Plan de administración del proyecto cpPlugins 3.0

cpPlugins 3.0

Integrantes:
Pablo Andrés Ariza Luna
Carlos Orlando Barón León
Santiago Chaustre Perdomo
Andrés Felipe Cocunubo Quintero
Ailín Ana María Rojas Bohorquez

Director:
Leonardo Flórez Valencia PhD.

Asesor:
Jaime Andrés Pavlich Mariscal PhD.

Mayo 2019

Historial de cambios

Fecha	No. Sección	Nombre de la sección	Responsable
30/04/2019	-	Estructura general del documento, títulos.	Andrés Cocunubo
30/04/2019	2.3 y 4.1	Supuestos y restricciones. Modelo de ciclo de vida	Andrés Cocunubo
02/05/2019	7.2	Control de calidad	Pablo Ariza
03/05/2019	5.2	Análisis de riesgos	Andrés Cocunubo, Ailín Rojas
03/05/2019	4.3	Organigrama y descripción de roles	Ailín Rojas
04/05/2019	6	Entrega del producto	Pablo Ariza
05/05/2019	2.1	Visión, propósito y alcance	Santiago Chaustre
05/05/2019	1	Introducción	Ailín Rojas
05/05/2019	-	Arreglos generales, revisión y corrección	Andrés Cocunubo
05/05/2019	7.1	Ambiente de trabajo	Ailín Rojas
05/05/2019	4.2	Lenguajes y herramientas	Carlos Barón
05/05/2019	5.3	Administración de la configuración	Santiago Chaustre
05/05/2019	5.1	Administración de requisitos	Carlos Barón

Tabla de contenidos

Lista de figuras

Lista de tablas

Lista de ecuaciones

1 Introducción

Este documento tiene como propósito dar a conocer los aspectos relevantes del plan de administración del proyecto cpPlugins 3.0. Está dirigido a todos los interesados en conocer las características del plan de gestión y su estructuración.

El documento se encuentra dividido en secciones que abarcan: la idea a desarrollar, el alcance o resultado que se espera obtener, la especificación de todos los aspectos necesarios para elaborar el proyecto, como también la organización del equipo, las herramientas y procesos de control de calidad definidos por el grupo, para poder cumplir con los objetivos propuestos.

La información presentada incluye la vista general del proyecto, en la cual se expone el propósito, alcance y objetivos. De igual forma, se busca presentar el contexto, que comprende las metodologías, herramientas y organización del equipo para el desarrollo de este.

Continúa con la definición de las condiciones y criterios de entrega del producto, basados en el contexto y las necesidades del cliente. Así mismo, se exponen los planes de soporte para la gestión de riesgos, monitoreo y control, control de calidad y ambiente de trabajo del equipo.

En efecto, con este documento se quiere dar a conocer el plan y las estrategias a seguir por el grupo de trabajo durante la elaboración e implementación del proyecto. Además de presentar al lector la idea de lo que es cpPlugins 3.0.

2 Vista general del proyecto

En esta sección se encuentra la visión, el propósito, el alcance, los objetivos, los supuestos, restricciones y la descripción de los entregables del proyecto.

2.1 Visión, Propósito y alcance.

2.1.1 Visión

Actualmente, las aplicaciones del procesamiento y visualización de imágenes han afecto diversas áreas del conocimiento clásico. Entre ellas se encuentra la medicina, la cual se ha visto ampliamente beneficiada por los diversos algoritmos y maneras de procesar datos médicos estructurados, los cuales facilitan el trabajo de los médicos a la hora de dar un diagnóstico o de ejercer alguna actividad investigativa, que requiera analizar este tipo de datos.

cpPlugins 3.0, nace como una evolución de cpPlugins 2.0. Añadiendo una interfaz Web que facilite a los diversos usuarios diseñar pipelines de procesamiento y ejecutarlos, realizando todo el trabajo de computo pesado en el lado del servidor, simplificando el cliente y brindando mayor portabilidad.

Los conceptos principales del proyecto se encuentran detallados de manera específica en el (**ver el anexo 1. Propuesta_cpPlugins 3.0.docx, *sección 1.1.3. Propuesta de solución***)

2.1.2 Propósito

El propósito de cpPlugins 3.0, es ofrecer una plataforma web que permita a profesionales e investigadores del área de la medicina y computación gráfica, diseñar flujos de procesamiento de datos médicos de una manera sencilla e intuitiva. Utilizando un amplio conjunto de algoritmos y visualizadores, accediendo al servicio desde cualquier lugar y dispositivo con acceso a internet, sin la necesidad de grandes equipos de cómputo.

Finalmente, se busca ayudar a médicos a realizar diagnósticos más precisos en menos tiempo con ayuda de las herramientas provistas por el núcleo de cpPlugins.

2.1.3 Alcance

Se propone crear un prototipo funcional, que permita realizar prototipado rápido de tubos y filtros, utilizando la biblioteca de algoritmos provista por la versión inmediatamente anterior de cpPlugins. Todo el procesamiento de datos médicos estructurados se debe realizar del lado del servidor y los resultados deben poder ser observados por el cliente.

2.2 Objetivos

Los objetivos se encuentran en la propuesta de trabajo de grado **(ver el anexo 1. Propuesta_cpPlugins 3.0.docx, sección 1.2 Objetivos).**

2.3 Supuestos y restricciones

En esta sección se presenta una lista de los supuestos y restricciones que presenta el proyecto. Dichos elementos están basados principalmente en las definiciones dadas por el Project Management Institute (PMI). **Ver sección 3** para definiciones.

2.3.1 Supuestos

- Los requerimientos establecidos al inicio del proyecto se mantendrán durante el desarrollo del proyecto.
- Se realizarán retroalimentaciones semanales con el director del proyecto sobre las actividades desarrolladas.
- Todos los integrantes tienen acceso a internet.

- Cada miembro del grupo tiene conocimiento de los supuestos y restricciones, así mismo de las reglas del equipo establecidas.
- El grupo cuenta con la información necesaria para desarrollar las diferentes partes del proyecto.

2.3.2 Restricciones

- La propuesta del proyecto debe ser aprobada por los evaluadores, de tal forma que el producto pueda ser desarrollado el próximo semestre.
- Se debe entregar el producto final en las fechas acordadas.
- El tiempo de los integrantes del equipo es limitado debido a otras actividades académicas o laborales.
- El director de grado cuenta con poca disponibilidad, por lo tanto, se debe solicitar cita previa para realizar una reunión.

2.4 Entregables

Esta sección se encuentra detallada en la propuesta de trabajo de grado **(ver anexo 1. Propuesta_cpPlugins 3.0.docx, sección 1.3 Entregables, estándares utilizados y justificación).**

cpPlugins 3.0

3 Glosario

Para más detalle, en la versión final de la propuesta de trabajo de grado **(ver el anexo 1. Propuesta_cpPlugins 3.0.docx, sección 5. Marco teórico)** se encuentra la definición de conceptos clave para la comprensión del documento y del proyecto a realizar.

Interfaz de usuario: es el medio con que el usuario puede comunicarse con una máquina, un equipo o una computadora, y comprende todos los puntos de contacto entre el usuario y el equipo. Normalmente suelen ser fáciles de entender y fáciles de accionar [1].

Pipeline: transformación de un flujo de datos en un proceso comprendido por varias fases secuenciales, siendo la entrada de cada una la salida de la anterior [2].

Restricción: es una restricción o limitación aplicable, ya sea interna o externa al proyecto, que afectará el rendimiento del proyecto o un proceso [3].

Scrum: es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto [4].

Stakeholder: se refiere a todas aquellas personas u organizaciones afectadas por las actividades y las decisiones de una empresa [5].

Supuesto: son factores que, para los propósitos de la planificación, se consideran verdaderos, reales o ciertos, sin necesidad de contar con evidencia o demostración [6].

Programación orientada a objetos: es un paradigma de programación enfocado en construir aplicaciones más ordenadas y fáciles de mantener gracias a la forma en la que se estructura la información [7].

REST: es un estilo arquitectural para facilitar la comunicación entre sistemas por medio de estándares. Se caracteriza por no guardar estados [8].

Herramienta CASE: (Computer Aided Software Engineering) son herramientas para aumentar la productividad en el desarrollo de aplicaciones de software. Brindan herramientas de diseño, documentación, compilación, entre otras [9].

BPMN: es un estándar que brinda la notación para visualizar de manera gráfica procesos de negocio [10].

4 Contexto del proyecto

4.1 Modelo de ciclo de vida

La metodología se describe detalladamente en la Propuesta final de trabajo de grado. **(ver el anexo 1. Propuesta_cpPlugins 3.0.docx, sección 3. Proceso).**

4.2 Lenguajes y herramientas

A continuación, se presenta una lista de los lenguajes y herramientas que se utilizarán durante el desarrollo del proyecto, los cuales se clasificaron de acuerdo con su función y se seleccionaron con base en las necesidades específicas del proyecto. Se expone cada lenguaje o herramienta y la respectiva justificación de cómo ayuda a elaborar el proyecto o por qué fue seleccionado.

4.2.1 Herramientas y lenguajes de desarrollo

Para llevar a cabo el desarrollo de software en el proyecto, se necesitan herramientas para dos partes importantes, el cliente y el servidor.

4.2.1.1 Servidor

El desarrollo en el servidor consiste en una aplicación encargada de realizar el procesamiento de datos médicos estructurados. Para ello se utilizarán las siguientes herramientas:

4.2.1.1.1 VTK

El Kit de herramientas de visualización (VTK) es un sistema de software de código abierto, disponible gratuitamente para gráficos de computadora en 3D, modelado, procesamiento de imágenes, representación de volúmenes, visualización científica y trazado de gráficos en 2D [11].

VTK es necesario en el desarrollo del proyecto para procesar datos médicos estructurados, de manera que puedan ser visualizados de la forma en la que el usuario final lo requiera. Se debe realizar este tipo de procesamiento en el servidor para conseguir un cliente más liviano.

4.2.1.1.2 ITK

ITK es un kit de herramientas de software de código abierto para realizar el registro y la segmentación. La segmentación es el proceso de identificar y clasificar los datos encontrados en una representación muestreada digitalmente. El registro es la tarea de alinear o desarrollar correspondencias entre datos [12].

ITK debe ser utilizado en el desarrollo del servidor, debido a que está altamente relacionado con la aplicación ya existente de cpPlugins, que se montará en el servidor. Por lo tanto, es importante tener en cuenta esta librería si en algún caso se requiere adaptar alguna funcionalidad.

4.2.1.1.3 C++

C++ es un lenguaje de programación orientado a objetos que toma la base del lenguaje C [13].

Se debe hacer uso de C++ en el proyecto debido a que cpPlugins está escrito originalmente en este lenguaje de programación y, por lo tanto, se requiere hacer uso de la implementación de VTK e ITK en C++.

4.2.1.1.4 Spring Framework

Es un framework de código abierto para la creación de aplicaciones empresariales Java, con soporte para Groovy y Kotlin. Tiene una estructura modular y una gran flexibilidad para implementar diferentes tipos de arquitectura según las necesidades de la aplicación [14].

Debido a la solución que plantea cpPlugins 3.0 es necesario realizar un método para el intercambio de datos médicos entre el cliente y el servidor. Para esto, se decide montar un servicio web REST, para descargar y subir archivos. Se plantea que el servicio web sea REST para garantizar una fácil integración con el cliente y maximizar la velocidad del intercambio de datos.
Por lo anterior, se usa Spring Framework para el desarrollo de este servicio, ya que es una herramienta dominada por el equipo de trabajo y, adicionalmente, permite un desarrollo rápido de la aplicación requerida.

4.2.1.1.5 Java

Java es un lenguaje de programación orientado a objetos y una plataforma informática comercializada por primera vez en 1995 por Sun Microsystems [15].
Se utilizará Java en la elaboración del proyecto debido a que es el lenguaje utilizado para el desarrollo en Spring Framework.

4.2.1.1.6 JNI

Es un mecanismo que permite la utilización de aplicaciones nativas desde un programa escrito en Java. Las aplicaciones nativas son bibliotecas o funciones escritas en lenguajes como C, C++ y ensamblador para un sistema operativo en donde se ejecuta la JVM [16].
Es necesario el uso de JNI en el proyecto, debido a que es una herramienta que permite la comunicación entre la implementación existente de cpPlugins y el mecanismo de comunicación con el cliente, el servicio web.

4.2.1.2 Cliente

El desarrollo en el cliente comprende todo lo relacionado con la aplicación que el usuario final visualizará en el navegador. Para desarrollar esta aplicación se utilizarán las siguientes herramientas:
Debido a que se desarrollará una aplicación web, los siguientes lenguajes son de uso obligatorio:

4.2.1.2.1 HTML

HTML es un lenguaje de etiquetado que se emplea para dar formato a los documentos que se quieren publicar en la World Wide Web [17].

4.2.1.2.2 CSS

Es el lenguaje utilizado para describir la presentación de documentos HTML o XML, esto incluye varios lenguajes basados en XML como son XHTML o SVG [18].

4.2.1.2.3 Javascript

JavaScript es un lenguaje de programación que permite realizar actividades complejas en una página web, como mostrar actualizaciones de contenido en el momento e interactuar con elementos de la página [19].
Por otra parte, también se escogieron las siguientes herramientas:

4.2.1.2.4 React

React es una biblioteca de Javascript para el desarrollo de interfaces de usuario altamente interactivas. Los componentes de React se actualizan inmediatamente cuando los datos cambian [20].
React es seleccionado para el desarrollo del proyecto, debido a que, brinda facilidades a la hora de mostrar información en tiempo real al usuario, lo cual es un aspecto importante en cpPlugins 3.0.

4.2.1.2.5 Vtk.js

Vtk.js es una biblioteca de renderización hecha para visualización científica en la Web. Adapta la estructura y la experiencia de VTK para brindar una representación de alto rendimiento a su navegador [21].

El uso de vtk.js es indispensable en cpPlugins 3.0 debido a que es una herramienta que brinda componentes muy importantes para interactuar y visualizar datos médicos estructurados de manera gráfica.

4.2.1.2.6 Draw2D

Draw2D es un framework de JavaScript puro que permite la creación aplicaciones gráficas de diagramas, utilizando un API para facilitar el desarrollo [22].

Se escoge utilizar Draw2D en el desarrollo del proyecto, debido a que hace que el desarrollo de la interfaz gráfica sea mucho más rápido, brindando todos los componentes necesarios para que el usuario pueda estructurar un flujo de datos a manera de grafo.

4.2.1.2.7 Bootstrap

Bootstrap es un conjunto de herramientas de código abierto, para diseñar sitios y aplicaciones web. Solo se ocupa del desarrollo Frontend. Sus elementos de diseño están basados en HTML, CSS, y extensiones de JavaScript adicionales [23].

Se selecciona Bootstrap para el desarrollo del proyecto para darle el formato adecuado a la presentación de las páginas, y además por la experiencia de algunos miembros del equipo con la manipulación del framework.

4.2.2 Herramientas de administración y despliegue

Para el despliegue de la aplicación y su gestión o administración, se seleccionan las siguientes herramientas:

4.2.2.1 Docker

Docker es una tecnología que permite la creación y el uso de contenedores de Linux. Permite compartir una aplicación o un conjunto de servicios con sus respectivas dependencias en diferentes entornos, para automatizar la implementación de una aplicación [24].

Se hará uso de Docker en el proyecto para conseguir una configuración que soporte todo lo relacionado con VTK e ITK, y así facilitar la integración y el despliegue.

4.2.2.2 Kubernetes

Es un sistema de código abierto para automatizar la implementación, escalado y administración de aplicaciones en contenedores [25].

Se hará uso de Kubernetes en conjunto de contenedores en Docker, para así facilitar la integración continúa, descrita en la administración de la configuración **(ver sección 5.3.3 Proceso de integración continua)**, y la escalabilidad y administración en proyectos futuros que involucren cpPlugins 3.0.

4.2.2.3 Node.js

Node.js es un entorno de ejecución de JavaScript orientado a eventos asíncronos, Node.js está diseñado para construir aplicaciones en red escalables [26].

Es necesario el uso de Node.js en el proyecto, debido a que en este entorno se ejecuta gran parte del código de Vtk.js.

4.2.2.4 Apache Tomcat

Apache Tomcat es una implementación de código abierto de Java Servlet, JavaServer Pages, Java Expression Language y Java WebSocket [27].

Es necesario el uso de Apache Tomcat en el proyecto, debido a que es el contenedor de las aplicaciones desarrolladas en Spring Framework.

4.2.2.5 Auto DevOps GitLab

Auto DevOps de GitLab permite detectar, construir, probar, implementar y monitorear automáticamente aplicaciones. Se encarga de simplificar la configuración y ejecución de un ciclo de vida de desarrollo de software [28].

Se hará uso de Auto Dev Ops de GitLab para la integración continúa descrita en la administración de la configuración **(ver sección 5.3.3 Proceso de integración continua).** Se escoge esta herramienta por la experiencia de algunos miembros del equipo de trabajo con la misma.

4.2.3 Herramientas de versionamiento

Para llevar control de las versiones del código fuente y realizar desarrollo colaborativo, se seleccionan las siguientes herramientas:

4.2.3.1 Git

Git es un sistema de control de versiones que se caracteriza y diferencia de los demás sistemas de control de versiones en su forma de modelar sus datos. La forma de modelar los datos en Git consiste en que cada vez que se confirma un cambio se hace una imagen de todo el proyecto en la que los archivos modificados se almacenan como archivos nuevos y los archivos sin modificar se guardan como enlaces [29].

Se hace uso de Git en el proyecto para poder realizar desarrollo colaborativo y por la experiencia del equipo de trabajo con la herramienta.

4.2.3.2 GitLab

Es una plataforma de desarrollo de software completamente integrada que permite, rapidez, efectividad y cohesión en la producción de un proyecto de software. Provee repositorios de código y diferentes herramientas para simplificar el desarrollo de software [30].

Se hace uso de GitLab en el proyecto, en cuanto a versionamiento, para hacer uso de sus repositorios de código. Se selecciona la herramienta por las funcionalidades de DevOps descritas anteriormente.

4.2.4 Herramientas de modelamiento

En primer lugar, para el modelamiento es indispensable la utilización del siguiente lenguaje:

4.2.4.1 UML

Por sus siglas en inglés, Unified Modeling Language o Lenguaje Unificado de Modelamiento, es un lenguaje gráfico destinado al modelado de sistemas y procesos [31].

4.2.4.2 Visual Paradigm

Visual Paradigm es una herramienta CASE que brinda ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación [32].

Se selecciona Visual Paradigm para el desarrollo de diagramas UML en el proyecto, por la existencia de su versión “community” y la experiencia del equipo de trabajo con la herramienta.

4.2.4.3 Bonitasoft Modeler

Bonitasoft es un software BPM que combina técnicas de la de mejora de procesos con la automatización de estos, para coordinar la forma de trabajar de una empresa [33].

Se selecciona Bonitasoft Modeler para elaborar los diagramas BPMN, por los mismos motivos descritos anteriormente en Visual Paradigm.

4.2.5 Herramientas para el manejo de documentos

En cuanto a lo relacionado al producto con documentos, se seleccionan las siguientes herramientas:

4.2.5.1 Microsoft Office

Microsoft Office es un conjunto de productos ofimáticos o empresariales, que incluye procesamiento de textos, hoja de cálculo, gráficos de presentación y programas de comunicación por correo electrónico [34].

Se selecciona Microsoft Office debido a que suple todas las necesidades ofimáticas y todo el equipo de trabajo posee acceso a este software.

4.2.5.2 SharePoint

SharePoint es una colección de productos y elementos de software que incluye componentes y funciones de colaboración, módulos de administración de procesos, módulos de búsqueda y una plataforma de administración de documentos [35].

Se selecciona SharePoint para la elaboración del proyecto debido a que brinda una administración de documentos completamente compatible con Microsoft Office y todo el equipo de trabajo posee acceso al software.

4.2.6 Herramientas de planeación, monitoreo y control

Finalmente, para llevar un correcto control del proyecto y facilitar la planeación, se seleccionan las siguientes herramientas:

4.2.6.1 Trello

Trello es una plataforma para la gestión de proyectos que funciona en tiempo real y a velocidad real. Permite formar equipos para mantener a las personas involucradas en un proyecto, conectadas, a través de un "tablero" al cual se pueden añadir "tarjetas" relacionadas a una tarea o comentario. Permite monitorear el avance del proyecto y controlar el trabajo del equipo gracias a la interacción visible en el tablero [36].

Se selecciona Trello para la ejecución del proyecto, ya que se ajusta a las necesidades descritas en el modelo del ciclo de vida **(ver sección 4.1. Modelo de ciclo de vida).**

4.2.6.2 Microsoft Teams

Microsoft Teams permite la colaboración entre personas de un mismo equipo o el desarrollo de un proyecto concreto, compartiendo recursos y cuya función principal es la comunicación constante entre los miembros del equipo [37].

Se hace uso de Microsoft Teams para complementar la administración de los documentos, en conjunto con SharePoint, brindando funcionalidades de comunicación. Se selecciona la herramienta debido a su integración con herramientas previamente seleccionadas.

4.3 Organigrama y descripción de roles

En esta subsección, se encuentra la organización del equipo de trabajo, en donde se especifican los roles que desempeña cada miembro y sus respectivas responsabilidades.

4.3.1 Organigrama

La organización del equipo de trabajo se presenta en la **Ilustración 1. Organigrama cpPlugins 3.0,** en la cual se muestran los roles y las personas que los desempeñan. En primer lugar, en la parte superior se sitúan el director del proyecto y *product owner*; el profesor Leonardo Flórez PhD. Que comunica al equipo los requisitos a implementar y representa los intereses de la universidad en la ejecución del proyecto. Así mismo, al lado derecho se encuentra el asesor del proyecto; el profesor Jaime Pavlich PhD. Que guía y realiza sugerencias al equipo en aspectos específicos durante el proceso.

Por otro lado, en la parte inferior se encuentra el equipo de trabajo, el cual está conformado por cinco estudiantes que llevan a cabo el desarrollo del proyecto y son los responsables de la comunicación con el director/*producto owner* y asesor del proyecto.
Cada miembro del equipo está a cargo de un área del proyecto, las cuales se establecieron para una mejor división del trabajo y para efectos de calidad en cada aspecto de este. Cada integrante fue asignado a su correspondiente área de acuerdo con las habilidades, fortalezas y gustos expresados por cada uno. Sin embargo, cabe aclarar que esto no limita a los integrantes para que se desenvuelvan en otras áreas.
De acuerdo con la metodología scrum, se designó al integrante *Santiago Chaustre* como *scrum master*, ya que posee las habilidades exigidas por el rol.

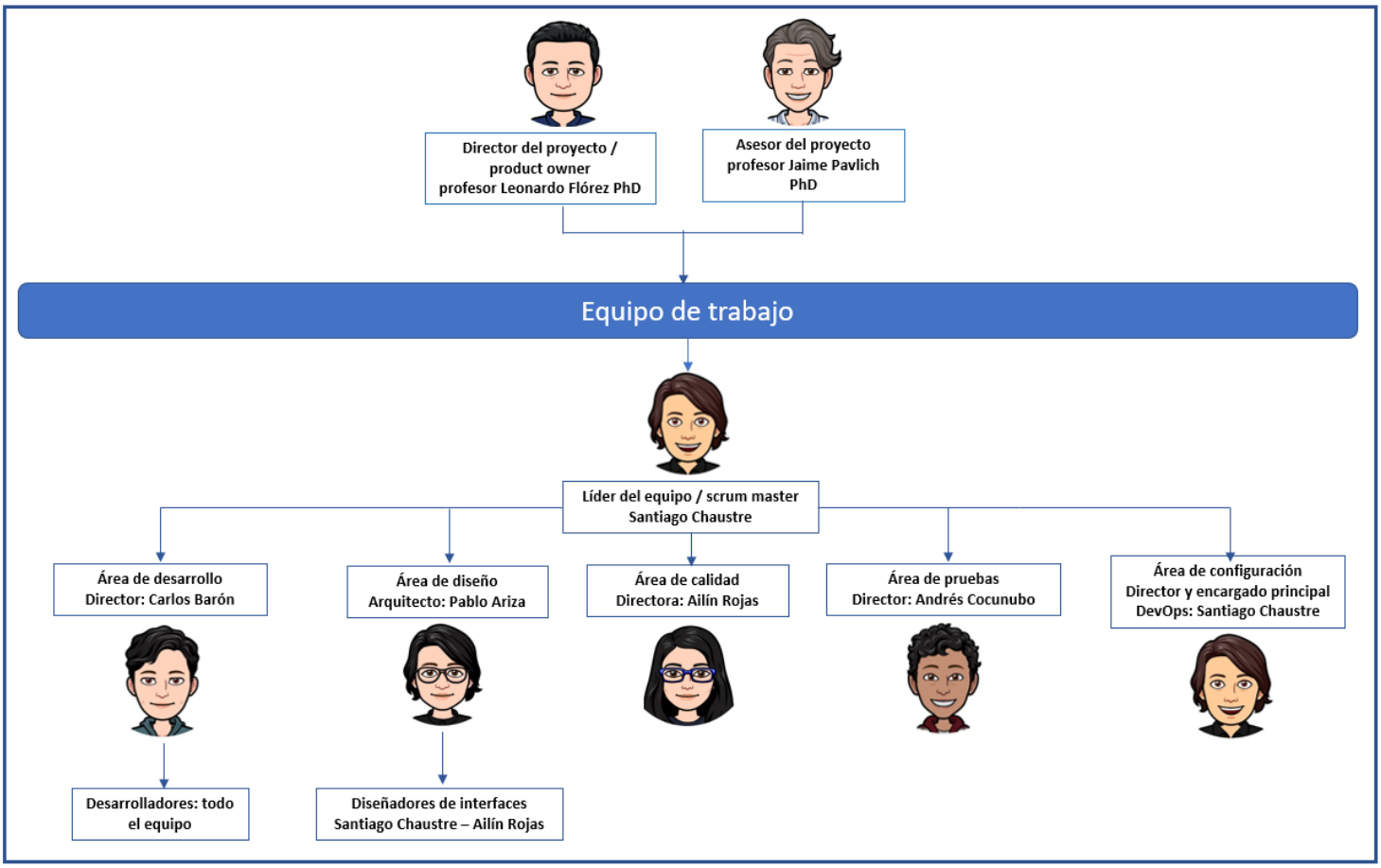

Ilustración 1 - Organigrama

4.3.2 Descripción de roles

Para la organización del equipo y la asignación de roles se tuvieron en cuenta los roles determinados por la metodología ágil *scrum*, **(ver anexo 1. Propuesta_cpPlugins 3.0, sección 3. Proceso)**, así como los roles determinados por los miembros del grupo de acuerdo con las áreas establecidas.
A continuación, se muestra la descripción de cada rol y las responsabilidades asociadas.

Rol	Descripción	Responsabilidades
Director del proyecto	Persona que guía al equipo de trabajo hacia el cumplimiento de los objetivos propuestos. Solicita avances periódicamente del proyecto, da instrucciones y monitorea el proyecto en ejecución.	• Orientar al equipo en la solución de problemas que puedan presentarse. • Asegurarse de que el proyecto cumpla con las normas establecidas por la universidad. • Calificar al equipo de trabajo.
Product owner	Persona que representa los intereses de los stakeholders, comunica los requisitos priorizados al equipo y establece los criterios de aceptación del producto. [38]	• Establecer los criterios de aceptación del producto. • Asegurarse de que todos los miembros del equipo comprenden los requisitos priorizados
Asesor del proyecto	Persona que guía al equipo ante dudas o dificultades que puedan surgir. Realiza sugerencias sobre cómo abordar aspectos relevantes del proyecto.	• Orientar y aconsejar al equipo de trabajo.
Líder del equipo	Persona encargada de dirigir y motivar al equipo de trabajo. Además de mantener la cohesión entre todos los miembros. Es quien orienta al equipo para ejecutar las instrucciones dadas por el director del proyecto.	• Estar al pendiente de lo que sucede en todas las áreas del proyecto. • Transmitir instrucciones de manera clara. • Ayudar a los miembros que presenten dudas o dificultades.
Scrum master	Encargado de promover y soportar las prácticas de scrum. Líder y facilitador del equipo scrum que asegura que la gestión del proyecto avance sin problemas [39].	• Se debe asegurar que los objetivos, el alcance y el dominio del producto sean comprendidos por todos en el equipo scrum de la mejor manera posible.
Director de desarrollo	Persona encargada de orientar, ayudar y entrenar en caso de ser necesario al equipo de desarrolladores. Suele ser el miembro del equipo más experimentado, también será capaz de asegurarse de que la ejecución sigue de cerca al diseño planteado. Influye en la calidad del código [40].	• Es quien selecciona los lenguajes de programación y frameworks de desarrollo. • Revisa que el código fuente solo contenga los componentes totalmente funcionales. • Comparte las mismas responsabilidades de los desarrolladores.
Desarrollador	Persona encargada de implementar las funcionalidades designadas por el director de desarrollo. Se asegura de que la aplicación funcione correctamente, que sea segura, que soporte el paso del tiempo, que sea fácilmente modificable y adaptable [41].	• Aplicar las reglas establecidas por el director de desarrollo. • Informar al director de área cualquier problema que se presente.

Arquitecto de software	Persona que en cooperación con el líder del proyecto participa en la toma de decisiones adecuadas para lograr una arquitectura del sistema que garantice el cumplimiento de los atributos de calidad exigidos por el cliente. Es quien traduce los requisitos a modelos para el desarrollo del sistema [42].	• Definir y diseñar la arquitectura del sistema teniendo en cuenta los requisitos funcionales y no funcionales, así como las reglas de negocio. • Realizar seguimiento para asegurar que se siga la estructura que planteó.
Diseñador de interfaces	Persona que configura la mejor experiencia posible para el público objetivo de la aplicación. Aplica bases teóricas, resultados del análisis del comportamiento de los usuarios y buenas prácticas de la usabilidad y accesibilidad web [43].	• Realizar el diseño de las pantallas que se mostrarán a los usuarios finales. • Mantener la comunicación con el equipo de desarrollo. • Diseñar propuestas para discutir con el equipo.
Directora de calidad	Asegura que la gestión de la calidad llegue a todas las áreas del proyecto. Se desenvuelve en varias áreas de trabajo, asegura que el trabajo cumpla y satisfaga los requisitos y criterios de aceptación del producto.	• Establece las políticas de calidad. • Constantemente revisa las partes realizadas por los miembros del equipo. • Realiza correcciones si es necesario
Director de pruebas	Encargado de la planificación, diseño, implementación y evaluación de las pruebas. Es quién posee los conocimientos necesarios en pruebas y herramientas para su realización.	• Generar el plan y modelo de pruebas. • Realizar un resumen de la evaluación de las pruebas
Director de configuración	Es el encargado de administrar y controlar los cambios en la evolución del software. Tiene como propósito establecer y mantener la integridad del producto de software [44].	• Definir un proceso formal de aprobación y seguimiento de los cambios. • Identificar elementos y componentes de configuración que serán gestionados.
Encargado principal DevOps	Responsable de la configuración inicial del pipeline de integración continua y despliegue, es quién explicará a los miembros del equipo como se utilizará la herramienta de DevOps.	• Monitorear que el proceso de integración y despliegue continuo se lleve a cabo de la manera correcta. • Gestionar las pruebas y configuraciones del pipeline. • Comunicarse con los miembros del equipo para gestionar la integración de las nuevas pruebas y nuevo código.

Tabla 1 - Descripción de roles

5 Monitoreo y control del proyecto

5.1 Administración de requisitos

Es el proceso establecido que se debe seguir para identificar o añadir nuevas a características del software a realizar. Entre las características se pueden encontrar nuevas funcionalidades, modificaciones a funcionalidades existentes y atributos de calidad del software. El proceso posee las siguientes fases:

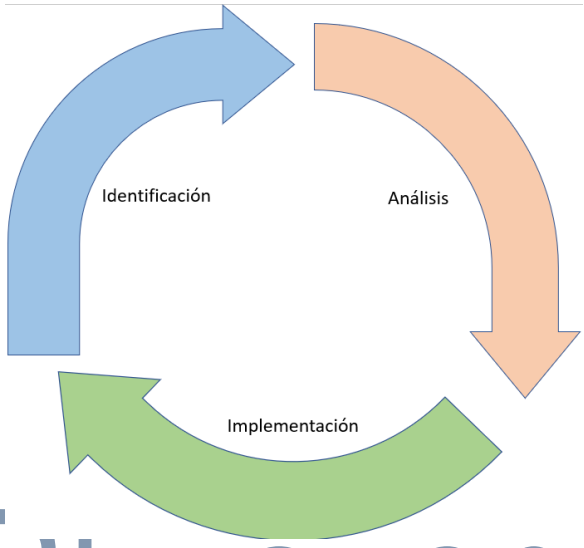

Ilustración 2 - Fases de la administración de requisitos

5.1.1 Identificación de requisitos

La primera fase o la fase de identificación de requisitos, consiste en una reunión con el cliente. En esta reunión, el cliente expone al equipo de trabajo todas las necesidades que necesita ver evidenciadas en el software. En la primera reunión de identificación de requisitos, durante la etapa de planeación del proyecto, solo se habla con el cliente para identificar las características del software. Sin embargo, en posteriores reuniones durante la ejecución del proyecto, se habla con el cliente para mostrar avances en el producto y recibir retroalimentación de este. La retroalimentación del producto puede dar como resultado la adición de una nueva característica, la modificación de una característica existente o la eliminación de una característica. Los resultados de esta fase quedan establecidos en un acta de reunión.

5.1.2 Análisis de requisitos

Después de haber identificado formalmente los requisitos del producto de la primera fase, se procede con la segunda, la cual consiste en un análisis del resultado. Este análisis comprende una reunión del equipo de trabajo, en donde se descomponen las diferentes tareas que implican los nuevos requisitos por parte del cliente. Una vez están identificadas las tareas a realizar, se hace un análisis en donde se obtiene la estimación de tiempo que comprende cada tarea, las herramientas necesarias para realizarla, los resultados esperados de cada tarea y el miembro del equipo adecuado para realizarla. Los resultados de esta fase quedan establecidos en un acta de reunión.

5.1.3 Implementación de requisitos

Finalmente, en la última fase, cada integrante realiza sus tareas para luego pasar al control de calidad (**ver sección 7.2 Control de calidad**), y así poder continuar con una siguiente iteración del proceso de administración de requisitos hasta culminar el proyecto.

5.2 Análisis de riesgos

Está sección tiene como objetivo presentar el análisis y la gestión de riesgos, el cual es un método sistemático de recopilación, evaluación, registro y difusión de información necesaria para formular recomendación orientadas a la adopción de una posición o medidas en respuesta a un peligro determinado [45, p. 5]. Además, dicho proceso se debe realizar junto a la mitigación y control de los riesgos.

Según el PMBOK, el análisis y administración de riesgos se divide en cuatro procesos fundamentales, los cuales se encuentran descritos en [46].

En la siguiente lista se encuentra cada proceso y sección del anexo donde está descrito a mayor detalle:

- **Clasificación de los riesgos:** los riesgos son documentados en esta fase, explicando las características de cada uno **(ver anexo 2. Riesgos.xls, primera hoja: Identificación)**.
- **Cuantificación de los riesgos:** se genera una valoración de los riesgos descritos en el proceso anterior en cuanto a su probabilidad de ocurrencia e impacto en el desarrollo del proyecto. **(ver anexo 2. Riesgos.xls, segunda hoja: Cuantificación)**.
- **Priorización:** se precisa una relación entre los valores de la fase anterior, para ello se utilizó la ecuación presentada a continuación. Como resultado se obtendrá el valor de cuantificación total de cada riesgo.

$$Cuantificación\ del\ riesgo = probabilidad * impacto$$

Ecuación 1 - Cuantificación del riesgo

- **Mitigación:** se generan los planes de mitigación y acción **(ver anexo 2. Riesgos.xls, cuarta hoja: Mitigación)**.

5.2.1 Priorización

En el **(anexo 2. Riesgos.xls, tercera hoja: Resumen cuantificación),** se encuentra listados los riesgos y la respectiva cuantificación obtenida según el proceso.

De acuerdo con la lista, solo se tomarán como riesgos prioritarios aquellos que tengan un resultado mayor a 8 en la cuantificación del riesgo. Estos son:

ID	Nombre riesgo	Resultado cuantificación	Plan de mitigación
1	Desarrollo incorrecto de la interfaz	12	Presentación de un modelo a los stakeholders para su aprobación antes de seguir con la programación o montaje del resto del proyecto. Dado el caso que la inconformidad se presente en etapas posteriores se dedicará un integrante del grupo a encontrar conceso con los stakeholders para retroalimentar el aspecto de la interfaz.
23	Fallas en la configuración de tecnologías	12	Recurrir a personas externas como el asesor de trabajo de grado, para solucionar los fallos con la mayor brevedad posible.

8	Cambio de los requisitos en la fase final	9	El director del proyecto revisará la viabilidad de la implementación y el coste de los cambios. Se presentará el informe al grupo de trabajo y se discutirá la posible implementación de estos cambios en el proyecto.
11	Falla en los equipos de cómputo	9	Todos los archivos, tanto de código como de documentación, cuentan con una copia en la nube (Google Drive, Sharepoint o Github). En cuanto los espacios de trabajo; se cuenta con las salas de cómputo de la Pontificia Universidad Javeriana o los computadores de la biblioteca de la Pontifica Universidad Javeriana.
13	Desconocer la tarea a realizar	8	Planificación de las tareas de acuerdo con las capacidades de cada uno de los integrantes del grupo de trabajo. Dado el caso que la tarea siga siendo desconocida por el encargado, este se verá en la obligación de pedir ayuda al resto del equipo o consultar cómo realizar dicha tarea.

Tabla 2 - Riesgos con mayor cuantificación

El proceso de monitoreo y control sobre el plan de mitigación se llevará a cabo por el líder del proyecto. El cual se comunicará constantemente con los encargados de las áreas involucradas en el plan de mitigación para asegurarse de que se están realizando las acciones necesarias para evitar que el evento suceda.

En caso de que el plan de mitigación no esté en ejecución, el líder del equipo será el encargado de ponerlo en marcha y debe informar al grupo los pasos a seguir.

5.3 Administración de la configuración

En esta sección, se describe el proceso de desarrollo, configuración y despliegue de cpPlugins 3.0.

5.3.1 Contenedores y su administración

Con el fin de facilitar el despliegue y configuración de dependencias en diferentes equipos de cómputo, se utilizarán contenedores Docker para agrupar los diferentes microservicios provistos por cpPlugins.

Este aplicativo se desplegará utilizando Kubernetes, con el objetivo de facilitar la labor de realizar despliegues automatizados y gestionar las diversas instancias. En un principio, se plantea utilizar algún servicio en la nube del orden de Google Cloud, AWS ó Azure, haciendo que las necesidades de hardware sean transparentes al usuario.

En principio, se crearán tres contenedores principales. En el primero de ellos, se encuentran los componentes necesarios para desplegar el servidor de presentación con node.js, en el segundo, se encuentran aquellos componentes necesarios para ejecutar la versión base de cpPlugins escrita en c++. Por último, en el tercer contenedor se agrega un servidor de tomcat con los microservicios provistos por cpPlugins 3.0. Esta organización, se puede ver en **La ilustración 3.**

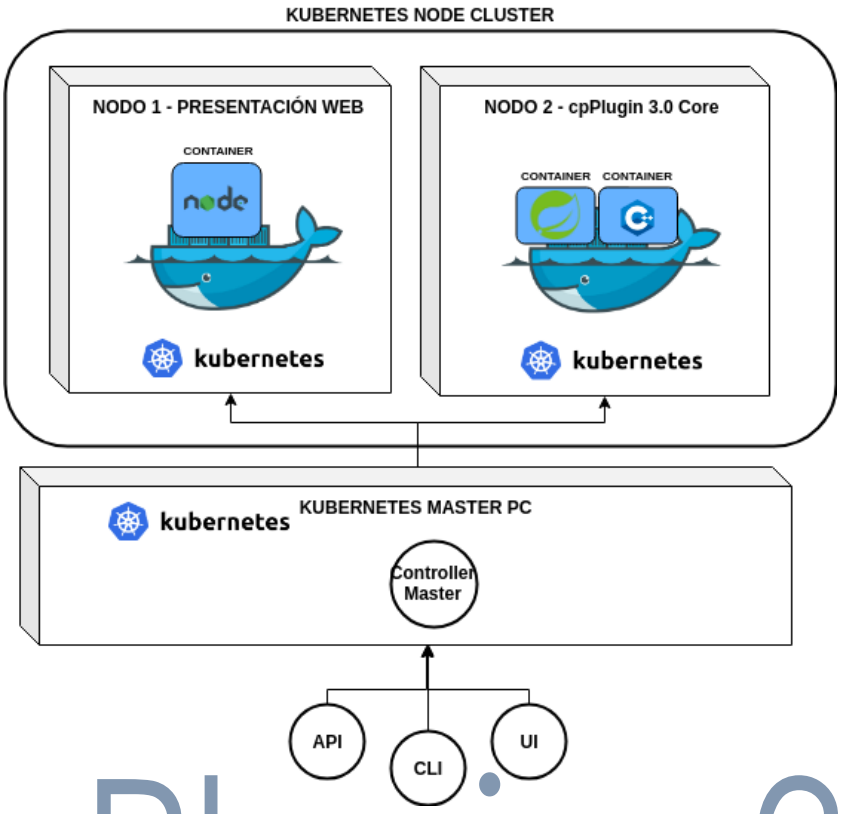

Ilustración 3 - Despliegue de cpPlugins 3.0

Adicionalmente, tal y como se evidencia en la ilustración anterior, se tiene un nodo adicional llamado "Kubernetes Master PC", este es de utilidad, para realizar el manejo de los diversos contenedores, ya que tiene comunicación directa con el cluster de cpPlugins.

5.3.2 Flujo de trabajo y versionamiento con Git

Para llevar un control claro del código y sus versiones, se utilizará un esquema de branching que permitirá tener un control de cada historia de usuario que se implementa en el código, utilizando como la guía de trabajo en git especificada en el siguiente documento anexo. **(ver anexo 1. versión final de la propuesta de cpPlugins 3.0.docx, sección 1.3 Entregables, estándares utilizados y justificación).**

El flujo empieza cuando es asignada una Historia de Usuario a un desarrollador del equipo. En ese momento, el miembro a cargo debe crear una rama en git nombrada de acuerdo al siguiente formato [No._Historia de_usuario – Nombre_Funcionalidad], por ejemplo: "HU001-FuncionalidadX".

Una vez que se ha terminado el desarrollo de la funcionalidad específica, el desarrollador a cargo debe crear un "Merge Request" de la rama y esperar la aprobación o correcciones del director de desarrollo del equipo, quien es la persona encarga de integrar las funcionalidades a la rama de desarrollo o de producción, **(ver anexo 3. Proceso de integración.jpg).** En la *ilustración 4* y en la *tabla número 4*, se puede observar la funcionalidad de cada rama y su respectivo nombre.

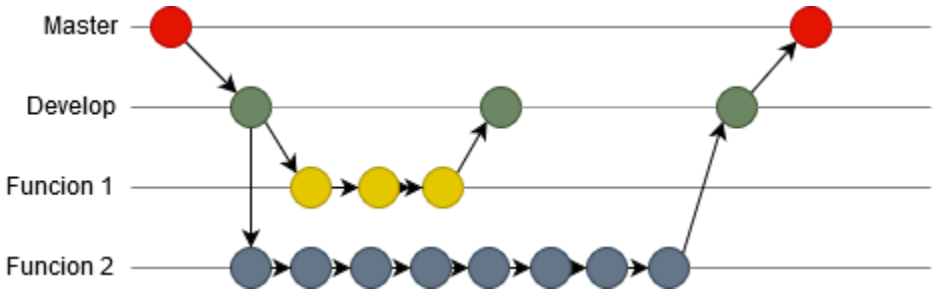

Ilustración 4 - GitFlow para el proyecto

Rama	Funcionalidad
Master	En esta rama, se encuentra el código fuente funcional y aprobado por el equipo a manera de prototipo y/o versión estable. Es la rama que debe tener menor interacción. Para facilitar el manejo de versiones, se etiquetarán estas versiones, para acceder a ellas de manera rápida.
Develop	En esta rama, se encuentra el código fuente funcional, que aún no ha sido aprobado para ser versionado, ya que incluye funcionalidades en desarrollo o que no son estables. También es utilizada a modo de ambiente de pruebas.
HUXX-Funcionalidad	En las ramas de funcionalidades, se encuentra el código especifico de una historia de usuario. Al momento de ser implementada dicha funcionalidad, la rama es integrada a la rama develop y eliminada.

Tabla 3 - Ramas a utilizar y su funcionalidad

Con respecto a la numeración de versiones, se utilizarán los tags provistos por git, para identificar puntos específicos en la "Historia" del código fuente, marcando estados del proyecto y numerándolos de acuerdo con el estándar establecido en el anexo. **(ver anexo 1.versión final de la propuesta de cpPlugins 3.0.docx, sección 1.3 Entregables, estándares utilizados y justificación).**

5.3.3 Proceso de integración continua

Para realizar el proceso de integración y despliegue continuo, se utilizará la herramienta de auto devOps provista por gitLab, donde es posible definir un pipeline de integración con diferentes tareas.

Este pipeline, se define en el archivo de configuración de gitlab *.gitlab-ci.yml*,. Este consta de tres diferentes etapas, descritas a continuación:

- **Compilación:** en esta etapa, se configuran una serie de comandos que permiten a gitlab, rectificar que el código fuente no contiene errores de compilación, de los diferentes componentes del proyecto.

- **Pruebas:** en esta etapa, se definen los diferentes tipos de pruebas que se deban realizar sobre el código y gitlab las ejecutará.
- **Despliegue:** esta es la etapa final, donde el sistema de gitlab CI, se encargará de desplegar el proyecto en kubernetes.

Estos pipelines, se ejecutarán secuencialmente en cada ocasión que se realice un "Merge Request", con el fin de mantener las ramas de despliegue y master con funcionalidades probadas. En caso de fallar cualquiera de las etapas, se revertirá la petición y la funcionalidad conflictiva debe ser reparada y no será agregada a la rama en cuestión, de otro modo, si todas las etapas se realizan de manera exitosa, la nueva funcionalidad se integrará y se desplegará. En la siguiente imagen se puede observar un ejemplo de pipeline.

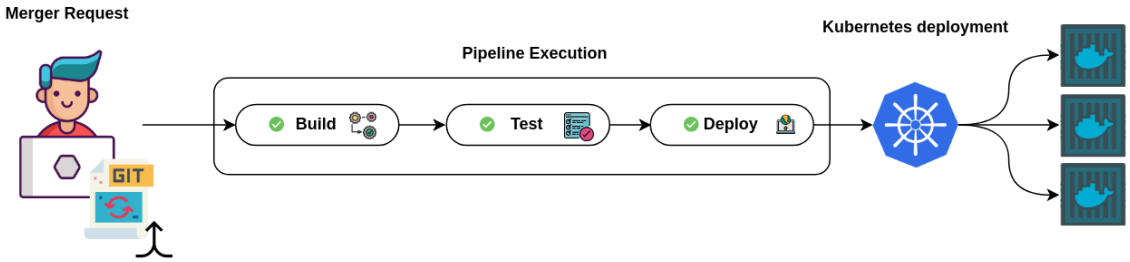

Ilustración 5 - PipeLine de ejemplo

6 Calendarización

La calendarización de los entregables especificados en la propuesta de grado se encuentran en la **Tabla 4 – Calendarización.** Aunque las fechas de los entregables de cada sprint no están especificadas en la tabla, se sabe que estos empezarán el 15/06/2019 y terminarán el 26/10/2019. Estos tendrán una duración de 1 semana.

Entregables	Fecha
SPMP	12/05/2019
SRS	23/05/2019
SDD	03/07/2019
Memoria	23/11/2019
Plan de Pruebas	03/07/2019
Reporte de Pruebas	23/11/2019
Código Fuente	26/10/2019
Producto	26/10/2019
Memoria	23/11/2019

Tabla 4 - Calendarización

7 Entrega del producto

A continuación, se presentan los criterios para la aceptación del producto, acordados entre el cliente y equipo de desarrollo. También se encuentra las actividades que se desarrollaran para

7.1 Criterios

Entregable	Criterios
Documentación (propuesta de trabajo de grado, SPMP, SRS, SDD, memoria, plan de pruebas, reporte de pruebas, manual de usuario)	• Sin errores ortográficas. • Referencias en IEEE. • Buena redacción. • La información debe ser consistente con lo dicho en todo el documento. • Cada sección debe ser autocontenida. • Las imágenes, tablas y ecuaciones deben estar correctamente numeradas y marcadas.
Código Fuente	• Sin errores de interpretación. • Sin errores de ejecución. • Código ordenado. • Implementación de los requisitos propuestos. • Estar debidamente documentado.
Producto	En el SRS se pueden encontrar con detalle los criterios que se deben cumplir para la aceptación del producto.

Tabla 5 - Criterios

7.2 Actividades

La entrega del producto se divide en 4 actividades principales

- Entrega del producto de software y documentación: el equipo de trabajo debe proporcionar el código y que incluye el producto. Además, se debe entregar una copia de la toda la documentación acordada.
- Validación del producto y revisión de la documentación: se verifica con los stakeholders que el producto cumpla con los criterios especificados y que la documentación cumpla sus objetivos. los stakeholders darán los comentarios y percepciones del producto entregado. Finalmente, comunican si el producto y la documentación son aceptados o no.
- Aprobación del producto: el equipo de desarrollo junto con los stakeholders certifica la entrega realizada.
- Plantear futuros cambios: el equipo de trabajo planteara los siguientes requisitos a ser desarrollados y agregados al producto.

8 Procesos de soporte

Esta sección contiene la especificación de los procesos transversales al proyecto que permiten soportar su ejecución, así como las reglas del trabajo en equipo y los mecanismos para asegurar su cumplimiento.

8.1 Ambiente de trabajo

Con motivo de generar un buen ambiente de trabajo, el equipo de cpPlugins 3.0 estableció un conjunto de reglas, para incentivar la productividad y garantizar una convivencia apropiada.

Se acordó, el manejo de un sistema de "strikes" o llamados de atención para el manejo de faltas, así como sanciones equivalentes a la falta cometida. Además de esto, para reconocer la excelencia del trabajo de los miembros del equipo se estableció un sistema de condecoración con estrellas. Para organizar el trabajo se utilizó un sistema de actividades con responsables y fechas límite.

Cada una de las reglas y sistemas establecidos fueron tomados y adaptados del ambiente de trabajo establecido por el equipo de Hellsoft de ingeniería de software [47], al cual pertenecían todos los integrantes de cpPlugins 3.0. A continuación se exponen las diferentes reglas y sistemas establecidos por el equipo.

8.1.1 Reglas de trabajo y convivencia en grupo

- Cada miembro del grupo debe ser respetuoso con los demás integrantes, dirigiéndose a los mismos con un trato afable.
- Cada miembro del grupo debe ser proactivo, responsable y puntual con cada actividad que se le designe, con el objetivo de realizar un trabajo de calidad.
- El principal medio de comunicación es el grupo de Whatsapp, donde cada integrante debe responder los mensajes para acusar recibido.
- Los problemas interpersonales entre miembros del equipo deben ser resueltos fuera del ejercicio profesional.
- Evitar las conversaciones que puedan desviar el interés principal del grupo.
- Evitar las conversaciones que puedan perjudicar la convivencia del grupo.
- Las decisiones cruciales sobre el proyecto deben ser sometidas a elección popular.
- La cantidad de *strikes* será un multiplicador para las acciones correctivas que se apliquen.
- La acumulación de tres faltas leves representa un strike para el individuo.

8.1.2 Reuniones

- Todos los miembros del equipo deben asistir de manera obligatoria a todas las reuniones realizadas los días que se demandan, principalmente en el horario establecido; jueves de 2:00pm a 4:00pm.
- Los miembros del equipo deben estar prestando atención de manera activa a los temas tratados en la reunión.
- El plazo máximo de llegada de un miembro a la reunión es de 15 minutos, después de esto tiempo representa una falta leve.
- La inasistencia a una reunión debe ser justificada mediante una excusa válida, y será sometida a escrutinio del equipo.
- Cada inasistencia deberá ser comunicada en un plazo máximo de 5 horas antes de la reunión.

- Todos los miembros del equipo tienen derecho a ser escuchados y sus ideas podrán ser discutidas a lo largo de la reunión.
- La inasistencia injustificada de un miembro a una reunión significa un strike más.

8.1.3 Excusas válidas de inasistencia

Para el equipo de cpPlugins 3.0, cada excusa debe ser sometida a juicio por los miembros del equipo, y solo será considerada válida si presenta alguna de las siguientes situaciones.

- Cita médica programada con anterioridad.
- Enfermedad.
- Razones de fuerza mayor comprobables como calamidades familiares.
- Eventos respaldados por entidades formales tales como universidades o estatales.

8.1.4 Faltas leves, medias y graves

Si alguno de los integrantes de equipo incurre en alguna falta, esta será clasificada de acuerdo con lo siguiente:

- **Leves:** Se considera una falta leve, aquella que no retrase el cumplimiento del proyecto por más de (1) día y no interfiera el desarrollo del proyecto.
- **Medias:** Se considera una falta media, aquella que retrase el proyecto de (2) a (4) días y sumarán un strike al miembro del equipo de manera automática.
- **Graves:** Son aquellas faltas que van en contra de la integridad de algún miembro del grupo, además de retrasar el cumplimiento del proyecto en más de (5) días de trabajo.

8.1.5 Acciones correctivas

De acuerdo con el nivel de la falta cometida, se realizará la acción correspondiente. Se debe realizar un pago simbólico, con lo cual compensar al equipo el daño generado como se describe en la tabla 5.

Leve	Media	Grave
Golosinas para compartir	Multa monetaria	Amonestación acordada con el director y el grupo

Tabla 6 - Pagos simbólicos

8.1.6 Sistema de reconocimiento a la excelencia

Para reconocer el excelente desempeño de los miembros del equipo, se estableció un sistema de condecoración con estrellas, en donde los motivos para ser merecedor de estas son los siguientes:

- Ayudar a otros integrantes a completar sus tareas si lo requieren.
- Adelantar de manera significativa trabajo del siguiente sprint.

Cuando un integrante del equipo complete tres estrellas será recompensado por los demás con un pago simbólico no monetario.

8.2 Control de Calidad

Para el control de calidad del producto se tienen en cuenta las siguientes subsecciones:

- **Calidad de los documentos.**
- **Calidad del producto.**

8.2.1 Calidad de los Documentos

Para la realización del control de calidad de los documentos se debe tener en cuenta lo explicado en la **sección 4.1 Modelo de ciclo de vida.** Allí se menciona la utilización de tableros Trello, los cuales ayudan a llevar la trazabilidad de las secciones realizadas.

La validación de la calidad inicia cuando una tarea es agregada a la columna *Revisión de Calidad*. Esta comprobación se basa en los criterios dichos en la sección **6. *Entrega del producto*** y será ejecutada por el área de calidad.
Finalmente, si la sección es aprobada, la tarea se moverá a la columna *Terminado*. De lo contrario, las correcciones requeridas serán resaltadas con los siguientes colores y será trasladada nuevamente a la columna *Tareas*.

- Amarillo: El texto necesita mejoras de redacción y/o ortografía.
- Verde: El texto tiene mucha información o tiene información redundante.
- Fucsia: El texto está incompleto. Expandir con más información.
- Gris: Las ideas presentadas no se entienden y/o no tienen una coherencia adecuada. La lectura es poco fluida. Es necesario organizar las ideas para conectarlas de mejor forma.
- Tierra: Frases muy largas sin puntos seguidos. Descomponer en frases más cortas separadas por puntos seguidos

El código de colores fue tomado del curso virtual *Planeacion del Proyecto Final 031339_1910_*2486 de Blackboard.

8.2.2 Calidad del producto

En esta subsección se especificará el proceso de validación de la calidad del producto.

8.2.2.1 Tipos de Pruebas

Para este proyecto se emplearán los siguientes tipos de pruebas:

- **Pruebas Funcionales:** tienen la finalidad de evaluar las funciones que el producto debería realizar. [48] La medición empleada para este proyecto será el porcentaje de requisitos funcionales que han pasado las pruebas.
- **Pruebas No Funcionales:** este tipo de prueba evalúan los siguientes atributos de calidad el rendimiento, la compatibilidad, la usabilidad, la confiabilidad, entre otros. [48] La medición empleada para este proyecto dependerá del atributo de calidad probado. Porque pueden llegar a ser tan diferentes, que no pueden ser agrupadas en una sola medida.

8.2.2.2 Niveles de Pruebas

Para este proyecto se realizarán los siguientes dos niveles, prueba de componentes y de integración de componentes. El nivel de componente, también conocido como pruebas unitarias, se enfoca en los componentes que se pueden probar por separado. El nivel de integración de componentes se centra en las interacciones e interfaces entre componentes del producto.

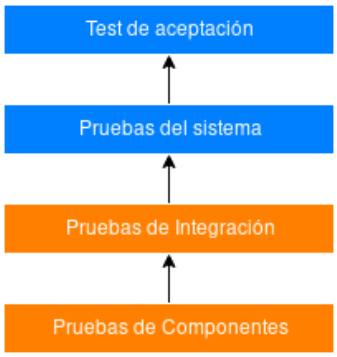

Ilustración 2 - Niveles de pruebas

Solo se tomaron estos dos niveles, por el tiempo del proyecto. Dado que solo se tiene cuatro meses para el desarrollo del producto.

8.2.2.3 Diseño de Pruebas

Para el diseño de pruebas se utilizará la técnica Black-Box, esta se concentra en las entradas y salidas del objeto de prueba sin hacer referencia a su estructura interna. [48] Para emplear esta técnica se usarán los siguientes elementos:

- **Tabla de Decisiones:** identifican las condiciones (a menudo entradas) y las acciones resultantes (a menudo salidas) del sistema. [48]
- **Casos de Uso:** describe las interacciones y actividades, así como las condiciones previas, las condiciones posteriores y el lenguaje natural, según corresponda. [48]

8.2.2.4 Ejecución de Pruebas

Las pruebas se realizarán tanto manuales como automatizadas. Porque la automatización de pruebas da una mejor ganancia a largo plazo [49], en este proyecto de corto plazo no tiene sentido tener el 100% de las pruebas automatizadas.

La funcionalidad y algunos atributos de calidad del sistema serán probados en su totalidad con pruebas manuales. Por ejemplo, la usabilidad será probada manualmente por los miembros del proyecto. Otros atributos de calidad como la extensibilidad serán probados automáticamente por la integración continua que se empleará en el proyecto.

9 Anexos

9.1 Versión final de la propuesta

Archivo Word que contiene la información correspondiente a la versión final de la propuesta de cpPlugins 3.0. Anexos\1.Propuesta_cpPlugins3.0.docx

9.2 Riesgos

Archivo Excel que contiene los resultados y planes de mitigación para los riesgos posibles durante el desarrollo del proyecto cpPlugins 3.0. Anexos\2.Riesgos.xlsx

9.3 Proceso de integración

Imagen png que muestra el proceso de integración de una funcionalidad realizada. Anexos\3.Procesodeintegración.png

cpPlugins 3.0

10 Referencias

[1] «Interfaz de usuario - EcuRed». [En línea]. Disponible en: https://www.ecured.cu/Interfaz_de_usuario. [Accedido: 05-may-2019].

[2] «MODELO DE ARQUITECTURA PIPELINE - Electrónica analogica y digital». [En línea]. Disponible en: https://www.electrontools.com/Home/WP/2018/04/18/modelo-de-arquitectura-pipeline/. [Accedido: 05-may-2019].

[3] «Glosario de Terminos en Project Management (PMI). Primera Parte». [En línea]. Disponible en: https://uv-mdap.com/blog/glosario-terminos-project-management-pmi/. [Accedido: 05-may-2019].

[4] «Qué es SCRUM – Proyectos Ágiles». [En línea]. Disponible en: https://proyectosagiles.org/que-es-scrum/. [Accedido: 05-may-2019].

[5] «Significado de Stakeholder (Qué es, Concepto y Definición) - Significados». [En línea]. Disponible en: https://www.significados.com/stakeholder/. [Accedido: 05-may-2019].

[6] «PMI – Glossary of Project Management terms». [En línea]. Disponible en: http://www.pmgloss.com/about/. [Accedido: 05-may-2019].

[7] «Qué es la programación orientada a objetos – Styde.net». [En línea]. Disponible en: https://styde.net/que-es-la-programacion-orientada-a-objetos/. [Accedido: 05-may-2019].

[8] «What is REST? | Codecademy». [En línea]. Disponible en: https://www.codecademy.com/articles/what-is-rest. [Accedido: 05-may-2019].

[9] «(PDF) Qué es una herramienta CASE | Pedro Beltran - Academia.edu». [En línea]. Disponible en: https://www.academia.edu/28037284/Qu%C3%A9_es_una_herramienta_CASE. [Accedido: 05-may-2019].

[10] «Estándar BPMN 2.0 para modelamiento de procesos y mucho más». [En línea]. Disponible en: https://www.bizagi.com/es/productos/beneficios/estandares. [Accedido: 05-may-2019].

[11] «VTK - The Visualization Toolkit». .

[12] «ITK - Segmentation & Registration Toolkit». [En línea]. Disponible en: https://itk.org/. [Accedido: 10-feb-2019].

[13] «Programación en C++/Introducción - Wikilibros». [En línea]. Disponible en: https://es.wikibooks.org/wiki/Programaci%C3%B3n_en_C%2B%2B/Introducci%C3%B3n. [Accedido: 05-may-2019].

[14] «Qué es Spring framework | OpenWebinars». [En línea]. Disponible en: https://openwebinars.net/blog/que-es-spring-framework/. [Accedido: 05-may-2019].

[15] «¿Qué es Java y para qué es necesario?» [En línea]. Disponible en: https://www.java.com/es/download/faq/whatis_java.xml. [Accedido: 05-may-2019].

[16] «Introducción a Java Native Interface - Geeky Theory». [En línea]. Disponible en: https://geekytheory.com/introduccion-a-java-native-interface. [Accedido: 05-may-2019].

[17] C. By-Nc-Nd, «Creative Commons License Deed», p. 249.

[18] «CSS | MDN». [En línea]. Disponible en: https://developer.mozilla.org/es/docs/Web/CSS. [Accedido: 05-may-2019].

[19] «What is JavaScript? - Aprende sobre desarrollo web | MDN». [En línea]. Disponible en: https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/Qu%C3%A9_es_JavaScript. [Accedido: 05-may-2019].
[20] «React – A JavaScript library for building user interfaces». [En línea]. Disponible en: https://reactjs.org/. [Accedido: 05-may-2019].
[21] «vtk.js». [En línea]. Disponible en: https://kitware.github.io/vtk-js/. [Accedido: 05-may-2019].
[22] «Draw2D touch». [En línea]. Disponible en: http://www.draw2d.org/draw2d/home.html. [Accedido: 05-may-2019].
[23] «Bootstrap · The most popular HTML, CSS, and JS library in the world.» [En línea]. Disponible en: https://getbootstrap.com/. [Accedido: 05-may-2019].
[24] «¿Qué es Docker?» [En línea]. Disponible en: https://www.redhat.com/es/topics/containers/what-is-docker. [Accedido: 05-may-2019].
[25] «Production-Grade Container Orchestration - Kubernetes». [En línea]. Disponible en: https://kubernetes.io/. [Accedido: 05-may-2019].
[26] «Acerca | Node.js». [En línea]. Disponible en: https://nodejs.org/es/about/. [Accedido: 05-may-2019].
[27] «Apache Tomcat® - Welcome!» [En línea]. Disponible en: http://tomcat.apache.org/. [Accedido: 05-may-2019].
[28] «Auto DevOps | GitLab». [En línea]. Disponible en: https://docs.gitlab.com/ee/topics/autodevops/. [Accedido: 05-may-2019].
[29] «Git - Fundamentos de Git». [En línea]. Disponible en: https://git-scm.com/book/es/v1/Empezando-Fundamentos-de-Git. [Accedido: 05-may-2019].
[30] «User documentation | GitLab». [En línea]. Disponible en: https://docs.gitlab.com/ee/user/index.html. [Accedido: 05-may-2019].
[31] «What is UML | Unified Modeling Language». [En línea]. Disponible en: https://www.uml.org/what-is-uml.htm. [Accedido: 05-may-2019].
[32] «Visual Paradigm - EcuRed». [En línea]. Disponible en: https://www.ecured.cu/Visual_Paradigm. [Accedido: 05-may-2019].
[33] «Bonita es una plataforma de aplicaciones basada en BPM». [En línea]. Disponible en: https://es.bonitasoft.com/business-process-management-bpm. [Accedido: 05-may-2019].
[34] «Microsoft Office | Herramientas para el hogar o la oficina». [En línea]. Disponible en: https://products.office.com/es-co/home. [Accedido: 05-may-2019].
[35] «¿Cuál es la explicación más corta y comprensible de qué es SharePoint? | Soluciones SharePoint». [En línea]. Disponible en: https://www.soluciones-sharepoint.com/2012/04/cual-es-la-explicacion-mas-corta-y.html. [Accedido: 05-may-2019].
[36] «Visita guiada de Trello». [En línea]. Disponible en: https://trello.com/tour. [Accedido: 05-may-2019].
[37] «¿Qué es Microsoft Teams? | Soluciones SharePoint». [En línea]. Disponible en: https://www.soluciones-sharepoint.com/2017/09/microsoft-teams.html. [Accedido: 05-may-2019].
[38] «What is a Product Owner?» [En línea]. Disponible en: https://www.scrum.org/resources/what-is-a-product-owner. [Accedido: 05-may-2019].

[39] «What is a Scrum Master?» [En línea]. Disponible en: https://www.scrum.org/resources/what-is-a-scrum-master. [Accedido: 05-may-2019].
[40] «php architect's Guide to Enterprise PHP Development - PDF Free Download». [En línea]. Disponible en: https://epdf.tips/php-architects-guide-to-enterprise-php-development.html. [Accedido: 05-may-2019].
[41] «Qué es un desarrollador y por qué necesitas uno». [En línea]. Disponible en: https://www.qualitydevs.com/2017/12/21/que-es-un-desarrollador/. [Accedido: 05-may-2019].
[42] «ARQUITECTO DE SOFTWARE». [En línea]. Disponible en: https://www.cessi.org.ar/perfilesit/detalle-de-arquitecto-de-software-3. [Accedido: 05-may-2019].
[43] «Emotion & Design - jnd.org». [En línea]. Disponible en: https://jnd.org/tag/emotion_design/. [Accedido: 05-may-2019].
[44] X. H. Moriones, «Administración de la Configuración», p. 35.
[45] «3.5. Análisis de riesgos - 2015_06_gps_Ulloa_rodolfo». [En línea]. Disponible en: https://sites.google.com/site/201506gpsulloarodolfo/3-5-analisis-de-riesgos. [Accedido: 05-may-2019].
[46] «Pmbok 5 edición». [En línea]. Disponible en: https://es.slideshare.net/miguelsarer95/pmbok-5-edicin. [Accedido: 05-may-2019].
[47] «Dropbox - HellSoft - Simplifica tu vida». [En línea]. Disponible en: https://www.dropbox.com/sh/okfmn3ojob248ww/AADoyqrJANzgfRNW0wEDWcFua/1830/SPMP/HellSoft?dl=0&preview=SPMP.docx&subfolder_nav_tracking=1. [Accedido: 05-may-2019].
[48] «208-ctfl-2018-syllabus.html..pdf». .
[49] «Integration Testing: What is, Types, Top Down & Bottom Up Example». [En línea]. Disponible en: https://www.guru99.com/integration-testing.html. [Accedido: 05-may-2019].