Bizativiti
Grupo 4
Descripción de la Arquitectura
Versión 1.6

Historia de revisiones

Fecha	Versión	Descripción	Autor
24/08/2013	1.0	Primer planteo de arquitectura	Juan Pablo Lorier
14/09/2013	1.1	Definición avanzada de la arquitectura	Juan Pablo Lorier
15/09/2013	1.1	Revisión SQA	Valeria Rocha
26/09/2013	1.2	Definición de la arquitectura	Juan Pablo Lorier
29/09/2013	1.2	Revisión SQA	Valeria Rocha
05/10/2013	1.3	Corrección urls web	Juan Picca
06/10/2013	1.4	Ajustes derivados de SQA	Juan Pablo Lorier
13/10/2013	1.4	Revisión SQA	Valeria Rocha
03/11/2013	1.5	Actualización urls web	Juan Picca
03/11/2013	1.5	Revisión SQA	Valeria Rocha
24/11/2013	1.6	Versión final	Juan Pablo Lorier
24/11/2013	1.6	Revisión SQA	Valeria Rocha

Índice

Introducción

Se describen los aspectos más sobresalientes de la arquitectura propuesta para el producto y se incluyen las razones para la elección de la misma.

Propósito

Se busca informar sobre los aspectos más destacados del proceso de elección de la arquitectura permitiendo tener una visión global de la misma así como un detalles de los aspectos más destacados.

Este documento está orientado a quienes pretendan comprender las motivaciones que condujeron a la arquitectura del sistema.

Alcance

En este documento se detallan aspectos referentes a la arquitectura y se describen los casos de uso principales que influyen en el diseño de la arquitectura propuesta.

Definiciones, siglas y abreviaturas.

Ver Glosario

Referencias

- Glosario versión 1.1.
- Especificación de requerimientos versión 3.

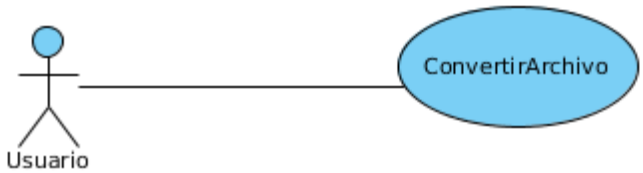
Visión general

La arquitectura que se adopta es la de plugins para permitir la extensión de las funcionalidades en forma sencilla en un diseño de aplicación web.

También se adopta la arquitectura de API REST para ofrecer los servicios mediante una interfaz web al usuario.

Vista del Modelo de Casos de Uso

Diagrama de Casos de Uso relevantes a la Arquitectura



Casos de Uso relevantes a la Arquitectura

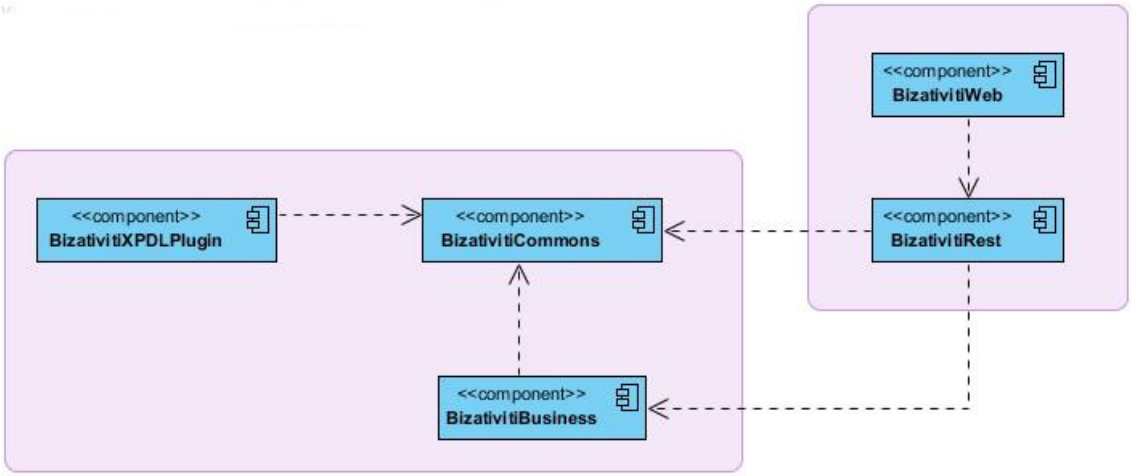
Convertir Archivo a BPMN 2.0

El caso de uso comienza cuando el usuario selecciona el archivo a convertir mediante una ventana de elección de archivo. Luego elige de una lista desplegable el tipo de archivo de entrada correspondiente de entre los tipos disponibles y con una ventana de selección de archivo elige la ruta y nombre del archivo XML de salida.

El usuario cuenta con un listado que le permite seleccionar los tipos de elementos del modelo que se van a tomar en cuenta en el momento de la conversión. Además, cuenta con un área de configuración que le permite introducir los valores por defecto que tomarán los atributos que no posean valor en el elemento original. Finalmente, presiona el botón de convertir y el sistema traduce el archivo de entrada a BPMN 2.0 y crea en la ruta de destino, el archivo XML de salida.

Trazabilidad desde el Modelo de Casos de Uso al Modelo de Diseño

Convertir Archivo a BPMN 2.0



Vista del Modelo de Diseño

Descomposición en Subsistemas

El sistema no presenta subsistemas

Diseño de Clases

BizativitiBusiness.Converter

La clase contiene un procedimiento eval que tiene como parámetro un objeto y su función es evaluar de que tipo es el objeto que se le pasa. Cuando determina el tipo del objeto, invoca un translator que es el encargado de hacer la conversión del objeto del metamodelo a uno de BPMN.

BizativitiXPDLPlugin.Converter

A modo de ejemplo, el plugin XPDL al igual que todos los plugins tiene una clase converter que en forma análoga al converter de BizativitiBusiness, detecta el tipo de objeto del estándar correspondiente al plugin que se le pasó como parámetro e invoca al translator correspondiente para que haga la conversión a un objeto del metamodelo.

Translator

Es una clase abstracta que incluye como agregación a todos los translators de cada tipo de elemento que se va a convertir. Los translators forman parte tanto del módulo BizativitiBusiness como de cada plugin y son los que en cada caso convierten ese tipo de elemento específico en su homólogo ya sea en el metamodelo (caso del translator del plugin) o de BPMN (caso del translator del business)

IPlugin

Es una interfaz que permite el uso de plugins para aportar la capacidad de poder entender un estándar de modelado BPM. Le brinda al sistema la habilidad de extenderse para poder importar otros estándares.

Metamodel

Es una colección de clases que nos brinda una representación intermedia de los objetos parseados de el standard de entrada elegido para que sean luego convertidos en BPMN.

Diseño de Casos de Uso

Diseño del caso de uso Convertir archivo

Todas las clases del diseño intervienen en el caso de uso. Este es el único caso de uso del sistema.

Trazabilidad desde el Modelo de Diseño al Modelo de Implementación

Se establece una relación entre el módulo BizativitiCommons y las clases definidas en el Metamodelo para representar los elementos del diagrama en conversión.

Se integran en este módulo también las clases encargadas del manejo de plugins, principalmente la clase IPlugin.

Para los módulos de BizactivitiBussiness y BizativitiXPDLplugin la correspondencia es con los módulos BizativitiBussiness y BizativitiXPLParser respectivamente.

Para el caso de BizativitiREST y BizativitiWeb la correspondencia es directa.

Vista del Modelo de Implementación

Componentes

BizativitiCommons

Este módulo contiene los objetos que van a ser utilizados por el resto

de los módulos. La idea es que estos objetos carecen de lógica. En particular en este módulo se encuentran los objetos del modelo intermedio o metamodelo y una interfaz que es la que deben implementar los distintos plugins.

BizativitiXPDLParser

Como ejemplo de plugin, este módulo es el que se encarga de leer el archivo de entrada y transformarlo a objetos del metamodelo. Implementa la interfaz de IPlugin que tiene el módulo BizativitiCommons y le aporta al sistema la posibilidad de convertir desde el standard XPDL de Bizagi.

BizativitiBusiness

Este módulo recibe una llamada para traducir un archivo y es capaz de invocar al plugin correspondiente, obtener un conjunto de objetos, pasarlos al metamodelo, y luego transformarlo en un XML que cumpla con el estándar BPMN 2.0 y sea soportado por Activiti.

BizativitiRest

Este componente se encarga de exponer una API REST sobre HTTP para poder brindar conversiones de los diferentes formatos soportados a BPMN 2.0. Éste módulo va a hablar con BizativitiBusiness, mediante una API bien definida para poder ejecutar los pedidos de conversión y obtener las respuestas en el formato final.

Se definen en este componente los siguientes URLs:

- http://<host>/bizativiti/upload
    - **Request**
        - **método HTTP:** POST
        - **payload:** file=<archivo en el formato "plugin">
        - **params:** type=<nombre del plugin a utilizar>
        - **opcionales:**

            Parámetros para la conversión selectiva de elementos (por defecto no se convierten):

            - **Task**=on|off|off
            - **UserTask**=on|off|off
            - **ManualTask**=on|off|off
            - **ScriptTask**=on|off|off
            - **BusinessTask**=on|off|off
            - **SendTask**=on|off
            - **StartEvent**=on|off
            - **TimeStartEvent**=on|off
            - **MessageStartEvent**=on|off
            - **EndEvent**=on|off
            - **TerminateEvent**=on|off
            - **EventoIntermedioSimple**=on|off
            - **TimerEvent**=on|off

- **MessageCatchEvent**=on|off
- **SignalCatchEvent**=on|off
- **SignalThrowEvent**=on|off
- **Association**=on|off
- **SequenceFlow**=on|off
- **Parallel**=on|off
- **Inclusive**=on|off
- **EventBased**=on|off
- **Exclusive**=on|off

Parámetros para agregar valores por defecto a los atributos extendidos:

- **nombre1**= <lista separada por coma de duplas valor,tipo con tipo en {string,long,date}>

    - **Response**
        - JSON con
            - **status:** ok/error
            - **ticket_id:** Entero con un número para chequear el estado del procesamiento (si status es ok)
            - **message:** Texto con mensaje de error (si status es error)
- http://<host>/bizativiti/status/<ticket_id>.json
    - **Request**
        - **método HTTP:** GET
        - **opcionales:** otros atributos como query string o en los headers
        - **parámetros de la URL:**
            - ticket_id: Entero devuelto por el endpoint anterior.
    - **Response**
        - JSON con
            - **status:** ok/failed
            - **ticket_id:** Entero con un número para chequear el estado del procesamiento.
            - **processing_status:** started/in progress/finished/error
- http://<host>/bizativiti/file/<ticket_id>.json
    - **Request**
        - **método HTTP:** GET
        - **opcionales:** otros atributos como query string o en los headers
        - **parámetros de la URL:**
            - ticket_id: Entero devuelto por el endpoint anterior.
    - **Response**
        - Archivo XML
- http://<host>/bizativiti/log/<ticket_id>.json
    - **Request**
        - **método HTTP:** GET

        - **opcionales:** otros atributos como query string o en los headers
        - **parámetros de la URL:**
            - ticket_id: Entero devuelto por el endpoint anterior.
    - **Response**
        - Archivo de log de la conversión.
- http://<host>/bizativiti/plugins
    - **Request**
        - **método HTTP:** GET
    - **Response**
        - Lista de plugins soportados por el servidor

BizativitiWeb

Este proyecto incluye la interfaz web que permite seleccionar un archivo desde el sistema de archivos traducir al formato de salida estándar. Se va a comunicar siempre con el BizativitRest consumiendo los servicios que los end-point provean.

Interfaces

Interfaz web de usuario

El sistema cuenta con una sencilla interfaz web que le permite al usuario conectarse a través de un navegador web al servidor de aplicaciones donde se le presenta una pantalla que permite que ingrese la ruta del archivo de entrada, elija el formato en el que éste se encuentra y pueda elegir que elementos del modelo se van a convertir así como agregar valores por defecto a atributos extendidos de objetos.

Interfaz REST

Esta interfaz le permite a otras aplicaciones consumir los servicios del sistema incluyendo a la interfase web que es el principal consumidor de esta interfaz.

Requerimientos no funcionales

Se pretende especificar los requerimientos de memoria RAM para poder ejecutar el sistema en las condiciones descriptas por el alcance.

El cliente no tiene limitaciones en cuanto a la cantidad de memoria RAM disponible, pero se estima que para procesar un modelo con el máximo de componentes se requiere 10 MB apróximadamente de memoria RAM.