REST WEBSERVICE APPLICATION

Manual Español

20 DE AGOSTO DE 2023

JUAN CARLOS SEPULVEDA

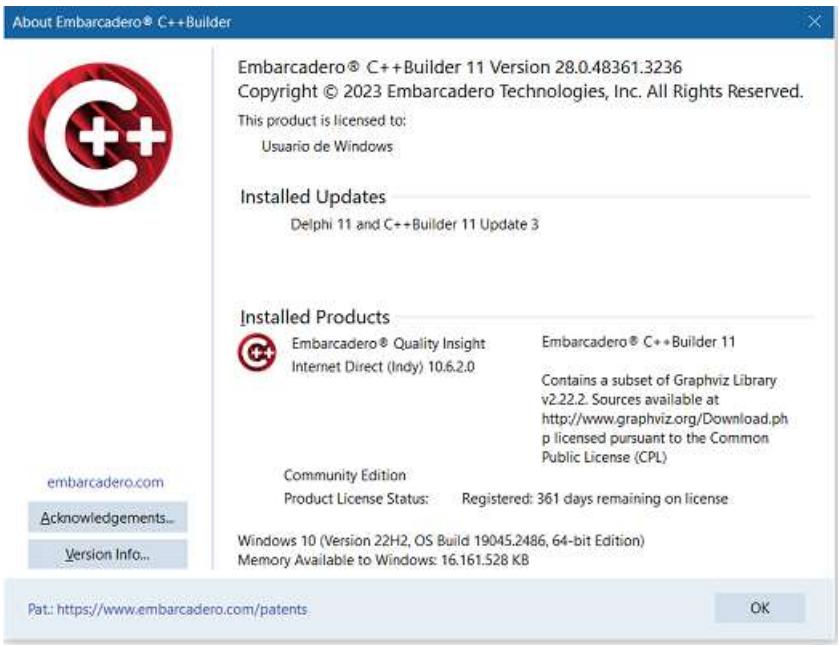
Contenido

Descripción de la aplicación.

Herramientas utilizadas.

La aplicación “Rest WebService Application” fue hecha usando la herramientas:

Embarcadero® C++ Buider 11.



Se uso la licencia versión “Community Edition” cuya licencia es gratis por 1 año, incluso para realizar aplicaciones comerciales, aunque tiene algunas limitaciones como son la ausencia de algunos componentes que están presente en las versión para empresa, entre otras limitaciones. Entre los componentes ausentes están los componentes de la paleta “DataSnap” y “REST Server”.

Al no estar presente estos componentes se utilizo el componente **TIdHTTPServer** para implementar el servidor REST

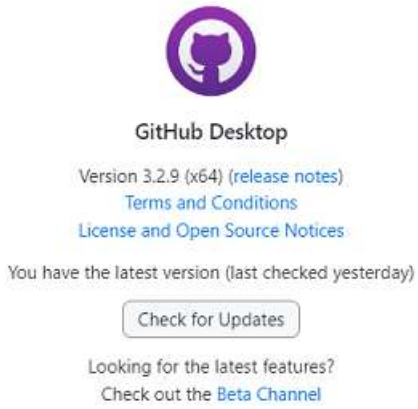
REST Debugger 10.5

Para depurar la aplicación se utilizó la herramienta “REST Debugger 10.5” simulando las peticiones JSon

SQLite Maestro.

Para el diseño y trabajo con la BBDD.

GitHub Desktop



Para subir los códigos fuentes al repositorio de GitHub.

Componentes utilizados

Para el desarrollo de la aplicación se utilizaron los siguientes componentes.

Paleta Indy Servers

TIdHTTPServer : Fue utilizado para recibir las peticiones HTTP.

Este componentes se utilizó al no estar presente los componentes de la paleta “REST Server” . El componente **TIdHTTPServer** par se usó para implementar el servidor REST. Se necesitan tener conocimientos de TCP y HTTP para usar dicho componente, además e una mayor cantidad de código que si se usa el componente “REST Server”

De este componente se utilizaron los eventos:

OnCommandGet: Para manipular las peticiones de tipo GET y POST.

OnCommandOther: Para manipular las peticiones de tipo PUT y DELETE.

Paleta FireDAC

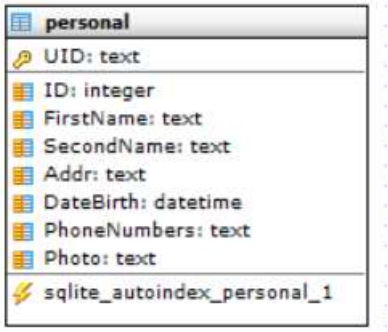
TFDConnection: Fue utilizado para realizar la conexión con la BBDD SQLite .

TFDQuery: Fue utilizado para realizar las consultas SQL la BBDD.

Base de Datos

Se utilizo una BBDD de tipo SQLite, conteniendo una sola Tabla denominada

Personal con los siguientes campos:



Como llave se usó el campo UID y en dicho campo la apelación almacena un Identificador universal conocido también como UUID, muy útil cuando se quiere utilizar un identificador que sea único universal. Se uso también un campo ID numérico para identificar al contacto con fines didácticos y de mostrar el uso de la aplicación trabajando. Podría haberse usado este campo como llave usándolo de tipo autoincremento, pero con las siguientes desventajas:

1. Limitación en la selección de valores: Los campos de tipo autoincremento solo permiten valores numéricos consecutivos, lo que limita la selección de valores personalizados.

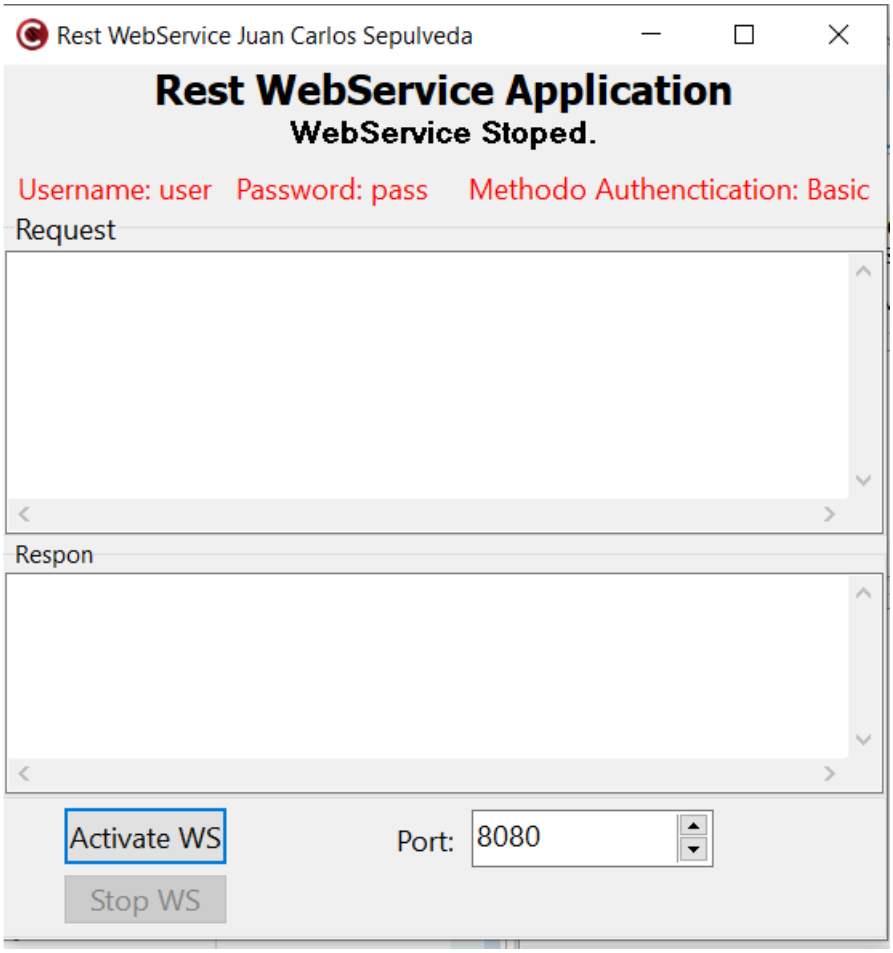
2. No se puede modificar el valor: Una vez que se ha asignado un valor a un campo auto incremental, no se puede modificar manualmente. Esto puede ser un problema si se necesita cambiar un valor específico en el futuro.
3. Problemas de sincronización: Si se trabaja con varias bases de datos y se utiliza un campo auto incremental, puede haber problemas de sincronización entre las diferentes bases de datos.
4. Problemas de escalabilidad: Si la base de datos crece demasiado, puede haber problemas con los campos auto incrementales, ya que pueden llegar a su límite máximo y no poder generar nuevos valores.
5. Problemas de seguridad: Si se utiliza un campo auto incremental como clave principal, puede haber problemas de seguridad, ya que los usuarios pueden adivinar fácilmente los valores de los registros.

Por supuesto ningunas de estas desventajas son importantes en esta aplicación pero para que se tenga una idea de cómo debe hacerse.

Nota: En el caso de las imágenes se guarda el enlace y se asume que físicamente están almacenadas en la nube o en un servidor externo.

Descripción de la aplicación.

La aplicación es una aplicación del tipo “Stand Alone”, no necesita ejecutarse dentro de ningún Servidor Web, como IIS, apache u otro. Al levantarla se muestra la siguiente interfase:



La aplicación al ser de prueba tiene fijado en el código los valores de UserName y Password. Actualmente usa los siguientes valores fijados en el código.

Username: user Password:pass

En una aplicación real estos valores deberían de cargarse desde la BBDD para poder usar varios usuarios e incluso dar una respuesta personalizada en función del usuario conectado.

Antes de “Activar” el servidor puede seleccionarse cual será el puerto de escucha. Por defecto usa el 8080, pero este puede ser cambiado mientras la aplicación no está activa.

Los “Memo” etiquetados como “Request” y “Respon” son usados para mostrar el logs de lo enviado, recibido y ocurrido en los mensajes Request y Respons. En una aplicación real esta información debería poder salvarse y además tener una estampa de tiempo.

Usando la aplicación

Instale la aplicación desde el instalador. Se recomienda no instalarla en los directorios Program File pues incluye una BBDD que se instala en el mismo directorio de instalación y las directivas de seguridad de Windows no permiten acceder sin permisos de Administacion a los archivos en esas carpetas.

Copie el instalador en la misma carpeta donde desea instalarlo para que no necesite cambiarlo de directorio

Para probar la aplicación debe usarse algún cliente.

Los clientes que se usaron durante el desarrollo fueron principalmente el navegador “Web Firefox 116.0.3 (64-bit)” y el “REST Debugger 10.5”

Probando solicitudes del Tipo GET.

Se selecciono el puerto 8095 para realizar las pruebas de este manual. Pudo ser otro.

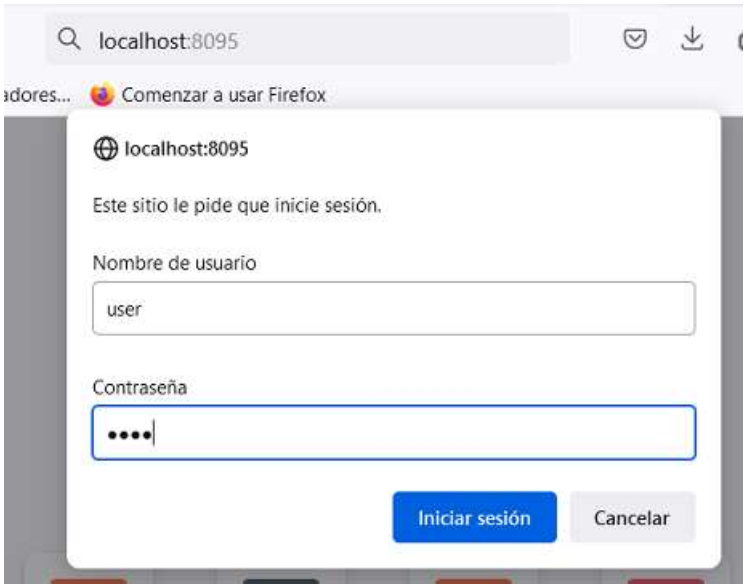
Usando el Firefox

La primera vez.

Al Introducir por primera vez la URL se solicitara contraseña

Introducir: http://localhost:8095/

Con lo cual pedirá usuario y contraseña:



Solicitando todos los datos

Para solicitar todos los datos basta con introducir la URL: http://localhost:8095/

Una vez entrado los valores se mostrara en el navegador todo el contenido de la base de datos, algo similar lo siguiente:

[{"UID":"97F9D2BF-23BB-4A37-B851-9D83AC675FAA","ID":1,"FirstName":"Juan","SecondName":"Carlos","Addr":"Marianao, Habana","DateBirth":"1976-08-01 00:00:00","PhoneNumbers":"55525659","Photo":"","Age":20,"UID":"4AC7B2C5-6AF7-49B0-B090-24F55A1C86F7","ID":2,"FirstName":"Lazaro","SecondName":"Sep","Addr":"La Lisa, Habana","DateBirth":"2020-10-16 00:00:00","PhoneNumbers":"5555555","Photo":"","Age":3,"UID":"BD2B7FD0-55D8-4987-B77E-145208F2D122","ID":3,"FirstName":"Lucia","SecondName":"Sep","Addr":"Camaguey, Camaguey","DateBirth":"2020-10-16 00:00:00","PhoneNumbers":"4444444","Photo":"","Age":3,"UID":"{220CDA9E-E91E-492B-B15E-0BA17B00BB64}","ID":222,"FirstName":"Mama","SecondName":"Moya","Addr":"UCI,116 104","DateBirth":"","PhoneNumbers":"5678","Photo":"www.pictures.com\/yuri\/carnet.jpg","Age":80,"UID":"{73DFF37B-100E-4261-8F32-5F49994D8D84}","ID":221,"FirstName":"Yurisleidy","SecondName":"Moya","Addr":"UCI,116 104","DateBirth":"","PhoneNumbers":"5678","Photo":"www.pictures.com\/yuri\/carnet.jpg","Age":31}]

En este caso se devolvieron 5 valores en formato JSon.

Solicitando contactos específicos por un campo

Si se deseara solamente retornar los valores de un determinado campo. Por ejemplo: ID=1. Teclear una de las 2 URL siguiente. Son equivalentes.

http://localhost:8095/1

http://localhost:8095/?ID=1

Resultado devuelto:

[{"UID":"97F9D2BF-23BB-4A37-B851-9D83AC675FAA","**ID":1**,"FirstName":"Juan","SecondName":"Carlos","Addr":"Marianao, Habana","DateBirth":"1976-08-01 00:00:00","PhoneNumbers":"55525659","Photo":"","Age":20}]

También se puede realizar solicitudes por otros campos

http://localhost:8095/?Age=3

Resultado devuelto:

[{"UID":"4AC7B2C5-6AF7-49B0-B090-24F55A1C86F7","**ID":2**,"FirstName":"Lazaro","SecondName":"Sep","Addr":"La Lisa, Habana","DateBirth":"2020-10-16 00:00:00","PhoneNumbers":"5555555","Photo":"","**Age":3**,"UID":"BD2B7FD0-55D8-4987-B77E-145208F2D122","**ID":3**,"FirstName":"Lucia","SecondName":"Sep","Addr":"Camaguey, Camaguey","DateBirth":"2020-10-16 00:00:00","PhoneNumbers":"4444444","Photo":"","**Age":3**}]

http://localhost:8095/?Age>40

Resultado devuelto:

[{"UID":"{220CDA9E-E91E-492B-B15E-0BA17B00BB64}","ID":222,"FirstName":"Mama","SecondName":"Moya","Addr":"UCI,116 104","DateBirth":"","PhoneNumbers":"5678","Photo":"www.pictures.com\/yuri\/carnet.jpg","**Age":80**}]

Solicitando por varias condiciones

Ejemplos de solicitudes por 2 campos

http://localhost:8095/?ID=1&Age=3

Resultado devuelto:

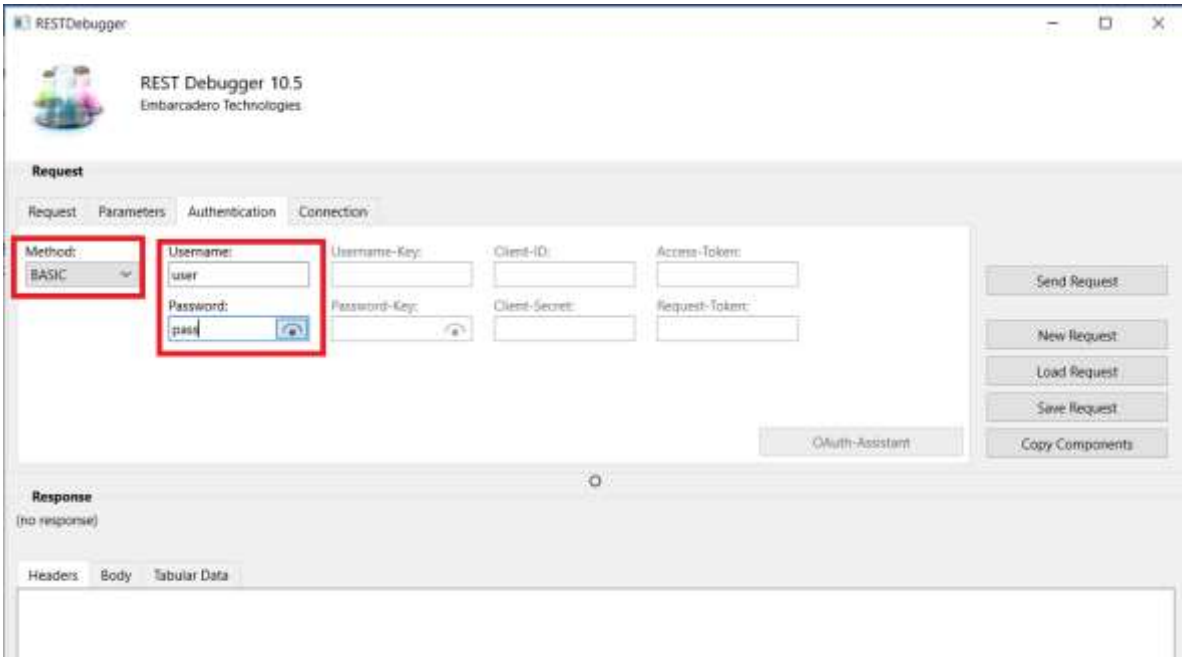
[{}]

http://localhost:8095/?ID=2&Age=3

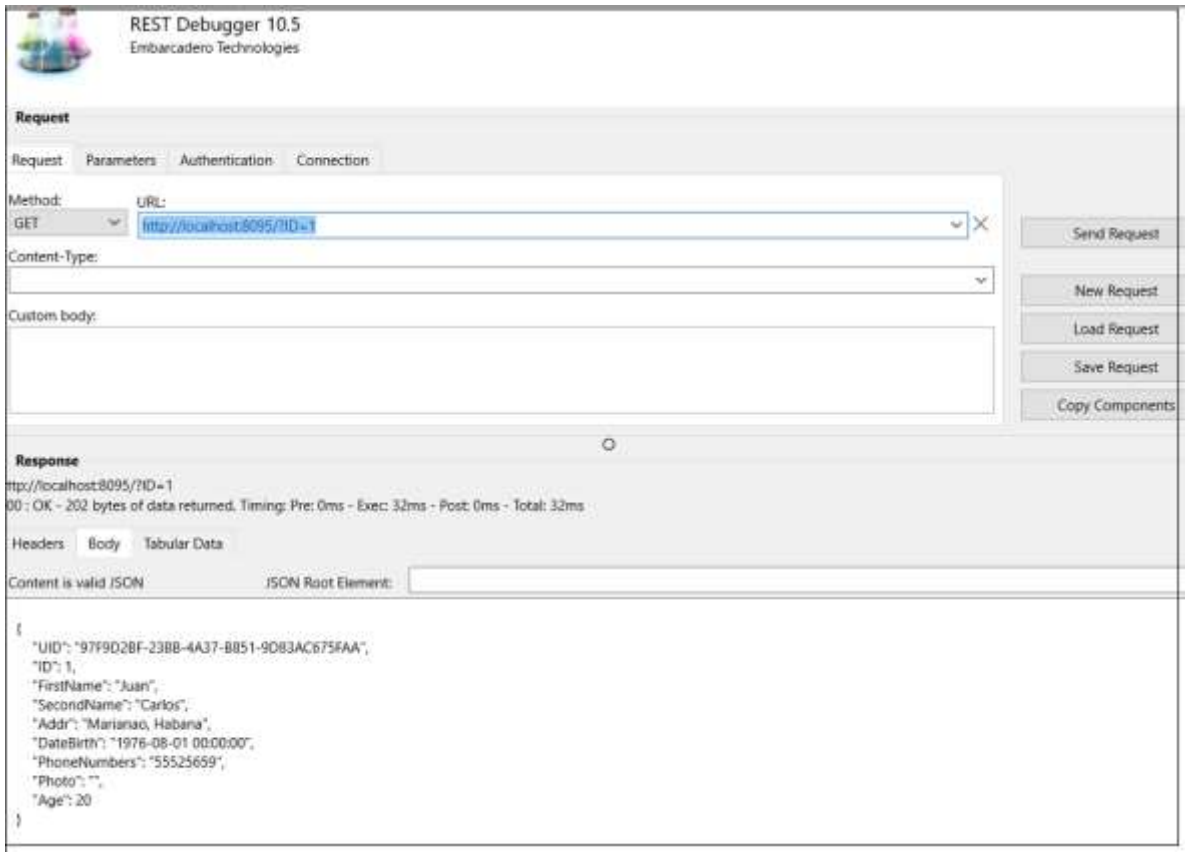
[{"UID":"4AC7B2C5-6AF7-49B0-B090-24F55A1C86F7","ID":2,"FirstName":"Lazaro","SecondName":"Sep","Addr":"La Lisa, Habana","DateBirth":"2020-10-16 00:00:00","PhoneNumbers":"5555555","Photo":"","Age":3}]

Usando REST Debugger 10.5

El primer paso para usar REST Debugger para probar la aplicación es configurarle la authenticacion para el sitio. En este caso de tipo “Basic” y con el login y password ya mencionados.



Para realizar solicitudes solo indicar las URL deseadas como ya se explicó.



Probando solicitudes del Tipo POST.

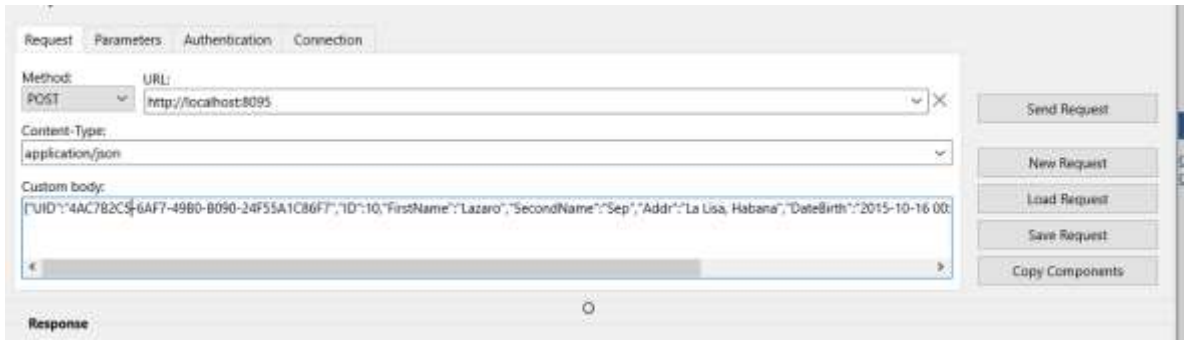
Se selecciono el puerto 8095 para realizar las pruebas de este manual. Pudo ser otro.

Usando REST Debugger 10.5

Para insertar un contacto debe conformar un texto JSON a enviar. Por ejemplo use este:

{"UID":"4AC7B2C5-6AF7-49B0-B090-24F55A1C86F7","ID":222,"FirstName":"Lazaro","SecondName":"Sep","Addr":"La Lisa, Habana","DateBirth":"2015-10-16 00:00:00","PhoneNumbers":"5555555","Photo":"","Age":3}

Seleccione método POST, y Content-Type = application/json



Si después realiza un chequeo de los actuales datos puede ver que se insertó un nuevo contacto el valor. El valor de UID que se especifico en el comando JSon es desechado ya que se genera automáticamente un valor de UUID internamente para garantizar que sea único.

Probando solicitudes del Tipo PUT.

Usando REST Debugger 10.5

Las actualizaciones se realizan a los contactos que tengan un ID igual al especificado como parte de la URL.

http://localhost:8095/15

Si existen contactos repetidos con el mismo ID todos se actualizaran con los datos enviados.

Para actualizar el valor previamente insertado, enviar por ejemplo el siguiente texto JSon con los valores a actualizar.

{"FirstName":"Haydee","SecondName":"Pena","Addr":"Marianao, Habana","DateBirth":"1976-08-01 00:00:00","PhoneNumbers":"55525659","Photo":"","Age":20}

En la cadena JSon no es necesario especificar ni el UID, ni el ID ya que ninguno de los dos será modificado

Probando solicitudes del Tipo DELETE.

Usando REST Debugger 10.5

Para eliminar algún contacto solamente tiene que indicar el valor del campo ID de los que se desean borrar.

Por ejemplo: http://localhost:8095/15

Se debe especificar que el Método es DELETE