# Introducción a la solución “services-base”

La estructura de la solución las podemos observar en la Ilustración 1.

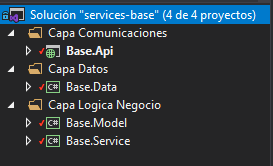


Ilustración Proyectos de la Solución

Los proyectos tienen el prefijo Base y son los siguientes: Service, Model, Data y Api. Los dos primeros corresponden a la Capa Logica Negocio y los otros dos a Capa Datos y Capa de Comunicaciones respectivamente.

# Revisión de la autenticación en “services-base”

En la Ilustración 2 observamos que la clase encargada de la autenticación en Factory Suite es AuthFactorySuiteController dentro del proyecto Api, específicamente dentro de la carpeta Controllers.

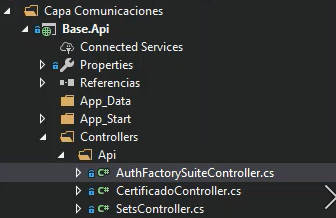


Ilustración Clase para la autenticación en FS

En la Ilustración 3 observamos el método Token a través del cual se realiza la autenticación, en donde observamos que la ruta establecida para solicitar la autenticación es “api/auth-/token” por POST.

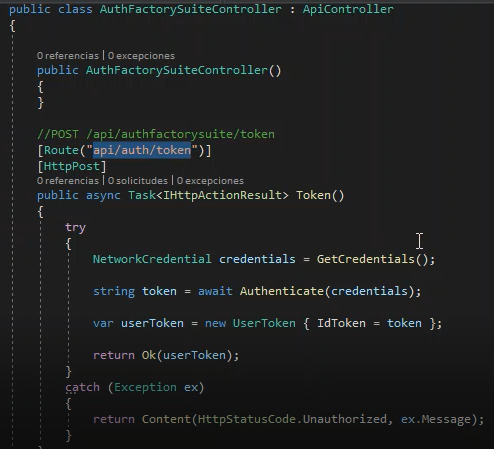


Ilustración Método Token

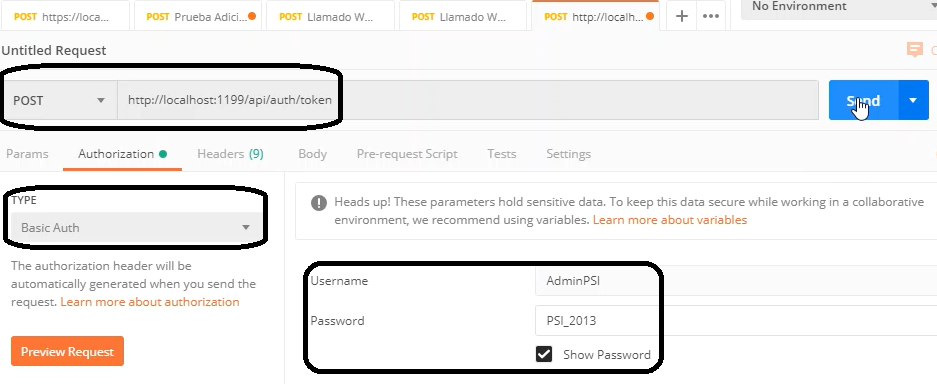


Ilustración Configuración en Postman para autenticar

Para la prueba correremos dentro de Visual Studio la solución donde el proyecto Base.Api está establecido como proyecto de inicio. Como cliente usaremos PostMan, para lo cual configuraremos la consulta al servicio como se muestra en la Ilustración 4. Tres puntos por resaltar: una llamada POST con la url ya comentada, la autenticación de tipo básica (BasisAuth) y con las credenciales respectivas (username y password). De resaltar que finalmente la autenticación se realiza con el servicio ServicioFactorySuiteProxy (WebServiceProxy.ashx).

Es importante resaltar la clase ServiceUrl en la cual esta configurado la ip o el dominio donde está el servicio de proxy mencionado en el párrafo anterior, la cual podemos encontrar en la carpeta mostrada en la Ilustración 5, y la implementación de la clase en la Ilustración 6. Igualmente, para configurar la fuente de la base de datos se indica en el webconfig en la entrada “DefaultConnection” y en “oracle.manageddataaccess.client” (<add name="Oracle Data Provider for .NET" invariant="Oracle.DataAccess.Client" description="Oracle Data Provider for .NET" type="Oracle.DataAccess.Client.OracleClientFactory, Oracle.DataAccess, Version=2.111.6.20, Culture=neutral, PublicKeyToken=89b483f429c47342" />) la cual observamos en la Ilustración 7. Para efectos del ejercicio nos estamos conectado al servidor de Pensemos SI donde está configurado el proxy, pero normalmente sería localhost ya que todos los servicios estarían en el mismo servidor.

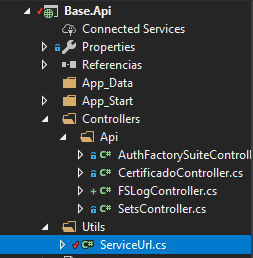


Ilustración Clase ServiceUrl dentro de la carpeta Utils

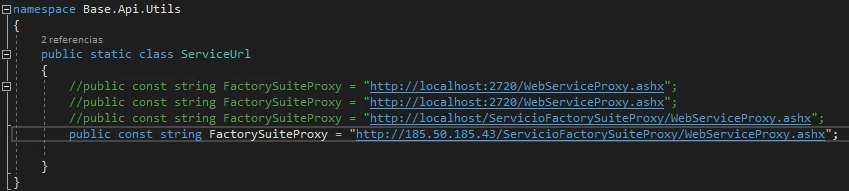
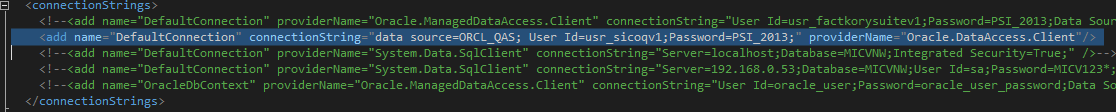


Ilustración Definición de la clase SeriveUrl

Finalmente, la respuesta enviada por el servicio base es mostrada en la Ilustración 8. La respuesta es mostrada en formatos json, siendo el atributo “id\_token” la que debemos conservar para futuras consultas (igualmente el cliente que se use para realizar las siguientes solicitudes al servicio).



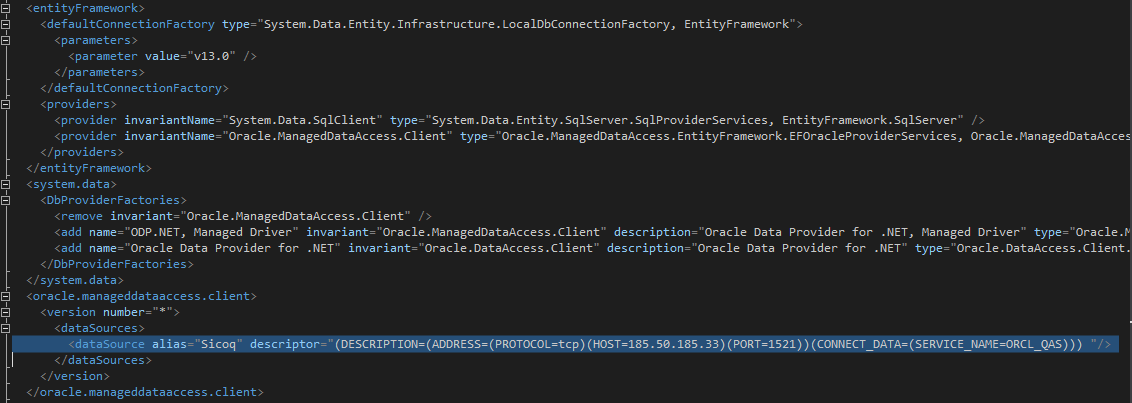


Ilustración Entrada del WebConfig donde se indica a cuál base de datos conectar

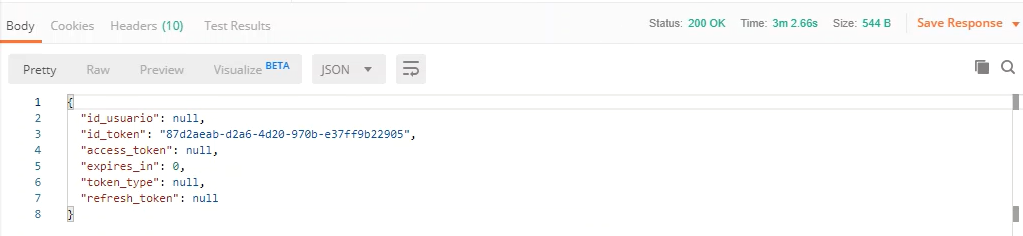


Ilustración Respuesta de la autenticación del Servicio Base

# Prueba del servicio “services-base”

La prueba del “services-base” lo realizaremos con una consulta sobre los certificados CCIE. En la Ilustración 9 apreciamos la clase “controller” a la cual pertenece la consulta en particular a probar. En la Ilustración 10 observamos la implementación del método que resuelve la consulta que vamos a probar. Observamos la url (“Route”) que debemos utilizar para obtener los resultados requerido.

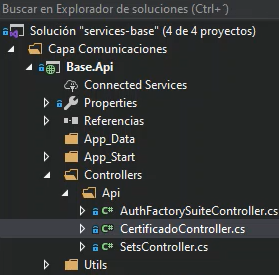


Ilustración API para consultar Certificados

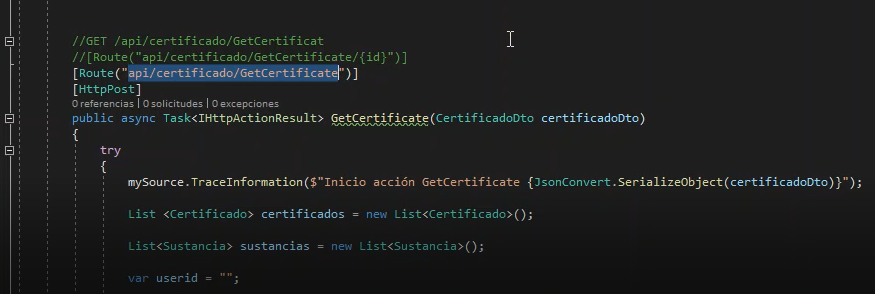


Ilustración Implementación del Método que resuelva la consulta a probar

En la Ilustración 11 observamos el mensaje tipo POST a enviar para realizar la consulta, en donde observamos la url ya comentada en el párrafo anterior, el tipo de autorización a usar es “Bearer Token” y parámetro “Token” correspondiente al obtenido en la prueba de autenticación anterior. En la Ilustración 12 observamos el body con el mensaje json el cual contiene “IdConsulta:”, “Codigo:” (en este caso CCCIE 82), y las coordenas (longitud y latitud).

En la Ilustración 13 mostramos la respuesta en formato json enviada por el servicio, tanto el certificado como las sustancias asociadas al certificado e igualmente las sucursales asociadas a las sustancias.

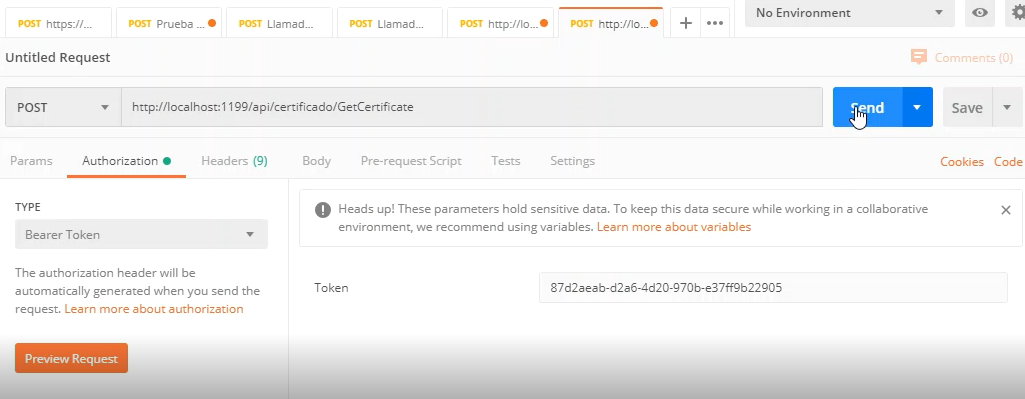


Ilustración Sección Authorization del mensaje a enviar

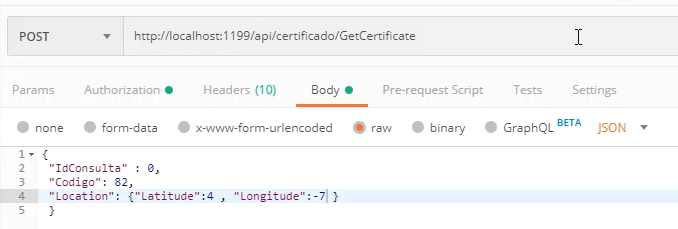


Ilustración Sección Body en formato en Json

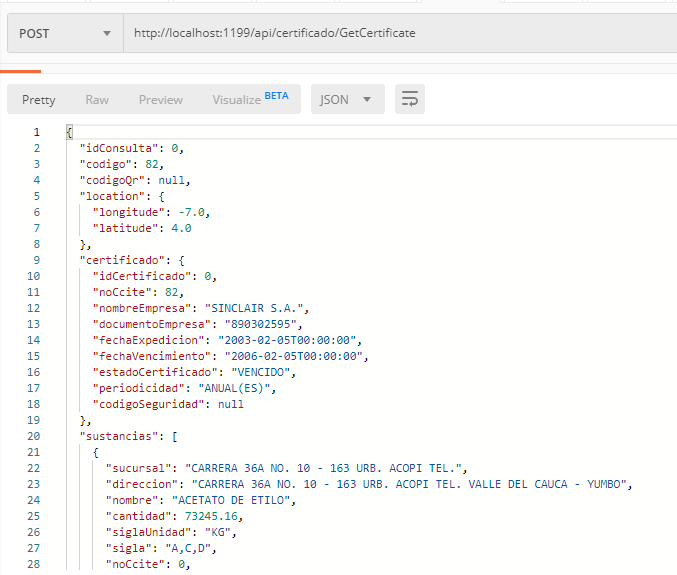


Ilustración Respuesta en formato en Json

# Agregar un nuevo servicio “services-base”

Para este ejercicio vamos a consultar la tabla “FSLogErrores” de Factory Suite, más específicamente los registrados en las ultimas 24 horas (hasta un día atrás). Lo primero es insertar en el archivo Queries.xml esta consulta como apreciamos en la Ilustración 15. Este archivo esta el proyecto en la ubicación indicada en la Ilustración 14., en la carpeta XML del proyecto Base.Data.

Posteriormente debemos agregar una clase a la carpeta “Models” del proyecto “Base.Model” (de la Capa Lógica Negocio) como apreciamos en la Ilustración 16. La implementación de la clase la apreciamos en la Ilustración 17. La manera más fácil de realizarlo es duplicar una clase existente de los modelos existente y ajustarlo cambiando el nombre de la clase y cambiando los miembros a los específicos de la consulta que va a representar.

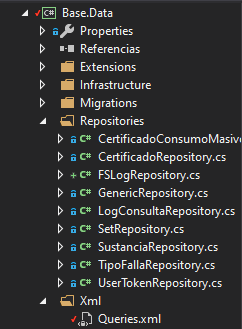


Ilustración Ubicación del archivo Queries.xml

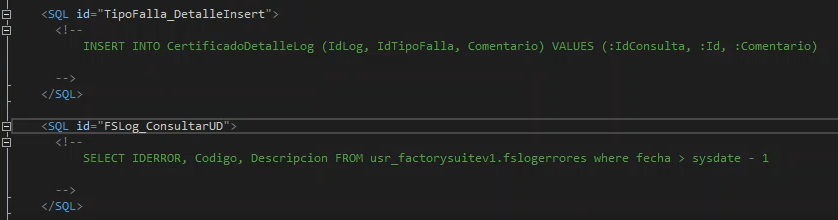


Ilustración Agregar consulta SQL a Queries.XML

En el siguiente paso agregamos una clase y una interface a la carpeta “Repositories” del proyecto “Base.Data” (de la Capa Datos) como apreciamos en la Ilustración 18. La implementación de la clase y de la interface la apreciamos en la Ilustración 19. Igual que la clase anterior la manera más fácil de realizarlo es duplicar una clase existente de los repositorios existente y ajustarlo cambiando el nombre de la clase y la interface.

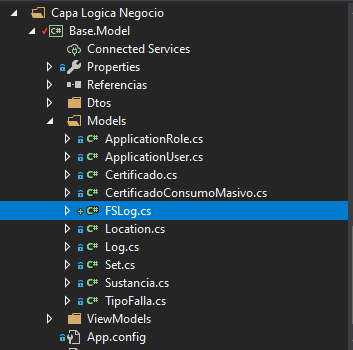


Ilustración Clase FSLog en Models (Base.Model)

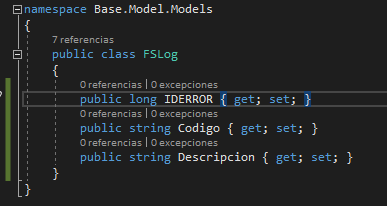


Ilustración Implementación de la clase FSLog

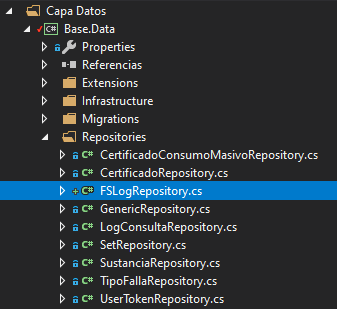


Ilustración Clase FSLogRepository en Repositories (Base.Data)

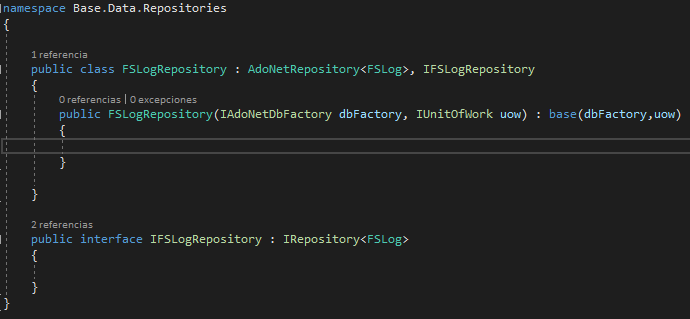


Ilustración Implementación de la clase FSLogRepository

En el siguiente paso agregamos una clase y una interface a la carpeta “Services” del proyecto “Base.Service” (de la Capa Lógica Negocio) como apreciamos en la Ilustración 20. La implementación de la clase y de la interface la apreciamos en la Ilustración 21. Igual que la clase anterior la manera más fácil de realizarlo es duplicar una clase existente de los servicios existente y ajustarlo cambiando el nombre de la clase y la interface.

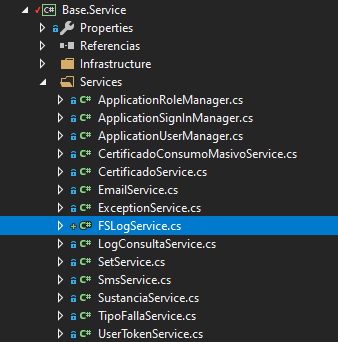


Ilustración Clase FSLogService en Services (Base.Service)

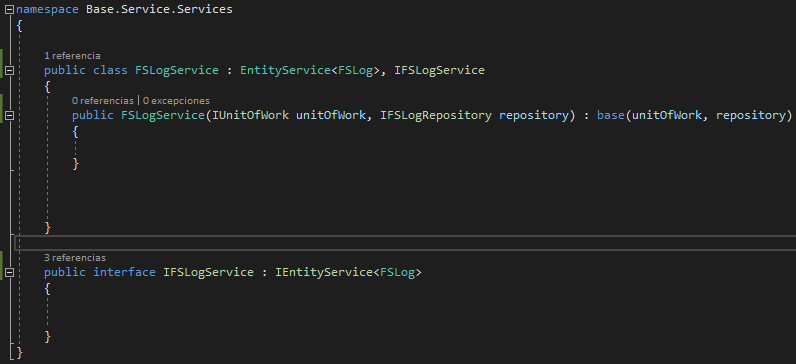


Ilustración Implementación de la clase FSLogService

Finalmente agregamos una clase de tipo “ApiController” a la carpeta “Controllers.Api” del proyecto “Base.Api” (de la Capa Comunicaciones) como apreciamos en la Ilustración 22, donde observamos los miembros: “\_FSLogService” (la cual construimos anteriormente), “\_userTokenService” usada para autenticarse y “mySource” usada para llevar un log. Los métodos son el constructor, “GetFSLogConsulta” la cual veremos en detalle la implementación, “GetToken” y “ValidateToken” para el apoyo de la autenticación.

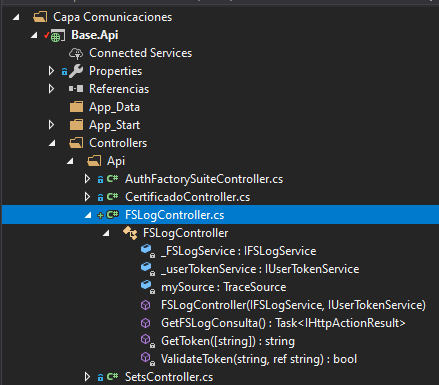


Ilustración Clase FSLogController en Controleler.Api (Base.Api)

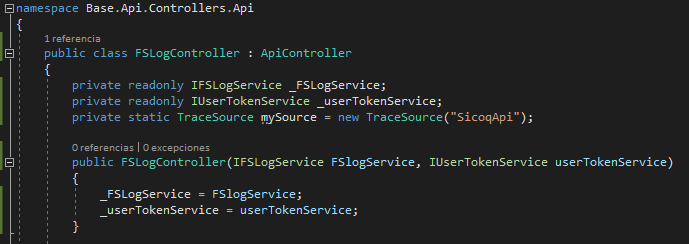


Ilustración Implementación de la clase FSLogController (Constructor)

La implementación del método constructor lo observamos en la Ilustración 23: “\_userTokenService” y “mySource” son estándar a todas las clases “ApiController” mientras “IFSLogService” es específica a esta implementación de acuerdo a lo que necesitemos, y por ende se pueden agregar otros servicios (en ese caso se debe agregar las mismas clases que ya explicamos para el nuevo servicio). La implementación de los métodos clase “GetToken” y “ValidateToken” la apreciamos en la Ilustración 24. Estos métodos son estándar para cada controlador y por ello se copian de otro existente.

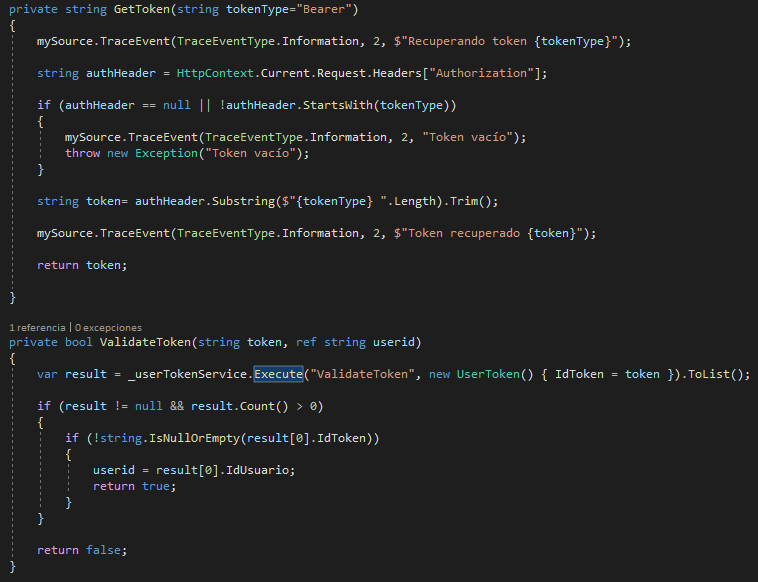


Ilustración Implementación de la clase FSLogController (GetToken y ValidateToken)

La implementación del método “GetFSLogConsulta” lo apreciamos en la Ilustración 25. Lo primero a resaltar es la anotación “[Route("api/FSLog/ConsultaUD")]” con la cual indicamos la url donde acceder este método. Lo siguiente es validar si se tiene un token vigente con los métodos “GetToken” y “ValidateToken” y llevando un registro de todas estas acciones con la ayuda de objeto “mySource”.

Una vez validado la autenticación en el servicio procedemos a crear una lista de FSLog donde se almacenar los registros devuelto por la consulta la cual ejecutamos en la llamada al método “\_FSLogService.Execute ("ConsultarUD", new FSLog()).ToList()”, en donde “ConsultaUD” es el nombre que le asignamos a la consulta en archivo “Queries.xml” (ver Ilustración 15) y FSLog es la clase que representa un registro de la consulta.

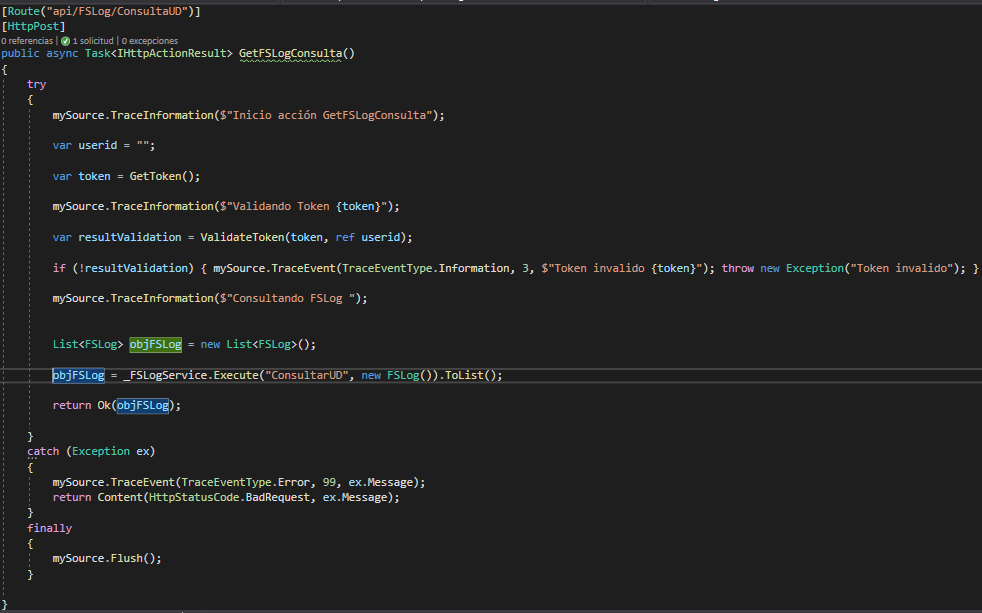


Ilustración Implementación del método GetFSLogConsulta de la clase FSLogController

En la Ilustración 26 mostramos la consulta al servicio desde Postman, en este caso sin parámetros. Y en Ilustración 27 apreciamos el resultado a este servicio que acabamos de implementar en este servicio base.

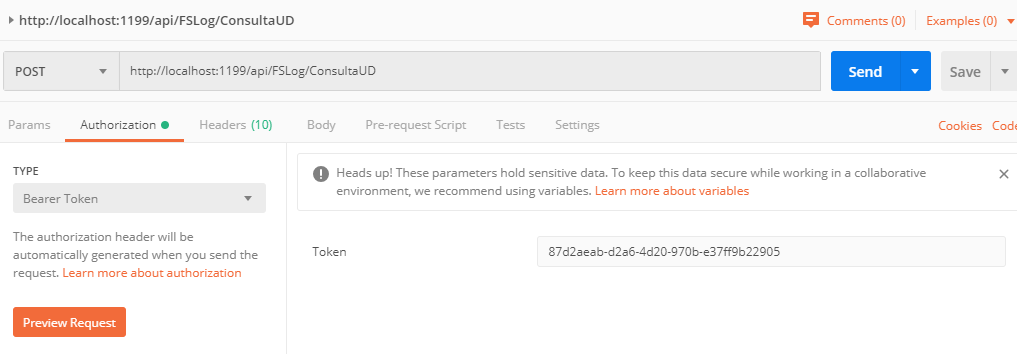


Ilustración Consulta al Servicio FSLog-ConsultaUD

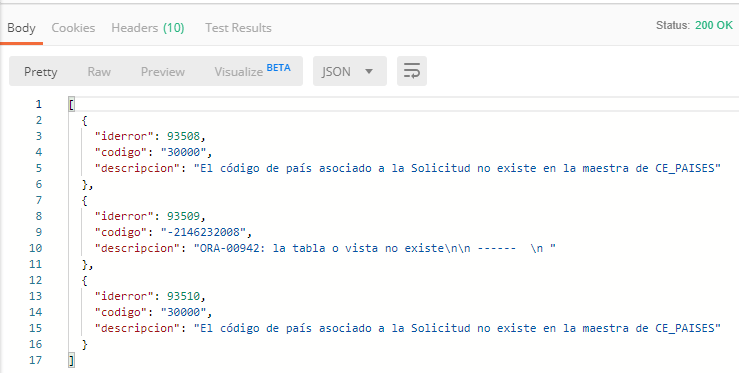


Ilustración Resultado de la consulta a FSLog-ConsultaUD