INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

Desarrollo de aplicaciones web

Semestre agosto - diciembre 2017

Third-party API

Profesor: *Juan Velez Ballesteros*

Fecha de entrega: 9 de septiembre de 2017

Grupo: 01

A01332891 Oscar Emiliano Cervantes del Valle

A01064754 Alison Ricardo González Cortés

A01332278 Guillermo Barrientos González

Introducción

Un API es una serie de rutinas, protocolos y herramientas para construir un software, especificando cómo cada componente interactúa entre sí.
Los APIs permiten una mejor y más fácil construcción de un software, y proveen una mejor visualización de los datos ya que se encuentra todo como si fueran bloques.

Ejemplos de APIs

De los APIs más populares actualmente se encuentran Google con Google Maps y YouTube; Flickr, Twitter y Amazon Product Advertising.

Implementando APIs

Existen diferentes categorías en las cuales se pueden clasificar los APIs, esto es dependiendo de si será implementado para un sistema operativo, una aplicación o un sitio web.
Casi todos los ambientes operativos tienen sus propias APIs para permitirle al desarrollador crear nuevas aplicaciones que sean consistentes con dicho ambiente operativo.
Dentro de los sistemas web se pueden generar APIs para que se puedan utilizar esos servicios web y se dé un enfoque especializado a los usuarios finales.
Generalmente son terceras partes quienes desarrollan sus propios APIs para darle soluciones a los usuarios finales, esto puede ser porque no existe una solución específica para la necesidad de ciertos usuarios o porque sólo requieren una parte de las funciones del servicio web.

Servicios RESTful

Representational State Transfer (REST) es una arquitectura para crear sistemas distribuidos. Se trata de una serie de reglas, restricciones y generar una interfaz uniforme.
Posee ciertos principios que son:

- Recursos:

    - Estructura del directorio de fácil entendimiento a través de identificadores de recursos uniformes (URIs).
- Representaciones:
    - Manejar la información y atributos a través de JSONs o XML.
- Mensajes:
    - Usar los métodos de HTTP para el intercambio de información.
- Interacciones Stateless:
    - Interacciones que no almacenen información del usuario dentro del servidor entre peticiones. De otra forma se limita y restringe la escalabilidad.

Métodos GET, POST, PUT, PATCH, DELETE

A través de los métodos de HTTP se pueden tener funciones de crear, recibir, actualizar y borrar (CRUD).

- **GET**
    - Obtener la información
- **POST**
    - El recurso en la URI hará algo con la entidad proveída.
- **PUT**
    - Crea o actualiza una entidad.
- **PATCH**
    - Actualizar campos específicos de una URI.
- **DELETE**
    - Remueve un recurso.

API de Openbravo

Introducción

Los web services son del tipo REST. Openbravo ERP REST consiste en un framework que ofrece seguridad y servicios de excepciones, así como implementación de un data access REST web service. El data access (DAL) REST

web services provee un web service de tipo CRUD, de modo que las aplicaciones externas puedan recuperar, actualizar, crear y borrar objetos de negocio a través de peticiones HTTP estándar.

Uso y ejemplos de uso

Para utilizar un web service se debe hacer mediante una URL de acuerdo a la instancia de Openbravo. Por ejemplo:
http://url.instancia.openbravo/ws/dal/NombreWebService

Si se quiere ver en un formato diferente, utilizar:
http://url.instancia.openbravo/ws/dal/NombreWebService?template=bolist.xslt

Con la finalidad de probar los web services, se puede utilizar la extensión/plugin de Firefox / Chrome: Poster.

Openbravo ofrece dos formas para enviar las credenciales, usuario y contraseña:

- Ingresar con nombre de usuario / contraseña pasándolos como parámetros de la petición (los nombres de parámetros son respectivamente l y p).
- Autenticación básica HTTP

Respecto a lo anterior hay dos puntos a considerar:

- El inicio de sesión y el contexto son almacenados en una sesión por razones de rendimiento
- Si el cliente que hace la petición no soporta cookies entonces también es posible incluir información de acceso en cada solicitud. Esta es la menos preferida desde el punto de vista del rendimiento.

Se puede actualizar/crear un objeto (producto, cliente, factura, etc) mediante el uso de los comandos POST o PUT:

- El XML enviado deberá apegarse al Esquema XML (.xsd) que se defina en cada caso; si el resultado es satisfactorio se regresa una respuesta satisfactoria (<success/>), si falla se regresa una respuesta XML de error.

- Cuando se inserta (INSERT) un registro nuevo, NO es necesario incluir el elemento id. SI es necesario enviar todos los elementos con el atributo minOccurs="1" pues son los mínimos requeridos. El caso de los elementos client y organization es especial puesto que son requeridos, sin embargo pueden ser tomados automáticamente por Openbravo de acuerdo al usuario con el que se esté utilizando el web service.
- En el caso de actualizar (UPDATE) SI es necesario incluir el id pero NO es necesario enviar todos los elementos, basta con enviar el elemento que cambiará de valor. Los elementos client y organization NO deben incluirse al enviar un UPDATE. Ejemplo:

  ```
  <?xml version="1.0" encoding="UTF-8"?>
  <ob:Openbravo xmlns:ob="http://www.openbravo.com" xmlns:xsi="http://
  www.w3.org/2001/XMLSchema-instance">
  <Product id="FF80818133916435013937EB31E0004">
  <active>false</active> </Product>
  </ob:Openbravo>
  ```

Se puede eliminar uno o varios objetos del mismo tipo (producto, cliente, factura, etc) mediante el uso del comando DELETE:

- Con un URI apuntando al objeto que se va a eliminar:

  http://url.instancia.openbravo/ws/dal/NombreWebService/Id

- Con un URI que incluya el parámetro “where” para eliminar varios objetos del mismo tipo:

  http://url.instancia.openbravo/ws/dal/NombreWebService?where=value=‘CO2’

Los resultados que puede devolver Openbravo son los siguientes:

- Códigos HTTP:
  - 200 (OK): for successful requests

    - 400 (Bad Request): in case the uri could not be parsed or in case of invalid xml
    - 401 (Unauthorized): is returned when a security exception occurs
    - 404 (Not found): if the entityname does not exist or the if an individual business object is addressed, if the business object does not exist. Note that a 410 (Gone) response could also be applicable here but it is less known therefore the 404 is used.
    - 409 (Conflict): is returned for a POST or PUT action, specifies that the data which was posted is out-of-date or invalid.
    - 500 (Internal Server Error): if an unrecoverable application error occurred.

- Mensajes XML:
    - Consulta sobre uno o varios objetos: envía como respuesta el objeto en forma de XML.
    - Update/Insert: devolverá un mensaje de éxito XML (<success/>) si se realizó correctamente, el mensaje de vuelta contendrá el identificador del registro insertado.

- Mensajes de error:
    - Si la petición falla entonces un mensaje XML de error es devuelto.

API de Userlike

- Mensajes offline:
    - Obtener la lista de mensajes de los clientes mientras se estuvo fuera de línea. Esto se realiza a través de GET y DELETE.
    - Se genera un JSON Array que consiste de objetos para cada correo.
- Lista de chats:
    - Extraer la lista de todas las sesiones previas. Se realiza a través de los métodos GET y DELETE.

    - El resultado es un JSON Array que consiste de objetos para cada sesión de chat.
    - Se pueden poner filtros a cada query a través de parámetros GET.
- Lista de operadores:
    - A través de este acceso se pueden ver los miembros del equipo.
    - Se puede extraer la lista de todos los operadores que se tengan en una misma cuenta.
    - Los detalles de cada operador pueden ser manipulados en otras llamadas al API.

Cómo se utilizará nuestro API

Ya que la necesidad de negocio de nuestro cliente, se generará un API que obtenga la información de Userlike y Openbravo y tenga la estructura para manejar y mostrar de manera visual al usuario final tal como la requieren visualizar.

Esto es debido a que no toda la información que se recopila en la base de datos de Openbravo es relevante para el cliente y para los operadores de soporte técnico, de modo que se generará una estructura estandarizada para los diferentes tipos de usuarios del portal.

Userlike es una herramienta para darle soporte a los clientes mediante chats y que la misma permite obtener información acerca de la buena y lo mala atención al cliente que se le está dando. Openbravo es una plataforma ERP donde se almacenan los tickets que deben ser atendidos por parte de la empresa.

La aplicación permitirá a los clientes de Alabóol la creación, eliminación y modificación de tickets a través de portal esto se realizará mediante el API de Openbravo. El otro requerimientos funcional es la obtención estadística de datos por medio gráficas del servicio que se les brindan a los cliente. Esto último se obtendrá por medio del API de Userlike.

Con esta herramientas, se permitirá concentrar en un solo portal los servicios de soporte que se les da a los clientes de la empresa con la finalidad de que el cliente se familiarice un este y no de otros más.

Referencias

Anónimo (s.f.). *API Tutorial.* OpenBravo. Recuperado de https://code.openbravo.com/docs/

Anónimo (s.f.). *API Tutorial.* Userlike. Recuperado de https://www.userlike.com/en/public/tutorial/api/intro

Anónimo (2017). *Understanding REST.* Spring. Recuperado de https://spring.io/understanding/REST

Beal, V. (s.f.). *API - application program interface.* Webopedia. Recuperado de http://www.webopedia.com/TERM/A/API.html

MuleSoft (s.f.). *API Plataform.* MuleSoft. Recuperado de https://www.mulesoft.com/platform/api