METODOLOGIA

Estándar para servicios API Rest

Versión 1.0

VERSIÓN PLANTILLA 1

1. CONSIDERACIONES

- Los servicios deben manejar estándares web donde que tengan sentido
- Los servicios deben ser amigable para el desarrollador y ser explorable mediante una barra de direcciones del navegador
- Los servicios deben ser sencillos, intuitivos y consistentes para que su adopción no sólo sea fácil, sino también agradable.
- Los servicios deberían proporcionar suficiente flexibilidad y ser eficientes.

2. HERRAMIENTAS DE IMPLEMENTACION

- NodeJs versión 12
- Express JS
- DockerFile

3. VERBOS OBJETOS (CRUD)

- **GET:** Consulta
- **POST:** Insertar
- **PUT:** Actualizar
- **DELETE:** Eliminar

RECURSO	POST / CREATE	GET / READ	PUT / UPDATE	DELETE
/productos	Crea un nuevo producto	Devuelve la lista de productos	Actualización en bloque de productos	Borrar todos los productos
/productos/711	Método no permitido (405)	Devuelve un producto específico	Actualiza un producto específico	Borra un producto específico

4. CARACTERISTICAS:

- Usar sustantivos plurales No verbos para declarar los endpoint. Ej: clientes, productos
- Incluir versionados a nivel de estructura en caso de ser necesario
  Ej: cliente\v1\
- Las relaciones se manejan a través del Path.
  Ej: clientes\{rut}\productos\{producto}
- Implementar Manejo de Errores el cual debe incluir (códigos de estados, mensaje para el desarrollador, mensaje para el usuario final, código de error interno, enlaces con más información)

Error único:

```
{
 coderror : 1234,
 message : Mensaje de error para el desarrollador,
 description : Descripción del Error para el usuario final,
 link: enlace a documentación u otro destino
}
```

Error en cadena:

```
{
 coderror : 1234,
 message : Mensaje de error para el desarrollador,
 link: enlace a documentación u otro destino,
 errors: [
          {
             coderror:1111,
             field: Campo 1,
             description Descripción del Error para el usuario final
          },
          {
             coderror:1112,
             field: Campo 2,
             description Descripción del Error para el usuario final

          }
        ]
   }
```

Sin Error: (opcional)

```
{
 coderror : 204,
 message : todo bien pero no se ha devuelto ningún contenido
}
```

- Usar estado en caso de ser necesario
- Usar Paginaciones (Registro Moderados + Información Paginación)
- Estandarización de las respuestas.

```
{
   "status":"OK",
   "error":{},
   "data":{}
}
```

- Aplicar Respuestas Parciales en caso de ser necesario, límite superior e inferior
- Usar solo formato JSON (application/json)
- Debe aplicar Formato UTF-8 (Content-Type: application/json; charset=utf-8)
- Funcionalidad tratarlas como un campo del recurso en la API.
  Ej.: clientes?ordenar=nombre
- Especificar campos opcionales como una lista separada por coma.
  Ej.: clientes?ordenar=ejecutivo,nombre
- Aplicar Autenticación (Básica, Api Keys, Oauth)
- Idioma Ingles
- Usar formato de Fecha según ISO 8601, en UTC. (yyyy-mm-dd)?
- Deben hacer uso del parámetro “mock” para datos de prueba, para simular dumis en etapa de desarrollo
- Soporte CORS.

5. REQUERIMIENTO DE ENTREGA

- Documentación de Instalación, de uso y desarrollo (Readme)
- Archivo DockerFile
- Fuentes del Servicios
- Pruebas Unitarias

6. ERRORES:

Los códigos de respuestas más habituales siguen la siguiente estructura:

- 2xx Indican que la petición se ha satisfecho correctamente
    - 200 todo fue bien y se ha encontrado el recurso y procesado la respuesta
    - 201 todo OK se ha creado el nuevo recurso
    - 204 todo bien pero no se ha devuelto ningún contenido
- 3xx indican que el recurso solicitado está en otra URI diferente
    - 301 el recurso se ha cambiado de ubicación definitivamente y nos indica la nueva ubicación
    - 302 el recurso que solicitas se ha movido temporalmente a este otro recurso
- 4xx indican códigos de error en el lado del cliente
    - 400 nos indica que hay un error en el formato o que es demasiado largo
    - 401 para decirnos que no tenemos autorización para ese recurso
    - 403, para indicarnos que el acceso al recurso no está permitido y el servidor ha rechazado nuestra petición.
    - 405 que nos indica que el método no se admite en la API.
- 5xx indican códigos de error en el lado del servidor
    - 500, cuando ha habido algún error interno del servidor
    - 502, cuando el servidor hace de Gateway o proxy y ha recibido una respuesta de error a donde quiera que intentase conectarse
    - 503 cuando el servicio en el servidor no está accesible, normalmente debido a una sobrecarga o algún problema similar.