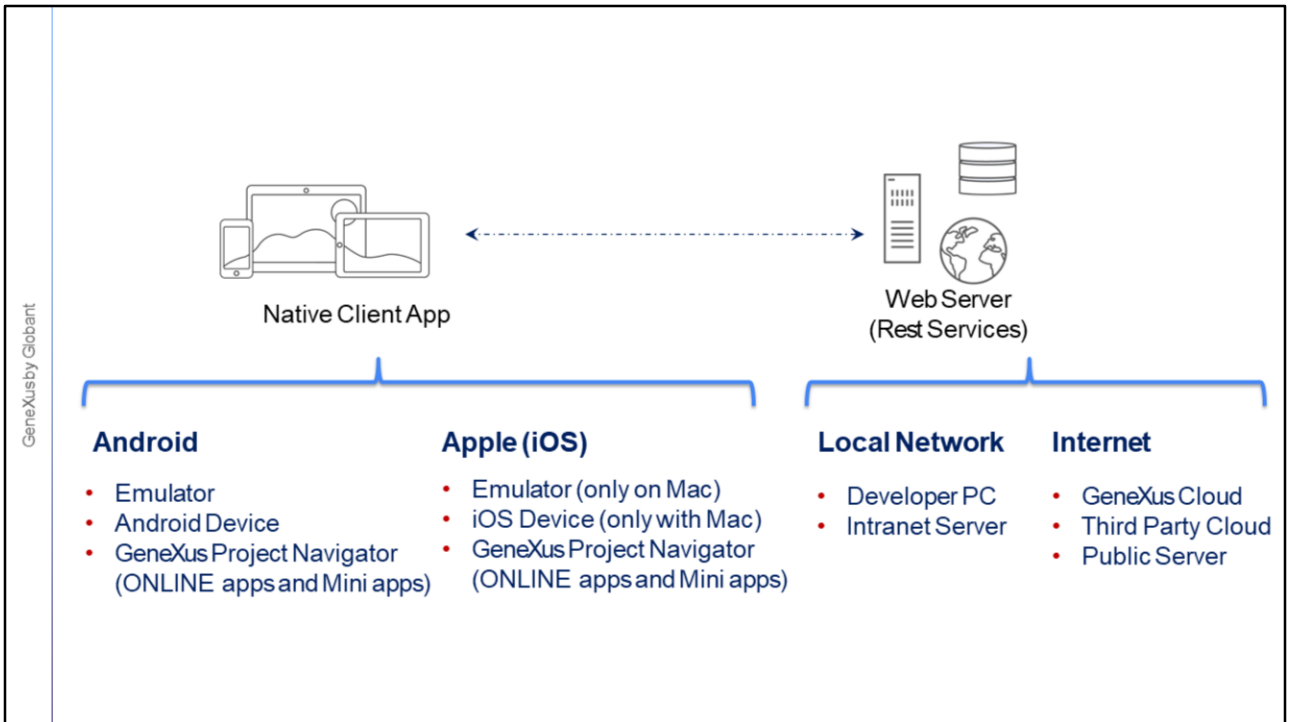


Prototyping a native mobile app



Rodolfo Roballo

In this video, we will see how we can test our Android or Apple application on an emulator, on a physical device, or using the GeneXus Project Navigator app. In addition, we will learn about the advantages and disadvantages of each option.

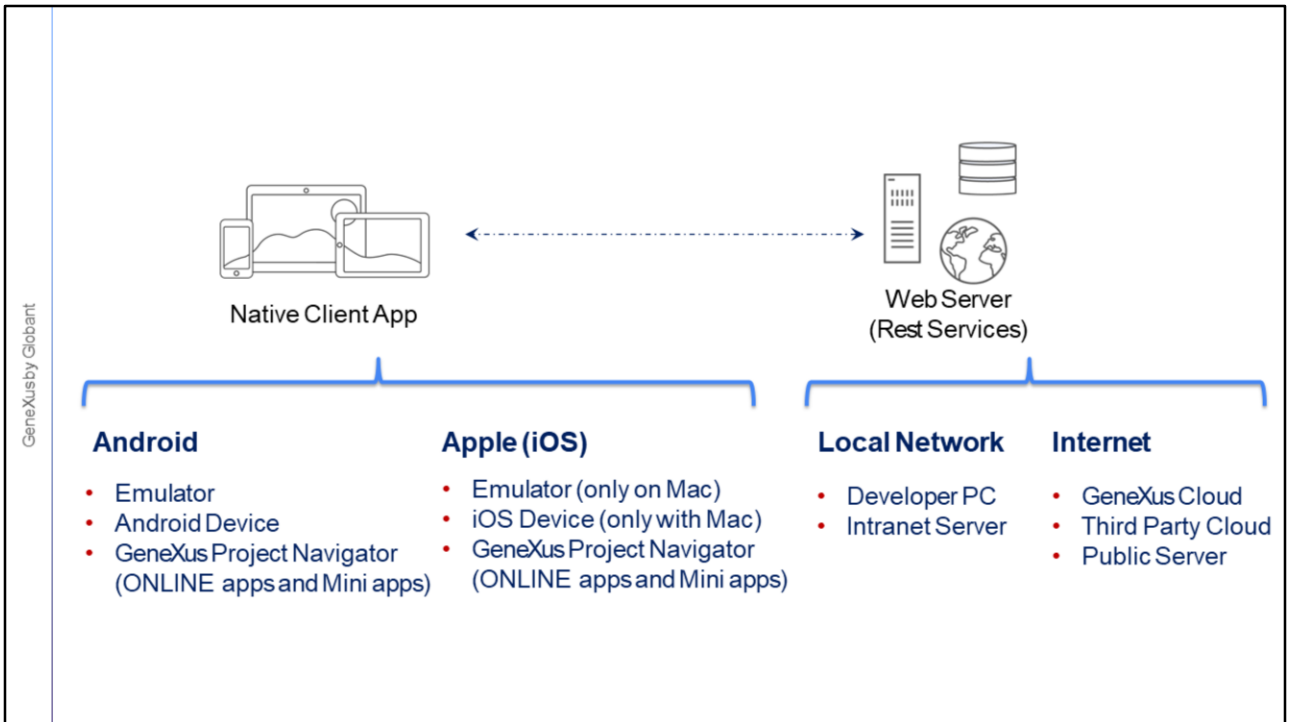


According to what we discussed when we studied the architecture of a native application, to run an application we need two components, the native mobile application itself and the REST services that handle the application's requests, hosted on a web server.

In the case of Android, the client application can run on an emulator as we have been using in the course, or we can run it on a standard physical Android device, or a Huawei Android device, connected to our Windows computer through a USB port. In both cases, the application is compiled before being executed and the .apk file is installed on the emulator or on the physical device through the USB connection.

In addition, we can also run que application with no need to compile it, using the GeneXus Project Navigator. The GPN is a native application created by GeneXus that you can download from Google Play, install it on a standard Android device and use it to prototype your application in an interpreted form with no need for the device to be physically connected to your development computer. This method is not used for Huawei because for the app to access the platform's own services, such as Huawei Analytics, Maps, Notifications, etc., a key is required and it can only be obtained with the compiled application.

Another restriction is that with the GPN we can only prototype applications ONLINE. The good news is that we can prototype Mini Apps. We will talk more about the GPN in a few moments.



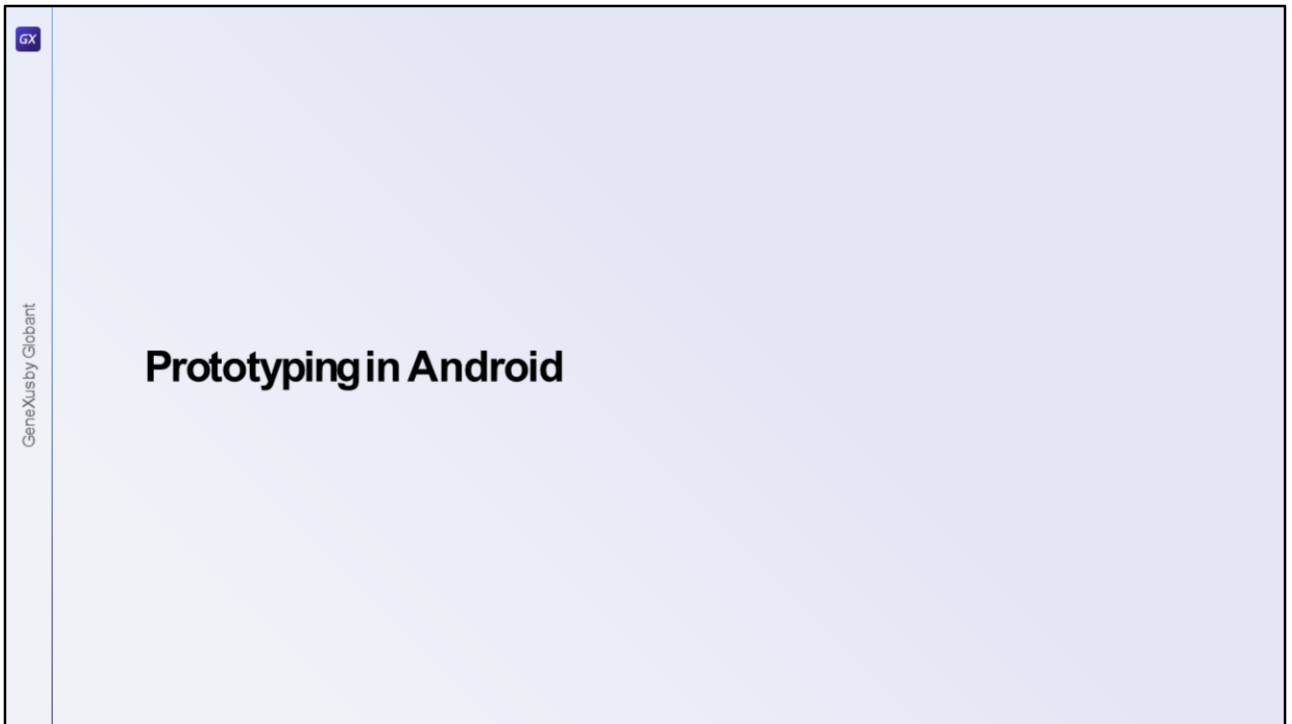
For Apple, the application can also run on an emulator or on a physical device, but we need a Mac computer to compile and to run the emulator or to connect a physical device via USB. In this case, we can also run the application in interpreted form on an iOS device using the GPN –as long as the application is Online or a Mini App– and we don't need a Mac, just an iPhone or iPad with the GPN installed from the Apple Store.

Regarding the services, they can run on the local network, either on the developer's computer or on some other web server available on the network; or they can run on a server on the Internet.

In this case, the server can be the GeneXus cloud server as we are using in the course, and this is the only option when using the trial version. If we are using GeneXus Full, the REST services can be hosted in a third-party cloud such as the one provided by Amazon or Google, or we can publish the services in our own server with public access.

What matters is that the mobile client application has access to these services; i.e. that it can connect via wifi to the server where the services are installed.

Next we will see all the options available and the issues that we will have to consider in each case.



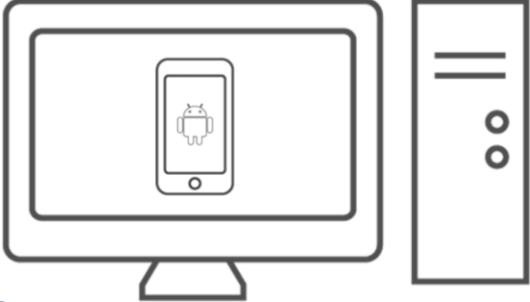
So far, we have run the native application on Android using the emulator.

Let's look a bit closer at the prototyping mechanisms on this platform.

GeneXusby Globant

Prototyping in Android using an Emulator

Generator: Frontend (Front end)	
Name	Frontend
Main Platform	Android
Dynamic Services URL	False
Services URL	https://trialapps3.genex...
SSL Pinning Pin Set	
Smart Devices Cache Management	On
Visual Testing Repository URL	
Generators	
Generate Android	True
Generate Huawei	False
Generate Apple	True
Generate Angular	False
Android Specific	
Android Execution Type	Emulator or Device
Android SDK directory	C:\Android-SDK
JDK Directory	C:\Program Files\Java\j...



Services URL

- Local Network
 - Wifi or LAN assigned IP Address
 - For Localhost use IP 10.0.2.2
- Internet Server (any)

To generate for Android, as we've seen in previous videos, we only need to meet the necessary requirements in order to run the application on this platform. Also, check that the Generate Android property is set to True, which is its default value, and that the Main Platform property is set to Android.

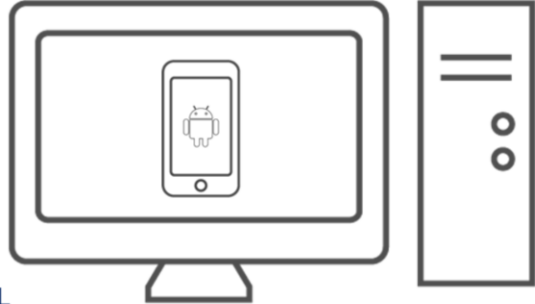
As we've said before, when running GeneXus, an Android emulator is opened to test the application. When using the emulator, the services can be hosted in any of the options we saw, but we must verify that the emulator has access to that server.

The services URL is set using the Services URL property of the Front end node.

In our case, this property is disabled because the Trial version doesn't allow running on any server other than the GeneXus cloud server. On the other hand, if we use the GeneXus Full version, in this property we can enter the corresponding URL.

Prototyping in Android using an Emulator

Generator: Frontend (Front end)	
Name	Frontend
Main Platform	Android
Dynamic Services URL	False
Services URL	https://trialapps3.genex...
SSL Pinning Pin Set	
Smart Devices Cache Management	On
Visual Testing Repository URL	
Generators	
Generate Android	True
Generate Huawei	False
Generate Apple	True
Generate Angular	False
Android Specific	
Android Execution Type	Emulator or Device
Android SDK directory	C:\Android-SDK
JDK Directory	C:\Program Files\Java\j...



Services URL

- Local Network
 - Wifi or LAN assigned IP Address
 - For Localhost use IP 10.0.2.2
- Internet Server (any)

When the host name is used as part of the URL, the server must be accessible from the emulator (i.e. it must be a public server) because the emulators can't resolve Localhost or any name of the internal network, so in those cases we must use the IP. For example, if we run the server on the development PC, we can use the IP 10.0.2.2 for Localhost, which will be accessible from the emulator.

A special case is when we need our application to connect to different servers. In this case, in the Services URL property we must indicate a default value or leave it empty and set the Dynamic Services URL property to True. When we run the application for the first time, a panel object built to ask for that value will be opened.

Prototyping in Android using a Physical Device

GeneXusby Globant

Generator: Frontend (Front end)	
Name	Frontend
Main Platform	Android
Dynamic Services URL	False
Services URL	https://trialapps3.genex.
SSL Pinning Pin Set	
Smart Devices Cache Management	On
Visual Testing Repository URL	
Generators	
Generate Android	True
Generate Huawei	False
Generate Apple	True
Generate Angular	False
Android Specific	
Android Execution Type	Emulator or Device
Android SDK directory	C:\Android-SDK
JDK Directory	C:\Program Files\Java\j_...

Developer options:

- USB debugging ON
- Install from any source allowed

• Device drivers installed

To run on a physical Android device (and from now on we include Huawei as a particular case of Android), we must connect the device to our development computer via USB cable.

When GeneXus detects that an Android device is connected to the development computer via USB, the emulator does not open and the application is installed and run on the device.

For the development computer to correctly detect that the device is connected, the corresponding drivers must be installed, the USB Debug option must be enabled on the device, and applications must be allowed to be installed from any source.

As we have already said, for the application to work correctly, the device must have access to the services. Since it is a physical device, using Wifi to connect the device to the same network where the services are running is the most frequent solution. If these services run on a server that can be accessed from the Internet, in addition to Wifi we can use a cellular data connection (such as 3G, LTE, etc.).

Prototyping in Android using a Physical Device

GeneXusby Globant

Generator: Frontend (Front end)	
Name	Frontend
Main Platform	Android
Dynamic Services URL	False
Services URL	https://trialapps3.genex...
SSL Pinning Pin Set	
Smart Devices Cache Management	On
Visual Testing Repository URL	
Generators	
Generate Android	True
Generate Huawei	False
Generate Apple	True
Generate Angular	False
Android Specific	
Android Execution Type	Emulator or Device
Android SDK directory	C:\Android-SDK
JDK Directory	C:\Program Files\Java\j...

Developer options:

- USB debugging ON
- Install from any source allowed

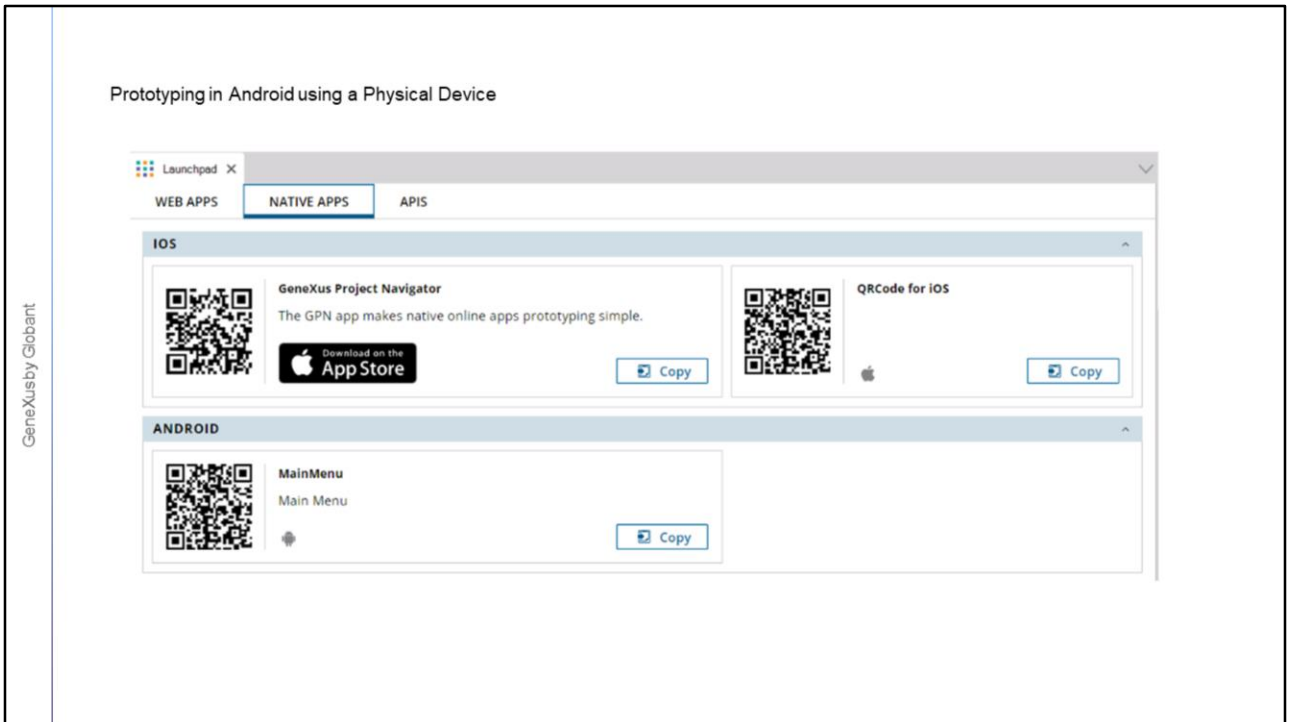
• Device drivers installed

If the server is on a local network, the server must be accessible by the device via Wifi with the necessary permissions.

If the server is localhost, that is, the web server is installed on the development computer to which the device is connected via USB, this wired connection does not establish a network connection for the device to access the localhost server. For this connection to be established, the cell phone and the development computer must be on the same network. This will only work if the computer allows remote connections and also verifies that the firewall doesn't block the connection.

If the server is private and remote, the network where the server is located may be accessed through a VPN and the local development computer may be configured as a proxy, but ultimately the device must be on the same network as the host.

In short, although it is possible to prototype locally and have the physical device access the web server, it is more convenient to prototype in the GeneXus cloud because regardless of where the device is located, as long as it has an Internet connection it will be able to reach the REST services of the application's backend.

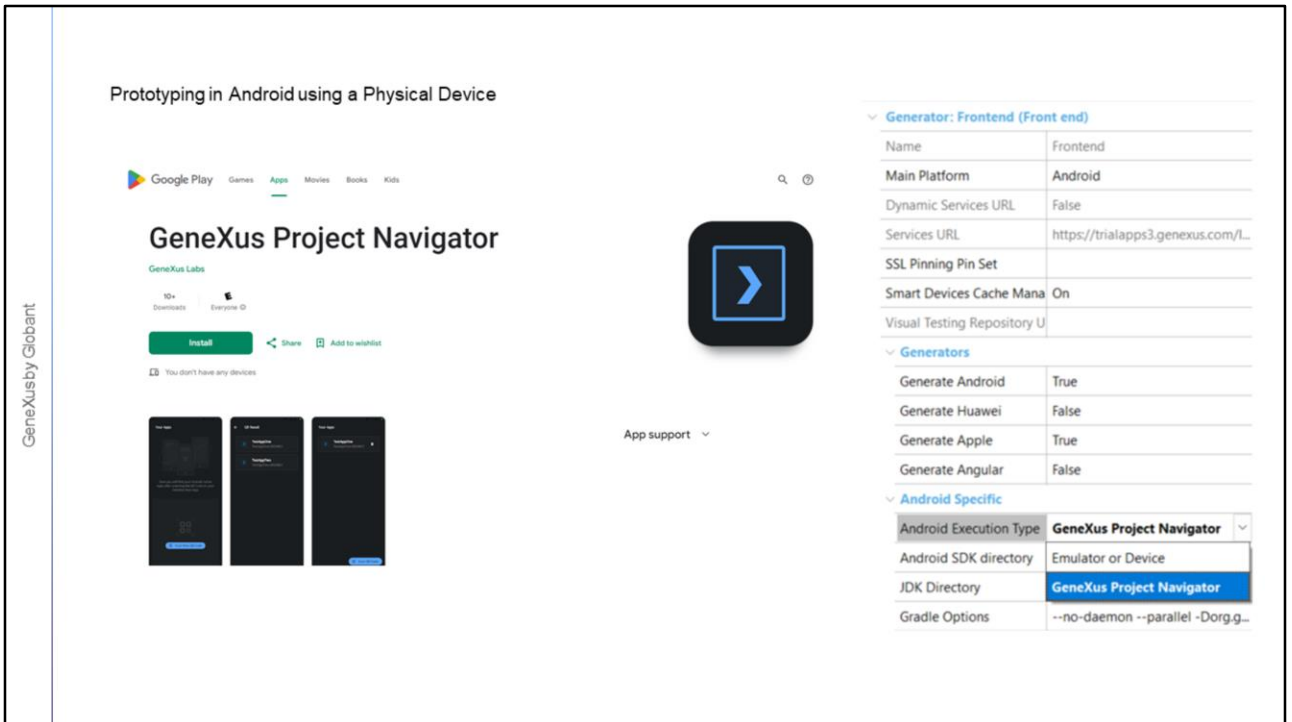


Instead of connecting the device via cable, another option to run on a physical device is to use QR codes.

To do so, we use the device to scan the QR code of the compiled app and install it. The device must have access to this URL, which is the same URL where the services run.

To access the QR codes, we must show the Launchpad window and select the Native Apps tab. The Launchpad is displayed if we run the Developer Menu or if we select View/Other Tool Windows/Launchpad. There we will find an IOS section and an ANDROID section. In the Android section, we will find the QR code to scan.

As in the previous case, the option to install from other sources must be enabled on the device.



A third option for prototyping in Android is to use the GeneXus Project Navigator app, which is available in the Google Play Store, and install it on the device.

To run our application in the GPN, we first add the project by scanning the QR code or by manually entering the URL of the Services URL property.

The application will run interpreted, because when we press F5 GeneXus doesn't compile the application (that is, it doesn't generate the apk). Instead, it updates the project's metadata on the server, which is automatically refreshed by the GPN, showing our changes on the device. This shortens development times because it reduces the time between making changes and seeing them running.

However, when prototyping with the GPN we have some limitations, which include the need for the application to be compiled –for example, when we want to develop offline applications, use user controls or external objects, use push notifications or access certain resources such as embedded fonts, images with various resolutions and animations, among others.



As we've said before, with GeneXus we can prototype on an emulator and on a physical device. Although prototyping on the emulator has the advantage of avoiding the need for a device, there are also several disadvantages.

First, with the emulator we don't have access to resources –such as the calendar, camera, GPS, or third-party applications– that we have on a physical device.

We also lose performance, because emulators are usually slower than devices when loading and running an application.

Another major disadvantage is that what you see on the emulator is not exactly what you see on a physical device. Maybe the differences are small, but it is very important to test the final application on a real device.

Finally, it is worth mentioning that running the application on the emulator doesn't "feel" the same, since you can't really experience touching the screen, swiping, making gestures, etc.

However, although it may seem that there is no reason to use an emulator, it is the fastest way to have an application running without using anything other than your computer.



Now let's see the different ways to prototype an application generated for iOS.

Prototyping in iOS

GeneXusby Cllobant

Generator: Frontend (Front end)	
Name	Frontend
Main Platform	Apple
Dynamic Services URL	False
Services URL	https://trialapps3.g
SSL Pinning Pin Set	
Smart Devices Cache Mana	On
Visual Testing Repository U	
Generators	
Generate Android	False
Generate Apple	True
Generate Angular	False
Apple Specific	

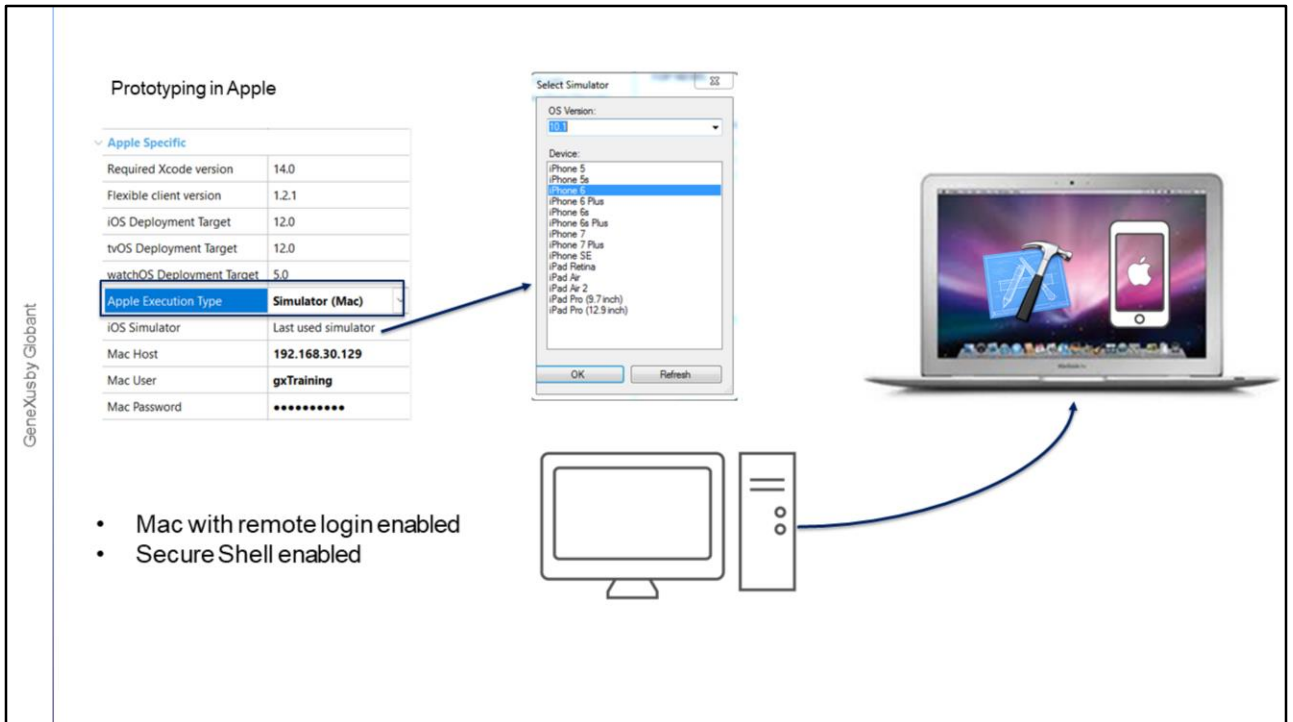
Apple Specific	
Required Xcode version	14.0
Flexible client version	1.2.1
iOS Deployment Target	12.0
tvOS Deployment Target	12.0
watchOS Deployment Ta	5.0
Apple Execution Type	GeneXus Project Navigator (...)
Execution Device	Simulator (Mac)
Mac Host	iOS Device (Mac)
Mac User	iTunes Sync (Local)
Mac Password	Build IPA (Local)
	GeneXus Project Navigator (Device)

Let's check the properties of the Front end node because we need to set the generation platform to Apple.

The Generate Apple property is set to True by default, but the Generate Android property is also set to True and the Main Platform property is set to Android by default.

The first thing we must do is set the Main Platform property to Apple. If we want, we can continue generating for Android, but if we are only going to develop an iOS app then it is preferable to set the Generate Android property to False in order to save generation time.

Next, to run the application, in the Apple Specific group we must configure some options that we will see next.



The Apple Execution Type property offers several prototyping options.

The first option of the combo box shows the Simulator (Mac) value, which lets you prototype on an Apple device emulator running on a Mac.

Of course, this requires a computer of this type, but it doesn't have to be the development computer that is running GeneXus; it can be a remote computer.

To configure the access, in the properties of the iOS Specific group, we need to set values for Mac Host, Mac User and Mac Password.

To this end, the Mac must be in the same network as the machine where GeneXus is installed. When running from GeneXus, the source code is transferred to the Mac where it is compiled; next, the emulator is opened and the compiled app is installed in it.

The iOS Simulator property allows selecting the emulators available on the Mac.

The Mac computer must support remote login and have SSH (Secure Shell) enabled.

Regarding Rest services, we must have access to these services from the Mac, which means that the web server where the services are located must be on the same network as the Mac computer.

GeneXusby Globant

Prototyping in Apple

Apple Specific	
Required Xcode version	14.0
Flexible client version	1.2.1
iOS Deployment Target	12.0
tvOS Deployment Target	12.0
watchOS Deployment Target	5.0
Apple Execution Type	iOS Device (Mac)
Mac Host	192.168.30.129
Mac User	gxTraining
Mac Password	*****

- iOS developer
- Mac Certificate
- AppID for each app
- Assign Development Team Id property in main object

The second option of the run type combo box has the value iOS Devices (Mac) that allows us to use a physical Apple device.

In this case, we need a Mac computer and an Apple device such as iPhone, iPad, Apple TV, or Apple Watch connected to the Mac via USB cable.

When using this option, we must complete the Mac Host, Mac User, and Mac Password properties to access the Mac computer, which must be on the same network as the development computer.

When running from GeneXus the application is generated and the sources are transferred to the Mac. At the end of the transfer, the application is compiled on the Mac and the .ipa extension package is transferred and installed on the Apple device connected to the Mac.

GeneXusby Globant

Prototyping in Apple

Apple Specific	
Required Xcode version	14.0
Flexible client version	1.2.1
iOS Deployment Target	12.0
tvOS Deployment Target	12.0
watchOS Deployment Target	5.0
Apple Execution Type	iOS Device (Mac)
Mac Host	192.168.30.129
Mac User	gxTraining
Mac Password	*****

The diagram illustrates the prototyping setup. A smartphone is connected to a laptop. The laptop screen shows a blueprint with a hammer icon, symbolizing development or prototyping. A server rack is also connected to the laptop, indicating a networked development environment.

- iOS developer
- Mac Certificate
- AppID for each app
- Assign Development Team Id property in main object

This prototyping option is the fastest and most realistic, because this app will be identical to the app that will be available for download from the Apple Store.

However, in order to compile the application, the developer must be registered as an iOS Developer on Apple's site, the Mac must have a development certificate installed and the device where it is to be tested must be registered within the certificate.

It is also necessary to create an App ID for each application we want to test and create a provisioning profile where all this information will be linked.

In addition, it is necessary to obtain the value of the Team-ID field of the Apple developer account and assign that value in the Development Team Id property of the main object of the application, so that it is included in the generated Xcode project.

GeneXusby Globant

GeneXus™ by Globant Recent pages Statistics 🔍 [Sign in](#) [Login](#)

Native Mobile Applications Development
Table of contents

Page Tools ▾ Page Info ▾ Also seen in ▾ Other document versions ▾

Prototyping in iOS with a compiled application

GeneXus™ by Globant Recent pages Statistics 🔍 [Sign in](#) [Login](#)

Native Mobile Applications Development
Table of contents

Page Tools ▾ Page Info ▾ Also seen in ▾

HowTo: Prototye an iOS Application on a Mac

This documentation is valid for: [GeneXus 18 Help](#) [GeneXus 17 Help](#) [GeneXus 16 Help](#) [GeneXus 15 Help](#) [GeneXus X Evolution 3 Help](#) [GeneXus X Evolution 2 Help](#)


The purpose of this article is to explain the necessary steps to prototype your iPhone or iPad application on your Mac.

Step 1: Install requirements

Check Mac requirements on [Apple Requirements](#)

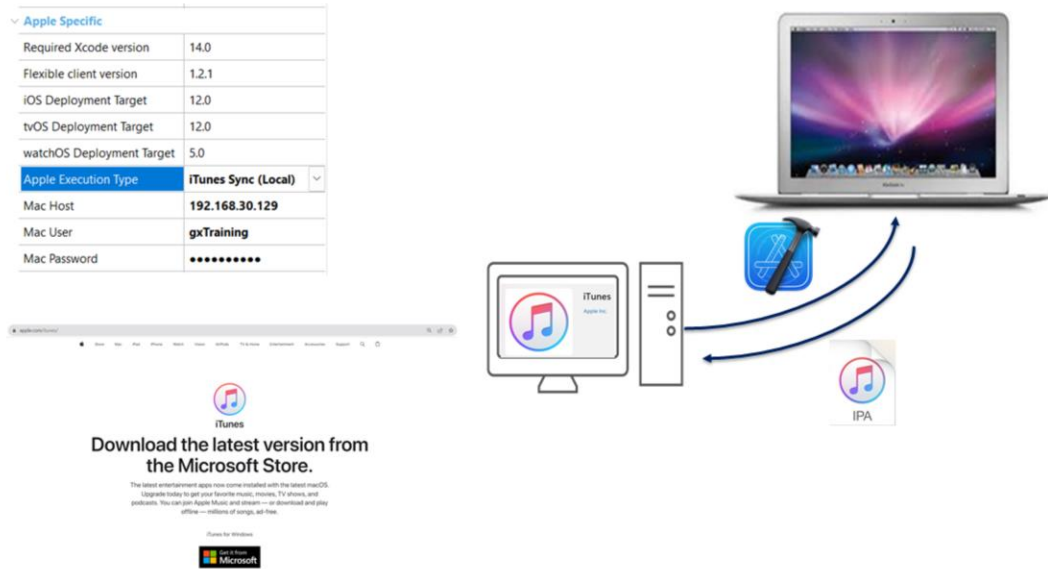
Step 2: Enable ssh

Go to System preferences.



- Introduction
 - Native Mobile Generator
 - Welcome to Native Mobile Applications Development
 - My first Android application
 - My first iOS application
 - My first Offline Native Mobile application
- Requirements and Installation
- Architecture
- Tabs offered in Panel and Work With objects
- Layouts
 - Orders and Filters in Grids of Panels
 - Transaction Rules in Native Mobile

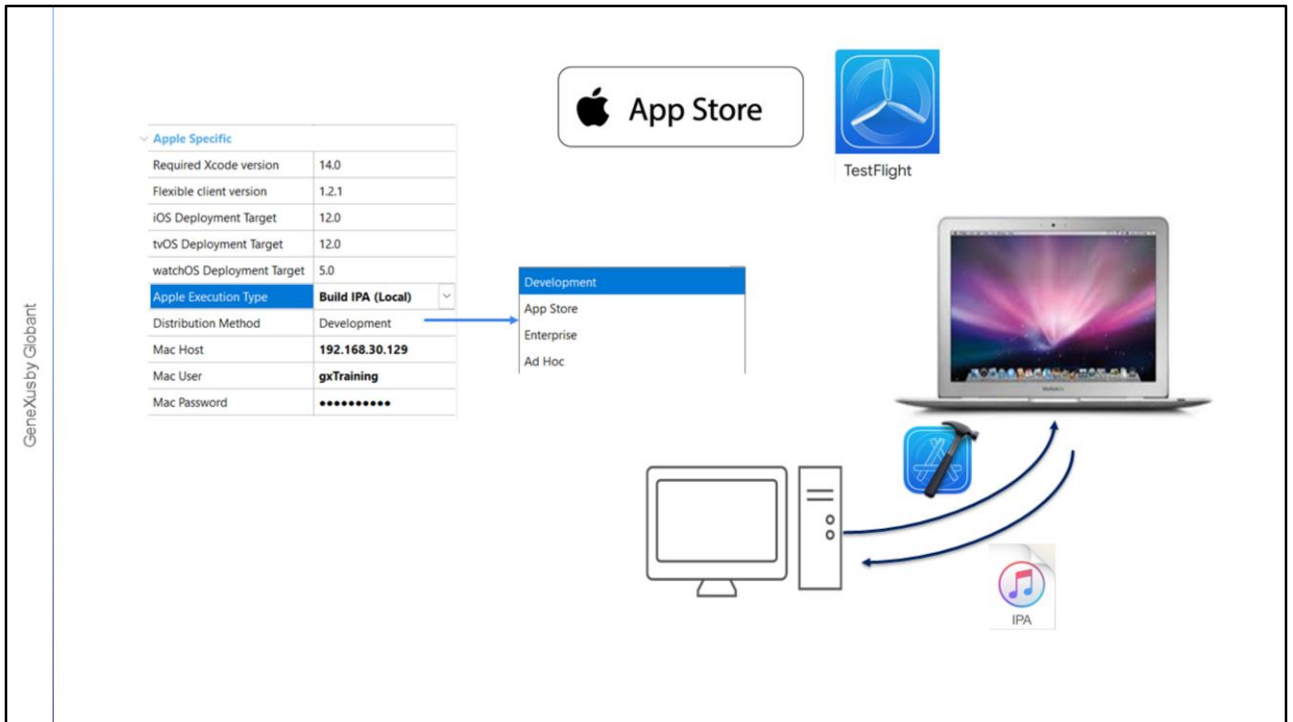
For more details on how to follow these steps, we suggest referring to the wiki.



By selecting the iTunes Sync (Local) option, the XCode project is transferred to a Mac and there the application is compiled, generating an *.ipa file.

Then the process copies the .ipa file from the Mac to the development computer and leaves it in the folder specified in the Output window of the GeneXus IDE.

Lastly, iTunes is automatically opened on the development computer to display the application in the application catalog.



With the Build IPA (local) option, the XCode project is transferred to a Mac and the application is compiled using the method specified in the Distribution Method property, generating an *.ipa file. The file is then copied to the folder associated with the KB environment.

If the Distribution Method property is set with the Development value, the application is compiled and signed with the development provisioning profile to be sent to the Test computer.

If App Store is selected, the application is compiled and signed with the development provisioning profile to be published in TestFlight and Apple Store.

TestFlight is an Apple application that allows application developers to invite internal or external testers to test their applications and, in this way, gather feedback and data related to their project's functionality.

If Enterprise is selected, the application is compiled and signed for internal distribution.

And if we choose Ad Hoc, the application is compiled and signed with a provisioning profile created for a specific case, and then distributed to the application testers.

GeneXusby Globant

Prototyping in Apple with GPN

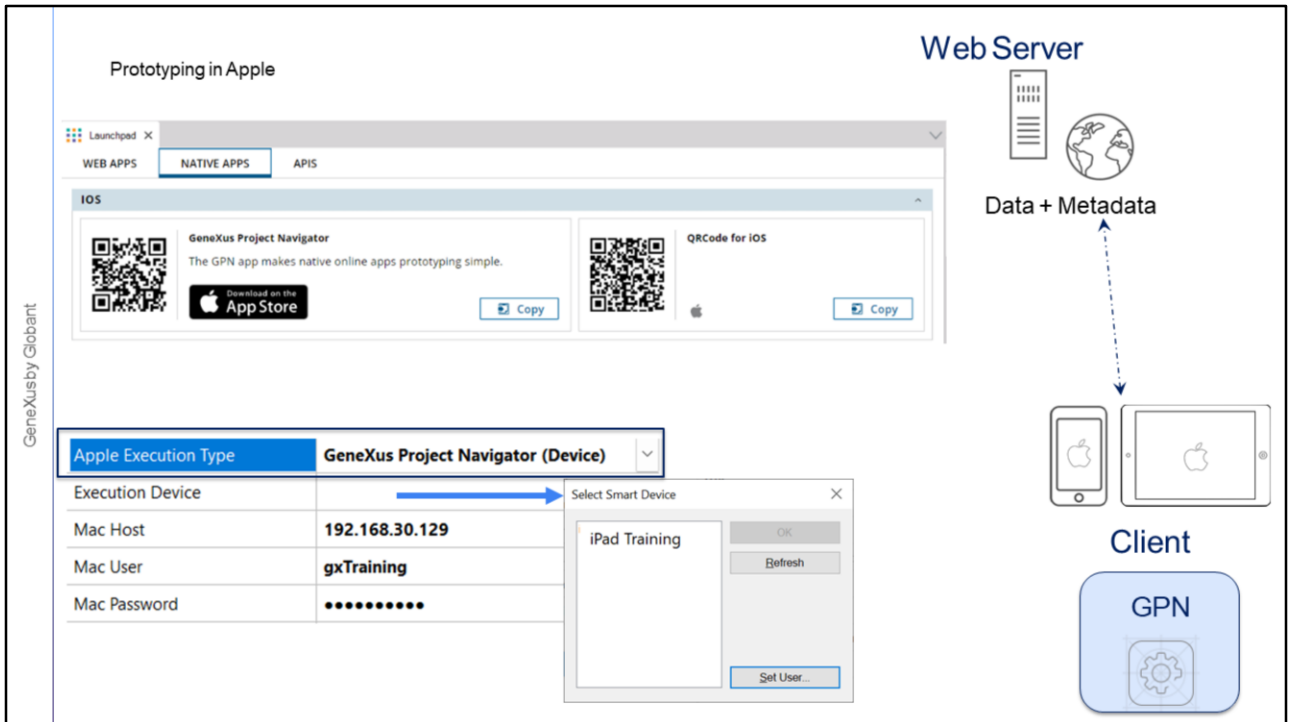
Apple Specific	
Required Xcode version	14.0
Flexible client version	1.2.1
iOS Deployment Target	12.0
tvOS Deployment Target	12.0
watchOS Deployment Target	5.0
Apple Execution Type	GeneXus Project Navigator (Device) ▾
Execution Device	
Mac Host	192.168.30.129
Mac User	gxTraining
Mac Password	*****

The last option in the combo box, the default option, reads GeneXus Project Navigator (Device).

This allows us to prototype on a physical device using the GeneXus Project Navigator, which must be downloaded from the Apple Store.

In this case, we don't need a Mac computer to compile the application and its behavior is similar to what we have already seen for Android.

The limitations of the GPN for prototyping in the Apple platform are the same as mentioned above.



The link to download the GPN from the Apple Store is available in the Launchpad. There is also the QR code of the application URL, which we will use from the GPN installed on the device to register the application.

When we use the GeneXus Project Navigator (Device) option, we must assign the device where we are going to prototype. In the Execution Device property, we will be able to choose one of the devices associated with our GeneXus Account. To do so, first we must open the Execution Device property dialog box, press the Set User button, log in to our GeneXus account, and choose the device from the list.

Once we've catalogued the application in the GPN (by entering the URL manually or through the QR code), every time we execute from GeneXus, the application is automatically updated with the changes made in GeneXus thanks to a Push Notification received from the server.

This way of prototyping has the advantage that we see the application on a real Apple device, with no need to have a Mac or to register as a developer on the Apple site.



In this video, we saw the flexibility we have to prototype native applications on both Android and Apple.

In the rest of the course, we will use the Android emulator to prototype our application.