FACULTAD DE INFORMÁTICA
UNIVERSIDAD NACIONAL DE LA PLATA

TESINA DE LICENCIATURA

Título: Aplicación web para la construcción colaborativa del Léxico Extendido del Lenguaje.
Autores: Sarmiento, Matias Sebastian y Zurbano, Mariela Fernanda
Director: Antonelli Leandro
Codirector: Sergio Firmenich
Asesor profesional: -
Carrera: Licenciatura en Sistemas

Resumen

En la presente tesina se investigó el Léxico Extendido del Lenguaje, haciendo hincapié en la construcción del mismo en forma colaborativa.
Se desarrolló una aplicación web que permite la creación de distintos LEL según se necesite. Construir un LEL se trata, básicamente, de crear símbolos de distintas categorías. Los símbolos son definidos a través de dos atributos: noció e impacto. Las nociones e impactos están compuestos por expresiones que pueden hacer referencia a símbolos ya definidos. En estos casos, existe una vinculación entre los mismos, facilitando de esta manera la "navegación" a través de los distintos símbolos del LEL que estén conectados. Los usuarios que trabajan sobre un mismo LEL pueden dar su opinión indicando like o dislike de un símbolo, y también agregar comentarios. Para los casos en los que se desee hacer algún arreglo o mejora en la descripción del LEL, se podrán realizar tareas de refactoring como por ejemplo unificar dos símbolos (merge). Por último, la aplicación cuenta con la posibilidad de integrar plug-ins desarrollados externamente a este proyecto.

Palabras Claves

Léxico Extendido del Lenguaje - Trabajo colaborativo - Símbolos - Usuarios

Conclusiones

A partir de la experiencia de desarrollar la aplicación web y de los objetivos planteados inicialmente, se concluyó que la utilización de una herramienta específica con funcionalidad pensada para tareas sociales es mucho más adecuado que utilizar un editor de texto plano. La aplicación desarrollada presenta claras ventajas para el trabajo colaborativo, proponiendo una forma clara, sencilla y ágil de colaborar y participar de manera activa en los proyectos de software.

Trabajos Realizados

- Investigación del Léxico Extendido del Lenguaje.
- Investigación y análisis de los trabajos realizados y herramientas colaborativas existentes para la construcción del Léxico Extendido del Lenguaje.
- Desarrollo de una aplicación web para la construcción de LEL en forma colaborativa, permitiendo futuras mejoras y/o extensiones.

Trabajos Futuros

Se proponen mejoras en algunos aspectos de la aplicación para que brinde el máximo beneficio y se definen algunas funcionalidades que podrían incorporarse tales como: procesamiento de lenguaje natural y soporte para análisis semántico.

Fecha de la presentación: MARZO 2019

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPITULO 1 - OBJETIVOS, MOTIVACIÓN Y DESARROLLO PROPUESTO

1.1 Introducción

En este capítulo se dará una breve presentación sobre los temas que se abordarán en esta tesina, comenzando por los objetivos, contexto y la motivación que originó el desarrollo de la misma y concluyendo con una breve descripción sobre el desarrollo propuesto.

1.2 Objetivos

Realizar el desarrollo de una aplicación web, que permita capturar el lenguaje del dominio del problema a través de la construcción de un glosario, conocido como Léxico Extendido del Lenguaje (LEL) [1], una representación hipertextual que contempla la descripción de la denotación y la connotación de cada término o entrada en el léxico.
El objetivo de la herramienta es lograr un espacio de construcción, mantenimiento y salida de datos del glosario que presente claras ventajas ante un procesador de textos tradicional. Además, ofrecer la posibilidad de trabajar en forma colaborativa [2] y respetando buenas prácticas [3] optimizando el tiempo invertido y agilizando la comunicación del grupo de trabajo.

1.3 Contexto

La ingeniería de software es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software [4], y el estudio de estos enfoques, es decir, el estudio de las aplicaciones de la ingeniería de software [5].
Existen distintas definiciones formuladas por prestigiosos autores, se citan las más reconocidas:

- Ingeniería de software es el estudio de los principios y metodologías para el desarrollo y mantenimiento de sistemas de software (Zelkovitz, 1978).
- Ingeniería de software es la aplicación práctica del conocimiento científico al diseño y construcción de programas de computadora y a la documentación asociada requerida para desarrollar, operar y mantenerlos. Se conoce también como el desarrollo de software o producción de software (Bohem, 1976).
- La ingeniería de software trata del establecimiento de los principios y métodos de la ingeniería a fin de obtener software de modo rentable, que sea fiable y trabaje en máquinas reales (Bauer, 1972).

- La ingeniería de software es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento del software [4]

El término Ingeniería de software apareció por primera vez a finales de la década de 1950 y fue estimulada por la crisis del software entre las décadas de 1960 y 1980 que identifica muchos de los problemas de desarrollo de software en aquellos años: muchos proyectos de software sobrepasaron el presupuesto y tiempo estimados, causaron daños a la propiedad e incluso causaron pérdidas de vidas [6].
El costo de propiedad y mantenimiento del software en la década de 1980 fue dos veces más caro que el propio desarrollo del software. Durante la década de 1990, el costo de propiedad y mantenimiento aumentó un 30% respecto a la década anterior. En 1995, las estadísticas mostraron que la mitad de los proyectos de desarrollo encuestados estaban operacionales, pero no eran considerados exitosos. Las tres cuartas partes de todos los grandes productos de software eran entregados al cliente con fallas tal que no fueron usados en absoluto, o no cumplieron con los requerimientos del cliente.
Cada nueva tecnología y práctica entre la década de 1970 y 1990 fue pregonada como la indicada para resolver la crisis del software. Así surgen herramientas como programación estructurada, programación orientada a objetos, herramientas CASE, el lenguaje de programación Ada, documentación y estándares. Por otra parte muchos abogaron el uso de procesos definidos y metodologías como el Modelo de Capacidad y Madurez. También se comenzó a trabajar en un código de ética, licencias y profesionalismo.
La búsqueda de una única clave para el éxito nunca funcionó. Todas las prácticas y tecnologías conocidas sólo han hecho mejoras incrementales en productividad y calidad.
El auge del uso de Internet llevó a un vertiginoso crecimiento en la demanda de sistemas internacionales de despliegue de información en la World Wide Web. Los desarrolladores se vieron en la tarea de manejar ilustraciones, mapas, fotografías y animaciones, a un ritmo nunca antes visto, con casi ningún método para optimizar la visualización y almacenamiento de imágenes. El uso de navegadores y utilización de lenguaje HTML cambia drásticamente la visión y recepción de la información.
Surge la necesidad entonces de crear soluciones de software a bajo costo, lo cual conlleva al uso de metodologías más simples y rápidas que desarrollan software funcional.

La Ingeniería de Software requiere llevar a cabo numerosas tareas agrupadas en etapas, al conjunto de estas etapas se las denomina *ciclo de vida*. El ciclo de vida del desarrollo de software es una estructura aplicada al desarrollo de un producto de software. Hay varios modelos a seguir para el establecimiento de un proceso para el desarrollo de software, cada uno de los cuales describe

un enfoque diferente para diferentes actividades que tienen lugar durante el proceso. El estándar internacional que regula el método de selección, implementación y monitoreo del ciclo de vida del software es ISO 12207 [7].

Las etapas comunes a casi todos los modelos de ciclo de vida son las siguientes:

Obtención de los requisitos

Se debe identificar el tema principal que motiva el inicio del estudio y creación del nuevo software o modificación de uno ya existente. A su vez identificar los recursos que se tienen. Es importante entender el contexto del negocio para identificar adecuadamente los requisitos.

Se tiene que tener dominio de la información de un problema, lo cual incluye los datos fuera del software (usuarios finales, otros sistemas o dispositivos externos), los datos que salen del sistema (interfaz de usuario, interfaces de red, reportes, gráficas y otros medios) y los almacenamientos de datos que recaban y organizan objetos persistentes de datos (por ejemplo, aquellos que se conservan de manera permanente).

También hay que ver los puntos críticos, lo que significa tener de una manera clara los aspectos que entorpecen y limitan el buen funcionamiento de los procedimientos actuales, los problemas más comunes y relevantes que se presentan, los motivos que crean insatisfacción y aquellos que deben ser cubiertos a plenitud. Por ejemplo: ¿El contenido de los reportes generados, satisface realmente las necesidades del usuario? ¿Los tiempos de respuesta ofrecidos, son oportunos?, etcétera.

Además, se tiene que tener en cuenta cómo será el comportamiento del software ante situaciones inesperadas como lo son por ejemplo una gran cantidad de usuarios usando el software o una gran cantidad de datos, entre otros.

La obtención de requisitos y el análisis se enfocan sólo en la visión del sistema que tiene el usuario.

Análisis de requisitos

La ingeniería de requisitos del software es un proceso de descubrimiento, refinamiento, modelado y especificación. Se refinan en detalle los requisitos del sistema y el papel asignado al software.

Tanto el desarrollador como el cliente tienen un papel activo en la ingeniería de requisitos (un conjunto de actividades que son denominadas análisis). El cliente intenta replantear un sistema confuso, a nivel de descripción de datos, funciones y comportamiento, en detalles concretos. El desarrollador actúa como interrogador, como consultor, como persona que resuelve problemas y como negociador.

El análisis y la especificación de requisitos pueden parecer una tarea relativamente sencilla, pero las apariencias engañan. El contenido de

comunicación es muy denso. Abundan las ocasiones para malas interpretaciones o falta de información. Es muy probable que haya ambigüedad. El dilema al que se enfrenta el ingeniero de software puede entenderse muy bien repitiendo la famosa frase de un cliente anónimo: “Sé que cree que entendió lo que piensa que dije, pero no estoy seguro de que se dé cuenta de que lo que escuchó no es lo que yo quise decir”.
El análisis de requisitos es una tarea de ingeniería del software que cubre el hueco entre la definición del software a nivel sistema y el diseño de software. El análisis de requerimientos permite al ingeniero de sistemas especificar las características operacionales del software (función, datos y rendimientos), indica la interfaz del software con otros elementos del sistema y establece las restricciones que debe cumplir el software [8].

Especificación

El resultado del análisis de requisitos con el cliente define un sistema que resuelve el problema. A tal definición se le llama especificación de los requisitos del software (SRS) y sirve como contrato entre el cliente y los desarrolladores. La SRS se estructura y formaliza durante el análisis para producir un modelo de análisis. Tanto la SRS como el modelo de análisis representan la misma información. Difieren sólo en el lenguaje y notación que usan. La SRS está escrita en lenguaje natural, mientras que el modelo de análisis se expresa, por lo general, en una notación formal o semiformal. La especificación del sistema es la base de la comunicación con los stakeholders. El modelo de análisis es la base de la comunicación entre los desarrolladores [14].
La obtención de requisitos y el análisis se enfocan sólo en la visión del sistema que tiene el usuario. La IEEE Std. 830-1998 normaliza la creación de las especificaciones de requisitos de software (Software Requirements Specification) [9].
Entre las técnicas más utilizadas para la especificación de requisitos se encuentran:

- Casos de uso

En 1986, Ivar Jacobson, importante contribuyente al desarrollo de los modelos de UML y proceso unificado, creó el concepto de caso de uso [10]. Un caso de uso expresa todas las formas de usar un sistema para alcanzar una meta particular para un usuario. En conjunto, los casos de uso le proporcionan todos los caminos útiles de usar el sistema e ilustran el valor que este provee. [11]. Se han realizado muchas mejoras al concepto que se estableció entonces, pero probablemente la más influyente y significativa, en términos de definición del término caso de uso, fue la de Alistair Cockburn en el libro Escribir casos de uso efectivos publicado en el año 2000 [12].

- Historias de usuario

Una historia de usuario es una representación de un requisito escrito en una o dos frases utilizando el lenguaje común del usuario. Las historias de usuario

son utilizadas en las metodologías de desarrollo ágiles para la especificación de requisitos.
Si bien el estilo puede ser libre, la historia de usuario debe responder a tres preguntas: ¿Quién se beneficia?, ¿qué se quiere? y ¿cuál es el beneficio? Por ello, algunos autores recomiendan redactar las historias de usuario según el formato: Como (rol) quiero (algo) para poder (beneficio) [13].

Desarrollo del software

- Implementación

Una implementación es la realización de una especificación técnica o algoritmos con un programa, componente software, u otro sistema de cómputo. El modelo de implementación es una colección de componentes y los subsistemas que contienen. Componentes tales como: ficheros ejecutables, ficheros de código fuente y todo otro tipo de ficheros que sean necesarios para la implementación y despliegue del sistema.
La etapa de implementación del diseño de software es el proceso de convertir una especificación del sistema en un sistema ejecutable. Siempre implica los procesos de diseño y programación de software, pero, si se utiliza un enfoque evolutivo de desarrollo, también puede implicar un refinamiento de la especificación del software. Esta etapa es una descripción de la estructura del software que se va a implementar, los datos que son parte del sistema, las interfaces entre los componentes del sistema, y algunas veces los algoritmos utilizados [14].

- Pruebas de software

Consiste en comprobar que el software realice correctamente las tareas indicadas en la especificación del problema. Se pueden realizar pruebas unitarias, que consisten en probar por separado cada módulo y luego realizar pruebas de manera integral para así llegar al objetivo, estas últimas se conocen como pruebas de integración.
Se considera una buena práctica que las pruebas sean efectuadas por alguien distinto al programador que desarrolló la solución, sin que esto exima al mismo de realizar sus propias pruebas. En general hay dos grandes maneras de organizar un área de pruebas, la primera es que esté compuesta por personal inexperto y que desconozca el dominio, de esta manera se evalúa que la documentación entregada sea de calidad, que los procesos descritos sean claros tal que cualquiera puede entenderlos y que el software hace las cosas como están descritas. El segundo enfoque es tener un área de pruebas conformada por programadores con experiencia, personas que saben sin mayores indicaciones en qué condiciones puede fallar una aplicación y que pueden poner atención en detalles que personal inexperto no consideraría.
De acuerdo con Roger S. Pressman, el proceso de pruebas se centra en los procesos lógicos internos del software, asegurando que todas las sentencias se han comprobado, y en los procesos externos funcionales, es decir, la realización de pruebas para la detección de errores.

- Documentación

Es todo lo concerniente a la documentación del propio desarrollo del software y de la gestión del proyecto, pasando por modelaciones (UML), diagramas de casos de uso, pruebas, manuales de usuario, manuales técnicos, etcétera; todo con el propósito de eventuales correcciones, usabilidad, mantenimiento futuro y ampliaciones al sistema.

- Mantenimiento

Fase dedicada a mantener y mejorar el software para corregir errores descubiertos e incorporar nuevos requisitos. Esto puede llevar más tiempo incluso que el desarrollo del software inicial. Alrededor de 2/3 del tiempo de ciclo de vida de un proyecto está dedicado a su mantenimiento. Una pequeña parte de este trabajo consiste en eliminar errores (bugs); siendo que la mayor parte reside en extender el sistema para incorporar nuevas funcionalidades y hacer frente a su evolución [8].

La ingeniería de software ha estado prestando mucha atención a la modelización de
productos, pero la Ingeniería de Requisitos comenzó examinando los aspectos de la generación de requerimientos y su intrínseca dependencia de los aspectos sociales que rodean la construcción y que rodearán a la operación del software [15].

El objetivo de la Ingeniería de Requisitos (IR) es sistematizar el proceso de definición de requisitos permitiendo elicitar, modelar y analizar el problema, generando un compromiso entre los Ingenieros de Requisitos (IRq) y los clientes/usuarios, ya que ambos participan en la generación y definición de los requisitos del sistema. La IR aporta un conjunto de métodos, técnicas y herramientas que asisten a los IRq para obtener requisitos lo más seguros, completos y oportunos posibles.

Diversos autores han propuesto el uso de escenarios para lograr estos objetivos, con resultados parciales muy aceptables. Los escenarios son descripciones de situaciones particulares del macrosistema. Están escritos en lenguaje natural y con una estructura simple. Estas características permiten la intervención directa del cliente/usuario tanto en su descripción como en su validación.

Diversos autores proponen el uso de glosarios como parte del proceso de construcción de escenarios, aunque en la mayoría de los casos su uso está restringido a un rol secundario de referencia o consulta. El enfoque propuesto por Rolland y Ben Achour (1998b) incluye el uso de un glosario de términos relacionado con la familia de casos de uso. Como bien sugieren, el empleo de lenguaje natural en casos de uso y escenarios puede implicar ambigüedades e inconsistencias. Estas desventajas persisten si sólo se maneja una lista de términos, pero pueden ser efectivamente reducidas mediante un vocabulario bien definido del Universo de Discurso (UdeD). Oberg et al. (1998) proponen el uso de un glosario con términos comunes a todos los participantes para facilitar

la comunicación y la comprensión. Este glosario se va refinando en sucesivas etapas del desarrollo y se lo utiliza para describir características del sistema y del modelo de caso de uso.
El Léxico Extendido del Lenguaje (LEL) es en sí mismo un glosario con roles y estructura más amplia que lo habitual. El principal objetivo del LEL es conocer el vocabulario del problema sin preocuparse por el problema [16]. De esta manera constituye el primer paso en el proceso de construcción del producto de software. Está escrito en lenguaje natural y compuesto por símbolos que son palabras o frases repetidas o relevantes del UdeD. Cuenta con vínculos hipertextuales (Leite y Franco 1990) entre símbolos [17].

El proceso tradicional para construir un LEL consiste en dos actividades realizadas por los IRq: identificar y describir los símbolos. El enfoque colaborativo propone además de estas dos actividades, una actividad social: expresar “like” a una expresión que define un símbolo. Es importante mencionar que en el enfoque colaborativo las actividades son llevadas al cabo directamente por stakeholders y no por IRq. Así, la identificación y la descripción de los símbolos también ocurre en forma colaborativa, y distintas personas cooperan para definir un símbolo. Por ejemplo, una persona identifica un símbolo y otra agrega una expresión. Luego, otra persona diferente incluye una expresión más para describir el símbolo. Finalmente, alguien más revisa el símbolo y su definición y puede indicar que le gusta alguna expresión [2].
Actualmente un LEL se construye mediante editores de texto tradicionales, lo cual no representa una facilidad para las actividades de colaboración entre los stakeholders.

1.4 Motivación

Es muy importante iniciar el desarrollo de software con los requerimientos necesarios lo más completos y correctos que sea posible. Es esencial comprender la naturaleza del problema que queremos resolver.
Comprender el contexto de un sistema de software durante la especificación de requerimientos es una tarea difícil. Es dificultoso elicitar y escribir el conjunto de requerimientos iniciales y a la vez cumplir con atributos tales como la correctitud y la completitud. Las fuentes principales de error durante la definición y análisis son: (i) falta de procedimientos y guías formales, (ii) falta de participación del usuario y (iii) falta de comunicación [14]. Es complejo realizar la especificación de requerimientos cuando hay muchos stakeholders involucrados, ya que les resulta difícil percibir sus propios errores o desvíos [18].
Todo aquello que no se detecte, o resulte mal entendido en la etapa inicial provocará un gran impacto negativo en los requisitos, propagando esta corriente negativa a lo largo de todo el proceso de desarrollo e incrementando su perjuicio cuanto más tardía sea su detección [19]. Esto es importante porque cualquier cambio en los requerimientos a futuro implica modificar diseño, desarrollo realizado, líneas de código, etcétera, que en algunos casos, llevan a

que productos ya cerrados tengan que modificarse e incluso (si estas modificaciones son muchas y/o significativas), prácticamente se descarten y se empiecen de nuevo.
Definir el lenguaje de dominio antes de especificar los requerimientos es una manera de hacer frente a este problema. Mientras que nos apoyamos en la colaboración para fomentar la cooperación entre los stakeholders, y así puedan explorar sus diferencias en forma constructiva y buscar soluciones que van más allá de su propia visión limitada.
El Léxico Extendido del Lenguaje que en sí mismo es un glosario con estructura más amplia que lo habitual, es una estrategia para capturar el lenguaje del contexto de la aplicación utilizando el lenguaje natural. El principal objetivo del LEL es conocer el vocabulario del problema. Está compuesto por símbolos que son palabras o frases repetidas o relevantes del universo del discurso [1] [20].
En el proceso de definición de requisitos se genera un compromiso entre los ingenieros y los clientes/usuarios, ya que ambos participan en la generación y definición de los requisitos del sistema, por lo tanto el uso de un glosario con términos comunes a todos los participantes facilita la comunicación y la compresión. Este glosario se va refinando en sucesivas etapas del desarrollo, y se lo utiliza para describir características del sistema y del modelo de caso de uso [16].
Para generar el LEL se registran símbolos (palabras o frases) peculiares o relevantes del dominio. Cada entrada del léxico se identifica con un nombre (o más de uno en caso de sinónimos) y tiene dos tipos de descripciones. Una llamada *Noción* que describe la denotación del símbolo y la otra *Impacto* que describe la connotación del mismo. Las entradas se clasifican en cuatro tipos de acuerdo a su uso general en el universo del discurso. Estos tipos son: Sujeto, Objeto, Verbo y Estado.
La aplicación que se desarrollará se basa en la estructura que tiene el LEL, resumida en el párrafo anterior.
La técnica LEL cuenta con la ventaja de poder obtener datos de salida a partir de los elementos definidos, por ejemplo el cálculo de los LEL Points.

1.5 Desarrollo propuesto

La aplicación web permitirá la creación de distintos LEL según se necesite.
Básicamente, construir un LEL se trata de crear símbolos de distintas categorías (sujeto, objeto, estado, verbo). Los símbolos son definidos a través de dos atributos: noción e impacto. El primer atributo describe la denotación, es decir, las características intrínsecas y sustanciales del símbolo, mientras que el segundo atributo describe la connotación. Como se definen en lenguaje natural se podrán agregar y/o quitar sinónimos.
Las nociones e impactos estarán compuestos por expresiones que pueden hacer referencia a símbolos ya definidos, en este caso, habrá una vinculación entre los mismos, facilitando de esta manera la “navegación” a través de los distintos símbolos del LEL que estén conectados.
Dado que se requiere como característica principal el trabajo colaborativo, existirá funcionalidad específicamente destinada a actividades sociales, que permitirá a los usuarios que trabajan sobre un mismo LEL dar su opinión respecto a lo aportado por sus compañeros, indicando like o dislike de un

símbolo y/o expresiones. Además, se podrán agregar comentarios, por lo tanto el autor de un símbolo o expresión, puede responder los comentarios recibidos si así lo desea y evaluar si es necesario realizar algún ajuste.
Para los casos en los que se desee hacer algún arreglo o mejora, se podrán realizar tareas de refactoring como renombrar un sinónimo o unificar dos símbolos (merge).
Para organizar el trabajo en grupo existirá la posibilidad de definir distintos roles. Por ejemplo, podrá haber un “Moderador” que sea el encargado de distribuir el trabajo entre todos los usuarios “comunes”, dar indicaciones y realizar las modificaciones que crea necesarias.
Finalmente, el sistema contará con la posibilidad de integrar plug-ins desarrollados externamente a este proyecto y así permitir la posibilidad de extender la herramienta.

1.6 Organización de la tesina

En el Capítulo 2 se abordará el concepto de Léxico Extendido del Lenguaje, cómo se construye en forma colaborativa actualmente y se presentará cómo se podrá construir a través de la aplicación propuesta en esta tesina. En el capítulo 3 se presentará la aplicación LELTool desarrollando sus funcionalidades y detallando la forma de utilización de la misma. Además se explicará la arquitectura y herramientas utilizadas para la implementación de la aplicación. En el capítulo 4 detallaremos los resultados obtenidos de realizar una prueba de usabilidad del sistema, utilizando una escala, conocida como SUS (System Usability Scale). Finalmente en el capítulo 5, se presentarán las conclusiones y trabajos futuros, en relación a la temática de la tesina.

CAPITULO 2 - BACKGROUND

2.1 Léxico Extendido del Lenguaje

El LEL (Language Extended Lexicon) [21] es un modelo que permite representar y documentar, con tecnología hipertextual un conjunto de símbolos que representan el lenguaje de la aplicación, sin necesidad de entender el problema. Consiste en una descripción de los términos significativos del macrosistema, acotando el lenguaje externo con el uso de símbolos definidos en el mismo LEL y a su vez, minimiza el uso de símbolos externos al lenguaje de la aplicación. El lenguaje de la aplicación es el lenguaje usado por los actores en todas las etapas de desarrollo y es una extensión del lenguaje natural usado normalmente. Una forma de obtener el lenguaje de aplicación es capturar los términos propios del mismo. Este lenguaje puede contener palabras y construcciones específicas del ambiente de la aplicación, o bien, expresiones del lenguaje natural con un significado distinto del corriente. Lo que se busca con el LEL es identificar estos símbolos, para su posterior definición con ayuda de los usuarios.

La Figura 1 muestra el Modelo del Léxico Extendido del Lenguaje.

LEL: representación de los símbolo en el lenguaje del dominio de la aplicación.
Sintaxis:
{Símbolo}$_1^N$
Símbolo: entrada del léxico que tiene un significado especial en el dominio de la aplicación.
Sintaxis:
{Nombre}$_1^N$ + {Noción}$_1^N$ + {Impacto}$_1^N$

Nombre: identificación del símbolo. Más de uno indica la presencia de sinónimos.
Sintaxis:
Palabra | Frase

Noción: denotación del símbolo. Debe ser expresada usando referencias a otros símbolos y usando el vocabulario mínimo.
Sintaxis:
Oración

Impacto: connotación del símbolo. Debe ser expresado usando referencias a otros símbolos y usando vocabulario mínimo.
Sintaxis:
Oración

donde Oración está compuesta solamente por Símbolos y No Símbolos, éstos pertenecientes al vocabulario mínimo

+ significa composición, {x} significa cero o más ocurrencias de x, | representa or

Figura 1- El modelo del Léxico Extendido del Lenguaje [17]

2.1.1 Objetivo

El principal objetivo del LEL es conocer el vocabulario del usuario. Que el usuario y el desarrollador compartan el mismo lenguaje asegura la comunicación entre ambos en particular, el uso del lenguaje propio del usuario mejora considerablemente esta comunicación. En el proceso de Ingeniería de Requerimientos, la validación de los diferentes productos requiere una fuerte interacción con el usuario, la que se ve facilitada por el vocabulario común usuario-desarrollador. Las diferentes representaciones que se construyen en el proceso de desarrollo de software encuentran en el vocabulario del usuario, un marco referencial que permite mantener la continuidad del vocabulario, lo que a su vez facilita el acceso a la documentación por parte de todos los participantes en el desarrollo. Otro objetivo de la utilización del LEL es la posibilidad de utilizarlo como herramienta para la capacitación de los miembros del equipo de desarrollo que se incorporan en etapas avanzadas del desarrollo, basado esto en una documentación consistente de los símbolos que representan el lenguaje de la aplicación. En [15] podemos encontrar un resumen de las relaciones establecidas en términos de objetivos a alcanzar y las consecuencias que se derivan de la satisfacción de esos objetivos relacionados con el uso del LEL en un proyecto de desarrollo. Este resumen lo podemos ver en la Tabla 1:

Objetivo	Consecuencia
Conocer el vocabulario del usuario	• Asegurar la comunicación • Facilitar la validación de los requerimientos con el usuario • Mantener el mismo vocabulario durante todo el proceso de desarrollo
Contar con un instrumento simple de trazabilidad	• Documentar consistentemente • Capacitar a nuevos miembros del equipo en la terminología empleada • Generar versiones del LEL a medida que evoluciona el proceso de desarrollo

Tabla 1: Objetivo-Consecuencia del LEL [15]

En [22] se propone construir el LEL como primer paso en la fase de elicitación de los requerimientos: “despreocuparse durante este procedimiento de entender el problema y abocarse sólo a conocer el vocabulario que utiliza el usuario en su mundo real”. Una vez familiarizados con ese léxico, el camino para comunicarse con el usuario y comprender el problema tendrá un obstáculo menos: el vocabulario.

2.1.2 Características del LEL

El LEL está compuesto por un conjunto de símbolos que identifican el lenguaje de la aplicación. Los símbolos son, en general, las palabras o frases utilizadas por el usuario y que repite con más frecuencia. También se incluyen aquellas palabras o frases que son relevantes para el dominio del problema, más allá de su frecuencia de repetición. Los símbolos se adquieren, por ejemplo, de entrevistas, observaciones, lectura de documentos [22]. Se genera una lista con todos los símbolos reconocidos. Durante el proceso de recolección, el ingeniero de software procura entender el significado de cada símbolo. La semántica de cada símbolo se representa con una o más nociones y uno o más impactos. El impacto puede no existir. La noción indica qué es el símbolo y el impacto, cómo repercute en el sistema. Por lo tanto, cada símbolo tiene un "nombre" que lo identifica, una "noción" y un "impacto" que lo describen [23].
A continuación, podemos ver un ejemplo de un símbolo del LEL:

Nombre	TRAMITE
Noción	El tramite tiene: • número de trámite • número de expediente • fecha de inicio • fecha ultimo estado • tipo de tramite • documentación faltante • observaciones El trámite tiene un estado y el historial asociado.
Impacto	El usuario con rol mesa de entrada inicia trámites. El administrativo inicia trámites. El administrativo modifica trámites. El administrativo finaliza trámites.

En la descripción de las nociones e impactos existen dos reglas básicas [15] [23], que se deben cumplir simultáneamente: una tiende a maximizar el uso de símbolos en el significado de otros símbolos, esto es acotar el lenguaje externo al dominio de la aplicación, utilizando para la descripción de la noción y el impacto símbolos definidos en el mismo LEL. Esto se denomina "principio de circularidad". La otra requiere que se minimice el uso de símbolos externos al lenguaje de la aplicación. Este principio se denomina "principio del vocabulario mínimo". Con estos principios se tiende a que se utilice lo más posible el lenguaje de la aplicación. Como resultado se obtiene un conjunto de símbolos que forman una red que permite representar al LEL en un hipertexto [22]. Los links lo determinan los símbolos que tienen una entrada en el LEL y se utilizan para definir otros. Lo que permite navegar entre ellos para conocer todo el vocabulario del dominio.

2.1.3 Proceso de construcción del LEL

Se ha propuesto que el proceso de construcción del LEL consta de 6 etapas interdependientes, que en algunos casos se desarrollan simultáneamente, las cuales podemos ver a continuación [15]:

- 1. Entrevistas.
- 2. Generación de la lista de símbolos.
- 3. Clasificación de los símbolos.
- 4. Descripción de los símbolos.
- 5. Validación con los clientes.
- 6. Control del LEL.

La Fig. 2 ilustra gráficamente las etapas recién mencionadas.
Las cuatro primeras muestran la corriente principal de construcción mientras que las dos restantes producen retroalimentaciones sobre aquéllas para permitir las correcciones necesarias.
Por otro lado, el LEL debe ser actualizado para reflejar la mejor comprensión del UdeD y, por lo tanto, puede ser mejorado durante el proceso de construcción de Escenarios [17].

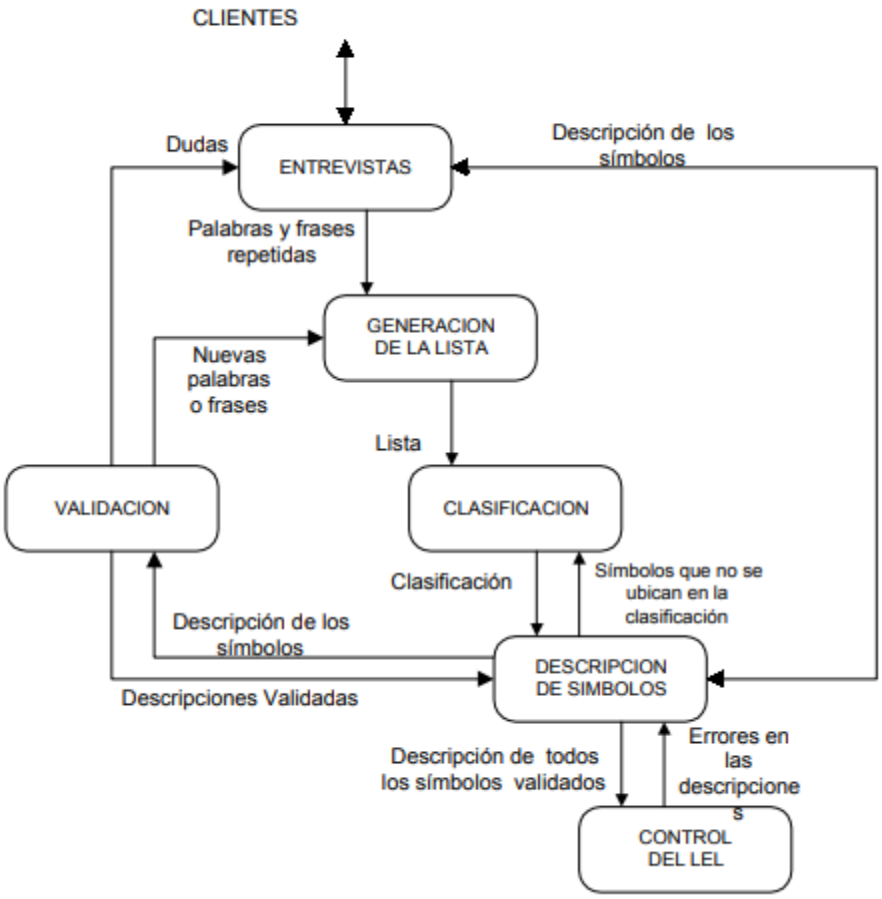

Figura 2: Etapas para la construcción del LEL [15]

2.1.3.1 Entrevistas

Permiten conocer el vocabulario que el cliente emplea en su medio ambiente. La cantidad de entrevistas necesarias depende de: [15]

- la complejidad de la aplicación;
- la experiencia que el ingeniero de requerimientos tiene en construir un LEL;
- los conocimientos que el cliente entrevistado tiene del tema en estudio.

Estas entrevistas, en primera instancia suelen ser reuniones con los usuarios. La cantidad de información que se puede extraer de una entrevista es grande. Una táctica recomendada en estas primeras reuniones, es que el entrevistador adopte una actitud pasiva, permitiendo que el entrevistado se exprese libremente con lo que se logrará que el mismo use su propio vocabulario y en forma inconsciente utilice el principio de circularidad. Con respecto a la elección de las personas a entrevistar, es recomendable buscar a las que toman decisiones, los jefes naturales, los responsables del proyecto y los supervisores. Hay que tener en cuenta que la idea principal es entender el lenguaje, de ningún modo se pretende determinar las funciones principales del sistema. Contrariamente al análisis de requerimientos tradicional, donde el analista identifica las funciones a partir de las entrevistas, el objetivo es familiarizarse con el lenguaje del dominio de la aplicación.

2.1.3.2 Generación de la lista de símbolos

La lista de símbolos es el conjunto de símbolos obtenidos durante las entrevistas. Pueden elegirse símbolos que parezcan estar fuera de contexto, esto es porque el significado con el que se utilizan es distinto al tradicional, así que es necesario definirlos dentro del ambiente de la aplicación. Primero se genera una lista de símbolos candidatos, que es la lista inicial obtenida de las primeras entrevistas, en las que se registran las palabras o frases que el cliente repite o enfatiza. Esta lista se modifica con frecuencia después de las primeras validaciones, que es cuando se incorpora mayor cantidad de símbolos y sinónimos. Se eliminan símbolos erróneos y se precisan los conceptos confusos. Esta lista rectificada pasa a ser la lista definitiva ya que las modificaciones futuras suelen ser mínimas. Durante la generación de la lista candidata se determina el punto de vista que tendrá el LEL. Esto permite homogeneizar las descripciones, más aún cuando participan varios ingenieros de requerimientos en la construcción del LEL [15].
En cuanto a la recolección de símbolos, un elemento a tener en cuenta es que, en la lista de símbolos obtenidos, pueden existir sinónimos. Estos sinónimos se pueden ir marcando en la medida de lo posible en esta fase. Sin embargo, los sinónimos quedan determinados de manera precisa en la parte de definición, donde se detectan descripciones similares para símbolos distintos.

2.1.3.3 Clasificación de los símbolos

La clasificación de los símbolos se utiliza para asegurar la integridad y homogeneidad de las descripciones [24]. Partimos de la clasificación general del LEL definida en [21], donde se dividen los símbolos en **Sujeto**, **Verbo**,

Objeto y **Estado**. Esta categorización guía y asiste a los ingenieros de requerimientos durante la descripción de distintos atributos. La Tabla 2 muestra cada categoría con sus características y cómo describirlas.

Categoría	Características	Noción	Impactos
Sujeto	Elementos activos que realizan acciones	Características o condición que el sujeto satisface	Acciones que realiza el sujeto
Objeto	Elementos pasivos sobre los cuales los sujetos realizan acciones	Características o atributos que tiene un objeto	Acciones que son realizadas en el objeto
Verbo	Acciones que los sujetos realizan sobre los objetos	Objetivo que el verbo persigue	Pasos necesarios para completar la acción
Estado	Situaciones en las que sujetos y objetos pueden estar	Situación representada	Acciones que deben ser realizadas para cambiar de un estado a otro

Tabla 2: Categorías del LEL

2.1.3.4 Descripción de los símbolos

Describir los símbolos es determinar su noción y su impacto. Durante la generación de la lista candidata se percibe inicialmente el significado de los símbolos, ya que las palabras o frases significativas del dominio están dentro de un contexto que el ingeniero de requerimientos está escuchando. Esto permite esbozar una primera descripción de algunos símbolos. Luego de validar con los clientes y determinar la clasificación, se describen los restantes símbolos. A continuación, se detallan una serie de reglas que pueden ayudar en este proceso:

- Un símbolo puede tener una o más nociones y cero o más impactos.
- Cada noción e impacto debe ser descrito con oraciones breves y simples.
- Nociones e impacto para un símbolo pueden representar diferentes puntos de vista o pueden ser complementarios.
- Las oraciones breves y simples de las nociones e impactos deben responder a los principios de circularidad y de vocabulario mínimo.
- Para los símbolos que son sujeto de una oración, los impactos deben indicar las acciones que realiza.

- Para los símbolos que cumplen el rol de verbo, las nociones deben decir quién ejecuta la acción, cuándo sucede y el proceso involucrado en la acción. Para los impactos se deben identificar las restricciones sobre la realización de la acción, qué es lo que origina esta acción y qué es lo que causa esta acción.
- Para los símbolos que son objetos de una oración, la noción debe identificar otros objetos con los cuales se relaciona y los impactos serán las acciones que se pueden realizar con este signo.

2.1.3.5 Validación con los clientes

La validación permite rectificar o ratificar el conocimiento del vocabulario del dominio obtenido durante las entrevistas. Las primeras validaciones permiten chequear la lista candidata y conocer el significado de los símbolos. Las siguientes validaciones son, generalmente, para chequear que el conocimiento obtenido sea el correcto y que no falte información, de no ser así se agrega o modifica el LEL.

2.1.3.6 Control del LEL

Este proceso consiste en controlar que todos los símbolos cumplan con la formulación de la clasificación a la que pertenecen, que no existan sinónimos como símbolos diferentes y que la sintaxis empleada sea la misma. Es decir, se controla que el LEL terminado para esta etapa, sea consistente y homogéneo. Este control comienza en las primeras descripciones de los símbolos y es fundamental cuando participan varios entrevistadores.

2.2 Colaboración

La colaboración es el proceso de dos o más personas u organizaciones que trabajan juntas para completar una tarea o lograr un objetivo [25]. La colaboración es similar a la cooperación. La mayor parte de la colaboración requiere liderazgo, aunque la forma de liderazgo puede ser social dentro de un grupo descentralizado [26].
Los métodos estructurados de colaboración fomentan la introspección del comportamiento y la comunicación [26]. Tales métodos apuntan a aumentar el éxito de los equipos a medida que se involucran en la resolución colaborativa de problemas.
En su sentido aplicado, "(a) la colaboración es una relación intencional en la que todas las partes eligen estratégicamente cooperar para lograr un resultado compartido" [27].

La colaboración en tecnología abarca una amplia gama de herramientas que permiten que grupos de personas trabajen juntos, incluidas redes sociales,

mensajería instantánea, *web sharing*, audio conferencia, video y telefonía. Muchas compañías grandes adoptan plataformas de colaboración para permitir que los empleados, clientes y socios se conecten e interactúen de manera inteligente.

La efectividad de un trabajo colaborativo está impulsada por tres factores críticos: comunicación, gestión de contenido y flujo de trabajo.

2.2.1 Edición colaborativa

La edición colaborativa es la edición realizada por grupos que producen en conjunto a través de contribuciones individuales. Las buenas decisiones, participación y coordinación grupal son fundamentales para el éxito de los resultados de la escritura colaborativa [28].
La escritura colaborativa es la escritura hecha por más de una persona. Se puede organizar de distintas formas, por ejemplo, discutir lo que se va a escribir antes de comenzar y discutir lo que se ha escrito después de terminar cada borrador [29]. También se puede organizar dividiendo la escritura en sub-tareas asignadas a cada miembro del grupo, continuando con la tarea siguiente una vez terminada la primera, o bien trabajando juntos en cada tarea [30] [31]. La escritura es planeada, escrita y revisada, con más de una persona involucrada en al menos uno de esos pasos [32]. Generalmente, las discusiones sobre la estructura y el contexto del documento involucran a todo el grupo [33].
La edición colaborativa, por lo general, se aplica a documentos de texto o código fuente programático. Dichas contribuciones asíncronas (no simultáneas) son muy eficientes en el tiempo, ya que los miembros del grupo no necesitan reunirse para trabajar juntos. En general, la gestión de este tipo de trabajo requiere la utilización de herramientas de software [34]. Las herramientas más comunes para editar documentos son wikis, mientras que para la programación, se utilizan los sistemas de control de versiones. La mayoría de los procesadores de texto también son capaces de registrar cambios, lo que permite a los editores trabajar en el mismo documento mientras que automáticamente se etiqueta quién realizó determinados cambios. Los nuevos entornos de escritura, como Google Docs, proporcionan funciones de escritura/ediciones colaborativas con control de revisión y edición síncrona/asíncrona.

2.2.2 Software colaborativo

El software colaborativo o software grupal es un software de aplicación diseñado para ayudar a las personas involucradas en una tarea común a lograr sus objetivos. Una de las primeras definiciones de software colaborativo es

"procesos grupales intencionales más la utilización de software para respaldarlos" [35].

En términos del nivel de interacción que permite el software colaborativo se puede dividir en: plataformas de edición colaborativa en tiempo real (Real Time Collaborative Edition, RTCE) que permiten a múltiples usuarios participar en la edición en vivo, simultánea y reversible de un solo archivo (generalmente un documento), y las plataformas de control de versión (también conocidas como control de revisión y control de fuente), permiten a los usuarios realizar ediciones paralelas a un archivo, al tiempo que conservan cada edición guardada por el usuario como archivos múltiples (que son variantes del archivo original).

El software colaborativo se puede dividir en tres categorías según el nivel de colaboración [36]:

- Comunicación: puede pensarse como un intercambio no estructurado de información.
- Colaboración: se refiere al trabajo interactivo hacia un objetivo compartido facilitando un espacio de trabajo virtual rico y compartido.
- Coordinación: se refiere al trabajo complejo e interdependiente hacia un objetivo compartido. Una buena metáfora para entender esto es pensar en un equipo deportivo: todos deben contribuir con el juego correcto en el momento adecuado, así como ajustar su juego a la situación en desarrollo, pero todos están haciendo algo diferente, para que el equipo gane. Eso es un trabajo complejo e interdependiente hacia un objetivo compartido: la gestión colaborativa.

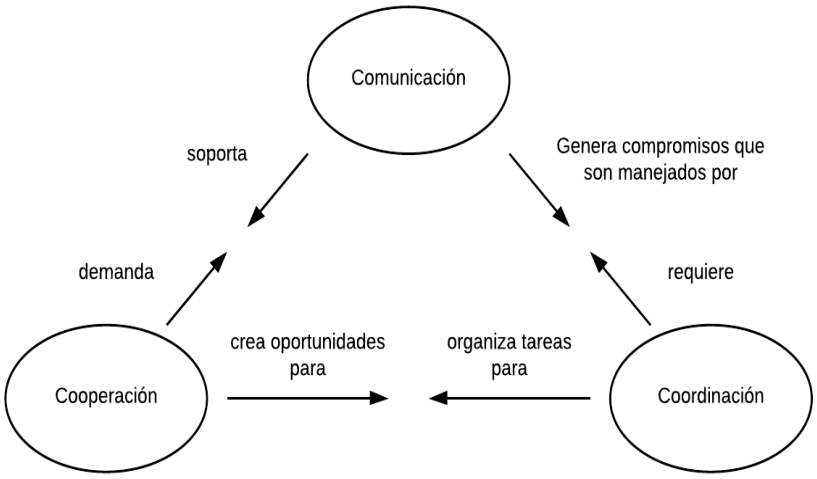

Figura 3 – Categorías del software colaborativo

2.2.3 Herramientas de colaboración

El propósito de una herramienta de colaboración es apoyar a un grupo de dos o más personas para lograr un objetivo en común [37]. Las herramientas de colaboración pueden ser de naturaleza tecnológica como lo son las herramientas de software, como no tecnológica, por ejemplo, papel, rotafolios, notas post-it o pizarras blancas [38].
Como vimos, según el nivel de colaboración el software colaborativo se puede dividir en tres categorías: comunicación, colaboración y coordinación. A continuación, veremos las herramientas de colaboración que existen actualmente según cada categoría:

Comunicación
Las herramientas de esta categoría permiten intercambiar información entre individuos [39]. Algunos ejemplos son:

- E-mail: la invención del correo electrónico como herramienta de colaboración cambió la forma en que solíamos comunicarnos, especialmente en el lugar de trabajo. Permite organizar la correspondencia diaria y contactar a varias personas con un solo clic [40]. Sin embargo, aunque el correo electrónico sigue siendo la herramienta utilizada más comúnmente en la colaboración para la comunicación, no es muy eficiente a gran escala. Pese a su flexibilidad, no está muy bien para conversaciones grupales, ya que crecen demasiado rápido y si se comparten documentos no hay forma de estar seguro de que se tiene la última versión de los mismos. Tampoco es posible realizar un seguimiento de las tareas que deben realizarse y la fecha límite de cada una de ellas [41].
- Correo de voz: el correo de voz como herramienta de colaboración se integra cada vez más en servicios como Google Voice. Como se señaló en un escenario futuro de IBM, el rol del correo de voz podría ser el que tiene el correo electrónico de hoy [42].
- Mensajería instantánea: a través de la mensajería instantánea como herramienta de colaboración, podemos llegar a las personas dentro de una organización en tiempo real. En el futuro, la mensajería instantánea ya no es un software independiente, pero está muy bien integrado en soluciones más grandes como la Comunicación Unificada [43].
- VoIP (*voice over IP*): la voz a través de IP como herramienta de colaboración ha ganado rápidamente popularidad entre las empresas y forma parte de su cartera de comunicaciones. Como señala un informe de Eclipse Telecom, VoIP se está moviendo en dirección a reemplazar totalmente nuestros teléfonos en nuestras oficinas y también para integrarse en los entornos de servicio de colaboración existentes [44].

Coordinación

Las herramientas de colaboración de esta categoría son las que le permiten configurar actividades de grupo, horarios y entregables. Algunos ejemplos son:

- Calendario online: los calendarios en línea son parte de nuestro comportamiento profesional en el trabajo y están totalmente integrados en otros sistemas. Como explica un artículo de investigación de la Universidad de Bath, los calendarios en línea podrían en el futuro estar estrechamente relacionados con otros datos, como datos sociales, y tener un efecto aún mayor [45].
- Time trackers: los contadores de horas se utilizan especialmente para medir el rendimiento de los empleados. La discusión sobre el efecto sobre la productividad es controvertido [46].
- Spreadsheets: las hojas de cálculo son como los correos electrónicos muy populares en el entorno corporativo y son una herramienta de colaboración esencial para el análisis o modelado financiero. Sin embargo, aunque son muy populares, varios estudios descubrieron que muchas hojas de cálculo contienen datos inexactos y, por lo tanto, son ineficientes [47].

Colaboración

Las herramientas de colaboración permiten a los grupos tener discusiones en tiempo real y dar forma a una idea o pensamiento juntos. Las tendencias en términos de colaboración apuntan a ayudar a mantener la "idea principal". También la idea de traer personas que no están trabajando en la organización y hacer uso de sus conocimientos [48].

- Video conferencia: en la mayoría de los casos, la videoconferencia es parte de la estrategia general de comunicación y colaboración de las organizaciones. Especialmente ahora, cuando todos los servicios se basan en la nube y, por lo tanto, los costos de implementación se vuelven muy accesibles. La visión a largo plazo para las video conferencias se basa en el uso correcto de la potencia de procesamiento de la computadora, el almacenamiento de datos o las velocidades de ancho de banda móvil para disminuir aún más los obstáculos de la colaboración [49].
- Teleconferencia: la propuesta de las soluciones de teleconferencia es llevar equipos, reuniones o eventos lo más cerca posible. Además de los entornos empresariales, las teleconferencias se utilizan actualmente en diversos campos, como la telemedicina, contribuyendo enormemente a la eficiencia y la productividad, ya que la distancia y el tiempo son factores limitantes [50].

Finalmente, las herramientas de colaboración se pueden clasificar en:

- Sincrónicas: una herramienta colaborativa es sincrónica cuando los usuarios colaboran al mismo tiempo [51]. Algunos ejemplos de estas herramientas son: pizarras compartidas, sistemas de comunicación a través de video, sistemas de chat, sistemas de soporte de decisión, video juegos multijugador, entre otros.
- Asincrónicas: una herramienta colaborativa es asincrónica cuando los usuarios colaboran en momentos diferentes [51]. Algunos ejemplos de estas herramientas son: e-mail, calendario grupal, sistemas de flujo de trabajo, entre otras.

2.2.4 Trabajo colaborativo en un proyecto LEL

Es muy difícil producir una especificación del lenguaje del dominio cuando hay muchos stakeholders involucrados. La interacción humana es el elemento que hace que la elicitación de requerimientos sea la actividad más dificultosa de escalar en la ingeniería de software [52] [53]. En un contexto colaborativo, todos los participantes construyen juntos incluso si la tarea se puede dividir en varias subtareas nuevas. La obtención de requisitos, como un proceso interdisciplinario, requiere competencias específicas de todos los usuarios y partes interesadas involucradas. La colaboración es por lo tanto necesaria, ya que ninguna persona puede poseer todas las competencias requeridas para esta tarea. Además, la buena obtención de requisitos de colaboración produce requisitos más ricos, más completos y más consistentes [54].

EL proceso tradicional para construir un LEL consiste en dos actividades: identificar y describir los símbolos. El enfoque colaborativo en el que se basa esta tesina propone
ambas actividades encuadradas en lo que formaría parte de un proyecto LEL, y en adición dos actividades sociales: todos los usuarios que colaboran en un mismo proyecto pueden agregar y responder comentarios y también, pueden indicar like o dislike de símbolos y/o expresiones.

CAPITULO 3 - CONTRIBUCION: LELTool

3.1 Introducción

Actualmente para construir un LEL pensamos en un editor de texto donde se va a escribiendo en forma secuencial los símbolos. No hace falta que el LEL sea demasiado grande para que se completen en forma rápida numerosas hojas, haciendo que el proceso de construcción del LEL sea algo lento y dificultoso a medida que este va creciendo. El ingeniero de software puede llegar a invertir demasiado tiempo en cuestiones que no hacen a la elicitación de requerimientos propiamente dicha, sino que necesita realizar tareas de administración de la información obtenida, como por ejemplo: insertar nociones e impactos a símbolos ya existentes, ordenar todas las entradas alfabéticamente por símbolo, marcar símbolos en distintos colores, etc. Este problema justifica el propósito de esta tesina, que es desarrollar una herramienta para soportar la producción de LEL en forma colaborativa, para facilitar la navegación entre los distintos elementos y la administración de la información obtenida para conocer todo el vocabulario del dominio.

LELTool es una aplicación web que provee utilidades para construir uno o más LEL en forma colaborativa. Posee una interfaz intuitiva pensada para que los usuarios puedan utilizarla en forma fluida, optimizando el tiempo de trabajo invertido y agilizando la comunicación entre los mismos.

Para llevar a cabo las funcionalidades de LELTool, se debió establecer la arquitectura y las herramientas a utilizar. La Arquitectura del Software es la organización de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implementarán, los principios que orientan su diseño y evolución. Naturalmente este diseño arquitectónico ha de ajustarse a las necesidades y requisitos del proyecto [55]. Una herramienta de desarrollo de software es un programa informático que se utiliza para crear, depurar, gestionar y mantener un programa. Generalmente hace referencia a programas relativamente simples, que pueden ser combinados para llevar a cabo una tarea.
A continuación, se detallarán los componentes correspondientes a la arquitectura y las herramientas utilizadas en LELTool.

3.2 Arquitectura

La aplicación LELTool se puede dividir en tres niveles:

1. **Nivel de presentación:** es el encargado de generar la interfaz de usuario en función de las acciones llevada a cabo por el mismo.

2. **Nivel de negocio:** contiene toda la lógica que modela los procesos de negocio y es donde se realiza todo el procesamiento necesario para atender las peticiones del usuario.
3. **Nivel de administración de datos:** encargado de hacer persistente toda la información, suministra y almacena información para el nivel de negocio.

Los dos primeros y una parte del tercero (el código encargado de las actualizaciones y consultas), suelen estar en el servidor mientras que la parte restante del tercer nivel se sitúa en la base de datos.

Al desarrollar la arquitectura se tomó como paradigma de programación la Programación Orientada a Objetos (POO). Este paradigma permite analizar, diseñar y desarrollar sistemas con una brecha menor entre la realidad y el modelo.
Por otro lado, la POO brinda herramientas que permiten que el código resultante tenga mayores niveles de flexibilidad, extensión y calidad.
Luego, la definición de objetos está contenida en los lenguajes de programación de la actualidad (.NET), de modo que se facilita la integración entre sistemas construidos con dichos lenguajes, en particular a través de servicios.

En los lenguajes de programación modernos (Java, .NET, etc.) existe gran variedad de frameworks que brindan las funcionalidades necesarias para cubrir distintas necesidades de base de las aplicaciones (logging, ORM, etc.), siendo algunos de estos frameworks estándares de mercado.
La arquitectura se basa en la utilización de frameworks considerados estándar (o de gran utilización en el mercado), y esto a su vez permite contar con amplia documentación y ejemplos de uso.

Para LELTool la arquitectura se implementa totalmente sobre el Framework ASP .NET Core de Microsoft, utilizando la versión 2 del mismo para aprovechar la totalidad de las funcionalidades provistas por el framework en las distintas capas de la arquitectura, lo cual favorece el desarrollo al contar con herramientas de base sin necesidad de desarrollar las mismas.

A continuación, la Figura 4 se muestra un diagrama de la arquitectura mencionada en el párrafo anterior:

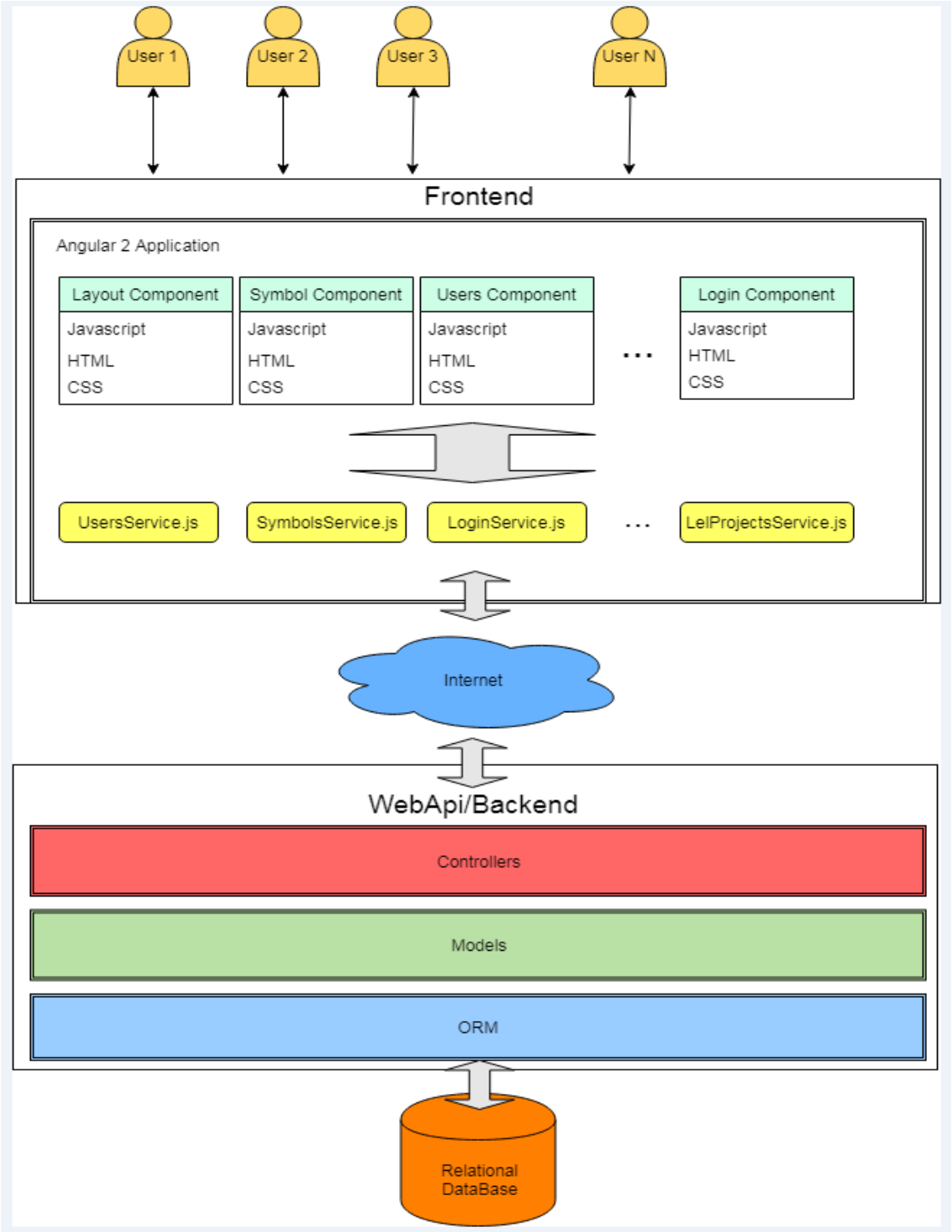

Figura 4 – Arquitectura

A continuación, se detallarán los frameworks/componentes que se utilizaron en la implementación de la arquitectura antes mencionada.

3.2.1 ASP .NET Core

ASP .NET Core [56] es una multiplataforma de código abierto, de alto rendimiento para la construcción de aplicaciones modernas, basadas en la nube, conectadas a Internet. Con ASP .NET Core, es posible crear aplicaciones y servicios web, aplicaciones de IoT (Internet of Things) y backends móviles, usar las herramientas de desarrollo en Windows, MacOs y Linux, desplegar en la nube y ejecutar .NET Core o .NET Framework.
ASP .NET Core provee los siguientes beneficios:

- Una story unificada para construir UI y APIs web.
- Integración de *workflows* de desarrollo y *client-side* frameworks.
- Inyección de dependencias incorporada.
- Un *pipeline* de solicitudes HTTP ligero, de alto rendimiento y modular.
- Posibilidad de host en IIS, Nginx, Apache, Docker o auto-host en su propio proceso.
- Herramientas que simplifican el desarrollo web moderno.
- Posibilidad de compilar y ejecutar en Windows, macOS y Linux.
- Es de código abierto y centrado en la comunidad.

3.2.2 Angular 6

Angular [57] es un framework para aplicaciones web desarrollado en TypeScript, de código abierto, mantenido por Google. Angular se utiliza para crear y mantener aplicaciones web de una sola página, combinando plantillas declarativas, inyección de dependencias, herramientas de punto a punto y buenas prácticas integradas para resolver los desafíos de desarrollo. Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles.
Entre las principales características de Angular podemos mencionar:

- Generación de código: Angular convierte plantillas en código altamente optimizado para las máquinas virtuales de JavaScript de hoy en día, ofreciendo todas las ventajas del código escrito a mano con la productividad de un framework.
- División del código: Angular posee un enrutador de componentes que ofrece una división automática de códigos para que los suuarios sólo carguen el código necesario para procesar la vista que solicitan.
- Plantillas: Angular permite crear rápidamente vistas de interfaz de usuario con una sintaxis simple y potente.
- Angular CLI: las herramientas de línea de comandos permiten empezar a desarrollar rápidamente, añadir componentes y realizar test, así como previsualizar de forma instantánea la aplicación.

- Animación: Angular permite crear animaciones complejas y de alto rendimiento con muy poco código a través de su intuitiva API.

3.2.3 Entity Framework Core

Entity Framework (EF) Core [58] es una versión ligera, extensible, de código abierto y multiplataforma de la popular tecnología de acceso a datos de Entity Framework.

EF Core puede servir como un mapeador relacional de objetos (O/RM), lo que permite a los desarrolladores .NET trabajar con una base de datos utilizando objetos .NET, y elimina la necesidad de la mayoría del código de acceso a datos que normalmente necesitan escribir.

3.2.4 TypeScript 2

TypeScript [59] es un lenguaje de programación libre y de código abierto desarrollado y mantenido por Microsoft. Es un superconjunto de JavaScript, que esencialmente añade tipado estático y objetos basados en clases. El sistema de tipos de Typescript realiza una formalización de los tipos de Javascript, mediante una representación estática de sus tipos dinámicos. Esto permite a los desarrolladores definir variables y funciones tipadas sin perder la esencia de Javascript. TypeScript extiende la sintaxis de JavaScript, debido a esto cualquier código JavaScript existente funciona sin problemas, por lo tanto puede ser usado para desarrollar aplicaciones JavaScript que se ejecutarán en el lado del cliente o del servidor.

Motor de base de datos: MySQL

MySQL [60] es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual: Licencia pública general/Licencia comercial por Oracle Corporation y está considerada como la base datos de código abierto más popular del mundo, y una de las más populares en general junto a Oracle y Microsoft SQL Server, sobre todo para entornos de desarrollo web.
El desarrollo de MySQL se fundamenta en el trabajo de los desarrolladores contratados por la empresa MySQL AB quienes se encargan de dar soporte a los socios comerciales y usuarios de la comunidad MySQL y dar solución a los problemas encontrados por los usuarios. Los usuarios o miembros de la comunidad MySQL pueden reportar bugs revisando el manual en línea que contiene las soluciones a problemas encontrados.

3.3 Herramientas utilizadas para el desarrollo

A continuación, se detallan las herramientas utilizadas para el desarrollo de la aplicación web LELTool.

3.3.1 Editor de código Visual Studio Code

Visual Studio Code [61] es un editor de código liviano pero potente que está disponible para Windows, macOS y Linux. Posee soporte incorporado para JavaScript, TypeScript y Node.js y además la posibilidad de incorporar extensiones para otros lenguajes (como C++ , C#, Python, PHP, Go) y runtimes (como .NET y Unity). Esta herramienta permite realizar diversas operaciones de desarrollo como depuración, ejecución de tareas y control de versiones. Su principal objetivo es proporcionar solo las herramientas que un desarrollador necesita para un ciclo rápido de desarrollo y depuración de código dejando los flujos de trabajo más complejos a los IDEs más completos. Es gratuito y código abierto.

3.3.2 MySQL Workbench

MySQL Workbench [62] es una herramienta visual unificada para arquitectos de bases de datos, desarrolladores y administradores de bases de datos (DBAs). Proporciona modelado de datos, desarrollo en SQL y herramientas de administración integral para la configuración de servidores, administración de usuarios, copia de seguridad, entre otros. Está disponible en Windows, Linux y Mac OS X.
Como características principales podemos mencionar:

- Diseño: permite a un DBA, desarrollador o arquitecto de datos diseñar, modelar, generar y administrar bases de datos visualmente. Incluye todo lo que un modelador de datos necesita para crear modelos de ER complejos, realizar ingeniería directa e inversa, y también ofrece funciones clave para realizar tareas difíciles de administración y documentación de cambios que normalmente requieren mucho tiempo y esfuerzo.
- Desarrollo: ofrece herramientas visuales para crear, ejecutar y optimizar consultas SQL. El Editor de SQL proporciona resaltado de sintaxis con color, autocompletado, reutilización de fragmentos de SQL e historial de ejecución de SQL. El Panel de conexiones de bases de datos permite a los desarrolladores administrar fácilmente las conexiones de bases de datos estándar, incluido *MySQL Fabric*.
- Administración: proporciona una consola visual para administrar fácilmente los entornos de MySQL y obtener una mejor visibilidad de las bases de datos. Los desarrolladores y DBA pueden usar las herramientas visuales para configurar servidores, administrar usuarios,

realizar copias de seguridad y recuperación, inspeccionar datos de auditoría y ver el estado de la base de datos.

- Tablero de rendimiento: proporciona un conjunto de herramientas para mejorar el rendimiento de las aplicaciones MySQL. Los DBA pueden ver rápidamente los indicadores clave de rendimiento utilizando el Tablero de rendimiento. Los informes de rendimiento brindan una fácil identificación y acceso a puntos de conexión IO, declaraciones SQL de alto costo y más. Además, con un solo clic, los desarrolladores pueden ver dónde optimizar su consulta con el *Visual Explain Plan*.
- Tablero de rendimiento visual: proporciona un conjunto de herramientas para mejorar el rendimiento de las aplicaciones MySQL. Los DBA pueden ver rápidamente los indicadores clave de rendimiento utilizando el Tablero de rendimiento. Los informes de rendimiento brindan una fácil identificación y acceso a puntos de conexión IO, declaraciones SQL de alto costo, entre otros.
- Migración de base de datos: proporciona una solución completa y fácil de usar para migrar Microsoft SQL Server, Microsoft Access, Sybase ASE, PostreSQL y otras tablas, objetos y datos de RDBMS a MySQL. Los desarrolladores y DBA pueden convertir rápida y fácilmente aplicaciones existentes para ejecutar en MySQL tanto en Windows como en otras plataformas. La migración también admite la migración de versiones anteriores de MySQL a las últimas versiones.

3.3.3 GitHub

GitHub [63] es una plataforma de desarrollo colaborativo para alojar proyectos y crear software junto a millones de desarrolladores utilizando Git. Git es un software de control de versiones pensado para la eficiencia y confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente. Su propósito es llevar registro de los cambios en archivos de computadora y coordinar el trabajo que varias personas realizan sobre archivos compartidos.
Las principales características de GitHub son:

- Wiki para cada proyecto
- Página web para cada proyecto.
- Gráfico para ver cómo los desarrolladores trabajan en sus repositorios y bifurcaciones del proyecto.
- Funcionalidades como si se tratase de una red social, por ejemplo: seguidores.
- Herramienta para trabajo colaborativo entre programadores.
- Gestor de proyecto de estilo Kanban.

3.4 Manual de usuario

LelTool permite que distintas personas se registren como usuarios y puedan crear, modificar y eliminar proyectos. Cada proyecto representa un LEL, dentro del mismo se pueden agregar, modificar y eliminar símbolos. Un usuario puede invitar a otros a que formen parte de un mismo proyecto, pudiendo trabajar en forma colaborativa a través de comentarios y de likes/dislikes de símbolos y/o expresiones. A continuación se detalla cada uno de ellos.

3.4.1 Login y registración

Al ingresar a la aplicación se encontrará con una pantalla como se muestra en la Figura 5:

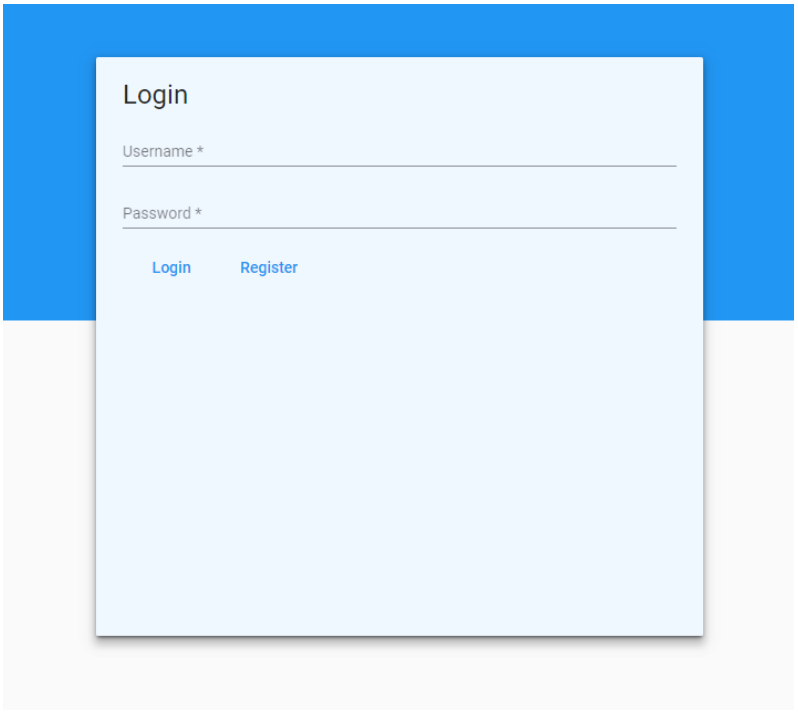

Figura 5 – Inicio LELTool

Si el usuario ya está registrado ingresa su nombre de usuario y contraseña y hace click en “Login”. En caso de no estar registrado, al hacer click en “Register” podrá ingresar su nombre de usuario, contraseña, correo electrónico y opcionalmente nombre y apellido. La Figura 6 muestra la pantalla de registración.

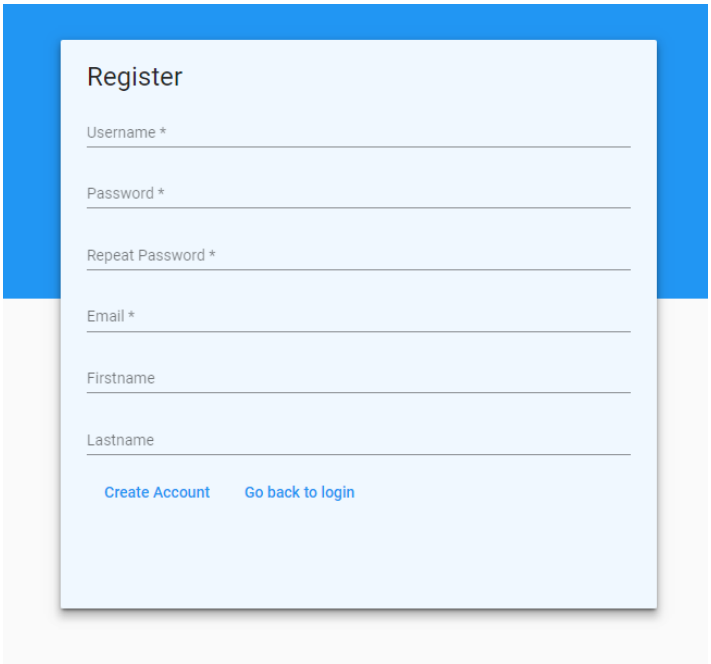

Figura 6 – Registro de usuario

3.4.2 Pantalla principal

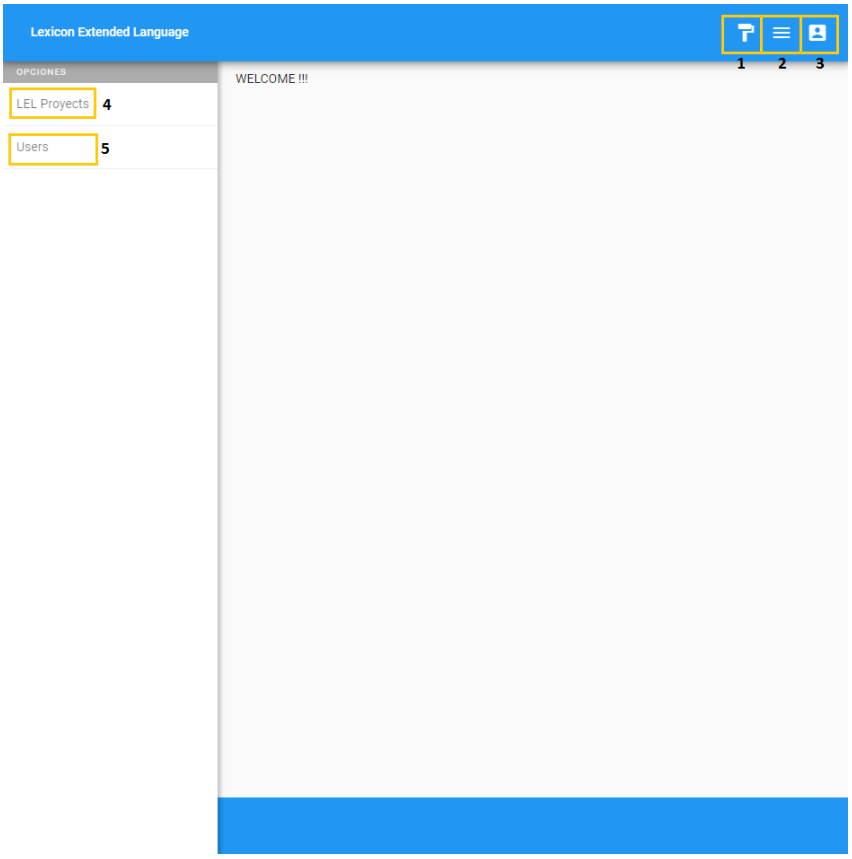

Figura 7 – Pantalla principal

Si se hace click en **(1)** se desplegará un panel para cambiar el color de la vista. Si se hace click en **(2)** se encontrá la opción “Help” que abre el manual de usuario. Si se hace click en **(3)** se encontrarán dos opciones “Settings” para cambiar los datos de usuario y “Logout” para cerrar sesión. Si se hace click en **(4)** se desplegarán los proyectos en los que el usuario logueado está participando. Si se hace click en **(5)** se podrá ver el listado completo de usuarios de LELTool. En las siguientes secciones se detallan estas funcionalidades.

3.4.3 Settings

Los datos de usuario se pueden editar desde la pantalla principal a través de la opción “Settings” como se mencionó anteriormente, se verá la siguiente pantalla como muestra la Figura 8.

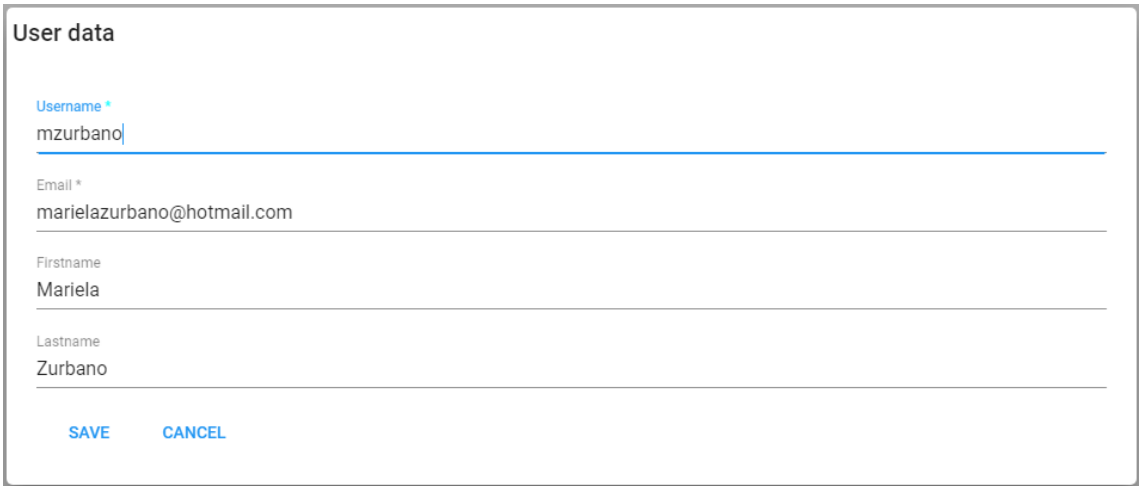

Figura 8 – Modificación de datos de usuario

Se pueden editar los campos deseados y al hacer click en “Save” se guardarán los cambios, o bien “Cancel” para descartarlos.

3.4.4 Proyectos

Haciendo click en “LEL Projects” se podrá ver en una grilla todos los proyectos en los que participa el usuario logueado, ya sea como administrador o como colaborador, como se muestra en la Figura 9.

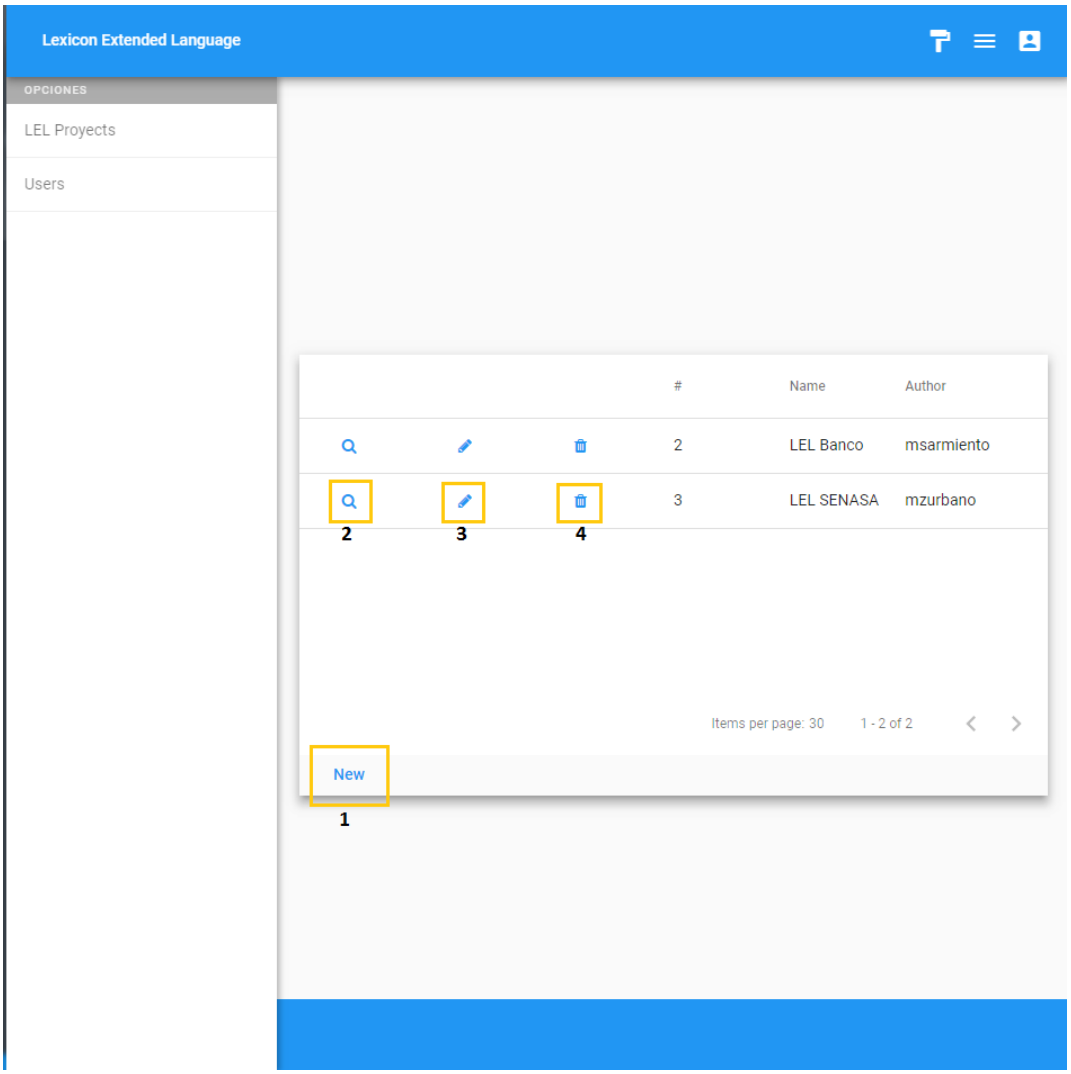

Figura 9 – Proyectos

Si se hace click en **(1)** se abrirá un modal panel para crear un proyecto ingresando el nombre del mismo. Si se hace click en **(2)** se accede al detalle de un proyecto. Si se hace click en **(3)** se abrirá un modal panel para editar el nombre del mismo. Por último, si se hace click en **(4)** se elimina un proyecto completo (esta acción requiere de confirmación). En las siguientes secciones se detallan estas funcionalidades.

3.4.4.1 Crear proyecto

En la Fig. 10 se muestra la pantalla para crear un proyecto ingresando el nombre que deseemos.

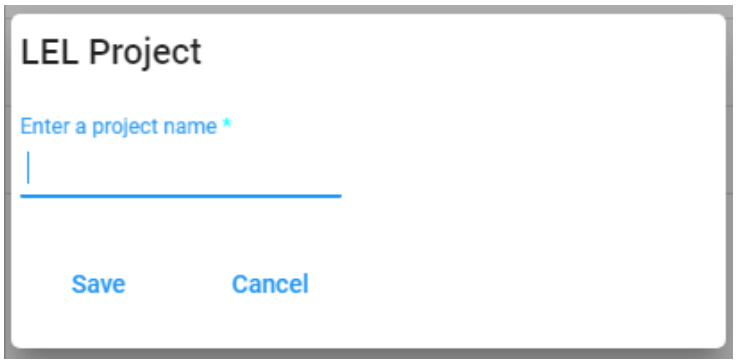

Figura 10 – Crear proyecto

3.4.4.2 Ver detalle de un proyecto

En la Figura 11 se muestra el detalle de un proyecto LEL donde podemos acceder a ver los símbolos de este y también ver quiénes están participando de su construcción en forma colaborativa.

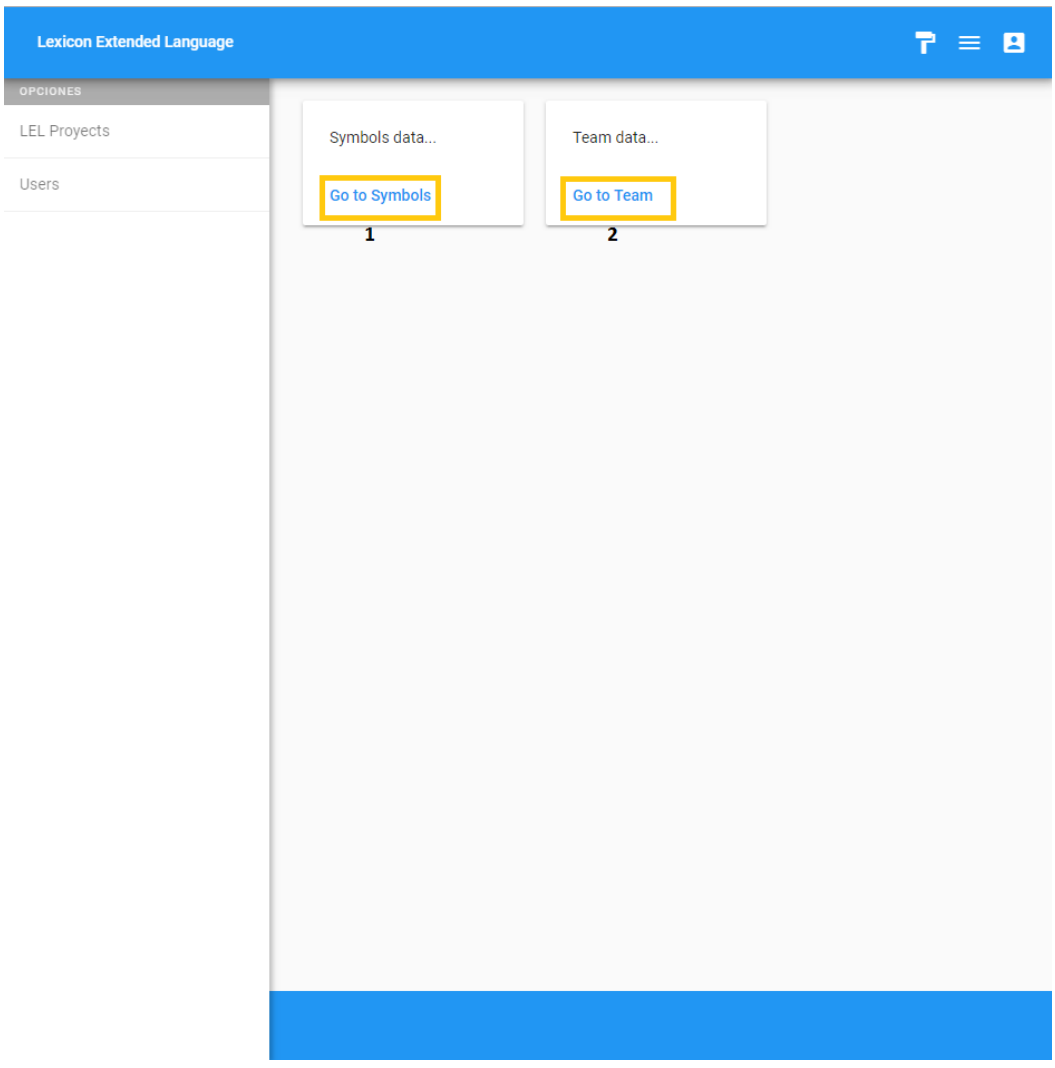

Figura 11 – Detalle de un proyecto

Si se hace click en **(1)** se podrá acceder al listado de símbolos del proyecto, en la sección 3.6.6 se mostrará en detalle toda la funcionalidad completa de símbolos. Si se hace click en **(2)** se podrá acceder al listado de usuarios que participan en la construcción del LEL, en caso de ser administrador se tendrá la opción de "invitar" a otros usuarios, en la sección 3.4.7 se mostrará en detalle esta funcionalidad.

3.4.4.3 Modificar proyecto

Figura 12 – Modificación datos de un proyecto

Si se hace click en "Save" se guardará el nuevo nombre. Si se hace click en "Cancel" se descartan los cambios.

3.4.4.4 Eliminar proyecto

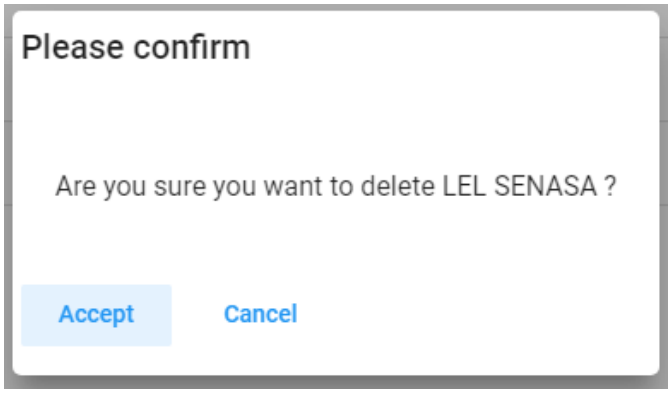

Figura 13 – Eliminar proyecto

Si se hace click en "Accept" se eliminará el proyecto. Si se hace click en "Cancel" la acción de eliminar no se lleva a cabo.

3.4.5 Usuarios

Al hacer click en “Usuarios” se verá el listado completo de usuarios de la herramienta LELTool, como se muestra en la Figura 14:

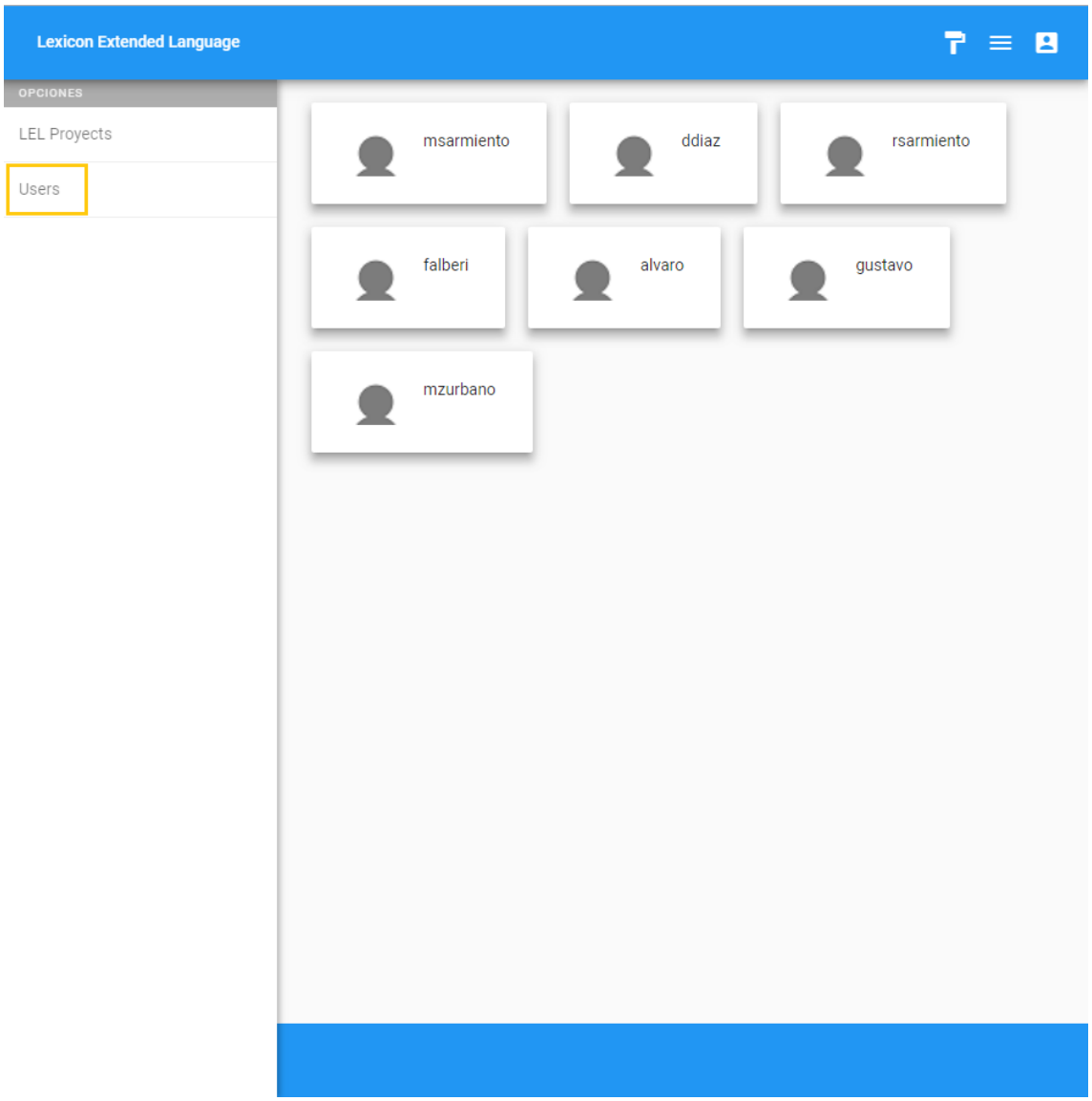

Figura 14 – Usuarios

3.4.6 Símbolos

En la Figura 15 se muestra un ejemplo del listado de símbolos de un proyecto LEL.

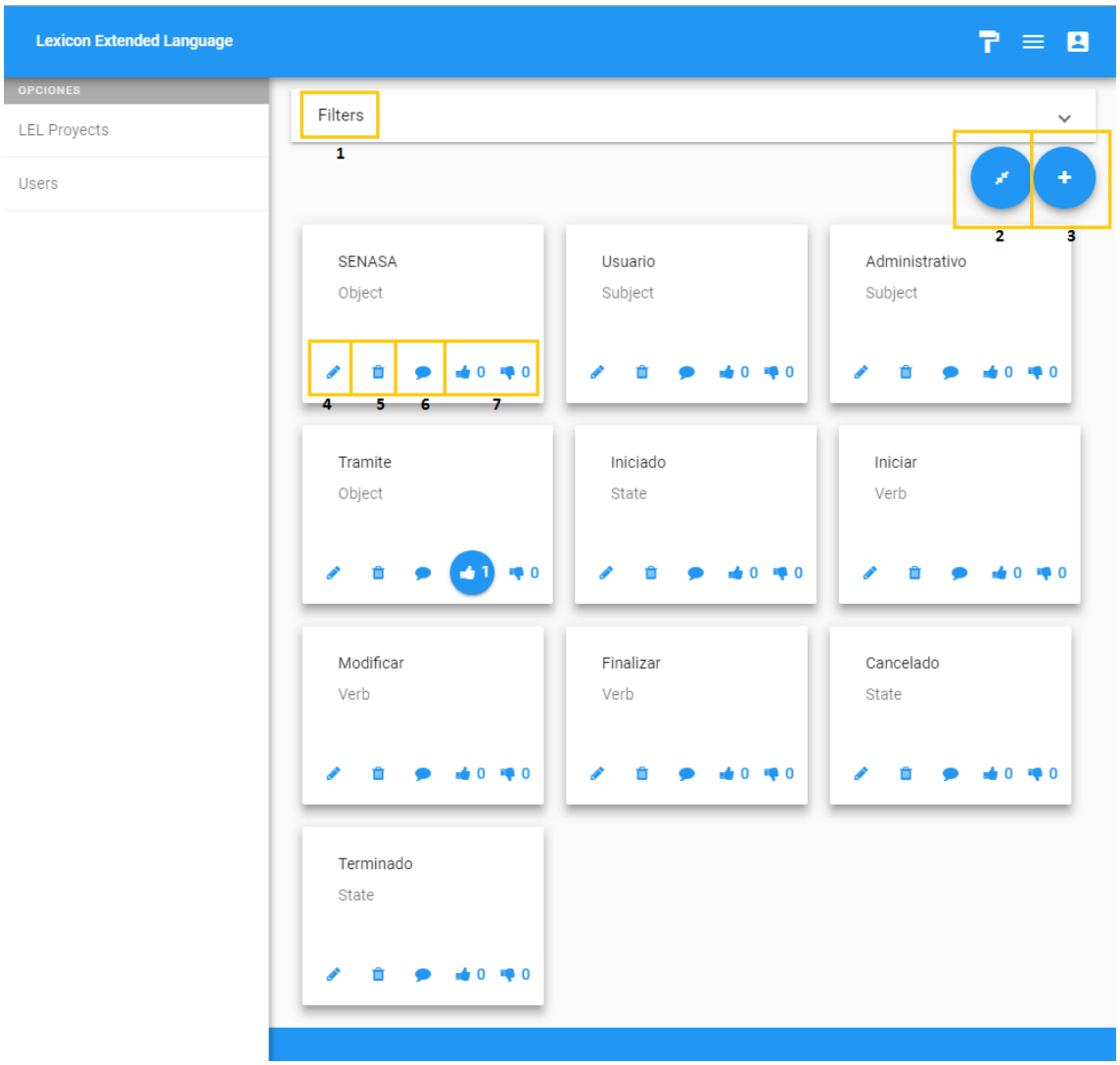

Figura 15 – Símbolos

Si se hace click en “Filters” **(1)** se desplegará un panel de filtros para buscar entre los símbolos. Si se hace click en “Merge two symbols” **(2)** se abrirá una nueva ventana para poder realizar el merge de dos símbolos. Si se hace click en “Add symbol” **(3)** se abrirá una pantalla para poder agregar un símbolo. Si se hace click en “Modificar” **(4)** se abrirá una pantalla para poder realizar modificaciones sobre el símbolo. Si se hace click en “Borrar” **(5)** se eliminará el símbolo. Si se hace click en “Comments” **(6)** se abrirá una ventana donde se pueden ver los comentarios realizados por los usuarios y además agregar/responder comentarios. Si se hace click en “Like” o “Dislike” **(7)** aumenta el contador de likes y/o dislikes del símbolo.

3.4.6.1 Filtros

Es posible filtrar símbolos por nombre y/o categoría. En la Figura 16 se muestra un ejemplo:

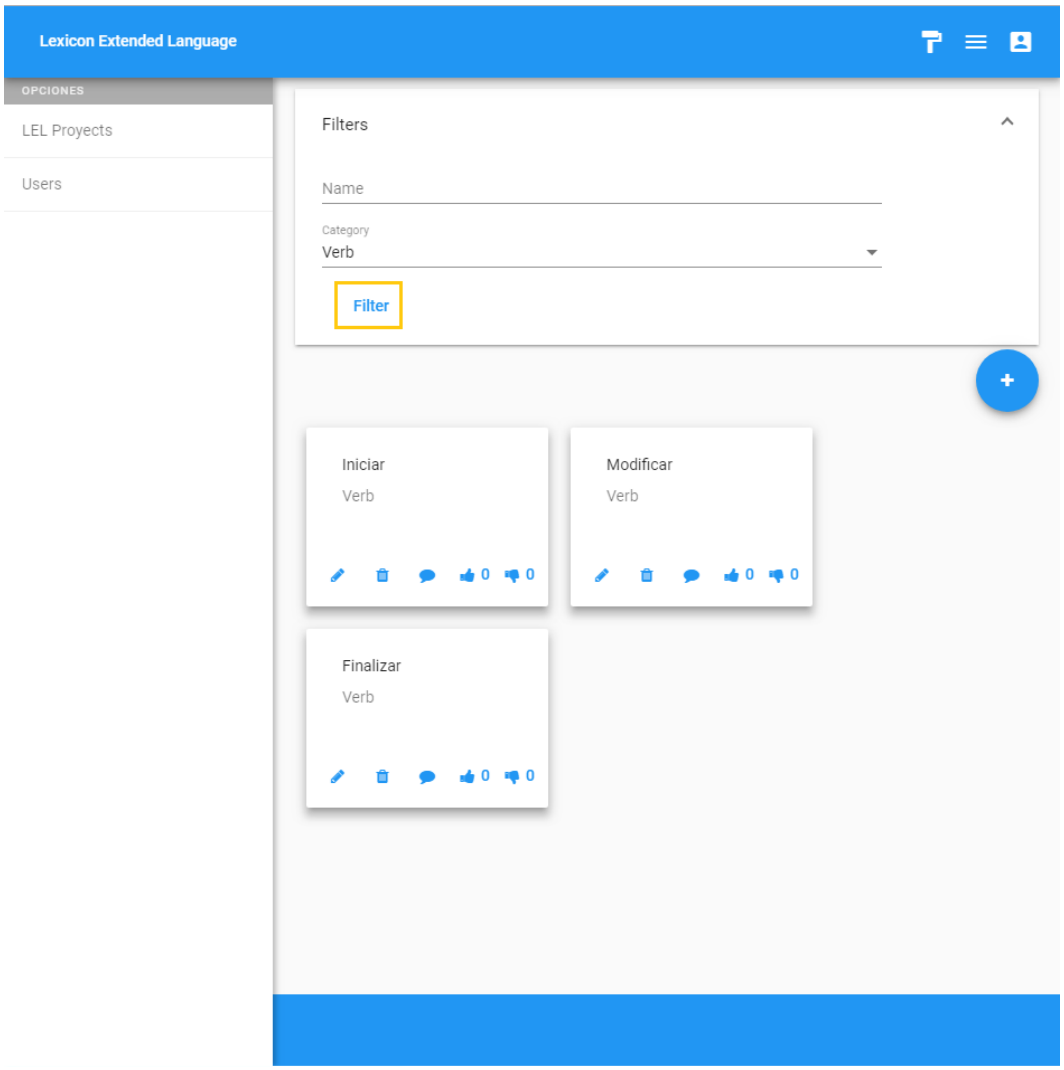

Figura 16 – Filtros

Al hacer click en “Filter” se filtrarán los símbolos según lo que se haya ingresado en los campos “Name” y “Category”.

3.4.6.2 Merge

Es posible realizar el merge entre dos símbolos. Se ingresa el nombre que llevará el símbolo nuevo resultante del merge y se seleccionan los símbolos a mergear. Cabe destacar que en primer lugar se elige un símbolo y luego, se mostrarán en el combo los símbolos disponibles de la misma categoría. La figura 17 muestra un ejemplo:

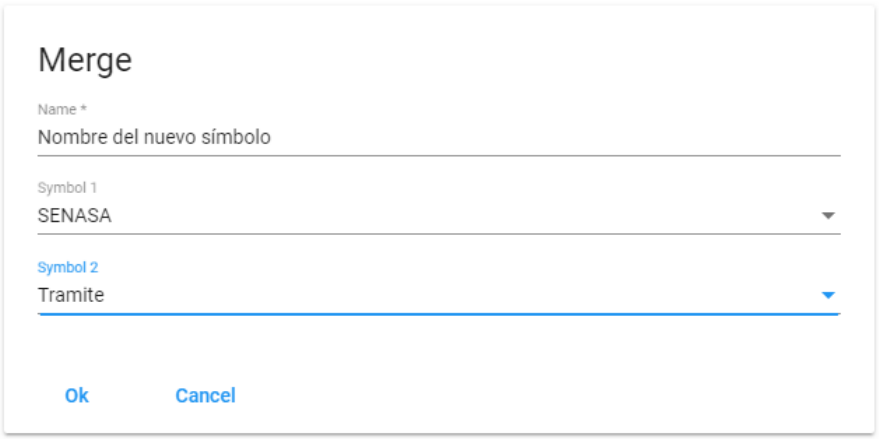

Figura 17 – Merge

Si se hace click en “Ok” se realizará el merge entre los símbolos. Si se hace click en “Cancel” se cancela la operación.

3.4.6.3 Agregar símbolo

En esta pantalla se podrán cargar los datos del nuevo símbolo. Se puede ingresar nombre, categoría, sinónimos, noción e impactos. La Figura 18 muestra la pantalla para agregar símbolo, a la derecha se puede ver el mensaje “No related symbol loaded” ya que aún no se han cargado impactos.

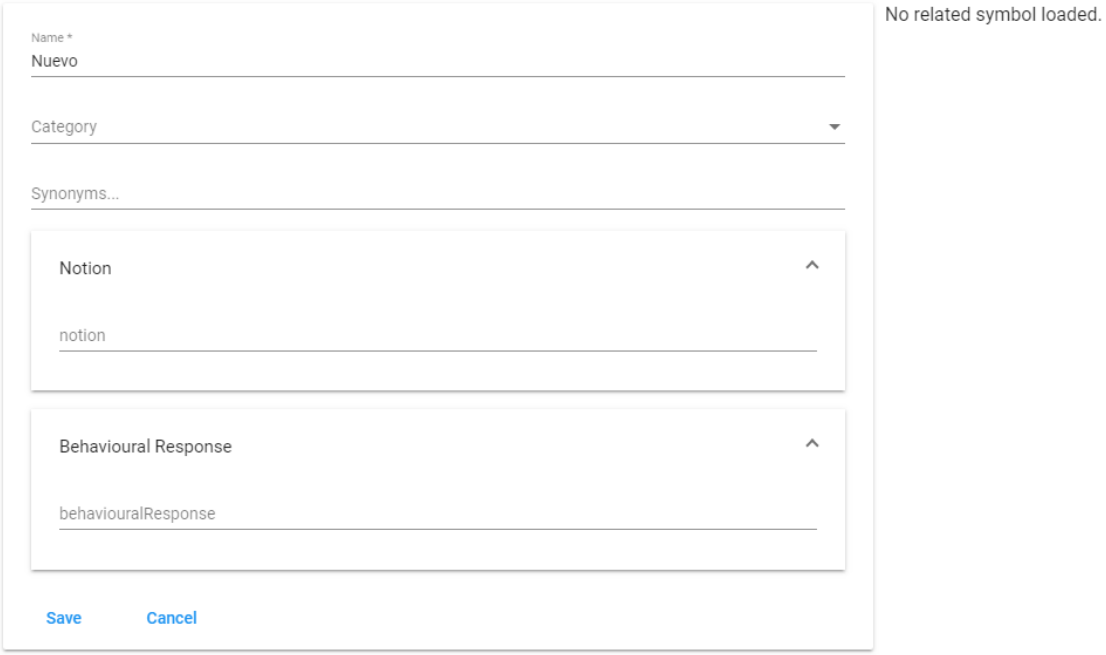

Figura 18 – Agregar símbolo

Si se hace click en “Save” se guardará el símbolo. Si se hace click en “Cancel” el símbolo no se guardará.

3.4.6.4 Eliminar símbolo

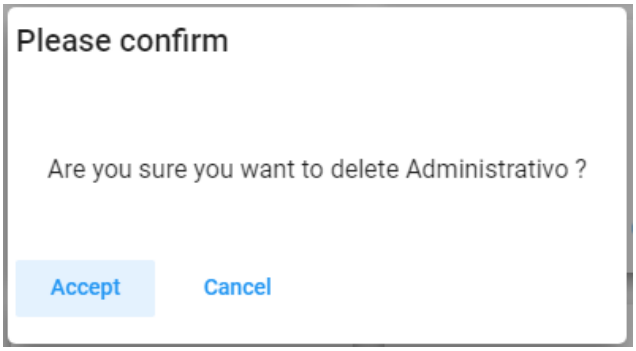

Figura 19- Eliminar símbolo

Si se hace click en “Accept” se eliminará el símbolo. Si se hace click en “Cancel” se cancela la acción de eliminar.

3.4.6.5 Modificar símbolo

La Figura 20 muestra, a modo de ejemplo, el símbolo “Trámite”:

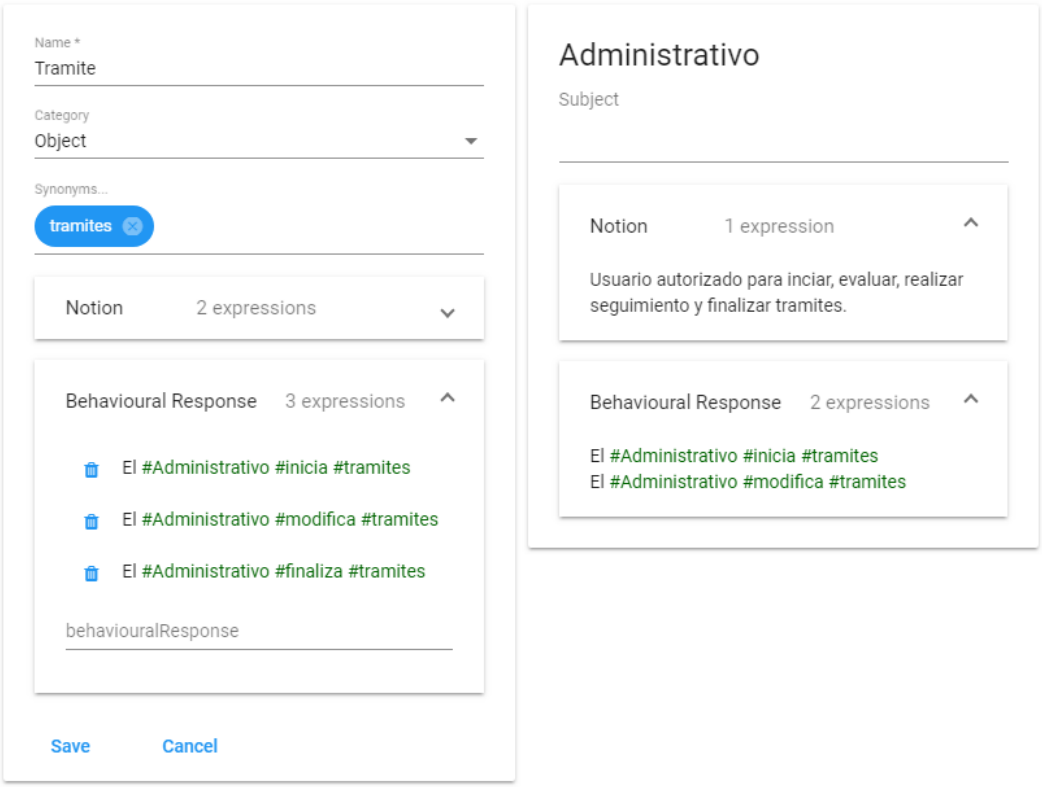

Figura 20 – Modificación de un símbolo

Se pueden editar todos los datos del símbolo. Con la notación “#” + “nombre_del_simbolo” se pueden agregar los símbolos relacionados (se puede

agregar el nombre del símbolo o algún sinónimo). A la derecha de la Figura se puede ver el símbolo relacionado “Administrativo”. Se puede visualizar cualquier símbolo relacionado haciendo click en el nombre del mismo. Los datos del símbolo relacionado no se pueden modificar.

3.4.6.6 Comentarios

Es posible que los colaboradores de un mismo proyecto dejen de comentarios en los símbolos. Sólo el autor de un comentario puede eliminarlo.

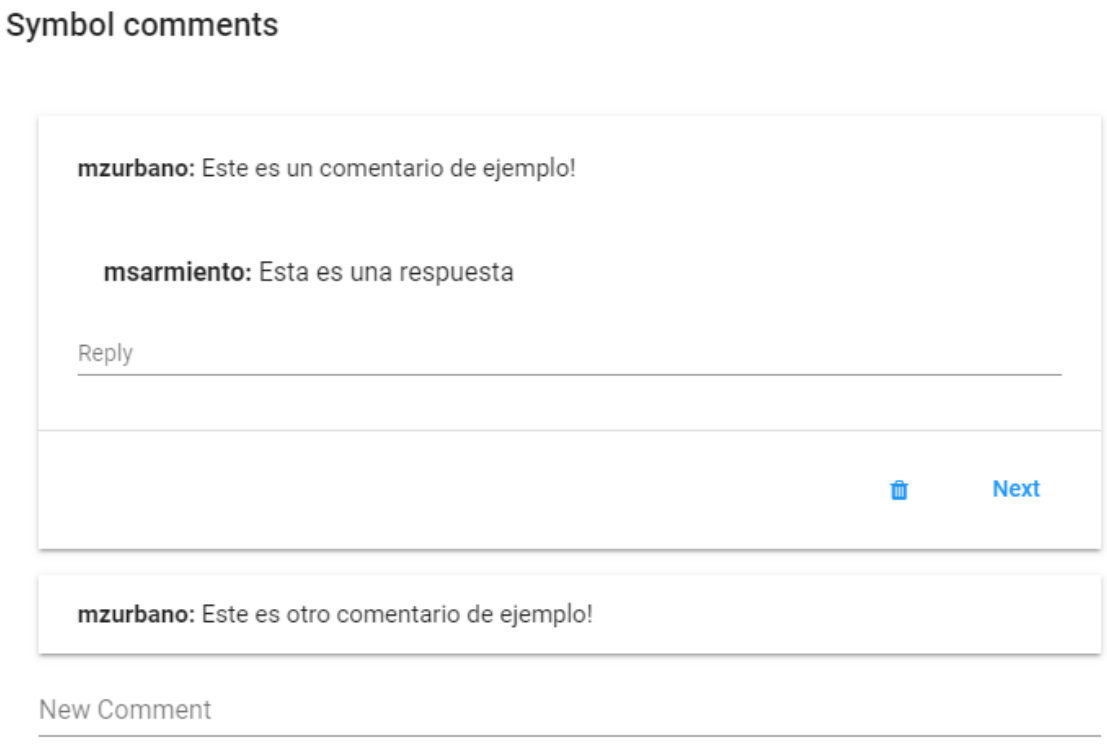

Figura 21 – Comentarios

3.4.7 Datos del equipo

En la Figura 22 se muestra un ejemplo de un listado de usuarios que participan en la construcción del LEL, en caso de ser administrador se tendrá la opción de “invitar” a otros usuarios.

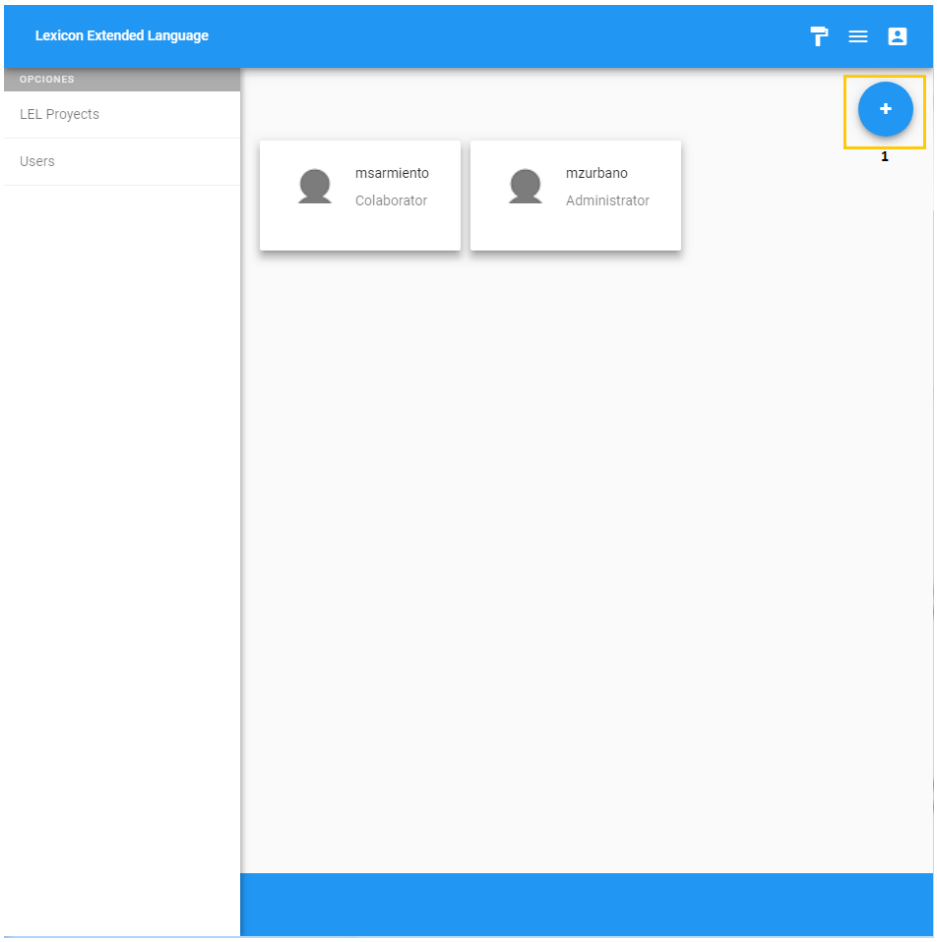

Figura 22 – Datos del equipo

Si se hace click en **(1)** “Invite” se podrá acceder a la pantalla para invitar a otros usuarios a participar de un proyecto LEL, como muestra la Figura 23:

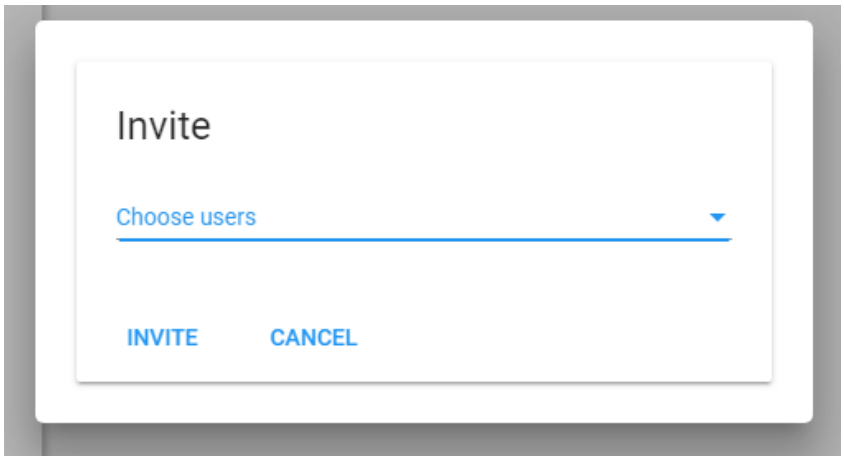

Figura 23 – Invitar

Si se hace click en “Invite” se agregarán como colaboradores los usuarios que hayan seleccionado en el combo de selección múltiple. Si se hace click en “Cancel” se cancela la operación.

CAPITULO 4 - PRUEBA DE USABILIDAD

4.1 Introducción

Se decidió llevar a cabo una prueba de usabilidad de la aplicación LELTool. Para ello se utilizó la Escala de Usabilidad del Sistema, conocido como SUS, por sus siglas en inglés. La escala de usabilidad del sistema proporciona una herramienta confiable y rápida para medir la usabilidad. Fue creada originalmente por John Brooke en 1986 [64], permite evaluar una amplia variedad de productos y servicios, incluidos hardware, software, dispositivos móviles, sitios web y aplicaciones. Se acepta comúnmente que el SUS tiene ciertos atributos que lo convierten en una buena opción para fines de evaluación general.

4.2 Escala de Usabilidad del Sistema

La escala de usabilidad del sistema consiste en un cuestionario de 10 items con cinco opciones de respuesta para los encuestados desde “fuertemente en desacuerdo” a “fuertemente de acuerdo” [65].

Los items son los siguientes:

1. Pienso que me gustaría usar este producto frecuentemente.

2. Encontré el producto innecesariamente complejo.

3. Pienso que el producto es fácil de usar.

4. Pienso que necesitaría soporte de una persona técnica para que sea posible
que utilice el producto.

5. Encontré que varias funciones en el producto estaban muy bien integradas.

6. Pienso que había mucha inconsistencia en el producto.

7. Imagino que la mayoría de las personas aprenderían a usar este producto
muy rápidamente.

8. Encontré el producto muy incómodo de usar.

9. Me sentí muy seguro usando el producto.

10. Necesito aprender muchas cosas antes de que pudiera seguir utilizando el producto.

Cada uno de los encuestados debe responder cada uno de los items. A continuación, se muestra el formato de respuesta del SUS:

Fuertemente en desacuerdo 1	**2**	**3**	**4**	**Fuertemente de acuerdo 5**

Figura 24 – Formato de respuesta del SUS

Por cada item de los mostrados anteriormente, el encuestado marca con una cruz uno de los casilleros indicando su respuesta.

Finalmente se calcula la puntuación del SUS de la siguiente manera:

- Para los items impares: a la respuesta del encuestado se le resta 1 (uno).
- Para los items pares: a 5 (cinco) se le resta la respuesta del encuestado. Esto escala todos los valores de 0 a 4 (siendo cuatro la respuesta más positiva)
- Se suma las respuestas de cada usuario y se multiplica ese total por 2,5. Esto convierte el rango de valores posibles de 0 a 100 en lugar de 0 a 40.

El rango recomendado para la medición de usabilidad según las clasificaciones de la escala de usabilidad del sistema [66], es el siguiente:

- 0 - 64: No aceptable
- 65 - 84: Aceptable
- 85 - 100: Exelente

El principal valor del SUS es que proporciona una puntuación de referencia única para la visión de los participantes de la usabilidad del producto.

4.3 Preparación

Se tomó parte de un LEL confeccionado previamente en un documento de texto para que sea utilizado en la prueba. La idea fue que los participantes tuvieran una guía de qué cargar en la aplicación LELTool manteniendo una coherencia y un sentido en la información cargada en la aplicación.
En el Anexo I se muestra el fragmento del LEL utilizado para la prueba.

Por último, se imprimieron copias en papel del cuestionario para que cada participante pueda responder en forma individual:

Por favor tilde el casillero que refleje su respuesta inmediata a cada oración.
No piense mucho sobre cada oración. Asegúrese de responder cada oración.
Si no sabe cómo responder, simplemente tilde el casillero 3.

	Fuertemente en desacuerdo 1	2	3	4	Fuertemente de acuerdo 5
Q1. Pienso que me gustaría usar este producto frecuentemente.					
Q2. Encontré el producto innecesariamente complejo.					
Q3. Pienso que el producto es fácil de usar.					
Q4. Pienso que necesitaría soporte de una persona técnica para que sea posible que utilice el producto.					
Q5. Encontré que varias funciones en el producto estaban muy bien integradas.					
Q6. Pienso que había mucha inconsistencia en el producto.					
Q7. Imagino que la mayoría de las personas aprenderían a usar este producto muy rápidamente.					
Q8. Encontré el producto muy incómodo de usar.					
Q9. Me sentí muy seguro usando el producto.					
Q10. Necesito aprender muchas cosas antes de que pudiera seguir utilizando el producto.					

Figura 25 – Cuestionario SUS

4.4 Desarrollo del cuestionario

Se reunió a un grupo de 5 personas, todas de diferentes profesiones, para que carguen en la aplicación LELTool el LEL mostrado en la sección anterior.
En primer lugar, se hizo una introducción acerca del Léxico Extendido del Lenguaje y se explicó brevemente cómo se define un símbolo. También se explicaron las funcionalidades que provee la herramienta además de la carga de símbolos de un LEL.
Luego, se asignaron los distintos símbolos que cada participante debía cargar en la aplicación, distribuyéndolos en forma “pareja” entre los mismos. Finalmente, se realizaron dos rondas de trabajo, en la primera, tomando turnos de a uno. En la segunda ronda, se permitió que cada participante en forma

"libre", es decir sin indicaciones, revise y utilice el resto de funcionalidades disponibles.

4.5 Análisis de resultados

A continuación, se muestran las respuestas y resultados obtenidos por cada participante:

Participante uno (P1)	Fuertemente en desacuerdo 1	2	3	4	Fuertemente de acuerdo 5
Q1. Pienso que me gustaría usar este producto frecuentemente.			**x**		
Q2. Encontré el producto innecesariamente complejo.	**x**				
Q3. Pienso que el producto es fácil de usar.			**x**		
Q4. Pienso que necesitaría soporte de una persona técnica para que sea posible que utilice el producto.		**x**			
Q5. Encontré que varias funciones en el producto estaban muy bien integradas.			**x**		
Q6. Pienso que había mucha inconsistencia en el producto.	**x**				
Q7. Imagino que la mayoría de las personas aprenderían a usar este producto muy rápidamente.				**x**	
Q8. Encontré el producto muy incómodo de usar.		**x**			
Q9. Me sentí muy seguro usando el producto.			**x**		
Q10. Necesito aprender muchas cosas antes de que pudiera seguir utilizando el producto.	**x**				

Tabla 3: Respuestas participante uno

Total participante uno: 29 x 2.5 = 72.5

Participante dos (P2)	Fuertemente en desacuerdo 1	2	3	4	Fuertemente de acuerdo 5
Q1. Pienso que me gustaría usar este producto frecuentemente.				**X**	
Q2. Encontré el producto innecesariamente complejo.	**X**				
Q3. Pienso que el producto es fácil de usar.				**X**	
Q4. Pienso que necesitaría soporte de una persona técnica para que sea posible que utilice el producto.			**X**		
Q5. Encontré que varias funciones en el producto estaban muy bien integradas.			**X**		
Q6. Pienso que había mucha inconsistencia en el producto.	**X**				
Q7. Imagino que la mayoría de las personas aprenderían a usar este producto muy rápidamente.					**X**
Q8. Encontré el producto muy incómodo de usar.	**X**				
Q9. Me sentí muy seguro usando el producto.		**X**			
Q10. Necesito aprender muchas cosas antes de que pudiera seguir utilizando el producto.		**X**			

Tabla 4: Respuestas participante dos

Total participante dos: 30 x 2.5 = 75

Participante tres (P3)	Fuertemente en desacuerdo 1	2	3	4	Fuertemente de acuerdo 5
Q1. Pienso que me gustaría usar este producto frecuentemente.					**x**
Q2. Encontré el producto innecesariamente complejo.	**x**				
Q3. Pienso que el producto es fácil de usar.					**x**
Q4. Pienso que necesitaría soporte de una persona técnica para que sea posible que utilice el producto.	**x**				
Q5. Encontré que varias funciones en el producto estaban muy bien integradas.				**x**	
Q6. Pienso que había mucha inconsistencia en el producto.	**x**				
Q7. Imagino que la mayoría de las personas aprenderían a usar este producto muy rápidamente.					**x**
Q8. Encontré el producto muy incómodo de usar.		**x**			
Q9. Me sentí muy seguro usando el producto.					**x**
Q10. Necesito aprender muchas cosas antes de que pudiera seguir utilizando el producto.	**x**				

Tabla 5: Respuestas participante tres

Total participante tres: 38 x 2.5 = 95

Participante cuatro (P4)	Fuertemente en desacuerdo 1	2	3	4	Fuertemente de acuerdo 5
Q1. Pienso que me gustaría usar este producto frecuentemente.		**x**			
Q2. Encontré el producto innecesariamente complejo.	**x**				
Q3. Pienso que el producto es fácil de usar.				**x**	
Q4. Pienso que necesitaría soporte de una persona técnica para que sea posible que utilice el producto.	**x**				
Q5. Encontré que varias funciones en el producto estaban muy bien integradas.		**x**			
Q6. Pienso que había mucha inconsistencia en el producto.			**x**		
Q7. Imagino que la mayoría de las personas aprenderían a usar este producto muy rápidamente.					**x**
Q8. Encontré el producto muy incómodo de usar.			**x**		
Q9. Me sentí muy seguro usando el producto.				**x**	
Q10. Necesito aprender muchas cosas antes de que pudiera seguir utilizando el producto.	**x**				

Tabla 6: Respuestas participante cuatro

Total participante cuatro: 28 x 2.5 = 70

Participante cinco (P5)	Fuertemente en desacuerdo 1	2	3	4	Fuertemente de acuerdo 5
Q1. Pienso que me gustaría usar este producto frecuentemente.					**x**
Q2. Encontré el producto innecesariamente complejo.	**x**				
Q3. Pienso que el producto es fácil de usar.				**x**	
Q4. Pienso que necesitaría soporte de una persona técnica para que sea posible que utilice el producto.			**x**		
Q5. Encontré que varias funciones en el producto estaban muy bien integradas.					**x**
Q6. Pienso que había mucha inconsistencia en el producto.	**x**				
Q7. Imagino que la mayoría de las personas aprenderían a usar este producto muy rápidamente.					**x**
Q8. Encontré el producto muy incómodo de usar.	**x**				
Q9. Me sentí muy seguro usando el producto.			**x**		
Q10. Necesito aprender muchas cosas antes de que pudiera seguir utilizando el producto.	**x**				

Tabla 7: Respuestas participante cinco

Total participante cinco: 35 x 2.5 = 87.5

La siguiente tabla muestra un resumen de los resultados obtenidos:

P1	**P2**	**P3**	**P4**	**P5**
72.5	75	95	70	87.5

Tabla 8: Resumen de resultados obtenidos

Podemos ver que todos los resultados superaron los 65 puntos. Además, si realizamos un cálculo del promedio obtenemos como resultado: 72.5 + 75 + 95

+ 70 +87.5 = 400/5 = 80 puntos. Por lo tanto, podemos concluir que la escala de usabilidad del sistema es “Aceptable”.

4.6 Conclusiones

Haciendo un análisis de las respuestas obtenidas por cada item, los resultados obtenidos fueron:

- Q1: Todas las respuestas oscilaron entre los valores 4 y 5 respecto a si les gustaría utilizar el producto frecuentemente.
- Q2: Obtuvimos un resultado unánime con valor 1 respecto a si encontraban el producto innecesariamente complejo.
- Q3: La mayoría indicó entre los valores 4 y 5 que creen que el producto es fácil de usar.
- Q4: La mayoría indicó con valor 3 respecto a si pensaban que necesitarían soporte de una persona técnica para poder utilizar el producto.
- Q5: Todas las respuestas oscilaron entre los valores 4 y 5 respecto a si encontraron varias funciones en el producto muy bien integradas.
- Q6: Obtuvimos un resultado unánime con valor 1 respecto a si pensaban que había mucha inconsistencia en el producto.
- Q7: Obtuvimos un resultado unánime con valor 5 respecto a si imaginaban que la mayoría de las personas aprenderían a usar este producto muy fácilmente.
- Q8: La mayoría indicó con valor 2 respecto a si encontraban el producto muy incómodo de usar.
- Q9: Todas las respuestas oscilaron entre los valores 2 y 4 respecto a si se sintieron seguros usando el producto.
- Q10: Obtuvimos un resultado unánime con valor 1 respecto a si creen que necesitan aprender muchas cosas antes de poder seguir utilizando el producto.

Finalmente podemos concluir que se podrían realizar algunos ajustes para que sea un poco más ágil la carga de los datos de símbolos, que entendemos fue uno de los motivos principales por el que no obtuvimos un resultado unánime de valor 1, en cuanto a si encontraban el producto muy incómodo de usar (Q8). Y también, se podrían agregar más avisos que indiquen si se realizó o no exitosamente una operación, para evitar cierta incertidumbre ante algunas acciones.

CAPITULO 5 - CONCLUSIONES Y TRABAJOS FUTUROS

5.1 Conclusiones

Es muy importante iniciar el desarrollo de software con los requerimientos necesarios lo más completos y correctos que sea posible. Comprender el contexto de un sistema de software durante la especificación de requerimientos es una tarea difícil. Es dificultoso elicitar y escribir el conjunto de requerimientos iniciales y a la vez cumplir con atributos tales como la correctitud y la completitud. Es complejo realizar la especificación de requerimientos cuando hay muchos stakeholders involucrados.

Todo aquello que no se detecte, o resulte mal entendido en la etapa inicial provocará un gran impacto negativo en los requisitos, propagando esta corriente negativa a lo largo de todo el proceso de desarrollo e incrementando su perjuicio cuanto más tardía sea su detección.

Definir el lenguaje de dominio antes de especificar los requerimientos es una manera de hacer frente a este problema. Mientras que nos apoyamos en la colaboración para fomentar la cooperación entre los stakeholders, y así puedan explorar sus diferencias en forma constructiva y buscar soluciones que van más allá de su propia visión limitada.

La aplicación desarrollada se basa principalmente en la estructura que tiene el LEL, se detallan a continuación sus características:

- **Construcción de proyectos LEL:** un proyecto representa un LEL que en sí mismo es un glosario con estructura más amplia que lo habitual, es una estrategia para capturar el lenguaje del contexto de la aplicación utilizando el lenguaje natural. El principal objetivo de un proyecto LEL es representar el vocabulario del problema. La aplicación permite crear uno o más LEL, indicando el nombre que identifica al mismo y donde el usuario que lo crea será el autor y administrador del mismo.
- **Creación, modificación y eliminación de símbolos:** para generar el LEL se registran símbolos (palabras o frases) peculiares o relevantes del dominio. Cada entrada del léxico se identifica con un nombre (o más de uno en caso de querer agregar sinónimos) y tiene dos tipos de descripciones. Una llamada Noción que describe la denotación del símbolo y la otra Impacto que describe la connotación del mismo. Las entradas se clasifican en cuatro tipos de acuerdo a su uso general en el universo del discurso. Estos tipos son: Sujeto, Objeto, Verbo y Estado. Cualquier usuario puede crear símbolos y modificarlos. La eliminación de un símbolo sólo puede realizarla el autor de dicho símbolo o el administrador del proyecto. Una característica muy importante de la aplicación es que se generan enlaces entre símbolos de manera tal que es muy fácil y práctico ver el/los símbolo/s relacionado. Esto mejora enormemente la lectura y comprensión de un símbolo.

- **Trabajo colaborativo:** en el proceso de definición de requisitos se genera un compromiso entre los ingenieros y los clientes/usuarios, ya que ambos participan en la generación y definición de los requisitos del sistema, por lo tanto, el uso de un glosario con términos comunes a todos los participantes facilita la comunicación y la compresión. Cada proyecto LEL se va refinando en sucesivas etapas del desarrollo, el trabajo colaborativo aquí es donde toma relevancia. Es posible que los usuarios dejen su opinión de un símbolo a través de comentarios, los cuales además, pueden ser respondidos generando un ida y vuelta entre los usuarios, lo que resulta muy enriquecedor para la construcción, mantenimiento y mejora de un LEL. Además, los usuarios pueden indicar like/dislike de símbolos y/o expresiones que en conjunto con los comentarios hacen posible un verdadero trabajo colaborativo.
- **Tareas de refactoring:** como hemos mencionado un proyecto LEL se va refinando en sucesivas etapas de desarrollo, por lo tanto, es muy factible que se necesite realizar arreglos o mejoras, para estos casos se podrán realizar tareas de refactoring como unificar dos símbolos (merge) en forma simple y rápida, evitando además, posibles errores como el borrado de símbolos que son importantes o la omisión de información en el nuevo símbolo que se crea a partir del merge.

5.2 Trabajos futuros

Se proponen mejoras en algunos aspectos de la aplicación para que brinde el máximo beneficio y se definen algunas funcionalidades que podrían incorporarse tales como:

- Procesamiento de lenguaje natural: el Procesamiento del Lenguaje Natural es el campo de conocimiento de la Inteligencia Artificial que se ocupa de investigar la manera de que una máquina comprenda lo que expresa una persona mediante el uso de lenguas naturales, como el español, el inglés o el chino. Creemos que incorporar funcionalidad de este tipo podría resultar muy útil en las expresiones que se escriben en Nociones e Impactos. Por ejemplo, en el caso de los verbos, el símbolo se define con el infinitivo del mismo, pero rara vez se utiliza en las expresiones, en general se utilizan las conjugaciones del mismo.
- Soporte para análisis semántico: uno de los componentes del procesamiento del lenguaje natural es el análisis semántico, el cual proporciona la interpretación de las oraciones, una vez eliminadas las ambigüedades morfosintácticas. Representaría una mejora interesante el hecho de poder agregar expresiones y que el análisis semántico se “encargue” de identificar símbolos.

Anexo I

Objeto: SENASA
Noción
Organismo responsable de garantizar y certificar la sanidad y calidad de la producción agropecuaria, pesquera y forestal.
Impactos

Sujeto: usuario
Noción
Persona que trabaja para el SENASA.
El usuario puede desempeñar el rol de administrativo.
Impactos
El usuario inicia trámites.

Sujeto: administrativo
Noción
Usuario autorizado para iniciar, evaluar, realizar seguimiento y finalizar trámites.
Impactos
El administrativo inicia trámites.
El administrativo modifica trámites.
El administrativo finaliza trámites.

Objeto: tramite
El tramite tiene:
número de trámite
fecha de inicio
El trámite tiene un estado asociado.
Impactos
El administrativo inicia trámites.
El administrativo modifica trámites.
El administrativo finaliza trámites.

Objeto: número de trámite
Noción
Número que identifica unívocamente un trámite en particular.
Impactos
Al iniciarse un trámite se le asigna un número.

Objeto: fecha de inicio
Noción
Dia, mes y año en que un usuario inicia un trámite.
Impactos
Siempre se registra la fecha en la que se inicia un trámite.

Estado: Iniciado
Noción
Situación donde el trámite ha sido creado y está disponible para modificaciones.
Impactos
El administrativo crea una evaluación para un trámite y es el propietario de un trámite asignado.

Estado: Asignado
Noción
Situación en donde el trámite está siendo evaluado por algún usuario del SENASA.
Impactos
El administrativo aprueba una evaluación y es el propietario de un trámite en curso.
El administrativo desaprueba una evaluación y es el propietario de un trámite cancelado.

Estado: En curso
Noción
Situación en la que el trámite ha superado la evaluación y puede continuar.
Impactos
El administrativo modifica el estado de un trámite y es el propietario de un trámite en estado terminado.

Estado: Cancelado (trámite)
Noción
Situación en la que el trámite no ha superado la evaluación y no se puede continuar.
Impactos

Estado: Terminado
Noción
Situación en la que el trámite ha sido finalizado. Ya no se pueden realizar modificaciones.
Impactos

Verbo: iniciar
Noción
Acción de ingresar a través del sistema un trámite de algún tipo.
Impactos
El administrativo selecciona el tipo de trámite que se va a iniciar.

Objeto: evaluación
Nocion
Una evaluación posee:
Trámite (que se está evaluando).
Area evaluadora.
Usuario (asignado a la evaluación).
Estado de la evaluación.
Fecha asignación.
Fecha aprobación.
Impactos
El administrativo crea una evaluación para un trámite.
El administrativo aprueba una evaluación.
El administrativo desaprueba una evaluación.

Verbo: crear evaluación
Noción
Acción de generar para un trámite específico una o varias evaluaciones que un usuario hace de los datos ingresados en dicho trámite.
Impactos
El administrativo selecciona el área técnica que hará la evaluación.
El administrativo selecciona el usuario a cargo de la evaluación (puede ser el u otro usuario con rol administrativo).
El sistema genera una evaluación para el trámite en cuestión asociada al usuario que se seleccionó.

Verbo: aprobar evaluación
Noción
Acción de verificar los datos del trámite y observaciones realizadas y concluir que la información es suficiente y correcta.
Impactos
El administrativo a través de la opción de modificación de una evaluación, aprueba la misma.
El tramite pasa a estado “En curso”.

Verbo: desaprobar evaluación
Noción
Acción de verificar los datos del trámite y observaciones realizadas y concluir que la información es incorrecta.
Impactos
El administrativo a través de la opción de modificación de una evaluación, desaprueba la misma.
El trámite pasa a estado “Cancelado”.

Verbo: modificar
Noción
Acción de ingresar/modificar datos correspondientes según el tipo de trámite.
Acción de modificar el estado de un trámite.
Impactos
El administrativo crea una evaluación para el trámite.
Si el administrativo desaprueba la evaluación, el trámite se cancela y no se puede continuar.
Si el administrativo aprueba la evaluación el trámite pasa a estado “En curso” y se puede continuar.

Verbo: finalizar
Noción
Acción de terminar un trámite.
Impactos
El administrativo ingresa a la opción de finalización de trámite.

BIBLIOGRAFIA

[1] Leite, J.C.S.P., Franco, A.P.M. : A strategy for conceptual model acquisition. RE 1993: pp 243-246

[2] Antonelli, L., Rossi G., Oliveros A.: A Collaborative Approach to Capture the Domain Language. WER 2015

[3] Antonelli, L., Rossi G., Leite, J.C.S.P., Oliveros, O.: Buenas prácticas en la especificación del dominio de una aplicación. WER 2013

[4] IEEE Standard Glossary of Software Engineering Terminology, IEEE std 610.12-1990, 1990. ISBN 155937067X.

[5] Pierre Bourque and Robert Dupuis, ed. Guide to the Software Engineering Body of Knowledge - 2004 Version. IEEE Computer Society. pp. 1-1. ISBN 0-7695-2330-7.

[6] Leveson, N. G., Turner, C. S. An Investigation of the Therac-25 Accidents. Julio 1993

[7] ISO/IEC 12207, https://www.iso.org/home.html

[8] Pressman, R.S. Pressman and Associates: Software Engineering: A Practitioner's Approach, 6/e ISBN: 0072853182. Copyright year: 2005

[9] IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications, Software Engineering Standards Committee of the IEEE Computer Society ISBN 0-7381-0332-2

[10] Jacobson, I., P. Jonsson, M. Christerson and Overgaard G., Ingeniería de Software Orientada a Objetos - Un acercamiento a través de los casos de uso. Addison Wesley Longman, Upper Saddle River, N.J., 1992.

[11] Jacobson, I., Spence I., Bittner, K., Casos de Uso 2.0 La guía para ser exitoso con los Casos de Uso 2005 - 2013

[12] Cockburn, A , Writing Effective Use Cases, Addison-Wesley. 2011

[13] Cohn, M., "User Stories Applied", Addison Wesley, ISBN 0-321-20568-5. 2004

[14] Sommerville, I. Software engineering. Boston: Pearson. 2011

[15] Hadad, G., Kaplan, G., Oliveros, A., Leite, J.C.S.P. : Integración de Escenarios con el Léxico Extendido del Lenguaje en la elicitación de requerimientos: Aplicación a un caso real 1997

[16] Leite, J.C.S.P., Franco, A.P.M.: A Strategy for Conceptual Model Acquisition, In Proceedings of the First IEEE International Symposium on Requirements Engineering, San Diego, California, IEEE Computer Society Press, 1993, pp 243-246.

[17] Kaplan, G. N., Hadad, G. D.S., Doorn, J. H., Leite, J.C.S.P: Inspección del léxico extendido del lenguaje. WER 2000

[18] Freeman E. R., Harrison J. S.: Stakeholder Theory: The State of The Art. University Priting House, Cambridge CB2 8BS, United Kingdom. 2010

[19] Firesmith, D. G.: “Common Requirements Problems, Their Negative Consequences, and Industry Best Practices to Help Solve Them”, in Journal of Object Technology, vol. 6, no. 1, January-February 2007, pp. 17-33

[20] Honderich, T. The Oxford Companion to Philosophy (2 ed.) Mar 10, 2005.

[21] Leite J.C.S.P., Franco A.P., O uso de Hipertexto na Elicitaçao de Linguagens da Aplicaçao". IV Simpósio Brasileiro de engenharia de software Sau Pedro, 1990

[22] Leite, J.C.S.P., Application Languages: A Product of Requirements Analysis, Departamento de Informática, PUC-/RJ, January 1989

[23] Leite, J.C.S.P., “Eliciting Requirements Using a Natural Language Based Approach: The Case of the Meeting Scheduler Problem”, March 1993

[24] Hadad,G.,Kaplan, G., Maiorana, V., Balaguer, F., Oliveros, A., Leite, J.C.S.P., Rossi, G. Informe Técnico: “Léxico Extendido del Lenguaje y Escenarios del Sistema Nacional para la Obtención de Pasaportes”. Proyecto de Investigación, Departamento de Investigación, Universidad de Belgrano, Buenos Aires, 1996.

[25] Marinez-Moyano, I. J. Exploring the Dynamics of Collaboration in Interorganizational Settings, Ch. 4, p. 83, in Schuman (Editor). Creating a Culture of Collaboration. Jossey-bass, 2006. ISBN 0-7879-8116-8.

[26] Spence, M. U. "Graphic Design: Collaborative Processes = Understanding Self and Others." (lecture) Art 325: Collaborative Processes. Fairbanks Hall, Oregon State University, Corvallis, Oregon. 13 April 2006.

[27] Hank, R. Collaborative leadership: developing effective partnerships for communities and schools (2nd ed.). Thousand Oaks, California, 2009. ISBN 978-1299395657. OCLC 842851754.

[28] Lowry, P. B., Curtis, A. M., Lowry, M. R., "A Taxonomy of Collaborative Writing to Improve Empirical Research, Writing Practice, and Tool Development". Journal of Business Communication (JBC), Vol. 41, No. 1, pp. 66-99, 2004.

[29] Dillon A. How Collaborative is Collaborative Writing? An Analysis of the Production of Two Technical Reports., pages 69--86. Springer-Verlag, London, 1993.

[30] Sharple M. Adding a Little Structure to Collaborative Writing. Structure in what way. Springer-Verlag, London, 1992.

[31] Rimmershaw R. Collaborative Writing Practices and Writing Support Technologies, pages 15--28. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1992.

[32] Ede L., Lunsford A. Singular Text/Plural Authors: Perspectives on Collaborative Authoring. Southern Illinois University Press, Carbondale, 1990.

[33] Beck E.D. A Survey of Experiences of Collaborative Writing, pages 87-112. Springer-Verlag, London, 1993.

[34] Kuutti, K., Karsten, E. H., Dourish, P., Fitzpatrick, G., Schmidt, K. ECSCW 2003: proceedings of the Eighth European Conference on Computer Supported Cooperative Work (14–18 September 2003, Helsinki, Finland). London: Kulwer. ISBN 978-1-4020-1573-1; OCLC 52784895

[35] Johnson-Lenz, P. "Rhythms, Boundaries, and Containers". Awakening Technology. 30 April 1990

[36] "Groupware - Communication, Collaboration and Coordination". Lotus Development Corporation. 1995. Archived from the original on July 13, 2011.

[37] Lomas, C., Burke, M., Page, C. L. Collaboration Tools. EDUCAUSE Learning Initiative. 2008

[38] Nunamaker Jr., Jay F., O Briggs, R., Romano Jr., Romano N. C,. Collaboration Systems: Concept, value, and Use. New York: Routledge. 2014 p. 55. ISBN 978-0765638458

[39] Deal, A. Collaboration Tools - A Teaching with Technology White Paper. Carnegie Mellon University. 2009. p. 3.

[40] Derks, D., Bakker, A. B. "The Impact of E-mail Communication on Organizational Life". Cyberpsychology: Journal of Psychosocial Research on Cyberspace. Erasmus University Rotterdam. 2010

[41] Anderson, R. "Collaboration Best Practices – 3 Reasons Why Email Hurts Your Productivity". Atlassian Blog. Atlassian Ltd. September 2012

[42] Prampolini, F. Telco 2015- five telling years, four future scenarios. IBM 2010 p. 35.

[43] 7 things you should know about Instant Messaging. Educause Learning Initiative 2005.

[44] Dyson, D. SIP, Hosted VoIP, and the Future of Voice Communications. Eclipse Telecom. 2015 pp. 2–3.

[45] Lovett, T., O'Neill, E., Irwin, J., Pollington, D. The Calendar as a Sensor: Analysis and Improvement Using Data Fusion with Social Networks and Location. Bath: University of Bath. 2010 pp. 1–3.

[46] Giersch, P., Zignego, V. Time Tracking to Improve Productivity. Cathedral Consulting Group. 2014 pp. 1–3.

[47] Shaun, S. Future of Spreadsheets as Financial Analysis Platforms. Data C Ltd. 2004 pp. 1–5.

[48] The Future of Collaboration Software - A Qualitative Study. Mikogo. 2015.

[49] Jystad, G. Video Collaboration and the Future Workplace (PDF). InFocus. 2014 pp. 1–9.

[50] Bridging The Future: Innovations and Value in Teleconferencing Technology and Applications. Forum Communications International. 2006.

[51] Wahlert, T. "Synchronous or Asynchronous Tools". Green Hills AEA. Green Hills Area Education Agency. 2012

[52] Cheng, B. H. C., Atlee, J. M.: Research directions in requirements engineering, in proceedings of the Conference on The Future of Software Engineering, 2007, pp. 285-303.

53] Northrop, L., Feiler, P., Gabriel, R. P., Goodenough, J., Linger, R., Longstaff, T., Kazman, R., Klein, M., Schmidt, D., Sullivan, K.: Ultra-Large-Scale Systems: The Software Challenge of the Future: Software Engineering Institute, 2006.

[54] Konaté, J., Sahraoui, A. E. K., Kolfschoten, G. L.: Collaborative Requirements Elicitation: A Process-Centred Approach, Group Decision and Negotiation, Springer July 2014, Volume 23, Issue 4, pp 847-877.

[55] 1471-2000 -IEEE Recommended Practice for Architectural Description for Software-Intensive Systems, http://standards.ieee.org/findstds/standard/1471-2000.html.

[56] ASP .NET Core, https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-2.1

[57] Angular 6, https://angular.io/docs

[58] Entity Framework Core, https://docs.microsoft.com/en-us/ef/core/

[59] TypeScript, https://www.typescriptlang.org/index.html

[60] MySQL, https://www.mysql.com/

[61] Visual Studio Code, https://code.visualstudio.com/

[62] MySQL Workbench, https://www.mysql.com/products/workbench/

[63] GitHub, https://github.com/

[64] Brooke J. “SUS: a retrospective”. Journal of Usability Studies. Volume 8 Issue 2, Pages 29-40. February 2013

[65] System Usability Scale, https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html

[66] McLellan S., Muddimer A. and Peres S.C. “The effect of experience on System Usability Scale ratings”. Journal of Usability Studies, 7(2), 56-67, February 2012

Todas las url mencionadas fueron accedidas entre los meses de Febrero y Marzo de 2019.