- Servicio integracion Consulta y Recaudo *.

Este documento detalla la especificación técnica para la integración con el servicio de consulta de facturas .

```
1. Información General
```

Nombre del Servicio: `ConsultaFacturaServlet` **Endpoint:** `/factura/consulta` **Método:** `POST` **Content-Type:** `application/json; charset=UTF-8` **Descripción:** Verifica la existencia, integridad y estado de una factura (vencimiento, pagos previos, anulación) antes de permitir un recaudo.

2. Estructura de la Petición (Request)

El servicio espera un cuerpo JSON con los siguientes campos obligatorios:

Campo	Tipo	Requerido	Descripción
:————	:—–	:——:	:———————————–
`usuario`	String	Sí	Usuario de servicios web autorizado.
`clave`	String	Sí	Contraseña del usuario.
`consecutivo`	String	Sí	Número de factura (Debe ser un número mayor a 0).
`valor`	String	Sí	Valor de la factura (Debe ser un número mayor a 0).

```
 Ejemplo de Request JSON:
```

{

```
"usuario": "ws_recaudo",
"clave": "pass123",
"consecutivo": "998877",
"valor": "250000"
```

}

3. Estructura de la Respuesta (Response)

El servicio retorna un objeto JSON con el resultado de la validación y codificación `UTF-8`.

3.1. Éxito (Código 200)

{

```
"codigo": 200,
"mensaje": "La factura existe y se encuentra disponible para pago."
```

}

3.2. Errores de Cliente (Códigos 4xx)

Código	Mensaje	Razón
:—	:—	:—
400	“La petición no contiene información.”	Falta datos en el json.

415	“El JSON recibido no es válido.”	Error de sintaxis en el JSON enviado.
400	“No se han especificado…”	Faltan campos obligatorios en el JSON.
412	“El campo 'consecutivo/valor' debe ser mayor que cero.”	Validación de negocio: los números deben ser positivos.
401	“El usuario no esta autorizado…”	Credenciales inválidas o estado inactivo.
404	“No se encontró una factura asociada…”	Factura inexistente o el valor no coincide con la base de datos.
409	“La factura ya fue pagada…”	Estado de factura 'Pagado' en el sistema.
410	“La factura fue anulada…”	Estado de factura 'Anulado' en el sistema.
422	“La factura se encuentra vencida”	La fecha de vencimiento es anterior a la fecha actual.

3.3. Errores de Servidor (Código 500)

{

```
"codigo": 500,
"mensaje": "Error interno."
```

}

4. Lógica de Negocio y Validaciones Actualizadas

1. **Validación Numérica:** Se han incorporado reglas estrictas para que `consecutivo` y `valor` sean cadenas que representen números mayores a cero. 2. **Manejo de Fechas:** La validación de vencimiento ahora normaliza la fecha actual (sin horas/minutos) para comparar únicamente por día contra la `fecha_vencimiento`. 3. **Seguridad:** Sigue utilizando el esquema de validación contra `ADMINISTRACION.usuario_servicios` mediante MD5. 4. **Control de Concurrencia:** Utiliza `ReentrantLock` para asegurar que las consultas sobre una misma factura se serialicen correctamente si es necesario.

5. Reglas de Negocio Aplicadas

1. **Seguridad:** Validación de acceso idéntica al proceso de recaudo (MD5). 2. **Validación de Valor:** El parámetro `valor` enviado debe coincidir exactamente con el valor almacenado en la vista `V_FACTURA_ACTIVA_RECAUDO`. 3. **Estados Restrictivos:**

- Si el estado es el definido en `Estados.getPagado()`, se bloquea el proceso.
- Si el estado es el definido en `Estados.getAnulado()`, se bloquea el proceso.

4. **Validación de Fecha:** Se comprueba que la factura no haya superado su fecha de vencimiento (`fecha_vencimiento`).

* Servicio Recaudo.

Este documento detalla la especificación técnica para la integración con el servicio de recaudo de facturas.

1. Información General

* **Nombre del Servicio:** `RecaudoFacturaServlet` * **Endpoint:** `/factura/recaudo` * **Método:** `POST`

* **Content-Type:** `application/json; charset=UTF-8` * **Descripción:** Servicio encargado de validar, procesar y generar comprobante y notificación de pago para una factura específica en el sistema.

2. Estructura de la Petición (Request)

El servicio espera un objeto JSON con los siguientes atributos:

Campo	Tipo	Requerido	Descripción
:————	:—–	:——-:	:———————————–
`usuario`	String	Sí	Usuario de servicios web autorizado.
`clave`	String	Sí	Contraseña del usuario (Validada vía MD5 en DB).
`consecutivo`	String	Sí	Número de factura a recaudar.
`valor`	String	Sí	Valor total del recaudo (sin decimales).
`banco`	String	Sí	Código del banco que procesa la transacción.

Ejemplo de Request:

{

```
"usuario": "ws_recaudo_dav",
"clave": "P@ssw0rd2024",
"consecutivo": "85562",
"valor": "150000",
"codbanco": "012"
```

}

3. Estructura de la Respuesta (Response)

El servicio siempre retorna un objeto JSON con un código de estado interno y un mensaje descriptivo.

3.1. Éxito (Código 201)

{

```
"codigo": 201,
"mensaje": "Recaudo exitoso.",
"comprobante": 1234567
```

}

```
3.2. Errores Comunes
```

| Código | Mensaje | Razón |

:—	:—	:—
415	“El JSON recibido no es válido.”	Error de sintaxis en el JSON enviado.
412	“El campo 'consecutivo/valor' debe ser mayor que cero.”	
400	“La petición no contiene información.”	Falta datos en el json.
401	“El usuario no esta autorizado…”	Credenciales inválidas o estado inactivo.

404	“No se encontró un banco…”	El código de banco no existe en el maestro.
404	“No se encontró una factura…”	Factura inexistente o el valor no coincide.
409	“La factura ya fue pagada…”	Factura en estado 'C' (Cancelada/Pagada).
410	“La factura fue anulada…”	Factura en estado 'N' (Anulada).
422	“La factura se encuentra vencida”	La fecha actual es superior a la de vencimiento.
500	“Error interno.”	Error inesperado en el servidor o base de datos.

4. Flujo de Validaciones y Reglas de Negocio

1. **Autenticación:** Validación contra la tabla `ADMINISTRACION.usuario_servicios` con cifrado MD5.
2. **Control de Concurrencia:** Se implementa un bloqueo por hilo (`ReentrantLock`) basado en el `consecutivo` y `valor`. Esto garantiza que una factura no sea procesada dos veces si se reciben peticiones simultáneas. 3. **Integridad de Datos:**

- Solo se permiten pagos por el **valor exacto** de la factura.
- La factura debe existir en la vista `RECAUDO.V_FACTURA_ACTIVA_RECAUDO`.

4. **Generación de Comprobante:**

- Se crea un registro en `ComprobanteDeIngreso`.
- Se realiza el `commit` de la transacción.
- Se ejecuta el proceso de **Aprobación Automática** del comprobante.
- Se guarda en un log el proceso de recaudo , tabla recaudo.log_pago_factura

5. Especificaciones Técnicas (Stack)

* **Java Versión:** 6 (JDK 1.6). * **Librerías:** `org.json` para el manejo de JSON. * **Persistencia:** JDBC sobre conexión gestionada por Hibernate (`SessionFactorySitu`). * **Logs:** Implementado vía `com.ada.utilidades.Log`.

> **Nota para Integradores:**
> Si el servicio retorna un error de red o un código 500, se recomienda realizar una consulta previa del estado de la factura antes de reintentar el pago para evitar duplicidades, a pesar de los controles de concurrencia del lado del servidor.

Métodos: `GET`, `POST`

Content-Type: `application/json;charset=UTF-8`

Parámetros de Entrada (Obligatorios)

Seguridad - **usuario** (string): Usuario registrado en la tabla `ADMINISTRACION.usuario_servicios` con estado activo. - **clave** (string): Contraseña del usuario (se valida con hash MD5 en base de datos).

Datos de Consulta - **consecutivo** (string): Identificador de la factura a consultar. - **valor** (string): Valor de la factura que debe coincidir exactamente con el registrado en BD. Usado para validar la factura y confirmar integridad de datos.

> La fecha de vencimiento se valida internamente contra la base de datos para asegurar que la factura no esté vencida. No debe enviarse como parámetro de la solicitud.