TEC | Tecnológico de Costa Rica

Área Académica de Administración de Tecnologías de Información

Propuesta de Diseño de Software de Alto Nivel de un Sistema de Generación de Lenguaje Natural para la Empresa Kleeen Software

Trabajo Final de Graduación para optar al grado de Licenciatura en Administración de Tecnología de Información

Elaborado por: Laura Daniela Carvajal Segura

Prof. Tutor : MSc. Jacqueline Solís Céspedes

Cartago, Costa Rica

I Semestre

Junio, 2022

INSTITUTO TECNOLÓGICO DE COSTA RICA

ÁREA ACADÉMICA DE ADMINISTRACIÓN DE TECNOLOGÍAS DE INFORMACIÓN

GRADO ACADEMICO: LICENCIATURA

Los miembros del Tribunal Examinador del Área Académica de Administración de Tecnologías de Información recomendamos que el siguiente Trabajo Final de Graduación de la Estudiante Laura Daniela Carvajal Segura, sea aceptado como requisito parcial para optar al grado académico de Licenciatura en Administración de Tecnología de Información.

JACQUELINE TATIANA SOLIS CESPEDES (FIRMA)
Firmado digitalmente por JACQUELINE TATIANA SOLIS CESPEDES (FIRMA)
Fecha: 2022.06.09 16:22:27 -06'00'

MSc. Jacqueline Solís Céspedes
Profesora Tutora

LUIS CARLOS NARANJO ZELEDON (FIRMA)
Firmado digitalmente por LUIS CARLOS NARANJO ZELEDON (FIRMA)
Fecha: 2022.06.09 12:12:05 -06'00'

Dr. Luis Naranjo Zeledón
Lector de Industria

TEC | Tecnológico de Costa Rica
Firmado digitalmente por MARIA JOSE ARTAVIA JIMENEZ (FIRMA)
Fecha: 2022.06.09 11:19:43 -06'00'

Ing. María José Artavia Jiménez, MBA.
Lector de Academia

TEC | Tecnológico de Costa Rica
Firmado digitalmente por YARIMA TATIANA SANDOVAL SANCHEZ (FIRMA)
Fecha: 2022.06.21 18:31:05 -06'00'

Ing. Yarima Sandoval Sánchez, MBA.
Coordinadora Trabajo Final de Graduación

Dedicatoria

A mi mamá, mi papá, mis hermanas y mi sobrina, por brindarme todo su apoyo a lo largo de estos años, también por su acompañamiento, ya que este fue vital para que lograra culminar con este proceso, y así cumplir mis metas.

Agradecimientos

A mi familia por creer en mí y darme la posibilidad de estudiar.

A Kleeen Software y a Mario mi jefe, por darme la oportunidad de trabajar en la empresa y apoyarme siempre en mi proceso académico.

A mis compañeros Hazel, Edwin, Óscar C, Alejandro H, Priscilla R, Dayana, Priscilla G, quienes me acompañaron y con quienes compartí grandes momentos.

A Néstor y Jacky que me guiaron y apoyaron increíblemente en la realización de este proyecto.

A Karla, Christian C con quienes viví mucho tiempo y quienes siempre estuvieron ahí, haciéndome reír y apoyándome a ser mejor.

Resumen

Carvajal, Laura. (2022). Propuesta de Diseño de Software de Alto Nivel de un Sistema de Generación de Lenguaje. (Trabajo Final de Graduación). Área Académica de Administración de Tecnologías de Información. Instituto Tecnológico de Costa Rica.

Este proyecto tiene como objetivo proponer un diseño de software de alto nivel de un sistema de generación de lenguaje natural, basado en las buenas prácticas de la industria, para la generación de documentación que brinde el contexto y entendimiento necesario en caso de una futura implementación e integración de este nuevo sistema con Kleeen Software IDE, con el fin de solventar la necesidad presentada, en la empresa Kleeen Software, durante el primer semestre del 2022.

El estudio se basó en una metodología aplicada de enfoque cualitativo, para esto se realizaron revisiones documentales de las fuentes primarias, así como entrevistas no estructuradas, y observación por medio de encuestas de Likert, para obtener y entender las características y requerimientos técnicos, así como recopilar información sobre los sistemas de la empresa.

La investigación concluyó que es necesario un diseño de un sistema de generación de lenguaje natural que permita brindar respuestas contextualizadas, e inteligentes al usuario final, basándose en su contexto e intenciones.

Por medio de las entrevistas no estructuradas se recopilaron los requerimientos iniciales para realizar el diseño del sistema.

Se propone un diseño de software de alto nivel de un sistema de generación de lenguaje como base sólida para una futura implementación dentro de la empresa, como resultado de este proyecto.

Laura Daniela Carvajal Segura

Palabras clave: Diseño de Software, Arquitectura de Software, Generación de Lenguaje Natural, Sistemas de Información.

Abstract

Carvajal, Laura. (2022). High Level Software Design Proposal for a Language Generation System. (End of Degree Project). Academic Area of Information Technology Administration. Technological Institute of Costa Rica.

The objective of this project is to propose a high-level software design of a natural language generation system, based on good industry practices, for the generation of documentation that provides the necessary context and understanding in case of a future implementation and integration of this new system with Kleeen Software IDE, to solve the need presented, in the company Kleeen Software, during the first semester of 2022.

The study was based on an applied methodology of qualitative approach, for which documentary reviews of the primary sources were carried out, as well as unstructured interviews, and observation through Likert surveys, to obtain and understand the characteristics and technical requirements, as well as collect information about company systems.

The research concluded that it is necessary to design a natural language generation system that allows providing contextualized and intelligent responses to the end user, based on their context and intentions.

Through unstructured interviews, the initial requirements were collected to carry out the design of the system.

As a result of this project, a high-level software design of a language generation system is proposed as a solid base for a future implementation within the company.

Laura Daniela Carvajal Segura

Keywords: Software Design, Software Architecture, Natural Language Generation, Information Systems.

Tabla de contenido

Índice de Figuras

Índice de Tablas

1. Introducción

En este trabajo se propone realizar un diseño de un sistema de generación de lenguaje natural, el cual será un componente parte del producto en desarrollo por la empresa Kleeen Software.

1.1. Descripción General

Actualmente, la empresa Kleeen Software se dedica al diseño y desarrollo de un sistema que permite la automatización al momento de la creación de aplicaciones basadas en las intenciones y visión de usuario. Parte de la generación de este sistema es configurada por los mismos desarrolladores, y al ser un producto que genera aplicaciones basadas en las intenciones del cliente, existen necesidades con respecto a la forma en la que el sistema se comunica con el usuario, tal como la generación de lenguaje natural.

Aunque la empresa cuenta con una arquitectura de *software* establecida, así como el propio diseño del sistema general, no existe ningún diseño para la implementación de un sistema que permita recibir datos puros y los convierta en información entendible por el ser humano.

De acuerdo con lo anterior, el presente proyecto busca proponer el diseño de alto nivel para un sistema de generación de lenguaje natural, el cual permite manipular datos del *back-end*, de manera que el cliente tenga una mayor claridad sobre la información almacenada y lo que significa esta para su negocio. Esto da como resultado una mayor eficiencia al momento de procesar la información, además de una mejor comprensión por parte de los clientes cuando utilizan la herramienta.

Por otro lado, el presente informe se encuentra divido por secciones, las cuales buscan un desarrollo limpio y claro, con el fin de brindar un mayor entendimiento por parte del lector. El documento se presenta de la siguiente manera: en la sección segunda se expone el contexto organizacional donde se va a desarrollar el proyecto final de graduación; en la sección tercera se presenta y describe la problemática o necesidad que se busca resolver con el presente proyecto; por último, en la sección final se definen los objetivos del proyecto, se delimita el alcance por ejecutar, así como las limitaciones, supuestos y exclusiones por considerar.

Adicionalmente se incluyen las secciones de apéndices, anexos, glosario y la bibliografía consultada.

1.2. Antecedentes

En este apartado se desglosan aspectos relacionados con la organización donde se realizará el presente proyecto, esto con el objetivo de conocer la naturaleza de este y contextualizar al lector.

1.2.1. Descripción de la organización

A continuación, se brinda una descripción general de la organización, se abarcan aspectos como una descripción general de la empresa, los inversores, la misión, visión, valores, y el equipo de trabajo.

1.2.1.1. La Organización

La Empresa Kleeen Software es un *start-up* que opera en el país desde Julio del 2019, tiene sus sedes en la región de la Bahía de California, Austin TX y Costa Rica. La empresa principalmente trabaja en el área de desarrollo web y automatización.

Uno de los objetivos que posee esta empresa es el facilitar y brindar accesibilidad a todas aquellas empresas o personas que necesiten crear un sistema de inteligencia empresarial para su negocio, de manera que, sin código, se logre una combinación y visualización de datos que permita la toma de decisiones empresariales basadas en conocimiento. Kleeen es una solución de automatización de *front-end* que genera y proporciona, a través de relaciones públicas, la interfaz de usuario, la experiencia de usuario y el código de infraestructura para crear aplicaciones web personalizadas de "conocimiento primero".

La plataforma de Kleeen Software automatiza el diseño y el desarrollo de *front-end* para aplicaciones comerciales con uso intensivo de datos que utilizan las mejores prácticas de *User Interface (UI) y User Experience (UX)*. El equipo de trabajo de Kleeen Software tiene una amplia experiencia en el diseño y desarrollo de interfaces y experiencias de usuario, y debido a esto, la empresa reconoce la oportunidad de reducir drásticamente el tiempo y los recursos necesarios para crear una automatización efectiva de UX / UI (Kleeen Software, 2021).

Kleeen ayuda y amplifica a los desarrolladores de *front-end* y *full stack* al automatizar las funciones de visualización procesable de mejores prácticas con una infraestructura segura. Construido sobre cualquier marco de UI / UX, como REACT, Angular y PHP, Kleeen permite al cliente dedicar su tiempo a innovar para diseñar y construir experiencias creativas.

A diferencia de otras herramientas de diseño sin código, Kleeen ofrece especificaciones de API y códigos limpios, legibles y totalmente exportables que permiten una fácil integración a fuentes de datos y sistemas *back-end*, lo que resulta en menores costos y reduce el tiempo de desarrollo a semanas en lugar de meses.

1.2.1.2. Inversores

Kleeen Software cuenta con inversores como: AI Fund, First Rays Venture Partners, Neotribe Ventures, Silicon Valley Data Capital y WestWave Capital.

1.2.1.3. Misión

Potenciar la innovación de productos al permitir que cualquier persona lleve al mercado una interfaz de usuario de producto asombrosa y escalable, sin atascarse en las transferencias diarias propensas a errores entre el producto, el diseño y la ingeniería (Kleeen Software, 2021).

1.2.1.4. Visión

Ser una organización líder e innovadora, reconocida como tal por nuestros clientes, por medio del desarrollo de sistemas integrados, automatizados e innovadores, esto al capacitar y mantener colaboradores competentes y cómodos en un ambiente laboral sano, inclusivo, abierto y de confianza, que incite a la innovación y mejora continua, permitiéndonos ofrecer ingeniería, proyectos y servicios especializados de alta calidad (Kleeen Software, 2021).

1.2.1.5. Nuestras tres "E"

De acuerdo con la filosofía de la empresa, las "E" en su nombre representan los tres principios de Kleeen Software, los cuales son:

- Visualizar *(Envision)*: consiste en comenzar por las necesidades del cliente, trabajando como piensan los humanos, de manera que los productos generados sean un éxito para el usuario.
- Comprometerse *(Engage)*: en Kleeen Software el compromiso es un punto clave, por lo que la vista previa, la validación y el *testing* son un punto importante para el desarrollo interno y externo.
- Exportar *(Export)*: siempre se busca la forma en la cual el cliente se sienta satisfecho y feliz, por esto, el código de la empresa será exportable a repositorios Git, de manera que el cliente tenga acceso total a su aplicación.

1.2.1.6. Valores

Según la página oficial de Kleeen Software, los valores de la empresa son los siguientes:

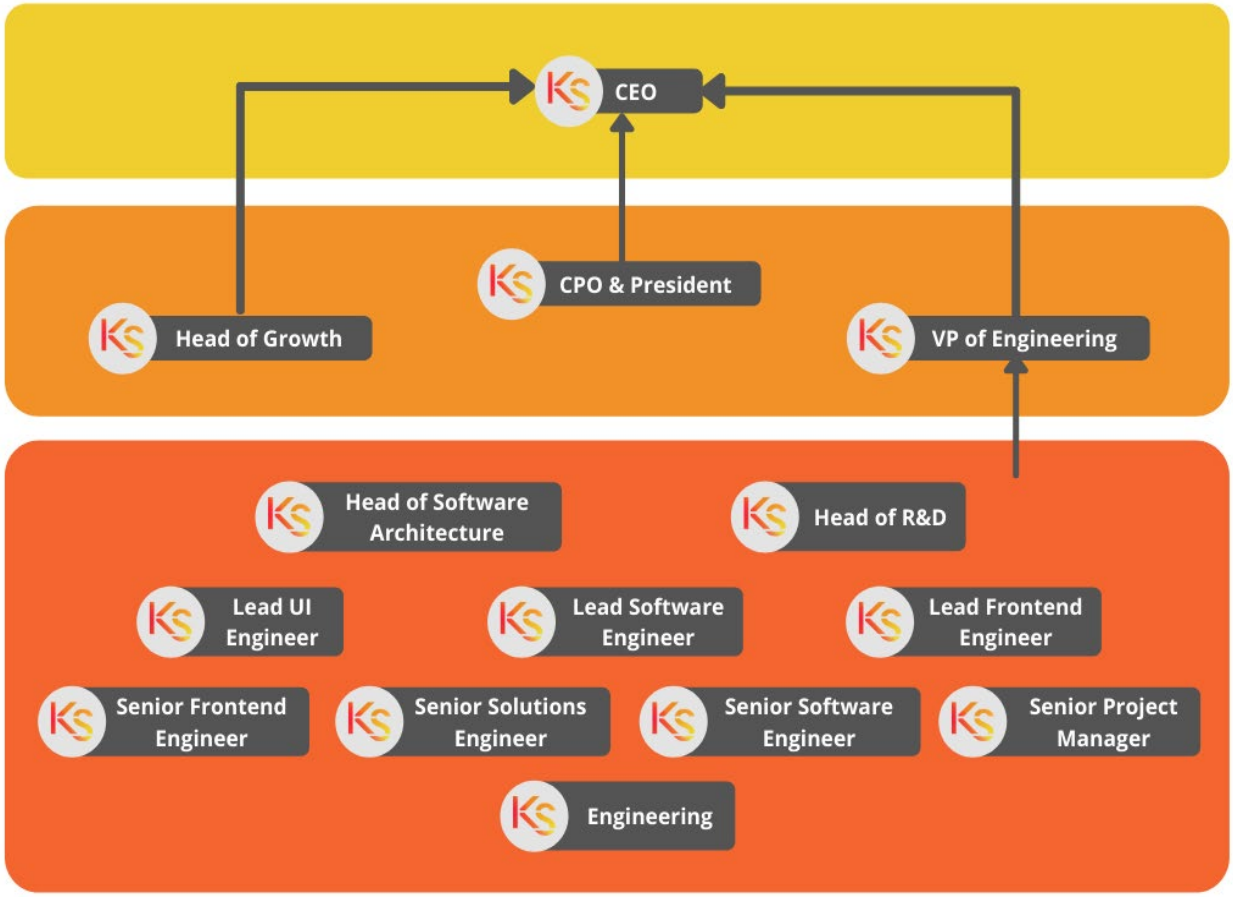
- **Comunicación:** no solo términos BS, sino que todo debe ser claro y factible.
- **Transparencia:** todos en la empresa deben saberlo todo. Ningún tema está fuera del límite de arriba a abajo.
- **Calle de dos vías:** si recibe una pregunta de alguien, debe comprender completamente la pregunta antes de comenzar a trabajar en ella. El autor de la pregunta debe dedicar tiempo a aclararla y resolver todas y cada una de las ambigüedades que puedan existir.

- **Inclusión:** debe tener en cuenta cómo se habla entre miembros del equipo, todos los empleados son bienvenidos. Nunca se debe tener miedo de hablar o mencionar cualquier aspecto de respeto o comunicación que le gustaría mejorar.
- **No está aquí por el dinero:** la empresa se asegurará de que todos reciban un pago y un trato justo... ninguno de estos debe ser un motivo por el que deba irse. Debería estar en la empresa porque el trabajo es increíble y está creciendo. Si no es así, dígaselo a alguien.
- **Compañía moral:** se toman "buenas" decisiones, decisiones éticas de arriba hacia abajo.

1.2.1.7. Equipo de trabajo

En la **Figura 1**, se presenta la estructura organizativa, la cual consta de un *CEO*, los roles de *CPO & President* (Director de Producto y Presidencia), *Head of Growth* (Director de crecimiento), y *VP of Engineering* (Vicepresidencia de Ingeniería), siendo estos los roles principales de gestión dependiendo de las líneas de acción definidas:

Figura 1:
Estructura Organizativa de Kleeen Software.



Nota. Fuente: Elaboración propia.

1.2.2. Trabajos similares realizados dentro y fuera de la organización

En esta sección se describen los trabajos que se han realizado dentro de la organización y que se encuentran relacionados con el tema de administración de procesos de negocio en el proceso de gestión de la innovación.

1.2.2.1. Proyectos de la organización

Dentro de los proyectos que se han realizado dentro de la organización se encuentran:

1.2.2.1.1. Kleeen Software IDE:

Kleeen Software IDE aprovecha una interfaz similar a un asistente para construir los elementos de rutina de la aplicación. Este se enfoca en construir la interfaz de usuario y la experiencia para las aplicaciones actuales centradas en datos, lo que permite a los diseñadores y desarrolladores del cliente concentrarse en los elementos personalizados de una aplicación.

Por otro lado, Kapitan es el producto interno de desarrollo impulsado por Kleeen Software. Este es un sistema generado por sí mismo utilizando la tecnología de *Kleeen Software Authoring,* donde, una vez lanzado y actualizado el producto por medio de la automatización llamada *"inception",* el equipo de ingeniería sólo se concentra en las funcionalidades específicas necesarias internamente por la compañía.

1.2.2.2. Proyectos externos

Dentro de los proyectos que se han realizado externos a la empresa Kleeen Software, los cuales se pueden usar como un insumo para el proyecto, se encuentran:

1.2.2.2.1. Smart Compose: Using Neural Networks to Help Write Emails

Es una función en los sistemas de Gmail pertenecientes a Google, el cual utiliza aprendizaje automático para ofrecer interactivamente sugerencias de finalización de oraciones a medida que el usuario escribe. Basado en la tecnología desarrollada para *Smart Reply, Smart Compose* ofrece una nueva forma para ayudar al momento de redactar mensajes, ya sea que se esté respondiendo a un correo electrónico entrante o redactando uno nuevo desde cero.

1.2.2.2.2. Gabriele

Gabriele redacta un millón de noticias al mes para 25 medios de todo el mundo, entre los que se encuentra el *Wall Street Journal*. Este sistema consiste en el desarrollo de contenidos destinados a la audiencia y los clientes en áreas como los deportes, las finanzas, la meteorología, los datos estadísticos, los sondeos o resultados políticos; pero aprovechará la inteligencia artificial para procesar datos, tanto de fuentes estructuradas como de internet y otras plataformas. De esta forma, Gabriele ofrece a los periodistas y editores material seleccionado, ordenado y jerarquizado sobre el cual trabajar para construir contenidos de alto valor añadido.

1.3. Planteamiento del problema

En esta sección, se describe la situación de problemática hallada dentro del entorno de la organización, la cual motiva el desarrollo del proyecto. Además, se mencionan los beneficios esperados.

1.3.1. Situación problemática

Actualmente, la empresa Kleeen Software se dedica al diseño y desarrollo de un sistema que permite la automatización para la creación de aplicaciones de gestión de información, las cuales se basan en las intenciones y visión del cliente, este se conoce como Kleeen Software IDE.

De acuerdo con una reunión realizada con el vicepresidente de ingeniería de la empresa (ver **Apéndice I.a**), y al aplicar una encuesta para conocer las inquietudes o problemas actuales (ver **Apéndice H.a**), se logra comprender cuál es la situación problemática actual.

El sistema se encuentra automatizado en su mayoría, tanto al momento de la creación y codificación, como al momento de generar productos terminados con respecto a la configuración del cliente. Esta automatización es configurada por los mismos desarrolladores, sin embargo, aún existen vacíos al manipular la información tomada del *back-end* al momento de mostrarla al usuario final, de manera que los datos no siempre están contextualizados en la visión de este, provocando así confusión por parte de este al no lograr comprender la información o el significado de los datos mostrados por el sistema.

Correspondiente a la naturaleza del producto que desarrolla la empresa Kleeen Software, el usuario final no siempre será un desarrollador o una persona con conocimientos en informática, por esto, cabe suponer que aquellas personas que requieren utilizar el sistema para realizar consultas o manipular los datos buscan una forma sencilla y eficiente de comunicarse.

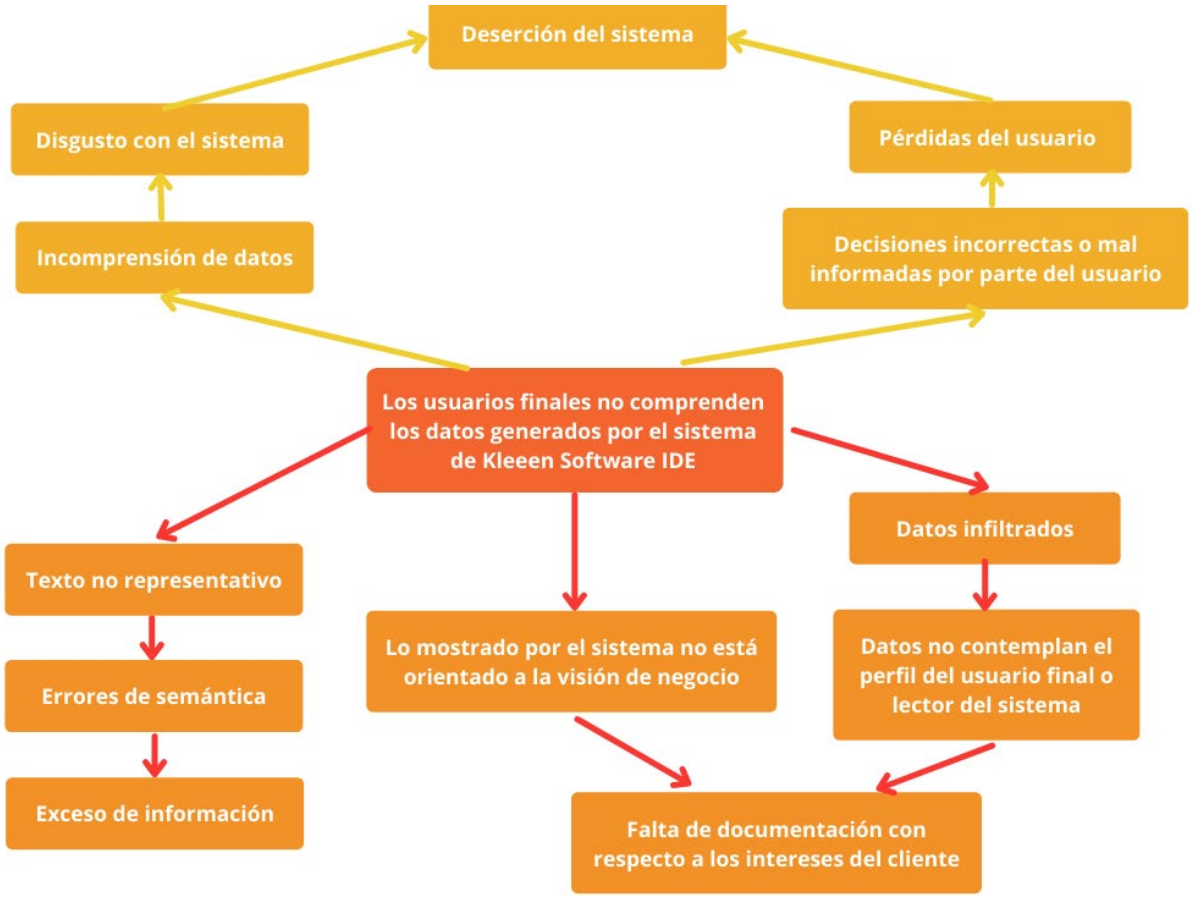
Aunque la empresa cuenta con una arquitectura de *software* establecida y un diseño de *software* para el sistema actual Kleeen Software IDE, no existe ningún diseño para la implementación de un sistema que permita recibir datos no procesados y los convierta a información entendible por el ser humano, así como algún sistema que permita corregir errores de semántica, representatividad de los textos, infiltraciones, y otros problemas que se presentan.

Es decir, se necesita un mecanismo que brinde puntos de acceso a datos con un contexto para compartir información de una manera fácil de entender para los humanos. En este sentido, los sistemas de generación de lenguaje natural pueden apoyar la contextualización de los resultados del sistema antes mencionado basándose en el contexto y las intenciones del usuario final.

La falta de un diseño integrado de generación de lenguaje natural puede solventar, al menos en forma parcial, la problemática actual de la empresa por lo que se presenta a continuación un diagrama de "Árbol del problema" y un diagrama de Ishikawa.

A continuación, en la **Figura 2**. se ilustran de manera clara las causas y consecuencias de la problemática presentada anteriormente:

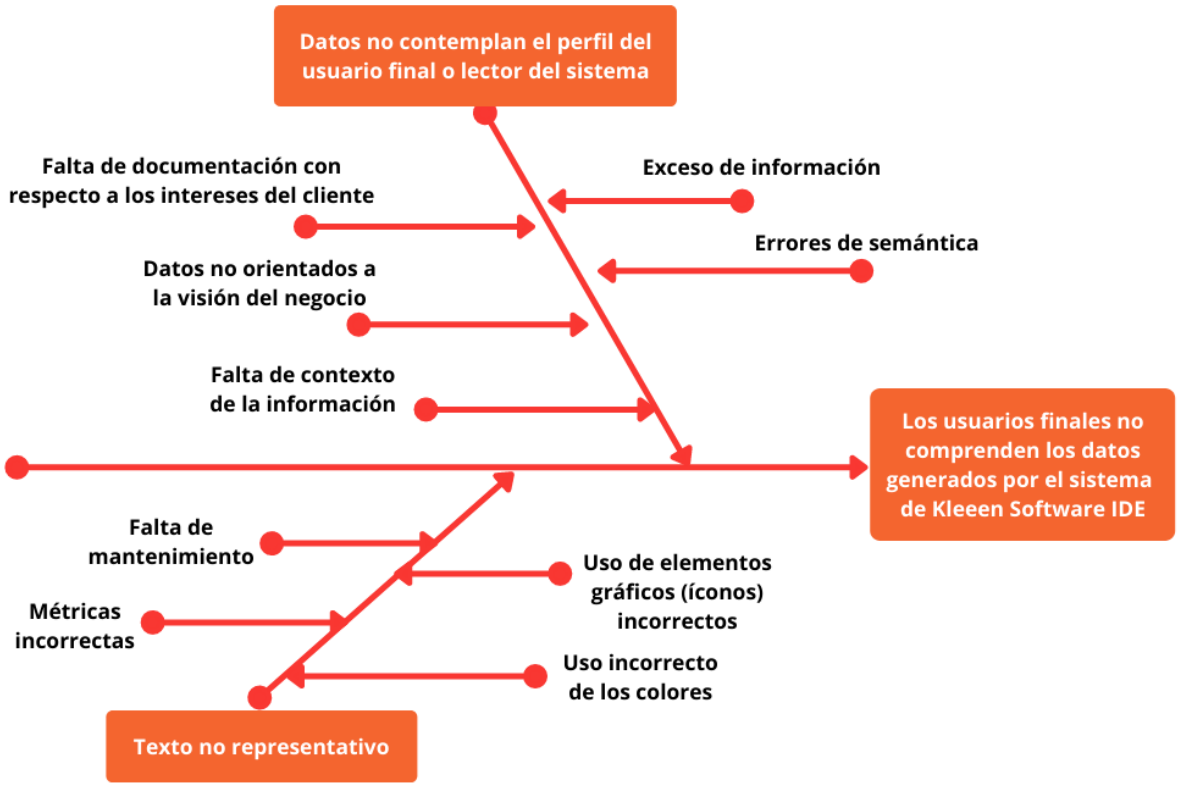
Figura 2:
Árbol de Problema.



Nota. Fuente: Elaboración propia.

A continuación, en **Figura 3,** se ilustra la causa raíz de la problemática presentada anteriormente:

Figura 3:
Diagrama Ishikawa



Nota. Fuente: Elaboración propia.

1.3.2. Justificación del proyecto

La presente propuesta se encuentra bajo el campo de diseño y especificación de *software*, donde se analizarán las necesidades de la empresa, se recolectarán los requerimientos y se visualizará una solución, de manera que se obtengan todos los insumos necesarios para la realización de un diseño de *software* acorde a lo requerido, para finalmente entregar a la organización una documentación que permita un posible desarrollo de un sistema de generación de lenguaje natural.

Con respecto a lo anterior, el presente proyecto buscará establecer un diseño de *software* basado en las mejores prácticas actuales, así como realizar la documentación necesaria, de manera que la organización, a un nivel gerencial y de ingeniería, comprendan aspectos como la definición y distribución de componentes para un posible desarrollo de un sistema de generación de lenguaje natural, garantizando también un correcto flujo comunicativo entre las diferentes áreas de la empresa.

El proyecto comprende diferentes aspectos relacionados con las etapas de la especificación y diseño de *software*, esto de acuerdo con los conocimientos obtenidos a lo largo de la carrera de Administración de Tecnología de Información. Para cada uno de estos elementos se presenta una justificación asociada.

1.3.2.1. Ingeniería de Requerimientos

Los requerimientos son peticiones que ayudan a definir correctamente “qué” se espera o necesita del *software* por desarrollar. Según Karl Wiegers (2013), estos se clasifican en: nivel de negocio, nivel de usuario y nivel de producto. Wiegers también piensa que “si no se consiguen los requerimientos de manera correcta, no importa lo bien que se trabaje cualquier cosa”, esto porque, aunque se realicen trabajos buenos, si estos no están enfocados en las necesidades correctas, no serán de utilidad.

Cualquier proyecto de *software* exitoso, existe detrás una ingeniería de requerimientos eficiente y eficaz, debido a esto, en el presente proyecto es requerida una ingeniería de requerimientos, con el objetivo de obtener una visión correcta de las necesidades de la empresa.

1.3.2.2. Diagrama UML 2.0

Según Paul, K., y Kimmel, P. (2011), UML *(Unified Modeling Language)* como su nombre lo dice, es un lenguaje de modelado, construcción y documentación de los elementos que forman un sistema de *software* orientado a objetos o componentes, de manera que, para desarrollar un diseño de *software* claro y correcto, se puede utilizar este tipo de diagramación.

Al realizar un diseño de software, es necesario definir un estándar de diagramación, por esto, para el presente proyecto se utiliza el lenguaje de modelado UML 2.0, con el objetivo de mostrar de manera clara la solución propuesta.

1.3.2.3. Diseño de Software

De acuerdo con Sommerville (2005), el diseño de *software* es un proceso para transformar los requisitos del usuario en alguna forma adecuada, lo que ayuda al programador en la codificación e implementación de este.

En el presente proyecto, tomando como base un análisis de requerimientos previo, se procede a realizar un diseño de *software* que permita, como salida, obtener los insumos necesarios para comprender y realizar la implementación del sistema de generación del lenguaje natural, en lenguajes de programación, o arquitecturas requeridas.

1.3.2.4. Arquitectura de Software

Según Sommerville (2005), la arquitectura de *software* es la parte de la ingeniería de *software* que se encarga de la descripción y el tratamiento de la estructura de un sistema, de forma que se pueda representar como una serie de componentes, con el fin de organizar adecuadamente los distintos subsistemas y permitir la integración de diferentes grupos de desarrollo en el mismo proyecto. Esta parte normalmente está ligada al diseño de *software*.

Como requisito para realizar la estructura de un sistema de manera correcta, es necesario definir una buena arquitectura, de manera que esta se vuelve vital para el desarrollo del presente proyecto.

1.3.2.5. Documentación de Software

Tomando lo que menciona Sommerville (2005), la documentación de un *software* es todo el registro de lo ocurrido y especificado para la implementación de una solución de *software*, esta permite una mayor claridad y una comunicación efectiva entre desarrolladores, gerencia, y demás involucrados en el proyecto. Esta documentación comprende aspectos como:

- Definiciones y especificación de requerimientos.
- Arquitectura del sistema.
- Diseño del sistema y modelo de datos.
- Descripción de procesos y servicios ofrecidos por el sistema.
- Documentación técnica - Especificación API.

Como resultado de esto, el presente proyecto tendrá como salida una robusta documentación, la cual contempla los conceptos definidos anteriormente, de manera que funcione para futuras implementaciones o diseño de software dentro de la empresa.

1.3.3. Beneficios esperados o aportes del Trabajo Final de Graduación

Debido a la problemática definida anteriormente, el presente proyecto espera generar documentación que afecte de manera positiva el entendimiento de la gerencia e ingeniería, ya que esto brindaría beneficios como:

- Los análisis generados como parte de la solución permitirán un mayor control y comprensión del proyecto a nivel interno de la empresa, esto es fundamental para comunicar el diseño a otras personas involucradas con la posible implementación del sistema diseñado en el presente proyecto.
- La empresa tendrá la documentación necesaria para implementar el sistema de generación de lenguaje natural, producida por el análisis y diseño del sistema previamente a una posible implementación.
- Se tendrá una ingeniería de requerimientos y un diseño de *software* para un sistema de generación de lenguaje natural. Estos diseños seguirán las buenas prácticas de desarrollo, permitiendo también la facilidad de mantenimiento del sistema y los módulos relacionados con el sistema principal de la empresa.
- Se darán recomendaciones en el proceso de generación y visualización de la información presentada por las diferentes partes del sistema.

1.4. Objetivos del Trabajo Final de Graduación

En esta sección, se definen el objetivo general y los objetivos específicos del proyecto, estos son las metas dentro del tiempo establecido, las cuales se puedan alcanzar durante el desarrollo del proyecto.

1.4.1. Objetivo General

Proponer un diseño de *software* de alto nivel de un sistema de generación de lenguaje natural, basado en las buenas prácticas de la industria, para producir la documentación que brinde el contexto y entendimiento necesario en caso de una futura implementación e integración de este nuevo sistema en Kleeen Software IDE, con el fin de solventar la necesidad presentada en la empresa Kleeen Software durante el primer semestre del 2022.

1.4.2. Objetivos Específicos

- Identificar la situación actual del sistema Kleeen Software IDE para la comprensión y descripción detallada de las necesidades y debilidades actuales de la empresa, esto con el fin de ser utilizadas en una futura integración a la propuesta de diseño del sistema de generación de lenguaje natural.
- Especificar los requerimientos funcionales y no funcionales para el diseño del sistema de generación de lenguaje natural, de acuerdo con la norma ISO/IEC/IEEE 29148, así como la organización y diseño de los componentes requeridos para el desarrollo del sistema.
- Evaluar tecnologías o plataformas de generación de lenguaje natural con el objetivo de recomendar aquellas que ofrezcan mejores medidas de aceptación y evaluación de los requerimientos para la propuesta planteada.
- Determinar el costo de implementación, así como el ROI relacionado con la propuesta de diseño de *software* por integrar en el sistema Kleeen Software IDE.

1.5. Alcance

En este apartado del documento, se expondrá el alcance del presente proyecto que se realizará en la empresa Kleeen Software, el cual consiste en una propuesta de diseño de *software* de un sistema de generación de lenguaje natural que será integrado en Kleeen Software IDE, esto en un lapso de dieciséis semanas.

El alcance del proyecto se limita a la propuesta de diseño, de manera que la implementación o integración de este queda excluida, así como cualquier otro entregable no especificado.

A continuación, se describen varios aspectos por incluir y aquellos que se excluirán de la realización del proyecto.

1.5.1. Supuestos

Como parte del desarrollo y puesta en marcha del presente proyecto, se supondrán los siguientes aspectos:

- Kleeen Software accederá y permitirá la programación de las reuniones pertinentes al proyecto siempre que sea necesario.
- La contraparte empresarial relacionada con el diseño del sistema notificará sobre la realización del presente proyecto a los involucrados pertinentes.
- Kleeen Software posee documentación que funciona como insumo de diseño y desarrollo, así como una metodología de trabajo establecida.
- Existirá una comunicación asertiva y eficiente entre los involucrados en el proyecto.

1.5.2. Entregables

A continuación, se mencionan los entregables esperados del proyecto planteado:

1.5.2.1. Entregables Académicos

En esta sección describen los entregables académicos que tendrá el proyecto.

1.5.2.1.1. Documento compuesto por:

- Capítulo 1 Introducción.
- Capítulo 2 Marco Conceptual.
- Capítulo 3 Desarrollo metodológico.
- Capítulo 4 Análisis de resultados.
- Capítulo 5 Propuesta de solución.
- Capítulo 6 Conclusiones.
- Capítulo 7 Recomendaciones.
- Referencias bibliográficas.
- Apéndices.
- Anexos.
- Glosario.

1.5.2.2. Entregables para la Empresa

- Documento que especifica los requerimientos del sistema.
- Documentación de todo el diseño de *software* de alto nivel propuesto, como arquitecturas y diagramas de:
  - Casos de uso.
  - Diagrama de Datos.
  - Diagrama de Arquitectura.
  - Diagrama de Componentes.
  - Diagrama de Despliegue.
  - Diagrama de Secuencia.
  - Diagrama de Actividad.
  - Etc.
- Validaciones costo-beneficio.
- Medidas de aceptación de la propuesta de diseño.
- Prueba de uso estadístico
  - Métricas de evaluación.
  - Documentación de gestión del cambio.
  - Plan de implementación del sistema propuesto.

1.5.3. Gestión del proyecto

A continuación, se detallarán los entregables de la gestión del proyecto.

1.5.3.1. Gestión de Cambios.

La gestión de cambios será utilizada como control y documentación relacionada con los posibles cambios que puedan afectar el alcance del presente proyecto. La plantilla de solicitud de cambio se encuentra en el **Apéndice B.**

1.5.3.1. Minutas.

Las minutas serán la documentación instantánea originada en cada reunión, en esta se describirán todos los aspectos o puntos abordados, así como el registro de participantes. La plantilla para las minutas se encuentra en el **Apéndice A**.

1.5.3.2. Cronograma de proyecto.

El cronograma del proyecto se utiliza como un apoyo visual y administrativo para la gestión de las fechas de entrega de los productos, además, sirven como medida de control de avance de los entregables académicos y para el negocio. El cronograma del proyecto se encuentra en el **Anexo I.**

1.5.4. Limitaciones

A continuación, se mencionan los posibles factores que podrían afectar en el desarrollo del proceso:

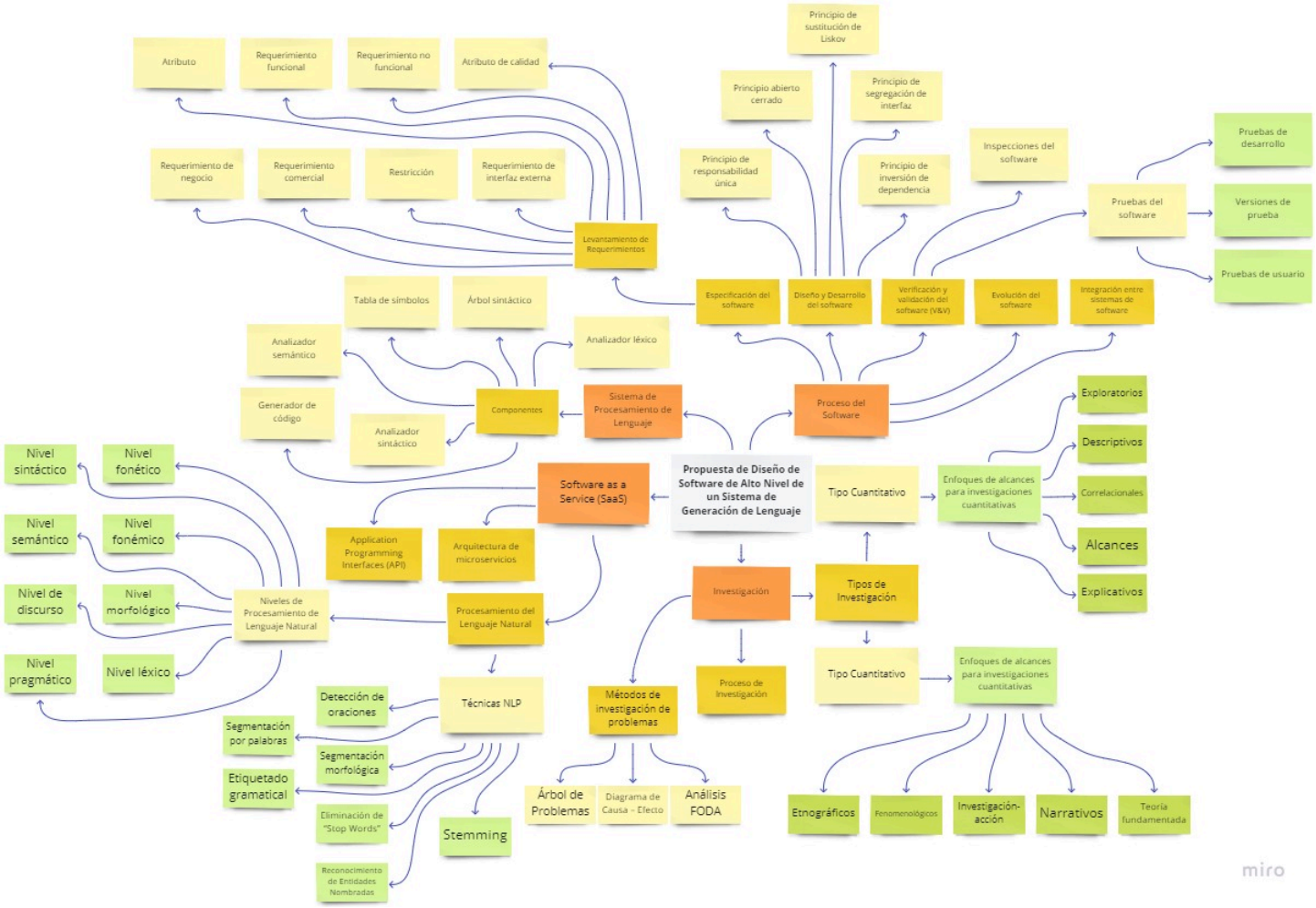
- Reuniones sujetas a la disponibilidad de los involucrados en el proyecto.
- Atrasos con reuniones o poca disponibilidad de los involucrados pueden generar atrasos al momento de recopilar información o requerimientos del proyecto.
- Requerimientos identificados o solicitados en la elaboración del proyecto que no sean realizables en el alcance del proyecto.
- Errores o faltantes en la documentación de los sistemas de la empresa pueden afectar el avance o entendimiento del contexto o acceso a la información.

2. Marco Conceptual

En el presente capítulo se muestra una recopilación de conceptos y/o consideraciones técnicas que permiten sustentar este trabajo final de graduación. Estos están relacionados con arquitectura, diseño y especificación de *software*, sumado a todos aquellos aspectos técnicos relacionados con la propuesta tecnológica al problema planteado.

Como introducción a los conceptos presentados a continuación, en la **Figura 4,** se muestra la relación entre estos y el proyecto.

Figura 4:
Mapa Conceptual



Nota. Fuente: Elaboración propia

2.1. Proceso del *Software*

Se define como un conjunto de actividades que contemplan la especificación, el diseño y el desarrollo de *software*.

Según Sommerville (2005), el proceso incluye las siguientes actividades:

2.1.1. Especificación del *software*

Esta tarea es vital en el proceso de *software*, debido a que establece los pilares para el desarrollo de un sistema, esto porque consiste en la definición del *software* que se desea producir, así como las restricciones sobre su operación.

Para el desarrollo de esta actividad se requiere realizar el proceso de levantamiento de requerimientos.

2.1.1.1. Levantamiento de Requerimientos .

Según Sommerville (2005), el levantamiento de requerimientos es la acción o tarea de recopilar las necesidades o peticiones para un sistema por parte del cliente o usuario. Estas tareas o acciones se definen como requerimientos, y, de acuerdo con el mismo actor, la definición formal de estos es:

Los requerimientos para un sistema son la descripción de los servicios proporcionados por el sistema y sus restricciones operativas. Estos requerimientos reflejan las necesidades de los clientes de un sistema que ayude a resolver algún problema como el control de un dispositivo, hacer un pedido o encontrar información. (p. 108)

También, de acuerdo con Pressman (2005), los requerimientos funcionan como una especificación de lo que debería ser implementado en el sistema, es decir, son descripciones de cómo el sistema debe comportarse, o de un sistema de propiedad o atributo.

De acuerdo con la norma ISO/IEC/IEEE 2914, los requerimientos deben tener las siguientes características:

- **Necesario:** El requisito define una capacidad esencial, característica, restricción y/o factor de calidad, si este se elimina, existirá una deficiencia que no puede ser satisfecha por otras capacidades del producto o proceso.
- **Libre Implementación:** Debe ser independiente de la implementación. El requerimiento establece lo que se requiere, no cómo se debe cumplir el requisito.
- **Sin ambigüedades:** El requerimiento se establece de tal manera que puede interpretarse de una sola manera.
- **Consistente:** El requerimiento está libre de conflictos con otros requerimientos.

- **Completo:** El requerimiento establecido no necesita más amplificación porque es medible y describe suficientemente la capacidad y las características para satisfacer la necesidad de la parte interesada.
- **Singular:** La declaración del requerimiento incluye solo un requisito sin uso de conjunciones.
- **Factible:** El requerimiento es técnicamente factible, no requiere grandes avances tecnológicos y se ajusta a las limitaciones del sistema (por ejemplo: costo, cronograma, técnico, legal, reglamentario) con un riesgo aceptable.
- **Trazable**: El requerimiento se puede rastrear, todas las relaciones padre-hijo para el requerimiento se identifican, de modo que el requisito se rastree hasta su origen e implementación.
- **Verificable:** El requerimiento tiene los medios para demostrar que el sistema satisface el requisito especificado. Se pueden recopilar pruebas que demuestren que el sistema puede satisfacer el requisito especificado. La verificabilidad se mejora cuando el requisito es medible.

2.1.1.1.1. Tipos de Requerimientos.

Existen diferentes tipos de requerimientos. De acuerdo con Wiegers y Beatty (2013), algunos de estos son:

- ***Requerimiento de negocio***

Es un objetivo de alto nivel donde el negocio describe un deseo o necesidad del cliente ya sea como un atributo de valor de un producto o una obligación administrativa.

- ***Requerimiento comercial***

Son aquellas leyes, reglas, políticas, directrices, estándares o bien, aquellas regulaciones que se deben cumplir de manera obligatoria, pues restringen o rigen algún aspecto legal o importante del negocio.

- ***Restricción***

Es una restricción que se impone a las opciones disponibles para el desarrollador, ya sea tanto para el diseño o desarrollo del sistema.

- ***Requerimiento de interfaz externa***

Es una descripción de las posibles conexiones entre el sistema y el usuario, u otro sistema de software o hardware. Suelen encontrarse en esta sección los requerimientos sobre las interfaces de comunicaciones o integraciones.

➢ ***Requerimiento funcional***

Es una descripción del comportamiento esperado del sistema bajo condiciones específicas.

➢ ***Requerimiento no funcional***

Es una descripción de un atributo o característica que el sistema debe exhibir o bien, una restricción que debe respetar.

➢ ***Requerimientos de calidad***

Es un tipo de requerimiento no funcional que describe un servicio, desempeño o característica mínima esperada del sistema. Estos contemplan los atributos de rendimiento, mantenibilidad y seguridad.

Dentro de los atributos de calidad, se encuentran los atributos de experiencia de usuario e interfaz de usuario, los cuales contemplan atributos de:

➢ Usabilidad: comprensibilidad, aprendibilidad, operatividad y atractividad.
➢ Accesibilidad: perceptible, operable, comprensible y robusto.

- Interfaz de Usuario

Según Ramírez (2017), se refiere a la interfaz visual de una herramienta de software con la que un usuario interactúa, esta es la suma de una arquitectura de información con elementos visuales y patrones de interacción.

- Experiencia de Usuario

De acuerdo con Costa (2012), se refiere específicamente a la experiencia del usuario mientras utiliza un sistema, esto como resultado de un trabajo para mejorar un producto o servicio y que sea funcional de manera agradable y correcta. Según Bargas-Avila & Hornbaek (2011) se define como un concepto sombrilla que buscaba ampliar la concepción de la interacción humano-máquina añadiendo al concepto de usabilidad una dimensión humana de la experiencia con los dispositivos electrónicos mediados por una interfaz.

2.1.1.1.2. Análisis de Requerimientos

Según Sommerville (1997), el análisis de requerimientos es una actividad que tiene el propósito de identificar problemas como falta de información, inconsistencias y conflictos de requisitos. Como parte del análisis de los requerimientos, se realizan tareas de:

➢ ***Descripción Detallada y Priorización de los Requerimientos***

Para esta tarea, según Lant, M. (2019), comúnmente se utiliza la técnica de MoSCoW con el objetivo de obtener un mejor entendimiento en común con las partes interesadas sobre la relevancia o importancia que tiene la entrega de cada requerimiento.

Sus acrónimos son:

- **M - Must:** requisitos fundamentales y obligatorios. Necesarios para el éxito del servicio.
- **S - Should:** requisitos que deberían ser cumplidos en la medida de lo posible. No depende directamente del éxito del servicio.
- **C - Could:** requisitos adicionales implementados en caso de disponer de tiempo para ello.
- **W - Won't:** requisitos que quedan excluidos.

- ***Puntuación de requerimientos***

Para la puntuación de requerimientos se suelen combinar los términos de urgencia y valor del negocio. Para esto, según Lant, M. (2019), se entiende que:

- Urgencia: se refiere a la necesidad de obtener un requerimiento de manera completa, rápida y correcta.
- Valor de Negocios: se refiere al valor que posee un requerimiento para la organización.

Para conocer el valor que recibe cada requerimiento, se utiliza una puntuación que va del uno al cinco, esto para cada término explicado anteriormente. Estos puntos se consiguen utilizando una guía también propuesta por Lant, M. (2019).

En la **Tabla 1**, se presenta la tabla propuesta por Lant, M (2019) para puntuar la urgencia de un requerimiento.

Tabla 1:
Guía para la puntuación de la urgencia de un requerimiento.

Valor	Guía
5	• Extremadamente limitado en el tiempo. • Nivel extremo de dependencia de otros elementos para completar esta tarea. • Si no se completa inmediatamente, tiene poco o ningún valor hacerlo.
4	• Altamente limitado en el tiempo. • Alto nivel de dependencia de otros elementos en la realización de esta tarea. • Importante para pasar al siguiente *sprint* debido a requisitos contractuales o del cliente.
3	• Moderadamente limitado en el tiempo. • Dependencia moderada de otros elementos en la realización de esta tarea. • Deseable para completar en los próximos uno o dos *sprints*.
2	• Mínimamente limitado en el tiempo.

Valor	Guía
	• Dependencia mínima de otros elementos para completar esta tarea. • La finalización en los siguientes dos o tres *sprints* es adecuada.
1	• Sin limitaciones de tiempo. • Sin dependencia. • Poco o ningún impacto.

Nota. Fuente: Lant, M. (2019).

En la **Tabla 2** se presenta la tabla propuesta por Lant, M (2019) para puntuar el valor del negocio de un requerimiento.

Tabla 2:
Guía para la puntuación del valor del negocio de un requerimiento.

Valor	Guía
5	• Extremadamente importante para la mayoría o todos los clientes. • Impacto extremo en la marca o reputación. • Crítico para el éxito del negocio.
4	• Importante para muchos clientes. • Impacto significativo en la marca o reputación. • Ventaja competitiva significativa.
3	• Importante para un número moderado de clientes. • Impacto significativo moderado en la marca o la reputación. • Ventaja competitiva importante moderada.
2	• Importante solo para unos pocos clientes. • Impacto menor en la marca o reputación. • Ventaja competitiva menor.
1	• Importante solo para unos pocos o incluso para ningún cliente. • Poco o ningún impacto en la marca o la reputación. • Poca o ninguna ventaja competitiva.

Nota. Fuente: Lant, M. (2019).

Según Lant, M. (2019), la priorización se calcula obteniendo el producto de la urgencia y el valor de negocio, este valor se posiciona en la **Tabla 3.**

Tabla 3:
Tabla de Priorización.

Valor del Negocio								
	5	5	10		15	20	25	
	4	4	8		12	16	20	M - Must
	3	3	6		9	12	15	S - Should
	2	2	4		6	8	10	C - Could
	1	1	2		3	4	5	W - Won't
		1	2		3	4	5	
		Urgencia						

Nota. Fuente: Lant, M. (2019).

- ***Descripción de los Casos de Uso***

De acuerdo con Pressman (2005), los casos de uso se detallan con el objetivo de brindar más detalles sobre la interacción entre los actores y el sistema.

Según Cockburn (2001), para esta descripción se recomienda el siguiente formato:

- Identificador del caso de uso.
- Nombre del caso de uso.
- Actor principal.
- Precondiciones.
- Postcondiciones.
- Escenario.
- Excepciones.

- ***Modelo UML***

Un modelo es una colección de imágenes y texto que representa algo, en el contexto del presente proyecto, representa software, componentes, y arquitecturas.

- Modelo General de Casos de Uso

Según Pressman (2005), un caso de uso es un conjunto de acciones realizadas por el sistema que especifica un comportamiento que el usuario puede realizar. Estos se definen desde la perspectiva

de un actor. Un actor se conoce como el usuario o dispositivo que interactúa con el sistema. Para esto, se realiza un modelado basado en UML, el cual da apoyo tanto de texto como gráfico.

Componentes de Diagramación

Los elementos de diagramación del modelo de casos se muestran en la **Tabla 4**:

Tabla 4:
Elementos del Diagrama de Casos de Uso

 Actor	Representa participantes en los casos de uso, ya sean personas o cosas. S
 Conector	Se utilizan para indicar la manera en que un actor y un caso de uso está asociados.
 Caso de uso	Es un conjunto de acciones realizadas por el sistema que especifica un comportamiento que el usuario puede realizar

Nota. Adaptación Kimmel, P. (2019). Manual UML.

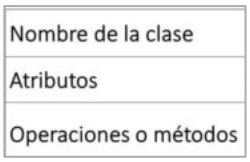
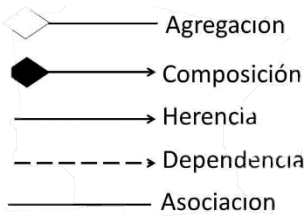
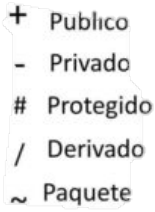
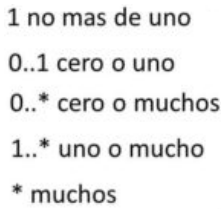
- Diagrama de clases

De acuerdo con Pressman (2005), el diagrama UML más comúnmente usado, y la base principal de toda solución orientada a objetos, posee elementos como las clases dentro de un sistema, atributos y operaciones, y la relación entre cada clase.

Componentes de Diagramación

Los elementos del diagrama de clases se muestran en la **Tabla 5**:

Tabla 5:
Elementos del Diagrama de Clases

 Clase	Una clase representa un objeto o un conjunto de objetos que comparte una estructura y un comportamiento comunes
 Relaciones	Se utilizan para indicar la manera en que las clases u otros elementos del diagrama están relacionados entre sí.
 Visibilidad	Define el nivel de acceso del elemento.
 Multiplicidad	Define la cantidad de elementos relacionados.

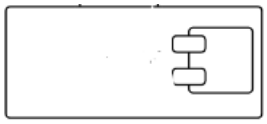
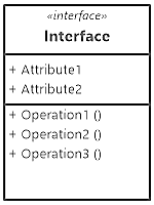
Nota. Adaptación Kimmel, P. (2019). Manual UML.

- Diagrama de componentes

Según Pressman (2005), este diagrama muestra la relación estructural de los elementos del sistema de software, muy frecuentemente empleados al trabajar en sistemas complejos con componentes múltiples.

Componentes de Diagramación
Los elementos del diagrama de componentes se muestran en la **Tabla 6**:

Tabla 6:
Elementos del Diagrama de Componentes

 Componente	Una entidad requerida para ejecutar una función de estereotipo.
 Relación Interface	Estos símbolos representan las interfaces donde un componente produce información utilizada por la interfaz requerida de otro componente.
 Interface	Muestra entradas o materiales que un componente recibe o proporciona.

Nota. Adaptación Kimmel, P. (2019). Manual UML.

- Diagramas de actividad

Según Pressman (2005), refleja flujos de trabajo de negocios u operativos representados gráficamente para mostrar la actividad de alguna parte o componente del sistema.

Componentes de Diagramación
Los elementos del diagrama de actividad se muestran en la **Tabla 7**:

Tabla 7:
Elementos del Diagrama de Actividad

 Símbolo de Inicio	Representa el inicio de un proceso o flujo de trabajo.
 Acción	Indica las actividades que componen un proceso modelado.
 Conectores	Muestra el flujo direccional o el flujo de control de la actividad. De acuerdo con la forma en la que se utilice, puede significar unir o separar dos flujos de actividades.
	Representa el final de un proceso o flujo de trabajo.

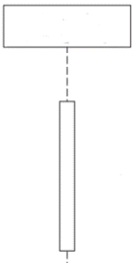
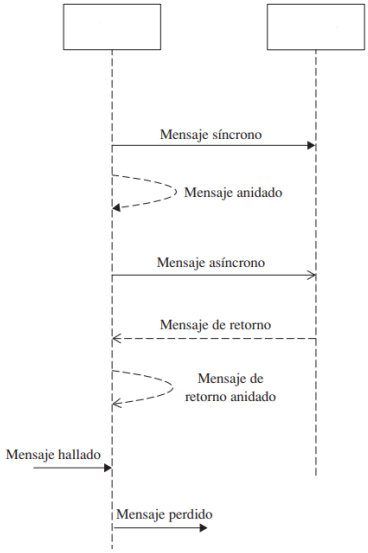
Nota. Adaptación Kimmel, P. (2019). Manual UML.

- Diagrama de secuencia

Según Pressman (2005), este diagrama muestra cómo los objetos interactúan entre sí y el orden de la ocurrencia, así como las interacciones para un escenario concreto.

Componentes de Diagramación
Los elementos del diagrama de secuencia se muestran en la **Tabla 8**:

Tabla 8:
Elementos del Diagrama de Secuencia

 Línea de vida	Representa un ejemplo de una clase, y la línea que desciende en forma vertical es un lugar conveniente para sujetar mensajes entrantes y salientes
 Activación de línea de vida	Reemplaza la línea de vida en el transcurso de la duración de la existencia de un caso en específico, teniendo presente que un objeto puede tener varios casos donde se crea y se destruye.
 Mensajes	Son líneas dirigidas que conectan líneas de vida, esta se inicia en una línea de vida de un objeto, y la flecha apunta hacia otra línea de vida objeto que contenga el mensaje invocado.

Nota. Adaptación Kimmel, P. (2019). Manual UML.

2.1.2. Diseño y Desarrollo del Software

En este apartado se establecen los pilares del diseño y la programación requerida para el desarrollo del *software*.

De acuerdo con Jaiswal (2019), el diseño de *software* es responsable del diseño a nivel de código sobre cómo y qué está haciendo cada módulo, las clases, las funciones, etc. El diseño de un sistema representa la organización del sistema o la estructura que debe tener, también proporciona una evidencia de la manera en que se comporta.

El diseño de *software* debe contemplar aspectos como: la calidad, el rendimiento, mantenibilidad y, en general, el éxito del sistema.

De acuerdo con Pressman (2005), la etapa del diseño del *software* encierra cinco fases:

- Diseño de los Datos: define la relación entre cada uno de los elementos estructurales del programa. Se crea un modelo de datos o información que se representa en un nivel de abstracción alto.
- Diseño Arquitectónico: describe cómo se comunica el *software* consigo mismo y con los sistemas que operan con él. Este presenta tres aspectos importantes:
  - Información sobre el dominio de la aplicación del *software* que se va a elaborar.
  - Elementos específicos del modelo de requerimientos, tales como diagramas de flujo de datos o clases de análisis.
  - Disponibilidad de estilos arquitectónicos y sus patrones.
- Diseño de la Interfaz: describe cómo se ve el sistema y la interacción que posee con el usuario.
- Diseño de Componentes: describe por completo los detalles internos de cada componente, se definen estructuras de datos para todos los objetos de datos locales y detalles algorítmicos para todo el procesamiento que tiene lugar dentro, así como la interfaz que permite el acceso a todas las operaciones de estos (comportamientos).
- Diseño del Despliegue: indica la forma en la que se organiza la funcionalidad del *software* y los subsistemas dentro del ambiente físico de la computación que lo apoyará.

También, según Jaiswal (2019), un buen diseño de *software* debe cumplir con los siguientes principios:

2.1.2.1. Principio de Responsabilidad Única

Indica que cada clase debe tener un solo propósito, una preocupación y una razón para cambiar.

2.1.2.2. Principio Abierto Cerrado

Está abierto para la extensión, pero cerrado para el ajuste. Indica que el diseñador podrá agregar más funcionalidades a la clase mas no editar las funciones actuales de una manera que rompa el código existente que las usa.

2.1.2.3. Principio de Sustitución de Liskov

Este principio guía al desarrollador a utilizar la herencia en un método de exceso que no romperá la lógica del dispositivo en ningún caso.

2.1.2.4. Principio de Segregación de Interfaz

Se pueden implementar múltiples interfaces, luego estructurar su código de una manera tan extrema que una categoría nunca se vea obligada a implementar una función que no sea necesaria para su propósito. Entonces, se pueden categorizar las interfaces.

2.1.2.5. Principio de Inversión de Dependencia

Las entidades deben depender de abstracciones, no de concreciones. Establece que el módulo de alto nivel no debe depender del módulo de bajo nivel, sino que deben depender de abstracciones.

2.1.3. Verificación y validación del software (V&V)

La validación de *software* es una tarea que permite asegurar que se completaron los requerimientos definidos y se cumple con las necesidades de los clientes.

De acuerdo con Salazar (2012), el proceso de verificación y validación brinda una evaluación objetiva del sistema, así como del proceso de desarrollo del *software*. Esta evaluación asegura que los requerimientos y necesidades del cliente se cumplieron de manera correcta, completa, precisa, consistente y de una manera fácil de probar.

En pocas palabras, el objetivo de esta actividad es facilitar la detección y corrección de errores, así como mejorar el proceso de desarrollo, mitigar y evitar los riesgos que se puedan presentar y asegurar el cumplimiento de los requerimientos de rendimiento, el cronograma establecido y el uso responsable de recursos y presupuesto definido.

Dentro de este proceso, se encuentra el uso de dos técnicas muy importantes, las cuales son:

2.1.3.1. Técnicas de V&V

Las siguientes técnicas son utilizadas para la comprobación y análisis de los sistemas de acuerdo con Pressman (2005):

2.1.3.1.1. Estrategia De Cuarto Limpio

Las estrategias de cuarto limpio son una serie de tareas que permiten la verificación y evaluación del *software*, dentro de sus tareas se encuentran:

- **Planeación del incremento:** Se desarrolla un plan de proyecto que adopte la estrategia incremental.
- **Recopilación de requerimientos:** Se desarrolla una descripción más detallada de los requerimientos del cliente (para cada incremento).
- **Especificación de estructura de caja:** Se usa un método de especificación que utiliza las estructuras de caja para describir la especificación funcional.
    - **Caja negra:** precisa el comportamiento de un sistema o de una parte de un sistema. **Caja de estado:** se representan las entradas, salidas y estados del sistema.
    - **Caja clara:** contiene el diseño de procedimientos para la caja de estado.
- **Diseño formal:** se especifican las cajas negras, estas se refinan y se construyen las cajas de estado y cajas claras.
- **Verificación de exactitud:** se realizan diferentes tareas de verificación, como el aplicar un conjunto de "preguntas de exactitud".
- **Generación, inspección y verificación de código:** se asegura la conformidad semántica, las estructuras y la exactitud sintáctica del código.
- **Prueba de uso estadístico:** se ejecuta una serie de pruebas derivadas de una muestra estadística.
- **Certificación:** Una vez completadas la verificación, inspección y prueba de uso (y todos los errores corregidos), el incremento se certifica como listo para su integración.

2.1.3.1.2. Inspecciones del software

Consiste en analizar y comprobar el documento de requerimientos, el diseño y el código fuente del programa.

2.1.3.1.3. Pruebas del software

Según Sommerville (2005), las pruebas de *software*:

Intentan demostrar que un programa hace lo que se intenta que haga, así como descubrir defectos en el programa antes de usarlo. Al probar el *software*, se ejecuta un programa con datos artificiales. Hay que verificar los resultados de la prueba que se opera para buscar errores, anomalías o información de atributos no funcionales del programa. (p. 206)

De acuerdo también con Sommerville (2005), para que un sistema sea confiable, como mínimo debe ser sometido a las siguientes tres etapas de pruebas:

- ***Pruebas de desarrollo***

En esta etapa se valida y prueba el código del sistema con el fin de descubrir errores (*bugs*) y defectos. Aquí intervienen los diseñadores y programadores del sistema.

- ***Versiones de prueba***

Aquí un equipo enfocado en calidad y pruebas hace una comprobación de las versiones del sistema. El objetivo es comprobar que el sistema cumpla con los requerimientos de manera integrada y completa.

- ***Pruebas de usuario***

En esta etapa los usuarios reales o potenciales del sistema se enfocan en probar el *software* en su propio entorno. Aquí se verifica si el *software* realmente cumple las expectativas de los usuarios, de manera que se decide si debe aceptarse o requiere más desarrollo, o del todo no se acepta.

2.1.3.2. Medidas de Aceptación y Valoración de Diseño de Software

Las medidas de aceptación y valoración de un diseño de *software* comúnmente están ligadas al término de verificación y validación del *software*, las cuales a su vez contemplan aspectos de calidad, esto porque, de acuerdo con Nuseibeh (2000), el detectar errores en las etapas tempranas del desarrollo de *software* permite la reducción de costos y la corrección de errores de diseño de *software*.

Según Juristo (2011), las medidas de aceptación se definen como aquellos aspectos que permiten forjar un criterio sobre los escenarios contemplados en el diseño de *software*, de manera que se conoce si cumple o no los requerimientos definidos.

Estas medidas están vinculadas con las pruebas de aceptación de tipo caja negra, donde el rol principal es el cliente. Esto se debe a que el objetivo de estas pruebas no es necesariamente validar el comportamiento lógico del diseño, pues este no se encuentra implementado, sino que se enfoca en los escenarios, casos de uso o casos de negocio en los cuales se contextualiza el diseño del sistema.

Por otra parte, la valoración de un diseño de *software* corresponde a aquellas actividades donde el diseño se estudia de manera que se pueda puntuar y valorar si este es apto o no para su posible implementación. La realización de esta tarea puede ser por medio de cuestionarios con preguntas como:

- ¿Cubre el diseño todos los requisitos funcionales?
- ¿Resulta ambigua la documentación del diseño?
- ¿Se ha aplicado la notación de diseño correctamente?
- ¿Se han definido correctamente las interfaces entre elementos del diseño?
- ¿Es el diseño suficientemente detallado como para que sea posible implementarlo en el lenguaje de programación elegido?

De acuerdo con Bedini (2002), el diseño puede ser evaluado por dos categorías de atributos:

- Atributos Internos: Tamaño, reúso, modularidad, acoplamiento, cohesividad, funcionalidad.

- Atributos Externos: calidad, complejidad, y mantenibilidad.

Otra manera de evaluar un diseño de *software* es por medio de instrumentos como:

- Matriz de cobertura de los requerimientos:

De acuerdo con Sommerville (2005), se entiende que la trazabilidad de un requerimiento se realiza desde el origen, hasta su implementación en el sistema y viceversa, justificando así, que un requerimiento deba ser rastreable desde que se define hasta que se concluye el desarrollo de *software*, esto debido a que, como menciona el mismo autor, para diseñar o testear cualquier componente del sistema es necesario saber qué requerimientos satisface (aunque sea parcialmente).

La matriz de cobertura de requerimientos o matriz de trazabilidad permite rastrear los requerimientos con el objetivo de hacer una evaluación en el diseño y en la implementación de un sistema.

- Diagramas de secuencia de los casos de uso.

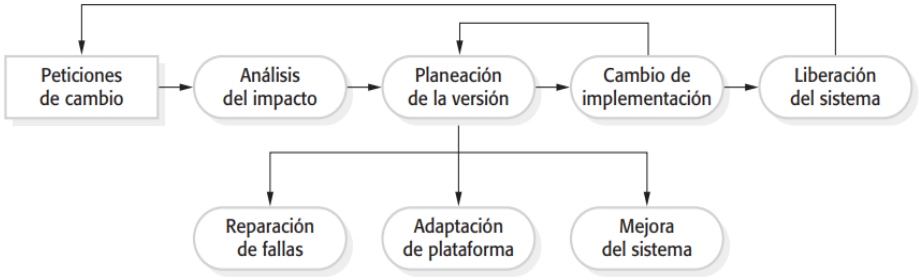
Según Kimmel (2008), los diagramas de secuencia se utilizan para representar un ordenamiento explícito en el tiempo y los actores, estos permiten visualizar los objetos, sus interacciones y cuáles mensajes se envían. Estos diagramas se utilizan para mostrar la manera en que varios objetos sustentan un caso de uso.

2.1.4. Evolución del *software*

En esta actividad el *software* se somete a cambios o modificaciones para adaptarlo a los cambios del mercado, o bien, ajustarlo a las nuevas necesidades del cliente.

El proceso de evolución de *software* no se puede representar de manera lineal, esto debido a que el constante crecimiento o surgimiento de cambios y nuevos requerimientos obligan a que el sistema se someta a muchas modificaciones en cualquier momento para tener una visión más realista de este proceso. En la **Figura 5**. se muestra el proceso de evolución de *software* según Sommerville (2005):

Figura 5:
Proceso de evolución de software.



Nota. Fuente: Sommerville (2005).

2.1.5. Integración entre sistemas de software

De acuerdo con Rivas (2006), esto se conoce como la práctica de conectar y unificar diferentes sistemas de *software*. Es común que las organizaciones requieran llevar a cabo esta actividad, ya sea porque están realizando una nueva aplicación y necesiten la integración con un sistema heredado, o porque están en transición de tecnologías.

2.1.5.1. Caracterización del Problema de Integración de Sistemas

Los sistemas normalmente presentan problemas al momento de ser integrados, estos suelen deberse a la autonomía y heterogeneidad. Según Rivas (2006), la autonomía se define cuando la construcción, el mantenimiento y la operación se realizan de forma independiente, sin la necesidad de considerar una posible integración con otros sistemas existentes en la organización. A su vez, esta misma autonomía da como resultado la heterogeneidad, debido a que favorece la utilización de diferentes tecnologías y la posible incompatibilidad entre los modelos tecnológicos.

La heterogeneidad se produce debido a la incompatibilidad de las versiones que cada sistema posee, por esto, al momento de realizar integraciones, se deben especificar criterios mínimos, como infraestructura, modelos y la administración de la información.

2.1.5.2. Tipos de Integración

Con base en las investigaciones de Rivas (2006) y Medina (2019), a continuación, se explican los diferentes tipos de integración de sistemas que existen en la actualidad.

2.1.5.2.1. Integración horizontal

Es el método para definir un sistema de comunicación, se enfoca en la transmisión y el monitoreo de mensajes, también proporciona servicios de transformación y mapeo de datos. Este tipo permite flexibilidad en la que los equipos pueden agregar, quitar o ajustar un sistema sin interrumpir el resto de los componentes.

2.1.5.2.2. Integración vertical

Es una solución a corto plazo, rápida y económica debido a que este método se concentra en el desarrollo de entidades funcionales que permitan una conexión con los sistemas de *software* de manera vertical. Sin embargo, este tipo de integración desarrolla una limitación al escalar el *software*, lo que significa que la información no se comparte correctamente y se aísla en cada sistema.

2.1.5.2.3. Integración de formato de datos común

Consiste en evitar el uso de un adaptador al convertir o transportar datos por medio de la estandarización de estos, de manera que un sistema debe ser aceptado por el otro sistema. Este tipo permite la traducción de datos y promueve la automatización.

2.2. Software as a Service (SaaS)

Este tipo de *software*, de acuerdo con Hernández (2009), se define como aquel modelo de distribución del *software* que proporciona a los usuarios un acceso a aplicaciones a través de la nube. Estos servicios son accesibles por cualquier cliente mediante un dispositivo inteligente como: *tablets*, celulares, o navegadores web.

Los SaaS poseen diferentes características que los hacen realmente útiles, dentro de ellas están:

- ➢ Personalización: en su mayoría permiten la adaptación con interfaces de usuario, esto para configurar los colores y logo de la empresa cliente.
- ➢ Entrega de funcionalidades acelerada: permiten una entrega de funcionalidades nuevas en periodos cortos, debido a que son servicios centralizados, así como los cambios realizados por el proveedor.
- ➢ Integración con protocolos abiertos: debido a que se ofrece a sistemas internos de empresas, debe permitir la posibilidad de acceder a datos de estos sistemas, por esto, en su mayoría se ofrecen integraciones mediante APIs (consultar apartado ***Application Programming Interfaces (API)***) por medio de los protocolos HTTP, SOAP o REST.
- ➢ Colaboración: debido al alto crecimiento de sistemas web 2.0 y posteriores, este servicio ofrece características de colaboración entre usuarios.

2.2.1. Arquitectura de microservicios

Según Jaiswal (2019), la arquitectura de microservicios depende del desarrollo de servicios estándar pequeños e independientes, en estos cada servicio resuelve un problema en específico, o bien, realiza una tarea singular. Estos módulos se comunican entre sí a través de una API (consultar apartado ***Application Programming Interfaces (API)***) bien definida para cumplir el objetivo comercial.

2.2.2. Application Programming Interfaces (API)

De acuerdo con Meng et al. (2018), un API se enfoca en romper las barreras y establecer enlaces entre diferentes herramientas de *software*, esto logrando una evolución de diferentes servicios y la integración entre sistemas, así como permitiendo mayor conectividad entre las tecnologías.

Las APIs buscan comunicar dos estructuras y establecer una transmisión de la información en ambos sentidos, lo cual genera reducción de procesos o recursos.

Según Matías, R y Vélez, J. (2016), en la actualidad existen APIs que permiten diferentes tipos de procesamientos o servicios. Para este proyecto, se profundizará en el Procesamiento del Lenguaje Natural (consultar apartado **Procesamiento del Lenguaje Natural (NLP)**).

2.2.3. Procesamiento del Lenguaje Natural (NLP)

De acuerdo con Matías, R & Vélez, J. (2016), este procesamiento es un área de investigación el cual se dedica a explorar las posibilidades de que un sistema comprenda, y manipule el lenguaje natural, ya sea escrito u oral, de manera que se pueda ser utilizado para desempeñar tareas o trabajos de interés.

2.2.3.1. Niveles de Procesamiento de Lenguaje Natural

Existen clasificaciones para los diversos tipos de procesamientos que se pueden realizar sobre un texto, estas clasificaciones se pueden representar como niveles, los cuales varían de acuerdo con la complejidad de cada uno (Matías, R y Vélez, J., 2016).

A continuación, también de acuerdo con Matías, R y Vélez, J., (2016), se describirán los niveles de procesamiento existentes:

2.2.3.1.1. Nivel fonético

Se encarga de la interpretación y comprensión del sonido dentro de las palabras, estudia la variación fonética y los elementos invariantes que caracterizan los sonidos o los patrones prosódicos.

2.2.3.1.2. Nivel fonémico

Se enfoca en el estudio de las variaciones que existen en la pronunciación al momento de conectar palabras dentro de oraciones.

2.2.3.1.3. Nivel morfológico

Se encarga de determinar la composición de las palabras, es decir, determinar la clase o categoría gramatical de cada una de estas y cómo se ubican dentro de una oración.

2.2.3.1.4. Nivel léxico

Etiqueta y clasifica a las palabras de acuerdo con su significado individual o dependiendo del contexto en el que se ubiquen.

2.2.3.1.5. Nivel sintáctico

Analiza el porqué de cada palabra dentro de una oración, con el fin de descubrir la estructura u organización de esta.

2.2.3.1.6. Nivel semántico

Busca el sentido que tiene una oración de acuerdo con la interacción de las palabras que la componen, de manera que determina sólo un significado correcto por oración.

2.2.3.1.7. Nivel de discurso

Similar al nivel semántico, este busca el sentido que tiene una composición de oraciones de acuerdo con la interacción entre sí dentro de un texto.

2.2.3.1.8. Nivel pragmático

Similar al nivel semántico, este busca las variaciones que existen dentro de un texto y analiza las variables que lo conforman, estas variables son:

- Emisor.
- Receptor.
- Situación.
- Enunciado.

2.2.3.2. Técnicas NLP

De acuerdo con Ramos, F. & Velez, J. (2016), con el fin de obtener mejores y completos resultados en el procesamiento de lenguaje natural, a lo largo de los años se ha definido un conjunto de técnicas las cuales permiten extraer del texto información determinada.

A continuación, se describirán algunas de las técnicas más comunes:

2.2.3.2.1. Detección de oraciones

Esta técnica comprende el **nivel sintáctico**, su funcionamiento se basa en recortar una secuencia de letras o caracteres entre dos signos de puntuación acompañados de espacios en blanco.

2.2.3.2.2. Segmentación por palabras

Luego de la identificación de las oraciones, se realiza una segmentación que utiliza el **Nivel léxico**, debido a que busca "claves", las cuales son unidades lingüísticas como las palabras, puntuaciones, números, caracteres alfanuméricos, etc.

2.2.3.2.3. Etiquetado gramatical

Una vez realizadas las técnicas anteriores, se realiza el etiquetado de las palabras de acuerdo con su rol o propósito dentro de una oración, algunas de estas etiquetas son: sustantivo, adjetivo, adverbio, etc.

2.2.3.2.4. Segmentación morfológica

Esta técnica comprende la clasificación de los morfemas (fragmento mínimo capaz de expresar el significado de una palabra) con el fin de realizar un análisis profundo dentro de un texto, para así obtener datos como: género, modo o tiempo. Existen diferentes tipos de morfemas, estos son:

1. **Independientes:** pronombres, preposiciones, conjunciones y determinantes.
2. **Derivativos:** prefijos, sufijos, o interfijos.
3. **Flexivos:** género, número, persona, modo y tiempo.

2.2.3.2.5. Eliminación de "Stop Words"

Elimina aquellas palabras que no aportan valor en una oración, comúnmente son palabras repetitivas. Algunas de estas son: artículos, los pronombres, las preposiciones, y las conjunciones.

2.2.3.2.6. Reconocimiento de Entidades Nombradas

Se encarga de la búsqueda y clasificación de elementos pertenecientes a categorías predefinidas por el usuario. Estas pueden ser: nombres de personas, entidades, organizaciones, lugares, expresiones temporales, cantidades, porcentajes, etc.

2.2.3.2.7. Stemming

Utilizando como resultado la salida de las técnicas anteriores, esta busca eliminar los prefijos y sufijos de un morfema, de manera que se pueda obtener la palabra raíz.

2.2.4. Sistema de Procesamiento de Lenguaje

Un sistema de procesamiento de lenguaje, según Sommerville (2005), se define como el programa o aplicación "que acepta lenguaje natural o artificial como entrada y se enfoca en generar alguna otra representación de ese lenguaje como una salida" (p. 279).

Cuando se habla de este término en ingeniería de *software*, comúnmente se referencia al uso de los compiladores, los cuales traducen un lenguaje de programación artificial de alto nivel a código máquina, pero no sólo se limitan a esto, también se ve el caso de aplicaciones que realizan tareas de conversión de archivos XML a consultas a una base de datos, o bien, traducen un lenguaje natural a uno de otro tipo. Los sistemas de procesamiento de lenguajes se utilizan en escenarios donde la forma más fácil de resolver un problema es especificar esa solución como uno o una serie de algoritmos que brinden o permitan generar una descripción de los datos del sistema.

2.2.4.1. Componentes

Según Sommerville (2005):

Los componentes que forman un sistema de procesamiento de lenguajes pueden organizarse de acuerdo con diferentes modelos arquitectónicos. Puede utilizarse una arquitectura de flujo de datos con la tabla de símbolos actuando como un repositorio para datos compartidos. Las fases de análisis léxico, sintáctico y semántico se organizan de forma secuencial. (p. 280)

Algunos de estos componentes son:

1. Analizador léxico: es la entrada de los símbolos del lenguaje, convierte estos a un formato interno.

2. Tabla de símbolos: se utiliza como un repositorio de datos, el cual almacena la información sobre los nombres de las entidades que se encuentran en el texto de entrada.
3. Analizador sintáctico: se encarga de la comprobación de la sintaxis del lenguaje que se está traduciendo. Este usa la gramática definida por el lenguaje, construye un árbol sintáctico.
4. Árbol sintáctico: es una estructura interna utilizada para la representación del programa que se encuentre en compilación.
5. Analizador semántico: se enfoca en tomar la información del árbol sintáctico y de la tabla de símbolos con el objetivo de comprobar la corrección semántica del texto en el lenguaje de entrada.
6. Generador de código: este se encarga de recorrer el árbol sintáctico anterior y generar el código de salida.

2.2.5. Generación de Lenguaje Natural (GLN)

De acuerdo con Vicente (2015), la GLN se define como la disciplina de investigar cómo se pueden crear sistemas de *software* que sean capaces de producir o generar textos de alta calidad que sean comprensibles por el ser humano, esto por medio de datos estructurados y procesables (ficheros binarios, datos numéricos, bases de datos, etc.), o bien, textos redactados naturalmente.

2.2.5.1. Clasificación de Sistemas de GLN

La clasificación de estos sistemas se realiza con base en los objetivos o necesidades del cliente y, a su vez, la naturaleza del proyecto que se desee llevar a cabo. Estas categorías son:

2.2.5.1.1. Según la entrada del sistema

Es cuando se espera que el sistema reciba datos no contextualizados que no conforman un texto completo, estos datos pueden ser:

- Números.
- Bases de datos.
- Bases de conocimiento.
- Corpus etiquetado (conjunto de datos destinados a una investigación científica).

Estos datos son utilizados con el fin de desarrollar un enfoque de “datos a texto”, el cual permite la generación de resúmenes o informes basados en datos puntuales.

2.2.5.1.2. Según los objetivos del sistema

Esta clasificación se enfoca en el propósito por el cual el sistema ha sido creado, en este caso comúnmente se habla de sistemas para la generación de informes a partir de datos factuales (información objetiva), debido a que se diseñan como herramientas de ayuda con el fin de solventar problemas de comprensión lectora.

2.2.5.2. Enfoques genéricos para abordar GLN

A continuación, con base en la investigación de Vicente (2015), se describen a un alto nivel los enfoques más relevantes para GLN.

2.2.5.2.1. Enfoques basados en conocimiento

Están constituidos por dos subsistemas, los cuales son una base de conocimiento y un motor de inferencia, estos sustentan las bases de información necesarias para el desarrollo de GLN.

2.2.5.2.2. Enfoques estadísticos

Se basan en las probabilidades tomadas de un volumen de texto base, estos pueden ser corpus, textos recopilados o textos originados en la web.

2.2.5.2.3. Enfoques híbridos

Este enfoque combina los enfoques anteriores, resultado de esto, se tiene la mezcla de las técnicas basadas en conocimiento y las probabilidades estadísticas.

2.2.5.3. Fases de un sistema GLN

Según Vicente (2015), existen múltiples formas de desarrollar un sistema de GLN, sin embargo, existen dos puntos claves para esto: la entrada de datos y la salida de información. Por esto, para especificar las fases que se deben realizar para lograr que un GLN satisfaga su propósito, se necesitan contemplar aspectos como: lexicalización y generación de expresiones de referencia, agregación y segmentación, y estructuración retórica y de ordenamiento.

Con base en lo descrito anteriormente, Vicente (2015) propone siete tareas o fases para desarrollar un GLN, las cuales se agrupan en tres módulos:

2.2.5.3.1. Macro planificación

Se refiere a la planificación y estructura de la información que define el "qué" y "dónde" del proyecto. Esto se desarrolla por medio de las tareas de:

- Selección de contenido.
- Estructuración del documento.

2.2.5.3.2. Micro planificación

Tomando como base la **Macro planificación**, se deciden las palabras o referencias adecuadas, se establecen estructuras para los mensajes y se agrupa la información en oraciones. Todo esto por medio de:

- Agregación.
- Lexicalización.
- Generación de expresiones referenciales.

2.2.5.3.3. Realización

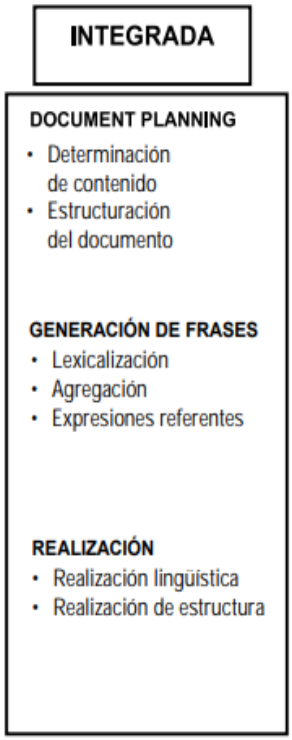
Al ser la última fase, este módulo genera la salida final del sistema, sea este texto o habla, por medio de la conformación de oraciones concretas basadas en la estructura definida. Las tareas de esta fase son:

- Realización lingüística.
- Realización de la estructura.

2.2.5.4. Arquitectura Integrada

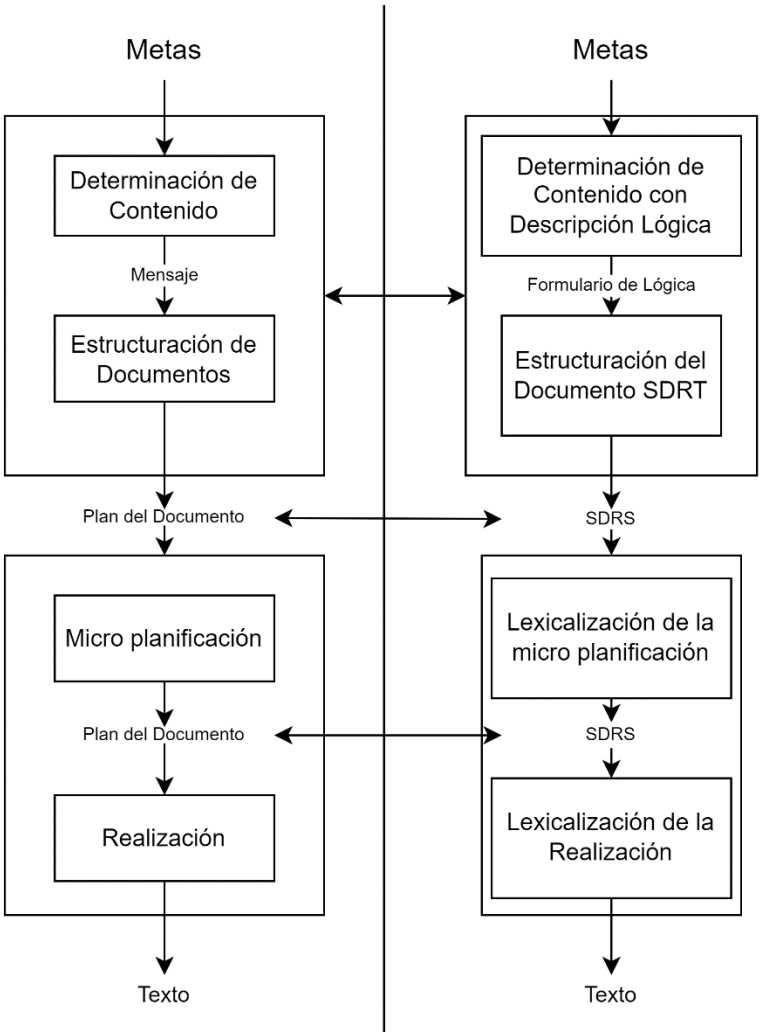
La forma en la que se conectan estas fases se conoce como arquitectura. Chala, G. J. (2007) propone una Arquitectura Integrada para la realización de sistemas de Generación de Lenguaje Natural (GLN), esta permite tomar todas las decisiones de un proceso estructurado de manera jerárquica, sin modularizar. En la **Figura 6**. se muestra la arquitectura propuesta por el autor, donde se entiende que el *Document planning* es el equivalente a la **Macro planificación** y la Generación de Frases a la **Micro planificación**, descritas anteriormente. También en la **Figura 7**, se muestra la comparación entre esta arquitectura y una estándar.

Figura 6:
Diseño de Arquitectura Integrada de un Sistema GNL.



Nota. Fuente: Tomado de Chala, G. J., Jordán, R. A., y Linares, D. L. (2007). GenLeNa: Sistema para la construcción de Aplicaciones de Generación de Lenguaje Natural. Sistemas & Telemática, 5(9), 45-60.

Figura 7:
La arquitectura estándar de un sistema NLG.



Nota. Adaptación de Danlos, L., y El Ghali, A. (2002). A complete integrated NLG system using AI and NLU tools. In Proceedings of COLING'02.

2.3. Investigación

Según Hernández (2014), se conoce como investigación a "un conjunto de procesos sistemáticos, críticos y empíricos que se aplican al estudio de un fenómeno o problema" (p. 25).

Una investigación puede tener un enfoque cuantitativo o cualitativo, y aunque ambos compartan estrategias o conceptos, cada uno posee sus propias características.

2.3.1. Tipos de Investigación

Una investigación puede tener un enfoque cuantitativo o cualitativo, y aunque ambos compartan estrategias o conceptos, cada uno posee sus propias características.

Estos tipos se detallan a continuación:

2.3.1.1. Tipo Cuantitativo

Este enfoque busca reflejar la necesidad de medir y estimar magnitudes de los fenómenos o problemas de investigación, es decir ¿cada cuánto sucede el problema?, o bien, ¿qué magnitud tiene el problema o sus consecuencias?

2.3.1.1.1. Enfoques de Alcances para Investigaciones Cuantitativas

Según Hernández (2014), existen diferentes alcances para las investigaciones cuantitativas, estos son:

- Exploratorios: investigan problemas poco estudiados, indagan desde una perspectiva innovadora y ayudan a identificar conceptos promisorios.
- Descriptivos: consideran al fenómeno estudiado y sus componentes, miden y definen variables.
- Correlacionales: permiten predicciones y cuantifican relaciones entre conceptos.
- Alcances: resultan de la revisión de la literatura y de la perspectiva del estudio, y dependen de los objetivos del investigador para combinar estos elementos.
- Explicativos: determinan las causas de los fenómenos, generan un sentido de entendimiento y son sumamente estructurados.

2.3.1.2. Tipo Cualitativo

Se conoce como un conjunto de prácticas interpretativas, las cuales muestran el mundo de manera visible. Este enfoque utiliza la recolección y análisis de los datos con el objetivo de formular preguntas de investigación. Estas preguntas se pueden realizar en cualquier etapa de la investigación, y las actividades son para lograr descubrirlas, perfeccionarlas y responderlas. Toman, transforman y convierten los datos en observaciones, anotaciones y documentos.

2.3.1.2.1. Enfoques de Diseño para Investigaciones Cualitativas

Según Hernández (2014), existen diferentes enfoques de diseño para las investigaciones cualitativas, estos son:

- Etnográficos: describe y explica los elementos y categorías que integran al sistema social.
- Fenomenológicos: responden preguntas sobre experiencias comunes y distintas categorías que se presentan en estas.
- Investigación-acción: proporciona un diagnóstico de problemáticas sociales, políticas, laborales, económicas o de naturaleza colectiva. Permite categorizar las causas y consecuencias de las problemáticas y sus soluciones.
- Narrativos: abordan información sobre historias, procesos, hechos, eventos y experiencias; siguen una línea de tiempo y están ensamblados en una narrativa general.
- Teoría fundamentada: proporciona información sobre las categorías del proceso o fenómeno y sus vínculos.

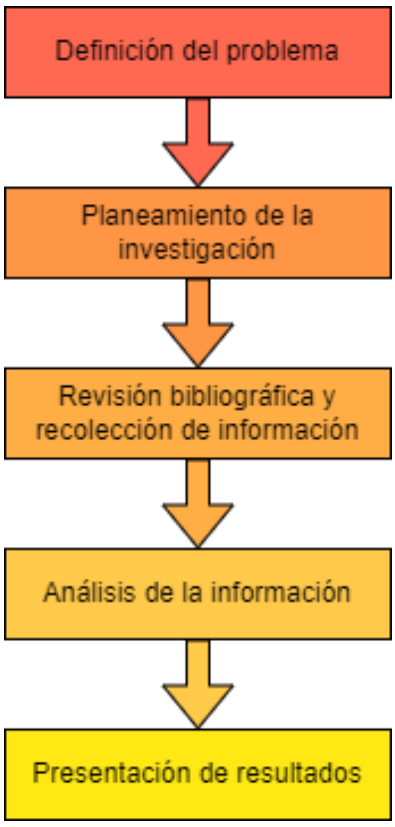
2.3.2. Proceso de Investigación

El proceso de investigación es un conjunto de actividades que busca localizar la información para un proyecto de investigación. Este permite obtener conocimientos científicos, los cuales serán objetivos, sistemáticos, claros, organizados y verificables. El sujeto responsable de la realización, supervisión y el desarrollo de las distintas tareas de este proceso se denomina investigador.

Según Sabino (1992), los objetos de estudio son los infinitos temas y problemas que reclaman la atención del científico, que suelen agruparse y clasificarse según las distintas ciencias o especialidades existentes.

En la **Figura 8,** se muestra una adaptación sobre las etapas del proceso de investigación según Sabino (1992).

Figura 8:
Etapas del Proceso de Investigación.



Nota. Adaptación del Proceso de Investigación. Sabino (1992).

2.3.2.1. Métodos de investigación de problemas

El método de investigación de problemas es el procedimiento o conjunto de procedimientos que se usan con el fin de obtener conocimientos, el modelo de trabajo o secuencia lógica que orienta la investigación.

De acuerdo con Hernández (2014), para analizar y lograr conceptualizar el problema de investigación, se pueden utilizar herramientas como:

2.3.2.1.1. Diagrama de Causa – Efecto

Este diagrama es una herramienta de calidad, la cual permite un análisis de los posibles problemas que se presentan dentro de una organización, equipo, sistema o donde sea que se desee investigar, y estos se relacionan con un efecto y sus posibles causas.

También, según Ponce (2007), una herramienta poderosa para la identificación y entendimiento del estado de una organización o proyecto es:

2.3.2.1.2. Análisis FODA

Este análisis consiste en una herramienta o instrumento que permite realizar un análisis organizacional con relación en factores que determinen el éxito en el cumplimiento de metas. También, este análisis estima a grandes rasgos el efecto que una estrategia tiene con el fin o enfoque de obtener un equilibrio o ajuste entre la capacidad interna de la organización y el entorno externo a esta.

Otra herramienta para la investigación de problemas, de acuerdo con Martínez y Fernández (2018), es el:

2.3.2.1.3. Árbol de Problemas

Este diagrama consiste en el desarrollo de ideas creativas con el propósito de identificar las posibles causas de un problema, de manera que se genera una forma organizada que explique las razones y consecuencias de este.

3. Marco Metodológico

Este capítulo presenta los elementos metodológicos que orientarán el desarrollo del proyecto. En particular, se habla del tipo de investigación, diseño de la investigación, fuentes de investigación, sujetos de investigación, variables de la investigación, instrumentos de investigación, procedimiento metodológico de la investigación, entre otros; esto para poder alcanzar los objetivos propuestos.

3.1. Tipo de Investigación

Tomando como referencia el punto **Marco Conceptual**, en el presente proyecto es una investigación aplicada, esto debido a que tiene como objetivo resolver un problema específico o desarrollar una aplicación práctica para satisfacer necesidades concretas.

3.2. Enfoque de Investigación

Tomando como referencia los conceptos presentados en el **Marco Conceptual**, se selecciona el tipo de enfoque cualitativo, esto de acuerdo con lo planteado anteriormente en el apartado **Objetivos del Trabajo Final de Graduación**. La naturaleza de estos objetivos implica una serie de actividades como recopilación, análisis y relación entre los datos cualitativos en la investigación. Además, se estudian cualidades que satisfagan o argumenten la solución propuesta, así como cada una de sus partes.

Otras razones que justifican la elección de este tipo de investigación son:

1. El proyecto, al no ser un proceso de negocio, no posee datos de uso o resultado, por ende, no permite un análisis o comparación de resultados cuantitativos.
2. El resultado esperado de este proyecto corresponde a una propuesta de solución basada en información y recopilación de datos, debido a que depende de requerimientos definidos previamente, de manera que estos ayudan a determinar la funcionalidad y uso.
3. Se realizará un análisis de costos, sin embargo, este no se analizará con el fin de compararlo con algún otro análisis o información, sino que será un insumo y argumento para propuestas de negocio.

A continuación, en la **Tabla 9**, se muestra el enfoque investigativo por cada objetivo del proyecto:
Tabla 9:
Enfoque investigativo por cada objetivo.

ID	Objetivo	Enfoque investigativo
Ob-01	Identificar la situación actual del sistema Kleeen Software IDE para la comprensión y descripción detallada de las necesidades y debilidades actuales de la empresa, esto con el fin de ser	Cualitativo

ID	Objetivo	Enfoque investigativo
	utilizadas en una futura integración a la propuesta de diseño del sistema de generación de lenguaje natural.	
Ob-02	Especificar los requerimientos funcionales y no funcionales para el diseño del sistema de generación de lenguaje natural, de acuerdo con la norma ISO/IEC/IEEE 29148, así como la organización y diseño de los componentes requeridos para el desarrollo del sistema.	Cualitativo
Ob-03	Evaluar tecnologías o plataformas de generación de lenguaje natural con el objetivo de recomendar aquellas que ofrezcan mejores medidas de aceptación y evaluación de los requerimientos para la propuesta planteada.	Cualitativo
Ob-04	Determinar el costo de implementación, así como el ROI relacionado con la propuesta de diseño de *software* por integrar en el sistema Kleeen Software IDE.	Cuantitativo (no comparativo)

Nota. Fuente: Elaboración propia.

3.3. Alcance de la Investigación

De acuerdo con Hernandez (2014), una investigación descriptiva es aquella que busca especificar las propiedades, características y perfiles de casos de un proyecto, así como cuantificar conceptos, y variables.

Por esto, la presente investigación posee un alcance descriptivo, ya que mide fenómenos o variables para estimar su ocurrencia o magnitud dentro del proyecto, esto respondiendo preguntas como: ¿Cuántos? ¿Qué cantidad? ¿Qué proporción o porcentaje? ¿Qué número? ¿Cuán frecuente? ¿Cuán a menudo? ¿Con qué frecuencia o periodicidad?.

3.4. Diseño de la Investigación

Tomando como referencia el punto **Marco Conceptual**, este proyecto presenta un enfoque de tipo investigación/acción práctico, esto debido a que responde preguntas sobre problemáticas o situaciones de un grupo (incluyendo cambios). Su desarrollo se basa en el diagnóstico de una problemática, así como el entendimiento de las categorías sobre las causas, consecuencias y sus soluciones.

También, la investigación cumple con las siguientes características:

1. Estudia prácticas locales.
2. Involucra indagación individual o en equipo.
3. Implementa un plan de acción o propuesta de solución (para resolver el problema, introducir la mejora o generar el cambio).

3.5. Fuentes de Investigación

Como parte del proceso investigativo, es necesaria la recolección y consulta de información, para esto, en el presente apartado se definen las fuentes de información que se consultarán.

De acuerdo con Monje (2011), las fuentes de investigación se dividen en dos:

- Primarias: son un escrito personal que se basa en las propias experiencias, investigaciones y resultados.
- Secundarias: son un escrito acumulativo que se basa en las experiencias y teorías de otros autores.

De acuerdo con Ulate, I. y Vargas, E. (2016), se establecen las fuentes de investigación como las presentadas en la **Tabla 10.**

Tabla 10:
Fuentes de Investigación.

Fuentes Primarias	Fuentes Secundarias
• Consulta a un experto. • Persona que observa el evento (testigo). • Escritos de una persona cuya biografía está construyendo. • Libros. • Artículos de publicaciones periódicas. • Artículos de revistas científicas y ponencias. • Trabajos presentados en congresos y ponencias, o simposios. • Monografías. • Tesis académicas. • Disertaciones. • Documentos oficiales. • Reportes de asociaciones. • Testimonios de expertos. • Documentales.	• Comentarios de libros, tesis, disertaciones y otros documentos especializados. • Índices que incluyen los datos de las referencias y un breve resumen de cada una.

Nota. Fuente: Ulate, I. & Vargas, E. (2016).

Con base en lo anterior, se establecieron las siguientes fuentes primarias de investigación:

- Revisión Documental:
  - Página web oficial de la empresa: información original y actualizada sobre la empresa.
  - Libro "Ingeniería del *software*", Séptima edición, por Ian Sommerville: información sobre el proceso de ingeniería de *software* (requerimientos, modelos, arquitecturas, etc.).
  - Libro "*Advanced Analytics: Moving Toward AI, Machine Learning, and Natural Language Processing*", por Fern Halper: información sobre inteligencia artificial y procesamiento de lenguaje natural, tecnologías, y mejores prácticas.
  - Libro "Manual de UML", por Paul Kimmel: información sobre la creación y el uso de los diagramas UML.
  - Proyectos similares realizados dentro de la empresa: información sobre estándares, diseños, arquitecturas, y tecnologías ya establecidas.
  - Trabajos finales de graduación externos a la empresa: información relevante de interés, tal como investigaciones, técnicas, mejores prácticas y posibles soluciones para el presente proyecto.

Además, se establecieron las siguientes fuentes secundarias de investigación:

- Revisión Documental:
  - Documentación en el espacio de *Confluence* de la empresa: información, acuerdos, resultados de investigaciones, realizados por los colaboradores de la empresa.

3.6. Sujetos de Investigación

De acuerdo con Hernández (2014), los sujetos de investigación son los que incluyen a aquellos grupos que interactúan por un periodo extendido, que están ligados entre sí por una meta y que se consideran a sí mismos como una entidad.

En la **Tabla 11** se define los siguientes:

Tabla 11:
Sujetos de Investigación.

Rol del sujeto	Años de experiencia en el rol	Caracterización del sujeto	Justificación de la importancia del sujeto
Vicepresidente de ingeniería	13	Frecuenta reuniones con los equipos de ingeniería, administrador de proyectos y gestor de información.	Es un punto clave debido a la cercanía con ingeniería, así como proveedor de requerimientos e información importante para el proyecto.
Desarrollador de *Software*	11	Desarrollo de los sistemas de la empresa, gestor de información y apoyo en decisiones de arquitectura.	Por su experiencia, posee conocimiento de primera mano con respecto a aspectos técnicos de la empresa.
Arquitecto de *Software*	35	Desarrollo de los sistemas de la empresa, realiza decisiones de arquitectura y desempeño.	Por su experiencia, posee conocimiento de primera mano con respecto a aspectos técnicos de la empresa.
Cliente	--	Es el usuario final del sistema Kleeen Software IDE, y por ende el posible usuario final del sistema producto de la propuesta de diseño del presente proyecto.	Brinda una perspectiva clara y real de los requerimientos o necesidades de uso del sistema Kleeen Software IDE y datos que servirán como insumos para la propuesta de diseño del sistema de generación de lenguaje natural.

Nota. Fuente: Elaboración propia.

3.7. Variables de la Investigación

Según Rivas (2015), las variables de investigación se definen como “algo” que, aunque resulte tautológico, varía o cambia de valores. A su vez, a una variable se le asocia un factor decisivo en la explicación o argumentación de un fenómeno o estado.

Por esto, en la **Tabla 12** se definen las siguientes variables de investigación para el presente proyecto:

Tabla 12:
Variables de la Investigación

Objetivo	Acción	Variable	Concepto	Indicadores	Instrumentos
Ob-01	Identificar	Estado del sistema Kleeen Software IDE	Contextualización y estado técnico actual del sistema Kleeen Software IDE necesario para la comprensión y descripción detallada de las necesidades y debilidades actuales.	• Cantidad de fortalezas, oportunidades, debilidades, y amenazas identificadas (Cuadro FODA)	• Entrevista no estructurada • Minutas de reunión
		Grado de satisfacción con el sistema Kleeen Software IDE	Consiste en recopilar la información para generar una descripción detallada sobre las opiniones y posibles aspectos de mejora del sistema Kleeen Software IDE.	• Porcentaje identificado de satisfacción del usuario • Opinión de los usuarios	• Encuesta - Escala de Likert

Objetivo	Acción	Variable	Concepto	Indicadores	Instrumentos
Ob-02	Especificar	Requerimientos del sistema de generación de lenguaje natural	Consiste en recopilar la información para definir las características del diseño de *software*.	• Lista de requerimientos especificados	• Entrevista no estructurada • Minutas de reunión
		Organización y diseño de componentes	Consiste en los diagramas de casos de uso, componentes, y secuencia de las actividades.	• Cantidad de diagramas especificados	• Revisión documental
Ob-03	Evaluar	Lista de tecnologías de generación de lenguaje natural	Se describen las cualidades de las tecnologías pertinentes para recomendar aquellas que ofrezcan mejores medidas de aceptación.	• Cantidad de tecnologías evaluadas. • Porcentaje de cumplimiento de las medidas de aceptación.	• Revisión documental
		Requerimientos del sistema de generación de lenguaje natural	Consiste en realizar una categorización y evaluación de los requerimientos funcionales para priorizar el diseño y uso de estos.	• Cantidad de requerimientos funcionales evaluados.	• Revisión documental • Minutas de reunión

Objetivo	Acción	Variable	Concepto	Indicadores	Instrumentos
Ob-04	Determinar	Costo de implementación	Es la estimación de los costes de implementación del proyecto.	• Cantidad de recursos requeridos. • Tiempo destinado a implementación. • Total de costos de implementación determinados.	• Revisión documental • Minutas de reunión
		ROI	Compara el beneficio esperado con la inversión prevista para visualizar futuros escenarios en aspectos financieros.	• Retorno de la inversión pronosticado.	

Nota. Fuente: Elaboración propia.

3.8. Instrumentos de Investigación

Para Martínez (2013) existen herramientas llamadas "instrumentos de investigación", las cuales permiten la recolección, medición y análisis de datos, de manera que se obtiene información como insumo para realizar una investigación.

De acuerdo con Ulate y Soto (2016), estos instrumentos se pueden clasificar de acuerdo con el enfoque que se presente en la investigación, de igual manera, estos pueden mezclarse o compartirse con el fin de obtener mejores resultados.

En la **Tabla 13** se muestra la clasificación que proponen estos autores:

Tabla 13:
Instrumentos de Investigación.

Enfoque Cualitativo	Enfoque Cuantitativo
• Observación. • Entrevistas. • Sesiones de grupos o reuniones (minutas). • Biografías. • Revisión documental. • Etnografías.	• Cuestionarios. • Registros de datos estadísticos. • Pruebas estandarizadas. • Entrevistas. • Encuestas.

Nota. Fuente: Ulate, I. y Vargas, E. (2016).

Para el presente proyecto se utilizarán los siguientes instrumentos de investigación:

3.8.1. Entrevista

De acuerdo con (Kvale, 2008), se entiende la entrevista cualitativa como un instrumento vital al momento de explorar o indagar a los sujetos de investigación, esto con el propósito de descubrir cómo experimentan y entienden su mundo.

Existen diferentes tipos de entrevistas:

- ➢ Estructuradas: el entrevistador no posee libertad de agregar preguntas, esta entrevista se realiza siguiendo una guía o cuestionario establecido.
- ➢ Semiestructuradas: el entrevistador se basa en una guía, este posee poca libertad de conducir la entrevista.
- ➢ No estructuradas o abiertas: se utiliza una guía general no estructurada y el entrevistador posee absoluta libertad de conducir la entrevista.

Los instrumento de investigación utilizado para aplicar las entrevistas se muestran en los **Apéndice D**, **Apéndice E**, **Apéndice F**, y **Apéndice G**.

Como resumen, se realizó la XXX:

Tabla 14:
Instrumentos de la Entrevista

Nombre Entrevista	Objetivo	Sujeto Entrevistado	Referencia
Entrevista para la Identificación del Problema y Situación Actual.	Determinar el problema y situación actual.	Vicepresidente de Ingeniería	**Apéndice D**
Entrevista para la recopilación y análisis de los requerimientos	Recopilar y analizar los requerimientos.	Vicepresidente de Ingeniería	**Apéndice E**
Entrevista para la comprensión de la situación actual del sistema Kleeen Software IDE	Comprender la situación actual del sistema Kleeen Software IDE	Arquitecto de Software	**Apéndice F**
Entrevista para la conocer el tiempo y costo relacionado a la documentación de un proyecto dentro de la empresa	Conocer el tiempo y costo relacionado a la documentación de un proyecto dentro de la empresa	Arquitecto de Software y vicepresidente de ingeniería (opcional)	**Apéndice G**

Nota. Fuente: Elaboración propia.

3.8.1. Revisión documental

Según Hurtado (2008), se conoce como revisión documental a la técnica en la cual se recolecta información sobre un determinado tema con el propósito de obtener variables relacionadas con la investigación, ya sea de manera directa o indirecta, y así comprender el estado actual de la problemática existente.

3.8.2. Minutas de reunión

De acuerdo con Ávila (2017), la minuta es el recurso escrito que registra los hechos o temas conversados en una reunión o audiencia, en estos se toman notas de asistentes, ideas abordadas, acuerdos y conclusiones. El instrumento de investigación utilizado para aplicar las minutas de reunión se muestra en el **Plantilla de minutas**.

3.8.3. Escala de Likert

De acuerdo con Orellana (2006), la observación es el análisis en primera persona que realiza el investigador de la situación o problemática de estudio. Esta debe ser de forma directa, entera y en el momento justo, aquí la participación varía según el propósito y el diseño de investigación previstos.

De acuerdo con Ávila (2017), la escala de Likert es una escala de medición que se utiliza para la comprensión de opiniones, y así medir el grado de aceptación o conformidad; o su contraparte, el grado de enajenación o inconformidad de un usuario o afectado con la situación o problemática de estudio.

3.9. Procedimiento metodológico de la investigación

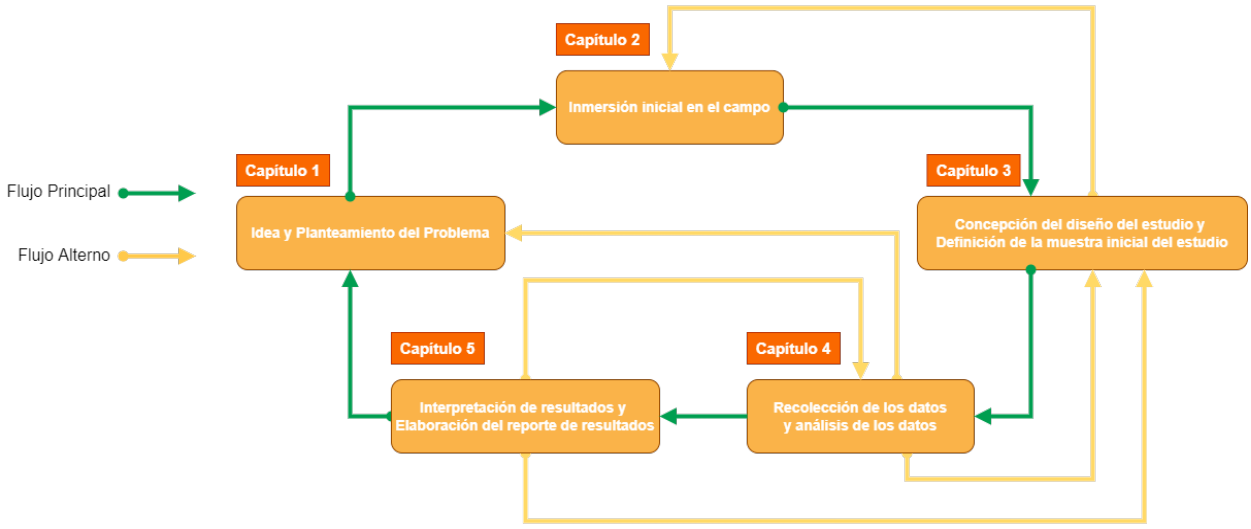
Según Hernández (2014), las fases para este enfoque de investigación son:

Fase 1. Idea.
Fase 2. Planteamiento del problema.
Fase 3. Inmersión inicial en el campo.
Fase 4. Concepción del diseño del estudio.
Fase 5. Definición de la muestra inicial del estudio.
Fase 6. Recolección de los datos .
Fase 7. Análisis de los datos.
Fase 8. Interpretación de resultados.
Fase 9. Elaboración del reporte de resultados.

Las fases enumeradas anteriormente no suelen ser estrictamente secuenciales debido a que se puede visitar una fase tantas veces como sea necesario a lo largo del desarrollo de la investigación.

Con base en lo anterior, para el presente proyecto se propone el proceso que se muestra en la **Figura 9.**

Figura 9:
Procedimiento metodológico de la investigación



Nota. Fuente: Elaboración propia.

Como se observa en la **Figura 9**, se plantea un flujo principal, el cual se representa con las flechas verdes. Las fases descritas anteriormente son agrupadas por cada capítulo del presente documento, de manera que cada actividad realizada tendrá como resultado un capítulo especializado. Las flechas amarillas muestran cómo el flujo principal puede variar de acuerdo con los resultados obtenidos por cada fase.

Cada capítulo del presente proyecto se relaciona con una o varias fases en específico, y a su vez, estas poseen tareas por desarrollar, las cuales son:

- Capítulo 1: Introducción
  - Idea:
    - Recopilar información sobre la empresa.
  - Planteamiento del Problema:
    - Identificar la situación problemática.
    - Establecer los objetivos del proyecto.
    - Delimitar el alcance.
- Capítulo 2: Marco Conceptual
  - Inversión inicial en el campo
    - Definir conceptos básicos necesarios para comprender el contexto del proyecto.
    - Definir los conceptos necesarios para elegir la metodología por aplicar.

- Capítulo 3: Marco Metodológico
  - Concepto de Diseño del Estudio:
    - Definir el tipo de investigación.
    - Definir el enfoque de investigación.
    - Definir el alcance de investigación.
    - Definir el diseño de investigación.
  - Definición de la muestra inicial del estudio:
    - Definir las fuentes de investigación.
    - Definir los sujetos de investigación.
    - Definir las variables de investigación.
    - Definir los instrumentos de investigación.
    - Definir el procedimiento metodológico.
- Capítulo 4: Análisis de Resultados
  - Recolección de los datos:
    - Recopilación de Información.
  - Análisis de los datos:
    - Análisis de la situación actual
    - Experimento de Mago de Oz
    - Análisis de satisfacción del usuario
- Capítulo 5: Análisis de Resultados
  - Interpretación de resultados y elaboración de reporte de resultados:
    - Diseño de la propuesta.
    - Estudio de costos.

De manera que cada capítulo se enfoca en una fase en específico, logrando así ejecutar tareas que permitan el desarrollo del presente proyecto.

3.10. Matriz de trazabilidad

A continuación, en la **Tabla 14**, se presenta la relación que poseen los objetivos específicos con el marco conceptual, metodología, análisis de resultados, conclusiones, recomendaciones y apéndices.

Tabla 15:
Matriz de trazabilidad

ID	Objetivo específico	Marco conceptual	Metodología	Análisis de resultados
Ob-01	Identificar la situación actual del sistema Kleeen Software IDE para la comprensión y descripción detallada de las necesidades y debilidades actuales de la empresa, esto con el fin de ser utilizadas en una futura integración a la propuesta de diseño del sistema de generación de lenguaje natural.	• **Proceso del *Software***	1. Idea 2. Planteamiento del problema 3. Inmersión inicial en el campo	• **Recopilación de información de Debilidades del sistema Kleeen Software IDE**
Ob-02	Especificar los requerimientos funcionales y no funcionales para el diseño del sistema de generación de lenguaje natural, de acuerdo con la norma ISO/IEC/IEEE 29148, así como la organización y diseño de los componentes requeridos para el desarrollo del sistema.	• **Proceso del *Software*** • **Sistema de Procesamiento de Lenguaje** • **Generación de Lenguaje Natural (GLN)**	4. Concepción del diseño del estudio 5. Definición de la muestra inicial del estudio	• **Necesidades del sistema Kleeen Software IDE**

ID	Objetivo específico	Marco conceptual	Metodología	Análisis de resultados
Ob-03	Evaluar tecnologías o plataformas de generación de lenguaje natural con el objetivo de recomendar aquellas que ofrezcan mejores medidas de aceptación y evaluación de los requerimientos para la propuesta planteada.	• **Application Programming Interfaces (API)**	6. Recolección de los datos 7. Análisis de los datos	• **Evaluación de Tecnologías**
Ob-04	Determinar el costo de implementación, así como el ROI relacionado con la propuesta de diseño de *software* por integrar en el sistema Kleeen Software IDE.	• **Investigación**	8. Interpretación de resultados 9. Elaboración del reporte de resultados	• **Estudio de Costos**

Nota. Fuente: Elaboración propia.

4. Análisis de Resultados

Los datos que se producen al aplicar todas las actividades e instrumentos descritos en el marco metodológico deben ser estudiados, comparados y contrastados para formular resultados, es decir, se deben analizar.

En el presente capítulo se muestra un aporte crítico para el Trabajo Final de Graduación, pues este contiene un grado de profundidad sobre la situación actual del problema abordado. Se discuten las razones que soportan la propuesta de alternativas de solución al problema, así como también se evidencian las ventajas de implementar dichas soluciones.

A continuación, se presentan los resultados obtenidos luego de una recopilación de información, así como de las observaciones realizadas para identificar el estado de la situación actual del sistema Kleeen Software IDE.

4.1. Recopilación de información

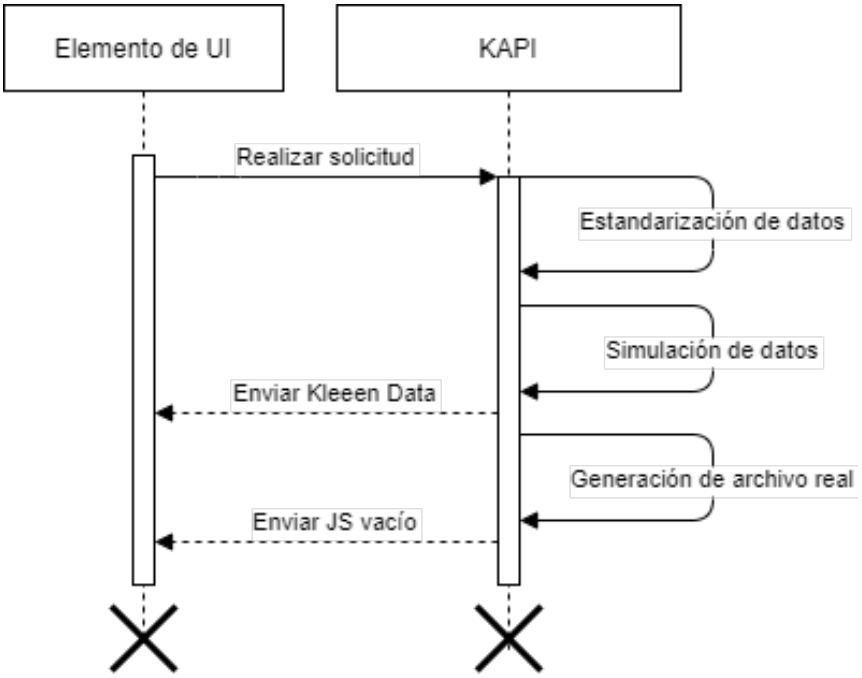
Con respecto al **Procedimiento metodológico de la investigación**, se contempla la Fase 1. En este apartado se exponen los resultados obtenidos luego de aplicar una revisión documental de fuentes teóricas de *software* y documentación interna provista por la organización. La información recopilada en esta sección fue producto de la investigación y la reunión con el arquitecto de software de la empresa (ver **Minuta Reunión 2**) y al entrevistar al arquitecto de software de la empresa (ver **Apéndice H.c**) se obtiene como resultado la siguiente información:

4.1.1. Entendimiento de KAPI

KAPI es una tecnología desarrollada por el equipo de Kleeen para actuar como intermediario entre la interfaz de usuario (también generada por Kleeen) y el *middleware* creado por los clientes. KAPI (basado en las necesidades del cliente) crea un archivo de esqueleto que el usuario final debe aprovechar para traer los datos reales al *front-end,* especificando tanto las llamadas como el formato de estos. Debido a que el archivo de esqueleto es independiente de la API, se pueden usar varias APIs en la misma llamada KAPI para obtener datos, luego combinarlos y devolverlos al *front-end*. El **Anexo II** presenta los tipos de datos utilizados y comprendidos por KAPI.

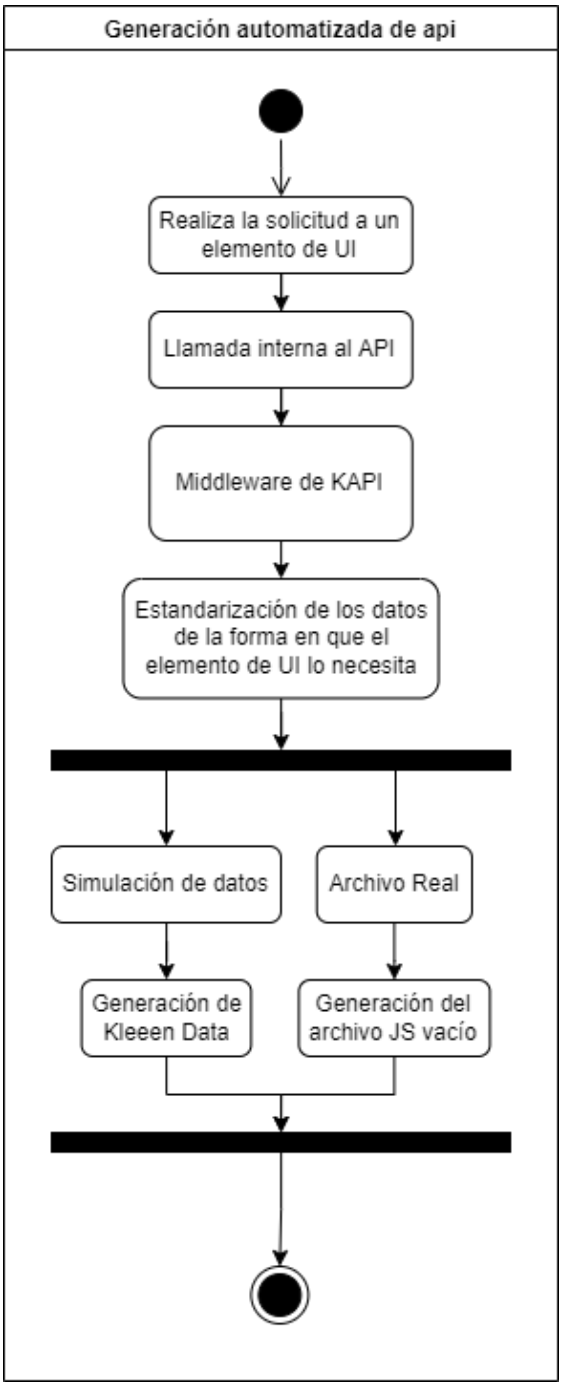
KAPI también verifica si los datos devueltos coinciden con las especificaciones JSON y, si estos fallan, los rechaza. A su vez, proporciona un *back-end* falso para los prototipos, lo que permite ejecutar una interfaz de usuario sin ningún tipo de conexión, con fines de prueba, control de calidad y depuración. En la **Figura 10**. se muestra el diagrama de secuencia y en la **Figura 11** el diagrama de actividad se muestra el flujo de operación y resultado que posee KAPI.

Figura 10:
Diagrama de Secuencia: Flujo de operación y resultado de KAPI.



Nota. Fuente: Elaboración propia.

Figura 11:
Diagrama de Actividad: Flujo de operación y resultado de KAPI.



Nota. Fuente: Elaboración propia.

4.1.2. Entendimiento del Modelo Conceptual (CM)

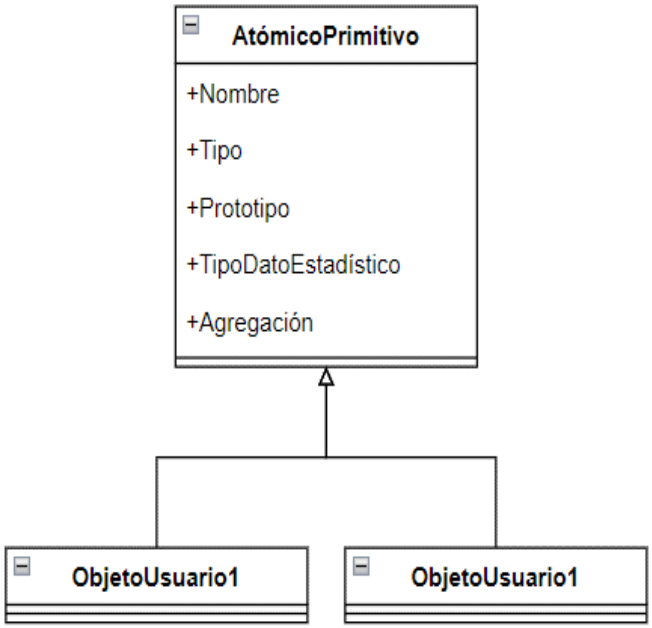
En un alto nivel, el modelo conceptual describe la forma en que se almacena una descripción del contexto que rodea un prototipo o producto de un cliente, este contempla tanto el contenido que se expondrá a los usuarios finales como la forma en que todo se encuentra interconectado.

Desde la perspectiva de Kleeen Software, el modelo conceptual es el sistema que cierra la brecha entre los objetivos de la tarea/producto (como los define el usuario creador) y los datos (y las funciones basadas en los datos) que respaldan esos objetivos.

La experiencia de crear un CM permite alinear la comprensión simplista de un usuario final con sus datos y sus relaciones. Por esto, el usuario debe ingresar cuál o cuáles entidades posee su producto (esto partiendo de un tipo de dato atómico soportado por el sistema) de manera que luego pueda añadirle atributos. Por lo tanto, todo corresponde a una entidad que tiene atributos, y a medida que el usuario final manipula su producto, solo debe pensar en agregar atributos a cualquier entidad previamente determinada.

A continuación, en la **Figura 12,** se muestra una adaptación de la estructura básica de alto nivel de un modelo conceptual desarrollado por la empresa, representado como una herencia de objetos.

Figura 12:
Diagrama de Clases: Jerarquía del Modelo Conceptual



Nota. Adaptación de Kleeen Software

En la **Figura 12** se muestra cómo los tipos primitivos y *Kleeen Atomics* solo existen una vez, y todas las entidades creadas por el usuario se conectan directamente a ellas, de manera que los usuarios pueden obtener referencias a los objetos básicos, pero no pueden cambiarlos.

4.1.3. Entendimiento del sistema Kleeen Software IDE

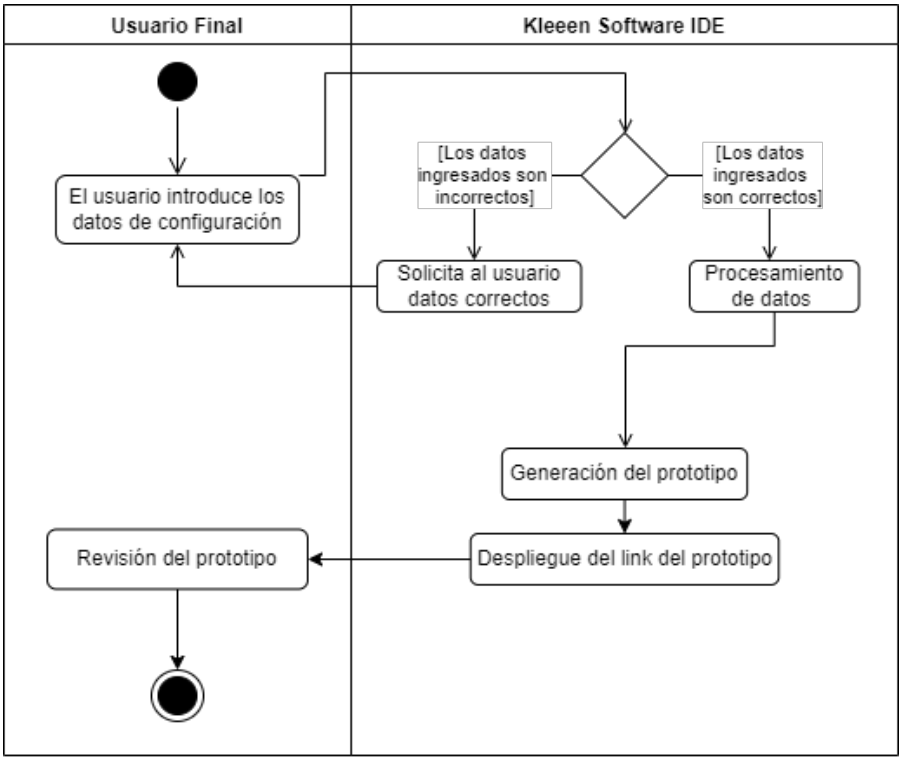
El objetivo de Kleeen Software IDE es, en primera instancia, simplificar radicalmente la creación de productos empresariales. El sistema permite la generación de código funcional como UI, *back-end* falso y conexiones API con Entendimiento de **KAPI**. Además, el sistema se encarga de la gestión de los detalles de implementación, de manera que el usuario final no se preocupe por discutir o decidir aspectos como la vinculación cruzada entre componentes o páginas, dimensiones de filtro seleccionadas automáticamente, gestión de la navegación, gestión de la estructura del código, etc.

Este sistema permite que el usuario se enfoque en el "qué" ver, no en "cómo" verlo. Un ejemplo de esto es sugerir cómo se pueden combinar los datos en lugar de elegirlos manualmente, o bien, el hecho de enfocarse en la intención del usuario y no en la implementación. Esto es sumamente importante para aquellos usuarios que no son expertos en desarrollo, diseño, o UI/UX.

Al automatizar el estilo, el diseño de la experiencia de usuario e interfaz de usuario, la selección de visualización, la ingeniería de *front-end* y las conexiones de *middleware*, es posible permitir que un miembro del equipo sin experiencia en programación construya algo de manera fácil y funcional con el fin de obtener una demostración, un prototipo o incluso un producto terminado. Para esto, el sistema debe recibir datos proporcionados por el usuario, de manera que pueda aprender lo necesario para lograr construir un producto.

En la **Figura 13** se muestra un diagrama de actividad del flujo normal que realiza un usuario final para crear un prototipo de gestión empresarial con el sistema de Kleeen Software IDE.

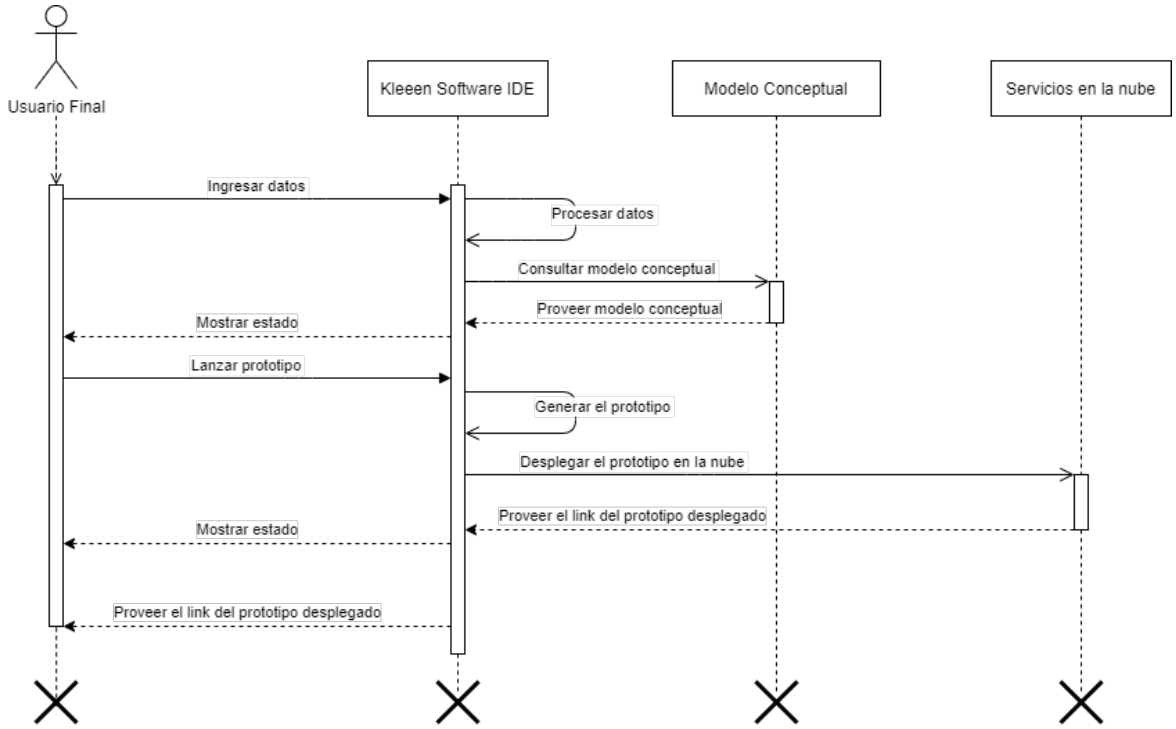
Figura 13:
Diagrama de Actividad: Interacción usuario final y el sistema Kleeen Software IDE.



Nota. Fuente: Elaboración propia.

A continuación, en la **Figura 14** se muestra un diagrama del flujo normal que sucede entre la interacción del usuario final con el sistema de Kleeen Software IDE para crear un prototipo de gestión empresarial.

Figura 14:
Diagrama de Secuencia: Interacción usuario final y el sistema Kleeen Software IDE



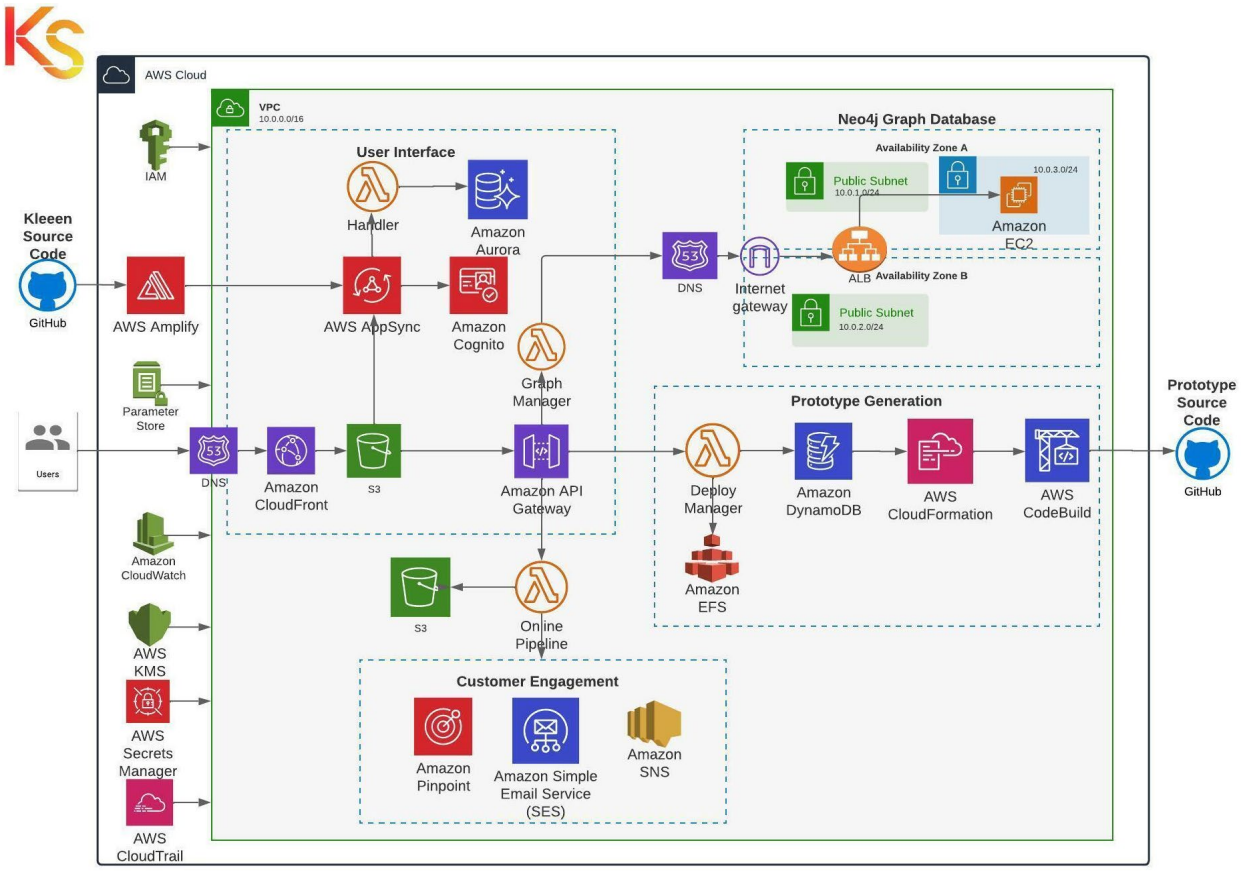
Nota. Fuente: Elaboración propia.

4.1.4. Entendimiento de la arquitectura del sistema Kleeen Software IDE

El sistema Kleeen Software IDE utiliza diferentes servicios en la nube por medio de tecnologías como *Amazon Web Services (AWS)*, las cuales brindan acceso a lambdas, sistemas de registro y *pool* de usuarios como *Cognito, AppSync*, entre otras.

En la **Figura 15**. se muestra la arquitectura de servicios utilizada. También, en el **Anexo III** se encuentra un diagrama que muestra esta arquitectura.

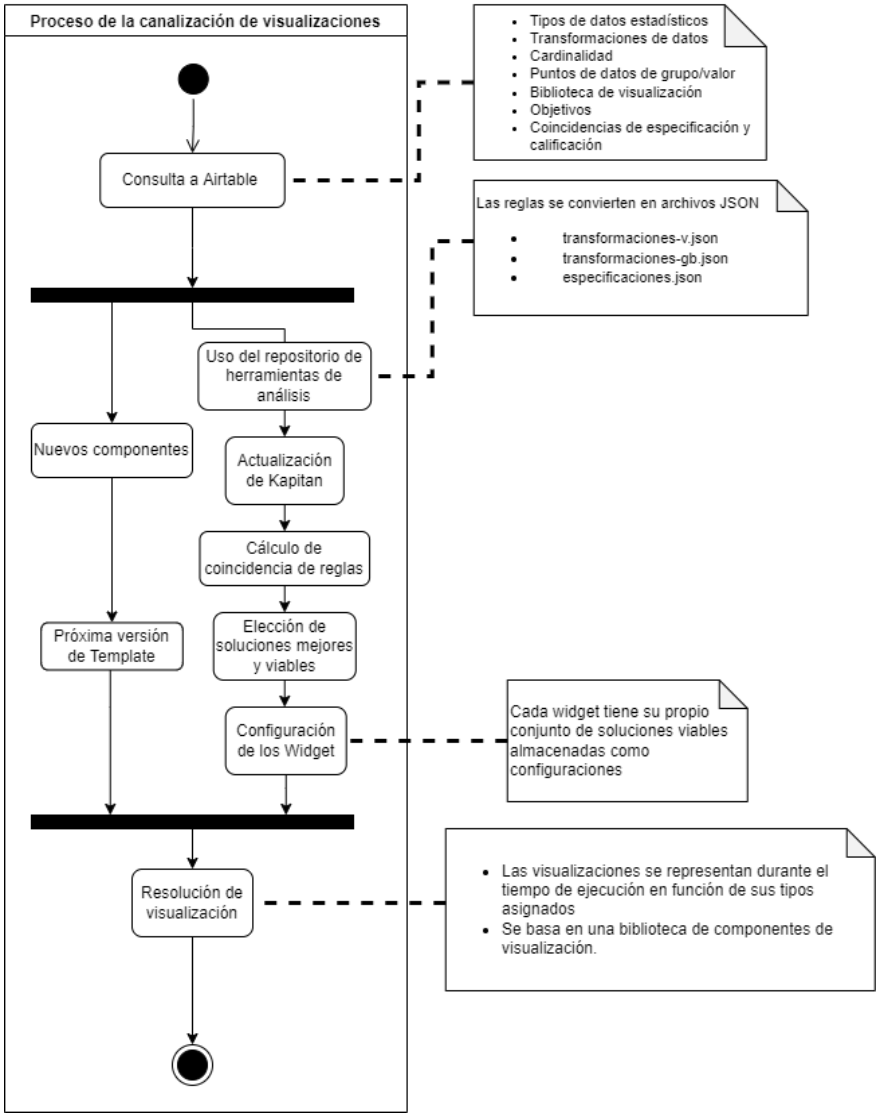
Figura 15:
Arquitectura de Servicios de Kleeen Software IDE.



Nota. Fuente: Kleeen Software Engineering

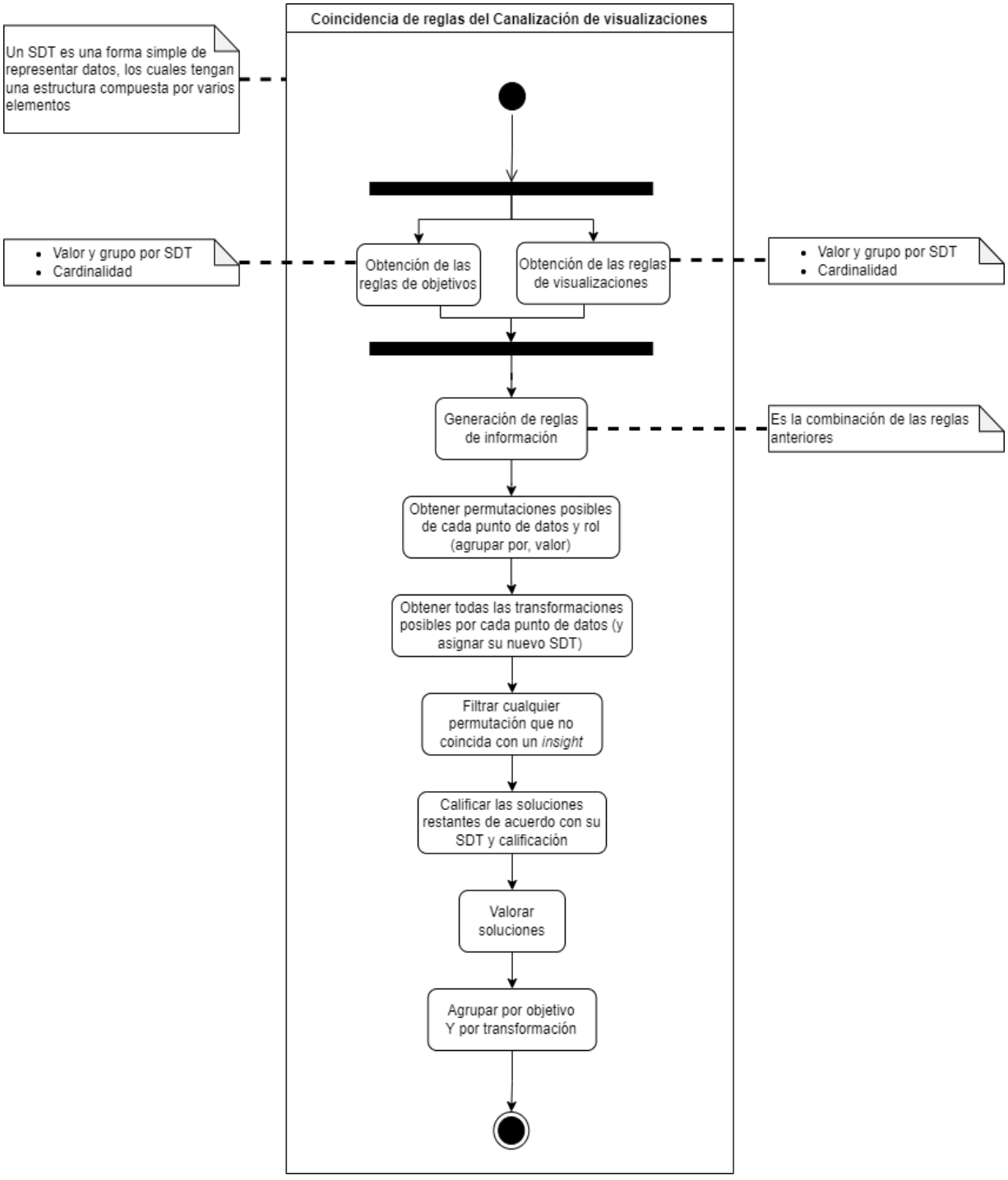
También, para comprender la arquitectura del sistema, así como su diseño, se debe primero comprender el flujo que poseen los datos entre el usuario final y el sistema, estos se muestran por medio de visualizaciones en el prototipo generado, las cuales se generan a través de una canalización que toma los datos proveídos por KAPI y, de acuerdo con la configuración realizada por el usuario, se calculan las mejores opciones para la representación correcta de los datos. En la **Figura 16** y **Figura 17**, se elaboró una adaptación del diagrama que muestra cómo es este proceso, con base en **Anexo V y Anexo IV**,

Figura 16:
Diagrama de Actividad: Adaptación del Viz Pipeline Process Overview.



Nota. Adaptación de Kleeen Software

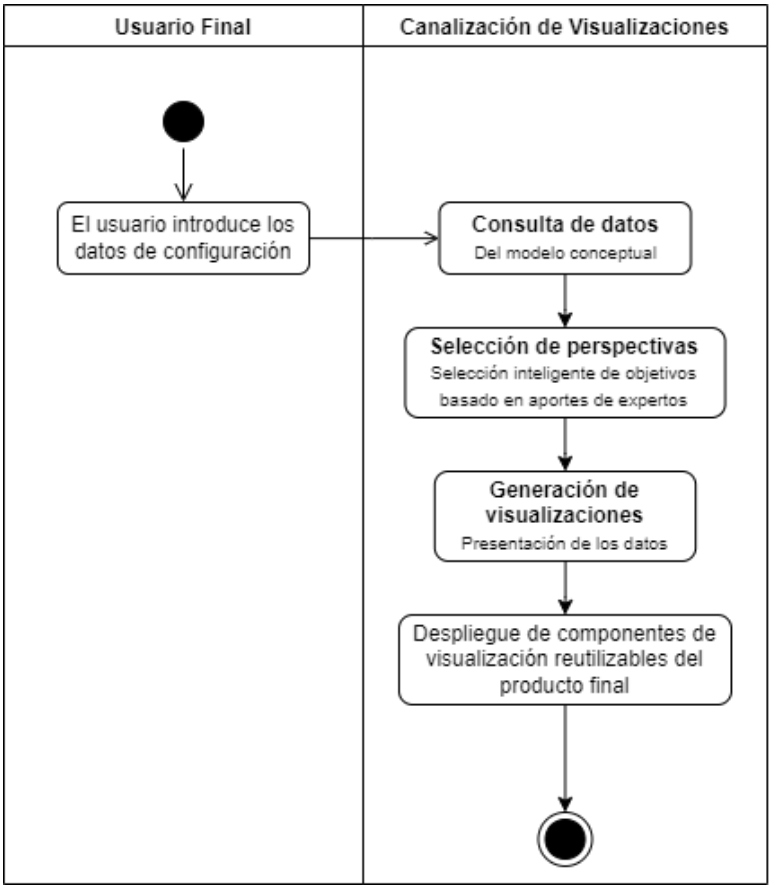
Figura 17:
Diagrama de Actividad: Adaptación del Viz Pipeline - Rule Matching.



Nota. Adaptación de Kleeen Software

A continuación, en la **Figura 18,** se muestra el flujo del funcionamiento de este proceso.

Figura 18:
Diagrama de Actividad: Canalización de Visualizaciones.



Nota. Fuente: Elaboración propia.

Dicho proceso es un punto clave para la generación de valor al cliente, y, a su vez, se vuelve un recurso fundamental para el sistema, esto porque posee una perfecta integración con el modelo conceptual, infiere en las perspectivas para los datos proveídos, sugiere el conjunto correcto de visualizaciones para un objetivo dado y evita errores comunes en la presentación de datos.

4.2. Análisis de la situación actual

En esta sección se expone el resultado de las observaciones obtenidas mediante la recopilación de información y se presenta un análisis cualitativo de las respuestas obtenidas de las entrevistas de levantamiento de requerimientos. En este apartado se contemplan las fases 3, 4 y 5 del punto **Procedimiento metodológico de la investigación**.

4.2.1. Debilidades del sistema Kleeen Software IDE

El sistema Kleeen Software IDE posee una arquitectura robusta según lo mostrado anteriormente. Este sistema, al integrarse con otros módulos como el Modelo Conceptual, KAPI, y la Canalización de Visualizaciones, permite el flujo de datos desde el *backend* hasta el *frontend*, de manera que el usuario final pueda ver y hacer uso del mejor tipo de gráficos según su configuración y los datos almacenados. Sin embargo, no existe manera o arquitectura que indique la forma en que estos datos puedan mostrarse o redactarse, o bien, descripciones contextualizadas de los gráficos generados de acuerdo con el perfil del usuario que los esté solicitando, esto genera que dichos usuarios, pese a que observan gráficos correctos, no siempre cuenten con el contexto necesario para comprenderlos.

A continuación, en la **Figura 19,** se presenta un análisis FODA del sistema Kleeen Software IDE producto de la reunión 0 (Ver **Minuta Reunión 0**) y tras realizar una entrevista al vicepresidente de ingeniería (ver **Apéndice H.a**) el sujeto entrevistado, expresa lo siguiente:

Figura 19:
Análisis FODA de Kleeen Software IDE.

Nota. Fuente: Elaboración propia.

De acuerdo con el concepto de Análisis FODA, y según el análisis FODA anterior, se evidencia cómo el hecho de no priorizar o investigar el pensamiento de usuario, o bien, no conocer el perfil del usuario final son debilidades que influyen en la problemática explicada en la **Introducción**, esto debido que no se conoce cómo o qué tipo de información espera este.

4.2.2. Necesidades del sistema Kleeen Software IDE

Como se describió en la **Situación problemática**, el sistema Kleeen Software tiene la necesidad de lograr recibir datos no procesados y convertirlos en información entendible por el ser humano.

Aunque actualmente la empresa cuenta con arquitectura de *software* establecida para Kleeen Software IDE, tal y como se evidenció en el apartado de **Debilidades del sistema Kleeen Software IDE**, es la falta de un diseño para *plugins* o para extensiones del sistema y debido a esto, es la falta de un diseño para un sistema de generación de lenguaje natural lo que provoca que el sistema actual no satisfaga la necesidad de poder brindar respuestas contextualizadas e inteligentes al usuario final, basándose en su contexto e intenciones.

A continuación, en la **Tabla 15** como parte de la información recopilada producto de la reunión 3 (Ver **Minuta Reunión 3**), y al entrevistar al vicepresidente de ingeniería (ver **Apéndice H.b**) se logran obtener los siguientes casos de uso, los cuales complementan los requerimientos iniciales que se definen más adelante en el **Análisis de los Requerimientos Recopilados** para satisfacer o aplacar las necesidades encontradas.

Tabla 16:
Casos de Uso.

ID	Nombre
CU0001	Gestión de Entidades
CU0002	Gestión de Reportes
CU0003	Gestión de Alertas y Notificaciones
CU0004	Gestión de Resúmenes de Tableros de Información
CU0005	Gestión de Integraciones

Nota. Fuente: Elaboración propia.

A continuación, en la **Tabla 16**, luego de llevar a cabo la reunión 3 (Ver **Minuta Reunión 3**),), y al entrevistar al vicepresidente de ingeniería (ver **Apéndice H.b**) se logra recopilar los siguientes requerimientos, los cuales están redactados con respecto a la guía de ISO/IEC/IEEE 29148, con el formato "[Asunto] [Acción] [Valor]":

Tabla 17:
Requerimientos del Sistema.

Requerimiento funcional	
ID	**Descripción**
REQ-1	El sistema debe permitir la integración con los sistemas de la empresa.
REQ-2	El sistema debe recibir datos de los sistemas de la empresa.
REQ-3	El sistema debe reconocer entidades.
REQ-4	El sistema debe permitir la gestión de entidades (crear, leer, actualizar, borrar).
REQ-5	El sistema debe permitir la creación de valores ejemplo para las entidades.
REQ-6	El sistema debe reconocer datos numéricos y agregaciones.
REQ-7	El sistema debe permitir la gestión de notificaciones de resumen (crear, leer, actualizar, borrar, envío de notificaciones).
REQ-8	El sistema debe permitir la gestión de reportes de informe con los indicadores de interés para el negocio (crear, leer, actualizar, borrar, divulgación de reportes, exportación de reportes).
REQ-9	El sistema debe generar resúmenes sobre *dashboards* (historia y propósito del *dashboard*) (crear, leer, actualizar, borrar, divulgación de resúmenes, exportación de resúmenes).
REQ-10	El sistema debe proporcionar recomendaciones de redacción de acuerdo con el interés del usuario (formalismos, populismos).
Restricción	
ID	**Descripción**
REQ-11	Existe una pantalla con la información de configuración del usuario.
REQ-12	El sistema tiene el logotipo de Kleeen Software.
REQ-13	La interfaz gráfica debe tener los colores de la paleta de colores de Kleeen Software.

Requerimientos de seguridad	
ID	**Descripción**
REQ-14	Al sistema sólo se podrá acceder por medio de Kleeen Software IDE o algún otro sistema de la empresa.
Rendimiento y mantenibilidad	
ID	**Descripción**
REQ-15	Vistas por módulos.
REQ-16	Validación por medio de *callbacks*.
REQ-17	Persistencia en la base de datos.
Interfaces de comunicaciones	
ID	**Descripción**
REQ-18	El sistema debe permitir la integración con servicios de mensajería (correos, SMS, Messenger, Whatsapp).
REQ-19	El sistema debe funcionar entre navegadores.

Nota. Fuente: Elaboración propia.

4.2.3. Análisis de los Requerimientos Recopilados

En la presente sección se presenta el análisis de los requerimientos recopilados partiendo de los conceptos planteados en la **Descripción Detallada y Priorización de los Requerimientos**.

4.2.3.1. Descripción detallada y priorización de los requerimientos funcionales

A continuación, en la **Tabla 17** se describen los requerimientos funcionales identificados anteriormente.

Tabla 18:
Descripción Detallada y Priorización de los Requerimientos Funcionales.

ID	Nombre	Urgencia	Valor del Negocio	Prioridad	MoSCoW
REQ-1	El sistema debe permitir la integración en los sistemas de la empresa.	5	5	25	M - Must
Descripción	Se debe garantizar la integración en el sistema Kleeen Software IDE en primera instancia (o algún otro sistema posterior).				
Precondiciones o supuestos	El sistema Kleeen Software IDE tendrá la apertura para la integración propuesta.				
REQ-2	El sistema debe recibir datos de los sistemas de la empresa.	5	5	25	M - Must
Descripción	Se debe permitir la recepción de datos numéricos, bases de datos, bases de conocimiento o corpus etiquetado.				
Precondiciones o supuestos	Los datos serán correctos a nivel de ortografía y uso de signos.				

ID	Nombre	Urgencia	Valor del Negocio	Prioridad	MoSCoW
REQ-3	El sistema debe reconocer entidades.	5	5	25	M - Must
Descripción	Se debe permitir el reconocimiento de entidades con el propósito de realizar el proceso de generación de lenguaje natural.				
Precondiciones o supuestos	El usuario ingresará ejemplos y categorías base para esta tarea.				
REQ-4	El sistema debe permitir la gestión de entidades (crear, leer, actualizar, borrar)	5	5	25	M - Must
Descripción	Se debe permitir que el usuario gestione las entidades de interés.				
Precondiciones o supuestos					
REQ-5	El sistema debe permitir la creación de valores ejemplo para las entidades.	5	5	25	M - Must
Descripción	Se debe permitir que el usuario ingrese valores ejemplos para las entidades de interés registradas.				
Precondiciones o supuestos	El usuario ingresará las entidades previamente.				

ID	Nombre	Urgencia	Valor del Negocio	Prioridad	MoSCoW
REQ-6	El sistema debe reconocer datos numéricos y agregaciones.	5	5	25	M - Must
Descripción	Se debe permitir el reconocimiento de datos recibidos en el sistema y agregaciones matemáticas como: sumas totales, porcentajes, máximos y mínimos, etc.				
Precondiciones o supuestos	Los datos serán correctos a nivel de tipado o escritura. Los números decimales serán separados por puntos y las agrupaciones de miles por comas.				
REQ-7	El sistema debe permitir la gestión de notificaciones de resumen (crear, leer, actualizar, borrar, envío de notificaciones).	3	4	12	S - Should
Descripción	Se debe permitir que el usuario gestione las notificaciones de resumen.				
Precondiciones o supuestos					

ID	Nombre	Urgencia	Valor del Negocio	Prioridad	MoSCoW
REQ-8	El sistema debe permitir la gestión de reportes de informe con los indicadores de interés para el negocio (crear, leer, actualizar, borrar, divulgación de reportes, exportación de reportes).	5	5	25	M - Must
Descripción	Se debe permitir que el usuario gestione los reportes de informe.				
Precondiciones o supuestos					
REQ-9	El sistema debe generar resúmenes sobre *dashboards* (historia y propósito del *dashboard*) (crear, leer, actualizar, borrar, divulgación de resúmenes, exportación de resúmenes).	5	5	25	M - Must
Descripción	Se debe permitir que el usuario gestione los resúmenes sobre *dashboards*.				
Precondiciones o supuestos					

ID	Nombre	Urgencia	Valor del Negocio	Prioridad	MoSCoW
REQ-10	El sistema debe proporcionar recomendaciones de redacción de acuerdo con el interés del usuario (formalismos, populismos).	2	3	6	C - Could
Descripción	Se permitirá la recomendación de redacción de textos cuando sea pertinente, ya sea por aspectos de formalismos o necesidades del cliente.				
Precondiciones o supuestos					

Nota. Fuente: Elaboración propia.

Como resultado del análisis de requerimientos presentado, se obtiene que:

- Existen un total de diecinueve requerimientos por parte del cliente, de los cuales diez son requerimientos funcionales.
- Se identifican ocho requerimientos de prioridad alta (must), uno de prioridad media (should), y uno de prioridad baja (could).

Como resumen de esto, se realiza la **Tabla 18:**

Tabla 19:
Ubicación de la Prioridad de los Requerimientos en el Mapa de Calor

ID	Prioridad	Ubicación en el mapa de calor (ver Tabla 3)
REQ-1	M - Must	
REQ-2	M - Must	
REQ-3	M - Must	
REQ-4	M - Must	
REQ-5	M - Must	
REQ-6	M - Must	
REQ-8	M - Must	
REQ-9	M - Must	
REQ-7	S - Should	
REQ-10	C - Could	

Nota. Fuente: Elaboración propia.

4.2.3.2. Descripción de los casos de uso

De acuerdo con el concepto de **Descripción de los Casos de Uso**, a continuación, en **Tabla 19** se presenta una descripción de alto nivel de los casos de uso identificados.

Tabla 20:
Descripción de los Casos de Uso.

Identificador del caso de uso	CU0001	**Nombre del Caso de Uso**	Gestión de Entidades	
Actor principal	Usuario			
Precondiciones	El sistema se encuentra integrado y funcional.			
Postcondiciones	El sistema creó la entidad. El sistema leyó la entidad. El sistema actualizó la entidad. El sistema eliminó la entidad. El sistema catalogó la entidad. El sistema creó valores ejemplo para la entidad.			
Escenario	1. El usuario selecciona el apartado de gestión de entidades. 2. El usuario selecciona la opción requerida. 3. El sistema permite el desarrollo de la acción partiendo de los conceptos básicos de CRUD *(Create, Read, Update, Delete)*, o bien, la acción de asignar categorías o ingresar valores ejemplo.			
Escenarios de CRUD	3.1.El usuario selecciona crear. 3.1.2.Se despliega el formulario correspondiente	3.2. El usuario selecciona leer. 3.2.1. Se despliega el formulario para	3.3.El usuario selecciona editar. 3.3.2.Se despliega el formulario correspondiente	3.4. El usuario selecciona eliminar. 3.4.1. Se despliega el formulario para

	para la creación de la entidad. 3.1.3. Se validan los datos. 3.1.4. Se guarda la entidad.	seleccionar la entidad por visualizar. 3.2.2. Se visualiza la entidad.	para la edición de la entidad. 3.3.3. Se validan los datos. 3.3.4. Se guarda la entidad.	seleccionar la entidad por eliminar. 3.4.2. Se elimina la entidad.
Excepciones	El sistema verificará que los datos ingresados sean correctos.			
Identificador del caso de uso	CU0002	**Nombre del Caso de Uso**	Gestión de Reportes	
Actor principal	Usuario			
Precondiciones	El sistema se encuentra integrado y funcional.			
Postcondiciones	El sistema creó el reporte. El sistema leyó el reporte. El sistema actualizó el reporte. El sistema elimina el reporte. El sistema divulgó el reporte. El sistema exportó el reporte.			
Escenario	1. El usuario selecciona el apartado de gestión de reportes. 2. El usuario selecciona la opción requerida. 3. El sistema permite el desarrollo de la acción partiendo de los conceptos básicos de CRUD *(Create, Read, Update, Delete),* o bien, la acción de divulgar reportes, o exportar reportes.			
Escenarios de CRUD	3.1. El usuario selecciona el *widget* del cuál desea el reporte. 3.1.1. Se crea el reporte.	3.2. El usuario selecciona el *widget* del cuál desea leer el reporte.	3.3. El usuario selecciona el *widget* del cuál desea editar el reporte.	3.4. El usuario selecciona el *widget* del cuál desea eliminar el reporte.

		3.2.1. Si el reporte existe, se muestra, si no: 3.2.2. Se ejecuta el escenario 3.1 y luego el 3.2.	3.3.1. Se despliega el formulario para editar el reporte. 3.3.2. Se validan los datos 3.3.3. Se guarda el reporte.	3.4.1. El usuario selecciona eliminar el reporte. 3.4.2. Se elimina el reporte.
Excepciones	El sistema verificará que los datos ingresados sean correctos.			
Identificador del caso de uso	CU0003	**Nombre del Caso de Uso**	Gestión de Alertas y Notificaciones	
Actor principal	Usuario			
Precondiciones	El sistema se encuentra integrado y funcional.			
Postcondiciones	El sistema creó la alerta. El sistema leyó la alerta. El sistema actualizó la alerta. El sistema eliminó la alerta. El sistema envió la alerta.			
Escenario	1. El usuario selecciona el apartado de gestión de alertas y notificaciones. 2. El usuario selecciona la opción requerida. 3. El sistema permite el desarrollo de la acción partiendo de los conceptos básicos de CRUD *(Create, Read, Update, Delete),* o bien, la acción de envío de alertas.			
Escenarios de CRUD	3.1. El usuario edita algún punto en el sistema. 3.1.1. El sistema detecta un cambio.	3.2. El sistema envía la alerta. 3.2.1. El usuario visualiza la alerta.	3.3. Se ejecuta el escenario 3.1 y el 3.2. 3.3.1. El usuario selecciona editar la alerta.	3.4. Se ejecuta el escenario 3.1 y el 3.2.

	3.1.2. Se crea la alerta.			3.4.1. El usuario selecciona eliminar la alerta.
Excepciones	El sistema verificará que los datos ingresados sean correctos.			
Identificador del caso de uso	CU0004	**Nombre del Caso de Uso**	Gestión de Resúmenes de Tableros de Información	
Actor principal	Usuario			
Precondiciones	El sistema se encuentra integrado y funcional.			
Postcondiciones	El sistema creó el resumen. El sistema leyó el resumen. El sistema actualizó el resumen. El sistema elimina el resumen. El sistema divulgó el resumen. El sistema exportó el resumen.			
Escenario	1. El usuario selecciona el apartado de gestión de resúmenes de tableros de información. 2. El usuario selecciona la opción requerida. 3. El sistema permite el desarrollo de la acción partiendo de los conceptos básicos de CRUD *(Create, Read, Update, Delete),* o bien, la acción de divulgar resúmenes o exportar resúmenes.			
Escenarios de CRUD	3.1. El usuario selecciona el tablero de información del cuál desea el resumen. 3.1.1. Se crea el resumen.	3.2. El usuario selecciona el tablero de información del cuál desea el resumen.	3.3. El usuario selecciona el tablero de información del cuál desea el resumen.	3.4. El usuario selecciona el tablero de información del cuál desea el resumen.

		3.2.1. Si el resumen existe, se muestra, si no: 3.2.2. Se ejecuta el escenario 3.1 y luego el 3.2.	3.3.1. Se despliega el formulario para editar el resumen. 3.3.2. Se validan los datos. 3.3.3. Se guarda el resumen.	3.4.1. El usuario selecciona eliminar el resumen. 3.4.2. Se elimina el resumen.
Excepciones	El sistema verificará que los datos ingresados sean correctos.			
Identificador del caso de uso	CU0005	**Nombre del Caso de Uso**	Gestión de Integraciones	
Actor principal	Usuario			
Precondiciones	El sistema se encuentra integrado y funcional.			
Postcondiciones	El sistema creó la integración. El sistema leyó la integración. El sistema actualizó la integración. El sistema eliminó la integración.			
Escenario	1. El usuario selecciona el apartado de gestión de integraciones. 2. El usuario selecciona la opción requerida. 3. El sistema permite el desarrollo de la acción partiendo de los conceptos básicos de CRUD *(Create, Read, Update, Delete).*			

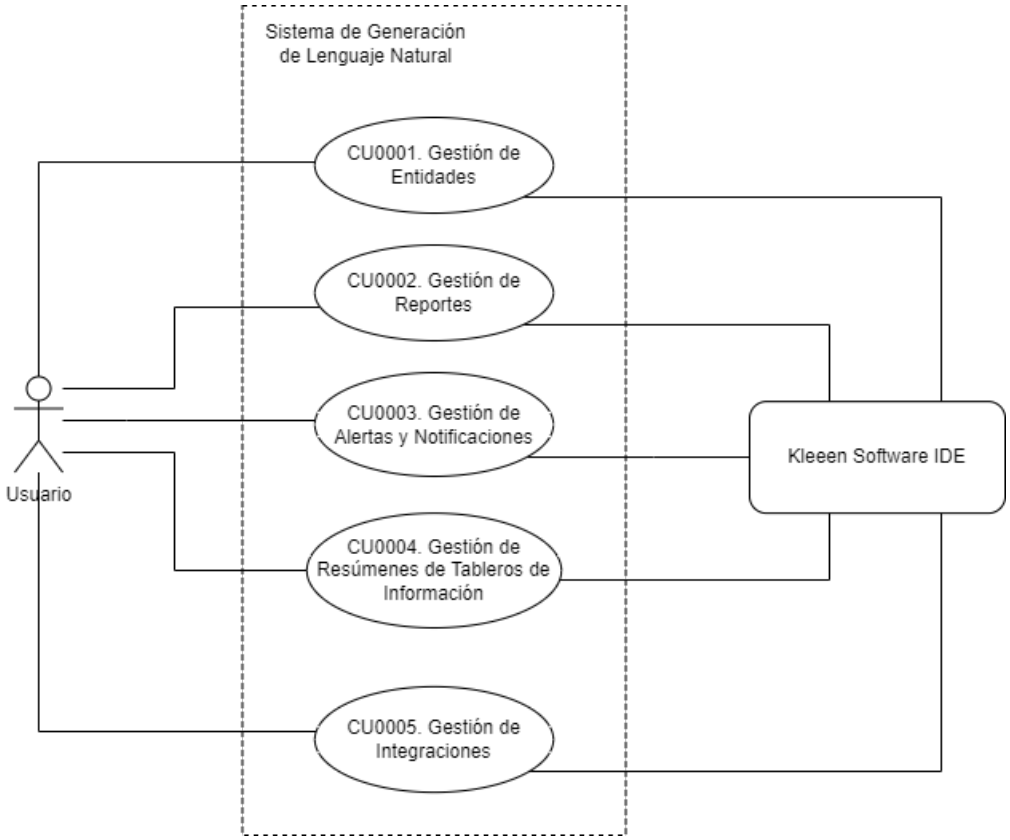
Escenarios de CRUD	3.1.El usuario selecciona crear. 3.1.2.Se despliega el formulario correspondiente para la creación de la integración. 3.1.3. Se validan los datos. 3.1.4. Se guarda la integración.	3.2. El usuario selecciona leer. 3.2.1. Se despliega el formulario para seleccionar la integración por visualizar. 3.2.2. Se visualiza la integración.	3.3.El usuario selecciona editar. 3.3.2.Se despliega el formulario correspondiente para la edición de la integración. 3.3.3. Se validan los datos. 3.3.4. Se guarda la integración.	3.4. El usuario selecciona eliminar. 3.4.1. Se despliega el formulario para seleccionar la integración por eliminar. 3.4.2. Se elimina la integración
Excepciones	El sistema verificará que los datos ingresados sean correctos.			

Nota. Fuente: Elaboración propia.

4.2.3.3. Modelo general de casos de uso

De acuerdo con el apartado "**Modelo General de Casos de Uso**", a continuación, en **Figura 20**, se presenta un diagrama de alto nivel de los casos de uso.

Figura 20:
Diagrama de Caso de Uso: Modelo General de Casos de Uso.



Nota. Fuente: Elaboración propia.

4.3. Experimento de Mago de Oz

El modelado de interacción nace de la necesidad de brindar un mejor soporte para la interacción entre humanos y computadoras. La interacción en este contexto, según Terveen (1995), se define como “un proceso en el que dos o más agentes trabajan juntos para lograr objetivos compartidos". De acuerdo con Fischer (2001), algunas cuestiones fundamentales (como objetivos compartidos, contexto compartido, control, co-adaptación, coevolución y aprendizaje) pueden derivarse de esta definición.

De acuerdo con Zapata y Carmona (2006), se entiende que el uso de interfaces de Lenguaje Natural en la interacción del usuario con el sistema incluye aspectos que permiten la evaluación y diseño de estas interfaces, para esto, se suele usar la técnica de “Experimento de Mago de Oz”. El Experimento de Mago de Oz es una técnica usada para evaluar la interacción entre un usuario y el sistema, este proceso consta de tres componentes esenciales, los cuales son:

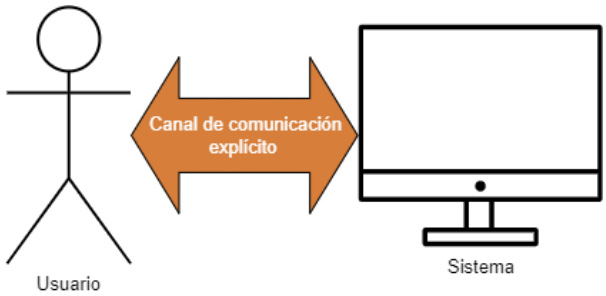
- Usuario: persona que prueba el sistema deseado.
- Sistema: simulación del sistema real por probar.
- Mago: persona o *software* que simula la actividad del sistema por probar.

En la actualidad, esta técnica ha mutado y cambiado, sin embargo, su objetivo inicial se mantiene. Según Fraser y Gilbert (1991) para emplear este experimento, se debe cumplir lo siguiente:

- Es posible simular el sistema futuro dadas las limitaciones humanas del Mago (velocidad de respuesta, o incapacidad de procesar ante una solicitud de un usuario).
- Es posible especificar el funcionamiento futuro del sistema.
- Es posible que la simulación sea convincente.

Según Fisher (2001), la interacción usuario y sistema se solía modelar como una simple díada en la que estos actores estaban conectados por un canal de comunicación explícito. Como ejemplo, se realizó una adaptación de este concepto en la **Figura 21**:

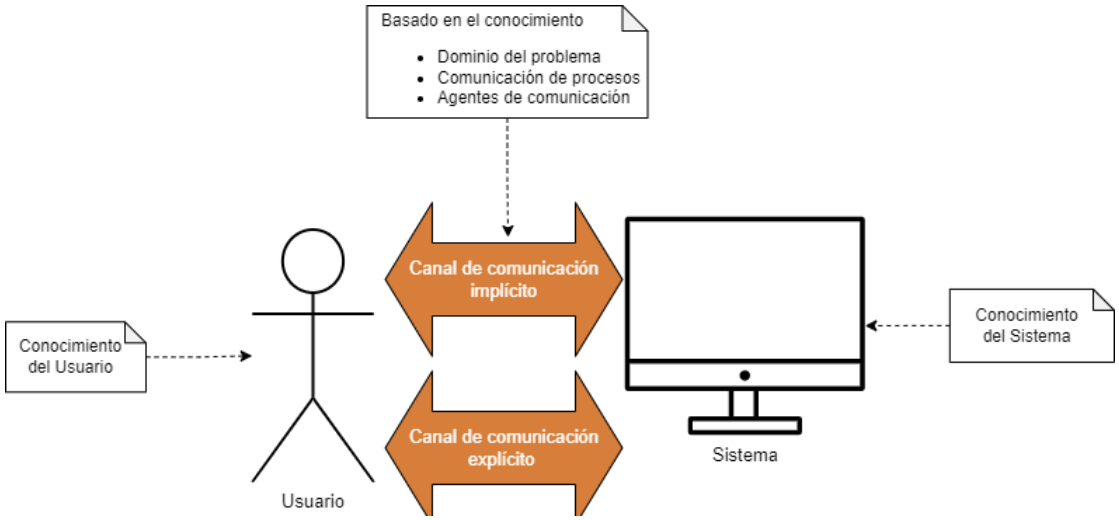
Figura 21:
Díada entre Usuario y Sistema.



Nota. Adaptación de User modeling in human–computer interaction. User modeling and user-adapted interaction Fischer, G. (2001).

El actual uso de elementos de comunicación como ventanas, menús, dispositivos de señalización, color, sonido y pantallas táctiles ha provocado que este canal de comunicación explícito se amplíe. Esta transformación vuelve al canal de comunicación implícito y apoya los procesos de comunicación que requieren que la computadora cuente con un cuerpo considerable de conocimiento sobre los dominios del problema, los procesos de comunicación y los agentes involucrados, por esto, se realizó una adaptación de este concepto en la **Figura 22:**

Figura 22:
Nueva Interacción Basada en el Conocimiento.



Nota. Adaptación de User modeling in human–computer interaction. User modeling and user-adapted interaction Fischer, G. (2001).

Debido a esto, para el presente proyecto se realizó un prototipo de interacción basándose en los principios del Experimento de Mago de Oz, en donde se presenta una simulación del sistema que emulará cómo funciona el resultado esperado por el usuario, de manera que este tenga una perspectiva clara y real de los datos que observa y satisfaga la necesidad planteada en el apartado "**Planteamiento del problema**". De acuerdo con Giannetti (2002), las interfaces técnicas (prototipos interactivos) desempeñan un papel semejante al de los medios que los seres humanos necesitamos para comunicarnos entre nosotros y facultan el acoplamiento entre diferentes sistemas.

El diseño de un prototipo, de acuerdo con Corina (2012) posee las etapas de:

1. Investigación preliminar: en esta fase se determina el problema y se identifica una idea para la solución.
2. Definición de los requerimientos del sistema: se registran todos los requerimientos y necesidades que los usuarios tienen en relación con el proyecto.
3. Diseño técnico: en esta fase el prototipo debe ser diseñado y documentado. Esta documentación incluye el diseño que especifica y describe la estructura del prototipo, el control de flujo, las interfaces de usuario y las funciones.

4. Programación y prueba: se implementa el prototipo y se prueba para asegurar la completitud de este con respecto a los requerimientos.
5. Operación y mantención: se pone en producción el prototipo y se le brinda el mantenimiento requerido.

De acuerdo con lo anterior, en la **Tabla 20** se adaptaron las etapas anteriores a las etapas del presente proyecto de la siguiente manera:

Tabla 21:
Adaptación de las Etapas de Desarrollo de Prototipos

Etapa según Corina (2012)	Etapa del presente proyecto
Investigación preliminar	**Debilidades del sistema Kleeen Software IDE.**
Definición de los requerimientos del sistema	**Necesidades del sistema Kleeen Software IDE**.
Diseño técnico	Se plantea el diseño de flujo, de interfaz de usuario y de función, por cada caso de uso elegido.
Programación y prueba	Se implementa un prototipo por medio de la herramienta Figma.
Operación y mantención	Debido a la naturaleza y contexto del prototipo, no se desarrolla esta etapa.

Nota. Fuente: Elaboración propia.

Conforme a las etapas planteadas, se presenta el desarrollo por cada una de estas:

4.3.1. Investigación preliminar

Como se definió en el **planteamiento del problema**, se conoce la actual problemática sobre la falta de un sistema que permita que los datos puedan mostrarse o redactarse de manera comprensible para el usuario que los está observando, o bien, la falta de descripciones contextualizadas de los gráficos generados de acuerdo con perfil del usuario que los esté solicitando.

4.3.2. Definición de los requerimientos del sistema

Anteriormente se especificaron los requerimientos del usuario, así como la identificación de los casos de uso principales. Debido a esto, en la **Tabla 21** se obtienen cinco casos de uso generales, de los cuales se eligen los siguientes para presentar en el prototipo.

Tabla 22:
Casos de uso por prototipar.

ID	Nombre	Por prototipar	Escenario por prototipar
CU0001	Gestión de Entidades		
CU0002	Gestión de Reportes	X	Creación y lectura de reportes
CU0003	Gestión de Alertas y Notificaciones	X	Creación y lectura de alertas
CU0004	Gestión de Resúmenes de Tableros de Información	X	Creación y lectura de resúmenes
CU0005	Gestión de Integraciones		

Nota. Fuente: Elaboración propia.

La elección de los casos de uso anteriores se basó en las necesidades del cliente, debido a que son los requerimientos más importantes que se pueden representar visualmente al integrar el sistema.

4.3.3. Diseño Técnico

A continuación, se presenta el diseño técnico del prototipo por cada caso de uso y escenario seleccionado.

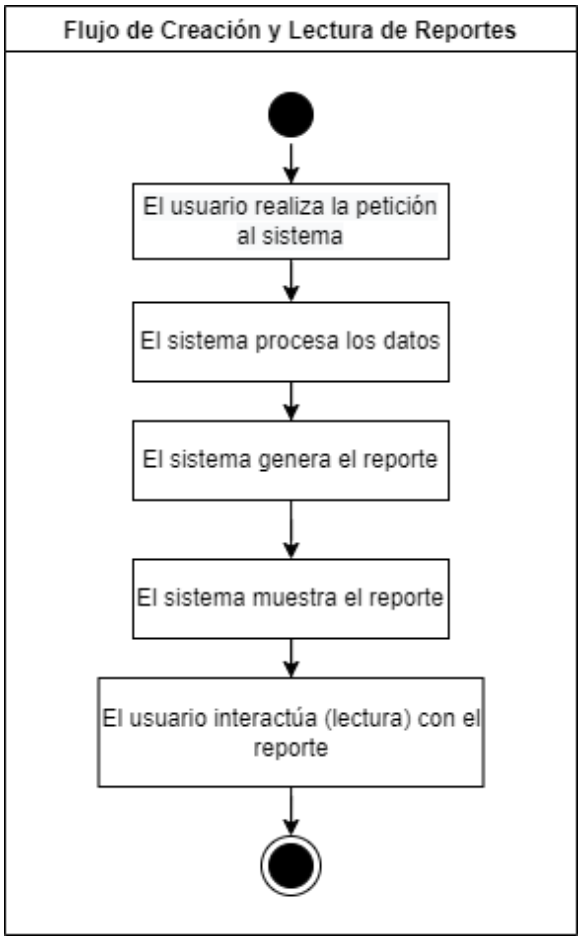
4.3.3.1. Gestión de Reportes

De acuerdo con los requerimientos recopilados, la gestión de reportes consta de una sección donde el usuario selecciona el indicador de interés o componente y se genera un reporte de este, permitiéndole al usuario la lectura correcta del mismo.

4.3.3.1.1. Diseño de flujo

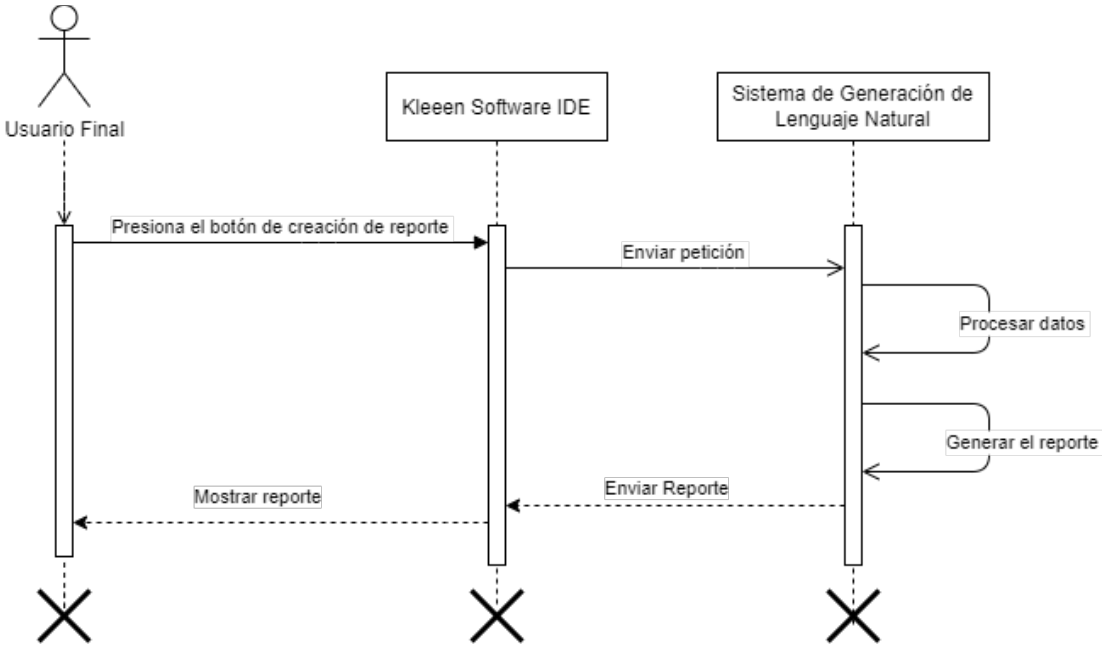
Para representar este flujo se realiza la **Figura 23** y la **Figura 24**.

Figura 23:
Diagrama de Actividad: Creación y Lectura de Reportes.



Nota. Fuente: Elaboración propia.

Figura 24:
Diagrama de Secuencia: Creación y Lectura de Reportes.



Nota. Fuente: Elaboración propia.

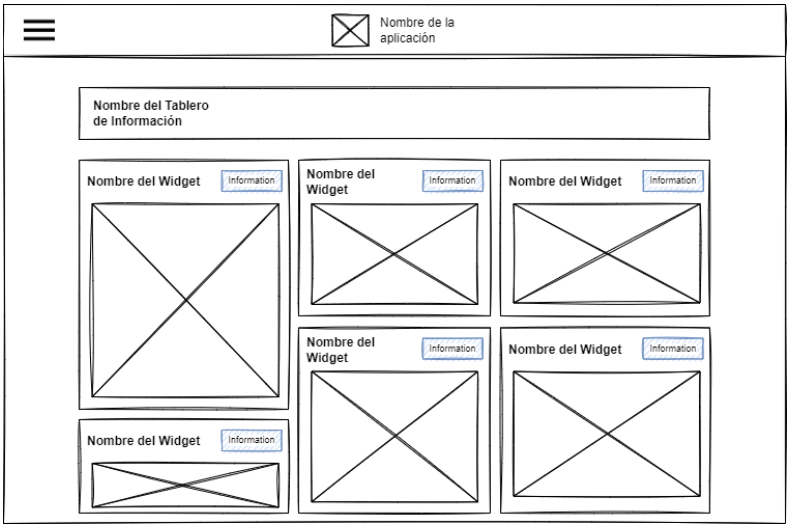
4.3.3.1.2. Diseño de interfaz de usuario

La interfaz será simple, adaptada al sistema Kleeen Software IDE, de manera que el usuario posea una interfaz similar a la cual está familiarizado. De acuerdo con Krause, R. (2021), la consistencia interna se relaciona con la consistencia dentro de un producto o una familia de productos, ya sea dentro de una sola aplicación o en una familia o conjunto de aplicaciones. Una familia de productos se refiere a varios productos, vendidos individualmente o en grupo, producidos por la misma empresa bajo la misma marca.

Por esto, al no alterar a gran escala la forma en que el usuario utiliza el actual sistema, se mantiene su usabilidad y facilidad de comprensión, sin complicar el proceso de aprendizaje de las funcionalidades proporcionadas por el nuevo sistema.

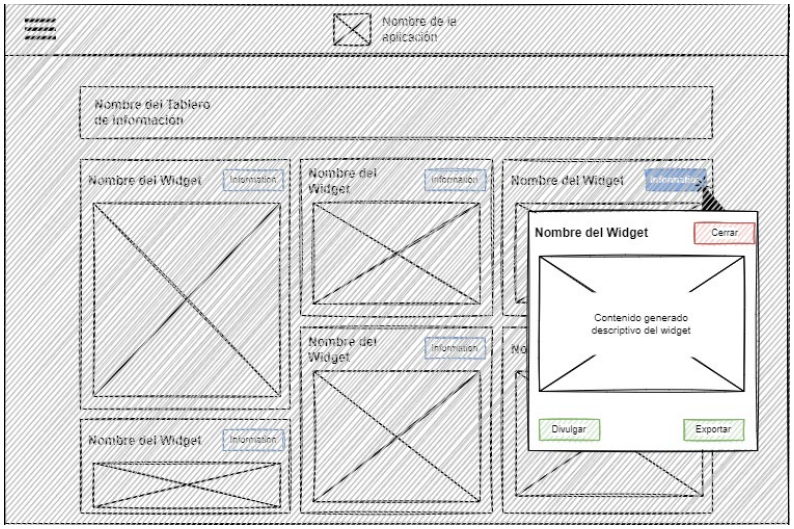
En la siguiente **Figura 25**. y la **Figura 26** se muestra un boceto del sistema propuesto.

Figura 25:
Boceto de Interfaz de Usuario: Creación y Lectura de Reportes (Vista del botón de creación).



Nota. Fuente: Elaboración propia

Figura 26:
Boceto de Interfaz de Usuario: Creación y Lectura de Reportes (Vista del reporte).



Nota. Fuente: Elaboración propia

4.3.3.1.3. Diseño de funciones

De acuerdo con el caso de uso y los escenarios seleccionados, se cuenta con dos funciones principales: creación y visualización de reportes.

La creación de reportes se realiza por medio de un botón ubicado en la parte superior de cada *widget*, el cual, al ser presionado, envía una petición al sistema, de manera que el sistema contextualiza quién es el componente que realizó el llamado y, de acuerdo con los datos de este componente, se genera un reporte, el cual será visualizado por el usuario.

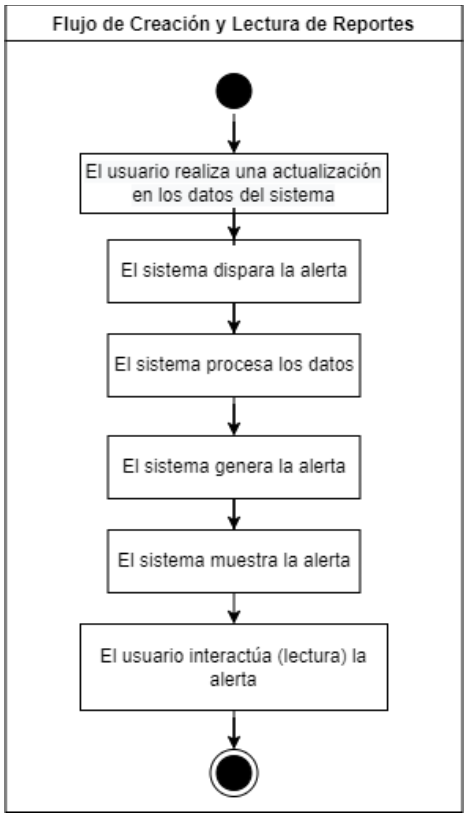
4.3.3.2. Gestión de Alertas y Notificaciones

Este caso de uso se encarga de la creación de alertas de acuerdo con disparadores, donde se genera una alerta entendible para el usuario, permitiéndole una comprensión correcta de las acciones ejecutadas.

4.3.3.2.1. Diseño de flujo.

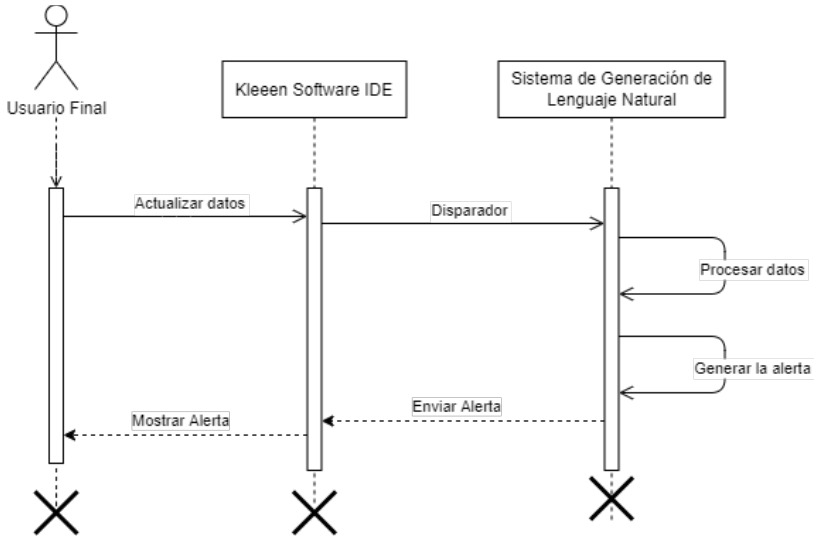
Para representar este flujo se realiza la **Figura 27** y la **Figura 28**:

Figura 27:
Diagrama de Actividad: Creación y Lectura de Alertas.



Nota. Fuente: Elaboración propia.

Figura 28:
Diagrama de Secuencia: Creación y Lectura de Alertas.



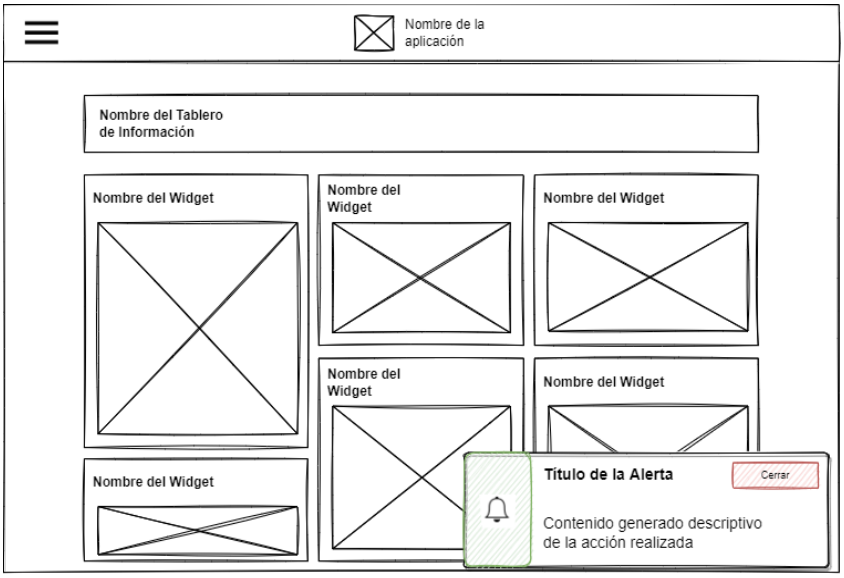
Nota. Fuente: Elaboración propia.

4.3.3.2.2. Diseño de interfaz de usuario

La interfaz será simple, adaptada al actual sistema Kleeen Software IDE. Las alertas de este caso serán diseñadas de manera que den la impresión de que el sistema está conversando con el usuario.

En la Figura 29 se muestra un boceto de las alertas.

Figura 29:
Boceto de Interfaz de Usuario: Alertas.



Nota. Fuente: Elaboración propia

4.3.3.2.3. Diseño de funciones

De acuerdo con los caso de uso y los escenarios seleccionados, se definen dos funciones principales: creación y visualización de alertas; sin embargo, la creación en este caso no es controlada por el usuario, debido a que la alerta se crea de acuerdo con disparadores que el sistema envía. El usuario sólo define cuándo se crean las alertas y el sistema se encarga de crearlas cuando la condición se cumpla.

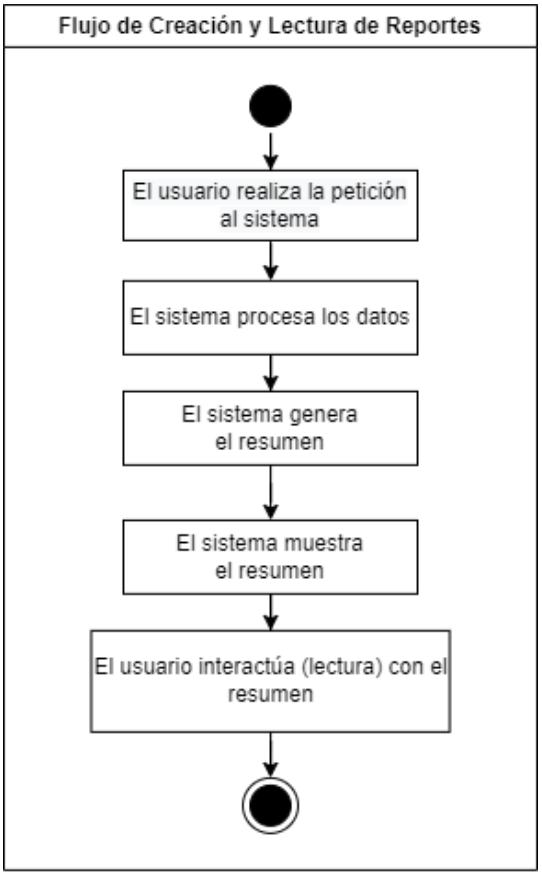
4.3.3.3. Gestión de Resúmenes de Tableros de Información

Este caso de uso permite que el usuario decida crear un resumen de un tablero en específico, donde se concentra y resume la información de este.

4.3.3.3.1. Diseño de flujo

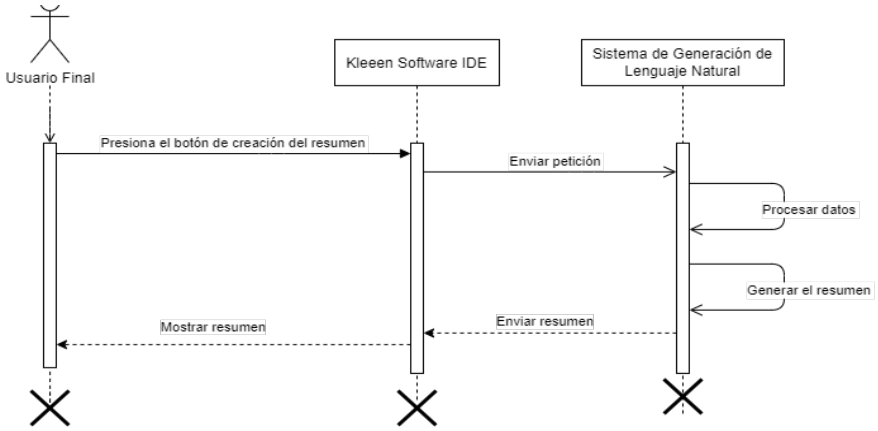
Para representar este flujo se realiza la **Figura 30** y la **Figura 31**:

Figura 30:
Diagrama de Actividad: Creación y Lectura de Resúmenes.



Nota. Fuente: Elaboración propia.

Figura 31:
Diagrama de Secuencia: Creación y Lectura de Resúmenes.



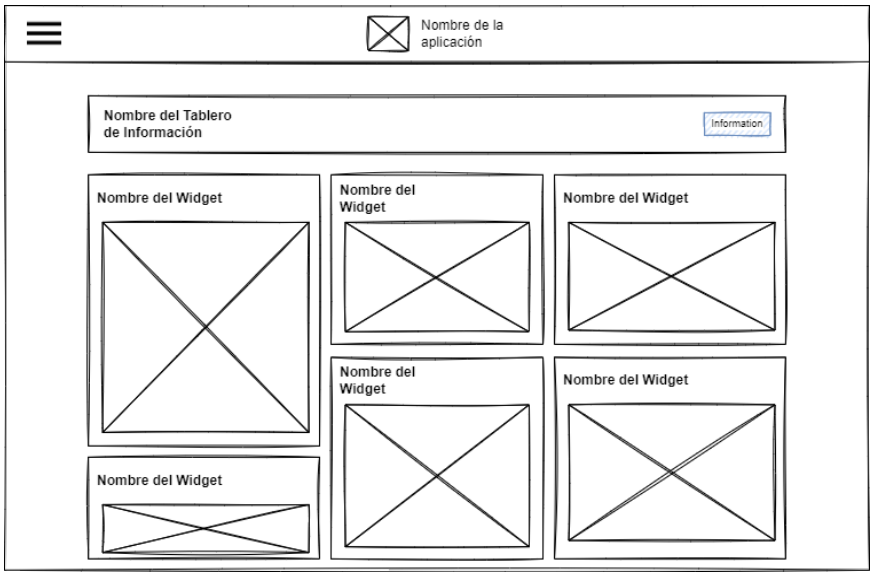
Nota. Fuente: Elaboración propia.

4.3.3.3.2. Diseño de interfaz de usuario

Los resúmenes se generarán por medio de un botón principal, siendo este parte del sistema Kleeen Software IDE.

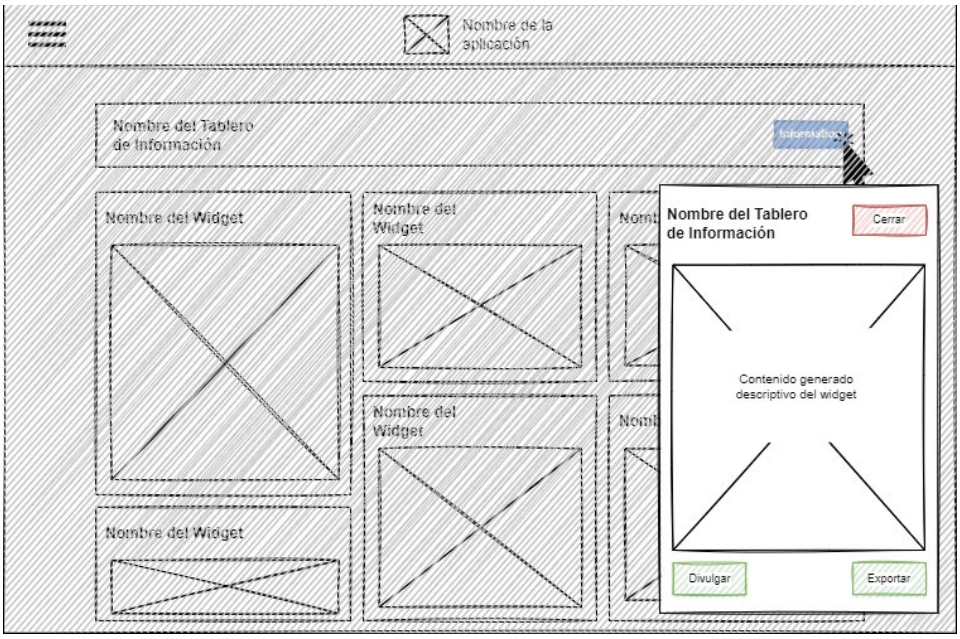
En la **Figura 32** y **Figura 33,** se muestra un boceto de las alertas.

Figura 32:
Boceto de Interfaz de Usuario: Creación y Lectura de Resúmenes (Vista del Botón de Creación)



Nota. Fuente: Elaboración propia

Figura 33:
Boceto de Interfaz de Usuario: Creación y Lectura de Resúmenes (Vista del Resumen).



Nota. Fuente: Elaboración propia

4.3.3.3.3. Diseño de funciones

Este caso de uso y los escenarios seleccionados poseen dos funciones: creación y visualización de resúmenes.

La creación de resúmenes se ejecuta por medio de un botón ubicado en el submenú de cada tablero de información. Este botón, al ser presionado, realiza la petición al sistema para que este contextualice y procese los datos y componentes relacionados con el tablero, y así generar un resumen, el cual será visualizado por el usuario.

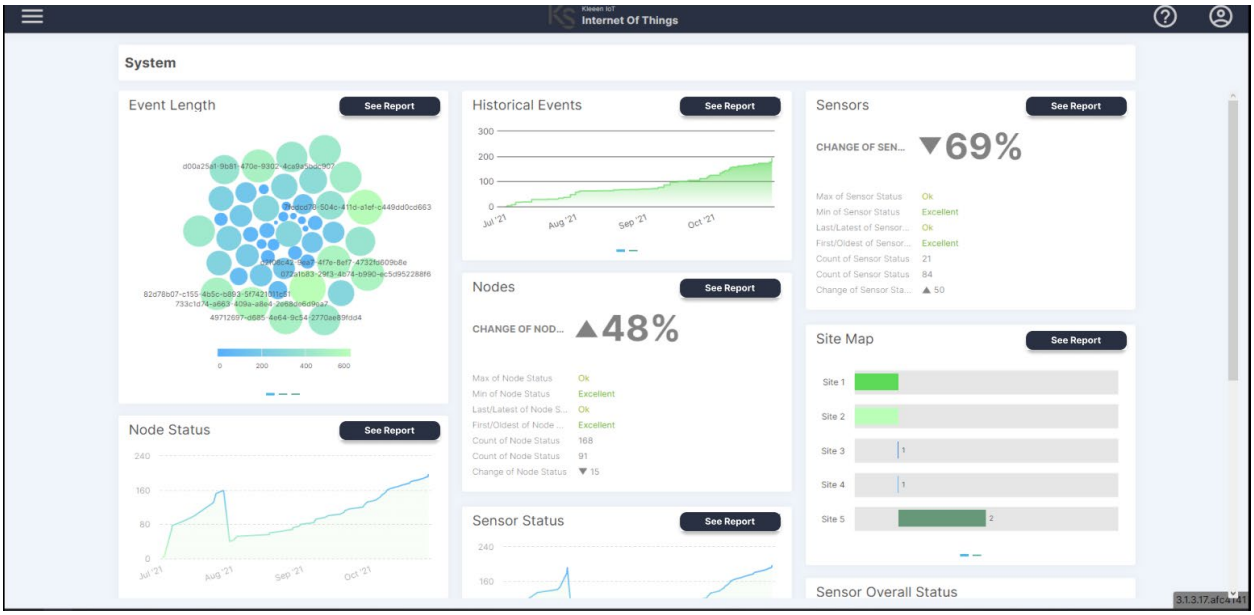
4.3.4. Programación y prueba

En esta sección se toma el diseño de interfaz realizado previamente y se implementa en la herramienta *Figma*.

4.3.4.1. Gestión de Reportes

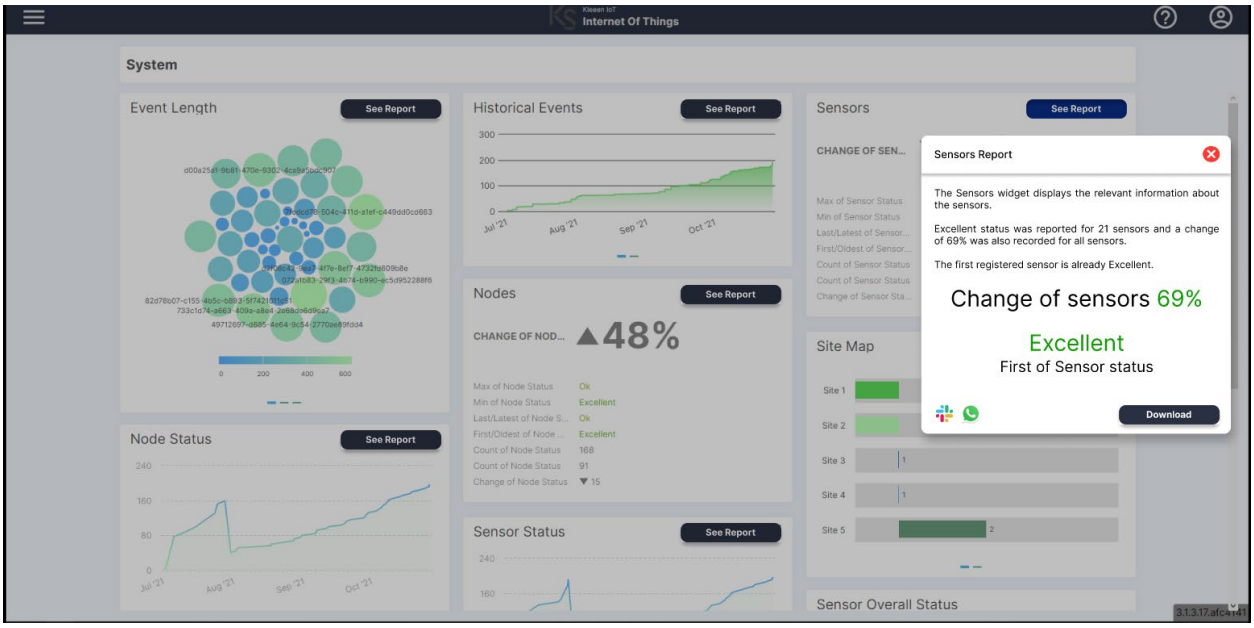
A continuación, en la **Figura 34 y** la **Figura 35**. se presenta la implementación del boceto realizado previamente.

Figura 34:
Prototipo de Interfaz de Usuario: Creación y Lectura de Reportes (Vista del botón de creación).



Nota. Fuente: Elaboración propia

Figura 35:
Prototipo de Interfaz de Usuario: Creación y Lectura de Reportes (Vista del botón de creación).

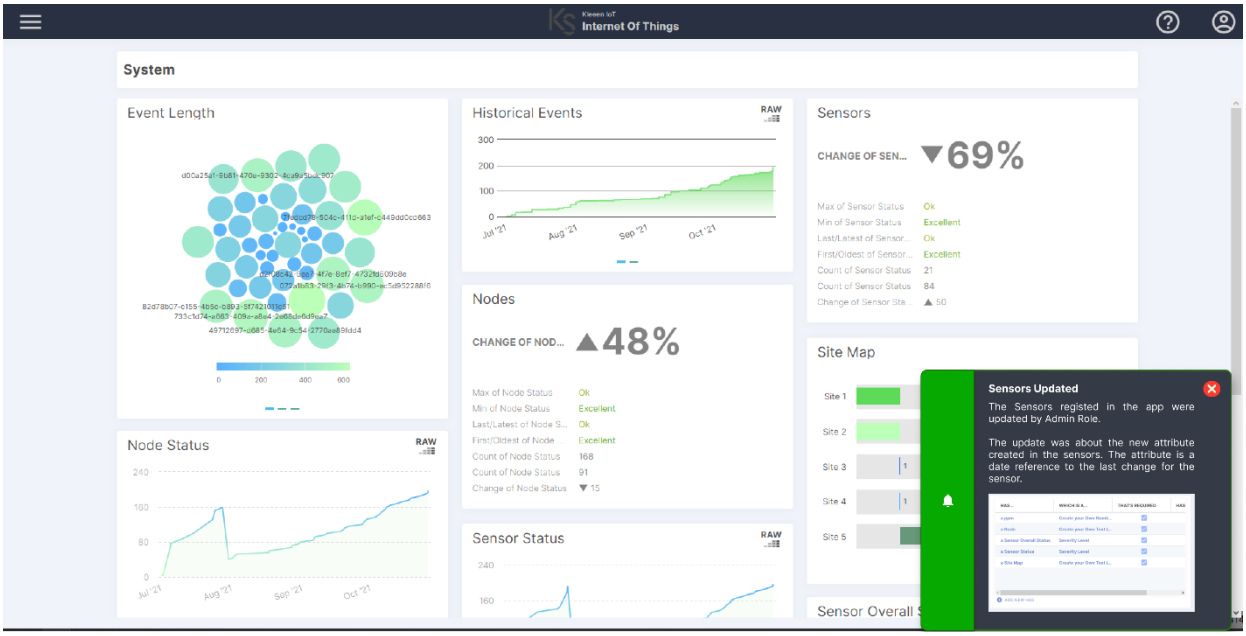


Nota. Fuente: Elaboración propia

4.3.4.2. Gestión de Alertas y Notificaciones

A continuación, en la **Figura 36** se presenta la implementación del boceto realizado previamente.

Figura 36:
Prototipo de Interfaz de Usuario: Alertas.

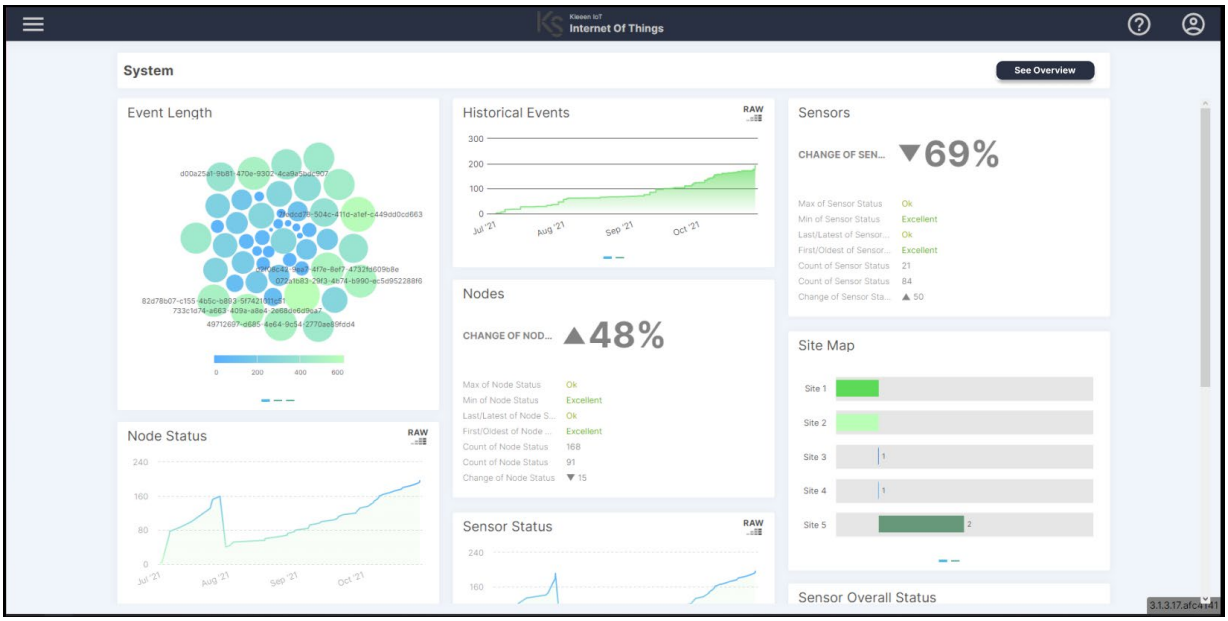


Nota. Fuente: Elaboración propia

4.3.4.3. Gestión de Resúmenes de Tableros de Información

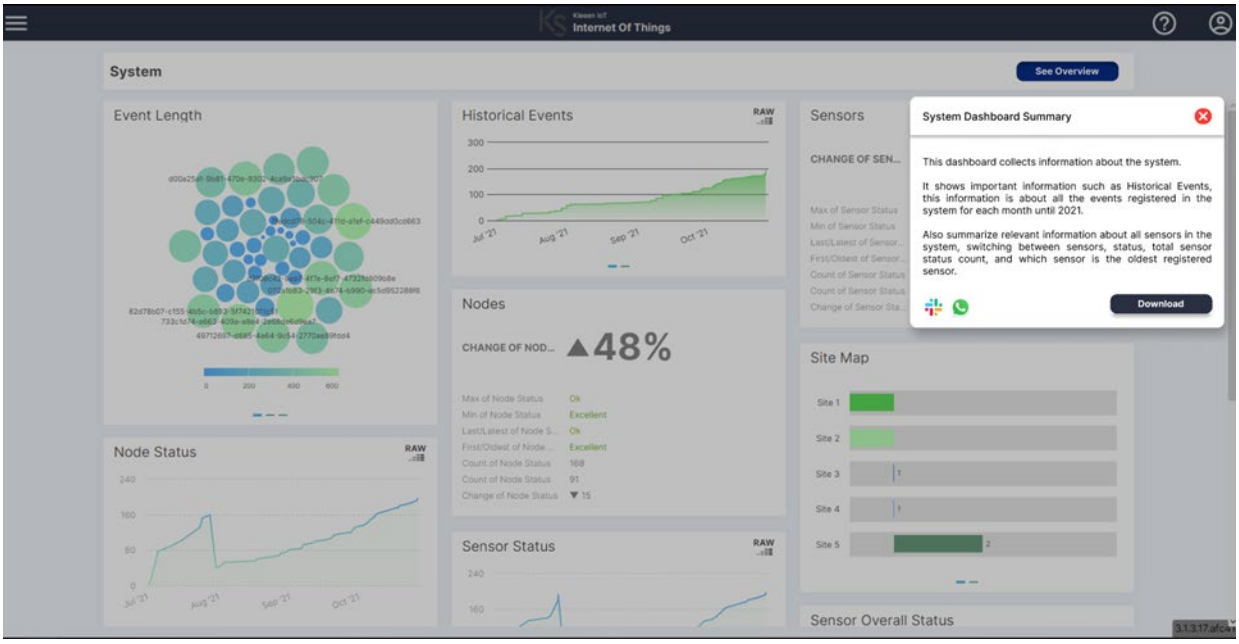
A continuación, en la **Figura 37** y la **Figura 38** se presenta la implementación del boceto realizado previamente.

Figura 37:
Prototipo de Interfaz de Usuario: Creación y Lectura de Resúmenes (Vista del botón de creación).



Nota. Fuente: Elaboración propia

Figura 38:
Prototipo de Interfaz de Usuario: Creación y Lectura de Resúmenes (Vista del resumen)



Nota. Fuente: Elaboración propia

4.4. Análisis de Satisfacción del Usuario

Con respecto a las **Variables de la Investigación**, se realizan diferentes reuniones con el vicepresidente de ingeniería de la empresa, y se obtiene información que permite la identificación de problemática actual, así como las percepciones manifestadas por el principal cliente de Kleeen Software, mostradas en el apartado de **Planteamiento del problema.**

Para analizar la satisfacción del usuario se realiza una encuesta de una escala de Likert la cual se puede encontrar en el **Apéndice C**.

Sin embargo, al intentar aplicar dicha encuesta, no se obtiene respuesta por parte del único cliente usuario de Kleeen Software IDE, por lo que se trabaja con lo recopilado en la información proveída por el vicepresidente de ingeniería de la empresa.

4.5. Conclusiones del Capítulo

Producto de lo desarrollado en el capítulo **Análisis de Resultados,** se concluye lo siguiente:

- De acuerdo con el análisis FODA realizado en el presente proyecto, se determinó que el escaso entendimiento del perfil de los clientes, la falta de diseño y arquitectura en la construcción de las extensiones del sistema y la centralización del conocimiento en pocas personas constituyen las principales debilidades de la organización.
- Por otro lado, el sistema Kleeen Software IDE cuenta con documentación de arquitectura y de procesos, también se apoya en tecnologías modernas como los servicios de AWS para la construcción de subsistemas modularizados como KAPI, el modelo conceptual y la canalización de las visualizaciones. Lo anterior es fundamental para plantear una estrategia que permita el diseño de aplicaciones o extensiones para el sistema actual.
- Durante el diagnóstico de la situación actual, también se determinó que existe un incremento en los competidores directos y cambios constantes en las tecnologías utilizadas, estos aspectos, junto con la difícil escalabilidad presente en los sistemas, representan amenazas que pueden evolucionar en riesgos para el negocio, por lo tanto, se deben tomar medidas para mitigar o afrontar las situaciones que se podrían manifestar.
- Al realizar diferentes reuniones con el vicepresidente de ingeniería de la empresa, se obtiene información que permite identificar la problemática actual, así como las percepciones manifestadas por el principal cliente de Kleeen Software; sin embargo, al aplicar una encuesta de Likert para validar estas opiniones, no se obtiene respuesta.
- Como resultado de diferentes reuniones, se obtienen un total de diecinueve requerimientos por parte del cliente, de los cuales diez son requerimientos funcionales.
- Tras el análisis de los diez requerimientos funcionales, se concluye que ocho de estos son de prioridad alta (*must*), uno es de prioridad media (*should*), y uno de prioridad baja (*could*).

5. Propuesta de Solución

En el presente capítulo se describe a fondo la propuesta de solución ante la problemática identificada anteriormente, de manera que permita la cobertura de los requerimientos y las necesidades de la empresa Kleeen Software, y, a su vez, el cumplimiento de los objetivos específicos y general. Este capítulo se subdivide en el diseño creado para la solución del problema, y el estudio de costos asociado al mismo.

5.1. Diseño de la Propuesta

En la sección **generación de lenguaje natural** se define cómo se pueden crear sistemas que sean capaces de generar textos o contenido comprensible por el ser humano, de manera que las computadoras y los humanos interactúen clara y fluidamente. Así pues, debido a la problemática identificada en el apartado **Situación Problemática** y el **Análisis de Resultados**, se identifica la necesidad y el deseo, por parte de la organización, de un sistema de generación de lenguaje natural (en este caso, se enfoca en el diseño del *software* que soporte este sistema), el cual permita brindar respuestas contextualizadas e inteligentes al usuario final, basándose en su contexto e intenciones.

Por esto, basándose en lo descrito en el **Procedimiento Metodológico de la Investigación**, a continuación, se propone un diseño de *software* para satisfacer los requerimientos recolectados en el apartado **Necesidades del sistema Kleeen Software IDE**.

Como parte de garantizar una mayor facilidad y comprensión del documento, el sistema de generación de lenguaje natural, del cual se propone el diseño de *software*, se denominará “Kleeen Software Natural Language Generation”, o “KS-NLG”, por su acrónimo.

De acuerdo con lo descrito en el apartado **Clasificación de Sistemas de GLN**, se propone que KS-NLG sea según la entrada del sistema, esto debido a la necesidad encontrada de generar resúmenes o informes con datos extraídos de bases de conocimiento, bases de datos, o números.

También, según lo explicado en la sección **Enfoques genéricos para abordar GLN**, se propone que KS-NLG tenga un enfoque basado en conocimiento, esto porque la naturaleza del sistema y el factor común de este tipo de sistemas es centrarse en la capacidad para representar explícitamente el conocimiento utilizando herramientas como las ontologías, conjuntos de reglas o tesauros.

Para la solución propuesta, solamente se mostrará una arquitectura y diseños de alto nivel del sistema, por lo tanto, no se incluye la vista física, ya que no se contempla la capa física de la arquitectura ni la vista lógica de los módulos a un bajo nivel.

5.1.1. Diseño de Software

Retomando lo explicado en el apartado **Diseño y Desarrollo del software**, se entiende que dicho proceso contempla cinco fases, las cuales se desarrollan a continuación como parte de la propuesta.

5.1.1.1. Diseño de los Datos

El diseño de datos corresponde a definir la relación entre los elementos estructurales de un sistema, para ello se debe crear un modelo de datos. Debido a la naturaleza del presente proyecto, se conoce la necesidad de recibir datos, manipularlos y generar una respectiva salida esperada. Los datos o elementos de información identificados en los requerimientos recopilados se muestran en la **Tabla 22.**

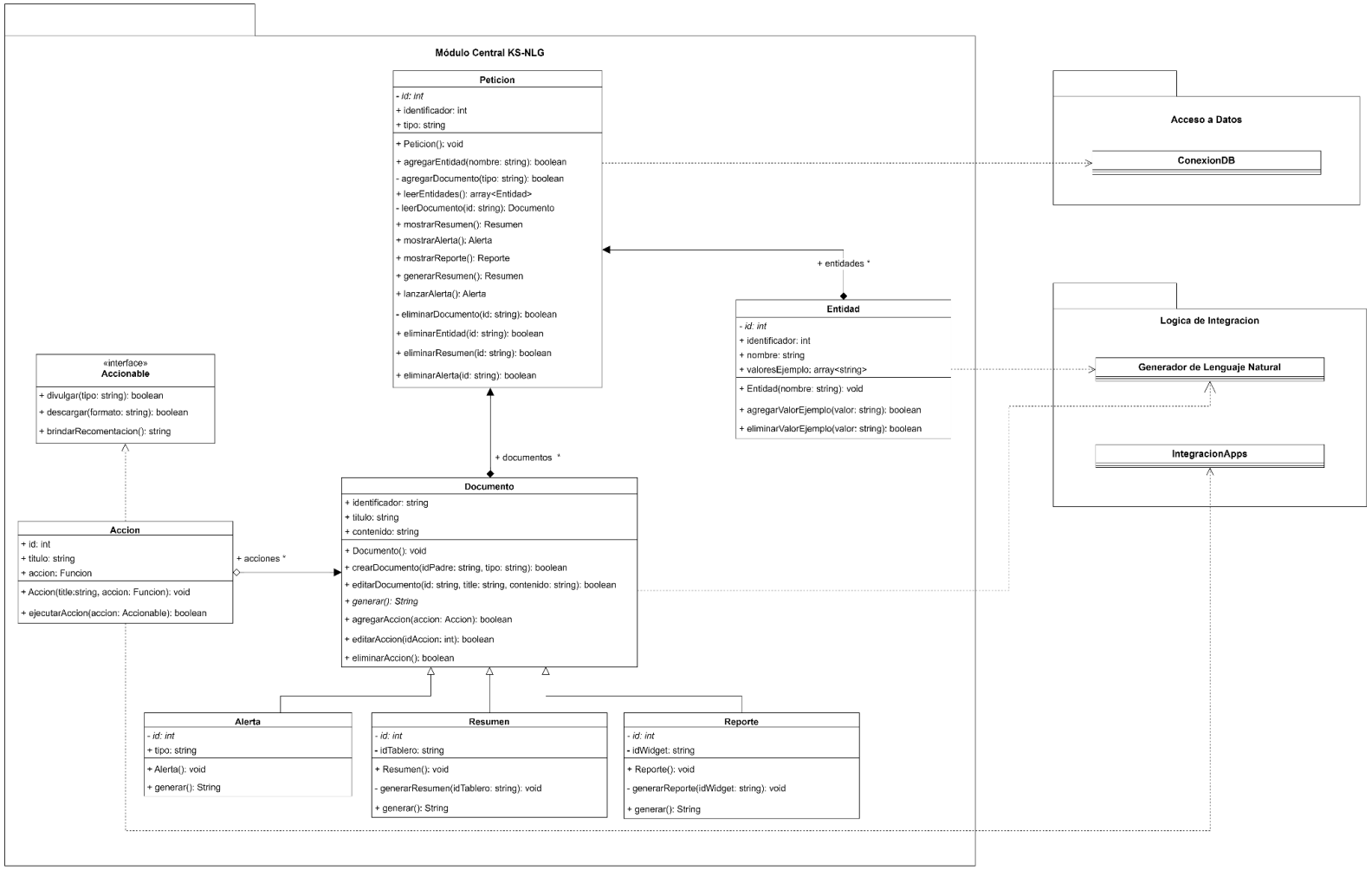
Tabla 23:
Datos Identificados para KS-NLG

Nombre	Objetivo	Característica
Peticiones	Son las solicitudes que se realizan a KS-NLG, estas serán para la generación de documentos y sus respectivas acciones requeridas.	Posee documentos
Entidad	Son las entidades necesarias para el reconocimiento de datos y la generación de lenguaje natural.	Posee valores ejemplos
Documento	Hace referencia a los documentos de lenguaje natural generados, estos pueden tomar distintas formas.	Pueden ser: alertas, resúmenes, reportes.
Alerta	Las alertas son texto generado basado en acciones o cambios en el prototipo del cliente. Se disparan cuando se cumplen las sentencias especificadas por el usuario.	
Resumen	Son texto generado que recopila la información referente a los tableros de información. Esta información no se enfatiza en los datos mostrados, sino en el tipo de datos, así como el propósito del tablero correspondiente.	
Reporte	Hace referencia a los datos de una tarjeta de información (*widget*) en específico, describen el comportamiento que posee, así como los datos relacionados con el mismo.	
Acción	Son acciones ejecutables asociadas a un documento.	Pueden especificar comportamientos como: descargar, compartir o divulgar.

Nota. Fuente: Elaboración propia.

En la **Figura 39** se muestra un diagrama de clases que explica mejor la relación de los datos expuestos en la **Tabla 16.**

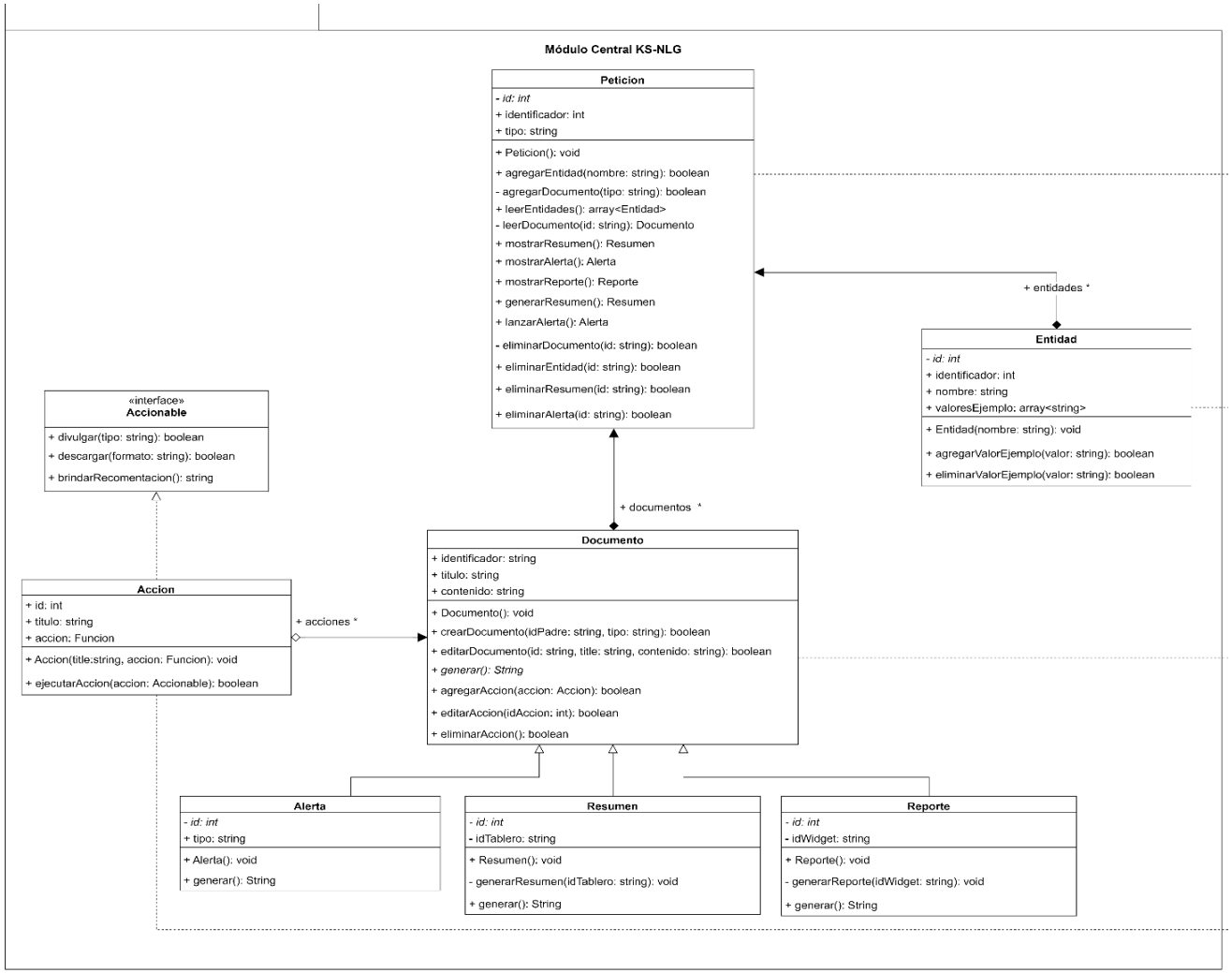
Figura 39:
Diagrama de Clases: KS-NLG



Nota. Fuente: Elaboración propia.

En la **Figura 40** se muestra un diagrama de clases del módulo central KS-NLG.

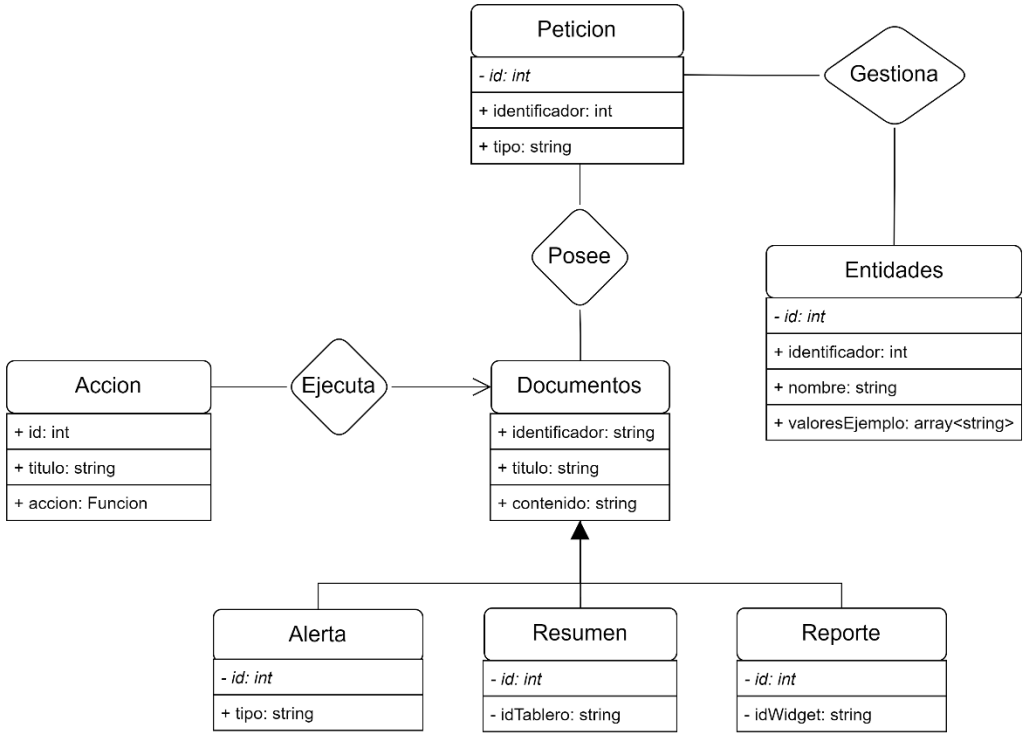
Figura 40:
Diagrama de Clases: Paquete del Módulo Central KS-NLG



Nota. Fuente: Elaboración propia.

A manera de simplificar la comprensión de la **Figura 40**, en la **Figura 41** se muestra el modelo de datos conceptual del sistema KS-NLG.

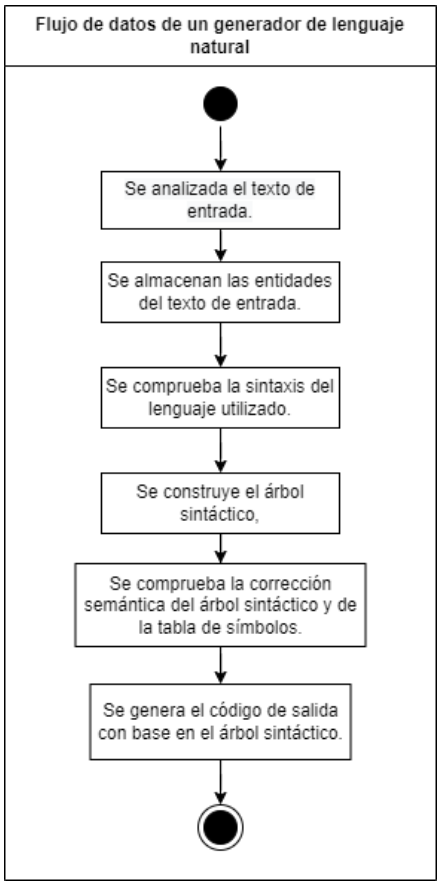
Figura 41:
Modelo de Datos Conceptual: KS-NLG



Nota. Fuente: Elaboración propia.

Como parte del modelo de datos propuesto, se contemplan los datos requeridos para la generación de los documentos. Estos datos se manipulan y cambian tomando como base lo propuesto por Sommerville (2005), también explicado en el apartado de **Componentes**. Un ejemplo de esto se observa con el siguiente diagrama de actividad (Ver la **Figura 42).**

Figura 42:
Diagrama de Actividad: Flujo de datos de un generador de lenguaje natural.



Nota. Fuente: Elaboración propia.

El diagrama anterior incluye los componentes genéricos para el funcionamiento de este sistema, los cuales son:

- Analizador léxico: realiza un análisis de los datos de entrada y los convierte a un formato interno preestablecido.
- Tabla de símbolos: se utiliza como un repositorio de datos, el cual almacena las entidades que se encuentran en el texto de entrada, así como las definidas por el usuario.
- Analizador sintáctico: comprueba la sintaxis del lenguaje esperado, utiliza la gramática requerida y construye un árbol sintáctico.
- Árbol sintáctico: permite interpretar la estructura del documento que se está generando.

- Analizador semántico: toma los datos del árbol sintáctico y de la tabla de símbolos generada anteriormente y corrige el texto.
- Generador de código: este se encarga de tomar los datos generados previamente y completar el texto de salida esperado.

5.1.1.2. Diseño Arquitectónico

El diseño arquitectónico permite definir la organización de un sistema y la estructura general a alto nivel de este. En esta fase se identifican los principales componentes estructurales del sistema, así como la relación entre ellos.

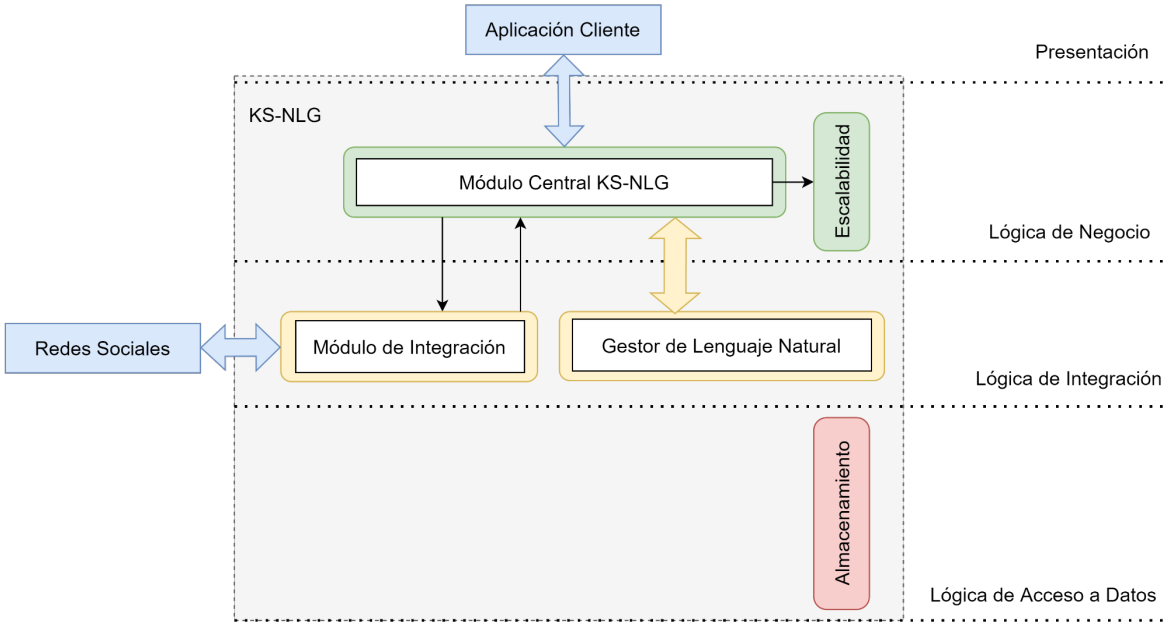
5.1.1.2.1. Diseño de la Arquitectura General del Sistema

Partiendo de los requerimientos definidos, así como los casos de uso, se encuentra la necesidad de diferentes módulos para el funcionamiento del KS-NLG. Estos son:

- Módulo Central KS-NLG: es el módulo encargado de la gestión del sistema. En este módulo se realizan las tareas de CRUD, divulgación y descarga de datos.
- Módulo de Integración: permite la conexión con sistemas o APIs externos, tal como las redes sociales o plataformas de mensajería como Gmail, Slack, Messenger, etc.
- Gestor de Lenguaje Natural: se refiere al gestor en el cual se realizan las tareas descritas en el apartado **Fases de un sistema GLN.**

En la **Figura 43** se muestra a alto nivel la arquitectura propuesta.

Figura 43:
Diseño de Arquitectura de Alto Nivel



Nota. Fuente: Elaboración propia.

5.1.1.2.2. Diseño de la Arquitectura del Módulo de Gestor de Lenguaje Natural

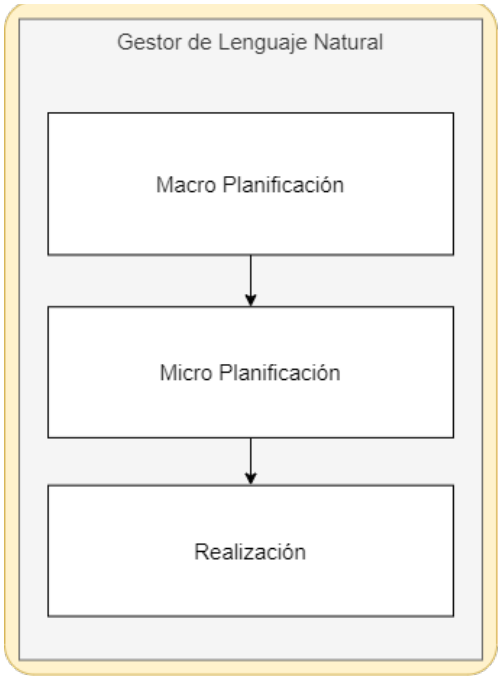
Como parte de la arquitectura y distribución del módulo "Gestor de Lenguaje Natural", se propone una arquitectura integrada, basada en la propuesta por Chal, G. (2007), donde todas las decisiones son tomadas dentro de un proceso, el cual se encuentra estructurado jerárquicamente, sin estar necesariamente modularizado.

Este módulo cuenta con submódulos, los cuales son requeridos como parte de la arquitectura propuesta. Estos son:

- Macro planificación: posee las siguientes tareas:
  - Determinar el Contenido: define la información que debe generarse como salida del sistema. Este proceso contempla aspectos como objetivos de comunicación, receptor de la información y restricciones.
  - Estructuración del Documento: se establece y define el orden y la estructura correcta para la información del texto generado. Además, contempla aspectos como la agrupación de la información y los tipos de recopilación o resumen, los cuales pueden ser generales o específicos según tema.
- Micro planificación: una vez que se ha ejecutado la macro planificación y se ha definido el contenido que se requiere, se realiza la construcción de las frases (estructuras sintácticas) deseadas. Este submódulo posee las siguientes tareas:
  - Agregaciones: este proceso consiste en unir las frases generadas de manera que se pueda producir un texto fluido y comprensible por el usuario lector.
  - Lexicalización: se seleccionan los verbos, adverbios, adjetivos, etc., necesarios para representar el contenido seleccionado por el módulo. Para simplificar este proceso se definen plantillas de resultados.
  - Generador de Referencias: se enfoca en decidir expresiones para referirse a las entidades seleccionadas por el usuario. Un ejemplo de esto es:
    - **Entidad:** marca Audi
    - **Referencias:**
      - La marca Audi
      - La marca
      - El automóvil Audi
      - Audi
      - El Audi
- Realización: al finalizar la micro planificación, se toma la salida del proceso de planificación y se convierte en texto real y de sentido. Posee las tareas de:
  - Realización Lingüística: se toman las oraciones abstractas resultado del proceso anterior y, por medio de reglas del lenguaje, como la morfología, sintaxis y ortografía, se convierte en texto real.
  - Estructuración del Texto: se toman las estructuras abstractas resultantes de las tareas anteriores y se les agrupa u ordena de acuerdo con su etiqueta (oración, párrafo, etc.).

En la **Figura 44**, se muestra la organización de la arquitectura para este módulo.

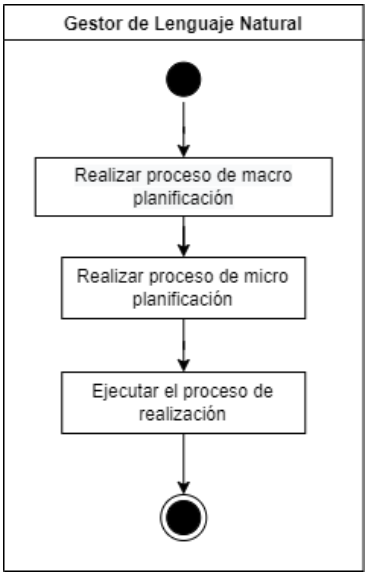
Figura 44:
Diseño de Arquitectura del Módulo Gestor de Lenguaje Natural



Nota. Fuente: Elaboración propia.

Cada uno de los submódulos dentro del Gestor de Lenguaje Natural realiza una serie de tareas, cuya salida es la entrada del siguiente, de manera que la ejecución es secuencial, dando como resultado la salida de texto generado de acuerdo con las entradas recibidas. En la **Figura 45** se muestra el diagrama de actividad correspondiente.

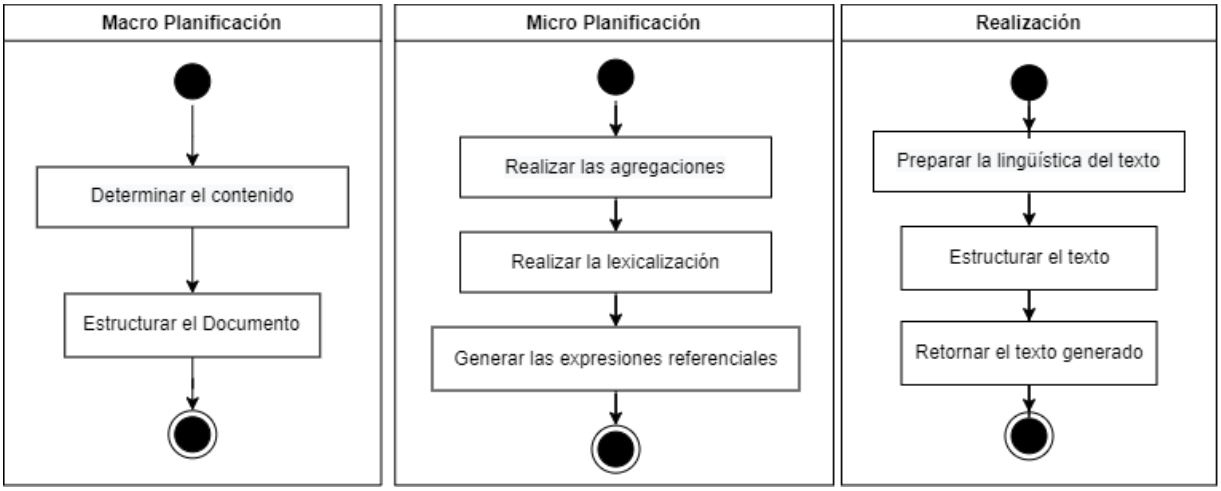
Figura 45:
Diagrama de Actividad: Módulo Gestor de Lenguaje Natural



Nota. Fuente: Elaboración propia.

Para esclarecer este proceso, en la **Figura 46** se muestra el diagrama de actividad para cada respectivo submódulo.

Figura 46:
Diseño de Arquitectura del Módulo Gestor de Lenguaje Natural



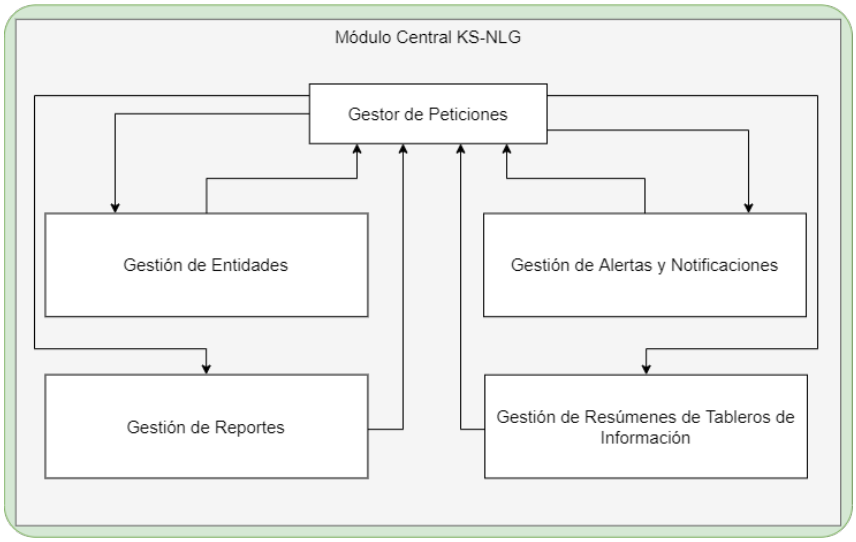
Nota. Fuente: Elaboración propia.

5.1.1.2.3. Diseño de la Arquitectura del Módulo Central KS-NLG

El módulo central es la estructura primordial del sistema, en el cual se llevarán a cabo tareas como la gestión de entidades, desarrollo de informes, resúmenes, notificaciones y demás, todas estas administradas por un gestor de peticiones, el cual se encargará de la lógica general de procesar y seleccionar las tareas requeridas y determinar cuál submódulo será utilizado según las necesidades del usuario.

En la **Figura 47 s**e muestra la organización de la arquitectura para este módulo.

Figura 47:
Diseño de Arquitectura del Módulo Central KS-NLG



Nota. Fuente: Elaboración propia.

5.1.1.3. Diseño de la Interfaz

De acuerdo con Pressman (2011), el diseño de la IU, también denominada diseño de la usabilidad, incorpora elementos estéticos, los cuales pueden ser: distribución, color, gráficos o interacciones, elementos ergonómicos como la distribución de componentes, información y colores, así como elementos técnicos (patrones de la IU y patrones reutilizables).

Para el presente proyecto no se cuenta con una interfaz completa o elaborada, debido a que el sistema propuesto no tiene relación visual directamente con el usuario, pues solo recibe peticiones, analiza datos y retorna una salida en forma de texto. Sin embargo, la aplicación cliente que consumirá este sistema requiere de componentes de IU como paneles, botones, interacciones y demás, esto con el fin de realizar las peticiones y mostrar los resultados esperados. Por tal motivo, se propone una guía que puede simplificar y esclarecer aspectos de IU necesarios para la implementación del KS-NLG.

De acuerdo con la ISO 9241-210:2010, se definen diferentes aspectos con respecto a cómo se debe presentar la información en un sistema de *software*, debido a que dicha norma dicta la manera correcta para que el usuario pueda realizar una tarea visual de forma satisfactoria y eficiente.

Como requerimiento de esta norma, la información siempre debe ser legible, comprensible y fácilmente interpretable, todo esto apegado a las necesidades del usuario.

Además, esta norma define tres aspectos o categorías para garantizar una buena presentación de la información, los cuales son:

- El tamaño y formato de los caracteres alfanuméricos.
- La organización de la información.
- La utilización de códigos técnicos.

De acuerdo con Krug, S. (2018), existen tres principios para la usabilidad de un sistema, estos son:

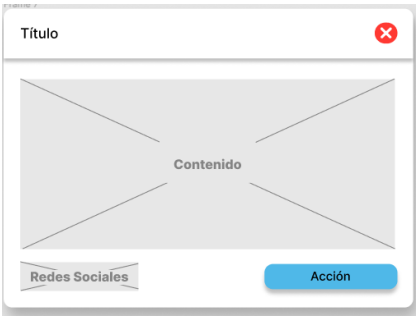
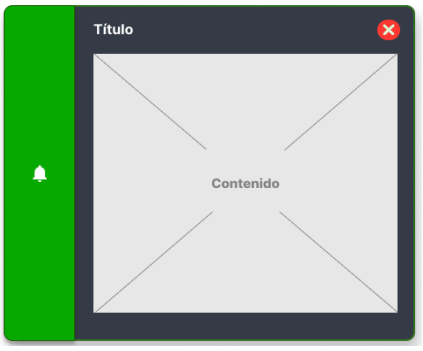
1) No me hagas pensar: el usuario no debe cuestionarse dónde se encuentra, por dónde empezar, o dónde se encuentran las cosas de interés.
2) No importa cuántos clics, sino que no sean difíciles: en ocasiones la intuición es más poderosa que otras cosas, por eso, el usuario debe ser capaz de intuir qué debe realizar o qué acciones se le permite hacer.
3) Quitar la mitad de las palabras de la mitad: extender las explicaciones no siempre mejora el entendimiento del usuario, ser conciso y puntual es más beneficioso en muchos casos.

Según Nielsen, J. (1994), en la usabilidad web existen ocho reglas importantes, las cuales son:

1) En internet el usuario es el que manda.
2) En internet la calidad se basa en la rapidez y la fiabilidad.
3) Seguridad.
4) La confianza es algo que cuesta mucho ganar y se pierde con un mal enlace.
5) Si quieres hacer una página decente, simplifica, reduce y optimiza.
6) Pon las conclusiones al principio.
7) No hagas perder el tiempo a la gente con cosas que no necesitan.

Como se observa en el apartado **Experimento de Mago de Oz**, la integración visual con el sistema Kleeen Software IDE es natural, es decir, al momento de implementar el sistema, se utilizan los componentes existentes, así como el uso de colores de acuerdo con tema del producto original. Sin embargo, a continuación, en la **Tabla 23** se presentan los componentes identificados.

Tabla 24:
Componentes

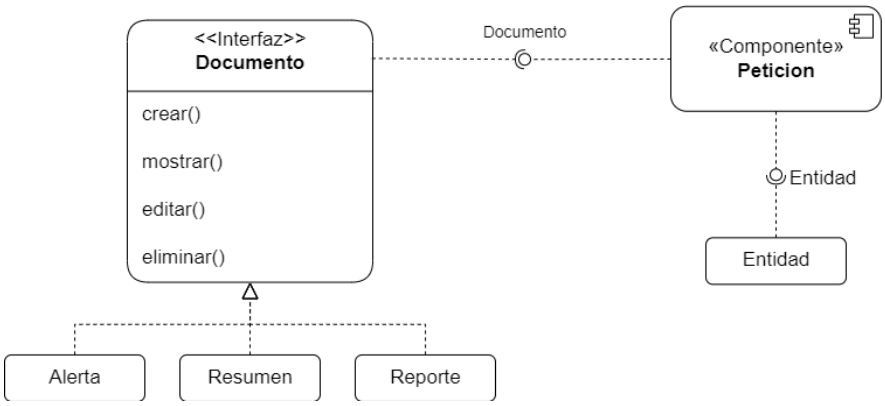
Componente	GUI	Propósito	Argumento de diseño
Botón Primario		Enviar la petición para la generación de resúmenes o informes.	De acuerdo con Ruiz Calle, J. (2018), un botón primario es aquel que está diseñado con elementos que requieran dar un mayor énfasis, distinguibles por su elevación y su color en todo el botón. Deberían contener acciones que son importantes para la aplicación.
Tarjeta		Mostrar los resúmenes o informes generados, habilitar la opción de divulgar el texto generado o descargarlo.	Según Laubheimer, P. (2016), una tarjeta es un contenedor de unas pocas piezas de datos relacionados. Se utiliza para agrupar información y presentar un resumen de datos o acciones de fácil acceso.
Alertas		Mostrar las alertas generadas, habilitar la opción de divulgar el texto generado o descargarlo.	De acuerdo con la página oficial de Patternfly, las Alertas (Notificaciones *Toast*) notifican al usuario sobre una ocurrencia del sistema.

Nota. Fuente: Elaboración propia.

5.1.1.4. Diseño de Componentes

En la **Figura 48** se muestra el diseño de componentes a alto nivel del Módulo Central KS-NLG, el cual permite una mejor perspectiva modular, desplegable y sustituible del sistema.

Figura 48:
Diseño de Componentes

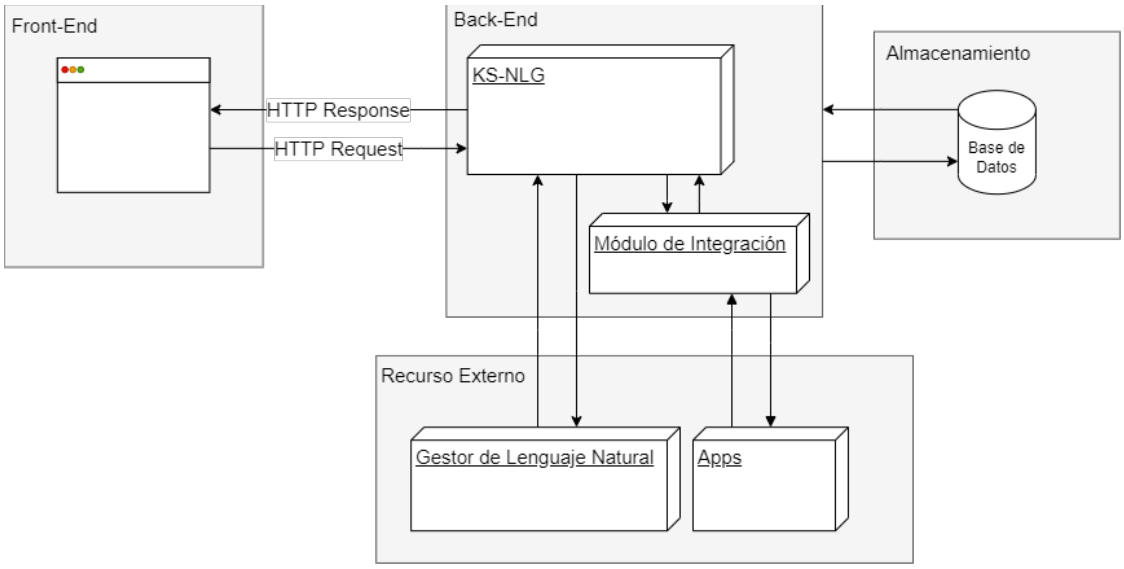


Nota. Fuente: Elaboración propia.

5.1.1.5. Diseño del Despliegue

A continuación, en la **Figura 49** se presenta el diseño de despliegue propuesto para KS-NLG. Los elementos incluidos indican cómo se distribuyen el *software* y los subsistemas dentro de un ambiente físico.

Figura 49:
Diseño de Despliegue



Nota. Fuente: Elaboración propia.

5.1.2. Evaluación de Diseño

Al momento de verificar un diseño de *software* existen diferentes estrategias o técnicas. De acuerdo con lo descrito en el apartado **Verificación y Validación del *Software* (V & V)**, se comprenden tres técnicas, de este modo, para el presente proyecto se utilizará la Verificación de Exactitud de **Estrategia De Cuarto Limpio**, combinado con las **Medidas de Aceptación y Valoración de Diseño de *Software*.**

5.1.2.1. Estrategia de Cuarto Limpio

Para el presente proyecto no se contemplan todas las fases de esta estrategia, esto debido a que no se encuentra disponible la implementación ni el plan del proyecto. A continuación, en la **Tabla 24** se muestran las fases expuestas, así como su equivalencia en el presente documento.

Tabla 25:
Fases de Estrategia de Cuarto Limpio en el Presente Proyecto

Fase Estrategia de Cuarto Limpio	Apartado del Proyecto
Planeación del incremento	No aplica
Recopilación de requerimientos	**Necesidades del sistema Kleeen Software IDE**.
Especificación de estructura de caja	**Especificación de Estructura de Caja**
Diseño formal	**Diseño de Software**
Verificación de exactitud	**Verificación de Exactitud**
Generación, inspección y verificación de código	No aplica
Prueba de uso estadístico	**Prueba de uso estadístico**
Certificación	**Certificación**

Nota. Fuente: Elaboración propia.

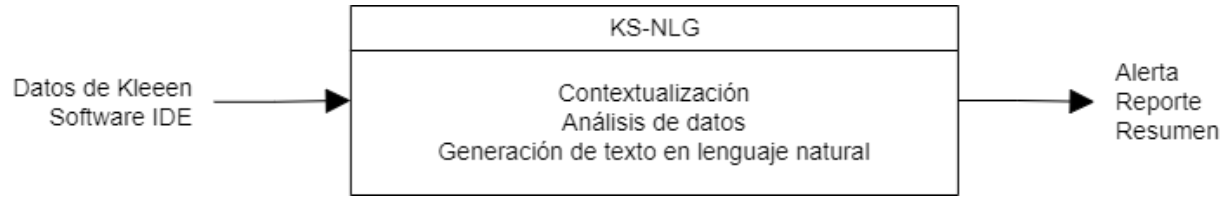
5.1.2.1.1. Especificación de Estructura de Caja

De acuerdo con lo descrito en el apartado de **Estrategia De Cuarto Limpio**, seguidamente se muestra la especificación funcional del sistema propuesto de acuerdo con los requerimientos del apartado **Necesidades del sistema Kleeen Software IDE**, para ello, se realizaron las especificaciones de caja negra, caja de estado, y caja clara.

- ***Especificación de Caja Negra***

A continuación, en la **Figura 50** se muestra la Caja Negra que especifica el comportamiento de KS-NLG.

Figura 50:
Caja Negra KS-NLG

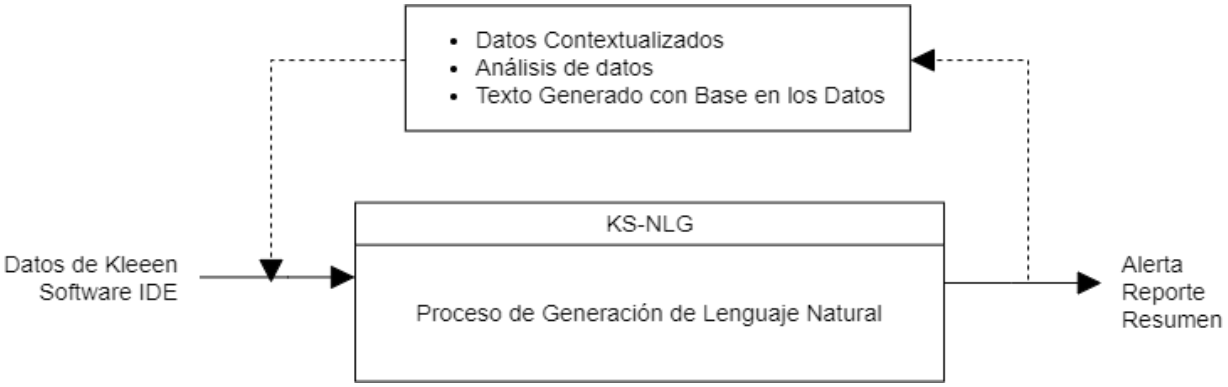


Nota. Fuente: Elaboración propia.

- ***Especificación de Caja de Estado***

A continuación, en la **Figura 51** se muestra la Caja de Estados que especifica qué estados posee el sistema durante el proceso descrito en la Caja Negra mostrada anteriormente.

Figura 51:
Caja de Estado KS-NLG

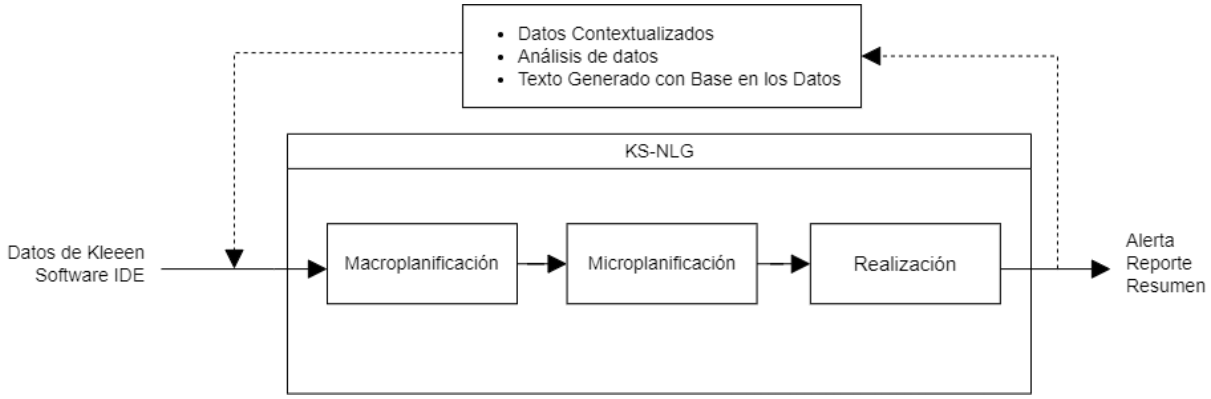


Nota. Fuente: Elaboración propia.

➢ ***Especificación de Caja Clara***

A continuación, la **Figura 52** se muestra la Caja Clara que especifica el procedimiento requerido para obtener los estados descritos en la Caja de Estados mostrada anteriormente.

Figura 52:
Caja Clara KS-NLG



Nota. Fuente: Elaboración propia.

La especificación de estructura de caja muestra, a nivel macro, cómo funciona el sistema propuesto, y este, a su vez, coincide con los diagramas propuestos en el apartado de **Diseño de *Software***, de manera que, a nivel conceptual, la propuesta de solución es correcta de acuerdo con lo indicado por Hevner, A. R. y H. D. Mills (1993).

5.1.2.1.2. Verificación de Exactitud

Para la realización de este proceso de evaluación, se toman en cuenta los conceptos descritos en **Medidas de Aceptación y Valoración de Diseño de Software**, donde se toman las preguntas de exactitud requeridas, las cuales son:

- ¿Cubre el diseño todos los requisitos funcionales?
- ¿Resulta ambigua la documentación del diseño?
- ¿Se ha aplicado la notación de diseño correctamente?
- ¿Se han definido correctamente las interfaces entre elementos del diseño?
- ¿Es el diseño suficientemente detallado como para que sea posible implementarlo en el lenguaje de programación elegido?

A continuación, se responden todas y cada una de estas preguntas.

- ***¿Cubre el diseño todos los requisitos funcionales?***

Para responder esta pregunta, se realiza una matriz de cobertura de los requerimientos, la cual se presenta en la **Tabla 25**.

Tabla 26:
Matriz de cobertura de los requerimientos

ID	Nombre del requerimiento	Descripción del requerimiento	Caso de Uso	Urgencia	Valor del Negocio	Prioridad	Diseño de Arquitectura	Diseño de Datos
REQ-1	El sistema debe permitir la integración con los sistemas de la empresa.	Se debe garantizar la integración con el sistema Kleeen Software IDE en primera instancia (o algún otro sistema posterior).	No aplica.	5	5	25	Módulo de Integración	Paquete: Logica de Integracion Clases: IntegracionApps
REQ-2	El sistema debe recibir datos de los sistemas de la empresa.	Se debe permitir la recepción de datos números, bases de datos, bases de conocimiento, o corpus etiquetado.	No aplica.	5	5	25	Módulo de Integración	Paquete: Logica de Integracion Clases: IntegracionApps

ID	Nombre del requerimiento	Descripción del requerimiento	Caso de Uso	Urgencia	Valor del Negocio	Prioridad	Diseño de Arquitectura	Diseño de Datos
REQ-3	El sistema de reconocer entidades.	Se debe permitir el reconocimiento de entidades con el propósito de realizar el proceso de generación de lenguaje natural.	Gestión de Entidades	5	5	25	Módulo Central KS-NLG	Paquete: Módulo Central KS-NLG Clases: Peticion y Entidad
REQ-4	El sistema debe permitir la gestión de entidades (crear, leer, actualizar, borrar).	Se debe permitir que el usuario gestione las entidades de interés.	Gestión de Entidades	5	5	25	Módulo Central KS-NLG	Paquete: Módulo Central KS-NLG y Acceso a Datos Clases: Peticion, Entidad, y ConexionDB

ID	Nombre del requerimiento	Descripción del requerimiento	Caso de Uso	Urgencia	Valor del Negocio	Prioridad	Diseño de Arquitectura	Diseño de Datos
REQ-5	El sistema debe permitir la creación de valores ejemplo para las entidades	Se debe permitir que el usuario ingrese valores ejemplos para las entidades de interés registradas.	Gestión de Entidades	5	5	25	Módulo Central KS-NLG	Paquete: Módulo Central KS-NLG y Acceso a Datos Clases: Entidad, y ConexionDB
REQ-6	El sistema debe reconocer datos numéricos y agregaciones	Se debe permitir el reconocimiento de datos recibidos en el sistema.	No aplica.	5	5	25	Módulo Central KS-NLG	Paquete: Módulo Central KS-NLG Clases: Peticion
REQ-7	El sistema debe permitir la gestión de notificaciones de resumen (crear, leer, actualizar, borrar, envío de	Se debe permitir que el usuario gestione las notificaciones de resumen.	Gestión de Alertas y Notificaciones	3	4	12	Módulo Central KS-NLG y Gestor de Lenguaje Natural	Paquete: Módulo Central KS-NLG y Acceso a Datos

ID	Nombre del requerimiento	Descripción del requerimiento	Caso de Uso	Urgencia	Valor del Negocio	Prioridad	Diseño de Arquitectura	Diseño de Datos
	notificaciones)							Clases: Peticion, Documento, Alerta, y ConexionDB
REQ-8	El sistema debe permitir la gestión de reportes de informe con los indicadores de interés para el negocio (crear, leer, actualizar, borrar, divulgación de reportes, exportación de reportes)	Se debe permitir que el usuario gestione los reportes de informe	Gestión de Reportes	5	5	25	Módulo Central KS-NLG y Gestor de Lenguaje Natural	Paquete: Módulo Central KS-NLG y Acceso a Datos Clases: Peticion, Documento, Reporte, y ConexionDB

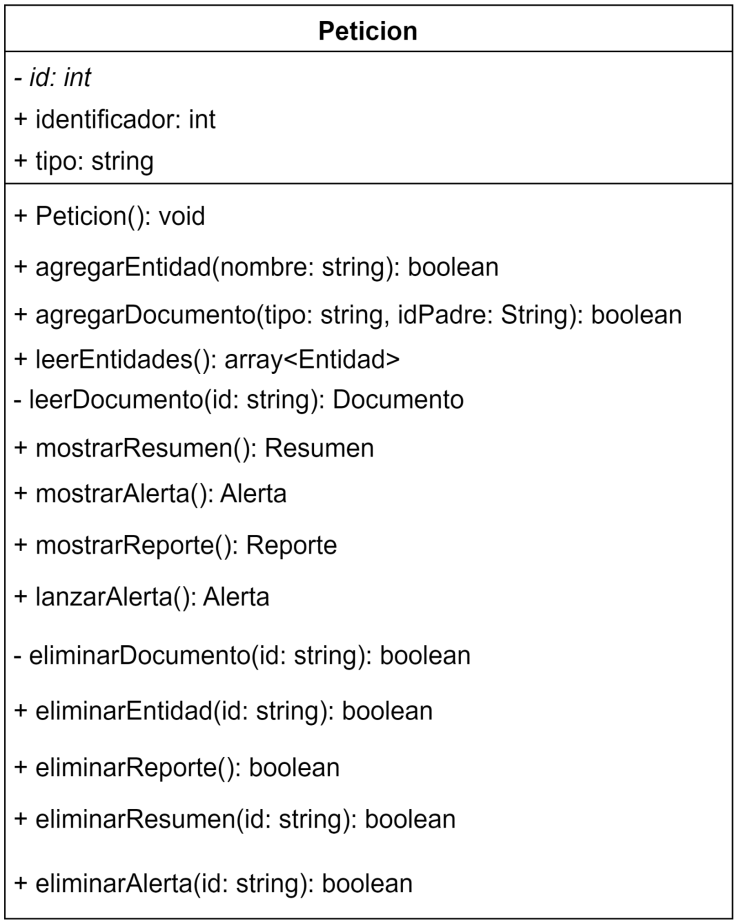
ID	Nombre del requerimiento	Descripción del requerimiento	Caso de Uso	Urgencia	Valor del Negocio	Prioridad	Diseño de Arquitectura	Diseño de Datos
REQ-9	El sistema debe generar resúmenes sobre dashboards (historia y propósito del dashboard) (crear, leer, actualizar, borrar, divulgación de resúmenes, exportación de resúmenes)	Se debe permitir que el usuario gestione los resúmenes sobre dashboards	Gestión de Resúmenes de Tableros de Información	5	5	25	Módulo Central KS-NLG y Gestor de Lenguaje Natural	Paquete: Módulo Central KS-NLG y Acceso a Datos Clases: Peticion, Documento, Resumen, y ConexionDB
REQ-10	El sistema debe proporcionar recomendaciones de redacción de acuerdo con interés del usuario (formalismos, populismos)	Se permitirá la recomendación de redacción de textos cuando sea pertinente, ya sea por aspectos de formalismos o necesidades del cliente.	No aplica.	2	3	6	Módulo Central KS-NLG y Gestor de Lenguaje Natural	Paquete: Módulo Central KS-NLG Clases: Peticion, y Accion

Nota. Fuente: Elaboración propia.

- ***¿Resulta ambigua la documentación del diseño?***

Para cada elemento incluido en el diseño, se utilizaron nombres de variables, métodos, clases o paquetes representativos directos y claros, esto con el objetivo de evitar malinformar o confundir al lector. En la **Figura 53** se muestra un ejemplo de esto, así como una descripción detallada de los datos requeridos en el sistema definidos en la **Tabla 21**.

Figura 53:
Clase Peticion



Nota. Fuente: Elaboración propia.

- ***¿Se ha aplicado la notación de diseño correctamente?***

Con respecto a lo explicado en el apartado **Diseño y Desarrollo del Software**, existen diferentes patrones que un buen diseño debe cumplir. Estos patrones son:

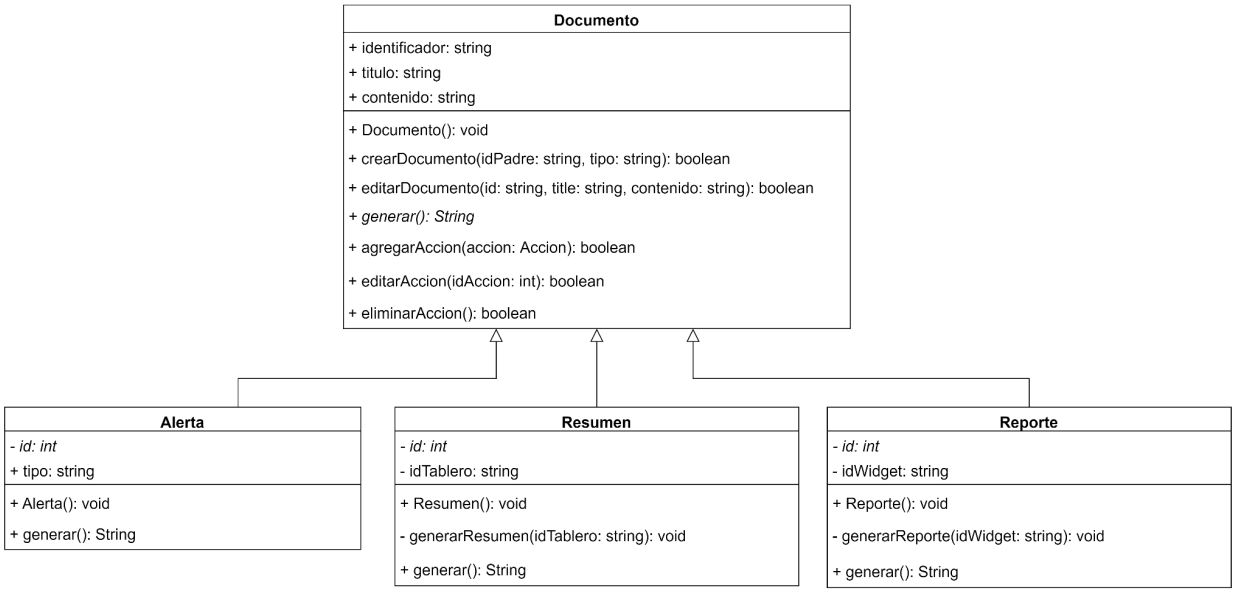
- Principio de Responsabilidad Única

De acuerdo con Jaiswal (2019), este principio supone que "una clase debe tener una y una sola razón para cambiar", esto significa que las clases del sistema deben tener una única razón de existir.

Para el diseño propuesto se utilizaron clases como *Documento*, la cual solo posee la responsabilidad de gestionar los documentos del sistema, y, en el caso de las funciones específicas que esta requiera, se creó una jerarquía, la cual permite que existan diferentes tipos de documentos con funciones específicas cuando se deseen agregar o modificar puntualmente de acuerdo con tipo deseado. Esto se evidencia en la **Tabla 21**, donde se explica la responsabilidad de cada clase diseñada.

En la **Figura 54** se muestra el diagrama que representa dicha jerarquía.

Figura 54:
Diseño de la jerarquía de la clase *Documento*



Nota. Fuente: Elaboración propia.

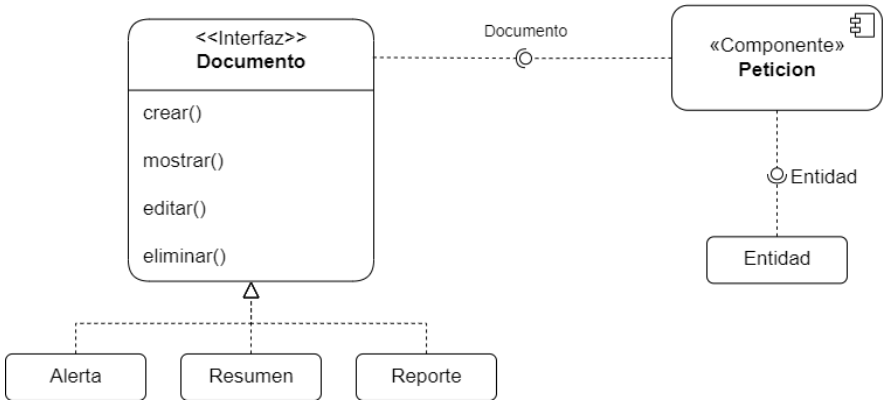
➢ Principio Abierto Cerrado

De acuerdo con el apartado **Diseño y Desarrollo del *Software*** y los principios de diseño de *software* explicados, se entiende que "un módulo [componente] debe ser abierto para la extensión, pero cerrado para la modificación" (Jaiswal, 2019). Es decir, los componentes deben definirse de manera que se puedan extender (funcionalmente) sin la necesidad de modificar su lógica interna.

Para el presente diseño se crean abstracciones que funcionan como un búfer entre la funcionalidad que sea probable extender y la clase del diseño definida.

En la **Figura 55** se muestra el diagrama de componentes de la clase *Peticion*. Por otro lado, es probable que, conforme pase el tiempo, el sistema requiera más tipos de documentos por generar, por esto, la interfaz *Documento* presenta una visión consistente con los componentes para las peticiones, de manera que, si se agrega un nuevo tipo de estos, no se requeriría hacer ningún cambio para la clase *Peticion* (componente), garantizando así el **Principio Abierto Cerrado**.

Figura 55:
Diseño de Componentes

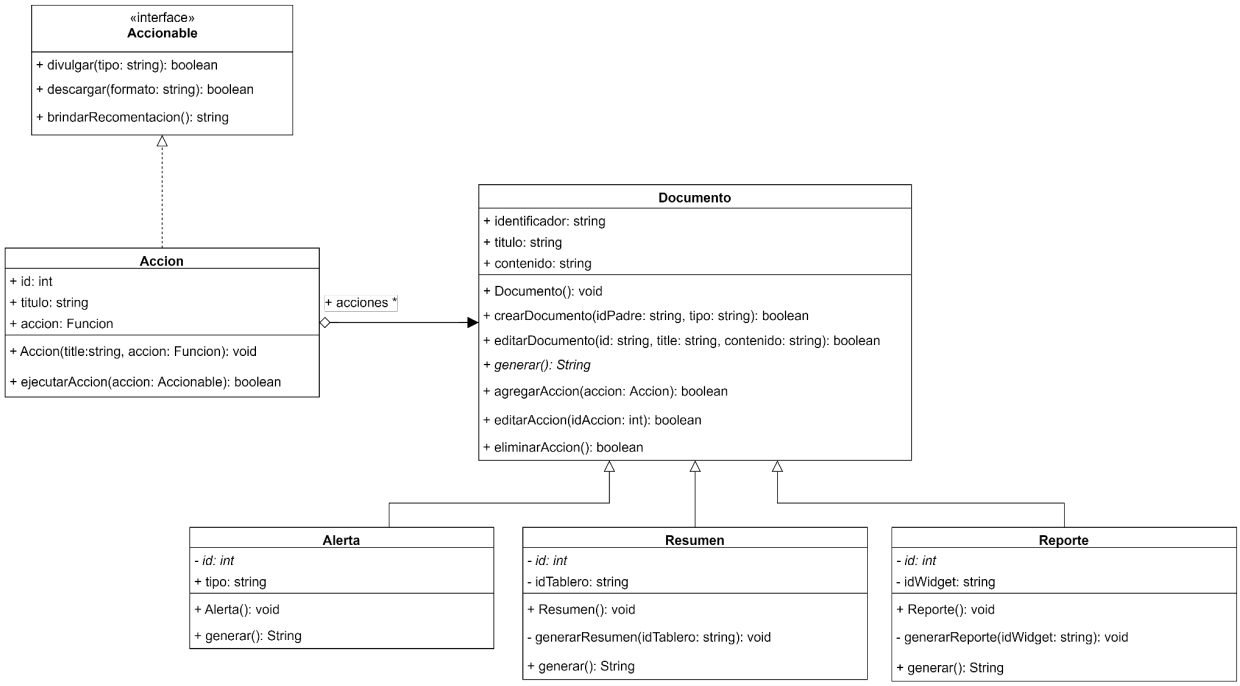


Nota. Fuente: Elaboración propia.

➢ Principio de Sustitución de Liskov

En la **Figura 56**. se evidencia cómo en la clase *Documento* solo se declaran métodos que permitan usarse en sus clases hijas. Para esto, en dicha clase se crearon métodos únicamente necesarios y válidos para sus subclases, de manera que puedan hacer uso de estos sin que existan inconsistencias al momento de ejecución, garantizando así **Principio de Sustitución de Liskov.**

Figura 56:
Diseño de la jerarquía de la clase *Documento*, la clase *Accion* y la interfaz *Accionable*.



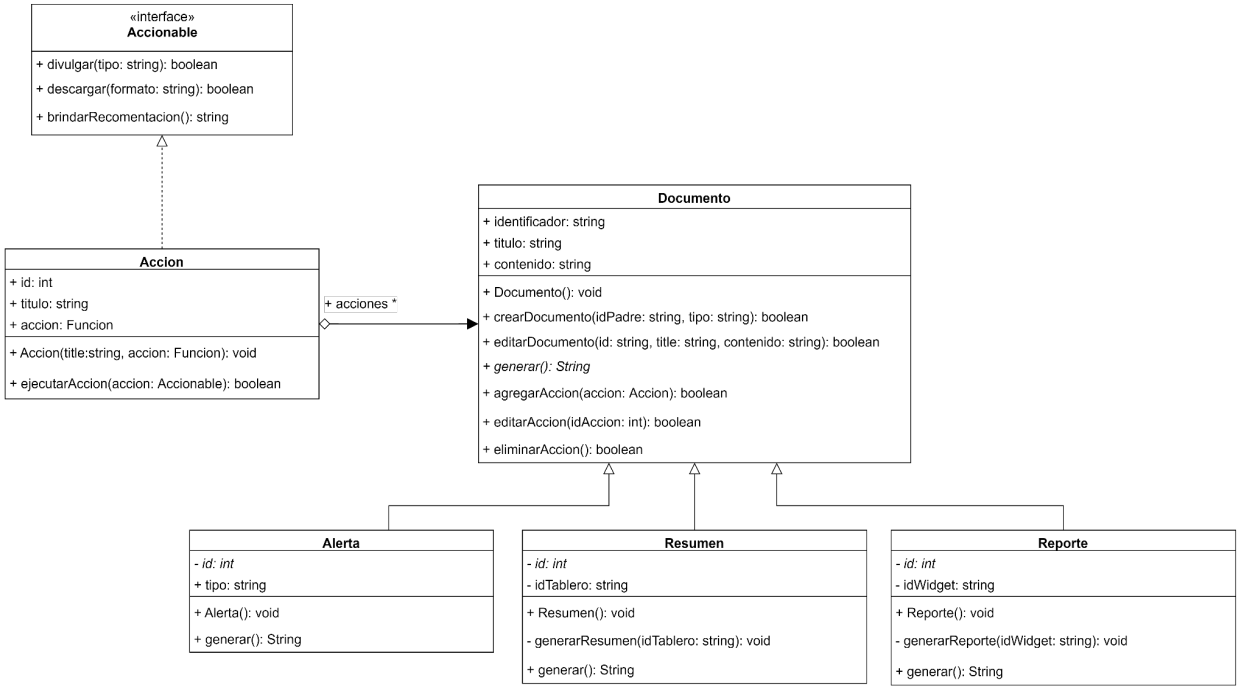
Nota. Fuente: Elaboración propia.

➢ Principio de Segregación de Interfaz

Según los conceptos repasados en **Diseño y Desarrollo del Software**, y los principios de diseño de software explicados, “es mejor tener muchas interfaces específicas del cliente que una sola de propósito general” de acuerdo con Jaiswal (2019), esto quiere decir que los clientes de un programa solo deberían conocer aquellos métodos que realmente usan

En los requerimientos especificados en **Necesidades del sistema Kleeen Software IDE**, una necesidad del cliente es que tanto las alertas como los resúmenes y los reportes puedan tener acciones como divulgar y descargar; por esto, en el diseño propuesto se crea el método *agregarAccion*, que se ejecuta por medio de la clase *Accion*, la cual consume la interfaz de Accionable, de manera que esta puede brindar los métodos requeridos para cada tipo de documento sin caer en errores sobre métodos o interfaces que no vayan a ser necesarios, esto se evidencia en la **Figura 57**.

Figura 57:
Diseño de la jerarquía de la clase *Documento*, la clase *Accion* y la interfaz *Accionable*.



Nota. Fuente: Elaboración propia.

➢ Principio de Inversión de Dependencia (DIP)

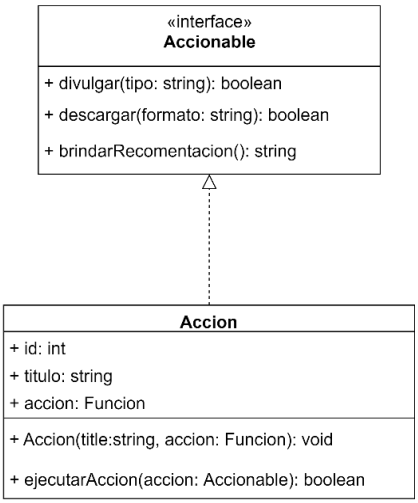
De acuerdo con Jaiswal (2019), en este principio se entiende que "las entidades deben depender de abstracciones, no de concreciones. Indica que el módulo de alto nivel no debe depender del módulo de bajo nivel, sino que deben depender de las abstracciones", esto significa que se puede intercambiar una implementación concreta por otra y el sistema debe continuar funcionando a la perfección.

Para el cumplimiento de DIP, se establece una relación de realización entre la clase de alto nivel *Accion* y la interfaz de *Accionable*.

La interfaz tiene los métodos de *divulgar*, *descargar*, y *brindarRecomendacion*, y la clase *Accion* implementa esta interfaz. Además, en lugar de escribir directamente la función requerida, en el método de *ejecutarAccion* de la clase *Accion*, se indica la clase *Accionable*, y, sin importar el tipo de función que se desee ejecutar, la clase *Accion* puede efectuarla sin ningún problema, esto también garantiza que no se viola el **Principio Abierto-Cerrado**.

En la **Figura 58** se muestra el diagrama que representa esta jerarquía.

Figura 58:
Diseño de la clase *Accion* y la interfaz *Accionable*.



Nota. Fuente: Elaboración propia.

➢ ***¿Se han definido correctamente las interfaces entre elementos del diseño?***

Para responder esta pregunta, se preparó un documento denominado: *Validación de Interfaces y Elementos del Diseño Web (UX & UI)*, para obtener la aceptación por parte del cliente. Este escrito valida y aprueba los ejemplos mostrados en el **Experimento de Mago de Oz**, así como en la propuesta de **Diseño de Interfaz**.

Para la creación de este documento, se evaluaron los tres principios definidos por Krug, S. (2018), los cuales son:

- No me hagas pensar.
- No importa cuantos clics, sino que no sean difíciles.
- Quitar la mitad de las palabras de la mitad.

De esta manera, se busca que el cliente comprenda la propuesta realizada, así como el objetivo de esta. La escala de Likert aplicada se encuentra en **Escala de Likert aplicada: Validación de Interfaces y Elementos del Diseño Web (UX & UI)**.

A continuación, en la **Tabla 26**, se muestra el resultado obtenido producto del documento aplicado al vicepresidente de ingeniería.

Tabla 27:
Puntaje de Aceptación: Validación de Interfaces y Elementos del Diseño Web (UX & UI)

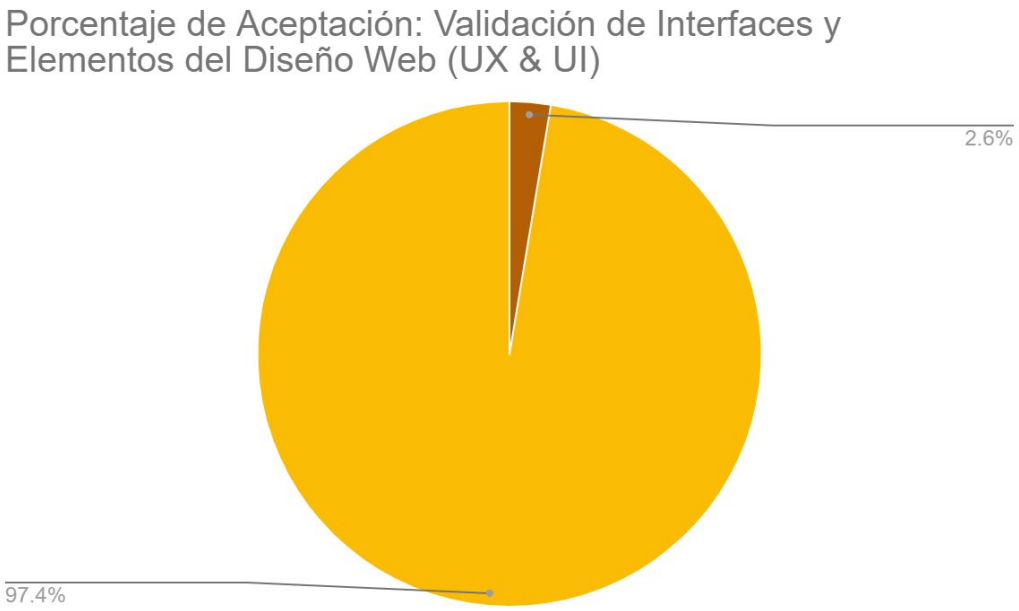
Pregunta	Opciones	Puntaje	Puntaje Obtenido
¿Cómo es tu nivel de satisfacción con el diseño de interfaz propuesto?	Muy satisfecho	5	5
	Satisfecho	4	
	Normal	3	
	Poco satisfecho	2	
	Nada satisfecho	1	
¿Cómo es tu nivel de satisfacción con la prueba de concepto mostrada en el Experimento de Mago de Oz?	Muy satisfecho	5	5
	Satisfecho	4	
	Normal	3	
	Poco satisfecho	2	
	Nada satisfecho	1	
¿Cómo valoras el diseño de interfaz propuesto?	Muy bueno	5	5
	Bueno	4	
	Indiferente	3	
	Malo	2	
	Muy malo	1	
¿Cómo valoras la prueba de concepto mostrada en el Experimento de Mago de Oz?	Muy buena	5	5
	Buena	4	

Pregunta	Opciones	Puntaje	Puntaje Obtenido
	Indiferente	3	
	Mala	2	
	Muy mala	1	
¿Qué tan clara fue la documentación suministrada?	Muy buena	5	5
	Buena	4	
	Indiferente	3	
	Mala	2	
	Muy mala	1	
¿Qué tan intuitiva fue la prueba de concepto mostrada en el Experimento de Mago de Oz?	Muy intuitiva	5	4
	Algo intuitiva	4	
	Indiferente	3	
	Poco intuitiva	2	
	Para nada intuitiva	1	
Con respecto a los clicks que se evidencian en la prueba de concepto mostrada en el Experimento de Mago de Oz, ¿Considera que estos fueron?	Excesivos	1	3
	Muchos	2	
	Justo los necesarios	3	
	Pocos	2	
	Nulos	1	
Con respecto al texto mostrado en la prueba de concepto mostrada en el Experimento de Mago de Oz, ¿Considera que estos fueron?	Excesivos	1	3
	Muchos	2	
	Justo los necesarios	3	
	Pocos	2	
	Nulos	1	
	Puntos Totales	36	35

Nota. Fuente: Elaboración propia.

En la Figura 59 se muestra un gráfico del porcentaje de aceptación.

Figura 59:
Porcentaje de Aceptación: Validación de Interfaces y Elementos del Diseño Web (UX & UI)



Nota. Fuente: Elaboración propia.

Con respecto a los datos anteriores, se observa que se obtuvo un 97.4% en la Validación de Interfaces y Elementos del Diseño Web (UX & UI), esto quiere decir que se obtiene la aceptación por parte del cliente, con respecto a los ejemplos mostrados en el **Experimento de Mago de Oz**, así como en la propuesta de **Diseño de Interfaz**.

- ***¿Es el diseño suficientemente detallado como para que sea posible implementarlo en el lenguaje de programación elegido?***

Se preparó un documento denominado: Validación del Diseño de Datos, Diseño Arquitectónico, de Componentes, y de Despliegue, para obtener la validación por parte de un ingeniero parte de la organización.

Este documento está creado con el objetivo de conseguir una validación y aprobación de las secciones de **Diseño de Software** y parte de la **Evaluación de Diseño.** De manera que el cliente comprenda la propuesta realizada, así como el objetivo de esta. La escala de Likert aplicada se encuentra en **Escala de Likert aplicada: Validación del Diseño de Datos, Diseño Arquitectónico, de Componentes, y de Despliegue.**

A continuación, en la **Tabla 27** se muestra el resultado obtenido producto del documento aplicado al ingeniero parte de la organización.

Tabla 28:
Puntaje de Aceptación: Validación del Diseño de Datos, Diseño Arquitectónico, de Componentes, y de Despliegue.

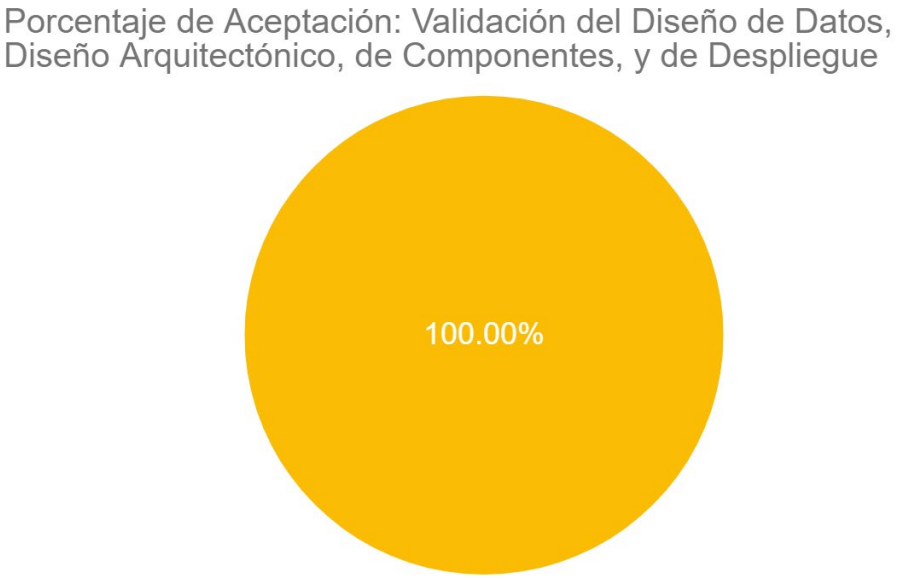
Pregunta	Opciones	Puntaje	Puntaje Obtenido
¿Cómo es tu nivel de satisfacción con el diseño de software propuesto?	Muy satisfecho	5	5
	Satisfecho	4	
	Normal	3	
	Poco satisfecho	2	
	Nada satisfecho	1	
¿Cómo es tu nivel de satisfacción con la evaluación de diseño realizada?	Muy satisfecho	5	5
	Satisfecho	4	
	Normal	3	
	Poco satisfecho	2	
	Nada satisfecho	1	
¿Cómo valoras el diseño de software propuesto?	Muy bueno	5	5
	Bueno	4	
	Indiferente	3	
	Malo	2	

Pregunta	Opciones	Puntaje	Puntaje Obtenido
	Muy malo	1	
¿Cómo valoras la evaluación de diseño realizada?	Muy buena	5	5
	Buena	4	
	Indiferente	3	
	Mala	2	
	Muy mala	1	
¿Qué tan clara fue la documentación suministrada?	Muy buena	5	5
	Buena	4	
	Indiferente	3	
	Mala	2	
	Muy mala	1	
¿Qué tan detallado es el diseño de software?	Muy detallado	5	5
	Algo detallado	4	
	Indiferente	3	
	Poco detallado	2	
	Para nada detallado	1	
	Puntos Totales	30	30

Nota. Fuente: Elaboración propia.

En la **Figura 60** se muestra un gráfico del porcentaje de aceptación.

Figura 60:
Porcentaje de Aceptación: Validación del Diseño de Datos, Diseño Arquitectónico, de Componentes, y de Despliegue



Nota. Fuente: Elaboración propia.

Con respecto a los datos anteriores, se obtuvo un 100% en la validación del diseño de datos, diseño arquitectónico, de componentes y de despliegue, esto quiere decir que se obtiene la aceptación por parte del cliente con respecto a los ejemplos mostrados en el **Diseño de *Software*** y parte de la **Evaluación de Diseño.**

5.1.2.1.3. Prueba de uso estadístico

Para esta sección, debido a la naturaleza del proyecto, no se cuenta con una implementación que compruebe directamente el desempeño o eficiencia del sistema puesto en marcha. Sin embargo, a continuación, se detallan las métricas de diseño presentes.

➢ ***Métricas de evaluación***

De acuerdo con Pressman (2005), las métricas de evaluación consisten en una serie de parámetros o sentencias que, al probarse en un sistema, dictan si este está desarrollado o diseñado de manera correcta. También, el mismo autor menciona que todo buen diseño de *software* debe determinar métricas para varios aspectos de la calidad de este, así como usar indicadores para guiar la forma en la que evoluciona el diseño.

- Métricas para Diseño Orientado a Objetos

Según Pressman (2005), en un diseño de *software* orientado a objetos se deben supervisar nueve características distintas y mensurables de un diseño OO:

➢ **Tamaño**: El tamaño se define en función de cuatro visiones:
  - Población: se mide al realizar un conteo estático de entidades OO, tales como clases u operaciones.
  - Volumen: son idénticas a las medidas de población, pero se recolectan de manera dinámica, en un instante de tiempo determinado.
  - Longitud: es una medida de una cadena de elementos de diseño interconectados (por ejemplo, la profundidad de un árbol de herencia es una medida de longitud).
  - Funcionalidad: proporcionan un indicio indirecto del valor entregado al cliente por una aplicación OO.

➢ **Complejidad:** en términos de características estructurales, es examinar cómo se relacionan mutuamente las clases de un diseño OO.

➢ **Acoplamiento:** las conexiones físicas entre elementos del diseño OO.

➢ **Suficiencia:** se pregunta: ¿qué propiedades debe poseer una clase para ser útil? Es decir, un componente de diseño (por ejemplo, una clase) es suficiente si posee las características requeridas.

➢ **Completitud**: es una implicación indirecta en el grado en el que puede reutilizarse la abstracción o el componente de diseño.

➢ **Cohesión:** es el conjunto de propiedades que posee parte del problema o dominio de diseño.

➢ **Primitivismo:** es el grado en el que una operación es atómica, esto quiere decir que la operación no puede construirse a partir de una secuencia de otras operaciones contenidas dentro de una clase.

➢ **Similitud**: El grado en el que dos o más clases son similares en términos de su estructura, función, comportamiento o propósito se indica mediante esta medida.

➢ **Volatilidad:** mide la probabilidad de que ocurra un cambio.

Para estas métricas, se compuso la **Tabla 28** la cual contempla las características mencionadas anteriormente, así como su aparición en el diseño propuesto.

Tabla 29:
Métricas para Diseño Orientado a Objetos

Módulo Central KS-NLG			Detalle
Tamaño			
Población	7	clases	4 Clases, 3 subclases
Volumen	4	clases	Se toma de manera dinámica que en cierto tiempo y espacio sólo se contempla un tipo de *Documento*
Longitud	2	niveles	• Nivel 1: *Documento* • Nivel 2: *Alerta, Resumen, Reporte*
Funcionalidad	12	funciones	Estas son las funciones públicas a las cuales los desarrolladores encargados de la integración tienen acceso en la clase *Peticion*
Complejidad	7	relaciones	• Una herencia por cada subtipo de *Documento (Alerta, Resumen, Reporte*) • Una realización entre *Accion* y *Accionable* • Una composición entre *Documento* y *Peticion* • Una composición entre *Entidad* y *Peticion* • Una agregación entre *Accion* y *Documento*
Acoplamiento	4	conexiones	Las conexiones son del Módulo Central KS-NLG con otros módulos del sistema
Suficiencia	Sí		Se abarcan todas las propiedades para que el sistema sea útil. Ver Tabla 19: Matriz de cobertura de los requerimientos
Completitud	2	clases	Las clases de *Accion* y *Documento* están diseñadas con un grado de abstracción el cual permite su reutilización en diferentes partes del sistema en caso de ser necesario.
Primitivismo	5	clases	Las clases *Accion, Alerta, Resumen, Reporte, y Entidad* son clases atómicas.
Similitud	3	clases	Las subclases de *Documento* (*Alerta, Resumen, Reporte*) son similares entre sí, sin embargo, su implementación es distinta.
Volatilidad	4	clases	Las clases *Accion*, y *Documento* son clases que, debido a su naturalidad de adaptación, pueden sufrir cambios a futuro.

Nota. Fuente: Elaboración propia.

- La Suite de Métricas CK

De acuerdo con Pressman (2005), la clase es la unidad fundamental de un sistema OO, es por esto que las medidas y métricas para una clase individual, la jerarquía de clase y las colaboraciones de clase serán invaluables cuando se requiera valorar la calidad del diseño OO. El autor propone diferentes métricas de diseño basadas en clases, las cinco principales son:

➢ **Métodos ponderados por clase (MPC):** Esta métrica permite conocer la cantidad de métodos y su respectiva complejidad, con el propósito de tomar estos datos como indicadores que permiten conocer el nivel de esfuerzo requerido para su implementación.

A continuación, en la **Tabla 29**, se presenta el MPC por cada clase del Módulo Central KS-NLG:
Tabla 30:
MPC del Módulo Central KS-NLG

1	2	3	4	5
Poco Complejo				Muy Complejo

Para la puntuación de complejidad, se utilizó como base la puntuación de tiquetes utilizada a lo interno de la empresa.

Clase	Peticion	Complejidad
Métodos	+ agregarEntidad(nombre: string): boolean	2
	+ agregarDocumento(tipo: string, idPadre: String): boolean	2
	+ leerEntidades(): array<Entidad>	4
	- leerDocumento(id: string): Documento	3
	+ mostrarResumen(): Resumen	2
	+ mostrarAlerta(): Alerta	2
	+ mostrarReporte(): Reporte	2
	+ eliminarReporte(id: string): boolean	4
	+ lanzarAlerta(): Alerta	4
	- eliminarDocumento(id: string): boolean	2
	+ eliminarEntidad(id: string): boolean	2
	+ eliminarResumen(id: string): boolean	2
	+ eliminarAlerta(id: string): boolean	2
	Complejidad de Clase	**2.538461538**

Clase	**Entidad**	Complejidad
Métodos	+ agregarValorEjemplo(valor: string): boolean	2
	+ eliminarValorEjemplo(valor: string): boolean	2
	Complejidad de Clase	**2**
Clase	**Documento**	Complejidad
Métodos	+ crearDocumento(idPadre: string, tipo: string): boolean	2
	+ editarDocumento(id: string, title: string, contenido: string): boolean	3
	+ generar(): String	2
	+ agregarAccion(accion: Accion): boolean	2
	+ editarAccion(idAccion: int): boolean	3
	+ eliminarAccion(): boolean	2
	Complejidad de Clase	**2.333333333**
Clase	**Alerta**	Complejidad
Métodos	+ generar(): String	4
	Complejidad de Clase	**4**
Clase	**Resumen**	Complejidad
Métodos	+ generar(): String	4
	- generarResumen(idTablero: string): void	2
	Complejidad de Clase	**3**
Clase	**Reporte**	Complejidad
Métodos	+ generar(): String	4
	- generarResumen(idTablero: string): void	2
	Complejidad de Clase	**3**
Clase	**Accion**	Complejidad
Métodos	+ ejecutarAccion(accion: Accionable): boolean	5
	Complejidad de Clase	**5**

Nota. Fuente: Elaboración propia.

- **Profundidad del árbol de herencia (PAH):** Esta métrica es "la máxima longitud desde el nodo hasta la raíz del árbol". Una jerarquía de clase profunda (PAH grande) también conduce a mayor complejidad de diseño. En el lado positivo, grandes valores de PAH implican que muchos métodos pueden reutilizarse.

En la **Tabla 30**, se observa el nivel de profundidad de la jerarquía de *Documento*:

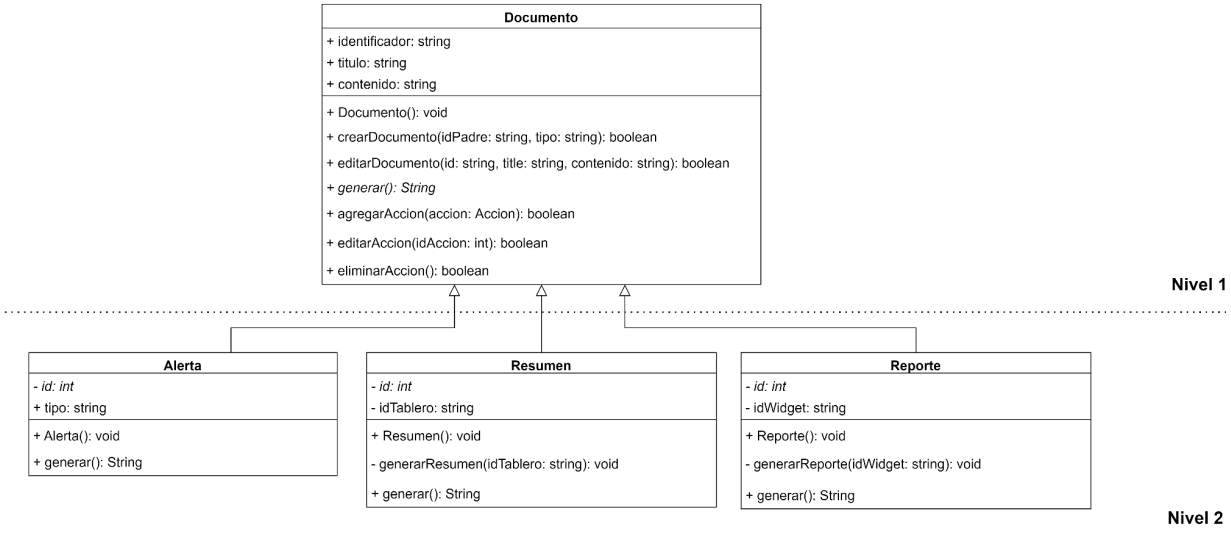
Tabla 31:
PAH de la Jerarquía *Documento*

Clase	Documento	**Nivel**
Subclases	Alerta	2
	Resumen	2
	Reporte	2
	Profundidad	2

Nota. Fuente: Elaboración propia.

En la **Figura 61,** se evidencian los niveles de la Jerarquía *Documento*.

Figura 61:
Niveles de la Jerarquía *Documento*.



Nota. Fuente: Elaboración propia.

- **Acoplamiento entre clases de objetos (ACO):** es el número de clases citadas dentro de una clase. Conforme el ACO aumenta, es probable que la reusabilidad baje; los valores de ACO para cada clase deben mantenerse tan bajos como sea razonable.

En la **Tabla 31**, se observa el ACO del Módulo Central KS-NLG:

Tabla 32:
ACO del Módulo Central KS-NLG

Clase	**Peticion**	**Total**
Clases Citadas	Entidad	1
	Documento	1
	Alerta	1
	Reporte	1
	Resumen	1
	Total Clases Citadas	**5**
Clase	**Entidad**	**Total**
Clases Citadas		0
	Total Clases Citadas	**0**
Clase	**Documento**	**Total**
Clases Citadas	Accion	1
	Total Clases Citadas	**1**
Clase	**Alerta**	**Total**
Clases Citadas		0
	Total Clases Citadas	**0**
Clase	**Resumen**	**Total**
Clases Citadas		0
	Total Clases Citadas	**0**
Clase	**Reporte**	**Total**
Clases Citadas		0
	Total Clases Citadas	**0**
Clase	**Accion**	**Total**
Clases Citadas		0
	Total Clases Citadas	**0**

Nota. Fuente: Elaboración propia.

- **Falta de cohesión en métodos (FCOM):** es el número de métodos que acceden a uno o más de los mismos atributos. Si ningún método accede a los mismos atributos, entonces la FCOM = 0; es deseable mantener alta la cohesión, es decir, mantener baja la FCOM.

En la **Tabla 32,** se observa el FCOM del Módulo Central KS-NLG:

Tabla 33:
FCOM del Módulo Central KS-NLG

Clase	Peticion	Atributos en común
Métodos	+ agregarEntidad(nombre: string): boolean	1
	- agregarDocumento(tipo: string): boolean	1
	+ leerEntidades(): array<Entidad>	1
	- leerDocumento(id: string): Documento	1
	+ mostrarResumen(): Resumen	1
	+ mostrarAlerta(): Alerta	1
	+ mostrarReporte(): Reporte	1
	+ eliminarReporte(id: string): boolean	1
	+ lanzarAlerta(): Alerta	1
	- eliminarDocumento(id: string): boolean	1
	+ eliminarEntidad(id: string): boolean	1
	+ eliminarResumen(id: string): boolean	1
	+ eliminarAlerta(id: string): boolean	1
	Total Atributos	**13**
Clase	**Entidad**	**Atributos en común**
Métodos	+ agregarValorEjemplo(valor: string): boolean	1
	+ eliminarValorEjemplo(valor: string): boolean	1
	Total Atributos	**2**

Clase	Documento	Atributos en común
Métodos	+ crearDocumento(idPadre: string, tipo: string): boolean	1
	+ editarDocumento(id: string, title: string, contenido: string): boolean	1
	+ generar(): String	1
	+ agregarAccion(accion: Accion): boolean	1
	+ editarAccion(idAccion: int): boolean	1
	+ eliminarAccion(): boolean	1
	Total Atributos	6
Clase	Alerta	Atributos en común
Métodos	+ generar(): String	0
	Total Atributos	0
Clase	Resumen	Atributos en común
Métodos	+ generar(): String	1
	- generarResumen(idTablero: string): void	1
	Total Atributos	2
Clase	Reporte	Atributos en común
Métodos	+ generar(): String	1
	- generarResumen(idTablero: string): void	1
	Total Atributos	2
Clase	Accion	Atributos en común
Métodos	+ ejecutarAccion(accion: Accionable): boolean	0
	Total Atributos	0

Nota. Fuente: Elaboración propia.

5.1.2.1.4. Certificación

A continuación, se presentan los documentos que permiten certificar la aceptación de la solución propuesta del **Diseño de Software** realizado.

- ***Documentación de validación***

Como se explicó en los apartados de **¿Se han definido correctamente las interfaces entre elementos del diseño?** y **¿Es el diseño suficientemente detallado como para que sea posible implementarlo en el lenguaje de programación elegido?** Los documentos aplicados al cliente respaldan la validación y aprobación del **Diseño de Software** y la **Evaluación de Diseño**, por lo tanto, se certifica el uso de estos.

- ***Documentación de gestión del cambio.***

Para la gestión del cambio, se preparó un documento llamado "Plan de Gestión de Cambio" (ver **Apéndice K**), el cual corresponde con los entregables de la organización.

- ***Plan de implementación del sistema propuesto.***

Para la gestión del cambio, se preparó un documento llamado "Plan de Implementación KS-NLG" (ver **Apéndice J**), el cual corresponde con los entregables de la organización.

5.1.3. Evaluación de Tecnologías

Actualmente existen múltiples tecnologías utilizadas para la generación de lenguaje natural, por esto, se presenta un análisis de estas con el objetivo de brindarle a Kleeen Software una perspectiva clara y recomendaciones sobre cuáles herramientas podrían utilizarse para la futura puesta en marcha de KS-NLG.

5.1.3.1. Herramientas

Para la realización de este análisis, se buscaron algunas herramientas, y se categorizaron de acuerdo con las **Fases de un sistema GLN**.

A continuación, se describen las herramientas encontradas:

5.1.3.1.1. Macro Planificación

Partiendo de lo definido anteriormente, existen dos tareas en la fase de **macro planificación**, las cuales son:

- Selección de contenido
- Estructuración del documento

Para esto, se identificó la herramienta SPUR, la cual realiza estas funciones:

- ***SPUR***
  - Descripción:

Esta herramienta es una biblioteca gratuita, y se encarga de la selección y organización o estructuración del contenido por comunicar, esto mediante la entrada de los atributos de las

opciones que se compararán, lo cual produce un plan de documentos con los atributos más importantes de tales opciones.

Este plan de contenido da lugar a realizaciones alternativas para recomendaciones, siguiendo un enfoque de abajo hacia arriba para la planificación de textos. Cada plan de contenido que se genere será un conjunto de afirmaciones (las cuales se deben comunicar al usuario), así como el conjunto de relaciones retóricas que se dan entre aquellas afirmaciones.

- Uso:

SPUR usa un modelo de usuario para dos aspectos de la selección de contenido:

1. Clasifica las opciones devueltas de una consulta de base de datos, y usa la clasificación para seleccionar un subconjunto de opciones para recomendar o comparar.
2. Determina una parte de atributos que se mencionan para cada opción, dependiendo el tamaño de esto y la configuración de un parámetro de concisión.

También, toma como entrada:

1. Una meta de estrategia de diálogo.
2. Un modelo de usuario.
3. Un parámetro de concisión.
4. Un conjunto de opciones devueltas por la base de datos que coinciden con las restricciones situacionales especificadas en la consulta del usuario.

Dadas las opciones y la configuración de concisión, la herramienta construye un plan de contenido específico para cada estrategia y modelo de usuario. El plan de contenido resultante sirve para filtrar la información que se presenta, de modo que solo se mencionen, contrasten y resalten las opciones y los atributos más relevantes.

5.1.3.1.2. Micro Planificación

Por otro lado, tal como se explicó previamente, existen tres tareas en la fase de **micro Planificación**, las cuales son:

- Agregación
- Lexicalización
- Generación de expresiones referenciales

Para esto, se identificó la herramienta SPaRKy, que realiza estas funciones:

- ***SPaRKy***

- Descripción:

Esta herramienta es una biblioteca gratuita, y está a cargo de la etapa de micro planificación basada en un sistema de plantillas. La entrada a este sistema es el plan de documento generado por SPUR, y la salida de esta herramienta es un plan de discurso con las afirmaciones que se representaron en la entrada.

- Uso:

Esta herramienta toma como entrada un plan de texto, que codifica las proposiciones que se expresarán en las hojas de un árbol, cuyos nodos internos están marcados con relaciones retóricas. SPaRKy luego reescribe el árbol del plan de texto usando una secuencia de reglas de lexicalización, combinación de cláusulas y expresión de referencia. También, las reglas de lexicalización obligatorias reescriben directamente las proposiciones específicas del dominio en un gráfico de dependencia léxico semántica de dominio general.

Después de la lexicalización, las reglas de combinación de cláusulas y expresión de referencia se aplican opcionalmente para reescribir la forma lógica en un conjunto de alternativas, entre las que idealmente hay una o más opciones que expresarán el contenido de manera concisa y fluida después de que se realice cada oración. Si no se aplica ninguna de las reglas de combinación de cláusulas y expresión de referencia, el texto se realizará como una secuencia de oraciones muy simples, con nombres propios utilizados para todas las referencias.

El proceso descrito se divide en dos fases o módulos:

- Generador de planes de oraciones.

En la primera fase, se genera un conjunto de árboles de planificación que contienen las relaciones (nodos internos) y las afirmaciones (nodos hoja) que aparecen en el texto final. En la segunda fase, esas afirmaciones se asignan a oraciones y luego se organizan.

- Ordenador de planes de oraciones.

Evalúa los diferentes planes generados basándose en un modelo basado en las calificaciones de los usuarios en la fase de capacitación. Sin embargo, utiliza algún conocimiento dependiente del dominio para tratar con los árboles de planificación, por lo que su adaptación a otros dominios también sería compleja.

5.1.3.1.3. Realización

A partir de lo definido en anteriormente, existen dos tareas en la fase de **realización**, las cuales son:

- Realización lingüística
- Realización de la estructura

Para esto, se identificaron las siguientes herramientas:

- ***RealPRO***

  - Descripción:

RealPRO está implementado en C++. Por lo tanto, es multiplataforma rápida y portátil. Se puede ejecutar como un servidor independiente, y tiene API de C++ y Java. Esta herramienta tiene licencia de forma gratuita para instituciones académicas, y se otorga con una tarifa para sitios comerciales, sin embargo, CoGenTex, Inc. ya no brinda soporte para la misma.

- Uso:

Como se explica en el Manual de Usuario según CoGenTex, Inc. (2000), el sistema recibe como entrada una estructura de dependencia sintáctica (en formato ASCII, HTML o RTF). Esta entrada se llama Estructura Sintáctica Profunda (DSyntS), la cual es una representación de un árbol desordenado con nodos y arcos etiquetados, se encuentra lexicalizado, lo que significa que los nodos están etiquetados con lexemas (palabras sin inflexiones) del idioma de destino. Se basa en una estructura de dependencia y no una estructura de frase, es decir, no hay nodos no terminales y todos los nodos están etiquetados.

- ***SimpleNLG***

  - Descripción:

Esta herramienta es una biblioteca gratuita, y se encuentra como una librería en el lenguaje java. Provee interfaces que ofrecen control directo sobre el proceso de realización, es decir, sobre la forma en que se construyen y combinan las frases, las operaciones morfológicas flexivas y la linealización. Está desarrollada con el objetivo de ayudar a escribir textos gramaticalmente correctos, define un conjunto de léxicos y tipos de oraciones, correspondientes a las principales categorías gramaticales, así como formas de combinarlos y establecer varios valores de características.

  - Uso:

Al construir una estructura sintáctica y linealizarla como texto con SimpleNLG, se llevan a cabo los siguientes pasos:

- Inicialización de los componentes básicos requeridos, con los elementos léxicos apropiados
- Usar las operaciones previstas en la API para establecer características de los componentes
- Combinar componentes en estructuras más grandes, nuevamente utilizando las operaciones provistas en API que se aplican a los componentes
- Pasar la estructura resultante al linealizador, que recorre la estructura del componentes, aplicando las inflexiones correctas y el ordenamiento lineal en función de las características, antes de devolver el texto realizado.

De acuerdo con Gatt, A., and Reiter, E., (2009), la sencillez de uso de SimpleNLG se refleja en su comunidad de usuarios. La distribución pública actualmente disponible ha sido utilizada por varios grupos con tres propósitos principales: (a) como un front-end para los sistemas NLG en proyectos donde la realización no es el enfoque principal de la investigación; (b) como un componente de lenguaje natural simple en interfaces de usuario para otros tipos de sistemas, por investigadores que no trabajan en NLG propiamente dicho; (c) como herramienta didáctica en cursos avanzados de grado y posgrado sobre Procesamiento del Lenguaje Natural.

5.1.3.1.4. Todas las Fases

También se identificaron herramientas las cuales realizan todas las fases requeridas, estas son:

- ***NaturalOWL***
  - Descripción:

Esta herramienta es una biblioteca gratuita, su enfoque es "datos a texto", debido a que con base en una ontología OWL genera texto con información referente a las entradas que reciba.

  - Uso:

A continuación se especifica el funcionamiento por cada una de las **Fases de un sistema GLN**.

    - Macro planificación:

Los textos de descripción que genera este sistema se estructuran a partir de un análisis basado en entradas de enciclopedia y posibles relaciones entre la información disponible. Esta se encuentra disponible y se divide en objetos de mensaje discretos.

Antes de construir la estructura, el sistema intenta resumir parte de esta información, por ejemplo, es posible combinar restricciones de cardinalidad sin perder datos de importancia.

La estructura de las descripciones consta de un pasaje introductorio, cuyo objetivo principal es brindar una rápida vista de la información sobre la clase, y una secuencia de secciones posteriores que presentan los datos restantes de acuerdo con las propiedades de esta. Esta etapa cierra con una presentación de las clases donde una de estas es disjunta. En general, cada elemento se realiza como una oración compleja.

    - Micro planificación:

Se realiza para derivar especificaciones de texto completo. La mayor parte de la estructuración que queda por hacer se realiza en los grupos de datos y está vinculada con operaciones de micro planificación como la agregación.

Dependiendo de los tipos de restricciones en los mensajes, se forman estructuras retóricas dentro de cada grupo. También permite que el usuario modifique el número máximo de frases por agregar.

    - Realización:

Se toma la salida de la micro planificación y la representa añadiendo los signos de puntuación y las letras mayúsculas necesarias. La entrada de esta última fase, como en la mayoría de los sistemas basados en esquemas, contiene el formato y el orden final en el que va a aparecer cada una de las palabras en el texto generado, por lo que no hace falta añadir nueva información, sino que es un proceso de transformación de los datos obtenidos al formato de salida.

5.1.3.1.5. Más herramientas para procesamiento de lenguaje natural

- ***ModelExplainer***

ModelExplainer es una herramienta innovadora con una interfaz de hipertexto/hiper gráfico desarrollada por CoGenTex, Inc. Esta genera automáticamente resúmenes textuales de diagramas

de modelado de datos orientados a objetos (OO) utilizando información de herramientas como *Rational Rose y Visio*. ModelExplainer permite al usuario centrarse en cualquier clase de un modelo y ver una descripción detallada de sus relaciones con otras clases.

Asimismo, esta herramienta permite la traducción instantánea de diagramas a un texto que ayuda a los analistas funcionales a depurar modelos de datos, aclarando aspectos de diseño como la cardinalidad y la dependencia, que se basan en notaciones complejas cuando se presentan gráficamente, además, puede generar texto de forma interactiva o en modo por lotes, con fines de documentación. Los textos facilitan la validación de requisitos con los usuarios finales (por ejemplo, expertos en el dominio), que pueden no estar familiarizados con los formalismos o las notaciones gráficas de modelos de datos particulares.

Los textos pueden incluir ejemplos generados automáticamente, que a menudo resaltan errores de diseño.

- ***CogentHelp***

CogentHelp es una herramienta de autoría para la ayuda de aplicaciones en línea, desarrollada por CoGenTex bajo un premio de Fase II de Investigación de Innovación en Pequeñas Empresas (SBIR) de Rome Laboratory (USAF).

CogentHelp automatiza parcialmente la producción de ayuda para interfaces gráficas de usuario (GUI), esto mediante la compilación de "fragmentos" de ayuda creados por humanos y asociados con cada componente de la GUI de una aplicación en un sistema de ayuda basado en HTML bien estructurado e interconectado. También genera dinámicamente temas de ayuda, de modo que reflejen el estado actual de la aplicación en tiempo de ejecución.

Esta herramienta admite applets de Java, lo que permite que las aplicaciones, así como su ayuda en línea, se implementen en intranets o en la web. También, verifica automáticamente la consistencia de los temas de ayuda orientados a la GUI con la propia aplicación, informando al autor de la ayuda sobre cualquier tema relacionado; brinda soporte de navegación a través de una tabla de contenido expandible; y permite integrar motores de búsqueda web.

- Gensim: Topic Modelling for Humans

Gensim es una biblioteca gratuita de Python, de código abierto, que representa documentos en forma de vectores semánticos de la manera más eficiente (computadora) y con el menor esfuerzo (humano) posible. Gensim está diseñado para procesar textos digitales sin estructura ("texto plano") utilizando algoritmos de aprendizaje automático no supervisados.

Los algoritmos de Gensim, como *Word2Vec*, *FastText, Latent Semantic Indexing (LSI, LSA, LsiModel), Latent Dirichlet Allocation (LDA, LdaModel),* etc., descubren automáticamente la estructura semántica de los documentos examinando patrones estadísticos de concurrencia dentro de un corpus de entrenamiento. Estos algoritmos no están supervisados, lo que significa que no se necesita intervención humana; solamente necesita un corpus de documentos de texto sin formato. Una vez que se encuentran estos patrones estadísticos, cualquier oración, frase y palabra se puede

expresar sucintamente en la nueva representación semántica y se puede consultar la similitud tópica con otras palabras o frases. Gensim tiene la licencia GNU LGPLv2.1 aprobada por OSI, esto quiere decir que es gratuito tanto para uso personal como comercial.

- ***Fasttext: NLP library for the learning of word embeddings and sentence classification created by Facebook's AI Research (FAIR) lab***

De acuerdo con Joulin, A. (2016), FastText es una biblioteca liviana, gratuita y de código abierto que permite a los usuarios aprender representaciones de texto y clasificadores de texto. Funciona en *hardware* genérico estándar. Posteriormente, los modelos se pueden reducir de tamaño para utilizarlos incluso en dispositivos móviles.

El modelo permite crear un algoritmo de aprendizaje no supervisado o de aprendizaje supervisado para obtener representaciones vectoriales de palabras. Facebook pone a disposición modelos pre entrenados para 294 idiomas.

5.1.3.2. Evaluación de Herramientas

Para la evaluación correspondiente, se establecieron dos pautas de aceptación que estas deben cumplir de acuerdo con el **Análisis de la Situación Actual**.

Las pautas definidas son:

- No sean de licenciamiento o membresía.
- Cumple al 100% con la **Arquitectura Integrada** propuesta.

A continuación, en la **Tabla 33** se presenta la evaluación de las herramientas de generación de lenguaje natural mencionadas en el apartado de **Herramientas**, con respecto a las pautas de aceptación definidas anteriormente.

Tabla 34:
Evaluación de Herramientas: Cumplimiento con las Pautas de Aceptación

Herramienta	¿Cumple con la Arquitectura Integrada?	Justificación	¿Es de licencia o membresía?
SPUR	No	La herramienta sólo realiza las tareas de **Macro planificación.**	No
SPaRKy	No	La herramienta sólo realiza las tareas de **Micro planificación.**	No
RealPro	No	La herramienta sólo realiza las tareas de **Realización.**	No
SimpleNLG	No	La herramienta sólo realiza las tareas de **Realización.**	No
NaturalOWL	Sí	La herramienta realiza las tareas de **Macro planificación**, **Micro planificación**, y **Realización.**	No

Nota. Fuente: Elaboración propia.

Con respecto a la **Tabla 33**, se evidencia que sólo una herramienta logra abarcar en un 100% las **Fases de un sistema GLN**, esto la vuelve en la única opción que cumple con la **Arquitectura Integrada** propuesta, por esto, la evaluación será con respecto a la herramienta **NaturalOWL**.

Con el propósito comprender de mejor manera la herramienta **NaturalOWL**, en la **Tabla 34** se definen tres posibles escenarios donde se puntuará el estado esta, estos escenarios se basaron en lo expuesto por Vicente, A. & Marsicano, J. (2017):

Tabla 35:
Evaluación de Herramientas: Escenarios de Aceptación

Escenario	Significado	Documentación Accesible		Soporte		Total
		Descripción	Puntaje	Descripción	Puntaje	
Oro	Escenario Ideal	Posee gran documentación accesible	5	Cuenta con soporte de parte del proveedor	5	10
Plata	Escenario Medio	Poca documentación accesible	3	Poco soporte de parte del proveedor	3	6
Bronce	Escenario Malo	Nula documentación accesible	0	Nulo soporte de parte del proveedor	1	0

Nota. Fuente: Elaboración propia.

Con base en la **Tabla 34**, se presenta la **Tabla 35** en la cual se muestra el escenario que mejor se adapta a la herramienta seleccionada.

Tabla 36:
Evaluación de Herramientas: Escenarios de **NaturalOWL.**

Herramienta	Documentación Accesible		Soporte		Total
	Descripción	Puntaje	Descripción	Puntaje	
NaturalOWL	Poca documentación accesible	3	Poco soporte de parte del proveedor	3	6

Nota. Fuente: Elaboración propia.

Con respecto a la **Tabla 35**, se observa cómo la herramienta se ubica en el escenario Plata, el cual es un escenario medio para el presente contexto. (ver referencia **[XXXI]**)

5.2. Estudio de Costos

Luego de realizar una reunión con el arquitecto de software de la empresa (ver **Minuta Reunión 6**), y al entrevistar al arquitecto de software de la empresa (ver **Apéndice H.d**) se obtiene como resultado el presente apartado donde se muestran los costos asociados al proyecto, tanto los relacionados con la presente propuesta como aquellos que corresponden a su futura implementación.

5.2.1. Costo de implementación

El costo de implementación comprende todos los gastos, inversiones o tiempo dedicado por parte del personal desde el momento en que se reconoce la necesidad y se comienza a trabajar en esta.

5.2.1.1. Costos de la Presente Propuesta

Para la presente propuesta de KS-NLG, se especifican los siguientes recursos y costos implicados:

5.2.1.1.1. Tiempo Estimado (Cronograma)

El presente proyecto se realizó en un periodo de dieciséis semanas laborables (jornada de cuarenta horas semanales), lo cual equivaldría a cuatro meses (ver **Anexo I**).

5.2.1.1.2. Recurso Humano

Debido a la naturaleza del proyecto, su desarrollo estuvo a cargo de una sola persona, sin tomar en cuenta las reuniones realizadas con otros miembros del equipo de Kleeen Software.

En la **Tabla 36** se desglosa el coste mensual por el salario del recurso.

Tabla 37:
Costos por recurso humano

Puesto	Salario Mensual	Meses Laborales	Total Salario
Estudiante a cargo del TFG	₡696,873.72	4	₡2,787,494.88
		Total Salarios	**₡2,787,494.88**

Nota. Fuente: Elaboración propia.

El salario anterior se tomó con base en lo estipulado por el Ministerio de Trabajo y Seguridad Social de Costa Rica en el 2022, el cual define que el salario mensual mínimo de un licenciado universitario es de ₡696,873.72. Por lo tanto, esta propuesta le costaría a la empresa ₡2,787,494.88 por los 4 meses laborados.

5.2.1.2. Costos de la Futura Implementación

Para la implementación de KS-NLG, se especifican los siguientes recursos y costos implicados:

5.2.1.2.1. Tiempo Estimado (Cronograma)

Para la implementación se dispone de dieciséis semanas laborables (jornada de cuarenta horas semanales), lo cual equivaldría a cuatro meses, como se observa en la **Tabla 37**.

Tabla 38:
Tiempo Estimado (Cronograma)

Recurso	Mes	Agosto				Septiembre				Octubre				Noviembre			
Recurso Humano	**Semana / Actividad**	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
AP	Analizar el sistema Kleeen Software IDE previo a la integración																
AP - Arquitecto de *Software*	Determinar los requerimientos de integración																
AP - Arquitecto de *Software*	Revalidar el diseño propuesto de KS-NLG de acuerdo con el análisis del sistema de Kleeen Software IDE																
AP	Establecer el alcance de la integración																
AP	Definir recursos necesarios																
AP	Valorar costos de implementación																
AP - Arquitecto de *Software* - Desarrollador de *Software*	Implementar KS-NLG																
AP - Arquitecto de *Software* - Desarrollador de *Software*	Diseñar el proceso de integración																
AP - Arquitecto de *Software* - Desarrollador de *Software*	Realizar la integración de software																
AP - Arquitecto de *Software* - Desarrollador de *Software*	Realizar comprobaciones de mantenimiento																
AP - Arquitecto de *Software* - Desarrollador de *Software*	Desarrollar pruebas de calidad, funcionalidad, y pruebas de usuario																
AP - Arquitecto de *Software* - Desarrollador de *Software*	Paso a Producción																

Nota. Fuente: Elaboración propia.

5.2.1.2.2. Recurso Humano

Con respecto a las tareas definidas en la **Tabla 37**, en la **Tabla 38** se proponen los siguientes recursos humanos para el proceso de integración:

Tabla 39:
Costos por recurso humano

Puesto	Salario Mensual	Meses Laborales	Total Salario
Administrador de Proyectos	₡941,001.00	4	₡3,764,004.00
Arquitecto de Software	₡1,186,751.00	3	₡3,560,253.00
Desarrollador de Software 1	₡887,600.00	2.5	₡2,219,000.00
		Total Salarios	**₡9,543,257.00**

Nota. Fuente: Elaboración propia.

Los salarios anteriores se tomaron con base en la página Glassdoor, la cual es una bolsa de trabajo basado en la web que permite a los empleadores y empleados conocer la situación laboral en las empresas así como obtener comparaciones de salarios.
De acuerdo con esta página, el salario para un administrador de proyectos equivale a ₡941,001.00, de un arquitecto de software a ₡1,186,751.00, y el de desarrollador de software ronda el ₡887,600.00. Por lo tanto, este desarrollo le costaría a la empresa ₡9,543,257.00 por los 4 meses laborados, en términos de recurso humano.

5.2.1.2.3. Tecnologías

La herramienta evaluadas en la **Evaluación de Herramientas,** es una recomendación por contemplar al momento de la implementación del sistema, y buscan el beneficio de la empresa. Estas recomendaciones son, en su mayoría, licencias y bibliotecas gratuitas, de manera que no se incurre en un costo de implementación extra al contemplar pago de licencias o membresías. Sin embargo, en caso de que la empresa desee implementar una tecnología diferente, o bien, desarrollarla internamente, se requiere un estudio de costos nuevo que evalúe los gastos y recursos necesarios para ese proceso.

5.2.1.2.4. Costos Ocultos

Como todo proyecto de software, el mantenimiento y la capacitación del personal a cargo son puntos clave para una buena funcionalidad y uso del este. Se deben contemplar estos costos, a continuación se presenta un aproximado de los costos por contemplar ante la posible futura implementación.

- ***Mantenimiento***

Para realizar la estimación de cuánto se puede incurrir en el mantenimiento del software a desarrollar, se utilizará el modelo constructivo de costos (COCOMO II).

COCOMO II es un proyecto de la Universidad del Sur de California (University of Southern California). El cual se utiliza para realizar estimaciones de proyectos de software.

Este utiliza tres submodelos, cada uno con mayor nivel de fidelidad que el anterior, estos modelos son:

- Composición de aplicaciones.
- Diseño inicial (Diseño temprano).
- Modelos post arquitectura.

Este modelo es usado para analizar el software a desarrollar y modificar, identificando componentes, subcomponentes, dividiéndolos y clasificándolos.

La estimación de costos por puntos de función se basa en la cantidad de funcionalidad involucrada en el software por desarrollar, esto es útil debido a que esta información se encuentra desde el diseño de los sistemas.

Para la estimación de esto costo, se utilizó la aplicación que la Universidad del Sur de California pone a disposición: http://softwarecost.org/tools/COCOMO/

Para hacer uso de esta aplicación, se define la cantidad de funciones aproximada que tendrá el software, y el lenguaje de programación utilizado.

Lenguaje de tercera generación se toma el escenario Nominal, es decir el escenario por defecto en cada una de las variables involucradas. Estas variables son:

- Controladores de escala de software.
  - Precedencia.
  - Flexibilidad de desarrollo.
  - Arquitectura / Resolución de Riesgos.
  - Cohesión de equipo.
  - Madurez del proceso.
- Controladores de costos del producto de software.
  - Confiabilidad de software requerida.
  - Tamaño de la base de datos.
  - Complejidad del producto.
  - Desarrollado para la reutilización.
  - Coincidencia de la documentación con las necesidades del ciclo de vida.
- Personal
  - Capacidad del analista.
  - Capacidad del programador.
  - Continuidad del Personal.
  - Experiencia de aplicación.
  - Experiencia de plataforma.
  - Experiencia con el lenguaje y el conjunto de herramientas.
- Plataforma
  - Limitación de tiempo.
  - Restricción de almacenamiento.
  - Volatilidad de la plataforma.
- Proyecto
  - Uso de herramientas de software.
  - Desarrollo multi sitio.
  - Cronograma de desarrollo requerido.

Para realizar el cálculo, se debe indicar costo de mano de obra de software de un colaborador por mes en dólares.

Luego de esto, el sistema calcula el costo de elaboración y construcción, he indica los siguientes datos:

- Esfuerzo: colaboradores requeridos.
- Horario: tiempo de desarrollo.
- Costo: es el costo mínimo sin incluir cargas sociales o atrasos en el tiempo.

Para concluir, se deben indicar las siguientes variables con el fin de calcular el costo de mantenimiento:

- Tamaño de cambio anual: por defecto es 50.
- Duración del mantenimiento (años): indica por cuanto tiempo se le dará mantenimiento al sistema.
- Comprensión del software (0% -50%).
- Desconocimiento (0-1): indica si se tiene o no conocimiento del sistema por mantener.

Y como resultado de este cálculo, el software muestra el costo del mantenimiento estimado, en forma de:

- Esfuerzo de mantenimiento anual: colaboradores requeridos.
- Costo de mantenimiento anual.
- Costo total de mantenimiento.

Para realizar este cálculo, se utilizaron los siguientes valores:

- Cantidad de funciones aproximada que tendrá el software:
- El lenguaje de programación utilizado: Lenguaje de tercera generación.
- Tamaño de cambio anual: 50.
- Duración del mantenimiento (años): 3
- Comprensión del software (0% -50%): 20%
- Desconocimiento (0-1): 1

A continuación, en la **Figura 62** el resultado al utilizar este sistema:

Figura 62:
Aplicación de COCOMO

Software Effort Distribution for RUP/MBASE (Person-Months)

Phase/Activity	Inception	Elaboration	Construction	Transition
Management	0.0	0.1	0.2	0.0
Environment/CM	0.0	0.1	0.1	0.0
Requirements	0.1	0.1	0.2	0.0
Design	0.0	0.2	0.3	0.0
Implementation	0.0	0.1	0.7	0.1
Assessment	0.0	0.1	0.5	0.1
Deployment	0.0	0.0	0.1	0.1

Maintenance
Annual Maintenance Effort = 0.1 Person-Months
Annual Maintenance Cost = $399
Total Maintenance Cost = $1199

Nota. Fuente: http://softwarecost.org/tools/COCOMO/

Esto quiere decir que, se requiere de un esfuerzo de mantenimiento anual igual a 0,1 meses-persona, con un costo de mantenimiento anual del $399, lo equivalente a ₡270,522.00, y un costo total de mantenimiento igual a $1199, lo equivalente a ₡812,922.00 por tres años.

- ***Capacitaciones***

Con respecto a las capacitaciones que pueden requerirse previo a la implementación del sistema, se seleccionaron una serie de cursos básicos necesarios para comprender los conceptos básicos de la generación de lenguaje natural.

Estos cursos se recomiendan como punto inicial para las capacitaciones requeridas, de manera que los colaboradores puedan comprender y manejar los conceptos y prácticas utilizadas en esta tecnología. Cabe recalcar que los proveedores en capacitaciones en temas de lenguaje natural son escasos, y de un alto costo, por esto, los cursos listados a continuación son recomendaciones.

Los programas y cursos seleccionados fueron elegidos por su puntuación registrada, o bien, por su cercanía con la empresa Kleeen Software.

A continuación, se listan los programas educativos seleccionados:

- Introducción al procesamiento de lenguaje natural (ver referencia **[XVIII]**)

Este curso es brindado por la Universidad Austral por medio de la plataforma de Coursera.

Este programa está enfocado en brindar los conocimientos introductorios sobre el procesamiento de lenguaje natural y las diversas tareas relacionadas al preprocesamiento de grandes volúmenes de texto.

Este curso es de nivel principiante, y posee una duración de 13 horas. El costo de este es de $49, en la modalidad en línea.

- Procesamiento de lenguaje natural utilizando aprendizaje automático (ver referencia **[XXXVI]**)

Este programa es parte de una serie de cursos brindados por la Escuela de Computación del Tecnológico de Costa Rica.

Este curso tiene una duración de 32 horas lectivas virtuales, y dentro de sus objetivos se encuentran:

- Objetivo General:
  - Al finalizar el curso, el estudiante contará con todas las bases para realizar proyectos que involucren actividades de NLP combinadas con aprendizaje automático.
- Objetivos Específicos:
  - Desarrollar un claro entendimiento del flujo de trabajo requerido por proyectos de NLP con aprendizaje automático.
  - Estar en capacidad de aplicar el paradigma de aprendizaje supervisado a problemas de NLP.
  - Conocer la problemática y las principales aplicaciones del aprendizaje automático al NLP.

De manera que la persona que realice este curso como capacitación ante la implementación de la presente propuesta, tendrá los conocimientos básicos sobre el procesamiento de lenguaje natural, por medio de la comprensión de algoritmos, redes neuronales profundas, búsqueda de similitudes, clasificación, reconocimiento de entidades, traducción y preguntas y respuestas.

El costo de este curso es de $350 más 2% de IVA.

- Cursos de Python para la Generación de Lenguaje Natural (ver referencia **[VII]**)

Este programa es brindado por NobleProg, el cual es un grupo internacional que imparte formaciones y consultorías, que durante los últimos diez años, ha formado a más de 40000 personas de más de 5000 empresas y organizaciones.

De acuerdo con la empresa Accenture Inc “En temas referentes a NLG, el equipo pudo aprender algo nuevo al final de cada tema. También hubo más actividades prácticas que diapositivas, lo cual fue bueno” *(Tomado de la página oficial de NobleProg)*

Los objetivos de este curso se enfocan en que los participantes comprendan cómo usar Python para producir texto en lenguaje natural de alta calidad construyendo su propio sistema NLG desde cero.

Este curso tiene como duración 21 horas, y al concluirlo, el estudiante podrá;

- Utilizar el NLG para generar automáticamente contenido para diversas industrias, desde periodismo, a bienes raíces, a informes meteorológicos y deportivos.
- Seleccionar y organizar el contenido fuente, planifique oraciones y prepare un sistema para la generación automática de contenido original

- Comprender la tubería NLG y aplicar las técnicas correctas en cada etapa, así como comprender la arquitectura de un sistema de generación de lenguaje natural (NLG)
- Entre otros.

Este curso tiene un costo de $2912 para la modalidad en línea.

Como resumen de esto, se tiene la **Tabla 39**:

Tabla 40:
Resumen de Costo de Capacitaciones

Nombre	Proveedor	Duración (horas)	Costo (Individual)	Costo por tres colaboradores capacitados
Introducción al procesamiento de lenguaje natural	Universidad Austral	13	$49	$147
Procesamiento de lenguaje natural utilizando aprendizaje automático	Escuela de Computación, del Tecnológico de Costa Rica	32	$350	$1,050
Cursos de Python para la Generación de Lenguaje Natural	NobleProg	21	$2912	$8,736
	Total	**66**	**$3,311**	**$9,933**

Nota. Fuente: Elaboración propia.

5.2.2. ROI

De acuerdo con Cuevas, C. (2001), el ROI o Retorno de Inversión *(Return On Investment)* es una razón financiera que se utiliza para calcular el rendimiento económico que se obtiene al realizar una inversión. Así pues, representa el resultado económico de las inversiones que ha realizado la empresa, ya sea que aporten un valor positivo o negativo.

Para realizar el cálculo de esta razón financiera, es necesario conocer dos valores principales: ganancia o beneficio esperado y el coste de inversión. Una vez teniendo estos dos datos, se utiliza la siguiente fórmula:

ROI % = (Ganancia o beneficio esperado – Coste de inversión) x 100/ Coste de Inversión

5.2.2.1. Beneficio esperado

Como parte de los **Beneficios esperados o aportes del Trabajo Final de Graduación** se encuentran beneficios como:

- Documentación necesaria para implementar el sistema de generación de lenguaje natural.
- Ingeniería de requerimientos y diseño de software para un sistema de generación de lenguaje natural.

Para obtener una estimación real de qué significan estos beneficios en términos financieros para la empresa, se realizó una reunión con el Arquitecto de Software de Kleeen Software (Ver **Minuta Reunión 6**). Producto de esta reunión, en la **Tabla 40** se realizó un estimado de cuánto invierte la empresa en realizar toda la documentación de un sistema implementado llamado *Playground*.

Tabla 41:
Recursos requeridos en un proyecto dentro de la empresa.

Proyecto: Playground	Encargado	Salario Mensual	Salario Semanal	Tiempo Estimado (semanas)	Total (Salario semanal x semanas)
Authoring					
Definir Arquitectura	Arquitecto de Software	₡1,486,751.00	₡345,756.05	12	₡4,149,072.56
Crear documentación asociada	Desarrollador de Software	₡1,039,386.00	₡241,717.67	3	₡725,153.02
Integration					
Definir Arquitectura	Arquitecto de Software	₡1,486,751.00	₡345,756.05	12	₡4,149,072.56
Crear documentación asociada	Desarrollador de Software	₡1,039,386.00	₡241,717.67	3	₡725,153.02
				Total Proyecto	**₡9,748,451.16**

Nota. Fuente: Elaboración propia.

Como resultado de la estimación anterior, se obtiene que el valor monetario del beneficio esperado corresponde a ₡9,748,451.16, esto debido a que, en un proyecto normal, estos son los costos en los que incurriría la empresa al realizar las labores de arquitectura, diseño y documentación de un sistema.

5.2.2.2. Coste de Inversión

En el apartado **Costos de la Presente Propuesta**, se calcularon los costos relacionados a la propuesta realizada en el presente proyecto, como resultado se encontró que el costo incurrido es de ₡9,543,257.00, el cual equivale a cuatro meses laborados, esto sin contar los posibles costos en mantenimiento y capacitaciones, debido a que estos son estimados de acuerdo a las recomendaciones brindadas, y no representan un dato estricto para la implementación.

5.2.2.3. Cálculo del ROI de la propuesta de diseño de software de alto nivel de un sistema de generación de lenguaje.

De acuerdo con el punto **Beneficio esperado** y **Coste de Inversión**, se obtienen las dos variables necesarias para realizar el cálculo de la razón financiera del retorno de inversión, estas son las siguientes:

Ganancia o beneficio esperado: ₡9,748,451.16
Coste de inversión: ₡9,543,257.00

Considerando que la fórmula del ROI es:

ROI % = (Ganancia o beneficio esperado – Coste de inversión) x 100/ Coste de Inversión

Entonces, al sustituir las variables dentro de la fórmula, se obtiene que:

ROI % = (9,748,451.16 – 9,543,257.00) x 100 / 9,543,257.00
ROI % = (205,194.16) x 100 / 9,543,257.00
ROI % = 20,519,416.28 / 9,543,257.00
ROI % = 2.15

Esto quiere decir que el retorno de inversión del proyecto es del 2.15%. Dado a que el resultado es mayor a 1, el departamento de desarrollo tendría utilidades.

5.2.2.4. Periodo de recuperación

El período de recuperación se calcula al tomar el costo de la inversión y dividirlo entre los ingresos esperados, o bien, el flujo de efectivo anual. Con los datos obtenidos se determina que:

Ganancia o beneficio esperado: ₡9,748,451.16
Coste de inversión: ₡9,543,257.00

Por lo tanto, al utilizar la siguiente fórmula:

Periodo de Recuperación = Coste de inversión / Flujo de efectivo anual

Se obtiene como resultado:

Periodo de Recuperación = 9,543,257.00 / 9,748,451.16
Periodo de Recuperación = 0.98 años; o su equivalente en semanas que serían 51 semanas.

5.2.3. Conclusiones del Capítulo

Producto de lo desarrollado en el capítulo **Propuesta de Solución**, se concluye lo siguiente:

- Como resultado de la propuesta realizada, se especifican diez diagramas para el diseño del *software*, de los cuales, tres corresponden al diseño de datos, cinco al diagrama de arquitectura, uno para el diagrama de componentes, y uno para el diagrama de despliegue. También se especifican tres diagramas para la evaluación del diseño.
- Tras la evaluación de tecnologías, se obtienen un total de nueve plataformas relacionadas a la generación y procesamiento de lenguaje natural, las cuales se incluyen con el propósito de ser contempladas por la empresa para la futura implementación de KS-NLG.
- Como parte de la evaluación de diseño y la matriz de requerimientos realizada, se concluye que el diseño de *software* presentado abarca en su totalidad los requerimientos funcionales identificados durante el análisis de la situación actual.
- Producto del plan de implementación y las reuniones llevadas a cabo con el equipo de Kleeen Software, se definió que, para la puesta en marcha de dicho plan, se requiere de un total de cuatro colaboradores distribuidos de la siguiente forma: un administrador de proyectos, un arquitecto de *software* y dos desarrolladores.
- Luego de identificar las posibles tareas mínimas para la implementación de KS-NLG, se estima un total de dieciséis semanas laborales requeridas.
- Tras el estudio de costos realizado en conjunto con colaboradores de la empresa, el costo total de la presente propuesta fue de ₡2,787,494.88, y el costo estimado para llevar a cabo la implementación del sistema es de ₡9,543,257.00, esto al evaluar el recurso humano requerido, así como el tiempo destinado.
- Con respecto al beneficio esperado, se pronostica que el retorno de inversión sea del 2.15%, teniendo este un periodo de recuperación de 51 semanas.

6. Conclusiones

A continuación, se presentan las conclusiones obtenidas a partir del análisis de resultados y la propuesta de solución realizada, estas se agrupan de acuerdo con el objetivo con el que se relacionen.

6.1. Objetivo Específico 1

Identificar la situación actual del sistema Kleeen Software IDE para la comprensión y descripción detallada de las necesidades y debilidades actuales de la empresa, esto con el fin de ser utilizadas en una futura integración a la propuesta de diseño del sistema de generación de lenguaje natural.

- De acuerdo con el análisis FODA realizado en el presente proyecto, se determinó que el escaso entendimiento del perfil de los clientes, la falta de diseño y arquitectura en la construcción de las extensiones del sistema y la centralización del conocimiento en pocas personas constituyen las principales debilidades de la organización.
- Por otro lado, el sistema Kleeen Software IDE cuenta con documentación de arquitectura y de procesos, también se apoya en tecnologías modernas como los servicios de AWS para la construcción de subsistemas modularizados como KAPI, el modelo conceptual y la canalización de las visualizaciones. Lo anterior es fundamental para plantear una estrategia que permita el diseño de aplicaciones o extensiones para el sistema actual.
- Durante el diagnóstico de la situación actual, también se determinó que existe un incremento en los competidores directos y cambios constantes en las tecnologías utilizadas, estos aspectos, junto con la difícil escalabilidad presente en los sistemas, representan amenazas que pueden evolucionar en riesgos para el negocio, por lo tanto, se deben tomar medidas para mitigar o afrontar las situaciones que se podrían manifestar.
- Al realizar diferentes reuniones con el vicepresidente de ingeniería de la empresa, se obtiene información que permite identificar la problemática actual, así como las percepciones manifestadas por el principal cliente de Kleeen Software; sin embargo, al aplicar una encuesta de Likert para validar estas opiniones, no se obtiene respuesta.

6.2. Objetivo Específico 2

Especificar los requerimientos funcionales y no funcionales para el diseño del sistema de generación de lenguaje natural, de acuerdo con la norma ISO/IEC/IEEE 29148, así como la organización y diseño de los componentes requeridos para el desarrollo del sistema.

- Como resultado de diferentes reuniones, se obtienen un total de diecinueve requerimientos por parte del cliente, de los cuales diez son requerimientos funcionales.
- Producto de la propuesta realizada, se especifican diez diagramas para el diseño del *software*, entre los cuales, tres corresponden al diseño de datos, cinco al diagrama de arquitectura, uno para el diagrama de componentes y uno para el diagrama de despliegue. También se especifican tres diagramas para la evaluación del diseño.

6.3. Objetivo Específico 3

Evaluar tecnologías o plataformas de generación de lenguaje natural con el objetivo de recomendar aquellas que ofrezcan mejores medidas de aceptación y evaluación de los requerimientos para la propuesta planteada.

- Tras el análisis de los diez requerimientos funcionales, se concluye que ocho de estos son de prioridad alta (*must*), uno es de prioridad media (*should*), y uno de prioridad baja (*could*).
- Tras la evaluación de tecnologías, se obtienen un total de nueve plataformas relacionadas a la generación y procesamiento de lenguaje natural, las cuales se incluyen con el propósito de ser contempladas por la empresa para la futura implementación de KS-NLG.
- Como parte de la evaluación de diseño y la matriz de requerimientos realizada, se concluye que el diseño de *software* presentado abarca en su totalidad los requerimientos funcionales identificados durante el análisis de la situación actual.

6.4. Objetivo Específico 4

Determinar el costo de implementación, así como el ROI relacionado con la propuesta de diseño de *software* por integrar en el sistema Kleeen Software IDE.

- Producto del plan de implementación y las reuniones llevadas a cabo con el equipo de Kleeen Software, se definió que, para la puesta en marcha de dicho plan, se requiere de un total de cuatro colaboradores distribuidos de la siguiente forma: un administrador de proyectos, un arquitecto de *software* y dos desarrolladores.
- Luego de identificar las posibles tareas mínimas para la implementación de KS-NLG, se estima un total de dieciséis semanas laborales requeridas.
- Tras el estudio de costos realizado en conjunto con colaboradores de la empresa, el costo total de la presente propuesta fue de ₡2,787,494.88, y el costo estimado para llevar a cabo la implementación del sistema es de ₡7,895,302.56, esto al evaluar el recurso humano requerido, así como el tiempo destinado.
- Con respecto al beneficio esperado, se pronostica que el retorno de inversión sea del 45.19%, teniendo este un periodo de recuperación de 36 semanas.

Con respecto al **Objetivo General** del proyecto:

Proponer un diseño de software de alto nivel de un sistema de generación de lenguaje natural, basado en las buenas prácticas de la industria, para la generación de documentación que brinde el contexto y entendimiento necesario en caso de una futura implementación e integración de este nuevo sistema con Kleeen Software IDE, con el fin de solventar la necesidad presentada, en la empresa Kleeen Software, durante el primer semestre del 2022.

Se concluye:

- Utilizando el modelo UML, se realizó un diseño de alto nivel para un sistema de generación de lenguaje natural, para esto fue necesario mapear las fortalezas y debilidades del sistema Kleeen Software IDE, realizar una recopilación y análisis de requerimientos, así como un experimento de mago de oz, de manera que la empresa cuenta con documentación suficiente para una futura implementación de KS-NLG y así solventar la necesidad presentada.

7. Recomendaciones

En este capítulo se describen las recomendaciones de acuerdo con el procedimiento metodológico propuesto, estas son de acuerdo con cada objetivo específico definido en el apartado de **Objetivos Específicos**.

7.1. Objetivo Específico 1

Identificar la situación actual del sistema Kleeen Software IDE para la comprensión y descripción detallada de las necesidades y debilidades actuales de la empresa, esto con el fin de ser utilizadas en una futura integración a la propuesta de diseño del sistema de generación de lenguaje natural.

- Implementar o contemplar un proceso que permita la gestión de conocimiento con respecto a la documentación de los sistemas internos de la empresa, por ejemplo, el Proceso de Gestionar el Conocimiento (BAI08) del COBIT 2019 (ver referencia **III**), de manera que se asegure que todos los colaboradores de la empresa que tengan contacto con los sistemas internos posean acceso a la documentación existente.
- Realizar revisiones periódicas sobre el estado de las tecnologías y competidores del mercado, esto con el propósito de obtener una perspectiva real y actualizada del área en la que se encuentra, de manera que la empresa pueda prepararse y afrontar los cambios tecnológicos o de negocio en el mercado.
- Repetir la evaluación con los clientes para actualizar la situación actual de satisfacción y así poseer datos correctos y reales sobre la percepción de los clientes.

7.2. Objetivo Específico 2

Especificar los requerimientos funcionales y no funcionales para el diseño del sistema de generación de lenguaje natural, de acuerdo con la norma ISO/IEC/IEEE 29148, así como la organización y diseño de los componentes requeridos para el desarrollo del sistema.

- Aplicar la norma ISO/IEC/IEEE 29148 en futuros proyectos, con el objetivo de documentar de manera clara y correcta los requerimientos y demás aspectos relacionados.
- Establecer un estándar para documentar los sistemas y procesos dentro de la empresa.

7.3. Objetivo Específico 3

Evaluar tecnologías o plataformas de generación de lenguaje natural con el objetivo de recomendar aquellas que ofrezcan mejores medidas de aceptación y evaluación de los requerimientos para la propuesta planteada.

- Evaluar el uso de las tecnologías contempladas en la propuesta realizada.
- Capacitar a los desarrolladores en temas de procesamiento de lenguaje natural con el propósito de comprender el funcionamiento de este campo y el de herramientas como ModelExplainer.

7.4. Objetivo Específico 4

Determinar el costo de implementación, así como el ROI relacionado con la propuesta de diseño de *software* por integrar en el sistema Kleeen Software IDE.

- Aplicar los costos externos que puedan generarse debido a actualizaciones o cambios de requerimientos, así como realizar evaluaciones de viabilidad y factibilidad de requerimientos nuevos que puedan solicitarse.
- Actualizar el estudio presentado en esta tesis con los montos reales a nivel empresarial para la valoración interna de la propuesta.
- Evaluar las funcionalidades adicionales que ofrecen las librerías contempladas en la propuesta, aunque estas se encuentren fuera de la lista de requerimientos inicial, esto para obtener una perspectiva amplia sobre las características nuevas que se pueden agregar al sistema, y así estimar el valor que estas le pueden aportar a la organización.
- Ampliar el estudio con herramientas de pago, y realizar una ponderación con el aporte real que brindan las tecnologías de uso libre.

Con respecto al **Objetivo General** del proyecto:

Proponer un diseño de software de alto nivel de un sistema de generación de lenguaje natural, basado en las buenas prácticas de la industria, para la generación de documentación que brinde el contexto y entendimiento necesario en caso de una futura implementación e integración de este nuevo sistema con Kleeen Software IDE, con el fin de solventar la necesidad presentada, en la empresa Kleeen Software, durante el primer semestre del 2022.

Se recomienda:

- Implementar la propuesta de solución realizada en el presente proyecto, debido a que esta cuenta con la aprobación de los requerimientos, un análisis de estos y un estudio de costos validado por la empresa.

8. Referencias

[I] Bargas-Avila, J.A., Hornbaeck, K. (2011). Old Wine in New Bottles or Novel Challenges? A Critical Analysis of Empirical Studies of User Experience. Conference on human factors in computing systems – proceedings. ACM. Vancouver, Canadá.

[II] Chala, G. J., Jordán, R. A., y Linares, D. L. (2007). GenLeNa: Sistema para la construcción de Aplicaciones de Generación de Lenguaje Natural. Sistemas & Telemática, 5(9), 45-60.

[III] Cockburn, A., (2001). Writing Effective Use-Cases, Addison-Wesley.

[IV] Corina, E., (2012) Prototipo, El Contexto Y La Ingeniería Del Software. Sistemas De Información Universidad Católica Andrés Bello, Caracas, Venezuela.

[V] Costa, M., y Silva, A. (2010). React-MDD - Reactive Traceability in Model-driven Development. ICEIS, (3), 483-488. Recuperado el 21 de noviembre 2021, de: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.1077.3217&rep=rep1&type=pdf

[VI] Cuesta, C. (2003). Arquitectura de software dinámica basada en reflexión. Alicante: Biblioteca Virtual Miguel de Cervantes. Recuperado el 23 de noviembre de 2021, de: http://www.cervantesvirtual.com/nd/ark:/59851/bmccr5p8. Edición digital a partir del texto original de la tesis doctoral.

[VII] Cursos de Python para la Generación de Lenguaje Natural. (2022, 5 abril). NobleProg. https://www.nobleprog.co.cr/cc/nlg?participants=4&how=private

[VIII] Danlos, L., y El Ghali, A. (2002). A complete integrated NLG system using AI and NLU tools. In Proceedings of COLING'02.

[IX] Doorn, J., (2007). Ingeniería de requisitos. Universidad Central de Buenos Aires.

[X] Ferré, X., y Sánchez, M. (n.d.). Desarrollo Orientado a Objetos con UML. Facultad de Informática – UPM

[XI] Fischer, G. (2001). User modeling in human–computer interaction. User modeling and user-adapted interaction, 11(1), 65-86.

[XII] Fraser, N. y Gilbert, G. (1991). Simulating Speech Systems. En Computer Speech and Language. University of Surrey, Surrey, UK.

[XIII] Giannetti, C. (2002). Estética digital. Sintopía del arte, la ciencia y la tecnología. Barcelona: Associació de Cultura Contemporània l'Angelot.

[XIV] Heineman, G., y Councill, W. (2001). Component-Based Software Engineering: Putting the Pieces Together (paperback) (1st ed.). Addison-Wesley Professional.

[XV] Hevner, A. R. y H. D. Mills, (1993). Box Structure Methods for System Development with Objects. IBM Systems Journal.

[XVI] Institute of Electrical and Electronics Engineers, Comisión Electrotécnica Internacional, Organización Internacional de Normalización. (2018). ISO/IEC/IEEE 29148. IEEE. Recuperado el 23 de noviembre 2021, de: https://ecs.wgtn.ac.nz/foswiki/pub/Courses/ENGR301_2021T1/Assessments/29148-2018-requirements.pdf

[XVII] Instituto Tecnológico de Aguascalientes. (2015). Árbol de Problemas del Análisis al Diseño y Desarrollo de Productos [Ebook]. Recuperado el 23 de noviembre 2021, de: https://www.redalyc.org/pdf/944/94443423006.pdf

[XVIII] Introducción al procesamiento de lenguaje natural. (2022) Coursera. https://www.coursera.org/learn/introduccion-al-procesamiento-de-lenguaje-natural

[XIX] Plataforma de Navegación en Línea (OBP). ISO 9241-210:2010 (en), Ergonomics of human-system interaction — Part 210: Humancentred design for interactive systems. Disponible en: https://www.iso.org/obp/ui/es/#iso:std:iso:9241:-210:ed-1:v1:en

[XX] ISO/IEC/IEEE. 2011. Systems and Software Engineering - Requirements Engineering. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/ Institute of Electrical and Electronics Engineers (IEEE), ISO/IEC/IEEE 29148.

[XXI] Jaiswal, M., (2019). Software Architecture and Software Design. SSRN Electronic Journal, 6(11), pp. 2452-2454. https://doi.org/10.2139/SSRN.3772387

[XXII] Krause, R. (2021). Maintain Consistency and Adhere to Standards (Usability Heuristic #4). Nielsen Norman Group. Recuperado 25 de abril de 2022, de https://www.nngroup.com/articles/consistency-and-standards/

[XXIII] Krug, S. (2018). Don't make me think!: Web & Mobile Usability: Das intuitive Web. MITP-Verlags GmbH & Co. KG.

[XXIV] Lant, M. (2019). How to Easily Prioritize Your Agile Stories - Michael Lant. Michael Lant - Software Archictecture, Development, Agile Methods and the Intersection of People Process and Technology. Recuperado de: https://michaellant.com/2010/05/21/how-to-easily-prioritize-your-agile-stories/

[XXV] Laubheimer, P. (2016, 6 noviembre). Cards: UI-Component Definition. Nielsen Norman Group. Recuperado 7 de abril de 2022, de https://www.nngroup.com/articles/cards-component/

[XXVI] Maria, L. S. (2020, 10 junio). Diseño web UX para crear el botón de llamada a la acción perfecto. Staff Creativa. Recuperado 7 de abril de 2022, de https://www.staffcreativa.pe/blog/diseno-web-ux-para-boton-de-llamada-a-la-accion-perfecto/

[XXVII] Martínez, R., y Fernández, A. (2008). Árbol de Problema y áreas de intervención. México: CEPAL

[XXVIII] Medina, A., (2019), Requisitos para la integración de sistemas de información. Estudio de caso: Facultad de Comunicación de la Universidad de La Habana. Alcance, 8(19), 75-87.

[XXIX] Meng, M., y Steinhardt, S., y Schubert, A., (2018). Application Programming Interface Documentation: What Do Software Developers Want?. Journal of Technical Writing and Communication 48, 295–330.

[XXX] Monje, C. (2011). Metodología de la Investigación Cuantitativa y Cualitativa Guía Didáctica. Universidad Surcolombiana.

[XXXI] nlgen in. (2009, 27 marzo). Launchpad. https://launchpad.net/nlgen

[XXXII] Nielsen, J. (1994, April). Usability inspection methods. In Conference companion on Human factors in computing systems (pp. 413-414).

[XXXIII] Nuseibeh B., Easterbrook S. (2000). Requirements Engineering: A Roadmap. ICSE2000. Limerick, Irlanda.

[XXXIV] Patternfly. (s. f.). Toast Notifications. Recuperado de: https://www.patternfly.org/v3/pattern-library/communication/toast-notifications/index.html#overview

[XXXV] Paul, K., y Kimmel, P. (2011). Manual de UML. McGraw-Hill Education.

[XXXVI] Ponce, H., (2007). La matriz foda: alternativa de diagnóstico y determinación de estrategias de intervención en diversas organizaciones. Enseñanza e Investigación en Psicología. Consejo Nacional para la Enseñanza en Investigación en Psicología A.C. Xalapa, México

[XXXVII] Programas De Educación Continua - Escuela De Computación (2022). Fundación Tecnológica De Costa Rica (Fundatec). https://www.tec.ac.cr/fundatec/programa-certificacion-educacion-continua-ti

[XXXVIII] Rivas, H. (2005). Integración De Sistemas Heredados Utilizando Web Services. Escuela Académica Profesional de Ingeniería Informática. Lima, Perú.

[XXXIX] Rivas, L. (2015). ¿Cómo hacer una tesis? (3ra ed.). IPNEditors.DOI: 10.13140/RG.2.1.3446.6644. México

[XL] Ruiz Calle, J. (2018). Modelización de Web Components reutilizables para simplificar el proceso de paso de información en aplicaciones web.

[XLI] Sabino, C. (2014). El proceso de investigación. Editorial Episteme.

[XLII] Sommerville, I., Galipienso, M. I. A., y Martinez, A. B. (2005). Ingeniería del Software. Pearson.

[XLIII] Ulate, I. y Vargas, E. (2016). Metodología para elaborar una tesis. Recuperado el 11 de marzo de 2022, de: https://ebooks.uned.ac.cr/pdfreader/metodologa-para-elaborar-una-tesis50072875

[XLIV] University of Tg-Jiu. (2017). Study To Determine a New Model of The Ishikawa Diagram for Quality Improvement [Ebook]. Recuperado el 23 de noviembre 2021, de: https://www.utgjiu.ro/rev_mec/mecanica/pdf/2017-01/39_liliana%20luca,%20minodora%20pasare,%20alin%20stancioiu%20-%20study%20to%20determine%20a%20new%20model%20of%20the%20ishikawa%20diagram%20for%20quality%20improvement.pdf

[XLV] Vicente, A. & Marsicano, J. (2017). Generación de Escenarios de Calidad a partir de Requerimientos Textuales.Universidad Nacional del Centro de la Provincia de B uenos Aires. Tandil, Argentina.

[XLVI] Vicente, M. (2015). La generación de lenguaje natural: análisis del estado actual. Computación y Sistemas. Computación y Sistemas, 19(4), 721-756.

[XLVII] Westfall, L. (2014). What, Why, Who, When, and How of Software Requirements. En Handbook of Research on Emerging Advancements and Technologies in Software Engineering (1-18). Pennsylvania: IGI Global.

[XLVIII] Wiegers K. y Beatty J., (2013). The essential software requirement. En Software Requirements (pp 36-55). Redmond, Washington: Christian Holdener, S4Carlisle Publishing Services.

[XLIX] Zapata, C., y Carmona, N,, (2006). El Experimento Mago De Oz Y Sus Aplicaciones: Una Mirada Retrospectiva. Grupo De Investigación En Ingeniería De Software. Facultad De Minas, Universidad Nacional De Colombia-Medellín.

9. Apéndices

A continuación, se adjuntan los apéndices del documento:

Apéndice A. Plantilla de minutas

Tabla 42:
Plantilla minuta

KLEEEN SOFTWARE

Minuta de la reunión

Reunión #:	**Fecha de realización:**	**Hora de inicio:**	**Hora de finalización:**
(Identificador de reunión).	*(Fecha cuando se realizó la reunión).*	*(Hora de inicio de la reunión).*	*(Hora de finalización de la reunión).*

Asistentes a la reunión

Nombre	**Puesto**
(Nombre de los participantes).	*(Nombre de los puestos o roles de los participantes).*

Agenda

(Puntos a tratar en la reunión).

Acuerdos

(Listar todos los acuerdos conversados en la reunión).

Aspectos Pendientes

(Puntos de la agenda pendientes de tratar, y que serán reprogramados para próximas reuniones).

Notas de la reunión

(Todas las notas, comentarios o asuntos de discusión relevantes para el presente proyecto, abarcados durante la reunión).

Firmas			
Elaborado por	XXXXXXXX (Estudiante de TFG)	**Revisado por**	XXXXXXXX (Prof. Tutor)

Nota. Fuente: Elaboración propia.

Apéndice B. Plantilla de solicitud de cambio

Tabla 43:
Solicitud de cambio

KLEEEN SOFTWARE — Solicitud de cambios				
Reunión #: *(Identificador de reunión).*	**Fecha de realización:** *(Fecha cuando se realizó la reunión).*	**Solicitante:** *(Nombre del solicitante del cambio).*		
Detalle del cambio				
Descripción del cambio		*(Detallar los cambios).*		
Causa del cambio		*(Explicar la causa del cambio).*		
Tipo del cambio				
Costo ( )	Tiempo **( )**	Calidad ( )	Procedimientos **( )**	
Recurso ( )	Alcance **( )**	Documentación **( )**	Otro ( )	
Impactos del cambio				
Nivel de impacto		**Bajo ( )**	**Medio ( )**	**Alto ( )**
Justificación del nivel de impacto		*(Justifica el nivel del cambio).*		
Implicaciones del cambio				
1.	*(Listar todas las implicaciones del cambio).*			
Aprobación del cambio				
Fecha de aprobación:	*(Fecha en la que se aprueba el cambio).*			
Firma del Estudiante.	*Firma del Profesor Tutor.*	*Firma del representante de la empresa.*		
Firma del Estudiante.	**Firma del Profesor Tutor.**	**Firma del representante de la empresa.**		

Nota. Fuente: Elaboración propia.

Apéndice C. Escala de Likert: Satisfacción del Usuario

Tabla 44:
Escala de Likert: Satisfacción del Usuario

Pregunta	Opciones	Marque con X
¿Qué objetivo comercial le permite KS lograr al usar nuestro producto?	Muy satisfecho	
	Satisfecho	
	Normal	
	Poco satisfecho	
	Nada satisfecho	
¿En qué medida KS satisface las necesidades de su aplicación?	Muy satisfecho	
	Satisfecho	
	Normal	
	Poco satisfecho	
	Nada satisfecho	
¿Crees que la forma en que se presenta la información en nuestros cuadros de mando es correcta?	Muy satisfecho	
	Satisfecho	
	Normal	
	Poco satisfecho	
	Nada satisfecho	
¿Con qué frecuencia se siente estresado debido a problemas con nuestra aplicación o servicio de soporte?	Muy satisfecho	
	Satisfecho	
	Normal	
	Poco satisfecho	
	Nada satisfecho	

Nota. Fuente: Elaboración propia.

Apéndice D. Estructura de Entrevista para la Identificación del Problema.

Figura 63:
Estructura de Entrevista para la Identificación del Problema

KLEEEN SOFTWARE

Entrevista

Entrevista para la Identificación del Problema y Situación Actual.

Sujeto Entrevistado	Vicepresidente de Ingeniería
Objetivo de la Entrevista	Determinar el problema y situación actual.

Estructura

1. ¿Cómo considera que es la situación actual de la empresa?
2. ¿Cuáles considera que son las fortalezas o debilidades que poseen en la empresa?
3. ¿Cuáles considera que son las debilidades oportunidades o amenazas que percibe externas, pero que afectan a la empresa?
4. ¿Posee la empresa actual alguna dolencia con algún servicio?
5. ¿Sabe a qué se debe este problema?
6. ¿Qué temor posee si este problema continúa o se llegara a agravar?

Notas extra

(Todas las notas, comentarios o asuntos de discusión relevantes para el presente proyecto, abordados durante la entrevista).

Firmas			
Elaborado por		**Sujeto Entrevistado**	XXXXXXXX

Nota. Fuente: Elaboración propia.

Apéndice E. Estructura de Entrevista para la Recopilación y Análisis de Requerimientos

Figura 64:
Estructura de Entrevista para la Recopilación y Análisis de Requerimientos

KLEEEN SOFTWARE

Entrevista

Entrevista para la recopilación y análisis de los requerimientos

Sujeto Entrevistado	Vicepresidente de Ingeniería
Objetivo de la Entrevista	Recopilar y analizar los requerimientos.

Estructura

1. ¿Qué problemas debe solucionar el Sistema? ¿Cómo se resuelven ahora? ¿Cómo los resolvería?
2. ¿Qué debe tener el sistema deseado?
3. ¿Quienes serán los usuarios del sistema?
4. ¿Cómo desea que sea este sistema?
5. ¿Quienes serán los usuarios del sistema?
6. ¿Hay aplicaciones/sistemas relevantes a tener en cuenta dentro de este sistema?
7. Con respecto a estas necesidades y deseos expresados, ¿Podría priorizar cada una del 1 al 5?

Notas extra

(Todas las notas, comentarios o asuntos de discusión relevantes para el presente proyecto, abordados durante la entrevista).

Firmas			
Elaborado por		**Sujeto Entrevistado**	XXXXXXXX

Nota. Fuente: Elaboración propia.

Apéndice F. Estructura de Entrevista para la Comprensión de la Situación Actual del Sistema Kleeen Software IDE

Figura 65:
Estructura de Entrevista para la Comprensión de la Situación Actual del Sistema Kleeen Software IDE

KLEEEN SOFTWARE

Entrevista

Entrevista para la comprensión de la situación actual del sistema Kleeen Software IDE

Sujeto Entrevistado	Arquitecto de Software
Objetivo de la Entrevista	Comprender la situación actual del sistema Kleeen Software IDE

Estructura

1. ¿La empresa posee documentación interna del sistema Kleeen Software IDE?
2. ¿El sistema posee módulos o sistemas externos? ¿Cómo son estos sistemas?
3. ¿Cuál es la arquitectura de software utilizada?
4. ¿Utilizan herramientas en la nube?
5. ¿Devops, flujos de AWS y algún otro sistema?
6. ¿Tienen diagramas de actores?
7. ¿Tienen diagramas de flujos?

Notas extra

(Todas las notas, comentarios o asuntos de discusión relevantes para el presente proyecto, abordados durante la entrevista).

Firmas			
Elaborado por		**Sujeto Entrevistado**	XXXXXXXX

Nota. Fuente: Elaboración propia.

Apéndice G. Estructura de Entrevista para Conocer el Tiempo y Costo Relacionado a la Documentación de un Proyecto Dentro de la Empresa.

Figura 66:
Estructura de Entrevista para Conocer el Tiempo y Costo Relacionado a la Documentación de un Proyecto Dentro de la Empresa

KLEEEN SOFTWARE

Entrevista

Entrevista para la conocer el tiempo y costo relacionado a la documentación de un proyecto dentro de la empresa

Sujeto Entrevistado	Arquitecto de Software y Vicepresidente de Ingeniería
Objetivo de la Entrevista	Conocer el tiempo y costo relacionado a la documentación de un proyecto dentro de la empresa

Estructura

1. ¿Existe algún proyecto que se está documentando en la empresa?
2. ¿De qué trata este proyecto?
3. ¿Cuál es la documentación relacionada a este proyecto?
4. ¿Cuál es el tiempo aproximado para realizar la documentación relacionada?
5. ¿Qué recursos están comprometidos o siendo utilizados en este proceso?
6. ¿Cuál es el costo monetario aproximado de este proceso?

Notas extra

(Todas las notas, comentarios o asuntos de discusión relevantes para el presente proyecto, abordados durante la entrevista).

Firmas			
Elaborado por		**Sujeto Entrevistado**	XXXXXXXX

Nota. Fuente: Elaboración propia.

Apéndice H. Entrevistas realizadas

a. Entrevista para la Identificación del Problema.

Figura 67:
Estructura de Entrevista para la Identificación del Problema

KLEEEN SOFTWARE

Entrevista

Entrevista para la Identificación del Problema y Situación Actual.

Sujeto Entrevistado	Vicepresidente de Ingeniería
Objetivo de la Entrevista	Determinar el problema y situación actual.

Estructura

1. **¿Cómo considera que es la situación actual de la empresa?**

Actualmente, la empresa se dedica al diseño y desarrollo de un sistema que permite la automatización para la creación de aplicaciones de gestión de información, este sistema busca agilizar el proceso de creación de aplicaciones empresariales.

2. **¿Cuáles considera que son las fortalezas o debilidades que poseen en la empresa?**

Una fortaleza de la empresa es que cuenta con mucha documentación de las arquitecturas, así como de los flujos de procesos. También intentamos mantener actualizadas nuestras tecnologías. Otra, es que el sistema que desarrollamos Kleeen Software IDE, está compuesto por varios otros sistemas que también son *inhouse*.

Como debilidad, diría que no hemos dedicado mucho tiempo a la investigación del pensamiento del usuario final, o cuál es el perfil de este. Tampoco tenemos establecido algún diseño o alguna arquitectura que permita crear aplicaciones tipo "extensiones" para el sistema Kleeen Software IDE.

3. **¿Cuáles considera que son las oportunidades o amenazas que percibe externas, pero que afectan a la empresa?**

Como oportunidades, diría que nos mantenemos al día de los competidores directos que tenemos, entonces conocemos cuando nuevas tecnologías surgen o aplicaciones competencias. Pero a su vez, también es una debilidad, porque los competidores crecen cada día, y cambian el paradigma de tecnologías que tenemos.

4. **¿Posee la empresa actual alguna dolencia con algún servicio?**

Algo que hemos notado, es que nuestro único cliente del sistema, en ocasiones nos ha manifestado que no comprende los datos generados, entonces esto nos crea incertidumbre y temor en que el cliente quiera seguir con nosotros si esto no se soluciona.

5. **¿Sabe a qué se debe este problema?**

Creemos que es porque actualmente no contamos con ningún sistema que verifique o permita que los datos sean comprensibles para los usuarios.

6. **¿Qué temor posee si este problema continúa o se llegara a agravar?**

Como mencioné antes, tememos que el cliente decida dejar de utilizar nuestros servicios,

Notas extra

(Todas las notas, comentarios o asuntos de discusión relevantes para el presente proyecto, abordados durante la entrevista).

Firmas

Elaborado por		**Sujeto Entrevistado**	

Nota. Fuente: Elaboración propia.

b. Entrevista para la Recopilación y Análisis de Requerimientos

Figura 68:
Entrevista para la Recopilación y Análisis de Requerimientos

KLEEEN SOFTWARE

Entrevista

Entrevista para la recopilación y análisis de los requerimientos

Sujeto Entrevistado	Vicepresidente de Ingeniería
Objetivo de la Entrevista	Recopilar y analizar los requerimientos.

Estructura

1. ¿Qué problemas debe solucionar el Sistema?

Básicamente el lograr mostrar la información generada de manera que el usuario final pueda entenderla, esta información puede ser mostrada por medio de reportes, resúmenes, o alertas que se puedan crear, editar o borrar. También algo que permita mandar estos documentos por slack u otra red social, así como también descargarlos.

2. ¿Qué debe tener el sistema deseado?

Debe poder conectarse con el sistema Kleeen Software IDE.

3. ¿Quienes serán los usuarios del sistema?

Los actuales usuarios de Kleeen Software IDE

4. ¿Hay aplicaciones/sistemas relevantes a tener en cuenta dentro de este sistema?

Sí, Kleeen Software IDE

5. Con respecto a estas necesidades y deseos expresados, ¿Podría priorizar cada una del 1 al 5?

La creación de los documentos debe ser de prioridad alta, entonces sería un 5,

Notas extra

(Todas las notas, comentarios o asuntos de discusión relevantes para el presente proyecto, abordados durante la entrevista).

Firmas			
Elaborado por		**Sujeto Entrevistado**	

Nota. Fuente: Elaboración propia.

c. Entrevista para la Comprensión de la Situación Actual del Sistema Kleeen Software IDE

Figura 69:
Entrevista para la Comprensión de la Situación Actual del Sistema Kleeen Software IDE

KLEEEN SOFTWARE

Entrevista

Entrevista para la comprensión de la situación actual del sistema Kleeen Software IDE

Sujeto Entrevistado	Arquitecto de Software
Objetivo de la Entrevista	Comprender la situación actual del sistema Kleeen Software IDE

Estructura

1. ¿La empresa posee documentación interna del sistema Kleeen Software IDE?

Sí, existen documentos en el confluence de la empresa.

2. ¿El sistema posee módulos o sistemas externos? ¿Cómo son estos sistemas?

Kleeen Software IDE utiliza sistemas como:

- KAPI
- Conceptual Model
- Viz Pipeline

3. ¿Cuál es la arquitectura de software utilizada?

El sistema utiliza diferentes servicios en la nube de AWS, algunos son los acceso a lambdas, sistemas de registro y pool de usuarios como Cognito, AppSync, entre otras.

4. ¿Utilizan herramientas en la nube?

La nube de AWS

5. ¿Devops, flujos de AWS y algún otro sistema?

Por ahora, creo que sólo AWS

6. ¿Tienen diagramas de actores?

Sí, tenemos unos diagramas de sistemas actores.

7. ¿Tienen diagramas de flujos?

Sí, tenemos unos diagramas de los flujos del pipeline de visualizaciones

Notas extra

El Arquitecto de Software facilitó imágenes y documentación que se adjuntará en el documento.

Firmas			
Elaborado por		**Sujeto Entrevistado**	

Nota. Fuente: Elaboración propia.

d. Entrevista para Conocer el Tiempo y Costo Relacionado a la Documentación de un Proyecto Dentro de la Empresa.

Figura 70: Entrevista para Conocer el Tiempo y Costo Relacionado a la Documentación de un Proyecto Dentro de la Empresa.

KLEEEN SOFTWARE

Entrevista

Entrevista para la conocer el tiempo y costo relacionado a la documentación de un proyecto dentro de la empresa

Sujeto Entrevistado	Arquitecto de Software
Objetivo de la Entrevista	Conocer el tiempo y costo relacionado a la documentación de un proyecto dentro de la empresa

Estructura

1. ¿Existe algún proyecto que se está documentando en la empresa?

Sí, el de Playground

2. ¿De qué trata este proyecto?

Es un subproyecto de Kleeen Software IDE en el cual se busca que el usuario final pueda manipular un demo real de la aplicación

3. ¿Cuál es la documentación relacionada a este proyecto?

Se documenta la arquitectura y el diseño del sistema

4. ¿Cuál es el tiempo aproximado para realizar la documentación relacionada?

Nos ha tomado como 3 meses aproximadamente.

5. ¿Qué recursos están comprometidos o siendo utilizados en este proceso?

Ahorita estamos yo como arquitecto, y un desarrollador más.

6. ¿Cuál es el costo monetario aproximado de este proceso?

Somos los mismos dentro de la empresa, asociaría el costo a que estamos tiempo completo dentro de esto.

Notas extra

(Todas las notas, comentarios o asuntos de discusión relevantes para el presente proyecto, abordados durante la entrevista).

Firmas			
Elaborado por		**Sujeto Entrevistado**	

Nota. Fuente: Elaboración propia.

Apéndice I. Minutas de Reunión

a. Minuta Reunión 0

Figura 71:
Minuta Reunión 0

KLEEEN SOFTWARE

Minuta de la reunión

Reunión #:	Fecha de realización:	Hora de inicio:	Hora de finalización:
0.	15 de febrero del 2022	2:00 pm	3:00 pm

Asistentes a la reunión

Nombre	Puesto
Laura Carvajal	Estudiante de TFG
Mario Octavio	Vicepresidente de Ingeniería - Contraparte

Agenda

1. Situación actual de la organización
2. Análisis FODA de la organización.

Acuerdos

En concreto, se acordó lo siguiente:
1.

Aspectos Pendientes

1.

Notas de la reunión

Firmas

Participantes

Laura Daniela Carvajal Segura *(Estudiante de TFG)*	
Mario Octavio Jimenez *(Contraparte)*	

Nota. Fuente: Elaboración propia.

b. Minuta Reunión 1

Figura 72:
Minuta Reunión 1

KLEEEN SOFTWARE

Minuta de la reunión

Reunión #:	Fecha de realización:	Hora de inicio:	Hora de finalización:
1.	*22 de febrero del 2022*	*3:30 pm*	*4:15 pm*

Asistentes a la reunión

Nombre	Puesto
Laura Carvajal	*Estudiante de TFG*
Mario Octavio	*Contraparte*
Jacqueline Solis	*Tutora Académica*

Agenda

1. *Presentación de las partes*
2. *Lectura del reglamento actualizado de TFG*
   a. *Repaso de las responsabilidades por cada rol*
   b. *Repaso de fechas importantes*
   c. *22 al 28 primera evaluación*
3. *Expectativas de la contraparte*

Acuerdos

En concreto, se acordó lo siguiente:

1. *Se acordó un horario donde el estudiante de TFG y la contraparte estén cómodos y respeten las responsabilidades de cada uno.*
2. *El estudiante de TFG será el responsable de agendar las reuniones necesarias, tanto con la tutora, como con la contraparte.*
3. *Diseño de alto nivel de un sistema de generación de lenguaje natural el cual sea capaz de brindar una idea clara para una posible implementación del sistema dentro de la empresa, e inclusive llegar a valorar una posible patente del mismo.*

Aspectos Pendientes

1. Agendar las reuniones pendientes, así como los rangos de evaluaciones por parte de la contraparte al estudiante de TFG.

Notas de la reunión

Firmas

Participantes

Laura Daniela Carvajal Segura *(Estudiante de TFG)*	
Mario Octavio Jimenez *(Contraparte)*	
Jacqueline Solis *(Tutora Académica)*	Firmado por JACQUELINE TATIANA SOLIS CESPEDES (FIRMA) PERSONA FISICA, CPF-01-1295-0649. Fecha declarada: 08/04/2022 08:38 PM Esta representación visual no es fuente de confianza. Valide siempre la firma.

Nota. Fuente: Elaboración propia.

c. Minuta Reunión 2

Figura 73:
Minuta Reunión 2

KLEEEN SOFTWARE

Minuta de la reunión

Reunión #:	Fecha de realización:	Hora de inicio:	Hora de finalización:
2.	23 de febrero del 2022	3:00 pm	3:40 pm

Asistentes a la reunión

Nombre	Puesto
Laura Carvajal	Estudiante de TFG
Jorge Ramírez	Software Developer en Kleeen Software

Agenda

1. Contextualización del TFG
2. Búsqueda de documentación interna
3. Explicación de la arquitectura de Kleeen Authoring Tool
   a. Devops, flujos de AWS y la herramienta.
   b. Diagramas de actores
   c. Visualization Pipeline
   d. Viz Pipeline Process Overview
   e. Viz Pipeline - Rule matching

Acuerdos

En concreto, se acordó lo siguiente:
1. Jorge será un recurso clave para la consulta de documentación o información técnica necesaria para el proyecto.
2. El estudiante de TFG realizará los diagramas que considere pertinentes.

Aspectos Pendientes

Revisión de la documentación proveída por Jorge, así como la creación de los diagramas que se consideren necesarios.

Notas de la reunión

Existe documentación desactualizada que no debe ser contemplada para el desarrollo del proyecto, porque puede provocar ambigüedades o malentendidos.

Firmas

Participantes

Laura Daniela Carvajal Segura *(Estudiante de TFG)*	
Jorge Ramírez Zamora *(Software Developer en Kleeen Software)*	

Nota. Fuente: Elaboración propia.

d. Minuta Reunión 3

Figura 74:
Minuta Reunión 3

KLEEEN SOFTWARE

Minuta de la reunión

Reunión #:	Fecha de realización:	Hora de inicio:	Hora de finalización:
3.	24 de marzo del 2022	2:30 pm	3:30 pm

Asistentes a la reunión

Nombre	Puesto
Laura Carvajal	Estudiante de TFG
Mario Octavio	Vicepresidente de Ingeniería - Contraparte

Agenda

1. Actualización del estado del proyecto
2. Recopilación de requerimientos
3. Investigación de campo (posibles requerimientos a futuro)

Acuerdos

En concreto, se acordó lo siguiente:
1. Se levantó una lista con los requerimientos a un alto nivel

Aspectos Pendientes

1. Redacción de requerimientos formal
2. Aceptación de los requerimientos por parte del Vicepresidente de Ingeniería

Notas de la reunión

Firmas

Participantes

Laura Daniela Carvajal Segura *(Estudiante de TFG)*	
Mario Octavio Jimenez *(Contraparte)*	

Nota. Fuente: Elaboración propia.

e. Minuta Reunión 4

Figura 75:
Minuta Reunión 4

KLEEEN SOFTWARE

Minuta de la reunión

Reunión #:	Fecha de realización:	Hora de inicio:	Hora de finalización:
4.	*07 de abril del 2022*	*3:00 pm*	*3:40 pm*

Asistentes a la reunión

Nombre	Puesto
Laura Carvajal	*Estudiante de TFG*
Jacqueline Solís	*Tutora*
Mario Octavio	*Vicepresidente de Ingeniería - Contraparte*

Agenda

1. *Presentación del primer avance a la contraparte*
2. *Retroalimentación por parte de la profesora tutora y la contraparte*

Acuerdos

En concreto, se acordó lo siguiente:

1.

Aspectos Pendientes

1.

Notas de la reunión

Firmas

Participantes

Laura Daniela Carvajal Segura *(Estudiante de TFG)*	
Mario Octavio Jimenez *(Contraparte)*	
Jacqueline Solís *(Tutora)*	Firmado por JACQUELINE TATIANA SOLIS CESPEDES (FIRMA) PERSONA FISICA, CPF-01-1295-0649. Fecha declarada: 08/04/2022 08:36 PM Esta representación visual no es fuente de confianza. Valide siempre la firma.

Nota. Fuente: Elaboración propia.

f. Minuta Reunión 5

Figura 76:
Minuta Reunión 5

KLEEEN SOFTWARE

Minuta de la reunión

Reunión #:	Fecha de realización:	Hora de inicio:	Hora de finalización:
5.	12 de mayo del 2022	2:00 pm	3:00 pm

Asistentes a la reunión

Nombre	Puesto
Laura Carvajal	Estudiante de TFG
Jacqueline Solís	Tutora
Mario Octavio	Vicepresidente de Ingeniería - Contraparte

Agenda

1. Presentación del avance final a la contraparte
2. Retroalimentación por parte de la profesora tutora y la contraparte

Acuerdos

En concreto, se acordó lo siguiente:
1.

Aspectos Pendientes

1.

Notas de la reunión

Firmas

Participantes	
Laura Daniela Carvajal Segura (Estudiante de TFG)	
Mario Octavio Jimenez (Contraparte)	
Jacqueline Solís (Tutora)	JACQUELINE TATIANA SOLIS CESPEDES (FIRMA) Firmado digitalmente por JACQUELINE TATIANA SOLIS CESPEDES (FIRMA) Fecha: 2022.05.26 10:41:12 -06'00'

Nota. Fuente: Elaboración propia.

g. Minuta Reunión 6

Figura 77:
Minuta Reunión 6

KLEEEN SOFTWARE

Minuta de la reunión

Reunión #:	Fecha de realización:	Hora de inicio:	Hora de finalización:
6.	17 de mayo del 2022	11:40 am	12:50 pm

Asistentes a la reunión

Nombre	Puesto
Laura Carvajal	Estudiante de TFG
Jorge Ramírez	Software y Architect Developer en Kleeen Software

Agenda

1. Revisión de tiempo y costo relacionado a la documentación de un proyecto.

Acuerdos

En concreto, se acordó lo siguiente:
1.

Aspectos Pendientes

Revisión de la documentación proveída por Jorge, así como la creación del estudio correspondiente.

Notas de la reunión

Existe documentación desactualizada que no debe ser contemplada para el desarrollo del proyecto, porque puede provocar ambigüedades o malentendidos.

Firmas

Participantes

Laura Daniela Carvajal Segura *(Estudiante de TFG)*	
Jorge Ramírez Zamora *(Software Developer en Kleeen Software)*	

Nota. Fuente: Elaboración propia.

Apéndice J. Plan de Implementación

A partir de la última página ↓

Apéndice K. Plan de Gestión de Cambio

A partir de la última página ↓

10. Anexos

Anexo I. Cronograma del proyecto

A continuación, se muestra el posible cronograma del proyecto a cumplir, las columnas numeradas del 1 al 16 hacen alusión a las semanas del semestre en las que se trabajarán las tareas de la columna izquierda.

Tabla 45:
Cronograma del proyecto

Actividad	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Ajustes anteproyecto																
Desarrollo del capítulo 1																
Entrega capítulo 1																
Identificación de la situación actual																
Análisis de la situación actual																
Entrega Marco Metodológico																
Entrega Marco Conceptual																
Recopilación de requerimientos																
Análisis de requerimientos completado																
Desarrollo del análisis de resultados																
Diseño de alto nivel																
Evaluación del Diseño																
Correcciones del diseño																
Documentación de la propuesta de diseño																
Plan de Implementación y gestión de cambios																
Entrega Resultados y propuesta de solución																
Entrega Conclusiones y Recomendaciones																
Entrega documento final																

Nota. Fuente: Elaboración propia.

Anexo II. Tipo de datos KAPI

A continuación, se muestran los tipos de datos que soporta KAPI.

Tabla 46:
Tipos de datos soportados por KAPI

Name	Type	Prototype	Statistical Data Type	Aggregations
% completed - point values			Data - Numeric - Continuous	average, min, max, sum, countTotal, countUnique
% completed - whole values			Data - Numeric - Discrete	average, min, max, sum, countTotal, countUnique
address	Atomic	string	Data - Categorical - free form (only alphabetical)	countTotal, countUnique
address1	Atomic	string	Data - Categorical - free form (only alphabetical)	countTotal, countUnique
address2	Atomic	string	Data - Categorical - free form (only alphabetical)	countTotal, countUnique
am_pm	Atomic	string	Data - Numeric - Continuous - Temporal - Looped (24 hr, 1 week, 1 month, quarter, year)	min, max, countTotal, countUnique
array_of_digits(n = 7)		boolean	Data - Numeric - Discrete	average, min, max, sum, countTotal, countUnique

Name	Type	Prototype	Statistical Data Type	Aggregations
array_of_doubles(n = 7)			Data - Numeric - Continuous	average, min, max, sum, countTotal, countUnique
array_of_integers(n = 7)			Data - Numeric - Discrete	average, min, max, sum, countTotal, countUnique
array_of_words(n = 7)			Data - Categorical - free form (only alphabetical)	countTotal, countUnique
biz-quarter			Data - Categorical - ordered (Ordinal)	countTotal, countUnique
boolean	Primitive	prime	Data - Categorical - Binary	countTotal, countUnique
building_number	Atomic	string	Data - Categorical - ordered (Ordinal)	countTotal, countUnique
Car Make			Data - Categorical - unordered (only alphabetical)	countTotal, countUnique
Car Model			Data - Categorical - unordered (only alphabetical)	countTotal, countUnique
card_data	Atomic		Data - Categorical - unordered (only alphabetical)	countTotal, countUnique
card_exp	Atomic	string	Data - Numeric - Continuous - Temporal	min, max, countTotal, countUnique

Name	Type	Prototype	Statistical Data Type	Aggregations
card_number(vendor)	Atomic	string	Data - Categorical - unordered (only alphabetical)	countTotal, countUnique
card_type	Atomic	string	Data - Categorical - unordered (only alphabetical)	countTotal, countUnique
catch_phrase	Atomic	string	Data - Categorical - free form (only alphabetical)	countTotal, countUnique
century	Atomic	string	Data - Numeric - Continuous - Temporal	min, max, countTotal, countUnique
city	Atomic	string	Data - Categorical - unordered - geo	countTotal, countUnique
coin_flip			Data - Categorical - Binary	countTotal, countUnique
color_name	Atomic	string	Data - Categorical - unordered (only alphabetical)	countTotal, countUnique
company_name	Atomic	string	Data - Categorical - free form (only alphabetical)	countTotal, countUnique
company_suffix	Atomic	string	Data - Categorical - unordered (only alphabetical)	countTotal, countUnique
Completeness			Data - Numeric - Percentage	average, min, max, sum, countTotal, countUnique

Name	Type	Prototype	Statistical Data Type	Aggregations
country	Atomic	string	Data - Categorical - unordered - geo	countTotal, countUnique
country_code	Atomic	string	Data - Categorical - unordered - geo	countTotal, countUnique
currency	Atomic		Data - Categorical - unordered (only alphabetical)	countTotal, countUnique
currency_code	Atomic	string	Data - Categorical - unordered (only alphabetical)	countTotal, countUnique
currency_name	Primitive		Data - Categorical - unordered (only alphabetical)	countTotal, countUnique
currency_symbol	Atomic	string	Data - Categorical - unordered (only alphabetical)	countTotal, countUnique
date(format = 'YYYY-MM-DD')	Atomic	string	Data - Numeric - Continuous - Temporal	min, max, countTotal, countUnique
day_of_month	Atomic	integer	Data - Numeric - Continuous - Temporal - Looped (24 hr, 1 week, 1 month, quarter, year)	min, max, countTotal, countUnique
day_of_week	Atomic	integer	Data - Numeric - Continuous - Temporal - Looped (24 hr, 1 week, 1 month, quarter, year)	min, max, countTotal, countUnique
day_of_year	Atomic	integer	Data - Numeric - Continuous - Temporal - Looped (24 hr, 1 week, 1 month, quarter, year)	min, max, countTotal, countUnique

Name	Type	Prototype	Statistical Data Type	Aggregations
description	Atomic	string	Data - Categorical - free form (only alphabetical)	countTotal, countUnique
domain	Atomic	string	Data - Categorical - free form (only alphabetical)	countTotal, countUnique
double(from = -1000, to = 1000)	Primitive	prime	Data - Numeric - Continuous	average, min, max, sum, countTotal, countUnique
email	Atomic	string	Data - Categorical - free form (only alphabetical)	countTotal, countUnique
file_extension	Atomic	string	Data - Categorical - free form (only alphabetical)	countTotal, countUnique
first_name	Atomic	string	Data - Categorical - free form (only alphabetical)	countTotal, countUnique
full_name	Atomic	string	Data - Categorical - free form (only alphabetical)	countTotal, countUnique
gender			Data - Categorical - unordered (only alphabetical)	countTotal, countUnique
height			Data - Numeric - Continuous	average, min, max, sum, countTotal, countUnique
image-icon			Data - Image	countTotal, countUnique

Name	Type	Prototype	Statistical Data Type	Aggregations
image-raster			Data - Image	countTotal, countUnique
image-vector			Data - Image	countTotal, countUnique
integer(from = -1000, to = 1000)	Primitive	prime	Data - Numeric - Discrete	average, min, max, sum, countTotal, countUnique
ip	Atomic	string	Data - Categorical - ordered (Ordinal)	countTotal, countUnique
ipv6	Atomic	string	Data - Categorical - ordered (Ordinal)	countTotal, countUnique
language_code	Atomic	string	Data - Categorical - unordered (only alphabetical)	countTotal, countUnique
last_name	Atomic	string	Data - Categorical - free form (only alphabetical)	countTotal, countUnique
latitude	Atomic	string	Data - Numeric - Continuous - geo	average, min, max, countTotal, countUnique
Length			Data - Numeric - Continuous	average, min, max, sum, countTotal, countUnique
letter	Atomic	string	Data - Categorical - ordered (Ordinal)	countTotal, countUnique
locale	Atomic	string	Data - Categorical - unordered (only alphabetical)	countTotal, countUnique

Name	Type	Prototype	Statistical Data Type	Aggregations
longitude	Atomic	string	Data - Numeric - Continuous - geo	average, min, max, countTotal, countUnique
mime_type	Atomic	string	Data - Categorical - unordered (only alphabetical)	countTotal, countUnique
moment	Atomic	string	Data - Numeric - Continuous - Temporal	min, max, countTotal, countUnique
Money			Data - Numeric - Discrete	average, min, max, sum, countTotal, countUnique
month_name	Atomic	string	Data - Numeric - Continuous - Temporal - Looped (24 hr, 1 week, 1 month, quarter, year)	min, max, countTotal, countUnique
month_number	Atomic	integer	Data - Numeric - Continuous - Temporal - Looped (24 hr, 1 week, 1 month, quarter, year)	min, max, countTotal, countUnique
name	Atomic	string	Data - Categorical - free form (only alphabetical)	countTotal, countUnique
name_prefix	Atomic	string	Data - Categorical - unordered (only alphabetical)	countTotal, countUnique
name_suffix	Atomic	string	Data - Categorical - unordered (only alphabetical)	countTotal, countUnique
on/off			Data - Categorical - Binary	countTotal, countUnique

Name	Type	Prototype	Statistical Data Type	Aggregations
order			Data - Numeric - Discrete	average, min, max, sum, countTotal, countUnique
password	Atomic	string	Data - Categorical - free form (only alphabetical)	countTotal, countUnique
phone	Atomic	string	Data - Categorical - unordered (only alphabetical)	countTotal, countUnique
random	Atomic	double	Data - Numeric - Continuous	average, min, max, sum, countTotal, countUnique
Rating-Binary			Data - Categorical - Binary	countTotal, countUnique
Rating-Multi-Level			Data - Categorical - ordered (Ordinal)	countTotal, countUnique
rgb_array	Atomic	string	Data - Categorical - unordered (only alphabetical)	countTotal, countUnique
rgb_hex	Atomic	string	Data - Categorical - unordered (only alphabetical)	countTotal, countUnique
rgba	Atomic	string	Data - Categorical - unordered (only alphabetical)	countTotal, countUnique
rgba_array	Atomic	string	Data - Categorical - unordered (only alphabetical)	countTotal, countUnique

Name	Type	Prototype	Statistical Data Type	Aggregations
rgba_hex	Atomic	string	Data - Categorical - unordered (only alphabetical)	countTotal, countUnique
safe_color_name	Atomic	string	Data - Categorical - unordered (only alphabetical)	countTotal, countUnique
sentence	Atomic	string	Data - Categorical - free form (only alphabetical)	countTotal, countUnique
sentences(n = 3)			Data - Categorical - free form (only alphabetical)	countTotal, countUnique
severity-buckets			Data - Categorical - ordered (Ordinal)	countTotal, countUnique
severity-bucketSocre			Data - Categorical - ordered (Ordinal)	countTotal, countUnique
severity-score			Data - Numeric - Percentage	average, min, max, sum, countTotal, countUnique
short_description	Atomic	string	Data - Categorical - free form (only alphabetical)	countTotal, countUnique
state	Atomic	string	Data - Categorical - unordered - geo	countTotal, countUnique
state_abbr	Atomic	string	Data - Categorical - unordered - geo	countTotal, countUnique
status			Data - Categorical - ordered (Ordinal)	countTotal, countUnique

Name	Type	Prototype	Statistical Data Type	Aggregations
street	Atomic	string	Data - Categorical - free form (only alphabetical)	countTotal, countUnique
string	Primitive	prime	Data - Categorical - free form (only alphabetical)	countTotal, countUnique
temperature			Data - Numeric - Continuous	average, min, max, countTotal, countUnique
text	Atomic	string	Data - Categorical - free form (only alphabetical)	countTotal, countUnique
time elapsed			Data - Numeric - Continuous	average, min, max, sum, countTotal, countUnique
time(format = 'HH:mm:ss')	Atomic	string	Data - Numeric - Continuous - Temporal - Looped (24 hr, 1 week, 1 month, quarter, year)	min, max, countTotal, countUnique
timestamp	Atomic	integer	Data - Numeric - Continuous - Temporal	min, max, countTotal, countUnique
timezone	Atomic	string	Data - Categorical - unordered (only alphabetical)	countTotal, countUnique
title	Atomic	string	Data - Categorical - free form (only alphabetical)	countTotal, countUnique

Name	Type	Prototype	Statistical Data Type	Aggregations
unix_time	Atomic	integer	Data - Numeric - Continuous - Temporal	min, max, countTotal, countUnique
url	Atomic	string	Data - Categorical - free form (only alphabetical)	countTotal, countUnique
USD	Atomic	integer	Data - Numeric - Discrete	average, min, max, sum, countTotal, countUnique
user_agent	Atomic	string	Data - Categorical - unordered (only alphabetical)	countTotal, countUnique
username	Atomic	string	Data - Categorical - free form (only alphabetical)	countTotal, countUnique
UTC offset			Data - Categorical - ordered (Ordinal)	countTotal, countUnique
uuid	Atomic	string	Data - Categorical - free form (only alphabetical)	countTotal, countUnique
weight			Data - Numeric - Continuous	average, min, max, sum, countTotal, countUnique
width			Data - Numeric - Continuous	average, min, max, sum, countTotal, countUnique
word	Atomic	string	Data - Categorical - free form (only alphabetical)	countTotal, countUnique

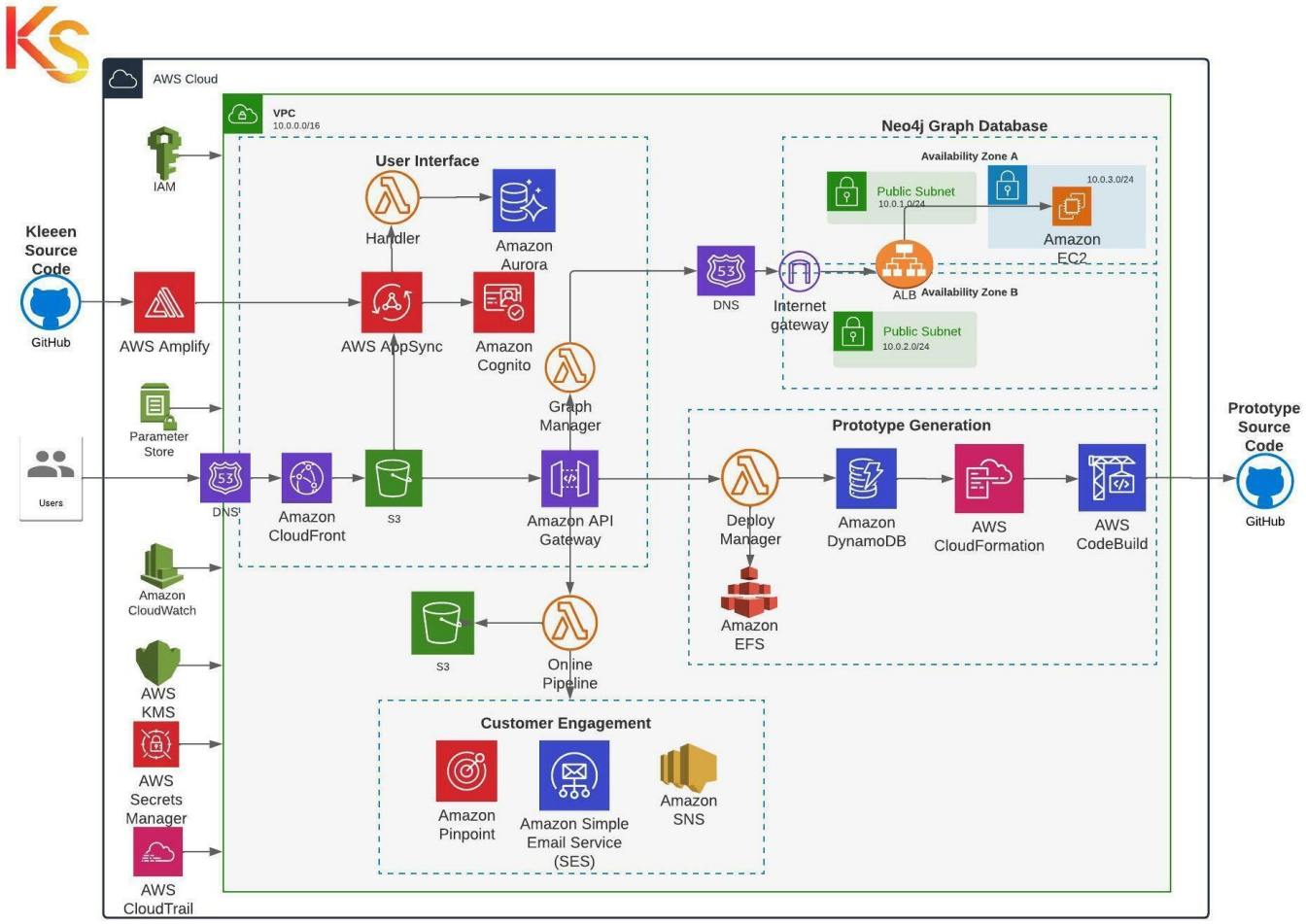
Name	Type	Prototype	Statistical Data Type	Aggregations
words(n = 7)			Data - Categorical - free form (only alphabetical)	countTotal, countUnique
year	Atomic	integer	Data - Numeric - Continuous - Temporal	min, max, countTotal, countUnique
yes/no			Data - Categorical - Binary	countTotal, countUnique
zip(digits = {5, 9})	Atomic	string	Data - Categorical - unordered - geo	countTotal, countUnique

Nota. Fuente: Kleeen Software

Anexo III. Arquitectura y flujo de los servicios consumidos por Kleeen Software IDE

A continuación, se muestra un diagrama con la arquitectura y flujo de los servicios consumidos por Kleeen Software IDE.

Figura 78:
Arquitectura y flujo de los servicios consumidos por Kleeen Software IDE

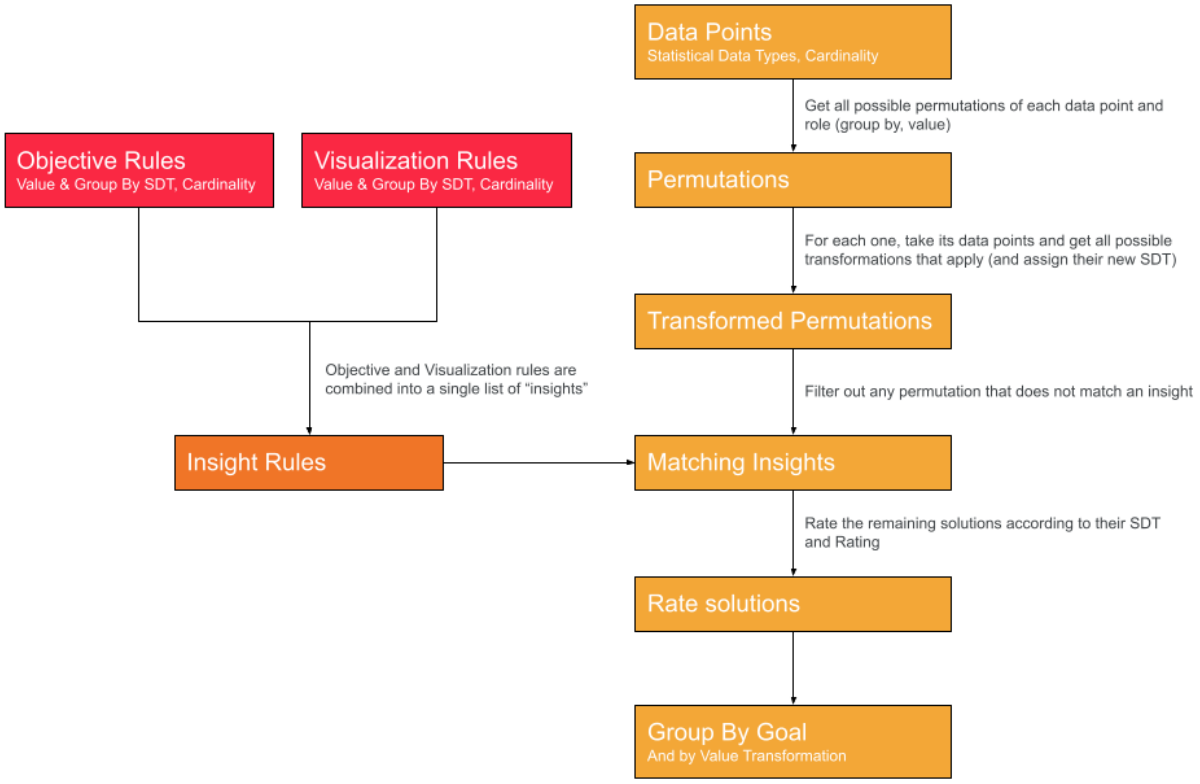


Nota. Fuente: Kleeen Software

Anexo IV. Viz Pipeline - Rule matching

A continuación, se muestra el diagrama del *Viz Pipeline - Rule matching* de Kleeen Software IDE.

Figura 79:
Viz Pipeline - Rule matching

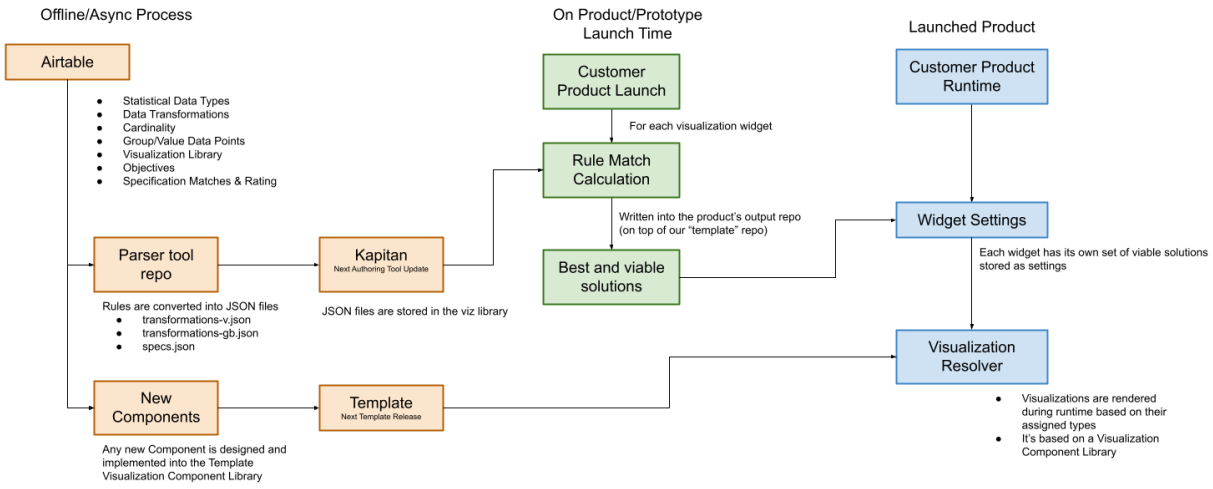


Nota. Fuente: Kleeen Software

Anexo V. Viz Pipeline Process Overview

A continuación, se muestra el diagrama del *Viz Pipeline Process Overview* de Kleeen Software IDE.

Figura 80:
Viz Pipeline Process Overview



Nota. Fuente: Kleeen Software

Anexo VI. Escala de Likert aplicada: Validación del Diseño de Datos, Diseño Arquitectónico, de Componentes, y de Despliegue.

Figura 81:
Escala de Likert aplicada: Validación del Diseño de Datos, Diseño Arquitectónico, de Componentes, y de Despliegue.

a. ¿Cómo es tu nivel de satisfacción con el diseño de software propuesto?
- ☑ ~~Muy satisfecho~~
- ☐ Satisfecho
- ☐ Normal
- ☐ Poco satisfecho
- ☐ Nada satisfecho

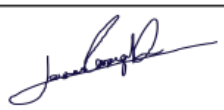
b. ¿Cómo es tu nivel de satisfacción con la evaluación de diseño realizada?
- ☑ ~~Muy satisfecho~~
- ☐ Satisfecho
- ☐ Normal
- ☐ Poco satisfecho
- ☐ Nada satisfecho

c. ¿Cómo valoras el diseño de software propuesto?
- ☑ ~~Muy bueno~~
- ☐ Bueno
- ☐ Indiferente
- ☐ Malo
- ☐ Muy malo

d. ¿Cómo valoras la evaluación de diseño realizada?
- ☑ ~~Muy buena~~
- ☐ Buena
- ☐ Indiferente
- ☐ Mala
- ☐ Muy mala

e. ¿Qué tan clara fue la documentación suministrada?
- ☑ ~~Muy buena~~
- ☐ Buena
- ☐ Indiferente
- ☐ Mala
- ☐ Muy mala

f. ¿Qué tan detallado es el diseño de software?
- ☑ ~~Muy detallado~~
- ☐ Algo detallado
- ☐ Indiferente
- ☐ Poco detallado
- ☐ Para nada detallado

Firmas	
Participantes	
Laura Daniela Carvajal Segura *(Estudiante de TFG)*	
Jorge Ramírez Zamora *(Software Developer en Kleeen Software)*	

Nota. Fuente: Kleeen Software

Anexo VII. Escala de Likert aplicada: Validación de Interfaces y Elementos del Diseño Web (UX & UI)

Figura 82:
Escala de Likert aplicada: Validación de Interfaces y Elementos del Diseño Web (UX & UI)

a. ¿Cómo es tu nivel de satisfacción con el diseño de interfaz propuesto?
   - [x] Muy satisfecho
   - [ ] Satisfecho
   - [ ] Normal
   - [ ] Poco satisfecho
   - [ ] Nada satisfecho

b. ¿Cómo es tu nivel de satisfacción con la prueba de concepto mostrada en el Experimento de Mago de Oz?
   - [x] Muy satisfecho
   - [ ] Satisfecho
   - [ ] Normal
   - [ ] Poco satisfecho
   - [ ] Nada satisfecho

c. ¿Cómo valoras el diseño de interfaz propuesto?
   - [x] Muy bueno
   - [ ] Bueno
   - [ ] Indiferente
   - [ ] Malo
   - [ ] Muy malo

d. ¿Cómo valoras la prueba de concepto mostrada en el Experimento de Mago de Oz?
   - [x] Muy buena
   - [ ] Buena
   - [ ] Indiferente
   - [ ] Mala
   - [ ] Muy mala

e. ¿Qué tan clara fue la documentación suministrada?
   - [x] Muy buena
   - [ ] Buena
   - [ ] Indiferente
   - [ ] Mala
   - [ ] Muy mala

f. ¿Qué tan intuitiva fue la prueba de concepto mostrada en el Experimento de Mago de Oz?
   - [ ] Muy intuitiva
   - [x] Algo intuitiva
   - [ ] Indiferente
   - [ ] Poco intuitiva
   - [ ] Para nada intuitiva

g. Con respecto a los clicks que se evidencian en la prueba de concepto mostrada en el Experimento de Mago de Oz, ¿Considera que estos fueron?
   - [ ] Excesivos
   - [ ] Muchos
   - [x] Justo los necesarios
   - [ ] Pocos
   - [ ] Nulos

h. Con respecto al texto mostrado en la prueba de concepto mostrada en el Experimento de Mago de Oz, ¿Considera que estos fueron?
   - [ ] Excesivos
   - [ ] Muchos
   - [x] Justo los necesarios
   - [ ] Pocos
   - [ ] Nulos

Firmas	
Participantes	
Laura Daniela Carvajal Segura *(Estudiante de TFG)*	
Mario Octavio Jimenez *(Vicepresidente de Ingeniería - Contraparte)*	

Nota. Fuente: Kleeen Software

11.Glosario

A continuación, se presenta una lista de palabras con su respectivo significado, palabras las cuales facilitan el entendimiento del proyecto:

- *Unified Modeling Language* (UML): El lenguaje de modelado unificado (UML) es un lenguaje de modelado de propósito general estandarizado en el campo de Ingeniería de software orientada a objetos. UML incluye un conjunto de técnicas de notación gráfica para crear modelos visuales de sistemas de software orientados a objetos.
- Kit de desarrollo de software (SDK): Un kit de desarrollo de *software* es generalmente un conjunto de herramientas de desarrollo de *software* que permite a un desarrollador de software crear una aplicación informática para un sistema concreto
- Ontologías: En Ciencia y Tecnologías de la Información, las ontologías son clasificaciones. Se utilizan como un medio para categorizar o agrupar la información en clases.
- Tesauros: Un tesauro es una lista de palabras o términos empleados para representar conceptos.

----------Última página----------

A continuación, se adjuntan los documentos de:

- Plan de Implementación
- Plan de Gestión de Cambios

Plan de Implementación
KS-NLG

Version 1.0
04 de Mayo, 2022

KLEEEN SOFTWARE

Tabla de contenido

Introducción

Propósito

La empresa Kleeen Software requiere un diseño de software para la implementación de un sistema que permita recibir datos no procesados, y los convierta a información entendible por el ser humano, es decir, el diseño de un sistema de generación de lenguaje natural.

Definición del producto

El producto esperado se enfoca en brindar puntos de datos con contexto, y con la capacidad de extraer y entender grandes cantidades de datos, ya sean numéricos o alfabéticos, con el fin de identificar patrones y compartir esa información de una manera sencilla de entender por los humanos.

Plan de Implementación

Objetivo

Elaborar un plan de implementación del diseño de software de alto nivel del sistema de generación de lenguaje natural (KS-NLG) propuesto, en la empresa Kleeen Software, durante el segundo semestre del 2022.

Tareas Mínimas

Para el presente plan se proponen una serie de tareas o requerimientos mínimos por realizar al momento de la implementación e integración del sistema propuesto KS-NLG, con el sistema actual Kleeen Software IDE.

Estas tareas son una recomendación con el objetivo de realizar un proceso limpio y ordenado de la gestión del proyecto.

Las tareas mínimas son las siguientes:

1. **Analizar el sistema Kleeen Software IDE previo a la integración**

Pese a que en la propuesta realizada existe un análisis del sistema Kleeen Software IDE, desde el momento en que se realizó dicho análisis, hasta el momento en el que se llevará a cabo la implementación de KS-NLG, es posible que hayan ocurrido cambios a nivel arquitectónico, de estructura, o de tecnologías, por lo que se requiere una nueva revisión de este.

2. **Determinar los requerimientos de integración**

Con requerimientos de integración se refiere a todos aquellas necesidades que el cliente exprese referentes a la integración del sistema, así como todos los requisitos que se identifiquen a nivel de diseño, desarrollo, o tecnologías.

3. **Revalidar el diseño propuesto de KS-NLG de acuerdo al análisis del sistema de Kleeen Software IDE**

Como se mencionó anteriormente, al realizar un nuevo análisis de Kleeen Software IDE, es probable encontrar cambios, es por esto, que se debe realizar también una re validación del diseño propuesto de KS-NLG, para así, realizar los cambios que sean necesarios, de manera que se obtenga información actualizada y correcta.

4. **Establecer el alcance de la integración**

Al definir los requerimientos mencionados en la tarea 2, se obtiene una perspectiva sobre los deseos o necesidades del cliente, de manera que se puede definir el alcance de la integración, es decir, en qué módulos, tecnologías, o pantallas se requiere la implementación de KS-NLG, así como la definición del tiempo de implementación o fecha límite de entrega sugerida.

5. **Definir recursos necesarios**

Una vez establecido el alcance y los requerimientos necesarios, se debe definir los recursos necesarios, ya sean recursos humanos, financieros, tecnológicos o tiempos de implementación.

6. **Valorar costos de implementación**

Al establecer los recursos humanos, financieros y tecnológicos, se deben calcular los costos de implementación, así como cualquier otro costo asociado.

7. **Implementar KS-NLG**

De acuerdo al diseño de alto nivel propuesto, y las correcciones obtenidas de la revalidación de la tarea 3, se debe implementar el sistema KS-NLG.

8. **Diseñar el proceso de integración**

El proceso de integración debe diseñarse de manera minuciosa y detallada, con el objetivo de evitar riesgos, o fallos al momento de integrar los sistemas involucrados. Se debe tomar en cuenta los diferentes tipos de integración existentes.

Con base en las investigaciones de Rivas (2006) y Medina (2019), a continuación, se explican los diferentes tipos de integración de sistemas que existen en la actualidad.

- Integración horizontal: Es el método para definir un sistema de comunicación, se enfoca en la transmisión y el monitoreo de mensajes, también proporciona servicios de transformación y mapeo de datos. Este tipo permite flexibilidad, en la que los equipos pueden agregar, quitar o ajustar un sistema sin interrumpir el resto de los componentes.
- Integración vertical: Es una solución a corto plazo, rápida y económica, debido a que este método se concentra en el desarrollo de entidades funcionales que permitan una conexión con los sistemas de software de manera vertical. Sin embargo, este tipo desarrolla una limitación al escalar el software, lo que significa que la información no se comparte correctamente y se aísla en cada sistema.
- Integración de formato de datos común: Consiste en evitar el uso de un adaptador al convertir o transportar datos, por medio de la estandarización de estos, de manera que un sistema debe ser aceptado por el otro sistema. Este tipo permite la traducción de datos y promueve la automatización.

9. **Realizar la integración de software**

En esta tarea se pone en marcha la integración diseñada previamente, velando por el correcto funcionamiento y el cumplimiento de los requerimientos especificados.

10. **Realizar comprobaciones de mantenimiento**

Una vez se concluya la integración, se deben realizar comprobaciones de mantenimiento para el sistema, de manera que se prevean y planifiquen cambios o mejoras pendientes.

11. Desarrollar pruebas de calidad, funcionalidad, y pruebas de usuario

Se deben realizar pruebas de calidad, funcionamiento y pruebas de usuario con el fin de garantizar el correcto desarrollo de la integración, así como la verificación de cumplimiento de los requerimientos especificados.

12. Paso a Producción

Concluidas las pruebas, el sistema se pasa a producción donde los clientes finales pueden hacer uso del mismo.

Recursos Requeridos

Para la implementación de KS-NLG se especifican los siguientes recursos necesarios:

Tiempo Estimado (Cronograma)

Para la implementación se dispone de dieciséis semanas laborables (jornada de cuarenta horas semanales), lo cual equivaldría a cuatro meses.

De acuerdo a las tareas mencionadas en el apartado **Tareas Mínimas**, se propone el siguiente cronograma.

Recurso	Mes	Agosto				Septiembre				Octubre				Noviembre			
Recurso Humano	**Semana / Actividad**	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
AP	Analizar el sistema Kleeen Software IDE previo a la integración																
AP - Arquitecto de Software	Determinar los requerimientos de integración																
AP - Arquitecto de Software	Revalidar el diseño propuesto de KS-NLG de acuerdo al análisis del sistema de Kleeen Software IDE																
AP	Establecer el alcance de la integración																
AP	Definir recursos necesarios																
AP	Valorar costos de implementación																
AP - Arquitecto de Software - Desarrollador de Software	Implementar KS-NLG																
AP - Arquitecto de Software - Desarrollador de Software	Diseñar el proceso de integración																
AP - Arquitecto de Software - Desarrollador de Software	Realizar la integración de software																
AP - Arquitecto de Software - Desarrollador de Software	Realizar comprobaciones de mantenimiento																
AP - Arquitecto de Software - Desarrollador de Software	Desarrollar pruebas de calidad, funcionalidad, y pruebas de usuario																
AP - Arquitecto de Software - Desarrollador de Software	Paso a Producción																

Recurso Humano

Con respecto a las tareas necesarias, se proponen los siguientes recursos humanos para el proceso de integración:

Puesto	Salario Mensual	Meses Laborales	Total Salario
Administrador de Proyectos	₡941,001.00	4	₡3,764,004.00
Arquitecto de Software	₡1,186,751.00	3	₡3,560,253.00
Desarrollador de Software 1	₡887,600.00	2.5	₡2,219,000.00
		Total Salarios	**₡9,543,257.00**

Nota. Fuente: Elaboración propia.

Los salarios anteriores se tomaron con base en la página Glassdoor, la cual es una bolsa de trabajo basado en la web que permite a los empleadores y empleados conocer la situación laboral en las empresas así como obtener comparaciones de salarios.
De acuerdo con esta página, el salario para un administrador de proyectos equivale a ₡941,001.00, de un arquitecto de software a ₡1,186,751.00, y el de desarrollador de software ronda el ₡887,600.00. Por lo tanto, este desarrollo le costaría a la empresa ₡9,543,257.00 por los 4 meses laborados, en términos de recurso humano.

Firmas	
Participantes	
Laura Daniela Carvajal Segura *(Estudiante de TFG)*	
Mario Octavio Jimenez *(Vicepresidente de Ingeniería - Contraparte)*	

Plan de Gestión de Cambio

Version 1.0

04 de Mayo, 2022

KLEEEN SOFTWARE

Tabla de contenido

Plan de Gestión del Cambio

Objetivo

Elaborar un plan de gestión de cambio para el proceso de implementación del diseño de software de alto nivel del sistema de generación de lenguaje natural (KS-NLG) propuesto, en la empresa Kleeen Software, durante el segundo semestre del 2022.

Gestión de Cambio

Al desarrollar un proyecto de TI los cambios son un elemento que se debe tener presente, esto debido a que la tecnología es algo volátil y cambiante, por ende, las modificaciones, los nuevos requerimientos, limitaciones y demás, son recurrentes.

La integración de KS-NLG con el sistema actual Kleeen Software IDE es un cambio relevante para el flujo de trabajo y uso, por esto se requiere que este proceso sea controlado y comunicado correctamente a todos los involucrados.

1. Descripción del cambio

La empresa Kleeen Software desea implementar el diseño de software de alto nivel del sistema de generación de lenguaje natural (KS-NLG) propuesto.

El resultado esperado se enfoca en brindar información con contexto, y con la capacidad de extraer y entender grandes cantidades de datos, ya sean numéricos o alfabéticos, con el fin de identificar patrones y compartir esa información de una manera sencilla de entender por los humanos.

2. Justificación de negocios

Actualmente, la empresa Kleeen Software se dedica al diseño y desarrollo de un sistema que permite la automatización para la creación de aplicaciones de gestión de información, las cuales se basan en las intenciones y visión del cliente, este se conoce como Kleeen Software IDE. El sistema se encuentra automatizado en su mayoría, tanto al momento de la creación y codificación, como al momento de generar productos terminados con respecto a la configuración del cliente.

Esta automatización es configurada por los mismos desarrolladores, sin embargo, aún existen vacíos al manipular la información tomada del backend al momento de mostrarla al usuario final, de manera que los datos no siempre están contextualizados en la visión del usuario final, provocando así confusión por parte de los usuarios al no lograr comprender la información o el significado de los datos mostrados por el sistema. Correspondiente a la naturaleza del producto que desarrolla la empresa Kleeen Software, el usuario final no siempre serán desarrolladores o personas con conocimientos en informática, por esto, cabe suponer que aquellas personas que requieren utilizar el sistema para realizar consultas o manipular los datos, buscan una forma sencilla y eficiente de comunicarse.

Aunque la empresa cuenta con arquitectura de software establecida y el diseño de software para el sistema actual Kleeen Software IDE, no existe ningún diseño para la implementación de

un sistema que permita recibir datos no procesados, y los convierta a información entendible por el ser humano. Es decir, falta un diseño para un sistema de generación de lenguaje natural, lo que provoca que el sistema antes mencionado, no satisfaga la necesidad de brindar respuestas contextualizadas e inteligentes al usuario final basándose en su contexto e intenciones

3. Análisis de impacto y riesgo

Para evaluar el impacto y el riesgo del proyecto, se elaboró el siguiente cuestionario:

1. ¿Cuál es el alcance del impacto diseñado para este cambio?
    a. *El sistema Kleeen Software IDE*
2. ¿Es esta una actividad compleja o de alto riesgo?
    a. *Es una actividad compleja, mas no de alto riesgo.*
3. ¿Puede este cambio afectar potencialmente la disponibilidad, integridad y/o seguridad de otros Sistemas de TI?
    a. *Sí, debido a que es una integración de una herramienta que se desarrolla por primera vez en la empresa, y requiere una integración con el sistema que actualmente se encuentra en producción, es posible que tanto la disponibilidad, integridad y seguridad del sistema se vea afectada.*
4. ¿Se ha probado este cambio?
    a. *No, nunca antes se había implementado un cambio o integración similar.*
5. ¿En qué estado estará el sistema Kleeen Software IDE durante la implementación?
    a. *El sistema seguirá funcionando en producción, sin embargo, al momento de su lanzamiento, el sistema podría verse interrumpido en un corto lapso.*
6. ¿Cuándo se producirá este cambio?
    a. *En cuanto comience el proceso mencionado en el Plan de Implementación, por ende, iniciaría tentativamente en Agosto 2022.*

Este cuestionario permite ver el estado, el alcance del impacto, el riesgo, la disponibilidad y el procedimiento general que se va a realizar y describe cómo va a estar el sistema mientras se realiza el cambio.

4. Plan de comunicación

A continuación se presentará una propuesta para un plan de comunicación para presentar el cambio que provocará la integración de los sistemas, el plan va a contener diferentes apartados, por cada uno de estos se presentará un posible ejemplo por apartado:

- Objetivos del Plan de Comunicación:
  - Dar a conocer el cambio con respecto a la integración del nuevo sistema KS-NLG con Kleeen Software IDE..
  - Proporcionar una base documental y material de referencia para futuros cambios de este tipo que se efectúen.
- Política de comunicación:
  - Aquí se citaría o se adjuntará toda o alguna parte de la política de comunicación existente.
- Audiencia destinataria:
  - Segmentación de los usuarios:
    - Cliente: se refiere a todos aquellos clientes de la empresa que hagan uso del sistema Kleeen Software IDE.
    - Inversionistas: se refiere a aquellas personas u organizaciones que se encuentren invirtiendo en el sistema.
    - Desarrolladores: son los desarrolladores involucrados con el sistema.
    - Gerencia y Directivos: son las personas encargadas de labores administrativas de la empresa.
  - Prioridades informativas:
    - Prioridad alta: todo aquel usuario primario (usuario que utiliza diariamente el sistema)
    - Prioridad Media: comunidad perteneciente a la empresa.
    - Prioridad Baja: comunidad externa.
- Canales de comunicación:
  - Slack.
  - Email empresarial.
  - Reuniones diarias.
- Calendario de las acciones:
  - Fase 1: Publicación del Plan de Comunicación.
  - Fase 2: Difusión y divulgación del comunicado.
  - Fase 3: Evaluación del Plan de Comunicación.
- Responsables:
  - Vicepresidente de Ingeniería.
  - Encargado de la puesta en marcha..
- Evaluación:
  - La evaluación dependerá de los objetivos presentados en el Plan de Comunicación, de esta manera se podrá evaluar la efectividad de la puesta en marcha del plan.
  - Posibles métricas para la evaluación:
    - Número de usuarios alcanzados con las publicaciones.
    - Número de solicitudes o consultas relacionadas al cambio.

Firmas	
Participantes	
Laura Daniela Carvajal Segura *(Estudiante de TFG)*	
Mario Octavio Jimenez *(Vicepresidente de Ingeniería - Contraparte)*