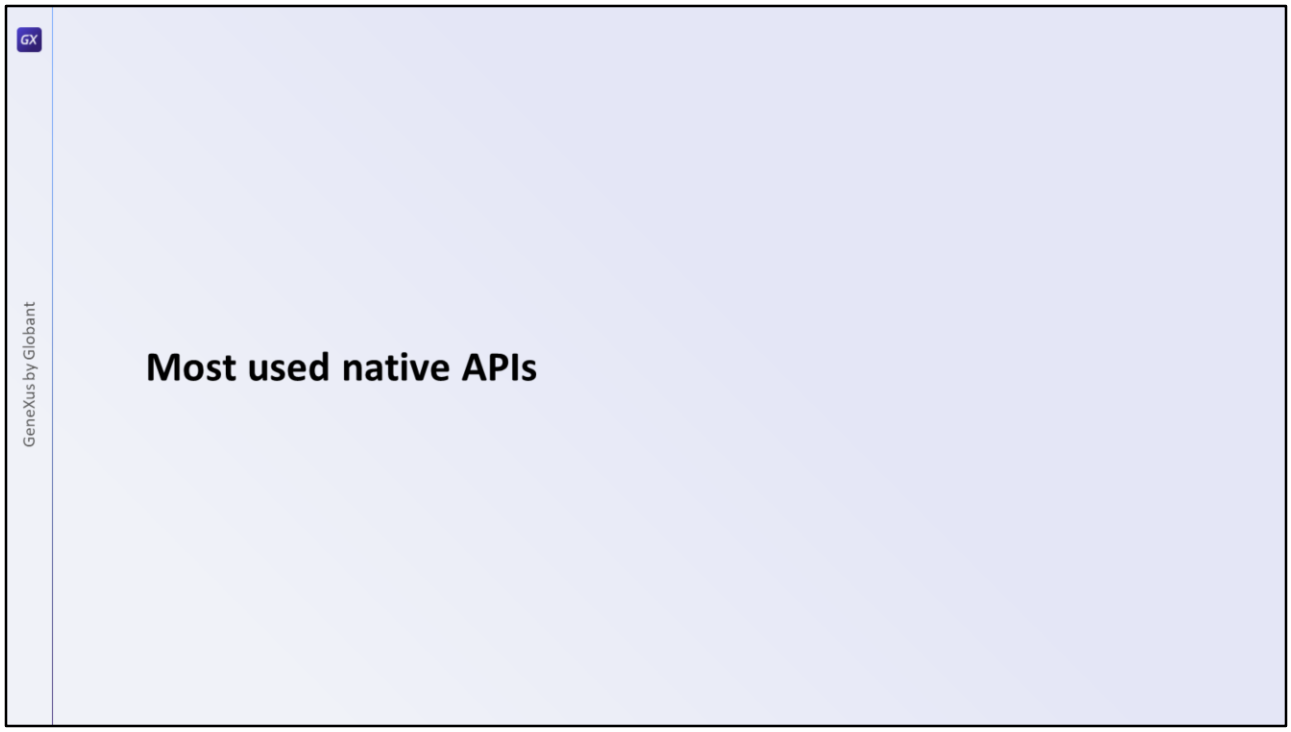


Utilizando objetos externos predefinidos em GeneXus, podemos acessar programaticamente muitas das funções que possui um dispositivo físico, como usar a câmera, acessar o gravador de áudio, fazer uma chamada ou imprimir determinado conteúdo.

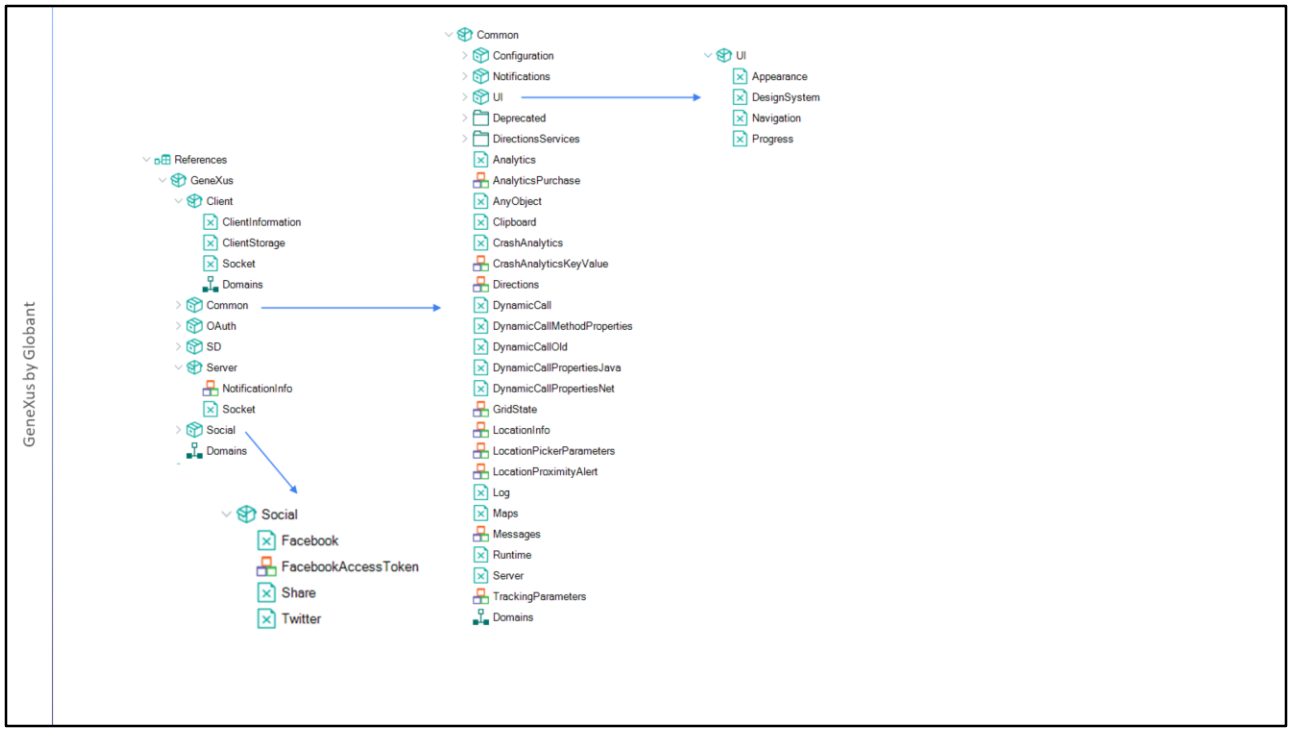
Também podemos acessar aplicações que já estão instaladas no dispositivo, como o calendário, os contatos, acesso aos arquivos, enviar um SMS, enviar um e-mail, abrir o browser ou acessar a store.

E também podemos interagir com redes sociais como Facebook ou Twitter.

A seguir veremos como acessar algumas dessas funcionalidades.



As APIs disponíveis em GeneXus podem ser de uso exclusivo em dispositivos móveis nativos, outras especiais para aplicações Web e outras que são válidas para ambas as plataformas.



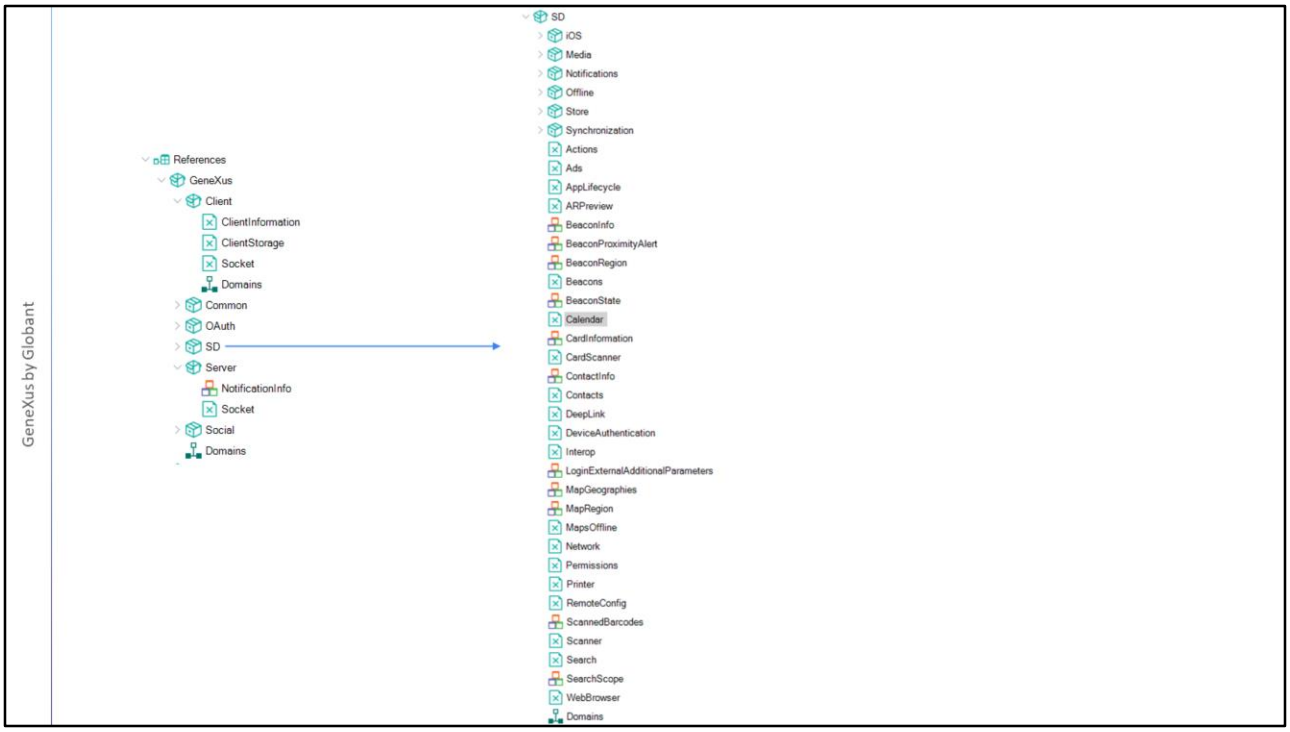
Encontramos estas APIs no KB Explorer sob o nó References, dentro do módulo GeneXus, onde também se encontram todos os SDTs, domínios e Procedimentos que estas APIs requerem. Este módulo, junto com seus submódulos são criados quando é criada a KB, e todos os seus objetos já estão compilados, razão pela qual são read-only. Isto faz com que não devam ser gerados novamente toda vez que é feito build da aplicação, o que melhora muito os tempos de compilação.

Por outro lado, qualquer pessoa pode desenvolver um módulo com objetos externos, domínios, sdts, etc. para implementar tanto funcionalidades nativas quanto para web e distribuí-lo. E poderemos importar esses módulos e utilizá-los da mesma forma que utilizamos as apis predefinidas. Desta forma, é facilitado o trabalho colaborativo na comunidade de desenvolvedores.

Se dermos uma olhada rápida, vemos que temos apis especiais para SD, outras especiais para Web e outras válidas para ambas as plataformas.

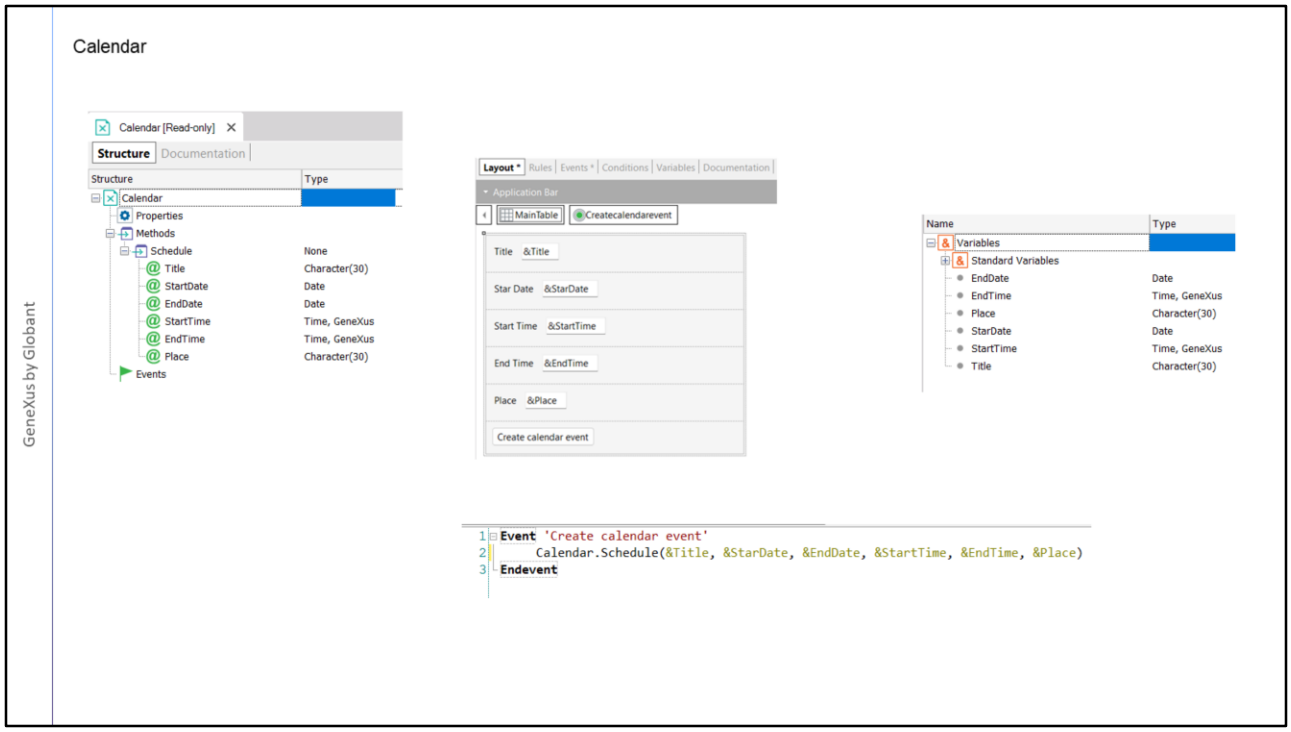
Por exemplo, dentro do submódulo Common vemos a API Maps, a API Log ou a API Analytics que podemos usar tanto em aplicações web quanto nativas; dentro do módulo UI, user interface, temos o objeto externo Navigation, que nos permitirá mostrar regiões da tela ou ocultá-las ou o objeto externo DesignSystem que permite que nossa aplicação possa configurar por código, aspectos de design.

Dentro do submódulo Social vemos apis para interoperar com Facebook ou Twitter, bem como para compartilhar informação com algum dos programas instalados no dispositivo.



Se abrirmos o grupo SD, veremos todas as APIs que podemos utilizar em um dispositivo móvel nativo.

Vamos ver como adicionar um compromisso ao calendário.



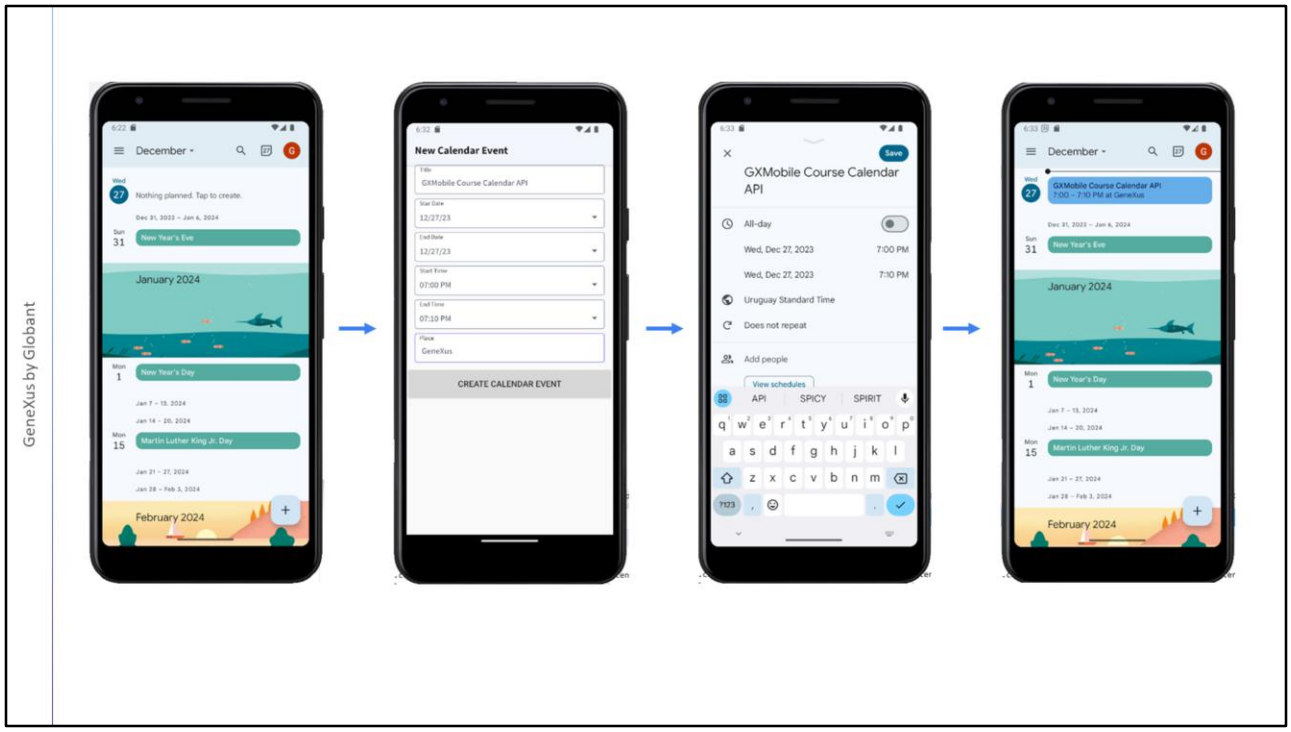
Se abrirmos a API Calendar, vemos que não possui propriedades e possui apenas um método, o método Schedule.

Se olharmos seus parâmetros, reconhecemos a informação necessária para adicionar um compromisso ao calendário, como o título, data e hora de início, data e hora de término e o local.

Uma coisa importante é que temos que passar todos os parâmetros e na ordem em que aparecem no método, se houver um parâmetro cujo dado não temos, deixamos o lugar desse parâmetro vazio, mas o escrevemos na própria invocação.

Em um panel denominado NewCalendarEvent, temos em tela as variáveis necessárias para os dados requeridos pelo método da API Calendar e um botão com o qual chamaremos o método Schedule, passando-lhe os parâmetros requeridos.

Vamos definir este panel como main, e como já temos o emulador aberto, executamos o panel.



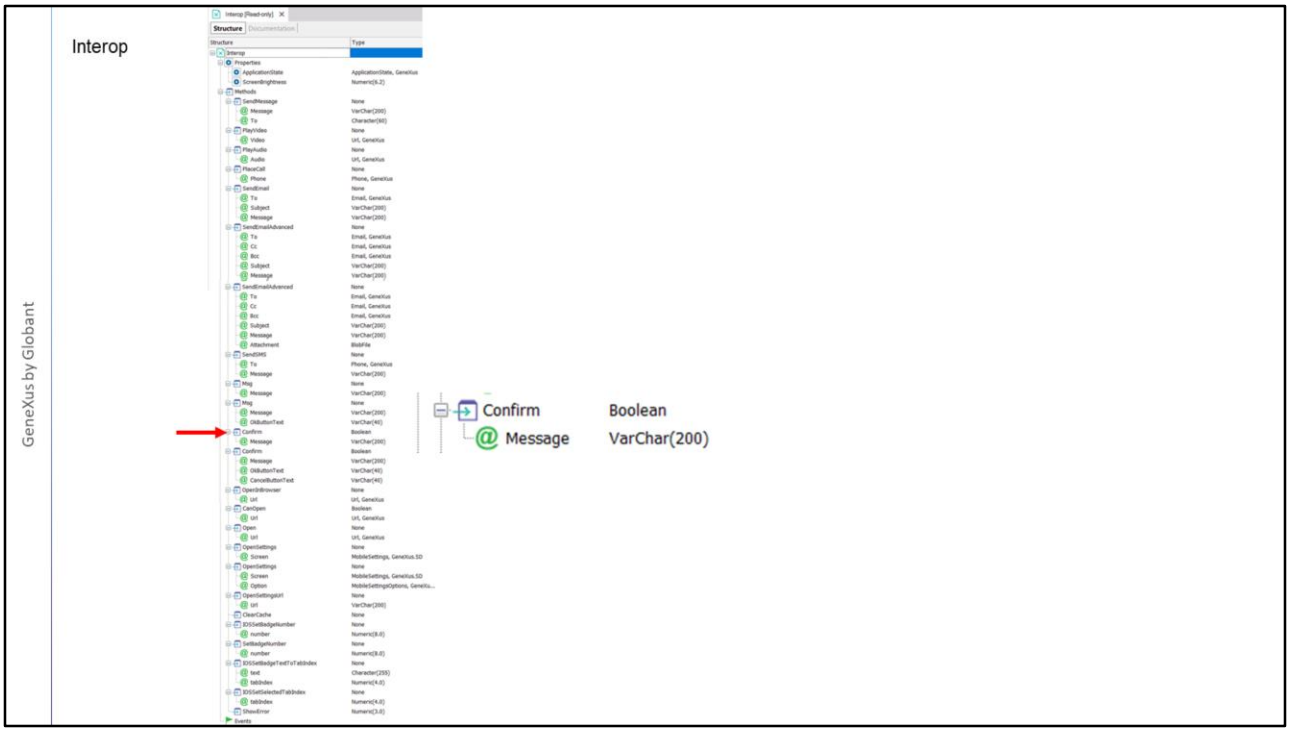
Vemos que é executado nosso panel. Antes de inserir dados, vamos à aplicação Calendário, fazemos login e vemos que nenhum evento foi adicionado para o dia de hoje.

Inserimos um evento chamado GXMobile Course API Calendar para hoje, para dentro de uma hora e que dure 10 minutos, no local colocamos GeneXus e pressionamos o botão CREATE CALENDAR EVENT.

Vemos que no dispositivo se abriu uma janela com os dados que inserimos, está tudo correto então pressionamos Save. E vemos que apareceu o evento que criamos por código a partir de nossa aplicação utilizando o Objeto Externo Calendar.

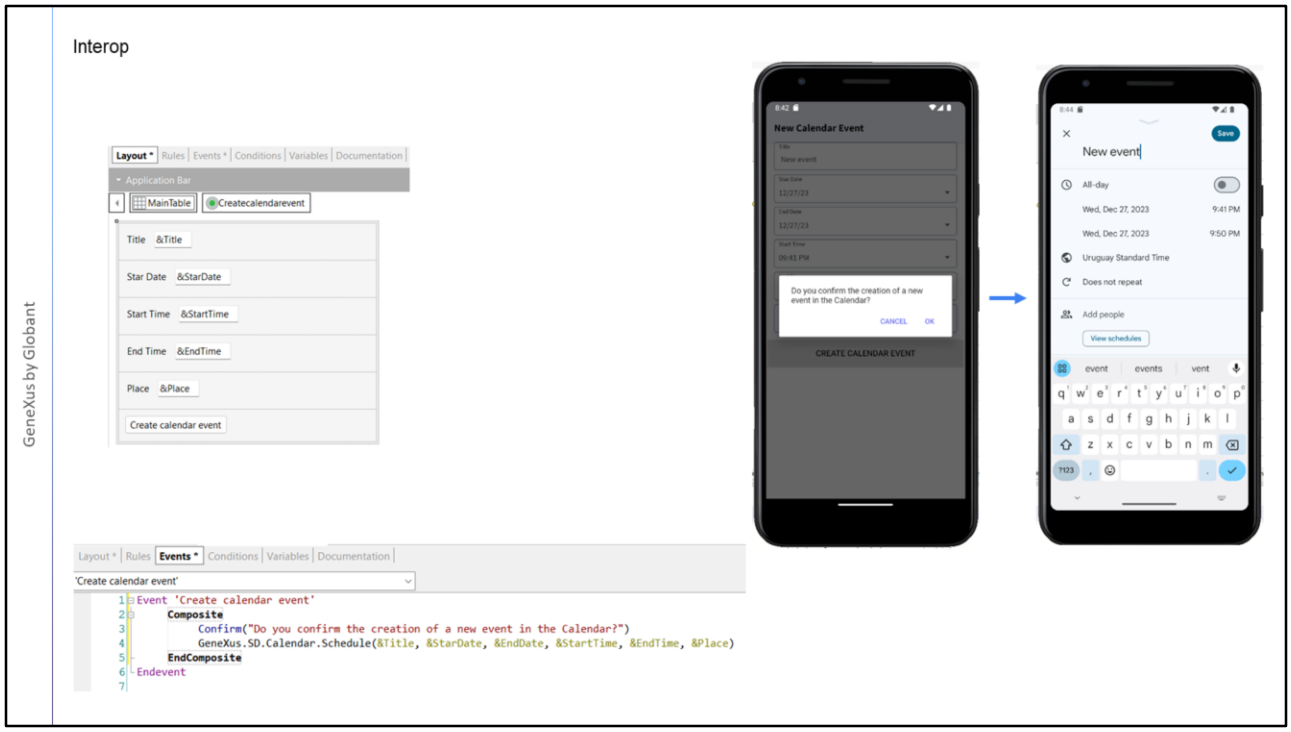
GeneXus possui alguns objetos externos que proporcionam ações comuns, como ir diretamente à tela principal, retornar à tela anterior, salvar um conteúdo e outras que proporcionam as funcionalidades mais comuns em um dispositivo móvel como enviar mensagens, confirmações, enviar e-mails ou mensagens SMS.

Vejamos algumas dessas APIs multifuncionais, a API Interop e a API Actions.



Uma API que é muito utilizada é a chamada Interop e que oferece interoperabilidade com app de mensagens, permite enviar mensagem por e-mail e SMS, bem como exibir mensagens ao usuário para informá-lo de algo ou para pedir confirmação sobre uma ação, executar vídeos ou áudios e muitas outras funções como podem ser vistas nos métodos do objeto.

Vamos utilizar o método Confirm para adicionar uma instância de confirmação antes de adicionar o compromisso de calendário que vimos no exemplo anterior.



Para isso, invocaremos o método Interop.Confirm, passando a ele como parâmetro a mensagem de confirmação que queremos que apareça. Dada a frequência da utilização do método Confirm, não precisamos escrever Interop.Confirm, podemos escrever diretamente Confirm e GeneXus o entende sem problemas.

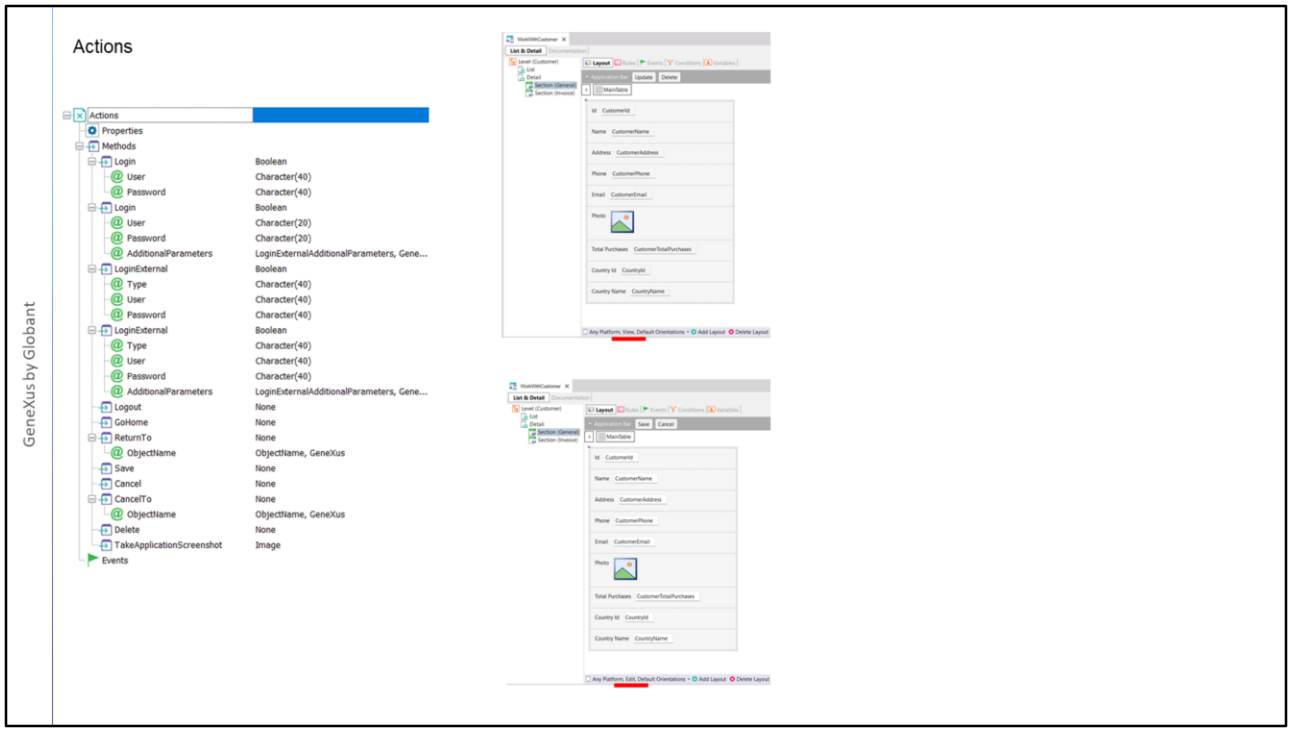
Vamos novamente ao panel NewCalendarEvent e no evento do botão podemos arrastar interop ponto e escolher o método Confirm ou como dissemos, escrever diretamente Confirm e depois entre parênteses vai o texto que queremos que seja exibido ao usuário. Observemos que embora o método Confirm retorne um valor booleano, não é necessário avaliar o valor retornado, pois GeneXus tem a inteligência de executar ou não a linha seguinte do código, dependendo se aceitamos ou cancelamos a ação.
Lembremos que o bloco Composite é opcional, mas o adicionamos porque nos fornece tratamento automático de erros.

Executamos…

Adicionamos os dados do novo evento.

Vemos que ao pressionar o botão, se abre uma janela pop-up com o texto de confirmação que escrevemos e temos a possibilidade de aceitar ou cancelar.

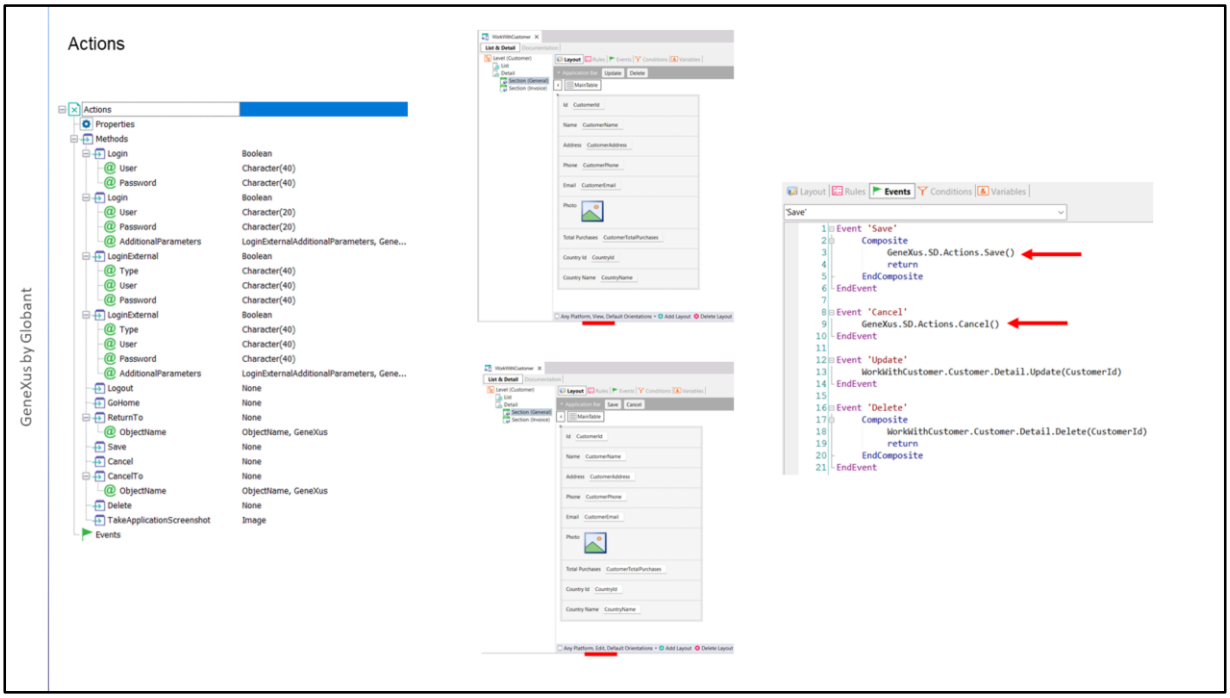
Se pressionarmos CANCEL voltamos à tela de entrada de dados do panel e se pressionarmos OK, vemos que se abre a tela de criação do novo evento.



Existe uma API que já havia aparecido anteriormente, mas que não tínhamos observado seu funcionamento, que era a API Actions, que permite, entre outras coisas, abstrair a funcionalidade de Login, Save, Cancel, métodos estes últimos, que nós já tínhamos visto nos eventos que implementava automaticamente o pattern Work with no nível do Detail.

Aqui vemos o objeto WorkWith que é criado ao aplicar o padrão WorkWith à transação Customer.

Por exemplo, vejamos o nó Detail, Section (General). Temos os botões Update e Delete para o layout do modo View e, para o layout do modo Edit, os botões Save e Cancel.



Se olharmos sua programação, vemos que o evento 'Save', que corresponde ao caso em que estamos editando a informação de um cliente e queremos gravar essa informação na base de dados, utiliza o método Save da API Actions.
O que o método Save desta API faz é encapsular a invocação ao business component correspondente, que é aquele que efetivamente realiza a inserção na base de dados, lembremos que este Business Component é um serviço REST, uma api que expôs nossa aplicação no servidor.

Poderíamos nos perguntar por que na invocação não aparece apenas Action.Save(), mas é usado todo o caminho: GeneXus. Sd. Actions. Save()
Lembremos que tanto GeneXus quanto SD são módulos e os objetos que estão dentro deles são visíveis dentro do módulo, mas podem existir outros módulos dentro da mesma kb com objetos de mesmo nome, mas que são objetos independentes. A maneira de desambiguar esta situação é utilizar todo o caminho nomeando o módulo contêiner principal e depois os submódulos até chegar ao objeto que queremos utilizar.

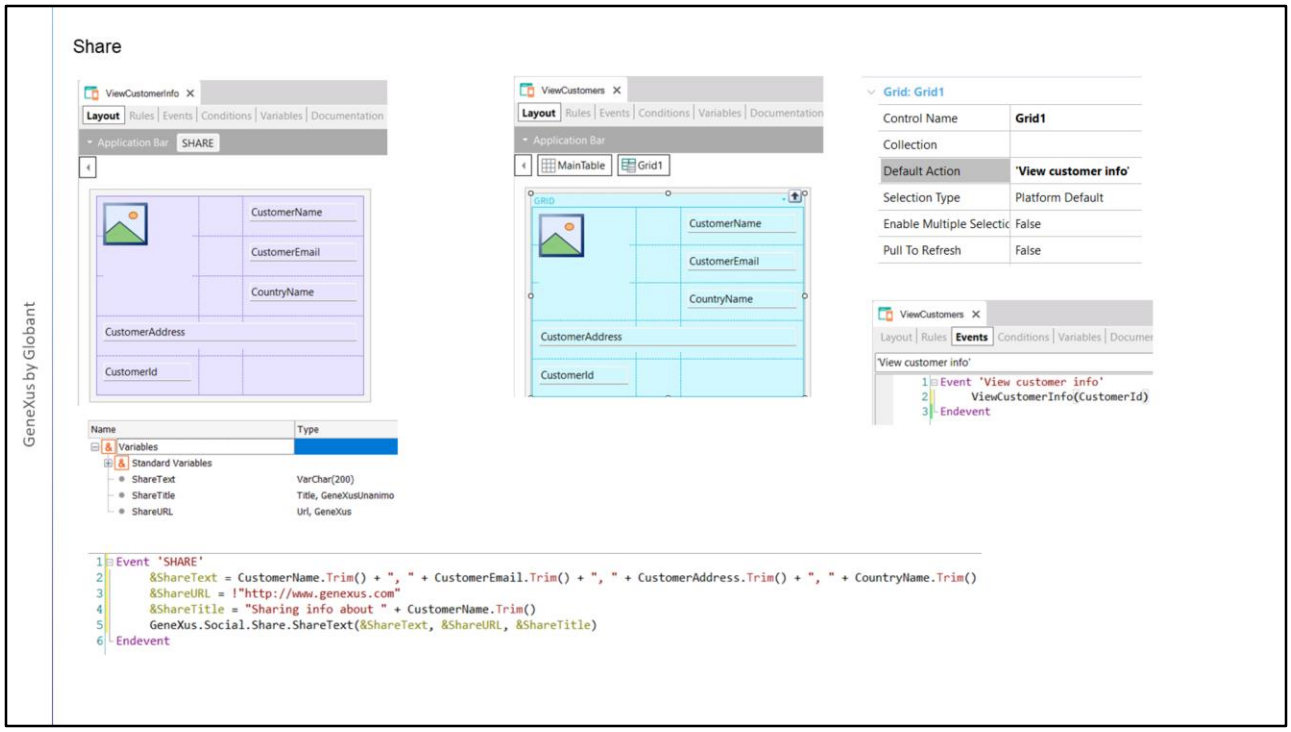
Isso depende se temos outros objetos com o mesmo nome, portanto, dar todo o caminho pode ser necessário ou não. Porém, é mais seguro incluir sempre todo o caminho, principalmente se forem incluídas APIs desenvolvidas especificamente para o projeto e se nele trabalham muitos desenvolvedores.

Voltando ao exemplo do WorkWith, observamos que o evento Cancel utiliza o método de mesmo nome da API Action. Tudo isso está programado de forma predeterminada no padrão WorkWith, portanto, se aplicarmos esse padrão a outra transação, observaremos uma programação análoga destes eventos utilizando esta API.

Dentro do módulo Social vimos APIs para interoperar com Facebook, Twitter e WeChat, bem como para compartilhar informação com algum dos programas instalados no dispositivo Vejamos exemplos de usos destas APIs.

Vemos que cada API possui propriedades e métodos de acordo com sua funcionalidade. Por exemplo, a API Share possui métodos que nos permitem compartilhar um texto, uma imagem ou um arquivo.

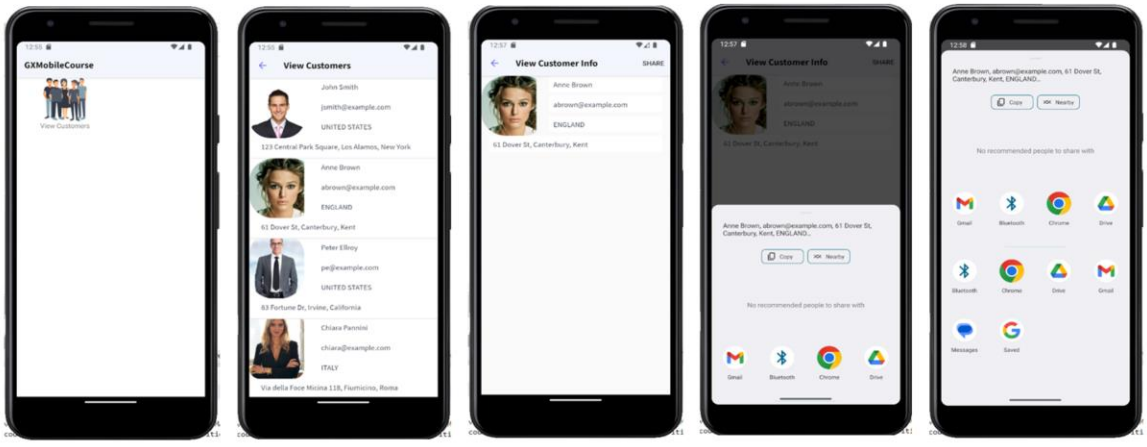
No caso das APIs das redes sociais Facebook e Twitter, os métodos permitem realizar as ações necessárias para interagir nessas redes, como, por exemplo, PostToWall do Facebook ou o método Follow do Twitter.



Vejamos um exemplo de uso da API Share para compartilhar os dados de um cliente.
Aqui já criamos um panel ViewCustomerInfo que tem os atributos de Customer e que recebe na regra Parm o CustomerId por parâmetro.
Já temos adicionado um botão SHARE à Application Bar.
Também temos definidas três variáveis, uma para cada parâmetro, ou seja, uma para o texto com os dados do cliente, outra para a url e outra com o título do que vamos compartilhar.
No evento do botão atribuímos valores a essas variáveis e escrevemos a invocação para a API Share, passando-lhe os parâmetros necessários.

No panel ViewCustomers adicionamos na propriedade Default Action do grid, uma ação com o nome "View customer info" e no evento correspondente invocamos o panel ViewCustomerInfo passando por parâmetro o identificador do cliente selecionado.

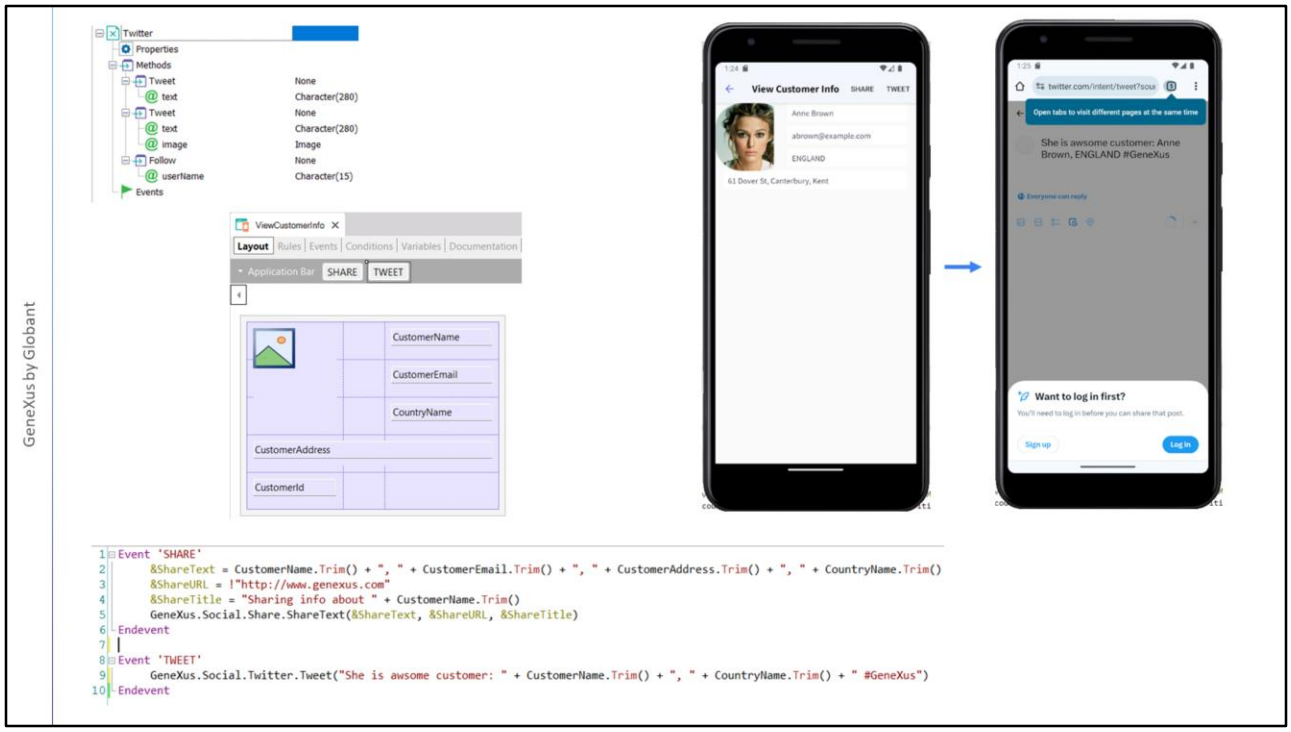
Executamos….



Tocamos sobre o ícone View Customers e depois tocamos sobre a cliente Anne Brown.

Vemos que se abre o panel com os dados de Anne e acima à direita aparece o botão SHARE.

Se o pressionarmos... Vemos que nos aparecem os dados da cliente para compartilhar e as aplicações que o dispositivo nos oferece nas quais podemos usar esta informação ou compartilhá-la.



Também poderíamos querer oferecer a ação de “twittar” uma mensagem fazendo referência ao cliente selecionado.

Para implementar isto utilizamos a API Twitter e o método Tweet. Se observarmos a API, nos oferece dois métodos Tweet: em um apenas enviamos a mensagem a ser “tweetada” e no outro o texto e a imagem. E por outro lado temos o método Follow.

Voltamos ao panel ViewCustomerInfo, adicionamos um botão TWEET à Aplication Bar e no evento do botão invocamos o método Tweet da API Twitter, passando por parâmetro o texto que queremos twittar com a hashtag no final.

Executamos... escolhemos novamente Anne Brown e vemos que é aberto o browser na página do Tweeter que nos pede para fazer login, mas ainda pode ser vista a mensagem que twittamos. Se tivéssemos a app do Twitter instalada, nos seria oferecido enviar mensagem direta, ou escrever um tweet geral.

Poderíamos querer fazer o mesmo, mas com Facebook, para o qual utilizaríamos a API correspondente.

Neste vídeo vimos a quantidade de APIs que temos para diferentes requisitos e mostramos alguns exemplos de uso.

À medida que GeneXus evolui, estarão disponíveis mais objetos externos com novas funcionalidades, por isso sugerimos consultar a documentação da wiki frequentemente