Contenido

Enunciado del Ejercicio de APIs para un Partner Engineer 1
Objetivo 1
Diseño de las APIs REST 1
Documentación de la **API A** y endpoints 1
Documentación de la **API B** y endpoints 2
Desarrollo del Informe de Integración de APIs 4
Descripción General 4
Flujo del Proceso de consulta de la API B 4
Diagrama de secuencia del consumo de las APIs en la integración 5
Configuración de las APIs en Postman 8
Capturas de pantalla Postman 8
Codigo export_postman.json: 9
Documentación de las APIs en Swagger 11
Capturas de pantalla Swagger 11
Codigo swagger_apis_partner_engineer.yml 12
Integración de las APIs 14
Codigo integracion.py: 14
Explicación del Script Python: 16

Enunciado del Ejercicio de APIs para un Partner Engineer

Objetivo

El objetivo es integrar 2 APIs. La API A es la API de la cual obtendremos los datos, y la API B es la que queremos alimentar. Los services managers y los desarrolladores deben tener toda la información necesaria para que su API B consiga devolver la información que se muestra en la respuesta API B.

Diseño de las APIs REST

Documentación de la **API A** y endpoints

URL o Endpoint: https://api.sistemaA.com/facturas

Método: GET

Partner Engineer: Eleana Rey Quijada
https://www.linkedin.com/in/eleanarey/

Descripción: API A es un servicio REST que permite la consulta de facturas emitidas en un rango de fechas. A través de la autenticación mediante OAuth 2.0, los usuarios pueden obtener una lista detallada de facturas, incluyendo el cliente, monto, fecha de emisión y estado de la factura.

Parámetros de consulta:

fecha_inicio: Fecha de inicio para el rango de consulta (Formato: YYYY-MM-DD).

fecha_fin: Fecha de fin para el rango de consulta (Formato: YYYY-MM-DD).

Seguridad: Autenticación mediante Token en el encabezado Authorization: Bearer <token> obtenido a través de client credentials (OAuth 2.0).

Manejo de errores y excepciones: Si los parámetros fecha_inicio y fecha_fin no son válidos, la API devolverá un código de error **400** (Bad Request). En general, en caso de un error interno de la parte cliente, se retornará un código **4XX** y en caso de un error de servidor, se retornará un código **5XX**. Si la petición se procesa correctamente la salida será de tipo **2XX**.

Encabezados de solicitud: Authorization: Bearer <token> (generado automáticamente al crear la configuración en Authorization).

Encabezados de respuesta: Content-Type: application/json; charset=utf-8

Respuesta: En formato JSON con la siguiente estructura.

```
{
  "facturas": [
    {
      "id": "123",
      "cliente": "Empresa XYZ",
      "monto": 1500.75,
      "fecha_emision": "2023-05-01",
      "estado": "pagada"
    },
....
  ]
}
```

Documentación de la **API B** y endpoints

URL o Endpoint: https://api.sistemaB.com/bills

Método: GET

Descripción: API B es un servicio REST que permite registrar facturas en su sistema a partir de datos como invoice_id, customer, amount_due, date_issued, y status. La API está diseñada para recibir facturas en formato JSON y almacenarlas para posterior procesamiento.

Parámetros de consulta:

start_date: Fecha de inicio para el rango de consulta (Formato: YYYY-MM-DD).

end_date: Fecha de fin para el rango de consulta (Formato: YYYY-MM-DD).

Seguridad: Autenticación mediante Clave API en el encabezado x-api-key: <api_key>. Este método de autenticación garantiza que solo clientes autorizados puedan interactuar con la API.

Manejo de errores y excepciones: Si los parámetros fecha_inicio y fecha_fin no son válidos, la API devolverá un código de error **400** (Bad Request). En general, en caso de un error interno de la parte cliente, se retornará un código **4XX** y en caso de un error de servidor, se retornará un código **5XX**. Si la petición se procesa correctamente la salida será de tipo **2XX**.

Las salidas de la petición que vamos a especificar por ser las más comunes, son:

200 OK. La solicitud fue exitosa. Es la respuesta más común y se utiliza en diversas operaciones como GET, POST, PUT, etc.

204 No Content. La solicitud fue exitosa, pero no hay contenido en la respuesta (generalmente utilizado con DELETE o PUT).

400 Bad Request. La solicitud tiene un error de sintaxis o está malformada.

401 Unauthorized. La solicitud requiere autenticación. El cliente debe proporcionar credenciales válidas (Bearer Token, API Key).

403 Forbidden. El servidor entiende la solicitud, pero no tiene permiso para ejecutarla (a diferencia del 401, las credenciales no ayudarían).

404 Not Found. El servidor no puede encontrar el recurso solicitado.

500 Internal Server Error. El servidor ha encontrado una condición inesperada que le impide completar la solicitud.

502 Bad Gateway. El servidor, actuando como puerta de enlace o proxy, recibió una respuesta no válida del servidor upstream.

503 Service Unavailable. El servidor no puede manejar la solicitud temporalmente, generalmente debido a mantenimiento o sobrecarga.

504 Gateway Timeout. El servidor, actuando como puerta de enlace o proxy, no recibió una respuesta a tiempo del servidor upstream.

Este detalle lo podemos consultar en la documentación de Swagger/OpenAPI.

Encabezados de solicitud: x-api-key: <api_key> (generado automáticamente al crear la configuración en Authorization).

Encabezados de respuesta: Content-Type: application/json; charset=utf-8

Respuesta: En formato JSON con la siguiente estructura.


```
{
  "invoices": [
    {
      "invoice_id": "456",
      "customer": "Company ABC",
      "amount_due": 2000.50,
```

```
            "date_issued": "2023-05-02",
            "status": "unpaid"
        }
    ]
}
```


¿Qué se pide realizar?

Realizar un documento explicando todo lo que se necesita realizar para la integración entre esas 2 APIs.

Desarrollo del Informe de Integración de APIs

Descripción General

En este informe se van a detallar los pasos necesarios para realizar la integración entre 2 APIs. El objetivo del desarrollo es extraer datos de la API A hacia la API B, y en la API B vamos a darle el formato requerido a los datos, para finalmente validar y mostrar la respuesta esperada en formato JSON.

Flujo del Proceso de consulta de la API B

PASO 1 - Consulta de Facturas desde API B:
Se realiza una petición GET al endpoint https://api.sistemaB.com/bills
Se envían los parámetros start_date y end_date, para definir el rango de las facturas que se desean consultar.
Se Autentica utilizando la clave API en el encabezado x-api-key
Si la autenticación es correcta, continuamos al paso 2.

PASO 2 – Consulta de Facturas en la API A:
Se realiza una petición GET al endpoint https://api.sistemaA.com/facturas.
Se envían los parámetros start_date y end_date para definir el rango de las facturas que se desean consultar. Estos serán los mismos parámetros fecha_inicio y fecha_fin que se han definido previamente en la petición GET a API B.
Se Autentica utilizando el token Bearer en el encabezado.
Si la autenticación es correcta, **se recibe la respuesta en formato JSON, y se extraen los datos para procesarlos en la API B.**

PASO 3 - Transformación de Datos en la API B:
En la API B vamos a convertir el formato de los datos de API A al formato que requiere API B.

Ejemplo de transformación:
id (API A) pasa a ser invoice_id (API B).
cliente pasa a ser customer.
monto pasa a ser amount_due.
fecha_emision pasa a ser date_issued.
estado pasa a ser status.

Partner Engineer: Eleana Rey Quijada
https://www.linkedin.com/in/eleanarey/

Partiendo de los principios de diseño de sistemas distribuidos y buenas práctica en la arquitectura de software, la transformación de los datos obtenidos en la **API A,** se realiza dentro de la **API B**. Porque la **API A** actúa únicamente como un proveedor de datos, mientras que la **API B** es responsable de asegurar que los datos sean recibidos y almacenados en el formato correcto. Este enfoque respeta los **principios de responsabilidad única**, donde cada API cumple con su rol específico.

También debe ser así para facilitar su **mantenimiento y escalabilidad,** al centralizar la lógica de transformación en la **API B**, ya que, por ejemplo, si se introducen nuevas fuentes de datos o cambia el formato de los datos, solo se necesita modificar la lógica en la **API B**, sin afectar a la **API A** ni a otros sistemas que consuman sus datos.

Otro principio que se ha tomado en cuenta a la hora de definir donde se van a transformar los datos es el de **Control y flexibilidad**, donde a **API B** tiene control total sobre los datos que recibe y cómo se deben transformar para ajustarse a sus propios requisitos, sin depender de cambios en la **API A**.

PASO 4 – Validación de la Respuesta. Se recibe y valida la información:
Ya tendríamos la respuesta en formato JSON y debemos continuar con la validación de los datos para finalmente recibir el documento JSON con la respuesta esperada.
Entonces, se recibe la respuesta en formato JSON.
Se verifica que la respuesta JSON de API B contenga las facturas correctamente enviadas y formateadas.
Revisar los campos como invoice_id, customer, amount_due, y status para asegurarse de que las facturas estén correctamente sincronizadas.

PASO 5 – Envío de la respuesta al cliente en formato JSON y fin del proceso.

Diagrama de secuencia del consumo de las APIs en la integración

A continuación, se muestra el diagrama de secuencia UML que muestra la interacción donde el Cliente envía las fechas a la API B, y esta a su vez consulta la API A para obtener la información solicitada y luego la devuelve al Cliente.

```
sequenceDiagram
    participant Cliente
    participant API_B as API B
    participant API_A as API A

    Cliente->>API_B: GET /bills con start_date y end_date
    API_B->>API_B: Validar autenticación (x-api-key)

    alt Autenticación fallida
```

Partner Engineer: Eleana Rey Quijada
https://www.linkedin.com/in/eleanarey/

```
    API_B -->> Cliente: 401 Unauthorized
  else Solicitud malformada
    API_B -->> Cliente: 400 Bad Request
  else Sin permiso
    API_B -->> Cliente: 403 Forbidden
  else No encontrado
    API_B -->> Cliente: 404 Not Found
  else Error interno del servidor
    API_B->>API_B: Validar conexión al servidor
    alt Error en la conexión al servidor
      API_B -->> Cliente: 500 Internal Server Error
    else Respuesta no válida de upstream
      API_B -->> Cliente: 502 Bad Gateway
    else Servicio no disponible
      API_B -->> Cliente: 503 Service Unavailable
    else Tiempo de espera agotado
      API_B -->> Cliente: 504 Gateway Timeout
    end
  else Autenticación exitosa
    API_B->>API_A: GET /facturas con fecha_inicio y fecha_fin
    API_A->>API_A: Validar autenticación (Bearer Token)
    alt Manejo de errores del lado de cliente o servidor
      API_A-->>API_B: Error de tipo 4xx-5xx
    else Autenticación exitosa
      API_A-->>API_B: 200 OK (facturas JSON)
      API_B->>API_B: Transformar datos (id a invoice_id, cliente a customer)
      API_B-->>Cliente: 200 OK (facturas transformadas)
      alt Respuesta sin contenido
        API_B -->> Cliente: 204 No Content
      end
    end
```

Partner Engineer: Eleana Rey Quijada
https://www.linkedin.com/in/eleanarey/

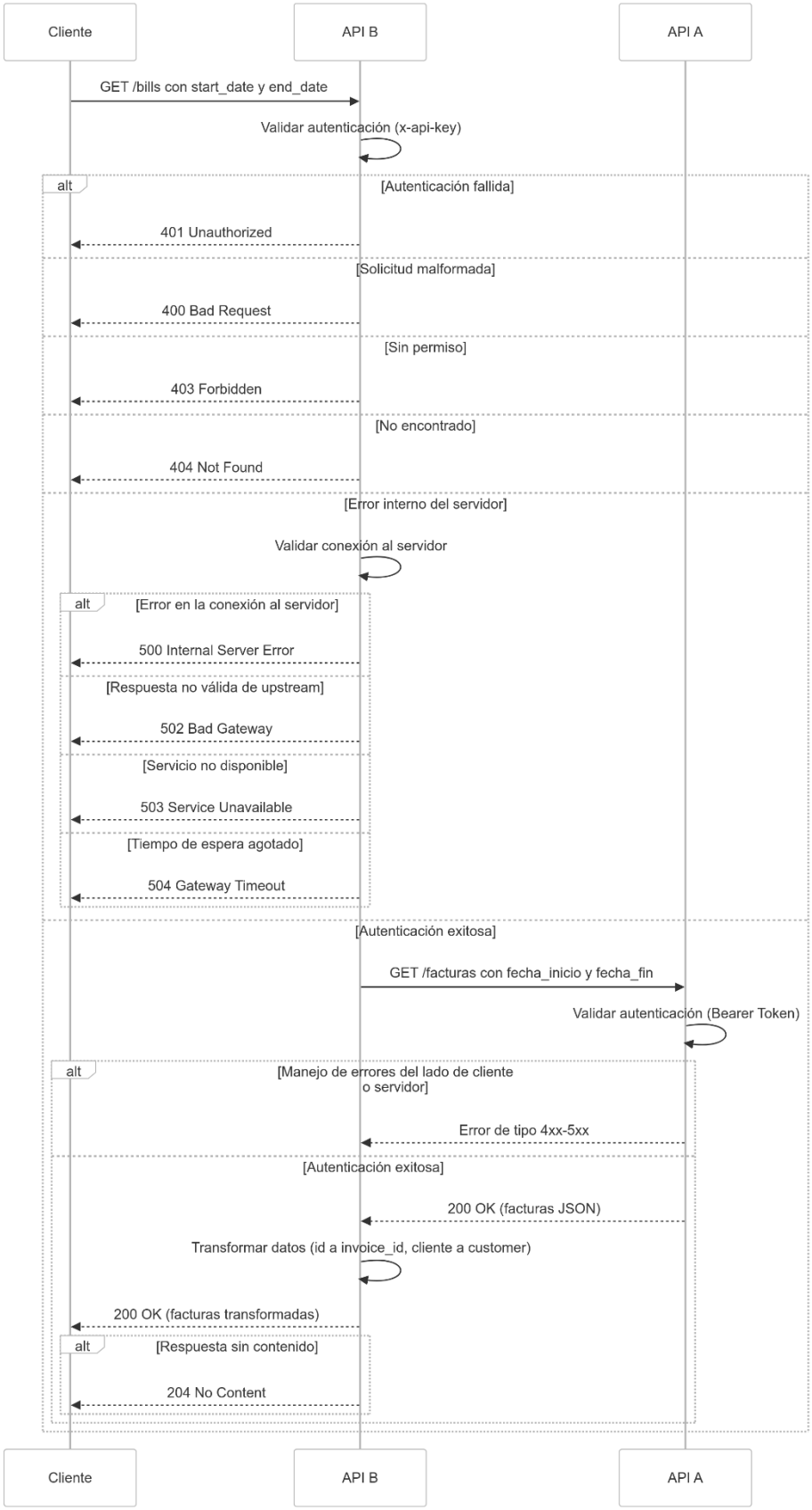


FIGURA 1 - DIAGRAMA DE SECUENCIA

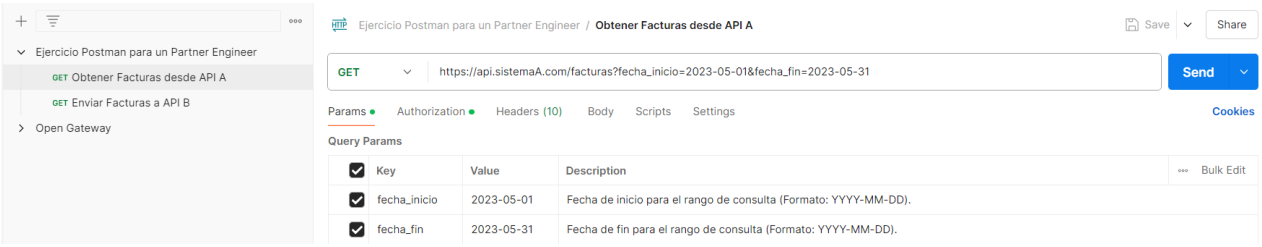
Partner Engineer: Eleana Rey Quijada
https://www.linkedin.com/in/eleanarey/

Configuración de las APIs en Postman

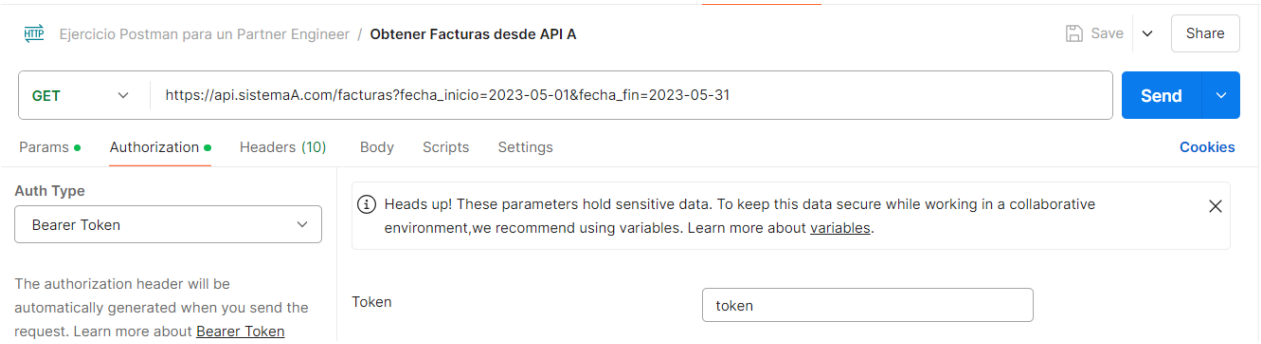
Las APIs se deben configurar en Postman estableciendo los endpoints, métodos HTTP, parámetros de consulta, y encabezados de autenticación, como se muestra en esta sección.

Capturas de pantalla Postman

Parámetros API A:



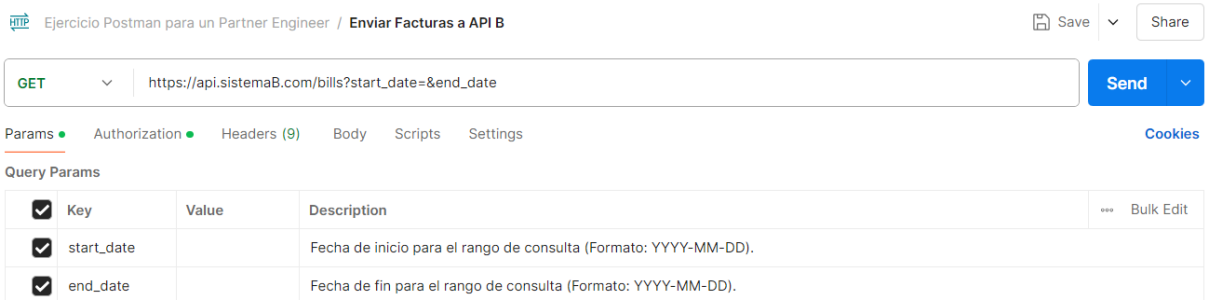
Autorización API A:



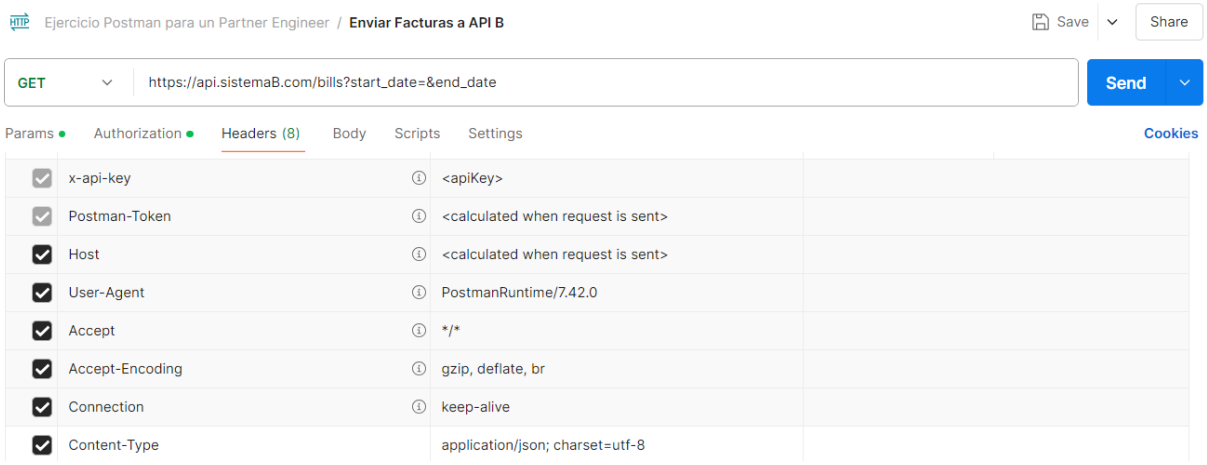
Encabezados API A:

Partner Engineer: Eleana Rey Quijada
https://www.linkedin.com/in/eleanarey/

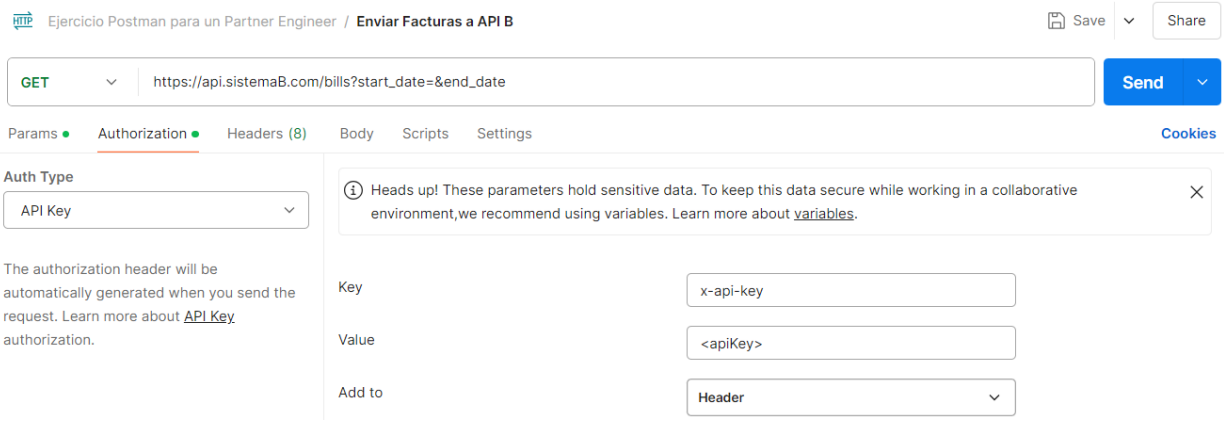
Parámetros API B:



Autorización API B:



Encabezados API B:



Codigo export_postman.json:


```
{
        "info": {
                "_postman_id": "adb7872f-886f-4ef7-8164-1915b8988806",
                "name": "Ejercicio Postman para un Partner Engineer",
                "schema": "https://schema.getpostman.com/json/collection/v2.1.0/collection.json",
                "_exporter_id": "38599945"
        },
        "item": [
```


Partner Engineer: Eleana Rey Quijada

https://www.linkedin.com/in/eleanarey/

```
{
	"name": "Obtener Facturas desde API A",
	"request": {
		"auth": {
			"type": "bearer",
			"bearer": [
				{
					"key": "token",
					"value": "token",
					"type": "string"
				}
			]
		},
		"method": "GET",
		"header": [
			{
				"key": "Authorization",
				"value": "Bearer <token>",
				"type": "text"
			},
			{
				"key": "Content-Type",
				"value": "application/json; charset=utf-8",
				"type": "text"
			},
			{
				"key": "",
				"value": "",
				"type": "text",
				"disabled": true
			}
		],
		"url": {
			"raw": "https://api.sistemaA.com/facturas?fecha_inicio=2023-05-
01&fecha_fin=2023-05-31",
			"protocol": "https",
			"host": [
				"api",
				"sistemaA",
				"com"
			],
			"path": [
				"facturas"
			],
			"query": [
				{
					"key": "fecha_inicio",
					"value": "2023-05-01",
					"description": "Fecha de inicio para el rango de
consulta (Formato: YYYY-MM-DD)."
				},
				{
					"key": "fecha_fin",
					"value": "2023-05-31",
					"description": "Fecha de fin para el rango de
consulta (Formato: YYYY-MM-DD)."
				}
			]
		}
	},
	"response": []
},
{
	"name": "Enviar Facturas a API B",
	"protocolProfileBehavior": {
		"disableBodyPruning": true
	},
	"request": {
		"method": "GET",
		"header": [
```


Partner Engineer: Eleana Rey Quijada

https://www.linkedin.com/in/eleanarey/

```
                                        {
                                                "key": "x-api-key",
                                                "value": "<api_key>",
                                                "type": "text"
                                        },
                                        {
                                                "key": "Content-Type",
                                                "value": "application/json",
                                                "type": "text"
                                        }
                                ],
                                "body": {
                                        "mode": "raw",
                                        "raw": "{\n    \"invoice_id\": \"123\",\n    \"customer\": \"Empresa
XYZ\",\n    \"amount_due\": 1500.75,\n    \"date_issued\": \"2023-05-01\",\n    \"status\": \"pagada\"\n}"
                                },
                                "url": {
                                        "raw": "https://api.sistemaB.com/bills",
                                        "protocol": "https",
                                        "host": [
                                                "api",
                                                "sistemaB",
                                                "com"
                                        ],
                                        "path": [
                                                "bills"
                                        ]
                                }
                        },
                        "response": []
                }
        ]
}
```

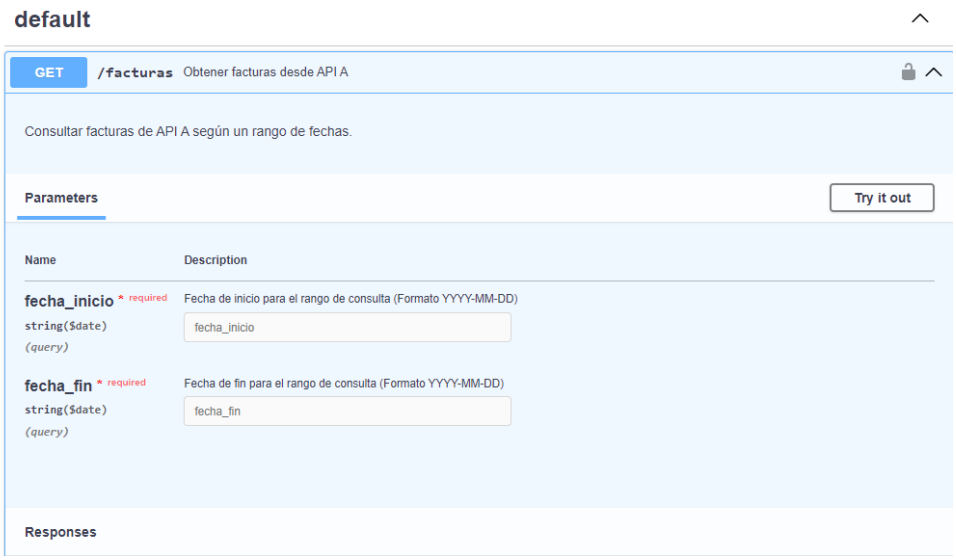

Documentación de las APIs en Swagger

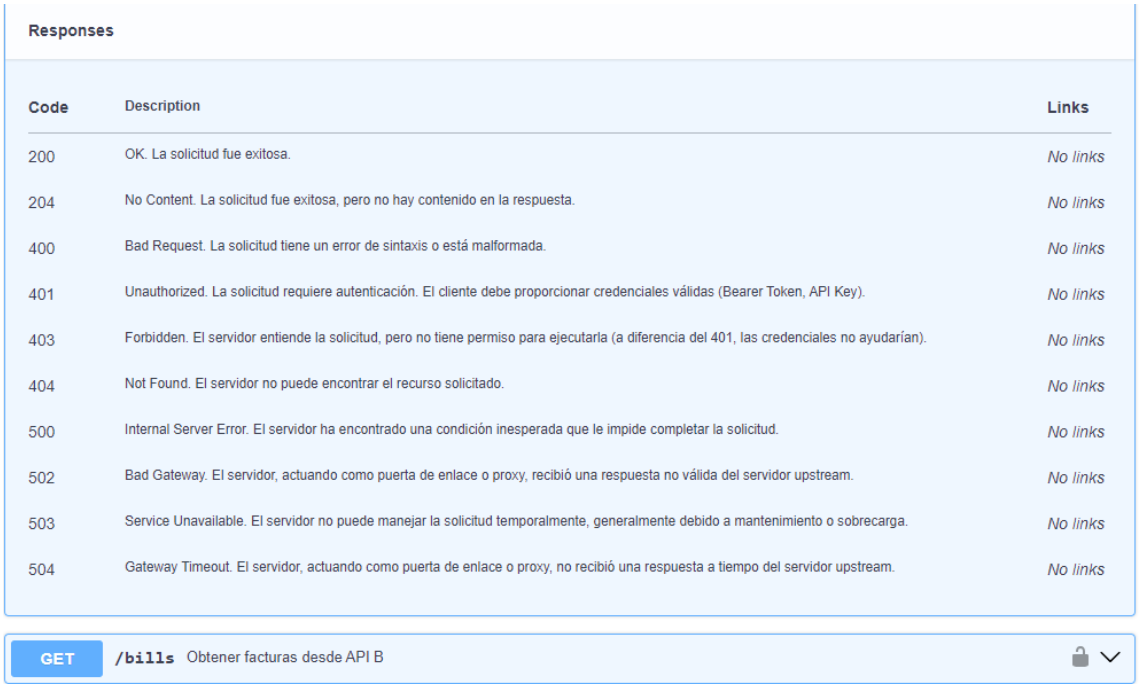
La documentación en Swagger será clave para entender los detalles técnicos de cada uno de los endpoints que necesitamos implementar en las APIs y nos facilitará la visualización y prueba de los endpoints directamente desde su interfaz. Esto nos permitirá simular y validar cómo funcionará API B antes de implementarla. También nos ayudará a que los equipos involucrados tengamos una visión clara y compartida de cómo debe operar la API. Por qué debe mantenerse actualizada. Actualmente, partimos de la información documentada en el fichero “swagger_apis_partner_engineer.yml”. Se dejan a continuación capturas de pantalla y el contenido del fichero mencionado.

Capturas de pantalla Swagger

Partner Engineer: Eleana Rey Quijada

https://www.linkedin.com/in/eleanarey/





Codigo swagger_apis_partner_engineer.yml

```
openapi: 3.0.1
info:
  title: Ejercicio de APIs para un Partner Engineer
  description: Servicio que permite realizar consultar de facturas por un rango de fechas. El objetivo es integrar 2 APIs. La API A es la API de la cual obtendremos los datos, y la API B es la que queremos alimentar.
  version: 1.0.0
servers:
  - url: https://api.sistemaA.com
    description: Sistema A Server
  - url: https://api.sistemaB.com
    description: Sistema B Server
paths:
  /facturas:
    get:
      summary: Obtener facturas desde API A
      description: Consultar facturas de API A según un rango de fechas.
      operationId: getFacturasSistemaA
      parameters:
```

Partner Engineer: Eleana Rey Quijada

https://www.linkedin.com/in/eleanarey/

```
      - name: fecha_inicio
        in: query
        description: Fecha de inicio para el rango de consulta (Formato YYYY-MM-DD)
        required: true
        schema:
          type: string
          format: date
      - name: fecha_fin
        in: query
        description: Fecha de fin para el rango de consulta (Formato YYYY-MM-DD)
        required: true
        schema:
          type: string
          format: date
    responses:
      '200':
        description: OK. La solicitud fue exitosa.
      '204':
        description: No Content. La solicitud fue exitosa, pero no hay contenido en la respuesta.
      '400':
        description: Bad Request. La solicitud tiene un error de sintaxis o está malformada.
      '401':
        description: Unauthorized. La solicitud requiere autenticación. El cliente debe proporcionar credenciales válidas
(Bearer Token, API Key).
      '403':
        description: Forbidden. El servidor entiende la solicitud, pero no tiene permiso para ejecutarla (a diferencia del
401, las credenciales no ayudarían).
      '404':
        description: Not Found. El servidor no puede encontrar el recurso solicitado.
      '500':
        description: Internal Server Error. El servidor ha encontrado una condición inesperada que le impide completar
la solicitud.
      '502':
        description: Bad Gateway. El servidor, actuando como puerta de enlace o proxy, recibió una respuesta no válida
del servidor upstream.
      '503':
        description: Service Unavailable. El servidor no puede manejar la solicitud temporalmente, generalmente
debido a mantenimiento o sobrecarga.
      '504':
        description: Gateway Timeout. El servidor, actuando como puerta de enlace o proxy, no recibió una respuesta a
tiempo del servidor upstream.
    security:
      - bearerAuth: []
  /bills:
    get:
      summary: Obtener facturas desde API B
      description: Consultar facturas de API B según un rango de fechas.
      operationId: getFacturasSistemaB
      parameters:
        - name: start_date
          in: query
          description: Fecha de inicio para el rango de consulta (Formato YYYY-MM-DD)
          required: true
          schema:
            type: string
            format: date
        - name: end_date
          in: query
          description: Fecha de fin para el rango de consulta (Formato YYYY-MM-DD)
          required: true
          schema:
            type: string
            format: date
      responses:
        '200':
          description: OK. La solicitud fue exitosa.
        '204':
          description: No Content. La solicitud fue exitosa, pero no hay contenido en la respuesta.
        '400':
          description: Bad Request. La solicitud tiene un error de sintaxis o está malformada.
```

Partner Engineer: Eleana Rey Quijada

https://www.linkedin.com/in/eleanarey/

```
      '401':
        description: Unauthorized. La solicitud requiere autenticación. El cliente debe proporcionar credenciales válidas (Bearer Token, API Key).
      '403':
        description: Forbidden. El servidor entiende la solicitud, pero no tiene permiso para ejecutarla (a diferencia del 401, las credenciales no ayudarían).
      '404':
        description: Not Found. El servidor no puede encontrar el recurso solicitado.
      '500':
        description: Internal Server Error. El servidor ha encontrado una condición inesperada que le impide completar la solicitud.
      '502':
        description: Bad Gateway. El servidor, actuando como puerta de enlace o proxy, recibió una respuesta no válida del servidor upstream.
      '503':
        description: Service Unavailable. El servidor no puede manejar la solicitud temporalmente, generalmente debido a mantenimiento o sobrecarga.
      '504':
        description: Gateway Timeout. El servidor, actuando como puerta de enlace o proxy, no recibió una respuesta a tiempo del servidor upstream.
    security:
      - apiKeyAuth: []
components:
  securitySchemes:
    bearerAuth:
      type: http
      scheme: bearer
      bearerFormat: JWT
    apiKeyAuth:
      type: apiKey
      in: header
      name: x-api-key
```

Integración de las APIs

Finalmente, para practicar las validaciones y simular el backend, se debe crear un script que realice la consulta de las facturas desde la API B usando los parámetros start_date y end_date. Luego, consultamos API A para obtener las facturas en el mismo rango de fechas que las obtenidas de API B. Luego los datos de las facturas de API A se transforman para coincidir con el formato que espera el cliente. En este caso, se va a realizar el script en Python.

Codigo integracion.py:

```
import requests
import json
# Configuración de las APIs
api_b_url = "https://api.sistemaB.com/bills"
api_a_url = "https://api.sistemaA.com/facturas"

api_b_key = "tu_api_key_de_sistemaB"
api_a_token = "tu_token_de_sistemaA"

# Función para consultar la API B
def consultar_api_b(start_date, end_date):
    headers = {
        'x-api-key': api_b_key
    }
    params = {
        'start_date': start_date,
        'end_date': end_date
    }
```

Partner Engineer: Eleana Rey Quijada
https://www.linkedin.com/in/eleanarey/

```
    response = requests.get(api_b_url, headers=headers, params=params)
    return response

# Función para consultar la API A
def consultar_api_a(start_date, end_date):
    headers = {
        'Authorization': f'Bearer {api_a_token}'
    }
    params = {
        'fecha_inicio': start_date,
        'fecha_fin': end_date
    }
    response = requests.get(api_a_url, headers=headers, params=params)
    return response
# Función para transformar los datos de la API A al formato de la API B
def transformar_datos(api_a_data):
    facturas_transformadas = []
    for factura in api_a_data['facturas']:
        factura_transformada = {
            'invoice_id': factura['id'],
            'customer': factura['cliente'],
            'amount_due': factura['monto'],
            'date_issued': factura['fecha_emision'],
            'status': factura['estado']
        }
        facturas_transformadas.append(factura_transformada)
    return facturas_transformadas
# Función principal que realiza la integración
def ejecutar_integracion(start_date, end_date):
    # Paso 1 - Consulta de API B
    print(f"Consultando API B para fechas {start_date} - {end_date}...")
    response_b = consultar_api_b(start_date, end_date)

    if response_b.status_code == 401:
        print("Error: 401 Unauthorized en API B.")
        return
    elif response_b.status_code == 400:
        print("Error: 400 Bad Request en API B.")
        return
    elif response_b.status_code == 403:
        print("Error: 403 Forbidden en API B.")
        return
    elif response_b.status_code == 404:
        print("Error: 404 Not Found en API B.")
        return
    elif response_b.status_code >= 500:
        print(f"Error: {response_b.status_code} Server Error en API B.")
        return

    # Paso 2 - Consulta de API A
    print(f"Consultando API A para fechas {start_date} - {end_date}...")
    response_a = consultar_api_a(start_date, end_date)

    if response_a.status_code == 401:
        print("Error: 401 Unauthorized en API A.")
        return
    elif response_a.status_code >= 400:
        print(f"Error: {response_a.status_code} Error en API A.")
```

Partner Engineer: Eleana Rey Quijada
https://www.linkedin.com/in/eleanarey/

```
            return

        # Extraer los datos de la API A
        api_a_data = response_a.json()

        # Paso 3 - Transformar los datos
        print("Transformando datos de la API A al formato de la API B...")
        facturas_transformadas = transformar_datos(api_a_data)

        # Paso 4 - Validación de la Respuesta
        print("Validando datos transformados...")
        for factura in facturas_transformadas:
            if not all(key in factura for key in ['invoice_id', 'customer', 'amount_due', 'date_issued', 'status']):
                print("Error: Factura transformada incompleta.")
                return

        # Paso 5 - Enviar la respuesta al cliente
        print("Respuesta final:")
        print(json.dumps(facturas_transformadas, indent=2))

    # Ejecutar el script de integración con un rango de fechas
    if __name__ == "__main__":
        start_date = "2023-01-01"
        end_date = "2023-01-31"
        ejecutar_integracion(start_date, end_date)
```

Explicación del Script Python:

Consulta de API B (consultar_api_b): Se envía una solicitud GET con los parámetros start_date y end_date, autenticando con la clave x-api-key. Se manejan diferentes errores HTTP como 401 Unauthorized, 403 Forbidden, y errores del servidor como 500 Internal Server Error.

Consulta de API A (consultar_api_a): Se realiza la solicitud a la API A con autenticación de tipo Bearer Token. Al igual que en la API B, se maneja la autenticación y otros errores de respuesta.

Transformación de Datos (transformar_datos): Los datos recibidos de la API A son transformados para cumplir con el formato esperado por la API B. Aquí, los campos se renombran (ej. id a invoice_id, cliente a customer, etc.).

Validación: Los datos transformados se validan para asegurarse de que contienen los campos obligatorios antes de ser enviados al cliente.

Envío de Respuesta: Se imprime la respuesta final con las facturas transformadas en formato JSON.

Ejecución:Al ejecutar este script con los parámetros de fecha, primero consultará la API B, luego la API A, transformará los datos, los validará, y finalmente imprimirá la respuesta en formato JSON.

Partner Engineer: Eleana Rey Quijada
https://www.linkedin.com/in/eleanarey/