

**Reto técnico
Desarrollador Backend Senior**

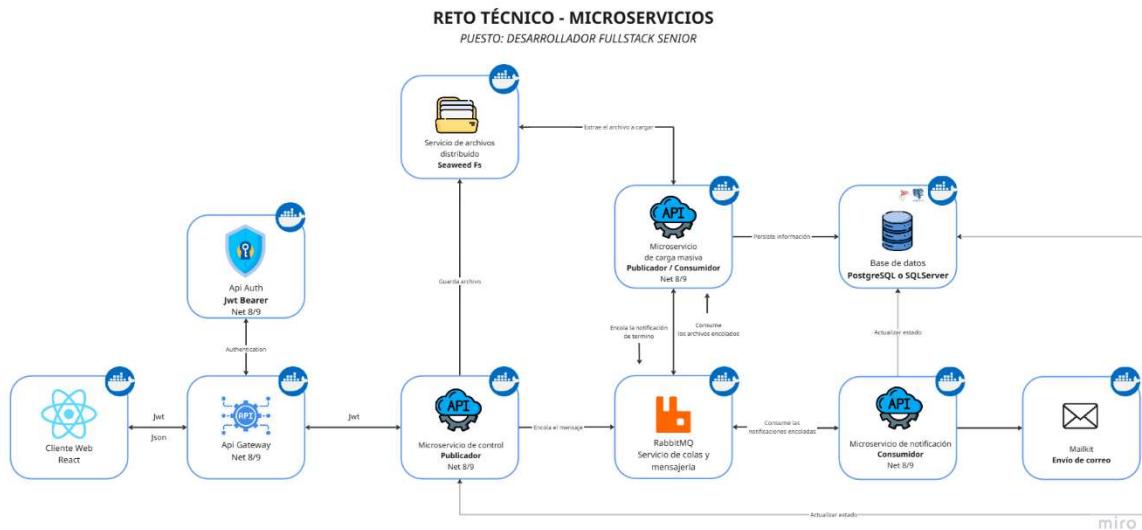
Casino Atlantic City - 2026

Reto Técnico – Microservicios

Postulación: Desarrollador Fullstack Senior

Este examen práctico evalúa tus habilidades en **arquitectura de microservicios, .NET 8/9, mensajería con Servicio de Colas (RabbitMQ/Kafka/otro), cliente web React, procesamiento asíncrono, implementación de colas, trazabilidad, persistencia en PostgreSQL o SQL Server, notificaciones por correo y buenas prácticas de desarrollo.**

El reto simula un flujo real de **carga masiva de datos**, completamente distribuido, siguiendo la arquitectura mostrada en el diagrama entregado.



1. Objetivo del Reto

Implementar un sistema de microservicios donde un **cliente web/Colecciones Postman** que permita subir un archivo Excel con información masiva, este archivo se procese de manera asíncrona mediante Servicio de Colas (RabbitMQ/Kafka/otro) y finalmente se envíe una notificación por correo al usuario una vez que la carga haya finalizado.

El reto debe ser **100% funcional, dockerizado** y siguiendo buenas prácticas **senior**.

2. Arquitectura General a Implementar

La solución completa consiste en:

1. Cliente Web React (Opcional pero valorado)

En caso no se realice la interfaz con React se deben enviar las colecciones postman con cada petición Permite:

- Iniciar Sesión
 - Subir un archivo Excel (.xlsx).
 - Consultar el historial de cargas.
 - Consultar el contenido del archivo excel subido.
 - Ver el estado de cada procesamiento, que puede ser:
 - **Pendiente**
 - **En proceso**
 - **Cargado**
 - **Finalizado**
 - **Notificado**
-

2. API Gateway (NET 8/9)

Punto de entrada centralizado que:

- Recibe todas las solicitudes del cliente web.
 - Valida JWT.
 - Reenvía peticiones al microservicio correspondiente.
-

3. Microservicio 0 — Authentication (NET 8/9)

Funciones mínimas:

- Expone un endpoint para autenticación de usuarios (/auth/login).
 - Valida credenciales contra la fuente de identidad (Base de datos, Identity Provider o servicio interno).
 - Genera y retorna un **JWT Bearer** con claims del usuario.
 - Ofrece endpoint para refreshar tokens (/auth/refresh) si se implementa Refresh Tokens (Opcional valorado).
-

✓ 4. Microservicio 1 — Control / Publicador (NET 8/9)

Funciones:

- Recibe desde el Gateway la solicitud para cargar el archivo.
 - Validar que archivo tenga no exceda el tamaño maximo configurado.
 - Guarda trazabilidad del archivo (estado inicial: **Pendiente**).
 - Publica un mensaje en Servicio de Colas (RabbitMQ/Kafka/otro) para que el archivo sea procesado.
 - Envía el archivo al servicio de almacenamiento SeaweedFS.
-

✓ 5. Microservicio 2 — Carga Masiva (Consumidor / Publicador) (NET 8/9)

Responsabilidades:

- Escucha la cola Servicio de Colas (RabbitMQ/Kafka/otro).
 - Descarga el archivo desde SeaweedFS.
 - Procesa el registro.
 - Realiza validaciones y limpieza de datos del Excel.
 - Inserta la información en PostgreSQL o SQL Server.
 - Marca la trazabilidad en los estados:
 - **En proceso**
 - **Cargado**
 - **Finalizado**
 - Publica una notificación en una segunda cola Servicio de Colas (RabbitMQ/Kafka/otro) indicando que el proceso ha terminado.
-

✓ 6. Microservicio 3 — Notificaciones (Consumidor) (NET 8/9)

- Escucha la cola de notificaciones.
 - Envía un correo al usuario indicando que la carga finalizó.
 - Usa MailKit.
 - Actualiza el estado final a **Notificado**.
-

✓ 7. Servicio de Colas (RabbitMQ/Kafka/otro)

- Cola 1: `carga_masiva`
 - Cola 2: `notificaciones`
-

✓ 8. Base de Datos — PostgreSQL o SQL Server

Tablas sugeridas:

- CargaArchivo (trazabilidad)
 - DetalleCarga (si fuera necesario)
 - DataProcesada (registros extraídos del Excel)
-

✓ 9. SeaweedFS

Servicio distribuido para almacenar los archivos Excel subidos.

🔗 3. Flujo Completo del Caso de Uso

0 Microservicio de Authentication

- El cliente web envía credenciales a /auth/login.
 - El microservicio valida las credenciales contra la fuente de identidad.
 - Genera y retorna un **JWT Bearer** con los claims del usuario.
 - (Opcional) Expone /auth/refresh para renovar el token.
 - El cliente usa este JWT en todas las solicitudes posteriores al Gateway.
-

1 El usuario sube un Excel desde el cliente web

- El archivo se envía al **API Gateway**, incluyendo el **JWT**.
 - El Gateway valida el token y reenvía la solicitud al **Microservicio de Control**.
-

2 Microservicio de Control

Funciones adicionales de negocio:

- Valida que el usuario tenga permiso para ejecutar cargas masivas.
- Valida que el archivo tenga un tamaño permitido y extensión correcta.
- Registra auditoría de quién subió el archivo y cuándo.

Flujo principal:

- Guarda un registro en PostgreSQL o SQL Server con estado inicial:
 - Pendiente
- Sube el archivo a **SeaweedFS**.
- Publica en Servicio de Colas (RabbitMQ/Kafka/otro) el mensaje:

```
{  
  "idCarga": 123,  
  "rutaArchivo": "seaweed://.../archivo.xlsx",
```

```
"usuario": "user@example.com"
}
```

3 Microservicio de Carga Masiva

Lógica de negocio incluida — Validación de duplicidad por periodo

1. El archivo Excel contiene un campo o columna Periodo.
2. Se consulta la base de datos para verificar si ya existe una carga previa para el mismo Periodo.
3. Reglas de negocio:
 - Si existe una carga previa con estado **Cargado, Finalizado o Notificado**, la carga debe ser **rechazada**.
 - Si existe una carga previa **Pendiente o En proceso**, la carga debe ser **bloqueada**, evitando cargas simultáneas para el mismo periodo.
 - Solo si el periodo no tiene cargas activas o finalizadas, el sistema registra una nueva carga con estado **Pendiente** y continúa el proceso.
En el caso de que no se cumpla con alguna validación se debe finalizar el proceso y almacenar los fallidos en una tabla de auditoria y trazabilidad

Lógica de negocio incluida — Validación de duplicidad de registros

1. El archivo Excel contiene un campo o columna Código Producto.
2. Se consulta la base de datos para verificar si ya existe un registro con el mismo Código.
3. Reglas de negocio:
 - Si existe un elemento con el mismo Código no se debe registrar y se debe reportar como **Existente**.

En el caso de que no se cumpla con alguna validación se deben almacenar los fallidos en una tabla de auditoria y trazabilidad

Esta validación permite asegurar consistencia de datos y evita duplicidades funcionales.

- Consume el mensaje.
- Actualiza el estado → **En proceso**.
- Descarga el archivo.
- Procesa todas las filas del Excel.
 - En el caso de que una columna esté vacía debe guardar un valor por defecto
 - Si hay filas vacías en el archivo no se deben registrar
- Inserta datos en PostgreSQL o SQL Server.
- Estado → **Finalizado**.
- Publica mensaje de notificación:

```
{  
  "idCarga": 123,  
  "usuario": "user@example.com",  
  "fechaFin": "2025-02-10T10:20:00"  
}
```

4 Microservicio de Notificaciones

- Lee la notificación.
 - Envía correo con MailKit.
 - Actualiza el estado final → **Notificado**.
-

5 Cliente Web

- Muestra el historial de cargas.
 - Permite ver estados en tiempo real (mediante pooling).
-

4. Requerimientos Técnicos Obligatorios

Backend – Todos los microservicios

- Lenguaje: **NET 8 o NET 9**
 - **Arquitectura limpia**
 - **CQRS o Inversión de dependencias**
 - **SOLID**
 - **JWT** (Refresh token opcional pero valorado)
 - **Manejo de excepciones global**
 - **Logging estructurado**
 - Dockerfile propio para cada microservicio (Opcional pero valorado)
 - docker-compose general orquestando (Opcional pero valorado):
 - todos los microservicios
 - Servicio de Colas (RabbitMQ/Kafka/otro)
 - seaweedfs
 - postgres o sqlserver
 - gateway
 - **Implementar Patrón Rate Limiting**
 - Uso de Dapper ó EntityFramework
 - Implementar Patrón Circuit Breaker (Opcional pero valorado)
 - Implementar Patrones de Reintentos (Opcional pero valorado):
-

Frontend (Opcional)

- React 16+
- Uso de componentes
- Pantallas requeridas:

0. Login

1. Subida de Excel

2. Historial de cargas (tabla)

3. Detalle del estado de una carga

Base de datos

- Debe incluir migraciones automáticas.
 - Uso de procedimientos almacenados
 - SqlServer o PostgreSQL
-

Mensajería

Servicio de Colas (RabbitMQ/Kafka/otro):

- Intercambio directo o topic
 - Mínimo 2 colas
-

Almacenamiento

SeaweedFS: - Servicio dockerizado - Endpoint para subir archivos (Abierto a usar otra herramienta)

Correo

- Usar MailKit
 - Configurable por variables de entorno
-
-



5. Estructura sugerida de la base de datos

- Para la construcción del modelo de datos se debe utilizar el criterio propio del candidato, se dejan scripts y nombres de referencia, sin embargo, se deben

contemplar los casos de uso y las reglas de negocio para construir la base de datos.

Script referencial

```
CREATE TABLE CargaArchivo (
    Id SERIAL PRIMARY KEY,
    NombreArchivo VARCHAR(200),
    Usuario VARCHAR(150),
    FechaRegistro TIMESTAMP,
    Estado VARCHAR(50),
    FechaFin TIMESTAMP NULL
);
```

6. Criterios de Evaluación

1. Arquitectura (25%)

- Microservicios independientes
- Limpieza del código
- Manejo de colas y estados

2. Funcionalidad (35%)

- Flujo completo operativo
- Procesamiento real del Excel
- Persistencia correcta

3. Docker / DevOps (20%)

- docker-compose funcional
- Servicios se levantan sin errores (opcional)

4. Frontend o Colecciones POSTMAN (20%)

Opción 1

- Colecciones Postman de cada endpoint **Opción 2**
 - Interfaz limpia y funcional
 - Manejo correcto de estados
 - UX básica pero consistente
-

7. Entrega final

El reto técnico debe ser completado en un plazo maximo de 48 horas.

El postulante debe entregar un repositorio con:

- ✓ Código fuente completo en repositorio GITHUB
 - ✓ Documentación en README
 - ✓ Instrucciones de despliegue (opcional pero valorado)
 - ✓ Scripts de base de datos
 - ✓ Postman collection (opcional en caso no se realice frontend)
 - ✓ Video corto (máximo 5 minutos) mostrando flujo completo funcionando
-

8. Resultado esperado

Una solución funcional, modular, distribuida, escalable y construida con estándares SENIOR.

Este reto está diseñado para verificar tu dominio práctico sobre: - microservicios - colas - .NET - React - procesamiento masivo - asincronía - docker - arquitectura limpia

 **MUCHA SUERTE**