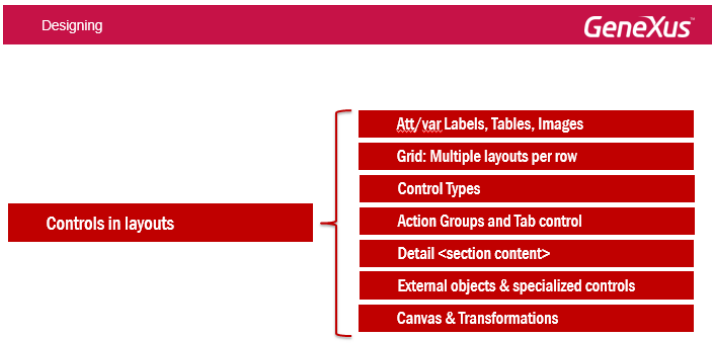
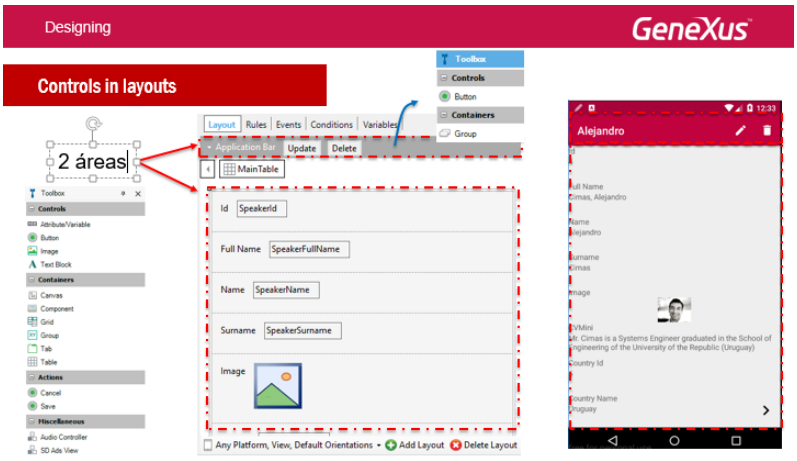


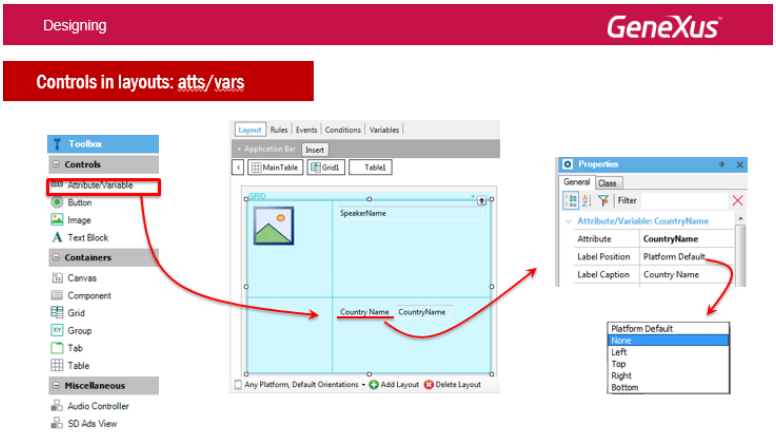
Agora vamos nos dedicar às particularidades que os controles assumem nos layouts, tudo isso comparado ao uso que já conhecemos para web panels, fundamentalmente, os aspectos que indicaremos aqui, começando pelo primeiro.



1 Duas áreas compõem um layout:

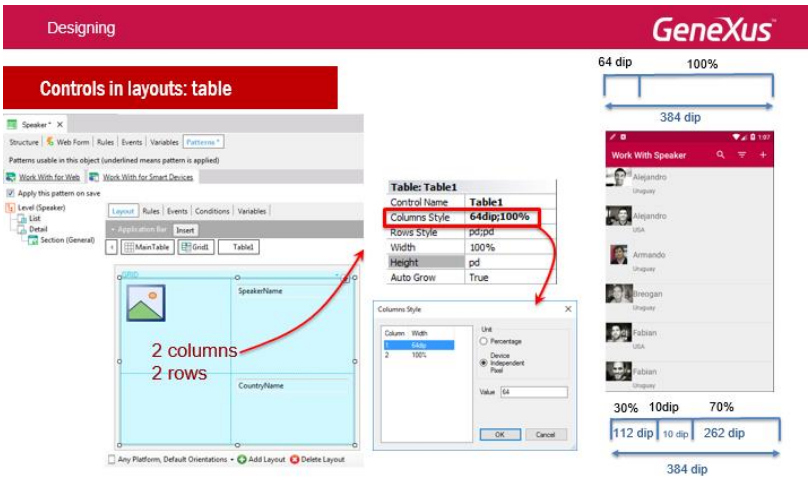
A área conhecida como **Application bar**, onde é possível colocar botões (que podem ter imagens asociadas ou não), e que irão corresponder a ações a serem realizadas sobre os dados ou chamar um tela ou o que quer que seja. Ainda é possível colocar grupos de ações (Group controle), para agrupar várias ações e disponibiliza-as, por exemplo, como menus drop-down.

E por outro lado vamos ter a área do layout propriamente dita, que nos oferece a toolbox para poder inserir os controles (se trata-se do nodo Detail, vamos ver depois que vão aparecer, além destes que estamos vendo aqui, os placeholders para as diferentes seções).
Bom, vamos nos concentrar nesta segunda área agora e deixaremos a primeira para mais adiante.



Aqui estamos vendo o List do Work with de Speakers. Vemos que, por default, estava mostrando SpeakerImage e SpeakerName. E vamos adicionar um atributo, ou country ou país do orador.

Para os controles atributos/variáveis, temos a propriedade **Label Position** que permite determinar a posição que o label ocupará em relação ao atributo/variável. Cada plataforma tem seu padrão (em Android, por exemplo, é Top, o que significa que o label vai aparecer em cima do controle atributo e não como estamos vendo aqui, que é à esquerda, porque de algum modo precisa mostrá-lo e esta tela é genérica, então não é uma tela específica para Android.
Se selecionamos o valor "None" então o label não será mostrado. Vamos ver isso no GeneXus.



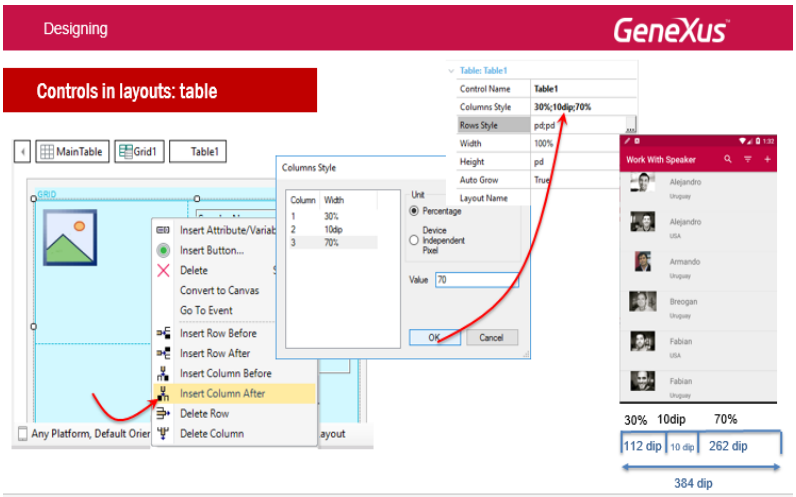
No work with speaker, vamos ao list e, bom, aqui vemos então os dois controles que temos agora, na grid do layout. Vamos à tollbox e arrastamos um novo atributo, escolhemos CountryName e vemos que, antes de tudo, foi adionada uma nova linha e coluna na tabela da grid e, por outro lado, se acessarmos as propriedades, vemos que dispõe-se da propriedade level position com o valor Platform default. Se executamos, agora veremos que para android está em cima. Vamos alterar para none para que o label não apareça, salvamos, e executamos para ver a alteração.

Executamos Run sobre o objeto dashword. No emulador, que ficou aberto desde a execução anterior, se observarmos antes de carregar o novo apk, vemos que estava mostrando somente a foto e o nome e agora, no novo, vai a ter que aparecer o Pais. Bom, podemos ver que está demorando para fazer a compilação. Neste caso, na verdade, vamos deixar assim porque não temos saída já que estamos adicionando um novo controle, um novo atributo, e esse vai ter que acessar o banco de dados para buscar a informação e, assim, neste caso, não podemos evitar todos estes passos de compilação e subida à nuvem, porém, vamos ver em quais caso é possível evitar tudo isso. Um pouco depois, então, vemos que aqui o apk foi carregado e se acessarmos speakers, agora está aparecendo o país. Vamos voltar ao ppt, bem..

O trabalho com as tabelas vai ser importantíssimo nas aplicações para Smart Devices. Todo layout, quando vazio, tem uma tabela raiz, chamada MainTable. Podemos vê-la aquí, quando estamos posicionados sobre a grid, depois sobre a tabela do grid e assim vai mostrando o controle sobre o qual estamos posicionados e todas as hierarquias, o pai e onde está inserido esse pai. No final, tudo controle está inserido na Main Table.

Por sua vez, toda grid tem uma tabela definida, quer dizer, esta tabela irá para conter os controles de cada linha a ser carregada, de cada elemento dessa grid que vai ser carregado do banco de dados. Neste caso, a tabela tem duas colunas e duas linhas. Observemos as duas propriedades: **Columns Style** e **Rows Style**, que vão permitir definir o tamanho que cada coluna e linha da tabela ocupará.
Se observamos as colunas, vemos que seus tamanhos podem ser definidos ou especificados em duas unidades: porcentagem ou dips (**d**evice **i**ndependent **p**ixel).
Esta unidade corresponde a uma abstração de um pixel que depois uma aplicação converte para pixels físicos, o que vai permitir escalar diferentes tamanhos de tela usando tamanhos uniformes, ou dip.
Os dips para cada plataforma tem diferente número de pixels.

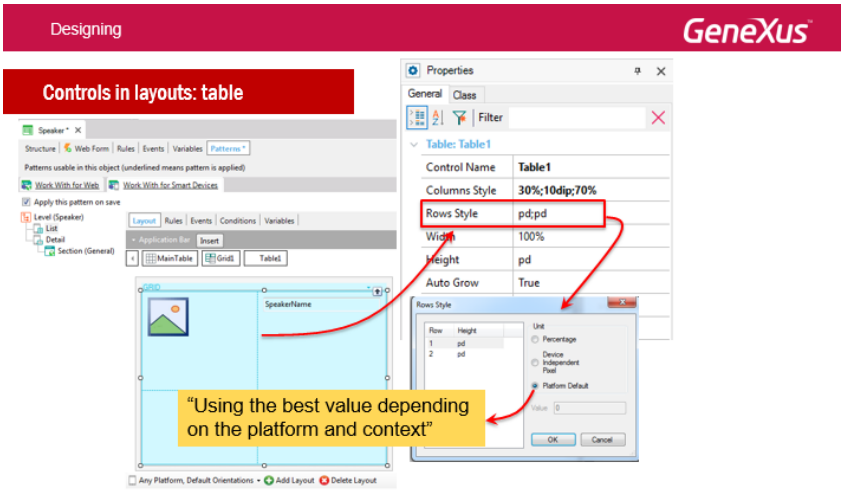
Os percentuais são relativos ao valor restante da largura total, os valores fixos (em dips). Assim, se a largura total da tela é de 384 dips e existe uma coluna de 64 dips, como vemos aquí, o 100% referente à coluna restante vai a ser a diferença entre a largura total e quantidade de dips de colunas fixas. Esse vai a ser o tamanho dessa coluna, no caso, 320 dips de largura.
Se tivéssemos três colunas, como mostramos aqui, a primeira com 30%, a segunda com valor fixo 10 dips e a terceira com 70%, os valores assumidos pela primeira e a terceira, que estão em porcentagem, serão obtidos aplicando esses percentuais ao valor resultante da subtração entre a soma dos valores fixos (aqui é só um, 10 dips) e o valor total da tela, neste caso 384. Ou seja 384 - 10= 374. Então, a quantidade de dips que a primeira e terceira colunas irão ocupar serão 30% e 70% desse valor (374).



Então, vamos fazer isso, vamos adicionar uma coluna. Nossa tabela tinha duas colunas. Vamos adicionar uma terceira de 10 dips para dar um espaço, como podemos ver aquí. um ar entre a primeira coluna e a tercera para que os dados não fiquem grudados, como ficavam neste caso da aplicação. Vamos voltar ao GeneXus, no world with do spikers e aqui nos posicionamos de qualquer lado e damos um insert column after, e agora vemos que temos 3 colunas. Nos posicionamos na tabela e modificamos suas propriedades. Vemos que os tamanhos da coluna assumidos são : a primeira 64dips, a segunda 50% restante e a terceira 50% restante. Então vamos modificar esses valores para que a primeira asuma os 30% a segunda um tamanho fixo de 10dips e a terceira os 70% restantes.

Bom, perfeito. Vamos testar, como fazemos? Por enquanto, vamos fazer à moda antiga, clicamos com o botão direito outra vez sobre o dashram, run, a menos que tívessemos um start up object, aí poderíamos dar F5 diretamente.
Novamente vamos colocar em edição. Esta vai ser uma alteração que não necesitamos compilar a aplicação e podemos ver instantaneamente e não esperar tudo este tempo.

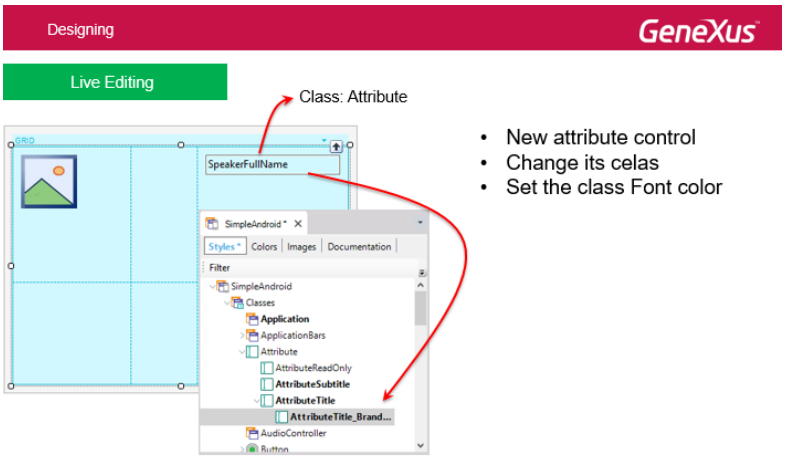
Bom, aí está, aparecendo a aplicação. Então vamos a speakers e agora é possível ver este espaço entre um dado e o outro. Isso é o que acabamos de ver, perfeito.



Por outro lado, também, se observarmos as linhas, vemos que foi adicionada a unidade pd, Platform Default, que não estava no caso anterior.

Este valor, platform default, é diferente de plataforma para plataforma. Justamente por isso é um default por plataforma, e para uma mesma plataforma, também depende do conteúdo da célula, por exemplo, se o campo tem label ou não. Se tiver, ela será mostrada em cima ou à esquerda, ou se é num layout em modo Edit ou View, tudo isso vai ser variável.

O valor platform default corresponde a: "Using the best value depending on the platform and the context". Por exemplo, para Android, com Label Postion = Top, corresponde a 64dips, enquanto que em iOS é 53.



Agora, se chegamos ao momento de diminuir os tempos de teste das alterações de desenho e então vamos ver a ferramenta live editing.
Então, até agora, cada vez que quisermos testar as alterações dos tamanhos das linhas ou das colunas das tabelas, como vimos, tínhamos que especificar e gerar a aplicação e compilar, e tudo isso demorava um bom tempo.
Vamos introduzir esta ferramenta que vai permitir realizar alterações de desenho e algumas alterações de comportamento e ver esses alterações instantaneamente sem siquer salvar a alteração de tal modo que podemos desfazer imediatamente sem esperar, sem nenhuma perda de tempo.

Vamos ver isso com um exemplo. Então, o que vamos a fazer? Vamos observar primeiro o atributo que tínhamos na grid de work with speakers (SpeakerName). Vamos ver que esse speakers name tinham a classe AttributeTitle associada a ele e, por isso, o valor desse atributo sai maior, o nome do speakers saia maior que o resto. Se virmos no emulador, podemos observar que justamente o nome está saindo muito maior. Então, se olharmos em speaker vemos a propriedade associada à classe AttributeTitle. Podemos ver que nessa classe, então, temos este tamanho e isso é o que está fazendo ele ser maior.

Bom, vamos alterar este atributo, vamos exclui-lo este que tinha a classe AttributeTitle e vamos colocar outro atributo em seu lugar, o atributo speaker full name. Vamos esconder o label e coloca-lo como none, perfeito.

Observe, como estamos inserindo um atributo novo, o patterns é que havia colocado a classe AttributeTitle no attribute anterior, O atributo de agora tem a classe Attribute, que tem o valor padrão, que vai ficar igual ao atributo country name.

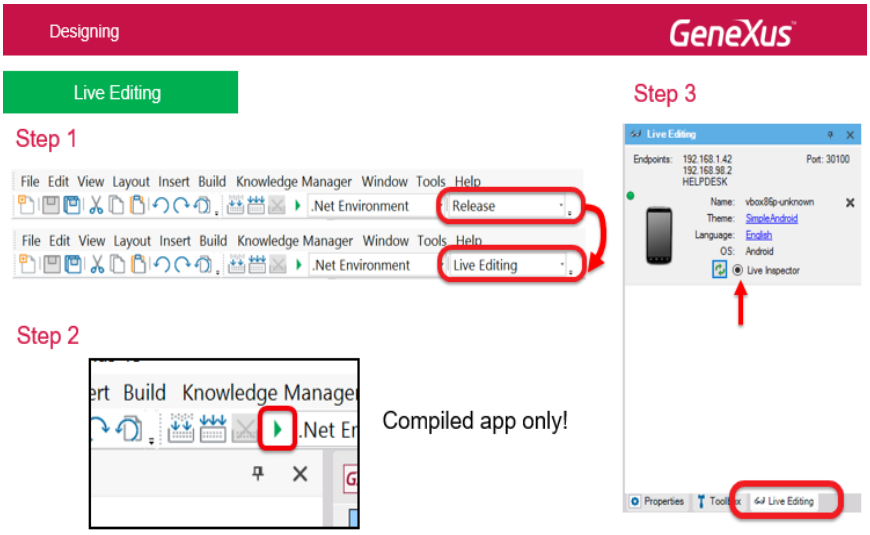
Então, primero, vamos alterar sua classe, por exemplo, em vez de colocar como título, poderíamos querer alterar para que tenha também a forma que tinha antes, porém o que vamos fazer não é só isso, mas que também apareça com a cor da marca, que apareça então com essa cor roxa que tinha na marca.

Então, para isso, vamos salvar este atributo Vamos deixar especificando e generando porque alteramos um atributo e, nesse caso, não tem outra saída, a não ser compilar novamente porque vai ter que ir até o banco de dados e buscar essa informação nova. Enquanto vou fazendo o run, vou dar uma olhada no tema do Android. Vou adicionar a classe AttributeTitle embaixo da classe Attribute. Bom, já estava adicionada. Já adicionei, na verdade, para evitar demorar.
Vejam que adicionei a classe AttributeTitlebrand que vai ter como four cor a cor do brand, e também vejam que adicionei a classe atribute traks e já fui alterando suas cores nas diferentes subclasses. Fiz a mesma coisa para as tabelas. Vejam que adicionei esta classe table track que, se abrirmos e expandirmos ela, também tem subclasses, por exemplo, blue para que a cor de fundo seja azul quando está selecionada ou branco e azul , etc.

Veremos isso mais adiante, vai a ser o que dará esse aspecto de desenho bonito à aplicação Android, que vai mudando a cor de acordo com a conferência, os traks.

Bom, então aqui vemos o que estamos falando no brand cor, que não sei por que não está mostrando, deveria estar mostrando a cor. Vamos agora ver os speakers e podemos observar, então, que agora o fullname está do mesmo tamanho que o country name, perfeito.
Então o que dissemos que queríamos fazer? Mudar a classe deste atributo para que assuma a classe nova que criei, porém para fazer isto, teríamos que alterar a classe, executar o Run sobre o dashboard outra vez, porém, não quero isso. Quero e fazer a alteração e isso seja refletido no emulador automaticamente, que eu possa ver a alteração sem fazer mais nada, sem nem sequer salvar. Para isso, dissemos que contamos com a ferramenta Live editing.

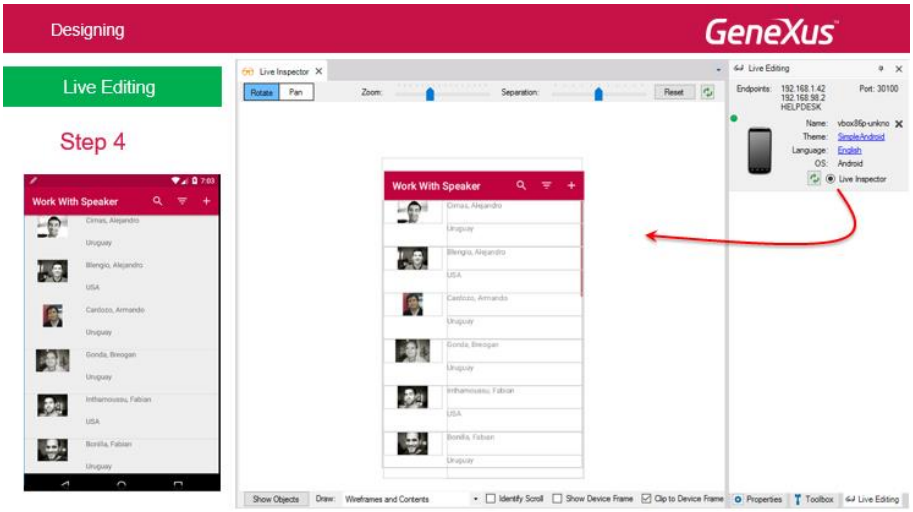
Como utilizamos essa ferramenta? Na parte superior do genexus tem um combo cujo valor está Release. Vamos mudar o valor para Live Editing. Vemos que, ao fazer isso, aparece esta aba aqui abaixo, que agora não tem nada. Agora sim, eu tenho que compilar a aplicação para que live editing comece a “escutar” essa aplicação compilada no dispositivo e se conecte. Vamos ver o live editing enquanto esperamos que termine a compilação.



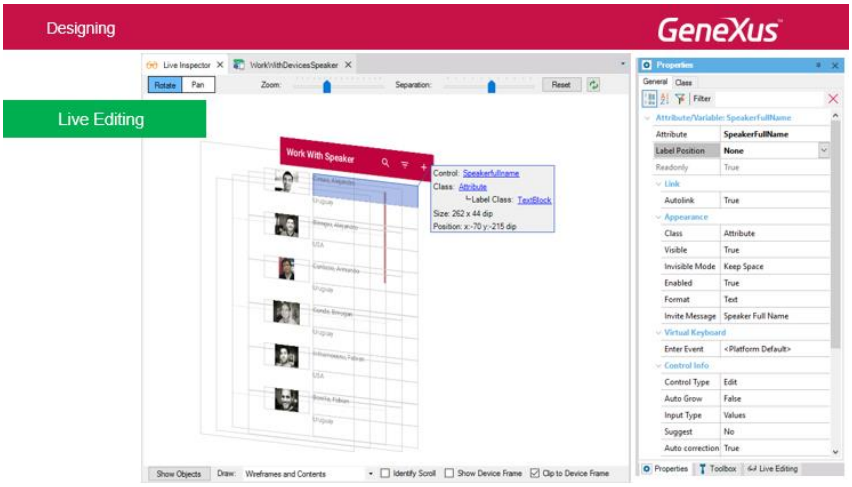
Live Editing é uma ferramenta de GeneXus que permite realizar modificações em themes, languages, layouts e eventos de usuario de nossos panels e ver as alterações instantaneamente na aplicação em execução, sem ter que especificar, gerar e compilar. E sem sequer ter que salvar as alterações.
Já repeti varias vezes porém isso é muito importante.

Desta maneira, o servidor de Live Editing vai ficar “escutando” as alterações que fizermos na aplicação e vai replicar estas alterações sobre o metadata que o dispositivo móvel acessa (nosso caso é o emulador, porém, poderia ser um dispositivo real)

Esse metadata é utilizado para desenhar a tela no dispositivo e para definir seu comportamento. Então, modificando o metadata vai ser possível fazer as modificações instantaneamente, sem necessidade de compilar nada.
Vamos ver se já executou…

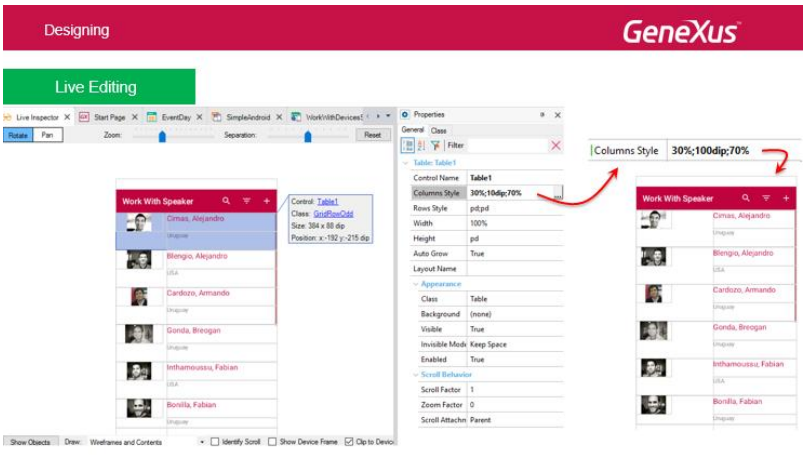


Vemos que agora, a aplicação apareceu em nosso emulador e que está aparecendo esta entrada aqui que está me dizendo que está escutando este emulador, que tem o tema simple Android, que o lenguaje é english e que o sistema operacional é Android.

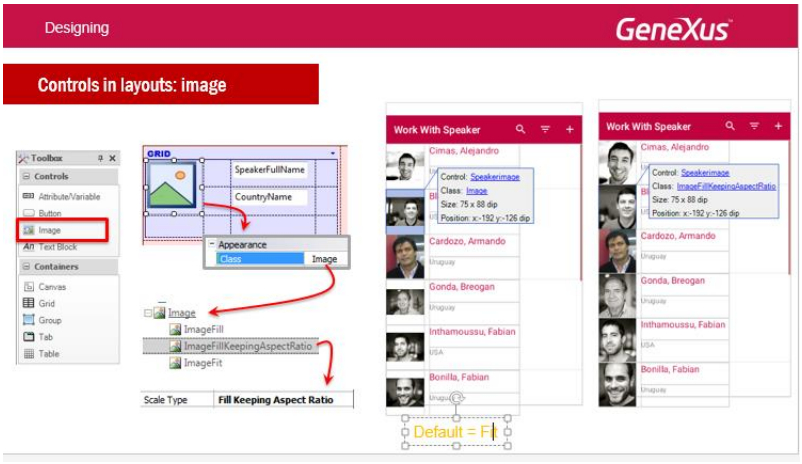


O que eu quero fazer? Ativar o live spector. Ao fazer isso, vemos que é aberta outra aba aquí, que agora parece não estar mostrando nada, porque o dashboard não é um elemento que inspescionável, modificável, e suas alterações não são visíveis nesta tela. Porém, se escolhemos speakers, no emulador, por exemplo, vemos que está refletindo automaticamente nesta outra tela aquí. Por exemplo, se dermos um tap sobre o orador Alejandro Cimas, vemos como a informação é carregada automaticamente na aba do live inspector ao mesmo tempo que vemos a informação em tempo real no dispositivo. Volto para tela anterior e vemos que atualiza outra vez. Além disso, nós podemos trabalhar diretamente nesta tela e alterar aspectos de desenho e vê-los automaticamente refletidos no emulador. Esta tela nos permite fazer um zoom, aumentar ou diminuir e, também girando-a com o mouse, podemos ver as diferentes camadas que compõem o layout que estamos visualizando para poder trabalhar melhor com ele. Por exemplo, separa-las mais ou não separa-las, voltar ao grau zero. Se, então, por exemplo, clicarmos no applicationbar, vamos às suas propriedades e vemos que aqui temos a classe. Vamos gira-lo um pouco ou deixa-lo plano. Posso resetar e também o temos de forma plana. Me posiciono sobre Cimas Alejandro. Vemos que me aparece de qual controle se trata, aparece ativo aqui nas propriedades do controle, aparece a classe e a subclasse, de maneira tal que eu posso modificar aspectos do controle tanto a nivel das propriedades, como das classes. Vamos a poder ver essas alterações automaticamente. Por exemplo, vamos a alterar a classe desse controle, mudando-a para essa que eu disse que tinha adicionado, que era atributetitle brand cor, e automaticamente, quando sair do campo, já posso ver a alteração.

Fiz a alteração diretamente no screen e vejam como foi refletida automaticamente no Live Inspector. O Live Inspector está escutando o que acontece no emulador. Então a alteração é feita automaticamente. Vejam todo o tempo que levamos neste caso, certo?



Bom, foi isso que dissemos, certo? Então podemos fazer estas alterações, ainda alterar o tamanho das colunas e tudo isso, ou poderíamos ter feito automaticamente.
Por exemplo, vamos a fazer isso agora, não nos custa nada. Nos posicionamos aqui no controle da tabela 1, o que tem as colunas e as linhas e, por exemplo, alteramos de 10 dips para 100 dips e quando saimos vemos que refletiu automaticamente a alteração e eu não tenho porque salvar esta alteração, mas quero e se não salvar, então vai voltar ao tamanho anterior.



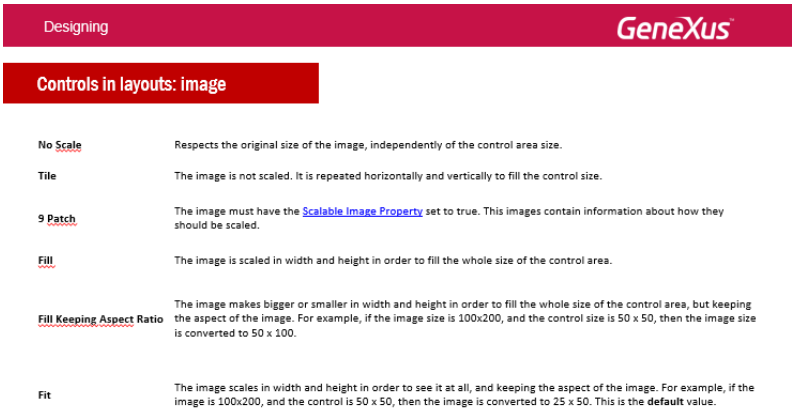
Outra questão interessante tem a ver com o uso das imagens. Aqui o que estamos vendo nestas imagens é que alteramos a propriedade Round span da imagem para que ocupe duas filas em vez de uma. Se observamos a classe Image default, não tem nenhum valor configurado para a propriedade scale type. Isso significa que vai assumir o valor fit, ou seja, que vai se encaixar no tamanho que tem. Se queremos alterar esse comportamento, vamos conseguir associando uma classe que configure a propriedade scale type correspondente, por exemplo, vemos aqui que esta está com o default (fit) e o que queremos que a propriedade tenha o valor fil keepin aspect reader, de maneira tal que a imagem se expanda, porém que mantenha as proporções, e então vai se ajeitar da melhor maneira.

Vamos ter que criar estas classes, que não existem, porque só vai a classe Image, fazendo variar então o scale type.

Vamos ao GeneXus, vou ao tema, depois vou à classe Image.
Bom e outra vez, como eu já importei o tema, e já havia feito estes alterações, aparecem estas três classes, porém no tema default não vão aparecer, teremos que criar. Por exemplo, esta: Image keeping Aspect Ratio que é uma subclasse da classe Image. Vejam que a única coisa que configurei foi a propriedade scale type.
Aqui podemos filtrar somente pelas propriedades que foram modificadas e aí podemos ver melhor a propriedade.

Bom, perfeito, então o que vamos fazer são duas coisas: 1 – configurar a imagem para que se expanda e ocupe 2 linhas e não somente uma e aí vemos o resultado. Bom, aqui, esta coluna vamos ter que mover porque se deslocarmos para lá ficará num lugar mais apropriado. Excluimos a última coluna indesejada (delete column). Agora vamos ao live inspector. Temos tudo em ordem e vamos colocar tudo para ver melhor e o que dissemos é que não está ocupando tudo. A classe atual é a classe padrão Image e não está ocupando todo o espaço. Então, vamos modificar a classe para Image fill keeping Ratio e vemos como tudo se acomodou dessa maneira. Outra vez, se vemos no emulador é a mesma coisa. Vemos como esta tudo funcionando tal como desejávamos.

E bom, os controles imagens tem essas possibilidades, tipo de imagem que, bom, deixamos para que vocês vejam, porém, particularmente temos was have fit para preencher sem redimensionar proporcionalmente, para que preencha toda a área da tela, etc.



Bem, com isso terminamos então o primeiro dos temas de controles de layout. Vamos passar ao seguinte, Multiples layouts per fit.