PRACTICE EXERCISES

GeneXus for Mobile Course

Training Department - GeneXus

January 2024

CONTENTS

THE PROBLEM

A multinational company in charge of managing amusement parks hires you to develop a system on a native Android mobile platform to view and occasionally update the information it works with. Suppose that the system is composed of a module:

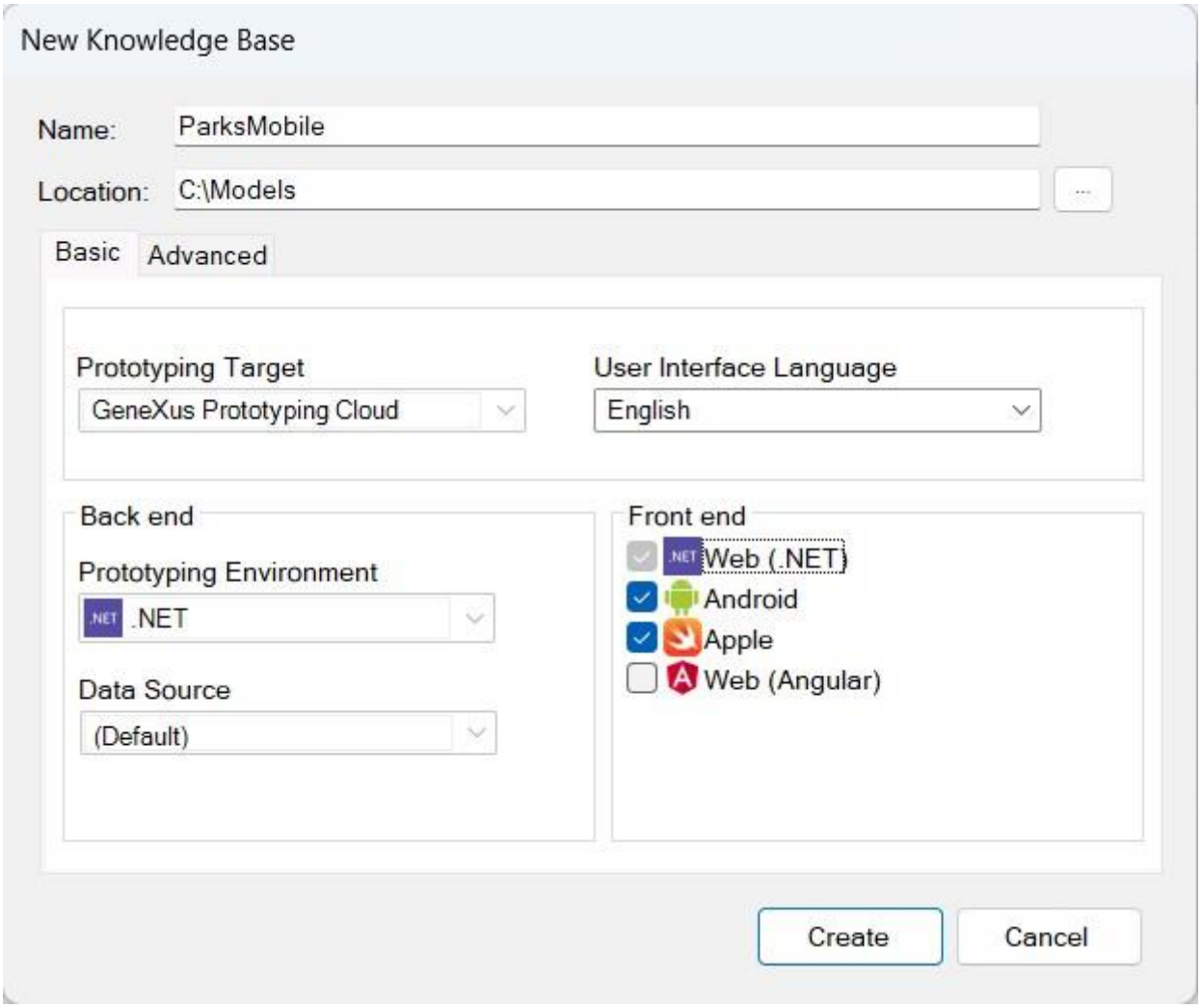
- **Back end**: part of the application that must run on a web server, composed of REST services generated in .NET, which must be accessible to the native mobile application from anywhere with an internet connection.
- **Mobile App (front end)**: part of the application generated in Android that will run on an emulator or on a native Android device, connected to the back-end web server via Wi-Fi, so that the application users can access the information from anywhere with an internet connection.

NEW PROJECT, NEW KNOWLEDGE BASE

Open GeneXus and create a knowledge base called *ParksMobile_FirstNameLastName* to start developing the application. Where it says “FirstNameLastName”, the first and last name of the student who is doing the practice exercise must be included. This name will be useful later when the KB is uploaded to GeneXus Server.

Tips:

- Choose .NET as your development environment. Make sure everything you need is installed (including SQL Server). If you use GeneXus Trial, the generation environment with .NET and SQL Server is already predefined, automatically prototyping in the Amazon cloud. If you use the GeneXus Learning or GeneXus Full version, you will have to enter this data when creating the KB.
- In the Front-end group, select Android. If you also want to generate the application for Apple, you can select that checkbox.
- Do not create the knowledge base in the “My Documents” folder or any other folder below “Documents and Settings” because these folders have special permissions granted by Windows.



TRANSACTION MODEL, BACK-OFFICE OBJECTS, AND TEST DATA

These practice exercises are focused on building native objects for a mobile application, so the first task is to import an .xpz file containing the transactions together with the data providers that allow populating the tables. Since all the transactions have the Work With for Web pattern applied, the objects generated by the pattern and other resources are also included, in order to have a complete and operational web back office.

The file to be imported is available on the course page, below the Materials section. Also available are the images that may be needed for the practice entities.

After importing the .xpz file, press **F5** to create the web back-office application and familiarize yourself with the data we are going to use. If you are using GeneXus Learning or GeneXus

Full, a dialog box will open to fill in the data corresponding to the database name, server name, and connection type.

Once the message that the Developer Menu has been successfully executed is displayed, in the launchpad window run the WWCountry object to view the country data.

On the left side of the screen, you will see a menu to work with other entities, such as amusement parks. We suggest you browse the different entities to see their data.

Tips:

- Keep the IDE properties window open (**F4**), as you will use it frequently.
- Verify that you have installed all the necessary Android generator requirements and the Android SDK.
- Make sure that in the Front-end node properties, the Generate Android property is set to True and the Main Platform property is set to Android. Optionally, you can leave the Generate Apple property set to True. Leave the rest of the properties with their default values.
- You can use the emulator that GeneXus creates automatically, or create your own as shown in the video "First steps with a native mobile application", but it is convenient that the Android emulator is started before running the native application.

LIVE EDITING CONFIGURATION

Before starting to create objects, configure Live Editing to speed up your work.

Remember:

- Change the combo box that is set to **Release** for **Live Editing**.
- Select *Rebuild All* for the change to be applied: this will only be necessary the first time.

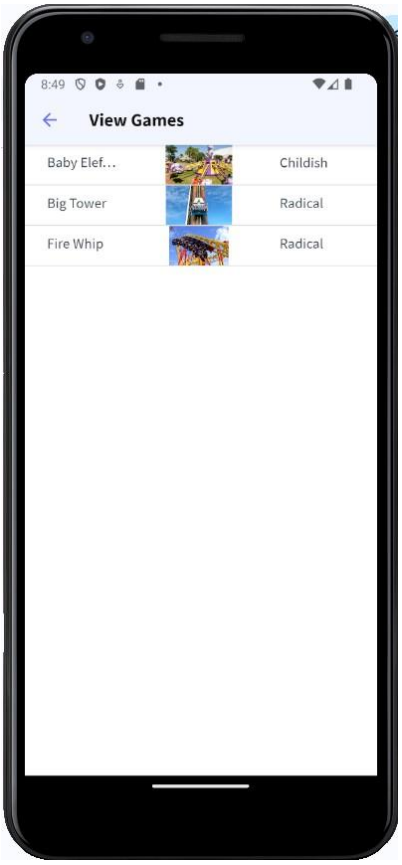
FIRST STEPS AND BASIC UI CONTROLS

The company that manages the parks has requested the implementation of a screen (Panel object) named ViewParks that shows the list of available amusement parks, with information of the identifier, name, photo, and name of the city and country of each one.

The screen should look similar to the following:

Tapping one of the parks should open another screen showing its rides, each one with the corresponding name, photo, and category name.

This screen should be called ViewGames and look similar to the following:



Remember:

- In order to display the rides of the selected park, the ViewGames panel must receive the amusement park identifier in a parameter and it must be passed as an argument in the invocation to ViewGames from ViewParks.

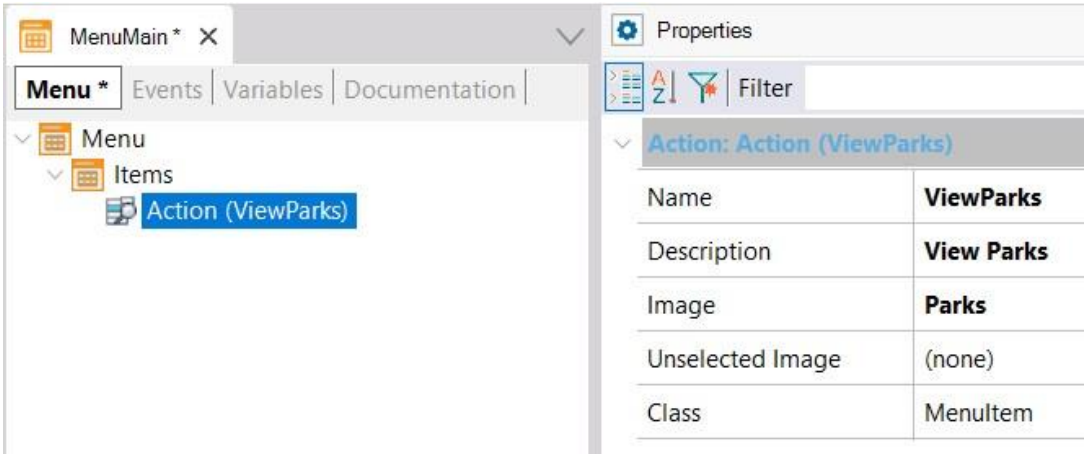
We're going to run the ViewParks screen from a menu object.

To do so, we create a Menu object and name it MenuMain. Then we drag the ViewParks object to the Items node of the menu object.

We go to the properties of the created action and set the Image property with the Parks image, which was previously imported in the xpz that allowed us to import the back office.

If we go to the events, we will see the ViewParks event and with the invocation to the ViewParks panel:

The menu object should look as follows:



Remember:

- Set the MenuMain object as startup by right-clicking on the object name and selecting **Set as startup object**. This will cause the MenuMain object to run on the emulator or device when you press F5.

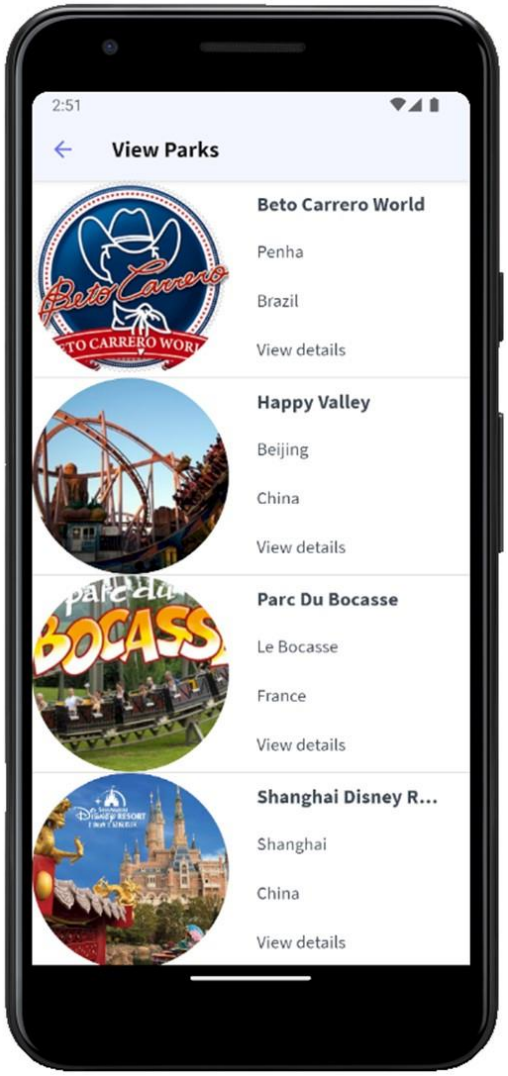
Now let's click on the MenuMain object's tab and go to its properties.

In the **Application Title** property, we type Amusement Parks. Then in the Android group, in the **Android Application Icon** property, we select the AmusementParksIcon image, which also came in the xpz we imported.

Make sure that you already have the emulator running and press **F5** to run the MenuMain object and see the ViewParks panel running. Take this opportunity to verify, in the device, the name of the application and the icon assigned. Also, check that the ViewGames panel with the rides of the selected park is displayed correctly.

APPLYING A DESIGN TO THE AMUSEMENT PARK SCREEN

Add a design to the park display screen that you have just created, and change the presentation of the attributes in the Grid to look as follows:



This implies that the photo of the park is on the left and the rest of the attributes are on the right slightly separated from the photo. Also, that the photo takes up all the grid rows, that it is round, and that the name of the park appears in bold.

Tips:

- Create a new object of Design System type called ParksMobileDSO that inherits from UnanimoMobile with the necessary classes to change the appearance of the controls.
- Use properties such as ***border-radius*** and ***gx-content-mode*** to make the required changes. As a guide, to make the photo round, you should use a value in pixels for border-radius, which is half the value of the dips assigned to the photo column (for example, if you set the column to 180 dips wide, then border-radius should be 90px).

- To show the title in bold, open the UnanimoMobile Design System Object to see how the class that uses the attributes is defined. Copy all the properties to a new class and change the value of the font-family property from `$fonts.primary-regular` to `$fonts.primary-bold`.

Don't forget:

- To configure the Design System you've just created so that it is considered by the platform for which you are developing. **Customization** node → **Platforms**
- To associate the classes with the controls.
- That even though the identifier of the park should not be displayed, you need it in the grid to pass it in a parameter.

In addition, the company requests another screen named ViewParkDetail with the detailed information of the park selected by the user when tapping "View details":

This new screen should look like this, with the name of the park highlighted, the photo taking up the entire width of the screen, then the website and the address of the park with its respective labels, and a button with the BACK label:

Tapping the BACK button should take you back to the previous screen (ViewParks).

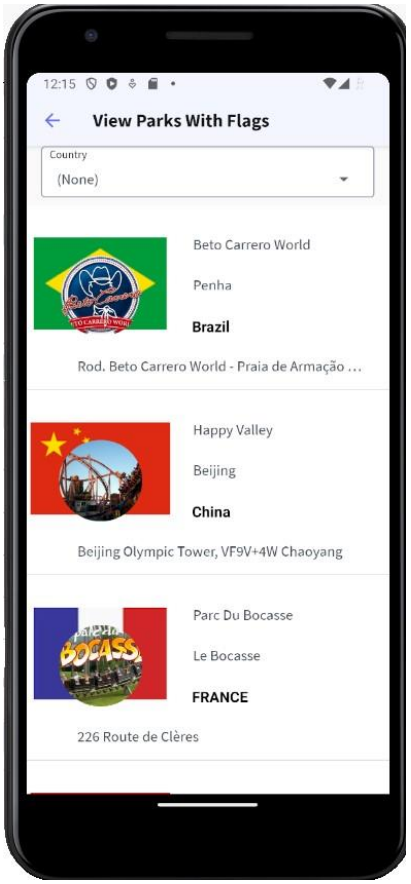
Tips:

- Create a variable and load it in the Start event with the text "View details".
- Use color tokens to facilitate your work on the styles.
- Use properties such as ***color***, ***background-color***, ***border-radius***, ***font-family*** and ***font-size*** to make the required changes.
- Use the ***Columns Style*** and ***Rows Style*** properties of the container table to define the space occupied by the elements in the rows and columns.
- Use the ***Horizontal Alignment*** and ***Vertical Alignment*** properties to align the elements in the cells.

CANVAS CONTROL

You are required to implement a panel named ViewParksWithFlags that displays the basic data of the amusement parks. In addition, each photo of the park should be displayed with the flag of its country in the background.

This screen should look similar to the following:



In addition, a combo box should be added at the top of the screen to select a country and filter the list of parks showing only those belonging to the selected country.

Lastly, we drag the ViewParksWithFlag panel to the Items node of the MenuMain menu, setting the Image property with the ParksWithFlags image.

Remember:

- The **Canvas** control is a container object of table type, so you can drag controls into it and adjust the Row Span or Column Span alignment values just like in a table.

- When inserting a control inside a Canvas, a Z-Order property is automatically added to define the layer where the control will be located. The value 0 is the bottom layer and

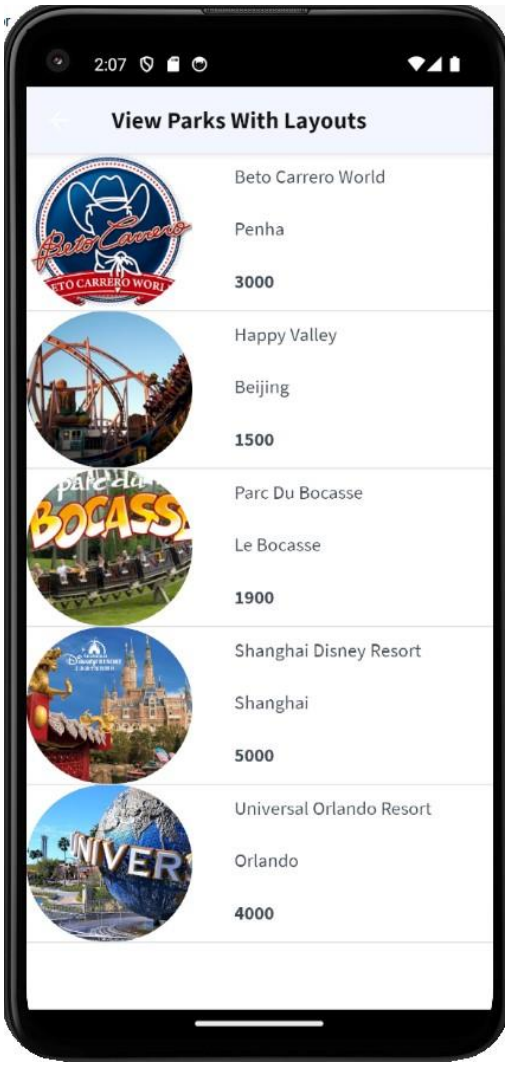
  the higher the number of **Z-Order**, the higher the layer of the control will be, so it is possible to superimpose controls.
- In addition to the Z-Order property, you must set the rest of the properties of the **Absolute position** group for the control to be correctly positioned in the canvas.
- For the selection of a value from the countries dynamic combo box to have an effect on the parks list, you must program the ControlValueChanged event of the dynamic combo box variable with a Refresh command and add a Parm rule with the combo box variable.

MULTIPLE LAYOUTS IN A GRID

The information displayed in the ViewParks panel has to be changed depending on certain conditions. Before proceeding, make a Save As of this panel and save it as ViewParksWithLayouts.

By default, you want it to display the amusement parks available in the database, showing the park's photo, name, city and average daily number of visitors.

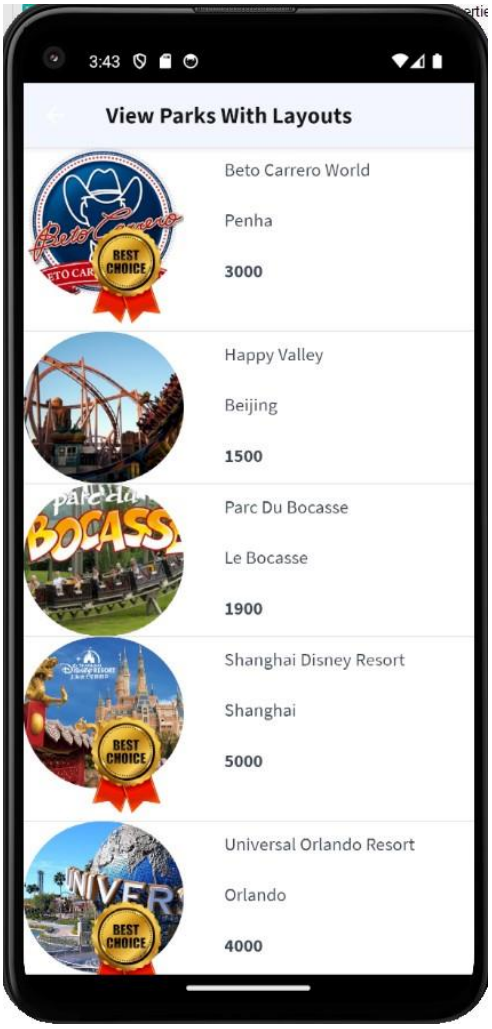
The screen with the default data should look similar to the following:



For those amusement parks where the number of visitors exceeds 2000 per day, the "Best Choice" stamp should be automatically superimposed on the park's photo. In addition, for

those parks, the number of visitors should be shown in red and highlighted in bold. For the rest, neither the stamp nor the number of visitors should be highlighted.

The screen with the “Best Choice” stamp superimposed should look similar to the following:

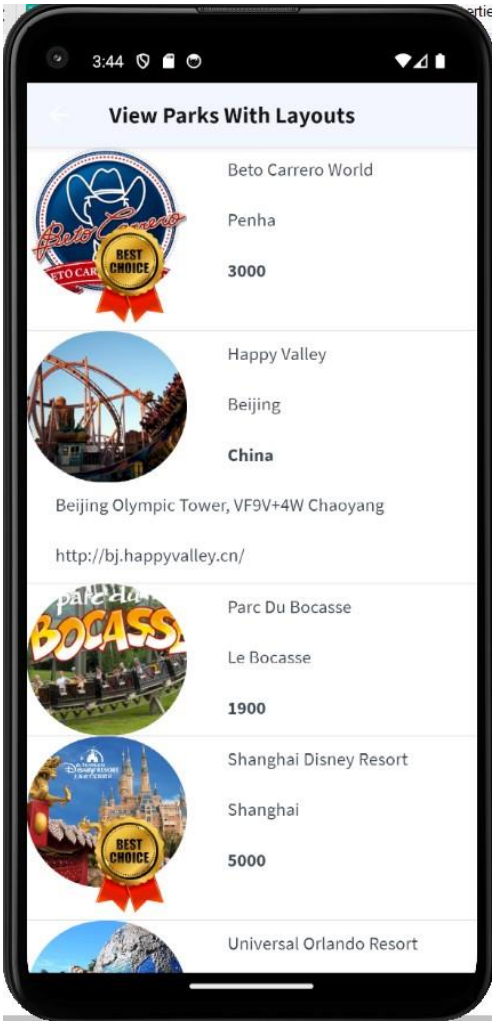


Tips:

- To be able to assign more than one layout to a panel, you must first create a new layout by clicking on the arrow in the upper right corner of the grid table and choosing Add New Item Layout.
- To superimpose the Best Choice stamp on the photo of the attraction, you can use a **canvas control**. To do so, you must first remove all the controls from the grid, insert the canvas in the grid and the stamp image, then insert a table control and insert the controls again.
- A simple way to do this is to first copy the grid table, then delete the controls from the table, insert the canvas with the photo and then paste the table on the canvas control. Finally, to the stamp image we assign a Z-Order value greater than that of the table containing the controls.
- Do not forget to set the properties of the **Absolute position** group of the image and give a height of 100% to the table.
- To dynamically assign a layout, you must do it in the Load event of the panel, checking the condition (in this case, that the number of visitors is greater than 2000) and assigning the ItemLayout property of the grid with the name of the corresponding layout.

In addition to the above, tapping any part of the park information should no longer invoke the panel displaying the rides, but should hide the number of visitors and display the address, country, and web page.

The screen with more information about the parks should look like the following:

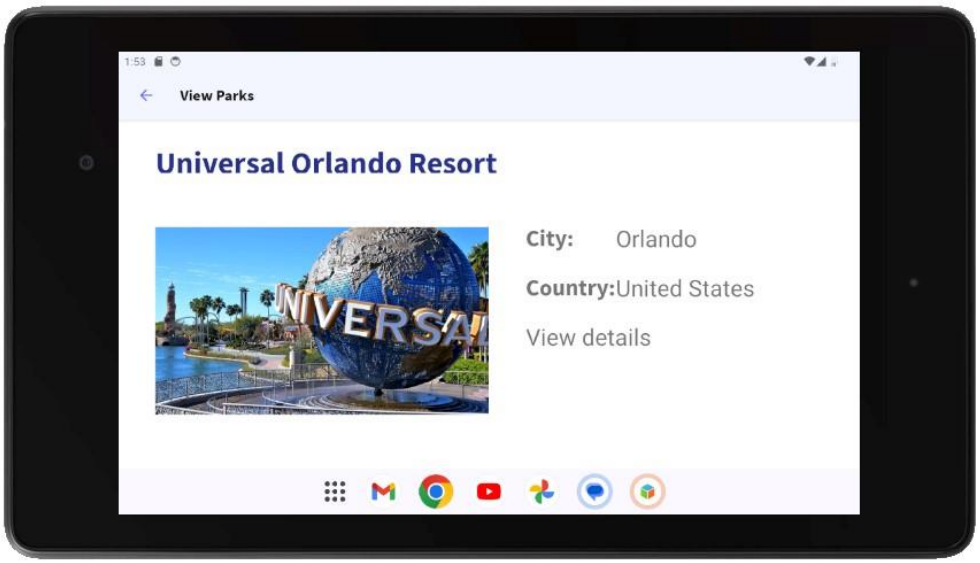


Remember:

- Until now, the value of the **Default Action** property of the grid was assigned to an event that invoked the ViewGames panel. In this way, when tapping on one of the parks, the screen showing the rides of the selected park would open. To invoke another layout different from the default layout (Layout1) when tapping it, you must restore the Default Action property to its default value and set the **Default selected item layout** property with the name of the layout that shows the details of the selected park.
- To run the ViewParksWithLayout panel, you can add it as an action to MenuMain and assign it an image as icon, or you can also set the panel as main object and run it directly.

MULTIPLE LAYOUTS FOR A PANEL

The ViewParks screen contents should be adjusted when the application is being used on a 7” Tablet in landscape orientation, so that it looks similar to the following:



Tips:

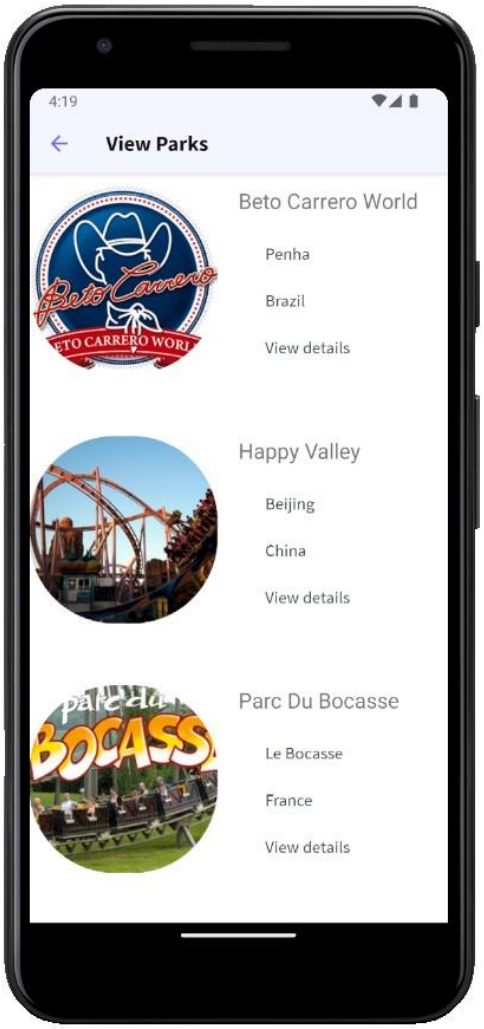
- Create a new device in Android Studio to simulate a 7” Tablet.
- Do not forget to create a new layout for the panel.
- Use the ***Columns Style***, ***Rows Style*** and ***Col Span*** properties of the containing table(s) to define the space taken up by the elements in the rows and columns.
- Use classes with the properties you used earlier to change the appearance of the park names and the size of the displayed information. To change the style of the attribute labels, use the gx-property `gx-label-class` and assign it a class to change that style.

Remember:

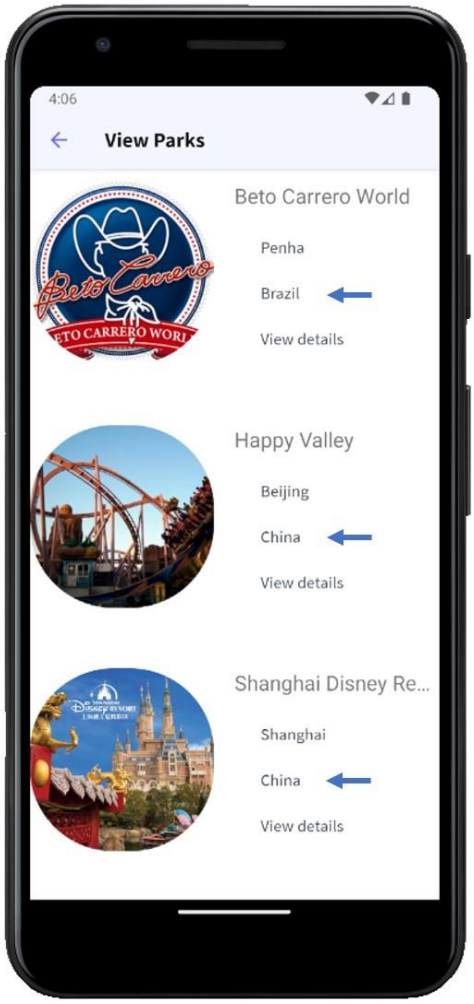
- To display a single record per row and scroll horizontally, you can use the ***Scroll direction***, ***Items Layout Mode*** and ***Items Per Column*** properties or change the Grid type to ***Horizontal.***

USE OF GRIDS

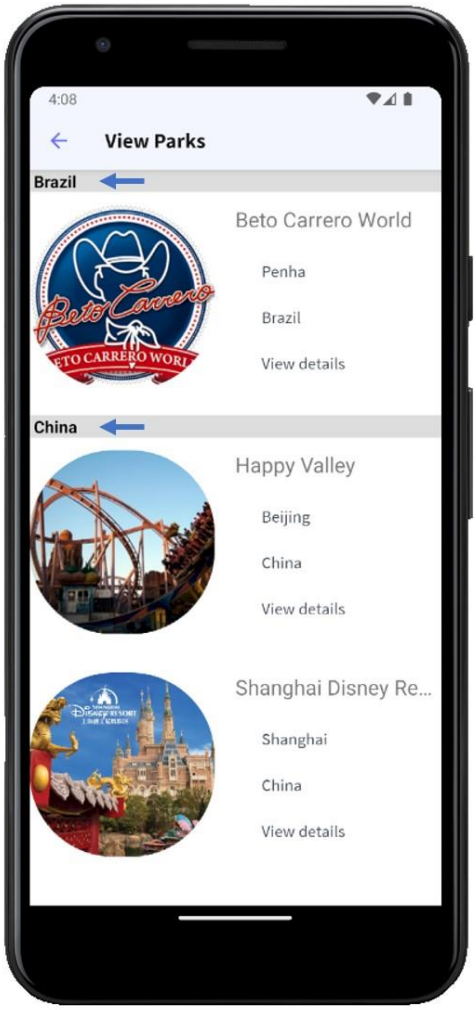
Modify the default layout of the ViewParks panel so that the Grid scrolls horizontally and displays a maximum of 3 rows per page. The screen should look similar to the following:



The data should now be sorted in ascending order by country name:



Now, amusement parks need to be grouped by country:



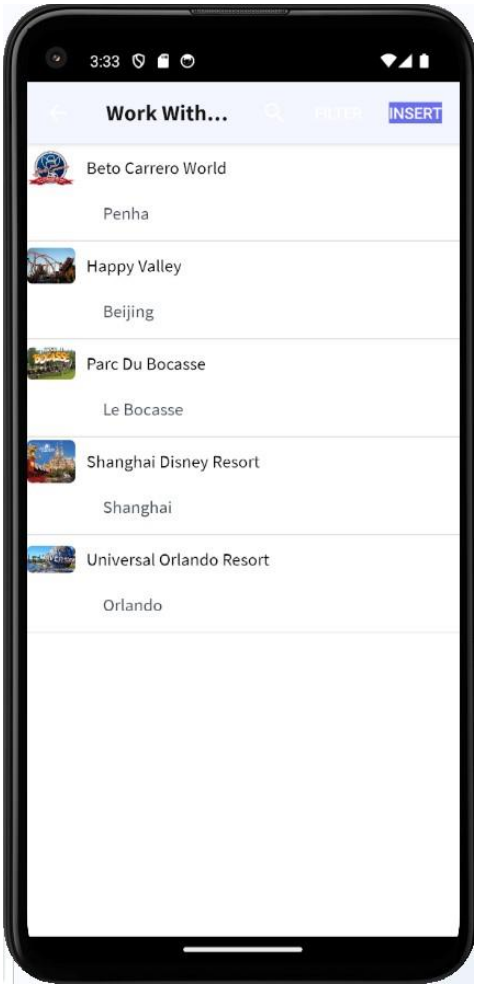
WORK WITH PATTERN

To perform CRUD operations (Create, Read, Update, Delete) on the amusement parks, in a simple way from a mobile device, the following is required:

Apply the Work With pattern to the AmusementPark transaction.

Add it as an item to the MenuMain Menu object.

When accessing the list of amusement parks through the Work With, we want to show, in addition to the name and photo of each park, the name of the city where it is located, as shown in the following image:



It is suggested that you improve the design of this screen.

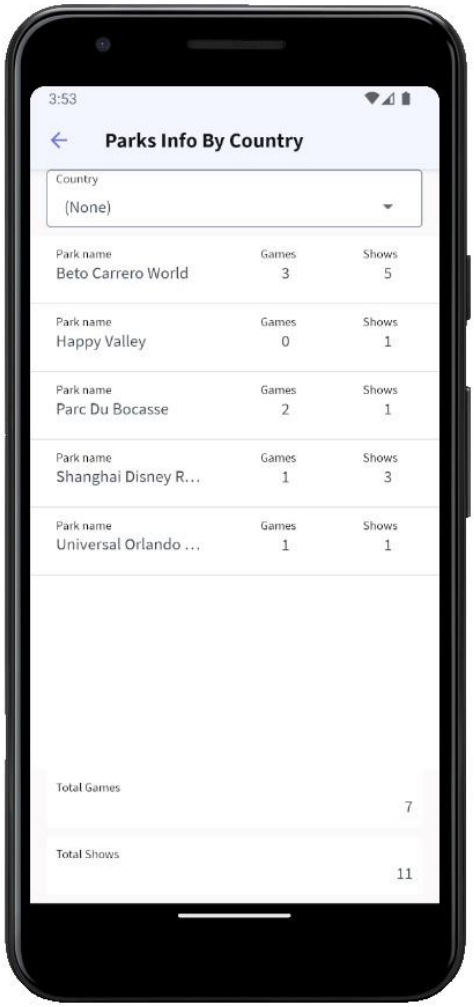
When making searches, in addition to being able to search by park name, we also want to search by city name in the same search. Therefore, when we search by park name, it should return the parks matching that name, and if we search by city name, it should return the parks in that city.

Tips:

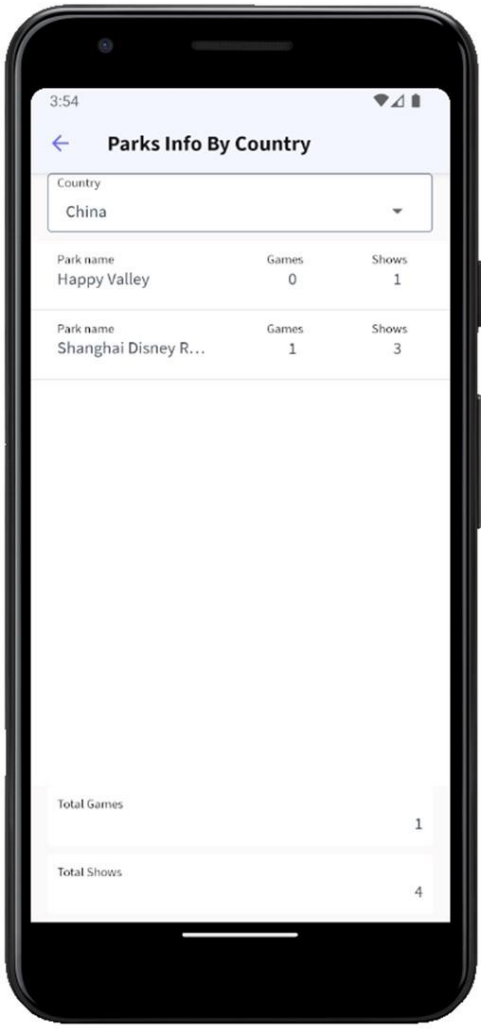
- To show the city next to the amusement park, we must do it from the List node of the WorkWith pattern, adding the corresponding attribute.
- To search by more than one attribute, the grid properties (in the List node) have to be configured by setting the "Search" property. The other attribute by which to search has to be added.

PROGRAMMING EVENTS ON A PANEL

It is necessary to develop a screen that shows, for each amusement park, its total number of rides and total number of shows. It should also allow filtering by country. In addition, below that list, it should show the overall totals. The screen should look similar to the following:



Filtering by country:



Add it as an item to the MenuMain Menu object.

Remember:

- Panels are programmed differently than WebPanels because the fixed part and the variable part use different Data Providers to be loaded.

Tips:

- Use the Load and Refresh events to load the variables of the Grid and the fixed part of the Panel, respectively.

USE OF APIS

Part 1)

It is necessary to invoke an API that allows creating an appointment in the device's calendar, in order to schedule a performance of a previously selected show.

Before proceeding, you need to view the list of shows of an amusement park and then view the details of the desired show. For this, in the ViewParksWithLayouts panel, an image representing the shows must be shown for each park, so that when clicking on it, it will take us to the list of shows of that park.

The shows icon in ViewParksWithLayouts can look as follows:

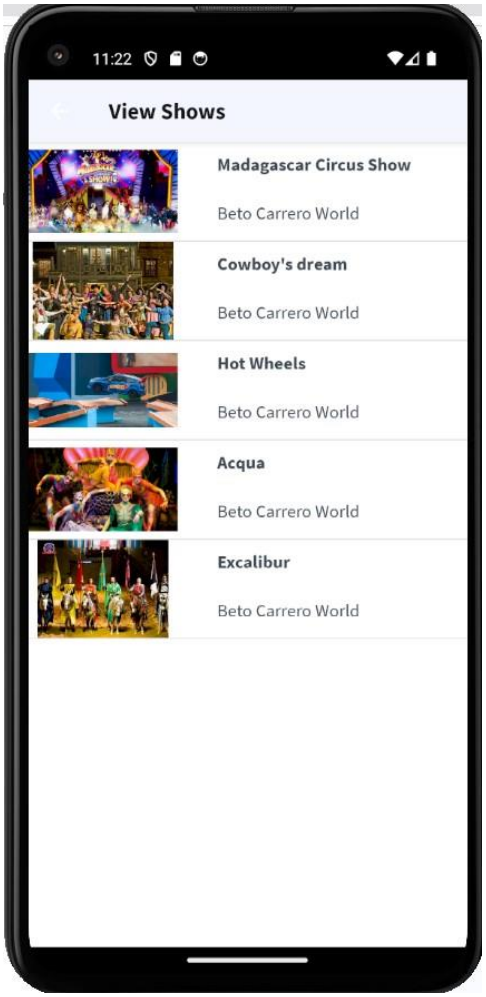
Tip:

- Create a &Shows variable of Image type, and in the Load event of the grid assign the image of the shows, with the FromImage method of the variable.

- Drag a table to the grid table and inside place the &Shows variable with the Label position property set to None. Repeat this for all the defined layouts that should be able to access the shows.

Clicking on the icon of a park's shows should open the ViewShows panel with the list of the park's shows, showing the photo and name of each show, and the selected park.

This panel should look similar to the following:



Then, when clicking on a show, the details of the invoked show should open. To this end, we will create a panel called ShowMoreInfo, with the enlarged photo of the show and the show's name, start and end time, city, and country.

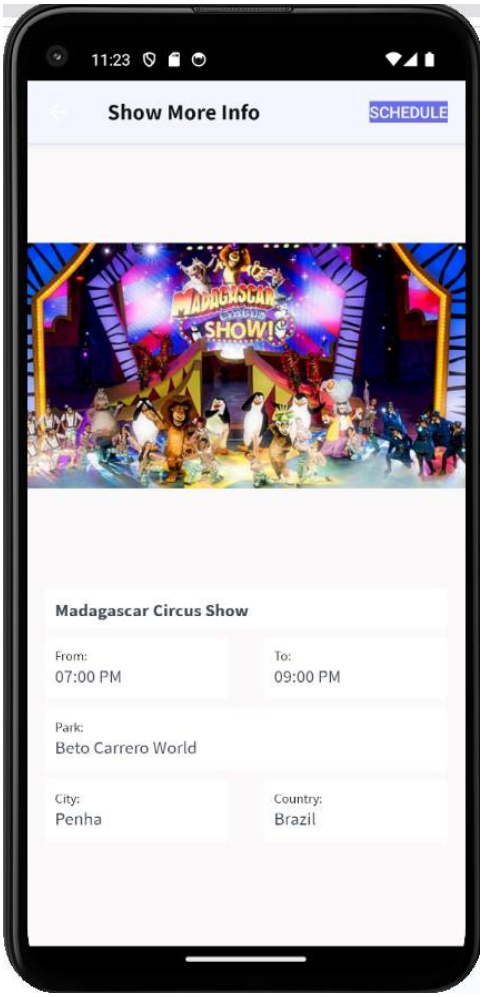
To create an appointment in the calendar as a reminder to attend this show, we add a button with the text SCHEDULE, and in its event we invoke a panel that allows entering the date of the selected show.

Remember:

- In the ShowMoreInfo panel, give the appropriate design to the table containing the controls, positioning them so that they look like the sample screen. Also, do not forget to assign appropriate values to the Column Style and Row Style properties of the table to display the Show details with the enlarged photo and readable data.

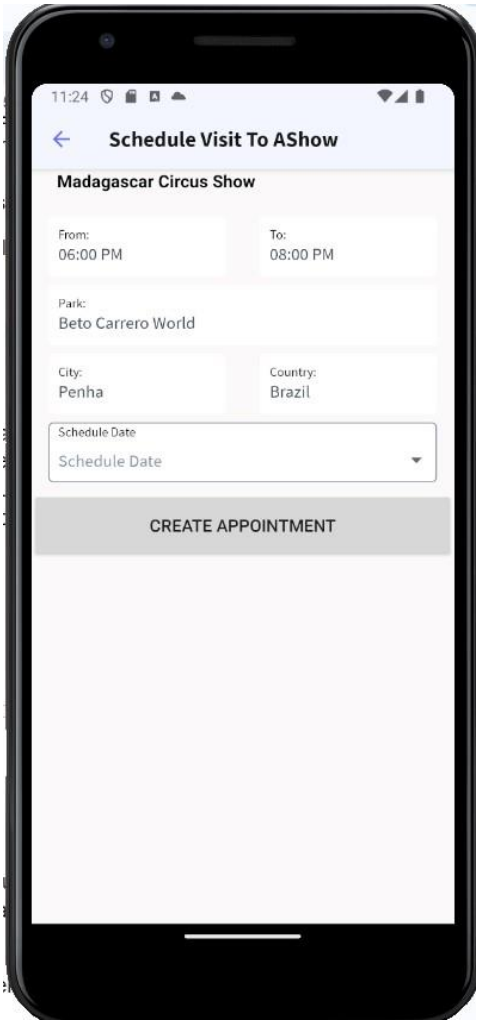
- You must insert the SCHEDULE button in the panel's Application Bar control by right-clicking on it and selecting Insert button....

The ShowMoreInfo panel with the show information and the SCHEDULE button should look as follows:



We must create the panel invoked by the SCHEDULE button and name it ScheduleVisitToAShow. It will display the show's data, including its start and end times, and it must allow entering the date we chose for the performance, since every day there is a show at the same time.

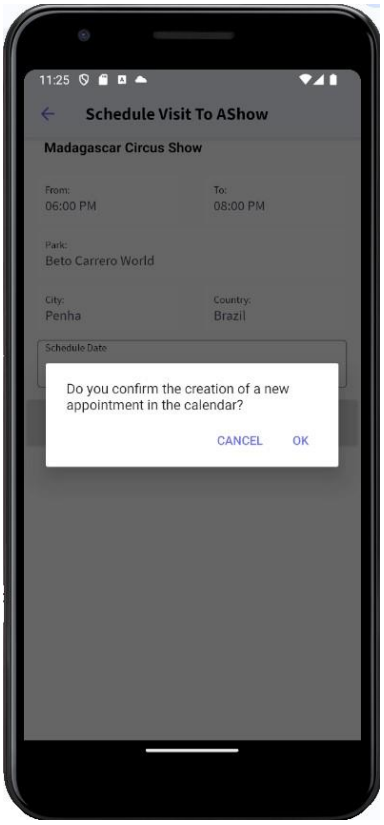
The ScheduleVisitToAShow panel screen should look similar to the following:



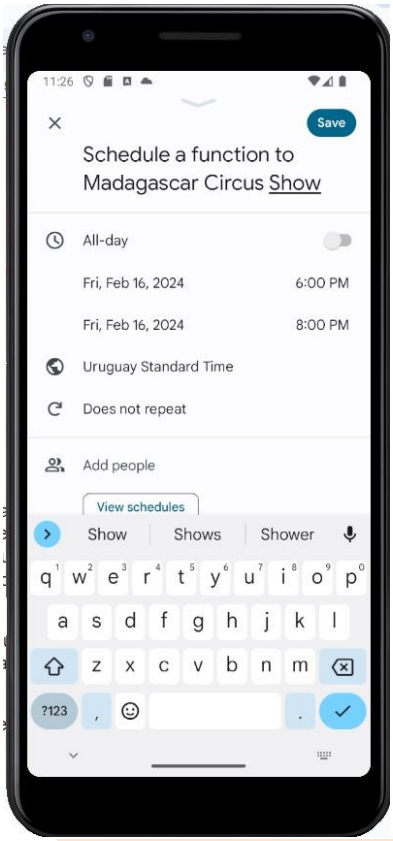
Then with that data, an API must be invoked to add an entry in the native calendar of the device. The invocation must include a text indicating that an appointment will be made for the selected show (for example “Schedule a performance of Madagascar Circus Show”) passing as parameters the required data that we have about the show.

Before the invocation is made, the user must be asked to confirm the creation of the new event in the calendar, with the option to confirm or decline the creation of the calendar appointment.

The popup with the dialog to confirm the action of creating the appointment in the calendar should look as shown:



Pressing the Confirm button should display the calendar appointment to be saved.



Remember:

- To have a confirmation dialog displayed, you must invoke the Confirm method of the Interop API, of the References node, GeneXus.SD module, passing it the corresponding message as a parameter.
- To invoke an API that adds an appointment in the device calendar, you must use the Schedule method of the Calendar API, located in the GeneXus.SD module, passing as a parameter the appointment text, start date, end date, start time, end time and a text with the location of the appointment.

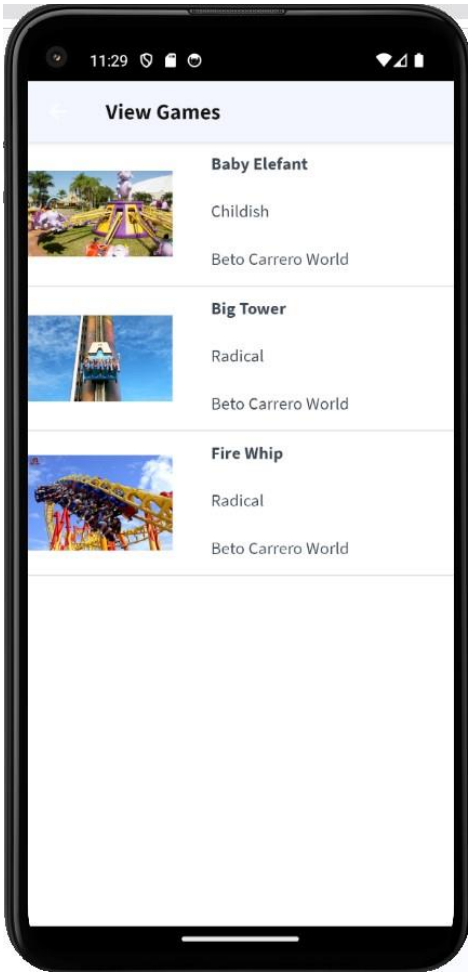
Part 2)

As part of the amusement parks promotion, a ride is to be advertised using a message on the X (Twitter) platform, showing a text with a maximum of 280 characters. The text of the message must include an invitation to visit the ride, mentioning the name of the ride, the park it belongs to, the city and country; for example: “Enjoy the Hulk ride at Shanghai Disney Resort amusement park located in Shanghai, China”.

To implement this, an icon to view the rides of that park must be added to the ViewParksWithLayouts panel, in a similar way to when we added the icon to view the shows.

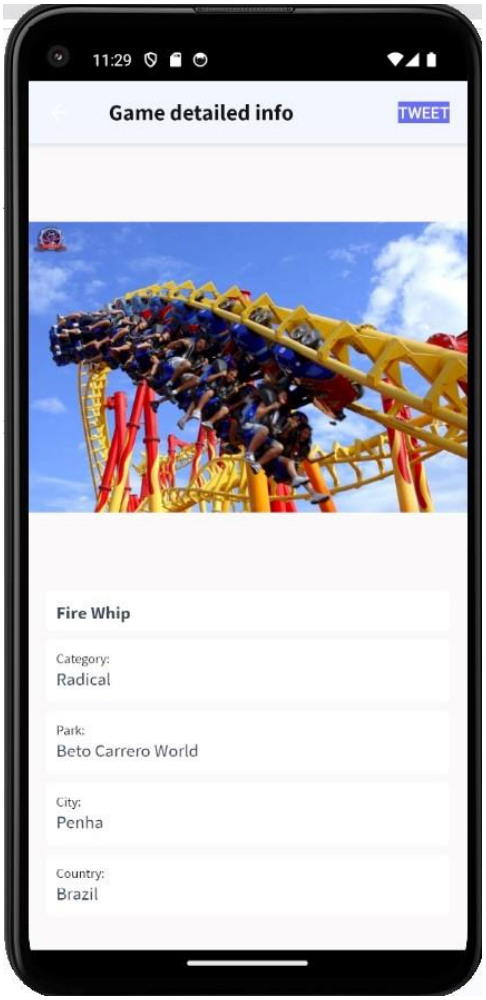
The ViewParksWithLayouts screen should look as follows:

Pressing the rides icon should invoke the ViewGames panel, which should look as shown below (add the appropriate design for that):



When clicking on a ride, a panel we will call GameMoreInfo should open with the ride's details, showing the ride's enlarged image and below it the name of the ride, its category, the park it belongs to, the city and the country. A TWEET button will also be added in the Application Bar to execute the action of posting the tweet.

The GameMoreInfo panel screen should look similar to the following:

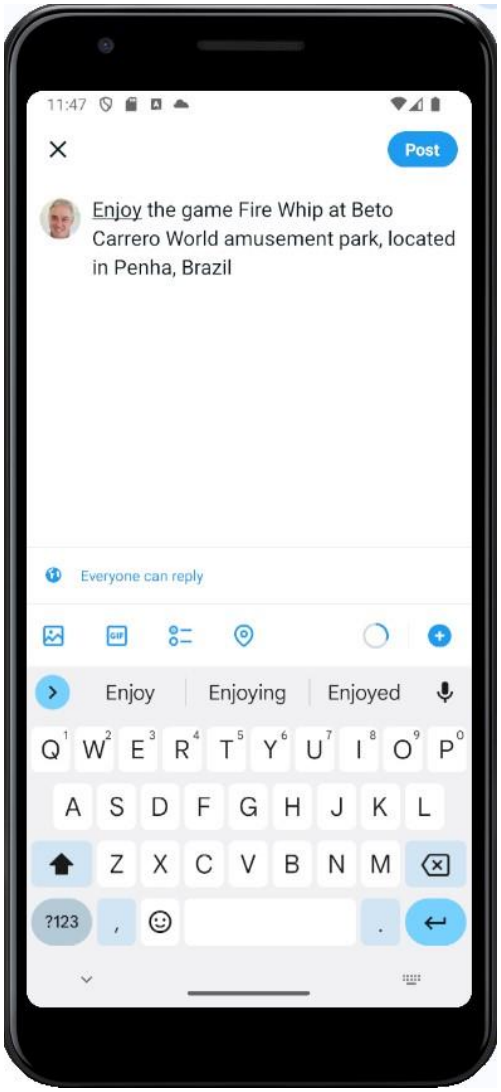


Tips:

- In the GameMoreInfo panel, give an adequate design to the table containing the controls, positioning them so that they look like the sample screen. Also, do not forget to assign appropriate values to the Column Style and Row Style properties of the table to view the details of the Ride with the enlarged picture and readable data.
- Insert the TWEET button in the panel's Application Bar control by right-clicking on it and selecting Insert button...

Pressing the TWEET button should invoke the API that posts the tweet, passing it the above-mentioned text as a parameter.

If the X application is installed on the device, it will open showing the message to be posted; otherwise, the device browser will open on the X website, showing the tweet to be posted.



Remember:

- To invoke an API that posts a Tweet, you must use the Twitter API that is located in the GeneXus.Social module, passing the tweet text as a parameter.
- If the X application is installed on the emulator, the application will be opened; otherwise, the browser will open on the X website. These actions may require you to log in to the platform to continue.

PANEL WITH OFFLINE OPERATION

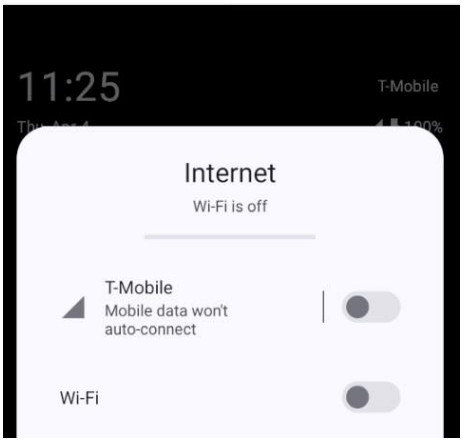
It should also be possible to use the whole application in offline mode.

Therefore, when opening it with an internet connection, the application will connect to the server and will be synchronized with the database. However, if it is opened without an internet connection, we still want it to be able to browse through the different screens.

Tips:

- To meet the requirement, you must set the Connectivity Support property of the MenuMain object to Offline.

After performing the corresponding procedure, to test that the application is indeed working offline, disconnect the application from the internet (turn off Wi-Fi and mobile data).



Open the AmusementPark WorkWith and add a new amusement park.

Confirm that this change has been made only in the device database. To do so, in the web interface, access WWAmusementPark through the Launchpad (View -> Other Tool Windows -> Launchpad) and check that the record you have just entered isn't displayed yet.

Next, reconnect the mobile device to the internet, close and open the application to synchronize the data with the server, and then go to the WWAmusementPark web interface again through the Launchpad. The new amusement park you entered while offline should be displayed now.

ADDING SECURITY TO THE APPLICATION

Apply GAM to the application, so that when executing any main object, it asks for an authentication login. In this case, we will test it by accessing the MenuMain menu object.

Remember:

- To apply GAM, you must set the Enable Integrated Security property to True from the KB version node in the KB explorer.
- To log in on the screen generated by the GAM, the mobile device should be connected to the internet, even if the app is still offline.

In order to implement the following, we must have an Online application again. Therefore, you will have to reconfigure the application in that mode.

Tips:

- To meet the requirement, you must set the Connectivity Support property of the MenuMain object to Online.

You need to create a user (with the name and password of your choice) from the GAM web back office. After login, this user should only be allowed to access the WorkWithAmusementPark object.

Run the application again and log in with this new user to check that you can only access the WorkWithAmusementPark object, and not the rest of the objects added to the MenuMain object.

Tips:

- To create a new login user, in the KB explorer go to the GAM_Examples folder and then run the GAMHome object.
- In the Knowledge Base node, you must change the Integrated Security property to the Authorization value, so that in addition to authenticating it also checks permissions.

www.genexus.com