



Manual Técnico sistema de requerimientos





Contenido

Introducción	3
Estructura de software del proyecto	4
Diagrama de base de datos	5
Estructura Front-end	6
Dependencias externas	6
Estructura de carpetas y archivos	7
Estructura Back-end	12
Dependencias externas	12
Estructura de carpetas y archivos	14
Base de datos (local)	16
Publicación del Sistema de requerimientos en IIS	17
Activar IIS	17
Descarga de e instalación de herramientas	18
Publicación de proyecto Angular	19
Publicación de proyecto nodejs	24
Reglas de firewall	28
Gestión de procesos utilizando PM2	34
Observaciones para una conexión entre angular y nodejs en iis exitosa	35
Conexión de puertos	35
Cambios en angular	37



Introducción

A continuación, se describe la estructura completa del proyecto sistema de requerimientos.

El funcionamiento de este sistema se fundamenta en la organización y documentación de distintos procesos, los cuales pueden abordarse en diferentes niveles, como la base de datos, el front-end y el back-end. Esto implica que, al utilizar la plataforma, podrás registrar y dar seguimiento a todas las alteraciones que se efectúen en los sistemas que conforman la red solidaria, garantizando un control efectivo y una documentación de cada cambio.

Este enfoque proporciona una mayor transparencia y trazabilidad en todo el proceso de desarrollo y mantenimiento de los sistemas, lo que, a su vez, contribuye a la mejora continua y a la eficiencia de la red solidaria.

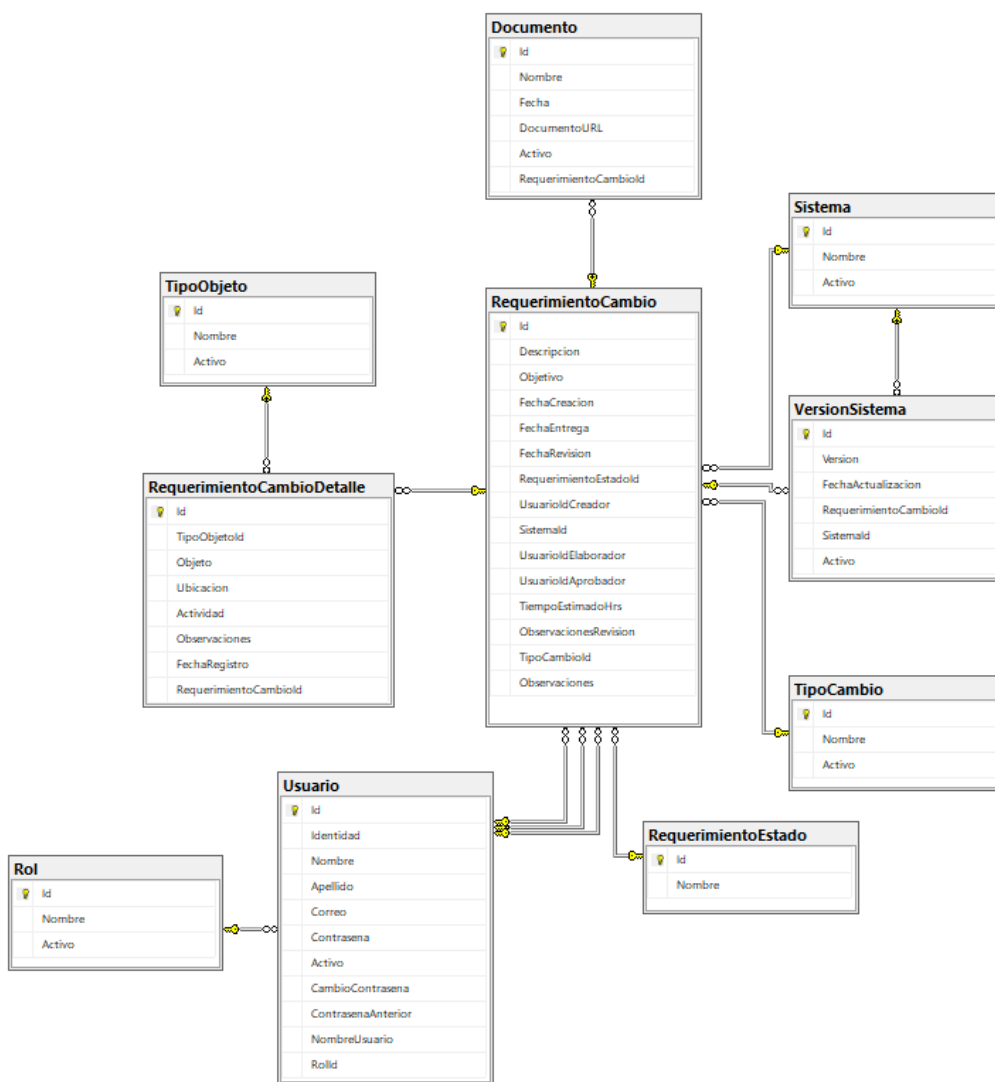


Estructura de software del proyecto





Diagrama de base de datos





Estructura Front-end

El proyecto utiliza Angular 16.1.0 como framework para aplicación web.

Angular es un framework para aplicaciones web desarrollado en TypeScript, de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página.

Dependencias externas

- alertifyjs: 1.13.1

Componentes para alertar notificaciones.

- bootstrap: 5.3.1

Componentes de estilos html con css.

- jwt-decode:3.1.2

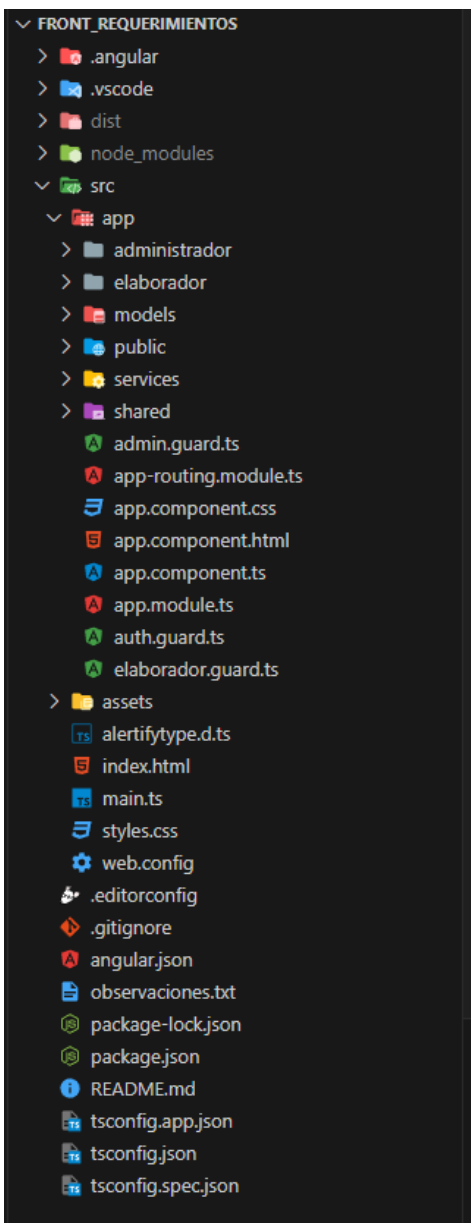
Gestor de credenciales jsonwebtoken.

- pdfmake: 0.2.7

Manipulación de archivos pdf.

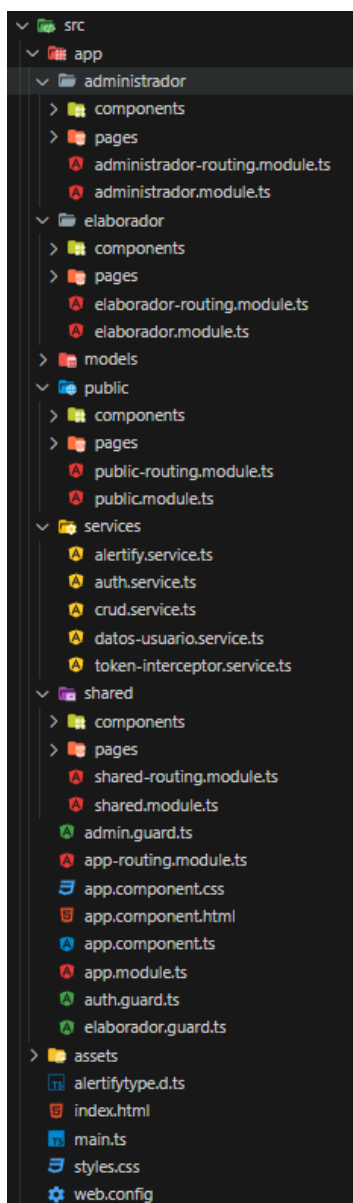


Estructura de carpetas y archivos



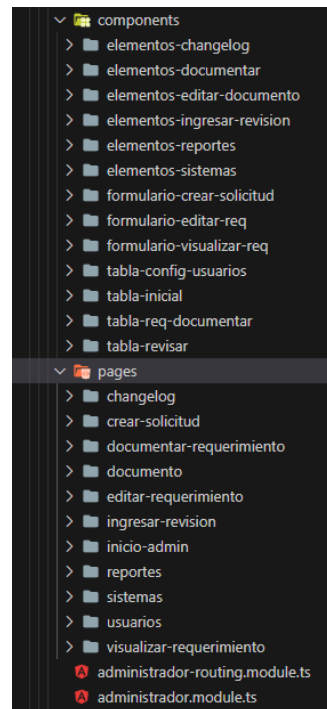


- Angular crea por defecto varias carpetas y archivos que no se trabajan sobre ellas ya que son configuraciones o dependencias para el mismo.
- Por lo que las carpetas donde se trabaja son las siguientes





- Algunas carpetas se dividen en subcarpetas que son components y pages, dentro de components se encuentran componentes necesarios para mostrarse en alguna página, como formularios, tablas, entre otros que deben ir dentro de una página específica.



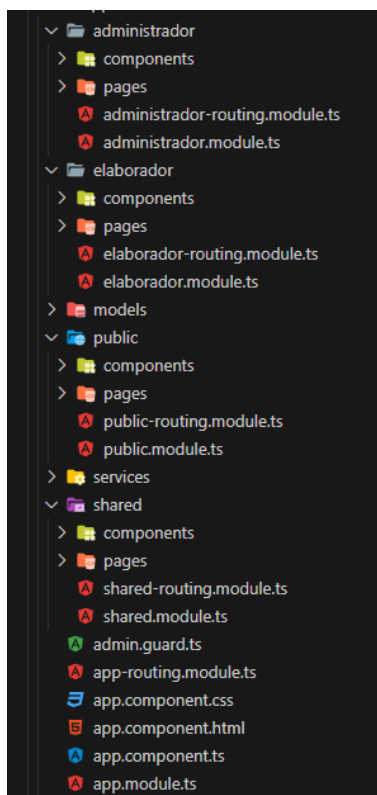


- Módulos y rutas

Los módulos contienen todas las páginas, componentes, rutas, librerías, etc., necesarios para el funcionamiento de procesos.

Angular tiene un módulo global (app.module.ts) donde se puede añadir todo eso de diferentes carpetas, pero al declarar todo en un mismo modulo hace que la aplicación cargue absolutamente todo, aunque no se utilice en ese momento para evitar esto se declaro un submódulo en algunas carpetas donde se contiene lo necesario para cada área.

Para cargar solo el modulo que se utiliza en ese momento se crearon subsistemas de rutas dentro de las carpetas que lo necesitan.



Y en el archivo de rutas global (app-routing.module.ts) se mandan a llamar los módulos correspondientes dependiendo las rutas.

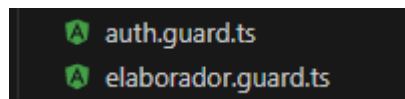
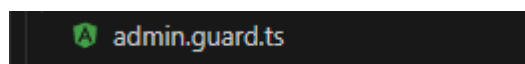


Por lo que el módulo global (app-module.ts) se declaran dependencias globales que son necesarias para el funcionamiento de toda la aplicación.

- Guards

Admin y elaborador cuidan que un usuario que no tiene el rol correspondiente o con estado inactivo no puedan acceder a rutas que no les pertenecen

El auth.guard vigila que los usuarios tengan el token de autorización y que este no haya sufrido ninguna modificación.

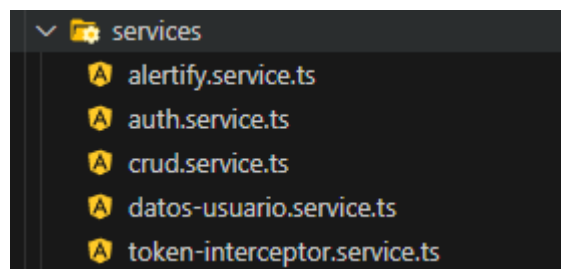


- Carpeta models

Esta carpeta contiene interfaces que modelan los datos que se traen o se envían a la base de datos.

- Carpeta services

Esta carpeta contiene todos los servicios que se pueden utilizar globalmente en el proyecto.



- Auth: Contiene los métodos relacionados con el login y el token.
- Crud: Contiene los métodos relacionados con el envío y recibimientos de datos del backend.
- datos-usuario: Envía el token del usuario y compara si la información corresponde con la información del usuario que recupera el backend en la base de datos.
- Token-interceptor: Intercepta todas las solicitudes que se envían al backend y agrega el token en el encabezado.



- Carpeta administrador: Dentro de ella se encuentra todo lo que se necesita en las interfaces de administrador.
- Carpeta elaborador: Dentro de ella se encuentra todo lo que se necesita en las interfaces de elaborador.
- Carpeta public: Contiene los elementos donde no se necesita ingresar sesión para verlos como crear usuario, login y recuperar contraseña.
- Carpeta shared: Esta carpeta contiene elementos que se mostraran para ambos usuarios sin importar el rol.
- Carpeta assets: En esta carpeta se almacena los recursos de personalización como imágenes, tipografías y papel membretado.

Estructura Back-end

El proyecto utiliza nodejs 18.17.0 como entorno de servidor para el fronted.

Node.js es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor basado en el lenguaje de programación JavaScript, asíncrono, con E/S de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google.

Dependencias externas

- bcrypt: 5.1.1

Librería para encriptación.

- body-parser: 1.19.0

librería para analizar y procesar solicitudes HTTP.

- cors: 2.8.1

Intercambio de recursos de origen cruzado, necesario para interactuar con el fronted.



- dotenv: 16.3.1

Paquete para node js que nos permite configurar o usar las variables de entorno en nuestro código

- excel4node: 1.8.2

Librería para manejo y creación de documentos de Excel.

- express: 4.17.1

Express.js es el framework de backend para nodejs.

- jsonwebtoken: 9.0.2

Manejo de credenciales jsonwebtoken.

- moment-timezone: 0.5.43

Configuración de zona horaria y fecha.

- mssql: 6.2.1

cliente Microsoft SQL Server para Node .js

- multer: 1.4.5-lts.1

"Middleware" de nodejs para el manejo de multipart/form-data, el cuál es usado sobre todo para la subida de archivos.

- nodemailer: 6.9.4

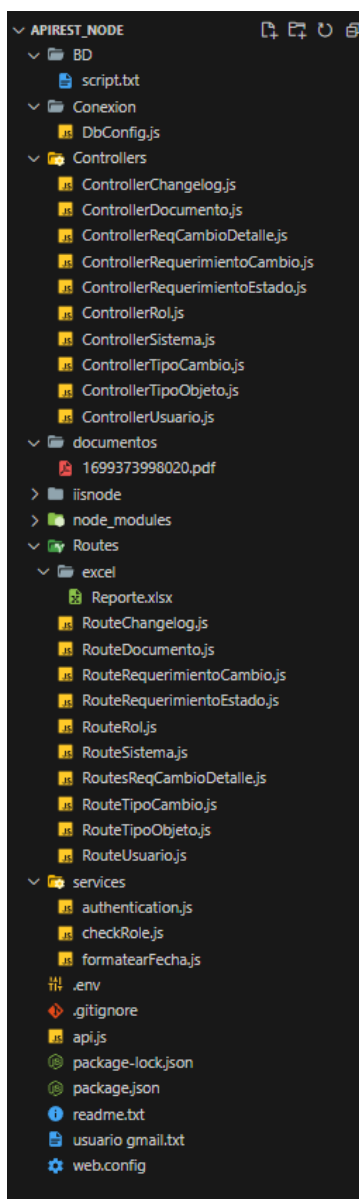
Módulo de Node. js que te permite enviar emails desde tu servidor.

- nodemon: 2.0.4

Vigila el sistema de archivos y reinicia automáticamente el proceso.



Estructura de carpetas y archivos





- Carpeta BD: contiene un archivo de texto con el script de la creación de la base de datos.
- Carpeta conexión: contiene un archivo con la configuración del backend hacia la base de datos.
- Carpeta controller: contiene varios archivos cada uno quiere representar las consultas a la base de datos de cierta tabla para obtener los datos.
- Carpeta documentos: Aquí se guardan los archivos pdf que se suben en el sistema. Esta carpeta siempre debe existir a menos que se cambie la dirección de donde se deben guardar los documentos.
- Archivo .env: aquí se configuran diferentes variables que son llamadas a lo largo del código, como ser, acces_token, email, password, user (variable que se usa para comparar el rol en el middleware checkrol), documentoUrl y una contraseña temporal para la cambiar la contraseña.
- Carpeta routes: Dentro de esta se encuentran los archivos routes que son rutas para poder acceder a la información desde el fronted.
- Carpeta Excel dentro de routes: dentro de esta carpeta se crea el reporte de Excel para que el fronted pueda descargarlo de esa carpeta.
- Carpeta services: dentro de esta carpeta están los archivos que actuaran como servicios para ser implementados a lo largo del código.
 - Authentication: La función authenticateToken es un middleware de Express que se utiliza para autenticar las solicitudes.
 - checkRole: La función checkRole es otro middleware de Express que se utiliza para verificar el rol del usuario autenticado.
 - formatearFecha: Función para formatear y registrar la fecha actual en la zona horaria deseada
- Archivo api.js: Archivo principal donde se declaran todas las rutas, puerto entre otros datos.



Base de datos (local)

- Gestor de base de datos: SQL Server 2022
- SQL Server Management Studio: 2019

Para el manejo de la base de datos el coordinador de desarrollo creo la base de datos en el servidor y cedió acceso para poder manipularla de forma local de esta forma tanto el api en nodejs y el gestor de base de datos local no presentaron problemas con incompatibilidad de versiones de sql.

- En caso de que se formateara la base de datos hay algunas tablas que tienen que tener datos por defecto.

Tabla RequerimientoEstado

	Id	Nombre
1	1	Requerimiento Creado
2	2	Requerimiento Iniciado
3	3	Revisar Requerimiento
4	4	Requerimiento Revisado
5	5	Requerimiento Eliminado
6	6	Requerimiento Terminado
7	7	Requerimiento Denegado

Tabla Rol

Resultados		Mensajes	
	Id	Nombre	Activo
1	1	Administrador	1
2	2	Elaborador	1

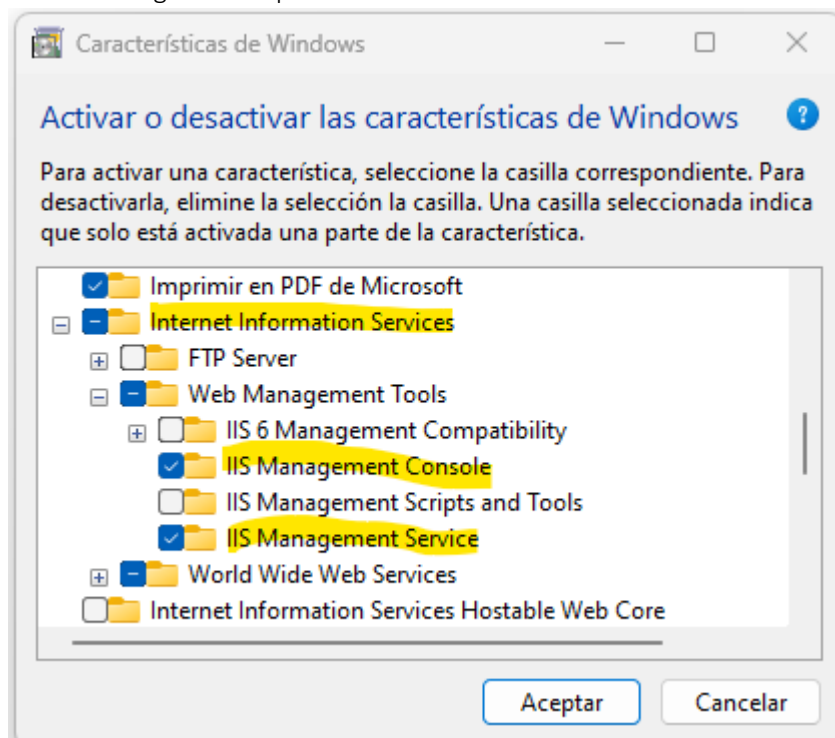
El número de Id también se tiene que mantener. Cuando se crea un usuario por defecto es asignado como elaborador por lo que para nombrar al primer administrador el RolId se debe cambiar de forma manual en la base de datos de ahí podrá cambiar el rol de los usuarios desde el sistema.



Publicación del Sistema de requerimientos en IIS

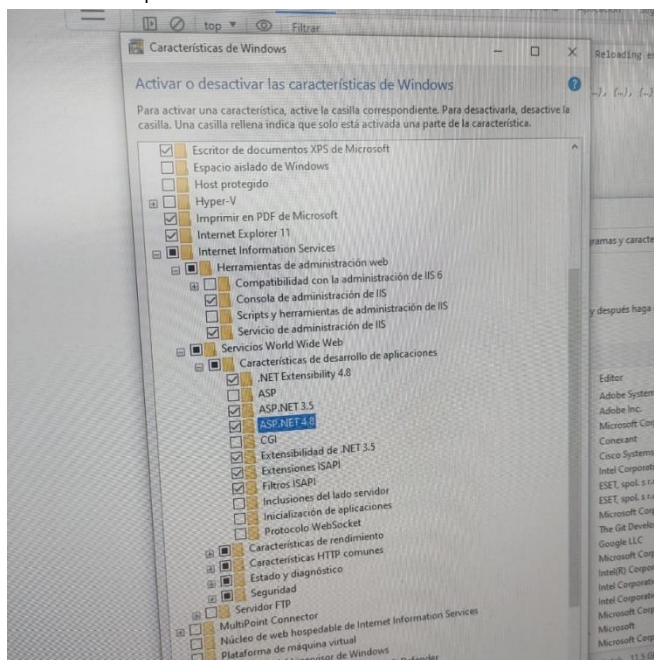
Activar IIS

1. Irse a “Activar o desactivar características de Windows”
2. Activar las siguientes opciones





Por defecto se activan otras opciones



Descarga de e instalación de herramientas

- Application Request Routing

<https://www.iis.net/downloads/microsoft/application-request-routing>

- URL Rewrite

<https://www.iis.net/downloads/microsoft/url-rewrite>

- IISNODE

<https://github.com/azure/iisnode/wiki/iisnode-releases>

- Nodejs (versión utilizada en el backend)

<https://nodejs.org/en>



Tanto el proyecto de angular como node deben estar en el disco c pero para más organización lo pondremos en la carpeta inetpub dentro de wwwroot C:\inetpub\wwwroot\nombreCarpeta

Pero si el proyecto de node da problemas de permisos se puede ejecutar desde una carpeta en el disco c.

Publicación de proyecto Angular

<https://dev.to/mezagini/subir-proyecto-de-angular-a-iis-3p4p>

1. Crear una carpeta vacía donde se guardará los archivos build de angular en la carpeta

C:\inetpub\wwwroot

2. Dentro del directorio src debe crear un archivo web.config Que resolverá las rutas de la APP.

Esta es la configuración de ese archivo

```
<configuration>
  <system.webServer>
    <rewrite>
      <rules>
        <clear /> <!-- Imperative Step, otherwise IIS will throw err -->
        <rule name="Angular Routes" stopProcessing="true">
          <match url=".*" />
          <conditions logicalGrouping="MatchAll">
            <add input="{REQUEST_FILENAME}" matchType="IsFile" negate="true" />
            <add input="{REQUEST_FILENAME}" matchType="IsDirectory" negate="true" />
          </conditions>
          <action type="Rewrite" url="/" />
        </rule>
      </rules>
    </rewrite>
  </system.webServer>
</configuration>
```





3. Ese mismo archivo, deberá ir en el build de la app. Por lo que hay que indicarlo. Esto se hace dentro del archivo angular.json dentro del arreglo de assets: []

```
"assets": [  
  "src/favicon.ico",  
  "src/assets",  
  "src/web.config"  
]
```

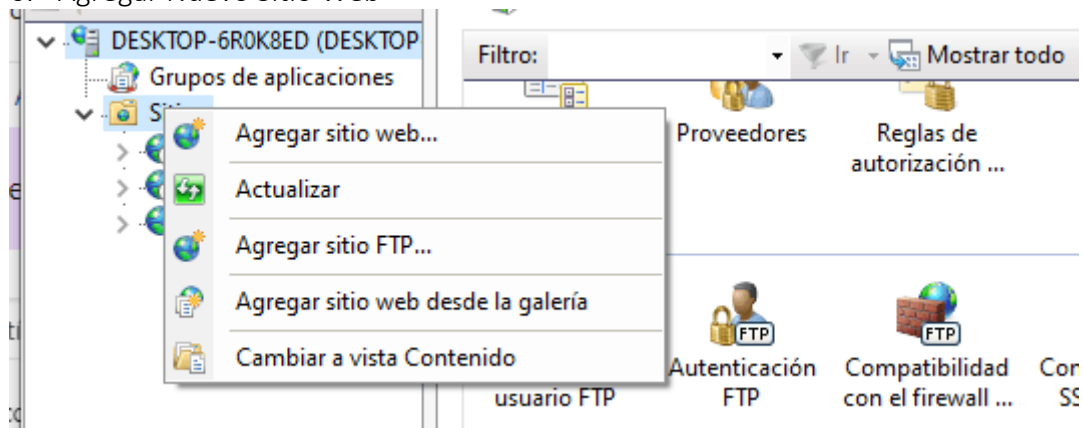
4. Generar build de la aplicación

En este caso, se hace un build para usar en el entorno de producción. Pero también puede ser para desarrollo.

Muy importante indicar la subcarpeta donde éste estará alojado con el flag --base-href. También hay que destacar que debe colocarse el nombre de la aplicación que está dentro del sitio que se creó al inicio. En la sección de preparación del entorno para el deploy. En este caso, la app se le llamó "NuevoProyecto"

ng build --configuration=production --base-href "/NombreDeLaCarpetaCreada/"

5. Copiar los archivos que están en el dist y guardar en /carpetaCreadaAlInicio
6. Agregar Nuevo Sitio Web





Agregar sitio web ? X

Nombre del sitio: Grupo de aplicaciones:

Directorio de contenido

Ruta de acceso física:

Autenticación de paso a través

Enlace

Tipo: Dirección IP: Puerto:

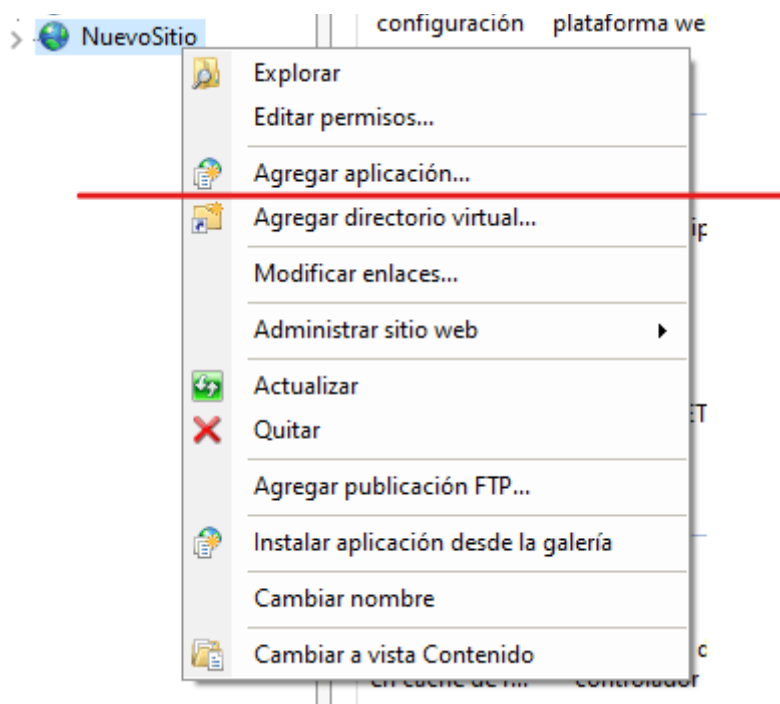
Nombre de host:

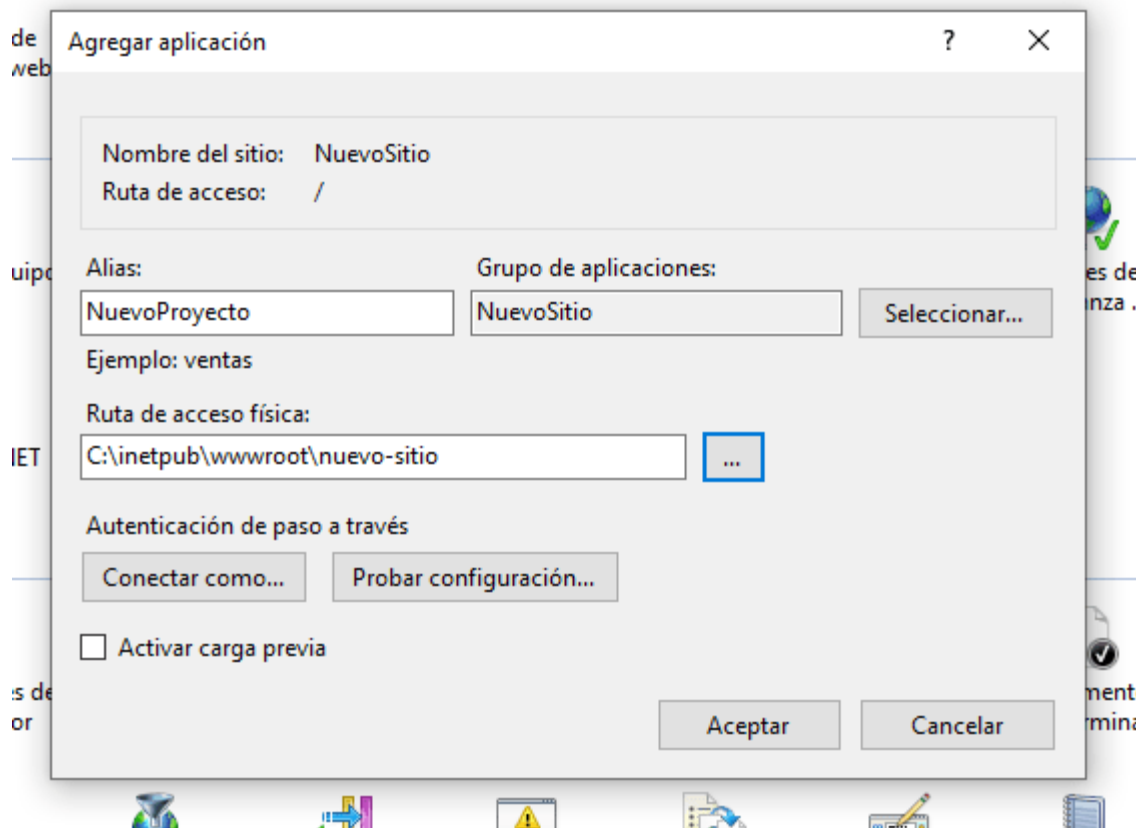
Ejemplo: www.contoso.com o marketing.contoso.com

☒ Iniciar sitio web inmediatamente



7. Coloca el nombre del sitio
8. La ruta donde se almacenarán los archivos de la publicación del proyecto (la carpeta creada) y, por último, el puerto. Y "Aceptar"
9. Agregar Aplicación





10. Colocas el Alias

11. La ruta de acceso al directorio donde irán los archivos de la publicación del proyecto. Por último, "Aceptar"

Obs: si se quiere ver de otro dispositivo en la misma red poner ipcomputadora/nombrecarpeta/
ejm: 172.16.242.78/requerimientos/



Publicación de proyecto nodejs

<https://harveywilliams.net/blog/installing-iisnode>

una vez instalados los programas indicados al inicio podemos proceder. La aplicación de node debe utilizar express.

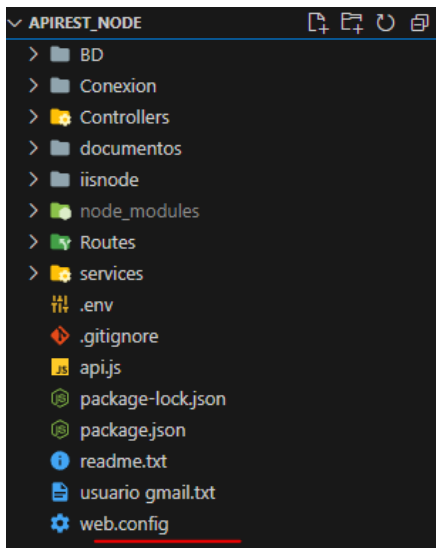
1. En el archivo principal de node debe definirse el puerto a utilizar

```
apijs > ...
1  const express = require("express");
2  const bodyParser = require("body-parser");
3  const cors = require("cors");
4
5  // Importa las rutas necesarias de la carpeta Routes
6  const routesRol = require("./Routes/RouteRol");
7  const routesSistema = require("./Routes/RouteSistema");
8  const routesTipoCambio = require("./Routes/RouteTipoCambio");
9  const routesDocumento = require("./Routes/RouteDocumento");
10 const routesReqEstado = require("./Routes/RouteRequerimientoEstado");
11 const routesTipoObjeto = require("./Routes/RouteTipoObjeto");
12 const routesUsuario = require("./Routes/RouteUsuario");
13 const routesReqCambio = require("./Routes/RouteRequerimientoCambio");
14 const routesReqCambioDetalle = require("./Routes/RouteReqCambioDetalle");
15 const routesChangelog = require("./Routes/RouteChangelog");
16
17 //configura la conexion con el fronted
18 const app = express();
19 app.use(bodyParser.urlencoded({ extended: true }));
20 app.use(bodyParser.json());
21 app.use(cors());
22
23 // Usa las rutas importadas bajo el prefijo 'api'
24 app.use("/api", routesRol);
25 app.use("/api", routesSistema);
26 app.use("/api", routesTipoCambio);
27 app.use("/api", routesDocumento);
28 app.use("/api", routesReqEstado);
29 app.use("/api", routesTipoObjeto);
30 app.use("/api", routesUsuario);
31 app.use("/api", routesReqCambio);
32 app.use("/api", routesReqCambioDetalle);
33 app.use("/api", routesChangelog);
34
35 //para que se puedan ver los documentos en el navegador con poner http://localhost:1054/nombredocumento----o http://direccionip:1054/nombredocumento-----
36 app.use(express.static(process.env.DocumentoURL));
37
38 //Establece el puerto a utilizarse
39 app.listen(1054);
40
41 app.get('/', (req, res) => {
42   res.send('¡Api en funcionamiento!');
43 });
```





2. Crear el archivo web.config



Con la siguiente configuración

```
<configuration>
  <system.webServer>

    <!-- indicates that the api.js file is a node.js application
    to be handled by the iisnode module -->

    <handlers>
      <add name="iisnode" path="api.js" verb="*" modules="iisnode" />
    </handlers>

    <rewrite>
      <rules>
        <rule name="sendToNode">
          <match url="/*" />
          <action type="Rewrite" url="api.js" />
        </rule>
      </rules>
    </rewrite>

  </system.webServer>
</configuration>
```

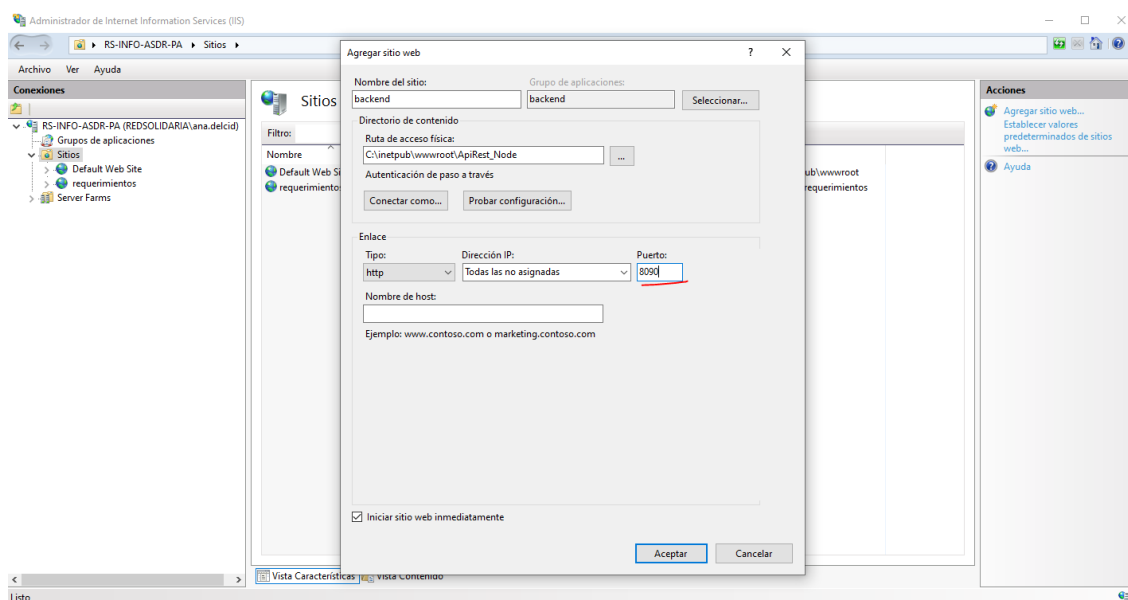




3. Agregar su sitio a IIS.

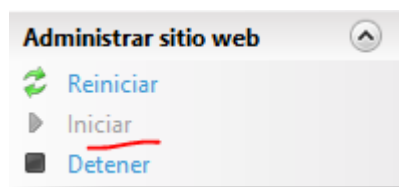
Para mayor accesibilidad es mejor que el proyecto de node este dentro de la carpeta C:\inetpub\wwwroot pero si da problemas de permisos se puede ejecutar desde una carpeta en el disco c.

El puerto ingresado debe ser diferente al que está declarado dentro de la aplicación de node



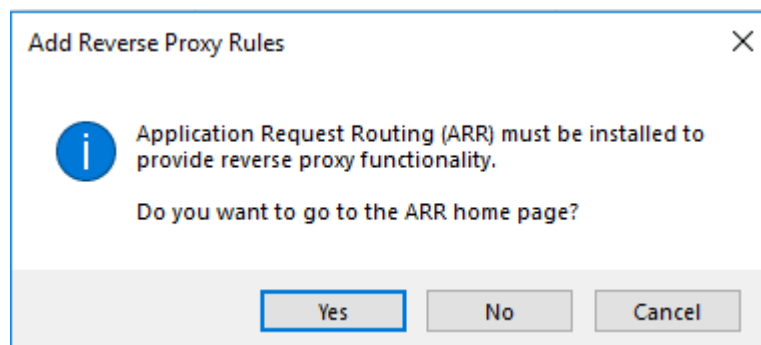
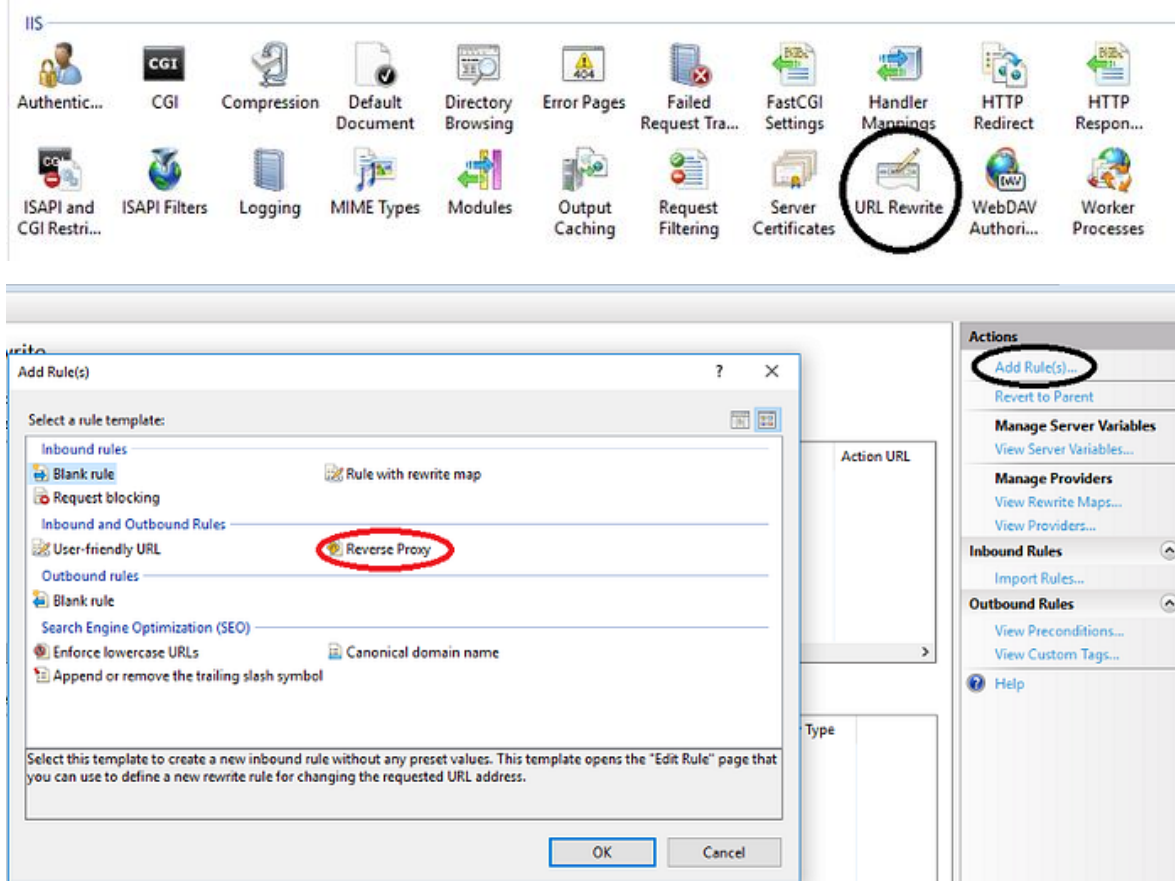
4. Iniciar el sitio, para poder ver si funciona en el navegador es necesario poner

<http://localhost:1054/> el puerto 1054 es el indicado en la aplicación de nodejs





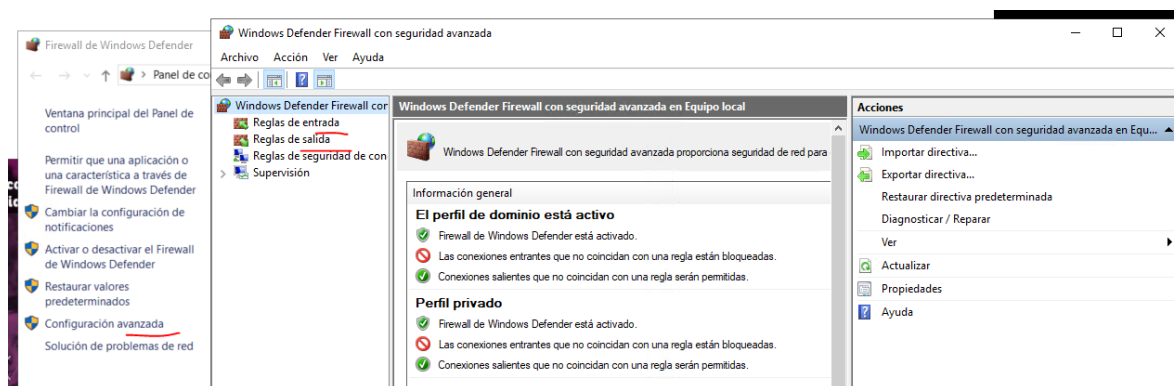
5. Paso (opcional, solo si hay error en las rutas) Si necesita vincular un nombre de host diferente al hosting en localhost, es posible que necesite usar un proxy inverso.



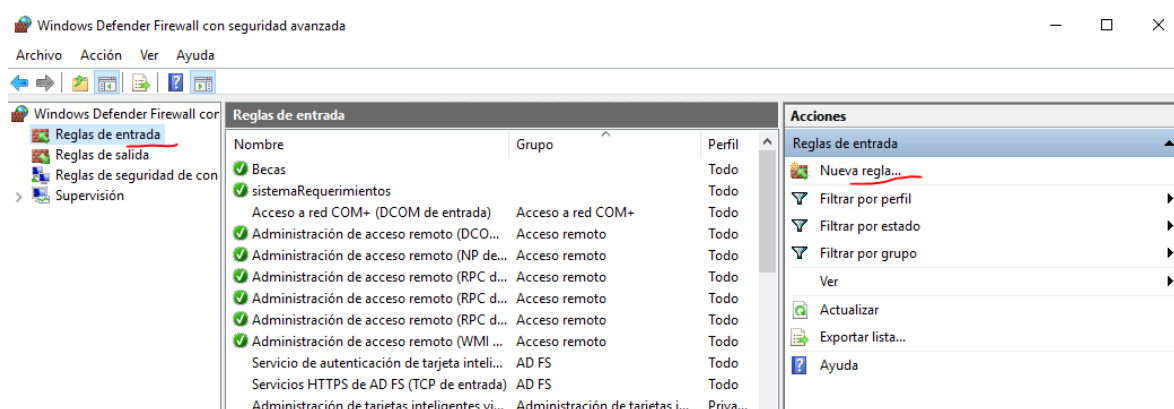


Reglas de firewall

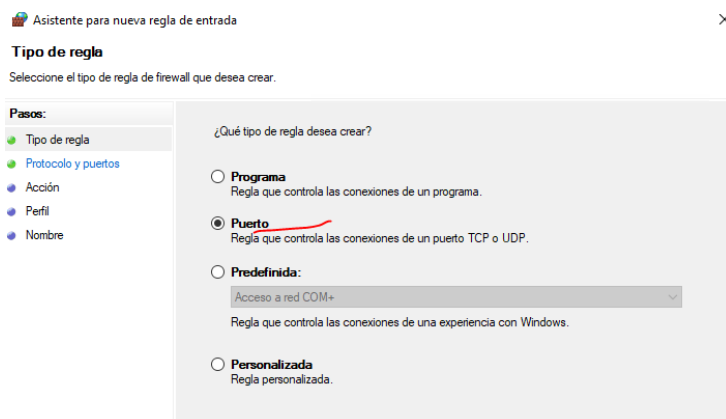
Si el sistema no se puede ver en otros dispositivos es necesario establecer los puertos utilizados en reglas de entrada y salida.



1. Nueva regla de entrada



2. Seleccionar puerto





3. Establecer los puertos y protocolo

Asistente para nueva regla de entrada

Protocolo y puertos
Especifique los puertos y protocolos a los que se aplica esta regla.

Pasos:

- Tipo de regla
- Protocolo y puertos
- Acción
- Perfil
- Nombre

¿Se aplica esta regla a TCP o UDP?

☒ TCP
☐ UDP

¿Se aplica esta regla a todos los puertos locales o a unos puertos locales específicos?

☐ Todos los puertos locales
☒ Puertos locales específicos:
Ejemplo: 80, 443, 5000-5010

< Atrás **Siguiente >** Cancelar

4. Permitir conexión

Asistente para nueva regla de entrada

Acción
Especifique la acción que debe llevarse a cabo cuando una conexión coincide con las condiciones especificadas en la regla.

Pasos:

- Tipo de regla
- Protocolo y puertos
- Acción
- Perfil
- Nombre

¿Qué medida debe tomarse si una conexión coincide con las condiciones especificadas?

☒ **Permitir la conexión**
Esto incluye las conexiones protegidas mediante IPsec y las que no lo están.

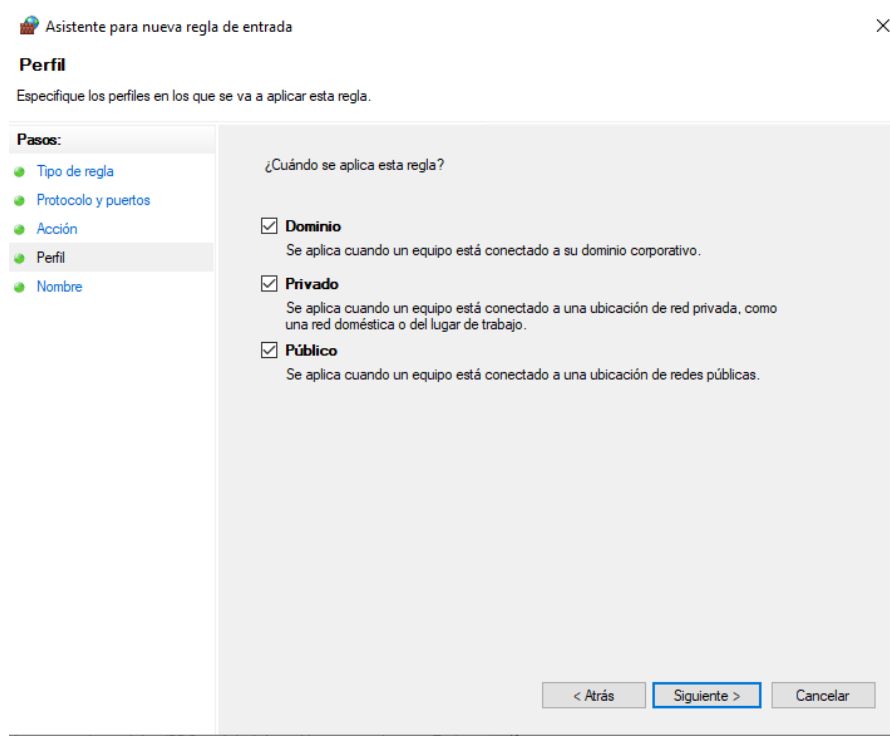
☐ **Permitir la conexión si es segura**
Esto incluye solamente las conexiones autenticadas mediante IPsec. Éstas se protegerán mediante la configuración de reglas y propiedades de IPsec del nodo Regla de seguridad de conexión.

☐ **Bloquear la conexión**

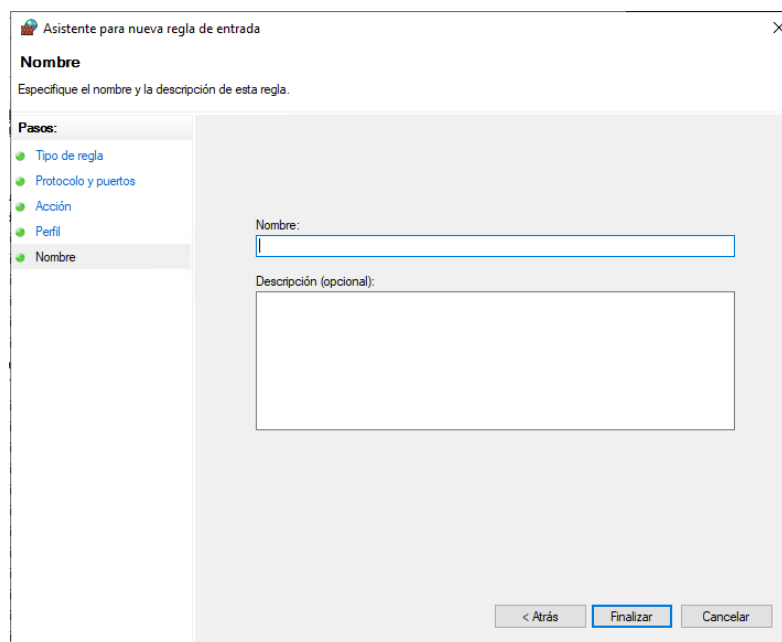
< Atrás **Siguiente >** Cancelar



5. Seleccionar cuando se debe aplicar



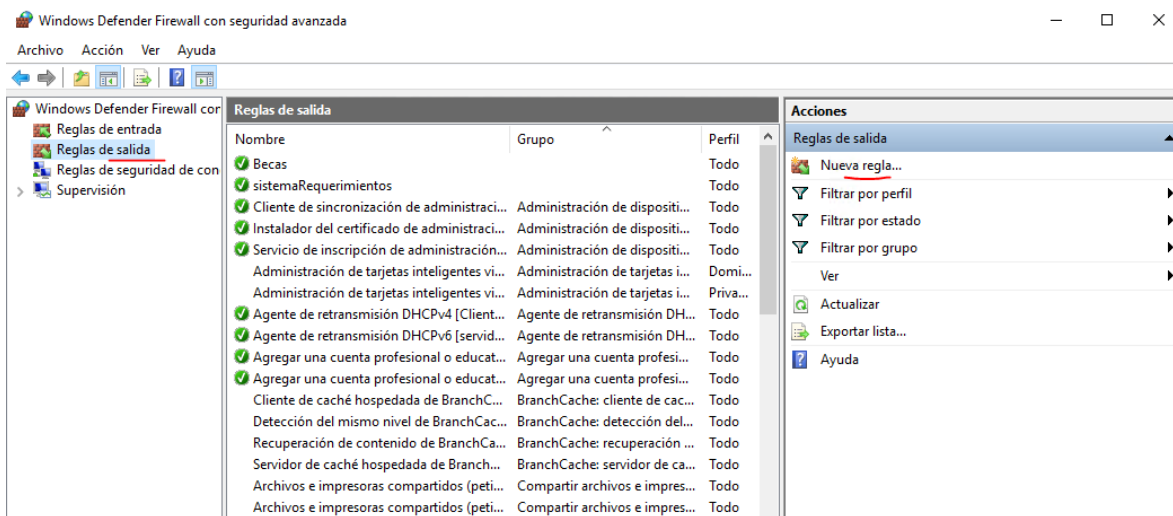
6. Establecer nombre



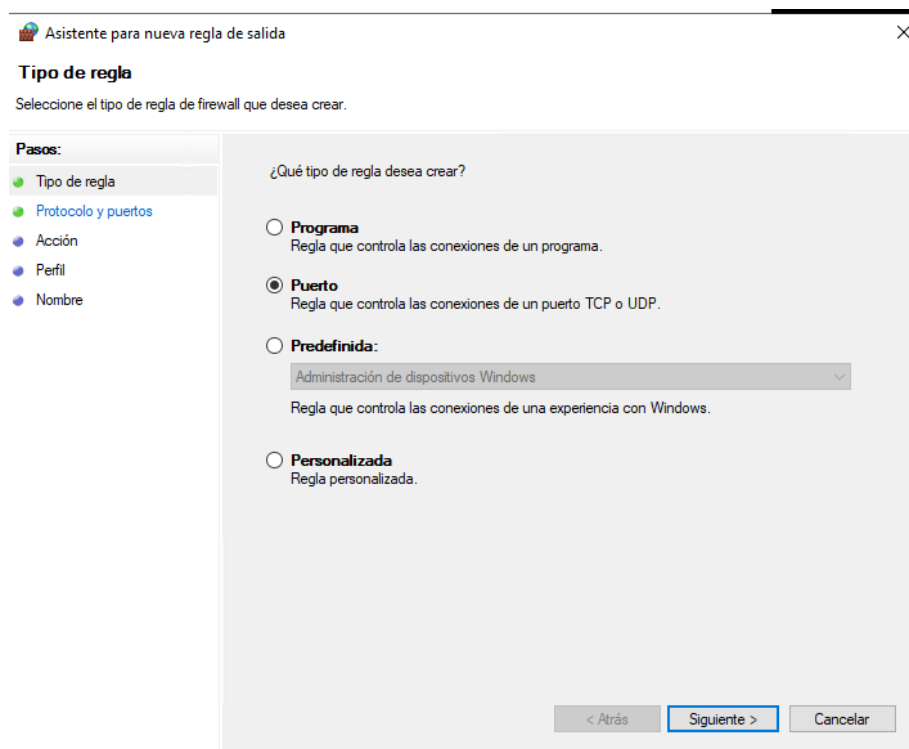
Edificio Plaza Azul, Pisos 3 y 5, Calle Viena, subida a Lomas del Guijarro,
Tegucigalpa, MDC, Honduras, C.A.



7. Nueva regla de salida

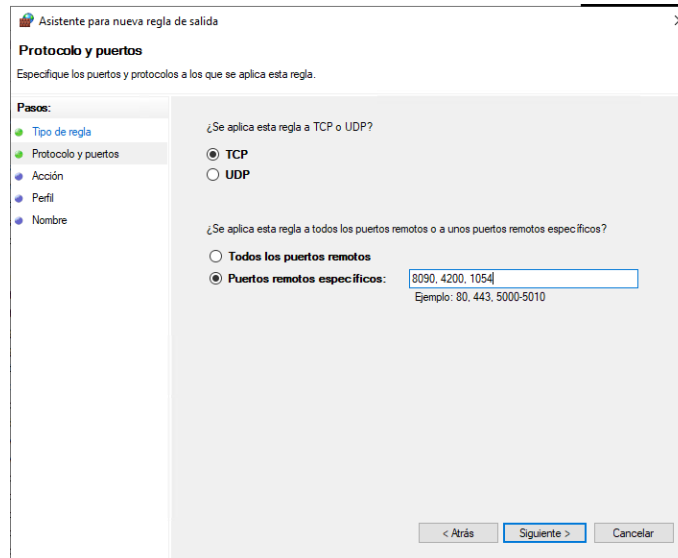


8. Establecer tipo de regla





9. Puertos y protocolo



Asistente para nueva regla de salida

Protocolo y puertos

Especifique los puertos y protocolos a los que se aplica esta regla.

Pasos:

- Tipo de regla
- Protocolo y puertos
- Acción
- Perfil
- Nombre

¿Se aplica esta regla a TCP o UDP?

☒ TCP

☐ UDP

¿Se aplica esta regla a todos los puertos remotos o a unos puertos remotos específicos?

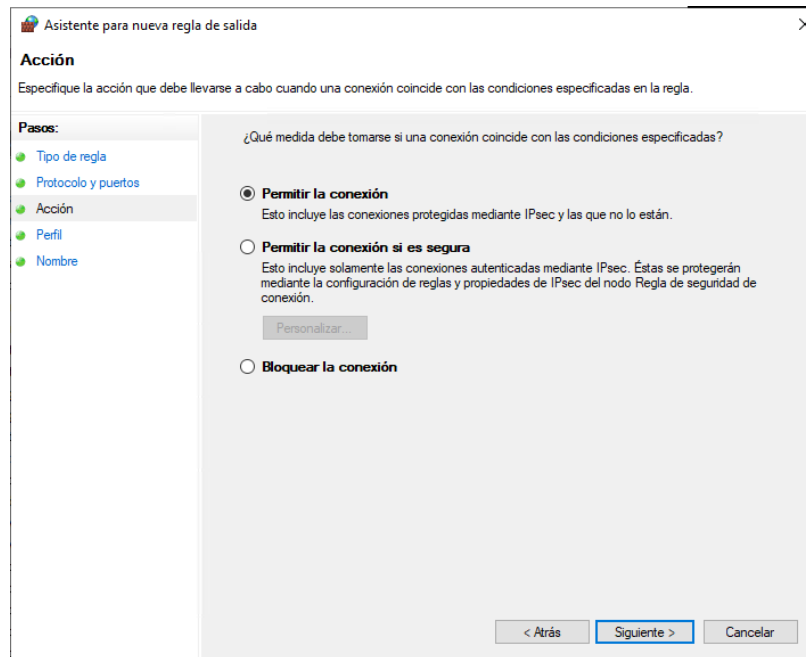
☐ Todos los puertos remotos

☒ Puertos remotos específicos:

Ejemplo: 80, 443, 5000-5010

< Atrás **Siguiente >** Cancelar

10. Permitir conexión



Asistente para nueva regla de salida

Acción

Especifique la acción que debe llevarse a cabo cuando una conexión coincide con las condiciones especificadas en la regla.

Pasos:

- Tipo de regla
- Protocolo y puertos
- Acción
- Perfil
- Nombre

¿Qué medida debe tomarse si una conexión coincide con las condiciones especificadas?

☒ **Permitir la conexión**

Esto incluye las conexiones protegidas mediante IPsec y las que no lo están.

☐ **Permitir la conexión si es segura**

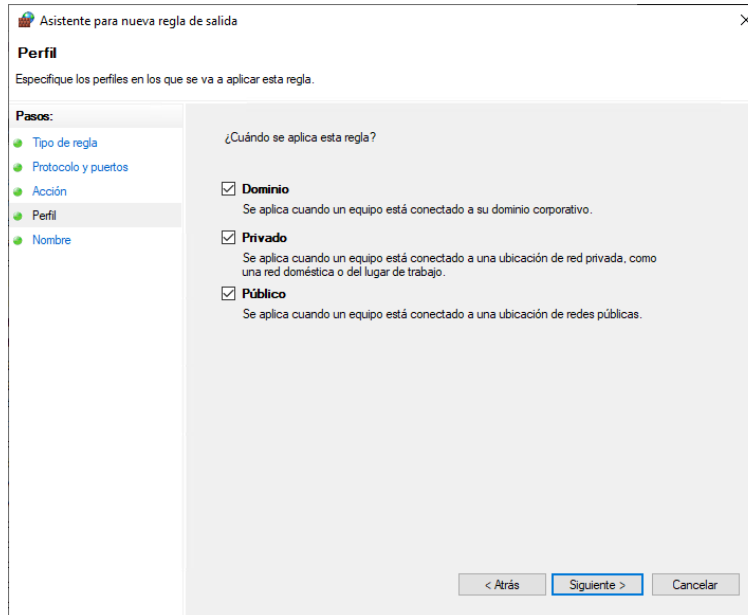
Esto incluye solamente las conexiones autenticadas mediante IPsec. Éstas se protegerán mediante la configuración de reglas y propiedades de IPsec del nodo Regla de seguridad de conexión.

☐ **Bloquear la conexión**

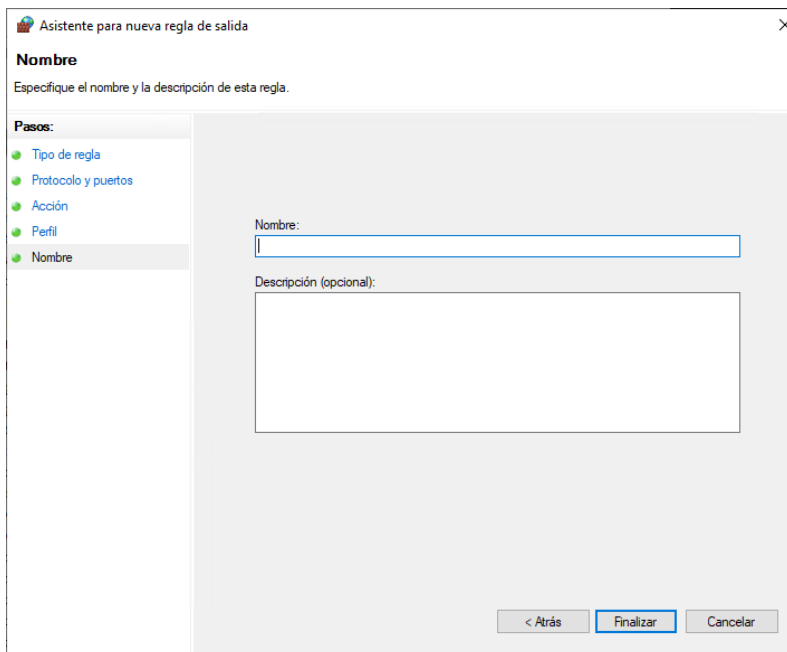
< Atrás **Siguiente >** Cancelar



11. Seleccionar cuando se aplicara la regla



12. Establecer nombre

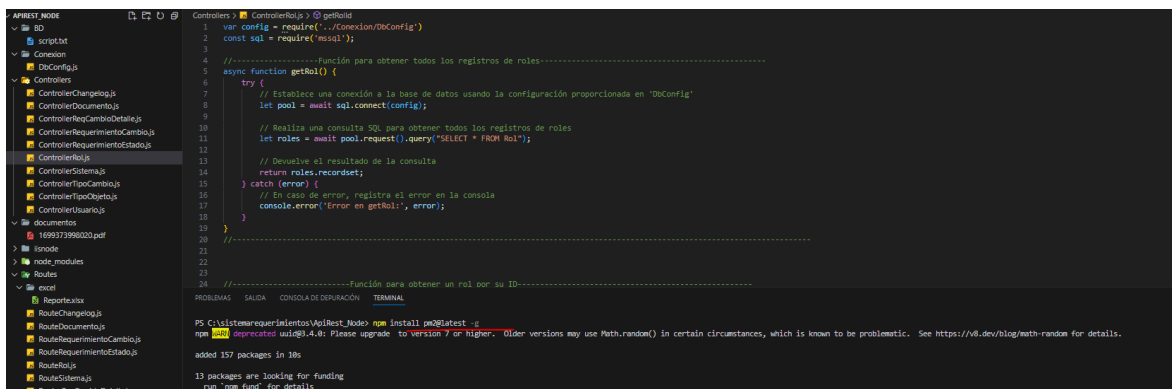




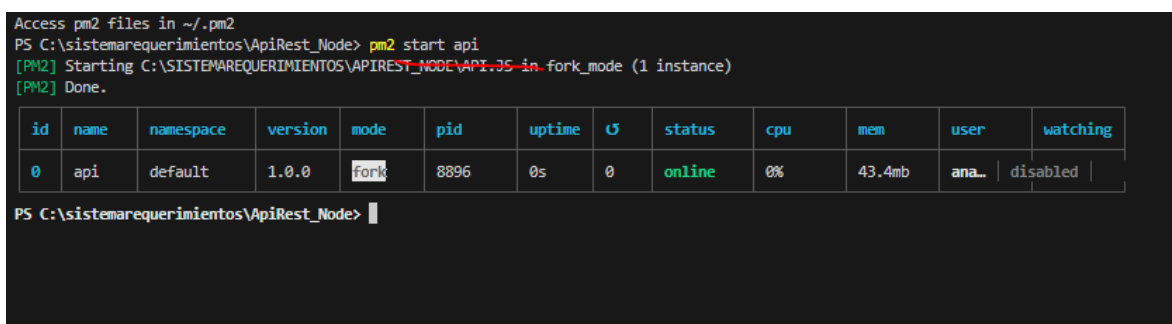
Gestión de procesos utilizando PM2.

PM2 es una herramienta de gestión de procesos que se puede utilizar para gestionar su aplicación Node.js.

1. Instale PM2 Type para comprobar el estado actual de los servidores. En nuestro caso, aún no iniciamos ni detuvimos ningún servidor. Con los comandos `npm install pm2@latest -g`, `pm2 ls`.



2. Escriba `pm2 start api.js` para iniciar el servicio y listo.



```
Access pm2 files in ~/.pm2
PS C:\sistemarequerimientos\ApiRest_Node> pm2 start api
[PM2] Starting C:\SISTEMAREQUERIMIENTOS\APIREST_NODE\API-JS in fork_mode (1 instance)
[PM2] Done.
```

id	name	namespace	version	mode	pid	uptime	U	status	cpu	mem	user	watching
0	api	default	1.0.0	Fork	8896	0s	0	online	0%	43.4mb	ana...	disabled

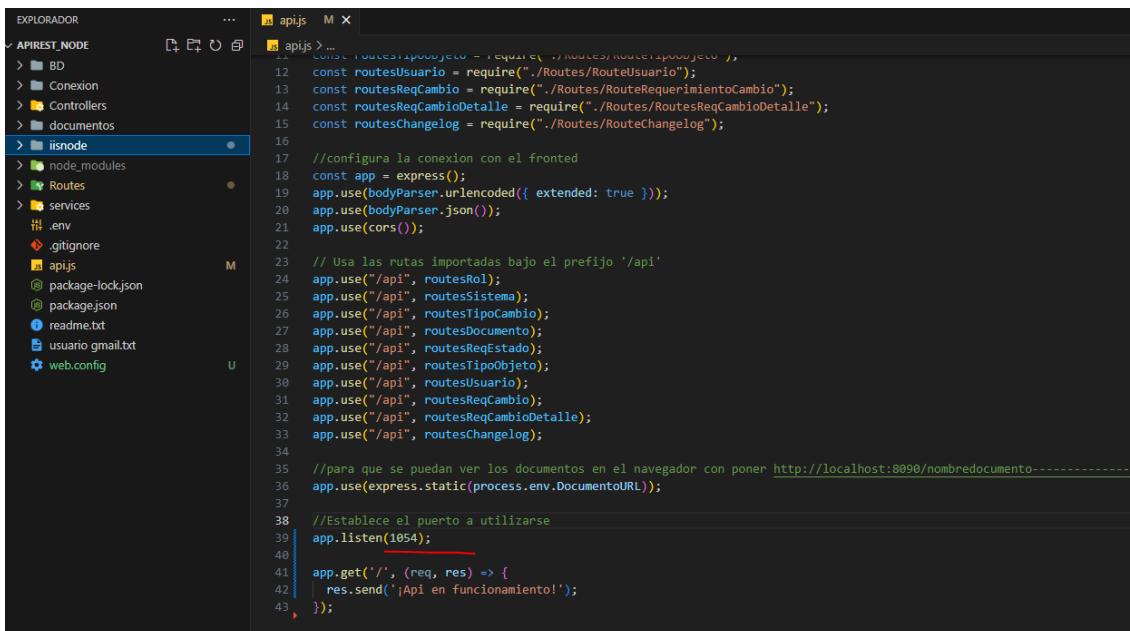
```
PS C:\sistemarequerimientos\ApiRest_Node>
```



Observaciones para una conexión entre angular y nodejs en iis exitosa

Conexión de puertos

1. Definir un puerto en el api de nodejs

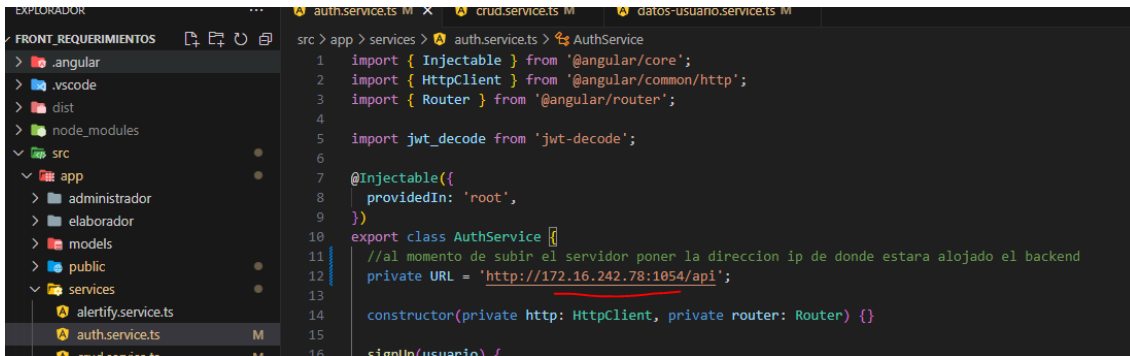


```
EXPLORADOR
  APIREST_NODE
    > BD
    > Conexion
    > Controllers
    > documentos
    > iisnode
    > node_modules
    > Routes
    > services
    .env
    .gitignore
    apijs
    @ package-lock.json
    @ package.json
    README.txt
    usuario gmail.txt
    web.config

apijs
  11 const routesTipoObjeto = require("../Routes/RouteTipoObjeto");
  12 const routesUsuario = require("../Routes/RouteUsuario");
  13 const routesReqCambio = require("../Routes/RouteRequerimientoCambio");
  14 const routesReqCambioDetalle = require("../Routes/RouteReqCambioDetalle");
  15 const routesChangelog = require("../Routes/RouteChangelog");
  16
  17 //configura la conexión con el frontend
  18 const app = express();
  19 app.use(bodyParser.urlencoded({ extended: true }));
  20 app.use(bodyParser.json());
  21 app.use(cors());
  22
  23 // Usa las rutas importadas bajo el prefijo '/api'
  24 app.use("/api", routesRol);
  25 app.use("/api", routesSistema);
  26 app.use("/api", routesTipoCambio);
  27 app.use("/api", routesDocumento);
  28 app.use("/api", routesReqEstado);
  29 app.use("/api", routesTipoObjeto);
  30 app.use("/api", routesUsuario);
  31 app.use("/api", routesReqCambio);
  32 app.use("/api", routesReqCambioDetalle);
  33 app.use("/api", routesChangelog);
  34
  35 //para que se puedan ver los documentos en el navegador con poner http://localhost:8090/nombredocumento-----
  36 app.use(express.static(process.env.DocumentoURL));
  37
  38 //Establece el puerto a utilizarse
  39 app.listen(1054);
  40
  41 app.get('/', (req, res) => {
  42   res.send('¡Api en funcionamiento!');
  43 });
```

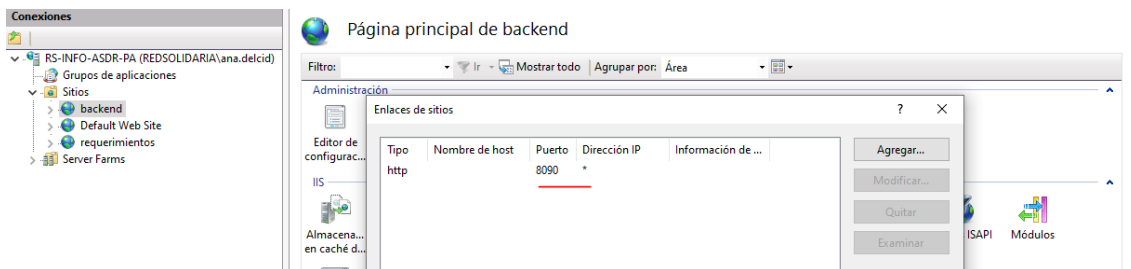


2. Donde angular llame al api poner la dirección ip de la computadora donde este alojado el backend junto con el puerto definido

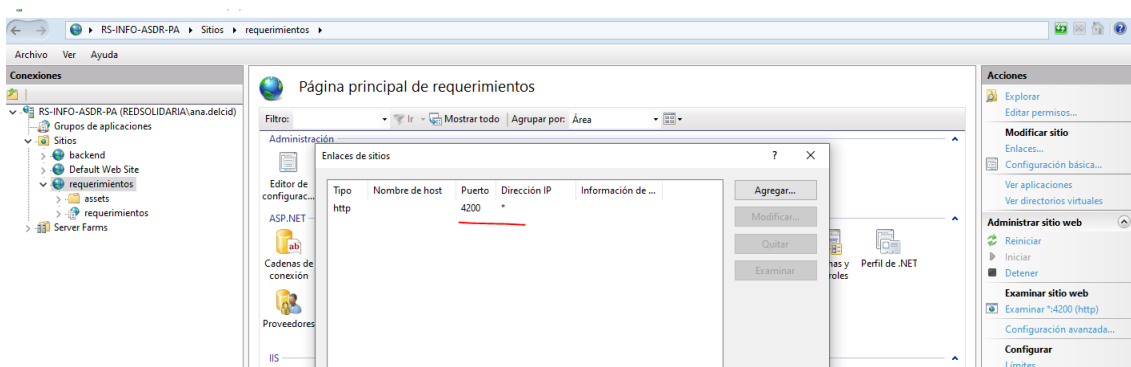


```
src > app > services > auth.service.ts > AuthService
1  import { Injectable } from '@angular/core';
2  import { HttpClient } from '@angular/common/http';
3  import { Router } from '@angular/router';
4
5  import jwt_decode from 'jwt-decode';
6
7  @Injectable({
8    providedIn: 'root',
9  })
10 export class AuthService {
11   //al momento de subir el servidor poner la direccion ip de donde estara alojado el backend
12   private URL = 'http://172.16.242.78:1054/api';
13
14   constructor(private http: HttpClient, private router: Router) {}
15
16   signIn(usuario) {
```

3. En iis en el sitio del backend, el puerto no tiene que ser el mismo definido en el api



4. El puerto del sitio del fronted si puede ser el mismo que muestra angular



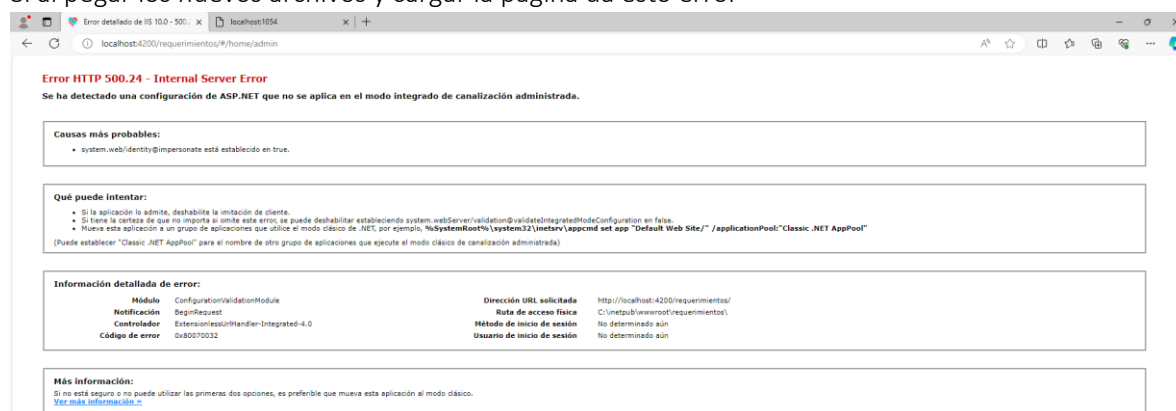


Cambios en angular

Cada vez que se realice un cambio en el proyecto de angular es necesario realizar la carpeta de build (dist) y pegar los archivos en la dirección del sitio como se mostró en al inicio.

Al pegar nuevos archivos en la carpeta del sitio el backend deja de funcionar. Para solucionar esto hay que reiniciar todas las aplicaciones desde el iis.

Si al pegar los nuevos archivos y cargar la página da este error



Solo es necesario desactivar esto

