# Índice general

# Índice de figuras

# 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.

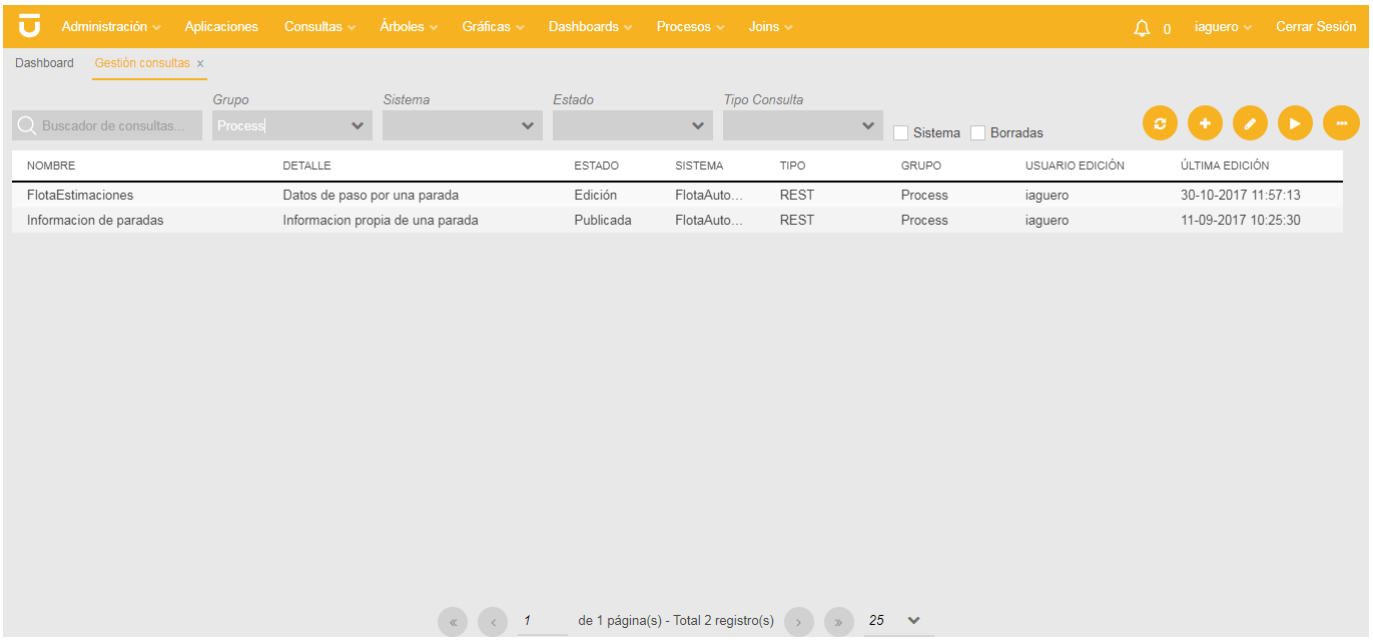


Figura 1.1: Vista de Gestión de Consultas

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.

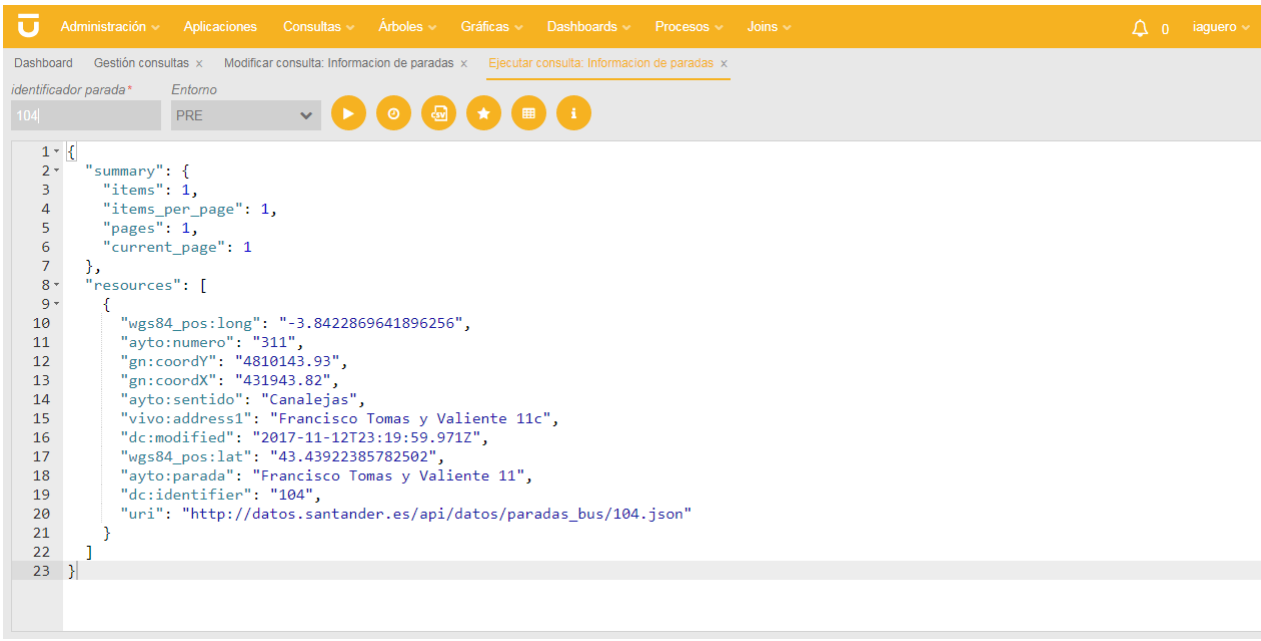


Figura 1.2: Vista de Ejecución de Consultas

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.

## 1.3. Ingeniería de Requisitos

En esta sección se contará el proceso llevado a cabo para aprender o entender la estructuración y definición de requisitos de la actual LUCA, así como la propia descripción de los requisitos que serán necesarios para la implementación del proyecto a realizar.

### 1.3.1. Proceso de integración en LUCA

El primer paso llevado a cabo para la introducción en el producto de LUCA y entender así su funcionamiento, fue una reunión con el Jefe de Proyecto y con el Gerente para describir, analizar y explicar todos los aspectos de LUCA.

### 1.3.2. Definición de Requisitos

Una vez aclarados los términos del actual LUCA, se abordo el incremento que se quería llevar a cabo, del que parte este Trabajo de Fin de Grado. Se explicaron los términos de Proceso, variables de entradas y de salidas, consultas y demás.

Además de la explicación, se proporcionaron unos documentos técnicos tanto del componente gráfico del proceso como del incremento sobre LUCA. Estos documentos pueden encontrarse en el Anexo adjunto a la memoria.

Dado que la aplicación a construir es un incremento de un producto ya existente, no hace falta identificar la fuente de los requisitos ya que es el propio gerente o impulsor de dicho incremento el que actúa como stakeholder, junto con el jefe del proyecto los encargados de formar los objetivos conjuntos de la captura de información y de los propios requisitos.

Cabe explicar también, que este proyecto se funda en la implementación o desarrollo de dos componentes. El primero representa una herramienta focalizada en el ámbito gráfico, cuya función es la creación, borrado y actualización de elementos que constan de variables de entradas y salidas, en términos generales, manipulación gráfica de los elementos constituyentes del proyecto. El otro representa un proyecto incremento del actual producto LUCA, centrado en la gestión de los procesos, los cuales son creados a través de consultas y enlaces entre sus entradas y salidas.

Como ya se ha mencionado, en estos documentos se pueden encontrar los requisitos técnicos atribuidos al proyecto, pero, de forma resumida, se centran en tres pilares o requisitos principales:

- Concatenación de las consultas entre si pertenecientes a un mismo proceso.
- Visualización del progreso de ejecución del proceso.
- Aplicar criterios de navegación a partir de los resultados de salidas.

La especificación de estos requisitos se encuentra en los documentos técnicos citados anteriormente.

## 1.4. Arquitectura LUCA

Esta sección se propone explicar y especificar la actual arquitectura de LUCA, así como, mostrar la integración con el componente o proyecto gráfico (Luca-Process), el cuál, a su vez, utiliza un componente javascript abstracto implementado con la librería javascript GO.JS.

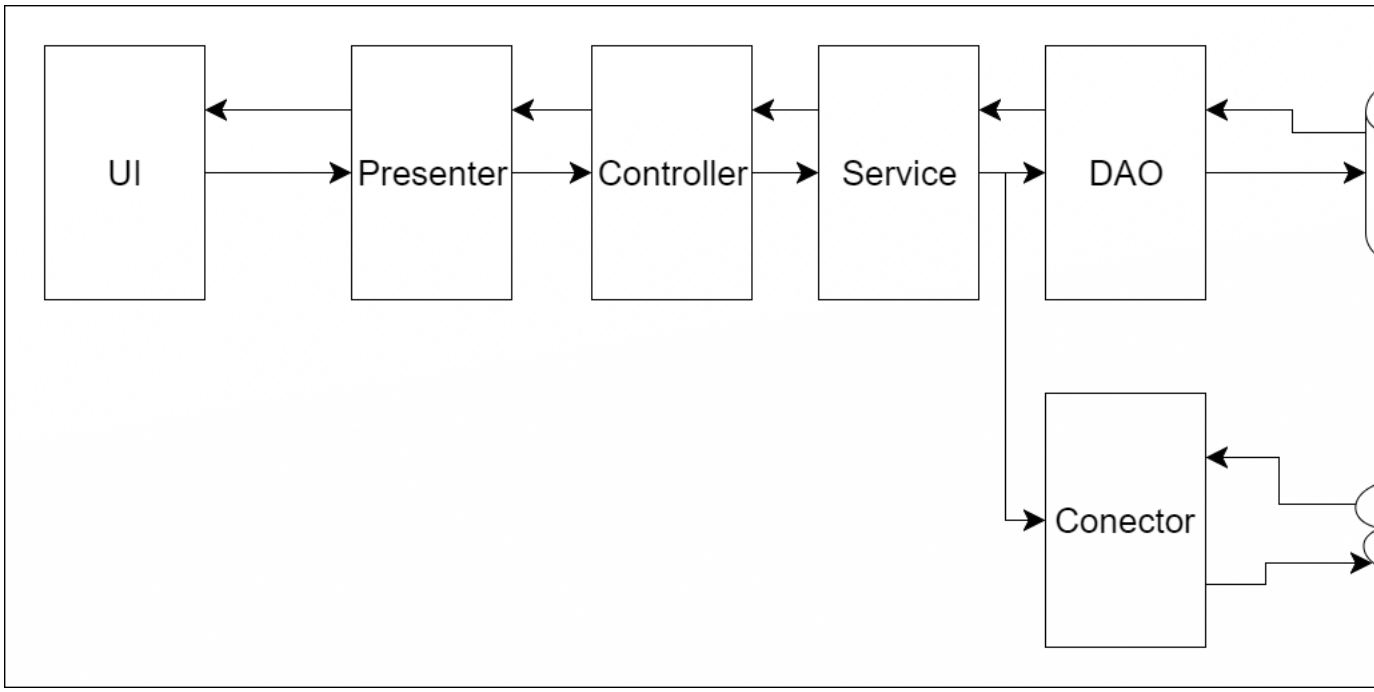


Figura 1.3: Arquitectura de LUCA

Para comenzar, se explica a continuación la arquitectura de LUCA previa al incremento que se propone realizar. En la figura superior podemos observar una arquitectura en tres capas bajo el patrón modelo-vista-presentador (MVP). Este patrón se caracteriza por separar de forma clara y concisa las vistas de la lógica de negocio.

En la arquitectura mostrada podemos observar, que la capa del presenter es la encargada de realizar la comunicación y control entre la vista y el modelo, y de albergar la gestión de eventos desde la vista.

Otro aspecto importante reside en la capa de servicio. Esta capa es la encargada de recibir los datos desde las diferentes fuentes de datos, ya sea de la base de datos de LUCA, o de un recurso externo de cualquier tipo (como por ejemplo la extracción de información de un servicio REST, SOAP o de otra base de datos ajena).

A continuación se muestran dos apartados que describen el funcionamiento y arquitectura del conector de LUCA y del Process-Component.

### 1.4.1. Arquitectura del Conector

El conector de LUCA es un componente que se encarga de recibir o recoger los datos de los diferentes recursos albergados en las diferentes fuentes de datos externas. Puede ser de diferentes tipos, como es un conector REST, SOAP, BBDD ...

En el ejemplo posterior podemos ver un diagrama que describe dicha interacción. El conector en función del tipo va a comunicarse con un cliente diferente, ya sea el HTTPClient [1] de Apache o JDBC [6] de Oracle, para realizar la comunicación y recepción de datos con los diferentes recursos.

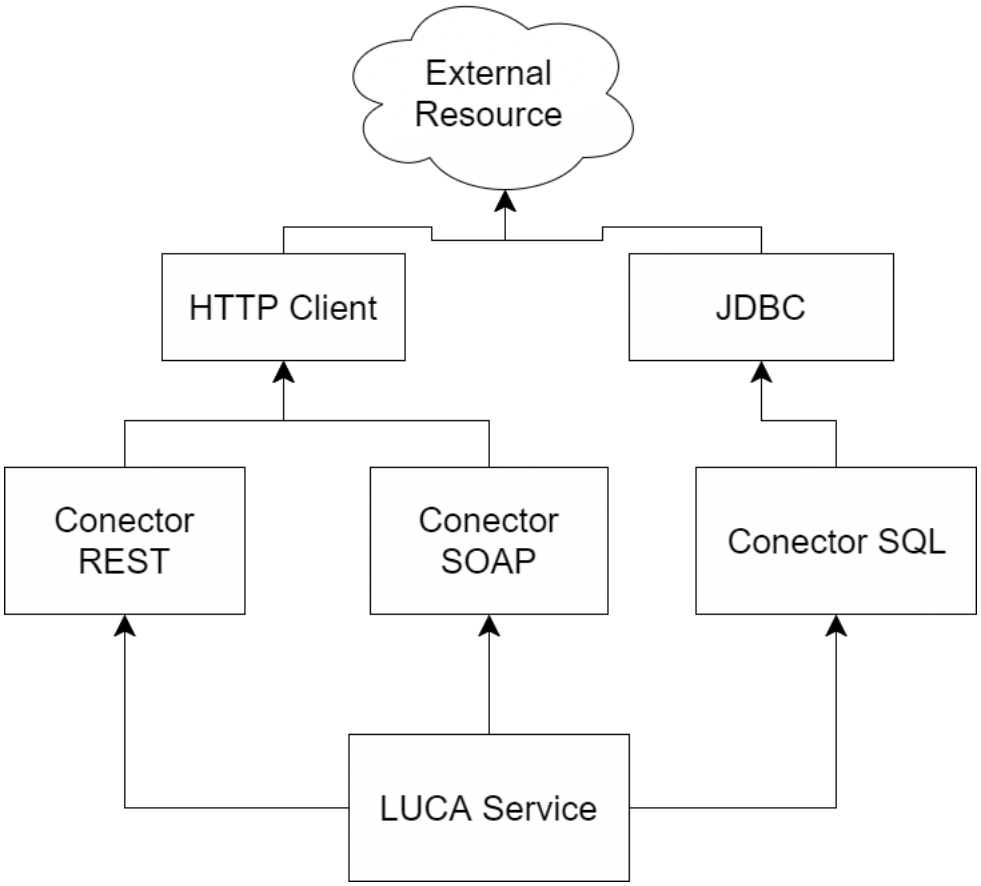


Figura 1.4: Arquitectura del Conector

### 1.4.2. Arquitectura del Process-Component

El Process-Component es un proyecto abstracto (debe de ser importado e implementado por un proyecto o componente padre) encargado de recibir y comunicar los eventos realizados sobre una interfaz construida a partir de la librería GO.JS.

La arquitectura del Process-Component se ostenta en dos pilares. El primero es la implementación de la lógica del propio componente y el segundo es la implementación de un conector que se comunica con una librería de GO.JS que también debe de ser definida.

La lógica del componente se basa en un estado ( el cuál alberga todos los elementos para poder formar la vista), y en una serie de acciones o comandos que se pueden realizar sobre él, y donde tras cada acción, se realiza una comunicación con el conector para que este se encargue de modificar el estado de los elementos de GO.JS.

El conector internamente se compone de la librería encargada de definir el ámbito gráfico con el que se va a trabajar, de una serie de eventos que serán trasladados a la lógica del Process-Component, y de modificar los elementos en activo en la vista bajo la orden especificada.

A continuación, se muestra una figura explicativa de dicha interacción interna entre los componentes:

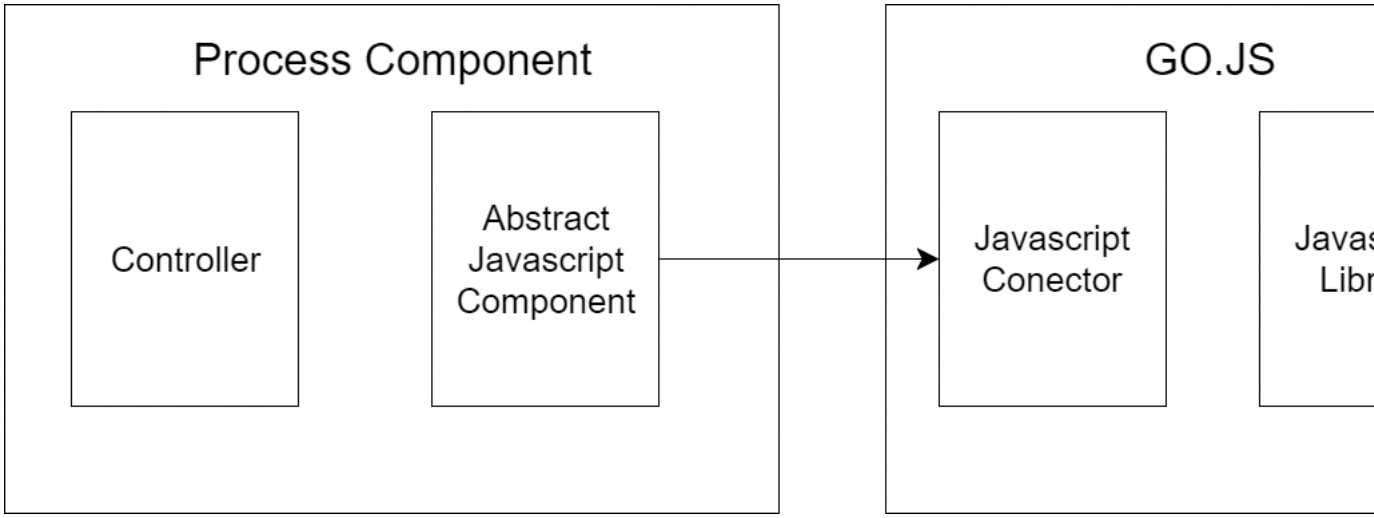


Figura 1.5: Arquitectura del Process-Component

Como resumen del funcionamiento del Process-Component, éste es el encargado de recibir eventos realizados sobre la interfaz gráfica, a través del conector hasta llegar a la lógica del componente para publicarlos o abstraerlos a un conjunto de elementos escuchadores que deberán de ser atendidos por el componente implementador de este Process-Component.

## 1.5. Planificación

La sección presente describe el proceso llevado a cabo para realizar la implementación tanto del componente gráfico (Process-Component) como del componente de Luca (Luca-Process).

El proceso que se llevó a cabo para realizar el desarrollo del proyecto se basó en dos fases:

- Una primera en la que se estudió el funcionamiento de la herramienta gráfica de GO.JS y después se realizó la implementación del Process-Componet.
- Una segunda en la que utilizando el componente descrito anteriormente, se implementa un incremento utilizando la lógica previa de LUCA.

### 1.5.1. Implementación

A continuación se explica brevemente el proceso de implementación llevado a cabo para ambas fases:

- Process-Component

  El primer paso para empezar la implementación del Process-Component, fue aprender y practicar con ejemplos y con pruebas de concepto para ver como funcionaba y aprender la sintaxis de la librería de GO.JS.

  Tras este primer paso, se realizó la implementación del modelo (se puede encontrar en los documentos técnicos), para posteriormente generar una primera prueba de concepto conjunta realizando una comunicación con la librería GO.JS.

  Por último se implementaron los eventos y escuchadores para posteriormente centrarse en completar dicho componente.

- Luca-Process

  En este componente se sigue un metodología similar al anterior. Primero se crea una prueba de concepto sencilla para ver la comunicación entre los componentes para posteriormente empezar a montar todo el sistema de recepción de eventos y llamadas al componente gráfico.

  El siguiente paso fue seguir el estilo arquitectónico ya existente en LUCA, utilizando el patrón MVP [3] para crear toda la jerarquíca de capas necesarias para la comunicación con los servicios ya existentes de LUCA y la propia base de datos de LUCA.

### 1.5.2. Pruebas

Este apartado describe el proceso de pruebas llevado a cabo para la comprobación y verificación de las diferentes funcionalidades del proyecto.

Las pruebas se centraron principalmente en el Luca-Process, ya que es el componente que alberga la mayor lógica del proyecto debido al conjunto de servicios que lo compone y a la lógica sobre los presenters desplegada.

Profundizando en las pruebas implementadas, solo se han realizado pruebas de integración por varios motivos. El primero es que las pruebas unitarias no son necesarias hacerlas ya que se centran sobre la capa de repositorio y esta capa ha sido implementada con Spring Data Jpa [7] y ofrece ya una fiabilidad. El motivo por el que no se realizaron pruebas de sistema, aunque en una primera planificación estaban previstas hacerlas integrandolas con Selenium [9] fue la complejidad que lleva la integración junto con Vaadin [2], ya que la ventaja de programar mediante una captura de eventos sobre la vista toda la secuencia de movimiento sobre dicha vista pasa a ser programática, y debido a que había que ceñirse a unas fechas de entrega y al tiempo que llevaría dicha implementación se decidió omitirlas.

Las pruebas de integración que se llevaron a cabo se implementaron con Spring Test [8] y JUnit [4]. Desglosando la composición de los mismos, se utilizó en cada test un fichero sql que declaraba las instrucciones de inserción de datos en la base de datos de pruebas necesarios para el correcto funcionamiento de los mismos. La base de datos de pruebas es una base de datos Mysql en local. relacional

# 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


Figura 2.1: GO.JS Logo

### 2.1.1. ¿Qué es GO.JS?

Go.JS [5] 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 [2] 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.

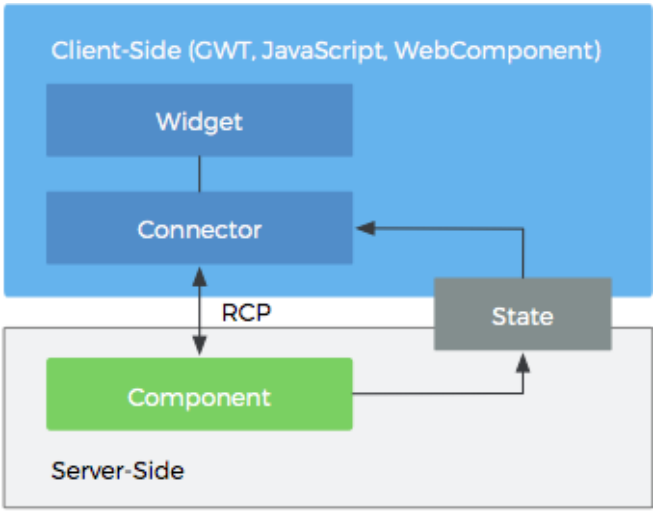


Figura 2.3: Esquema Cliente-Servidor

# Capítulo 3

# Process-Component

Este fragmento del proyecto o componente se dedica exclusivamente al apartado gráfico, el cuál posee una sintaxis ya definida en el apartado del diseño arquitectónico.

Este componente es una herramienta que se dedica a proveer métodos para interactuar con él y además es capaz mediante escuchadores de avisar a los clientes que lo requieran sobre los eventos ocurridos sobre la interfaz gráfica.

Su implementación se centra en dos pilares centrales. Por una parte se ha realizado un proyecto Vaadin encargado de mantener el estado de la aplicación gráfica, y por otro lado un conjunto de ficheros Javascript implementados sobre GO.JS que se encargan de modificar el entorno gráfico mediante sentencias .

- Proyecto Vaadin

  Este proyecto es el encargado de crear los metodos necesarios para interactuar desde el exterior con el esquema creado previamente. Además debe de permitir insertar escuchadores para los eventos proporcionados desde GO.JS, de forma que se pueda establecer dos direcciones de comunicaciones. Una desde el exterior con el proyecto Vaadin y este con el conector y directamente con el entorno gráfico de GO.JS, y otro desde interacción con los eventos (por parte del usuario) desde el fichero Javascript de configuración (explicado en el apartado posterior), con el conector y este con el estado, es decir, con el proyecto Vaadin.

- Ficheros Javascript

Existe un primer fichero que permite configurar el esquema gráfico que se va a llevar a cabo (Procesos, Subprocesos, InputVariables, OutputVariables ...), así como todo el resto de propiedades gráficas, ademas de ser capaz de lanzar eventos preconfigurados.

El segundo fichero esencial para el funcionamiento de esta estructura es el fichero conector. Este es el encargado de declarar y configurar todos los eventos que se pueden lanzar, además de ser el encargado de realizar todos los metodos CRUD [1]necesarios para que puedan interactuar con las propiedades configuradas en el primer fichero citado previamente.

Esta imagen trata de resumir el esquema de actuación que se lleva a cabo para la comunicación entre los distinto elementos del Process-Component.

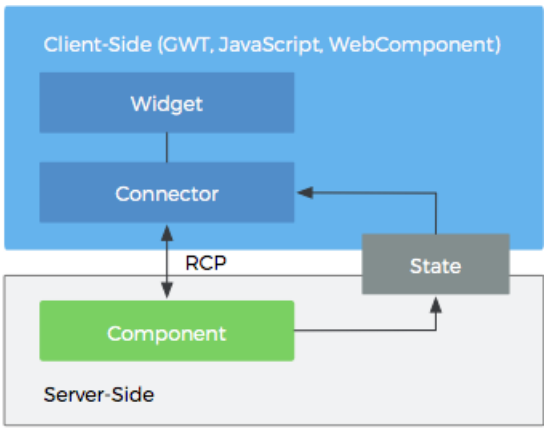


Figura 3.1: Estructuración en módulos

[1]CRUD es el acrónimo de crear, leer, actualizar y borrar, en esete contexto significa el conjunto de métodos para poder realizar dichas acciones sobre los distintos elementos existentes.

# Capítulo 4

# Luca-Process

Este capitulo describe el proceso de desarrollo e integración del componente gráfico (Process-Component). Se analizarán los aspectos del diseño y arquitectura a realizar así como las etapas de desarrollo del mismo.

## 4.1. Arquitectura

La arquitectura de Luca-Process tras la integración del Process-Component, cambia ligera o mínimamente respecto a la arquitectura demostrada en la introducción. La siguiente figura describe la diferencia respecto a la figura anterior:

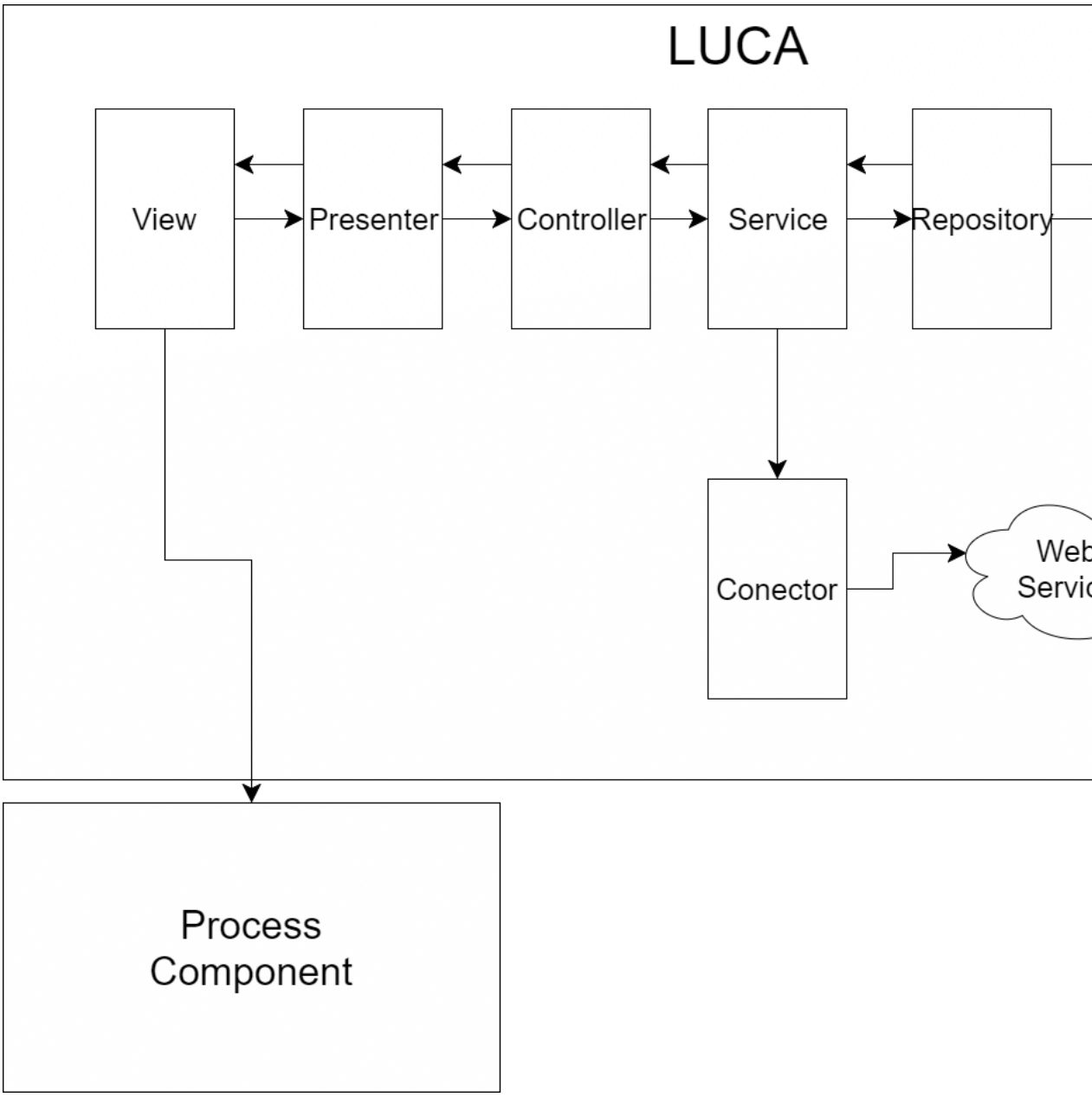


Figura 4.1: Nueva Arquitectura de LUCA

Como se puede apreciar, la única diferencia reside en que en esta citada nueva funcionalidad se utiliza otro proyecto aparte, el cuál se encarga de forma exclusiva de proveer componentes de visualización para poder abordar la sintaxis de procesos y subprocesos que se enlazan entre sí mediante entradas y salidas.

## 4.2. Desarrollo

En esta sección se describe el proceso de desarrollo o implementación del componente de LUCA que integra el componente gráfico.

El primer paso para empezar a comprender el diseño y arquitectura de LUCA, fue realizar una pequeña prueba de concepto aplicando el patrón MVP [3] e importando el componente gráfico para mostrar un proceso provisional, comprobando así el comportamiento del componente.

Una vez comprobado el funcionamiento, se comienza la implementación por capas del proyecto, creando la capa controladora junto con sus anotaciones de seguridad, la capa de servicio que realiza llamadas a la capa de repositorio (implementada con Spring Data [7]) para acceder a la base de datos, así como, al resto de servicios ya existentes en LUCA para la obtención de datos y ejecución de consultas. También se realizaron e implementaron, como se citó en la introducción, el conjunto de pruebas sobre la capa de servicio.

# Capítulo 5

# Conclusiones

Para finalizar, a continuación se relatan las conclusiones sacadas tras el transcurso y ejecución del proyecto.

El objetivo principal del proyecto consistía en crear un proyecto que fuese capaz de permitir al usuario de forma gráfica y sencilla crear una jerarquía de procesos para enlazar las diferentes consultas existentes en el producto LUCA.

Tras completar con éxito la implementación del proyecto, se implanto en el producto LUCA, para comprobar el correcto funcionamiento e integración y comprobar así la correcta y eficiente satisfacción de los requerimientos impuestos al proyecto.

En el apartado persona, ha sido una gran oportunidad poder realizar este proyecto en una empresa como es CIC, ya que ayuda a tener una constancia y hábito de trabajo y nos prepara para el ámbito empresarial. Además ha sido muy satisfactorio poder trabajar con mas de una tecnología ya que aporta muchos conocimientos.

# Bibliografía


[1] Apache. hc.apache.org/httpcomponents-client-ga/.

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

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

[4] Erich Gamma  Kent Beck. junit.org/junit5/.

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

[6] Oracle. www.oracle.com/technetwork/java/javase/jdbc/index.html.

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

[8] Pivotal Software. docs.spring.io/spring/docs/current/spring-framework-reference/

[9] SeleniumHQ. www.seleniumhq.org.