**Decisiones tomadas**

**Patrones utilizados**

Se decide utilizar un estilo orientado a servicios, que consta de los servicios de:

* Presentación: UI para la consulta de las facturas
* Router Service: Servicio para el enrutamiento de la factura de acuerdo con el número de convenio
* Registry Service: Servicio para la ejecución de los pagos de acuerdo con el número de factura y el convenio al que pertenece la misma

Así mismo, se decide usar el patrón de diseño DISPATCHER, para, de acuerdo con los códigos de los convenios, direccionar la consulta al servicio deseado.

Se toma la decisión de utilizar un patrón Object Mapper para realizar el mapeo de información de la aplicación a la información de los servicios. Para este mapeo, se toma la decisión que el mapeo se hará a servicios tipo REST o tipo SOAP, para lo que se deberá almacenar en cada caso la información requerida para la invocación del servicio.

Para el mapeo de los servicios SOAP, se toma la decisión de utilizar esquemas XSLT para el armado del ENVELOPE de la petición. Cada convenio debe tener asociado un esquema XSLT para el armado del ENVELOPE, así como la URL del ENDPOINT, la acción (SOAPAction) a invocar y el dato de la respuesta que se desea extraer de la misma.

Para el mapeo de los servicios RESTful, se toma la decisión de realizar tal mapeo, mediante cadenas de caracteres con la siguiente estructura:

campo1Servicio:campo1ObjetoPago,campoNServicio:campoNObjetoPago

Esto significa que para cada mapeo que se haga se debe almacenar una cadena de caracteres con la equivalencia de campos entre el servicio y los campos en el objeto Pago. El objeto pago tiene la siguiente estructura (definida en contrato RAML):

Pago:

properties:

numeroFactura: number

numeroConvenio: number

numeroIdCliente: string

tipoIdCliente: string

totalPago: number

example:

numeroFactura: 12345678790

numeroConvenio: 1234

numeroIdCliente: "123454544"

tipoIdCliente: "CC"

totalPago: 45000

Los mapeos que se pueden hacer son:

* **Path params y query params:** son los mapeos que se hacen a datos en la URL del servicio, por lo que se toma la decisión que cualquier parámetro en la URL del servicio debe colocarse en ella, en el siguiente formato:

**{<nombre\_parámetro>}**

* **Payload:** este mapeo hace referencia a objetos que se deban enviar en la petición como payload.
* **Headers:** Este mapeo hace referencia a la posibilidad de asignar los headers de la petición, de la siguiente forma:

**<nombre del header>:<Valor del header>**

Por ejemplo Accept:application/json

Se toma, así mismo, la decisión de que en el formato de los mapeos no debe haber espacios después de los dos puntos (“:”).

Se toma la decisión de programar los servicios en JAVA, por la experiencia con el lenguaje.

**Estilo de servicios**

Para la implementación de los servicios se decide hacer uso de la tecnología RESTful, para lo cual se define el siguiente contrato RAML

#%RAML 1.0

title: Bank123API

version: 1.0

baseUri: api/resources

types:

RESTData:

properties:

metodo: string

funcion: string

recurso: string

pathParamMapping: string

paramsMapping: string

headers: string

payloadMapping: string

example:

metodo: "GET"

funcion: "CONSULTAR"

recurso: "payment/{idFactura}"

pathParamMapping: "idFactura:numeroFactura"

paramsMapping: ""

payloadMapping: ""

headers: "Accept:application/json"

SOAPData:

properties:

soapAction: string

xsltDefinition: string

funcion: string

responseElement: string

example:

soapAction: "consultar"

xsltDefinition: "…"

funcion: "CONSULTAR"

responseElement: "totalPagar"

Convenio:

properties:

id: number

idConvenio: number

nombre: string

tipoServicio: string

host: string

puerto: number

urlServicio: string

tipo: string

soapData: SOAPData[]

restData: RESTData[]

example:

id: 1

idConvenio: 1234

nombre: "Claro"

tipoServicio: "Celular"

host: "localhost"

puerto: 8080

urlServicio: "claro/payments/service"

tipo: "REST"

soapData: []

restData: []

Pago:

properties:

numeroFactura: number

numeroConvenio: number

numeroIdCliente: string

tipoIdCliente: string

totalPago: number

example:

numeroFactura: 12345678790

numeroConvenio: 1234

numeroIdCliente: "123454544"

tipoIdCliente: "CC"

totalPago: 45000

/convenio:

get:

description: Permite devlover todos los convenios registrados

responses:

200:

body:

application/json:

type: Convenio[]

post:

description: Permite crear un convenio a partir de el objeto dado

body:

application/json:

type: Convenio

/{id}:

delete:

description: Permite borrar un convenio registrado

responses:

200:

description: Borra un convenio

put:

description: Permite modificar la información almacenada de un convenio

body:

application/json:

type: Convenio

get:

description: Permite devolver el convenio asociado a un número de convenio dado

responses:

200:

body:

application/json:

type: Convenio

post:

description: Permite invocar el servicio asociado a un convenio

body:

application/json:

/pago:

post:

description: Permite efectuar un pago

body:

application/json:

type: Pago