

Manual Técnico

Tema: Servicio Reportes

Tabla de contenido

[Introducción 3](#_Toc473414208)

[Operaciones del servicio 5](#_Toc473414209)

[deploy 5](#_Toc473414210)

[download 6](#_Toc473414211)

[view 7](#_Toc473414212)

[Cliente JAVA 8](#_Toc473414213)

[deployReport 8](#_Toc473414214)

[download 9](#_Toc473414215)

Introducción

El servicio de reportes es el encargado de almacenar todos los reportes que sean solicitados por el banco y de permitir generarlos en varios formatos de acuerdo a la necesidad del desarrollador que lo utilice.

El mismo ha sido desarrollado para dar soporte a la nueva plataforma de procesos de bonita 7 y se encuentra desplegado en el AS del Carbon.

Arquitectura

Este servicio está desarrollado bajo una arquitectura de 3 capas la cual está soportada por los frameworks Hibernate, Spring y Jackson para la conversiones a JSON, el objetivo es exponer un recurso REST el cual pueda ser consumido desde JavaScript (formularios de bonita), Java (conectores de procesos) y aplicaciones nativas de celulares.

Para consumir el servicio desde JavaScript se deben utilizar llamados de tipo XMLHttpRequest y para consumirlo desde Java se debe utilizar el cliente desarrollado por Trycore para tal fin el cual detallaremos más adelante.

A continuación el diagrama de arquitectura:



Operaciones del servicio

En esta sección describiremos todas las funcionalidades que posee el servicio así como la lista de parámetros y retorno que genera cada una, cabe destacar que este servicio está diseñado para trabajar con mensajes JSON por lo que el cuerpo de las solicitudes esperadas y de las respuestas que se retornen tendrán este tipo de contenido.

Las operaciones tienen diferentes tipos de parámetros de acuerdo el tipo de solicitud (GET; POST, PUT, DELETE), algunos parámetros son parte de la URL y están rodeados por llaves los cuales llamaremos PathVariable, luego están los que forman parte del QueryString, los que van luego del signo de interrogación, estos los llamaremos RequestParam para los llamados GET y finalmente están los parámetros en viajan en el body de la solicitud, también los llamaremos RequestParam pero son utilizados para los llamados POST y PUT.

## deploy

Permite desplegar un nuevo reporte en la plataforma

**Método**: POST

**Ruta de acceso**: /deploy

**Body:** form-data

**Parámetros**:

|  |  |  |  |
| --- | --- | --- | --- |
| **Nombre** | **Descripción** | **Tipo** | **Obligatorio** |
| name | Nombre del reporte | String | Si |
| file | JRML del reporte | File | Si |

**Respuesta**: void

## 

## download

Permite descargar un reporte previamente desplegado

**Método**: GET

**Ruta de acceso**: /download?r=…&f=…&u=…&p=…

**Parámetros**:

|  |  |  |  |
| --- | --- | --- | --- |
| **Nombre** | **Descripción** | **Tipo** | **Obligatorio** |
| r | Nombre del reporte | String | Si |
| f | Formato del reporte | String (Posibles valores: TEXT, CSV, HTML, PDF, XLS, XLSX, DOCX, PPTX) | Si |
| u | Usuario que genera el reporte | String | Si |
| p | Lista de parámetros | List<String> | No |

*El parámetro p corresponde a los parámetros del reporte y debe setearse de igual forma para todos los parámetros de la siguiente manera:*

p=NOMBRE\_PARAMETRO=VALOR

**Respuesta**: Descarga del archivo asociado a la generación.

## view

Permite visualizar un reporte online siempre y cuando el formato seleccionado sea compatible con el browser.

**Método:** GET

**Ruta de acceso:**  /view?r=…&f=…&u=…&p=…

**Parámetros:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nombre** | **Descripción** | **Tipo** | **Obligatorio** |
| r | Nombre del reporte | String | Si |
| f | Formato del reporte | String (Posibles valores: TEXT, CSV, HTML, PDF, XLS, XLSX, DOCX, PPTX) | Si |
| u | Usuario que genera el reporte | String | Si |
| p | Lista de parámetros | List<String> | No |

*El parámetro p corresponde a los parámetros del reporte y debe setearse de igual forma para todos los parámetros de la siguiente manera:*

p=NOMBRE\_PARAMETRO=VALOR

**Respuesta**: Genera la visualización para que sea mostrada en el explorador (ej: HTML, Texto, CSV, PDF). Si el formato no es compatible con el browser el archivo se descarga.

Cliente JAVA

Dado que los servicios son de tipo REST con mensajes JSON se creó un cliente JAVA para consumir los servicios a través HTTPClient.

Para poder utilizar el widget se debe importar la librería JAR llamada **report-client.jar** en el studio de bonita o en el proyecto java que se desee, una vez en el classpath se podrá utilizar la clase llamada **ReportClient** la cual contiene las operaciones que permiten invocar el servicio. A continuación detallaremos las operaciones existentes y los parámetros que espera cada una.

## deployReport

Esta operación invoca la operación deploy del servicio detallada anteriormente recibe los siguientes parámetros:

|  |  |  |  |
| --- | --- | --- | --- |
| **Nombre** | **Descripción** | **Tipo** | **Obligatorio** |
| serviceUri | URI del servicio, se debe indicar protocolo, dns, puerto y contexto (por defecto es report-svc) | String | Si |
| reportName | Nombre del reporte | String | Si |
| jrxmlFile | Archivo JRXML del reporte | String | Si |

**Respuesta:** Sin respuesta

## download

Operación utilizada para invocar la descarga de un reporte al servicio la misma espera los siguientes parámetros:

|  |  |  |  |
| --- | --- | --- | --- |
| **Nombre** | **Descripción** | **Tipo** | **Obligatorio** |
| serviceUri | URI del servicio, se debe indicar protocolo, dns, puerto y contexto (por defecto es report-svc) | String | Si |
| reportName | Nombre del reporte | String | Si |
| format | Formato de generación y descarga | String (Posibles valores: TEXT, CSV, HTML, PDF, XLS, XLSX, DOCX, PPTX) | Si |
| user | Usuario que solicita la generación utilizado con fines de LOG | String | Si |
| parameters | Mapa de parámetros que espera el reporte | Map<String, String> | No (se debe enviar null en caso no se requiera) |