TECNOLOGÍA EN DESARROLLO DE SOFTWARE

Fundación Universitaria
SAN MATEO

FACULTAD DE INGENIERÍA Y AFINES

TECNOLOGÍA EN DESARROLLO DE SOFTWARE

AUTOMASOFT: DESARROLLO DE SOFTWARE PARA NO DESARROLLADORES

TRABAJO DE GRADO MODALIDAD DE OPCIÓN DE GRADO

JUAN PABLO ÁNGEL PARRA ARÉVALO

DIRECTOR

M.Sc. RICARDO CEBALLOS GARZÓN

BOGOTÁ D.C

2022

NOTA DE SALVEDAD DE RESPONSABILIDAD INSTITUCIONAL

"La Fundación Universitaria San Mateo NO se hace responsable de los conceptos emitidos en el presente documento, el departamento de investigaciones velará por el rigor metodológico de la investigación".

CONTENIDO

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

DEDICATORIA

Esta investigación está dedicada a todas las personas que tienen la motivación de avanzar, pero que no siempre tienen claro un punto de partida. A mis familiares y amigos, que siempre me apoyaron y a todos los docentes que hicieron posible la realización del presente proyecto.

AGRADECIMIENTOS

Quisiera agradecer a los docentes que hicieron posible la realización de este proyecto, incluyendo al Profesor Ricardo Ceballos Garzón, quien realizó una labor de guía al momento de elaborar este proyecto.

También quiero agradecer a las personas y comunidades que sirven a proyectos de software, traductores, encargados del diseño y documentación de los diferentes aplicativos y herramientas que se utilizan para desarrollar software funcional.

RESUMEN

El desarrollo de software es un campo con crecimiento notable, aplicándose a diferentes áreas para la solución de problemáticas presentadas en diferentes contextos, teniendo así una gran importancia en los procesos de quienes utilizan herramientas informáticas en la realización de tareas para reducir el esfuerzo, tiempo y problemas relacionados con el acceso a la información.

Sin embargo, aunque el software actualmente juega un papel fundamental para personas y empresas, el proceso de desarrollo de software estaba restringido a aquellos con conocimientos y habilidades específicas. Esto impedía a personas con ideas, pero sin conocimientos, realizar su software sin tener que acudir a una empresa de software.

Esta situación ha ido cambiando durante los últimos años. Sin embargo, con el crecimiento de las tecnologías No-code/Low-code en las que las personas pueden utilizar interfaces amigables para generar su propio software, sin la necesidad de saber programar.

La investigación presentada en este documento se fundamenta en los aspectos más relevantes sobre las tecnologías No-code/Low-code con el objetivo de definir cómo diseñar una herramienta para crear software sin programar.

PALABRAS CLAVE: Automatización; No-code; Low-code; desarrollo de software; desarrollador ciudadano; arquitectura de software.

ABSTRACT

Software development is a field that has expanded quite a lot, being applied to different areas, thus taking a great importance in the processes of those who use computer tools to perform different tasks by reducing the effort, time and knowledge needed to access information.

However, although software currently plays a fundamental role for people and companies, the software development process was restricted to those with specific knowledge and skills. This prevented people with ideas, but no knowledge, from developing their software without having to go to a software company. This situation has been changing during the last years, with the growth of No-code/Low-code technologies in which people can use friendly interfaces to generate their own software, without the need to know how to program.

With the present document it is intended to carry out a research in which more aspects about No-code/Low-code technologies will be consulted with the objective of defining how to design a tool to create software without writing code.

KEYWORDS: Automatization; No-code; Low-code; software development; citizen developer; software architecture.

INTRODUCCIÓN

El presente proyecto plantea una investigación sobre la programación de software en personas sin conocimientos de programación, mediante soluciones conocidas como “Low-code/No-code”.

A lo largo del presente documento, se evidencia la investigación realizada en temas como:

- ¿Qué es software?
- ¿Cómo se crea software?
- ¿Qué es el software para empresas?
- ¿Cómo se crea el software para empresas?
- ¿Qué son las tecnologías Low-code?
- ¿Qué son las tecnologías No-code?
- ¿Cuál es la diferencia entre Low-code y No-code?
- ¿Qué impacto han tenido las tecnologías Low-code/No-code?
- ¿Qué características deben cumplir estas tecnologías?
- ¿Qué se debe buscar al momento de elegir una solución Low-code/No-code?
- ¿Qué alcance tienen?
- ¿En qué situaciones conviene utiliza tecnologías Low-code/No-code y cuándo no es recomendable?

Para dar respuesta a las preguntas anteriores se accederá a diferentes fuentes de información, tales como bases de datos especializadas y repositorios, dentro de los cuales están Google Scholar, Scopus, Web Of Science, Semantic Scholar, entre otros.

Posteriormente se analizará la información obtenida con miras al establecimiento de la propuesta.

CAPÍTULO I: DESCRIPCIÓN DEL PROYECTO

PRESENTACIÓN DEL PROBLEMA DE INVESTIGACIÓN

La gran mayoría de personas buscan optimizar las actividades o procesos que realizan, una de las opciones más llamativas, en especial cuando se trata de información, es a través de software, uno de los principales problemas que surgen, están relacionados con la accesibilidad a la programación de software, ya sea por tiempo, conocimientos, recursos, entre otros.

JUSTIFICACIÓN

Después de realizar una revisión sobre tecnologías de desarrollo sin código, incluyendo artículos e informes, se llega a la idea de que, dada la información recolectada se encuentra viable desarrollar un software que permita este objetivo. Por este motivo, el presente proyecto busca dar respuesta a la siguiente pregunta problema:

¿Cómo desarrollar una aplicación que permita crear software sin saber programar?

Para llegar a una respuesta, se ha establecido una metodología de trabajo cuyo inicio es establecido desde la recolección de información, acudiendo a repositorios y bibliografía sobre tal temática, especificando sobre los aspectos generales y específicos que una aplicación No-code/Low-code debe cumplir, su alcance, sus restricciones, a quién está orientada, entre otros. Una vez obtenida la información necesaria, se realizará un proceso de ingeniería de software, en el que se define la arquitectura, metodología, herramientas y conceptos para desarrollar una aplicación que permita desarrollar aplicaciones sin la necesidad de programar.

OBJETIVOS

Objetivo general

- Proponer la arquitectura para el desarrollo de una aplicación no-code/low-code.

Objetivos específicos

- Definir el alcance del sistema, así como sus restricciones, limitaciones y necesidades.

- Establecer un modelo que cumpla con el alcance propuesto, incluyendo los diagramas, documentación y técnicas de levantamiento de información necesarias para las etapas de análisis y diseño.

- Seleccionar las tecnologías, herramientas y técnicas de programación a utilizar para el desarrollo del software planteado.

CAPÍTULO II: MARCO TEÓRICO

Como se ha comentado anteriormente, el proceso de programar sin escribir código ha tenido un impacto en diferentes sectores, haciendo que se expanda en popularidad y desempeño. Con los desarrollos que se generan desde la tecnología han surgido nuevos términos, cuyo entendimiento es clave para saber con claridad en qué consiste la programación sin código:

Low-code: Consiste en las herramientas que facilitan parcialmente la tarea de escribir código (Forrester, 2014).

No-code: Consiste en las herramientas que facilitan totalmente la tarea de programar (Beranic, Tina; Rek, Patrik; Heričko, Marjan, 2022).

Desarrollador ciudadano: Término con el que se le conoce a los usuarios de tecnologías no-code (Marten Oltrogge; Erik Derr; Christian Stransky, et al, 2018).

Actualmente, existe mucha información disponible referente a este tipo de tecnologías, muchas veces estos reportes vienen de los mismos proveedores, por lo que se cuenta con información de primera mano, en la que generalmente se exponen cifras que relatan la posición actual de diferentes tipos de usuarios en cuanto a tecnologías no-code/low-code.

Un estudio realizado por Forrester (2020) aclara que, según los usuarios, las razones para elegir un servicio no-code son principalmente:

-Facilidad: Que el software sea lo más amigable posible con el usuario. Esto se logra a partir de interfaces simples pero intuitivas, en las que las opciones más relevantes están al alcance de una mínima cantidad de clics, dejando las opciones menos relevantes en menús y herramientas.

-Adaptabilidad: El software tiene que poder resolver los problemas que tengan sus clientes, incluyendo la mayor cantidad de utilidades para diferentes propósitos, ventas, publicidad, generación de informes, entre otros

-Facilidad de acceso: Para poder utilizar el software, no se debería realizar un esfuerzo mayor, tanto en ingeniería, como económicamente.

Es necesario aclarar que, como todo software, las tecnologías No-code/low-code presentan ciertas desventajas, como consumo de recursos, tanto de software y hardware como económicos, alcance reducido, posibles fallos de seguridad, entre otros. También es importante notar que uno de los principales obstáculos para que los no desarrolladores se adentren en el mundo del software es por su dificultad. Un ejemplo de esta dificultad parte principalmente de proyectos open source, es decir abiertos al público (Raza et al, 2011). Estos proyectos durante su programación no tuvieron en cuenta al usuario final. Por eso no siempre se crean con interfaces intuitivas, o guías simples para poder usar el software (Llerena, R et al , 2022).

Cuando se piensa en un software que ofrece muchas posibilidades pero que es poco amigable con el usuario se encuentra Vim como uno de los ejemplos más claros. Vim es un editor de texto muy utilizado entre programadores, estando entre los cinco (5) editores de código más utilizados (Stack Overflow, 2022), la principal desventaja de Vim es el esfuerzo y tiempo que se debe dedicar para hacer cosas básicas como editar y guardar un archivo. Esta elevada curva de aprendizaje requiere de tanto esfuerzo que la personas deciden utilizar otro software. Es por esta razón por la que el software siempre debe tener al usuario en mente a la hora de planearlo, diseñarlo y programarlo, ya que el usuario final espera un software funcional y fácil de usar. De lo contrario, se presentará una reducción en el número de usuarios (de Oliveira & Zuchi, 2020).

Por esta razón es necesario incorporar técnicas de usabilidad de software, lo que incluye localización, facilidad de usar, simplicidad en las interfaces, entre otros (Castro, J. et al, 2018). Si se implementan estas técnicas en los proyectos de software se puede mejorar, no solo en el funcionamiento de la aplicación, sino en la posición frente a otras aplicaciones (Llerena et al 2019).

Un caso de estudio realizado en el software HistoryCal muestra cómo se vería este sistema antes y después de implementar técnicas de usabilidad (Llerena et al 2018).

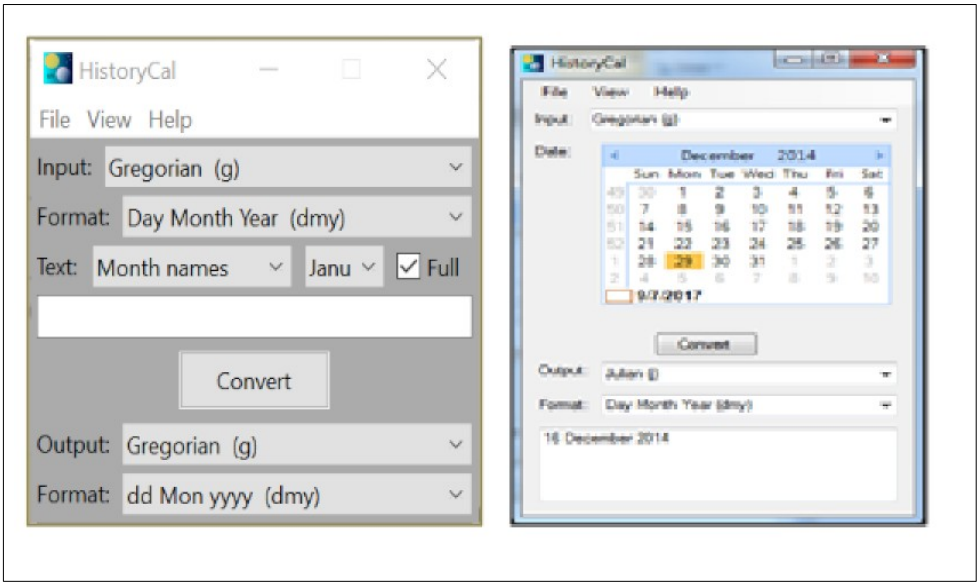

Figura 1: Ilustración de HistoryCal antes y después de aplicar técnicas de usabilidad. Se puede notar que se remplazan las listas desplegables y campos de texto para utilizar controles más dinámicos y sencillos como calendarios.

Aunque también es cierto que la programación no es el único elemento que se relaciona con el software, por ejemplo, los usuarios pueden participar en proyectos de software libre (OSS) sin saber programar (Iivari, N. 2009), los posibles escenarios en que esto sucede son las traducciones, donaciones, pruebas, interfaces gráficas, sugerencias, recomendaciones, etc. Es evidente que desde hace mucho tiempo los usuarios no necesitan saber desarrollar para estar en contacto con el proceso de creación de un software. También existen las listas de correo, que involucran la interacción directa de los usuarios y los desarrolladores. Un estudio que analizó la interacción de los desarrolladores y los no desarrolladores en el proyecto Debian (Sowe, et al 2007) encontró que la interacción y el intercambio de conocimiento por medio de listas de correo es muy elevado, ya que se comparten ideas, mejoras, preguntas, se reportan problemas y se presenta una interacción que invita a los usuarios a integrarse cada vez más en el software.

Los ejemplos anteriores están enfocados a la conexión del usuario final con los desarrolladores, pero no se tiene en cuenta la intervención directa del usuario final con el rumbo que puede tomar el software. Este tipo de usuarios generalmente desempeña cargos administrativos, toman decisiones, proveen los requerimientos, las necesidades, los criterios de aceptación, los controles de calidad, las relaciones con terceros, la promoción del software y el manejo de capital. Este número de responsabilidades puede presentar una carga difícil de sostener, generando problemas en el proyecto (Martin A, 2004). También es cierto que tomar decisiones sobre el software sin conocer de software puede presentar errores, así lo indica un reporte que estudió la asignación de productos en las organizaciones Mozilla y Gnome. En dicho reporte se define que la mayoría de actividades administrativas son llevadas a cabo en parte por no desarrolladores y que presenta un mayor porcentaje de errores (29%) frente al porcentaje de error de tareas asignadas por desarrolladores (18%) lo que puede llevar a retrasos en las entregas o nuevos errores en el software (Xie, J et al, 2013). Para solucionar este problema se crearon herramientas como PAR (Product Assignment Recomender), que, por medio de datos estadísticos, deduce la prioridad de las tareas a asignar y genera una ruta de trabajo con posibles perfiles para asignar las tareas. De esta manera, los no desarrolladores tienen la posibilidad de tomar decisiones más acertadas, reduciendo los riesgos de tiempo y desarrollo drásticamente (Xie, J et al, 2014).

Otro de los roles que se pueden delegar a los usuarios sin conocimientos de programación son las pruebas de software, que consiste en analizar el funcionamiento de un software mediante su uso para encontrar errores. Esto puede ser una tarea que

requiere mucho tiempo, por este motivo se crearon diferentes herramientas para la automatización de pruebas, en las que el usuario define acciones y las pruebas se realizan de manera automática. Todo esto es posible sin saber programar (Gafurov, D. et al, 2018).

Otro perfil que pueden llevar a cabo los usuarios sin conocimientos de programación es la documentación. Este es un aspecto muy importante, ya que en las diferentes metodologías de desarrollo es crucial que el cliente entienda el sistema que se desarrollará (Cesar, I. et al 2014). La manera de comunicar las especificaciones técnicas de una manera que el cliente entienda es con la documentación. Existen numerosas maneras de documentar un software que será desarrollado, por ejemplo, en metodologías de desarrollo tradicionales se utiliza UML, que consiste en una serie de diagramas que relatan la estructura, el funcionamiento y las partes que integran un sistema, así como la interacción con sus usuarios (Fontela, 2012).

Por otra parte, en metodologías de desarrollo ágiles, se cuenta con documentos que definen los tiempos, las tareas a realizar, los posibles caminos que puede tomar el software, las especificaciones de cada ciclo o iteración (denominados sprints). Una de las principales ventajas de las metodologías ágiles es que no solo está lista para el cambio en los requerimientos, sino que los fomenta ya que se tiene en cuenta que el cliente o el usuario final no siempre está seguro de qué quiere, por lo que las posibilidades de se encuentren cambios en los requerimientos son elevadas (Cohen, 2003).

ANTECEDENTES DE LA INVESTIGACIÓN

Las tecnologías Low-code/No-code presentan un nuevo camino a la hora de programar aplicaciones, y es gracias al impacto que han tenido las diferentes soluciones que ofrecen estos servicios que han adquirido tal popularidad, llegando así a ser una opción que vale la pena considerar a la hora de desarrollar aplicaciones.

Según un reporte realizado por Forrester, (2020), en el que se estudia el impacto que ha tenido Mendix, una plataforma para desarrollar aplicaciones sin código se encontró que en tres años se ha obtenido una ganancia de más de 20 millones de dólares. Las tecnologías de poco o nada de código aceleran el proceso de desarrollar software, mientras que reducen las herramientas necesarias, así como el costo de utilizarlas, sin la ayuda de tecnologías de poco código, las empresas necesitarían personal especializado y diferentes herramientas de trabajo, lo que se traduce en costos elevados. Pero con una solución de poco o nada de código, se reduce el tiempo y costos necesarios. Según este reporte, varias de las razones por las que este tipo de servicios son usados son:

Facilidad de responder a las necesidades de un cliente, habilidad de una compañía de manejar sus propias mejoras, facilidad de acceso, facilidad de adaptabilidad y cambio, entre otras.

También se reportó que un 45% de los participantes del estudio aceptaron que sus empresas tuvieron o tienen planes de utilizar una herramienta de poco o nada de código (Low-code/No-code).

Otro reporte realizado por Kintone (2017) aclara que un 74% de las empresas participantes tienen debilidades en la velocidad de entrega de sus aplicaciones. Lo que se traduce en aumento de costos por el hecho de tener que mantener el proceso de desarrollo por más tiempo y reducción de ganancias, al no contar con el impulso que tendrían si se contara con su aplicativo. El 76% de los entrevistados anuncian que una o varias de sus aplicaciones fueron desarrolladas por fuera de la empresa. Lo que da una impresión acertada de la situación de las empresas que no cuentan con personal especializado y por eso tienen que ajustarse a las tarifas de empresas externas para el desarrollo de sus aplicaciones. Esto significa que una empresa debe invertir más si se desarrolla su aplicación por medio de un tercero. Los “Desarrolladores ciudadanos” crean sus aplicaciones más rápido que departamentos grandes de IT. Los retos que enfrentan son la seguridad y la adquisición de habilidades de programación para el manejo de datos.

Este es uno de los aspectos más importantes de las tecnologías Low-code/No-code, ya que se definen no solo los principales beneficios, sino que también se exponen los desafíos a los que los usuarios de las tecnologías Low-code/No-code se deben enfrentar: La seguridad en sus aplicaciones y la capacidad de darles un uso adecuado para evitar problemas con el manejo de datos. La razón por la que los desarrolladores ciudadanos trabajan de esta manera es porque sienten que los departamentos de IT

son muy lentos. Del mismo modo que el punto anterior, esto representa uno de los principales beneficios de las Tecnologías Low-Code/No-code: la reducción de tiempos. Esta reducción de tiempos se traduce en la reducción de costos. Un tercio de las empresas apoyan a los desarrolladores ciudadanos. Aunque un cuarto de los ejecutivos no tiene habilidades de programación. Esta es una de las limitaciones que se han impuesto a las Tecnologías Low-code/No-code: A la hora de desarrollar una aplicación estos servicios no se tienen en cuenta y, en muchos casos, se desaconsejan (probablemente debido al desconocimiento del tema).

Tomando en consideración el artículo elaborado por Lionel Mew y Daniela Field (2020), en el que se presenta una descripción de la plataforma Mendix, para desarrollo de aplicaciones de alto nivel, es decir, más cercano al usuario, se puede apreciar la manera en la que esta plataforma funciona, basándose en los siguientes principios:

Colaboración: Este ítem hace referencia al trabajo en conjunto y simultáneo que se realiza por parte de dos o más desarrolladores ciudadanos

Datos: Para generar una base de datos, Mendix ofrece la posibilidad de crear un diagrama de clases, en donde se indica qué información se almacenará. De esta manera, Mendix genera las sentencias SQL para automatizar la creación de la base de datos.

Lógica de negocio: Funciona igual que el ítem anterior, en este caso se crea un diagrama o mapa de procesos, a partir del cual Mendix realiza la lógica para evitar el proceso de programar.

Interfaz y experiencia de usuario: Mendix ofrece interfaces sencillas e intuitivas en las que no son necesarios conocimientos avanzados.

Seguridad y autenticación: Mendix cuenta con varias formas para proteger la información, entre las que se encuentran la restricción de permisos y accesos y diferentes métodos de verificación de usuarios.

Despliegue: Mendix ofrece la posibilidad de integración con varios proveedores, de esta manera las aplicaciones están sincronizadas con la nube para tener acceso público.

Tal como se indica en (Lionel Mew, Daniela Filed, 2020), esta tecnología surgió en una época en la que las herramientas, tecnologías y procesos del momento no permitían cumplir las altas expectativas, lo que llevó a la pérdida del interés por parte de los usuarios. Actualmente, esas expectativas son posibles. Esta es la razón de la popularidad de Mendix, ya que ha sabido implementar robustez, escalabilidad y trabajo colaborativo, sin perder velocidad o facilidad al momento de desarrollar aplicaciones.

Es importante notar que Mendix ofrece un software estándar que se va moldeando a los procesos de los usuarios, llegando así a solucionar problemas más específicos conforme el usuario va parametrizando su funcionamiento. Pero también existen numerosos sistemas de información dirigidos específicamente a objetivos concretos.

Entre las aplicaciones con enfoques concretos se encuentra Pydpiper, es un proyecto de software libre escrito en el lenguaje de programación Python que se especializa en ofrecer técnicas y herramientas para visualizar varios aspectos del sistema nervioso (Friedel, M. et al, 2014). Pydpiper está en constante desarrollo y su versatilidad ha hecho posible diferentes implementaciones, una de ellas es MAGeT, que permite crear plantillas con información que se tenga a la mano para generar mapas que a partir del procesamiento automático en diferentes etapas para llegar a un resultado deseado (Mallar M. et al, 2012).

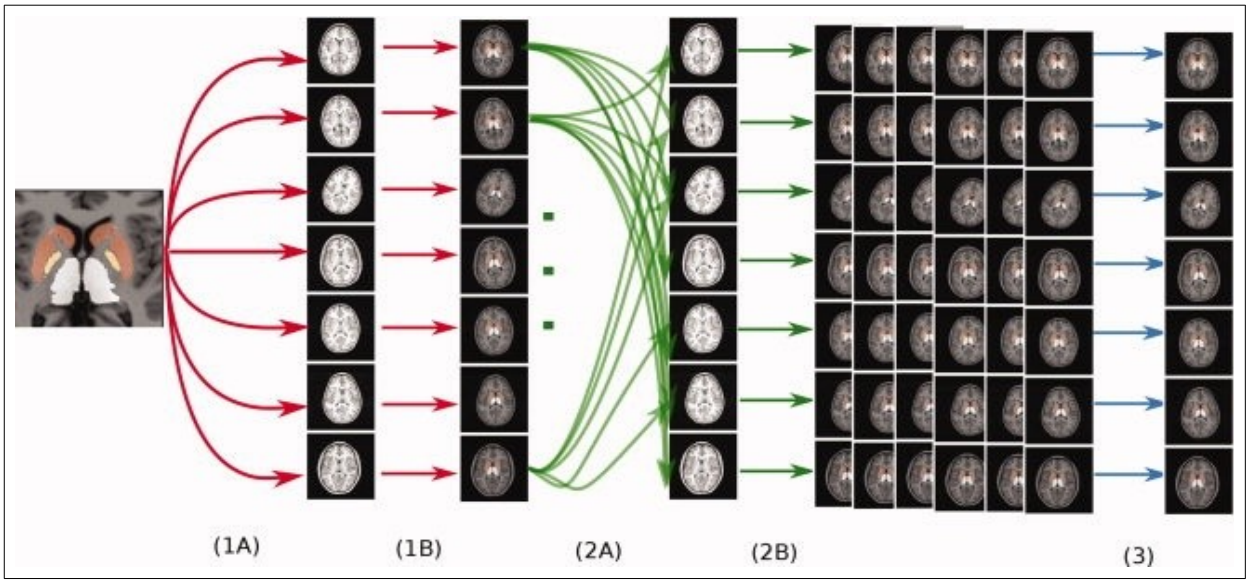

Figura 2: Ilustración de la implementación de Pydpiper, MAGeT en la que se procesan datos en diferentes etapas para obtener información automáticamente

Como se puede apreciar, Pydpiper está orientado a la neurociencia, pero también existe software especializado en otras áreas, por ejemplo, ASDL (Aviation Scenario Definition Language). Se trata de una aplicación para gestionar información de salidas, rutas y escenarios de aterrizajes utilizados por los aeropuertos (Jafer, S. et al 2017). Dicha aplicación toma un modelo ingresado por el usuario para convertirlo en una interfaz gráfica.

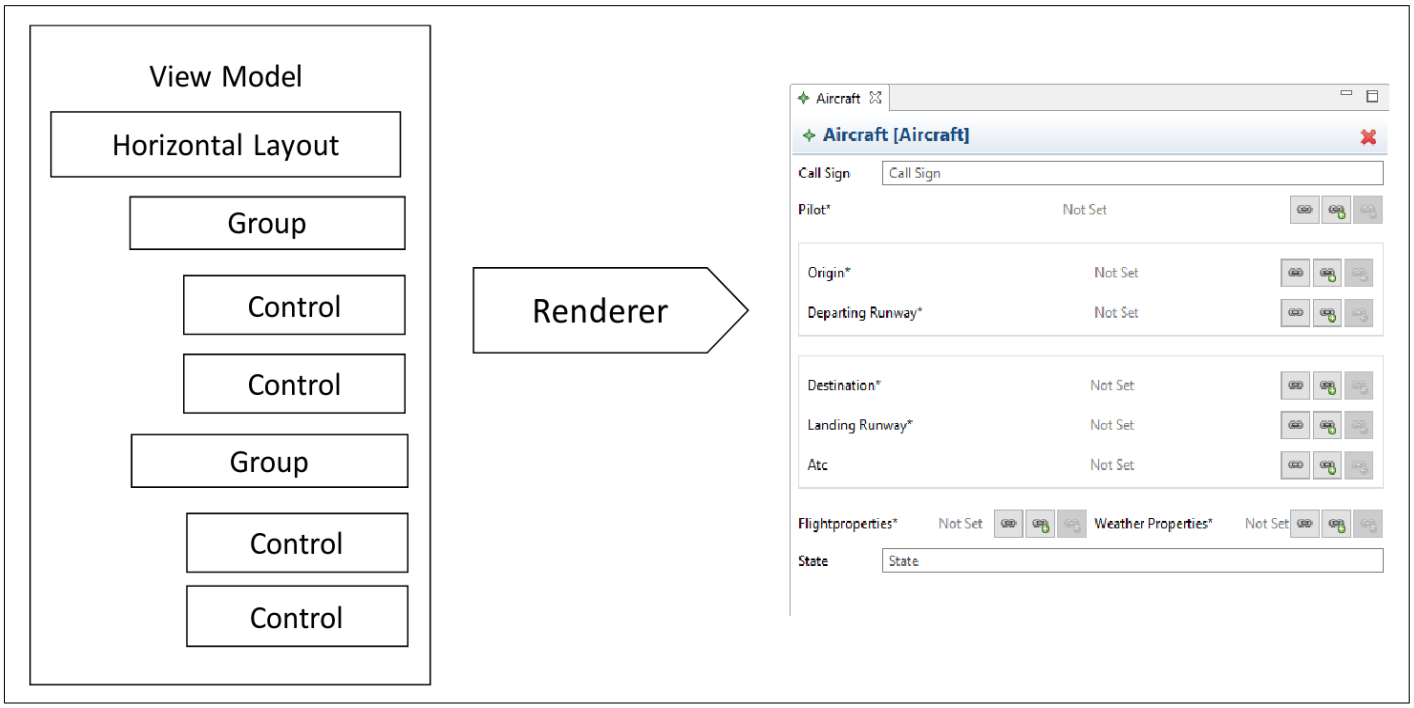

Figura 3: El modelo es interpretado por un motor de renderizado que convierte los controles definidos por el usuario en elementos para crear interfaces gráficas.

Otro ejemplo es uAdventure, un motor para crear videojuegos de manera sencilla para no programadores (Pérez C et al, 2019), ofreciendo diferentes controles para varios tipos de juegos, incluyendo juegos interactivos, novelas visuales, videojuegos con mapas extensos, etc.

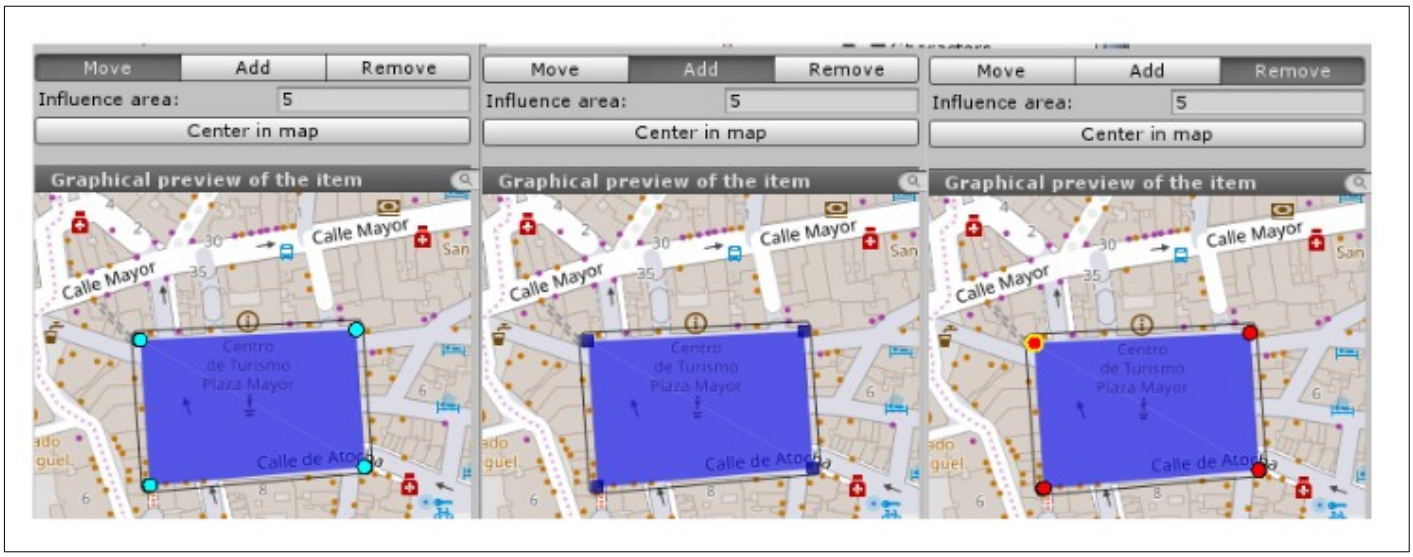

Figura 4: Posicionamiento de elementos en mapas utilizando uAdventure.

BASES TEÓRICAS O FUNDAMENTOS CONCEPTUALES

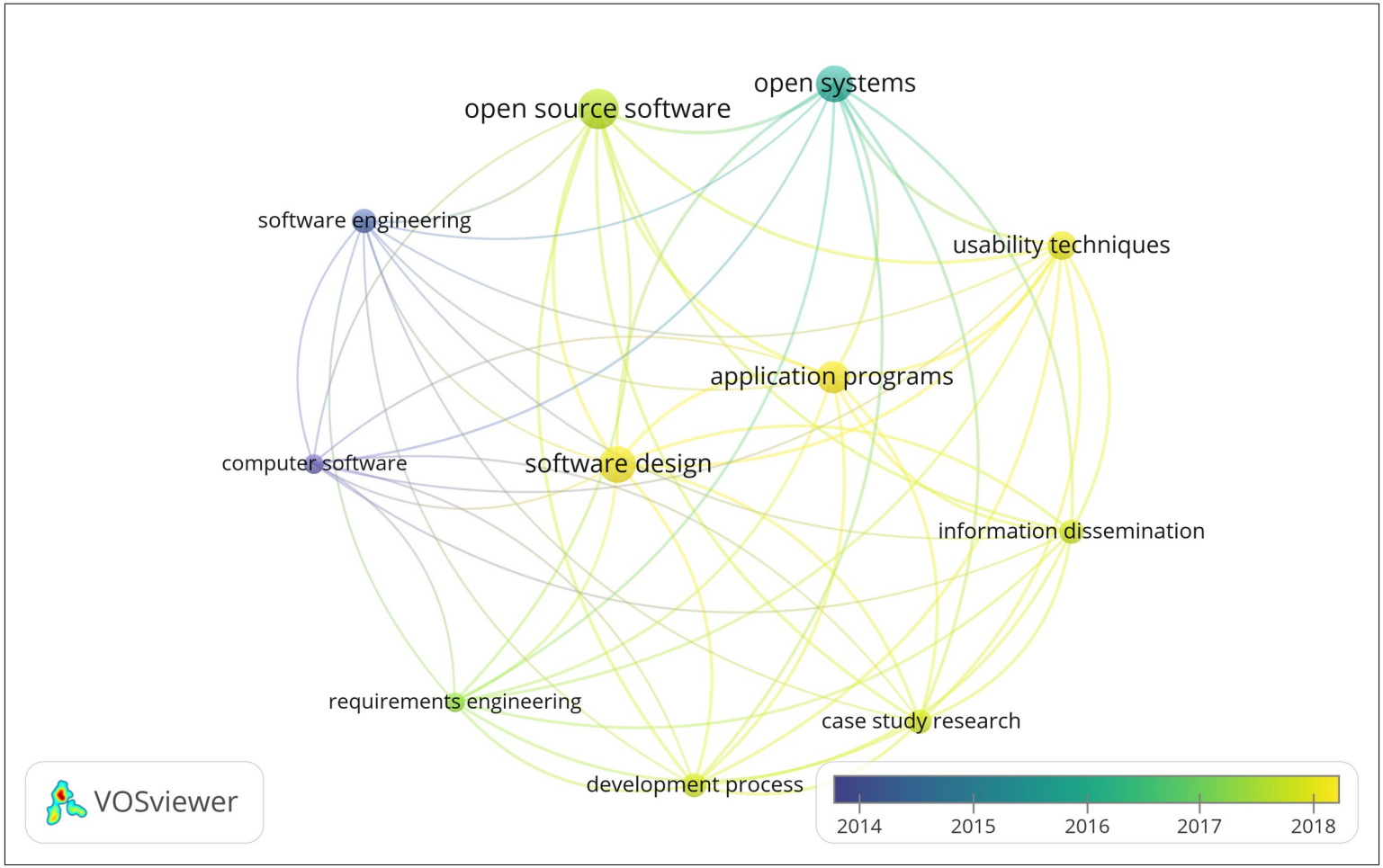

Figura 5: Tipo de análisis: Co-ocurrence, unidad de análisis: all keywords. utilizando la herramienta VOSviewer.

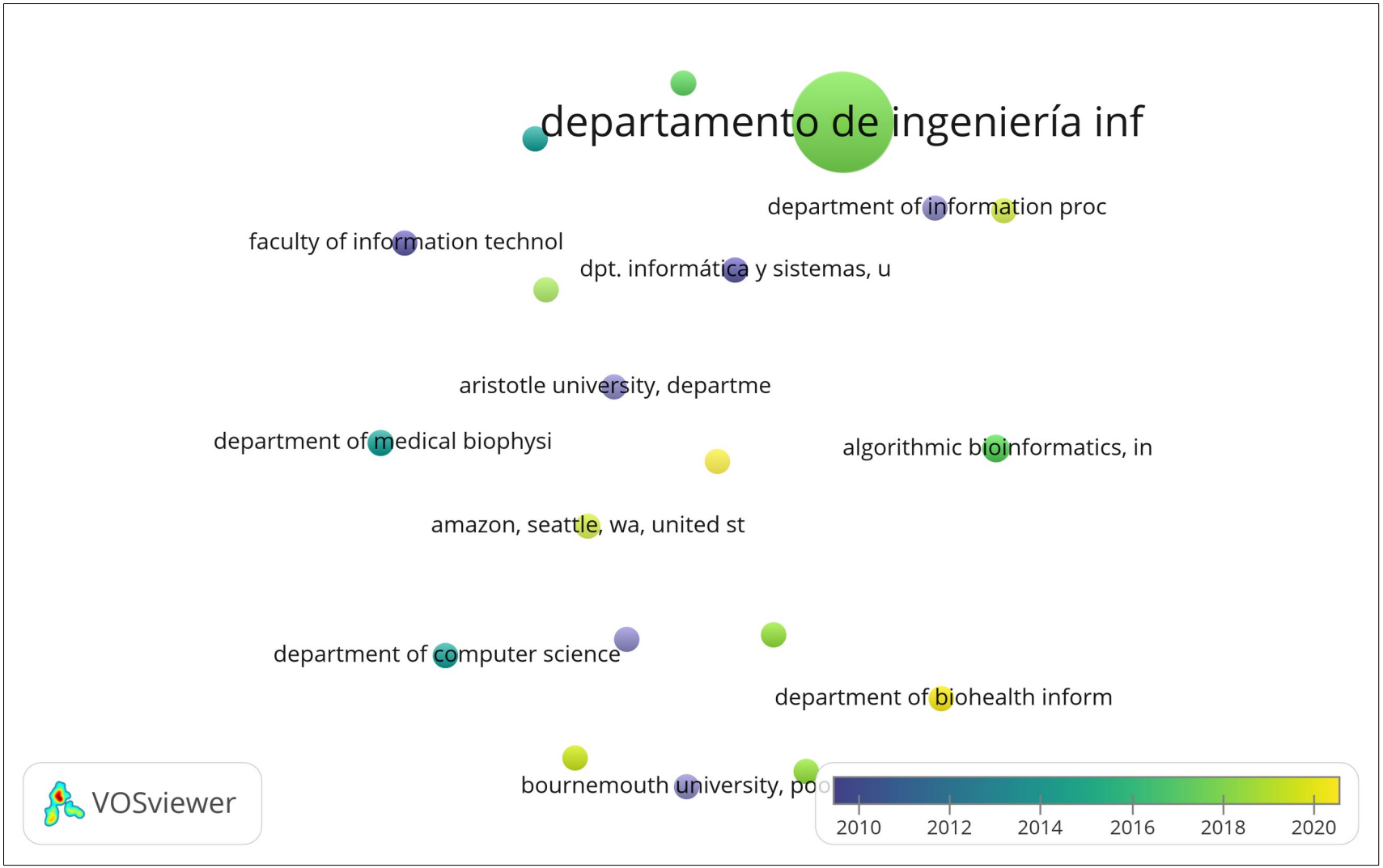

Figura 6: Tipo de análisis: Co-authorship, unidad de análisis: countries. utilizando la herramienta VOSviewer.

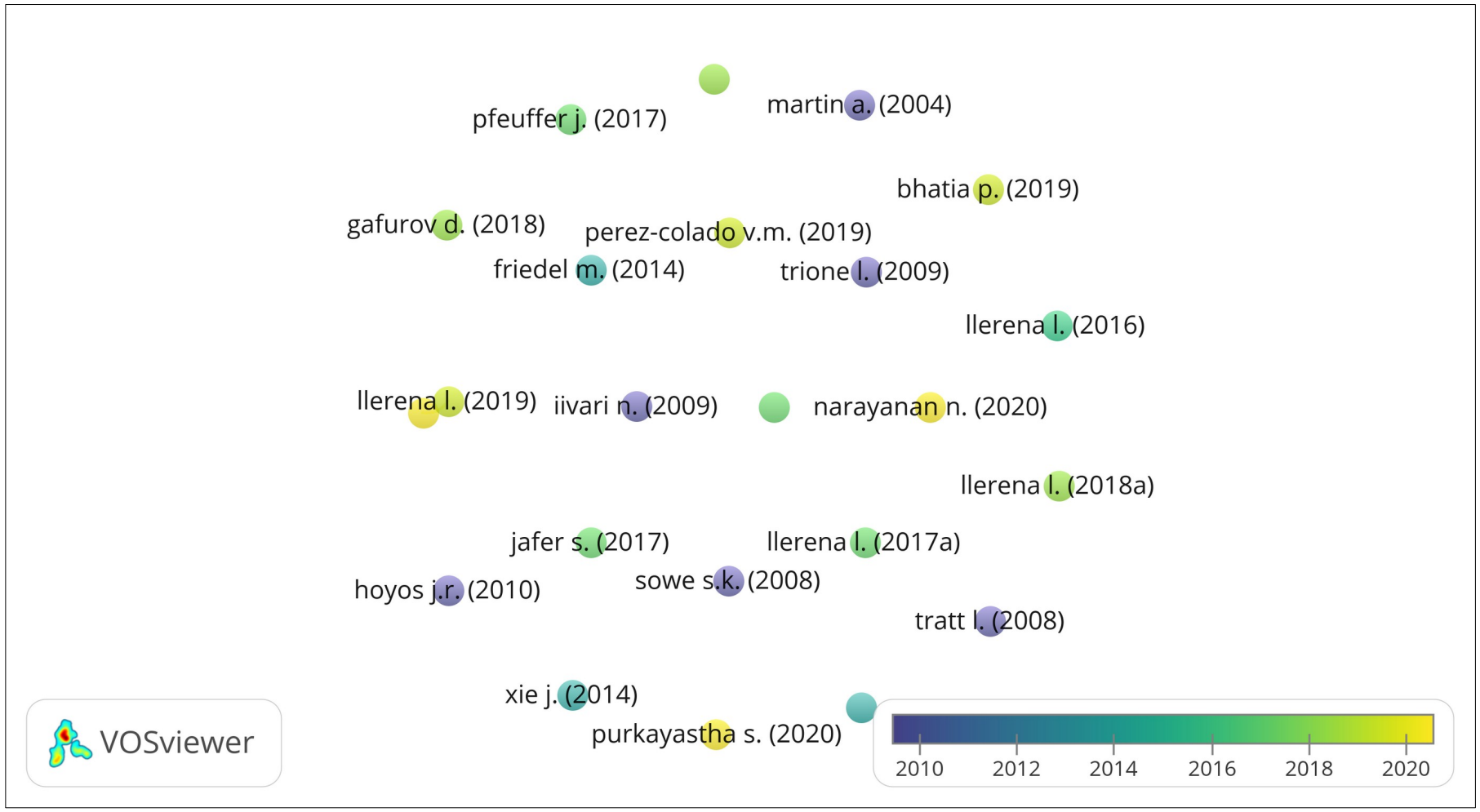

Figura 7: Tipo de análisis: Citation, Unidad de análisis: Documents. Utilizando la herramienta VOSviewer.

BASES LEGALES DE LA INVESTIGACIÓN

Para analizar las bases teóricas de un proyecto de software se deben contemplar dos aspectos, el manejo que el usuario le dará al software y la manera en la que el software maneja los datos de los usuarios.

Teniendo en cuenta el manejo por parte de los usuarios que utilizan el software existen documentos denominados Licencias de software, se trata de condiciones que regulan el manejo del software, su distribución y comercialización (Ballhausen, M. 2019.).

En cuanto al manejo de datos por parte del software existen diferentes regulaciones dependiendo del país en que se consulte, en el caso de Colombia existe la LEY ESTATUTARIA 1581 DE 2012 “Por la cual se dictan disposiciones generales para la protección de datos personales” que restringe el manejo de información sensible de los usuarios, salvo que se cuente con autorización, o se utilice para cuestiones médicas, la protección de un proceso jurídico, o se trate de un tratado del cual Colombia hace parte.

CAPÍTULO III: DISEÑO METODOLÓGICO

Para definir el alcance del software se llevará a cabo un proceso de ingeniería de software en el que se analizarán los posibles escenarios en los que el software puede realizarse, de una manera u otra, teniendo en cuenta los usuarios a los que estará orientado, contemplando los diferentes aspectos como las necesidades o la población a la cual puede estar orientado el software. Una vez se tenga la información a un nivel general, se partirá hacia lo específico, para definir los procesos a realizar para diseñar el software, entre los que se encuentran:

-Levantamiento de información.

-Análisis de requerimientos.

-Diseño de la arquitectura del sistema.

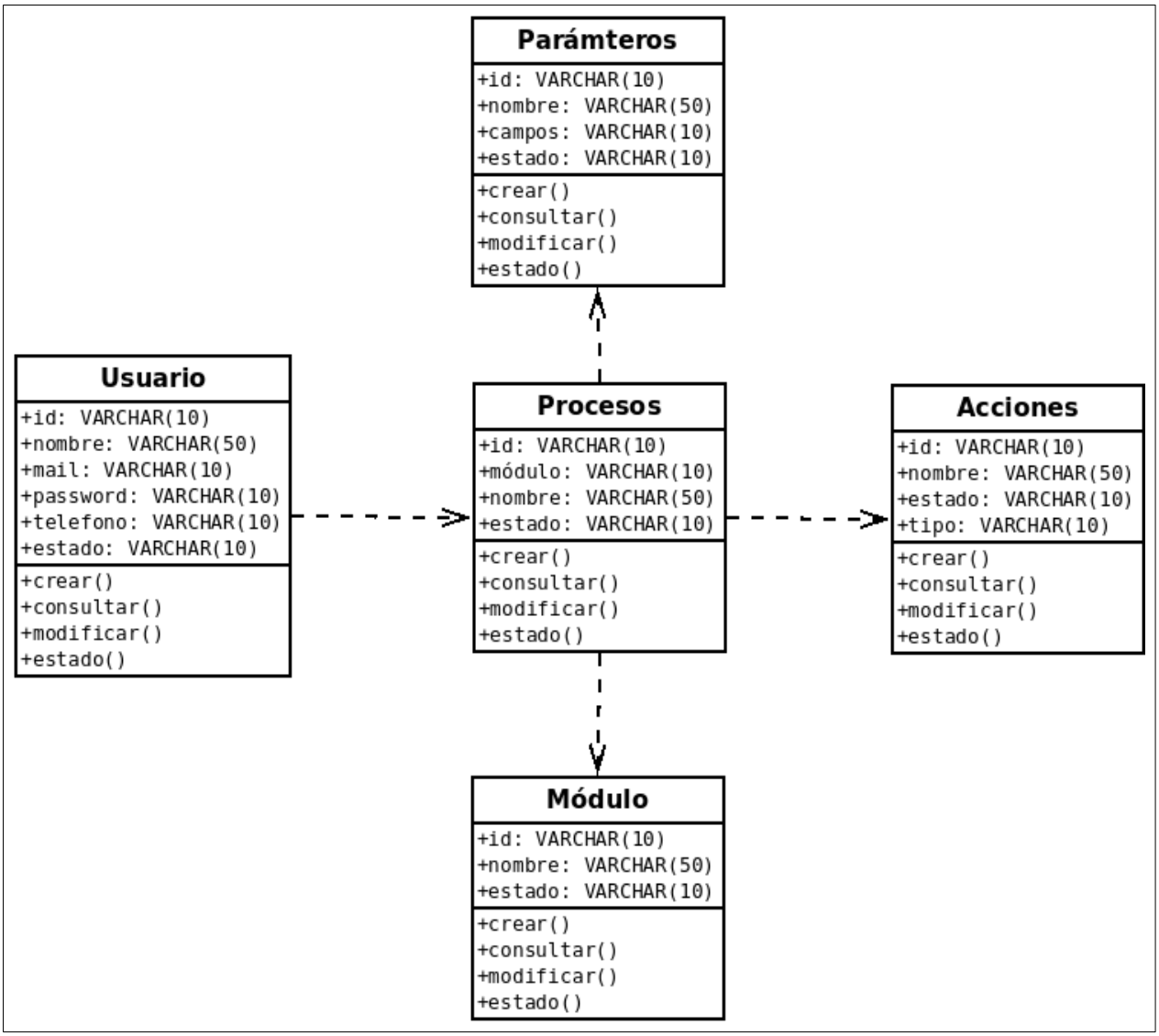

Figura 8: *Diagrama de clases para la aplicación elaborada por el autor en el programa DIA.*

En este diagrama de clases se describe la manera en la que se almacenará la información. Puede parecer un modelo simple y genérico, pero de esta manera se garantiza la escalabilidad y la posibilidad de crear módulos y procesos de manera continua.

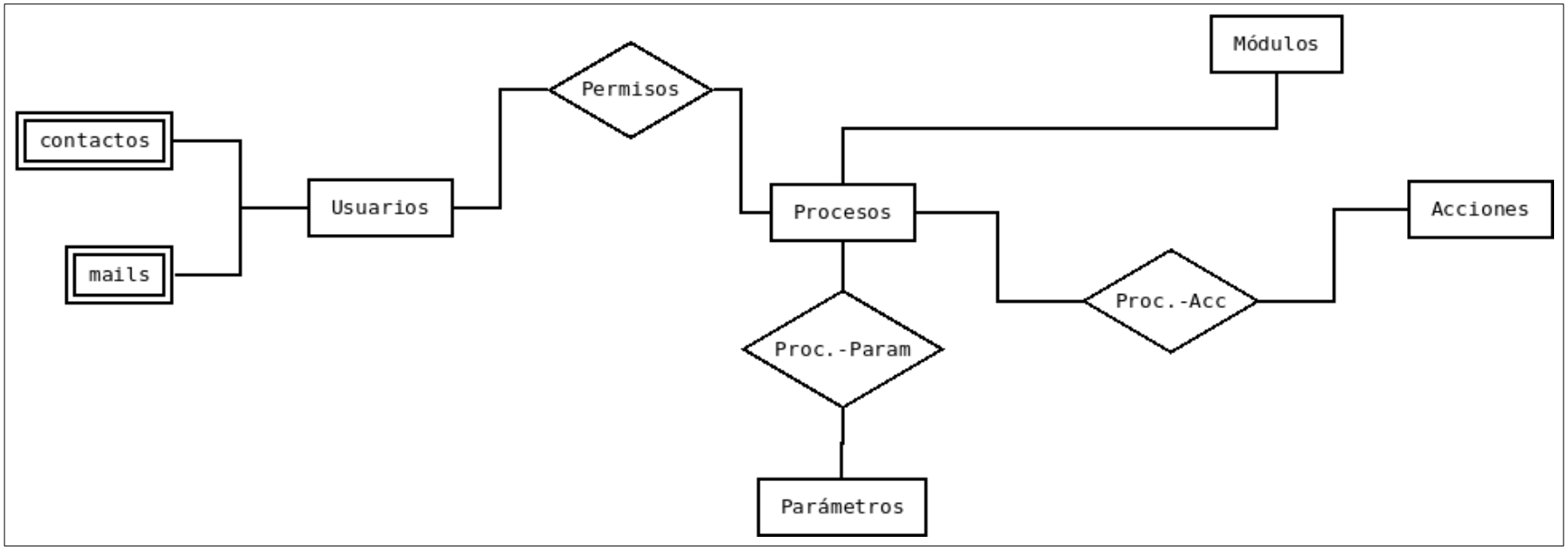

Figura 9: *Modelo entidad-relación MER elaborado por el autor en el programa DIA.*

Para garantizar que la integridad y evitar la duplicidad de la información, los módulos planteados se dividirán, generando así las conexiones entre módulos y permitiendo que se puedan crear diferentes módulos, procesos, acciones, permisos y usuarios, sin que exista conflicto entre los datos.

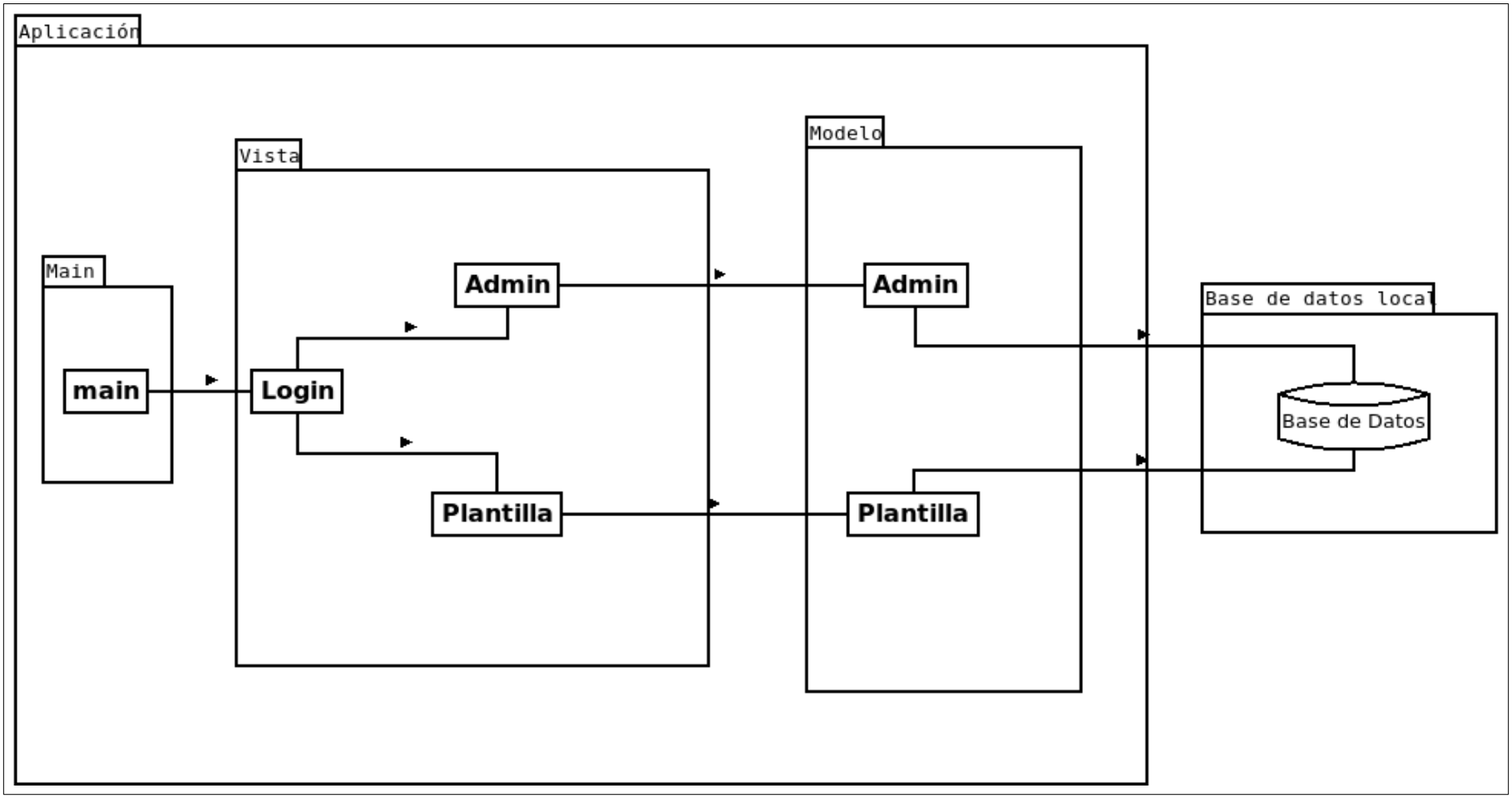

Figura 10: *Arquitectura del sistema elaborada por el autor en el programa DIA.*

En este diagrama se puede apreciar la arquitectura del software, utilizando un modelo de capas, para garantizar que la información se procese de diferentes maneras dependiendo del nivel donde se encuentra. También se pueden observar dos rutas: la ruta definida y la ruta indefinida. La ruta definida recorre el único camino que se tiene claro, el camino del administrador, en donde se crean los módulos, los usuarios, los procesos y demás funcionalidades del sistema. La ruta indefinida es el resultado de la ruta definida, es en donde los usuarios creados pueden interactuar con el sistema parametrizado en el camino del administrador.

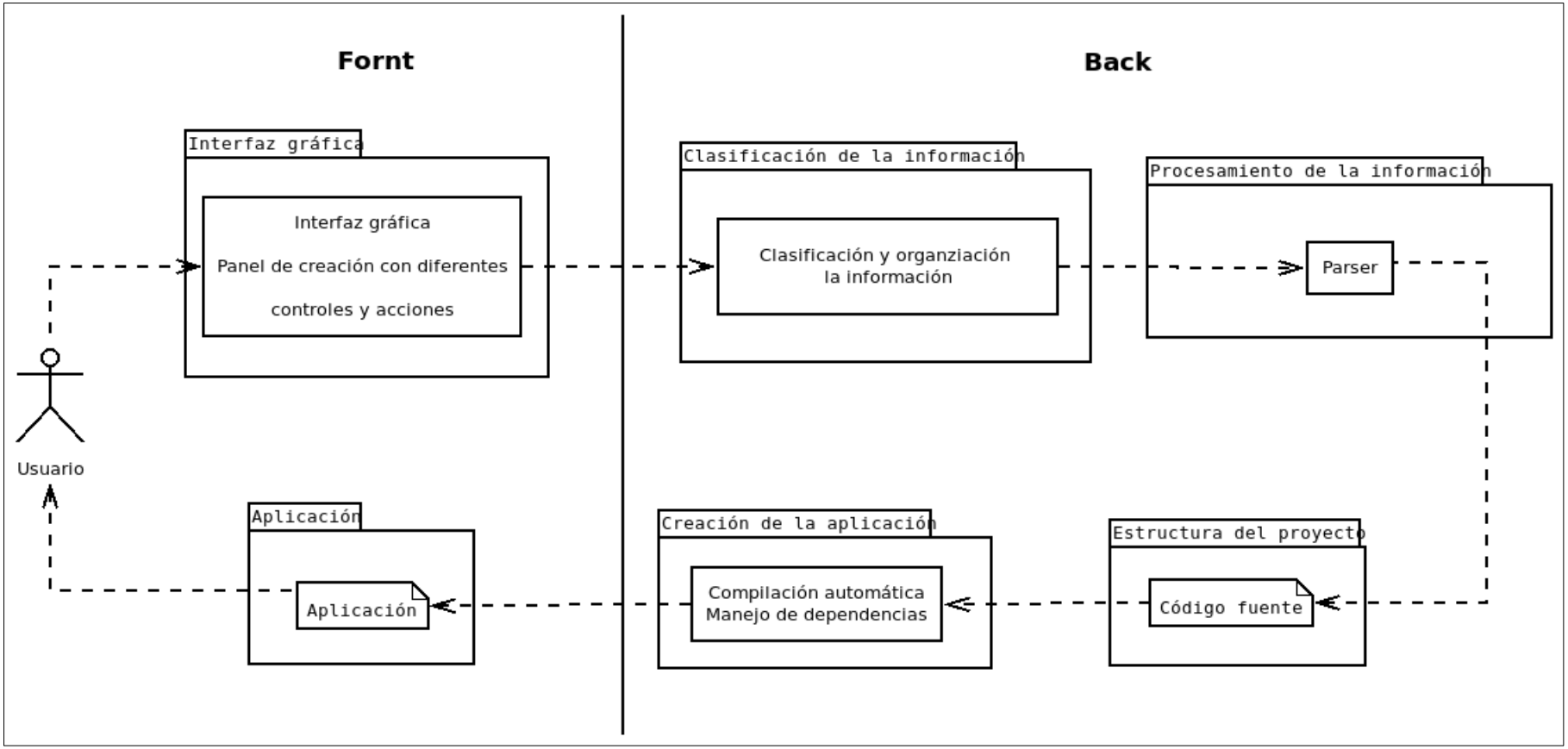

Figura 11: *Proceso planteado por el autor en el que se genera una aplicación por medio de la interacción con el usuario*

En la imagen anterior se presenta un modelo en el cual el usuario interactúa con un panel de creación con diferentes controles y acciones. Cada acción, por ejemplo, dar clic a un botón, lanza un trigger o un evento que envía la información necesaria a la siguiente capa, la capa de clasificación. Esta capa toma la información recibida, la procesa y la organiza para generar blobs o componentes separados en los que se indica de qué se debe hacer con esa información, de qué manera y qué componentes son necesarios para hacerlo.

La siguiente capa es la más importante, el parser, que se encarga de recibir los blobs generados en la capa anterior y crea el código fuente de cada uno, luego de tener todos los blobs procesados, los integra en la estructura de un proyecto de software, indicando las dependencias que se encontraron para cada archivo generado.

Una vez se tiene el código fuente listo, se resuelven las dependencias indicadas y se procede a compilar el código fuente, obteniendo así una aplicación para el usuario.

TIPO DE INVESTIGACIÓN

Se realizará una investigación aplicada con el objetivo de documentar la arquitectura de un sistema de información, para poder describir cómo será el comportamiento del software.

POBLACIÓN

Ya que la presente investigación se realiza para un proyecto de software, definir la población a estudiar resulta complejo, por la cantidad de posibilidades que el software ofrece, por lo tanto, se utilizarán técnicas de levantamiento de información que permitan llegar a un público general.

TÉCNICAS E INSTRUMENTOS DE RECOLECCIÓN DE DATOS

Se realizará una encuesta con las siguientes preguntas

¿Sabe programar?

Si pudiera desarrollar una aplicación, ¿Qué funcionalidad tendría?

¿Con qué fines utiliza software?

¿Conoce software que permita desarrollar sin saber programar?

Si es así ¿ha usado alguno?

¿Qué espera de una aplicación que le permita desarrollar software sin saber programar?

La encuesta se puede consultar en:

https://docs.google.com/forms/d/e/1FAIpQLScAKezWUifz151BEEx6RRqRvKX4sgG3VkAFaW109jtDOFHo9w/viewform?usp=sf_link

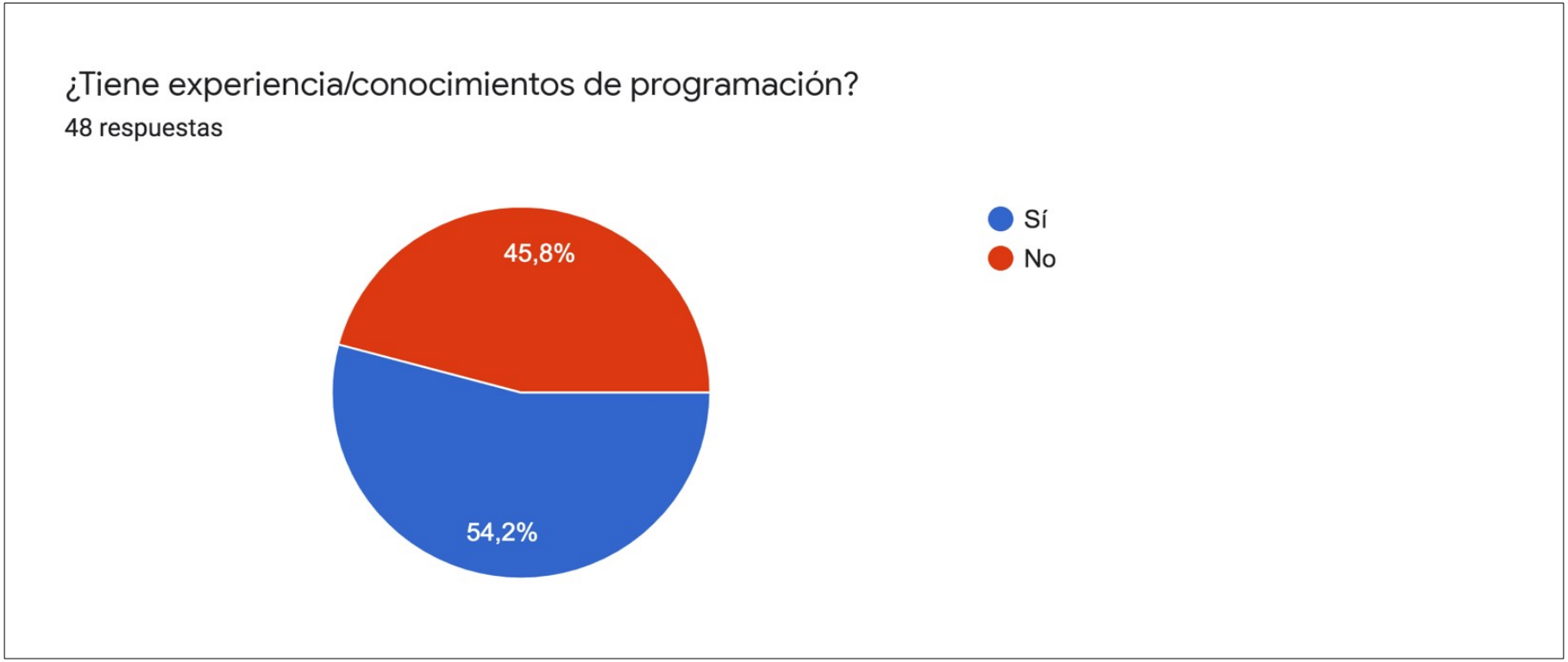

Figura 12: *Respuestas de la pregunta 1.*

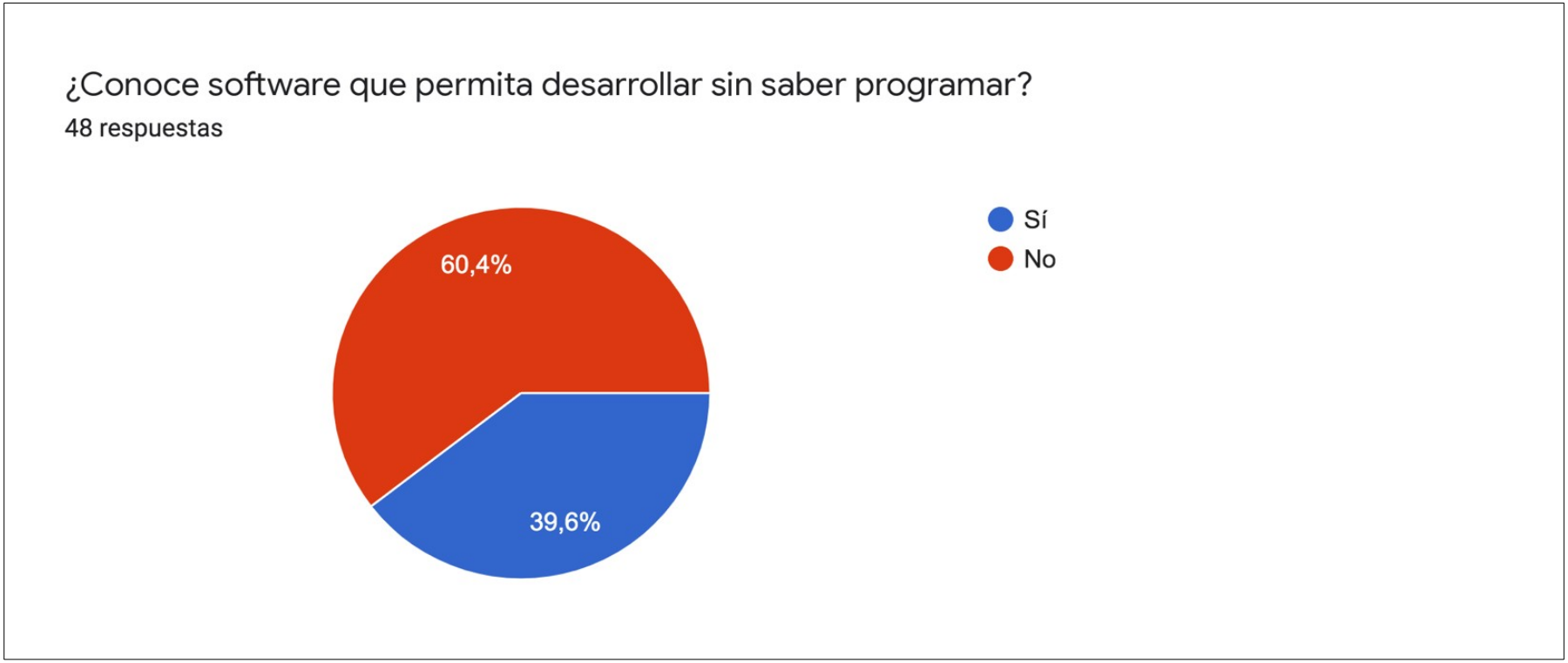

Figura 13: *Respuestas de la pregunta 4.*

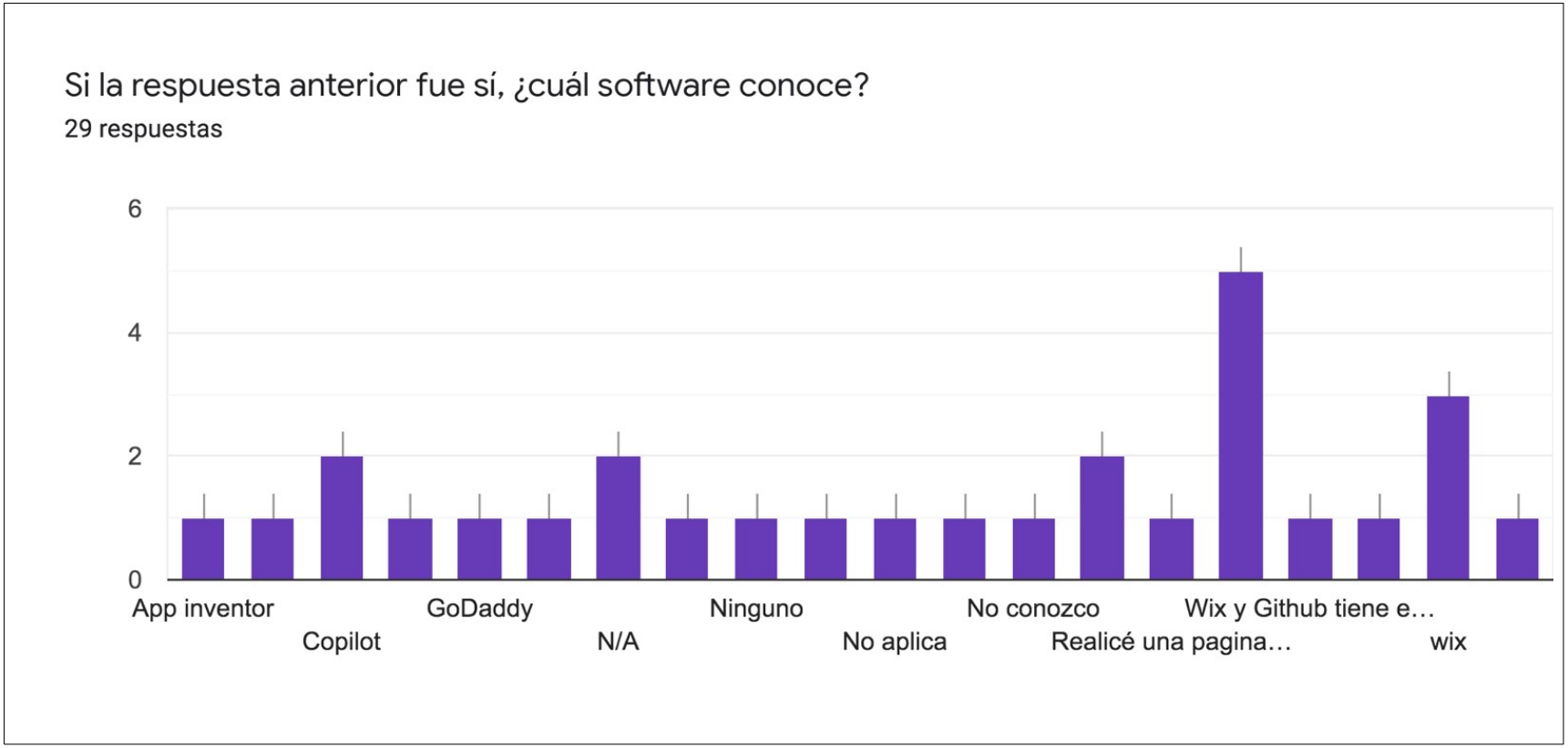

Figura 14: *Respuestas de la pregunta 5.*

Analizando los resultados de la encuesta, es posible apreciar que la mayoría de encuestados no tienen conocimientos de desarrollo de páginas web, aunque tienen ideas de nuevas aplicaciones que les serán de utilidad para gestionar información y automatizar ofertas en sus trabajos o estudios. También se puede apreciar cómo tienen en sus prioridades la seguridad, la velocidad, la funcionalidad y que la aplicación cumpla con principios de experiencia de usuario como una interfaz intuitiva, que sea fácil de usar, o que puedan acceder desde diferentes lugares y dispositivos

Teniendo en cuenta la información recolectada en la encuesta se encuentra viable reducir el alcance de la aplicación para enfocar el desarrollo en las utilidades que más se solicitaron y que cumplen con un objetivo estándar, es decir, independientemente del tipo de proyecto se pueden integrar para lograr diferentes funcionalidades.

La función principal del proyecto es la generación de código a partir de la interacción con el usuario. Ya que la facilidad de uso se encuentra en los objetivos de la aplicación, se tiene en cuenta que el usuario espera, en un nivel técnico, velocidad y disponibilidad, es decir, que la aplicación que desarrolle el usuario sea rápida, estable y funcione en diferentes sistemas operativos. Para solventar esta necesidad se pensaron dos lenguajes de programación principalmente: Java y C++. Se eligió C++ para esta tarea por su potencia, robustez, posibilidades que ofrece y su integración con otros componentes como librerías y compatibilidad con diferentes sistemas operativos sin aumentar drásticamente el consumo de recursos. Java también ofrece estas características, pero para lograr la compatibilidad se necesita de una pieza de software adicional denominada JVM (Java Virtual Machine). Al tratarse de una máquina virtual, una aplicación desarrollada en Java tiene la posibilidad de ser rápida y robusta, pero a cambio de un mayor consumo de recursos (Oracle, 2015). C++ presenta una mayor complejidad a la hora de programar y generar código para una aplicación, pero a menos que se utilicen librerías y funciones específicas de un sistema operativo, una aplicación en C++ puede ser compatible con Windows y con sistemas operativos basados en GNU/Linux.

Al elegir C++ sobre Java se llega a un aspecto muy importante: La interfaz gráfica, ya que a diferencia de Java, C++ no ofrece la posibilidad de crear interfaces gráficas de manera nativa. Por esta razón, surge un paso adicional en la elección de herramientas: la elección de una librería para crear interfaces gráficas.

Al ser este un proyecto multiplataforma, se debe elegir un marco de herramientas de desarrollo que sea multiplataforma. Las dos opciones principales son GTK y QT, ambas funcionan con C++, pero GTK tiene dos ventajas sobre QT, una licencia más permisiva

(GTK, 2012) y una integración con más lenguajes de programación (GTK, sf). Esto resulta muy útil en caso de que se vea necesario integrar otro lenguaje. Por parte de QT, cuenta con dos licencias, una de ellas comercial, lo que limita el posible desarrollo de la aplicación en caso de que sea comercial (QT, sf).

El siguiente aspecto que se debe tener en cuenta es la optimización de tiempos, este es un problema que se ve más notorio con la integración de interfaces gráficas ya que, el proceso de compilación consiste en diferentes etapas que toman el código fuente y lo traducen a lenguaje de máquina, es decir código binario (Stevanovic, M. 2014)

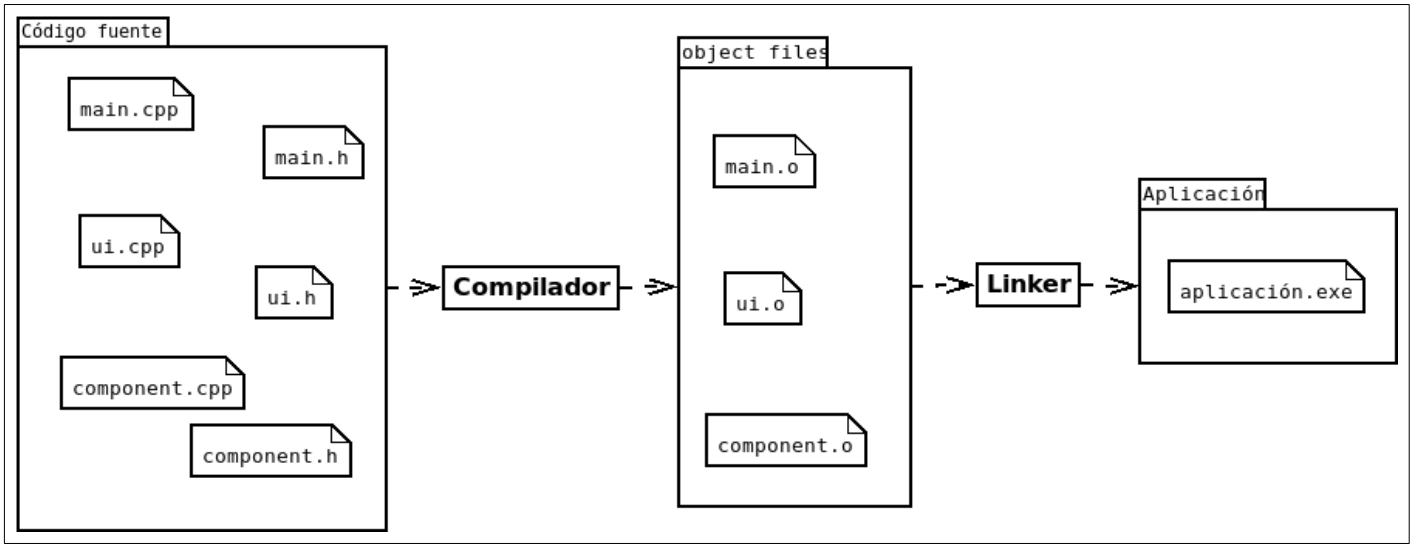

Figura 15: *Proceso de compilación de archivos fuente para crear object files que son enlazados para formar un ejecutable, es decir, una aplicación.*

Este es un proceso lento, ya que cada vez que se realiza un cambio, se debe compilar de nuevo el proyecto, y para compilar el proyecto se deben compilar todos los archivos. El tiempo de compilación toma más tiempo cuando se trabaja con archivos complejos, como interfaces gráficas. Para reducir este tiempo se empleará la herramienta para la automatización de compilación llamada Make. Que consulta la fecha de modificación de los archivos necesarios para crear una aplicación (en el caso de C++, los object files) y solo compila los archivos que se han modificado desde la

última compilación. De esta manera, si se modificó un solo archivo, no es necesario compilar todo el proyecto nuevamente (GNU, 2020).

El siguiente aspecto es uno de los más importantes: La base de datos, ya que es el lugar en donde se guardará toda la información. Para la base de datos, se requiere un servidor de base de datos, lo que conlleva a pensar en temas como la arquitectura del servidor, el tipo de conexión, la seguridad, la disponibilidad, entre otros. Al ser este un proyecto con un alcance reducido, se optó por implementar una base de datos local, lo que evita pasar por los temas relacionados con la administración de servidores. Se eligió SQLite como herramienta para administrar la base de datos local, ya que implementa un motor de base de datos que puede integrarse directamente con C++ y permite gestionar la información de manera eficiente, sin la necesidad de mantener un servidor dedicado (SQLite, sf).

Hasta el momento se cuenta con las herramientas necesarias para la creación de una aplicación de escritorio, con interfaz gráfica, tiempos de compilación reducidos y manejo de la información por medio de una base de datos. Estas herramientas resultan suficientes para desarrollar aplicaciones funcionales, pero al tratarse de necesidades de diferentes usuarios y tipos de proyectos, se encuentra necesario la integración directa con más componentes que permitan la realización de diferentes tareas que, tomando en cuenta los resultados de la encuesta realizada, facilitarán las tareas que realizarán los usuarios.

El primer componente adicional consiste en la visualización de datos, para generar gráficos e indicadores con la información disponible en la base de datos. Para este objetivo se utilizará la Librería Matplot++, una librería escrita para C++ que permite la realización de gráficos en 2D y 3d a partir de diferentes fuentes de información (Matplot++, sf).

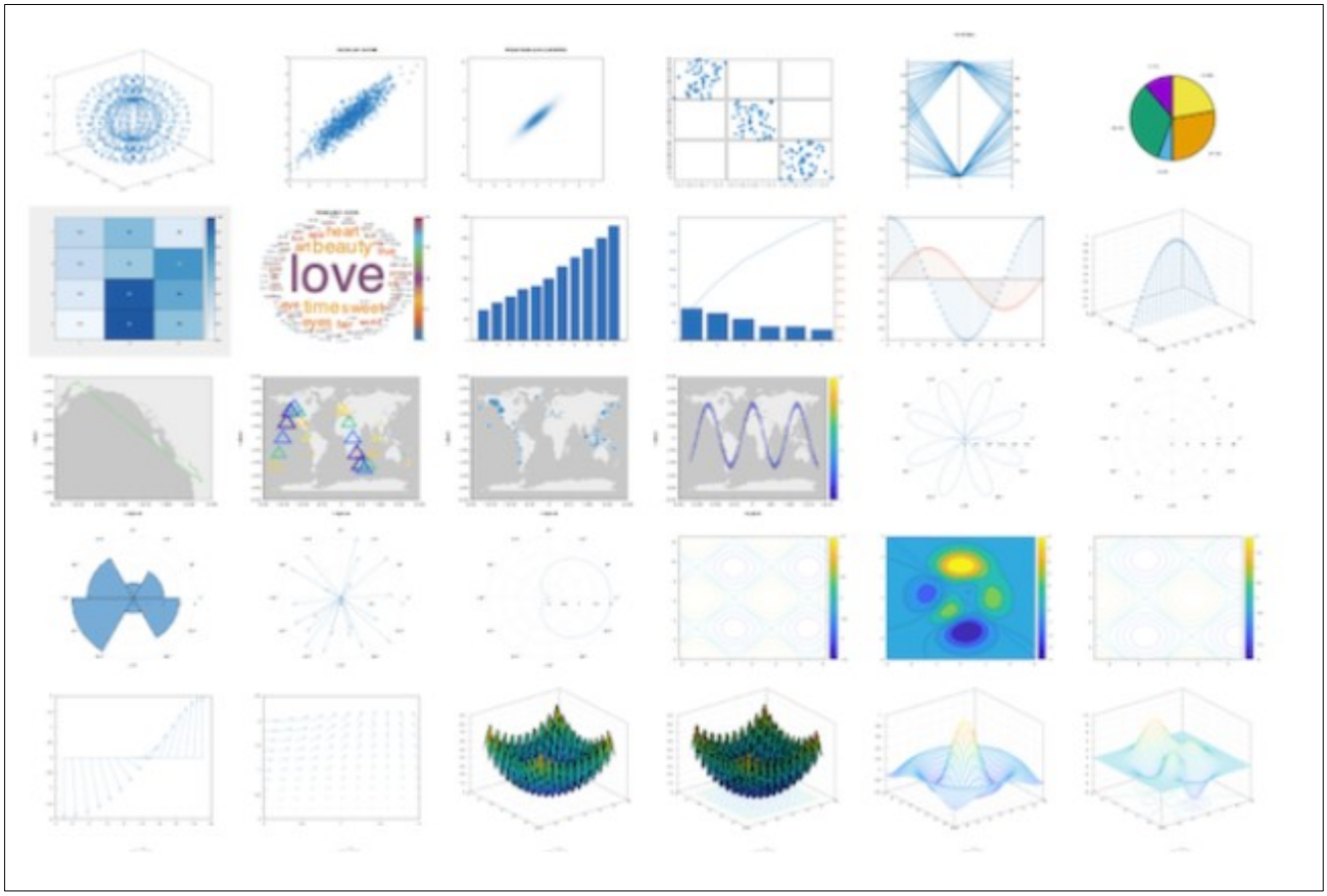

Figura 16: *Algunos de los gráficos que se pueden realizar con Matplot++*

Con las herramientas elegidas hasta el momento, toda la información está contenida dentro del software, pero una de las solicitudes de los encuestados es la conexión con un entorno exterior a la aplicación. Para esto se plantea la implementación de técnicas como la exportación de datos, la generación de informes, la visualización de contenido, la comunicación con otros usuarios y la integración con mapas.

Para la generación de informes automáticos se implementarán inicialmente dos posibilidades, PDF y Excel. Para la creación de documentos en PDF, se utilizará principalmente la librería PoDoFo, que permite la creación y lectura de archivos PDF (PoDoFo. sf). Para la generación de Excel se implementará la librería OpenXLSX, ya que ofrece las herramientas de codificación necesarias para la creación y lectura de datos de archivos con extensión XLSX (OpenXLSX, sf).

Para el procesamiento de archivos multimedia (imagen, audio y video) se utilizará Ffmpeg, una herramienta completa, que permite la manipulación multimedia. Se eligió Ffmpeg debido a su estructura dividida en diferentes componentes, facilitando la utilización de métodos especializados para objetivos en específico: (Ffmpeg, sf).

- libavutil: Contiene métodos adicionales, estructuras de datos, cadenas y operaciones matemáticas necesarias para la integración con otros componentes.

- libswscale: Se desempeña en el escalado optimizado y la conversión de formatos de pixeles.

- libswresample: Se encarga de la conversión y procesamiento de audio, la conversión de formatos, configuración de la frecuencia de audio, entre otros.

- libavcodec: contiene los diferentes códecs multimedia y las herramientas necesarias para procesar y configurar los canales de audio, video y subtítulos.

- libavformat: Soporta varios protocolos de entrada y salida para acceder a los recuros de audio y video. Además de las herramientas necesarias para mezclar canales de audio, video y subtítulos en formatos contenedores.

- libavdevice: Provee un marco de trabajo para leer y escribir canales optimizados para diferentes dispositivos de entrada y salida.

- libavfilter: Contiene herramientas genéricas para aplicar filtros de audio y video. Además de los filtros, también incluye fuentes y demás herramientas para realizar un procesamiento más profundo.

- libpostproc: Provee las herramientas necesarias para el post-procesamiento de la información. Una vez terminado el proceso aplicado, esta librería se encarga de generar la salida deseada.

Ffmpeg ofrece un marco de desarrollo completo, y su funcionalidad se extiende a tal grado que sus componentes son utilizados por empresas como Google (Ffmpeg, sf), y la NASA (Maki, N, 2020).

Para la integración con GPS se optó por Open Street Map (OSM) un proyecto que inició como una herramienta para visualizar calles pero que se ha ido extendiendo hasta el punto de albergar mapas del mundo con información específica como tiendas, hoteles, fuentes de agua y comida, rutas a tomar, árboles individuales, entre otros (Bennett J. 2010).

CAPÍTULO III: RESULTADOS DE LA INVESTIGACIÓN

RESULTADOS DEL OBJETIVO GENERAL

Para proponer la arquitectura de un software es necesario establecer escenarios que van desde la planeación hasta el diseño, para esto se realizaron consultas en bases de datos o repositorios especializados con miras a revisar los antecedentes de autores nacionales e internacionales, teniendo en cuenta algunos de los trabajos más relevantes se procedió a elaborar una encuesta para delimitar las características propias que debiese tener una aplicación Low-code/no-code para personas con y sin conocimientos de programación.

Desde los resultados obtenidos en la muestra, se presenta una propuesta para la arquitectura de un software cuya población objetivo son personas con y sin conocimientos de programación.

RESULTADOS DEL OBJETIVO ESPECÍFICO NO. 1

Realizando un análisis de la información obtenida durante la investigación y las respuestas de la encuesta realizada, se logra definir un alcance, limitaciones y necesidades del cliente:

El sistema estará orientado para plataformas de escritorio, es decir funcionará en Sistemas operativos como Windows y en diferentes distribuciones de GNU/Linux. Esto teniendo en cuenta que para una primera implementación se debe contar con un servidor remoto, lo cual conlleva a muchas dificultades de infraestructura y seguridad. Por tal razón, la primera implementación estará restringida a un alcance local

RESULTADOS DEL OBJETIVO ESPECÍFICO NO. 2

Basándose en la funcionalidad y la manera en que se operaría el software, se implementará un modelo de capas, como se define en los diagramas de arquitectura presentados en este documento, con el fin de que el usuario final pueda definir sus procesos, así como para garantizar que la información se procese en diferentes niveles y etapas.

Para el proceso de desarrollo se utilizará un modelo de metodología de desarrollo ágil, con el objetivo de presentar el software en diferentes iteraciones y prever los posibles cambios que se pueden presentar. Para ello se define la documentación de las iteraciones en las siguientes tablas:

ID	Historia de Usuario			Descripción	Nº	Criterios Aceptación		
	COMO	QUIERO	PARA			DADA	CUANDO	ENTONCES
HU1	Administrador	Administrar usuarios y sus permisos	Gestionar quién tiene acceso al sistema	*Poder gestionar la información deo los usuarios*	1	El ingreso de usuarios al sistema	no existe	se crea
					2		existe	se actualiza
					3		existe	se inactiva
					4		no tiene todos los datos	alerta solicitando los datos faltantes
					5		tiene un dato no permitido	error informando que dato no es permitido
HU2	Administrador	Establecer los módulos	Gestionar las secciones en las que el sistema debe ser divido	*Poder gestionar los módulos*	1	El ingreso de módulos al sistema	no existe	se crea
					2		existe	se actualiza
					3		existe	se inactiva
					4		No tiene todos los datos	alerta solicitando los datos faltantes
					5		tiene un dato no permitido	error informando que dato no es permitido
HU3	Administrador	Establecer los procesos	Gestionar las secuencia de acciones que se podrán realizar en cada módulo	*Poder gestionar los procesos*	1	El ingreso de procesos a cada módulo	no existe	se crea
					2		existe	se actualiza
					3		existe	se inactiva
					4		No tiene todos los datos	alerta solicitando los datos faltantes
					5		tiene un dato no permitido	error informando que dato no es permitido
HU4	Administrador	Establecer las acciones	Gestionar las operaciones que se pueden realizar en cada proceso	*Poder gestionar los acciones*	1	El ingreso de acciones a cada módulo	no existe	se crea
					2		existe	se actualiza
					3		existe	se inactiva
					4		No tiene todos los datos	alerta solicitando los datos faltantes
					5		tiene un dato no permitido	error informando que dato no es permitido
HU5	Usuario	Establecer los parámteros	Gestionar la información que almacenará el sistema	*Poder gestionar los parámetros*	1	El ingreso de parámteros al sistema	no existe	se crea
					2		existe	se actualiza
					3		existe	se inactiva
					4		No tiene todos los datos	alerta solicitando los datos faltantes
					5		tiene un dato no permitido	error informando que dato no es permitido

Tabla 1: *Historias de usuario de la aplicación.*

PRODUCT BACKLOG				Codificación	
				FOR-001	
Id	**Priodidad**	**Backlog Item**	**Estado**	**Criterios de aceptación**	**Resultado**
PP-0001	Alta	Necesito ingresar a la aplicación con usuario y contraseña	Pendiente	El usuario ingresa, inicia sesión, accederá a la aplicación	El sistema debe validar si el usuario se encuentra registrado, Una vez haga esto le permitirá el acceso a la aplicación.
				El usuario no ingresa a la aplicación, porque no se encuentra registrado	El sistema debe vaidar si el usuario se encuentra registrado, de lo contrario debe generar una alerta solicitando verificar el usuario y/o la contraseña.
PP-0002	Alta	Necesito ingresar información de los usuarios	Pendiente	El administrador ingresa los datos del usuario	El sistema valida los datos e ingresa el nuevo usuario funcional al sistema
				El administrador ingresa un dato inválido	El sistema valida el dato y muestra un mensaje de error
				El administrador no ingresa todos los datos	El sistema valida los datos faltantes y muestra un mensaje de error
PP-0003	Alta	Necesito ingresar los módulos al sistema	Pendiente	El administrador ingresa los datos del módulo	El sistema crea el nuevo módulo
				El administrador ingresa un dato inválido	El sistema valida el dato y muestra un mensaje de error
				El administrador no ingresa todos los datos	El sistema valida los datos faltantes y muestra un mensaje de error
PP-0004	Alta	Necesito ingresar los procesos al sistema	Pendiente	El administrador ingresa los datos del proceso	El sistema crea el nuevo proceso asociado a un módulo
				El administrador ingresa un dato inválido	El sistema valida el dato y muestra un mensaje de error
				El administrador no ingresa todos los datos	El sistema valida los datos faltantes y muestra un mensaje de error
PP-0005	Alta	Necesito ingresar las acciones al sistema	Pendiente	El administrador ingresa los datos de la acción	El sistema crea la nueva acción asociada a un proceso
				El administrador ingresa un dato inválido	El sistema valida el dato y muestra un mensaje de error
				El administrador no ingresa todos los datos	El sistema valida los datos faltantes y muestra un mensaje de error
PP-0006	Alta	Necesito ingresar los parámetros al sistema	Pendiente	El administrador ingresa los datos del parámetro	El sistema crea el nuevo parámetro
				El administrador ingresa un dato inválido	El sistema valida el dato y muestra un mensaje de error
				El administrador no ingresa todos los datos	El sistema valida los datos faltantes y muestra un mensaje de error
Creado por:		**Juan Parra**		**Fecha: 2022-05-07**	
				Versión: 01	

Tabla 2: *Backlog de la aplicación.*

Identificador (ID) de item de product backlog	Enunciado del item de Product Backlog	Tarea	Dueño / Voluntario	Estatus	Horas estimadas totales
PP-0001	Como Usuario necesito ingresar al aplicativo con usuario y contraseña para acceder a los servicos ofrecidos en el sistema	Crear Base de Datos	Juan	Pendiente	4
		Crear tabla de usuarios	Juan	Pendiente	4
		Interfaz gráfica	Juan	Pendiente	8
		Proceso de login	Juan	Pendiente	8
		Implementar la arquitectrura	Juan	Pendiente	6
		Crear conexión	Juan	Pendiente	4
		Pruebas de funcionamiento	Juan	Pendiente	4
		Pruebas de seguridad	Juan	Pendiente	2
		Pruebas de UX	Juan	Pendiente	1
		Pasar a producción	Juan	Pendiente	1
PP-0002	Como administrador necesito ingresar información de los usuarios	Interfaz gráfica	Juan	Pendiente	4
		CRUD con usuarios	Juan	Pendiente	8
		Implementar la arquitectura	Juan	Pendiente	4
		Pruebas conexión BD	Juan	Pendiente	2
		Pruebas de integración con Login	Juan	Pendiente	2
		Pruebas de funcionamiento	Juan	Pendiente	2
		Pruebas de integridad de información	Juan	Pendiente	2
		Pruebas de seguridad	Juan	Pendiente	2
		Pruebas de UX	Juan	Pendiente	2
		Pasar a producción	Juan	Pendiente	2
PP-0003	Como administrador necesito ingresar información de los módulos	Crear tabla de módulos	Juan	Pendiente	4
		Interfaz gráfica	Juan	Pendiente	4
		CRUD con módulos	Juan	Pendiente	8
		Generación de tablas necesarias	Juan	Pendiente	8
		Implementar la arquitectrura	Juan	Pendiente	6
		Crear conexión	Juan	Pendiente	1
		Pruebas unitarias	Juan	Pendiente	2
		Pruebas de seguridad	Juan	Pendiente	2
		Pruebas de UX	Juan	Pendiente	2
		Pasar a producción	Juan	Pendiente	2

PP-0004	Como administrador necesito ingresar información de los procesos	Crear tablas de procesos	Juan	Pendiente	4
		Interfaz gráfica	Juan	Pendiente	4
		CRUD con procesos	Juan	Pendiente	8
		Implementar la arquitectrura	Juan	Pendiente	8
		Crear conexión	Juan	Pendiente	6
		Pruebas unitarias	Juan	Pendiente	1
		Pruebas de integración	Juan	Pendiente	2
		Pruebas de seguridad	Juan	Pendiente	2
		Pruebas de UX	Juan	Pendiente	2
		Pasar a producción	Juan	Pendiente	2
PP-0005	Como administrador necesito ingresar información de las acciones al sistema	Crear tablas de acciones	Juan	Pendiente	4
		Interfaz gráfica	Juan	Pendiente	4
		CRUD de acciones	Juan	Pendiente	8
		Implementar la arquitectrura	Juan	Pendiente	8
		Crear conexión	Juan	Pendiente	6
		Pruebas unitarias	Juan	Pendiente	1
		Pruebas de integración	Juan	Pendiente	2
		Pruebas de seguridad	Juan	Pendiente	2
		Pruebas de UX	Juan	Pendiente	2
		Pasar a producción	Juan	Pendiente	2
PP-0006	Como usuario necesito ingresar información de los parámetros para utilizarse en diferentes módulos del sistema	Crear tablas de parámetros	Juan	Pendiente	4
		Interfaz gráfica	Juan	Pendiente	4
		CRUD con paramétros	Juan	Pendiente	8
		Implementar la arquitectrura	Juan	Pendiente	8
		Crear conexión	Juan	Pendiente	6
		Crear conexión	Juan	Pendiente	1
		Pruebas unitarias	Juan	Pendiente	2
		Pruebas de integración	Juan	Pendiente	2
		Pruebas de UX	Juan	Pendiente	2
		Pasar a producción	Juan	Pendiente	2

Tabla 3: *Definición de los sprints de la aplicación.*

RESULTADOS DEL OBJETIVO ESPECÍFICO NO. 3

Ya que un software que esté alineado con las tecnologías Low-code/No-code presenta muchas opciones para los usuarios, se utilizan herramientas que se acoplen al proceso de desarrollo ofreciendo diferentes posibilidades como generación de gráficos para visualizar los datos, procesamiento multimedia, generación de documentos en PDF, integración con GPS, entre otros.

La función principal del software propuesto es entender los requerimientos del usuario y convertirlos en código para que un sistema operativo sea capaz de reconocerlo y ejecutarlo, pero también se tendrán en cuenta diferentes características como la generación de informes o procesamiento multimedia. Para ello, su utilizarán librerías externas al lenguaje de programación elegido, ya que como menciona Tratt (2008) la mayoría de lenguajes de programación modernos, como C++ solo permiten la expansión por medio de librerías. De esta manera, es posible distribuir las funcionalidades del software en módulos y enfocar el desarrollo en la función principal: la generación de código.

Objetivo	Opción principal	Opción alternativa
Sistema operativo	GNU / Linux	Windows
Lenguaje de programación	C++, Bash, Shell, SQL	C, Java, NoSQL
Interfaz gráfica	GTK (con gtkmm)	Tauri, Electron, QT
Compilación	gcc, makefile	g++
Base de datos	SQLite	MySQL, UnQLite
Servidor de base de datos	SQLite	WAMPP / LAMPP
API / conexión con la base de datos	SQLite, C++	PHP
Visualización de datos	Matplot++	xChart
Generación / edición de PDFs	PoDoFo	LitePDF, DynaPDF
Hojas de cálculo / Excel	OpenXLSX, xInt	Xlslib
Multimedia	Ffmpeg	mpv-lib, Vireo
GPS	OpenStreetMap	

Tabla 4: *Herramientas a utilizar para desarrollar el aplicativo*

Algunas de estas herramientas están sujetas a cambios al ser este un software que se presenta en términos de las necesidades de los diferentes usuarios. Por esta razón se presentan opciones principales y opciones alternativas. En caso de que la opción principal no cumpla con los objetivos planteados, se integrará o se reemplazará con la opción alternativa.

CAPÍTULO V: CONCLUSIONES Y RECOMENDACIONES

Luego del análisis de toda la información recolectada, fue posible definir un modelo, restricciones, herramientas y metodologías necesarias para la planeación de un software que le permita a los usuarios programar sin la necesidad de escribir código.

El alcance del sistema se redujo a un entorno local, pero con la idea de escalabilidad en mente para que en caso de querer expandir las funcionalidades y fronteras del sistema no se tenga que rehacer una implementación total.

Las herramientas a utilizar fueron elegidas teniendo en cuenta las necesidades de los usuarios encuestados, por eso se utilizan herramientas como C++, porque pese a su complejidad, ofrece muchas posibilidades, con un rendimiento favorable frente a otros lenguajes.

También es necesario notar que un software para crear sistemas de información completamente parametrizables es un concepto que se puede abordar de diferentes formas, es por tal razón que se limita a una aplicación de escritorio, pero en el caso de ser viable, podría ser posible redefinir los requerimientos y extender el alcance a entornos Web y Móviles.

BIBLIOGRAFÍA

Ballhausen, M. (2019). Free and open source software licenses explained. *Computer*, *52*(6), 82-86.

Bennett, J. (2010). *OpenStreetMap*. Packt Publishing Ltd.

Castro, J. W., Llerena, L., Rodríguez, N., & Acuña, S. T. (2017). Adoption of the focus groups technique in the open source software development process.

Cesar, I., Fertalj, K., & Batoš, V. (2014, June). Towards a method to retrieving business process model from source code. In 2014 9th Iberian Conference on Information Systems and Technologies (CISTI) (pp. 1-6). IEEE.

Cohen, D., Lindvall, M., & Costa, P. (2003). Agile software development. *DACS SOAR Report*, *11*, 2003.

Chakravarty, M. M., Steadman, P., Van Eede, M. C., Calcott, R. D., Gu, V., Shaw, P., ... & Lerch, J. P. (2013). Performing label-fusion-based segmentation using multiple automatically generated templates. *Human brain mapping*, *34*(10), 2635-2654.

de Oliveira, B. C., & Zuchi, J. D. (2020). EFFICIENCY IN WRITING SOFTWARE WITH VIM. *Revista Interface Tecnológica*, *17*(2), 386-397.

FFmpeg. (s.f.). *FFmpeg*. Recuperado el 1 de Julio, 2022, de https://ffmpeg.org/

FFmpeg. (s.f.). *Projects*. Recuperado el 1 de Julio, 2022, de https://trac.ffmpeg.org/wiki/Projects

Friedel, M., van Eede, M. C., Pipitone, J., Chakravarty, M. M., & Lerch, J. P. (2014). Pydpiper: a flexible toolkit for constructing novel registration pipelines. Frontiers in neuroinformatics, 8, 67.

Fontela, C. (2012). *UML: modelado de software para profesionales*. Alpha Editorial.

Gafurov, D., Hurum, A. E., & Markman, M. (2018, September). Achieving test automation with testers without coding skills: an industrial report. In 2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE) (pp. 749-756). IEEE.

GNU. (2020, January 19). *Make - GNU Project - Free Software Foundation*. Make. Recuperado el 1 de Julio, 2022, de https://www.gnu.org/software/make/

GTK. (2012, February 27). *COPYING · main · GNOME / gtk ·*. GitLab. Recuperado el 1 de Julio, 2022, de https://gitlab.gnome.org/GNOME/gtk/-/blob/main/COPYING

Iivari, N. (2009). User Participation in'Configuring the User'in OSS Development. ICIS 2009 Proceedings, 199.

Jafer, S., Chhaya, B., & Durak, U. (2017). Graphical specification of flight scenarios with aviation scenario defintion language (ASDL). In AIAA modeling and simulation technologies conference (p. 1311).

Kintone. (2017). The Rise of The Empowered Citizen Developer. https://resources.kintone.com/citizen-developer-business-application-report-2017

Llerena, R., Rodríguez, N., Llerena, L., Castro, J. W., & Acuña, S. T. (2020, July). Adoption of the HTA Technique in the Open Source Software Development Process. In International Conference on Human-Computer Interaction (pp. 184-198). Springer, Cham.

Llerena, L., Rodríguez, N., Castro, J. W., & Acuña, S. T. (2016, October). Adoption of the user profiles technique in the open source software development process. In International Conference on Software Process Improvement (pp. 201-210). Springer, Cham.

Llerena, L., Rodríguez, N., Sacca, G., Castro, J. W., & Acuña, S. T. (2016, September). Adoption of the persons technique in the open-source software development process. In Proceedings of the XVII International Conference on Human Computer Interaction (pp. 1-4).

Llerena, L., Rodriguez, N., Castro, J. W., & Acuña, S. T. (2018). How to Incorporate a Usability Technique in the Open-Source Software Development Process. In SEKE (pp. 182-181).

Llerena, L., Rodriguez, N., Castro, J. W., & Acuña, S. T. (2019). Adapting usability techniques for application in open-source software: A multiple case study. Information and Software Technology, 107, 48-64.

Maki, J. N. (2020, November 24). *The Mars 2020 Engineering Cameras and Microphone on the Perseverance Rover: A Next-Generation Imaging System for Mars Exploration*. SpringerLink. Recuperado el 1 de Julio, 2022, de https://link.springer.com/article/10.1007/s11214-020-00765-9

Matplot++. (s.f.). *Home - Matplot++*. Recuperado el 1 de Julio, 2022, de https://alandefreitas.github.io/matplotplusplus/

Martin, A., Biddle, R., & Noble, J. (2004, June). The XP customer role in practice: three studies. In Agile development conference (pp. 42-54). IEEE.

Mendix. (2020). Forrester: Total Economic Impact (TEI) of the Mendix Platform. https://www.mendix.com/resources/forrester-total-economic-impact-of-the-mendix-platform/

M. Oltrogge et al., "The Rise of the Citizen Developer: Assessing the Security Impact of Online App Generators," 2018 IEEE Symposium on Security and Privacy (SP), 2018, pp. 634-647, doi: 10.1109/SP.2018.00005.

OpenXLSX. (s.f.). *GitHub - troldal/OpenXLSX: A C++ library for reading, writing, creating and modifying Microsoft Excel® (.xlsx) files.* GitHub. Recuperado el 1 de Julio, 2022, de https://github.com/troldal/OpenXLSX

Oracle. (2015). *Chapter 1. Introduction*. Java Virtual Machine Specification. Recuperado el 1 de Julio, 2022, de https://docs.oracle.com/javase/specs/jvms/se8/html/jvms-1.html#jvms-1.2

Pérez-Colado, V. M., Pérez-Colado, I. J., Freire-Morán, M., Martínez-Ortiz, I., & Fernández-Manjón, B. (2019, July). UAdventure: Simplifying narrative serious games development. In 2019 IEEE 19th International Conference on Advanced Learning Technologies (ICALT) (Vol. 2161, pp. 119-123). IEEE.

PoDoFo. (s.f.). *PoDoFo*. Recuperado el 1 de Julio, 2022, de http://podofo.sourceforge.net/index.html

QT. (n.d.). *Qt Commercial License | Qt 5.15*. R Recuperado el 1 de Julio, 2022, de https://doc.qt.io/qt-5/commerciallicense.html

Raza, A., Capretz, L. F., & Ahmed, F. (2011). An empirical study of open source software usability: The industrial perspective. International Journal of Open Source Software and Processes (IJOSSP), 3(1), 1-16.

Team, T. G. (s.f.). *The GTK Project - A free and open-source cross-platform widget toolkit*. The GTK Team. Recuperado el 1 de Julio, 2022, de https://www.gtk.org/docs/language-bindings/

Tina Beranic, T. B., Patrik Rek, P. R., & Marjan Hericko, M. H. (2022). Adoption and Usability of Low-Code/No-Code Development Tools. https://www.proquest.com/openview/a6e9a1210ef714ead2f9695c6a71fb6f/1?pq-origsite=gscholar&cbl=1986354

Tratt, L. (2008). Domain specific language implementation via compile-time meta-programming. ACM Transactions on Programming Languages and Systems (TOPLAS), 30(6), 1-40.

Sowe, S. K., Stamelos, I., & Angelis, L. (2008). Understanding knowledge sharing activities in free/open source software projects: An empirical study. Journal of Systems and Software, 81(3), 431-446.

SQLite Home Page. (s.f.). SQLite. Recuperado el 1 de Julio, 2022, de https://www.sqlite.org/index.html

Stack Overflow (2022). Stack Overflow Developer Survey

Stevanovic, M. (2014). *Advanced C and C++ compiling*. Apress.

Xie, J., Zheng, Q., Zhou, M., & Mockus, A. (2014, May). Product assignment recommender. In Companion Proceedings of the 36th International Conference on Software Engineering (pp. 556-559).

Xie, J., Zhou, M., & Mockus, A. (2013, October). Impact of triage: a study of mozilla and gnome. In *2013 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement* (pp. 247-250). IEEE.