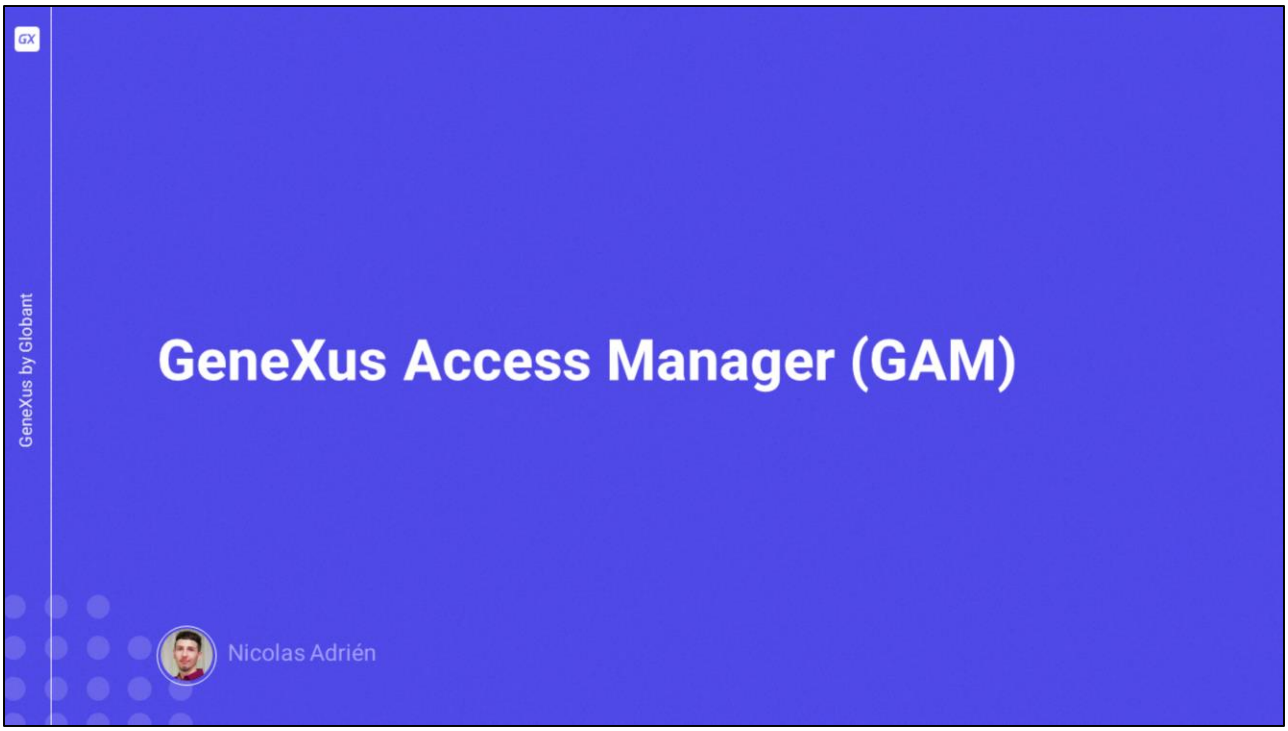
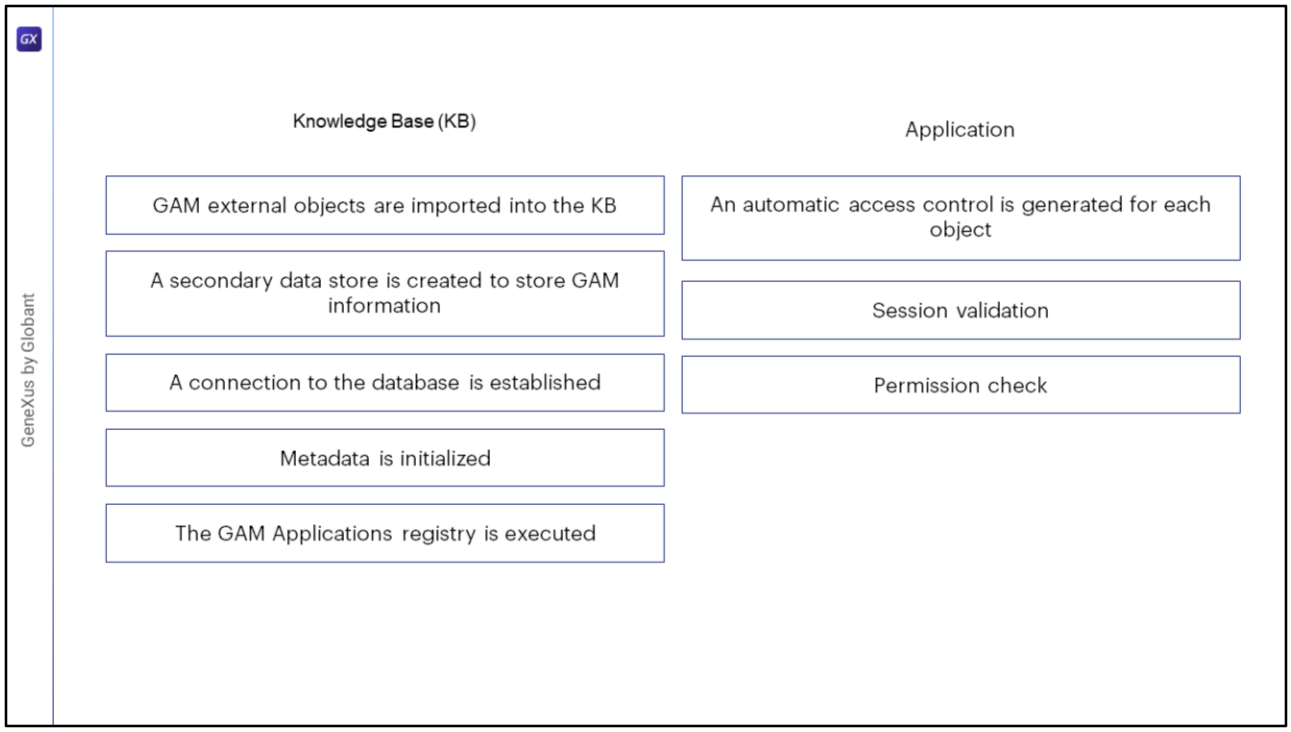

GeneXus by Globant
GeneXus™
by Globant
training.genexus.com




El módulo de seguridad de cualquier aplicación GeneXus (tanto aplicaciones web como móviles) lo proporciona GeneXus Access Manager, o llamado comúnmente como GAM.


Este se creó con el fin de resolver las funcionalidades de autenticación y autorización de las aplicaciones.
Los controles de seguridad se realizan automáticamente al habilitar la seguridad integrada en la aplicación.

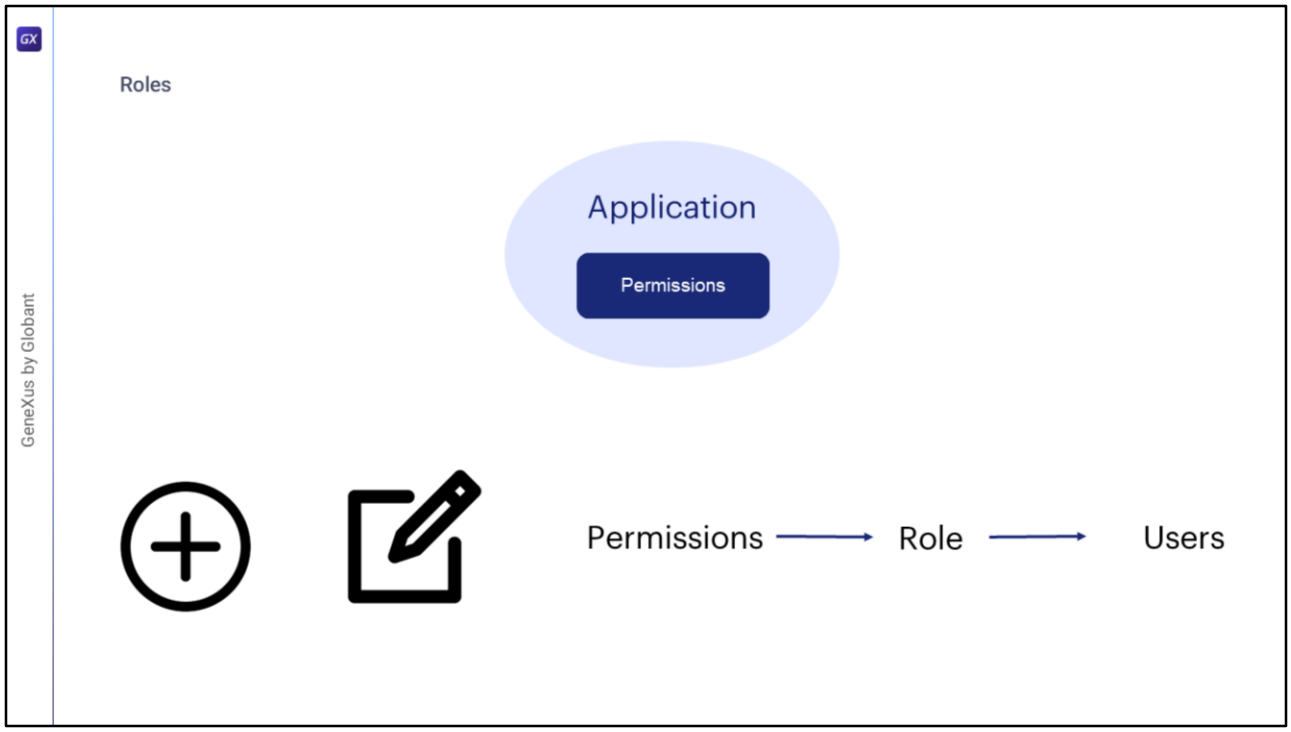



¿Qué sucede a nivel de KB después de habilitar la seguridad integrada?

- Se importan a la KB los objetos externos GAM.
- Se crea un almacén de datos secundario para almacenar información GAM.
- Se establece una conexión con la base de datos.
- Se inicializan los metadatos.
- Se ejecuta el registro de Aplicaciones GAM.

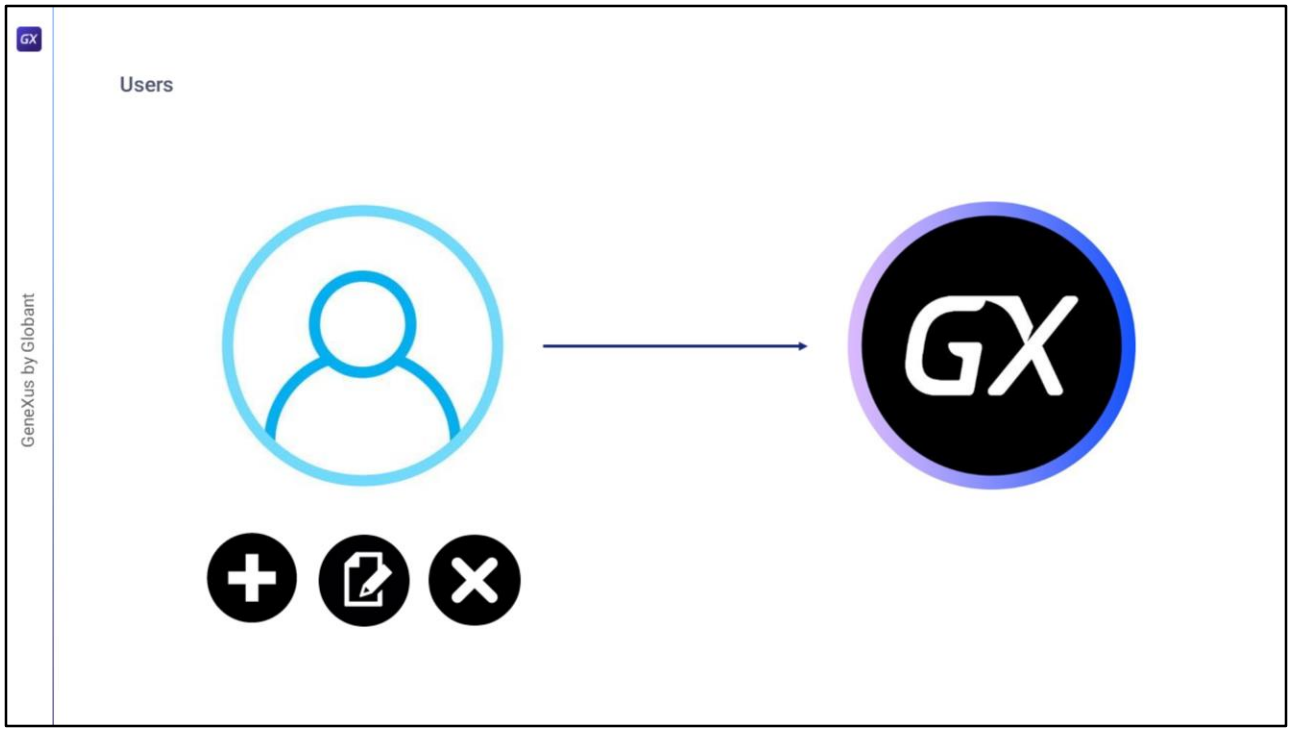
Ahora, ¿qué sucede a nivel de aplicación después de habilitar la seguridad integrada?

- En este caso se genera un control de acceso automático en cada objeto que tiene definida la propiedad de Nivel de Seguridad Integrado.
- Antes de ejecutar cualquier objeto, el código generado valida si la sesión es válida; de lo contrario, se ejecuta un Objeto de inicio de sesión para Web o mobile, para iniciar la sesión.
- Además, se comprueba automáticamente los permisos al inicio (esto último ocurre cuando la propiedad Nivel de seguridad integrado está con el valor “Autorización”).



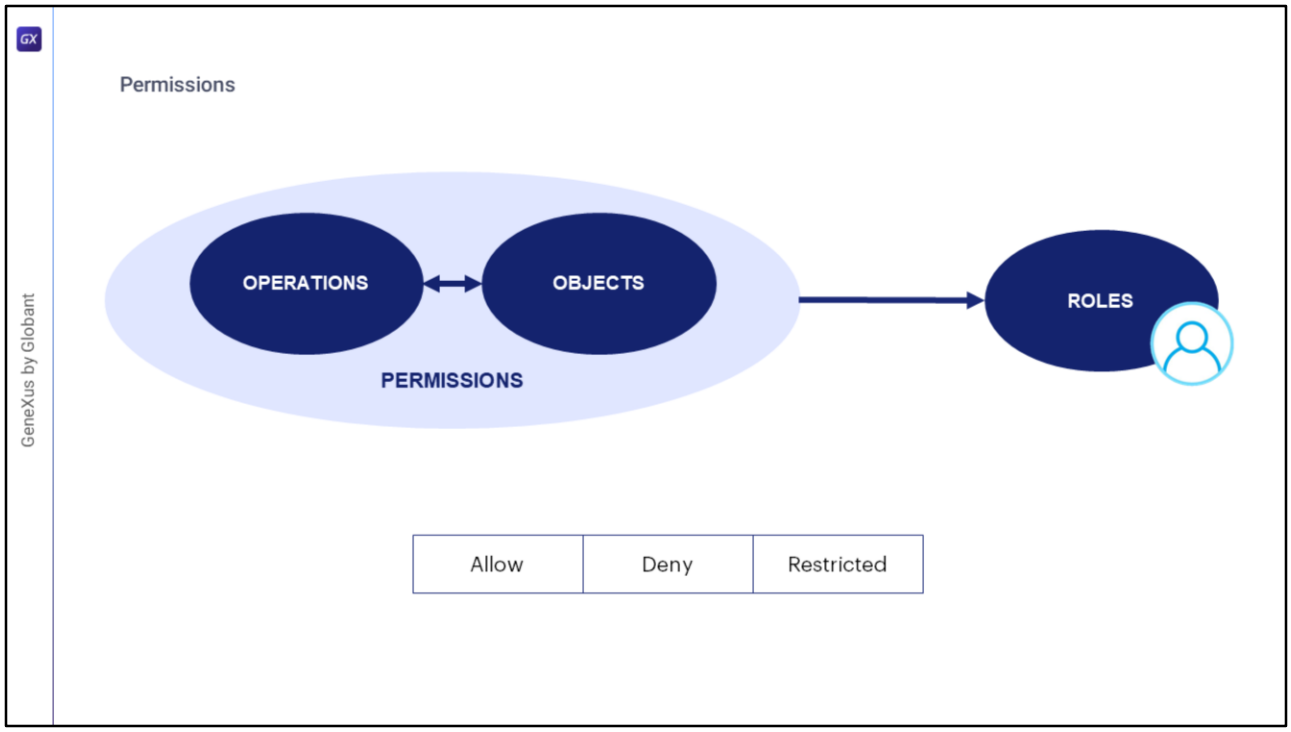

Yendo a los elementos propios del GAM, tenemos nuevamente a los Roles.
Un rol en GAM es la forma de agrupar permisos en una aplicación, y están organizados en jerarquías de roles.

Se pueden crear nuevos roles, editar existentes y agregar permisos a roles ya definidos en una aplicación desde el Backoffice web de GAM.
Si un rol está asociado con un usuario, los permisos del rol están asociados con ese usuario. Esto significa que cuando se comprueban los permisos de usuario en tiempo de ejecución, se tienen en cuenta los permisos asociados al usuario a través de los roles. Sin embargo, los permisos asociados directamente con el usuario tienen prioridad.




En cuanto a los usuarios, estos son aquellos individuos que utilizan de forma habitual una aplicación GeneXus.
Con GAM se pueden agregar, editar y eliminar usuarios.

A su vez, un usuario de GAM puede tener varios Roles asociados y un Rol Principal.



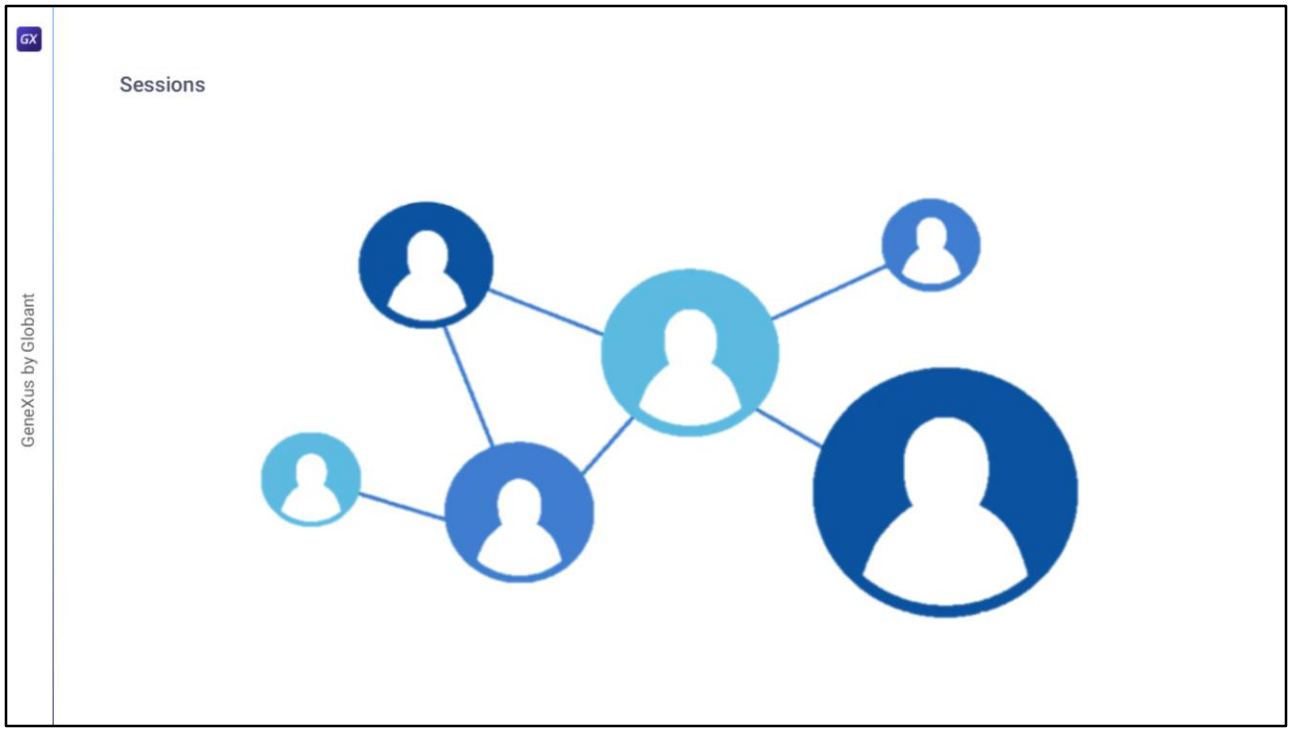

Los permisos, como dijimos anteriormente describen la posibilidad de realizar operaciones sobre objetos.
Estos pueden estar asociados a uno o más roles.

El permiso tiene un tipo de acción predeterminada que especifica si el permiso está restringido o no: Allow o Restricted. Por defecto está habilitado para todos los usuarios.

Los permisos son por aplicación, y al asignarlo a un Rol se puede especificar su acción para ese Rol en específico. Las opciones son Allow, Deny o Restricted. Cuando se asigna Allow, es un permiso por la positiva. Si queremos denegar un permiso, tenemos las otras dos opciones: Deny o Restricted.

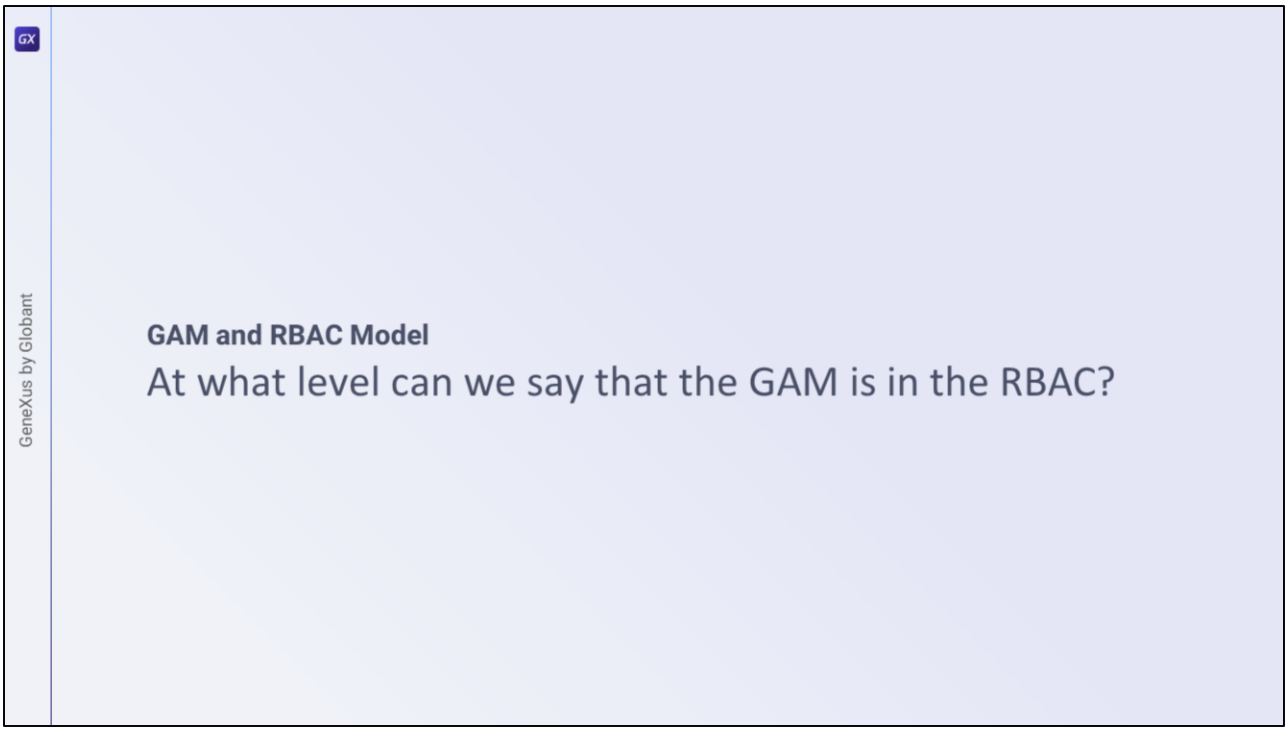
Al momento de crearse una sesión para un usuario, GAM obtiene los roles asociados a este. Para chequear si un Usuario tiene un permiso válido, GAM verifica en todos esos Roles: Si uno de ellos tiene un permiso Restricted y otro tiene el mismo permiso Allow, gana el Allow, pero si uno de ellos tiene un permiso Deny y otro tiene el permiso Allow, gana el Deny, donde incluso este es más fuerte que el Restricted.

Finalmente, el permiso es asignado directamente a un usuario.

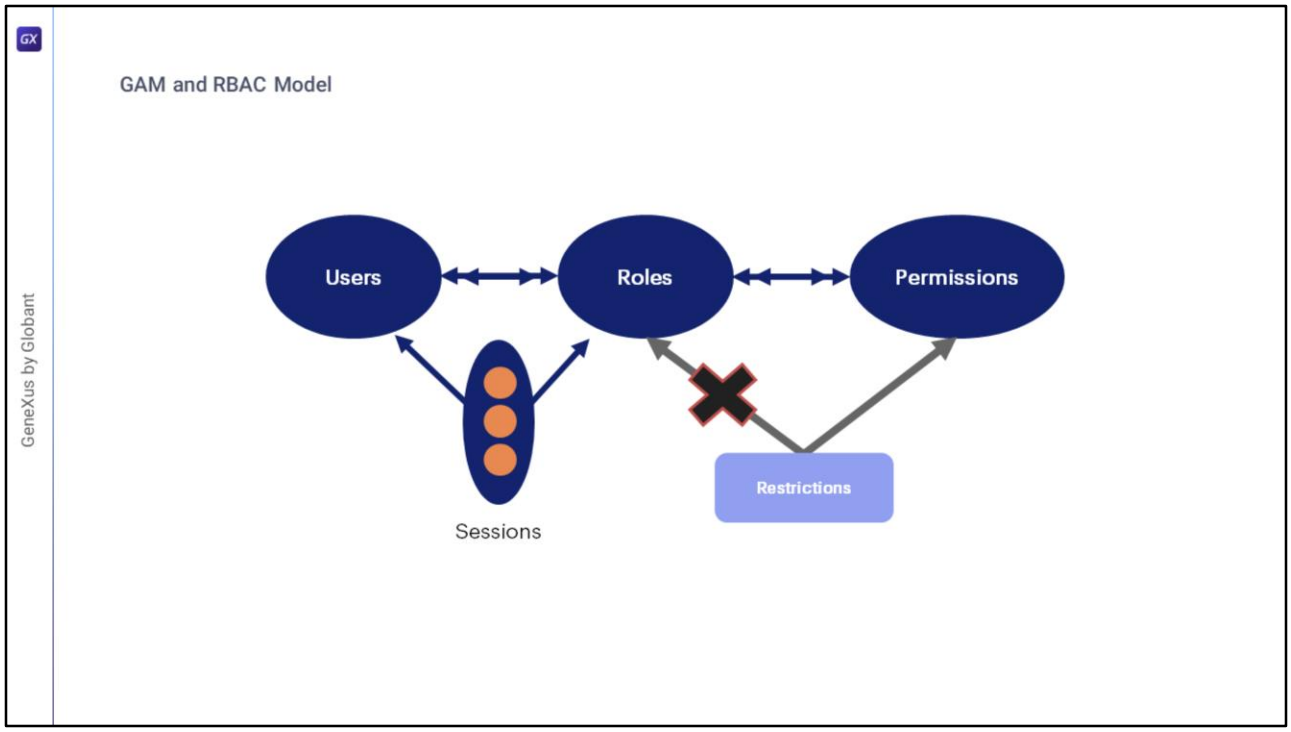



Al igual que en el RBAC tenemos a las Sesiones, que realizan el mapeo entre los usuarios y sus roles activos.

Conceptualmente, se manejan exactamente igual que como las describimos en el RBAC.
Cada sesión corresponde a cada usuario y es manejada automáticamente por GAM.
Contienen un token de identificación, y expiran según las políticas de seguridad definidas.

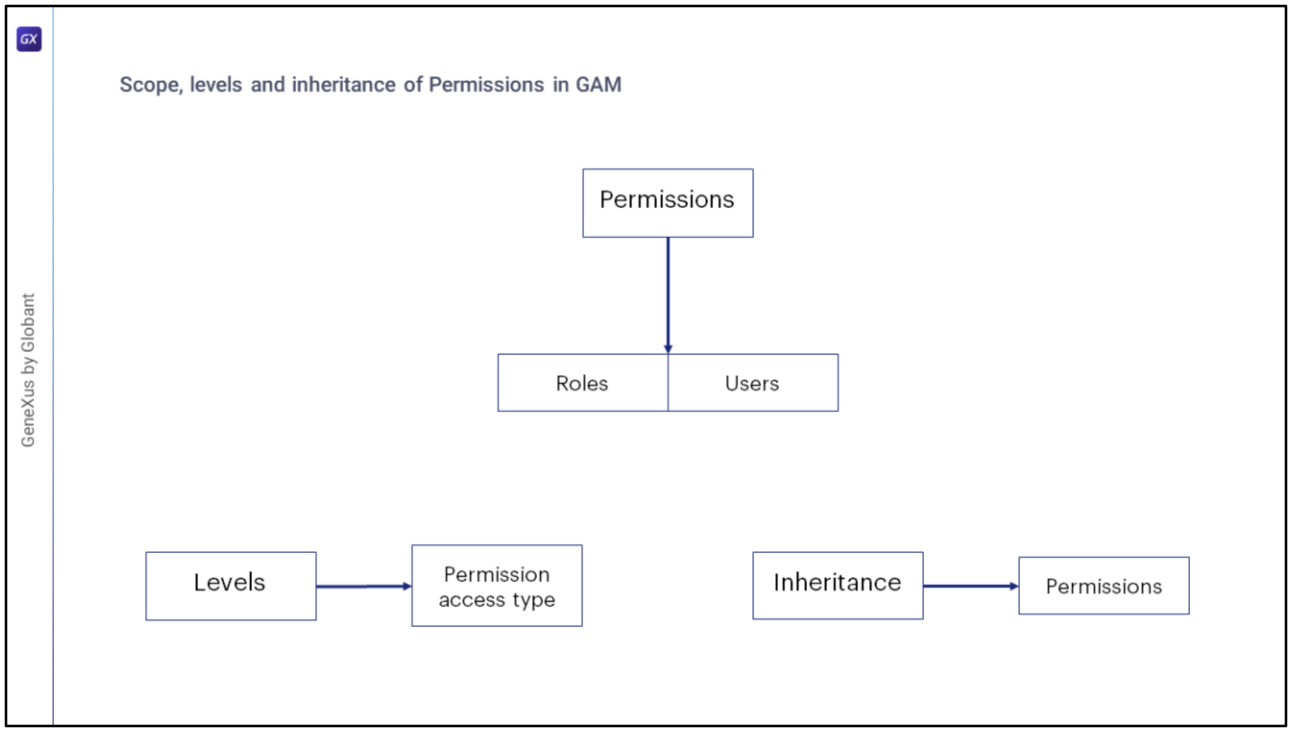



¿En qué nivel podemos decir que se encuentra el GAM en el RBAC?




El GAM se encuentra ubicado en el nivel 3, donde se aplican los mecanismos de separación por roles, con determinados permisos para cada usuario, que, si recordamos, era lo que correspondía a este nivel.

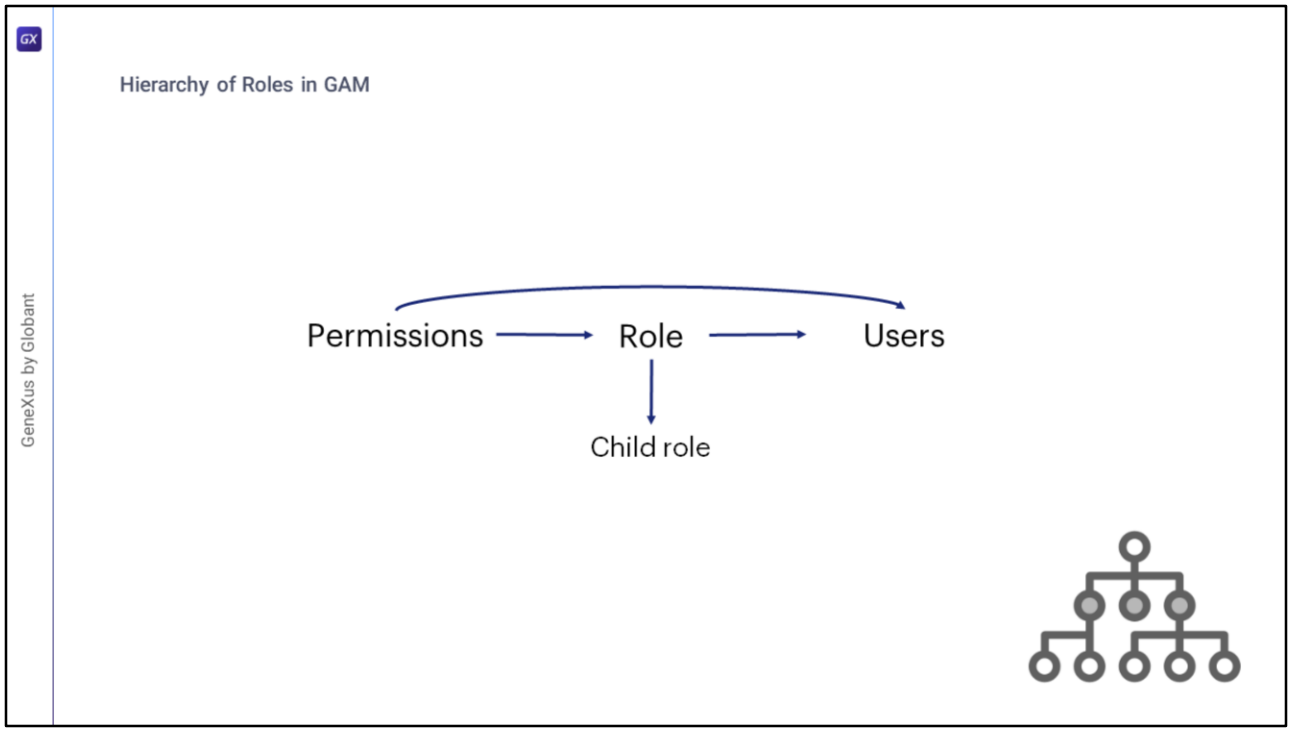
En el caso de las Restricciones, si bien existen restricciones a nivel de permisos, no existen a nivel de roles.




Los Permisos en GAM existen dentro del alcance de las Aplicaciones y se asignan a Roles y a Usuarios en el Repositorio GAM.

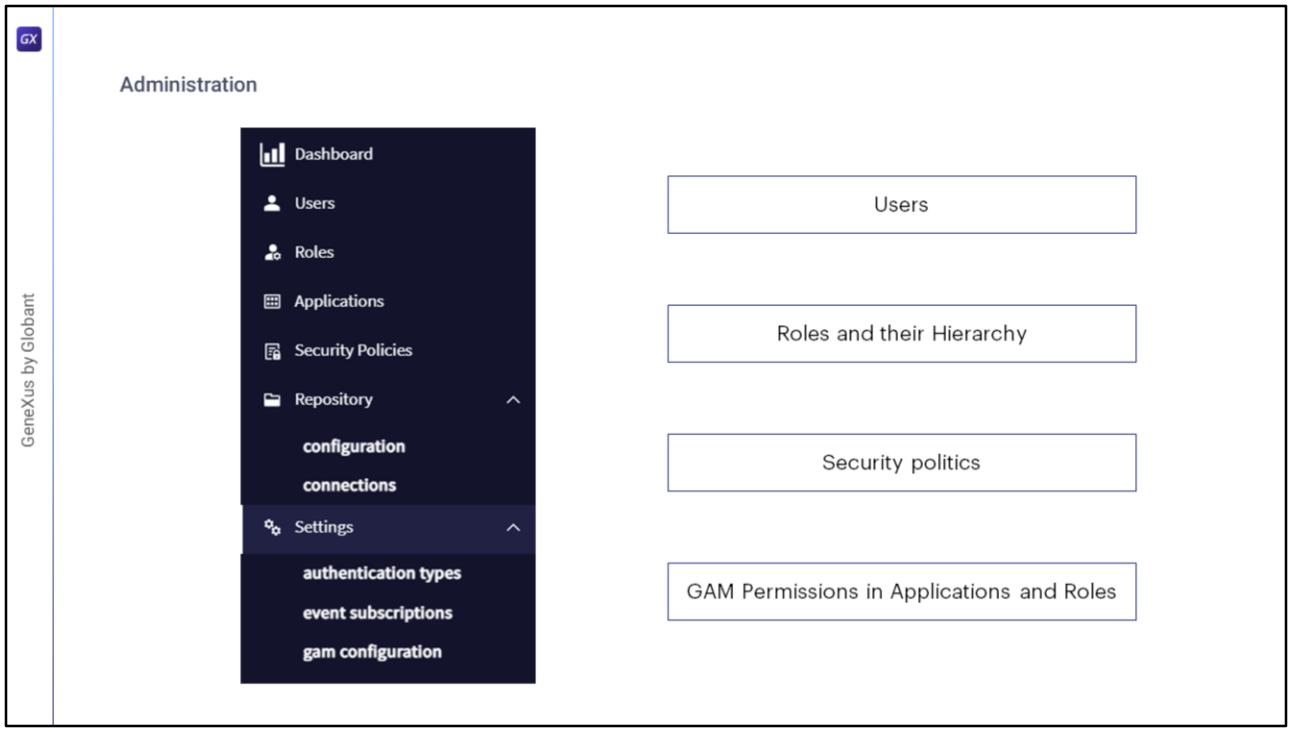
En cuanto a los niveles, el nivel de permisos que tiene un usuario en tiempo de ejecución depende del tipo de acceso de permiso.

Finalmente, cuando se agrega cualquier permiso de control total a un rol, de forma predeterminada todos los permisos secundarios de este control total también se agregan al rol, a menos que se haya perdido la herencia.




Como se explicó anteriormente, si un rol está asociado con un usuario, los permisos del rol están asociados indirectamente con ese usuario. Esto significa que cuando se comprueban los permisos de usuario en tiempo de ejecución, se tienen en cuenta los permisos asociados al usuario a través de los roles.

Además, si el rol tiene hijos, los permisos de los roles hijos también se asocian con el usuario.



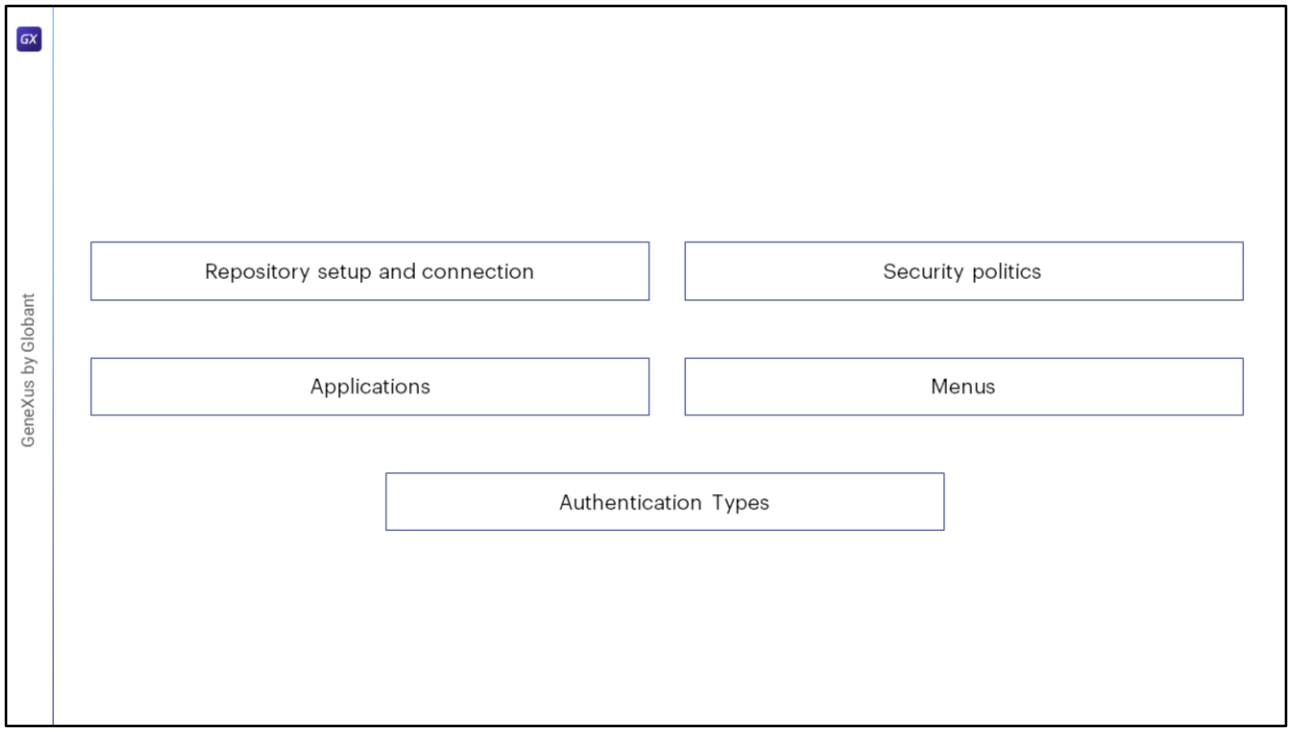

El backend de GAM es una aplicación web que le permite al administrador de GAM administrar todo el sistema por completo.
Entre estas funciones, se encuentra todo lo que mencionamos anteriormente relacionado al RBAC:

- Usuarios
- Roles y su Jerarquía
- Políticas de Seguridad
- Permisos GAM en Aplicaciones y Roles

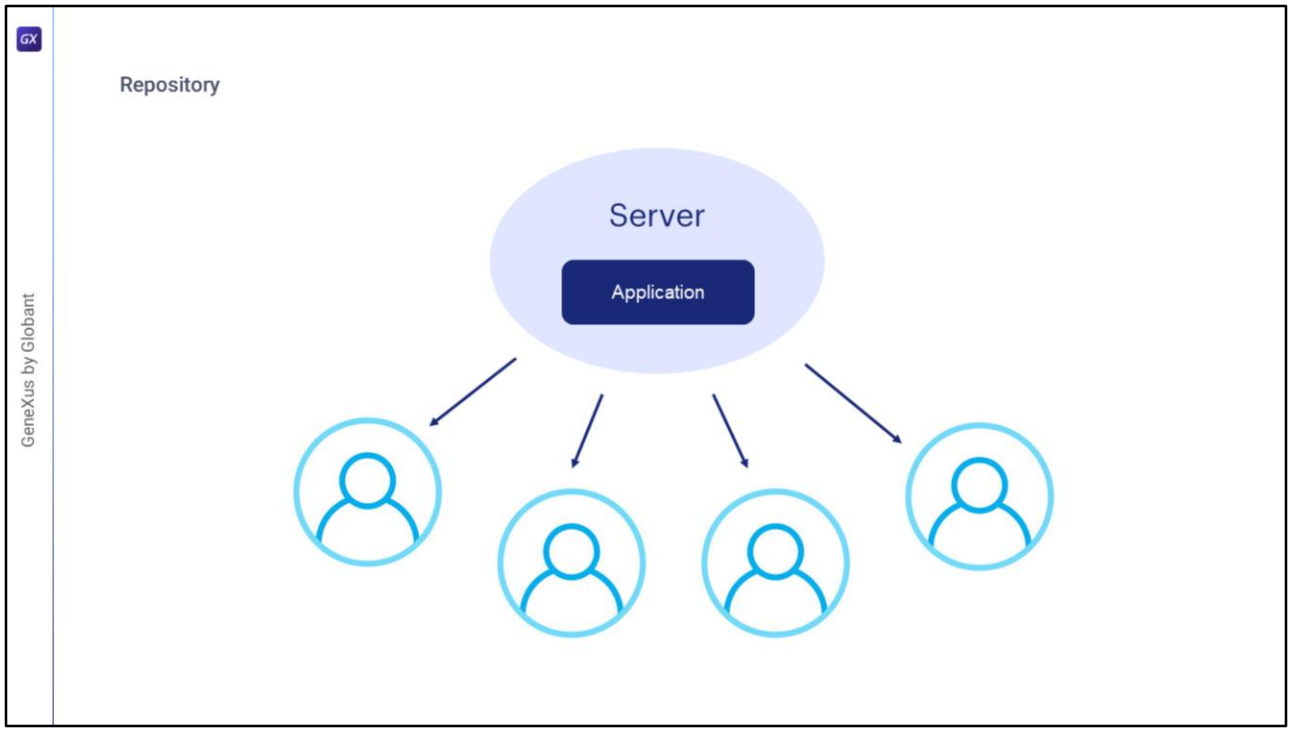
A su vez, también le brinda la posibilidad de administrar las aplicaciones, configurar el repositorio y sus conexiones, cambios de contraseña, tipos de autenticación, entre otros, que los vamos a ver a continuación…

GeneXus by Globant

# Other fundamental concepts of GAM



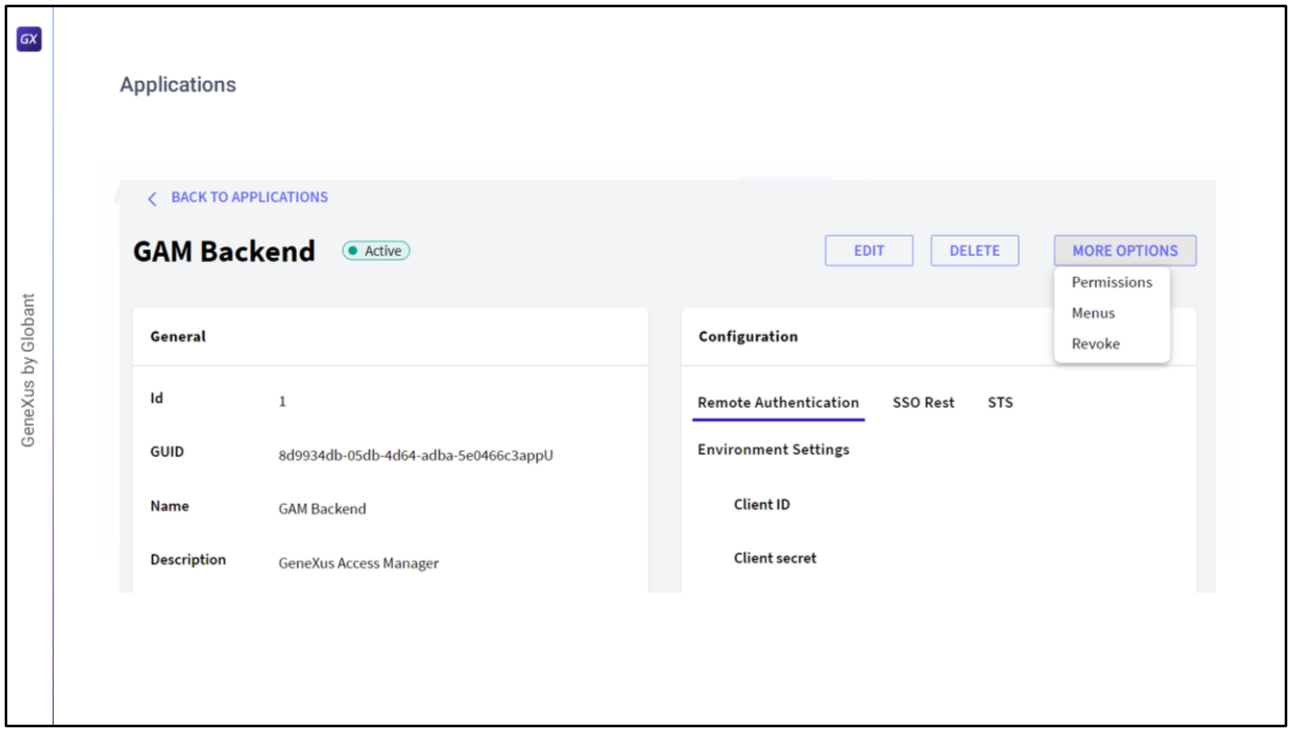

Además de los conceptos mencionados hasta el momento, GAM brinda la posibilidad de administrar más opciones, como puede ser la configuración y conexión del repositorio, sus aplicaciones y políticas de seguridad, los menús, y los tipos de autenticación existentes.




Primero veamos el Repositorio.

Un repositorio es una entidad de GAM que soporta una arquitectura donde una sola instancia de la aplicación se ejecuta en un servidor y sirve a múltiples usuarios.

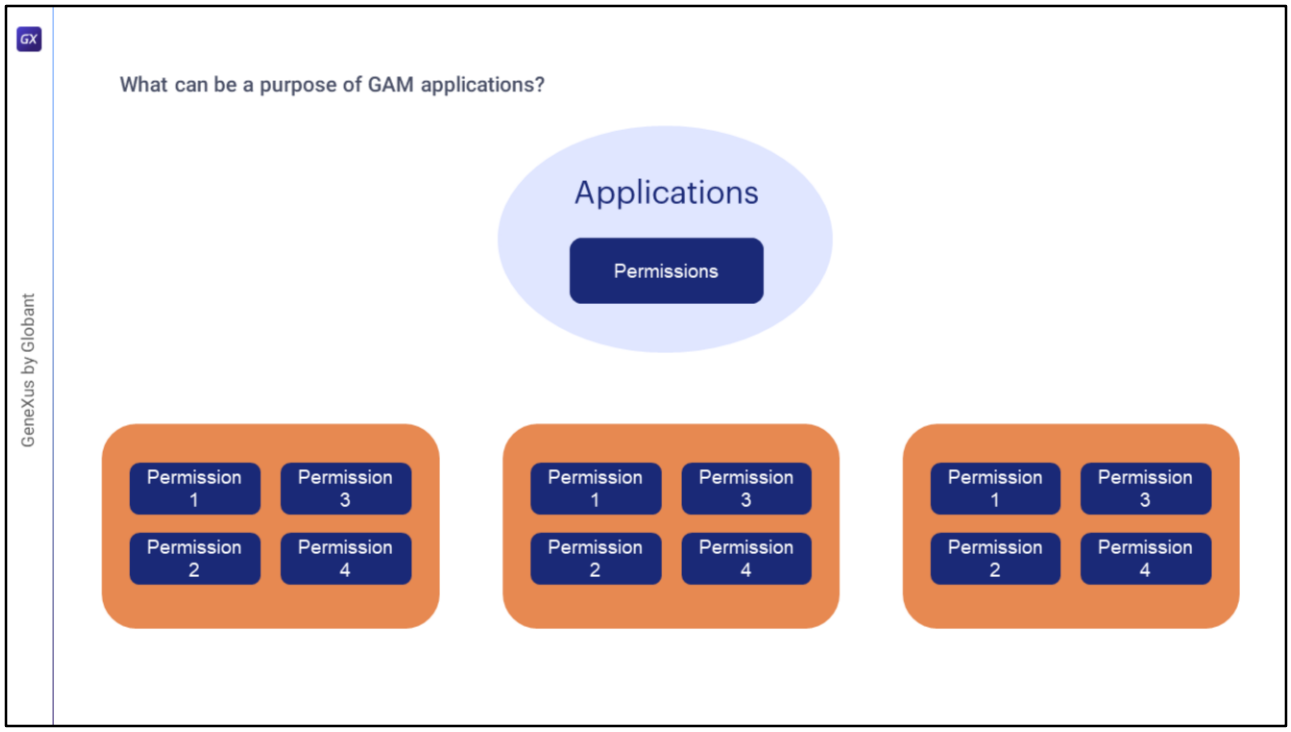
En este escenario, estos usuarios deben usar la misma base de datos GAM y diferentes repositorios GAM dentro de la base de datos. A esto comúnmente se le llama aplicación multiusuario, y permite a cada usuario un conjunto exclusivo de roles, aplicaciones y políticas de seguridad de GAM.




¿Qué son las aplicaciones GAM?
GeneXus tiene un ambiente web, y tiene una aplicación web GAM, la cual puede ser identificada mediante un GUID.
Además, la aplicación tiene un nombre (el nombre de la KB) e incluye los permisos de todos los objetos web de la KB

En cuanto a Aplicaciones móviles:
Las aplicaciones GAM Mobile agrupan los permisos de todos los objetos mobile principales de la KB. Cada objeto principal de estos determina la existencia de una aplicación GAM.

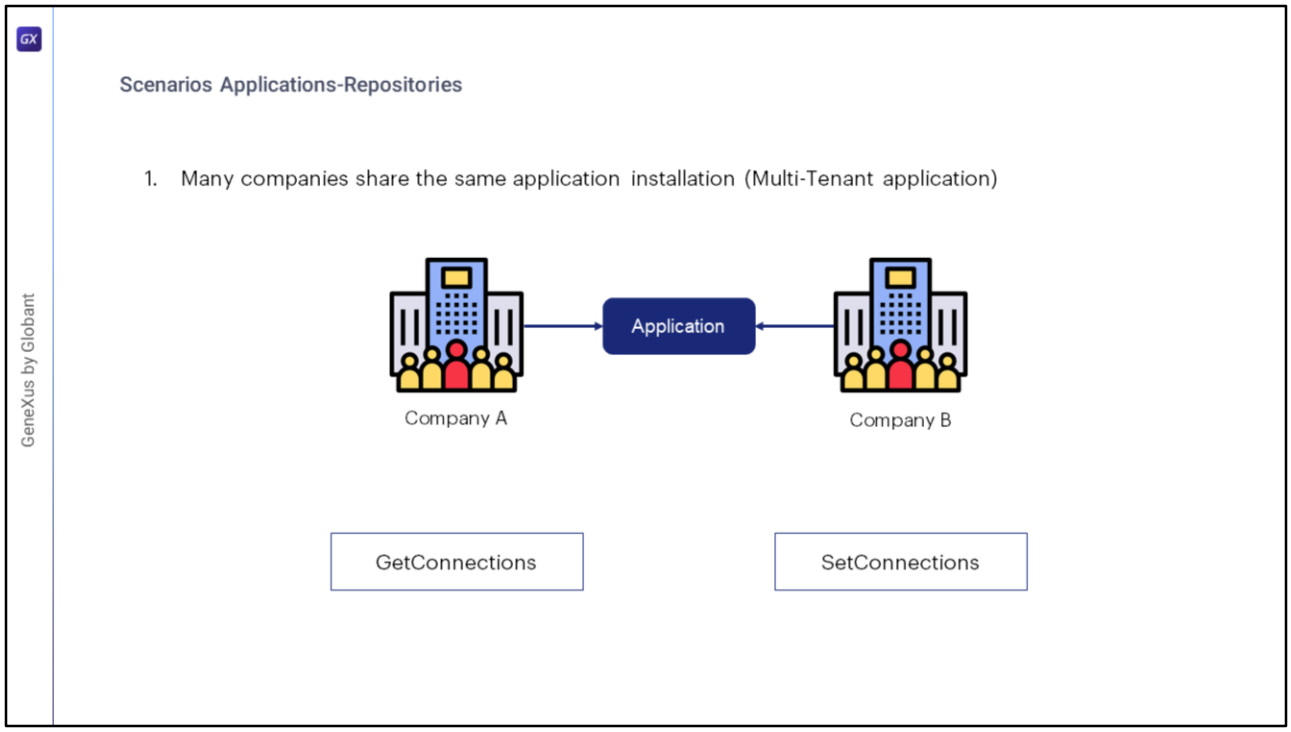
Las aplicaciones GAM se definen dentro de un repositorio y como ya dijimos, cada uno de estos puede contener más de una aplicación.




¿Cuál puede ser un propósito de las aplicaciones GAM?

Uno de ellos puede ser asociar permisos a estas aplicaciones y formar grupos de permisos.

Como los permisos se comprueban teniendo en cuenta la aplicación que se está ejecutando, entonces cuando el usuario inicia sesión en un repositorio y se necesita un permiso para ejecutar una acción, el permiso debe estar definido en la aplicación GAM que está ejecutando (y debe tener un rol donde se permita este permiso).



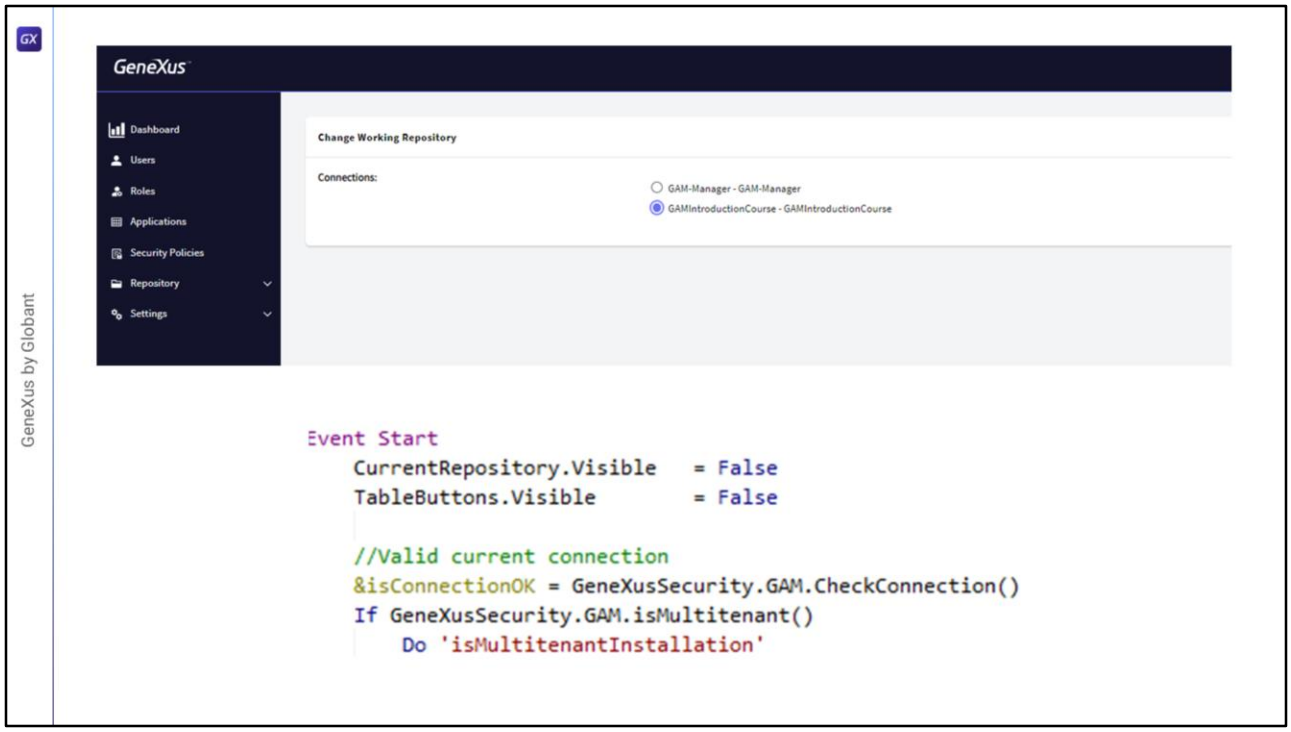

Pongamos ejemplos de escenarios con Aplicaciones y Repositorios.

**El primer escenario está compuesto por muchas empresas que comparten la misma instalación de la aplicación (lo que se define como aplicación Multi-Tenant).**
Como dijimos antes, el diseño del modelo de GAM permite conectarse a múltiples Repositorios para resolver muchos escenarios donde un solo Repositorio GAM no sería suficiente.
En este caso, cada Repositorio tendrá sus propios usuarios administradores, y la información de un Repositorio no será accesible desde los demás.

Es el caso de una aplicación multiusuario en la que una sola instancia del software se ejecuta en un servidor, sirviendo a múltiples organizaciones de clientes.

Llevando este escenario a la práctica, tenemos dos métodos de GAM para poder utilizar: **GetConnections** y **SetConnections**.
El método GAM.GetConnections devuelve en una colección una lista de conexiones que contiene la clave almacenada en el archivo connection.gam.
El método GAM.SetConnection devuelve verdadero si la conexión se estableció correctamente. Esto significa que todos los métodos GAM accederán al nuevo conjunto de repositorios.

Veamos un ejemplo de esto.




Antes de profundizar en el mismo, veamos un concepto que se utiliza en la implementación de esto y es el GAM Manager Repository.
Este es un Repositorio particular que se utiliza para administrar el resto de Repositorios, y son los usuarios de este Repositorio, en conjunto a los usuarios de repositorios que tengan habilitado la propiedad de Manager, los únicos que pueden crear nuevos Repositorios y administrarlos.

No entraremos en detalle en este tema, por lo cual recomendamos ver la documentación del mismo en la Wiki de GeneXus para entenderlo a fondo. Nosotros simplemente asumiremos que ya lo tenemos creado y configurado.

Podemos mostrar esto con uno de los ejemplos que nos brinda GAM. En este caso utilizaremos GAMExampleLogin.
Vemos que en el evento Start, se chequea la conexión actual con el Repositorio, y si estamos ante un caso de Multitenant, se realiza un llamado a una subrutina.

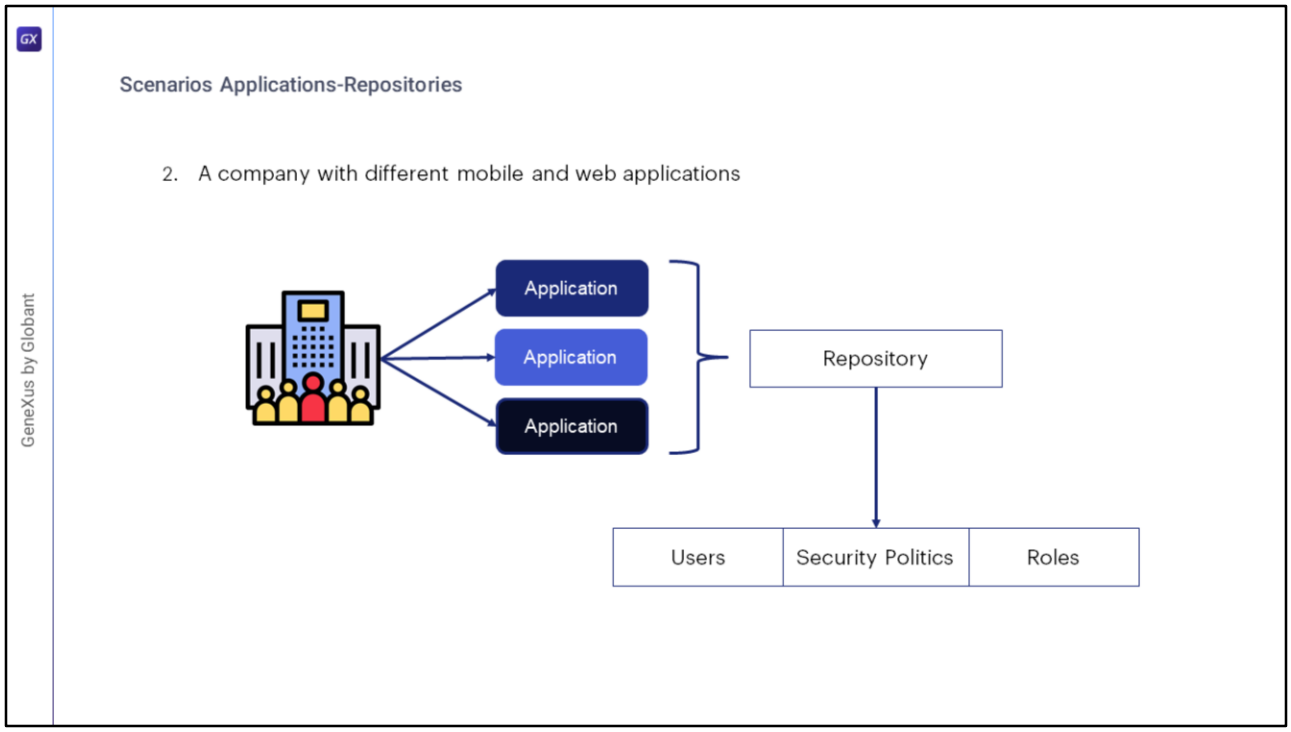
```
Sub 'isMultitenantInstallation'
    //Read Curent Repository
    &GAMRepository = GeneXusSecurity.GAMRepository.Get()
    //Check if the current repository uses an authentication master repository
    If not &GAMRepository.AuthenticationMasterRepositoryId.IsEmpty()
        &isConnectionOK = GeneXusSecurity.GAM.SetConnectionByRepositoryId(&GAMRepository.AuthenticationMasterRepositoryId, &Errors)
    Endif
    If not &isConnectionOK
        If GeneXusSecurity.GAM.GetDefaultRepository(&RepositoryGUID)
            &isConnectionOK = GeneXusSecurity.GAM.SetConnectionByRepositoryGUID(&RepositoryGUID, &Errors)
        Else
            &ConnectionInfoCollection = GeneXusSecurity.GAM.GetConnections()
            If &ConnectionInfoCollection.Count > 0
                //The first connection found is established by default
                &isConnectionOK = GeneXusSecurity.GAM.SetConnection(&ConnectionInfoCollection.Item(1).Name, &Errors)
            EndIf
        Endif
    Endif

    If &isConnectionOK
        &GAMRepository = GeneXusSecurity.GAMRepository.Get()
        //Check if the current repository uses an authentication master repository
        If not &GAMRepository.AuthenticationMasterRepositoryId.IsEmpty()
            &isConnectionOK = GeneXusSecurity.GAM.SetConnectionByRepositoryId(&GAMRepository.AuthenticationMasterRepositoryId, &Errors)
            &GAMRepository  = GeneXusSecurity.GAMRepository.Get()
        Endif
        CurrentRepository.Caption = "Repository: " + &GAMRepository.Name
        CurrentRepository.Visible = True
    Endif
EndSub
```

En la subrutina, vemos que lo primero que se trae es el repositorio actual, para luego hacer el chequeo si es master o no y conectarse.
Si la conexión no fue exitosa, pregunta si el repositorio actual es el por defecto para setear una conexión con él. En caso de que no lo sea, utiliza los métodos que vimos anteriormente: **GetConnections** y **SetConnections**.
Primero hace el Get obteniendo la lista de conexiones, y se queda con la primera.

Quedémonos con esa parte del código. En este caso se queda con la primera dado que es a modo de ejemplo de implementación básica. En un caso personalizado el desarrollador puede elegir a que repositorio quiere conectarse según las condiciones que quiera o desea exponer.
Recordemos que estas implementaciones son de ejemplo y guía para que realicen sus propias implementaciones.

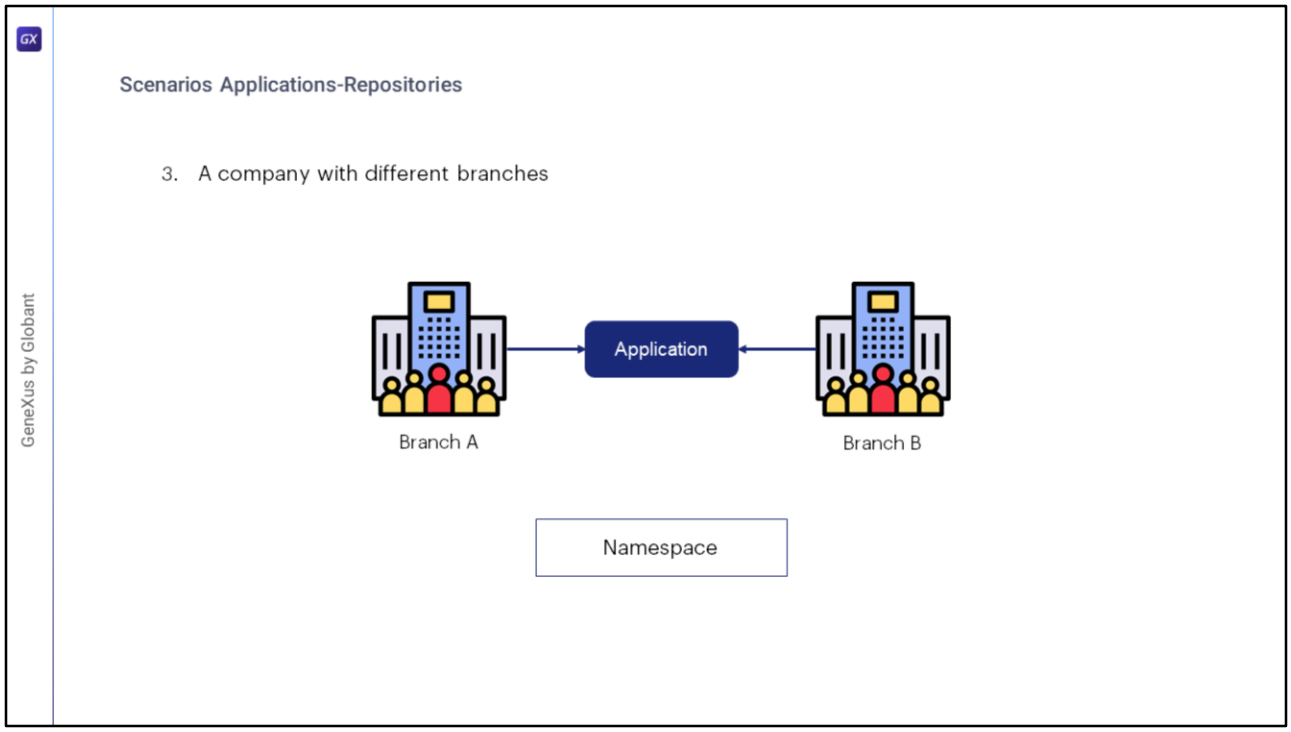
Con esos dos métodos, tenemos todo el control de las conexiones a los repositorios. El resto del código ya no viene al caso.




**En el segundo escenario tenemos una empresa con distintas aplicaciones móviles y web.**

En este escenario de uso, la empresa tiene diferentes aplicaciones, y se comparten los usuarios, políticas de seguridad, y roles, teniendo un solo repositorio.

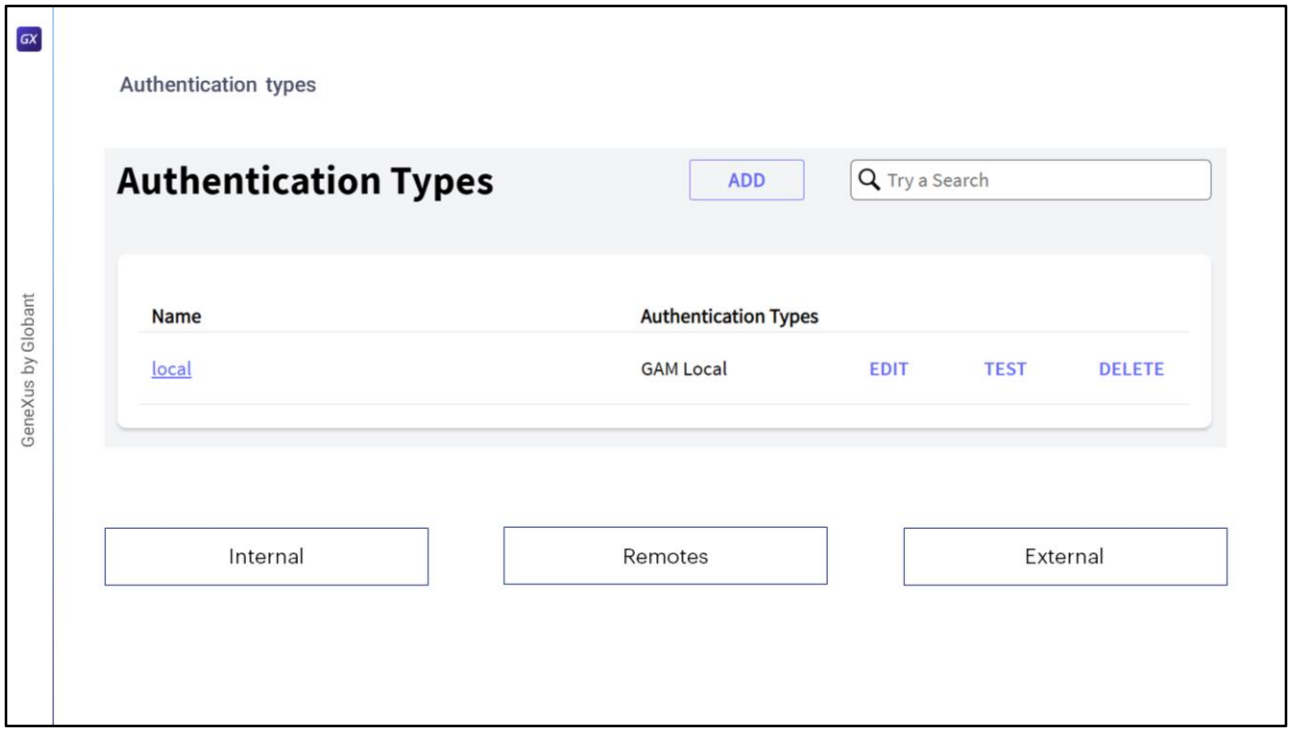
Además, cada aplicación contiene sus propios permisos.




**En el tercer y último escenario, tenemos una empresa con diferentes sucursales.**
En este escenario de uso, la empresa tiene una sola aplicación que es usada por todas las sucursales diferentes que existen en la empresa. O sea, cada sucursal es representada por un repositorio.
A diferencia del escenario 1, en este caso los usuarios son los mismos para todas las sucursales, aunque tienen la salvedad de que cada uno pueda tener distintas políticas de seguridad, roles y permisos, según la sucursal de la aplicación a la que están conectados.

Dado el diseño del modelo de repositorios en GAM, los usuarios podrían habilitarse en muchos repositorios si tienen el mismo namespace que el del repositorio en el cual están habilitados.
El Namespace de los Repositorios GAM, tiene como finalidad agrupar en un contenedor abstracto todos los Repositorios que pertenecen a una misma Empresa.

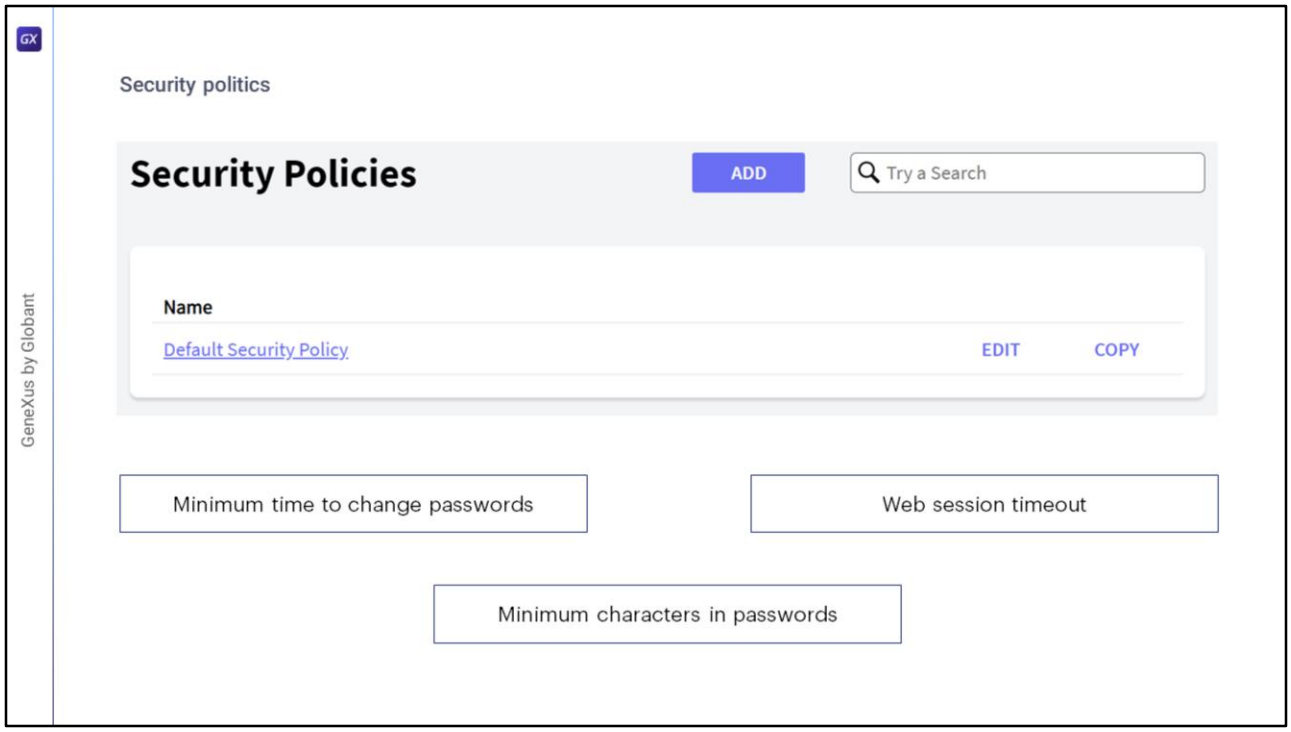
Algo a destacar, es que los tres escenarios se resuelven con solo una base de datos, sin necesidad de crear otras por cada sucursal o aplicación.




Tipos de Autenticación.
La autenticación es el acto o proceso de confirmar que algo (o alguien) es quien dice ser.

GAM ofrece distintos tipos de autenticación, ya sea internos, remotos, y externos (como puede ser servicios web, redes sociales o Google).

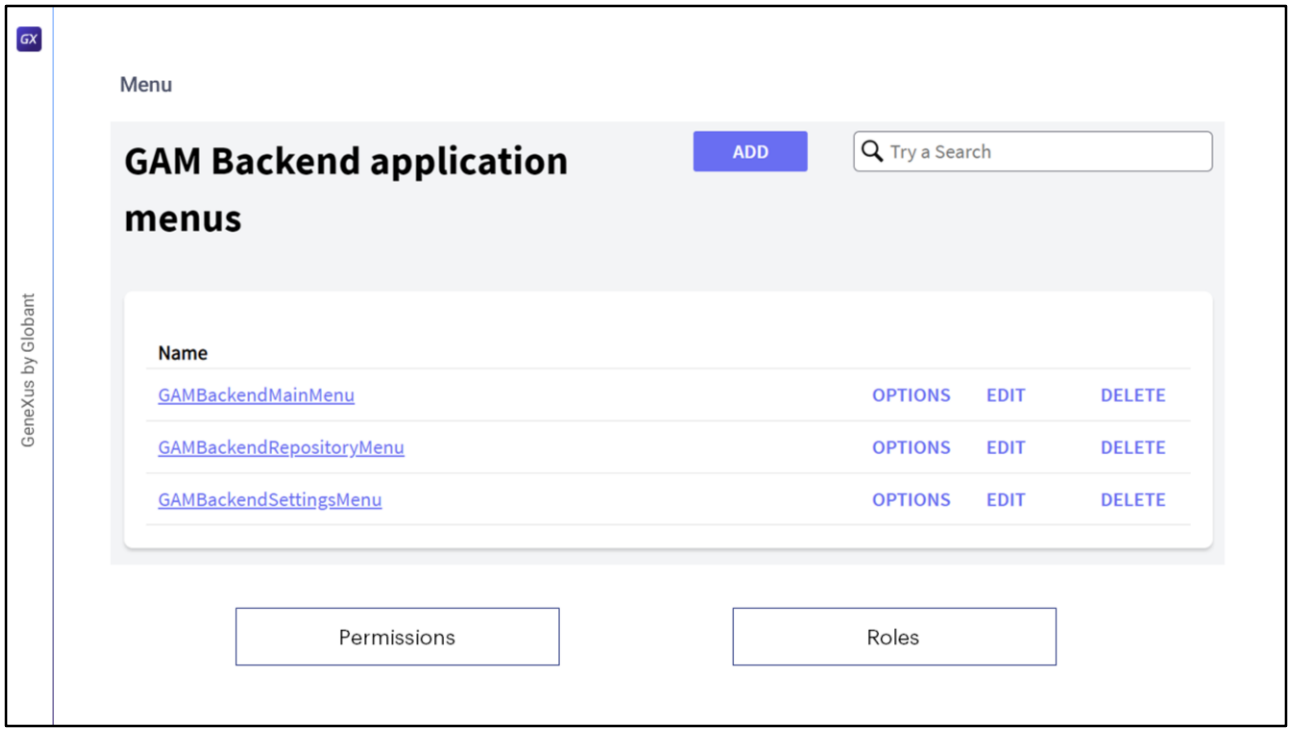
Esto lo veremos con detalle más adelante en el curso.




Las políticas de seguridad son un conjunto de reglas, normas y protocolos de actuación que se encargan de velar por la seguridad de los roles y usuarios en GAM.

Se pueden definir utilizando el Backoffice web de GAM, o programáticamente usando la API de GAM.

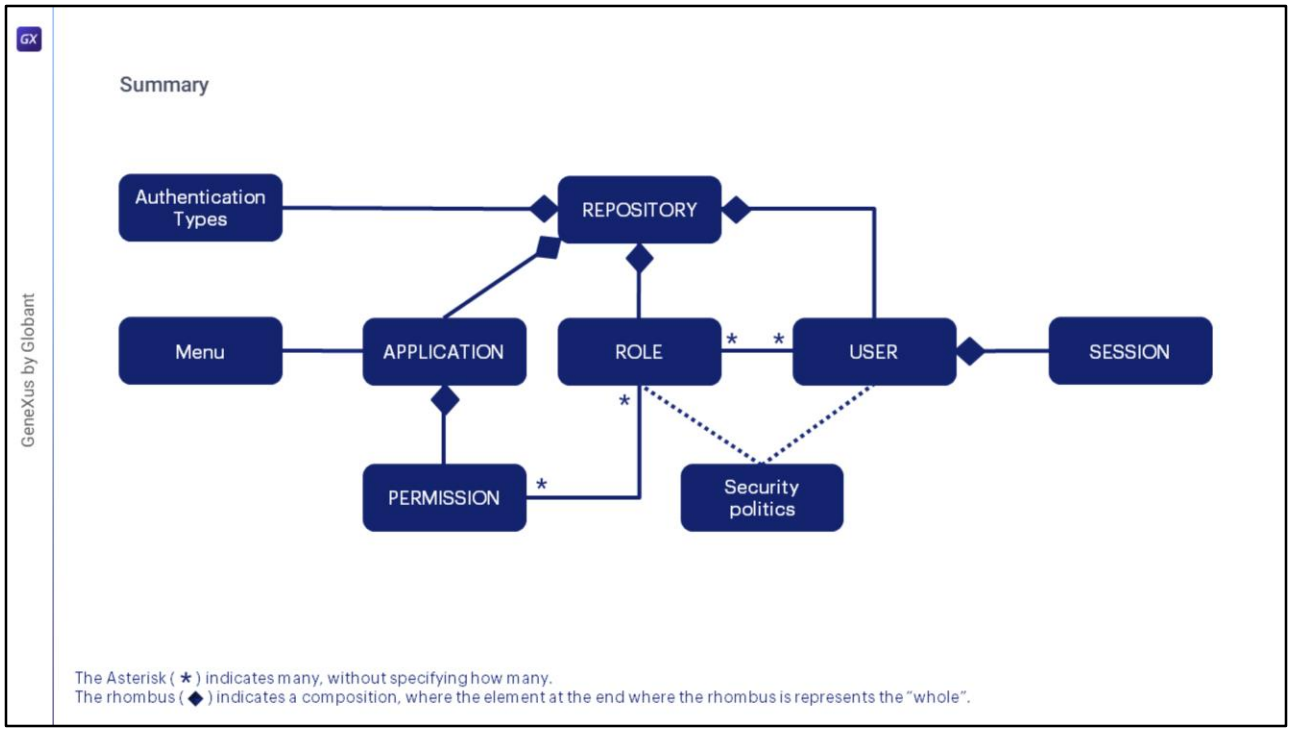
Entre las que utiliza GAM, se puede destacar el Tiempo mínimo para cambiar contraseñas, el Tiempo de espera de la sesión web, el Mínimo de determinados caracteres en contraseñas (como son los caracteres numéricos, especiales, mayúsculas y minúsculas), entre otros.




Utilizando GAM se puede definir (dinámicamente) un menú de su aplicación web o mobile en tiempo de ejecución, en base a los permisos y roles GAM del usuario logueado.

GAM devuelve la estructura del menú, dependiendo de los permisos del usuario para que esta estructura pueda cargarse en tiempo de ejecución con cualquier Control de usuario.

El menú debe definirse para una aplicación GAM en particular. Se pueden definir tantos menús como se quiera dentro de una aplicación GAM.
Una opción de menú tendrá un permiso y un recurso asociado.




Resumiendo todo lo visto hasta ahora, en el diagrama que vemos en pantalla, tenemos todos los componentes que mencionamos del GAM.

Antes de profundizar en el diagrama, en cuanto a la nomenclatura utilizada, como se indica en la slide el asterisco indica muchos, y el rombo indica una composición, donde hace referencia a que por ejemplo no pueden existir sesiones si no existe un usuario que la contenga.

Ahora si yendo al contenido del diagrama, podemos detallar que el conjunto de Permisos GAM cae dentro del alcance de una sola Aplicación y Repositorio. Por lo tanto, las Aplicaciones GAM se pueden asociar a n Permisos GAM, y estos están determinados por una Aplicación en un Repositorio.
Luego, los roles se definen dentro de un Repositorio, y están asociados a Permisos donde estos se pueden usar en muchos Roles.
Los usuarios se pueden habilitar en uno o más repositorios de GAM, y son el contenido de las sesiones. Además, pueden tener muchos roles e indirectamente asignarse uno o más permisos.

Los Tipos de Autenticación y Políticas de Seguridad están dentro del alcance de un Repositorio. Los Menús están dentro del alcance de la aplicación, que, a su vez, estas están dentro del alcance de un Repositorio


GeneXus by Globant
GeneXus™
by Globant
training.genexus.com