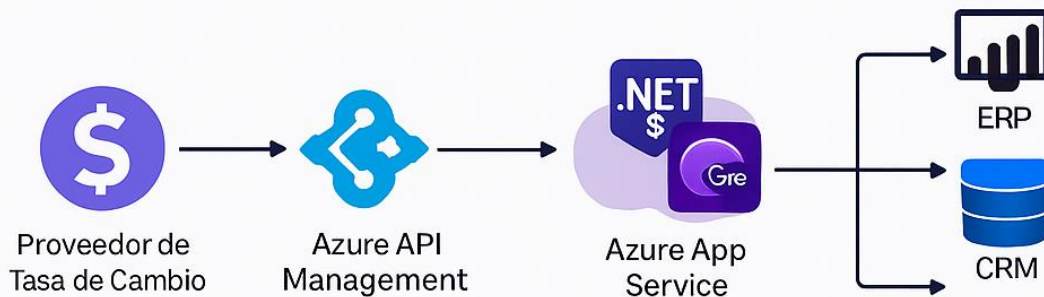




**Paper técnico: Arquitectura y desarrollo de
microservicio empresarial para Tasa de Cambio oficial
en Azure con .NET Core 8**

Implementación de Microservicio Empresarial para Tasa de Cambio Oficial



Código Ejemplo

```
[Authorize]
[ApiController]
public class ExchangeRatesController <
{
    [HttpGet]
    public async Task<
        ActionResult > GetLatestRate()
    {
        // Código oniso consulta API externa
        return Ok(exchangeRate).
    }
}
```

Código ejemplo

```
[Authorize]
[ApiController]
public class ExchangeRates
{
    [HttpGet]
    public async Task<
        ActionResult > GetLatest
    {
        // Código omrito consulta API externa
        return OK(exchangeRate).
    }
}
```

Servicios en Azure

- Azure API Management
- Azure Key Vault
- Azure App Service
- Azure SQL Database

Consideraciones

- ✓ **Gobierno de TI**
Control de versiones y consumo (monitoreo más baja latencia)
- ✓ **Gobierno de Datos**
Creación de una fuente oficial de datos
- ✓ **Seguridad de Información**
Control de acceso, protección a datos integración y autenticación segura

Paper técnico: Arquitectura y desarrollo de microservicio empresarial para Tasa de Cambio oficial en Azure con .NET Core 8

1. Introducción

La gestión coherente de datos críticos como la Tasa de Cambio oficial impacta directamente en operaciones contables, financieras y comerciales. En este paper proponemos la construcción de un microservicio interno empresarial que actúe como fuente oficial de consulta para esta tasa, integrando el ecosistema Azure con .NET Core 8 bajo principios de arquitectura empresarial, seguridad y gobierno de datos.

2. Escenario y objetivo

Escenario: La organización cuenta con un proveedor externo de tasas de cambio (servicio pago vía API REST). Sin embargo, diferentes áreas (ERP, CRM, vertical de operaciones) consultan esta fuente de forma descoordinada y duplicada, generando inconsistencias y trazabilidad limitada.

Objetivo: Centralizar la consulta y distribución de la Tasa de Cambio a través de un **microservicio oficial**, que se alimente del servicio externo, almacene/control de versiones, y sirva a todos los sistemas internos.

3. Arquitectura propuesta

Servicios utilizados en Azure:

- **Azure App Service:** Despliegue del microservicio en .NET Core 8.
- **Azure Key Vault:** Almacenamiento seguro de claves de API del proveedor.
- **Azure API Management:** Publicación, control de acceso y analítica del servicio interno.
- **Azure SQL Database / Cosmos DB:** Almacenamiento local para cacheo y auditoría.
- **Azure Functions / Logic Apps:** Consulta diaria al proveedor externo.
- **Azure Monitor / Application Insights:** Trazabilidad y monitoreo.
- **Azure AD (opcional):** Integración con control de acceso corporativo.

Diagrama (versión textual):

[Proveedor externo API]

|

| (Azure Function diaria)

v

[Microservicio oficial - App Service - .NET Core 8] <--- Azure Key Vault

|

| (Exposición vía API Management)

v

[Azure SQL / Cosmos DB]

|

|-----|

v

v

[ERP]

[CRM]

|

|

[Vertical de Negocio] <-----

4. Ejemplo básico de código en .NET Core 8

[ApiController]

[Route("api/tasa-cambio")]

public class TasaCambioController : ControllerBase

{

private readonly ITasaCambioService _service;

public TasaCambioController(ITasaCambioService service)

{

_service = service;

}

[HttpGet("oficial")] // api/tasa-cambio/oficial

```

public async Task<IActionResult> GetOficial()
{
    var tasa = await _service.ObtenerTasaCambioOficialAsync();
    return Ok(tasa);
}
}

```

Servicio ejemplo:

```

public class TasaCambioService : ITasaCambioService
{
    private readonly IHttpClientFactory _httpClientFactory;
    private readonly IConfiguration _config;

    public TasaCambioService(IHttpClientFactory httpClientFactory, IConfiguration config)
    {
        _httpClientFactory = httpClientFactory;
        _config = config;
    }

    public async Task<TasaCambioDto> ObtenerTasaCambioOficialAsync()
    {
        var client = _httpClientFactory.CreateClient();
        var apiKey = _config["TasaCambio:ApiKey"];
        var response = await client.GetAsync($"https://api.externa.com/latest?key={apiKey}");

        if (response.IsSuccessStatusCode)
        {
            var content = await response.Content.ReadAsStringAsync();
            var data = JsonConvert.DeserializeObject<TasaCambioDto>(content);
            return data;
        }
    }
}

```

```
}

    throw new Exception("Error al obtener la tasa de cambio");
}
}
```

5. Consideraciones de seguridad

- Uso de **Azure Key Vault** para almacenar y rotar claves.
 - Integración opcional con **Azure AD** o JWT interno.
 - Registro de accesos y errores mediante **Application Insights**.
 - API protegida con throttling y versionamiento desde **API Management**.
-

6. Gobierno de TI y de datos

- **Fuente oficial única (SSOT):** El microservicio actúa como autoridad oficial de la tasa.
 - **Políticas de calidad:** Se validan estructuras de datos y control de versiones.
 - **Auditoría:** Registro de actualizaciones y accesos a la API.
 - **Catálogo de datos:** La Tasa de Cambio se incorpora al catálogo corporativo como dato maestro.
 - **Segregación de funciones:** La gestión del servicio se separa de los consumidores (ERP, CRM, etc.).
-

7. Conclusión

Este enfoque permite consolidar una gestión moderna de datos críticos mediante microservicios, acelerando la interoperabilidad, reduciendo errores y fortaleciendo los principios de gobierno corporativo y seguridad. Es un caso que puede escalarse a cualquier otro tipo de dato clave (tributario, financiero, operativo).