DISEÑO DE SISTEMAS

Trabajo Práctico Anual Integrador - Implementación SSO

Comisión:
K3001 - K3101

Profesor:
Ezequiel Escobar

Ayudantes:
Alberio, Valentina
Sandoval, Aylén Martina

Grupo:
Nro. 3

Alumnos:
Barbieri, Tomás Iván (209.945-7)
Benitez, Camila Nahir (208.898-8)
Hygonenq, Federico (203.730-0)
Szapari Marcos, Marco (209.202-5)
Vazquez, Lucas Manuel (209.224-4)

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Universidad Tecnológica Nacional (FRBA) - 2024

Comparación y Elección del SSO

Single Sign-On (SSO): Es un Componente que permite tener acceso a múltiples aplicaciones ingresando solo con una Cuenta a los diferentes sistemas y Recursos.

Es posible acceder mediante una única “contraseña” y se desea evitar el ingreso repetitivo de estas cada vez que el Usuario se desconecte del Servicio.

Para los Usuarios supone una gran comodidad, ya que identificándose solo una vez es posible mantener la Sesión válida para el resto de las aplicaciones que hacen uso del SSO.

Acelera el acceso de los Usuarios a sus aplicaciones; reduce la carga de memorizar diversas Contraseñas; es fácil de implementar y conectar a nuevas fuentes de datos; al fallar el SSO, se pierde acceso a todos los sistemas relacionados; suplementación de Identidades en los accesos externos de los Usuarios.

Comparación de 5 opciones del mercado:

1. Okta

- Ventajas:
  - Interfaz fácil de usar: Panel de control intuitivo para administradores y usuarios finales.
  - Alta compatibilidad: Soporta una gran cantidad de aplicaciones y protocolos como SAML, OAuth y OpenID Connect.
  - Foco en seguridad: Funciones avanzadas como autenticación adaptativa basada en contexto y reglas de acceso granular.
  - Escalabilidad: Adecuado para organizaciones de cualquier tamaño, desde startups hasta grandes empresas.
  - Disponibilidad multiplataforma: Integra bien con aplicaciones móviles y de escritorio.
- Desventajas:
  - Costo: Puede ser caro para empresas pequeñas.
  - Curva de aprendizaje para características avanzadas: Configurar integraciones complejas puede requerir tiempo.
  - Dependencia de internet: Necesita conexión constante para la autenticación.

2. Microsoft Azure Active Directory (Azure AD)

- Ventajas:
  - Integración con Microsoft: Funciona perfectamente con productos como Office 365, SharePoint, y Microsoft Teams.
  - Seguridad robusta: Funciones avanzadas como acceso condicional y MFA (autenticación multifactorial).
  - Compatibilidad empresarial: Soporta integraciones con una variedad de aplicaciones empresariales de terceros.
  - Precios competitivos: Los paquetes base son asequibles y escalables.
  - Soporte híbrido: Excelente para organizaciones que manejan entornos híbridos (nube y on-premise).
- Desventajas:

    - Configuración compleja: La configuración inicial puede ser complicada, especialmente para empresas que no están completamente en el ecosistema de Microsoft.
    - Dependencia del ecosistema Microsoft: Es más beneficioso si ya utilizas productos de Microsoft.
    - Costos adicionales por características avanzadas: Algunas funciones avanzadas requieren niveles de licencia más caros.

3. Google Workspace (SSO)

- Ventajas:
    - Simplicidad de uso: Ideal para empresas ya integradas en el ecosistema de Google.
    - Precio accesible: Incluido en la suscripción a Google Workspace.
    - Soporte para OAuth y SAML: Compatible con muchas aplicaciones externas.
    - Fácil implementación: Ideal para pequeñas y medianas empresas.
- Desventajas:
    - Limitaciones de integración: No soporta tantas aplicaciones externas como Okta o Azure AD.
    - Foco en entornos de Google: Es más útil si ya utilizas Google Workspace.
    - Falta de características avanzadas: Menos personalización y configuraciones avanzadas en comparación con soluciones específicas de SSO.

4. Ping Identity

- Ventajas:
    - Alta personalización: Funciona bien en entornos complejos y personalizados.
    - Compatibilidad con múltiples protocolos: SAML, OAuth, OpenID Connect, entre otros.
    - Seguridad avanzada: Incluye autenticación adaptativa y acceso condicional basado en riesgo.
    - Escalabilidad: Adecuado para grandes empresas con necesidades complejas.
- Desventajas:
    - Curva de aprendizaje pronunciada: Requiere experiencia técnica para aprovechar al máximo sus características.
    - Costo elevado: Es más costoso que algunas alternativas, especialmente para empresas pequeñas.
    - Configuración inicial compleja: Puede requerir consultoría profesional para implementaciones grandes.

5. Auth0

- Ventajas:
    - Desarrollador-friendly: Ofrece APIs y SDKs bien documentados para personalización.
    - Escalabilidad técnica: Ideal para desarrolladores que necesitan construir soluciones específicas.
    - Compatibilidad con múltiples protocolos: Incluye OAuth, OpenID Connect, y SAML.

    - Soporte para aplicaciones B2C y B2B: Excelente para empresas que necesitan gestionar acceso para clientes.
- Desventajas:
    - Costo variable: Puede volverse caro rápidamente si la base de usuarios crece.
    - Menos amigable para no técnicos: Está diseñado pensando en desarrolladores, lo que puede ser complicado para usuarios no técnicos.
    - Complejidad inicial: Requiere esfuerzo para configurar y adaptar a necesidades específicas.

Cuadro Comparativo Resumido

Opción	Ventajas Principales	Desventajas Principales	Mejor para…
Okta	Interfaz intuitiva, amplia compatibilidad.	Costoso, curva de aprendizaje.	Empresas de cualquier tamaño.
Azure AD	Integración Microsoft, seguridad avanzada.	Complejo fuera del ecosistema Microsoft.	Organizaciones híbridas y usuarios de Microsoft.
Google SSO	Fácil de usar, accesible.	Limitado a Google y menos personalizable.	PyMES y ecosistemas Google.
Ping	Alta personalización, seguridad robusta.	Costoso, curva de aprendizaje.	Grandes empresas y entornos complejos.
Auth0	Personalización técnica, APIs sólidas.	Costos variables, técnico.	Startups técnicas y desarrolladores.

Conclusión: Auth0 nos pareció una solución ideal para organizaciones que buscan una plataforma flexible y escalable para gestionar autenticaciones, especialmente por su enfoque centrado en desarrolladores. A diferencia de las otras opciones analizadas, Auth0 ofrece un mayor nivel de personalización gracias a la buena documentación de sus APIs, lo que nos permite construir experiencias de autenticación únicas y completamente integradas en sus aplicaciones. Esto lo hace una opción especialmente poderosa para un trabajo como el solicitado por la ONG, donde la experiencia del usuario final es una prioridad.

Otra gran ventaja de Auth0 es su compatibilidad con múltiples protocolos de autenticación, como OAuth, OpenID Connect y SAML, lo que facilita su integración con casi cualquier sistema o aplicación. A diferencia de soluciones más generalistas como Google Workspace SSO o Microsoft Azure AD, Auth0 no está limitado a un ecosistema específico, lo que brinda una mayor libertad para combinar tecnologías y plataformas según las necesidades que surjan. Además, su capacidad para gestionar usuarios externos de manera eficiente lo diferencia de opciones como Okta, que están más orientadas al entorno corporativo interno.

Además, su flexibilidad y capacidad para escalar lo convierten en una opción superior para organizaciones que necesitan una solución de autenticación avanzada, adaptable y preparada para el futuro. En comparación con opciones más tradicionales o con configuraciones rígidas, Auth0 permite y facilita diseñar y controlar completamente los flujos

de inicio de sesión de la organización, garantizando una experiencia segura y optimizada para sus usuarios.

Propuesta de Implementación en TPA

Desde el punto de vista del **usuario**, se encontraría con la página de inicio de sesión, con el diseño que ya implementamos, pero debajo de las opciones para rellenar con su usuario o mail y contraseña, tendría un botón con el símbolo de Google, lo que le daría la opción de ingresar utilizando su cuenta de Google. Esto le generará un token temporal, por lo que si cierra sesión o pasa más de un tiempo determinado (por nosotros), deberá volver a ingresar, ya sea con esta opción o rellenando los datos.

Si bien Auth0 nos permite utilizar otras cuentas aparte de la de Google, como son Facebook o Github, consideramos que la cuenta de Google es más coherente y común. Otras cuentas como la de Facebook podrían ser muy personales; o en el caso de Github, poco "coherentes". Por lo tanto, nos apegamos a una opción ampliamente utilizada y aceptada por los usuarios en las redes.

En cuanto al punto de vista de los **desarrolladores** (es decir, nosotros), comenzamos utilizando las dependencias correspondientes en Maven, y pasamos a implementar un archivo "*web.xml*", el cual nos permite manejar de forma más sencilla las credenciales de uso que nos facilita Auth0 para la implementación del SSO; estas son: Dominio, ID del Cliente y Secreto de Cliente.

Por otro lado, en cuanto a las clases que utilizaríamos:

- **Auth0Filter:** Es un filtro que agrega una capa de seguridad, ya que se usa para verificar si un usuario está autenticado antes de permitirle el acceso a ciertas rutas específicas de nuestra aplicación web.
- **AuthenticationControllerProvider:** Es una clase de utilidad que implementa el patrón Singleton para gestionar una única instancia compartida de AuthenticationController y un proveedor de claves públicas (JwkProvider). Este diseño se usa para manejar autenticación con Auth0, optimizando eficientemente el uso de recursos.

Pero además, hay clases como los **Servlets** (*Callback*, *Home*, *Login* y *Logout*) que no utilizaríamos por ser innecesarias; en su lugar consideramos posible una implementación a partir de Javalin. Esto se debe a que Javalin nos permite manejar las rutas y la lógica con nuestras propias configuraciones, siempre y cuando integremos correctamente los endpoints necesarios para el flujo de autenticación. A continuación, breve explicación de la funcionalidad de los **Servlets**:

- **LoginServlet:** Se invoca cuando el usuario intenta iniciar sesión. Usa el Dominio y el ID de Cliente para crear una URL de autorización válida y redirige al usuario ahí.
- **CallbackServlet:** Captura solicitudes a nuestra URL de Callback y procesa los datos para obtener las credenciales. Después de iniciar sesión correctamente, las credenciales se guardan en la sesión HTTP de la solicitud.
- **HomeServlet:** Lee los tokens guardados previamente y los muestra en el recurso *home*.
- **LogoutServlet:** Se invoca cuando el usuario hace click en el enlace de cierre de sesión. Invalida la sesión del usuario y lo redirige a la página de inicio de sesión, manejada por *LoginServlet*.