

Prueba técnica para Developer de Dynamics 365

* SU NOMBRE

FECHA

* Preparado por

Infoavan

26/10/2020

Versión 1.0 Final

|  |  |
| --- | --- |
|  |  |



CONTENIDO

[CONTENIDO 2](#_Toc69402921)

[TABLAS 3](#_Toc69402922)

[IMAGENES 4](#_Toc69402923)

[1. Escenarios de prueba 5](#_Toc69402924)

[1.1. Introducción 5](#_Toc69402925)

[1.2. Pre requisitos 5](#_Toc69402926)

[1.3. Escenario 5](#_Toc69402927)

[1.4. Requisitos 5](#_Toc69402928)

[1.4.1. Requisito 1 5](#_Toc69402929)

[1.4.2. Requisito 2A 5](#_Toc69402930)

[1.4.3. Requisito 2B 6](#_Toc69402931)

[1.4.4. Requisito 3 (O justificar si no es viable) 6](#_Toc69402932)

[1.4.5. Requisito 4 6](#_Toc69402933)

[1.4.6. Requisito 5 7](#_Toc69402934)

TABLAS

[Tabla 1: Alternativas con sus pros y contras. 7](#_Toc69402917)

[Tabla 2: Posibles opciones. 7](#_Toc69402918)

IMAGENES

**No se encuentran elementos de tabla de ilustraciones.**

# Escenarios de prueba

## Introducción

A continuación, se presentan los escenarios para la prueba técnica.

La prueba se deberá enviar en un tiempo máximo de 5 horas a partir de que el candidato decida que tiene tiempo para dedicarle.

Tiempo estimado: 5 horas

## Pre requisitos

Es necesario disponer de un entorno trial de Dynamics 365 CE para poder realizar esta prueba si quieres enviar la solución o código fuente implementado (IDEAL).

En caso de no disponer del entorno, se puede responder la prueba escribiendo el código más o menos exacto. Otra opción es utilizar LinqPad.

## Escenario

Queremos poder asociar una calificación numérica entre 1 y 10 a los contactos asociados a una cuenta. Cada contacto tendrá una tipología -no la tipología OOB, un campo nuevo- almacenada en un campo de tipo OptionSet (llamado customtype).

## Requisitos

## Requisito 1

Se desea que al entrar en el formulario principal de cuenta se muestre una notificación estándar de Dataverse de tipo warning, si la media de las calificaciones de todos los contactos es inferior a 4 y siempre que haya al menos un contacto asociado a la cuenta.

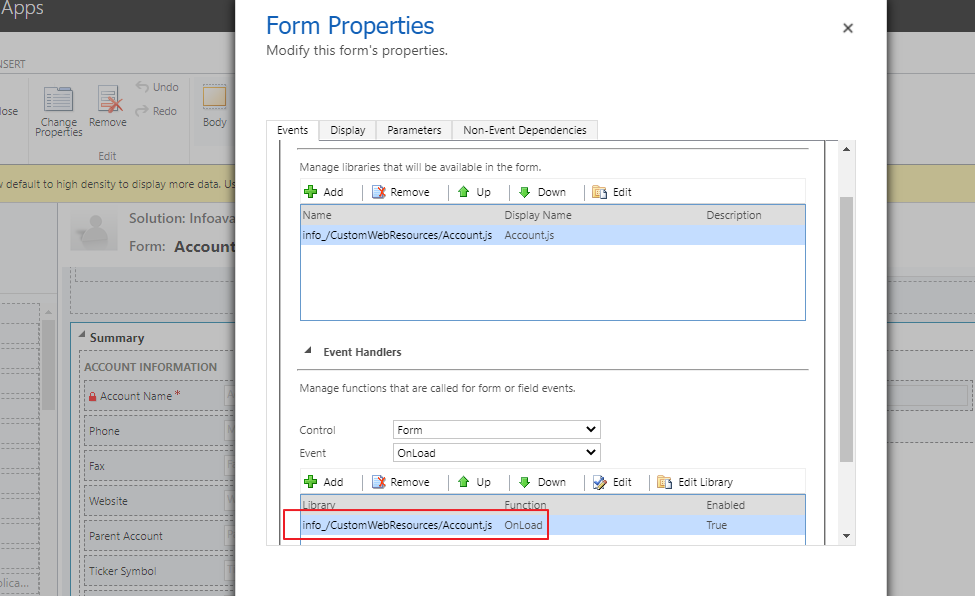
Ejercicio:

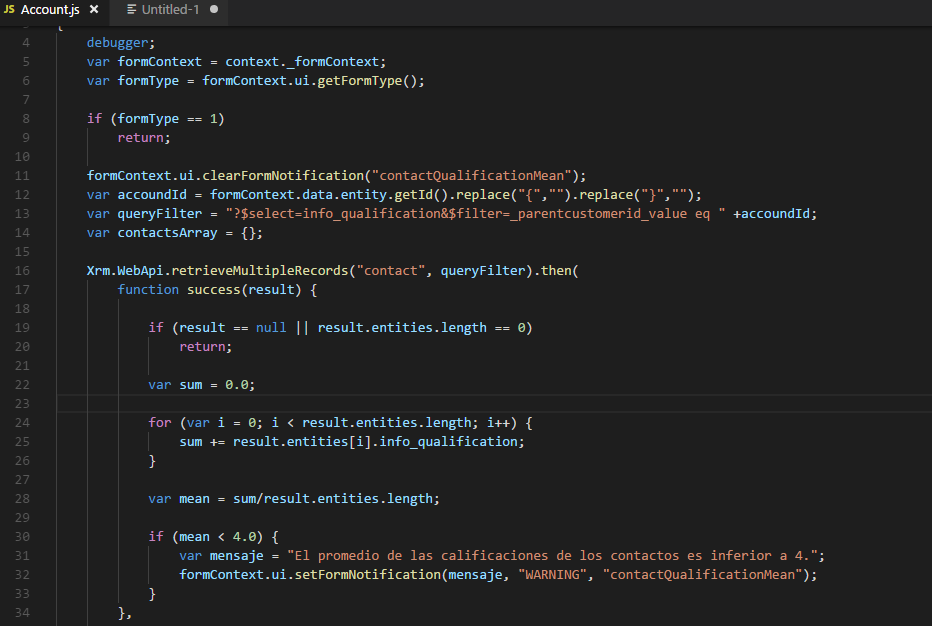
* Escribe a continuación un diseño técnico básico de la solución propuesta
* Implementa los cambios necesarios en un entorno Trial (Personalizaciones de la entidad y el código/ficheros necesarios para lograr lo anteriormente descrito.
* Crea una solución que se pueda volcar en otro entorno.

**SOLUCIÓN**

Se define una rutina de javascript en la carga del formulario Account.

Esta rutina básicamente lo que realiza es ir a buscar todos los contactos asociados a la cuenta, sumar la totalidad de calificaciones y dividirlo por la cantidad de calificaciones (promedio). Si el promedio da menor a 4 se muestra un formNotification. Caso contrario no se toma ninguna acción.





Dentro del repositorio GIT se puede observar el archivo js (Account.js) y una demo de este funcionamiento en video (VideoDemos\Requerimiento 1).

Se incluye solución administrada con todos los componentes en \SolucionesCRM\Infoavan\_1\_0\_0\_0\_managed - Requerimiento 1

## Requisito 2A

Una vez puesto en producción, el usuario de negocio decide que es interesante tener esta media disponible para cada registro en tiempo real/casi real y no sólo cuando se accede al formulario, de manera que esta media se pueda utilizar para reporting o exportar a reports Power BI o poder trabajar en Dataverse con vistas que utilicen la misma.

Expón las alternativas de diseño posibles respecto a la solución implementada, así como los pros y contras y en qué momento estaría disponible el valor del campo para cada una de ellas, de cara a que el usuario de negocio pueda decidir en función de los pros/Cons.

|  |  |  |  |
| --- | --- | --- | --- |
| Alternativa / Diseño | Pros | Contras | Comentarios |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

*Tabla 1: Alternativas con sus pros y contras.*

**SOLUCIÓN**

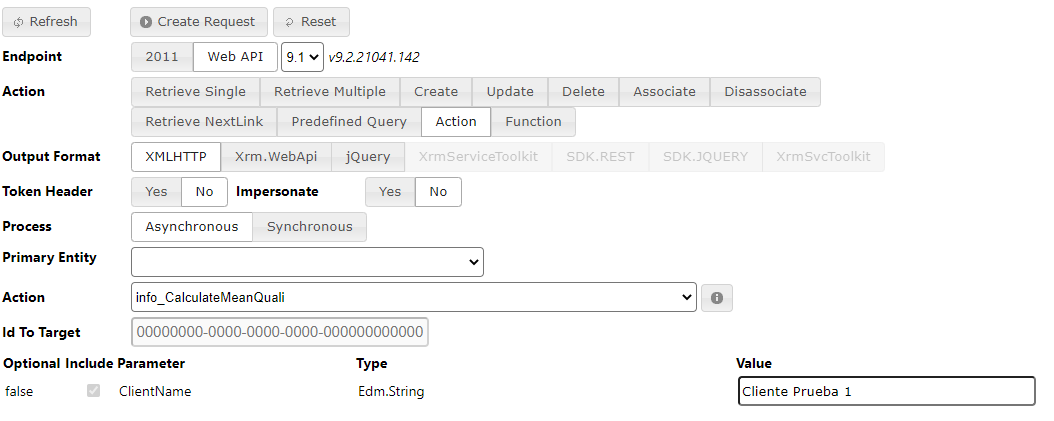
**Alternativa 1:**

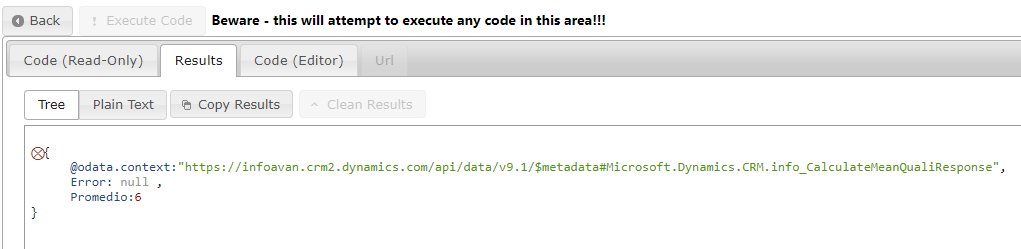
Custom Action (con Plugin Registrado) + Workflow asincrónico

Se define la siguiente estrategia.

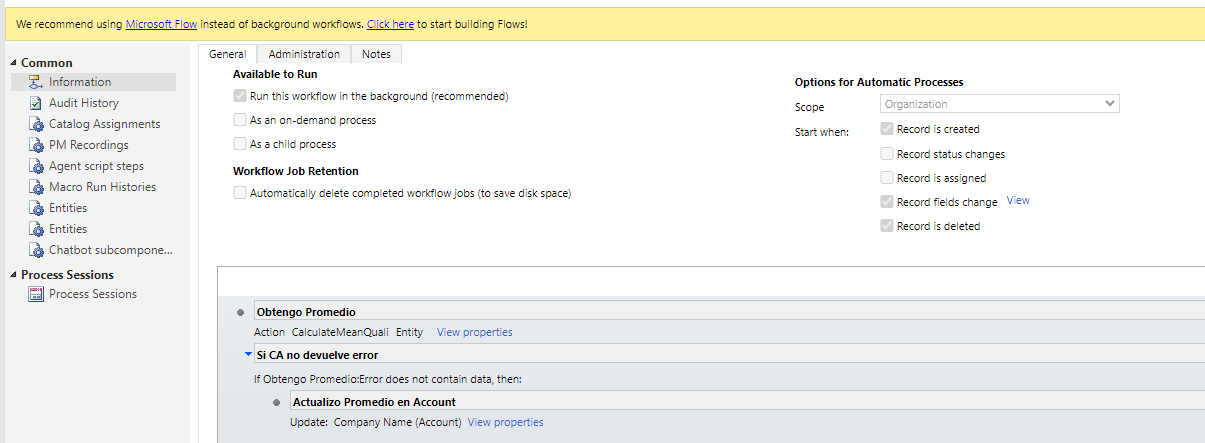
Se implementa un campo a nivel Account que consolidará los promedios de los contactos que tenga.

A continuación se define un Custom Action (info\_CalculateMeanQuali) que mediante un input de dato de cliente (en este caso el nombre), nos devuelve el promedio de sus contactos.





Si existe algún error al procesar este promedio se devuelve el error en el campo “Error”.



Luego se define un Workflow Asincrónico que ante el evento de que se cree/modifique la calificación/borre un contacto, se dispara.

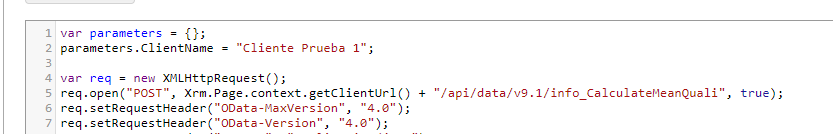
Como primer medida se llama al CustomAction en cuestión, enviándole el nombre del cliente asociado a ese contacto (campo name de Account – entidad relacionada). Si el CA no devuelve error se hace Update del cliente en cuestión dentro del campo info\_contactsqualmean.

**Pros:**

- Se actualiza en tiempo casi/real los valores de los promedios para cada cliente.

- Como el campo queda a nivel de account, se puede utilizar para popular vistas/dashboards, etc.

- Se puede obtener el resultado del promedio desde fuera de CRM consultando el Custom Action a través de la WebAPI de CRM:



- A futuro el calculo de promedio, puede modificarse por una necesidad particular del promedio, por ej: en vez de calcular el promedio de los contactos necesitasmos calcular el valor max/min/mediana, percentil, etc. Se establece un único punto de cálculo (el Custom Action) que es modular y se puede modificar a futuro sin interferir con el resto de los componentes asociados.

**Contras:**

- Como se trata de un workflow asincrónico (ya que si es sincrónico no se realizar las modificaciones al borrar un contacto), el cambio no es exactamente en tiempo real y está atado a la congestión de las operaciones asincrónicas.

- Si bien se utilizó un WorkFlow estándar el mismo se puede reemplazar por un flujo de Power Automate para mayor versatilidad.

Dentro del repositorio GIT se puede observar el proyecto de Visual Studio con el código fuente del Custom Action (Plugins\Infoavan.Plugins\Infoavan.Plugins\CalculateMeanQuali.cs) y una demo de este funcionamiento en video (VideoDemos\Requerimiento 2A).

Se incluye solución administrada con todos los componentes en \SolucionesCRM\Infoavan\_1\_0\_0\_0\_managed - Requerimiento 2A

## Requisito 2B

El contacto tiene su propio campo calificación, a partir del cual se calculan las medias. Sin embargo, se quiere que cada vez que se informe este campo, por cualquier vía, incluido integraciones, se actualice el campo estándar “name” del contact de manera que contenga la calificación entre paréntesis + el nombre y el primer apellido

Ejemplo: JAIME HERRERO, calificación 6 -- (6) JAIME HERRERO.

**Enumera las opciones posibles evitando las que impliquen escribir código.**

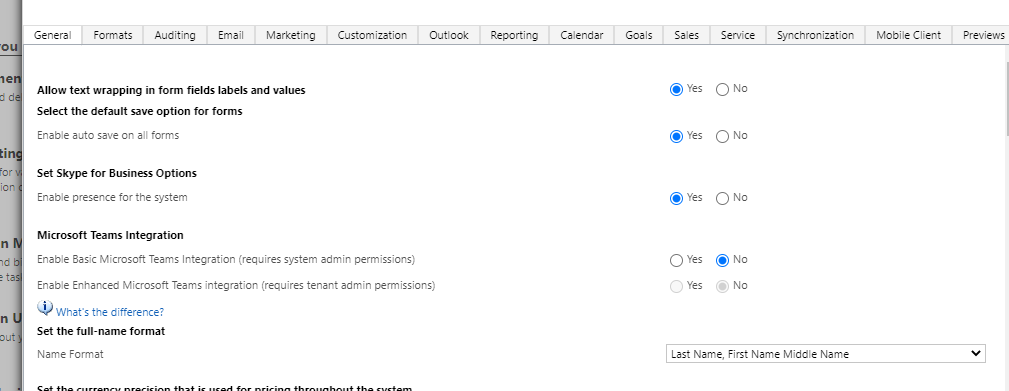
|  |  |
| --- | --- |
| Opción | Detalles |
|  |  |
|  |  |
|  |  |
|  |  |

*Tabla 2: Posibles opciones.*

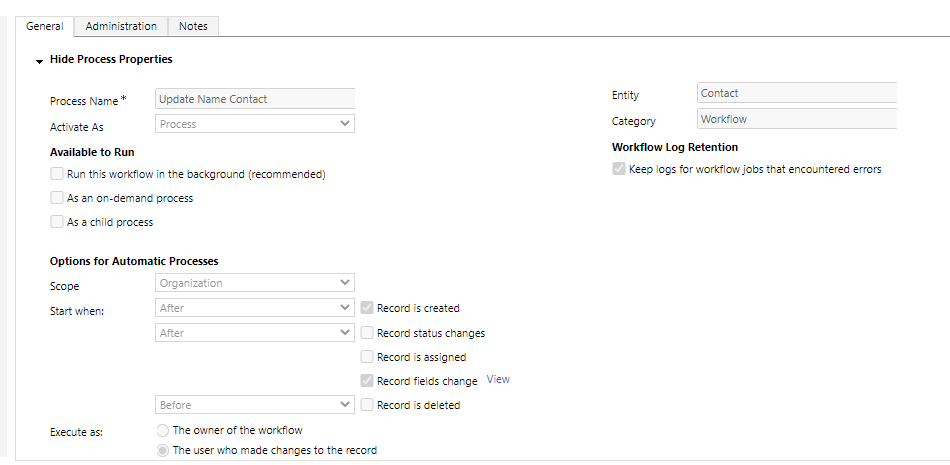
**SOLUCIÓN**

**Alternativa 1**

1)Setear el full-name format como Last Name, First Name Middle Name

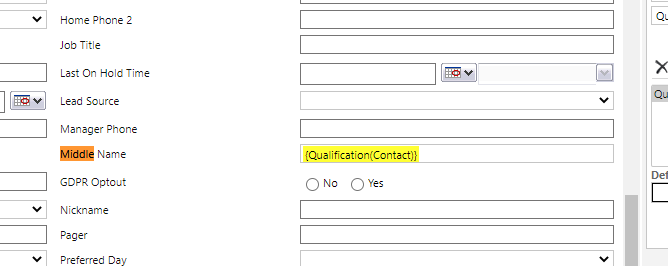


2) Crear un WorkFlow sincrónico en la creación/update (solo cuando se modifica la calificación) de los contactos.



3) Este Workflow deberá mapear el contenido de la calificación al Middle Name, con lo cual al modificar tanto First Name, como Last Name como Qualification se obtendrá el siguiente Full Name:

Last Name, First Name Qualification



Pros:

- Es robusto. Funciona en todos los casos, ya sea se modifique desde CRM o mediante algún tipo de integración

- Se realiza en tiempo real.

- No se aplica código.

- Se puede personalizar el formato de varias formas.

Contras:

- El formato del Full Name modificado en la configuración aplica tanto para contact como para account.

- Si se quiere ingresar el dato del Middle Name, se deberá ingresar dentro del First Name, no se podrán ingresar por separado.

Dentro del repositorio GIT se puede observar una demo de este funcionamiento en video (VideoDemos\Requerimiento 2B).

Se incluye solución administrada con todos los componentes en \SolucionesCRM\Infoavan\_1\_0\_0\_0\_managed - Requerimiento 2B

**Alternativa 2**

Se podría hacer algo similar utilizando reglas de negocio dentro del formulario. Este método permite realizar la misma funcionalidad sin utilizar código pero solamente funcionaría al modificar los campos desde el formulario de CRM, y no funcionaría al realizarse modificaciones desde integraciones o sistemas externos.

También se podría realizar algo similar utilizando flujos de trabajo de PowerAutomate.

## Requisito 3 (O justificar si no es viable)

Si consideras que plugin es una opción válida en el punto 1.4.2 (requisito 2A), escribe una implementación de este plugin, regístrala y explica las decisiones tomadas a la hora de registrar.

Explica y muestra pantallas de los pasos a seguir para debugar tu plugin en Visual Studio con la ayuda de Plugin Registration Tool en caso de que recibas un error persistente y no dispongas de trazas suficientes.

**SOLUCIÓN**

Se definen los pasos para implementar un Plugin registrado en la ejecución de un Custom Action.

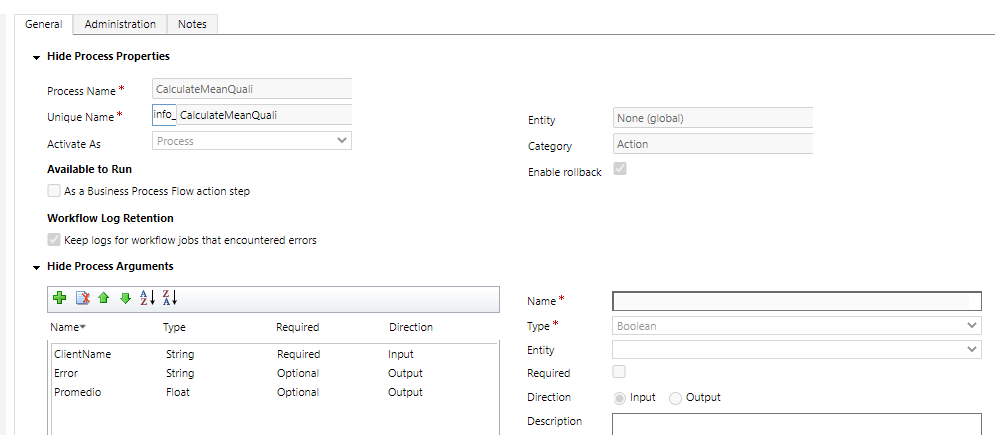
1) Se crea el CustomAction con los siguientes parámetros:

Unique Name: info\_CalculateMeanQuali

Entity: None (global)

Process Arguments:

* ClientName – String – Required – Input Parameter (Acá se enviará el nombre del cliente)
* Error – String – Optional – Output Parameter (Este parámetro se devolverá si la ejecución del Plugin devuelve errores)
* Promedio – Float – Optional – Output Parameter (Si el Plugin ejecuta de forma correcta devolverá el promedio de calificación de los contactos asociados al cliente.



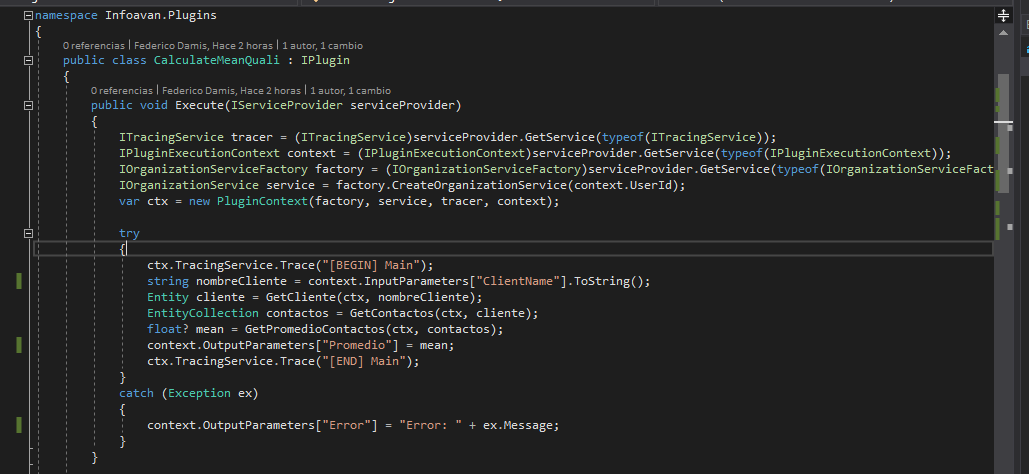
2) Acto seguido se escribe el Plugin en C#. La rutina supone los siguientes pasos:

Obtengo el nombre del cliente enviado como input y busco el registro en la entidad Account.

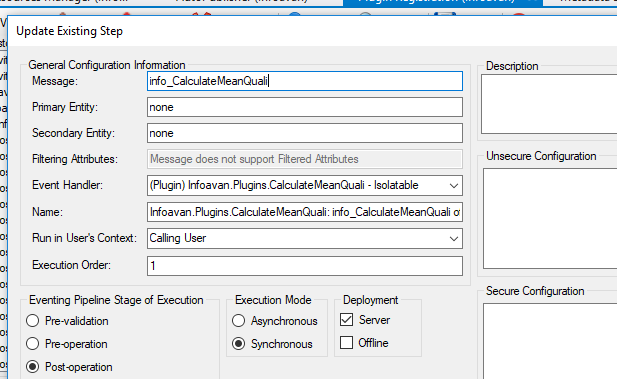
Obtengo un EntityCollection con todos los contactos asociados a ese cliente.

Recorro todos los contactos obtenidos (con un foreach) y calculo un valor global promedio a partir del valor de calificación de cada contacto.

Si ocurre alguna Excepción, devuelvo el resultado en el parámetro “Error”.

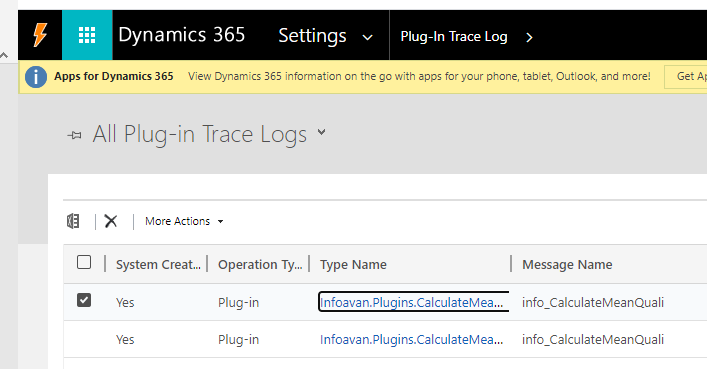


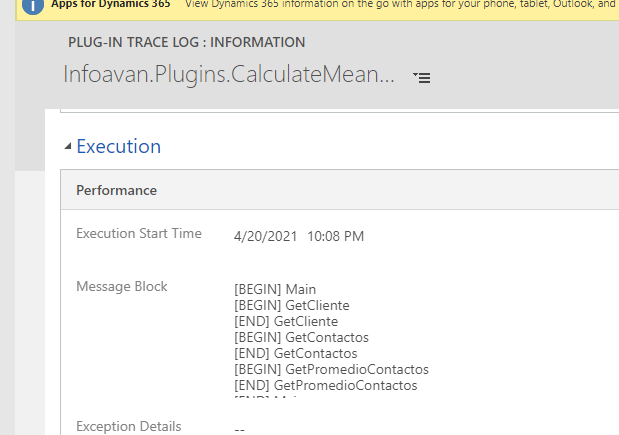
3) Se registra el Plugin para ejecute a través del Sandbox mediante Plugin Registrator y se crea el siguiente step de plugin:



4) Se crea un WorkFlow asincrónico según los pasos descritos en el Requisito 2A, que garantizarán la ejecución del CustomAction.

5) En el caso de contar con errores se puede proceder a analizar las trazas invocadas en el Plugin:





6) En el caso de que se quiera realizar debug sobre el Plugin se deberá realizar lo siguiente:

**PENDIENTE POR NO PODER CONECTAR PLUGIN REGISTRATION TOOL**

## Requisito 4

Supongamos que tenemos un servicio externo para recibir una información de viabilidad de un contacto desde un ERP central, accesible mediante un token OAuth.

Se trata de una llamada que se utiliza muchas veces desde Dataverse y su implementación varía mucho, por lo que se desea centralizar la implementación y que este servicio esté disponible desde workflows, JavaScript y Power Automate. ***Un mismo código invocable desde cualquier parte de Dataverse***.

Supongamos que la URL del servicio es <http://serviciocontacto.azurewebsites.net> y que devuelve una estructura formada por un código de resultado (code) y una descripción (description), en formato JSON. El payload del servicio sólo recibe como entrada un parámetro “contactid”.

Si el campo código es -1 significa que ha habido un error de negocio (no de comunicación) y su descripción vendrá en el campo “description”. Si la operación es satisfactoria, el servicio devuelve un código “1” y description el texto de actualización.

Describe la mejor opción posible para que el código pueda ser reutilizable desde estas tres plataformas (requisito indispensable, la llamada al servicio externo tiene que ser directamente invocable desde cualquier parte de Dataverse o desde Power Automate)

Escribe a continuación un prototipo de código que se comunica con el servicio, así como el tratamiento de la respuesta recibida

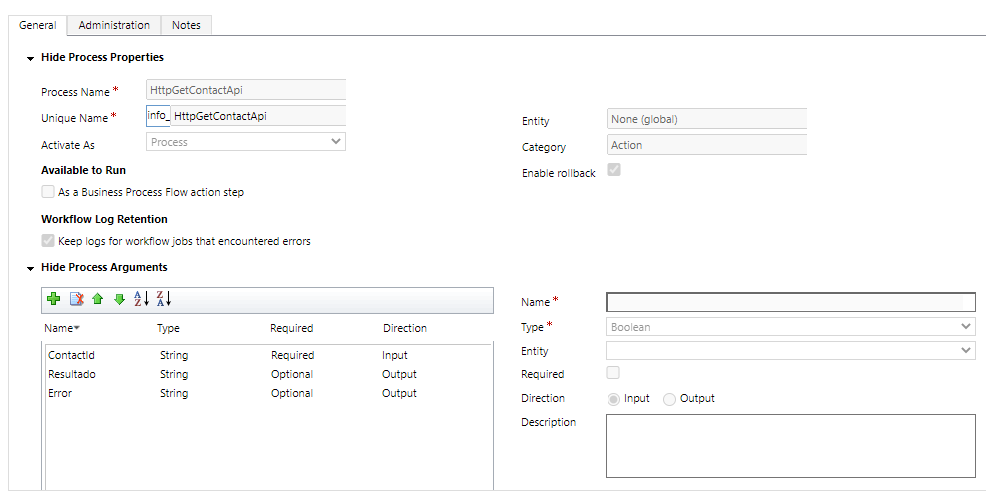
**SOLUCIÓN**

1) Se define un Custom Action con los siguientes parámetros:

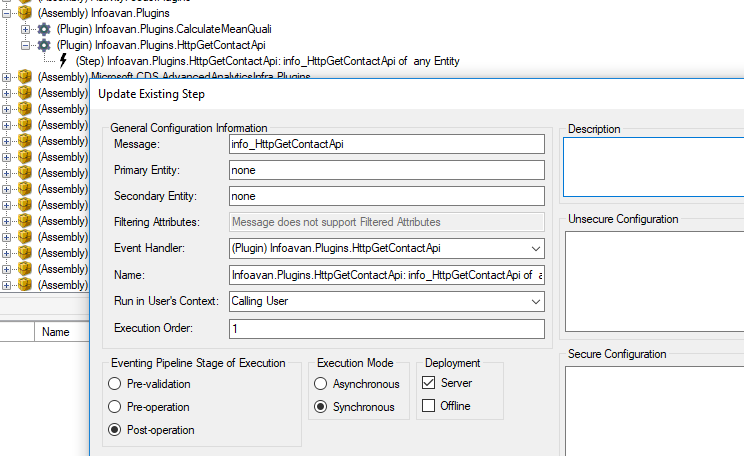
a) ContactId – String – Required – Input – Acá se enviará el GUID del contacto

b) Resultado – String – Optional – Output – Acá se devolverá el resultado del servicio invocado si la comunicación fue correcta.

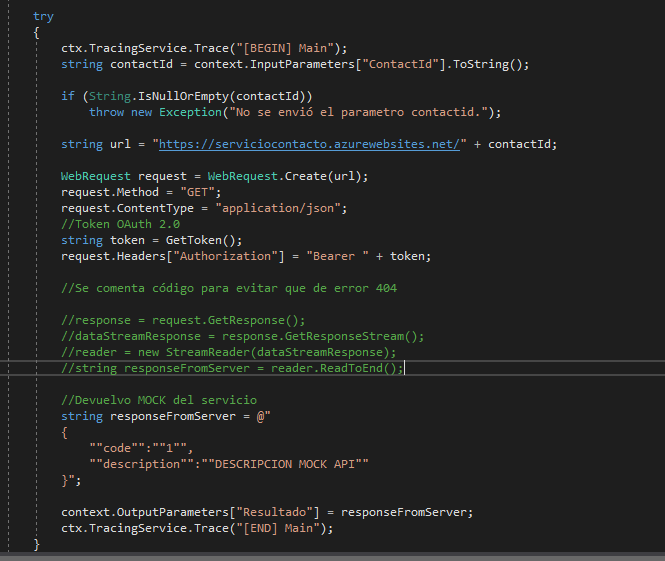
c) Error – String – Optional – Output – Acá se devolverán los errores/excepciones no controladas, además de los errores de comunicación con el servicio invocado.



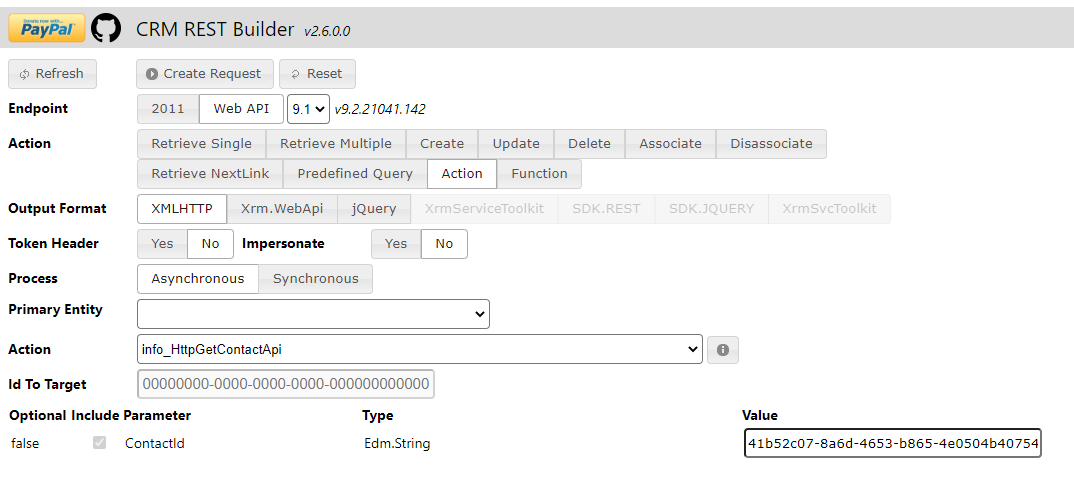
2) Se registra un Plugin a ser ejecutado cuando se invoca al CustomAction:

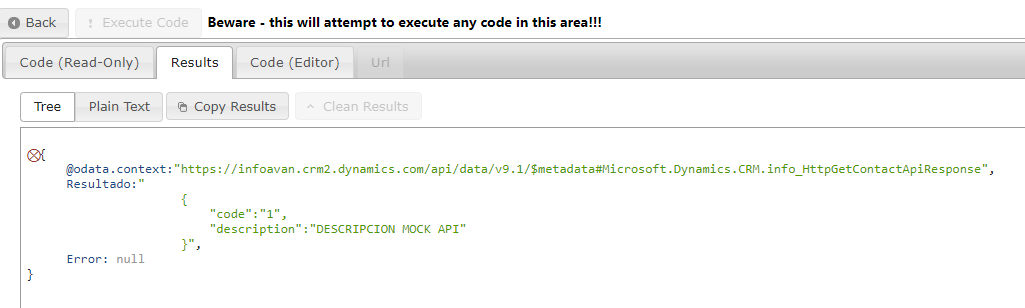


Básicamente lo que realiza este Plugin, es tomar el ContactId enviado, agregarlo como sufijo a una url determinada y realizar un WebRequest. Si la invocación al servicio es correcta se devuelve en el parámetro “Respuesta” la salida de la invocación al servicio en JSON.



Este Custom Action se puede invocar tanto por Javascript, como a través de un WorkFlow o PowerAutomate.

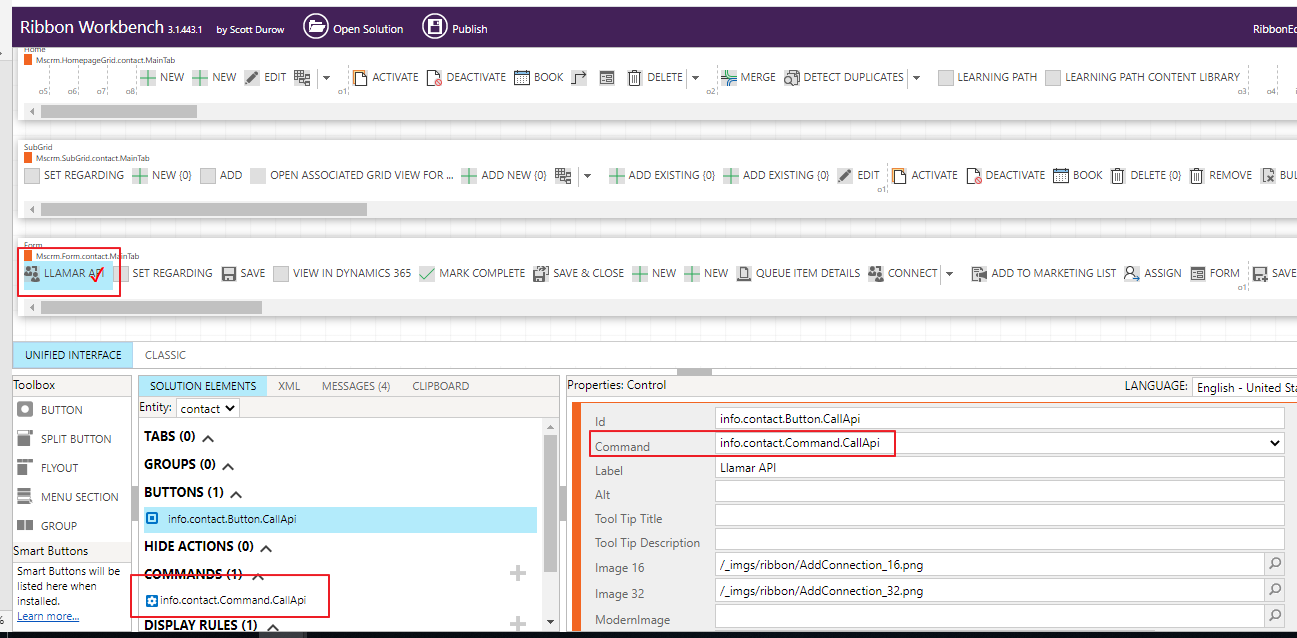




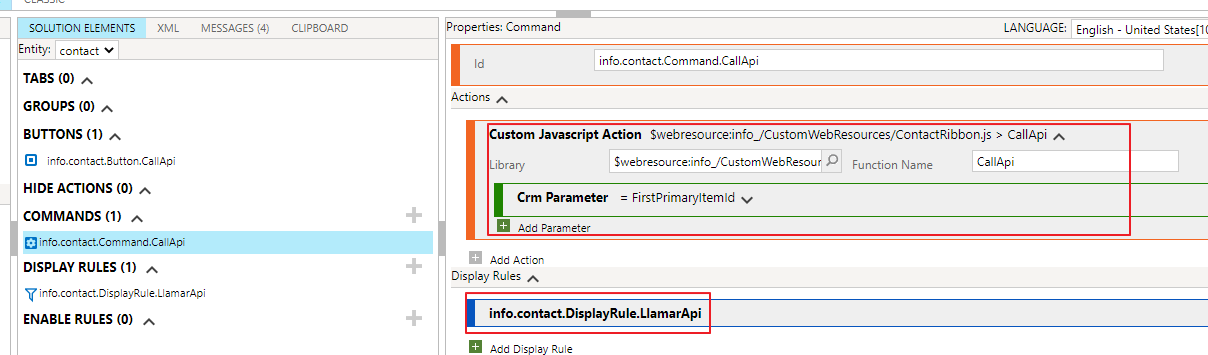
Invoca a demanda el mecanismo implementado desde un nuevo botón del ribbon del formulario de contacto, que sólo sea visible cuando el contactid está informado (cuando el formulario NO está en modo creación).

**SOLUCIÓN**

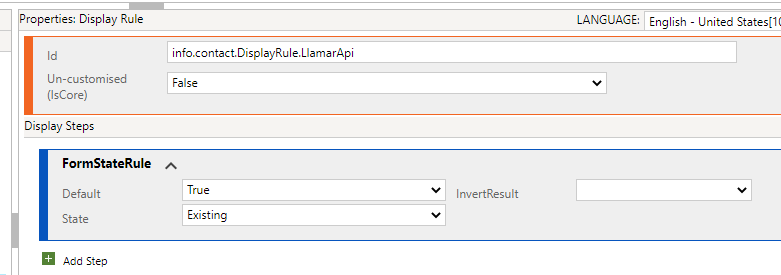
1) Se define un botón mediante RibbonWorkbench con un determinado Command:



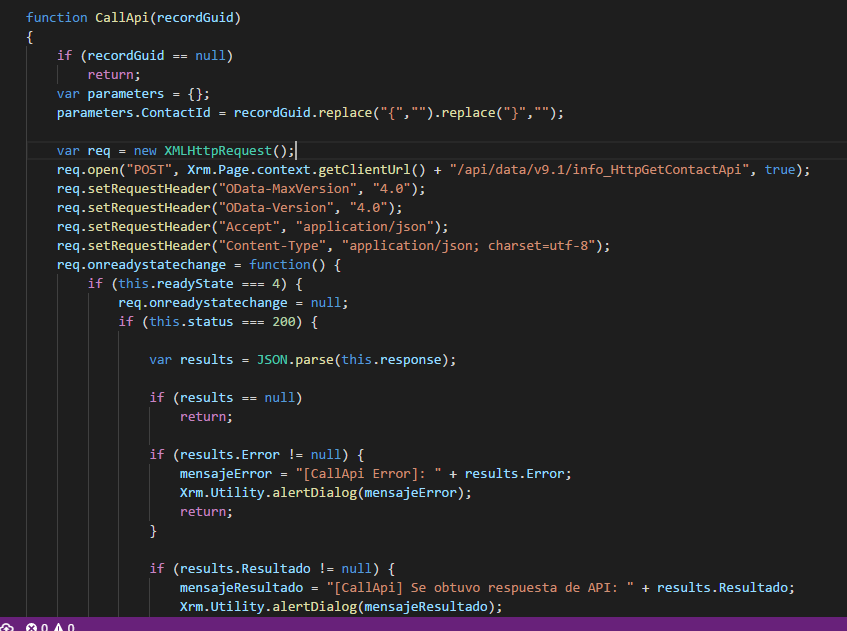
2) Dentro de ese Command definimos un Action y una Display Rule:



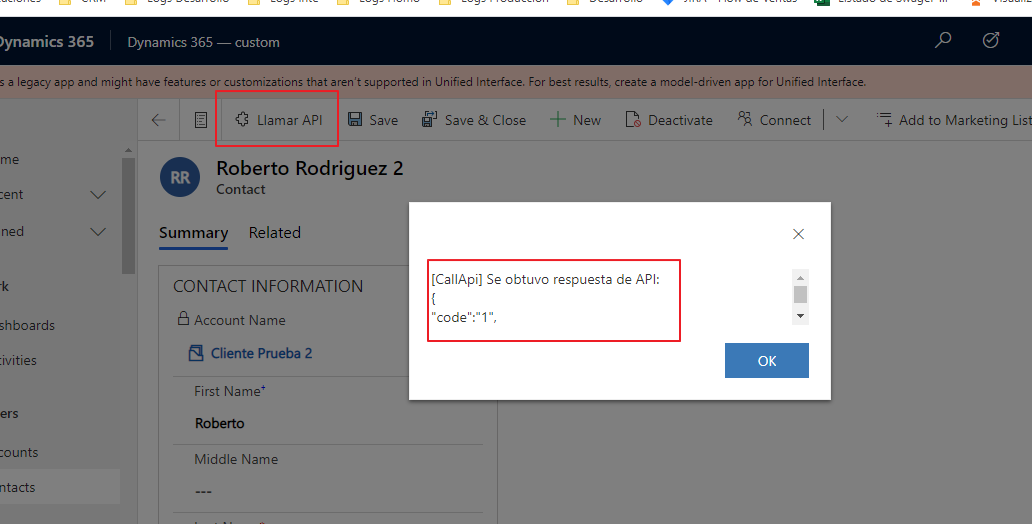
3) Dentro de esa Display Rule, la definimos como sigue:



4) Definimos nuestro Javascript que hace el llamado al CustomAction:



5) Utilizamos el botón para disparar el Custom Action, luego el Plugin y por último el llamado a la API, que nos devuelve la información.



Dentro del repositorio GIT se puede observar el proyecto de Visual Studio con el código fuente del Custom Action (Plugins\Infoavan.Plugins\Infoavan.Plugins\HttpGetContactApi.cs), el archivo JS ContactRibbon.js y una demo de este funcionamiento en video (VideoDemos\Requerimiento 4).

Se incluye solución administrada con todos los componentes en \SolucionesCRM\Infoavan\_1\_0\_0\_0\_managed - Requerimiento 4

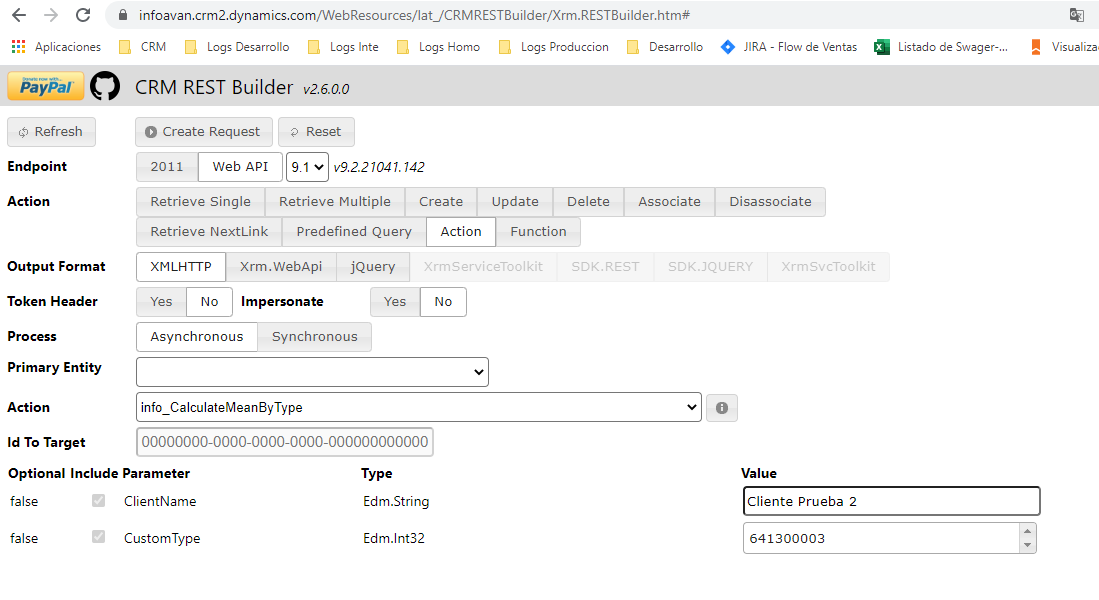
## Requisito 5

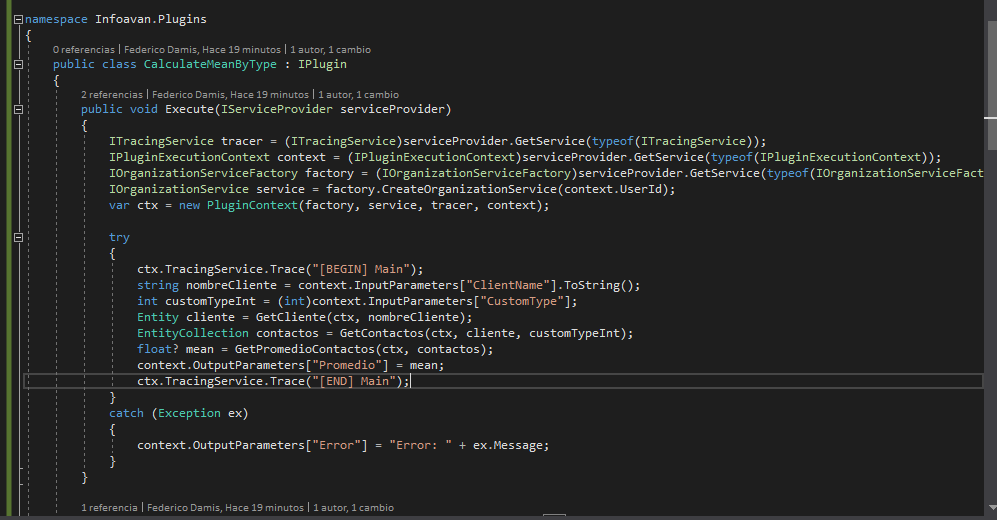
Por último, realiza un diseño con la mejor opción para poder conseguir desde JavaScript, workflow o código de servidor la media de calificación por un tipo concreto de contacto para una cuenta.

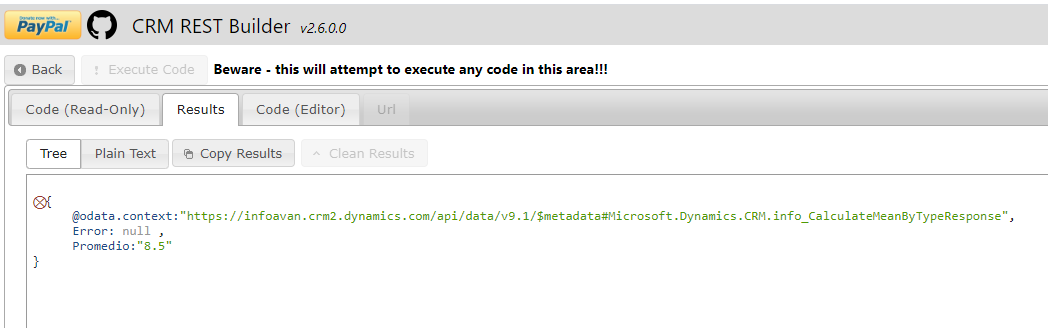
Recuerda que cada contacto tiene su tipología en el campo creado (customtype) al principio del ejercicio.

**SOLUCIÓN**

De la misma forma que antes se define un Custom Action con un Plugin asociado, pero agregándole como parámetro de entrada el tipo de contacto.







Dentro del repositorio GIT se puede observar el proyecto de Visual Studio con el código fuente del Custom Action (Plugins\Infoavan.Plugins\Infoavan.Plugins\CalculateMeanByType.cs)

y una demo de este funcionamiento en video (VideoDemos\Requerimiento 5).

Se incluye solución administrada con todos los componentes en \SolucionesCRM\Infoavan\_1\_0\_0\_0\_managed - Requerimiento 5