UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

TÍTULO: Herramienta para relevar especificaciones basadas en lenguaje natural en forma colaborativa.

AUTORES: Ruffolo, Nicolás Simón – Sansone, Emiliano Andrés

DIRECTOR/A: Antonelli, Leandro

CODIRECTOR/A: Fernández, Alejandro

CARRERA: Licenciatura en Sistemas

Resumen

Esta tesina se centra en una fase fundamental del ciclo de vida del desarrollo de software: la obtención y análisis de requerimientos. Se propone una herramienta que implementa un método para la consolidación de especificaciones básicas, denominadas kernel sentences. Estas especificaciones son propuestas por expertos en el dominio y posteriormente validadas de manera colaborativa por los mismos expertos. La herramienta ofrece soporte tecnológico para facilitar el trabajo colaborativo y para realizar ciertas revisiones estilísticas de la redacción.

Palabras Clave

Elicitación de requerimientos, Validación de requerimientos, Kernel sentences, crowd requirements.

Conclusiones

La herramienta implementa el método propuesto en el artículo “A Collaborative Approach to specify Kernel Sentences using Natural Language” de Leandro Antonelli, Alejandro Fernandez, Nicolás Ruffolo, Emiliano Sansone, y Diego Torres presentado en el Workshop in Requirements Engineering (WER), Natal, Brasil, Agosto
23 – 26 (2022). La implementación ha sido puesta aprueba, verificándose su relevancia al obtener una calificación positiva.

Trabajos Realizados

Se desarrolló una herramienta denominada Ruems Requeriments con el objetivo de asistir en el relevamiento y validación de especificaciones de requerimientos. La herramienta implementa un método que fue mejorado y ajustado durante el proceso de desarrollo. Paralelamente a la implementación, se realizó un estudio exhaustivo de las herramientas existentes, así como de las técnicas colaborativas y los frameworks de procesamiento de lenguaje natural. Finalmente, se evaluó la usabilidad de la herramienta mediante una encuesta System Usability Scale (SUS).

Trabajos Futuros

Las líneas de trabajo futuro se dividen en dos vertientes. Por un lado, se continuará mejorando el método implementado por la herramienta, y por otro, se enfocará en la optimización de la usabilidad de la misma. Para ambas tareas, es fundamental desarrollar una aplicación móvil que facilite una mayor aceptación y practicidad por parte de los usuarios.

Fecha de la presentación: NOVIEMBRE 2024

UNIVERSIDAD NACIONAL DE LA PLATA
FACULTAD DE INFORMÁTICA

TESINA DE GRADO

"Herramienta para relevar especificaciones basadas en lenguaje natural en forma colaborativa"

Licenciatura en Sistemas

Autores:

Sansone Emiliano Andrés- Ruffolo Nicolás Simón

Director y Co-Director:

Antonelli, Leandro – Fernández, Alejandro

Fecha de Entrega

Noviembre 2024

Agradecimientos

A nuestros directores Leandro Antonelli y Alejandro Fernández los cuales estuvieron siempre a disposición y acompañándonos desde el comienzo de esta tesina y su gran brecha de aprendizaje los cuales permitieron contribuir a lo que hoy en día nos gusta hacer. Sin su apoyo a lo largo de este trabajo no hubiera sido de la misma manera.

Agradecemos a todos nuestros compañeros que estuvieron a lo largo de toda nuestra carrera para brindarnos sus consejos, apoyo y compañerismo a lo largo de todos estos años.

“La amistad no pide nada a cambio, excepto un poco de mantenimiento (Georges Brassens)”

Agradecemos a la facultad de informática y a la Universidad Nacional de La Plata por la formación dada, a cada uno de nosotros. A todos ellos que dan sustento y apoyo a la educación pública,gratuita y de calidad para que cada vez le sea posible a más personas lograr educarse de una manera profesional.

Agradecemos finalmente a nuestras familias por incentivarnos y el apoyo constante desde nuestros inicios en la decisión de estudiar una carrera de grado para poder desarrollarnos como profesionales que somos. Por el esfuerzo incondicional para cuidar y priorizar siempre nuestro crecimiento personal, y para permitirnos completar nuestros estudios y acompañándonos durante todo este recorrido.

“Sin una familia, el hombre solo en el mundo, tiembla con el frío (Andre Maurois)”

ÍNDICE

1 Introducción

Este capítulo busca introducir al lector acerca de las problemáticas principales que se presentan en la tesina, definiendo cuáles son los objetivos a obtener, la motivación que conllevó a desarrollar la herramienta y la contribución que se esperaba tanto en el ámbito académico como en el laboral.

1.1 Motivación

Este capítulo muestra como con paso de los años el ser humano empezó a reemplazar tareas manuales por tareas automatizadas. En esta automatización, los sistemas de software cumplen un rol importante, ya que permiten administrar procesos y asignar recursos [35]. Esta automatización tiene por objetivo reducir tiempo, costo y esfuerzo para maximizar el beneficio para el ser humano y que pueda resolver sus objetivos de manera más precisa con los parámetros recién mencionados.

Los desarrollos de sistemas de software se debe llevar a cabo a través de un proceso estandarizado, el cual es conocido como ciclo de vida de desarrollo de software. Si bien existen muchos modelos (cascada, v, espiral, etc.) [41] y dos corrientes bien diferenciadas (clásica y ágil) [32], en general, el comienzo de un desarrollo se realiza con la etapa de ingeniería de requerimientos.

La ingeniería de requerimientos [10] es la disciplina que tiene por objetivo lograr una especificación del software a desarrollar. Esta especificación debe describir las funciones que realizará el sistema y servirá como acuerdo entre las partes involucradas. Es por esto, que la especificación debe ser completa, consistente y no ambigua entre otras características; ya que si la especificación tiene falencias, el software basado en una especificación de tales características no cumplirá las expectativas y puede llevar a errores a futuro [32].

La especificación de requerimientos es una tarea compleja, ya que debe unir dos mundos muy diferentes: (i) él mundo real, con expertos del dominio, y (ii) el mundo de desarrollo de software. Cada mundo tiene su lenguaje y sus reglas. Es por eso que es todo un desafío el poder unificarlos y obtener el conocimiento y especifico de forma de que sea entendible por todos [32]. Boehm [8] sostiene que un error con costo relativo 1 que se introduce en la etapa de requerimientos puede costar de entre 100 a 200 veces más corregirlo en la etapa de mantenimiento o cuando el software es entregado. Lograr una especificación de requerimientos lo más completa y correcta posible permite reducir tiempo y esfuerzo, ya que reduce la reelaboración por errores originados por una mala comprensión de los requerimientos. Además, mejora la comunicación entre los stakeholders (todos los involucrados en el proyecto de desarrollo, tanto del lado de los usuarios como del equipo de desarrollo que afectan directa o inderectamente el proyecto), ya que se cuenta con un cuerpo de conocimiento aceptado y consensuado por todos los involucrados que pueden utilizar, resolver conflictos o dudas. Todo esto, redunda en una mejora continua de la calidad del

software resultante, ya que el foco del desarrollo se puede poner en el diseño técnico de la solución y no se invierte esfuerzo en entender el problema que se debe resolver [5].

El conocimiento y los requerimientos para desarrollar un software está generalmente distribuido entre muchos Stakeholders [39]. Existen dos tipos de stakeholders: (i) internos y (ii) externos. Los stakeholders internos son por ejemplo: empleados, gerentes, junta directiva o inversionistas. Su interés en una empresa es distinto en cada caso. Los empleados quieren prosperar en su trabajo. Los propietarios están interesados en maximizar el beneficio que obtiene el negocio. Los inversores están preocupados por obtener retorno de su inversión. Los stakeholders externos son personas que no pertenecen a la empresa en sí, pero que se preocupan o se ven afectadas por el desempeño de la misma. Por ejemplo: consumidores, distribuidores o proveedores. Al igual que sucede con los stakeholders internos, los externos tienen diferentes motivaciones. Los clientes quieren que el negocio proporcione bienes o servicios de alta calidad a bajo costo. Los proveedores quieren que el negocio continúe comprandoles. Los acreedores quieren que se les pague en tiempo y en su totalidad. Dada la gran cantidad y variedad de stakeholders, es muy difícil involucrar a todas las personas en un proceso de relevamiento. Y si esto fuera posible, es más difícil aunque las personas se pongan de acuerdo debido a los intereses bien diferentes que poseen.

Hay dos herramientas muy habituales para especificar requerimientos que son: (i) Use Cases y (ii) User Stories. Use Cases proveen una descripción de las acciones de un sistema desde el punto de vista del usuario. La funcionalidad del sistema es modelada usando actores que hacen acciones sobre el sistema. Los Use Cases son servicios o funciones provistas por el sistema para sus usuarios. User Stories por su parte, son una descripción breve de una funcionalidad de software tal y como la percibe el usuario [15]. Las Users Stories sé usan principalmente en metodologías ágiles. La más utilizada es Scrum que fue creada por Jeff Sutherland [52]. Él afirma que Scrum logra que hagamos el doble de trabajo en la mitad de tiempo.

Tanto los Uses Cases cómo las User Stories usan oraciones simples en sus descripciones. Harris [27] definió a estas oraciones con el nombre de Kernel Sentences.Las mismas se caracterizan por tener un solo verbo, ser afirmativas, activas, declarativa, no contiene ningún estado de ánimo. Se denominan kernel, ya que son la base sobre la cual se forman otras oraciones más complejas.

1.2 **Objetivos**

El objetivo de la tesina fue el desarrollo de una herramienta que brinde soporte colaborativo a un grupo de stakeholders con el objetivo de relevar y validar requerimientos especificados a través de kernel sentences como oraciones específicas sobre las cuales se pueden generar otras. Logrando de esta manera

agilizar el uso de diferentes puntos de vista de diferentes stakeholders para luego unificarlos en una misma idea y llegar a una conclusión mutua por las partes involucradas. Logrando con esto que esta etapa de análisis de requerimientos en el ciclo de vida del software ya nombrado sea mucho más eficiente.

Tareas realizadas para desarrollar este trabajo:

(I) Procesar las kernel sentences ingresadas por los stakeholders.

(II) Dar respuesta a dicho procesamiento a los analistas.

(III) Devolver el resultado a los usuarios.

(IV) Responder en base a la respuesta de los mismos, pudiendo tomar una decisión sobre dicha respuesta ya sea caso positivo o negativo.

Este capítulo muestra como primera medida que para llevar a cabo un ciclo de vida exitoso en la construcción de cualquier software es indispensable haber tenido una etapa de elicitación y validación de requerimientos completa y exitosa. Muestra además que esta etapa dentro del ciclo de vida del software es la más importante de todas y a la que más se le debe poner un foco de atención; porque si se producen errores graves en esta etapa puede resultar en que se obtenga un software erróneo que no satisfaga completamente las necesidades de las partes interesadas.

Algunos de los problemas más habituales en esta etapa del ciclo de vida del software, que ya ha mencionado este capítulo pueden ser, **falta de comunicación** cuando el software es muy grande (suele tener muchas personas involucradas) es posible que no haya una comunicación fluida entre todas estas personas, lo cual hace que muchos desconozcan nuevos requerimientos o cambios en los mismos. **Un requerimiento mal escrito** en muchas ocasiones puede llevar a que el software tenga una mala especificación de requerimientos; esto puede traer confusión entre las partes involucradas a la hora de comprender y analizar los requerimientos.

Otro problema se debe a **diferentes conocimientos del dominio**. Hoy en día, el software esta distribuido en diversas áreas; las personas involucradas en cada área tienen su propio conocimiento sobre el dominio del software. Esto a veces puede traer conflictos cuando se debe tomar alguna decisión importante dé forma global.

Para reducir la ocurrencia de muchos de estos problemas, este capítulo propone el desarrollo de una herramienta llamada Ruems Requeriments, la cual brinda soporte al proceso de relevar y validar requerimientos. Esta herramienta se encarga de convertir los requerimientos escritos o establecidos por los usuarios en oraciones más simples y más comprensibles.

La herramienta Ruems Requeriments consiste en una aplicación web que permite el ingreso de dos tipos de usuarios; por un lado existen usuarios analistas que son los encargados de administrar el proceso de relevación de requerimientos en la herramienta. Por otro lado, están los usuarios corrientes que son

los encargados de relevar y validar los requerimientos.

Ruems Requeriments brinda la posibilidad de trabajar en distintos proyectos de software y además permite que un mismo usuario ya sea analista o usuario corriente participen en varios de ellos.

Los usuarios corrientes pueden registrar nuevos requerimientos a un determinado proyecto de software. Por otro lado los analistas son los encargados de aprobar estos requerimientos o solicitar cambios en los mismos previo a su aprobación, como así también pueden llevar a cabo un control general de trabajo de los usuarios participantes de ese proyecto de software.

1.3 **Aportes de la tesina**

El cuadro 1 representa las líneas generales en donde se describen los aportes realizados de la tesina.

Cuadro 1: Aporte realizado

Aporte realizado
Se desarrolló una aplicación la cual permite el procesamiento de kernel sentences por parte de diferentes stakeholders con distintos puntos de vista, logrando una asistencia en el proceso de evaluación de usabilidad a través de la utilización de la misma que permite generar diferentes puntos de vista a los usuarios que interactúen con la misma logrando así pruebas de usabilidad con posibles usuarios finales
Se evaluó la aplicación por parte de diferentes usuarios para evaluar el proceso de evaluación de usabilidad, logrando una interacción correcta por las mismas partes que están involucradas
Contribuir a mejorar la evaluación de interfaces de usuario mediante la creación de diferentes versiones de interfaz para una misma aplicación
Fomentar la utilización de kernel sentences correctas, como primera etapa del ciclo de vida de un producto de software. Lo ideal es que estas etapas preliminares contribuyan a encontrar cualquier problema de requerimientos o usabilidad mientras está todavía en fase inicial y así ajustar los diseños, estructura, funcionalidad o requerimientos para así poder crear productos de acuerdo a las necesidades para las cuales fueron pensados
Se planteó desarrollar una nueva forma de metodología para lograr una fácil agilización a lo largo del proceso de elitación de requerimientos

2 Background

Este capítulo describe los distintos conceptos que se necesitan para comprender el desarrollo realizado. Se comienza nombrando las metodologías ágiles para así con esto ir introduciendo al lector en el procesamiento de lenguaje natural.

2.1 Metodologías ágiles

En la década de los noventa surgieron metodologías de desarrollo de software ligeras, más adelante nombradas como metodologías ágiles, que buscaban reducir la probabilidad de fracaso por subestimación de costos, tiempos y funcionalidades en los proyectos de desarrollo de software. Existen dos tipos de metodologías, por un lado, las metodologías tradicionales, que buscan imponer disciplina al proceso de desarrollo de software y de esa forma volverlo predecible y eficiente. El principal problema de este enfoque es que hay muchas actividades que hacer para seguir la metodología y esto retrasa la etapa de desarrollo. Las metodologías ágiles tienen dos diferencias fundamentales con las metodologías tradicionales; la primera es que los métodos ágiles son adaptativos no predictivos. (Cuadro 2) La segunda diferencia es que las metodologías ágiles son orientadas a las personas y no orientadas a los procesos [32].

Cuadro 2: Metodologías tradiciones vs metodologías ágiles

Metodologías tradicionales	**Metodologias ágiles**
Predictivos	Adaptativos
Orientados a procesos	Orientados a personas
Proceso rígido	Proceso flexible
Se concibe como un proyecto	El proyecto se divide en proyectos pequeños
Poca comunicación con el cliente	Comunicación constante con el cliente
Entrega de software al finalizar el desarollo	Entregas constantes de software
Documentación extensa	Poca documentación

Las **metodologías ágiles** son flexibles, pueden ser modificadas para que se ajusten a la realidad de cada equipo y proyecto. Los proyectos ágiles sé subdividen en proyectos más pequeños mediante una lista ordenada de características. Cada proyecto es tratado de manera independiente y desarrolla un subconjunto de características durante un periodo de tiempo corto, de entre dos y seis semanas. La comunicación con el cliente es constante al punto de requerir un representante de él durante el desarrollo. Los proyectos son alta-

mente colaborativos y se adaptan mejor a los cambios; de hecho, el cambio en los requerimientos es una característica esperada y deseada, al igual que las entregas constantes al cliente y la retroalimentación por parte de él. Tanto el producto como el proceso son mejorados frecuentemente. Las metodologías ágiles se caracterizan por el desarrollo iterativo e incremental; la simplicidad de la implementación; las entregas frecuentes; la priorización de los requerimientos o características a desarrollar a cargo del cliente; y la cooperación entre desarrolladores y clientes. Las metodologías ágiles dan como un hecho que los requerimientos van a cambiar durante el proceso de desarrollo.

Entre las metodologías ágiles más habituales se encuentra **scrum**.Su nombre no corresponde a una sigla, sino a un concepto deportivo, propio del rugby, relacionado con la formación requerida para la recuperación rápida del juego ante una infracción menor. La metodología scrum para el desarrollo ágil de software es un marco de trabajo diseñado para lograr la colaboración eficaz de equipos en proyectos, que emplea un conjunto de reglas y artefactos y define roles que generan la estructura necesaria para su correcto funcionamiento.Fue creada por Jeff Sutherland [52]. Él afirma que scrum logra que las personas hagan el doble de trabajo en la mitad de tiempo. Los llamados equipos scrum son autogestionados, multifuncionales y trabajan en iteraciones. La autogestión les permite elegir la mejor forma de hacer el trabajo, en vez de tener que seguir lineamientos de personas que no pertenecen al equipo y carecen de contexto. Los integrantes del equipo tienen todos los conocimientos necesarios (por ser multifuncionales) para llevar a cabo el trabajo. La entrega del producto se hace en iteraciones; cada iteración crea nuevas funcionalidades o modifica las que el dueño del producto requiera. Scrum define tres roles: el Scrum master, el dueño del producto y el equipo de desarrollo. El Scrum master tiene como función asegurar que el equipo está adoptando la metodología, sus prácticas, valores y normas; es el líder del equipo pero no gestiona el desarrollo. El dueño del producto es una sola persona y representa a los interesados, es el responsable de maximizar el valor del producto y el trabajo del equipo de desarrollo; tiene entre sus funciones gestionar la lista ordenada de funcionalidades requeridas o Product Backlog. El equipo de desarrollo, por su parte, tiene como responsabilidad convertir lo que el cliente quiere, el Product Backlog, en iteraciones funcionales del producto; el equipo de desarrollo no tiene jerarquías, todos sus miembros tienen el mismo nivel y cargo de desarrollador.

En otra instancia existe otra metodología muy ampliamente utilizada la cual es **Extreme Programming [XP]** .XP es la metodología ágil más conocida. Fue desarrollada por Kent Beck buscando guiar equipos de desarrollo de software pequeños o medianos, entre dos y diez desarrolladores, en ambientes de requerimientos imprecisos o cambiantes. XP tiene como base cinco valores:Simplicidad, Comunicación, Retroalimentación, Respeto y Coraje. Estos valores, a su vez, son la base para la definición de sus principios. De ellos, los fundamentales son: la retroalimentación rápida, asumir simplicidad, el cambio incremental,la aceptación del cambio y el trabajo de calidad. Las prácticas de esta metodología se derivan de sus valores y principios y están enfocadas en

darle solución a las actividades básicas de un proceso de desarrollo, esto es: escribir código, realizar pruebas, escuchar (planear) y diseñar. Las prácticas de XP incluyen: planning game, pequeñas entregas, diseño simple, programación en pareja, pruebas,refactoring, integración continua, propiedad común del código, paso sostenible, cliente en sitio, metáfora y estándares de código.

Como mencionó este capítulo anteriormente, las metodologías ágiles más utilizadas actualmente son Scrum y XP; sin embargo, las metodologías ágiles no se limitan únicamente a estas dos sino que incluyen varias más. Existen además otras metodologías ágiles las cuales son utilizadas pero no tan conocidas como las ya mencionadas. Entre estas podemos encontrar a Kanban,Agile Inception, Design Sprint,DSDM,Crystal methods,ASD Adaptative Software Development y FDD.

Aplicación de metodología ágil en Ruems Requeriments

Luego de haber analizado las metodologías ágiles más utilizadas hoy en día (scrum,xp), se puede decir que Ruems Requeriments se aproxima más al trabajo de una metodología ágil de tipo scrum. Hay varias razones que llevan a garantizar esta afirmación.Entre lo más notable se puede ver el **trabajo colaborativo**. El capítulo detalló anteriormente que en scrum se fomenta mucho la colaboración entre los miembros del equipo y el cliente. Ruems Requeriments brinda la posibilidad de tener varios equipos con distintos roles de personas en constante colaboración. Esto quiere decir que entre los distintos roles se ayudan mutuamente para llevar a cabo las tareas del equipo.

También se destaca la **comunicación**. Ruems requeriments permite una comunicación fluída entre clientes y usuarios de un mismo equipo, para desempeñar las tareas del grupo y satisfacer correctamente el trabajo colaborativo.

Finalmente, se menciona el muy notorio rol del **Scrum Master**. Se destaca que en scrum existe un usuario con el rol de scrum master el cual es el encargado de liderar al equipo. En Ruems Requeriments puede haber varios usuarios con este mismo rol en un mismo equipo. Estos usuarios son los que están en constante comunicación con los demás miembros y clientes del equipo; y además administran el trabajo de todo el equipo.

2.2 Procesamiento de lenguaje natural

El procesamiento del lenguaje natural consiste en el estudio y análisis de los aspectos lingüísticos de un texto a través de programas informáticos. Un sencillo ejemplo de PLN no es más que un corrector ortográfico de un procesador de textos que todos hemos empleado alguna vez.

Menciona Verdejo [56] que el lenguaje natural se distingue de los lenguajes artificiales por su riqueza (en vocabulario y construcciones), flexibilidad (reglas con múltiples excepciones), ambigüedad (pudiendo darse diversos significados de una palabra o una frase según el contexto),indeterminación (permitiendo referencias y elipsis) y posibles interpretaciones del sentido literal según la

situación en que se produce. Lo que son ventajas para la comunicación humana se convierten en problemas a la hora de un tratamiento computacional, ya que implican conocimiento y procesos de razonamiento que aún no sabemos ni cómo caracterizarlos ni cómo formalizarlos.

Hay que señalar que PLN surge en la década de los cincuenta, y su historia se entrelaza con las investigaciones que sobre el lenguaje se llevan a cabo en otras disciplinas, principalmente lingüística formal, psicología cognitiva, lógica y la informática, pero sobre todo, la inteligencia artificial, dando lugar a una disciplina denominada lingúísíica computacional. La lingüística computacional es la intersección de la lingüística y la informática con el fin de procesar o generar las lenguas.

Modelos de Procesamiento de lenguaje natural

Existe mucha información de la Inteligencia Artificial, hoy en día aún se está investigando distintas ramas de la misma, entre ellas el Procesamiento de Lenguaje Natural, sobre todo por los dos grandes problemas que este procedimiento conlleva, los cuales son la ambigüedad y la dimensionalidad de los textos. Es por ello que para lograr llegar a un entrenamiento es importante conocer bajo qué modelo se realizaría el procesamiento,existen dos modelos que se mencionan a continuación:

Modelos lógicos

Noam Chomsky estableció ya hace 70 años las bases de la estructura lógica del lenguaje. Este concepto ha mantenido su validez a lo largo de décadas gracias a su interpretación fidedigna de los esquemas que usamos al comunicarnos. En la actualidad, los modelos lógicos de procesamiento del lenguaje natural recogen su esencia para aplicarla a la comunicación entre máquina y persona.

Los modelos del PLN lógicos son los creados por los lingüistas especializados basándose en determinadas formas gramaticales. De esta manera, es posible que las máquinas puedan reconocer diferentes patrones de estructuras lingüísticas.

Esto se combina con la información que ya existe en los diccionarios de computación para poder definir los modelos de resolución de una tarea concreta a través del proceso de automatización de los procesos conversacionales.

Modelos probabilísticos

Son mencionados como basados en corpus, basándose en datos, donde su esencia es el análisis de la información recopilada lingüísticamente, se pueden tomar de muchas formas como las grabaciones, registros, en la actualidad son tomados desde las conversaciones de las redes sociales, un conjunto infinitos de enunciados gramaticalmente, donde sus términos, reglas gramaticales,fonemas, palabras, frases etc. Donde ese lenguaje es tomado para encontrar un algoritmo que determine a qué grupo pertenece el lenguaje, es decir que reconozca el lenguaje.

Para ello se es empleada con la utilización de estructura gramatical, que pueden ser de forma secuencial como de pares o de tríos, aquí también se incluyen la estadística del uso de la regla de gramática. El enfoque de los diferentes algoritmos probabilísticos se describen como N-gram,Unigram,Bigram,Trigram [45].

El cuadro 3 resume las características más destacables de los modelos de procesamiento de lenguaje natural.

Cuadro 3: Modelo de procesamiento de lenguaje natural

Lógicos (gramaticales)	**Probabilísticos (basado en corpus)**
Se basan en determinadas formas gramaticales	Se basan en los datos como eje principal del análisis
Son creados por lingüistas especializados	los lingüistas recopilan información para analizar y establecer la frecuencia de aparición de todas las unidades que forman la lengua
Las máquinas pueden reconocer diferentes patrones de estructuras lingüísticas	Los algoritmos pueden establecer la probabilidad de que aparezcan en un contexto
Identifica acrónimos existentes,seleccionando los tokens de oraciones o palabras que deseen	Recopilan una cantidad de información para analizar y establecer la frecuencia de aparición de todas las unidades
Se combina con la información que ya existe en los diccionarios de computación para poder definir los modelos de resolución de una tarea concreta a través del proceso de automatización de los procesos conversacionales	Se realiza sin tener que valorar los requisitos establecidos en las reglas gramaticales en ningún momento
Recogen su esencia para aplicarla a la comunicación entre máquina y persona	Esencia, de lo que se denomina aprendizaje automático en inteligencia Artificial

Herramientas de procesamiento de lenguaje natural

Una herramienta computacional servirá para estructurar textos en la medida en que sea capaz de extraer la información del texto (tareas de NER y Tagging), para lo cual resulta necesario comprender el significado de los términos en el texto. También permite devolver la información de manera estructurada en base a las necesidades del usuario.

Las tareas más importantes para extraer información relevante de un texto son Tokenization, NER, POS tagging y RI. La estructuración de textos pasa por reconocer entidades relevantes (NER), clasificarlas (etiquetado) e identificar las relaciones entre ellas (RI) ası como la información relacionada con cada entidad. NER y TAGGING se pueden comprender como una secuencia de etiquetado (frase=conjuntos de palabras/tokens). Por otro lado RI se utiliza cuando una vez detectadas las entidades y realizado su etiquetado, se deben detectar las relaciones entre estas.

Librerías de procesamiento de lenguaje Natural

Se entiende como librería o biblioteca el conjunto de implementaciones funcionales codificadas en un lenguaje de programación, que ofrece una interfaz bien definida para la función que se invoca. Esta pretende ser utilizada de forma simultánea por distintos programas independientes.

En programación, se define paquete como una agrupación de clases e interfaces relacionadas. Estas clases son ocultas y nadie ajeno al sistema puede conocer su funcionamiento interno.

Algunas de las librerías o programas que se pueden descargar para estructurar textos son Spark NLP;Spacy;Deep Pavlov; Sci Lite Annotations; como las más utilizadas.

Dos de los artefactos utilizados actualmente para especificar requerimientos son: (i) Use Cases y (ii) User Stories. Use Cases proveen una descripción de las acciones de un sistema desde el punto de vista del usuario. La funcionalidad del sistema es modelada usando actores que hacen acciones sobre el sistema. Los Use Cases son servicios o funciones provistas por el sistema para sus usuarios. User Stories por su parte, son una descripción breve de una funcionalidad de software tal y como la percibe el usuario [15]. Tanto los Uses Cases como las User Stories usan oraciones simples en sus descripciones. Harris [27] definió a estas oraciones con el nombre de Kernel Sentences.Las mismas se caracterizan por tener un solo verbo, ser afirmativas, activas, declarativa,no contiene ningún estado de ánimo. Se denominan kernel ya que son la base sobre la cual se forman otras oraciones más complejas. La actividad de la especificación de kernel sentences se compone de tres pasos: (i) la descripción, (ii) la verificación sobre si realmente es una oración del kernel, y (iii) la revisión para identificar si es significativa para el alcance del sistema de software.

El primer paso, la descripción de kernel sentences, es realizado por algún experto que especifica una kernel sentences. El experto se convierte en el autor de la misma. Es importante rastrear cada oración del kernel hasta su autor porque los otros expertos (diferentes del autor) se deben validar si están de acuerdo o no con la contribución. Además, la identificación del autor también es relevante para supervisar su actividad.

El segundo paso, el paso de verificación, comprueba si la kernel sentences es realmente una oración del kernel. También proponemos alguna revisión adi-

cional para asegurarnos de que la contribución del experto sea lo más simple posible.

La primera verificación consiste en comprobar que la oración tiene la estructura: sujeto + verbo + objeto para comprobar que tiene un solo verbo y está escribiendo en voz activa. Algunos ejemplos de esta verificación son: Los tomates son fertilizados por el agricultor (voz pasiva, no correcta);es necesario fertilizar los tomates (sujeto nulo, no correcto); El agricultor fertiliza y riega los tomates (dos verbos, no correctos);El agricultor fertiliza los tomates (correcto).

La segunda verificación consiste en comprobar la presencia de conjunciones. Existen diferentes tipos: (i) conjunciones coordinadoras como: y, o, para, pero, etc., (ii) conjunciones correlativas como: no solo, sino también, ya sea, ninguna, etc. (iii) y conjunción subordinante como: después, siempre y cuando, si sólo, dónde, según, etc. La presencia de las conjunciones no determina que la contribución no sea una oración con núcleo. Sin embargo, podría proporcionar una pista de que una gran cantidad de información se describe en una sola oración. Es importante mencionar que el siguiente ejemplo (**El agricultor evalúa la humedad del suelo**) proporciona el conocimiento implícito en la segunda oración (**El agricultor riega los tomates de acuerdo con la humedad del suelo**). Es decir, **según** significa que el agricultor debe **evaluar la humedad**. Además, el agricultor tiene algunos criterios para decidir cuándo la humedad es suficiente y no es necesario regar. Dicho ejemplo del agricultor lo situamos en el cuadro 4, junto con los detalles analizados.

Cuadro 4: Ejemplo del agricultor

Oración
El agricultor fertiliza y riega los tomates (conjunción **y** para expresar dos verbos)
El agricultor riega los tomates de acuerdo con la humedad del suelo (correcto, pero **de acuerdo con** sugiere la presencia de más conocimiento)
El agricultor evalúa la humedad del suelo (correcto, se infiere del anterior)

La tercera verificación consiste en comprobar la presencia de adjetivos y adverbios. Aunque su presencia no determina que la contribución no sea una oración del núcleo, este tipo de palabras caracterizan sustantivos y verbos, y su presencia podría proporcionar una pista de que se podría agregar más información en otra oración del núcleo.

3 **Trabajos relacionados**

Este capítulo analiza en líneas generales las principales investigaciones que se realizaron sobre lineamientos ideales que concluyeron en él desarrolló de la aplicación Ruem Requerimients, como así también aplicaciones que utilizan el procesamiento de lenguaje de procesamiento natural para el procesamiento de elitación de requerimientos las cuales algunas presentan ciertas falencias que las mismas fueron analizadas y contempladas. Hoy en día existe una amplia variedad de aplicaciones de procesamiento de lenguaje natural para utilizar, en el apartado de 3.1 aplicaciones relacionadas se mencionan las herramientas más utilizadas junto con los beneficios que proporcionan cada una de estas y las diferentes características que las hacen únicas en su función.

3.1 **Aplicaciones relacionadas**

Pensando en el proceso de implementación de Ruems Requeriments este capítulo recurre a diferentes ideologías por parte de distintos autores, tal fue el caso de Garner que afirma lo importante que es la interacción humana y la asociación en actividades de resolución de problemas como el desarrollo de software [11]. Se destaca que la iteración entre diferentes usuarios por medio de una aplicación es crucial para un buen desarrollo de software; esto llevo a pensar una idea de una construcción colaborativa y una validación del conocimiento.

Se plantea además como base para el desarrollo de la herramienta; la herramienta basada en **StakeRare** planteada por **Vijayan** [31] en la cual nombra que es importante la idea de elicitar el conocimiento del dominio. La misma herramienta plantea la construcción de una red de partes interesadas en la cual plantean sus ideas para luego de estas mismas obtener un conocimiento concreto. Se toma esta idea de red de usuarios las cuales todas las partes interesadas pueden interactuar y dejar sus ideas planteadas para luego ser debatidas. No obstante se sigue la idea planteada además de Meng [25] la cual es de alto rendimiento y valioso la voluntad de intercambio de conocimientos entre diferentes usuarios.

La idea planteada por **Unkelos** [26] termina de forjar el corazón de la herramienta la cual fue la de una herramienta de colaboración,ya que proponen un proceso de ingeniería de requisitos colaborativo y gamificado. De este mismo autor se toman los tipos de roles que va a tener la herramienta, el mismo propone tres roles que se detallan en el cuadro 5 los cuales están relacionados con los roles propuestos.

De **Goncalves** [16] se toma la idea de la importancia de las reglas de negocio para consolidar el conocimiento del dominio. Por lo que se propone en la herramienta 3 reglas que deben cumplir los usuarios para poder presentar sus ideologías.

Cuando se busca aplicaciones relacionadas con el procesamiento de lengua-

Cuadro 5: Roles planteados

Roles
El creador (es el experto en nuestro enfoque)
El revisor (es el experto en nuestro enfoque, ya que revisa el conocimiento)
El cliente (es el analista en nuestro enfoque, ya que va a consumir el conocimiento obtenido)

je natural se encuentran diferentes opciones muy diversas que tienen desde casos puntuales de filtros de corrector de textos hasta asistentes inteligentes o textos predictivos. Un caso particular que nos da ciertos lineajes para crear Ruems Requeriments fue **Text2LEL** [13].

Text2LEL es una herramienta diseñada para identificar símbolos e impactos, que son elementos de un glosario LEL, a partir de información textual proporcionada por el usuario. Aunque esta idea sirvió como punto de partida, hemos añadido la posibilidad de que los usuarios tengan la libertad de determinar si el texto ingresado es válido para sus propios propósitos.

Luego con el ejemplo ya de una herramienta similar concreta se generó la idea de intercambio colaborativo y se concluyo en una herramienta muy conocida y a su vez muy utilizada habitualmente en varios ambientes de trabajo para diferentes causas conocida como **Google Docs**.

Google Docs [47] es una herramienta de procesamiento de textos en línea que permite a los usuarios crear y editar documentos de manera colaborativa. Esta herramienta inspira en cuanto a la capacidad de múltiples usuarios para interactuar e intercambiar ideas constantemente sobre un mismo documento. En contraste a Ruems Requerimients presenta funcionalidades limitadas, como la necesidad de conexión a internet para su uso completo. Además de que no ofrece una guía concreta para ayudar al usuario a corregir su enfoque sobre el texto o las kernel sentences planteadas originalmente.

Otra herramienta conocida que es utilizada y la cual brinda ciertas ideas es **Etherpad** [1]. Es una herramienta de edición de texto en tiempo real que permite a múltiples usuarios colaborar simultáneamente en documentos de texto, además de publicarlos en ciertos sitios web. Sin embargo, una desventaja que presenta en comparación con Ruems Requerimients es que Etherpad impone ciertas restricciones en algunos documentos, ya que estos pueden expirar después de un tiempo determinado. Otra gran desventaja es la falta de seguridad si no se utiliza en un entorno controlado, y el mecanismo para compartir texto o kernel sentences con otros usuarios se limita a compartir una URL, lo cual no resulta muy conveniente.

Con las ideas de concretar Ruems Requerimients, se detalló en la iteración

entre usuarios y o usuario y analistas del sistema por lo que se tuvó en cuenta el ejemplo de la herramienta **Quora** [57]. Es una plataforma de preguntas y respuestas donde los usuarios pueden plantear preguntas y obtener respuestas de la comunidad. Este sistema destaca por el intercambio de preguntas y respuestas, que en Ruems Requeriments se convirtieron en kernel sentences, y la participación de expertos, lo que permite validar los perfiles de los usuarios y sus kernel sentences, otorgando mayor credibilidad. Sin embargo, una desventaja es que algunos usuarios expertos pueden usar esta plataforma para autopromocionarse, algo que no ocurre con Ruems Requerimients. Además, la calidad de las respuestas puede variar considerablemente y extenderse exponencialmente.

Con la idea de que Ruems Requerimients tenga discusión entre diferentes kernel sentences se indagó hasta dar con la herramienta de vBulletin. **VBulletin** [23] es un software de foros comerciales que ofrece una plataforma robusta para discusiones en línea. Permite la creación de foros específicos para discutir frases núcleo relacionadas con distintos aspectos de un proyecto. Por ejemplo, una empresa podría utilizar vBulletin para crear un foro interno donde los equipos de desarrollo, pruebas y negocios discutan frases núcleo que describen los requisitos del sistema. Los usuarios pueden responder con preguntas y sugerencias para asegurar que los requisitos sean claros y precisos. Sin embargo, al igual que otras herramientas, vBulletin es una plataforma comercial que requiere la compra de una licencia para soporte técnico y actualizaciones, además de ciertos requisitos del servidor para funcionar de manera óptima.

Stack Exchange [59] es un sitio web de preguntas y respuestas sobre diversos temas, conocido por su formato estructurado y su enfoque en respuestas de calidad. Un aspecto particular que encontramos útil como referencia es la posibilidad de votaciones por parte de los usuarios, donde las mejores contribuciones son destacadas. Sin embargo, una gran desventaja de Stack Exchange, que consideramos al diseñar nuestra herramienta, es su comunidad exclusiva, lo que puede desalentar la participación de nuevos usuarios. Además, las reglas restrictivas y difíciles de seguir de Stack Exchange sirvieron como guía para evitar esos problemas en nuestra aplicación.

Answerbag [60] es un sitio web de preguntas y respuestas que permite a los usuarios formular preguntas sobre una amplia variedad de temas y recibir respuestas de la comunidad. Se destaca de esta herramienta su capacidad para abordar tanto preguntas muy generales como otras más específicas, además de las votaciones y comentarios en las respuestas. Este sitio web inspiró a ampliar la gama de consejos que deben seguir los usuarios al ingresar las kernel sentences en Ruems Requerimients. Sin embargo, un problema de Answerbag es que no es intuitivo y tiene un diseño anticuado, a diferencia de Ruems Requerimients.

Spiceworks [58] es una plataforma web de preguntas y respuestas orientada a profesionales de TI y administradores de sistemas. Lo que distingue a esta aplicación es su enfoque exclusivo en profesionales con experiencia, ya que

se basa en preguntas y respuestas muy específicas. Esto inspiró en diseñar a Ruems Requerimients con una división de proyectos, permitiendo que cada usuario se ubique en el área o proyecto correspondiente y formule kernel sentences basadas en su conocimiento. Sin embargo, la principal desventaja de Spiceworks es que puede resultar compleja para usuarios no técnicos, un aspecto considerable al generalizar a Ruems Requeriments para que sea accesible a una audiencia más amplia.

3.2 **Conclusiones sobre las aplicaciones relacionadas**

A diferencia de estas aplicaciones, el sistema de Ruems Requeriments permite hacer un seguimiento interactivo entre los diferentes usuarios en los cuales ellos mismos, con ayuda de Ruems Requeriments, terminan corrigiendo ciertas ideas incorrectas. Esta funcionalidad logra mejorar la comunicación y la colaboración dentro del equipo, y a su vez, aumenta la eficiencia en la gestión de requisitos. El impacto de Ruems Requeriments no se define por la limitación de corrección de ideas incorrectas, si no que también promueve una cultura de aprendizaje y una mejora continua. Este sistema permite introducir a los usuarios en un proceso de formación de la cual estos mismos aprenden de sus errores, por lo cual dicho proceso culmina en un resultado de conocimientos acumulativo, que beneficia a todo el equipo. En resumen, Ruems Requeriments ofrece una solución integral que va más allá de la simple gestión de requisitos. Su enfoque interactivo y colaborativo, junto con su capacidad de integración y adaptabilidad, lo posiciona como una herramienta esencial para equipos de desarrollo que buscan mejorar la eficiencia, la comunicación y la calidad de sus proyectos.

4 **Método de relevamiento y validación de kernel sentences propuesto en esta tesina**

La ingeniería de requerimientos es, por tanto, una actividad esencialmente de interacción con los interesados en el sistema. Es incorrecto y extremadamente riesgoso que los Ingenieros de Software establezcan los requerimientos del sistema sin haber mantenido numerosas reuniones con diferentes representantes del cliente, sin haberles mostrado prototipos del sistema, sin haberles hecho una y otra vez las mismas preguntas,sin haber comprendido el negocio del cliente.

Como se detalló en el capítulo uno de la tesina,un requerimiento es una característica del sistema o una descripción de algo que el sistema es capaz de hacer, con el objeto de satisfacer el propósito del sistema.

No es lo mismo una solicitud de un usuario o cliente que un requerimiento. No todas las solicitudes de un usuario o cliente se convierten necesariamente en requerimientos, pero sí todos los requerimientos se originan en una solicitud de un usuario o cliente. Para que una solicitud de un usuario o cliente se convierta en requerimiento, esta debe ser o estar documentada apropiadamente y él solicitante debe validarla.

Los ingenieros de software no originan los requerimientos; su función es convertir solicitudes de los usuarios o clientes en requerimientos. Luego, deben proveer un sistema que los implemente.

Los requerimientos siempre están en el entorno del sistema que se va a desarrollar, nunca dentro de él.

Luego de este análisis introductorio sobre la ingeniería de requerimientos se concluye, en términos generales, que la ingeniería de requerimientos es el proceso de recopilar, analizar y verificar las necesidades del cliente para un sistema de software.La meta de la ingeniería de requerimientos es entregar una especificación de requerimientos de software correcta y completa. La ingeniería de requerimientos apunta a mejorar la forma en que comprendemos y definimos sistemas de software complejos, a continuación se detallan los procesos de la ingeniería de requerimientos [10].

4.1 **Procesos de la ingeniería de requerimientos**

El proceso del establecimiento de requerimientos de un sistema de software,(ya mencionado), es el primer paso esencial en entregar lo que el cliente desea. A pesar de esto, la insuficiencia de tiempo y esfuerzo son a menudo encontrado en esta actividad y existen pocos métodos sistemáticos para soportarlo. Entre los métodos conocidos se puede citar a los siguientes: según Pressman el proceso de análisis de requerimientos del software se puede identificar en cinco tareas o etapas fundamentales [41].

Reconocimiento del problema.En esta etapa, la función primordial del ana-

lista en todo momento es reconocer los elementos del problema tal y como los percibe el usuario.

Evaluación y síntesis. En esta etapa el analista debe centrarse en el flujo y estructura de la información, definir las funciones del software, determinar los factores que afectan el desarrollo de nuestro sistema, establecer las características de la interfaz del sistema y descubrir las restricciones del diseño. Todas las tareas anteriores conducen fácilmente a la determinación del problema de forma sintetizada.

Modelización. Durante la evaluación y síntesis de la solución, se crean modelos del sistema que servirán al analista para comprender mejor el proceso funcional, operativo y de contenido de la información.

Especificación. Las tareas asociadas con la especificación intenta proporcionar una representación del software.

Revisión. Una vez que se han descrito la información básica, se especifican los criterios de validación que han de servir para demostrar que se ha llegado a un buen entendimiento de la forma de implementar con éxito el software.

El proceso de ingeniería de requerimientos en el que se basará este trabajo de tesina es el que propone Loucopoulos [32], en el que se plantea que en esta fase hay que considerar por lo menos tres aspectos fundamentales como es **comprender el problema**, **describir formalmente el problema** y **obtener un acuerdo sobre la naturaleza del problema**.

Esto lleva a simplificar el proceso a tres etapas para obtener los requerimientos del problema que se intentan abordar.**Elicitación de requerimientos,** cuyo propósito es ganar conocimientos relevantes del problema, que se utilizarán para producir una especificación formal del software necesario para resolverlo.Existen diversas técnicas para llevar a cabo esta etapa del proceso (entrevistas,cuestionarios,etc).**Especificación** en la que una especificación puede ser vista como un contrato entre usuarios y desarrolladores de software, que define el comportamiento funcional deseado del artefacto de software sin mostrar como será alcanzada tal funcionalidad. El capítulo uno de esta tesina muestra que, para esta etapa se dispone de diversas herramientas para especificar requerimientos. Alguna de ellas son casos de uso e historias de usuario.**Validación** es el proceso que certifica que el modelo de los requerimientos es consistente con las intenciones de los clientes y los usuarios; esta es una visión más general que el concepto común de validación, pues se produce en paralelo con la elicitación y la especificación, tratando de asegurar que tanto las ideas como los conceptos presentados en una descripción se identifican y explican con claridad. La necesidad de validación aparece cuando se incorpora una nueva pieza de información al modelo actual. O bien cuando diferentes piezas de información se incorporan en un todo coherente.

En la figura 1, se observa un esquema del proceso de ingeniería de requerimientos planteado por Locopoulos [32].

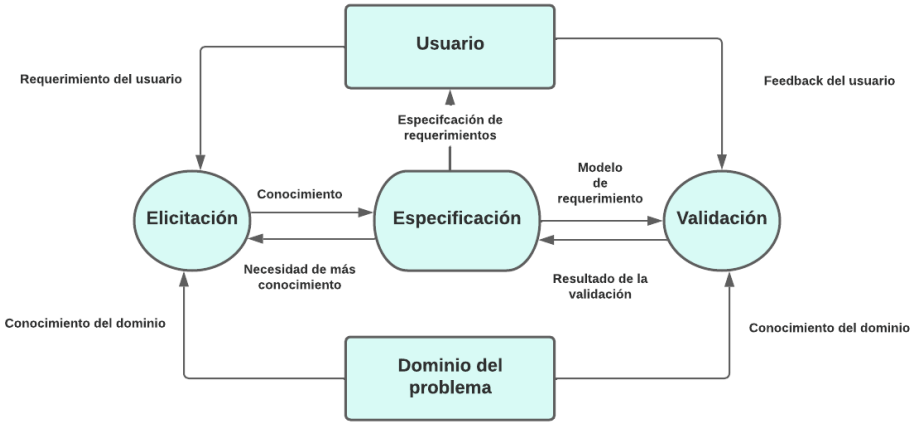

Figura 1: Esquema del proceso de ingeniería de requerimientos (Fuente: Loucopoulos)

4.2 **Kernel sentences**

Una kernel sentences no es más que una construcción simple con un solo verbo que puede estar en vos pasiva, activa o declarativa. Este tipo de oraciones se las denomina como núcleo, ya que son la base para la construcción de oraciones más complejas. Por ejemplo tomemos dos posibles ejemplos: **el agricultor fertiliza los tomates** y **el agricultor riega los tomates cuando hace calor**. En la primera oración o frase se denota que el sujeto agricultor realiza una acción que es fertilizar sobre un determinado objeto tomate. La segunda oración tiene la misma estructura, pero describe una acción diferente y también agrega la descripción sobre “cuándo” se realiza la acción. Con esto es importante mencionar que el verbo ser no tiene un significado semántico y esto se debe porque la segunda oración tiene 2 verbos regar y el verbo hace. La segunda oración demuestra que no es una kernel sentences, ya que esta completa por más de un verbo. Este ejemplo muestra cómo la oración original con dos verbos saca una conclusión sobre la primera oración del agricultor que fertiliza y agrega nutrientes. Por más detalle ver **anexo 2** [17].

4.3 **Método a dar soporte con la herramienta**

El enfoque consiste en obtener de forma colaborativa kernel sentences de los expertos del dominio para consolidar el conocimiento del dominio de la aplicación. Se lo menciona como proceso colaborativo porque pueden participar muchos expertos. Al mismo tiempo. Las personas contribuyen con diferentes oraciones centrales, ya que las partes interesadas pueden tener diferentes puntos de vista sobre el dominio. Y esto Una visión complementaria y muy impor-

tante para producir una visión integrada y completa. El proceso considera dos roles diferentes: **expertos del dominio** y **analistas**. Los expertos del dominio proporcionan las kernel sentences y validan a las mismas las cuales pueden ser propuestas por otros expertos. Por lo tanto, los analistas pueden identificar oraciones centrales que no pertenecen al dominio y eliminarlos del proceso. Finalmente, los analistas monitorean la actividad del experto identificando expertos con algún comportamiento particularmente sesgado. Por ejemplo, un experto que acepta como válida cada kernel sentences o que rechaza como inválidas todas ellas no está comprometido con la actividad y su contribución debe ser omitida, ya que no es confiable. Así, el proceso propuesto considera tres actividades diferentes: (i) especificación de kernel sentences, (ii) validación de kernel sentences y (iii) evaluación de expertos. Estas tres actividades se realizan en paralelo. Así, mientras se especifican oraciones, otras oraciones pueden ser validadas y, al mismo tiempo, se puede evaluar a los expertos.

La primera actividad (especificación de kernel sentences) se basa en la definición de las mismas por algún experto (quien se convierte en su autor). Luego, se debe realizar alguna revisión para verificar que la oración cumple con las condiciones para ser considerada una kernel sentences (desde la perspectiva gramatical). Finalmente, algún analista debe revisar el conocimiento expresado por la kernel sentences para determinar si es valioso o no para la descripción del dominio. La segunda actividad (validación de kernel sentences) se basa en recopilar la opinión (acuerdo) de los expertos (diferentes del autor) sobre las kernel sentences de la primera actividad. Finalmente, el analista decide si acepta una kernel sentences como válida, basándose en las opiniones de los expertos. La tercera actividad (evaluación de expertos) se basa en monitorear las contribuciones de los expertos para identificar cuán confiable es algún participante. Si una persona no es confiable, el analista debe excluir a la persona de la actividad, así como sus contribuciones. Para más detalle ver **anexo 2** [17].

5 Arquitectura propuesta en esta tesina

Este capítulo presenta la arquitectura de la herramienta, describiendo los componentes fundamentales que la conforman y cómo interactúan entre sí. Se aborda la definición y el rol de un cliente web, que actúa como la interfaz principal de usuario, permitiendo la interacción directa con la herramienta. También se detalla la función de los servicios, que son responsables de procesar las solicitudes del cliente y gestionar la lógica de negocio.

Además, se explora el concepto de API (Interfaz de Programación de Aplicaciones), que sirve como un puente de comunicación entre el cliente y los servicios, garantizando que los datos se intercambien de manera eficiente y segura. Finalmente, se enumeran y explican las tecnologías utilizadas en el desarrollo de la herramienta, destacando su contribución a la robustez, escalabilidad y rendimiento del sistema.

Cliente web es una interfaz de programación que integra diferentes servicios con una aplicación web o móvil que se intercomunican por medio de un protocolo rest. Por medio de este protocolo, el cliente web se puede comunicar con un servicio externo para cumplir con las necesidades del usuario.

La interfaz de usuario es el medio con que el usuario puede comunicarse con una máquina, equipo, computadora o dispositivo, y comprende todos los puntos de contacto entre el usuario y el equipo. Las interfaces básicas de usuario son aquellas que incluyen elementos como menús, ventanas, contenido gráfico, curso y en general, todos aquellos canales por los cuales se permite la comunicación entre el ser humano y la computadora.

El cliente web/interfaz de usuario de la aplicación Ruems Requeriments es react js.

Servicio es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre diferentes aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos, la mayoría se intercambian por medio del protocolo HTTP.

El término **API** es una abreviatura de Application Programming Interfaces, que en español significa interfaz de programación de aplicaciones. Se trata de un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones, permitiendo la comunicación entre dos aplicaciones de software a través de un conjunto de reglas.

Así pues, podemos hablar de una API como una especificación formal que establece cómo un módulo de un software que se comunica o interactúa con otro para cumplir una o muchas funciones. Todo dependiendo de las aplicaciones que las vayan a utilizar, y de los permisos que les dé el propietario de la API a los desarrolladores de terceros.

Hay un tipo en particular de API la cual se la denomina **API rest**, que es la

abreviación de Representational State Transfer, es un conjunto de restricciones que se utilizan para que las solicitudes HTTP cumplan con las directrices definidas en la arquitectura. Las restricciones determinadas por la arquitectura Rest se detallan en el siguiente cuadro 6.

Cuadro 6: Restricción arquitectura rest

Restricción	Descripción
Cliente-servidor	Las aplicaciones existentes en el servidor y el cliente deben estar separadas
Sin estado	Las requisiciones se realizan de forma independiente, es decir, cada una ejecuta solo una determinada acción
Caché	La API debe utilizar la caché para evitar llamadas recurrentes al servidor
Interfaz uniforme	Agrupa otros cuatro conceptos en los que se determina que los recursos deben ser identificados, la manipulación de los recursos debe ser a través de la representación, con mensajes autodescriptivos y utilizando enlaces para navegar por la aplicación

Luego, cuando se habla de Rest API, significa utilizar una API para acceder a aplicaciones back-end, de manera que esa comunicación se realice con los estándares definidos por el estilo de arquitectura Rest.

En la aplicación Ruems Requeriments la api rest creada es Web API rest symfony.

Procesamiento de lenguaje natural

El procesamiento del lenguaje natural (NLP, por sus siglas en inglés) es una rama de la inteligencia artificial que ayuda a las computadoras a entender, interpretar y manipular el lenguaje humano. El procesamiento del lenguaje natural funciona a través del aprendizaje automático (ML o machine learning). Los sistemas de aprendizaje automático almacenan las palabras cargas por el usuario por medio de un formulario de una interfaz gráfica. Frases, oraciones y a veces libros enteros se introducen en los motores de ML donde se procesan en base a reglas gramaticales, los hábitos lingüísticos de la vida real de la gente, o ambos. El ordenador utiliza estos datos para encontrar patrones y extrapolar lo que viene después. Tomemos el software de traducción, por ejemplo: En francés, Voy al parque se traduce como Je vais au parc, así que el aprendizaje automático predice que voy a la tienda también comenzará con Je vais au. Todo lo que la computadora necesita después de eso es la palabra para tienda.

Funcionamiento de Ruems Requeriments

La aplicación Ruems Requerimentes en principio levanta en uno de sus puertos el sistema con la interfaz gráfica que ve el usuario. Este mismo está conformado por el framework de react js. En paralelo se inicia en otro puerto el servicio web api rest symfony. Este levanta todos los servicios necesarios para su interacción con el cliente react. Una vez que el cliente react js y el servicio web api rest están paralelamente trabajando, se comunican mediante el protocolo HTTP; por lo general el cliente react js envía una request HTTP al servicio web y este le devuelve un json como respuesta, utilizando métodos tales como get,post,put y delete del protocolo HTTP. A su vez el web Api Rest symfony debe estar constantemente interactuando con la base de datos para poder responder las peticiones solicitadas por el cliente react. Esta interacción se lleva a cabo mediante el orm de symfony doctrine. Este orm permite mapear los datos de la base de datos a objetos de la aplicación. El web api rest symfony trabaja con estos objetos y convierte las respuestas al cliente en objetos json. La aplicación Ruems Requeriments se comunica además con un servidor externo de procesamiento de lenguaje natural. El web api rest symfony se comunica con este servicio externo mediante el método post de HTTP y en el cuerpo de la solicitud envía un objeto json de kernel sentences. Este objeto será recibido por el servicio externo y por cada kernel sentences se valida si la misma cumple con las reglas del sistema. Luego, el servicio retorna la respuesta a el web api rest symfony y este último le da la respuesta final al cliente. Todo este procesamiento de solicitarle peticiones al web api rest se realiza con el fin de cargar los formularios para que el usuario pueda utilizar todas las funcionalidades que Ruems Requerimentes provee. La figura 2 muestra un diagrama que detalla la arquitectura de nuestra herramienta [49].

SpaCy es una biblioteca gratuita de código abierto para el procesamiento avanzado del lenguaje natural (NLP) en Python. Al trabajar con muchos textos,se utiliza spaCy el cual está diseñado específicamente para crear aplicaciones que procesan y **entienden** grandes volúmenes de texto. Se puede usar para construir sistemas de extracción de información o comprensión del lenguaje natural, o para pre procesar texto para aprendizaje profundo. Esta biblioteca permite analizar las especificaciones producidas por los stakeholders dentro de Ruems Requeriments [54].

Las características de Spacy aplicadas en Ruems Requeriments son la de **Tokenization** que permite segmentación de texto en palabras, signos de puntuación, etc. **Part-of-speech (POS) Tagging** que realiza la asignación de tipos de palabras a tokens, como verbo o sustantivo. **Dependency Parsing** El cual realiza la asignación de etiquetas de dependencia sintáctica, que describen las relaciones entre tokens individuales, como sujeto u objeto. **Lemmatization** que permite la asignación de las formas básicas de las palabras. Por ejemplo, el lema de “fue“ es “ser“, y el lema de “ratas“ es “rata“. **Sentence Boundary Detection (SBD)** Para encontrar y segmentar oraciones individuales. **Named Entity Recognition (NER)** Con el que etiquetamos objetos del “mundo real“

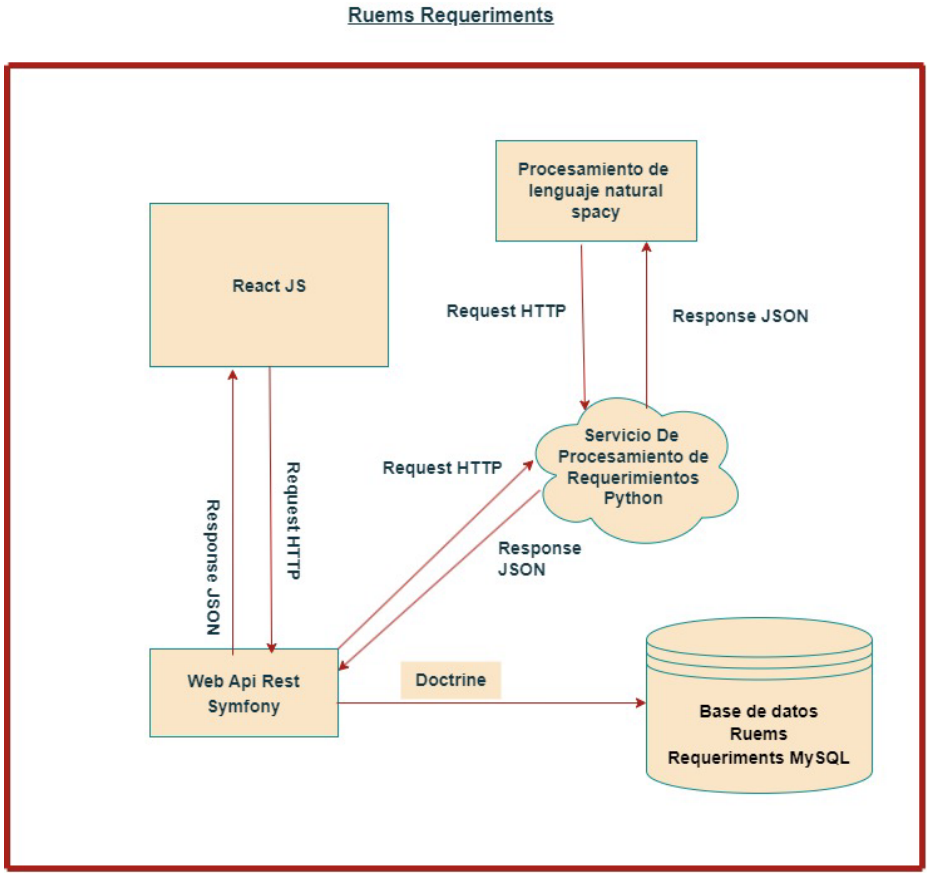

Figura 2: Arquitectura de la herramienta

con nombre, como personas, empresas o ubicaciones. **Entity Linking (EL)** permite la desambiguación de entidades textuales a identificadores únicos en una base de conocimiento. **Similarity** se utiliza para comparar palabras, extensiones de texto y documentos y qué tan similares son entre sí. **Text Classification** para asignación de categorías o etiquetas a un documento completo o partes de un documento. **Rule-based Matching** lo utilizamos para encontrar secuencias de tokens en función de sus textos y anotaciones lingüísticas, similares a las expresiones regulares. **Training** permite actualización y mejora de las predicciones de un modelo estadístico. **Serialization** lo utilizamos para guardar objetos en archivos o cadenas de bytes.

Cuando llama nlp a un texto, spaCy primero tokeniza el texto para producir un objeto llamado Doc. Luego, se procesa en varios pasos diferentes; esto también se conoce como tubería de procesamiento. El procesamiento utilizado por estas tuberías suele incluir un tagger, un lematizer, un parser y un entity recog-

nizer. Cada componente de tuberías devuelve el Doc que se ha procesado y que luego se pasa al siguiente componente.

En la figura 3 se muestra un pequeño diagrama que detalla todo el proceso interno de spacy desde que recibe un texto hasta que el mismo es descompuesto y llegando a un doc resultante.

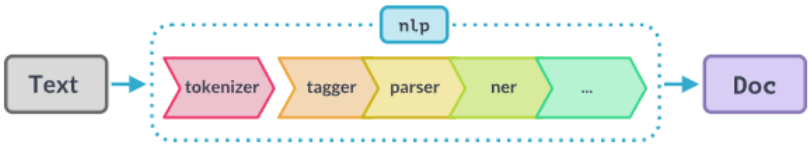

Figura 3: Procesamiento de spacy

En la tesina se utiliza esta herramienta con este modelo de procesamiento para validar y analizar aquellos requerimientos que son registrados por los usuarios y que requieren un análisis más profundo, para verificar si satisfacen las reglas del sistema [55].

En la figura 4 se muestra una configuración inicial de spacy en lenguaje python.

```python
import spacy

nlp = spacy.load("en_core_web_sm")
doc = nlp("Apple is looking at buying U.K. startup for $1 billion")
for token in doc:
    print(token.text, token.pos_, token.dep_)
```

Figura 4: Ejemplo de código de uso de spacy

Symfony es un completo framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web.

Symfony está desarrollado completamente con PHP y ha sido probado con éxito en sitios como Yahoo! Answers, delicious, DailyMotion y muchos otros sitios web de primer nivel. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en

plataformas Windows.

Características de symfony

Symfony se diseñó para que se ajustara a los siguientes requisitos: **fácil** de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y unix estándares). **Independiente** del sistema gestor de bases de datos. **Sencillo de usar** en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos. **Preparado** para aplicaciones empresariales y adaptable a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo. **Código** fácil de leer que incluye comentarios de phpDocumentor y que permite un mantenimiento muy sencillo. **Fácil** de extender, lo que permite su integración con librerías desarrolladas por terceros.

Automatización de características de proyectos web

Symfony automatiza la mayoría de elementos comunes de los proyectos web, como por ejemplo: **la capa** de internacionalización que incluye Symfony permite la traducción de los datos y de la interfaz, así como la adaptación local de los contenidos.**La capa** de presentación utiliza plantillas y layouts que pueden ser creados por diseñadores HTML sin ningún tipo de conocimiento del framework. Los helpers incluidos permiten minimizar el código utilizado en la presentación, ya que encapsulan grandes bloques de código en llamadas simples a funciones. **Los formularios** incluyen validación automatizada y relleno automático de datos, lo que asegura la obtención de datos correctos y mejora la experiencia de usuario. **Los datos** incluyen mecanismos de escape que permiten una mejor protección contra los ataques producidos por datos corruptos. **La gestión** de la caché reduce el ancho de banda utilizado y la carga del servidor. **La autenticación y la gestión**de credenciales simplifican la creación de secciones restringidas y la gestión de la seguridad de usuario.**El sistema** de enrutamiento y las URL limpias permiten considerar a las direcciones de las páginas como parte de la interfaz, además de estar optimizadas para los buscadores.

Las bases de datos siguen una estructura relacional, mientras que PHP y Symfony son orientados a objetos. Por este motivo, para acceder a la base de datos como si fuera orientada a objetos, es necesario una interfaz que traduzca la lógica de los objetos a la lógica relacional. Esta interfaz se denomina mapeo de objetos a bases de datos (ORM, de sus siglas en inglés object relational mapping).

Un ORM consiste en una serie de objetos que permiten acceder a los datos y que contienen en su interior cierta lógica de negocio.

Una de las ventajas de utilizar estas capas de abstracción de objetos/relacional es que evita utilizar una sintaxis específica de un sistema de bases de datos concreto. Esta capa transforma automáticamente las llamadas a los objetos en consultas SQL optimizadas para el sistema gestor de bases de datos que se

está utilizando en cada momento [48].

En la presente tesina utilizaremos symfony en su versión 5 para la construcción de la api de Ruems Requeriments que dispondrá de los servicios necesarios para responder a peticiones HTTP con los métodos clásicos; GET,POST,PUT, DELETE.

React es una librería Javascript para crear interfaces de usuario. La misma biblioteca tiene una serie de características que lo destacan.

Declarativo. React te ayuda a crear interfaces de usuario interactivas de forma sencilla. Diseña vistas simples para cada estado en tu aplicación, y React se encargará de actualizar y renderizar de manera eficiente los componentes correctos cuando los datos cambien. Las vistas declarativas hacen que tu código sea más predecible, por lo tanto, fácil de depurar.

Basado en componentes. Crea componentes encapsulados que manejen su propio estado, y conviértelos en interfaces de usuario complejas. Ya que la lógica de los componentes está escrita en JavaScript y no en plantillas. Los componentes de React implementan un método llamado render() que recibe datos de entrada y retorna qué mostrar.

Estados. Además de obtener datos de entrada (a los que accedes a través de this.props), un componente puede tener datos en su estado interno (a los que accedes a través de this.state). Cuando los datos del estado de un componente cambian, se vuelve a invocar render con los nuevos valores en this.states [51].

Aplicación en Ruems Requeriments

Se decidió utilizar react js en Ruems Requeriments, ya que permite mayor facilidad y comodidad para el diseño de las interfaces de la herramienta. Dado que react permite la creación de componentes nos facilita la tarea del diseño del menú de navegación, el pie de página y el cuerpo de la página en Ruems Requeriments tanto para las interfaces del analista como del usuario corriente.React js permite reutilizar los componentes creados para alguna otra interfaz que comparta las mismas características visuales. De este modo, se reduce el esfuerzo de desarrollo y se garantiza un rendimiento muy satisfactorio.

Material UI es una biblioteca de componentes React de código abierto que implementa Material Design de Google .Incluye una colección integral de componentes pre construidos que están listos para usar en producción desde el primer momento. Material UI tiene un diseño hermoso y presenta un conjunto de opciones de personalización que facilitan la implementación de su propio sistema de diseño personalizado sobre nuestros componentes.

Material ui es utilizada por medio de la componentizacion que nos ofrece react para estilizar toda la aplicación de Ruems Requeriments dando un estilo más elegante,llamativo e interactivo para los usuarios que utilicen la aplicación.

Algunos ejemplos en los cuales se utilizó la librería Material UI en Ruems Requeriments son los siguientes en la figura 5 representando un componente

botón de la librería material ui; y en la figura 6 un componente spinner utilizado en Ruems Requeriments al momento de esperar respuestas por parte del servidor. [46].

Figura 5: componente button de material ui utilizado en Ruems Requeriments

Figura 6: componente circleprogress de material ui utilizado en Ruems Requeriments

MySQL es el sistema de administración de bases de datos SQL de código abierto más popular, es desarrollado, distribuido y respaldado por Oracle Corporation.Una base de datos es una colección estructurada de datos. Puede ser cualquier cosa, desde una simple lista de compras hasta una galería de imágenes o la gran cantidad de información en una red corporativa. Para agregar, acceder y procesar datos almacenados en una base de datos de computadora, necesita un sistema de administración de bases de datos como MySQL. Una base de datos relacional almacena datos en tablas separadas en lugar de poner todos los datos en un gran almacén. Las estructuras de la base de datos están organizadas en archivos físicos optimizados para la velocidad. La parte SQL de MySQL significa Lenguaje de consulta estructurado. SQL es el lenguaje estandarizado más común utilizado para acceder a las bases de datos. Según su entorno de programación, puede ingresar SQL directamente (por ejemplo, para generar informes), incrustar declaraciones de SQL en código escrito en otro idioma o usar una API específica del idioma que oculta la sintaxis de SQL. Un posible ejemplo se visualiza en la figura 7.

Características más destacadas de mysql:

Internos y Portabilidad: Escrito en C y C++, probado con una amplia gama de diferentes compiladores, funciona en muchas plataformas diferentes, utiliza un diseño de servidor de varias capas con módulos independientes, proporciona motores de almacenamiento transaccional y no transaccional.

Tipos de datos: enteros con/sin signo de 1, 2, 3, 4 y 8 bytes de longitud, FLOAT, DOUBLE, CHAR, VARCHAR, BINARY, VARBINARY, TEXT, BLOB, DATE, TIME, DATETIME, TIMESTAMP, YEAR, SET, tipos de cadena de longitud fija y de longitud variable.

Seguridad: un sistema de privilegios y contraseñas que es muy flexible y seguro, y que permite la verificación basada en host. Seguridad de contraseñas mediante el cifrado de todo el tráfico de contraseñas cuando se conecta a un servidor [40].

MySQL en Ruems Requerimients

Elegimos MySQL como sistema de almacenamiento de base de datos Ruems Requeriments porque posee ciertas ventajas como su **rapidez**. La cualidad más destacada por quienes desarrollan MySQL es su velocidad y así cómo el software fue diseñado desde un principio, pensando principalmente en la rapidez. **MySQL es gratis** bajo la licencia GPL de código abierto, y el costo por licencia comercial es muy razonable. **Fácil de usar**. De construir e interactuar con una base de datos MySQL, siguiendo simples reglas en el lenguaje SQL. **Soporta** bases de datos de gran tamaño. MySQL maneja bases de datos de hasta 50 millones de filas o más. El límite de tamaño de archivo predeterminado para una tabla es de 4 GB, pero este se puede incrementar. En la figura 7 se visualiza una imagen de la interfaz que posee MySQL para interactuar con la base de datos.

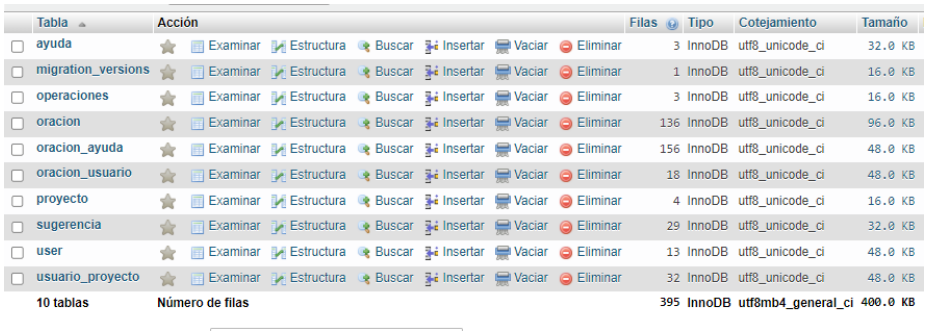

	Tabla	Acción	Filas	Tipo	Cotejamiento	Tamaño
☐	ayuda	Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	utf8_unicode_ci	32.0 KB
☐	migration_versions	Examinar Estructura Buscar Insertar Vaciar Eliminar	1	InnoDB	utf8_unicode_ci	16.0 KB
☐	operaciones	Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	utf8_unicode_ci	16.0 KB
☐	oracion	Examinar Estructura Buscar Insertar Vaciar Eliminar	136	InnoDB	utf8_unicode_ci	96.0 KB
☐	oracion_ayuda	Examinar Estructura Buscar Insertar Vaciar Eliminar	156	InnoDB	utf8_unicode_ci	48.0 KB
☐	oracion_usuario	Examinar Estructura Buscar Insertar Vaciar Eliminar	18	InnoDB	utf8_unicode_ci	48.0 KB
☐	proyecto	Examinar Estructura Buscar Insertar Vaciar Eliminar	4	InnoDB	utf8_unicode_ci	16.0 KB
☐	sugerencia	Examinar Estructura Buscar Insertar Vaciar Eliminar	29	InnoDB	utf8_unicode_ci	32.0 KB
☐	user	Examinar Estructura Buscar Insertar Vaciar Eliminar	13	InnoDB	utf8_unicode_ci	48.0 KB
☐	usuario_proyecto	Examinar Estructura Buscar Insertar Vaciar Eliminar	32	InnoDB	utf8_unicode_ci	48.0 KB
	10 tablas	Número de filas	395	InnoDB	utf8mb4_general_ci	400.0 KB

Figura 7: Interfaz gráfica de MySQL

Python es un lenguaje de programación potente y fácil de aprender. Tiene estructuras de datos de alto nivel eficientes y un simple pero efectivo sistema de programación orientado a objetos. La elegante sintaxis de Python y su tipado dinámico, junto a su naturaleza interpretada lo convierten en un lenguaje ideal para scripting y desarrollo rápido de aplicaciones en muchas áreas, para la mayoría de plataformas. El intérprete de Python es fácilmente extensible con funciones y tipos de datos implementados en C o C++ (u otros lenguajes que permitan ser llamados desde C). Python también es apropiado como un lenguaje para extender aplicaciones modificables.

Entre sus características más relevantes se debe destacar que **es un lenguaje interpretado**. Un lenguaje interpretado o de script es aquel que se ejecuta utilizando un programa intermedio llamado intérprete, en lugar de compilar el código a lenguaje máquina que pueda comprender y ejecutar directamente una computadora (lenguajes compilados). La ventaja de los lenguajes compilados es que su ejecución es más rápida. Sin embargo, los lenguajes interpretados son más flexibles y más portables. Python tiene, no obstante, muchas de las características de los lenguajes compilados, por lo que se podría decir que es semi interpretado. **Tipado Dinámico**. La característica de tipado dinámico se refiere a que no es necesario declarar el tipo de dato que va a contener una determinada variable, si no que su tipo se determinará en tiempo de ejecución según el tipo del valor al que se asigne, y el tipo de esta variable puede cambiar sí se le asigna un valor de otro tipo. **Fuertemente tipado**. No se permite tratar a una variable como si fuera de un tipo distinto al que tiene, es necesario convertir de forma explícita dicha variable al nuevo tipo previamente.**Orientado a objetos**, la orientación a objetos es un paradigma de programación en el que los conceptos del mundo real relevantes para nuestro problema se trasladan a clases y objetos en nuestro programa. La ejecución del programa consiste en una serie de interacciones entre los objetos [20].

El lenguaje de programación Python,es utilizado en Ruems Requeriments para el nexo entre la aplicación y el uso de la librería de NLP spacy, ya que la misma está escrita en este mismo. Por medio de otra librería de Python llamada hug que funciona como servidor local podemos generar un servicio el cual puede ser consumido por la interfaz gráfica y visualizar las respuestas de las kernel sentences ingresadas por los usuarios.

6 **Diseño de la herramienta propuesto en esta tesina**

Este capítulo describe el diseño detallado de la herramienta desarrollada, abordando los principales componentes y procesos que la conforman. A lo largo del capítulo, se presentan y explican los diferentes diagramas utilizados en la etapa de diseño, como el diagrama de clases, el diagrama de entidades y relaciones (DER), y el diagrama de procesos de negocio (BPM). Estos diagramas no solo ayudan a visualizar la estructura y funcionalidad del sistema, sino que también facilitan la comprensión de cómo interactúan entre sí los distintos elementos que lo componen.

6.1 **Diagrama entidad relación**

En el diagrama entidad relación de la herramienta Ruems Requerimients podemos ver las siguientes entidades:

Cada **sugerencia** que se almacene en la base de datos, contará con un número que la identificará entre todas las sugerencias existentes.

Al nombrar **ayuda** en el diagrama de entidad relación, se referencia a las reglas predefinidas en Ruems Requeriments. Cada Ayuda que se almacene en la base de datos, contará con un número que la identificará entre todas las ayudas existentes. Por otra parte, cada ayuda almacenada en la base de datos constará con un número de ayuda, que servirá de referencia al momento posterior del registro de una oración para que de esta manera, un servidor externo analicé la oración con cada ayuda en base a su número vinculado.

Al nombrar **oraciones** en el diagrama entidad relación se referencia a las Kernel sentences que son registradas por los usuarios. Cada Oración que se almacene en la base de datos, contará con un número que la identificará entre todas las oraciones existentes. Por otra parte, esta entidad cuenta con un atributo denominado tipo que en nuestra herramienta representa al estado de la oración (validado,aceptado,rechazado, analizado). Como último detalle a destacar, esta entidad contiene un atributo llamado análisis; que indica si el procesamiento de la oración fue satisfactorio o no durante todo el análisis de la oración con respecto a las reglas del sistema llevada a cabo por un servidor externo; esto significa cuantas más reglas cumpla la oración, más posibilidades de que el análisis sea satisfactorio.

Cada **proyecto** que se almacene en la base de datos contará con un número que la identificará entre todos los proyectos existentes y además de su nombre.

Cada **usuario** que se registra en la base de datos contará con un número que lo identificará entre todos los usuarios existentes. Cada Usuario posee un email y un nombre de usuario; estos atributos también son considerados únicos para cada usuario, lo cual significa que no pueden existir dos o más usuarios con igual nombre de usuario y/o email. Como último detalle, se destaca el atributo rol que indicará si el usuario en cuestión es un analista o un usuario corriente en Ruems Requeriments.

En el diagrama entidad relación de la herramienta Ruems Requerimients se describen las siguientes relaciones:

Tiene. Esta relación conecta la entidad Sugerencia con la entidad Oración. Esta es una relación de tipo (1,N), esto significa que según las cardinalidades establecidas en ambos lados de la relación, una sugerencia va a pertenecer a una oración mientras que una oración puede tener (o,N) sugerencias.

Op. Esta relación conecta la entidad Oración con la entidad Proyecto. Esta es una relación de tipo (1,N), esto significa que según las cardinalidades establecidas en ambos lados de la relación una Oración va a pertenecer a un Proyecto mientras que un proyecto puede tener (o,N) Oraciones.

Orac_ayud. Esta relación conecta la entidad Oración con la entidad Ayuda. Esta es una relación de tipo (N,N), esto significa que según las cardinalidades establecidas en ambos lados de la relación una Oración va a tener asociadas (O,N) ayudas mientras que una Ayuda puede estar vinculadas en (0,N) oraciones. Al tratarse de una relación (N,N), esta se transformará en una tabla de base de datos al igual que las entidades y por cada Oración vinculada a una Ayuda se dispone de un atributo llamado aprobación que indicará si el proceso de análisis de oraciones fue correcto o no para esa oración con su ayuda asociada. En esta nueva entidad que se crea en base a la relación, el identificador estará compuesto por el identificador de la oración más el identificador de la ayuda.

Registra. Esta relación conecta la entidad Oración con la entidad Usuario. Esta es una relación de tipo (1,N), esto significa que según las cardinalidades establecidas en ambos lados de la relación una Oración fue registrada por un Usuario mientras que un Usuario puede registrar (o,N) oraciones.

Pu. Esta relación conecta la entidad Usuario con la entidad Proyecto. Esta es una relación de tipo (N,N), esto significa que según las cardinalidades establecidas en ambos lados de la relación un Usuario podrá ser miembro de (0,N) proyectos mientras que un Proyecto puede tener (0,N) usuarios. Al tratarse de una relación (N,N), esta se transformará en una tabla de base de datos al igual que las entidades. En esta nueva entidad, que se crea en base a la relación el identificador estará compuesto por el identificador del proyecto más el identificador del usuario.

Orac_usr. Esta relación conecta la entidad Usuario con la entidad Oración. Esta es una relación de tipo (N,N), esto significa que según las cardinalidades establecidas en ambos lados de la relación un Usuario puede votar (0,N) oraciones mientras que una Oración puede ser votada por (0,N) usuarios. Al tratarse de una relación (N,N), esta se transformará en una tabla de base de datos al igual que las entidades. En esta nueva entidad que se crea en base a la relación, el identificador estará compuesto por el identificador del usuario más el identificador de la oración. Además, tenemos un tercer atributo llamado respuesta que indica por cada Oración que respuesta fue proporcionada por el Usuario a la cual se vincula.

En la figura 8 se muestra el diagrama entidad-relación resultante.

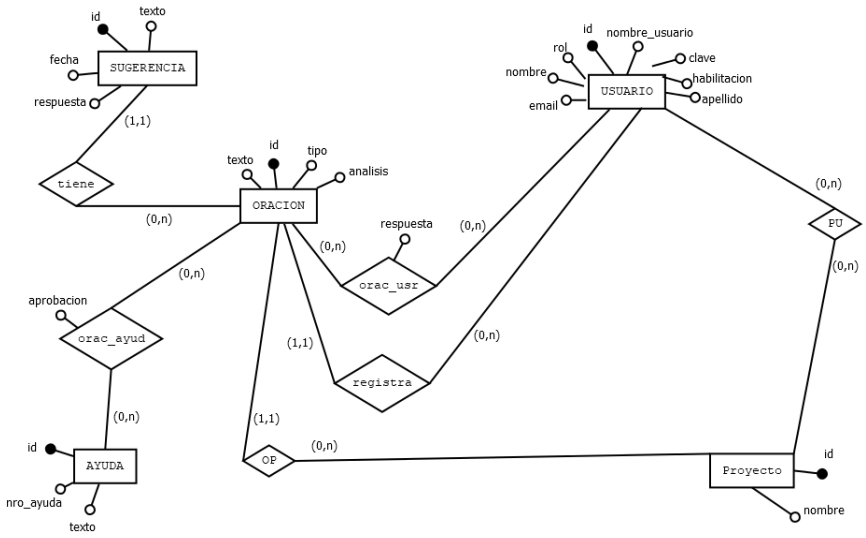

Figura 8: Diagrama entidad relación

Finalmente, luego de haber concluido con el análisis del diagrama entidad-relación de Ruems Requeriments, se escala un nivel más en este diagrama y detalla un modelo físico de base de datos con las tablas correspondientes, sus atributos y sus claves primarias. Dicho diagrama físico se verá de la siguiente manera en la figura 9:

SUGERENCIA = (id_sugerencia, texto, fecha, respuesta, oracion(FK))

AYUDA = (id_ayuda, nro_ayuda, texto)

ORACION = (id_oracion, texto, tipo, analisis, usuario(FK), proyecto(FK))

USUARIO = (id_usuario, nombre, rol, nombre_usuario, email, clave, habiltacion, apellido)

PROYECTO = (id_proyecto, nombre)

ORAC_AYUD = (id_oracion(FK), id_ayuda(FK), aprobacion)

ORAC_USR = (id_oracion(FK), id_usuario(FK), respuesta)

PU = (id_usuario(FK), id_proyecto(FK))

Figura 9: Diagrama fisico de base de datos

6.2 Diagrama business process model (BPM)

Esquema del proceso 1

El proceso representa a dos tipos de participantes llamados “Analista y Usuario Corriente”. A continuación, se describe brevemente la funcionalidad de cada tarea y la toma de decisiones en cada compuerta, siguiendo el flujo de ejecución del proceso: El proceso inicia cuando el usuario corriente presiona la opción registrar requerimientos.

Ingreso de texto. Esta tarea de usuario, visualiza un campo de texto para que el usuario corriente ingrese una kernel sentences.

Validación del texto. Esta tarea de servicio, procesa la kernel sentences recibida para determinar cuáles reglas (de las predefinidas en el sistema) son cumplidas e incumplidas. A partir de esta instancia la ejecución del proceso continúa en el usuario analista.

Revisión de requerimiento. Esta tarea de usuario, visualiza una tabla con las kernel sentences registradas en un proyecto junto con tres opciones posibles. (aceptar,rechazar,reajustar)

¿Confirma requerimiento?. En esta compuerta, el analista brinda una respuesta a la tarea anterior. Si la respuesta es si la ejecución continua a la tarea **Registrar requerimiento a validación**. Si la respuesta es no la kernel sentences se descarta del registro y finaliza todo el proceso. Si la respuesta es reajustar la ejecución continua a la tarea de **Reajuste de requerimiento**.

Registrar requerimiento a validación. Esta tarea de servicio registra la kernel sentences a producción para que la misma pueda recibir respuestas en el proyecto. Una vez finalizada esta tarea,el proceso se da por finalizado.

Reajuste de requerimiento. En esta tarea de usuario se visualiza la kernel sentences original y un campo de texto para modificarla.

Envío de sugerencia. Esta tarea de servicio, envía una notificación al usuario corriente que creó la kernel sentences con la modificación que proporcionó el analista. A partir de esta instancia, el proceso continúa su ejecución nuevamente al usuario corriente.

Revisión de sugerencia. En esta tarea de usuario se visualiza la sugerencia propuesta por el analista en la tarea anterior con dos opciones disponibles: aceptar y rechazar.

¿Acepta sugerencia?. En esta compuerta el usuario corriente brinda una respuesta a la tarea anterior. Si la respuesta es si la kernel sentences con la modificación del analista pasa a producción para recibir respuestas, luego de esto el proceso finaliza. Si la respuesta es no va a la tarea de envío de notificación.

Envío de notificación. Esta tarea de servicio, envía una notificación al analista que realizó la sugerencia, informando el rechazo del usuario corriente. Luego de esto, la ejecución retorna a la tarea de revisión de requerimiento. Repitiendo

el ciclo de ejecución del analista.

La figura 10 detalla el diagrama del proceso 1.

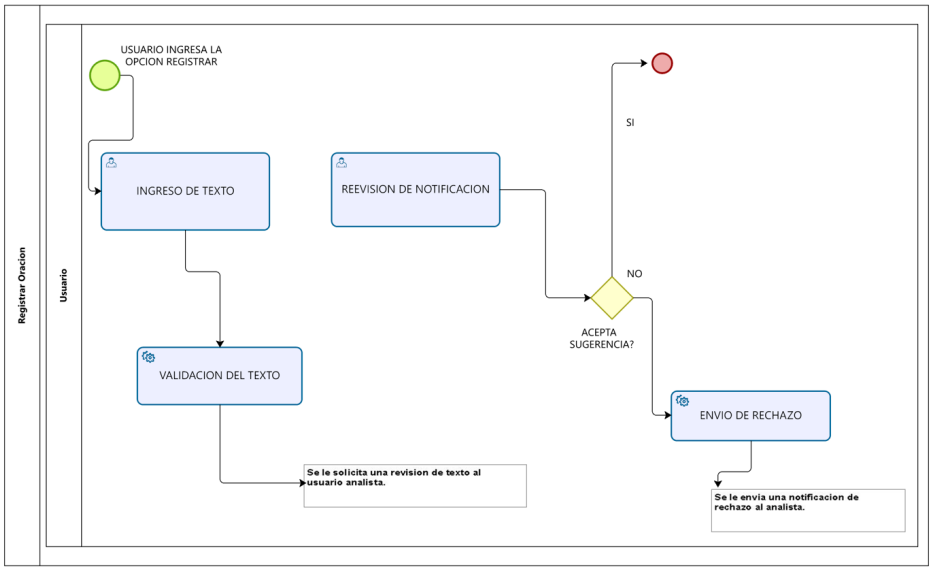

Figura 10: Diagrama BPM

Esquema del proceso Validar Requerimiento

El proceso representa a dos tipos de participantes llamados “Analista y Usuario Corriente”. A continuación, se describe brevemente la funcionalidad de cada tarea y la toma de decisiones en cada compuerta, siguiendo el flujo de ejecución del proceso:

El proceso inicia cuando el usuario corriente presiona la opción validar requerimientos.

Visualización de un conjunto de requerimientos aleatorios. Esta tarea de servicio junta un conjunto aleatorio de kernel sentences del usuario corriente en el proyecto que esté involucrado y las devuelve.

Selección de respuesta. Esta tarea de usuario visualiza el listado de kernel sentences recopiladas en la tarea anterior y permite al usuario la selección de una respuesta entre tres posibles para cada kernel sentences.

Confirmación de respuesta. Esta tarea de usuario visualiza un mensaje de alerta para que el usuario confirme las respuestas proporcionadas a las kernel sentences.

Control de respuestas totales. Esta tarea de servicio recibe las kernel sentences con las respuestas proporcionadas por el usuario corriente y recalcula su porcentaje de aprobación, juntando las respuestas recibidas. A partir de esta

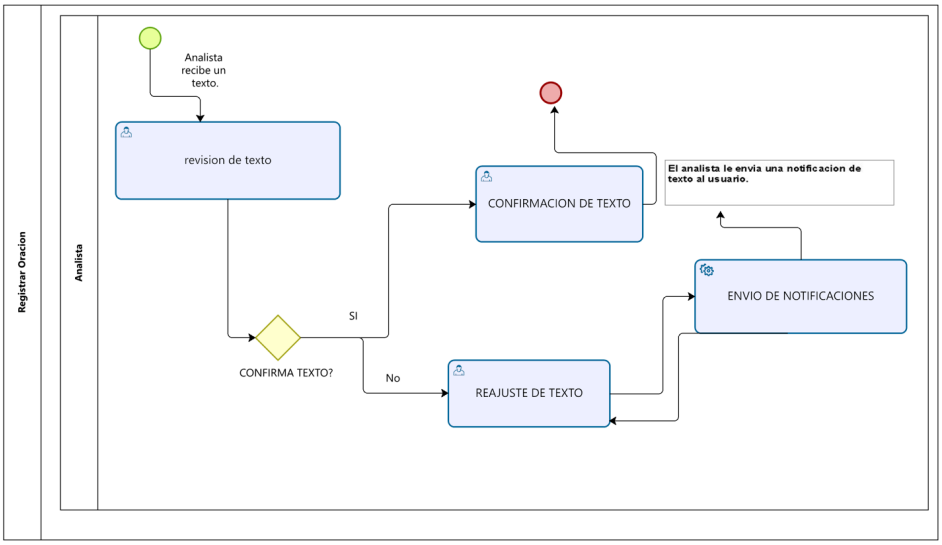

Figura 11: Diagrama BPM

instancia, el flujo de ejecución del proceso continúa a partir del analista.

¿Aprueba requerimiento?. En esta compuerta el analista, evalúa la respuesta de la tarea anterior. En caso de que retorne falso, la siguiente tarea a ejecutar es visualización de un conjunto de requerimientos aleatorios. En caso de que retorne verdadero, va la tarea de aprobación de requerimientos.

Aprobación de requerimiento. En esta tarea de usuario el analista aprueba/válida la kernel sentences para el proyecto involucrado.

Una vez finalizada la última tarea, el proceso se da por finalizado.

La figura 12 muestra el diagrama representativo del proceso de validar un requerimiento.

6.3 **Diagrama de clase**

De acuerdo al diagrama entidad relación que acabamos de analizar anteriormente, también podemos hablar de un diagrama UML que representa a los objetos que trabajarán en la aplicación Ruems Requeriments. Podemos decir que estos objetos son los siguientes:

Proyecto. Este objeto responderá a los clásicos mensajes de getters y setters a los atributos de este objeto como así también podrá responder a mensajes para agregar un usuario al proyecto,eliminarlos,agregar una oración al proyecto y eliminarla.

Usuario. Este objeto responderá a los clásicos mensajes de getters y setters a

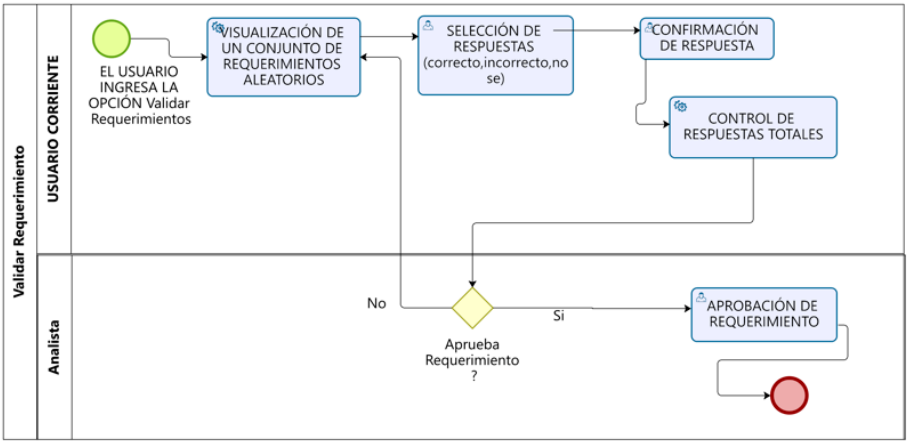

Figura 12: Diagrama BPM validar requerimiento

los atributos de este objeto, como así también podrá responder a mensajes para vincular un usuario con una oración y eliminar esa relación. Por otro lado responderá a mensajes para agregar y eliminar una relación entre oración/usuario al usuario en cuestión. Además, responderá a mensajes para agregar y eliminar una relación entre usuario/proyecto al usuario en cuestión.

Ayuda. Este objeto responderá a los clásicos mensajes de getters y setters a los atributos de este objeto, como así también podrá responder a mensajes para vincular una oración con una ayuda y eliminar esa relación.

Sugerencia. Este objeto responderá a los clásicos mensajes de getters y setters a los atributos de este objeto, como así también podrá responder a mensajes para agregar oraciones a la sugerencia, agregar una respuesta a esa sugerencia.

Oración. Este objeto responderá a los clásicos mensajes de getters y setters a los atributos de este objeto, como así también podrá responder a mensajes para vincular una oración con una sugerencia y eliminar esa relación. Por otro lado, responderá a mensajes para agregar y eliminar una relación entre oración/ayuda a la oración en cuestión. Además, responderá a mensajes para agregar y eliminar una relación entre oración/proyecto a la oración en cuestión.

La figura 13 muestra el diagrama de clases final.

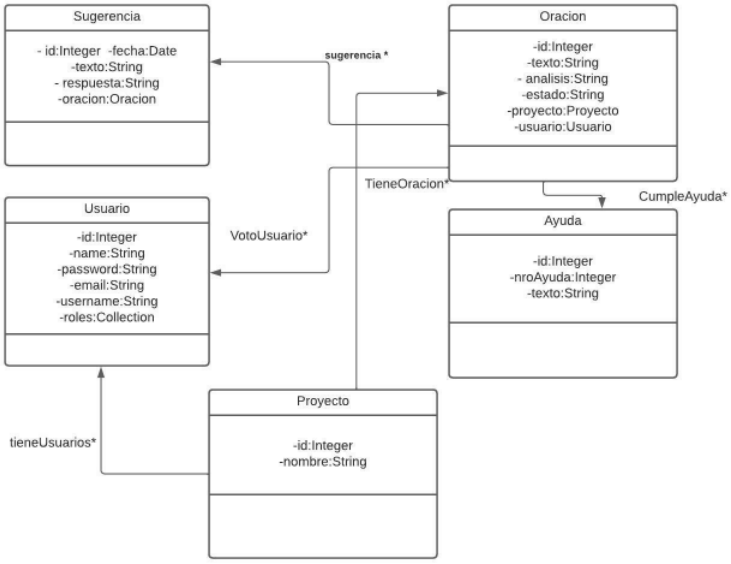

Figura 13: Diagrama de clases

La imagen anterior muestra la relación, entre los usuarios y las kernel sentences que se registran en los diferentes proyectos, como así también las sugerencias para cada una de ellas.

7 Guía de uso propuesta en esta tesina

Este capítulo proporciona una guía completa para el uso de la herramienta Ruems Requeriments, describiendo en detalle cada una de sus funciones y características. Aquí se explica cómo los usuarios pueden interactuar con las distintas funcionalidades, desde las operaciones más básicas hasta las más avanzadas. El objetivo de esta guía es asegurar que los usuarios puedan aprovechar al máximo todas las capacidades del sistema, facilitando su adopción y optimizando la experiencia de uso.

7.1 Autenticidad en la herramienta

Inicio De Sesión: El sistema de Ruems Requeriments consta con una sección que permite a cualquier usuario del sistema la opción de autenticarse por medio de un nombre de usuario y una clave. Se le solicita por medio de un formulario al usuario su nombre de usuario y contraseña, como así también un botón de confirmación de estos datos. La figura 14 muestra la sección de inicio de sesión que se encuentra en Ruems Requeriments.

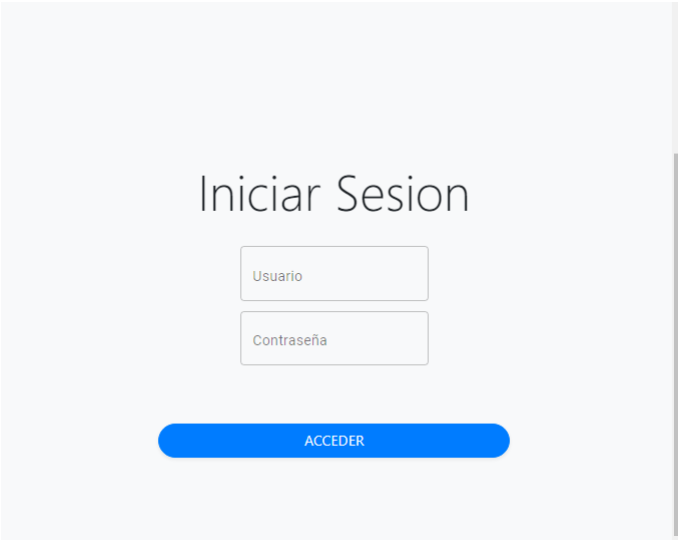

Figura 14: Formulario de inicio de sesión

El sistema envía una solicitud (request) al sistema y este último busca en su base de datos interna si existen un par de datos que coincidan con el nombre de usuario y contraseña ingresados, devuelve una respuesta al usuario (response). Las dos posibles respuestas que se pueden dar para esta solicitud son las siguientes:

Autenticación fallida. Esta respuesta puede darse ante los siguientes eventos:

I El usuario acepta los datos sin completar los dos campos solicitados. En este caso el sistema responde con un mensaje de error informando al usuario que debe completar los campos faltantes (tanto nombre de usuario como contraseña son requeridos); la figura 15 muestra un ejemplo de autenticación fallida cuando el usuario no ingresa ambos campos que son requeridos.

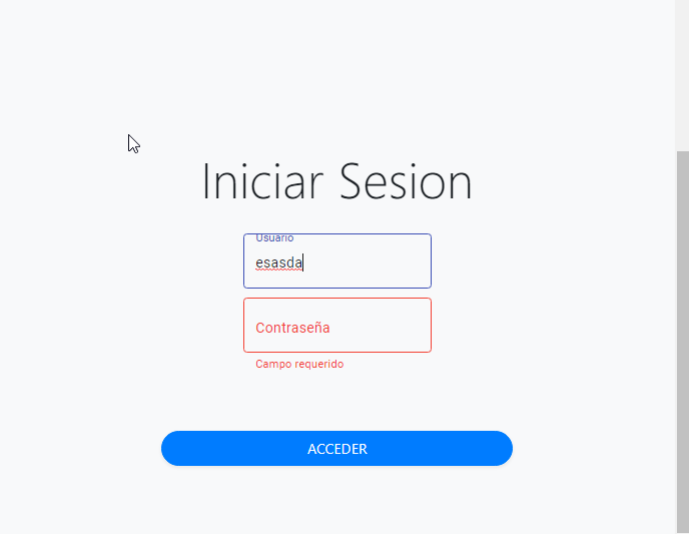

Figura 15: Ejemplo de error por campos requeridos

II El usuario ingresa los dos campos solicitados y acepta los datos, el sistema verifica los datos recibidos y no encuentra coincidencias en su registro de usuarios. En este caso el sistema responde con un mensaje de error informando al usuario que los datos ingresados son incorrectos (al menos uno de los dos datos requeridos son incorrectos). En la figura 16 podemos ver la respuesta de Ruems Requeriments al intentar autenticarse con un nombre de usuario o contraseña inexistentes.

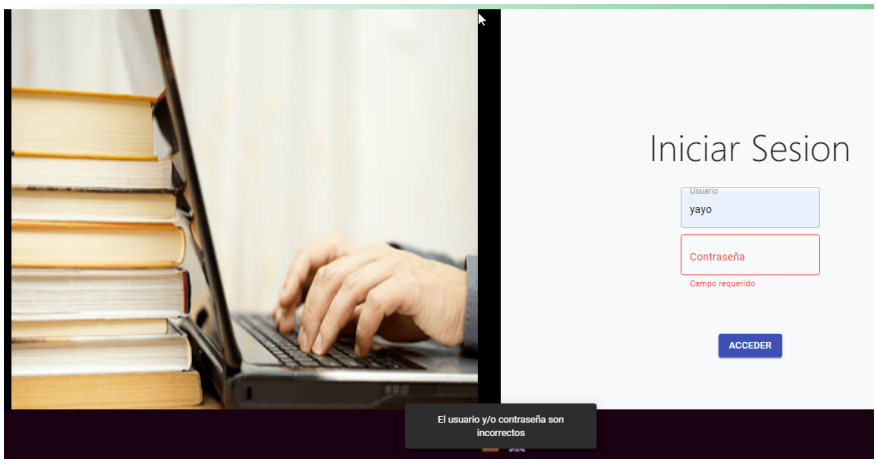

Figura 16: Ejemplo de usuario inexistente

Autenticación exitosa . El usuario ingresa los dos campos solicitados y acepta los datos, el sistema verifica los datos recibidos y encuentra un usuario en su base de datos interna cuyo nombre de usuario y contraseña coinciden con los datos proporcionados por el usuario. En este caso el sistema crea un token de sesión para este usuario y lo redirige a la pantalla de selección de proyectos en donde el usuario deberá escoger uno de los proyectos en donde él es partícipe, tal como se muestra en la figura 17.

Figura 17: Ejemplo de selección de proyecto luego de una autenticación exitosa

Cerrar Sesión. En cualquier momento el usuario tiene la posibilidad de cerrar su sesión, para esto tiene disponible la opción de cerrar sesión la cual la misma se encuentra en su barra de navegación superior, tal como se muestra en la figura 24. Al confirmar esta operación se envía una solicitud (request) al sistema. El sistema destruye el token de sesión generado al momento de la autenticación y elimina toda la información respectiva del usuario almacenada en su sistema de almacenamiento local. El sistema retorna una respuesta (response) indicando que la operación se realizó exitosamente y devuelve al usuario a la pantalla principal de autenticación.

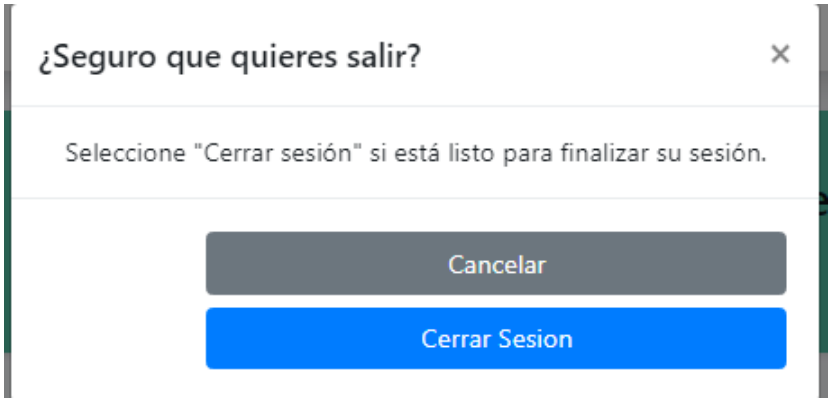

Figura 18: Alerta de confirmación de cierre de sesión

7.2 **Usuarios y roles**

El sistema Ruems Requeriments contiene dos tipos de roles para los usuarios. Estos roles son los siguientes:

Usuario corriente. Es aquel que tendrá acceso únicamente a su actividad y a la interacción con sus proyectos personales. Entre sus funciones disponibles se encuentran:

Registrar requerimientos. Cargar a un proyecto específico nuevos requisitos a validar por otros usuarios que sean miembros de dicho proyecto.

Validar requerimientos. Votar con tres tipos de respuesta (correcto, incorrecto,no sé) los requerimientos de otros usuarios asociados a sus proyectos personales.

Visitar las reglas. En todo momento el usuario puede hacer seguimiento de las reglas definidas para crear nuevos requerimientos.

Controlar las notificaciones de sus proyectos. El usuario puede confirmar aquellas notificaciones recibidas en sus proyectos provenientes de los analistas.

A continuación en la figura 40 se muestra el panel de navegación para un usuario corriente autenticado.

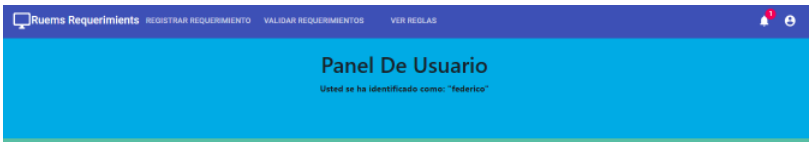

Figura 19: Panel del usuario corriente

Analista. Es aquel usuario que podrá tener acceso al panel de control, por lo tanto, podrá tener una visión global de todo el equipo de colaboradores. Entre sus funciones disponibles se encuentran:

Administrar requerimientos. El analista puede visualizar y administrar los requerimientos de los proyectos de los cuales participa sea cual sea su estado (analizado,validado,rechazado,aceptado).

Administrar usuarios. El analista puede visualizar y administrar todos los usuarios de los proyectos en los cuales participa. (registrar nuevos usuarios y ver su historial).

Reglas. Se detalla una sección de reglas, donde se contemplan todas las reglas que el usuario debe respetar a la hora de registrar una kernel sentences.

Estadísticas. Se detalla una sección de control para que el analista pueda en cualquier momento realizar un seguimiento general sobre el proyecto en cuestión.

Las figuras 20 y 21 muestran el panel de navegación del usuario analista con las operaciones detalladas anteriormente.

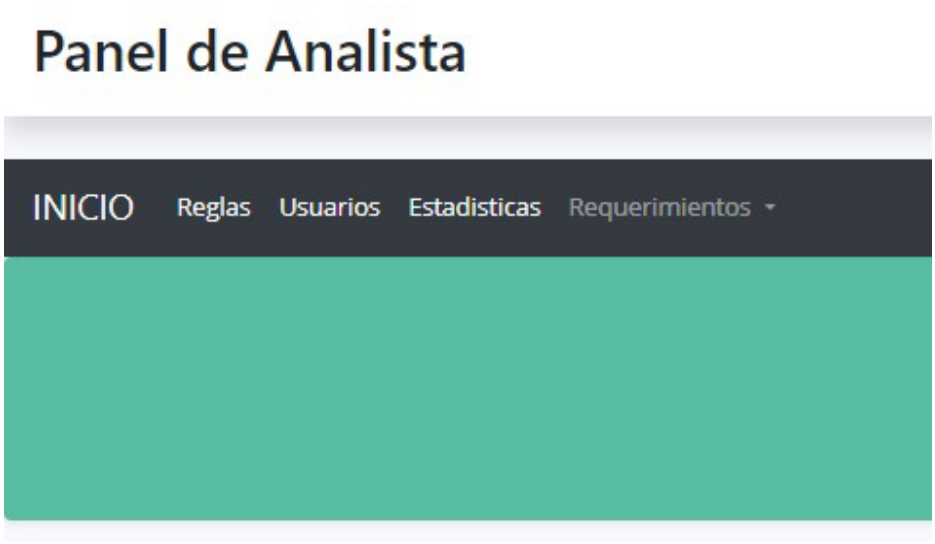

Figura 20: Panel del analista

7.3 **Características de la herramienta**

A continuación se detallan las funcionalidades que se proveen en el sistema Ruems Requeriments para ambos tipos de usuario. Estas funcionalidades están asociadas a la colaboración entre usuarios,del mismo proyecto el seguimiento por parte de los analistas y la interacción con los usuarios y la adminis-

Figura 21: Panel del analista

tración general del sistema.

7.4 **Registro de requerimiento en la herramienta**

El usuario corriente selecciona la opción registrar requerimiento. La aplicación visualiza un campo para ingresar un nuevo requerimiento y un botón enviar. El usuario ingresa un requerimiento y presiona el botón enviar tal como se muestra en la figura 22.

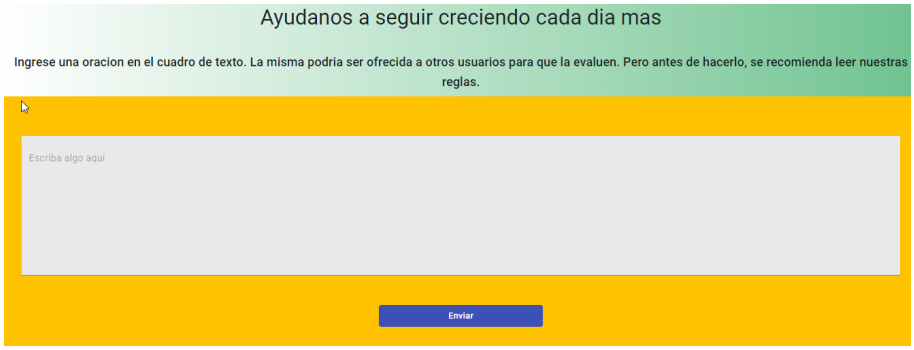

Figura 22: Pantalla principal de registrar requerimiento

El sistema recibe el requerimiento y se comunica con un servidor externo de procesamiento de lenguaje natural. Este servidor valida el requerimiento por cada regla predefinida en la aplicación, retornando el resultado de la validación de cada regla. El sistema recibe esta respuesta del servidor externo y la devuelve a la aplicación Ruems Requeriments. La aplicación recibe la respuesta y abre una ventana que visualiza el resultado obtenido al usuario tal como se puede obsvervar en las figuras 23 y 24.

Figura 23: Ventana de resultado de análisis final

Esta ventana muestra cada regla y el resultado procesado según el requerimiento; además muestra el resultado final sobre el análisis general del requerimiento, además un botón registrar requerimiento. Luego se visualiza la opción de registrar el requerimiento. El usuario selecciona dicha opción y se visualiza una alerta con un mensaje de éxito de requerimiento registrado exitosamente, tal como se puede observar en la figura 25.

7.5 **Registro de usuario en la herramienta**

El usuario analista dispone de una opción para poder registrar nuevos usuarios a Ruems Requerimients. El analista ingresa a la sección de administración de usuarios y presiona la opción “agregar nuevo usuario”. El sistema visualiza un formulario con los siguientes campos: nombre de usuario,email,contraseña y rol. En el caso del campo rol el analista dispone de las opciones usuario y analista para dar de alta. El analista presiona la opción registrar usuario y el sistema válida los datos ingresados. En este punto pueden darse las siguientes situaciones:

- Si el nombre de usuario y/o email ingresados ya se corresponden con los de otros usuarios ya existentes en Ruems Requeriments el sistema informa esta excepción con un mensaje de error de datos ya existentes. Tal como se puede ver en la figura 26 el usuario ingresar por error un usuario ya existente en el sistema de Ruems y se visualiza un cartel con el error correspondiente.

Figura 24: Otro ejemplo de ventana de resultado de análisis final

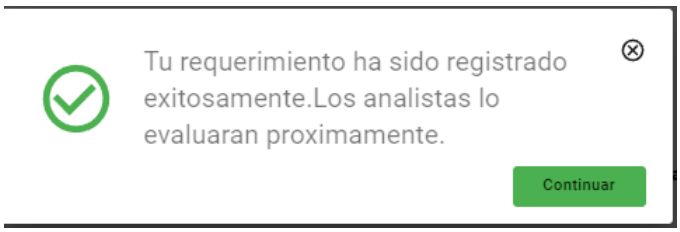

Figura 25: Ventana de registro exitoso

II En el caso de que el dato ingresado en el campo email no cumpla con el formato solicitado de un email real ruems requeriments visualiza un error de email incorrecto.

III Si el analista dejo campos sin completar Ruems Requeriments informa que dichos campos no fueron completados. En la figura 27 se visualiza un ejemplo en el cual el usuario no ingresa el email ni el rol, por lo tanto, se visualiza un mensaje de error.

IV La contraseña ingresada por el analista y su respectiva confirmación no coinciden una con otra, por lo tanto, Ruems Requeriments visualiza un mensaje de error.

V En caso de que las validaciones anteriores sean exitosas o no se produzcan errores,Ruems Requeriments realiza una petición HTTP a la base de

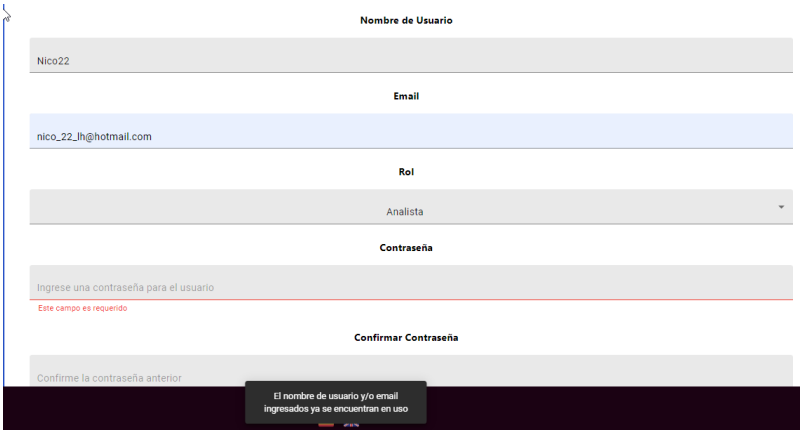

Figura 26: Autenticación fallida por registro de usuario existente

Figura 27: Ejemplo de autenticación fallida por campos no completados

datos enviando los datos ingresados por el analista en el cuerpo de la solicitud. El sistema almacena los datos y responde con un mensaje de éxito al analista el cual dice el usuario fue dado de alta correctamente. En la figura 28 se visualiza un mensaje resultante de alerta exitoso, para el caso de que el usuario ingrese correctamente los datos y se validen de forma exitosa.

7.6 **Voto de requerimientos**

El usuario corriente selecciona la opción validar requerimiento. La aplicación le envía al sistema el identificador del proyecto que actualmente se encuentra el usuario. El sistema recibe dicho identificador y realiza dos filtros al sistema.

Primero, se obtiene de todos los requerimientos cargados en el proyecto aque-

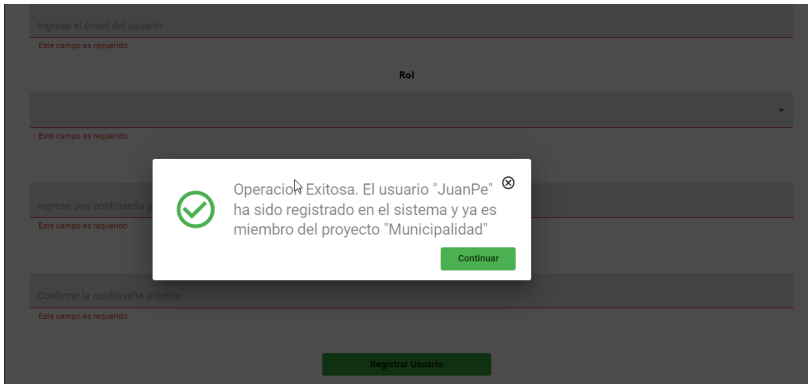

Figura 28: Registro del usuario JuanPe exitoso

llos que no fueron creados por el usuario que actualmente va a votar. Una vez obtenido este subconjunto de requerimientos, el sistema obtiene solamente aquellos que fueron aceptados por un analista (su estado debe ser igual a aceptado).

Luego, el sistema realiza otra búsqueda la cual consiste en obtener todos los requerimientos que el usuario corriente ya ha votado en dicho proyecto.

Una vez establecidas las consultas anteriores el sistema realiza una diferencia entre las dos consultas realizadas. Por ejemplo: el primer filtro se llama A y el segundo se llama B. El sistema realiza A - B; esto quiere decir que el resultado final se conforma por todos los requerimientos obtenidos en A que no estén en el filtro B. Es decir todos los requerimientos del proyecto actual en estado aceptado que no fueron creados por el usuario y que además no hayan sido votados por el mismo.

Finalmente, si el resultado de A - B supera ampliamente los 5 requerimientos disponibles el sistema solo devuelve los 5 primeros resultados del total. Esto quiere decir que el usuario solo podrá votar de a 5 requerimientos por votación; y en caso de que el resultado de A - B sea de menos de 5 requerimientos se devuelve todos los que haya encontrado. Seguido de esto la aplicación muestra los 5 (o menos) requerimientos encontrados en los pasos anteriores. Por cada requerimiento del listado el usuario corriente dispondrá de 3 posibles opciones de respuesta las cuales son correcto, incorrecto, no se. Además, se visualizan dos botones enviar y cancelar. En la figura 29 se detalla un posible caso en el cual el usuario puede seleccionar que respuesta darle a cada una de las kernel sentences y enviar dicha respuesta en el sistema de Ruems Requeriments.

Si el usuario corriente presiona la opción cancelar el sistema le visualiza una alerta de confirmación tal como se ve en la figura 30. El usuario confirma la alerta y la aplicación redirecciona a la pantalla principal del panel del usuario corriente.

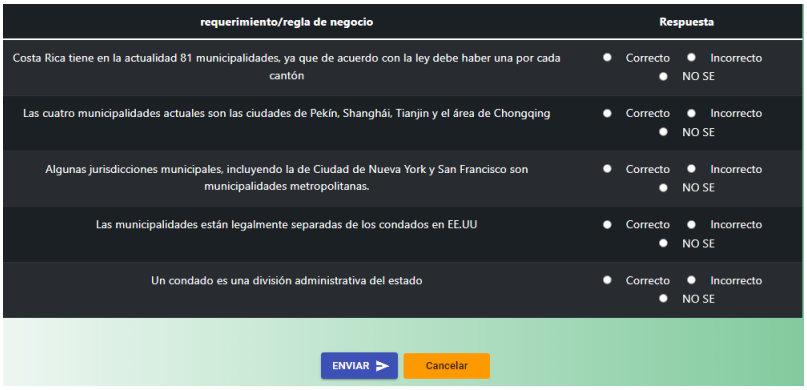

Figura 29: Ejemplo de votación de requerimientos

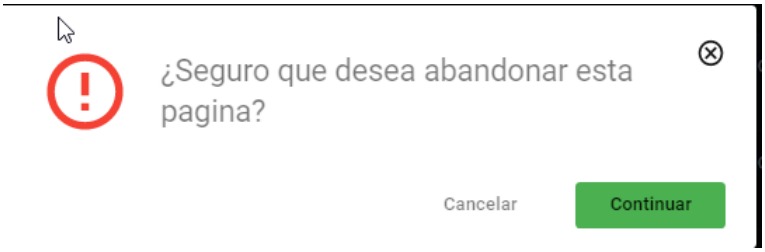

Figura 30: Alerta de cancelación de votación de requerimiento

Si el usuario corriente presiona la opción enviar se pueden generar dos posibles caminos:

I) Existen requerimientos sin responder. En este caso la aplicación le informa al usuario que debe responder la totalidad de los requerimientos como se visualiza en la figura 31.

II) El usuario respondió todos los requerimientos. En este caso la aplicación envía al sistema los requerimientos con las respuestas seleccionadas por el usuario.

III) El sistema recibe el listado de los requerimientos con las respuestas y almacena cada una con su respectiva respuesta en el historial del usuario corriente.

IV) Luego de haber realizado la votación la aplicación muestra un mensaje de éxito como se visualiza en la figura 32 en la validación de requerimientos y redirecciona al usuario a la pantalla principal del panel.

Cabe destacar que solo se le permite al usuario corriente realizar una votación por día en un proyecto determinado. Lo cual quiere decir que luego de haber

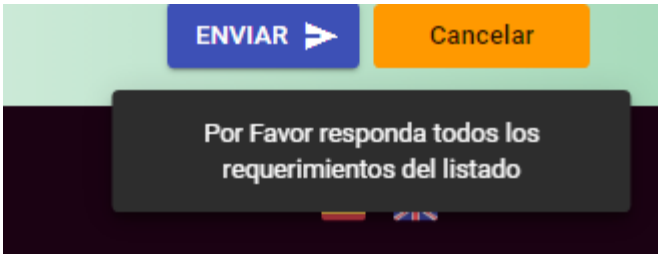

Figura 31: Mensaje de respuestas sin responder

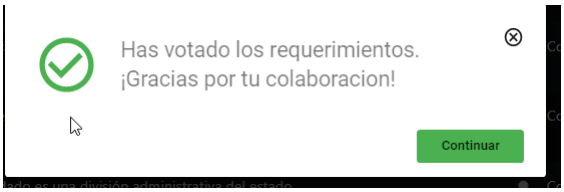

Figura 32: Mensaje de votación de requerimientos exitosa

realizado una votación deberá esperar 23 horas para poder a volver a votar requerimientos en el mismo proyecto. En la figura 33 en el sistema de Ruems Requeriments se visualiza el mensaje de que ya se realizó la votación y el tiempo transcurrido para poder volver a votar nuevamente.

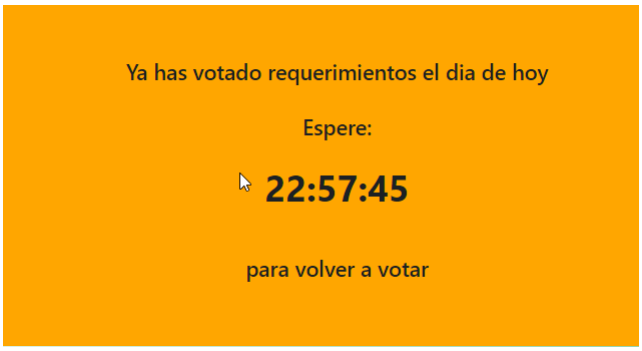

Figura 33: Pantalla de espera para volver a votar requerimientos

7.7 **Gestión de notificaciones de analistas/usuarios**

En Ruems Requeriments los usuarios(analista y usuarios corrientes)deben estar en constante comunicación para que de esa forma se logre una mayor fluidez en el trabajo con los requerimientos. Es por este motivo que ambos usuarios disponen de una sección de notificaciones en sus respectivos paneles. En

esta sección pueden mantenerse informados mutuamente acerca de los procesos de alta de requerimientos en un proyecto en específico.

Notificaciones en panel de usuario

En la barra de navegación del panel del usuario corriente, se dispone de la opción ver notificaciones como se ilustra en la figura 34 se simboliza con un icono de campana junto a un número que indica la cantidad de notificaciones recibidas.

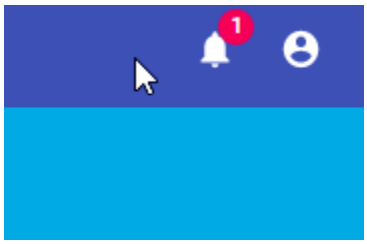

Figura 34: Símbolo que indica las notificaciones de un usuario corriente

Al ingresar a esa sección Ruems Requeriments visualiza un listado con las notificaciones del usuario en el proyecto que se encuentra actualmente y pendientes de confirmación. Esto quiere decir que se muestran aquellos requerimientos que fueron registrados por el usuario y que obtuvieron alguna solicitud de cambio por parte de alguno de los analistas.

Por cada notificación visualizada en el listado se muestran:

I) Requerimiento original: Se visualiza el requerimiento en su estado original; esto quiere decir se muestra el requerimiento en la forma en que el usuario lo registró en un principio.

II) Sugerencia analista: Se visualiza el cambio propuesto por el analista sobre este requerimiento.

III) Operaciones: Se visualizan dos opciones sobre esta notificación. Por un lado una opción de aceptar;en caso de elegir esta opción se acepta la propuesta del analista y el requerimiento pasa a estado activo (ya puede obtener respuestas por parte de otros usuarios). Por otro lado, una opción de rechazar; en caso de elegir esta opción se descarta la propuesta del analista. En ambos casos se le notifica al analista la operación elegida por el usuario.

   En la figura 35 podemos ver un ejemplo de lo mencionado anteriormente en donde en el primer formulario de requerimiento original se visualiza el requerimiento tal cual fue ingresado por el usuario, y en el segundo formulario de sugerencia por parte del analista se visualiza la propuesta por parte del analista. Por debajo de eso tenemos las opciones de aceptar y cancelar dicha sugerencia.

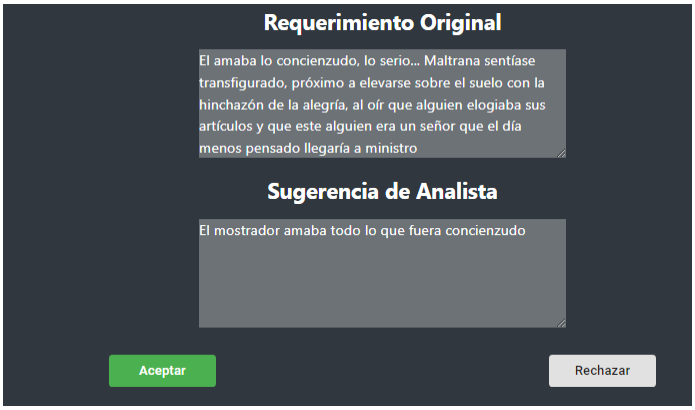

Figura 35: Visualización de una notificación de usuario corriente

Notificaciones en panel de analista

En la barra de navegación del panel del analista, como se visualiza en la figura 36 se dispone de la opción ver notificaciones simbolizada por un icono de campana junto a un número que indica la cantidad de notificaciones recibidas.

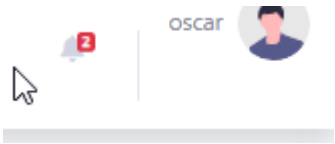

Figura 36: Símbolo que indica las notificaciones de un usuario analista

Al presionar esta opción, se despliega una lista con las notificaciones recibidas en el proyecto actual. Por cada notificación recibida se muestra la fecha en la que fue confirmada junto a la respuesta proporcionada por el usuario a el cual se le envió; en caso de haber sido una notificación de un rechazo de requerimientos la misma es simbolizada por un icono de alerta/exclamación, en cambio, si se trata de una notificación basada en una aprobación de una propuesta la misma se simboliza con un símbolo azul. Esto se puede visualizar en las figuras 37 y 38 donde se detalla el nombre del usuario junto con el número de sugerencia.

El analista puede acceder a una determinada notificación del listado desplegable, en caso de hacerlo la aplicación redirige a una pantalla donde se visualiza los datos de la notificación:

I) Si se trata de una propuesta de cambio de requerimiento que fue aprobada por su usuario autor se visualiza un campo que muestra el requerimiento que se había registrado en un principio y otro campo con el reajuste propuesto por el analista y aprobado por el usuario.

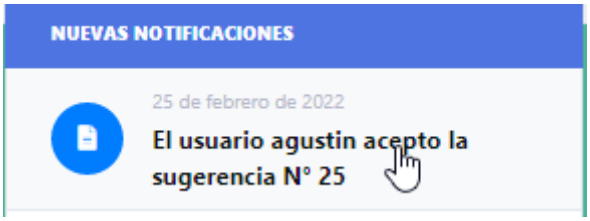

Figura 37: Notificación de analista que indica aceptación

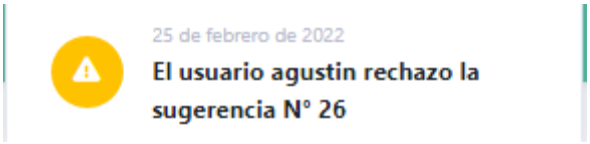

Figura 38: Notificación de analista que indica rechazo

La figura 39 muestra el resultado correcto del caso de que el usuario decida aceptar el reajuste por parte del analista.

Figura 39: Vista de una notificación que fue aprobada por un usuario corriente

II) Si se trata de una propuesta de cambio de requerimiento que fue rechazada por su usuario autor,además de mostrar los campos anteriormente mencionados, se le visualiza una opción al analista para volver a realizar un nuevo reajuste a este requerimiento.

La figura 40 muestra el resultado del caso en que el usuario decida rechazar el reajuste por parte del analista y se muestra un botón para volver a reajustar el requerimiento según su criterio.

Cabe destacar que solamente se van a mostrar las notificaciones que tengan como máximo tres días de antigüedad; esto quiere decir que una notificación

Figura 40: Vista de una notificación que fue rechazada por un usuario corriente.

dejará de verse en el listado desplegable luego de tres días de haber sido recibida.

7.8 **Análisis de estadísticas**

El usuario analista puede en todo momento visitar las estadísticas de un determinado proyecto. Para esto ingresa en la sección (estadísticas). La aplicación envía al sistema el identificador del proyecto actual.

El sistema recibe el identificador del proyecto y realiza el siguiente filtro; busca todos los requerimientos del proyecto recibido que se encuentren en estado aceptado y los devuelve en una lista, la aplicación recibe dicha lista y visualiza cada requerimiento en una tabla.

Por cada requerimiento de la tabla el analista podrá llevar un control de tres posibles estadísticas que se detallan en el siguiente cuadro 7.

Además de proveer estos tres tipos de estadísticas el analista dispondrá de una operación adicional: “requerimiento validado”; esta opción se utiliza para que el analista de por finalizado el periodo de validación por parte de los usuarios a este requerimiento en el proyecto actual si el mismo ha alcanzado un porcentaje de estadísticas favorable para el analista. Esto está ilustrado en la figura 41 donde se muestra dicha operación junto con los requerimientos y sus porcentajes de correctitud,incorrectitud y porcentaje sin respuestas.

Cuadro 7: Estadística de evaluación de requerimiento

Estadística	Descripción
Porcentaje de correctitud	Porcentaje del total de respuestas dadas como correctas para dicho requerimiento sobre el conjunto de respuestas totales.
Porcentaje de incorrectitud	Porcentaje del total de respuestas dadas como incorrectas para dicho requerimiento sobre el conjunto de respuestas totales.
Porcentaje sin respuesta	Porcentaje del total de respuestas dadas como inconclusas para dicho requerimiento sobre el conjunto de respuestas totales.

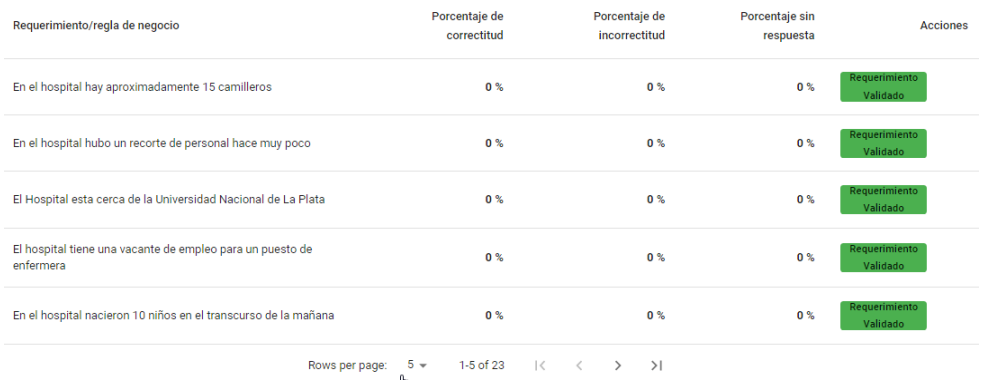

Figura 41: Ejemplo de tabla de estadísticas

7.9 **Análisis de historial de usuarios**

El analista puede ingresar a la sección de administrar usuarios de un proyecto en específico y visualizar a todos los miembros de dicho proyecto (usuarios corrientes), en el listado de los miembros del proyecto, el analista dispondrá de la opción de ver historial. Al presionar esta opción Ruems Requeriments visualiza un resumen general de la participación de dicho miembro en el proyecto actual. El historial cuenta con las siguientes secciones:

Gráfico de torta: se visualizan tres gráficos:

I) Porcentaje de correctitud. Se visualiza un control total de las respuestas dadas como correctas por parte de dicho miembro en el proyecto actual con respecto a las respuestas totales.

II) Porcentaje de incorrectitud. Se visualiza un control total de las respuestas

dadas como incorrectas por parte de dicho miembro en el proyecto actual con respecto a las respuestas totales.

III) Porcentaje sin respuesta. Se visualiza un control total de las respuestas dadas como inconclusas por parte de dicho miembro en dicho proyecto con respecto a las respuestas totales.

La figura 42 muestra un ejemplo de las estadísticas anteriores mencionadas para que un analista vea la participación por parte de los usuarios corrientes.

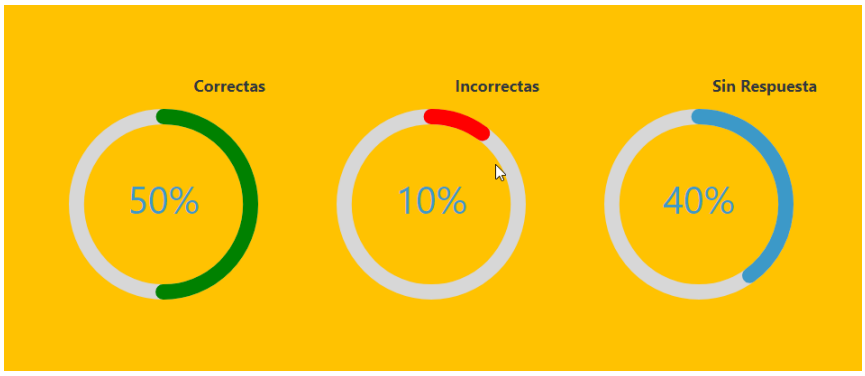

Figura 42: Visualización de gráficos de torta del usuario

Requerimientos resueltos: La figura 43 visualiza además una tabla con todos los requerimientos que dichos miembros respondieron en el proyecto actual. Por cada requerimiento se muestra la respuesta brindada por el usuario y la fecha y hora de la votación. Esta tabla se ordena por los requerimientos votados más recientemente a los más antiguos.

requerimiento/regla de negocio	Respuesta	Fecha de votacion
Un condado es una división administrativa del estado	nose	2022-02-22, 01:30:52
Las municipalidades están legalmente separadas de los condados en EE.UU	incorrecto	2022-02-22, 01:30:52
Algunas jurisdicciones municipales, incluyendo la de Ciudad de Nueva York y San Francisco son municipalidades metropolitanas.	correcto	2022-02-22, 01:30:52
Las cuatro municipalidades actuales son las ciudades de Pekín, Shanghái, Tianjin y el área de Chongqing	nose	2022-02-22, 01:30:52
Costa Rica tiene en la actualidad 81 municipalidades, ya que de acuerdo con la ley debe haber una por cada cantón	correcto	2022-02-22, 01:30:52

Figura 43: Tabla de historial del usuario

7.10 **Reglas del sistema**

Cualquier usuario de Ruems Requeriments puede en todo momento visitar las reglas del sistema. Las cuales son:

I) Registre requerimientos que cumplan con el formato: Sujeto + Verbo + Objeto Directo. Adicionalmente, puede agregar un Objeto Indirecto. Por ejemplo: una oración con las siguientes características, “los kayakistas reservan travesías en kayak en la empresa.”

II) No deje sujetos tácitos en el requerimiento. Especifique sobre qué o quién se realiza la acción (verbo). Para darle una mejor simplicidad a los requerimientos se recomienda a los usuarios no registrar requerimientos tales como “queríamos llegar temprano, pero nos retrasamos”. En este caso el sujeto tácito sería “nosotros” el cual está implícito en la oración.

III) No registre requerimientos muy extensos que vayan más allá de su idea fundamental. Esta regla se utiliza para informarle al usuario que al momento de registrar un requerimiento no se desvíe del foco de atención del requerimiento. Por ejemplo: “los kayakistas reservan travesías de kayak a la empresa, pero el jefe es un administrador”. En la idea anterior el foco de atención es la reserva de travesías, pero la misma se desvía al jefe de la empresa.

La figura 44 muestra como se ven las reglas mencionadas anteriormente en Ruems Requeriments.

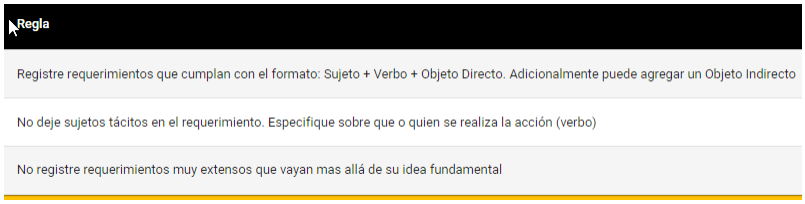

Figura 44: Tabla de reglas del sistema

7.11 **Confirmación de requerimientos**

Los requerimientos son el foco fundamental en el desarrollo de un proyecto. Por este motivo Ruems Requeriments proporciona cuatro tipos de requerimientos que los analistas deben tener en cuenta a la hora de administrar los proyectos. El primer tipo de requerimiento que se encuentra en la aplicación son los requerimientos “analizados”.

Estos requerimientos son aquellos que fueron inmediatamente registrados por los usuarios corrientes; esto quiere decir que una vez que un usuario corriente lo da de alta este requerimiento pasa por un proceso interno de análisis para determinar qué reglas (de las predefinidas en el sistema) satisfacen y luego queda a la espera de una revisión posterior por parte de los analistas.

La figura 45 muestra una tabla con los requerimientos y que reglas cumplieron de las mencionadas anteriormente y cuáles no con las opciones de aceptar,reajustar o rechazar.

Como segundo tipo de requerimiento que se encuentra en la aplicación están

Figura 45: Tabla de requerimientos analizados

los requerimientos aceptados.

Este tipo de requerimientos son aquellos los cuales fueron aceptados por los analistas de forma posterior al registro de los usuarios corrientes. Estos requerimientos también se conocen como requerimientos activos, ya que son los que están disponibles para recibir respuestas por parte de los usuarios corrientes.

La figura 46 muestra una tabla con las estadísticas realizadas por parte de los requerimientos aprobados.

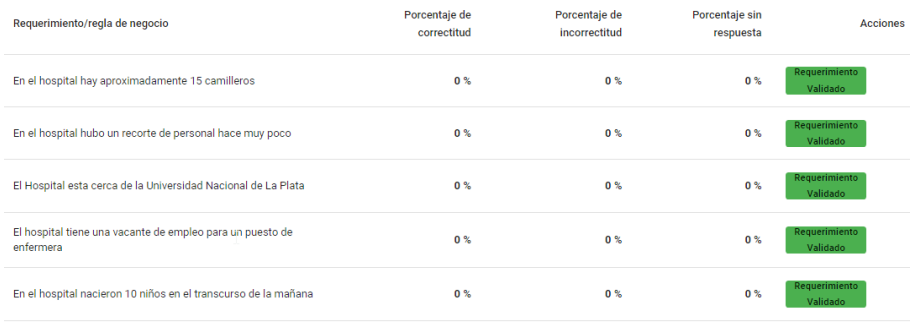

Figura 46: Tabla de estadísticas que indica los requerimientos aceptados de un proyecto

Como tercer tipo de requerimiento que se encuentran en la aplicación están los requerimientos rechazados.

Este tipo de requerimientos son aquellos que luego de procesamiento interno del sistema (tras haberse registrado por parte de los usuarios corrientes), fueron rechazados tras la revisión de los analistas. Esto quiere decir que los requerimientos rechazados quedan totalmente desvinculados de los proyectos a los cuales son registrados por lo que no podrán recibir respuestas por parte de

otros usuarios.

En La figura 47 se visualiza un listado con los requerimientos rechazados en un proyecto, en donde también se muestran al analista y el usuario correspondiente en dicho requerimiento.

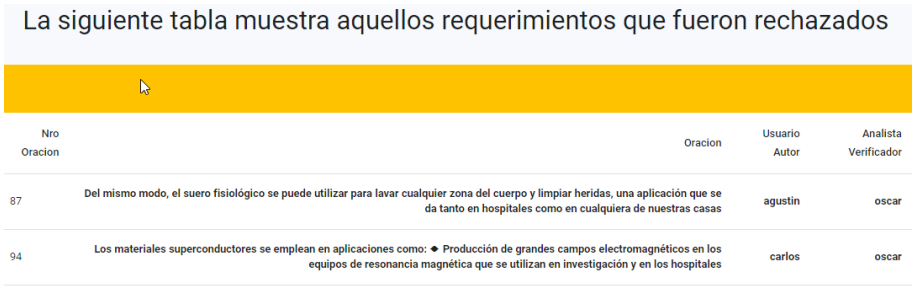

Figura 47: Tabla de estadísticas que indica los requerimientos rechazados de un proyecto.

Como cuarto y último tipo de requerimiento presente en Ruems Requeriments se encuentran los requerimientos validados.

Este tipo de requerimiento son aquellos que tras recibir una cantidad de respuestas acorde a la conformidad de los analistas, pasan a estar aprobados para llevarse a cabo en el proyecto.

La figura 48 muestra una tabla con el listado de requerimientos aprobados en Ruems Requeriments.

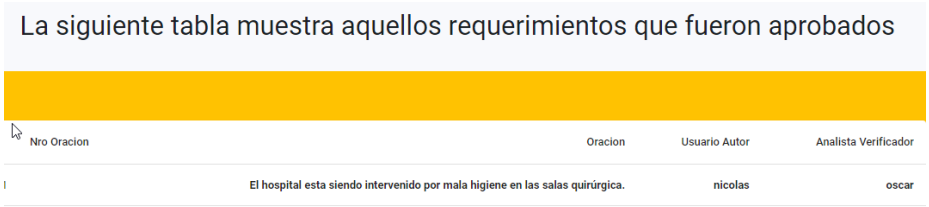

Figura 48: Tabla de requerimientos validados

Ruems Requeriments provee las siguientes operaciones sobre los requerimientos

Aceptar requerimientos. El requerimiento debe estar en estado analizado. En todo momento el analista puede visitar la sección requerimientos analizados dentro de un proyecto determinado. Ruems Requeriments visualizará una tabla con todos los requerimientos que se encuentren en este estado y por cada uno de ellos se muestra que reglas se establecieron como cumplidas e incumplidas tras el procesamiento interno del sistema. Finalmente, el analista puede optar por aceptar el requerimiento; en este caso pasa a estar activo para recibir respuestas por parte de los miembros del proyecto.

La figura 49 ilustra un botón el cual permite aceptar el requerimiento analizado en Ruems Requeriments.

Figura 49: Opción aceptar en la tabla de requerimientos analizados.

Rechazar requerimiento. El requerimiento debe estar en estado analizado. En la tabla de requerimientos analizados también se encuentran disponibles la opción de rechazar requerimientos. Los analistas pueden decidir sobre esta opción cuando el requerimiento no cumple ninguna de las reglas establecidas por Ruems Requeriments, cuando el requerimiento está basado en un dominio distinto al definido por el proyecto o cuando el requerimiento está desarrollado con lenguaje inapropiado. Al rechazar un requerimiento, este queda totalmente fuera del proyecto sin posibilidad de ser activado. El panel de administración también cuenta con una sección de requerimientos rechazados para que el analista pueda ver todos los requerimientos que no han sido aprobados en un determinado proyecto.

La figura 50 ilustra la opción para rechazar un requerimiento en Ruems Requeriments.

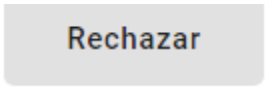

Figura 50: Opción rechazar en la tabla de requerimientos analizados

Reajustar requerimiento. El requerimiento debe estar en estado analizado. El requerimiento no debe tener un reajuste previo pendiente de confirmación. En la tabla de requerimientos analizados también se encuentra disponible la opción de reajustar requerimientos. Por lo general esta opción es utilizada por los analistas cuando el requerimiento incumple muy pocas reglas (por lo general una de tres). Al presionar esta opción la aplicación redirige a una pantalla donde se visualizan, por un lado, el requerimiento el cual va a ser reajustado y por otro lado un campo para ingresar el requerimiento cambiado. El analista

cambiará el requerimiento corrigiendo aquellas partes que incumplieron con algunas de las reglas. Una vez cambiado el requerimiento y se presiona el botón enviar se le envía una notificación al usuario corriente,creador de este requerimiento con la posibilidad de aceptar este cambio o pedir uno nuevo. Una vez enviada la notificación este requerimiento no puede volver a ser reajustado por ningún otro analista hasta que su usuario creador confirme el reajuste previo.

La figura 51 ilustra un botón en el cual se procesa la descripción mencionada anteriormente en ruems requeriments.

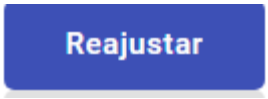

Figura 51: Opción reajustar en la tabla de requerimientos analizados

La figura 52 ilustra un formulario para ingresar el requerimiento para que el mismo pase a ser reajustado.

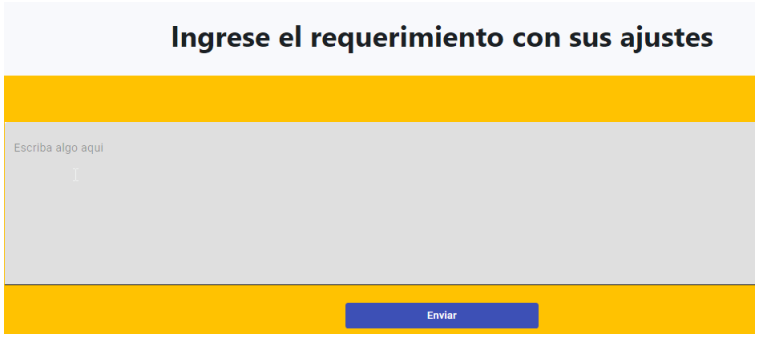

Figura 52: Formulario para reajustar un requerimiento

Validar requerimiento. El requerimiento debe estar en estado aceptado o activo dentro del proyecto. En la tabla de estadísticas se encuentran los requerimientos activos del proyecto (aquellos que pueden recibir respuestas); por cada uno de ellos se dispone de la opción requerimiento validado. El analista puede decidir sobre esta opción si nota que el requerimiento ha llegado a un número de respuestas que muestra una conformidad por parte de los miembros del proyecto. Al confirmar esta operación el requerimiento pasa a estar aprobado dentro del proyecto y ya no podrá recibir más respuestas. Por otra parte, la aplicación cuenta con una sección requerimientos validados donde los analistas pueden llevar un control de todos los requerimientos que han sido aprobados dentro del proyecto en cuestión.

En resumen, la validación de un requerimiento es un proceso crucial que asegura que las necesidades del proyecto han sido debidamente consideradas y

que las respuestas han alcanzado un nivel de acuerdo suficiente. Una vez validado, el requerimiento se consolida como parte del plan de trabajo aprobado, lo que ayuda a garantizar la eficiencia y la calidad en la ejecución del proyecto.

La figura 53 ilustra un botón en el cual se procesa la descripción mencionada anteriormente en ruems requeriments.

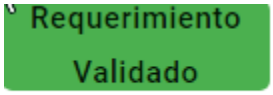

Figura 53: Opción validar en la tabla de estadísticas

8 Evaluación de usabilidad

Para la realización de las pruebas de usabilidad del sistema Ruems Requeriments se utilizó una herramienta de evaluación de usabilidad conocida como la Escala de Usabilidad del Sistema(SUS). En ingeniería de sistemas, la escala de usabilidad del sistema(SUS)es una escala simple de diez ítems que brinda una visión global de las evaluaciones subjetivas de usabilidad.

El SUS como herramienta para proyectos ha sido probado durante casi 30 años de uso y ha demostrado ser un método confiable para evaluar la usabilidad de los sistemas en comparación con los estándares de la industria. Cabe destacar que en comparación con otras pruebas, la escala de usabilidad del sistema es más barata y más rápida. [53]

La escala de usabilidad del sistema nos provee de ante mano un conjunto de 10 preguntas las cuales van a responder los usuarios que van a trabajar con la herramienta.

Experiencia de los usuarios

Se seleccionaron 10 usuarios para realizar las pruebas. Seleccionamos diferentes tipos de profesionales de diversas áreas entre los cuales tuvimos encuenta abogados,informaticos,licenciados en nutrición,marketing y médicos. Elegimos un conjunto de usuarios de diversos sectores con justa razón para que podamos ampliar el dominio de los requerimientos en diferentes ciencias laborales. En líneas generales los resultados fueron bastante favorables, la mayoría hacía hincapié en tener en cuenta diversas variedades de autoayuda o guías para poder terminar de comprender el flujo de ejecución de la herramienta; sin pasar por alto este detalle luego quedaron muy satisfechos con el funcionamiento en sí. Lo más destacable fue que alguno de los usuarios nos brindaron opiniones y consejos en como la herramienta se podría seguir dotando de más funcionalidades para que en un futuro pueda seguir creciendo y automatizándose aún más.

Desarrollo de las pruebas

Se les proporcionó a los usuarios un PowerPoint como guía introductoria con imágenes de la herramienta de lo que deberían realizar. Seguido de esto sé les solicito ingresar a cada uno de los mismos a la herramienta de Ruems Requeriments con su propio usuario y clave. Los usuarios en el sistema de Ruems Requeriments realizaron el proceso de cargar de un requerimiento y posteriormente la validación de otros requerimientos. Finalizada estas tareas se les solicitó a los usuarios realizar un cuestionario en las cuales estaban detalladas las preguntas de la herramienta de usabilidad SUS y obtener de forma eficiente los resultados de las pruebas. Dichas preguntas se encuentran detalladas en el cuadro 8.

Se muestra en una escala con valores del 1 al 5 en donde el valor 5 de puntuación más alta significa que está totalmente de acuerdo con su respuesta y el valor 1 en donde está totalmente en desacuerdo con la respuesta.

Cuadro 8: Modelo de preguntas establecidas por el SUS.

Pregunta
¿Le gustaría usar este sistema con más frecuencia ?
¿ Encontro al sistema muy complejo ?
¿ Penso que el sistema es fácil de usar ?
¿Necesito la ayuda de una persona para poder usar este sistema ?
¿ Las funciones del sistema se entienden y son claras ?
¿Hay muchas inconsistencias en este sistema ?
¿Cree que Otras personas aprenderían a usar este sistema muy rápido ?
¿Le parece sistema parece bastante engorroso de usar?
¿Se sintió satisfecho usando el sistema ?
¿Necesito aprender muchas cosas antes de comenzar a usar el sistema ?

Una vez obtenidos los valores de cada una de las preguntas contestadas por los usuarios según su puntuación. Continuamos con el siguiente paso del SUS, teniendo en cuenta las siguientes consideraciones; para cada una de las **preguntas impares**, restar 1 a la puntuación y para cada una de las **preguntas pares**, restar su valor a 5. Esto se ejemplifica en el cuadro 9 tomando como ejemplo al usuario 1.

Cuadro 9: Tabla con los resultados que muestra la puntuación que le dio el usuario 1 a cada una de las preguntas contestadas con su sumatoria correspondiente.

Usuario1			
Preguntas	**SumaImp**	**SumaPar**	**SumaTotal**
Pregunta1	5-1=4		
Pregunta2		5-1=4	
Pregunta3	4-1=3		
Pregunta4		5-1=4	
Pregunta5	4-1=3		
Pregunta6		5-1=4	
Pregunta7	4-1=3		
Pregunta8		5-1=4	
Pregunta9	5-1=4		
Pregunta10		5-2=3	
SumTotImp	**17**		
SumTotPar		**19**	
SumTaTotal			**36**

Paso final de la prueba

Finalmente, el cuadro 10 resultado determina el resultado final de hacer los cálculos promedios de la suma de cada una de las preguntas según la puntuación que le dieron cada uno de los usuarios luego de probar la aplicación.

En base a los resultados que se probaron anteriormente de cada pregunta obtuvo los valores resultantes de la puntuación de las mismas por parte de la opinión de cada usuario. El mismo resultado además muestra la suma en la columna **Resul** de la puntuación que hizo cada usuario según cada una de las preguntas planteadas por el SUS.

Una vez obtenidos la suma de las preguntas pares e impares por cada uno de los usuarios multiplicamos por 2,5 a cada uno de los resultados obtenidos y con esto construimos la columna **Resul Final** en donde obtenemos el promedio por cada una de las sumas de las preguntas de cada uno de los usuarios.

Finalmente, sacamos los promedios (suma de las filas de resul final / 10 y de la suma de resul/10), con esto obtenemos el score final de la prueba del SUS el cual es **86**.

En base a este resultado se consideró la escala del sus la cual detalla que si se obtiene el **80.3** o superior es una A; a la gente le encanta su sitio y lo recomendará a otras personas. Si se obtiene el **68** o más te da una C; lo estás haciendo bien pero podrías mejorar. Finalmente, si se obtiene **51** o menos te da una gran F; haz de la usabilidad tu prioridad ahora y soluciona este problema rápidamente.

Cuadro 10: Tabla con los resultados que muestra la puntuación que le dio cada usuario a cada una de las preguntas más la sumatoria de puntuación que obtuvo cada usuario.

SUS												
Usuarios	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	Resul	Resul Final
Usuario1	5	1	4	1	4	1	4	1	5	2	32	80
Usuario2	5	2	4	3	4	2	5	1	5	3	32	80
Usuario3	5	2	3	1	3	1	4	1	5	1	34	85
Usuario4	4	3	4	1	4	2	4	2	3	1	30	75
Usuario5	5	1	5	1	5	1	5	1	5	1	40	100
Usuario6	5	1	5	1	5	1	5	1	5	1	40	100
Usuario7	5	1	5	1	5	1	5	1	5	1	40	100
Usuario8	3	1	4	3	3	3	4	2	3	2	26	65
Usuario9	4	4	5	1	5	2	5	2	4	1	33	82,5
Usuario10	4	1	4	1	5	1	4	1	5	1	37	92,5
PromedioFinal											**34,4**	**86**

Conclusión final

El análisis de usabilidad de la herramienta Ruems Requirement ha arrojado un porcentaje de **86**, lo cual la posiciona como una aplicación altamente valorada

según los principios del Sistema de Usabilidad de SUS. Este resultado, que corresponde a una calificación de **A**, indica que la herramienta no solo cumple con los estándares esperados en términos de funcionalidad y facilidad de uso, sino que también ha generado una experiencia positiva para los usuarios.

Al obtener una puntuación de este nivel, se intuye que la mayoría de los usuarios encontraron la aplicación intuitiva, eficiente y agradable de utilizar. Esto se traduce en una percepción general favorable, donde es probable que los usuarios no solo adopten la herramienta para su uso continuo, sino que también la recomienden a otras personas.

Además, este nivel de satisfacción refuerza el potencial de Ruems Requirement para seguir evolucionando, con una base sólida de usuarios satisfechos que podrían participar activamente en la retroalimentación para futuras mejoras. En resumen, esta calificación es un indicativo de éxito en cuanto a diseño centrado en el usuario y representa una excelente base para expandir su alcance en el mercado.

9 Conclusiones y trabajos futuros

A lo largo de la tesina se hizo un análisis e investigación de la construcción de un software desde sus etapas iniciales hasta el producto final como así también las complejidades que pueden surgir durante su desarrollo. Se detalló todo el proceso del ciclo de vida del software centrándonos en la etapa de análisis y obtención de requerimientos, que al ser la primera etapa de este ciclo necesita ser abordada correctamente con la mínima presencia de errores ya que un error en esta etapa puede ser más costoso para las etapas posteriores del ciclo de vida del software. Se diseñó, desarrollo e implemento una aplicación llamada Ruems Requeriments basada en la etapa de análisis y obtención de requerimientos con el uso del procesamiento de lenguaje natural, la cual permite gestionar diferentes requerimientos en formato de kernel sentences por parte de los usuarios. También podemos concluir que nuestra aplicación contribuye a la obtención de métricas de usabilidad de manera temprana. Algunas de las métricas que son posibles de obtener son el nivel de dificultad y satisfacción de los usuarios para la obtención de kernel sentences las cuales considere ser aprobadas. Sin embargo, las muestras tomadas durante el desarrollo de la aplicación ha sido realizada en un grupo reducido de personas. Motivo por el cual, hay que tener en consideración que para que los resultados de la muestra sean realmente representativos, deberán realizarse con un número mayor de usuarios que pertenezcan a diferentes grupos sociales. De esta manera obtendremos una retroalimentación de muchas personas con gran agilidad. Logrando ampliar las métricas que se pueden obtener con la aplicación. Una idea que se especula mucho es implementar un mecanismo que nos permita ejecutar de manera independiente las diferentes pruebas de usabilidad creadas en la aplicación. Así de esta forma podríamos permitir a un mayor número de usuarios participar de las mismas y como resultado obtendremos información más completa y fiable mediante la opinión de un número relevante de personas. Otra idea muy latente y que puede abarcar a un mayor número de usuarios es implementar una aplicación mobile basada en Ruems Requeriments que permita un mejor funcionamiento y uso de los usuarios. Esto puede llevar a tener que incluir nuevas tecnologías a la aplicación. El tener un mayor número de usuarios en el sistema va a permitir escalar la aplicación a diferentes idiomas para que pueda ser utilizada a nivel mundial. Además, se propone expandir la funcionalidad para el registro de requerimientos en RuemsRequeriments que permita que la aplicación, además de evaluar y analizar las kernel sentences registradas, también ofrezca una lista de posibles sugerencias. Esta lista consistiría en la misma kernel sentences escrita de diversas formas y que el usuario tenga la opción de seleccionar la que más le conviene. Esto nos ayudaría a reducir la cantidad de requerimientos reajustados. Asimismo, permitir que RuemsRequeriments se pueda vincular dentro de otra aplicación más conocida para que de esa forma pueda llegar a abarcar a más usuarios y convertirse en una aplicación más popular; como así también que la aplicación tenga la posibilidad de reconocimiento de voz entre otras funciones para que de esa forma pueda ser utilizada por personas con distintas discapacidades. Finalmente, se consideró

la posibilidad de ofrecer a los usuarios la opción de registrarse y autenticarse utilizando cuentas externas a la aplicación Ruems Requeriments, tales como Gmail, Facebook y Outlook. Esta integración no solo tiene como objetivo mejorar la experiencia de usuario al simplificar el proceso de inicio de sesión, sino también reforzar la seguridad de la aplicación mediante el uso de plataformas ampliamente reconocidas y confiables en términos de protección de datos.

Permitir que los usuarios utilicen sus cuentas de servicios externos facilita el acceso, ya que muchos usuarios ya tienen cuentas activas en estos proveedores y prefieren no crear y gestionar nuevas credenciales. De esta manera, se reducen las barreras de entrada y se mejora la comodidad para los usuarios que buscan una experiencia más fluida y rápida al interactuar con la aplicación.

Desde el punto de vista de la seguridad, delegar el proceso de autenticación a plataformas como Gmail, Facebook y Outlook ofrece ventajas significativas. Estas plataformas cuentan con avanzados sistemas de protección de datos, como autenticación en dos pasos, monitoreo constante de actividades sospechosas y encriptación avanzada de la información, lo que garantiza un nivel de seguridad que sería difícil de replicar a nivel individual para cada aplicación. Al aprovechar estas medidas, Ruems Requeriments puede ofrecer un entorno más seguro para sus usuarios, minimizando el riesgo de brechas de seguridad y acceso no autorizado.

Además, la integración con cuentas externas podría facilitar la gestión de usuarios, ya que permite centralizar las credenciales en servicios de confianza que los usuarios ya están acostumbrados a utilizar, lo cual también contribuye a reducir la carga de trabajo relacionada con la administración de contraseñas y el manejo de cuentas.

Referencias

[1] JD Zamfirescu y David Greenspan Aaron Iba. etherpad. https://etherpad.org/, 2008.

[2] Russell L Ackoff et al. Redesigning the future. *New York*, 29, 1974.

[3] Carlos Rene Angarita, Claudia Yamile Gomez Llanez, and Judith del Pilar Rodriguez. Metodo para interactuar con los stakeholders en el proceso de captura de requerimientos de software. *Revista GTI*, 15(41):47–56, 2016.

[4] Antonio Argadoña. *La teoria de los stakeholders y el bien comun*. Universidad de Navarra, Barcelona., 1998.

[5] Michael Arias. La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software. *InterSedes: Revista de las Sedes Regionales*, VI:0–, 01 2005.

[6] Marko Bajec, M Krisper, and R Rupnik. Using business rules technologies to bridge the gap between business and business applications. In *Proceedings of the IFIP 16th World Computer Congress*, pages 77–85, 2000.

[7] Fernando Berzal. El ciclo de vida de un sistema de información. *Recuperado de*, 2004.

[8] Barry W. Boehm. *Software engineering economics*. Prentice-Hall advances in computing science and technology series. Prentice-Hall, Englewood Cliffs, N.J, 1981.

[9] Barry W Boehm. Software engineering economics. *IEEE transactions on Software Engineering*, 10(1):4–21, 1984.

[10] Grady Booch, Douglas L. Bryan, and Charles G. Petersen. *Software engineering with Ada*. The Benjamin/Cummings series in object-oriented software engineering. Benjamin/Cummings, Redwood City, Calif, 3rd ed edition, 1994.

[11] Deakin University Brian J. Garner. *Collaborative Knowledge Management Requirements for Experiential Learning (CKM)*. Proceedings IEEE International Conference on Advanced Learning Technologies), 2001.

[12] Ghezzi Carlo, Jazayeri Mehdi, and Mandrioli Dino. *Fundamentals_of_-Software_Engineering*. The Benjamin/Cummings series in object-oriented software engineering. Prentice Hall PTR, Better World Books, Mishawaka, IN, Estados Unidos de America, 1rd ed edition, 1992.

[13] Guillermo Federico Carrilao Ávila. Construcción semiautomática de un documento lel utilizando técnicas de procesamiento de lenguaje natural, 2021.

[14] Michael Arias Chaves. La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software. *InterSedes: Revista de las Sedes Regionales*, 6(10):1–13, 2005.

[15] Mike Cohn. *User stories applied: for agile software development*. Addison-Wesley signature series. Addison-Wesley, Boston, 2004.

[16] João Carlos de A.R. Gonçalves; Flávia Maria Santoro; Fernanda Araujo Baião. *Collaborative narratives for business rule elicitation*. IEEE International Conference on Systems, Man, and Cybernetics, 2011.

[17] Antonelli Leandro Fernandez Alejandro Ruffolo Nicolas Simon Sansone Emiliano Andres Torres Diego. A collaborative approach to specify kernel sentences using natural language. *Journal of Computational Linguistics*, 2021.

[18] Daniel Díaz Araya, Sandra Oviedo, Leandro Muñoz, and Francisco S Ibañez. Software e innovación: desarrollando productos con hardware y software flexible. In *XVIII Congreso Argentino de Ciencias de la Computación*, 2013.

[19] International Organization for Standardization. Seguridad, gestion de la calidad del software - iso 9126. *New York*, 2015.

[20] Python Software Foundation. python. https://www.python.org, 2001.

[21] Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of software engineering*. Prentice-Hall, Inc., 1991.

[22] Gustavo Daniel Gil. *Herramienta para implementar LEL y escenarios (TILS)*. PhD thesis, Universidad Nacional de La Plata, 2002.

[23] James E. Grove. vbulletin. https://www.vbulletin.com/es/, 2000.

[24] Carl A Gunter, Elsa L Gunter, Michael Jackson, and Pamela Zave. A reference model for requirements and specifications. *IEEE Software*, 17(3):37–43, 2000.

[25] Qingliang Meng; Xinxin Guo. *Identification of key user knowledge source in crowdsourcing innovation mode*. Conference on Service Systems and Service Management (ICSSSM), 2015.

[26] Naomi Unkelos-Shpigel; Irit Hadar. *requirements engineering*. IEEE Fifth International Workshop on Empirical Requirements Engineering (EmpiRE), 2015.

[27] Zellig S. Harris. Co-Occurrence and Transformation in Linguistic Structure. *Language*, 33(3):283, July 1957.

[28] Ramón Ventura Roque Hernández, Juan Manuel Salinas Escandón, and Adán López Mendoza. La complejidad de entender y enfrentar la formación de futuros desarrolladores de software. *CIENCIA ergo-sum, Revista Científica Multidisciplinaria de Prospectiva*, 22(2):153–159, 2015.

[29] Diguital Guide IONOS. Herramientas colaborativas para mejorar la productividad. *Productos IONOS*, 2021.

[30] Felix Freiling Irene Eusgeld and Ralf Reussner. *Dependability Metrics*. Bonn (2003) Lovric, GI-Dagstuhl Research Seminar, Dagstuhl Castle, Germany, 1rd ed edition, 2008.

[31] Jaya Vijayan; G. Raju; Mary Joseph. *Collaborative requirements elicitation using elicitation tool for small projects*. International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES), 2016.

[32] Pericles Loucopoulos and Vassilios Karakostas. *System Requirements Engineering*. McGraw-Hill, Inc., USA, 1995.

[33] D. Kelly M. Barne. Play by the rules. *Journal of Game Studies*, 1997.

[34] Yukio Mizuno. Software quality improvement. *Computer*, 16(03):66–72, 1983.

[35] Juan Carlos Moreno Perez and Arturo Francisco Ramos Perez. *Administración software de un sistema informático*. Ra-Ma, Paracuellos de Jarama, Madrid, 2014. OCLC: 890302552.

[36] A Oliveros, F Balaguer, G Hadad, G Kaplan, G Rossi, J Leite, and V Maiorana. Enhancing a requirements baseline with scenarios. In *Proceedings*, pages 44–53, 1997.

[37] Alejandro Oliveros and Ruben Leandro Antonelli. Técnicas de elicitación de requerimientos. In *XXI Congreso Argentino de Ciencias de la Computación (Junín, 2015)*, 2015.

[38] Alejandro Oliveros, Fernando Danyans, and Matias Mastropietro. Stakeholders en los requerimientos de aplicaciones web. *Tecno Lógicas*, 2014.

[39] Alejandro Oliveros and Sandra Noemi Martinez. La noción de stakeholder en la Ingeniería de Requerimientos. In *XVII Worshop de Investigadores en Ciencias de la Computación*, May 2015.

[40] Oracle. Oracle. https://www.mysql.com/, 1995.

[41] Roger S Pressman, Jess Elmer Murrieta Murrieta, Eloy Pineda Rojas, and Victor Campos Olgun. *Ingenieria de software: un enfoque practico*. McGraw-Hill, Mexico, D.F., 2005. OCLC: 682964292.

[42] Oscar Revelo-Sánchez, César A. Collazos-Ordóñez, and Javier A. Jiménez-Toledo. El trabajo colaborativo como estrategia didáctica para la enseñanza/aprendizaje de la programación: una revisión sistemática de literatura. *Tecno Lógicas*, 21(41):115–134, 2018.

[43] Ramón Ventura Roque-Hernández, Rebeca Díaz-Redondo, and Ana Fernández-Vilas. La especificación de requerimientos de software desde la perspectiva de un nuevo paradigma: los aspectos. *CienciaUAT*, 6(3):56–59, 2012.

[44] Walter Ovidio Sanchez et al. La usabilidad en ingeniería de software: definición y características. *New York*, 2015.

[45] Lady M Sangacha-Tapia, Ricardo Javier Celi-Párraga, Eleanor A Varela-Tapia, and Ivan Leonel Acosta-Guzmán. Modelos probabilísticos ia del procesamiento de lenguaje natural en conversaciones de personas contagiadas con covid-19. *Polo del Conocimiento*, 6(9):1124–1139, 2021.

[46] Material UI SAS. Material ui. https://v4.mui.com/es/, 2014.

[47] Sam Schillace. googledocs. https://docs.google.com/document, 2006.

[48] SensioLabs. Symfony. https://symfony.com/, 2005.

[49] Ruffolo Nicolas Simon and Sansone Emiliano Andres. Ruemsrequeriments, herramienta para relevar especificaciones basadas en lenguaje natural en forma colaborativa, 2024.

[50] Miguel Angel Solinas. *Elicitación y trazabilidad de requerimientos utilizando patrones de seguridad*. PhD thesis, Universidad Nacional de La Plata, 2012.

[51] Facebook Open Source. React. https://es.reactjs.org/, 2023.

[52] Jeff Sutherland, J.J Sutherland, and Victoria Eugenia Gordo del Rey. *Scrum: el revolucionario método para trabajar el doble en la mitad de tiempo*. Editorial Ariel S.A., 2018. OCLC: 1140958394.

[53] Nathan Thomas. How to use the system usability scale (sus) to evaluate the usability of your website.

[54] Yuli Vasiliev. *Natural language processing with Python and spaCy: a practical introduction*. No Starch Press, Inc, San Francisco, 2020.

[55] Ramon Venturalosion. Natural language processing. https://spacy.io/, 2023-01-03.

[56] M Felisa Verdejo. Comprension del lenguaje natural: avances, aplicaciones y tendencias. *Arbor*, 151(595):39, 1995.

[57] Adam D'Angelo y Charlie Cheever. quora. https://es.quora.com/, 2010.

[58] Scott Abel y Greg Kattawar . Spiceworks. https://www.spiceworks.com/, 2006.

[59] Jeff Atwood y Joel Spolsky . stackexchange. https://stackexchange.com/, 2008.

[60] Andy Sacks y Russell Dicker . Answerbag. https://www.answerbag.com/, 2003.

[61] Anthony Youdeowei. *The B_Rule Methodology: A Business Rule Approach to Information Systems Development*. The University of Manchester (United Kingdom), 1997.

10 Anexo 1 Conceptos centrales de la ingeniería de requerimientos

El software es complejo porque el mismo debe tener todas las condiciones, características y restricciones que se especifican en el problema del mundo real. Esta complejidad implica que el software deba adaptarse a varias características fundamentales de cualquier sistema de software, lo que aumentará su confiabilidad. Mientras más características cumpla el software, más confiable será su utilización [28].

Un sistema de software presenta la característica de **correctitud** cuando es funcionalmente correcto y se comporta de acuerdo a la especificación de las funciones (especificación de requerimientos funcionales) que debería proveer. Esta definición de correctitud asume que existe una especificación de requerimientos funcionales del sistema y que es posible determinar en forma no ambigua si las cumple o no. Se presentan diversas dificultades cuando no existe dicha especificación, o si existe, pero está escrita informalmente utilizando, por ejemplo, lenguaje natural por lo que es posible que contenga ambigüedades. La correctitud es una propiedad matemática que establece la equivalencia entre el software y su especificación, por lo que cuanto más riguroso se haya sido en la especificación, más precisa y sistemática podrá ser su evaluación. Esta misma puede ser evaluada mediante diversos métodos, algunos de enfoque experimental como las pruebas, otros de enfoque analítico como verificación formal de la correctitud.

Esta definición de correctitud no toma en consideración que la especificación puede ser incorrecta por contener inconsistencias internas [12].

El término **usabilidad**, es un atributo cualitativo definido comúnmente como la facilidad de uso en cualquier sistema que interactúe con un usuario. Otra definición del término es que se trata de una medida de la calidad de la experiencia que tiene un usuario cuando interactúa con un producto o sistema. Esto es medido a través del estudio de la relación que se produce entre las herramientas y quienes las utilizan. Esto último se realiza para determinar la eficiencia en el uso de los diferentes elementos ofrecidos en las pantallas. Como así también para la efectividad en el cumplimiento de las tareas que se pueden llevar a cabo a través de ellas [44].

Nelson definen a la **flexibilidad** de la tecnología como las características de la tecnología que permiten habilitar ajustes u otros cambios a los procesos de negocio. Desde la perspectiva del desarrollo de software la flexibilidad es una temática actual. Una gran cantidad de técnicas y métodos tiene a la flexibilidad como objetivo. Por ejemplo, el desarrollo dirigido por modelos, las líneas de productos de software, la programación generativa son paradigmas que intentan construir fábricas de software.Estas permiten generar software a partir de solo describir un modelo que la fábrica de software interpretada para generar una aplicación. Esto es claramente un proceso de desarrollo flexible que permite modelar rápidamente los cambios que se producen en el ambiente y generar el código ejecutable que satisface a las nuevas necesidades [18].

Según la norma iso/iec 9126 [30] la **portabilidad** se define como la característica que posee un software para ejecutarse en diferentes plataformas. El código fuente del software es capaz de reutilizarse en vez de crearse un nuevo código cuando el software pasa de una plataforma a otra. A mayor portabilidad menor es la dependencia del software con respecto a la plataforma. El entorno puede incluir entornos organizacionales, de hardware o de software.

Según la norma iso/iec 9126 [30] la **seguridad** es la capacidad del software para proteger los datos y la información.El fin de esto es que personas no autorizadas nunca puedan ingresar al sistema, leer o modificar los datos. Las personas que sí estén autorizadas no se les niega el acceso al sistema ni a la información que necesiten. Esta característica garantiza la integridad de los datos a nivel de accesos, modificaciones no permitidas y transacciones. También aplica para la transmisión de datos. Depende de la habilidad del software en la prevención de accesos no autorizados, accidentales o intencionales, por parte de personas, de programas ajenos (por ejemplo, virus) o de cualquier agente externo no autorizado [19].

Ciclo de vida del Software

En el proceso de desarrollo de software normalmente se involucra lo que se conoce **como ciclo de vida del desarrollo del software**. Aquí se agotan una serie de faces o etapas, con sus respectivas ventajas o desventajas según él paradigma adoptado.Hay diversos modelos y alguno de ellos tienen un enfoque de desarrollo de software seguro. [21]. La figura 54 representa un esquema sobre las etapas del ciclo de vida del software.

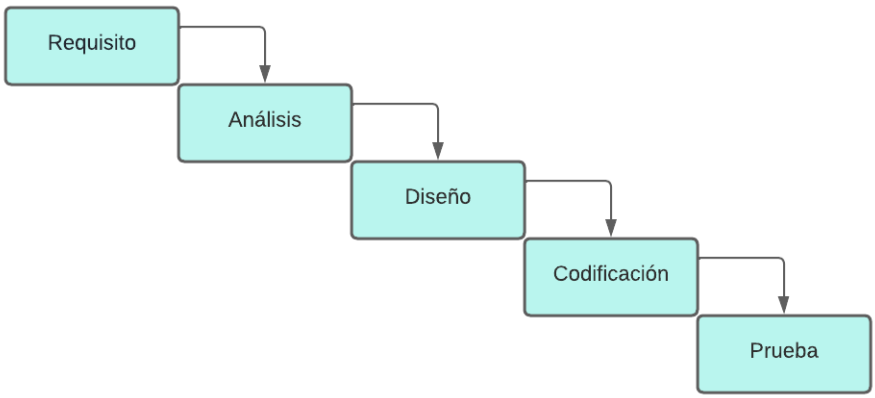

Figura 54: Etapas del ciclo de vida del software

En la etapa de **requisito** se incluyen tareas como la determinación del ámbito del proyecto, un estudio de viabilidad, análisis de riesgos, costes estimados, asignación de recursos en las distintas etapas. Son tareas que influyen en el

éxito del proyecto, por eso es necesaria una planificación inicial. En esta etapa es fundamental conocer, lo que el usuario quiere,cómo lo quiere, cuándo y por qué. Para lograr los objetivos de esta etapa es fundamental conseguir una buena comunicación con el cliente.

La comunicación es la base para la obtención de las necesidades del cliente. Es la principal fuente de error por falta de procedimientos y guías formales, falta de participación del usuario, mala interpretación de las necesidades y falta de comunicación. A continuación, en la figura 55 se muestra un dibujo representando distintos escenarios basándonos en a los problemas de comunicación.

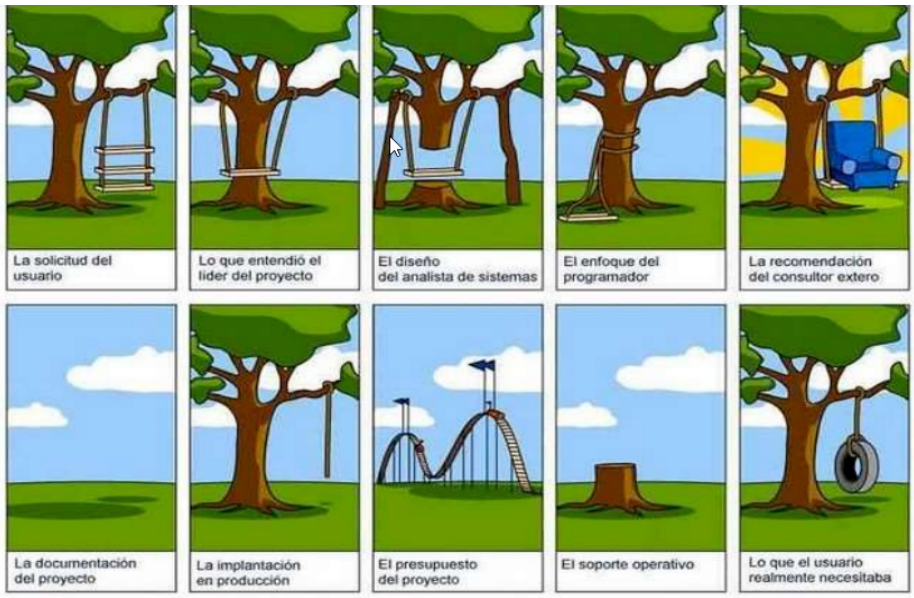

Figura 55: Problemas de la comunicación

Después de la recolección de requisitos, el equipo idea un plan para procesar el software.

En la etapa de **análisis** lo que debemos hacer para construir un sistema de información es averiguar qué es exactamente lo que tiene que hacer el sistema. La etapa de análisis en el ciclo de vida del software corresponde al proceso mediante el cual se intenta descubrir qué es lo que realmente se necesita. Además, se llega a una comprensión adecuada de los requerimientos del sistema. La etapa de análisis es importante porque si no sabemos con precisión qué es lo que se necesita, ningún proceso de desarrollo nos permitirá obtenerlo. El problema es que, en principio, puede que ni nuestro cliente sepa en primera instancia qué es exactamente lo que necesita. Por tanto, deberemos ayudarlo a averiguarlo con ayuda de distintas técnicas, por ejemplo; técnicas de elicitación de requerimientos, herramientas de modelado de sistemas, metodologías de

análisis de requerimientos, entre otros. El equipo analiza si el software puede hacerse para cubrir todos los requisitos del usuario y si hay alguna posibilidad de que el software ya no sea necesario. Se investiga si el proyecto es viable a nivel financiero, práctico, y a nivel tecnológico para que la organización acepte la oferta.

Los modelos que se utilizan en la etapa de **diseño** representan las características del sistema que nos permitirán implementarlo de forma efectiva. Un software bien diseñado debe exhibir determinadas características. Su diseño debería ser modular en vez de monolítico. Sus módulos deberían ser cohesivos (encargarse de una tarea concreta y solo de una) y estar débilmente acoplados entre sí (para facilitar el mantenimiento del sistema). Cada módulo debería ofrecer a los demás unas interfaces bien definidas y ocultar sus detalles de implementación.Por último,debe ser posible relacionar las decisiones de diseño tomadas con los requerimientos del sistema que las ocasionaron. Igual que en la etapa de análisis creábamos distintos modelos en función del aspecto del sistema en que centrábamos nuestra atención. El diseño de un sistema de información también presenta distintas facetas. Por un lado, es necesario abordar el diseño de la base de datos. Por otro lado, también hay que diseñar las aplicaciones que permitirán al usuario utilizar el sistema de información. Tendremos que diseñar la interfaz de usuario del sistema y los distintos componentes en que se descomponen las aplicaciones.

Una vez que sabemos qué funciones debe desempeñar nuestro sistema de información (análisis) y hemos decidido cómo vamos a organizar sus distintos componentes (diseño), se pasa a la etapa de **codificación**. Antes de escribir una sola línea de código (o de crear una tabla en nuestra base de datos) es fundamental haber comprendido bien el problema que se pretende resolver. También es importante haber aplicado principios básicos de diseño que nos permitan construir un sistema de información de calidad. Para la fase de implementación debemos seleccionar las herramientas adecuadas. Es decir, un entorno desarrollo, un lenguaje de programación. La elección de estas herramientas dependerá en gran parte de las decisiones de diseño que hayamos tomado hasta el momento y del entorno en el que nuestro sistema deberá funcionar. A la hora de programar, debemos procurar que nuestro código no resulte indescifrable. Para que nuestro código sea legible, hemos de evitar estructuras de control no estructuradas. Elegir cuidadosamente los identificadores de nuestras variables. Seleccionar algoritmos y estructuras de datos adecuadas para nuestro problema. Mantener la lógica de nuestra aplicación lo más sencilla posible. Comentar adecuadamente el texto de nuestros programas. Además, debemos facilitar la interpretación visual del código mediante el uso de sangrías y líneas en blanco.

La etapa de **pruebas** tiene como objetivo detectar los errores que se hayan podido cometer en las etapas anteriores del proyecto (y, eventualmente, corregirlos). Consiste en hacerlo antes de que el usuario final del sistema los tenga que afrontar. De hecho, una prueba es un éxito cuando se detecta un error. La

búsqueda de errores que se realiza en la etapa de pruebas puede adaptar distintas formas, en función del contexto y de la fase del proyecto. Las pruebas de unidad sirven para comprobar el correcto funcionamiento de un componente concreto de nuestro sistema. Las pruebas de integración son las que se realizan cuando vamos juntando los componentes que conforman nuestro sistema y sirven para detectar errores en sus interfaces [7].

El desarrollo de software es una tarea que involucra complejidad proveniente del área de aplicación del producto final, de los factores técnicos y humanos y de la gestión del proceso. En cuanto a recursos técnicos para llevar a cabo el proceso de desarrollo de software será necesario contar con el hardware requerido en este desarrollo, tales como computadora con sistemas operativos los más actuales posibles, conexión estable a internet,editores de código que faciliten las tareas a los desarrolladores, servidores para hacer pruebas con el software, contar con una base de datos,herramientas para mantener comunicado al equipo, servicios para compartir el código entre todo el equipo. Para tener todos estos recursos disponibles para el desarrollo de software, este mismo debe presentar una remuneración económica inicial.

El desarrollo de software es una actividad que incorpora complejidad de diversas áreas. Una aplicación no solo debe implementar todos los requisitos funcionales, sino también los no funcionales como por ejemplo robustez, capacidad de respuesta, mantenibilidad, verificabilidad, escalabilidad, seguridad, soporte, monitoreo y recuperación de desastres con el propósito de satisfacer no sólo las necesidades inmediatas de las organizaciones, sino ser a la vez lo suficientemente flexible como para adaptarse a las crecientes y cambiantes necesidades futuras de el mismo. Es por ello que si un desarrollador de software debe realizar una aplicación para automatizar el proceso de generación de una nómina, es su obligación conocer todos los detalles del proceso de nómina con el que va a trabajar. Debe conocer,manejar, e implementar con algoritmia los conceptos de contabilidad, impuestos, leyes fiscales, prestaciones, deducciones,criterios gerenciales, recursos financieros y humanos, entre otros muchos, sin ser contador de profesión ni tener un perfil académico afín a esa área.

Además, debe conocer los métodos de desarrollo de software, ya que va a analizar, diseñar, implementar, probar y mantener una aplicación completa. Debe también conocer notaciones y paradigmas de análisis y diseño para documentar y comunicar procesos y resultados, debe conocer por lo menos un lenguaje de programación, un manejador de bases de datos, y las diferentes estructuras de datos que puede utilizar.Por si esto fuera poco, además debe tener la habilidad de congeniar con los usuarios, pues los usuarios son piezas clave.

Por otra parte, también debe ser capaz de trabajar en equipo, pues los grandes sistemas rara vez se desarrollan por una sola persona. La utilización de equipos para la organización del desarrollo de software incrementa la relevancia de la influencia de los colegas de trabajo comparado contra el empleo de herramientas de tecnologías de información enfocadas a la productividad individual.

A lo largo de los años se ha podido constatar que los requerimientos o requisitos son la pieza fundamental en un proyecto de desarrollo de software, ya que marcan el punto de partida para actividades como la planeación, básicamente en lo que se refiere a las estimaciones de tiempos y costos, así como la definición de recursos necesarios y la elaboración de cronogramas que será uno de los principales mecanismos de control con los que se contará durante la etapa de desarrollo. Además, la especificación de requerimientos es la base que permite verificar si se alcanzaron o no los objetivos establecidos en el proyecto ya que estos son un reflejo detallado de las necesidades de los clientes o usuarios del sistema y es contra lo que se va a estar verificando si se están cumpliendo las metas trazadas. Es muy frecuente escuchar entre los conocedores del desarrollo de software (programas de computadoras), que un gran número de los proyectos de software fracasan por no realizar una adecuada definición, especificación, y administración de los requerimientos. Dentro de esa mala administración se pueden encontrar factores como la falta de participación del usuario, requerimientos incompletos y el mal manejo del cambio a los requerimientos.

Durante la etapa de especificación de requerimientos se pueden presentar muchos inconvenientes los cuales son importantes de identificar y prevenir, en el cuadro 11 se presentan los problemas más comunes en este proceso.

Cuadro 11: Problemas de especificación

Problema
Los requerimientos no son obvios y vienen de muchas fuentes
Son difíciles de expresar en palabras (el lenguaje es ambiguo)
La cantidad de requerimientos en un proyecto puede ser difícil de manejar
Un requerimiento puede cambiar a lo largo del ciclo de desarrollo
El usuario no puede explicar lo que hace
Tiende a recordar lo excepcional y olvidar lo rutinario
Hablan de lo que no funciona
Los usuarios tienen distinto vocabulario que los desarrolladores
Usan el mismo término con distinto significado

Boehm [8], establece que reparar errores en las etapas posteriores a la de requerimientos puede llegar a ocasionar un costo de entre 100 a 200 veces mayor que hacerlo en la de definición de requerimientos. Pero las palabras de Ackoff [2] son las que mejor sintetizan la importancia de los requerimientos: “Fallamos más a menudo porque resolvemos el problema incorrecto, que porque obtenemos una solución inadecuada del problema correcto”.

Entre los múltiples trabajos que destacan la importancia de los requerimientos para la producción de software de calidad y por extensión, software seguro, se

destaca a Mizuno,quien organiza los errores que se pueden producir a lo largo del proceso de desarrollo de software, en un modelo de catarata de errores [34]. El modelo muestra que una especificación incorrecta de requerimientos puede ocasionar errores ocultos dentro del sistema. Sí trasladamos este modelo a una aplicación con requerimientos de seguridad, el resultado se agudiza, desde el momento en que se sabe que dicha aplicación requiere cumplir objetivos de seguridad. Es entendible que una incorrecta especificación de requerimientos de seguridad y un diseño incorrecto de los mismos se traduzca en aplicaciones más vulnerables por desconocimiento ya no de errores, sino de amenazas ocultas, amenazas no identificadas o amenazas identificadas que no son tratadas adecuadamente por falta de interés. Situación a la que se ha llegado por un inadecuado tratamiento de la seguridad desde las etapas tempranas del proceso de desarrollo de software [50]. La figura 56 representa un diagrama que encadena los diferentes errores planteados por Mizuno.

Conflictos entre los usuarios de un software

En la actualidad los requerimientos se extraen de las personas conocidas como stakeholders (concepto que profundizaremos más adelante) en el mundo del software; de forma hablada o escrita en documentos(o incluso podemos obtener los requerimientos de otras aplicaciones ya en funcionamiento), ya que cada uno tiene su punto de vista y su forma de razonar el dominio del problema. Es necesario mantener cierta organización entre los stakeholders de un mismo dominio, que puedan poseer distinto conocimiento, ya que esto puede traer varios conflictos. Puede suceder que existan stakeholder interesados en una parte del dominio y por otro lado, tener otros stakeholders con otro tipo de interés y conocimiento del dominio. Como cada parte interesada tiene sus propios conocimientos, la especificación de requerimientos puede resultar conflictiva en el proceso de análisis y obtención. Es por eso que se necesita de un procedimiento o estrategia para unificar los distintos puntos de vista e interpretaciones en el software para lograr una especificación de requerimientos más comprensible. Esto lleva a concluir en que la tarea de recolectar requerimientos no es una tarea sencilla, sino que conlleva un gran proceso de análisis y es en lo que se va a hacer mucho hincapié en gran parte de este capítulo, en como llevar un correcto proceso de obtención de requerimientos para que todas las partes involucradas en el mismo puedan satisfacer sus objetivos. Un posible ejemplo de esta última idea, puede ser un sistema de venta de productos alimenticios donde por un lado tenemos a los vendedores los cuáles tienen su punto de vista o interés en términos de llevar cuantas ventas se realizaron,qué productos se vendieron,como se vendió a que precio entre otros. Y por el otro lado tenemos el perfil de los compradores, los cuales simplemente les interesa poder comprar el producto,ver si hay alguna oferta y el medio de pago que se puede realizar. Esto concluye a que tenemos diferentes stakehdolers en el mismo sistema con diferentes puntos de vista y es necesario poder lograr una unificación entre los distintos puntos de vista y lo cual no es una tarea sencilla.

Introducción a los requerimientos

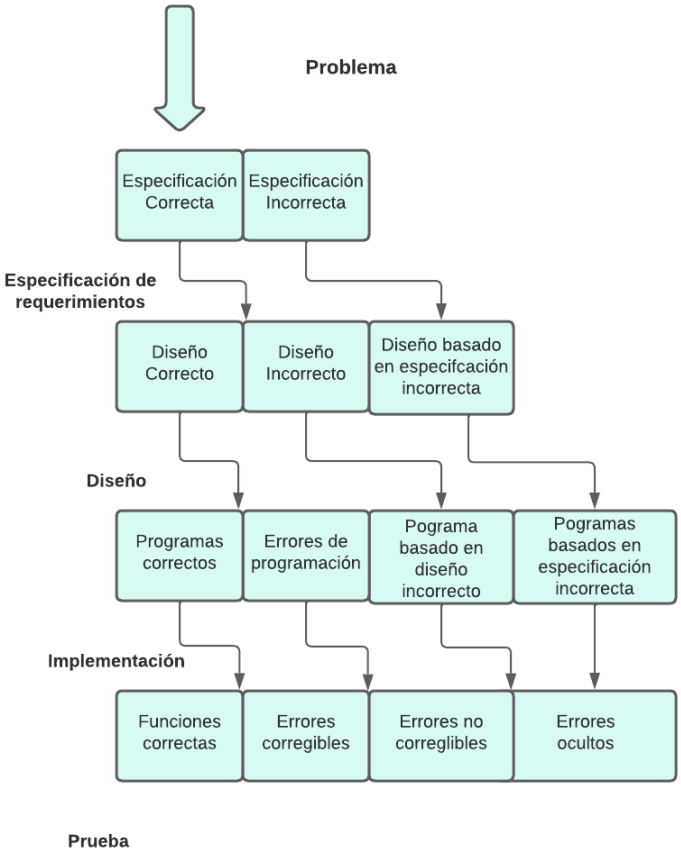

Figura 56: Catarata de errores de mizuno

A lo largo de los años se ha podido constatar que los requerimientos o requisitos son la pieza fundamental en un proyecto de desarrollo de software, ya que marcan el punto de partida para actividades como la planeación, básicamente en lo que se refiere a las estimaciones de tiempos y costos, así como la definición de recursos necesarios y la elaboración de cronogramas que será uno de los principales mecanismos de control con los que se contará durante la etapa de desarrollo. Es muy frecuente escuchar entre los conocedores del desarrollo de software (programas de computadoras), que un gran número de los proyectos de software fracasan por no realizar una adecuada definición, especificación, y administración de los requerimientos. Dentro de esa mala administración se pueden encontrar factores como la falta de participación del usuario, requerimientos incompletos y el mal manejo del cambio a los requeri-

mientos.

Los **Requerimientos** fueron definidos por la IEEE como [IEEE90], la misma los define en el cuadro 12.

Cuadro 12: Requerimientos según IEEE

Definición
Condición o capacidad requerida por el usuario para resolver un problema o alcanzar un objetivo
Condición o capacidad que debe satisfacer o poseer un sistema o un componente de un sistema para satisfacer un contrato, un estándar, una especificación u otro documento formalmente impuesto
Representación documentada de una condición o capacidad como en 1 o 2

En el cuadro 13 se visualizan definiciones de requerimientos brindadas por diferentes autores.

Habiendo analizado las distintas definiciones de requerimientos brindadas por los autores antes mencionados podemos extraer ideas en común de las definciones y establecer una definición de requerimientos más general y completa. Entonces podemos decir que los **requerimientos** son una descripción de una condición o capacidad que debe cumplir un sistema de software, ya sea derivada de una necesidad de usuario identificada, o bien, estipulada en un contrato, estándar, especificación u otro documento formalmente impuesto al inicio del proceso de desarrollo. Entonces podemos concluir que un requerimiento es una propiedad o restricción determinada con precisión que un producto de software debe satisfacer [22] y cubre las siguientes características detalladas en el cuadro 14.

Los requerimientos de software pueden dividirse en 2 categorías, requerimientos funcionales y requerimientos no funcionales y se caracterizan en el cuadro 15.

Regla De Negocio en los requerimientos

Una regla de negocio es una descripción que define o restringe algún aspecto del negocio, el cual se pretende que imponga la estructura o controle o influya en el comportamiento del mismo. Muchos autores tratan de dilucidar y acomodar la definición del concepto de reglas de negocio, como se puede observar en el cuadro 16.

De las definiciones anteriormente analizadas podemos extraer las siguientes ideas comunes sobre una regla de negocio que se detallan en el cuadro 17.

Especificación de los requerimientos

La especificación de requerimientos es una actividad primordial en el desarrollo

Cuadro 13: Ingeniería de requerimientos según distintos autores

Autor	Definición
Zave [24]	1) Rama de la ingeniería del software que trata con el establecimiento de los objetivos, funciones y restricciones de los sistemas software. 2) Asimismo, se ocupa de la relación entre estos factores con el objeto de establecer especificaciones precisas
Boehm [9]	Ingeniería de requerimientos es la disciplina para desarrollar una especificación completa, consistente y no ambigua, la cual servirá como base para acuerdos comunes entre todas las partes involucradas y en dónde se describen las funciones que realizará el sistema
Loucopoulos [32]	Trabajo sistemático de desarrollo de requisitos, a través de un proceso iterativo y cooperativo de análisis del problema, documentando los resultados en una variedad de formatos y probando la exactitud del conocimiento adquirido
Leite [36]	Es el proceso mediante el cual se intercambian diferentes puntos de vista para recopilar y modelar lo que el sistema va a realizar. Este proceso utiliza una combinación de métodos, herramientas y actores, cuyo producto es un modelo del cual se genera un documento de requerimientos

de sistemas de software que orienta en todo momento las acciones del equipo de trabajo; la manera de realizarla varía de acuerdo a la naturaleza del proyecto y a las prácticas adoptadas por los desarrolladores. Una forma relativamente reciente de especificar requerimientos es mediante el paradigma orientado a aspectos: un enfoque que propone una manera diferente de conceptualizar las características importantes del software. Hay diferentes técnicas para especificar los requerimientos de software, las dos herramientas más habituales son por medio de historias de usuarios y casos de uso [43].

Casos de Uso

Un caso de uso es la descripción de una secuencia de interacciones entre el sistema y uno o más actores en la que se considera al sistema como una caja negra. Los casos de uso son una técnica para especificar el comportamiento de un sistema. Los actores son personas u otros sistemas que interactúan con el sistema cuyos requerimientos se están describiendo. Un actor puede participar en varios casos de uso y un caso de uso puede estar relacionado con

Cuadro 14: Características de un requerimiento

Característica	Descripción
Especificación escrita	Como todo contrato o acuerdo entre dos partes
Posible de probar o verificar	Si un requerimiento no se puede comprobar,entonces ¿cómo se sabe si se cumplió con él o no?
Conciso	Un requerimiento es conciso si es fácil de leer y entender. Su redacción debe ser simple y clara para aquellos que vayan a consultarlo en un futuro
Completo	Un requerimiento está completo si no necesita ampliar detalles en su redacción, es decir, si se proporciona la información suficiente para su comprensión
Consistente	Un requerimiento es consistente si no es contradictorio con otro requerimiento
No ambiguo	Un requerimiento no es ambiguo cuando tiene una sola interpretación. El lenguaje usado en su definición, no debe causar confusiones al lector

varios actores; se puede ver un posible ejemplo en la figura 4. Los casos de uso presentan ciertas ventajas sobre la descripción meramente textual de los requerimientos funcionales [Firesmith97], ya que facilitan la elicitación de requerimientos y son fácilmente comprensibles por los clientes y usuarios. Además, pueden servir de base a las pruebas del sistema y a la documentación para los usuarios. La especificación de requerimientos mediante la técnica de casos de uso se lleva a cabo mediante dos componentes. **Diagrama de casos de Uso**: ilustra las interacciones entre el sistema y los actores. **Escenarios** (narración del CU): descripción de la interacción entre el actor y el sistema para realizar la funcionalidad. La figura 57 representa un diagrama base de CU.

Cuadro 15: Requerimientos funcionales y no funcionales

Requerimientos funcionales	Requerimientos no funcionales
Son los que definen las funciones que el sistema será capaz de realizar, describen las transformaciones que el sistema realiza sobre las entradas para producir salidas. Es importante que se describa él ¿Qué? y no el ¿Cómo?, se deben hacer esas transformaciones. Estos requerimientos al tiempo que avanza el proyecto de software se convierten en los algoritmos, la lógica y gran parte del código del sistema	Tienen que ver con características que de una u otra forma puedan limitar el sistema, como por ejemplo, el rendimiento (en tiempo y espacio), interfaces de usuario, fiabilidad (robustez del sistema, disponibilidad de equipo), mantenimiento, seguridad, portabilidad, estándares, etc [14]

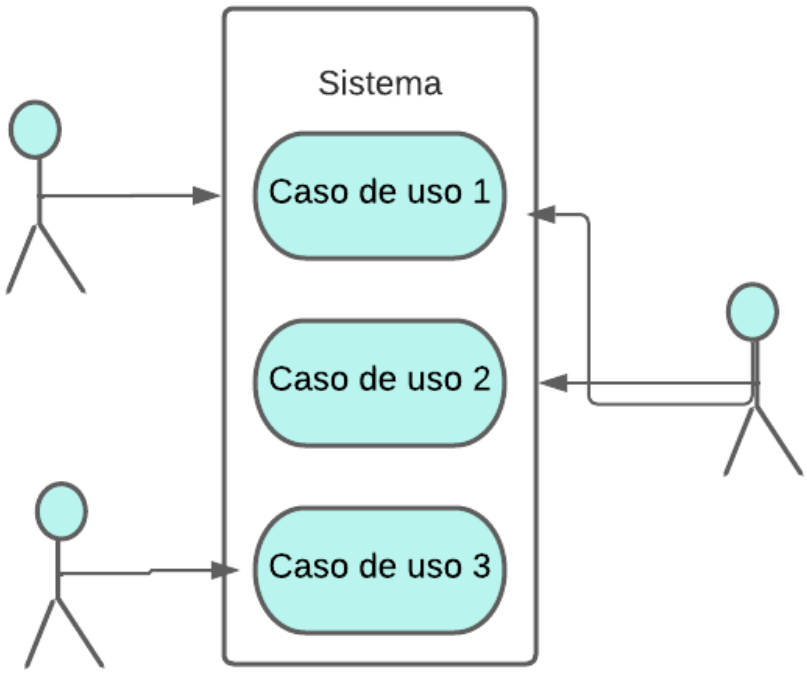

Figura 57: Ejemplo de un diagrama ilustrativo de caso de uso

Cuadro 16: Reglas de negocio según distintos autores.

Autor	Definición
Barne,1997 [33]	Son reglas que contienen algunos casos que impactan el negocio, y la interpretación de las reglas puede tener un impacto cualitativo del sistema de información para el futuro
Youdeowei,1997 [61]	Son requerimientos que se obtienen desde los objetivos de negocio de una empresa
Bajec [6]	2000) Son sentencias explícitas que estipulan una condición que debe existir en el ambiente de información del Negocio para extraer la información desde su propio ambiente siendo consistente con las políticas del negocio. 2006) Son sentencias que restringen o definen algunos aspectos del negocio, esto trata de asegurar la estructura del negocio o de controlar o influir en el comportamiento del mismo

La figura 58 representa un escenario básico de CU.

Historia de usuario es una representación de un requisito de software escrito en una o dos frases utilizando el lenguaje común del usuario. Si bien el estilo puede ser libre, la historia de usuario debe responder a tres preguntas:¿quién se beneficia?,¿qué se quiere?,¿cuál es 0el beneficio?. Entonces el esquema de una historia de usuario debe ser el siguiente: **Como (rol) quiero (algo) para poder (beneficio)**.

Las historias de usuarios se escriben respetando el siguiente formato:

-Id: cada historia de usuario debe tener su propio identificador.

-Título: el título de la historia de usuario índica la funcionalidad o el requerimiento a describir.

-Descripción: es una breve introducción a la historia de usuario siguiendo el esquema definido anteriormente.

-Criterios de prueba de aceptación: indica todos los criterios o pruebas de aceptación que se deben cumplir para llevar a cabo la historia de usuario.

-Dependencia: indica si la historia de usuario depende de otra.

La tabla 18 representa un, template general para una historia de usuario.

Relevamiento de requerimientos (elicitación)

La producción de los requerimientos del software es sustancialmente interacti-

Cuadro 17: Definiciones y restricciones del negocio

Regla
Una sentencia que define o restringe algunos aspectos del negocio
Establece restricciones a la estructura del negocio, controlando o influyendo en el comportamiento del mismo
No podrá ser fraccionada o descompuesta en RN más detalladas
En caso de ser reducida perdería información importante sobre el negocio

va e iterativa, el punto de arranque es el conocimiento que resulta del proceso de elicitación para el que se dispone de una serie de técnicas. Vimos anteriormente que cuando tenemos usuarios de varias áreas con distintos conocimientos del dominio, esto puede traer conflictos para construir la especificación de requerimientos. Hemos mencionado que necesitamos de técnicas para solucionar o reducir estos conflictos. El resultado del proceso de elicitación es todo el conocimiento relevante necesario para producir un modelo de los requerimientos de un dominio de problema, para el que se dispone de una serie de técnicas.

Técnicas de elicitación de requerimientos

Existen dos tipos de técnicas de elicitación de requerimientos [37]. Una técnica es la de **métodos discretos** los cuales son menos perturbadores que otras formas de averiguar los requerimientos. Se consideran insuficientes para recopilar información cuando se utilizan por sí solos, por lo que deben utilizarse junto con uno o varios de los métodos. Utilizar diferentes métodos para acercarse a la organización es una práctica inteligente mediante la cual podrá formarse un panorama más completo de los requerimientos. Entre los más conocidos esta el **muestreo de la documentación, los formularios y los datos existentes**. Se caracterizan por ser documentos que describen la funcionalidad del negocio que está siendo analizada. Tipos de documentos que pueden enseñar algo acerca del sistema, pueden ser organigramas,memos notas internas, minutas, registros contables. Además permiten conocer el historial que origina el proyecto. Otro método muy utilizado es el de **investigación y visitas al lugar**. Se centra en investigar el dominio y realizar patrones de soluciones (mismo problema en otra organización) como asi también buscar problemas similares en internet o bien también se podría consultar otras organizaciones.Finalmente hay un tercer método de muy cotidiano uso que es el de **observación del ambiente de trabajo**. Consiste en que el analista se convierte en observador de las personas y actividades con el objeto de aprender acerca del sistema. El proceso de observación del ambiente de trabajo consiste en determinar quién y cuándo será observado,obtener el permiso de la persona y explicar el porqué será observado. Para lograr esto se necesita mantener el perfil bajo,tomar

Nombre	**Iniciar Sesión**	
Actores	usuario no logueado	
Precondiciones	-	
Postcondiciones/ Objetivo	el usuario se loguea correctamente en el sistema.	
Camino Feliz	**Acción del actor:**	**Acción del sistema:**
	1 El usuario ingresa usuario y contraseña	2 el sistema valida los datos y el sistema lo lleva al panel correspondiente
Curso Alterno	2.1 El usuario y contraseña ingresados son incorrecto vuelve al paso 1.	
Prioridad	alta	

Figura 58: Ejemplo de un escenario de caso de uso

nota de lo observado,revisar las notas con la persona apropiada y no interrumpir a la persona en su trabajo. Se detallan las ventajas en el cuadro 19 y las desventajas en el cuadro 20.

La otra técnica de elicitación de requerimientos es la de **métodos interactivos**. Hay métodos interactivos que pueden usarse para obtener los requerimientos de los miembros de la organización aunque son distintos en su implementación, estos métodos tienen muchas cosas en común. La base es hablar con las personas en la organización y escuchar para comprender.Los más conocidos son los **cuestionarios** y **entrevistas**. El uso de cuestionarios permite a los analistas reunir información proveniente de un grupo grande de personas. El empleo de formatos estandarizados para las preguntas puede proporcionar datos más confiables que otras técnicas; por otra parte, su amplia distribución asegura el anonimato de los encuestados, situación que puede conducir a res-

Cuadro 18: Ejemplo de una historia de usuario.

ID	Las historias deben tener un número asociado
TITULO	Agregar un nuevo destino
DESCRIPCIÓN	Como jefe de logística quiero poder agregar un nuevo destino ingresando su nombre para poder ampliar el catálogo de los destinos a los que llega la empresa.
CRITERIOS-PRUEBAS DE ACEPTACIÓN	▪ Tener el formulario para completar el campo del nombre del destino. ▪ Probar el ingreso de un destino que ya existe y comprobar que se informa el error. ▪ Probar el ingreso de un destino dejando el campo nombré en blanco () y comprobar que se informa el error. ▪ Probar el ingreso de un destino que no existe y revisar que se haya agregado al catálogo de destinos.
DEPENDENCIA	(indicar el número de las historias de las cuales depende)

puestas más honestas.

El inconveniente es que la respuesta puede ser limitada, ya que es posible que no tenga mucha importancia para los encuestados llenar el cuestionario. Es recomendable conseguir apoyo de la alta dirección para solicitar a las personas de la organización que contesten el cuestionario.

Al igual que con las entrevistas, se debe seleccionar a los encuestados.El analista debe asegurar que el conocimiento y experiencia de estos califiquen para dar respuestas a las preguntas. Dentro de los cuestionarios tenemos tipos de preguntas cerradas y abiertas. Las **abiertas** son las que dejan abiertas todas las posibles opciones de respuesta. **Posible ejemplo**: describa los problemas que experimenta en la actualidad con los informes de las salidas, en su opinión, ¿qué tan útiles son los manuales de usuario para la aplicación de contabilidad del sistema actual?. Por otro lado, las **cerradas** limitan o cierran las opciones de respuestas disponibles. **Posible ejemplo**: ¿es útil el reporte que utiliza actualmente? SI NO.

Aplicación de cuestionarios

El uso de cuestionarios debería estar dedicado para el caso en donde las perso-

Cuadro 19: Ventajas del ambiente de trabajo

Ventaja
Datos confiables
El analista puede ver exactamente lo que se hace (tareas difíciles de explicar con palabras)
Análisis de disposiciones físicas, tránsito, iluminación, ruido
Económica en comparación con otras técnicas

Cuadro 20: Desventajas del ambiente de trabajo

Desventaja
La gente se siente incómoda siendo observada
Algunas actividades del sistema pueden ser realizadas en horarios incómodos.
Las tareas están sujetas a interrupciones
Tener en cuenta que la persona observada puede estar realizando las tareas de la forma “correcta” y no como lo hace habitualmente

nas están dispersas geográficamente ya sea en diferentes oficinas o ciudades y los mismos pueden responder sin problemas. Este mecanismo es efectivo cuando queremos analizar un problema grande mediante opiniones de diversas personas que se encuentran involucradas desde distintos puntos o sitios. Con respecto a los cuestionarios podemos obtener una serie de ventajas las cuales pueden ser de respuestas rápida,anonimas y de un presupuesto económico logrando que los usuarios no tengan que trasladarse. Pero en su contra partida podemos toparnos también con una serie de desventajas como obtener un número bajo de respuesta, que el usuario cuestionado no responda todas las preguntas o no podemos percibir su análisis corporal y todo esto puede llevar a que un cuestionario sea difícil de preparar.

En la figura 59 podemos observar las características de los tipos de preguntas que se realizan en un cuestionario.

Las **entrevistas** son una técnica de exploración mediante la cual el analista de sistemas recolecta información de las personas a través de la interacción cara a cara. Es una conversación con un propósito específico, que se basa en un formato de preguntas y respuestas en general. Conocer opiniones y sentimientos del entrevistado. Los tipos de información obtenida de una entrevista son: opiniones, objetivos, procedimientos informales y sentimientos. Existen dos tipos de entrevistas:

Estructuradas (Cerradas). La misma consiste en que el encuestador tiene un

Figura 59: Tipos de preguntas en cuestionario

conjunto específico de preguntas para hacérselas al entrevistado. Se dirige al usuario sobre un requerimiento puntual. La desventaja que posee es que no permite adquirir un amplio conocimiento del dominio.

No Estructuradas (Abiertas). La misma consiste en que el encuestador lleva a un tema en general sin ninguna preparación de preguntas específicas. Por lo general inician con preguntas que no dependen del contexto, para conocer el problema, la gente involucrada, entre otras ideas generales. Las ventajas son que el entrevistado se siente incluido en el proyecto y es posible obtener una retroalimentación del mismo. Además es posible adaptar las preguntas de acuerdo al entrevistado. Hay que tener en cuenta que este tipo de entrevistas son costosas,requieren tiempo, recursos humanos y dependen en gran parte de las habilidades del entrevistador.

En la figura 60 se observa las características de los tipos de preguntas que se realizan en una entrevista.

Tipos de preguntas de una entrevista

Existen 3 tipos de preguntas en una entrevista,por un lado tenemos las que se consideran abiertas las cuales permite al encuestado responder de cualquier manera. Por ejemplo,preguntas como las siguientes: ¿qué opinión tiene del sistema actual?,¿cómo describe su trabajo?. Por otro lado, están las cerradas las cuales sus respuestas son directas, cortas o de selección específica por ejemplo:¿quién recibe este informe?, ¿cuántas personas utilizan el sistema?. Finalmente, están las de sondeo que permiten obtener más detalle sobre un te-

ma puntual por ejemplo: ¿podría dar detalles sobre…?, ¿podría dar un ejemplo de…?.

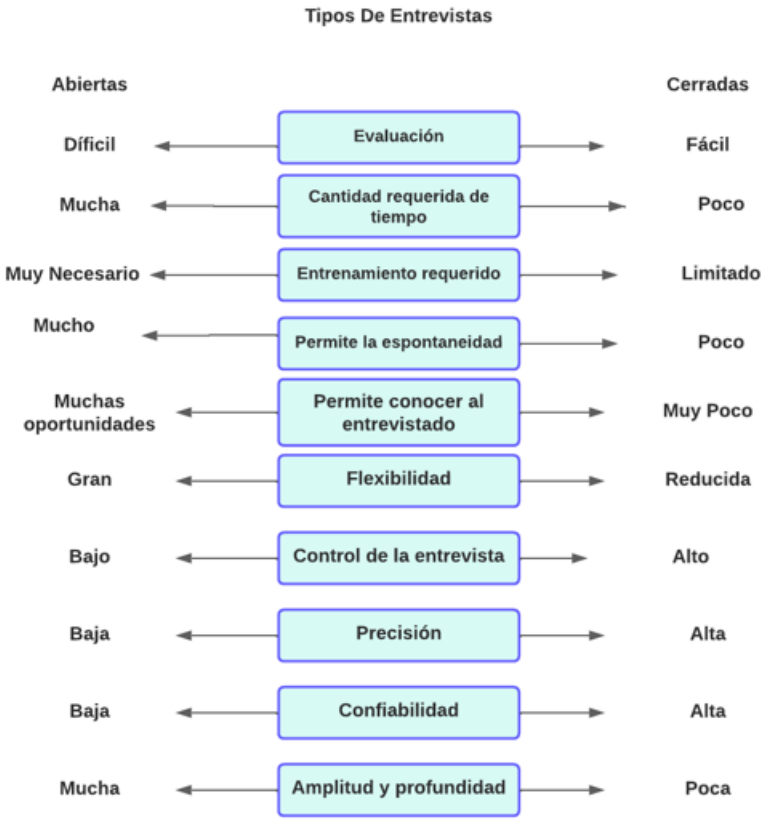

Figura 60: Tipos de entrevistas

Planeación conjunta de Requerimientos (JRP o JAD)

La planeación conjunta de requerimientos es el proceso mediante el cual se conducen reuniones de grupo altamente estructurado con el propósito de analizar problemas y definir requerimientos. Este proceso requiere de extenso entrenamiento, ya que reduce el tiempo de exploración de requisitos y amplía la participación de los integrantes. Entre las ventajas del jrp encontramos que ahorra tiempo,que hay varios usuarios involucrados y obtenemos desarrollos creativos. Con respecto a las desventajas frente a otras técnicas notamos que es difícil organizar los horarios de los involucrados y es complejo encontrar un grupo de participantes integrados y organizados. La forma de planear las sesiones de JRP es mediante la selección de una ubicación para llevar a cabo las mismas,como así también la selección de los participantes y la preparación de la agenda. La técnica de jrp nos ofrece como beneficios la involucración activa

de los usuarios y la gerencia en el proyecto de desarrollo. También se destaca una reducción del tiempo en la etapa de requerimientos y además de que si se incorporan prototipos, los mismos ya confirman el diseño del sistema.

Lluvia de Ideas - Brainstorming

La técnica de brainstorming a diferencia de jrp permite generar ideas al alentar a los participantes para que ofrezcan tantas ideas como sea posible en un corto tiempo sin ningún análisis hasta que se hayan agotado las ideas. Con esto se promueve el desarrollo de ideas creativas para obtener soluciones. También se realizan reuniones del equipo involucrado en la resolución del problema y estas son conducidas por un director. Entre las ventajas más destacables tenemos que es clave resolver la falta de consenso entre usuarios,es útil combinarlo con la toma de decisiones que nos ayuda a entender el dominio del problema,encarar la dificultad del usuario para transmitir y ayudar a entender. La idea del brainstorming se basa en una serie de principios que son en que cuantas más ideas se sugieren mejores resultados se conseguirán, esto quiere decir que la producción de ideas en grupos puede ser más efectiva que la individual. A su vez las ideas de una persona pueden hacer que aparezcan otras por **contagio** y otras veces puede suceder que las mejores ideas aparecen tarde y es mejor elegir sobre una variedad de soluciones. Dentro del brainstorming se organizan una serie de fases de aplicación entre las que se destacan el **calentamiento**, define que en la fase 1 se presenta el grupo y se realizan una serie de ejercicios para mejorar el funcionamiento colectivo y así preparar a los participantes. Es uno de los temas más importantes, ya que se sienta aquí la base para lograr el correcto proceso de elaboración de ideas. Después del calentamiento comienza la **generación de ideas**.

Para comenzar con esta etapa y asegurar que todos los integrantes parten de la misma base, se les debe plantear un objetivo al que se pretende llegar mediante la sesión creativa de ideas, pero antes de ponerse con dicho trabajo es necesario conocer una serie de reglas básicas: toda crítica está prohibida durante el proceso de generación; toda idea es bienvenida; se generarán tantas ideas como sea posible; el desarrollo y asociación de las ideas generadas es deseable.

Luego de esto continuamos con la siguiente fase de **trabajo con las ideas** la cual en esta se pretenden mejorar las ideas existentes y definirlas en mayor profundidad y así poder descartar aquellas que sean menos viables.

Finalmente, se completa con la etapa de **evaluación** tras haber definido en mayor profundidad las ideas y haber descartado mediante un primer filtrado las menos viables, el grupo debe establecer una serie de criterios con los cuales se van a evaluar las ideas.De este análisis y evaluación en conclusión se elige la más adecuada para solucionar el problema planteado al principio de la sesión y, por lo tanto, se da por terminada la sesión de brainstorming.

El concepto de stakeholder

Ya mencionado anteriormente, cuando hablamos de stakeholders se hace re-

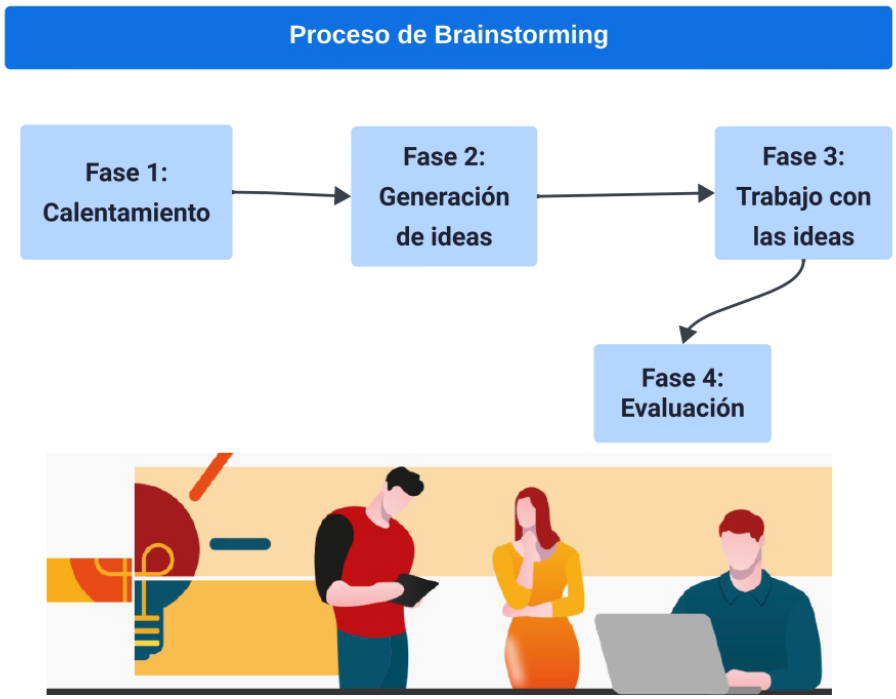

Figura 61: Ilustración de un proceso brainstorming

ferencia a las personas involucradas en el software. Los stakeholders desempeñan un papel clave en la elicitación, especificación y validación de los requerimientos de software. El desarrollo de un sistema de información requiere una serie de actividades y entre las más importantes se encuentra comprender las necesidades de parte de los stakeholders. Los requerimientos de software no solamente son un recurso para basar el desarrollo de un sistema, sino que brindan a los stakeholders la oportunidad de expresar qué es lo que quieren y exhiben estas necesidades; y ello en parte es el fundamento de que desempeñen en un papel relevante en la especificación y casi excluyente en la validación. En conclusión los requerimientos provienen de los stakeholders y más allá de la formulación de los requerimientos los stakeholders de un proyecto juegan un papel determinante en el éxito de estos.

La noción de stakeholder no ha sido tratada en forma consistente en la literatura en ingeniería de software. En un proyecto de desarrollo de software hay tres entidades básicas en interacción: los requerimientos del producto a construir, las actividades ejecutadas para la construcción y el producto que resulta del desarrollo. En consecuencia el interés de una entidad cualquiera puede estar en distintas componentes del proyecto a desarrollar [38].

Según Loucopoulos en primer lugar se trata de stakeholders de requerimientos y son aquellos que tienen un interés en cuanto a ganar o perder con el cambio en consideración [38].

Según Freeman [4] un stakeholder es aquel que puede afectar a, o es afectado por, los propósitos de la corporación. Es una concepción que extiende la responsabilidad empresaria más allá de la que debe a los accionistas (los shareholders). Este concepto se extendió rápidamente a otros ámbitos más allá de la teoría de la firma y también a la ingeniería de software donde hoy se encuentra plenamente instalado.

En resumen del análisis del concepto de stakeholders se deduce que este término se utiliza para referirse a cualquier persona o grupo que se verá afectado por el sistema, directa o indirectamente.

Gestión de stakeholders

La gestión de los stakeholders es el proceso de identificar y comunicarse efectivamente con aquellas personas o grupos que tienen interés en los resultados de los programas/proyectos permitiendo gestionar las relaciones con las partes interesadas como una forma de lograr influencia y resultados positivos de los programas y proyectos. Entre las características principales de la gestión de los stakeholders podemos encontrar que los stakeholders a distintos niveles, tanto dentro como fuera de la organización, deberán analizarse y comprometerse con eficacia para alcanzar los objetivos del programa en términos de apoyo y compromiso. La gestión de los stakeholders incluye la planificación de las comunicaciones,el uso e identificación efectivo de los diferentes canales de comunicación y las técnicas que permiten alcanzar los objetivos del programa. Con respecto a la interacción con otros,la comunicación con los stakeholders debe ser clara, consistente,enfocada en lo esencial y en un lenguaje comprensible para todos y en términos de proceso la gestión debe ser vista como un proceso continuo en todas las iniciativas del programa y vinculado al ciclo de vida de la iniciativa y los controles de la institución [3].

Punto de vista de los stakeholders

Existen tres tipos genéricos de puntos de vista que se detallan en el cuadro 21.

Existen dos tipos de stakeholders que se detallan en el cuadro 22, y algunos ejemplos de estos stakeholders se pueden apreciar en el cuadro 23.

Cuadro 21: Los 3 puntos de vista de los stakeholders

Punto de vista	Definición
Punto de vista de los interactuadores	Representan a las personas u otros sistemas que interactúan directamente con el sistema. Pueden influir en los requerimientos del sistema de algún modo
Punto de vista indirecto	Representan a los stakeholders que no utilizan el sistema ellos mismos pero que influyen en los requerimientos de algún modo
Punto de vista del dominio	Representan las características y restricciones del dominio que influyen en los requerimientos del sistema

Cuadro 22: Tipos de stakeholders [39]

Stakeholders	Definición
Stakeholders internos	Varios miembros del organismo gubernamental, incluyendo a la dirección ejecutiva y los representantes de ministerios, agencias y departamentos,como también a los empleados
Stakeholders externos	Miembros públicos, grupos de intereses especiales, la prensa, y otros niveles de gobierno. Tienen derecho a desafiar u objetar las decisiones tomadas por los gerentes de los programas y proyectos

Cuadro 23: Ejemplos de tipos de stakeholders externos e internos

Stakeholder	Concepto
Clientes	Paga por el producto
Consultores	Internos o externos,son los que tienen la potestad para ayudar a abarcar los requerimientos correctos
Consumidores	Compra el producto
Desarrolladores	
Estándares industriales	Código que deben satisfacerse para los productos que se consumen en una industria
Expertos	Incluye consultores de negocios,analistas de negocios y cualquiera que tenga conocimiento especializado de que puede contribuir al sistema
Expertos técnicos	Requeridos para asesorar en la construcción técnica del producto
Fuerza del mercado	Posiblemente representado por el depto de marketing
Gerentes	
Gobierno	Para el caso de los productos que generan información para el gobierno o la reciben de él
Grupos especiales	Extranjeros,tercera edad,jovenes,etc
Inspectores	De seguridad,auditores,eventualmente del gobierno,etc
Intereses culturales	Productos de dominio público que deben considerar no ofender a ninguna comunidad
Legislación	Representado por abogados o exigencias legales,se incluyen los estándares

Trabajo Colaborativo

El trabajo colaborativo es un proceso en el que un individuo aprende más de lo que aprendería por sí solo, fruto de la interacción de los integrantes de un equipo, quienes saben diferenciar y contrastar sus puntos de vista, de tal manera, que llegan a generar un proceso de construcción de conocimiento. En el trabajo colaborativo un gran número de individuos es coordinado y conducido hacia un objetivo común. Esta coordinación generalmente le corresponde a comunidades autoorganizadas en las que el trabajo se lleva a cabo sin obligaciones formales y sin que se reconozcan un autor exclusivo del resultado [42].

Diferencias con el trabajo en equipo

No debe confundirse el trabajo colaborativo con el trabajo en equipo, a pesar de que toda colaboración implique necesariamente unas labores conjuntas o de equipo.Mientras que el trabajo en equipo reconoce un líder definido, colectiviza la responsabilidad y organiza un grupo humano para cumplir una tarea puntual, el trabajo colaborativo suele llevarse a cabo sin presencia de jefes formales, a través de aportaciones individuales de conocimiento, y con el propósito final de compartir el conocimiento alcanzado.

Características del trabajo colaborativo

A grandes rasgos el trabajo colaborativo, se caracteriza porque; **se trata** de una forma de colaboración voluntaria entre los miembros de una comunidad especializada, capaz de compartir esfuerzo, tiempo y saberes en la construcción de nuevas plataformas de conocimiento. Además,**consiste** en la sinergia de los saberes y los esfuerzos de una comunidad organizada, generalmente de personas que comparten pasiones o saberes. **Suele estar** regido por los principios del libre flujo de la información, del altruismo y el voluntariado, por lo que a menudo se considera una cultura laboral en sí misma. **A diferencia** de otras formas de trabajo, no genera un sentimiento de grupo, pero sí uno de pertenencia a un proyecto colectivo.

Importancia del trabajo colaborativo

El trabajo colaborativo facilita la consecución de objetivos a largo plazo, al permitir que los continuos aportes individuales avancen libremente hacia la meta.

Sin embargo, existe un margen importante de coordinación: la dinámica colaborativa, al ser libre y voluntaria, recibe continuos aportes que incrementan el valor y el alcance del proyecto. De hecho, este tipo de trabajo no persigue resultados óptimos en calidad o productividad, sino que apunta a la integración del conocimiento compartido.

Ventajas del trabajo colaborativo

A diferencia de otras formas de trabajo, el trabajo colaborativo no requiere de una estructura formalmente definida, de modo que resulta menos rígido y coercitivo en sus formas: sin jefes, sin reparto oficial de las labores, este tipo de trabajo se sustenta en la continua actualización de los saberes compartidos. Se trata de una labor que depende más que nada del grupo.

Del mismo modo, se trata de un modelo de trabajo masivo y disperso, que no se rige por horarios y que, además, se sostiene de manera voluntaria, de modo que sus costes de producción son, al menos en ese sentido, muy bajos. Por otro lado, el proyecto no depende de ninguno de sus colaboradores por completo, sino que se trata de una forma de colectivización del saber, ajena a jefaturas y a liderazgos tradicionales.

Herramientas colaborativas

Las herramientas colaborativas son programas para trabajar en equipo que incluyen funciones para gestionar proyectos y tienen como objetivo mejorar el proceso de trabajo. Para ello, contienen funciones específicas para planificar, organizar y analizar tareas [29]. Algunas de las funcionalidades más habituales en estas herramientas colaborativas suelen ser los **mapas conceptuales**. Muchas de estas herramientas están pensadas para asistir a los equipos en el proceso creativo con la elaboración de mind maps y otras variantes de brainstorming, animando a los implicados en un determinado proyecto a compartir sus ideas y participar, así, activamente en su concepción. Además, se puede incluir al cliente en este proceso, si este así lo desea, y se le proporciona acceso a la herramienta. Otra herramienta muy común consiste en **compartir archivos.** Uno de los pilares en los que se basa una herramienta de colaboración es la posibilidad de compartir tablas, documentos y otros archivos con el resto de los miembros del equipo. Muchas herramientas permiten almacenar los datos en una plataforma central, así como definir permisos de acceso individuales, de modo que todos los archivos importantes están disponibles en cualquier momento y desde cualquier lugar. Los directores o administradores del proyecto son los encargados de establecer a qué datos tiene acceso cada miembro.

Muchas herramientas colaborativas contienen funciones que permiten la **comunicación en tiempo real**. Por ejemplo, en un panel de proyecto se pueden compartir las tareas o los avances del mismo, y a través de los comentarios, es posible aclarar dudas sobre cualquier cuestión de forma inmediata. Muchas aplicaciones siguen utilizando formas tradicionales de comunicación como la videollamada, la mensajería instantánea o el email, o disponen de interfaces a programas tan conocidos como Outlook. Además, la mayoría de los proveedores permiten guardar los documentos en la nube, de forma que un mayor número de usuarios puede tener acceso a ellos al mismo tiempo.

Se describe como una reunión, de una llamada importante o de una fecha de entrega, los correos, las llamadas o los mensajes de chat son un medio útil para informar eventos próximos, aunque de esta manera también se olvidan fácilmente. Cuantas más citas tengamos, más complicado resulta recordarlas todas. Por este motivo, las herramientas de colaboración también contienen **funciones de calendario** por medio de las cuales se pueden compartir todas las fechas relevantes y visualizarlas en cualquier momento.

Un buen software colaborativo también permite al administrador del proyecto o al jefe del equipo planificar los diferentes procesos de trabajo de la mejor forma posible sin perder la visión de conjunto. Por este motivo, también contiene componentes que permiten repartir y organizar los recursos de forma eficiente. Esto se conoce como **funciones de gestión**. Muchas de las herramientas ofrecen, asimismo, soluciones para poder documentar y valorar los avances en el proyecto. Si se utilizan en todos los departamentos, pueden facilitar la comunicación entre los mismos.

Ejemplos de herramientas colaborativas

Gmail es una de las aplicaciones de correo electrónico más populares (utilizada por más de 1 billón de personas), y por buenas razones está incluido en Google Drive, lo que hace que se integre con el software de Google. También tiene una interfaz intuitiva. Y, contiene muchas funciones que puedes utilizar, como: Cancelar el envío, reenviar, búsqueda poderosa y más.

En el año 2014 se creó la herramienta de groupware **slack**, simplificando desde ese momento la comunicación en empresas y agencias y encargándose de que todo usuario pueda acceder en cualquier momento a los servicios y recursos que necesita en su jornada de trabajo.

Skype es un software que permite comunicaciones de texto, voz y vídeo sobre Internet. Fue diseñado en 2003 por el danés Janus Friis y el sueco Niklas Zennström (también creadores de Kazaa) y desarrollada en su solución técnica por los estonios Priit Kasesa-lu, Ahti Heinla y Jaan Tallinn, ya que de hecho Skype nació en Tallin, Estonia. El código y protocolo de Skype permanecen cerrados y son privativos de la aplicación.

Trello es una aplicación de gestión de proyectos basada en un tablero que usamos para planear asignaciones de tareas. Puedes crear tantas Tarjetas y Columnas como necesites. Por ejemplo, puedes tener una Columna para cada sección Por Hacer,Haciendo y Terminado y luego asignar a cada tarea una Tarjeta. Cada Tarjeta se mueve a través del Tablero mientras el trabajo avanza. También es realmente fácil comunicar las tareas específicas de un proyecto utilizando el sistema de comentarios de Trello.

Google Docs tiene su punto más fuerte cuando las personas colaboran online. Como procesador de palabras es ágil, pero como una suite de herramientas para personas que trabajen juntas en documentos, es robusta. Múltiples personas pueden trabajar en el mismo documento al mismo tiempo, dejando cada una notas, realizando ediciones todo desde múltiples locaciones.

Los documentos no son lo único en lo que la gente colabora. La programación es una de las tareas de colaboración más importantes que realizan muchas compañías. Cientos de programadores pueden estar trabajando en el mismo proyecto a la vez. **GitHub** es una herramienta para asegurarte que todo se mantenga en orden. Cada persona tiene una copia del código en su computadora. Cuando realizan cambios ellos lo envían en github lo cual mantiene todo administrado. Si los cambios de una persona rompe las cosas, es fácil deshacer los cambios. Si tu equipo está colaborando en la programación, github es esencial.

11 **Anexo 2 Publicación relacionada**

A continuación se hará referencia al anexo A Collaborative Approach to specify Kernel Sentences using Natural Language, el cual describe ideologías de la iteración de requerimientos por medio de diferentes usuarios con diferentes propuestas las cuales se buscan que todas sean evaluadas. [17]

A Collaborative Approach to specify Kernel Sentences using Natural Language

Leandro Antonelli[1], Alejandro Fernandez[1 2], Nicolas Ruffolo[1],
Emiliano Sansone[1], Diego Torres[1 2 3]

[1]Lifia, Fac. de Informática, UNLP, La Plata, Bs As, Argentina
[2] Comisión de Investigaciones Cientificas (CICPBA)
[3] Departamento de Ciencia y Tecnología, UNQ

{leandro.antonelli, alejandro.fernandez, nicolas.ruffolo,
emiliano.sansone, diego.torres}@lifia.info.unlp.edu.ar

Abstract—Requirements engineering is a critical part of software development. Errors in the requirements, if not found and corrected early in the engineering process, become costly problems later on. Analysts commonly rely on Use Cases or Users Stories to capture requirements. However, there is domain knowledge that these artifacts don't capture well (for example, business rules and given-then-when scenarios). Such domain knowledge is generally distributed among multiple stakeholders and domain experts with complementing perspectives. Therefore, it is important to use a collaborative technique with a simple artifact to acquire and validate their knowledge. Kernel sentences is a linguistic definition about small sentences (with only one verb) written in active voice. Some authors relate kernel sentences to business rules. We argue that kernel sentences are adequate to use in the collaborative acquisition and they can be used as the input to produce more complex artifacts. This paper proposes a collaborative approach to acquire and validate kernel sentences. The process has three main activities: acquisition of the kernel sentences, validation of them, and assessment of the activity of the experts who participate in the activity. This paper also describes a prototype to support the process. Finally, the paper shows the result of a preliminary evaluation with promising results about the applicability of the process.

Keywords- requirements; specifications; kernel sentences; collaboration; natural language

1. Introduction

Requirements engineering is a critical stage of software development. Errors made at this stage can cost up to 200 times to repair when the software is delivered to the client [3]. There are two main philosophies of software development life cycle: classic and agile. Both strategies organize the development process and deal with the requirements in different ways. In a classic life cycle, an extensive documentation is produced and consumed. It consists for example of software requirement specification

with hundreds of Use Cases. In agile development, the communication of the requirements relies on a specific role within the team the, product owner, and on a simple documentation artifact, the User Story. User Stories have minimum information and constitute “an invitation to talk” [1].

Requirements described as Use Cases or as User Stories define the goals, the scope and the functionality of the software system. Nevertheless, software applications are “packed knowledge about the domain” [7]. This knowledge needs to be captured in a complementary artifact to Use Cases and User Stories, for example in business rules [17] or given-then-when scenarios [22].

While goals and requirements for the software application can be elicited from a small group of people (the client or the sponsor) the knowledge of the domain relies in a wider group of stakeholder, the domain experts, who generally have a different and complementary point of view of the domain. Thus, it is important to involve as many experts as possible to collaborative [14] acquire their knowledge.

Experts and development team belong to different worlds and use different languages [20]. The experts use the language of the domain while development team uses a computer science language. In order to cope with this communication gap it is important to use artifacts in natural language that are readable by both parties [14].

The concept of the kernel sentence was introduced in 1957 by linguist Z.S. Harris [12] and featured in the early work of linguist Noam Chomsky [8]. Kernel sentences are also known as basic sentences. They are declarative constructions, in active voice, always affirmative with only one verb. Boyd [4] suggests the use of Kernel Sentences to describe models in Software development.

This article proposes a collaborative process to specify Kernel Sentences. It consists of three main activities. The first activity consists in the specification of kernel sentences by the experts. The second activity consists in the validation of the kernel sentences specified in the first activity by the other experts. Finally, the third activity consists in assessing the behavior of the experts to determine how reliable their contributions are. This article also describes a prototype that can be used to support the proposed process. Finally, the paper presents some preliminary evaluation using the SUS survey [5] [6] that shows the applicability of the process.

We believe that this approach can be used in both philosophies of software development: agile and classic. The effort for performing the process is more related to classic development cycle and it can be integrated with early stages of requirements engineering, before defining the scope of the software system as well as the requirements. Nevertheless, we think that the proposed approach can also be used in agile because the prototype tool proposed can deal with the complexity and reduce the effort. Moreover, the knowledge consolidated is a good complement to User Stories.

The rest of the paper is organized in the following way. Section 2 describes some background about kernel sentences. Section 3 details our contribution namely, the proposed collaborative process. Section 4 describes the tool to support the process. Section 5 presents the preliminary evaluation. Section 6 reviews some related work. Finally, Section 7 discusses some conclusions.

2. Kernel Sentences

A kernel sentence is a simple construction with only one verb. It is also active, positive and declarative. This basic sentence does not contain any mood. It is termed as “kernel” since it is the basis upon which other more complex sentences are formed. For example, Figure 1 describes two kernel sentences. The first sentence states that the subject “farmer” performs an action (“fertilizes”) on a certain object (“tomatoes”). The second sentence has the same structure while it describes a different action (“water”) and it also adds the description about “when” the action is performed. It is important to mention that the verb “to be” does not have a semantic meaning. That is why the second example has two verbs: “to water” and “to be”. Figure 2 shows two sentences that are not kernel, since both sentences has two verbs. First sentence uses the verbs “to fertilize” and “to add”, while second sentence uses the verbs “to water” and “to prevent”. The first sentence can be rewritten into two kernel sentences (Figure 3). The farmer is the subject of the first verb, that is, “the farmer fertilizes…”. And the fertilization activity is the subject of the second action that is “the fertilization adds nutrient”. This example shows how the original sentence (Figure 2) with two verbs draws a conclusion about the role of the farmer who fertilizes and adds nutrient. Nevertheless the correct responsibilities are stated in Figure 3, that describes that the farmer fertilizes and because of this activity, the nutrients are added. This precision in the description is very important to understand the domain.

The farmer fertilizes the tomatoes
The farmer waters the tomatoes when it is hot

Figure 1. Kernel Sentences

The farmer fertilizes the tomatoes to add nutrients that are not present in the soil.
The farmer waters the tomatoes to prevent them of drying out.

Figure 2. No Kernel Sentences

The farmer fertilizes the tomatoes
The fertilization adds nutrient to the soil

Figure 3. Sentences rewritten as kernel sentences

3. The Proposed Approach

The proposed process has the objective of collaboratively obtaining kernel sentences from the domain experts in order to consolidate the knowledge of the application domain. The process is collaborative because many experts can participate at the same time. People contribute with different kernel sentences, since different stakeholders may have different point of view about the domain. And this complementary vision is very important to produce an integrated and complete description of the domain.

The process considers two different roles: experts of the domain and analysts. The experts of the domain provide the kernel sentences and validate the kernel sentences proposed by other experts. Thus, the collaborative characteristic is

reinforced. Analysts participate as moderators of the process. They have a complete vision of the set of kernel sentences provided by all the experts and the scope of the software system. Thus, analysts can identify kernel sentences that do not belong to the domain and remove them from the process. Analysts can also identify duplicated kernel sentences proposed by different experts. Finally, analysts monitor the activity of the expert by identifying experts with some particular biased behavior. For example, an expert who accepts as valid every kernel sentence, or that rejects as invalid all of them is not concerned with the activity and his contribution should be omitted since it is not reliable. The analyst not necessarily must be an expert of the domain. Moreover, we believe that he should not be an expert in order to avoid biases based on his subjectivity.

Thus, the process proposed considers three different activities: (i) kernel sentences specification, (ii) kernel sentences validation, and (iii) experts assessment. These three activities are conducted in parallel. Thus, while sentences are specified, other sentences can be validated and at the same time experts can be assessed.

The first activity (kernel sentences specification) relies on the definition of kernel sentences by some expert (who becomes its author). Then, some revision should be made to verify that the kernel sentence satisfies the conditions to be considered a kernel sentence (from the grammar perspective). Finally, some analyst should review the knowledge stated by the kernel sentence to determine if it is valuable or not for the description of the domain.

The second activity (kernel sentences validation) relies on collecting the opinion (agreement) of the experts (different from the author) about the kernel sentences of the first activity. Finally, the analyst decides whether to accept a kernel sentence as valid, based on the opinions of the experts.

The third activity (experts assessment) relies on monitoring the contributions of the experts, to identify how reliable some participant is. If a person is not reliable the analyst should exclude the person from the activity as well as his contributions. Figure 4 summarizes the whole process.

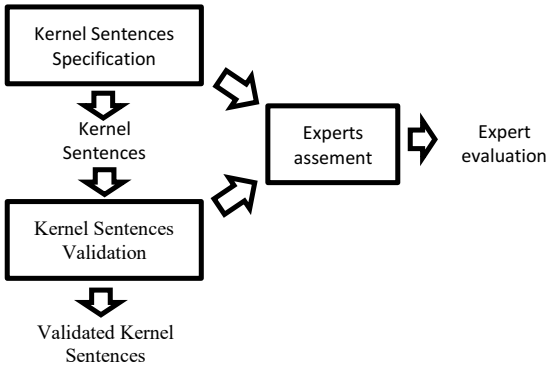

Figure 4. The proposed approach

3.1. Kernel Sentences Specification

The activity of kernel sentences specification is composed of three steps: (i) the description of the kernel sentence, (ii) the verification of its kernel sentence quality, and (iii) the revision of the kernel sentence, to confirm that it is significant for the scope of the software system.

The first step, the description of kernel sentence, is performed by some expert. The expert becomes the author of the sentence. It is important to trace every kernel sentence to its author because the other experts (different from the author) must indicate if they agree or not with the contribution. Moreover, identifying the author is also important to monitor his participation during the expert assessment activity.

The second step, the verification step, checks whether the kernel sentence is really a kernel sentence. We also propose some extra verification to make sure that the contribution of the expert is as simple and clear as possible. The following paragraphs describe the three verification steps we propose.

The first verification consists in checking that the sentence has the structure subject + verb + object to verify that it has only one verb, and that it is written in active voice. Figure 5 provides some examples of this verification.

> The tomatoes are fertilized by the farmer (passive voice, not correct)
> It is necessary to fertilize the tomatoes (null subject, not correct)
> The farmer fertilizes and waters the tomatoes (two verbs, not correct)
> The farmer fertilizes the tomatoes (correct)

Figure 5. Revision of structure subject + verb + object

The second verification consists in checking the presence of conjunctions. There are different types of conjuntions: (i) coordinating conjunctions such as ‘and’, ‘or’, ‘for’, ‘but’, etc., (ii) correlative conjunctions such as ‘not only’, ‘but also’, ‘either’, ‘neither’, etc, (iii) and subordinating conjunction such as ‘after’, ‘as long as’, ‘if only’, ‘where’, ‘according to’, etc. The presence of conjunctions does not determine that the contribution is not a kernel sentence. Nevertheless, it could provide a clue that too much information is contained in only one sentence. Figure 6 provides some examples of this revision. It is important to mention that the third sentence in Figure 6 (“The farmer assesses the humidity of the soil”) provides knowledge that is implicit in the second sentence of the same figure (“The farmer waters the tomatoes according to the humidity of the soil”). That is, “according to” means that the farmer should “assess the humidity”. Moreover, the farmer has some criteria to decide when humidity is enough and no watering is necessary.

> The farmer fertilizes and waters the tomatoes (conjunction “and” to express two verbs)
> The farmer waters the tomatoes according to the humidity of the soil (correct, but “according to” suggests the presence of more knowledge).
> The farmer assesses the humidity of the soil (correct, it is inferred from the previous one)

Figure 6. Revision of presence of conjuctions

The third verification consists in checking the presence of adjectives and adverbs. Although their presence does not confirm that the contribution is not a kernel sentences, these types of words characterize nouns and verbs, and their presence could provide a clue that more information could be added in another kernel sentence. Figure 7 provides some examples of this revision. The first sentence uses the word "carefully" and the farmer knows what "carefully" means. Nevertheless, it is important to state explicitly its meaning. Thus, the second sentence describes that "carefully" means "waters the soil of the tomatoes" (avoiding pouring directly on the plant).

> The farmer carefully waters the tomatoes (it is a kernel sentence, but it uses the adverb "carefully", that should be describe)
> The farmer waters the soil of the tomatoes ("carefully" means avoiding pouring the water directly to the tomatoes)

Figure 7. Revision of presence of adjectives and adverbs

The third (and last) step of the activity of the kernel sentence specification consist in analyzing whether the sentence is significant for the domain or not. This analysis is performed by the analyst and since he is not an expert, his criteria could not be accurate. Nevertheless, it is important to perform some preliminary analysis of the suitability of the sentence before involving the rest of the experts in the validation. Figure 8 provides an example of this revision. The analysts know that safety of the workers is priority, that is why "the farmer uses sun protection" is important. Nevertheless it is outside of the scope of the software system. Thus, this sentence should be rejected.

> The farmer uses sun protection (this is important for the farmer health, but it is out of the boundary of the software application to develop)

Figure 8. Revision of suitability for the domain

3.2. Kernel Sentences validation

The activity of kernel sentences validation has the goal of deciding whether to accept or not the kernel sentences specified in the first activity. Thus, the accepted ones will integrate the knowledge about the domain. This activity is composed of two steps: (i) experts give their opinion about the kernel sentences specified by another author, and (ii) the analyst takes a decision about the kernel sentence (accept it or not) regarding the opinions of the experts.

The first step, the opinion of the experts, consists in providing one of three possible alternatives: "accept", if the expert considers that the kernel sentence should be accepted (because the expert agree with the author), "reject", if the expert considers that the kernel sentence should be rejected (because the expert does not agree with the author), (iii) and "don't have opinion", if the expert does not have knowledge about the statement of the kernel sentence.

The second step, deciding about the kernel sentence, consists in analyzing the opinions and, if they are conclusive, make a decision: accept or reject. If no, the analyst can wait until more opinions are collected to decide. As a suggestion, more

than half of the experts should have provided their opinion, and more than half of the opinions should agree on accept or reject.

3.3. Experts assessment

The main reason for the experts assessment activity is to monitor the contribution of the experts to identify people that are not concerned with the activity and whose contributions are not reliable. For example (i) people to contribute with junk information (because it is not finally accepted), (ii) people with a systematic tendency to contradict (that is, providing always false information), (iii) people that do not think thoroughly and automatically accept or reject everything. In our experience, we have identified two extreme behaviors. One represented by experts that specified a lot of kernel sentences, but only few of them were accepted by their colleagues. Another behavior corresponded to people that gave their opinion about accepting most of the kernel sentences, but many of them were finally rejected by their colleagues.

It is important to identify these unreliable contributors and take some actions to avoid biasing the result of the activity. The simplest measure consists in excluding the person from the activity since he is not committed with it. Considering the amount of people excluded from the activity, it could be necessary to review the kernel sentences where they were involved, since they would have been biased by unreliable opinions.

Beyond this proposed process, the assessment of the experts can be used in later stages of the requirements engineering process. For example, the information collected about the behavior of the expert can identify experts that prefer writing or validating, or people that have a lot of knowledge and their contribution are mainly accepted. For example, the ratio regarding kernel sentences validated and kernel sentences written can identify if the expert is a “writer” or a “validator”. Then, the ratio considering the kernel sentences written and kernel sentences accepted suggests that the person is an expert among the other experts. Thus, this information can be used to plan furthers stages in the requirements engineering process. For example, to discuss some particular topic with one some specific person, or to provide some information to the “validator” to obtain his feedback.

4. Tool Support

A software prototype was implemented that can be used to support the application of the proposed approach. The prototype is a web application implemented following a service-oriented architecture. The services are implemented in PHP [19] using the Symphony framework [23]. Moreover, Python [21] is also used to communicate to the SpaCy library [25] used to deal with natural language processing.

The application is responsive, that means, the interface of the application adapts itself to the device used: a computer (desktop or laptop) or mobile device (phone or tablet). Thus, the application provides a variety of platforms to be used and experts will have a wide range of possibility to contribute with knowledge acquisition.

The prototype implements two roles of users, (i) experts and (ii) analysts. All participants can work asynchronously. The experts can add contributions (kernel sentences) and validate other contributions. Figure 9 shows the interface to provide the opinion about kernel sentences written by the other experts. The analysts can verify the contributions of experts, and finally accept or reject them regarding the opinion of the experts. Figure 10 shows the interface where the kernel sentences are shown with the opinion provided by the experts about them. The percentage of "accepts" and "rejects" are displayed as well as the number of opinions provided. The analysts can also monitor the activity of the experts. Figure 11 depicts the interface that shows the contribution of some expert. It shows the kernel sentences provided as well as the kernel sentences that he provides his opinion. The interface shows the opinion of the expert and the final decision. And there are some stats about this activity. Thus, the analyst can assess the activity of the experts and identifying unreliable contributors.

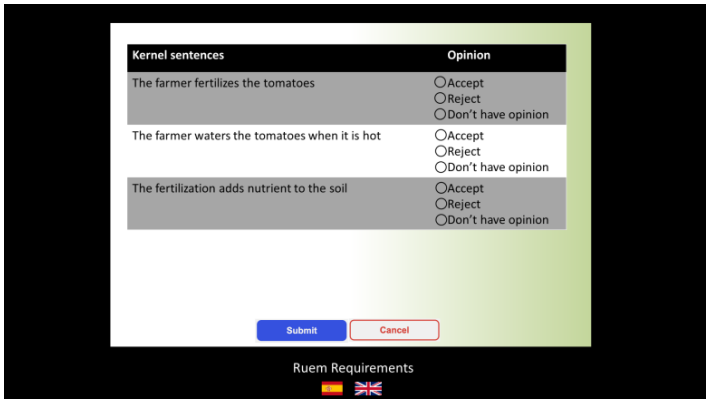

Figure 9. Interface of expert role

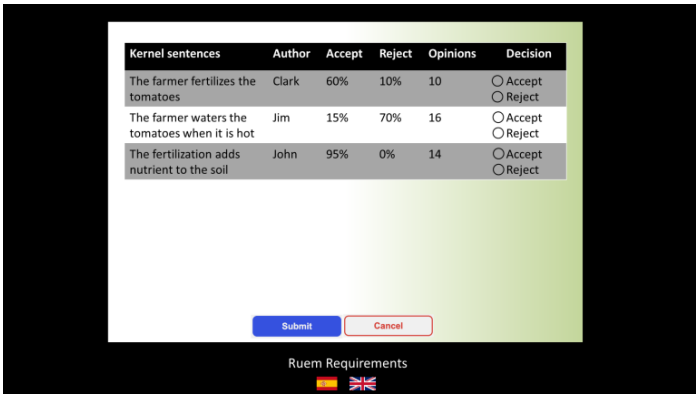

Figure 10. Interface of the analyst role

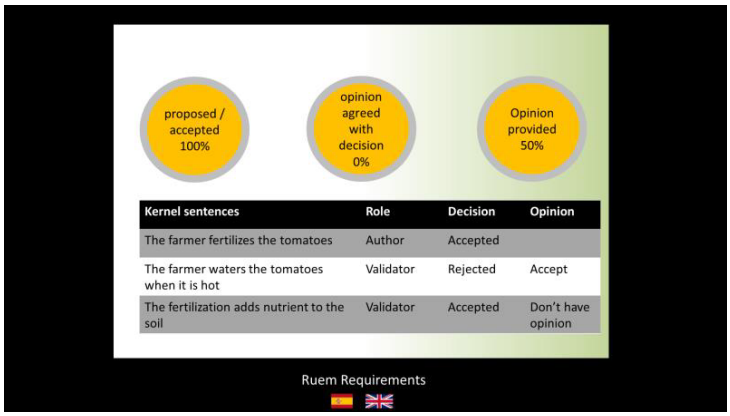

Figure 11. Interface of the experts assesment.

Thus, the complete workflow for a kernel sentence from its contribution to its finally inclusion in the data base of consolidate knowledge is the following. Some expert writes a contribution and becomes his author. The tool verifies whether the contribution is a kernel sentence or not (that is, verifies the rules described in the section of the approach).The analyst checks if the kernel sentence describes knowledge within the scope of the system. He can discard (reject) or consider that the kernel sentence is eligible (accept) to show the experts to express their opinion.

The application shows the kernel sentences accepted by analysts to the experts to ask their opinion about the correctness of the fact the kernel sentence state. Thus, the experts can answer "accept", "reject" or "don't have opinion". When the majority of experts provided their opinion, the analysts evaluate the percentages of "accept / reject / don't have opinion" to finally accept or discard the kernel sentence.

5. Evaluation

The collaborative process proposed was evaluated. The evaluation was performed using general collaborative tools like google spreadsheet instead of using the prototype tool presented in this paper, because the goal of the evaluation was to assess the applicability of the process instead of evaluating the usability of the tool. Moreover, only the main two activities were evaluated: specification and evaluation, because we trusted in the commitment of the participants, thus it was not necessary to assess the reliability of their contributions.

The participants of the evaluation were 14 students of a graduate course on requirements engineering. All of them have experience in industry, in software development. Nevertheless, the most important characteristic of them is the experience with the topic of the case study. They played the role of experts of the domain, and they had to specify and validate kernel sentences following the proposed approach. All the participants contributed to specify and validate kernel sentences for the same knowledge base. They were asked to contribute with a range between 10 and 20 kernel sentences, and they have one week to perform the task. In general, all the

kernel sentences were correctly defined, although there were many repeated contributions, because when experts contribute, they do not know the contributions of the others (unless they receive the contribuation to validate. One of the authors of this paper was the lecturer of the course and played the role of the analyst. He checked that the kernel sentences satisfied the conditions to be considered in the first activity (kernel sentences specification) and he also assigned the kernel sentences to the participants to obtain their opinion and finally accept or reject them.

The application domain used in the evaluation was the market-place application domain. All the participants had experience, as user with different roles (buyers and sellers) and some of them as developer of some application in the domain. In order to solve some ambiguities, it was defined one specific web site of the market place domain to follow their functionality.

The Systems Usability Scale (SUS) was used to evaluate the results of the case study [5] [6] in terms of the applicability of the proposed approach. Although SUS is mainly used to assess usability of software systems, it was probe to be effective to assess products and processes [2].

The System Usability Scale (SUS) consists of a 10-item questionnaire; every question must be answered in a five-options scale, ranging from ”1” (”Strongly Disagree”) to ”5” (”Strongly Agree”). Although there are 10 questions, they are related by pairs, asking the same question but in a complementary point of view in order to obtain a result of high confidence.

The calculation of the SUS score is performed in the following way. First, items 1, 3, 5, 7, and 9 are scored considering the value ranked minus 1. Then, items 2, 4, 6, 8 and 10, are scored considering 5 minus the value ranked. After that, every participant’s scores are summed up and then multiplied by 2.5 to obtain a new value ranging from 0 to 100. Finally, the average is calculated. The approach can have one of the following results: “Non acceptable” 0-64, “Acceptable” 65-84, and “Excellent” 85-100 [15]. The score obtained was 71,07. Thus, the approach can be considered as “acceptable”.

6. Related works

Garner et al. [9] state how important the human interaction is, and the associate sharing in problem solving activities as software development. Thus, our proposed approach relies on a collaborative construction and a validation of the knowledge.

Giraldo et al. [10] propose an approach to transform BPMN models to a model with more precision called CIAM. They emphasize in the importance of capturing the knowledge as early as possible, for example in early meeting as we propose.

Vijayan et al. [26] agree in the importance of eliciting domain knowledge and they propose a tool based on StakeRare [14]. Since they work in a very early phase, their method needs as input a definition of the scope of the system. Then, it builds a network of stakeholders based on recommendations. Finally, it elicits the knowledge from them. It is interesting the idea of building the network using a snowball rolling technique. Our proposed approach does not suggest how to involve the stakeholder. Nevertheless, Meng et al. [16] describes their finding in the identification of key user

for extracting knowledge in a crowdsourcing environment. They assess some characteristics that we agree: user knowledge value and willingness of knowledge exchange. Zhang et al. [28] performed a literature review about the characteristics of participants in order to perform the best selection of them. They agree with the characteristic defined by Meng et al. [16] and add more characteristics: interest, skills, expertise, willingness to achieve, and reputation. It is a very interesting contribution although it will increase the effort of applying our approach if we would like to add some of them.

Unkelos et al. [24] propose a gamified collaborative requirements engineering process. Particularly, they developed a tool to support their approach. They defined three roles that are related with our proposed roles: the creator (it is the expert in our approach), the reviewer (it is the expert in our approach since he review the knowledge) and the costumer (it is the analyst in our approach since he is going to consume the knowledge obtained). They also evaluate the participants but they do with a different objective from our evaluation. They evaluate participants in order to foster and reward their participants according to the philosophy of the gamified techniques. Kifetew et al. [13] also agree in the need for gamify collaborative requirements practices. In particular they propose a tool to prioritize requirements.

Nejad et al. [18] present a collaborative method for knowledge management in architectural design. Although the goal of their method is different from our goal, the two proposals agree in collecting, verifying and validating knowledge in collaborative way. They use a "trust" computation for validation purpose. Gonçalves et al. [11] also agree with the overall method and they also consider the importance of business rules to consolidate the knowledge of the domain. Although we use kernel sentences, some authors state their similarities [4]. Wen et al. [27] make an interesting proposal of a platform for eliciting requirements defined through 4 attributes: stakeholder, context, functional and non-functional requirements. It is similar to our approach the linking between the stakeholder and the functional requirements. And it is interesting the definition of the context. We believe that it can help to improve the description.

7. Conclusions and future work

This paper proposes a process to consolidate the knowledge of the application domain by capturing, in a collaborative way, kernel sentences directly from the experts. The proposed process consists of three activities: (i) kernel sentences specification, (ii) kernel sentences validation, and (iii) experts assessments. Two roles participate in this process, (i) experts who describe and validate the kernel sentences, and (ii) analysts who perform additional validations and take the final decision on the kernel sentences.

Although the process proposed has the objective of consolidating the knowledge of the domain, the process and the kernel sentences can be considered as a first step in a bigger strategy to manage requirements. Kernel sentences can be used in more complex artifacts of knowledge specification as well as requirements. Moreover, the assessment of the participants is useful to draw a profile of the people involved in order to plan further stages in the requirements engineering process.

The results of the case study are promising, although there is too much work to do. A full implementation of the tool with a complete assessment of both, the tool and the process, is necessary. Moreover, more experiments about range of values and percentages should be done in order to obtain precise definition of the values to accept or reject kernel sentences and define a profile of the participants. We also consider that it is very interesting to include some kind of glossary or ontology to the approach in order to deal with ambiguities for providing a more precise definition.

Acknowledgment

This paper is partially supported by funding provided by the STIC AmSud program, Project 22STIC-01.

References

1. Alexander, I. and Maiden, N.: Scenarios, Stories, Use Cases, through the system development life cycle, West Sussex: John Wiley & Sons, 2004.
2. Bangor, A., Kortum, P. T., Miller, J. T.: "An empirical evaluation of the system usability scale." Intl. Journal of Human–Computer Interaction 24.6, pp. 574-594, 2008.
3. Boehm, B.W.: Software Engineering, Computer society Press, IEEE, 1997.
4. Boyd, N. S.: “Using Natural Language in Software Development.” In: Journal of Object-Oriented Programming - Report on Object Analysis and Design, 11-9, 1999
5. Brooke, J.: “SUS-A quick and dirty usability scale” Usability evaluation in industry, 189(194), pp. 4-7, 1996.
6. Brooke, J: "SUS: a retrospective", Journal of usability studies 8.2, pp.29-40, 2013.
7. Brooks, F., The Mythical Man-Month: Essays on Software Engineering, Addison-Wesley Professional, 2 edition 1995.
8. Chomsky, N.: The Logical Structure of Linguistic Theory. Plenum Press, New York, 1975.
9. Garner, B. J.: "Collaborative knowledge management requirements for experiential learning (CKM)," Proceedings IEEE International Conference on Advanced Learning Technologies, doi: 10.1109/ICALT.2001.943989, pp. 488-489, 2001.
10. Giraldo, F., Alzate, A., Duarte, L., Tobón, M. and Hoyos, B.: "Deriving collaborative models from business process models," 2011 6th Colombian Computing Congress (CCC), doi: 10.1109/COLOMCC.2011.5936278, pp. 1-5, 2011.
11. Gonçalves, J. C. de A.R., Santoro, F. M. and Baião, F. A.: "Collaborative narratives for business rule elicitation," 2011 IEEE International Conference on Systems, Man, and Cybernetics, doi: 10.1109/ICSMC.2011.6083954, pp. 1926-1931, 2011.
12. Harris, Z. S.: Co-occurrence and transformation in linguistic structure. (Linguistic Society of America) pp. 390- 457, 1957.
13. Kifetew, F., Munante, D., Perini, A., Susi, A., Siena, A. and Busetta, P.: "DMGame: A Gamified Collaborative Requirements Prioritisation Tool," 2017 IEEE 25th International Requirements Engineering Conference (RE), doi: 10.1109/RE.2017.46, pp. 468-469, 2017.
14. Lim, S. L., Finkelstein, A.: “StakeRare: Using Social Networks and Collaborative Filtering for Large-Scale Requirements Elicitation”, IEEE transactions on software

engineering, Volume 38, Issue 3, May-Jun 2012, DOI 10.1109/TSE.2011.36, pp 707-735, 2012
15. McLellan, S., Muddimer, A., Peres, S. C.: "The effect of experience on System Usability Scale ratings." Journal of usability studies 7.2, pp. 56-67, 2012.
16. Meng, Q,. and Guo, X.: "Identification of key user knowledge source in crowdsourcing innovation mode," 2015 12th International Conference on Service Systems and Service Management (ICSSSM), doi: 10.1109/ICSSSM.2015.7170263, pp. 1-6, 2015.
17. Meservy, T. O., Zhang, C., Lee, E. T. and Dhaliwal, J.: "The Business Rules Approach and Its Effect on Software Testing," in IEEE Software, vol. 29, no. 4, doi: 10.1109/MS.2011.120, pp. 60-66, July-Aug, 2012.
18. Nejad, M. S., Moaven, S., Habibi, J. and Alidousti, R.: "Toward a collaborative method for knowledge management of software architectural decisions based on trust," 2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), doi: 10.1109/FSKD.2015.7382050, pp. 828-834, 2015.
19. PHP, https://www.php.net/, accessed: 2022-03-05
20. Potts, C.: "Using schematic scenarios to understand user needs," in Proceedings of the 1st conference on Designing interactive systems: processes, practices, methods, & techniques, 1995
21. Python, https://www.python.org/, accessed: 2022-03-05
22. Rose, S., Nagy, G.: Formulation: Document examples with Given/When/Then, Independently published, 979-8723395015, 2021.
23. Symfony, https://symfony.com/, accessed: 2022-03-05
24. Unkelos-Shpigel, N. and Hadar, I.: "Inviting everyone to play: Gamifying collaborative requirements engineering," 2015 IEEE Fifth International Workshop on Empirical Requirements Engineering (EmpiRE), doi: 10.1109/EmpiRE.2015.7431301, pp. 13-16, 2015.
25. Vasiliev, Y.: Natural Language Processing with Python and SpaCy: A Practical Introduction. No Starch Press, 2020.
26. Vijayan, J., Raju, G. and Joseph, M.: "Collaborative requirements elicitation using elicitation tool for small projects," 2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES), doi: 10.1109/SCOPES.2016.7955848, pp. 340-344, 2016.
27. Wen, B., Luo, Z. and Liang, P.: "Distributed and Collaborative Requirements Elicitation Based on Social Intelligence," 2012 Ninth Web Information Systems and Applications Conference, doi: 10.1109/WISA.2012.14, pp. 127-130, 2012.
28. Zhang, X., Gong, B., Ni, H., Liang, Z. and Su, J.: "Identifying Participants' Characteristics Influencing Participant Estimation in Knowledge-Intensive Crowdsourcing," 2019 8th International Conference on Industrial Technology and Management (ICITM), doi: 10.1109/ICITM.2019.8710681, pp. 358-363, 2019.