

Índice general

1. Introducción	1
1.1. Introducción	1
1.2. Planificación del proyecto	2
1.3. Estructura del documento	3
2. Ingeniería de Requisitos	5
2.1. Introducción	5
2.2. Captura de requisitos	5
2.3. Modelado de requisitos	7
3. Análisis y documentación	9
3.1. GO.JS	10
3.1.1. ¿Qué es GO.JS?	10
3.1.2. ¿Porqué GO.JS?	10
3.1.3. Características	10
3.1.4. Explicación	10
3.2. Vaadin	11
3.2.1. ¿Qué es GO.JS?	11
3.2.2. Características	11
3.2.3. Aplicación	11
3.3. Spring	12
4. Diseño Arquitectónico	13
4.1. Process Component	13
4.2. Luca Process	14
4.2.1. Esquema de componentes	15
5. Implementación	17
5.1. Process-Component	18
5.2. Luca-Process	19

6. Diseño Arquitectónico	21
6.1. Process Component	21
6.2. Luca Process	22
6.2.1. Esquema de componentes	23

Índice de figuras

1.1. Metodología En Cascada	2
2.1. Jerarquía de Requisitos	7
3.1. GO.JS Logo	10
3.2. Vaadin Logo	11
3.3. Esquema Cliente-Servidor	12
3.4. Spring Logo	12
4.1. Diagrama de clases	14
4.2. Diagrama de clases	15
4.3. Diagrama de Componentes	16
5.1. Estructuración en módulos	17
5.2. Estructuración en módulos	19
6.1. Diagrama de clases	22
6.2. Diagrama de clases	23
6.3. Diagrama de Componentes	24

Índice de cuadros

2.1. Conjunto de Requisitos del Proyecto	6
2.2. Conjunto de Requisitos del Componente Gráfico	7

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 agradecer 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 utilizan ya no 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 puede 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 capítulo del documento se propone a describir, en leves pinceladas, los objetivos principales que el proyecto debe de completar.

1.1. Introducción

En los últimos años, el volumen de datos que una empresa o entidad necesita y/o es capaz de gestionar o manipular, ha aumentado de forma vertiginosa. Además del volumen de datos a manipular, nos damos cuenta de que para acceder a lo diversos datos es necesario muchas veces establecer comunicaciones con los diversos recursos existentes para alcanzar el objetivo.

Con el objeto de facilitar este proceso de recuperación de información almacenada en sistemas y fuentes de datos heterogéneas, 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.

Actualmente LUCA proporciona mecanismos o abstracciones 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.

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.

1.2. Planificación del proyecto

En este apartado se describe la metodología de desarrollo que ha sido utilizada para la cumplimentación del proyecto.

En el proyecto se optó por utilizar una metodología en cascada, en la que de forma iterativa se va puliendo el producto hasta completar los requisitos y objetivos establecidos.

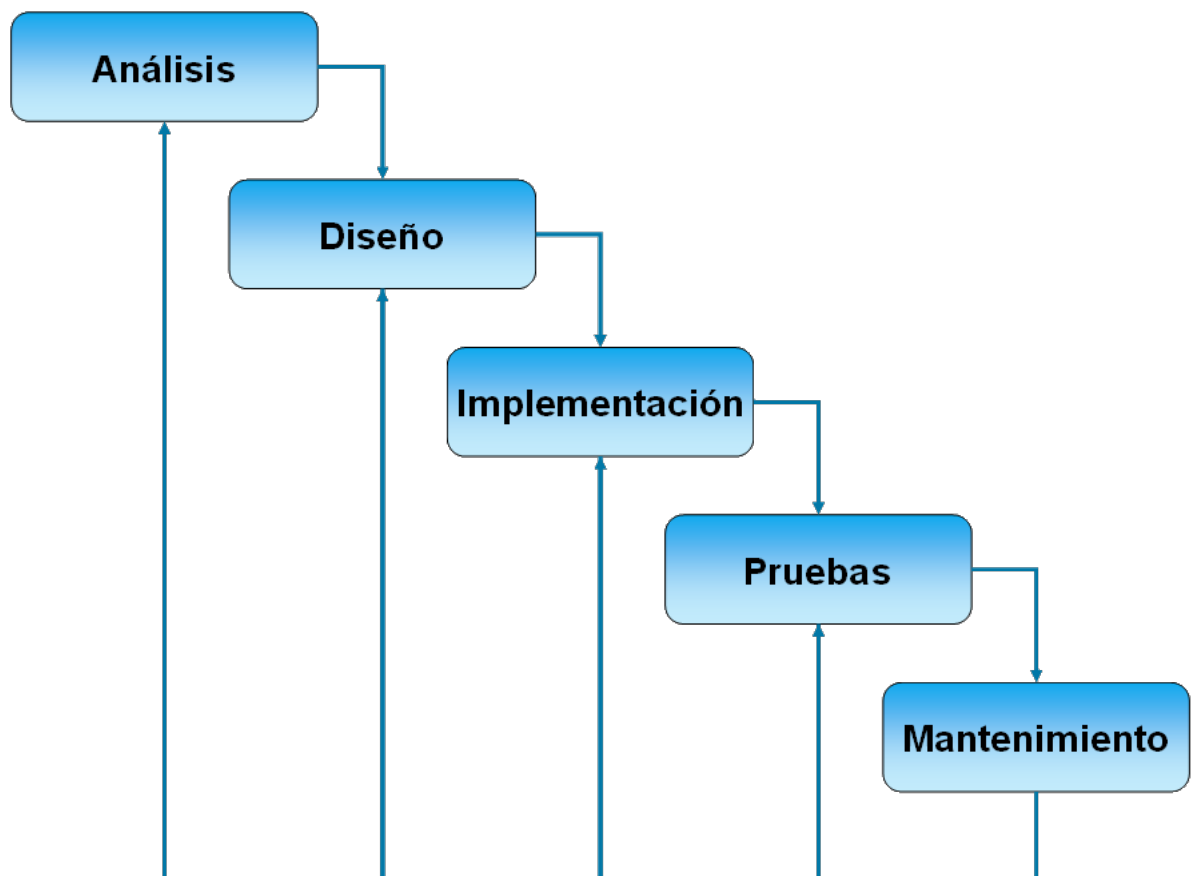


Figura 1.1: Metodología En Cascada

Acorde a la imagen mostrada previamente, la primera etapa de la metodología que se llevo a cabo para la cumplimentación del proyecto fue el análisis de los requisitos y objetivos del proyecto. El objetivo de esta etapa es describir de forma detallada el conjunto de requisitos y funcionalidades necesarias para la satisfacción del stakeholder y la completitud del sistema.

Una vez en posesión de la especificación de requisitos, el siguiente paso fue definir el diseño arquitectónico del proyecto en bloques independientes que pudiesen ser elaborados y probados de forma aislada. Otro requisito importante que se tuvo que tener en cuenta fue, que el producto debe de estar diseñado para poder hacer frente a futuros incrementos.

Utilizando el diseño arquitectónico ya definido, se focalizó el esfuerzo en el diseño de la aplicación. En esta se tuvo en cuenta que se pretenden realizar dos elaboraciones o componentes software:

- En una primera instancia se desarrollara un componente genérico cuya funcionalidad es de índole gráfica, por lo que no precisa de un desarrollo en capas común, ya que consta unicamente de un modelo de datos y una implementación web a través del Framework GO.JS, así la integración con Vaadin [1].
- En una segunda instancia se desarrollará una aplicación incremento del producto actual ya elaborado llamado LUCA. El cuál sigue un patrón MVP [2], y esta elaborado en Spring [5].

Para aplicación desarrollado con Spring, se creó una base de datos relacional asociada, la cuál servirá de fuente de persistencia de los datos almacenados.

Tras la especificación de la base de datos comenzó un proceso iterativo de implementación, verificación y pruebas en el que en función de los diversos requisitos se fue completando el proyecto.

1.3. Estructura del documento

Esta sección se propone describir la ingeniería de requisitos llevada a cabo, así como, profundizar en el diseño arquitectónico. También como introducción

inicial, se relatará de forma breve y concisa la librería javascript GO.JS y el framework Vaadin.

En primera instancia explicará el proceso de captura y especificación de requisitos llevado a cabo, así como, la toma de decisiones seguidas. También se centrará en detallar y explicar las decisiones para la especificación de la base de datos, que deberán de estar de acuerdo con las implementación actuales de las bases de datos de LUCA, y con la especificacion basada en Spring. Por último se explicará el proceso de desarrollo de las capas de repositorio, servicios y presentación.

Capítulo 2

Ingeniería de Requisitos

Este capítulo describe el proceso de captura de requisitos, así como, el modelado de los requisitos tanto funcionales como no funcionales.

Contents

2.1. Introducción	5
2.2. Captura de requisitos	5
2.3. Modelado de requisitos	7

2.1. Introducción

El capítulo actual se propone explicar de forma detallada la fase de ingeniería de requisitos, en la que se recoge toda la información necesaria y relativa a los requerimientos para poder presentar al stakeholder los objetivos que debe de tener la aplicación para poder considerarse como completada.

Por tanto, teniendo en cuenta lo citado anteriormente, este capítulo tratará primero el proceso de captura de requisitos llevado a cabo, para posteriormente, mostrar y detallar el modelado de requisitos subyacente de la etapa anterior.

2.2. Captura de requisitos

En la presente sección se describe el proceso de identificación de requisitos llevado a cabo para que nuestra aplicación se pueda considerar como completada.

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 objetivos conjuntos de la captura de información y de los propios requisitos.

A raíz de una demostración del producto actual LUCA, y de la reunión para la captura de requisitos con el gerente de la empresa CIC, así como del jefe de proyecto bajo el que estuvo supervisado el proyecto actual, se muestra el resultado del proceso de la captura de requisitos llevada a cabo.

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.

A continuación se muestra, por cada componente, una tabla desglosando los diferentes requisitos, y posteriormente, una breve explicación general del mismo:

Referencia	Requisito
REF-01	Los usuarios podrán consultar los datos referentes a los diversos elementos o c
REF-02	Los usuarios podrán modificar los datos de los elementos.
REF-03	Los usuarios podrán persistir los nuevos elementos creados.
REF-04	Los usuarios podrán concatenar los diferentes elementos (procesos, subprocesos
REF-05	Los usuarios podrán persistir, a partir de las consultas almacenadas en el siste

Cuadro 2.1: Conjunto de Requisitos del Proyecto

El usuario deberá de ser capaz de interactuar con los diferentes elementos existentes, como son las consultas almacenadas. Desde poder crear conjuntos de consultas anidadas a través de sus entradas y salidas para obtener una salida final objetivo, hasta poder modificar o eliminar los esquemas o procesos ya almacenados.

Referencia	Requisito
REF-01	Los usuarios deberán de ser capaces de enlazar los diferentes elementos entre sí.
REF-02	Los usuarios deberán de ser capaces de seleccionar de diferentes formas los elementos a
REF-03	Los usuarios deberán de ser capaces mover e interactuar con los diferentes elementos.
REF-04	Los usuarios deberán de ser capaces de interactuar con eventos y escuchadores para la

Cuadro 2.2: Conjunto de Requisitos del Componente Gráfico

El usuario podrá realizar acciones de forma gráfica como son la selección o arrastrado de los diversos elementos que conforman la estructura semántica del componente.

2.3. Modelado de requisitos

Este apartado muestra el proceso de modelado de requisitos, los cuales fueron especificados en el apartado anterior, así como la explicación y especificación del mismo. Además se llevará a cabo un análisis de requisitos para determinar en que nivel de la jerarquía de requisitos se encuentra.

El modelado de requisitos se ha regido bajo el concepto de jerarquía de requisitos escrito por Klaus Pohl [?], el cual define cinco niveles de abstracción para los requisitos desglosándolos en objetivos blandos y duros, y tareas que se corresponden con casos de uso de diferentes niveles. A continuación se muestra un esquema de jerarquía de requisitos:

Figura 2.1: Jerarquía de Requisitos

Tras haber categorizado los requisitos, se procederá a describir un caso de uso de cada tipo o nivel. Para ello se utilizará un diagrama de casos de uso junto a una especificación de casos de uso por cada caso de uso a explicar.

//TODO metodologia usada para la especificacion de los casos de usos de klaus pohl

caso de uso: gestionar procesos, alta baja etc

Tras haber detallado el conjunto de requisitos junto con sus niveles y desglose en objetivos y requisitos

Capítulo 3

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 implementación de la aplicación. De esta forma se puede realizar una planificación mas cercana a la realidad y partir de unos conocimientos mínimos para empezar a elaborar el proyecto.

Contents

3.1. GO.JS	10
3.1.1. ¿Qué es GO.JS?	10
3.1.2. ¿Porqué GO.JS?	10
3.1.3. Características	10
3.1.4. Explicación	10
3.2. Vaadin	11
3.2.1. ¿Qué es GO.JS?	11
3.2.2. Características	11
3.2.3. Aplicación	11
3.3. Spring	12

3.1. GO.JS



Figura 3.1: GO.JS Logo

3.1.1. ¿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.

3.1.2. ¿Porqué GO.JS?

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

3.1.3. 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.

3.1.4. 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.

3.2. Vaadin



Figura 3.2: Vaadin Logo

3.2.1. ¿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.

3.2.2. 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.

3.2.3. 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.

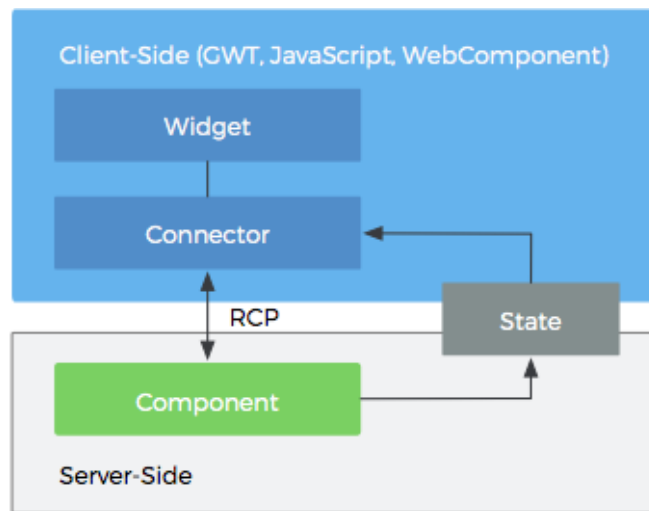


Figura 3.3: Esquema Cliente-Servidor

3.3. Spring



Figura 3.4: Spring Logo

Spring [5] es un Framework que proporciona un modelo de programación y configuración integral para aplicaciones empresariales modernas basadas en Java y en cualquier tipo de plataforma de implementación. Un elemento clave de Spring es el soporte de infraestructura a nivel de aplicación, ya que Spring se centra en abstraer a las aplicaciones empresariales del uso de diferentes frameworks para que los equipos puedan centrarse en la lógica empresarial a nivel de aplicación, sin lazos innecesarios con entornos de implementación específicos.

Capítulo 4

Diseño Arquitectónico

Tras haber adquirido la información suficiente respecto al conjunto de requisitos objetivos de satisfacción, y del conjunto de tecnologías que deben de integrarse en el proyecto, se muestra a continuación la especificación del diseño arquitectónico.

El diseño arquitectónico va a describir tanto la estructura arquitectónica del componente de procesos implementado con GO.js, tanto su integración en el producto LUCA, el cual, requiere de un diseño paralelo que implementar.

Contents

4.1. Process Component	13
4.2. Luca Process	14
4.2.1. Esquema de componentes	15

Para poder diferenciar estos dos diseños, el componente que implementa GO.JS se nombrará como 'Process Component' y el diseño del producto LUCA que integra el Process Component se llamara 'LUCA Process'.

4.1. Process Component

Debido a que es un proyecto orientado al entorno gráfico y hará las funciones de widget, el Process-Component ha recibido una arquitectura monocapa, centrándose en ella todos los elementos necesarios para montar una interfaz gráfica.

Para poder realizarlo, se establecerá una estructura de clases que se comunicará con un framework Javascript llamado GO.JS a través de Vaadin.

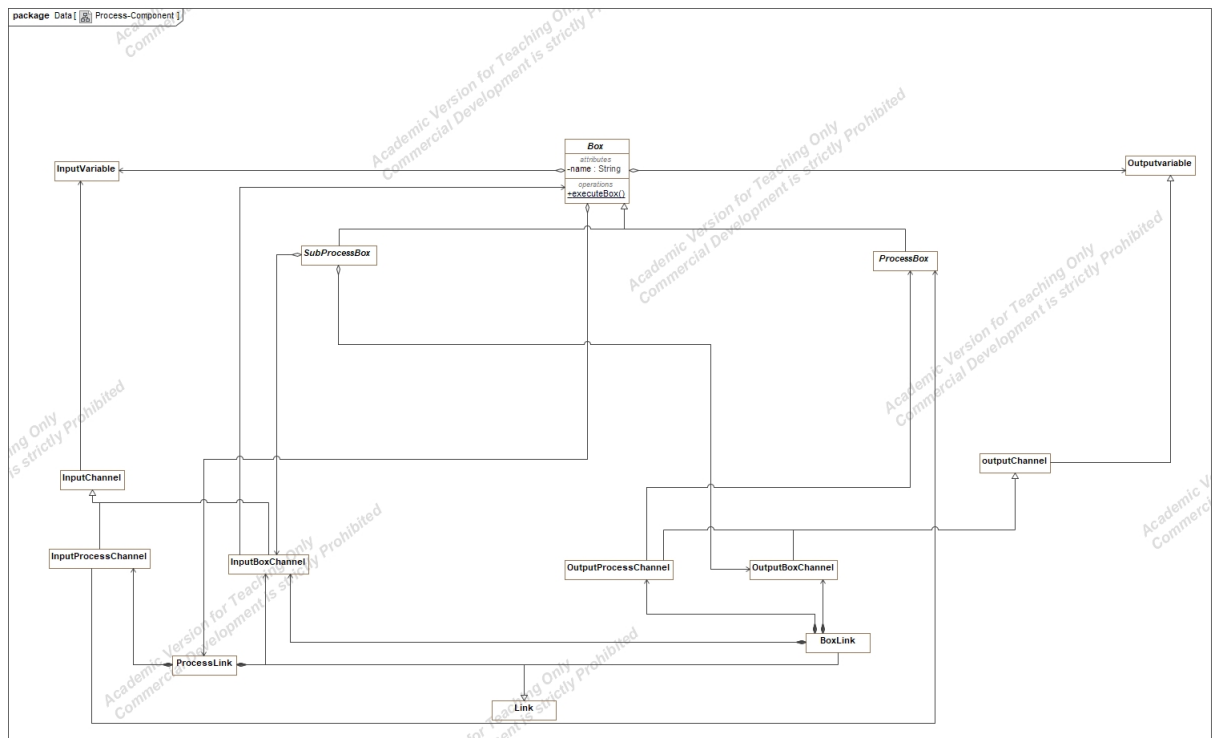


Figura 4.1: Diagrama de clases

4.2. Luca Process

El Luca-Process seguirá un estilo arquitectónico estándar, estará basado en una arquitectura en tres capas convencional [6]. Esta arquitectura ha sido elegida por varios motivos fundamentales.

- Porque es el modelo o esquema que mejor se plantea para este tipo de proyecto que consta de una interfaz gráfica que se comunica con un módulo para realizar transacciones entre ellos. Además, al constar de un apartado de persistencia, es muy beneficioso separar los módulos.
- Por capacidad de actuación frente a cambios, de esta forma se pueden cambiar módulos sin afectar al resto.
- Por compatibilidad y semejanza con el resto de proyectos y productos desarrollados en la empresa.
- Uno primero es porque es un estándar en el conjunto de productos o aplicaciones software.

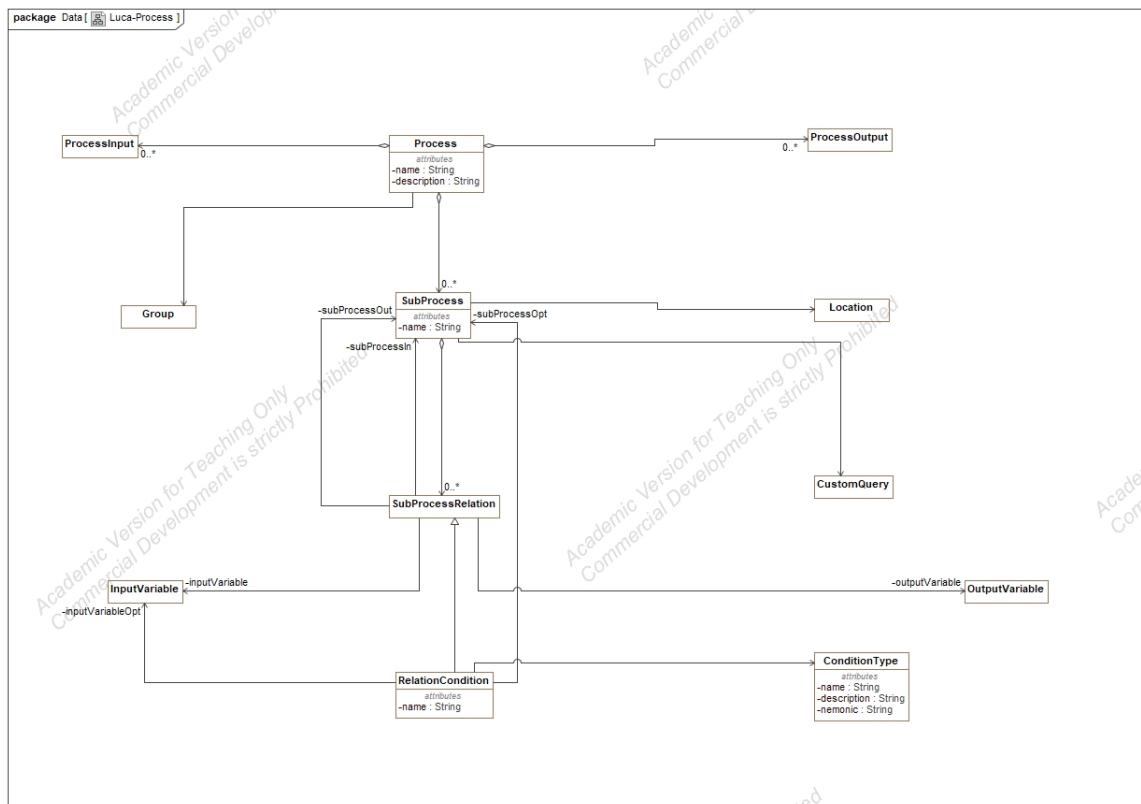


Figura 4.2: Diagrama de clases

4.2.1. Esquema de componentes

A continuación se muestra el esquema de componentes que explica la interrelación entre las diferentes capas de la aplicación con las consecutivas.

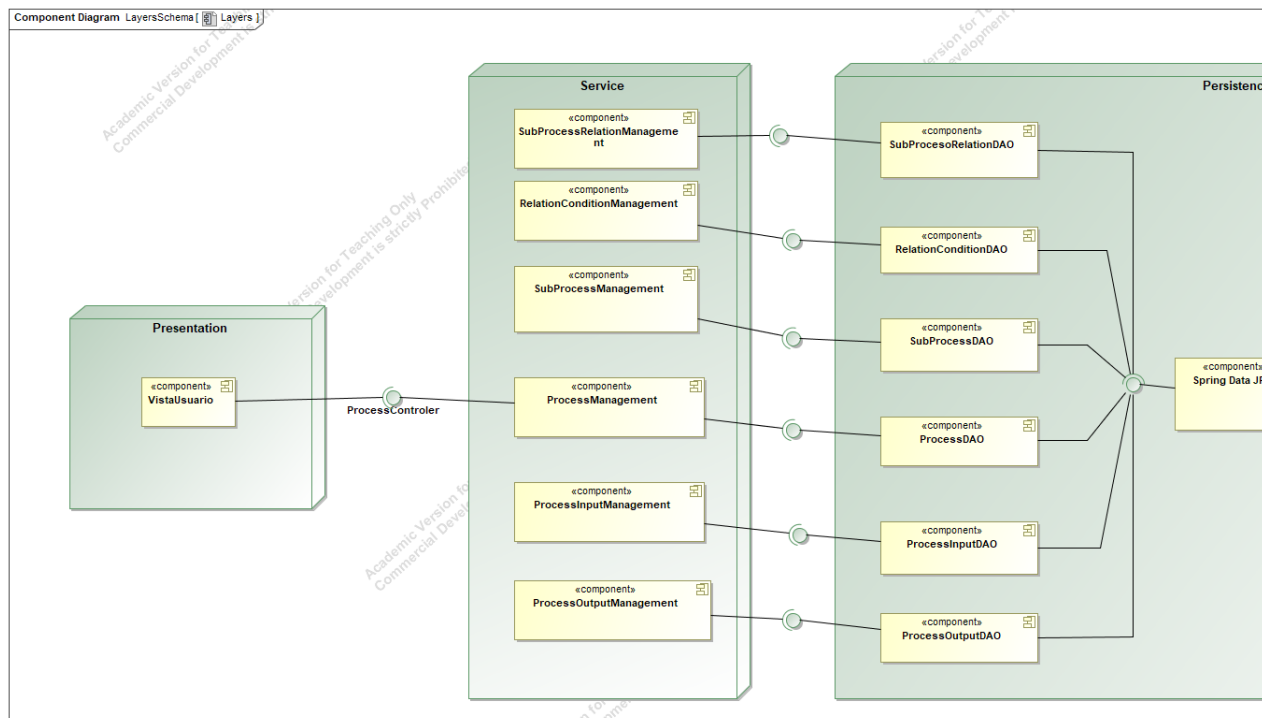


Figura 4.3: Diagrama de Componentes

En este esquema se puede apreciar que la vista del usuario se comunica con la capa de servicios a través de la interfaz del procesos, la cual a su vez se comunica con el resto de servicios y su propia interfaz de persistencia implementada con Spring Data JPA [4].

Capítulo 5

Implementación

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

A continuación, se declara un breve desglose de los dos componentes con los fragmentos que han sido implementados para poder satisfacer los el diseño arquitectónico.

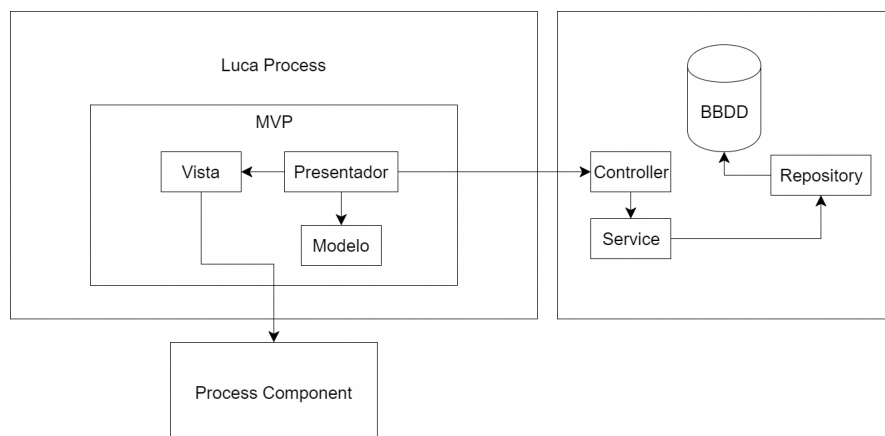


Figura 5.1: Estructuración en módulos

Contents

5.1. Process-Component	18
5.2. Luca-Process	19

5.1. 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, además 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 ¹necesarios para que puedan interactuar con

¹CRUD 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.

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 distintos elementos del Process-Component.

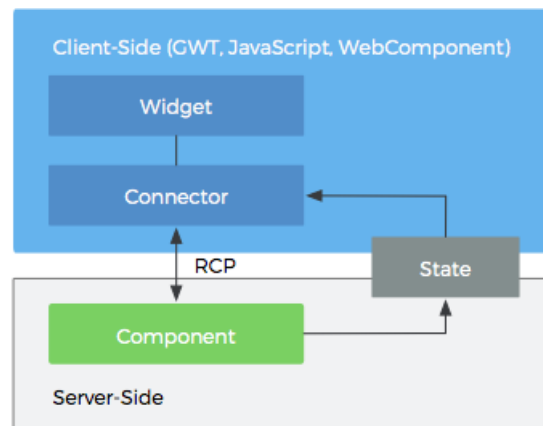


Figura 5.2: Estructuración en módulos

5.2. Luca-Process

Capítulo 6

Diseño Arquitectónico

Tras haber adquirido la información suficiente respecto al conjunto de requisitos objetivos de satisfacción, y del conjunto de tecnologías que deben de integrarse en el proyecto, se muestra a continuación la especificación del diseño arquitectónico.

El diseño arquitectónico va a describir tanto la estructura arquitectónica del componente de procesos implementado con GO.js, tanto su integración en el producto LUCA, el cual, requiere de un diseño paralelo que implementar.

Contents

6.1. Process Component	21
6.2. Luca Process	22
6.2.1. Esquema de componentes	23

Para poder diferenciar estos dos diseños, el componente que implementa GO.JS se nombrará como 'Process Component' y el diseño del producto LUCA que integra el Process Component se llamara 'LUCA Process'.

6.1. Process Component

Debido a que es un proyecto orientado al entorno gráfico y hará las funciones de widget, el Process-Component ha recibido una arquitectura monocapa, centrándose en ella todos los elementos necesarios para montar una interfaz gráfica.

Para poder realizarlo, se establecerá una estructura de clases que se comunicará con un framework Javascript llamado GO.JS a través de Vaadin.

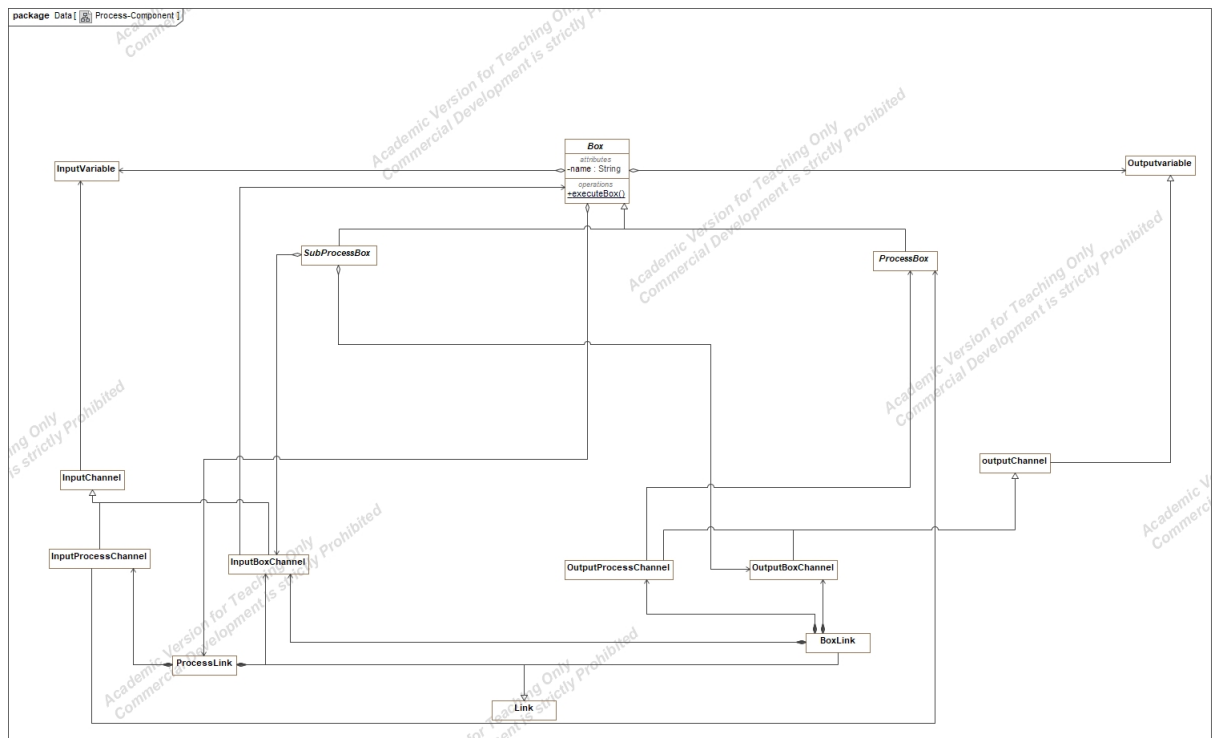


Figura 6.1: Diagrama de clases

6.2. Luca Process

El Luca-Process seguirá un estilo arquitectónico estándar, estará basado en una arquitectura en tres capas convencional [6]. Esta arquitectura ha sido elegida por varios motivos fundamentales.

- Porque es el modelo o esquema que mejor se plantea para este tipo de proyecto que consta de una interfaz gráfica que se comunica con un módulo para realizar transacciones entre ellos. Además, al constar de un apartado de persistencia, es muy beneficioso separar los módulos.
- Por capacidad de actuación frente a cambios, de esta forma se pueden cambiar módulos sin afectar al resto.
- Por compatibilidad y semejanza con el resto de proyectos y productos desarrollados en la empresa.
- Uno primero es porque es un estándar en el conjunto de productos o aplicaciones software.

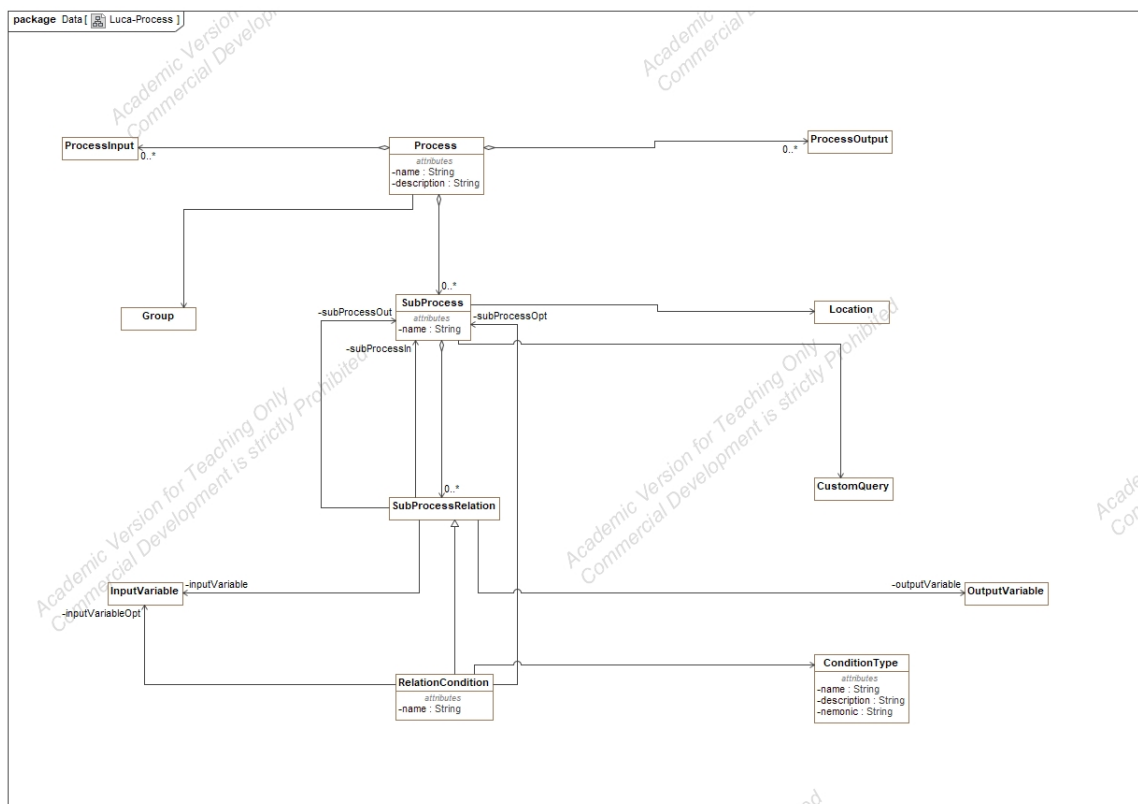


Figura 6.2: Diagrama de clases

6.2.1. Esquema de componentes

A continuación se muestra el esquema de componentes que explica la interrelación entre las diferentes capas de la aplicación con las consecutivas.

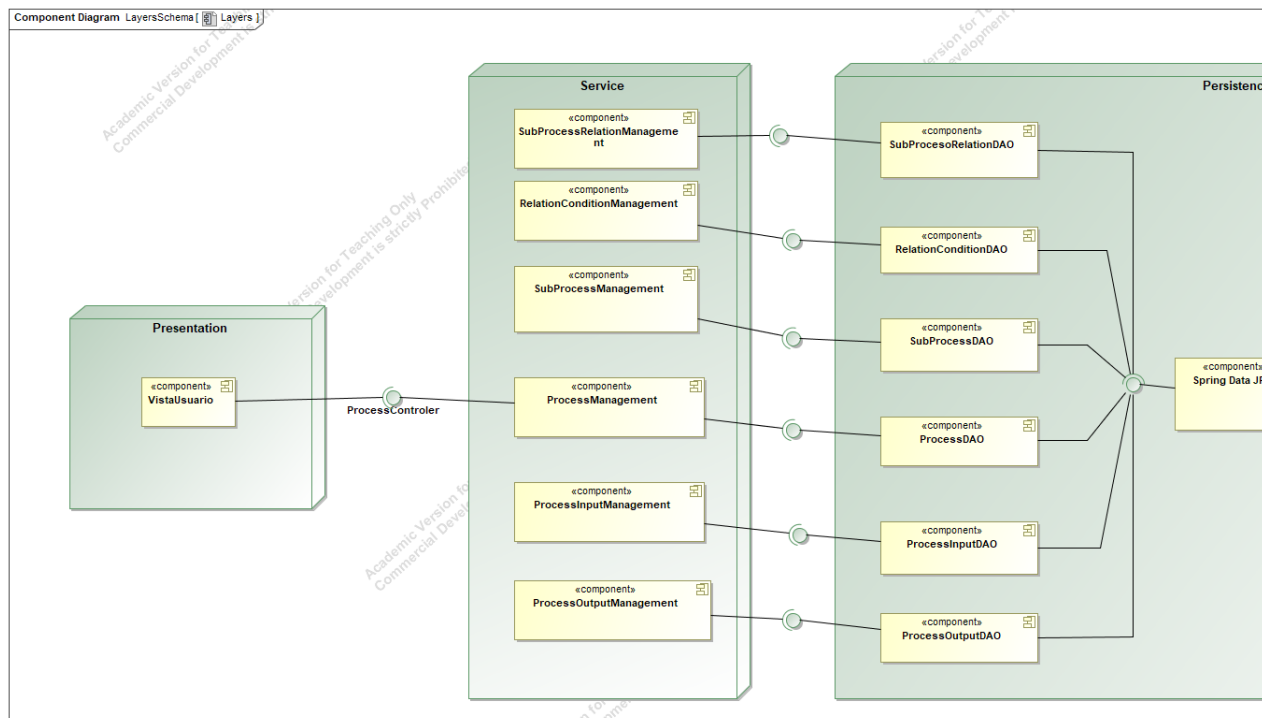


Figura 6.3: Diagrama de Componentes

En este esquema se puede apreciar que la vista del usuario se comunica con la capa de servicios a través de la interfaz del procesos, la cual a su vez se comunica con el resto de servicios y su propia interfaz de persistencia implementada con Spring Data JPA [4].

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/.