Solución Configurar reglas de acceso en Azure Entra ID

1. **Registrar aplicación en Azure AD**

Primero, registrar la aplicación en Azure Entra ID para permitir la autenticación.

1. Ir al **Portal de Azure** → **Azure Entra ID**.
2. En el menú lateral, selecciona **"Registros de aplicaciones"**.
3. Haz clic en **"Nuevo registro"**.
   * **Nombre:** Poner un nombre para la app (ej., "MiAppSegura").
   * **Tipos de cuenta:** Seleccionar "Cuentas en este directorio organizacional" (o según necesidades).
   * **URI de redirección:** Si la app usa autenticación OAuth/OpenID Connect, añadir la URL de redirección (ej., <https://miapp.com/auth>).
4. Guardar los cambios y tomar nota del **ID de la aplicación (client ID)** y el **ID del directorio (tenant ID)**.

Referencia: <https://learn.microsoft.com/es-es/entra/identity/role-based-access-control/custom-create#asignaci-n-de-un-rol-personalizado-con-mbito-de-un-recurso>

1. Configurar grupos en Azure Entra ID

Definir grupos de seguridad en Azure Entra ID para organizar a los usuarios según sus roles.

1. En Azure Entra ID seleccionar "Grupos" en el menú lateral.
2. Hacer clic en "Nuevo grupo".
   1. Tipo de grupo: Selecciona Seguridad.
   2. Nombre: Por ejemplo, "SG-Directors", "SG-Managers" o ”SG-Officers”.
   3. Rol Microsoft Entra puede ser asignado al grupo: Si
   4. Dueño: La cuenta de administrador (la tuya/mía), considero se puede omitir.
   5. Miembros: Agregar los usuarios que desea incluir en este grupo.
3. Repetir proceso para crear “SG-Anonymous” con las salvedades del caso.

Referencia: <https://learn.microsoft.com/es-es/entra/identity/role-based-access-control/groups-create-eligible?tabs=ms-powershell#microsoft-entra-admin-center>

1. Crear roles en la aplicación

Azure Entra ID permite usar roles personalizados para definir permisos dentro de tu aplicación.

1. Crear roles en el manifiesto de la aplicación
   * Ve a "Registros de aplicaciones" → seleccionar la aplicación.
   * En el menú lateral, seleccionar "Manifiesto".
   * Agregar roles personalizados al manifiesto JSON. Ejemplo:

"appRoles": [  
{  
 "allowedMemberTypes": [ "Group" ],  
 "description": "Permite acceso al panel de administración",   
 "displayName": "Administrador",  
 "id": "11111111-1111-1111-1111-111111111111",  
 "isEnabled": true,  
 "lang": null,  
 "origin": "Application",  
 "value": "Admin"

},

{

"allowedMemberTypes": [ "Group" ],

"description": "Permite acceso de usuario estándar",

"displayName": "Usuario",

"id": "11111111-1111-1111-1111-111111111111",

"isEnabled": true,

"lang": null,

"origin": "Application",

"value": "User"

}

]

* Guardar el manifiesto.

Referencia: <https://learn.microsoft.com/es-es/entra/identity-platform/howto-add-app-roles-in-apps#app-roles-ui>

1. Asignar roles a los usuarios o grupos

Una vez definidos los roles, asignar estos roles a los grupos o usuarios.

1. Volver a "Registros de aplicaciones" → seleccionar la aplicación.
2. En el menú lateral, seleccionar "**Usuarios y grupos**".
3. Hacer clic en "**Agregar asignación**".
   * Seleccionar el grupo o usuario al que quieres asignar el rol.
   * Escoger el rol (por ejemplo, "SG-Directors", "SG-Managers" o ”SG-Officers”).
4. Guardar los cambios.

Referencias: <https://learn.microsoft.com/es-es/azure/role-based-access-control/role-assignments-portal>  
<https://learn.microsoft.com/es-es/entra/identity/role-based-access-control/groups-assign-role?tabs=ms-powershell>

1. Configurar la aplicación para validar los roles

En el código, asegurar que la aplicación valide los roles asignados por Azure Entra ID. Ejemplos en ASP.NET Core y Node.js:

**Ejemplo en ASP.NET Core**

1. Asegurarse de tener instalado el paquete de Microsoft.Identity.Web.
2. En tu Startup.cs o Program.cs:

services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)

.AddMicrosoftIdentityWebApi(options =>

{

Configuration.Bind("AzureAd", options);

options.TokenValidationParameters.RoleClaimType = "roles";

}, options => { Configuration.Bind("AzureAd", options); });

services.AddAuthorization(options =>

{

options.AddPolicy("AdminPolicy", policy => policy.RequireRole("Admin"));

options.AddPolicy("UserPolicy", policy => policy.RequireRole("User"));

});

1. Proteger los controladores o endpoints con los roles:

[Authorize(Roles = "Admin")]

public IActionResult AdminEndpoint()

{

return Ok("Este endpoint es solo para administradores.");

}

[Authorize(Roles = "User")]

public IActionResult UserEndpoint()

{

return Ok("Este endpoint es para usuarios estándar.");

}

**Ejemplo en Node.js**

1. Usar el paquete passport-azure-ad para manejar la autenticación.
2. Validar los roles desde el token JWT.

const passport = require('passport');

const BearerStrategy = require('passport-azure-ad').BearerStrategy;

const options = {

identityMetadata: `https://login.microsoftonline.com/<tenantID>/v2.0/.well-known/openid-configuration`,

clientID: '<clientID>',

audience: '<clientID>',

validateIssuer: true,

passReqToCallback: false,

};

passport.use(new BearerStrategy(options, (token, done) => {

if (token.roles.includes("Admin")) {

return done(null, token, { role: "Admin" });

} else if (token.roles.includes("User")) {

return done(null, token, { role: "User" });

} else {

return done(null, false);

}

}));

app.use(passport.initialize());

app.get('/admin', passport.authenticate('oauth-bearer', { session: false }),

(req, res) => {

if (req.authInfo.role === "Admin") {

res.send("Bienvenido, administrador.");

} else {

res.status(403).send("Acceso denegado.");

}

});

1. Probar el acceso
2. Usar herramientas como Postman o tu frontend para realizar pruebas.
3. Autenticarse con un usuario que pertenezca a un grupo/rol específico y verificar que:

* Los endpoints protegidos respondan correctamente según el rol.
* El acceso anónimo esté permitido en los endpoints públicos.