

GeneXus Applications Architecture



As we saw at the beginning of the course, GeneXus allows us to create different types of applications, on different platforms.


We will now look at their architecture and how it relates to their logic; that is, how they are developed.

Web Applications with Back-office Focus



RecentsCountries — Country

Country

Id	6
Name	<input type="text" value="Argentina"/>
City Id	1
Flag	<input type="button" value="Change"/> 
City	
Id Name	
x	1: Buenos Aires

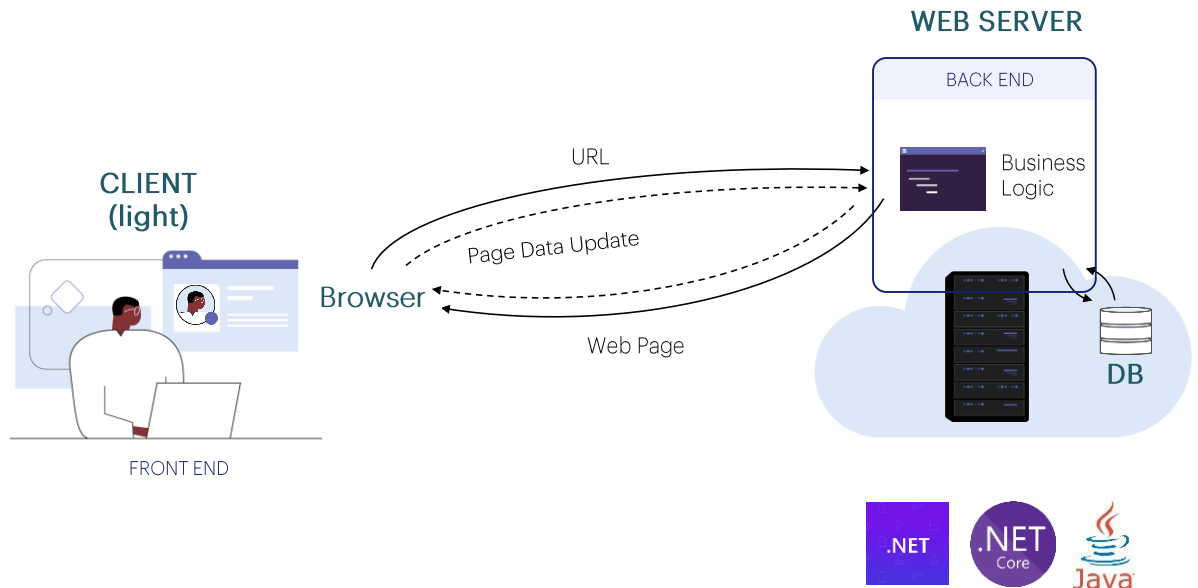
Id	Name	UPDATE	DELETE
6	Argentina	UPDATE	DELETE
1	Brazil	UPDATE	DELETE
9	Chile	UPDATE	DELETE
3	China	UPDATE	DELETE
2	France	UPDATE	DELETE
7	Jamaica	UPDATE	DELETE
5	Japan	UPDATE	DELETE
10	Kingdom of Saudi Arabia	UPDATE	DELETE
11	Mexico	UPDATE	DELETE
8	Uruguay	UPDATE	DELETE

« < > »

So far, we have seen how transaction screens work; where data is validated when you exit each field and you receive a message, or that when entering a foreign key value the name corresponding to that key is inferred. This immediate response received by the user from the application is called Client Side Validation.

Also, we saw that in screens generated with the Work With pattern (web panel objects that we'll discuss later) the information was dynamically loaded when we changed the page of a grid or when we applied a filter, and that its size was automatically adjusted in a responsive way.

Architecture of Web Applications with Back-office Focus



Some of these operations are performed on the client side of the web application (i.e. the browser) and others on the server side; for example, when it is necessary to search in the database.

The application logic is on the server side, so when we run an object in our application, the browser requests the corresponding page from the server. The server prepares the information, obtains data from the database if necessary, builds the screen, and sends it to the browser for display.

In some cases, once the page is being viewed in the browser, we start an action such as when applying a filter to a query (in a grid, for example). As a consequence, the code on the client side communicates with the code on the server to return the data that meets the filter, and can then refresh only the part of the screen involved in the action (in this example, the grid). In turn, if we sort a grid column or if in a transaction we exit a field that causes a message to be displayed, it is resolved by the client alone, with no need to access the server.

In this architecture the client has some intelligence, but the decisions are made on the server because that is where the complete logic of the application resides. What we have developed so far is basically the application back office, as we have seen how to manipulate the database (transactions) and how to query its information in a more hierarchical way, to achieve those additions, deletions and modifications (that is, everything built by the Work with pattern). However, with this same architecture we can also build customer-facing applications.

In GeneXus, to build these applications we use .Net, .Net Core or Java. GeneXus uses these languages for both client (front-end) and server (back-end) code.

High-performance applications with more interactive content

12°
CLOUDY
FRANCE

HUMIDITY 64% WIND 12 K/M

Twitter Feed #France

- Don't Forget your sunscreen tomorrow!
- Amazing weather in Paris!

Weather Forecast:

Day	Temp	Condition
MON	9°	RAINING
TUE	15°	SUNNY
WED	11°	CLOUDY
THU	7°	STORM
FRI	18°	SUNNY

Customer Details:

Supplier	Contact No	Instrument
Supplier: Norman Edwards normanedwards@gmail.com +44 657 4112	020-837-3578	Baby Crib
Supplier: Samuel Milson SamuelMilson@gmail.com +44 657 4112	020-837-3578	Obstetric Tables

TIENDA inglesa

Buscar

INGRESAR

- Inicio
- Categorías
- La Huerta
- Mailing
- Ocasiones
- Bienestar
- Gift Card
- Sucursales
- Mis Puntos
- Catálogo
- Contacto

Buscar

ESPECIAL COSMÉTICA, PERFUMERÍA Y LIMPIEZA
DESDE EL 12 AL 26 DE AGOSTO

Paga Online con

Scotiabank
18 cuotas sin recargo

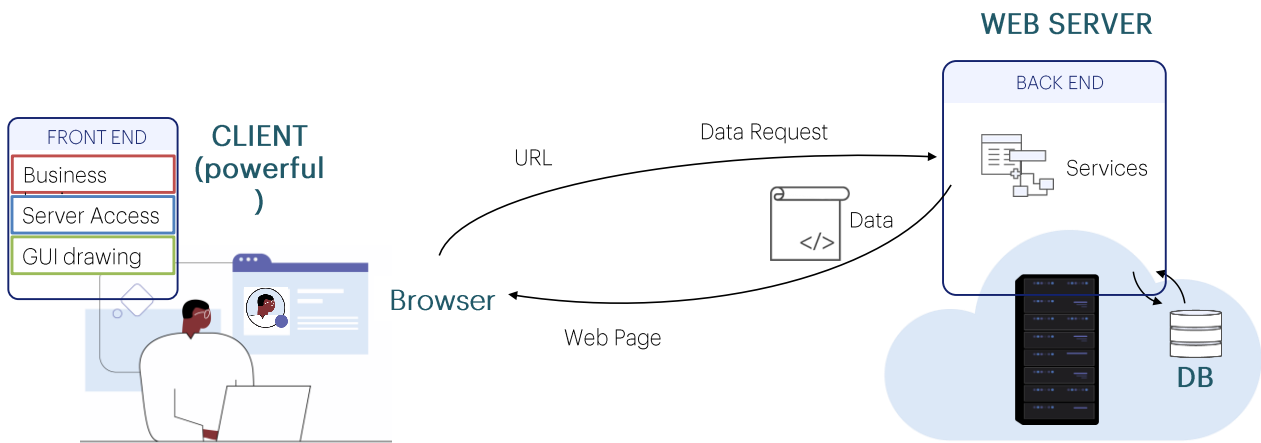
Productos Destacados

- Afeitadora philips s1030/04 (2x\$1500) - U\$S 65
- Pack 2 cervezas erdinger (2x\$400) - \$ 339
- Depilador 16 Imágenes (1x\$139) - U\$S 139
- Cerveza cabezas pack las 3 (3x\$349) - \$ 349

There are other types of applications that can be web or native for mobile devices, which are very powerful to display information fast and with a special focus on design and interactivity.

They are customer-facing applications, i.e. user-centered, with rich on-screen information, that load parts of the page only when needed because they are designed for extremely high performance with the best possible user experience.

Architecture of Web Applications with UX Focus



This architecture also runs one part on the client and one part on the server, but in this case most of the application logic is on the client. In the client there are 3 distinct layers: business logic, communications with the server, and the part that takes care of drawing the screen in the browser.

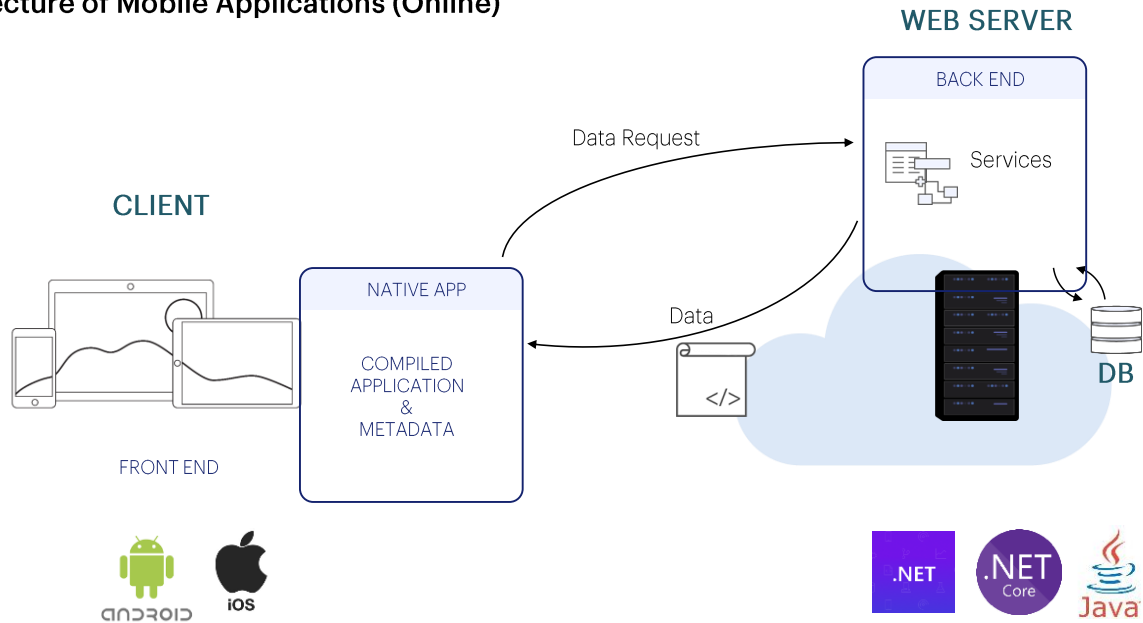
With this scheme, most of the functions are performed on the client and the server is only accessed to request data from the database or modify it and other resources provided by services.

This type of customer-facing applications that require a powerful client in order to deliver a high level of user experience and interactivity are more recent than the responsive applications we saw earlier.

To be able to handle these demands, frameworks such as Angular, React, or Vue have emerged in the market, with all the power to build a smarter client as required by these applications.

As we've said before, GeneXus also allows developing customer-facing applications in Java, Net, or .Net Core, but the advantage of programming them in Angular is that the same panel objects designed for the user interface can be reused almost unchanged to generate the native mobile application, i.e. for Android or iOS.

Architecture of Mobile Applications (Online)



So far, we have focused on web applications, and now we'll look at the architecture of a native mobile application.

In particular, we will see the architecture of mobile applications always connected through Wi-Fi (online applications), which is the most common case of native applications, while keeping in mind that GeneXus also builds offline applications.

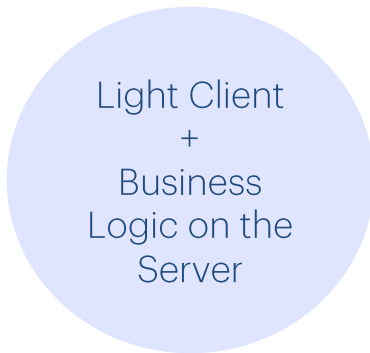
A mobile application also has a part that runs on the client (in this case, the mobile device) and a part that runs on the server, which provides information to the client.

The application code runs on the device, which accesses the server only when it needs data.

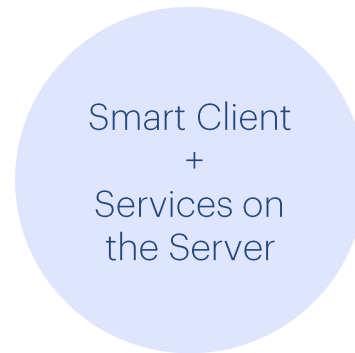
With GeneXus we can generate these applications for Android or iOS devices. The server services (which provide data to the client) can be generated in Java, .Net, or .Net Core.

As we can see, this architecture is very similar to that of high-performance web applications, and for that reason the programming method is very similar, with some differences that we will see later on.

In short, there will be two programming methods.



**Web Applications with
Back-office Focus**



**Web Applications
with UX Focus**

**Native Applications
for Mobile Devices**

The applications' architecture conditions many aspects of the application; in particular, the way they are programmed.

When the client is light, almost all application requirements will be solved on the server, so our programming will always be focused on server operations such as accessing the database.

On the other hand, when we have a powerful client, many operations can be performed on the client itself, without having to resort to the server. However, there are other operations mainly related to database interaction that do require the server programs.

For this reason, we must have a different syntax to indicate when we want certain code to be executed on the client and another on the server. This also applies to the programming of native applications for mobile devices.

Next, we will see how applications of each type are programmed.

GeneXus™

training.genexus.com
wiki.genexus.com
training.genexus.com/certifications