**Integrar OnlyOffice a .Net Framework**

1. Contexto de Seguridad: ONLYOFFICE requiere JWT

**Texto

El contenido generado por IA puede ser incorrecto.**

Esto implica que:

* Cada request del navegador y del backend debe llevar un JWT firmado con el secreto u2QNSuIFTxXPte5LUYG1u8WWlwuBwdS1.
* Si no lo envías, tendrás errores como “Invalid token”, “No token provided”, o el editor ni siquiera cargará.

**2. ¿Qué debes hacer en tu integración?**

1. **Generar el JWT en tu backend (ASP.NET)**

Debes firmar un JWT con el payload adecuado y ese secreto cada vez que generes la configuración para el editor OnlyOffice.

Recomendación:

* Usa la librería JWT más común para .NET: System.IdentityModel.Tokens.Jwt
* O, si usas .NET Framework, también puedes usar Jose.JWT.

1. **Agregar el token en la configuración del editor**

El objeto de configuración que entregas al frontend debe incluir el token, por ejemplo:

Pantalla de computadora con fondo negro

El contenido generado por IA puede ser incorrecto.{

...,

"document": {...},

"editorConfig": {...},

"token": "<AQUÍ\_EL\_JWT>"

}

1. **El token también debe ir en los headers de las requests entre tu servidor y el DocumentServer**

Esto normalmente lo maneja OnlyOffice automáticamente si pones bien el token en la configuración.

**3. ¿Cómo generar el JWT en C#?**

Aquí tienes un ejemplo usando Jose.JWT (puedes instalarlo con NuGet):

using Jose;

using System;

using System.Collections.Generic;

public string GenerarOnlyOfficeToken(object payload)

{

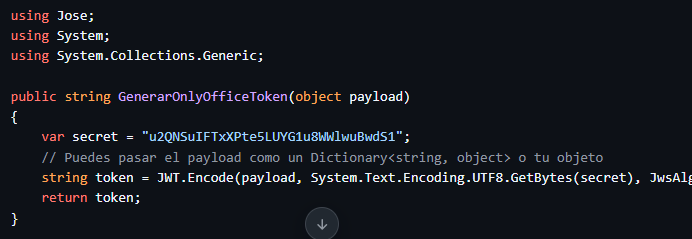
var secret = "u2QNSuIFTxXPte5LUYG1u8WWlwuBwdS1";

// Puedes pasar el payload como un Dictionary<string, object> o tu objeto

string token = JWT.Encode(payload, System.Text.Encoding.UTF8.GetBytes(secret), JwsAlgorithm.HS256);

return token;

}



**El payload debe incluir al menos lo mismo que mandas en tu configuración:**

var payload = new Dictionary<string, object>

{

{ "document", new {

fileType = extensionArchivo.Replace(".", ""),

key = uniqueKey,

title = nombreArchivo,

url = rutaArchivo,

permissions = new { edit = true }

}},

{ "editorConfig", new {

lang = "es",

mode = "edit"

}}

// ... agrega más campos si tu config lo tiene

};

var token = GenerarOnlyOfficeToken(payload);

Pantalla de computadora con letras

El contenido generado por IA puede ser incorrecto.

**4. Modifica tu función para enviar el token**

Cuando devuelvas la configuración al frontend, envía el token:

var config = new {

// tu config...

token = token

};

return Json(config, JsonRequestBehavior.AllowGet);

Texto

El contenido generado por IA puede ser incorrecto.

**5. En el frontend JS, pásale el token a DocsAPI.DocEditor**

Asegúrate de que tu objeto de configuración JS tenga el campo "token": "...":

var docEditor = new DocsAPI.DocEditor("onlyoffice-editor", {

// ...

token: config.token,

// ...

});

Texto

El contenido generado por IA puede ser incorrecto.

**6. CORS**

Tu configuración CORS en Web.config es correcta, pero recuerda que también debes permitir OPTIONS (preflight) y que los puertos internos sean accesibles entre tu app y el Docker.

**7. Resumen visual de flujo**

1. Usuario pide visualizar documento.
2. Tu backend arma la configuración, genera el JWT y lo pone en el JSON.
3. El frontend recibe la configuración, toma el token y lo pasa a DocsAPI.DocEditor.
4. OnlyOffice valida el token y permite la edición/vista.

**8. Recomendación extra**

Haz un endpoint específico en tu backend para entregar la configuración de ONLYOFFICE (con el token generado).