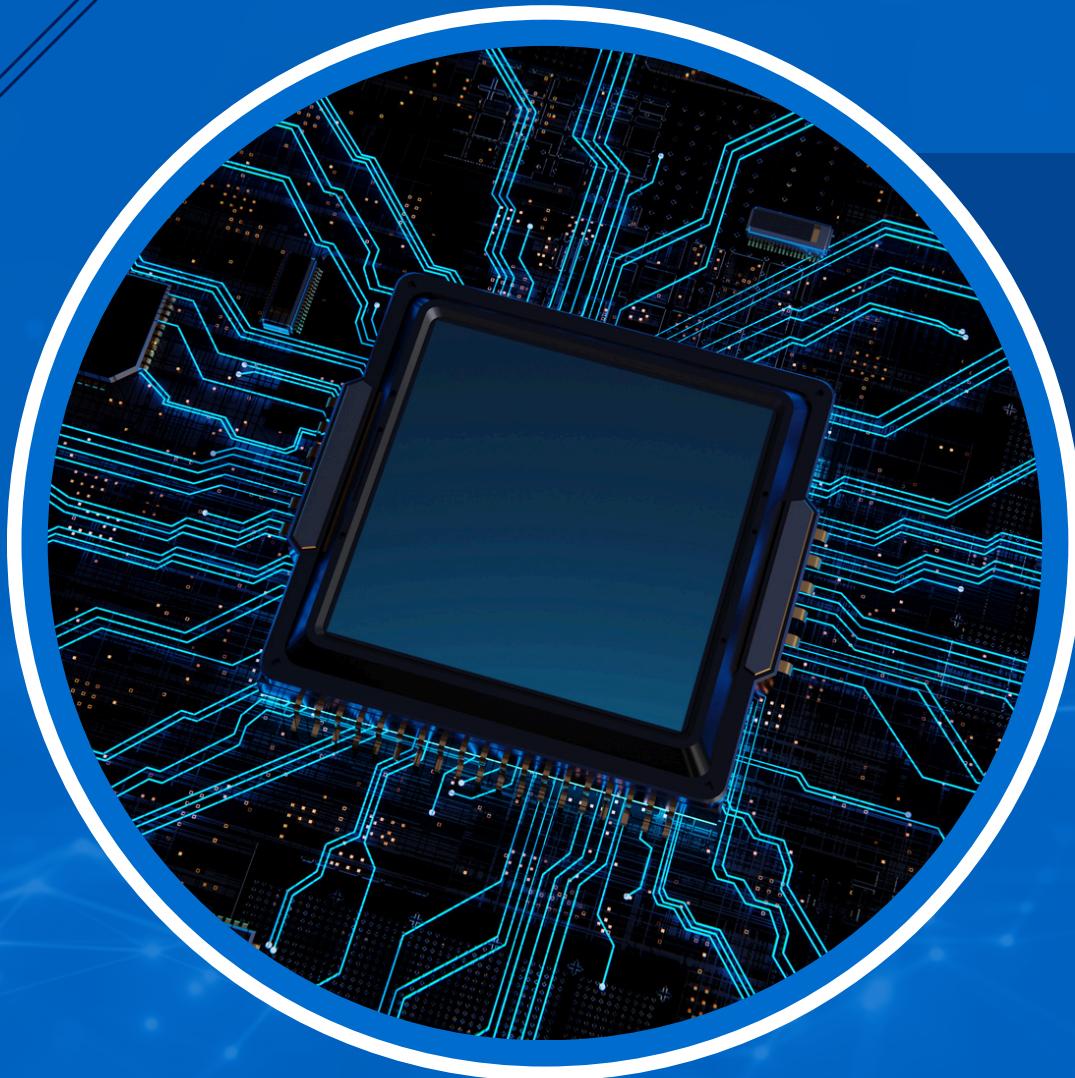


SEGURIDAD EN APLICACIONES WEB



La seguridad en aplicaciones web se refiere a las prácticas, técnicas y medidas implementadas para proteger una aplicación web de amenazas y vulnerabilidades que pueden comprometer la integridad, confidencialidad y disponibilidad de los datos y sistemas.





■ Control de Acceso Roto

Permite que los usuarios accedan a recursos o realicen acciones fuera de sus permisos.

■ Fallas Criptográficas

Uso incorrecto o insuficiente de cifrado, exponiendo datos sensibles.

■ Inyección de Código

Un atacante inyecta código malicioso en una consulta o script.

■ Diseño Inseguro

Falta de enfoque en la seguridad desde el diseño de la aplicación.



TOP 10 - VULNERABILIDAD

Publicado por: Open Web
Application Security Project





■ Configuración Incorrecta de Seguridad

Uso de configuraciones predeterminadas o mal gestionadas.

■ Fallas Criptográficas

Uso incorrecto o insuficiente de cifrado, exponiendo datos sensibles.

■ Componentes Vulnerables y Desactualizados

Uso de bibliotecas, frameworks o sistemas sin actualizar con vulnerabilidades conocidas.

■ Fallos en Identificación y Autenticación

Fallos en mecanismos de autenticación y gestión de sesiones.



TOP 10 - VULNERABILIDAD

Publicado por: Open Web
Application Security Project





■ Fallas de Integridad en Software y Datos

Uso de software sin verificar su autenticidad o sin proteger su integridad.

■ Fallos en Registro y Monitoreo de Seguridad

Falta de logs y monitoreo que permitan detectar ataques o actividad sospechosa.

■ SSRF – Falsificación de Petición del Lado del Servidor

Un atacante hace que el servidor realice solicitudes no autorizadas



TOP 10 - VULNERABILIDAD

Publicado por: Open Web
Application Security Project





■ ¿Qué es?

JSON Web Token (JWT) es un estándar abierto (RFC 7519) para la creación de tokens compactos y seguros que permiten la autenticación y el intercambio de información entre partes. Estos tokens están firmados digitalmente, lo que garantiza su integridad y seguridad.

■ Estructura de un JWT

Header (Encabezado)

- Indica el algoritmo de firma (ej. HS256, RS256).

Payload (Carga útil)

- Contiene los datos (información sobre el usuario o sesión).

Signature (Firma)

- Se genera combinando el header, el payload y una clave secreta o clave privada.
- Se usa para garantizar que el token no haya sido alterado.

JSON WEB TOKEN (JWT)



USO DE JWT EN AUTENTICACIÓN

■ Inicio de sesión (Login)

- El usuario envía sus credenciales al servidor.
- El servidor las valida y genera un JWT con los datos del usuario.
- El JWT se envía al cliente.

■ Autorización en solicitudes

- El cliente almacena el JWT (usualmente en localStorage o sessionStorage).
- En cada solicitud, el cliente envía el JWT en el header Authorization
- El servidor verifica la firma y, si el token es válido, permite el acceso.

■ Expiración y Renovación

Un JWT tiene una expiración, después de la cual el usuario debe autenticarse nuevamente o usar un refresh token para obtener uno nuevo.



EJEMPLO DE USO EN: BACKEND_CON_JWT

RBAC (ROLE-BASED ACCESS CONTROL)

CONTROL DE ACCESO BASADO EN ROLES

Es un modelo de control de acceso en el que los permisos y privilegios se asignan a los roles, y los usuarios obtienen acceso según el rol que se les asigne. En lugar de otorgar permisos individuales a cada usuario, se agrupan en roles, lo que facilita la gestión de seguridad.



RBAC (ROLE-BASED ACCESS CONTROL)

🔍 Ejemplo de RBAC :

Sistema de gestión de empleados para una empresa con un sistema en el que hay tres tipos de usuarios:

Administrador → Puede gestionar empleados, asignar roles y ver todos los datos.

Gerente → Puede ver y modificar información de los empleados de su departamento.

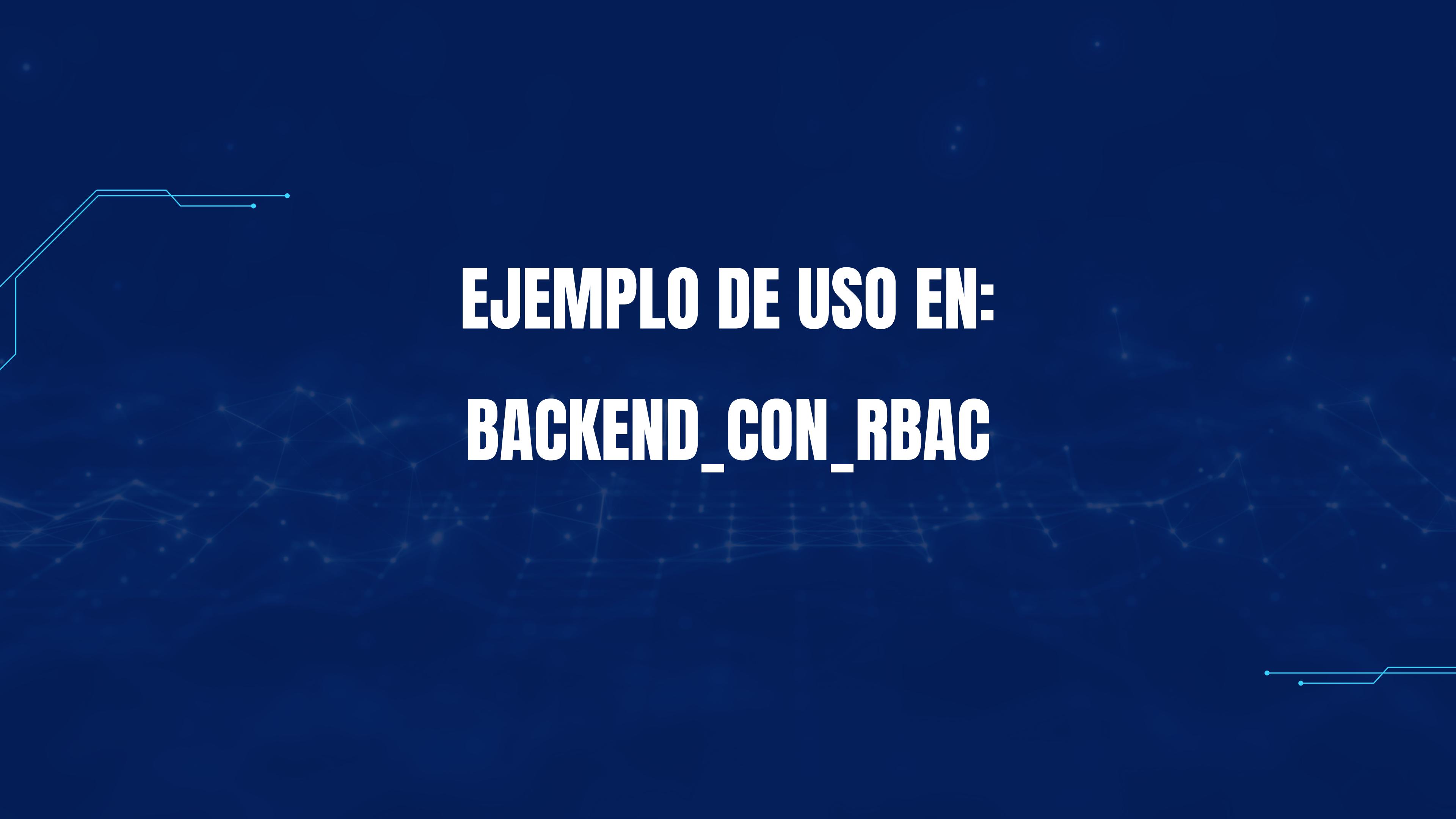
Empleado → Solo puede ver su propia información.



Rol	Ver Datos	Modificar Datos	Gestionar Usuarios
Administrador	✓	✓	✓
Gerente	✓ (limitado)	✓ (limitado)	✗
Empleado	✓ (Propios)	✗	✗

DIFERENCIAS CLAVE ENTRE ACL Y RBAC

Características	ACL (Access Control List)	RBAC (Role-Based Access Control)
Concepto	Control basado en listas de permisos individuales por usuario/recurso.	Control basado en asignación de roles con permisos predefinidos.
Asignación de permisos	Se otorgan permisos directamente a cada usuario o grupo en un recurso específico.	Se otorgan permisos a roles, y los usuarios obtienen permisos al recibir un rol.
Escalabilidad	Difícil de administrar en sistemas grandes, ya que cada usuario puede tener permisos específicos en múltiples recursos.	Más escalable, ya que los permisos se asignan a roles en lugar de a usuarios individuales.
Administración	Más compleja, ya que cada usuario puede requerir configuraciones individuales.	Más sencilla, ya que los permisos se gestionan a nivel de rol y no por usuario.
Flexibilidad	Más detallado, útil en entornos donde cada usuario necesita permisos específicos.	Más estructurado y organizado para empresas o grandes sistemas.



EJEMPLO DE USO EN: BACKEND_CON_RBAC