Integrantes:

Juan José Lemus Vásquez - 201404412  
Oscar Francisco Oliva Monterroso - 201403651  
Herberth Guillermo Obregón Espino - 201314237  
Evely Fabiola Sandoval Gámez - 201314608

Documento de propuesta de topologia y servicios

Software Avanzado

Contenido

[Proyecto de Laboratorio 2](#_Toc5306914)

[Planteamiento de la Topología 2](#_Toc5306915)

[Definición de los Servicios 3](#_Toc5306916)

[Cada uno de los servicios web a utilizarse en cada una de las entidades del SOA: 3](#_Toc5306917)

[Definición de protocolos a utilizar en cada caso. 4](#_Toc5306918)

[Definición de cada una de las operaciones en cada caso, incluyendo. 5](#_Toc5306919)

[PIM 5](#_Toc5306920)

[Método getAllProducts: 5](#_Toc5306921)

[Método getSomeProducts: 5](#_Toc5306922)

[Método postProduct 6](#_Toc5306923)

[ERP 7](#_Toc5306924)

[Método getProducts: 8](#_Toc5306925)

[Método postConsultaDisponibilidad: 9](#_Toc5306926)

[Método postPedido: 9](#_Toc5306927)

[BODEGA 10](#_Toc5306928)

[Autenticación oAuth2: 11](#_Toc5306929)

[Roles de OAuth 12](#_Toc5306930)

# Proyecto de Laboratorio

## Planteamiento de la Topología

Considerando la naturaleza del caso de negocio propuesto, se considera una mezcla entre dos modelos de negocio los cuales se describen a continuación:

Texto resaltado, adquirido de la documentación SOA y servicios web de IBM, fuente:

<https://www.ibm.com/developerworks/library/ws-soa-esbtop/index.html>

**Múltiples Geografías:** Las empresas pueden organizarse geográficamente, al igual que los depósitos y tiendas se encuentran en distintas zonas geográficas, cada región opera de manera independiente y sinérgica en apoyo de los objetivos globales de la empresa, cada tienda puede administrar la gestión de sus productos de forma independiente a las demás tiendas, pero sigue orientada a los objetivos globales. Además, se dice que este tipo de modelo ofrece sus servicios a los clientes e interactúa con socios y proveedores dentro de un país o región específica, lo cual va en contra de las restricciones del caso ya que, según disposiciones contractuales, el inventario que venden en línea de negocio compartida solo puede ser provisto por el consorcio así que se ve necesario fusionar otro modelo de negocio.

**Red de tiendas / sucursales:** Las empresas con redes de sucursales a menudo implementan funciones de integración, procesos y capacidades bajo el control central o regional, debido a que cada tienda debe comunicarse específicamente a través de funciones descritas por el consorcio, Una infraestructura central a la que todas las sucursales están vinculadas, y desde la cual se administran todas las sucursales, es el rasgo clave, las tiendas a través de su ERP estarán conectadas con el PIM, el cual es el que brindará un catálogo global, el cuál será el mismo para todas las tiendas.

# Definición de los Servicios

# Cada uno de los servicios web a utilizarse en cada una de las entidades del SOA:

# Definición de protocolos a utilizar en cada caso.

## Restful

REST es una tecnología mucho más flexible que transporta datos por medio del protocolo HTTP, pero este permite utilizar los diversos métodos que proporciona HTTP para comunicarse, como lo son GET, POST, PUT, DELETE, PATCH y a la vez, utiliza los códigos de respuesta nativos de HTTP (404,200,204,409). REST es tan flexible que permite transmitir prácticamente cualquier tipo de datos, ya que el tipo de datos está definido por el Header Content-Type, lo que nos permite mandar, XML, JSON, Binarios (imágenes, documentos), Text, etc.

## ¿Porque Restful ?

1. Si lo medimos por el número de servicios Web que lo utilizan, en los últimos años REST ha emergido como un modelo de diseño predominante para los servicios Web. De hecho, REST ha tenido tanto impacto en la Web que ha desplazado enormemente al diseño de interfaces basado en SOAP y WSDL, porque es un estilo considerablemente más fácil de utilizar.
2. SOAP solo permite la transmisión de datos en formato XML, mientras que con Rest podemos utilizar JSON, Otra de las grandes ventajas que presenta JSON sobre SOAP es el performance, ya que los JSON son considerablemente más livianos en peso y mucho más rápido en su procesamiento.

## Ventajas que ofrece REST para el desarrollo

##### **Separación entre el cliente y el servidor**:

El protocolo REST separa totalmente la interfaz de usuario del servidor y el almacenamiento de datos. Eso tiene algunas ventajas cuando se hacen desarrollos. Por ejemplo, mejora la portabilidad de la interfaz a otro tipo de plataformas, aumenta la escalabilidad de los proyectos y permite que los distintos componentes de los desarrollos se puedan evolucionar de forma independiente.

##### **Visibilidad, fiabilidad y escalabilidad**.

La separación entre cliente y servidor tiene una ventaja evidente y es que cualquier equipo de desarrollo puede escalar el producto sin excesivos problemas. Se puede migrar a otros servidores o realizar todo tipo de cambios en la base de datos, siempre y cuando los datos de cada una de las peticiones se envíen de forma correcta. Esta separación facilita tener en servidores distintos el front y el back y eso convierte a las aplicaciones en productos más flexibles a la hora de trabajar.

##### **La API REST siempre es independiente del tipo de plataformas o lenguajes**:

la API REST siempre se adapta al tipo de sintaxis o plataformas con las que se estén trabajando, lo que ofrece una gran libertad a la hora de cambiar o probar nuevos entornos dentro del desarrollo

# Definición de cada una de las operaciones en cada caso, incluyendo.

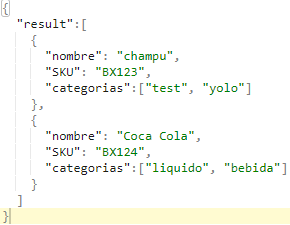
## PIM

El servicio de PIM será desarrollado en Node exponiendo servicios restful donde los mensajes que se enviarán y recibirán son json con una estructura definida, teniendo como punto de entrada <http://dominio/PIM_WS/metodo> teniendo los siguientes métodos disponibles:

## Método getAllProducts:

El tipo de método es “GET”, no recibe ningún parámetro, como resultado devolverá un json con un atributo llamado “result” que contendrá un arreglo de json y cada json del arreglo tendrá los siguiente atributos nombre de tipo texto, SKU de tipo texto y categorías será un arreglo que contendrá cada el nombre de cada categoría.

Ejemplo del json de respuesta:



**Versión reducida**

|  |  |
| --- | --- |
|  | Resultado |
| Nombre de Función | getAllProducts |
| Explicación | Devuelve un arreglo de productos con la información mínima. |
| Parámetros de Entrada | NA |
| Repuesta | String: JSON |

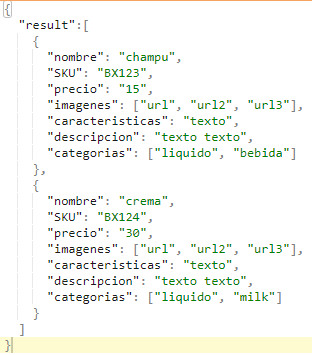
## Método getSomeProducts:

El tipo de método es “POST”, el método recibe como parámetro un json con los siguiente atributo “producto” que es un arreglo de SKU de los cuales se requiere todo la información del producto, como respuesta el método devolverá un json con un atributo llamado “result” que contendrá un arreglo con jsons y cada json del arreglo tendrá los siguientes atributos nombre de tipo texto, SKU de tipo texto, precio de tipo numérico, imágenes de tipo arreglo que contendrán las direcciones de las imágenes, características de tipo texto, descripción de tipo texto, categóricas un arreglo que contendrá el nombre de cada categoría.

Ejemplo de json de parámetro:



Ejemplo de json de respuesta:

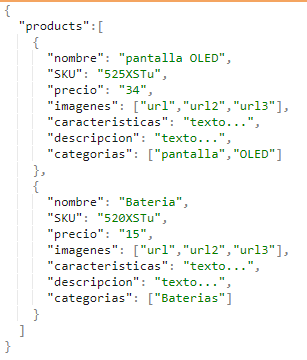


**Versión reducida**

|  |  |
| --- | --- |
|  | Resultado |
| Nombre de Función | getSomeProducts |
| Explicación | Devuelve un arreglo de productos solicitados por la tienda, contiene la información completa. |
| Parámetros de Entrada | String: JSON |
| Repuesta | String: JSON |

## Método postProduct

Este método es tipo “POST”, este método recibe como parámetro “producto” que es un arreglo de productos en formato JSON, que contendrá el nombre, SKU, precio, imagen, características, descripción y categoría; este método será utilizado para agregar productos al catálogo.



Respuesta



**Versión reducida**

|  |  |
| --- | --- |
|  | Resultado |
| Nombre de Función | postProduct |
| Explicación | Envía un arreglo de productos que serán agregados al PIM |
| Parámetros de Entrada | String: JSON |
| Repuesta | Stirng: JSON |

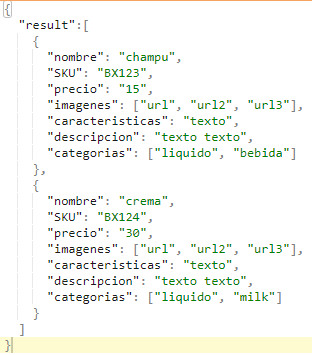
## ERP

El servicio de ERP será desarrollado en Node exponiendo servicios restful donde los mensajes que se enviarán y recibirán son json con una estructura definida, teniendo como punto de entrada <http://dominio/ERP_WS/metodo> teniendo los siguientes métodos disponibles:

## Método getProducts:

Método utilizado para abastecer la tienda con todos los productos que se encuentran en el inventario local unificado con los productos del PIM.

El tipo de método es “GET”, no recibe ningún parámetro, como resultado devolverá un json con un atributo llamado “result” que contendrá un arreglo tipo JSON con todos los productos.



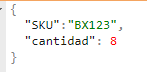
**versión reducida**

|  |  |
| --- | --- |
|  | Resultado |
| Nombre de Función | getProducts |
| Explicación | utilizado para abastecer la tienda con todos los productos que se encuentran en el inventario local unificado |
| Parámetros de Entrada | NA |
| Repuesta | String: JSON |

## Método postConsultaDisponibilidad:

Método utilizado para verificar la existencia de pedidos en la cantidad solicitada.

El tipo de método es “POST”, se le envía como parámetro un texto JSON que contendrá el producto a verificar, como resultado devolverá un booleno True en caso exista y False en caso no haya disponibilidad.



Respuesta



**versión reducida**

|  |  |
| --- | --- |
|  | Resultado |
| Nombre de Función | postConsultaDisponibilidad |
| Explicación | Método utilizado para verificar la existencia de pedidos en la cantidad solicitada. |
| Parámetros de Entrada | String: JSON |
| Repuesta | String: JSON |

## Método postPedido:

Método utilizado para realizar un pedido, en la tienda se pueden escoger uno o varios productos los cuales se desea adquirir.

El tipo de método es “POST”, se le envía como parámetro un texto JSON que contendrá un arreglo de los productos solicitados, como resultado devolverá un json con una respuesta exitosa o fallida OK/NOK.



Respuesta



**versión reducida**

|  |  |
| --- | --- |
|  | Resultado |
| Nombre de Función | postPedido |
| Explicación | Método utilizado para realizar un pedido, a través de un arreglo de productos. |
| Parámetros de Entrada | String: JSON |
| Repuesta | String: JSON |

## BODEGA

El servicio de Bodega será desarrollado en Node exponiendo servicios restful donde los mensajes que se enviarán y recibirán son json con una estructura definida, teniendo como punto de entrada <http://dominio/BODEGA_WS/metodo> teniendo los siguientes métodos disponibles:

# Autenticación oAuth2:

## Roles de OAuth

OAuth define cuatro roles:

1. Propietario del recurso
2. Cliente
3. Servidor de recursos
4. Servidor de autorización

Detallaremos cada rol en las siguientes subdivisiones.

### Propietario del recurso: Usuario

El propietario del recurso es el "usuario" que da la autorización a una aplicación, para acceder a su cuenta. El acceso de la aplicación a la cuenta del usuario se limita al "alcance" de la autorización otorgada (e.g. acceso de lectura o escritura).

### Servidor de Recursos / Autorización: API

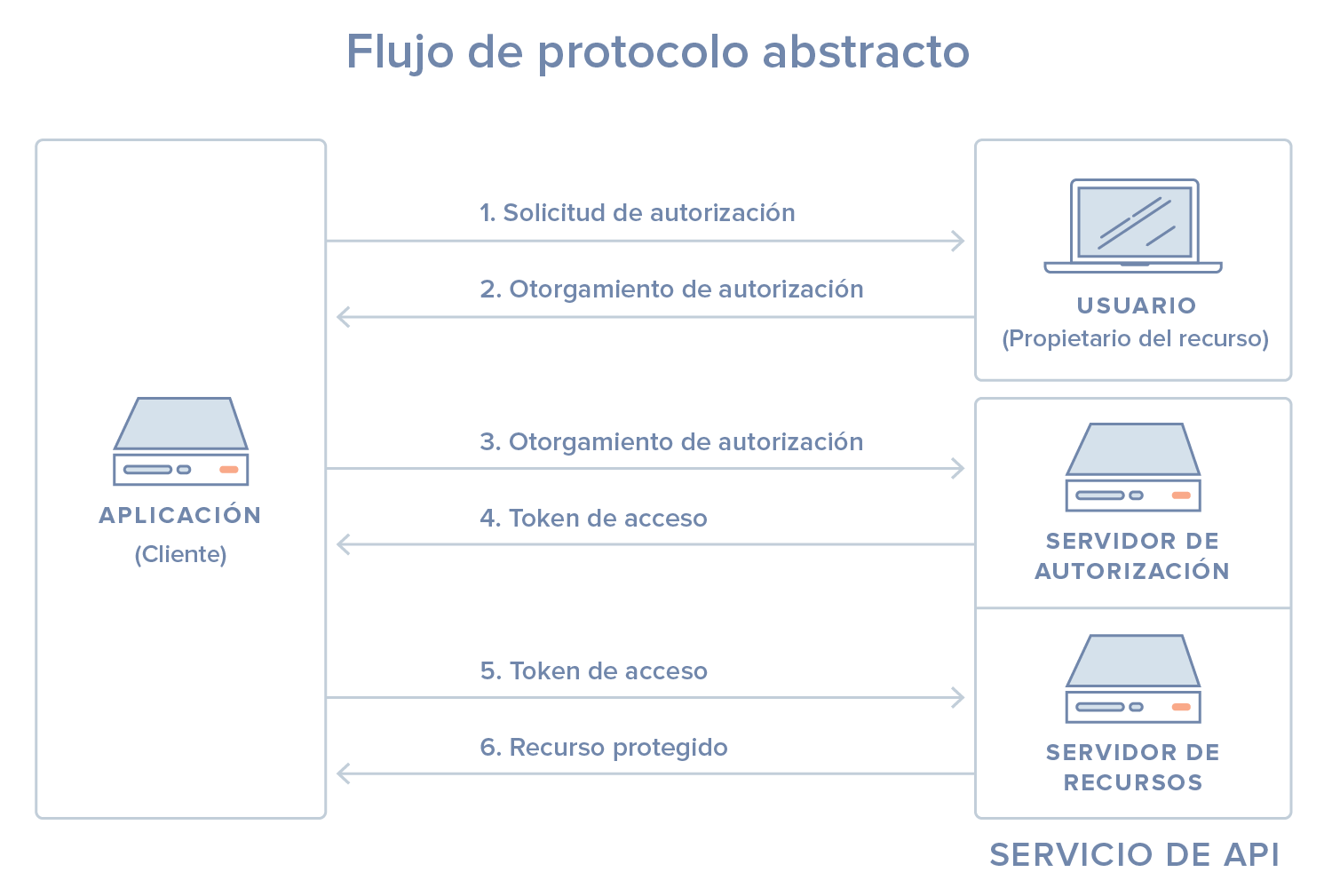
El servidor de recursos aloja las cuentas de usuario protegidas, y el servidor de autorizaciones verifica la identidad del usuario y luego genera tokens de acceso a la aplicación.

Desde el punto de vista del desarrollador de una aplicación, la API del servicio atiende tanto a los roles de recursos como a los de autorización. Nos referiremos a ambos roles combinados, como al rol de servicio o de API.

### Cliente: Aplicación

El cliente es la aplicación que desea acceder a la cuenta del usuario. Antes de que pueda hacerlo, debe ser autorizado por el usuario, y dicha autorización debe ser validada por la API.

## Flujo de protocolo abstracto



A continuación se encuentra una explicación más detallada de los pasos en el diagrama:

1. La aplicación solicita autorización para acceder a los recursos de servicio del usuario
2. Si el usuario autoriza la solicitud, la aplicación recibe la autorización
3. La aplicación solicita al servidor de autorización (API), presentando la autenticación de su identidad y la autorización otorgada La aplicación solicita al servidor de autorización (API) un token de acceso presentando la autenticación de su propia identidad y la autorización otorgada
4. Si la identidad de la aplicación es autenticada y la autorización es válida, el servidor de autorización (API) emite un token de acceso a la aplicación. La autorización finaliza
5. La aplicación solicita el recurso al servidor de recursos (API) y presenta el token de acceso para autenticarse
6. Si el token de acceso es válido, el servidor de recursos (API) provee el recurso a la aplicación

El flujo real de este proceso variará dependiendo del tipo autorización que esté en uso, sin embargo, esta es la idea general. Examinaremos diferentes tipos de autorizaciones en una sección posterior.

### Registro de la aplicación

Antes de utilizar OAuth, se deben registrar las tiendas que podrán hacer uso de este servicio en la cual se solicitaran los siguientes datos

* Nombre de la aplicación
* Sitio web de la aplicación
* Redirect URI o Callback URL

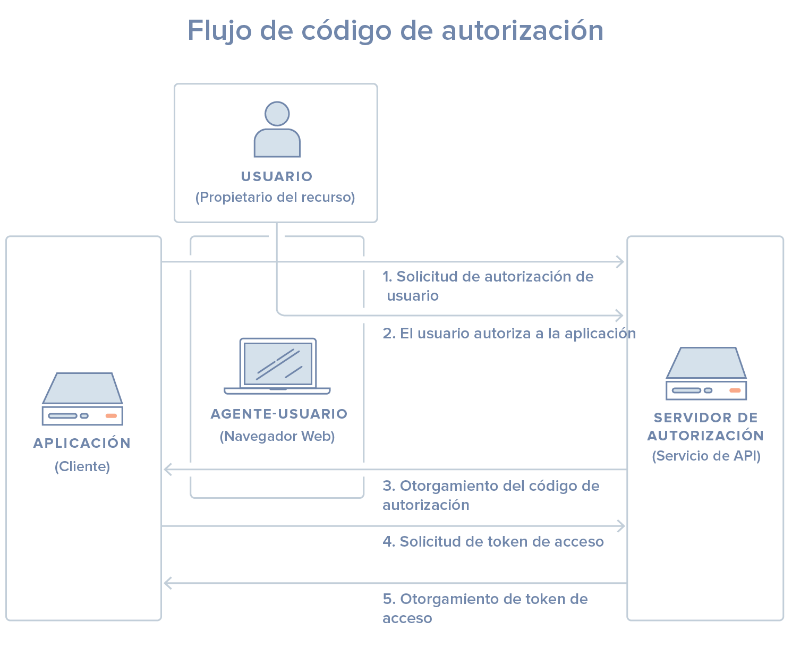
*Redirect URI* es donde el servicio reorientará al usuario después de que se autorice (o deniegue) su solicitud y, por consiguiente, la parte de su aplicación que manejará códigos de autorización o tokens de acceso.

### Identificador del cliente y secreto de cliente

Una vez esté registrada tu aplicación, el servicio emitirá "credenciales del cliente" en forma de un identificador de cliente y un secreto de cliente. El identificador (ID) de cliente es una cadena pública que utiliza la API de servicio para identificar la aplicación y para generar las URL de autorización que se presentan a los usuarios. Una vez la aplicación solicita el acceso a la cuenta de un usuario, el secreto de cliente se utiliza para autenticar la identidad de la aplicación al API de servicio; y se deberá mantener la confidencialidad entre la aplicación y la API.

## Código de autorización

El tipo de otorgamiento más usado es el código de autorización, ya que ha sido optimizado para aplicaciones del lado del servidor, en donde el código fuente no está expuesto públicamente y se puede mantener la confidencialidad del secreto de cliente. Este es un flujo basado en la reorientación (redirection), que significa que la aplicación debe ser capaz de interactuar con el agente de usuario (i.e. el navegador web del usuario) y recibir códigos de autorización API que se enrutan a través del agente de usuario.



### Paso 1: Enlace de código de autorización

Primero se le da al usuario un enlace de código de autorización similar al siguiente:

https://api.commerce.com/v1/oauth/authorize?response\_type=code&client\_id=CLIENT\_ID&redirect\_uri=CALLBACK\_URL&scope=read

A continuación, se presenta una explicación de los componentes del enlace:

* **https://api.commerce.com/v1/oauth/authorize**: El punto de conexión de la API de autorización
* **client\_id=client\_id**: el ID de cliente (cómo la API identifica la aplicación)
* **redirect\_uri=CALLBACK\_URL**: donde el servicio reorienta al agente-usuario después de que se otorgue un código de autorización
* **response\_type=code**: especifica que tu aplicación está solicitando un código de autorización
* **scope=read**: especifica el nivel de acceso que la aplicación está solicitando

### Paso 2: El usuario autoriza a la aplicación

Cuando el usuario hace clic en el enlace, debe primero iniciar sesión en el servicio para autenticar su identidad (a menos que ya haya iniciado sesión). Luego, el servicio solicitará autorizar o denegar el acceso de la aplicación a su cuenta.

### Paso 3: La aplicación recibe el código de autorización

Si el usuario hace clic en "Authorize Application", el servicio reorienta el agente-usuario al "redirect URI" de la aplicación, que se especificó durante el registro del cliente, junto con un código de autorización. La reorientación sería algo así (suponiendo que la aplicación es "usac.edu.gt"):

https://usac.edu.gt /callback?code=AUTHORIZATION\_CODE

### Paso 4: La aplicación solicita token de acceso

La aplicación solicita un token de acceso de la API, pasándole el código de autorización junto con los detalles de autenticación, incluido el secreto del cliente, a la terminal del token de la API. A continuación, se presenta un ejemplo de una solicitud POST para la conexión del token:

https://api.commerce.com/v1/oauth/token?client\_id=CLIENT\_ID&client\_secret=CLIENT\_SECRET&grant\_type=authorization\_code&code=AUTHORIZATION\_CODE&redirect\_uri=CALLBACK\_URL

### Paso 5: La aplicación recibe el token de acceso

Si la autorización es válida, la API enviará una respuesta a la aplicación, con el token de acceso (y, opcionalmente, un token de actualización). La respuesta completa se verá más o menos así:

{"access\_token":"ACCESS\_TOKEN","token\_type":"bearer","expires\_in":2592000,"refresh\_token":"REFRESH\_TOKEN","scope":"read","uid":100101,"info":{"name":"Mark E. Mark","email":"mark@thefunkybunch.com"}}