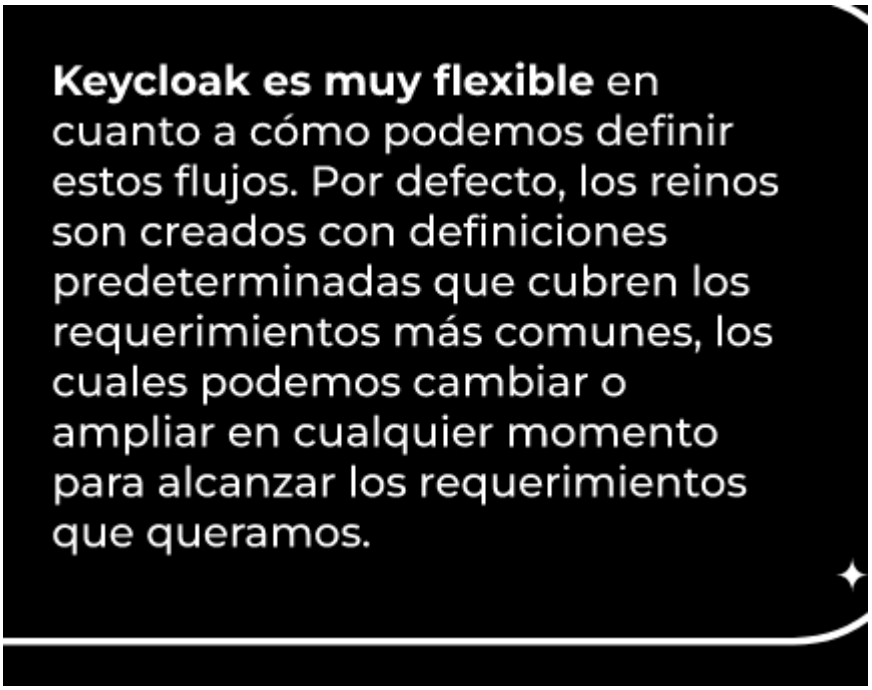
1. Keycloak no permite la modificación de los pasos de flujo de autenticación.

FALSO



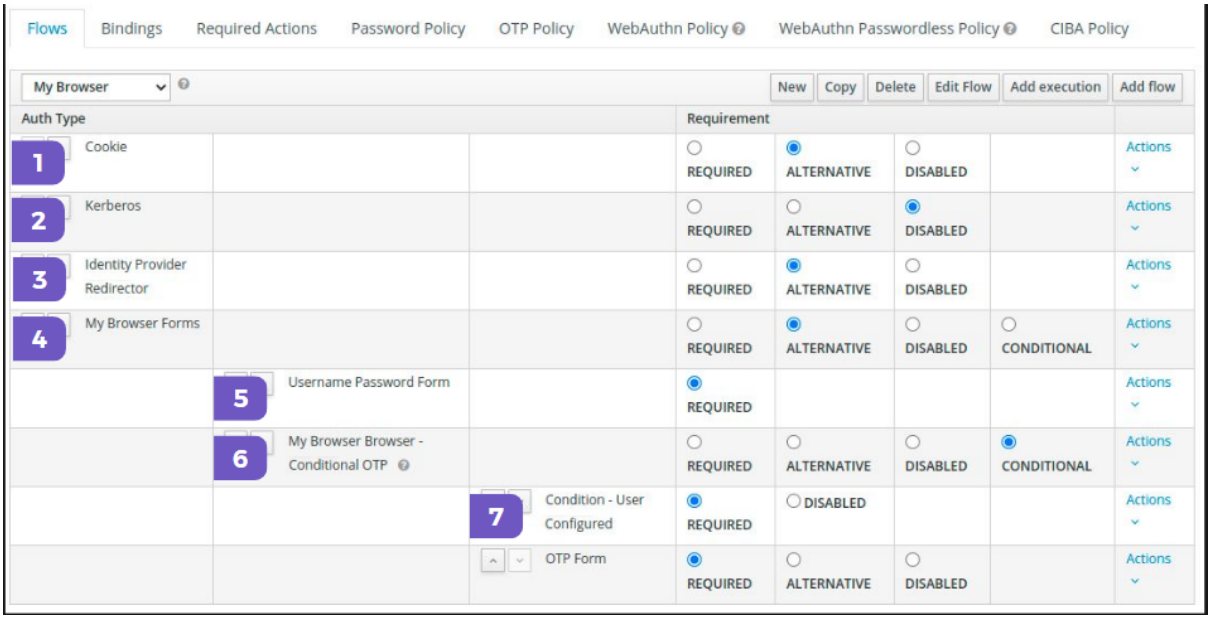

Ahora, cambiemos la forma en que los usuarios se autentican en el reino. En lugar de solicitar las credenciales usando una sola página de inicio de sesión, recopilemos el nombre de usuario y la contraseña en diferentes pasos y desde diferentes páginas. Para eso, hacemos clic en el menú "**Actions**" en el lado derecho de la ejecución del formulario de "**Username Password Form**" y hacemos clic en la opción "**Delete**". Por el momento, nuestro flujo debería tener el siguiente aspecto:

Una vez que agregamos la ejecución al flujo, deberíamos verla dentro del subflujo. Por defecto, las ejecuciones se agregan al final del flujo, pero en nuestro caso queremos que esta ejecución esté en la parte superior del subflujo para poder obtener primero el nombre de usuario. Para eso, hacemos clic en la flecha hacia arriba en el lado izquierdo de "**Username Form**" hasta que se convierta en la primera ejecución en el subflujo. Luego, repetimos los mismos pasos que acabamos de hacer para agregar la ejecución de autenticación "**Password Form**" al subflujo para obtener la contraseña y autenticar al usuario. Debemos asegurarnos de que "**Password Form**" sea la segunda ejecución en el subflujo. También, asegurémonos de que las ejecuciones de "**Username Form**" y de "**Password Form**" estén marcadas como "**REQUIRED**". Este es un paso importante, ya que obliga a nuestros usuarios finales a proporcionar ambas piezas de información cuando inician sesión en el reino. Ahora, el flujo de autenticación de "**My Browser**" debería verse así:

Como capa adicional de seguridad, Keycloak nos permite usar un segundo factor, o evidencia, al autenticar a los usuarios. Además de proporcionar una contraseña —algo que los usuarios saben—, los usuarios están obligados a proporcionar evidencia secundaria sobre su identidad —algo que tienen—, que puede ser un código o una clave de seguridad en su posesión.

**OTP** es probablemente una de las formas más comunes de habilitar 2FA para cuentas de usuario. Son relativamente fáciles de usar y agregan una capa adicional de seguridad a la hora de autenticar usuarios. Sin embargo, aunque es un método útil para **2FA**, tiene algunas desventajas. Se basa en una clave compartida entre el servidor y los usuarios. Además, no proporciona la mejor usabilidad para los usuarios finales y sigue abierto a ataques comunes como phishing o estafas. Como veremos más adelante, Keycloak nos ayuda a superar estas limitaciones usando un dispositivo de seguridad como segundo factor, aprovechando **WebAuthn**.

2.Los pasos de los flujos de autenticación son obligatorios. FALSO


3.Qué acción es fundamental para Keycloak en ambiente productivo. Como se ve en la página la única correcta era la de BD

En esta clase nos toca ahondar un poco más en configuraciones avanzadas de Keycloak, adentrándonos en un escenario más práctico y real para nuestra aplicación empresarial.

Para esto vamos a realizar las siguientes operaciones:

- Asignar un nombre de host a Keycloak.
- Habilitar TLS.
- Configurar una base de datos como repositorio.
- Habilitar el clustering.
- Configurar un proxy reverso.
- Probar el nuevo ambiente.

4. Tipos de arquitectura a secularizar:


5.Forma segura de invocar API rest externa.

La forma más segura de invocar una **API REST** externa desde una SPA es con una **API** intermedia desplegada en el mismo dominio de la SPA, delegando la gestión del cliente confidencial hacia nuestro IAM y gestión de tokens de "forma invisible" para el web browser, minimizando la posibilidad de alteración de tokens de **refresh**.

Esta arquitectura se llama **Back end for Front end (BFF)** y trata de hacer más portable la SPA, orientando la misma a la vista y su comportamiento, desligándose de lo referido a la lógica propia de la aplicación. La lógica la delega a una API intermediaria que terminará invocando a los servicios finales de negocio, por ejemplo, la imputación de un pago con tarjeta de crédito al microservicio de tarjetas.

Otra ventaja clara de esta solución es que no tendremos que lidiar o configurar **CORS** (**cross-origin resource sharing**) dado que la API intermedia está en el mismo dominio. En caso de que no lo fuese, deberíamos gestionar el permiso adecuado para poder llamar APIs de diferente dominio desde la SPA de origen.

6.La integración de seguridad con redes sociales permite:

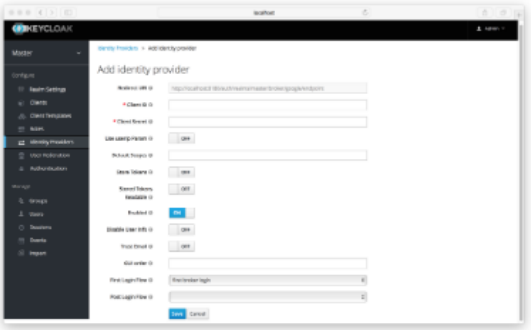
Cuando configuramos un proveedor de identidad para iniciar sesión, Keycloak actúa como intermediario, delegando el proceso de autenticación al nuevo proveedor de identidad configurado.

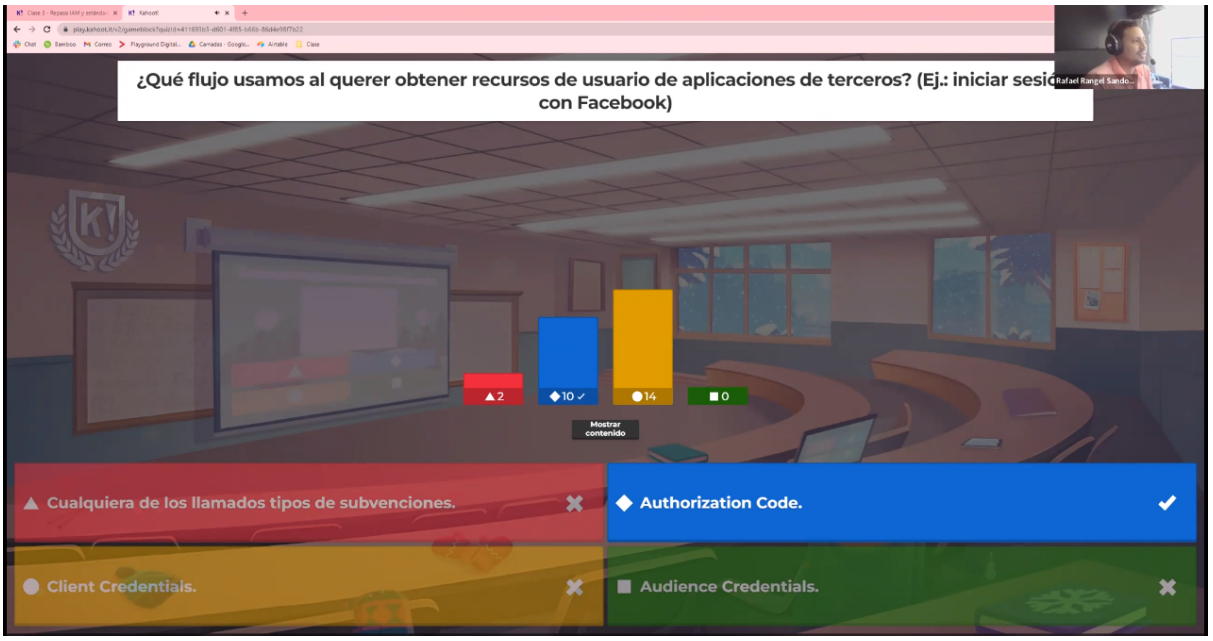
Para utilizar un proveedor de identidad tenemos que configurarlo desde "Identity Providers" en el menú de la izquierda.

7.Integración con redes sociales basadas en OpendID y OAuth. Obvimente para esto obtenemos de provedores clientid y client secret.

01 Primero, debemos ir a la opción “**Identity Providers**” en el menú de la izquierda y seleccionar “**Google**” de la lista desplegable en “**Add provider**”. Esto nos lleva a la página de “**Add identity provider**”

Pero no nos olvidemos que, para poder guardar, debemos obtener un **client ID** y un **client secret** de Google. Es importante guardar un dato de esta página: la **redirect URI** (quizás quieran copiarla al portapapeles), ya que en el proceso de registrar Keycloak en la página de Google, nos solicitará este dato.
Para habilitar el login con Google, primero debemos crear un proyecto y un cliente en **Google Developer Console**, para ello debes hacer el proceso de sign-in a **Google Clouds** (si es que todavía no lo hiciste en la clase 10).




8.Cluster de varios nodos. También está explicado en el minuto 1:40 del video resumen. El DNS está declarado en el proxy-pass.

## Configuración de proxy reverso

Un proxy reverso es un componente clave para alcanzar HA en producción, ya que permite acceder de forma transparente a múltiples instancias de Keycloak redistribuyendo la carga.

Por seguridad, las instancias de Keycloak estarán en una red privada, por lo que se necesitará un proxy para poder acceder a ellas. Dado que el cliente final no conoce cada instancia de Keycloak, sino que conoce el punto común de ingreso por su proxy reverso, podemos agregar o remover instancias en el clúster de Keycloak sin afectar la disponibilidad de este.

Se puede utilizar cualquier proveedor de proxy reverso, como **NGINX, F5** y **HA Proxy.** Independientemente del proveedor, los siguientes requerimientos son fundamentales:

- TLS y encriptación
- Balanceo de carga
- Afinidad de sesión
- Redirección de datos de cliente

## Balanceo de carga

En nuestro ejemplo, vamos a utilizar el proxy reverso **NGINX**, donde definiremos los nodos que conformaran el clúster de Keycloak dentro del archivo **/conf/nginx.conf** que se encuentra en nuestra carpeta de descarga del proxy.

```
upstream mykeycloak {
    server localhost:8443 fail_timeout=2s;
    server localhost:8553 fail_timeout=2s;
}

server {
   listen       8000;
   location / {
                proxy_pass https://mykeycloak;
    }
}
```

La cantidad de nodos que definamos dependerá de la carga esperada de nuestro Keycloak y será lo que se defina en NGINX, no necesitamos ninguna configuración adicional en nuestro Keycloak.

Podemos observar que la configuración del upstream "mykeycloak" indica los nodos que conforman el clúster, mientras que en la configuración **server** se indica qué puerto atiende nuestro proxy y bajo qué URL o **proxy_pass** tomará los nodos del upstream "mykeycloak".

9.Intercambio constante de usuario y contraseña.

## Antipatrones de integración de seguridad con IAM

**Podemos vernos tentados a desarrollar una pantalla de login en nuestra aplicación e intercambiar de forma constante el usuario y password por un token en cada petición de un recurso del servidor con Keycloak. Sin embargo, esto es un grave error. Con la simple intercepción de esta información, tendremos acceso a todos los recursos que el usuario pueda ver dentro de la organización. Además, no podremos agregar otras funcionalidades del IAM, como autenticación de dos factores.**

10. Mi respuesta personal a un ejercicio que está mal redactado.

*"1.Las consideraciones que se deben tener en cuenta para implementar Keycloak en producción deben ser: asignar un nombre de Host a Keycloak (establecemos las URLs para Back, Front y administración de la consola); habilitar TLS (la comunicación se establecerá*

*bajo el protocolo HTTPS); configurar una base de datos como repositorio (esto incluye instalar el driver JDBC elegido y configurarlo y establecer los atributos para crear una conexión con la BD elegida); habilitar el clustering (levantar el servidor en modo HA y la distribución de la caché); configurar un proxy reverso(configurar el balanceo de carga, sesiones de tipo Sticky y el redireccionamiento de las cabeceras en las peticiones). Implementamos estas acciones con la finalidad de que la infraestructura de produccion sea tolerante a fallos, disponga de alta escalabilidad y la base de datos sea consistente.*
*2.El ms de cuentas debería contar con una clase de configuración donde se limite el consumo a usuarios autenticados, una clase que pueda interceptar el token devuelto por el IAM y leer sus respectivos claims, y además, cada endpoint en particular deberia protegerse con validaciones (del tipo @PreAuthorized) donde se especifique el grupo, scope o rol que tiene acceso al recurso, en este caso, administrador.*
*”*

Estas respuestas están elaboradas con con la siguiente información:

## Introducción

Hasta el momento vimos diferentes aspectos de seguridad, revisamos temas a nivel conceptual, pero también vimos su implementación según las diferentes arquitecturas que pueda tener el sistema. Además, fuimos volcando lo aprendido en un ejercicio productivo real y logramos que este sistema (siempre apoyándose en Keycloak para la gestión de seguridad) mantenga una estabilidad óptima.

En esta clase nos toca ahondar un poco más en configuraciones avanzadas de Keycloak, adentrándonos en un escenario más práctico y real para nuestra aplicación empresarial.

Para esto vamos a realizar las siguientes operaciones:

- Asignar un nombre de host a Keycloak.
- Habilitar TLS.
- Configurar una base de datos como repositorio.
- Habilitar el clustering.
- Configurar un proxy reverso.
- Probar el nuevo ambiente.



y la información de cada sección en particular de la clase 22A.

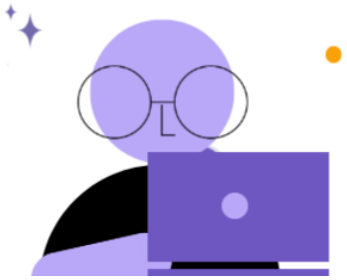
La segunda pregunta está redactada a partir de lo visto en la clase8A y mi experiencia en el proyecto integrador en el cual trabaje desde el inicio de la cursada

**Esta es la típica clase de configuración de la seguridad en donde definimos cuáles endpoints queremos que tengan seguridad y cuáles no. En nuestro caso, todos los endpoints requerirán estar autenticados para poder ser consumidos.**

En ambos métodos recorremos el jwt, que lo tenemos en una clase JsonNode, y obtenemos los roles y la audiencia.

# Configuración en los endpoints

Tenemos cuatro posibles endpoints para consumir, cada uno de ellos tiene validaciones diferentes.



```
{
@RestController
@RequestMapping("/hello")
public class HelloRestController {
@GetMapping("/user")
public String hello() {
  return "hello";
}

  @GetMapping("/role")
+ @PreAuthorize("hasRole('USER')")
  public String withRole() {
    return "Only for User role";+
  }

  @GetMapping("/scope")
  @PreAuthorize("hasAnyAuthority + ('SCOPE_publish')")
  public String withScope() {
    return "Only for Scope Publish";+
  }

  @GetMapping("/aud")
  @PreAuthorize("hasAnyAuthority('AUD_account')")
  public String withAud() {
    return "Only for aud account";+
  }
}
```