Índice general

1. Introducción **1**
1.1. Introducción 1
1.2. LUCA 2

2. Análisis y documentación **7**
2.1. GO.JS 8
2.1.1. ¿Qué es GO.JS? 8
2.1.2. ¿Porqué GO.JS? 8
2.1.3. Características 8
2.1.4. Explicación 8
2.2. Vaadin 9
2.2.1. ¿Qué es GO.JS? 9
2.2.2. Características 9
2.2.3. Aplicación 9

3. Process-Component **11**
3.1. Introducción 11

4. Luca-Process **13**
4.1. Introducción 13

5. Conclusiones **15**
5.1. Introducción 15

Índice de figuras

1.1. Vista de Gestión de Consultas 3
1.2. Vista de Ejecución de Consultas 4

2.1. GO.JS Logo 8
2.2. Vaadin Logo 9
2.3. Esquema Cliente-Servidor 10

Índice de cuadros

Agradecimientos

Me gustaría dar agradecimientos a mi familia y facultad, ya que sin ellos esto no habría sido posible nada de estos.

Es importante agradecer también a CIC Consulting Informático por permitirme la oportunidad de realizar el desarrollo del proyecto en su empresa, sin olvidarme de mis compañeros de LUCA, que han sido un gran apoyo durante el mismo..

Para finalizar, me gustaría gradecer a mi mentor Pablo, por guiarme durante el desarrollo del proyecto con eficacia y ayudarme a afrontar este trabajo de fin de grado.

Resumen

Las empresas actuales no utilizan un único sistema de información que de soporte a sus procesos de trabajo, sino un ecosistema de sistemas información que dan soporte a diferentes procesos de negocio ejecutados dentro de dicha organización. Como consecuencia de esta nueva situación, cuando un usuario quiere obtener una información concreta cuyos datos residen en varios de estos sistemas, necesita acceder a cada uno de estos sistemas, extraer de cada sistema la información que precisa, filtrarla y unificarla para finalmente obtener los datos requeridos.

Por ejemplo, una tienda de electrodomésticos podría tener sistemas informáticos diferentes para el departamento de atención al cliente, para el departamento técnico de postventa y para el departamento de compras y adquisiciones.Por tanto, para conocer el estado actual de una reparación, podríamos necesitar:

- Acceder al primer sistema para obtener el identificador de la incidencia y en qué fase de su gestión se encuentra.
- Comprobado que la incidencia está actualmente en reparación, recuperaríamos otro sistema el estado detallado de la reparación, comprobando que está a la espera de una pieza.
- Finalmente accederíamos al sistema de compra y adquisiciones para comprobar cuando está prevista la entrega de dicha pieza. Los sistemas de almacenamiento de la información pueden ser diversos, incluyendo desde un servicio web, una base de datos relacional, un repositorio de ficheros accesible vía FTP o una base de datos NoSQL.

El objetivo de este proyecto es facilitar dicho proceso de composición al usuario mediante el desarrollo de un mecanismo gráfico para la especificación de estos procesos de composición de consultas.

Palabras clave:

Preface

Keywords:

Capítulo 1

Introducción

Este primer capítulo describe de forma general la información principal y relevante para poder comprender el funcionamiento del producto actual LUCA, así como los objetivos del proyecto a desarrollar junto con la explicación y motivación del mismo.

1.1. Introducción

En los últimos años, el volumen de datos que una empresa o entidad necesita o es capaz de gestionar o manipular ha aumentado de forma vertiginosa. Estos datos se almacenan en fuentes de diversos tipos, abarcando elementos tan dispares como bases de datos relacionales, hojas XML o repositorios FTP, a los cuales se accede mediante diferentes formas y lenguajes. Por tanto, un nuevo problema que debemos enfrentar, adicional al del volumen de datos a manipular, es que para acceder cierta información es necesario muchas veces establecer comunicaciones entre diferentes fuentes.

Como consecuencia de esta nueva situación, cuando un usuario quiere obtener una información concreta cuyos datos residen en varios de estos sistemas, necesita acceder a cada uno de estos sistemas, extraer de cada sistema la información que precisa, filtrarla y unificarla para finalmente obtener los datos requeridos.

Por ejemplo, una tienda de electrodomésticos podría tener sistemas informáticos diferentes para el departamento de atención al cliente, para el departamento técnico de postventa y para el departamento de compras y adquisiciones.Por tanto, para conocer el estado actual de una reparación, podríamos necesitar:

- Acceder al primer sistema para obtener el identificador de la incidencia y en qué fase de su gestión se encuentra.
- Comprobado que la incidencia está actualmente en reparación, recuperaríamos otro sistema el estado detallado de la reparación, comprobando que está a la espera de una pieza.
- Finalmente accederíamos al sistema de compra y adquisiciones para comprobar cuando está prevista la entrega de dicha pieza. Los sistemas de almacenamiento de la información pueden ser diversos, incluyendo desde un servicio web, una base de datos relacional, un repositorio de ficheros accesible vía FTP o una base de datos NoSQL.

1.2. LUCA

Con el objetivo de facilitar este proceso de recuperación de información almacenada en sistemas y fuentes de datos hetereogéneas, dentro de la empresa CIC, se está desarrollando una aplicación denominada LUCA. Para facilitar este proceso de recuperación de información, LUCA proporciona un lenguaje común para todas las fuentes de datos a unificar, permitiendo al usuario abstraerse de los detalles de cada fuente.

A continuación se explica brevemente el funcionamiento de Luca con algunos ejemplo gráficos orientativos para ayudar a la comprensión del propio funcionamiento.

Para explicar el funcionamiento principal de dicho producto, se pretende describir el proceso de creación y ejecución de una consulta a un servicio REST.

En la vista inicial nos encontramos con una lista de consultas ya almacenadas, así como, una serie de filtros y opciones para la creación, ejecución y edición de consultas. Como se pretende explicar la creación y ejecución de una consulta, para avanzar en la vista se seleccionaría la creación de una nueva consulta.

A continuación, se muestra un ejemplo sobre un servicio REST, concretamente sobre el servicio REST de información de la flota de autobuses de Santander (TUS), donde tras introducir el identificador de una parada de autobús, se devolverá la información relativa a dicha parada, como es la localización.

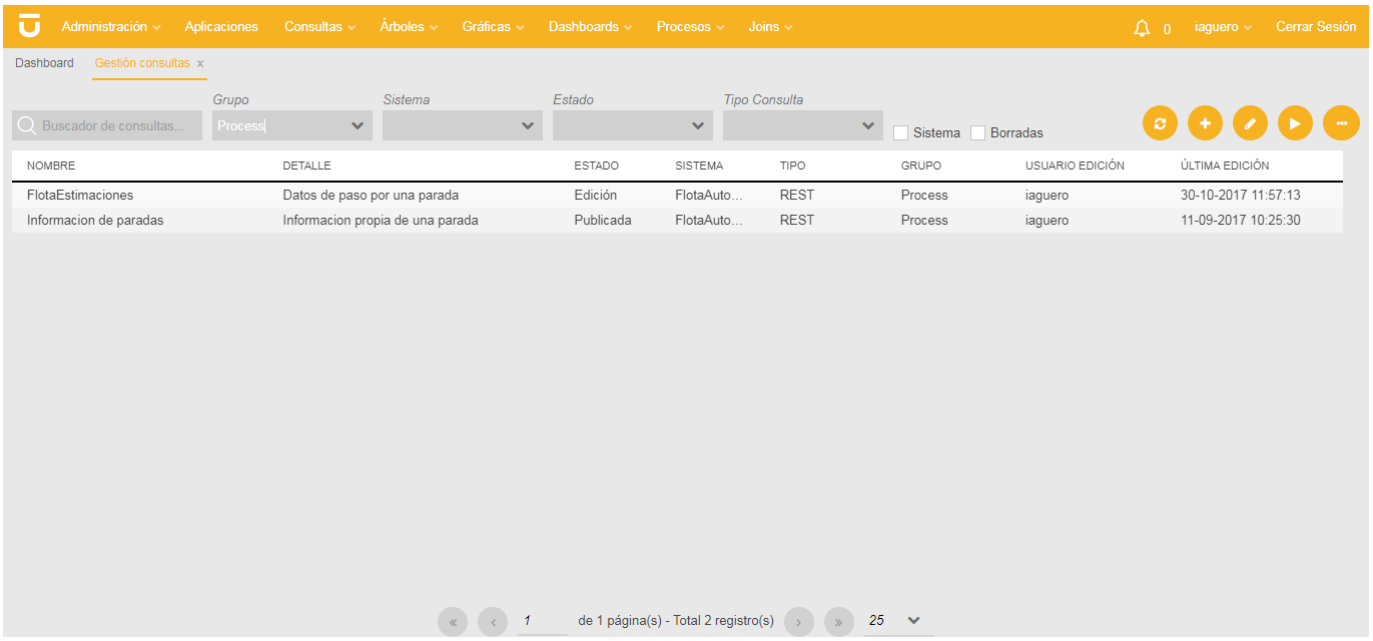

Figura 1.1: Vista de Gestión de Consultas

Tras seleccionar la opción de creación de una nueva consulta, se muestra una vista en la que se permite agregar toda la información relativa a la consulta, como son las variables de entrada o de salida, o el tipo de salida que se desea recibir.

Una vez introducido el identificador de la parada, y seleccionado el botón de ejecución para llevar a cabo la ejecución de la consulta, se muestra el resultado de la consulta, el cuál puede ser visualizado de diferentes formas en función del recurso al que se llama.

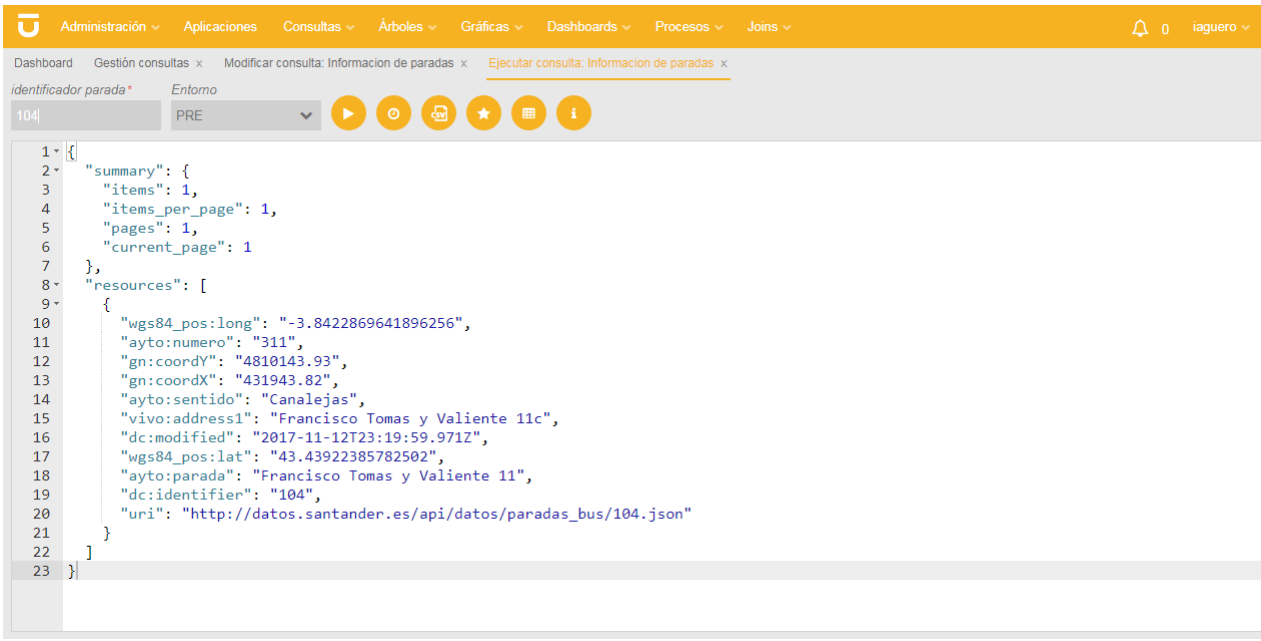

Figura 1.2: Vista de Ejecución de Consultas

Actualmente LUCA proporciona mecanismos para permitir al usuario recuperar de manera uniforme información de diferentes fuentes de datos. No obstante, LUCA actualmente sólo es capaz recuperar información de una única fuente de datos a la vez. Por tanto, cuando es necesario combinar información procedente de distintas fuentes, el propio usuario es el que debe realizar dicho proceso de composición, ejecutando cada consulta a mano, y utilizando las salidas de cada una de ellas como las entradas de las siguientes.

Un ejemplo de dicho proceso de composición sería la necesidad de un dependiente de una tienda de electrodomésticos de obtener la edad de los usuarios que compraron lavadoras durante el mes pasado. Actualmente, los pasos o consecución de consultas que debería de realizar serían las siguientes:

- Primero necesitaría obtener el registro de compras del mes pasado del sistema.
- Después, tras apuntarse dicho registro, tendría que, uno por uno, seleccionar los que se corresponden con lavadoras.
- Una vez que el usuario tiene las lavadoras compradas el mes pasado, éste tendría que recoger que usuarios han comprado las lavadoras.

- Por último, el usuario debería de buscar en el sistema cada usuario que ha realizado la compra, con el nombre obtenido previamente.

Se puede observar que el usuario tiene un engorroso desarrollo de acciones para poder obtener el resultado deseado.

El objetivo de este proyecto es facilitar dicho proceso de composición al usuario mediante el desarrollo de un mecanismo gráfico para la especificación de estos procesos de composición de consultas.

Para llevar a cabo esta tarea, se pretende realizar un sistema que permita visualmente utilizar las consultas ya almacenadas, para posteriormente relacionarlas entre sí mediante sus entradas y salidas y los tipos de las mismas. De esta forma, se podrá ejecutar automáticamente una cadena de consultas para obtener un resultado concreto, bajo un único concepto llamado Proceso.

Capítulo 2

Análisis y documentación

Este capítulo describe el proceso de adquisición de conocimientos necesarios para poder llevar a cabo el proceso de diseño arquitectónico y de construcción o implementacíon de la aplicación. De esta fomra se puede realizar una planificación mas cercana a la realidad y partir de unos conocimientos mínimos para empezar a elaborar el proyecto.

Contents

2.1. GO.JS 8

2.1.1. ¿Qué es GO.JS? 8

2.1.2. ¿Porqué GO.JS? 8

2.1.3. Características 8

2.1.4. Explicación 8

2.2. Vaadin 9

2.2.1. ¿Qué es GO.JS? 9

2.2.2. Características 9

2.2.3. Aplicación 9

2.1. GO.JS

Figura 2.1: GO.JS Logo

2.1.1. ¿Qué es GO.JS?

Go.JS [3] es una biblioteca de JavaScript con múltiples funciones para implementar diagramas interactivos personalizados y visualizaciones complejas en navegadores y plataformas web modernos.

2.1.2. ¿Porqué GO.JS?

GoJS facilita la construcción de diagramas de JavaScript de nodos, enlaces y grupos complejos con plantillas y diseños personalizables.

2.1.3. Características

GoJS ofrece muchas características avanzadas para la interactividad del usuario, tales como arrastrar y soltar, copiar y pegar, edición de texto en el lugar, información sobre herramientas, menús contextuales, diseños automáticos, plantillas, vinculación y modelos de datos, administración de estado transaccional y deshacer, paletas , vistas generales, controladores de eventos, comandos y un sistema de herramientas extensible para operaciones personalizadas.

2.1.4. Explicación

GoJS ofrece muchas características avanzadas para la interactividad del usuario, tales como arrastrar y soltar, copiar y pegar, edición de texto en el lugar, información sobre herramientas, menús contextuales, diseños automáticos, plantillas, vinculación y modelos de datos, administración de estado transaccional y deshacer, paletas , vistas generales, controladores de

eventos, comandos y un sistema de herramientas extensible para operaciones personalizadas.

2.2. Vaadin

Figura 2.2: Vaadin Logo

2.2.1. ¿Qué es GO.JS?

Vaadin [1] es un framework de desarrollo de SPA que permite escribir el código de dichas aplicaciones en Java o en cualquier otro lenguaje soportado por la JVM 1.6+. Esto permite la programación de la interfaz gráfica en lenguajes como Java 8, Scala o Groovy, por ejemplo.

2.2.2. Características

Uno de las características diferenciadores de Vaadin es que, contrario a las librerías y frameworks de JavaScript típicas, presenta una arquitectura centrada en el servidor, lo que implica que la mayoría de la lógica es ejecutada en los servidores remotos. Del lado del cliente, Vaadin está construido encima de Google Web Toolkit, con el que puede extenderse.

2.2.3. Aplicación

En este proyecto, Vaadin se encargara de realizar la comunicación entre el cliente y el servidor. De esta forma, será capaz de enviar y recibir datos, eventos y peticiones entre el componente Javascript (cliente) y el servidor.

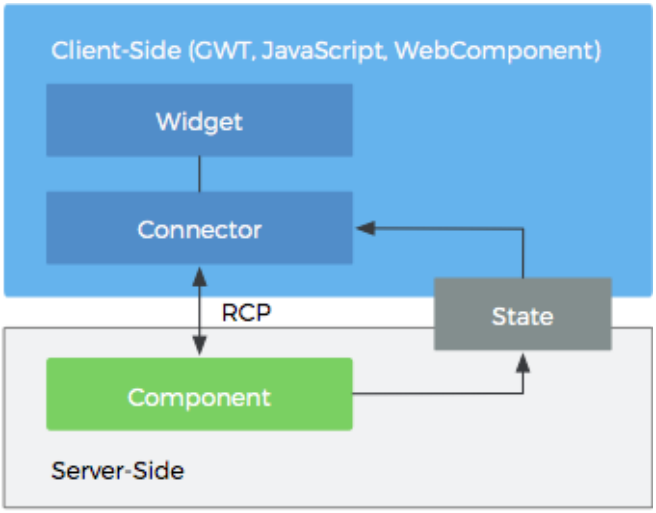

Figura 2.3: Esquema Cliente-Servidor

Capítulo 3

Process-Component

Este capitulo...

3.1. Introducción

Capítulo 4

Luca-Process

Este capitulo...

4.1. Introducción

Capítulo 5

Conclusiones

Este capitulo...

5.1. Introducción

Bibliografía

[1] Ari Handler Gamboa. adictosaltrabajo.com/tutoriales/introduccion-a-vaadin/.

[2] Eduardo Barrio. mhp-net.es/spring-vaadin-hibernate-tutorial-8-el-patron-mvp/.

[3] Northwoods Software. gojs.net/latest/index.html.

[4] Pivotal Software. docs.spring.io/spring-data/jpa/docs/current/reference/html.

[5] Pivotal Software. spring.io.

[6] Víctor Gómez. instintobinario.com/arquitectura-en-tres-capas/.