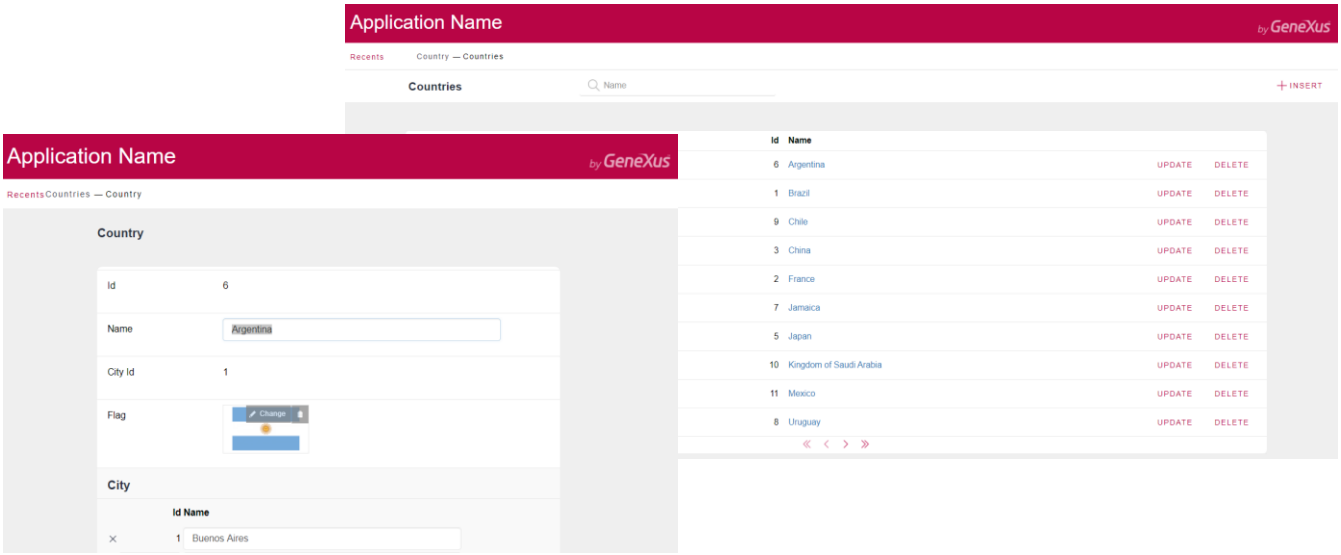
Aplicaciones GeneXus y su arquitectura

GeneXus™

Como vimos al principio del curso, GeneXus nos permite crear distintos tipos de aplicaciones, en distintas plataformas.

A continuación veremos su arquitectura y cómo ésta se relaciona con su lógica, es decir, con la forma de desarrollarlas.

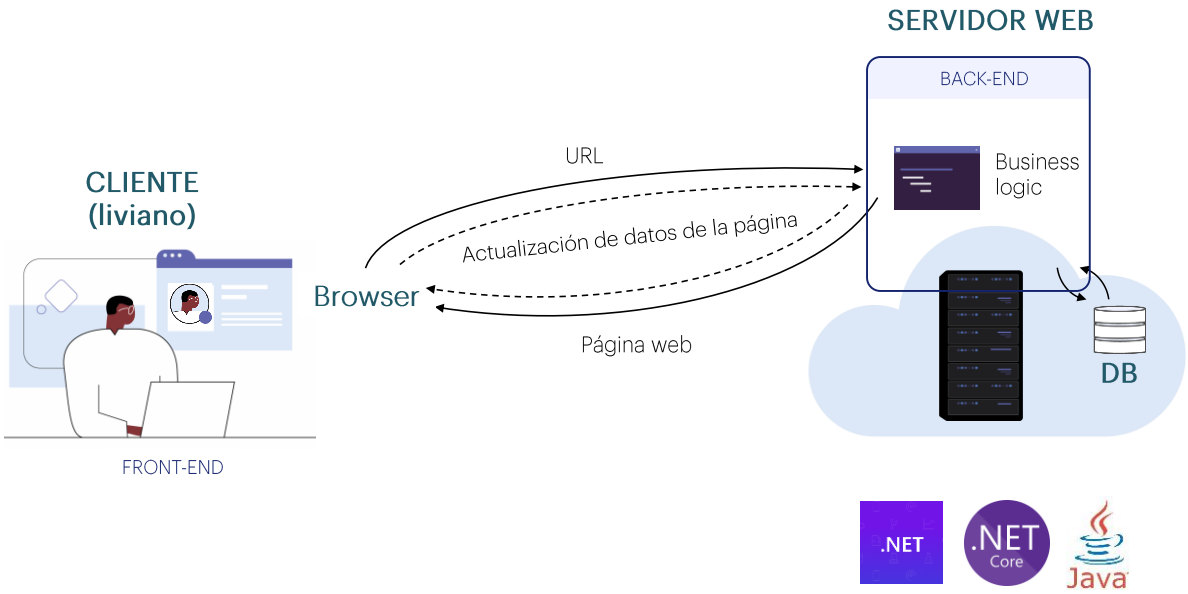
Aplicaciones web con énfasis en back-office



Hasta ahora hemos visto cómo funcionan las pantallas de las transacciones, en las que al salir de cada campo se validan los datos y podemos recibir un mensaje, o que al colocar un valor de una clave foránea se infiere el nombre correspondiente a esa clave. Le llamábamos Client Side Validation a esa respuesta inmediata que el usuario obtenía de parte de la aplicación.

También vimos que en las pantallas generadas con el patrón Work With (los objetos web panels, en los que nos detendremos luego) la información se cargaba dinámicamente cuando cambiábamos de página de un grid o cuando aplicábamos un filtro, o también vimos que se ajustaba su tamaño automáticamente en forma responsiva.

Arquitectura de las aplicaciones web con énfasis en back-office



Algunas de estas operaciones se realiza en la parte cliente de la aplicación web (o sea en el browser) y otra se hace en el servidor, por ejemplo cuando se requiere ir a buscar información a la base de datos.

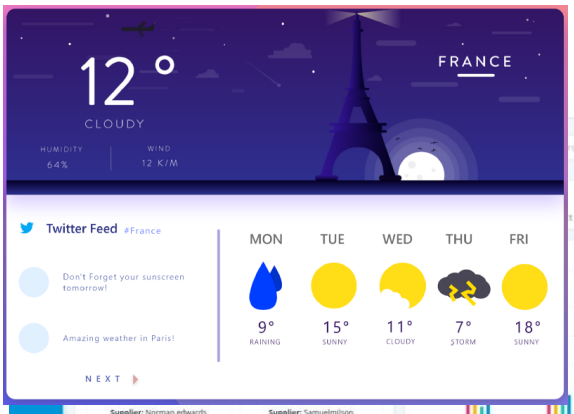
La lógica de la aplicación está en la parte del servidor, así que cuando ejecutamos un objeto en nuestra aplicación, el navegador pide la página correspondiente al servidor. El servidor prepara la información, obtiene datos de la base de datos si es necesario, arma la pantalla y la envía al navegador para que la muestre.

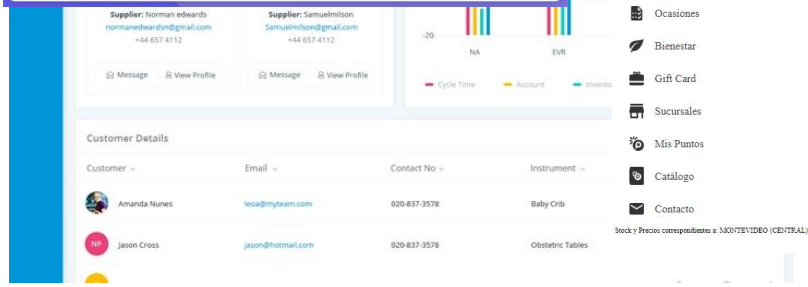
En algunos casos, luego de que la página se está viendo en el navegador, nosotros iniciamos una acción como cuando aplicamos un filtro a una consulta (en una grilla, por ejemplo). Como consecuencia, el código de la parte cliente se comunica con el código en el servidor para que le devuelva los datos que cumplen con el filtro, y pueda luego refrescar solamente la parte de la pantalla involucrada en la acción (en este ejemplo, el grid). A su vez, si ordenamos una columna de una grilla o si en una transacción salimos de un campo que produce que se muestre un mensaje, lo resuelve únicamente el cliente, sin necesidad de acceder al servidor.

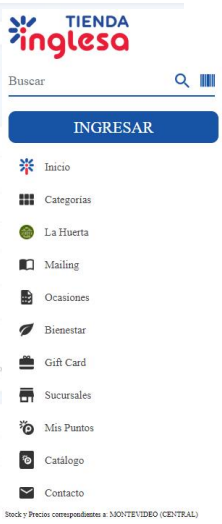
En esta arquitectura el cliente tiene alguna inteligencia, pero es en el servidor donde se toman las decisiones ya que allí reside la lógica completa de la aplicación.
Lo que hemos desarrollado hasta el momento es básicamente el back-office de la aplicación, ya que hemos visto cómo manipular la base de datos (transacciones) y cómo consultar su información de manera más jerarquizada, para lograr esas altas, bajas y modificaciones (o sea todo lo construido por el pattern Work with),.Sin embargo, con esta misma arquitectura también podemos construir aplicaciones customer-facing.

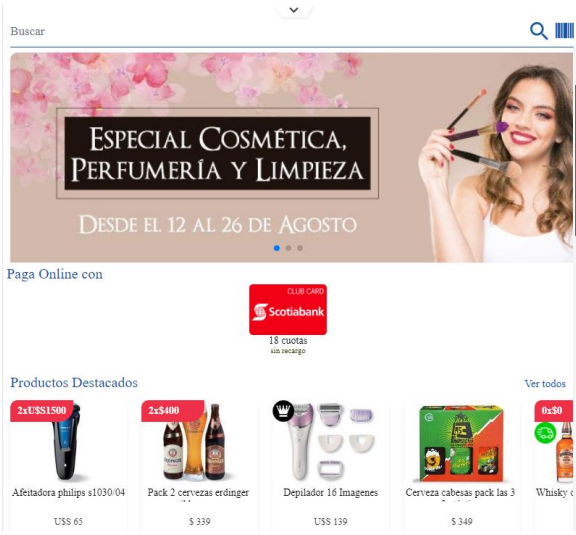
En GeneXus para construir estas aplicaciones utilizamos .Net, .Net Core o Java, y GeneXus utiliza estos lenguajes tanto para el código en el cliente (front-end) como para el código en el servidor (back-end).

Aplicaciones de alta performance con contenido más interactivo





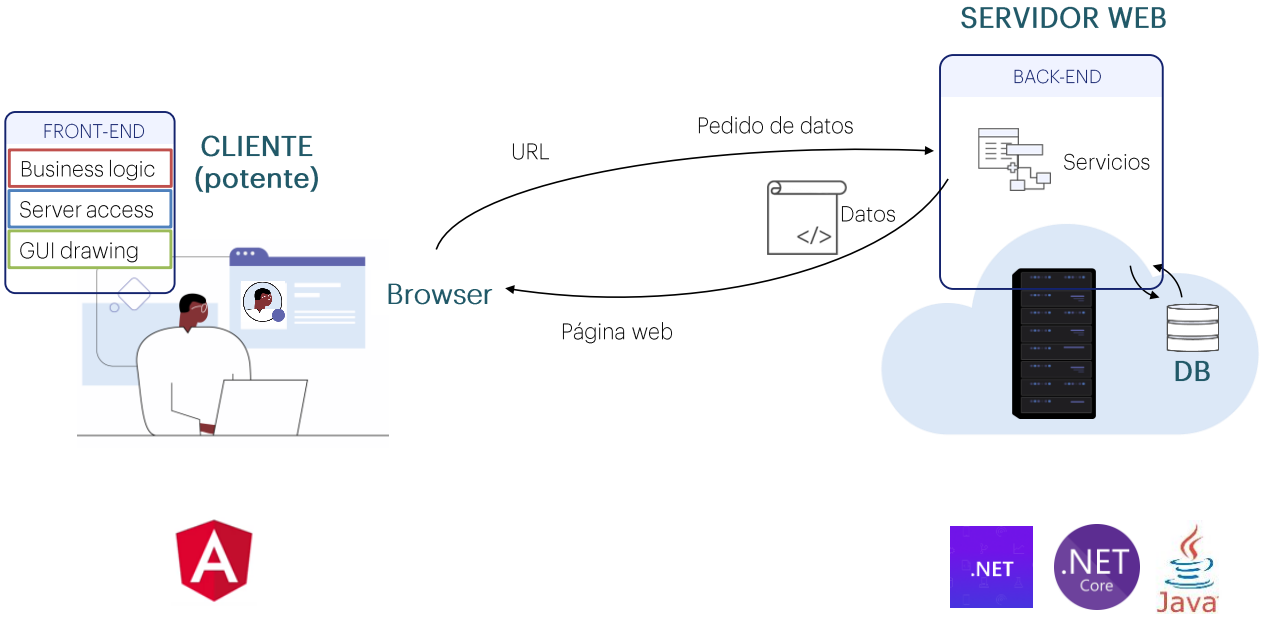




Hay otro tipo de aplicaciones que pueden ser web o nativas para dispositivos móviles, que tienen una gran potencia para mostrar la información en forma muy rápida y con especial foco en el diseño y la interactividad.

Son aplicaciones del tipo customer-facing, es decir orientadas al usuario, con gran riqueza de información en pantalla, que van cargando las partes de la página solo cuando se va necesitando porque están diseñadas para una altísima performance con la mejor experiencia de usuario posible.

Arquitectura de aplicaciones web con énfasis en la UX



En esta arquitectura también ejecuta una parte en el cliente y otra en el servidor, pero en este caso buena parte de la lógica de la aplicación está en el cliente. En el cliente hay 3 capas bien diferenciadas: la lógica del negocio, las comunicaciones con el servidor y la parte que se encarga del dibujo de la pantalla en el navegador.

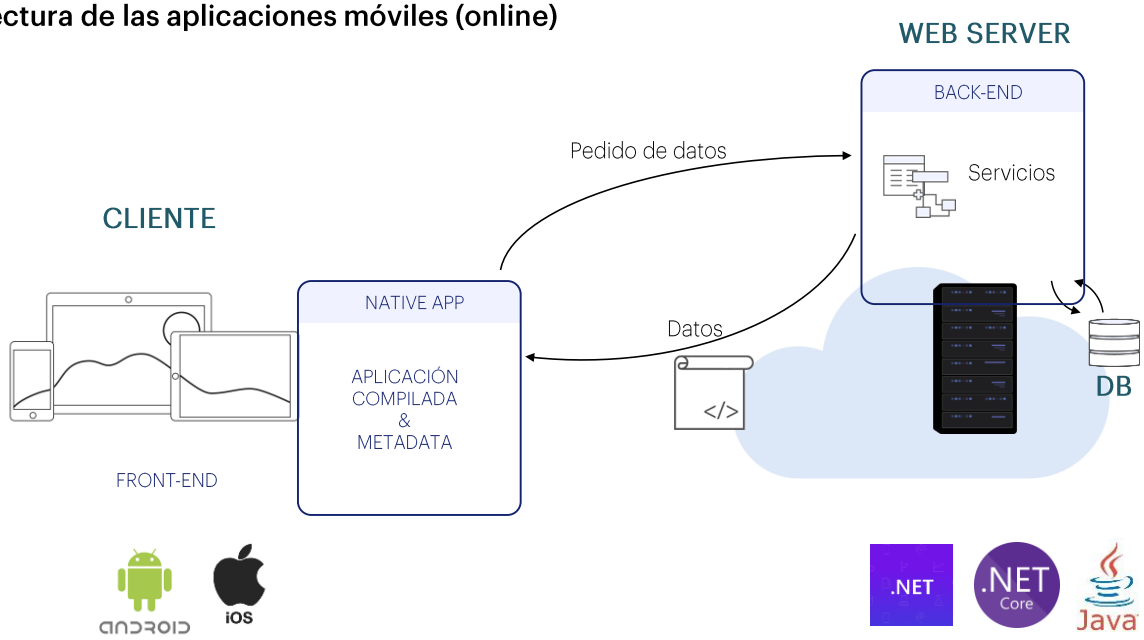
Con este esquema, la mayoría de las funciones se cumplen en el cliente y solamente se accede al server para pedir datos de la base de datos o modificarlos y otros recursos brindados por servicios.

Este tipo de aplicaciones customer-facing que requieren de la potencia en el cliente para poder ofrecer altas prestaciones de experiencia de usuario e interactividad, son más recientes que las aplicaciones responsivas que vimos antes.

Para poder manejar estas exigencias, han surgido en el mercado frameworks como Angular, React o Vue, que tienen toda la potencia para poder construir un cliente más inteligente como el que necesitan estas aplicaciones.

Sin embargo, como mencionamos antes, con GeneXus también es posible desarrollar aplicaciones customer-facing en Java, Net o .Net Core, aunque la ventaja de programarlas en Angular es que los mismos objetos paneles que diseñemos para la interfaz de usuario podremos reutilizarlos prácticamente sin cambios para generar la aplicación mobile nativa, es decir, para Android o iOS.

Arquitectura de las aplicaciones móviles (online)



Hasta ahora nos hemos concentrado en las aplicaciones web, veamos ahora la arquitectura de una aplicación móvil nativa.
En particular veremos la arquitectura de aplicaciones móviles siempre conectadas a través de wifi (aplicaciones online), que es el caso más común de las aplicaciones nativas, pero sabiendo que GeneXus también construye las desconectadas.

Una aplicación móvil también tiene una parte que ejecuta en el cliente (en este caso el dispositivo móvil) y una parte que ejecuta en el servidor, que provee información al cliente.
El código de la aplicación ejecuta en el dispositivo, que accede al servidor solo cuando necesita datos.
En GeneXus podemos generar estas aplicaciones para dispositivos Android o iOS. ´Los servicios en el server (que proveen los datos al cliente) se pueden generar en Java. .Net o .Net Core.

Como vemos, esta arquitectura es muy similar a la de las aplicaciones web de alta performance y por esa razón la forma de programar es muy similar, con algunas diferencias que veremos más adelante.

En resumen, tendremos dos formas de programación

Cliente liviano
+
Lógica de negocios en el server

Aplicaciones web con énfasis en back-office

Cliente inteligente
+
Servicios en el server

Aplicaciones web con énfasis en la UX

Aplicaciones nativas para móviles

La arquitectura de las aplicaciones condicionan muchas cosas de la aplicación, pero fundamentalmente la forma de programarlas.

En el caso de que el cliente sea liviano, prácticamente todos los requerimientos de la aplicación serán resueltos en el servidor, por lo que nuestra programación siempre estará orientada a operaciones en el servidor como por ejemplo, el acceso a la base de datos.

En cambio cuando contamos con un cliente potente, muchas operaciones se podrán realizar en el cliente mismo, sin necesidad de recurrir al servidor. Sin embargo, hay otras operaciones que tienen que ver fundamentalmente con la interacción con la base de datos que sí requiere de los programas del server.

Por este motivo, deberemos tener una sintaxis diferente para indicar cuándo queremos que tal código se ejecute en el cliente y cuál en el servidor. Esto es válido también para la programación de aplicaciones nativas para dispositivos móviles.

A continuación, veremos cómo se programan aplicaciones de cada tipo.

GeneXus™

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