

API Legacy

En el contexto de la migración del proyecto SICOF, se ha identificado la necesidad de desarrollar una API Legacy que permita la comunicación entre las aplicaciones legacy de la empresa (SICOF, BPP y Rentas) con los nuevos servicios y microfrontend. Esta API actuará como un puente entre el pasado y el futuro, facilitando la integración de datos y procesos entre los diferentes entornos tecnológicos.

Objetivos

Los objetivos principales del desarrollo de la API Legacy son:

- **Facilitar la migración de datos:** La API proporcionará un mecanismo estandarizado para la transferencia de datos de sesión entre las aplicaciones legacy y los nuevos sistemas microfrontend, minimizando la necesidad de reescritura de código y simplificando el proceso de migración.
- **Exponer funcionalidades legacy:** La API permitirá exponer las funcionalidades clave de las aplicaciones legacy a los nuevos sistemas, permitiendo su reutilización y evitando la necesidad de replicarlas desde cero.
- **Promover la integración:** La API servirá como base para la integración de las aplicaciones legacy con los nuevos sistemas, permitiendo una comunicación fluida y el intercambio de información en tiempo real.

Arquitectura

La API Legacy se basará en una arquitectura RESTful, siguiendo los principios de diseño de APIs REST. Esto permitirá una fácil integración con las aplicaciones legacy y los nuevos sistemas, utilizando protocolos HTTP estándar y formatos de datos como JSON.

Componentes

La API Legacy estará compuesta por los siguientes componentes:

Capa de acceso a datos

Esta capa encapsulará el acceso a las bases de datos de las aplicaciones legacy, proporcionando una interfaz única y simplificada para la consulta y manipulación de datos.

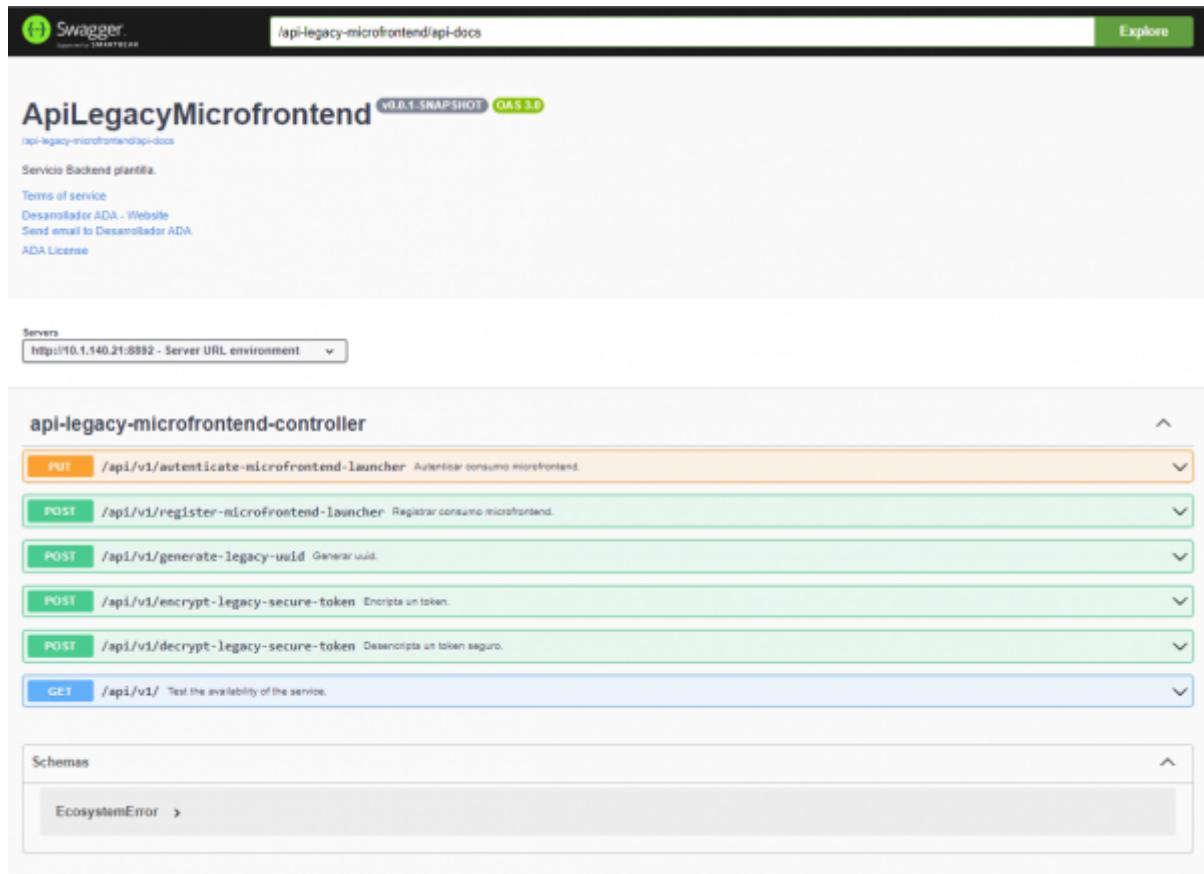
Se define el siguiente modelo.

```
/**  
 * The Class FrontendConsumingUrlDto.  
 */  
@Schema(description = "Entidad que almacena los registros de tokens")
```

```
heredados para consumo microfronend.")  
@Data  
public class FrontendConsumingUrlDto implements Serializable{  
  
    /** The Constant serialVersionUID. */  
    private static final long serialVersionUID = 1L;  
  
    /** The id. */  
    @Schema(description = "Identificador del registro.")  
    private Long id;  
  
    /** The microfrontend url. */  
    @Schema(description = "url del microfronend desplegado.")  
    private String microfrontendUrl;  
  
    /** The path param. */  
    @Schema(description = "Parámetro que contiene la cadena de la sesión  
    encriptada que será utilizada en la inicialización del microfronend.")  
    private String pathParam;  
  
    /** The consume status. */  
    @Schema(description = "Estado del registro. Es utilizado para validar si  
    el token heredado está vigente.")  
    private String consumeStatus;  
  
    /** The uuid. */  
    @Schema(description = "Identificador del token heredado.")  
    private String uuid;  
}
```

Capa de lógica de negocio

Esta capa implementará la lógica de negocio específica de las aplicaciones legacy, exponiendo las funcionalidades de estas aplicaciones a través de la API.



api-legacy-microfrontend-controller

- PUT** /api/v1/autenticate-microfrontend-launcher Autenticar consumo microfrontend.
- POST** /api/v1/register-microfrontend-launcher Registrar consumo microfrontend.
- POST** /api/v1/generate-legacy-uuid Generar uuid.
- POST** /api/v1/encrypt-legacy-secure-token Encripta un token.
- POST** /api/v1/decrypt-legacy-secure-token Desencripta un token seguro.
- GET** /api/v1/ Test the availability of the service.

Schemas

- EcosystemError

La documentación del api se expone en los siguientes endpoint.

- [Documentación OpenAPI](#): JSON de documentación del servicio.
- [Documentación Swagger](#): Visor Swagger de documentación del servicio.

Seguridad

La seguridad de la API Legacy será una prioridad absoluta. Se implementarán medidas de seguridad como autenticación, autorización, cifrado y validación de datos para proteger la información confidencial y garantizar la integridad de las transacciones.

Integración del API Legacy: Escenarios y Tipos

El API Legacy se integrará con las aplicaciones legacy y los nuevos sistemas de diversas maneras, abriendo un abanico de posibilidades para la migración y el desarrollo futuro. A continuación, se describen algunos de los escenarios de integración más comunes:

Integración Microfrontend

El API Legacy se debe utilizar para implementar una arquitectura de microfrontend, donde cada aplicación frontend se integra con el API para autorizar el acceso a las funcionalidades que se migran. Esta estrategia facilita el desarrollo y mantenimiento de las aplicaciones frontend, ya que cada una puede ser desarrollada y actualizada de forma independiente.

El flujo para autorizar los accesos desde aplicaciones legacy es:

Paso 1

Las aplicaciones legacy consumiran la url del microfrontend y enviarán un parámetro llamado legacyToken de tipo string.

Paso 2

El parámetro legacyToken es un string con el siguiente formato **contextClient@uuid** el cual al ser separados por el simbolo @ se obtendran 2 fragmentos.

- **contextClient**: Código del cliente.
- **uuid**: token de autenticación.
- **separador**: Caracter de separación de los fragmentos del token legacy. Por defecto se define @

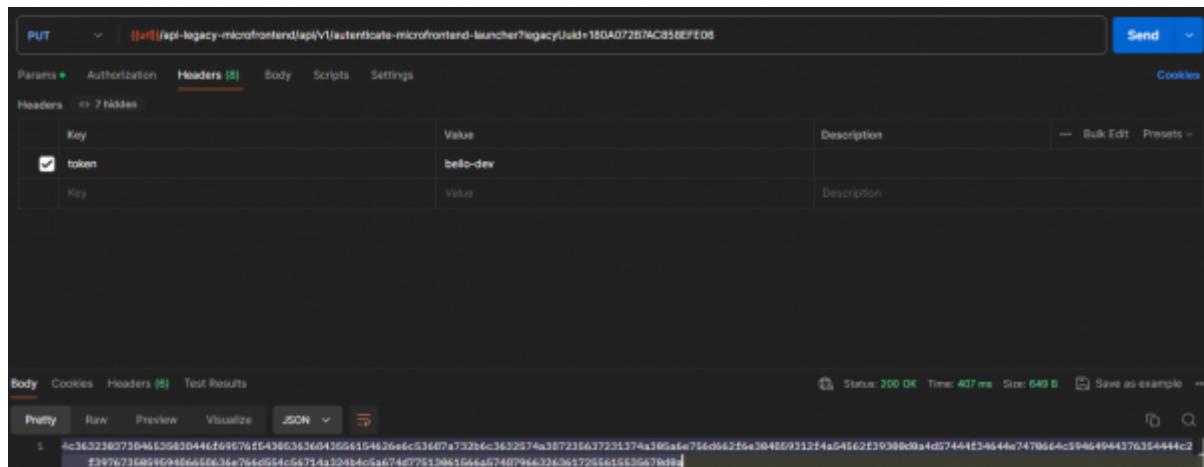
Ejemplo: legacyToken=bello-dev@180A072B7AC858EFE06

- **contextClient**: bello-dev
- **uuid**: 180A072B7AC858EFE06
- **separador**: @

Paso 3

Se debe consumir el método del api **autenticate-microfrontend-launcher** el cual puede ser consultado en la documentación de la [Capa de lógica de negocio](#) y pasar el contextClient como token en el header y el uuid como parametro.

Si el uuid está vigente se devolverán los parametros de sesión encriptados con el código 200. Lo que permitirá el acceso a la funcionalidad.



POST {{url}}/api-legacy-microfrontend/api/v1/autenticate-microfrontend-launcher?legacyUuid=180A072B7AC858EFE06

Headers

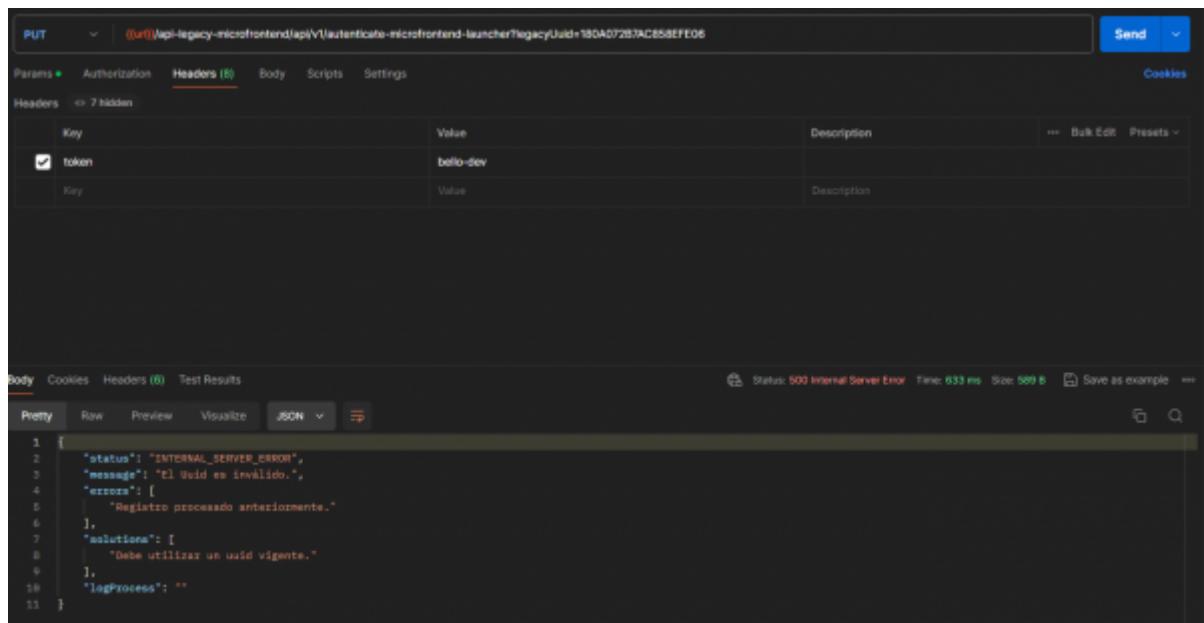
Key	Value	Description
token	bello-dev	

Body

```
1 4c36323073b462983d446d69676f543803636043266154626e6c53407a732b6c3632574a387236637231374a385a6e756d622f6e304089312f4a54562e393000c8a4d57444f34644a7470664c59464944376314444c2  
139767316958948645364766d654c56714a32444c5a674d77513961664c67407944336361725f6155367949
```

Si el uuid está vencido o es inválido se devolverá una excepcion con el código 500. Lo que evitara el

acceso al microfrontend.



The screenshot shows a POST request to the URL `((url))api-legacy-microfrontend/api/V1/authenticate-microfrontend-launcher?legacyUuid=10407287AC858EFE06`. The 'Headers' tab is selected, showing a single header `token` with the value `bello-dev`. The 'Body' tab is selected, showing an empty JSON object. The 'Test Results' tab shows a response with status 500 Internal Server Error, time 633 ms, and size 569 B. The response body is a JSON object:

```
1: {  
2:   "status": "INTERNAL_SERVER_ERROR",  
3:   "message": "ElUuid es invalido.",  
4:   "errors": [  
5:     "Registro procesado anteriormente."  
6:   ],  
7:   "solutions": [  
8:     "Debe utilizar un uuid vigente."  
9:   ],  
10:  "logProcess": ""  
11: }
```

Paso 4

Al realizar la autenticación de forma correcta se obtendrá un hash que contendrá los parámetros de sesión pero estarán encriptados por lo tanto se debe consumir el servicio **decrypt-legacy-secure-token** el cual recibirá 2 argumentos uno en el header llamado token el cual recibirá el contextClient y un parámetro secureToken donde se le enviará el hash obtenido del proceso de autenticación como se ve en la imagen.

[← Regresar](#)

From:
<http://wiki.adacsc.co/> - Wiki



Permanent link:
<http://wiki.adacsc.co/doku.php?id=ada:howto:sicoerp:factory:new-migracion-sicoerp:apilegacy&rev=1717719741>

Last update: 2024/06/07 00:22