PROTOTIPO DE SOFTWARE PARA LA CREACIÓN DE AUTOMATIZACIÓN ROBÓTICA DE PROCESOS – RPA ORIENTADA A SOFTWARE CONTABLES PARA ORGANIZACIONES DEL SECTOR PÚBLICO

GLORIA ZULAY CÁCERES GRANADOS
FANNY QUINTERO PARRA

UNIVERSIDAD AUTÓNOMA DE BUCARAMANGA – UNAB
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS
PROYECTO DE GRADO
GRUPO DE INVESTIGACIÓN EN TECNOLOGÍAS DE INFORMACIÓN
LÍNEA DE INVESTIGACIÓN EN TELEMÁTICA
BUCARAMANGA, MAYO 08 DE 2020

PROTOTIPO DE SOFTWARE PARA LA CREACIÓN DE AUTOMATIZACIÓN ROBÓTICA DE PROCESOS – RPA ORIENTADA A SOFTWARE CONTABLES PARA ORGANIZACIONES DEL SECTOR PÚBLICO

GLORIA ZULAY CÁCERES GRANADOS
FANNY QUINTERO PARRA

Trabajo de Grado para optar por el título de Ingeniero de Sistemas

DIRECTOR:
Johan Smith Rueda Rueda

CO-DIRECTOR
José David Ortiz Cuadros

UNIVERSIDAD AUTÓNOMA DE BUCARAMANGA – UNAB
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS
PROYECTO DE GRADO
GRUPO DE INVESTIGACIÓN EN TECNOLOGÍAS DE INFORMACIÓN
LÍNEA DE INVESTIGACIÓN EN TELEMÁTICA
BUCARAMANGA, MAYO 08 DE 2020

TABLA DE CONTENIDO

PROTOTIPO DE SOFTWARE PARA LA CREACIÓN DE AUTOMATIZACIÓN ROBÓTICA DE PROCESOS – RPA ORIENTADA A SOFTWARE CONTABLES PARA ORGANIZACIONES DEL SECTOR PÚBLICO 2
1. INTRODUCCIÓN 8
2. PLANTEAMIENTO DEL PROBLEMA 10
3. JUSTIFICACIÓN 12
4. OBJETIVOS 14
4.1 OBJETIVO GENERAL 14
4.2. OBJETIVOS ESPECÍFICOS 14
5. MARCO REFERENCIAL 15
5.1 MARCO CONCEPTUAL 15
5.2 MARCO TEÓRICO 20
5.3 ESTADO DEL ARTE 35
5.4 MARCO LEGAL 41
6. METODOLOGÍA 42
7. RESULTADOS 45
7.1 CARACTERIZACIÓN 45
7.2 SELECCIÓN DE TECNOLOGÍAS Y MÉTODOS 54
7.3 COMPONENTE DE VISIÓN COMPUTACIONAL 64
7.4 PROTOTIPO SOFTWARE FUNCIONAL 69
7.5 INFORME COMPARATIVO DEL RENDIMIENTO DE PROCESOS CON SOFTWARE Y SIN SOFTWARE. 86
8. CONCLUSIONES Y TRABAJO A FUTURO 92
9. REFERENCIAS 94

LISTADO DE TABLAS

Tabla 1 Criterios de búsqueda 35
Tabla 2 Trabajos recopilados. 36
Tabla 3 Actividades y entregables del proyecto. 44
Tabla 4 Criterios para selección de algoritmo de detección de objetos. 63
Tabla 5 Criterios de selección para framework de RPA. 63
Tabla 6 Criterios para selección entorno de desarrollo. 64
Tabla 7 Tecnologías y métodos seleccionados para el desarrollo del prototipo software. 64
Tabla 8 Intervalo de tiempo de desarrollo de la RPA manual y automatizada. 89
Tabla 9 Llenado de facturas manualmente. 89
Tabla 10 Llenado de facturas de manera automática. 90
Tabla 11 Tiempo de llenado de factura de manera manual y automatizada. 90

LISTADO DE FIGURAS

Figura 1 Arquitectura básica de una CNN con dos capas convolucionales y tres clasificaciones. 21
Figura 2 Pasos de la Visión Computacional. 23
Figura 3 Operador de Sobel. 24
Figura 4 Detección de objetos. 26
Figura 5 Fases del Modelo Incremental. 28
Figura 6 Modelo CRISP-DM. 29
Figura 7 Automatización de Procesos de Robótica. 31
Figura 8 Diagrama de Procesos. 32
Figura 9 Diagrama de aplicación. 33
Figura 10 Diagrama de Robot. 33
Figura 11 Diagrama de interacción. 34
Figura 12 Sector Empresarial en el que labora. 46
Figura 13 Tipo de Empresa. 46
Figura 14 Sector económico de la empresa. 47
Figura 15 Tipo de licencia del software. 47
Figura 16 Sistema de ejecución del software contable. 48
Figura 17 Sistema operativo compatible al software contable. 48
Figura 18 Software contable de la empresa. 49
Figura 19 Errores Contables. 49
Figura 20 Tipos de errores contables. 50
Figura 21 Tiempo de contabilización de una factura. 50
Figura 22 Módulos del software contable. 51
Figura 23 Diagrama de proceso de facturas. 51
Figura 24 Diagrama subproceso recepción de facturas. 52
Figura 25 Diagrama Subproceso creación recibo de caja. 52
Figura 26 Diagrama de procesamiento de facturas automatizado. 53
Figura 27 Lista de Frames. 64
Figura 28 Etiquetador LabelIng. 65
Figura 29 Selección de región de imágenes. 65
Figura 30 Coordenadas de un frame. 66
Figura 31 Entrenamiento del algoritmo YOLO. 67
Figura 32 Uso de inspect para encerrar elementos en un cuadro de color amarillo. 69
Figura 33 Conexión a escritorio remoto. 70
Figura 34 Imagen binaria obtenida después de aplicar detección de color amarillo. 70
Figura 35 Detección de color verde. 71
Figura 36 Coordenadas detectadas. 71
Figura 37 Recortes de las imágenes. 72

Figura 38 Datos obtenidos al realizar el barrido con la librería pytesseracy de python en formato .csv. 72
Figura 39 Estructura de conversación del chatbot. 73
Figura 40 Imagen del script generado de la RPA en el lenguaje de robot framework desde el chatbot. 73
Figura 41 Diseño de la interfaz gráfica del chatbot. 74
Figura 42 Inicio de conversación con el chatbot. 75
Figura 43 Eliminación de coordenadas duplicadas y validación de los elementos GUI. 75
Figura 44 Lanzamiento de la RPA. 76
Figura 45 Chatbot con Interfaz gráfica de usuario final. 76
Figura 46 Reconocimiento de color amarillo y encierro de la ubicación del elemento. 77
Figura 47 Seleccionando elementos GUI. 78
Figura 48 Errores de la RPA al momento de acceder a los controles para ingresar los datos. 78
Figura 49 Ingresando datos al software. 79
Figura 50 Ingreso de datos en el mismo control y datos erróneos. 80
Figura 51 Ingresando datos en cada control. 80
Figura 52 Prueba, falla de ingreso de datos en controles seleccionados. 81
Figura 53 Se ingresan todos los datos al software contable. 81
Figura 54 La RPA seleccionó los controles utilizados. 82
Figura 55 Estadísticas de las pruebas y estado del proceso. 83
Figura 56 Controles a los que accedió los datos que ingreso. 84
Figura 57 Gráfica proceso de llenado de facturas manual y automatizado. 88

1. INTRODUCCIÓN

El profesional contable, dentro de sus funciones al interior de una organización, realiza un alto número de tareas repetitivas. La contabilidad es el proceso que permite identificar, clasificar, registrar, resumir, interpretar, analizar y evaluar, en términos monetarios, las operaciones y transacciones de una empresa (Díaz Moreno, 2006). Asimismo, su principal objetivo es informar razonable y objetivamente la situación de la compañía, su desempeño o resultados obtenidos por el desarrollo de su actividad y las causas de esos resultados; para ello, se realiza la consolidación del registro de sus operaciones que se hace en los estados financieros (FAEDIS, 2018).

Con el fin de suministrar información sobre el negocio, se necesita estructurar los procesos de registro de las operaciones y actividades de la empresa o negocio, ya sea por medio de procedimientos manuales, mecánicos o electrónicos. Toda la actividad económica se manifiesta con transacciones comerciales, como compra de inventarios, pagos de nómina, venta de productos, cancelación de obligaciones, entre otros (Díaz Moreno, 2006).

Según el artículo 354 de la Constitución Política de Colombia, la persona encargada de llevar a cabo la contabilidad del país recibe el nombre de contador general de la nación, el cual lleva a cabo funcionar tales como uniformar, centralizar y consolidar la contabilidad pública, elaborar el balance general y determinar las normas contables que deben regir en el país, conforme a la ley (Leyes desde 1992 - Vigencia expresa y control de constitucionalidad [DECRETO_2811_1974], 2006).

Las tecnologías de la información han contribuido a facilitar la realización de estas tareas, y en muchos casos, disminuir la intervención humana; especialmente, en las tareas repetitivas a través de la automatización de procesos. Una de las tecnologías es la RPA –por sus siglas en inglés, *Robotic Process Automation*–, un software RPA que permite la automatización de tareas repetitivas como el procesamiento de formularios o la entrada de datos, por nombrar algunos ejemplos.

En el mercado existen varias soluciones basadas en RPA, aunque en el mercado existen soluciones basadas en RPA, estas pueden llegar a presentar dificultades en su incorporación en los procesos organizacionales. Algunas de las dificultades que se presentan son: (i) herramientas solo permiten acciones específicas y predeterminadas; (ii) si el software en el que se está automatizando presenta cambios en la interfaz gráfica con las actualizaciones del software contable, o se decida cambiar de software contable, se debe volver a desarrollar la RPA. Esto implicaría iniciar un nuevo proceso de construcción, lo que implica un incremento de costos y tiempo para la organización.

Por las razones anteriormente citadas, nace la idea del diseño de un prototipo software, basado en visión computacional para la generación y creación de RPA, que permitan la automatización en el proceso de registro contable en el sistema. Asimismo, brinda una manera sencilla y fácil de manejo para un usuario sin conocimientos básicos en programación y así, reducir los costos de tiempo y dinero en el proceso de la programación de una nueva RPA.

El presente documento corresponde a un proyecto de grado del Programa de Ingeniería de Sistemas. Este proyecto se pretende desarrollar en cuatro fases, a saber: en la primera, se caracterizará el proceso y herramientas necesarias para realizar el registro contable en una organización. En la segunda fase, luego de una revisión de literatura, se seleccionarán las tecnologías y métodos que se usarán para el desarrollo del prototipo. En la tercera fase, se construirá el componente de visión computacional que permitirá identificar los componentes de interfaz de usuario de los softwares contables. Finalmente, se desarrollará el prototipo software para automatizar la creación de RPA.

2. PLANTEAMIENTO DEL PROBLEMA

El proceso contable es el proceso mediante el cual las transacciones de una empresa son registradas y clasificadas para luego, elaborar los estados financieros. El proceso contable se lleva a cabo a partir de una serie de paso en donde sobresalen los siguientes: realizar las transacciones en el diario general, pasar la información del diario general al mayor general, obtener la balanza de comprobación, registrar los asientos de ajuste, obtener la balanza de comprobación ajustada, formular los Estados financieros, hacer los asientos de cierre y obtener la balanza de comprobación después del cierre (Alarcón, 2014). A partir de este proceso, se logra la obtención del ciclo contable de la empresa, con el fin de generar información contable confiable y precisa (Alarcón, 2014). En ese mismo sentido, las instituciones públicas, para la elaboración y presentación de los estados financieros deben llevar sus registros contables bajo el marco normativo para las entidades del gobierno; el cual es aplicable a las entidades de gobierno que se encuentran bajo el ámbito del Régimen de Contabilidad Pública (CGN, 2014).

El ejercicio de ingresar información a un software, sean contable o no, es una tarea repetitiva, en la cual se invierte un tiempo considerable para extraer los datos de los soportes físicos; considerando también, que en el proceso se puedan incurrir en errores en la trascripción o entrada de datos manual. Dentro de los errores de entrada de datos manual se encuentran los siguientes: errores de transcripción, que se producen cuando se ingresa de manera incorrecta; errores de transposición, el cual ocurre cuando la información se ingresa en el orden incorrecto; y los errores de sustitución, omisión e inserción (Dhakal et al., 2018; Identigate, 2018.; scannmore, 2018; ThinkAutomation, 2018). Este tipo de errores afectan a organizaciones de todas las áreas y sectores. El área de la salud, la transferencia de datos desde instrumentos de laboratorio a sistemas de información de laboratorio y registros de salud electrónicos; en el área de la investigación en el análisis estadístico, en el que errores pueden llegar a arruinar los resultados estadísticos y las conclusiones (Barchard & Pace, 2011; Mays & Mathias, 2019), y la contabilidad.

En el caso particular de la contabilidad, los errores en la trascripción de datos pueden generar errores contables. Entre los errores que se pueden presentar en los documentos financieros, que describen una discrepancia o diferencia no fraudulenta, se encuentran los siguientes: el error por omisión, que hace referencia a la transacción financiera que no se registra por error; el error por comisión, que es cuando un registro de transacción se realiza por el valor incorrecto en la cuenta correcta; error de principio, que consiste en el valor correcto de la entrada pero colocado incorrectamente; el error de transposición, que ocurre cuando dos o más dígitos se invierten, o transponen, individualmente o como parte de una secuencia más grande; y la reversión de entradas, que puede ocurrir cuando las entradas contables se invierten por completo, por lo que las entradas se cargan en una cuenta y se acreditan en la otra (Luenendonk, 2017). Las causas anteriormente citadas, se traducen en procesos ineficientes dentro de la empresa y costos financieros importante, porque implica invertir recursos de tiempo para rectificar los registros

financieros, así como emitir una declaración de culpa y liberar las entradas correctas (LuenenDonk, 2017).

Para contribuir con la automatización de tareas repetitivas y reducir la tasa de errores en el proceso de entrada de datos, han surgido soluciones tecnológicas como la Automatización Robótica de Procesos —RPA, por sus siglas en inglés, *Robotic Process Automation*—. Las RPA permiten a las empresas, automatizar determinadas tareas repetitivas de forma más eficiente (IBM Robotic Process Automation, 2019).

La implementación de RPA para la automatización de tareas por parte de una organización, representa ventajas como: reducción de costos, lo que representa un escenario muy viable comparadas con dichas tareas realizadas por un ser humano; optimización de tiempos, ya que permite que los procesos sean completados en menor tiempo; soporte 24/7, para brindar una mejor experiencia y satisfacción al cliente; reducción de riesgo operacional, reproducción de tareas sin errores, ya que al ser manipulado por una máquina, está no presentará cansancio o errores debido a falta de conocimiento; y procesos internos optimizados, esto debido a que la RPA puede ser adaptada a un sistema existente, permitiendo la optimización en la gestión de talento humano, permitiendo que los empleados puedan invertir su tiempo en otras tareas relacionadas a su campo de acción, lo que generará mayor rendimiento y desempeño de sus capacidades (ISOL, 2019).

Para la automatización de procesos a través de RPA se debe considerar actividades como: (i) análisis de las tareas a automatizar y revisar si los parámetros de estas actividades están dentro las acciones de la herramienta de RPA; (ii) registro de las acciones en la herramienta de RPA para su desarrollo; (iii) realizar los cambios y ajustes necesarios para refinar el script de la RPA; y (iv) se realizan pruebas periódicas para evaluar su funcionamiento. Sin embargo, el tiempo de desarrollo de una RPA puede variar dependiendo de las tareas a automatizar. Una tarea simple puede llegar a durar días o semanas, y no necesitaría conocimientos avanzados de programación, pero sí algunos conceptos básicos por parte del usuario. Al contrario, cuando se trata de tareas más complejas su implementación puede llegar a durar semanas y meses, y requiere apoyo de un equipo especializado sobre esta área (BMind Licencias, 2019).

Para automatizar el ingreso de información al software contable se debe diseñar una RPA para dicho software. Si el software al que se le ha realizado el ingreso de datos a través de RPA presenta cambios en su interfaz gráfica, o la empresa decide migrar a otro software para realizar dicha tarea o proceso, incurriría en tener que modificar la RPA ya desarrollada o tener que desarrollar una nueva RPA, lo que conlleva costos adicionales para la organización. Además, del costo económico que representa estas herramientas de los cuales sus licencias están entre USD$990 o USD$3990 (Capterra, 2019).

Lo anterior da origen a la siguiente pregunta de investigación: ¿De qué manera se puede lograr optimizar la creación de RPA por medio de visión computacional para la automatización de los procesos contables en las empresas públicas?

3. JUSTIFICACIÓN

El presente proyecto se enfoca en desarrollar una solución que permita la automatización de RPA, y esta tecnología a su vez, se implemente para automatización de tareas repetitivas en el área contable. Esta tecnología aprovecha la combinación de interfaz de usuario (UI) y características de nivel de superficie para crear scripts que automatizan el trabajo rutinario y predecible de la transcripción de datos. Por lo tanto, los principales casos de uso para las soluciones RPA se basan en la integración de datos. Según una encuesta de Gartner 2019, alrededor del 80% de los líderes financieros han implementado RPA o planean implementar. Aun así, la adopción de nuevas tecnologías digitales y en la nube sigue siendo una hazaña desafiante (Ray et al., 2019). La implementación de esta tecnología ha sido más alta en las industrias bancarias y de seguros. Este sector ha liderado en RPA para operaciones de procesos y los departamentos financieros y de contabilidad.

RPA sigue siendo un mercado relativamente pequeño con un ingreso total de poco menos de $ 850 millones en 2018. Sin embargo, RPA es el subsegmento de software de más rápido crecimiento rastreado oficialmente por Gartner, con crecimiento interanual de más del 63% en 2018 (Ray et al., 2019). Algunos casos de uso de RPA, podemos encontrar Bancolombia una institución financiera que necesitaba desarrollar una fuerza de trabajo virtual que combinara capacidades humanas, robóticas, cognitivas y de análisis para mejorar la experiencia de sus clientes bancarios, automatizar las tareas repetitivas y aumentar la eficacia de manera general. Con los robots de *Automation Anywhere* empresa de RPA, Bancolombia ordena los datos estructurados, semiestructurados y sin estructura para transformar su administración de procesos empresariales – BPM. Los robots, también llamados *bots*, automatizan cientos de procesos y aumentan en gran medida la eficacia de la administración (Automation Anywhere, 2020).

Hitachi Vantara quería aprovechar la automatización robótica de procesos – RPA no solo para mejorar la calidad y confiabilidad de sus procesos, sino también con el objetivo principal de hacer que los miembros del personal dejaran de realizar tareas rutinarias y de bajo valor. Hitachi Vantara decidió implementar IQ Bot, la oferta cognitiva de inteligencia artificial – IA que permite extraer datos semiestructurados de documentos y procesarlos sin esfuerzo (Automation Anywhere, 2020).

Estas tecnologías personalizan y contextualizan la interacción humano-tecnología, permitiendo a las empresas proporcionar información y servicios personalizados basados en el lenguaje y la imagen, con una participación humana mínima o nula (Deloitte, 2020).

Según el estudio *10 strategic technology trends for 2020* publicado por Gartner, una de las tendencias actuales es la hiper automatización, un nuevo concepto que va más allá de la automatización o el uso de tecnología para ejecutar tareas manuales repetitivas que hacen las personas. La hiper-automatización se ocupa de combinar RPA y la aplicación de tecnologías avanzadas, incluida la inteligencia artificial (IA) y el aprendizaje automático (ML), para automatizar cada vez más los procesos complejos y aumentar las capacidades de los humanos (Burke et al., 2020).

La hiper-automatización, una disciplina que una vez incorporada en el negocio puede optimizar el crecimiento de este, con ventajas como: reducción del esfuerzo humano al ejecutar gran volumen de transacciones en un tiempo muy reducido; reducción del riesgo del error humano y aumentando la calidad y precisión; mejorar la calidad de los puestos de trabajo, eliminando tareas repetitivas y tediosas que no aportan valor; e incrementar la productividad y la competitividad.

La implementación de una API que permita la creación de RPA independientemente del software que se use, le permitirá a la empresa disfrutar de beneficios como: la reducción de costos y tiempo, ya que permite realizar las mismas tareas que haría un trabajador en un menor tiempo; aumentar la productividad lo que es equivalente a ganancias; brindar al usuario una mejor experiencia, reducir el riesgo operacional, reproducir tareas humanas sin errores, y se integraría al sistema existente, y sumado a esto, la optimización de la gestión de talento, es decir, que los empleados podrán enfocar sus recursos en otras tareas de más valor para la empresa.

4. OBJETIVOS

A continuación, se presenta el objetivo general y los objetivos específicos de este proyecto.

4.1 OBJETIVO GENERAL

Implementar un prototipo de software usando técnicas de visión computacional para la creación de automatización robótica de procesos - RPA de software contable para organizaciones del sector público.

4.2. OBJETIVOS ESPECÍFICOS

- Caracterizar el proceso y herramientas tecnológicas usadas en organizaciones para registrar su contabilidad.
- Seleccionar tecnologías y métodos que permitan la creación de RPA usando visión computacional.
- Construir el componente de visión computacional que permita identificar los elementos de interfaz de usuario del software contable.
- Desarrollar un prototipo de software usando las tecnologías y métodos seleccionados, para la creación de RPA, a partir del componente de visión computacional construido.
- Evaluar el desempeño del software en un ambiente simulado.

5. MARCO REFERENCIAL

En este capítulo, se presenta la información teórica y conceptual. Además, identifica y expone los fundamentos teóricos, las regulaciones y/o los lineamientos del proyecto de investigación. El principal objetivo es el de recopilar los antecedentes del tema de estudio (teorías, experimentos, datos, estadísticas, etc.). Al hacer esto, se pueda identificar vacíos e interrogantes por explorar que justifican el proyecto. Este capítulo se estructura en 4 secciones.

En la sección 5.1 se presenta el marco conceptual, en donde se describen conceptos relevantes y fundamentales relacionados con el proyecto de investigación.

En la sección 5.2 se presenta el marco teórico, en donde se recopilan los antecedentes, investigaciones previas y consideraciones teóricas que fundamentan este proyecto de investigación.

En la sección 5.3 se presenta el marco normativo y estándares, el cual proporciona las bases para desarrollar el proyecto de investigación junto con el alcance y participación política de este. Además, se identifican las leyes interrelacionadas a este proyecto.

En la sección 5.4 se presenta el estado del arte, en el cual se recopila la información de resultados de otras investigaciones afines al del proyecto de investigación. Se tuvo en cuenta trabajos previos en una ventana de tiempo de los último 5 años hasta.

5.1 MARCO CONCEPTUAL

A continuación, se presentan los conceptos más importantes que abarca este proyecto de investigación relacionados con los procesos contables, automatización y visión computacional. Los conceptos descritos están divididos en 5 áreas: Ingeniería del Software, Automatización Robótica de Procesos (RPA), Interfaz Gráfica de Usuario (GUI), Visión Computacional y Contabilidad.

5.1 Ingeniería del Software. Es la aplicación sistemática de conocimientos, métodos y experiencia científica y tecnológica al diseño, implementación, prueba y documentación de software (Standardization & Normalisation, 1987).

5.1.1 Proceso. Es un conjunto de actividades interrelacionadas o interactivas que transforma las entradas en salidas (Standardization & Normalisation, 1987).

5.1.2 Metodología. Es un sistema de prácticas, técnicas, procedimientos y reglas utilizados por quienes trabajan en una disciplina (Standardization & Normalisation, 1987).

5.1.3 Software. Es la totalidad o parte de los programas, procedimientos, reglas y documentación asociada a un sistema de procesamiento de información (Standardization & Normalisation, 1987).

5.1.4 Proyecto. Conjunto de actividades para desarrollar un nuevo producto o mejorar un producto existente (Standardization & Normalisation, 1987).

5.1.5 Modelo (CRISP-DM). Es la guía de referencia más ampliamente utilizada en el desarrollo de proyectos de Data Mining (Gallardo Arancibia, 2013).

5.1.6 Modelo de Gestión Incremental. El modelo incremental combina elementos del modelo en cascada con la filosofía interactiva de construcción de prototipos. Se basa en la filosofía de construir incrementando las funcionalidades del programa. Este modelo aplica secuencias lineales de forma escalonada mientras progresa el tiempo en el calendario. Cada secuencia lineal produce un incremento del software (Ortiz, 2017).

5.2 Automatización Robótica de Procesos (RPA). La automatización robótica de procesos es la tecnología que permite que cualquiera pueda configurar un software informático que hace posible que un “robot” emule e integre las acciones de una interacción humana en sistemas digitales para ejecutar un proceso comercial. Los robots emplean la interfaz de usuario para capturar datos y manipular aplicaciones existentes del mismo modo que los humanos. Estos robots realizan interpretaciones, activan respuestas y se comunican con otros sistemas para operar en una amplia gama de tareas repetitivas. Y lo hacen considerablemente mejor, pues los robots software nunca duermen, no cometen errores y son mucho menos costosos que los empleados (UiPath, 2017).

5.2.1 Automatización de Interfaz Gráfica de Usuario (GUI). Es una forma de imitar las acciones del usuario, como el movimiento del *mouse* o presionar teclas en el teclado, para automatizar procesos que involucran *clicks*, completar campos, reconocimiento de imágenes y hacer *scroll*. Si bien no se considera el método más elegante para la automatización, en algunas ocasiones la automatización de GUI es la única alternativa (helpsystems, 2020).

5.2.2 Automatización. Es la conversión de procesos o equipos a operación automática, o los resultados de la conversión (Standardization & Normalisation, 1987).

5.2.3 Robótica. Las técnicas involucradas en el diseño, construcción y uso de robots (Standardization & Normalisation, 1987).

5.3 Interfaz Gráfica de Usuario (GUI). Las interfaces de los sistemas informáticos se utilizan como mediadores sistema y un usuario. En el caso en el que sean dos sistemas los que se tienen que comunicar, el mediador entre ellos es un mecanismo, entorno o herramienta. Por tanto, se definen dos tipos de interfaces: la física y la virtual o gráfica. La interfaz física puede ser alterada por el usuario mediante el uso del ratón y el teclado

de un ordenador. La interfaz virtual permite la interacción con los elementos gráficos convirtiendo a la persona en usuario de aplicación. En ambos tipos, se trata de relaciones del tipo entrada de datos. Sin embargo, se necesita un elemento que facilite la salida de datos, donde se visualizan las interfaces gráficas. Como conclusión, GUI se define como la Interfaz Gráfica de Usuario con la que éste es capaz de interactuar con la información digital a través de un entorno gráfico de simulación (Ingenier & Ingenier, 2016).

5.3.1 Prueba de Interfaz Gráfica de Usuario (GUI). Se debe probar tantas veces como escenarios tenga. En caso de que uno de ellos resulte en error se deberá volver a comprobar el funcionamiento de la aplicación con las modificaciones realizadas. Además, hay que tener en cuenta que en muchas ocasiones un evento sólo aparece tras otro anterior, con distintos mensajes o pop-ups hacia el usuario según la situación, etc. Esto hace que el número de ejecuciones de una misma prueba sea muy elevado y que conlleve mucho más tiempo que otro tipo de pruebas (Ingenier & Ingenier, 2016).

5.4 Visión. Visión es la ventana al mundo de muchos organismos. Su función principal es reconocer y localizar objetos en el ambiente mediante el procesamiento de las imágenes. La visión computacional es el estudio de estos procesos, para entenderlos y construir máquinas con capacidades similares. El objetivo de la visión computacional es extraer características de una imagen para su descripción e interpretación por la computadora (Sucar & Gómez, 2011).

5.4.1 Procesamiento de Lenguaje Natural (NLP). Es una rama de la inteligencia artificial que ayuda a las computadoras a entender, interpretar y manipular el lenguaje humano. NLP toma elementos prestados de muchas disciplinas, incluyendo la ciencia de la computación y la lingüística computacional, en su afán por cerrar la brecha entre la comunicación humana y el entendimiento de las computadoras (Moreno, 2017).

5.4.2 Proyección de imagen. La proyección puntual es la transformación de la imagen que se presenta al pasar a muchos de los dispositivos visuales, incluyendo nuestros ojos y una cámara. La aproximación más simple a este fenómeno es el modelo de la “cámara de agujero de alfiler” (pinhole cámara) que consiste en proyectar todos los puntos de la imagen a través de un punto al plano de la imagen. De esta forma, un punto (X, Y, Z) en el espacio, se proyecta a un punto (x, y) en el plano de la imagen (Sucar & Gómez, 2011).

5.4.3 Red Neuronal. Una red neuronal es un modelo de computación cuya estructura de capas se asemeja a la estructura interconectada de las neuronas en el cerebro, con capas de nodos conectados. Una red neuronal puede aprender de los datos, de manera que se puede entrenar para que reconozca patrones, clasifique datos y pronostique eventos futuros (The MathWorks, 2019).

5.4.4 Aprendizaje Profundo. El aprendizaje profundo, también conocido como redes neuronales profundas, es un aspecto de la inteligencia artificial (AI) que se ocupa de emular el enfoque de aprendizaje que los seres humanos utilizan para obtener ciertos

tipos de conocimiento. En su forma más simple, el aprendizaje profundo puede considerarse como una forma de automatizar el análisis predictivo (Rouse, 2017).

5.4.5 Red Neuronal Convolucional (CNN). La CNN es un tipo de Red Neuronal Artificial con aprendizaje supervisado que procesa sus capas imitando al córtex visual del ojo humano para identificar distintas características en las entradas que en definitiva hacen que pueda identificar objetos y ver. Para ello, la CNN contiene varias capas ocultas especializadas y con una jerarquía: esto quiere decir que las primeras capas pueden detectar líneas, curvas y se van especializando hasta llegar a capas más profundas que reconocen formas complejas como un rostro o la silueta de un animal (Bagnato, 2018).

5.4.6 Convolución. La convolución es una operación matemática, en la que una función se "aplica" de alguna manera a otra función. El resultado se puede entender como una "mezcla" de las dos funciones. La convolución se representa por un asterisco (*), que se puede confundir con el operador * que generalmente se utiliza para multiplicar en muchos lenguajes de programación (Cowley, 2018).

5.4.7 Red Neuronal Recurrente (RNN). Una red neuronal recurrente no tiene una estructura de capas definida, sino que permiten conexiones arbitrarias entre las neuronas, incluso pudiendo crear ciclos, con esto se consigue crear la temporalidad, permitiendo que la red tenga memoria (Calvo, 2017).

5.4.8 Redes recurrentes Simples – SRN o Elman. Las redes neuronales recurrentes son la arquitectura base sobre la que se implementan el resto. Se diferencian del resto de redes en que incorporan la retroalimentación lo que se consigue crear la temporalidad, permitiendo a la red que tenga memoria (Calvo, 2017).

5.4.9 Redes LSTM. Las redes LSTM (Long Short Term Memory) están compuestas por unidades LSTM y son un tipo especial de red neuronal recurrente descritas en 1997 por Hochreiter & Schmidhuber. Las redes neuronales recurrentes convencionales presentan problemas en su entrenamiento debido a que los gradientes retropropagados tienden a crecer enormemente o a desvanecerse con el tiempo debido a que el gradiente depende no solo del error presente sino también los errores pasados. La acumulación de errores provoca dificultades para memorizar dependencias a largo plazo (Calvo, 2017).

5.4.10 Lenguaje Específico de Dominio (DSL). lenguaje que esté especializado en modelar o resolver un conjunto específico de problemas. Este conjunto específico de problemas es el llamado dominio de aplicación o de negocio. La mayor parte de los lenguajes de programación no se pueden considerar DSL ya que no están diseñados para resolver un conjunto específico de problemas, sino para resolver cualquier tipo de problema. Son pues, lenguajes generalistas, o *turing* completos (Amodeo, 2010).

5.4.11 Aprendizaje Supervisado. El aprendizaje supervisado son un conjunto de técnicas que permiten inferir modelos para extraer conocimiento de conjuntos de datos donde a priori se desconoce (Calvo, 2019).

5.4.12 Regresión. Los modelos de regresión describen la relación entre una variable de respuesta (salida) y una o varias variables de predicción (entrada) (The MathWorks, 2019).

5.4.13 Reconocimiento de Patrones. El reconocimiento de patrones es un componente importante de las aplicaciones de redes neuronales en visión artificial, procesamiento de radar, reconocimiento de voz y clasificación textual. Funciona mediante la clasificación de los datos de entrada en objetos o clases en función de características clave, ya sea mediante la clasificación supervisada o no supervisada (The MathWorks, 2019).

5.4.14 Aprendizaje No Supervisado. Las técnicas de aprendizaje no supervisado se pueden aplicar sin necesidad de tener los datos etiquetados para el entrenamiento (Calvo, 2019).

5.4.15 Clustering. El *clustering* es un enfoque de aprendizaje no supervisado en el cual se pueden emplear redes neuronales para el análisis de datos exploratorio a fin de localizar patrones ocultos o agrupaciones de datos. Este proceso implica la agrupación de datos por similitud. Entre las aplicaciones del análisis de *clusters* están el análisis de secuencias genéticas, la investigación de mercados y el reconocimiento de objetos (The MathWorks, 2019).

5.5 Contabilidad. Es la ciencia y técnica que enseña a recopilar, clasificar y registrar, de una forma sistemática y estructural, las operaciones mercantiles realizadas por una empresa con el fin de producir informes que, analizados e interpretados, permitan planear, controlar y tomar decisiones sobre la actividad de la empresa (Urueña, 2010).

5.5.1 Proceso contable. Estado contable que refleja la información económica-financiera que dispone la empresa al comienzo del ejercicio económico, es decir; refleja la situación de sus activos (liquidez, valores, bienes, solvencia), sus pasivos (obligaciones con terceras personas) y su patrimonio (Alaña C., Solórzano S. &, Sayonara, 2015).

5.2 MARCO TEÓRICO

En esta sección se definen los fundamentos teóricos de las áreas del conocimiento que tienen relación con este proyecto de investigación.

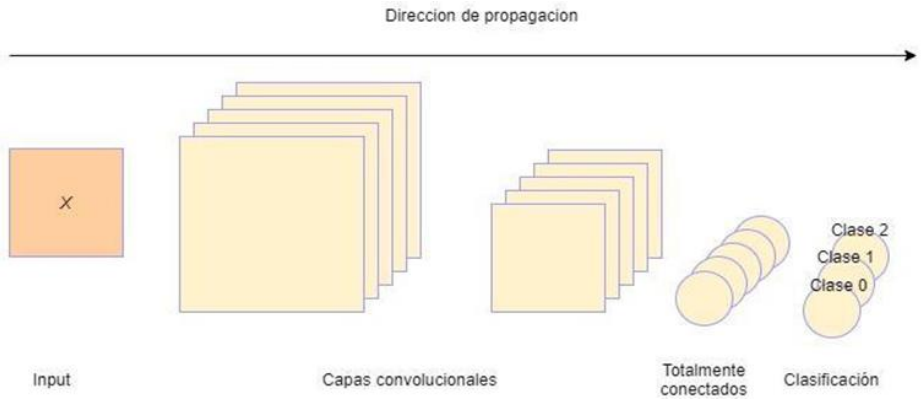
5.2.1 Aprendizaje automático o Machine learning. El aprendizaje automático es uno de los tipos de inteligencia artificial (AI) que proporciona a las computadoras la capacidad de aprender, y esta misma con el tiempo va mejorando su aprendizaje utilizando su propia experiencia. El objetivo de un algoritmo de aprendizaje es producir un resultado en forma de una regla que sea precisa al máximo (Gollapudi, 2019). Ethem Alpaydın dice que el aprendizaje automático consiste en programar computadoras para optimizar un criterio de rendimiento utilizando datos de ejemplo o experiencias pasadas. Su modelo puede hacer predicciones en el futuro, o descriptivo para obtener conocimiento de los datos, o ambos (Alpaydin, 2014). Estos modelos utilizan la teoría de la estadística para la construcción de modelos matemáticos, ya que la tarea principal es hacer inferencia a partir de una muestra. Además, el aprendizaje automático ha permitido encontrar soluciones a los problemas de visión, reconocimiento de voz y robótica.

5.2.2 Deep learning. El aprendizaje profundo es un subcampo del aprendizaje de máquina. A diferencia de los modelos de aprendizaje automático, de lo cual su capacidad de aprendizaje es finita independientemente de cuántos datos adquieran. Para François Chollet, representa la idea de la representación sucesiva y jerarquizada de los datos por medio de capas. La cantidad de capas que contribuyen a un modelo es denominada la profundidad del modelo (Chollet, 2018). Los modelos de aprendizaje profundo mejoran su rendimiento al acceder a una mayor cantidad de datos, permitiendo así la máquina obtener más experiencia. Los modelos de aprendizaje profundo están estructurados por decenas o cientos de capas sucesivas de representación, y todos estas aprenden automáticamente mediante la exposición de modelos datos de entrenamiento. Estas capas de representaciones aprenden a través de modelos llamados redes neuronales, estructurados en capas literales apiladas unas sobre otras.

5.2.3 Redes neuronales convolucionales. Las redes neuronales convolucionales son una variación de las redes neuronales regulares, estas redes son un tipo de red neuronal artificial con aprendizaje supervisado. Las redes neuronales convolucionales resuelven el problema de modelación de piezas de información más pequeñas y las combinan usando redes profundas. Este procesamiento ocurre en múltiples capas (Gollapudi, 2019). Logrando así identificar las diferentes características en las entradas que permiten en la identificación de objetos. Estas redes contienen varias capas ocultas especializadas y con una jerarquía. De esta forma las primeras capas detectan líneas, curvas y se van especializando hasta llegar a las capas profundas que logran reconocer formas más complejas como un animal o un rostro (Convolutional Neural Networks: La Teoría explicada en español | Aprende Machine Learning, 2018).

Los modelos CNN de aprendizajes profundos para realizar el proceso de entrenamiento y de probar, cada imagen de entrada lo introduce a través de una serie de capas de convolución con filtros para producir un mapa de respuestas. Luego se aplica una función de activación al mapa de respuesta, este proceso se repite a través de varias capas convolucionales dependiendo de la arquitectura de red. Para que la red realice la clasificación, la salida de la capa final de convoluciones se aplica a una o varias capas apiladas completamente conectadas y se aplica la función softman para la clasificación del objeto con valores probabilísticos entre 0 y 1. La estructura estándar CNN anteriormente descrita se representa en la Figura 1.

Figura 1 Arquitectura básica de una CNN con dos capas convolucionales y tres clasificaciones.



Nota. La CNN tiene tres tipos de capas: capa convolucional, capa de reserva (*pooling layer*), y capa totalmente conectada (*fully connected layer*). Fuente: Gollapudi, (2019).

En CNN, las capas se establecen en tres dimensiones: altura, anchura y profundidad. Las neuronas de una capa oculta se conectan sólo a un conjunto parcial de neuronas de la otra capa y no se conectan a todas las neuronas. Además, la salida se reduce a un solo vector de puntajes de probabilidad, organizado a lo largo de la dimensión de profundidad. Las capas ocultas ayudan en la extracción de características; esto se hace por las capas de convolución y agrupación, y la clasificación final se hace por la capa totalmente conectada (Gollapudi, 2019).

5.2.3.1 Capa de convolución. La es la primera capa de que extrae las propiedades de una imagen de entrada. Esto quiere decir, que la red aprenderá patrones específicos dentro de la imagen y será capaz de reconocer todas las partes de la imagen. Una capa de convolución está compuesta por n_f filtros del mismo tamaño y profundidad. Para cada filtro, lo convolucionamos con el volumen de entrada para obtener n_f salidas. A continuación, las salidas se pasan a alguna función de activación, ReLU, por ejemplo. Finalmente, esas salidas n_f se apilan juntas en un volumen (h - f_h + 1) x (w - f_w + 1) x n_f (Interns Explain CNN - Data Wow, 2018).

La convolución es una multiplicación por elemento. Estas consisten en tomar grupos de píxeles cercanos de la imagen de entrada e ir operando matemáticamente (producto escalar) contra una pequeña matriz que se llama *kernel*. Supongamos que tenemos una imagen de 5 x 5 cuyos valores en píxeles son 0, 1 y una matriz de filtro 3 x 3. Luego, la convolución de la matriz de imagen de 5 x 5 se multiplica con la matriz de filtro de 3 x 3, y la suma de las matrices forma un mapa de características.

5.2.3.2 Capa de reserva (Pooling layer). Pooling es una operación que acepta un conjunto de valores de datos como entrada y genera un valor de ellos para ser pasado a la siguiente capa. Existen dos propósitos importantes de las operaciones de agrupación (i) reducir el tamaño del espacio de datos para reducir la sobrecarga y (ii) es la invariabilidad de la transición (Lemley et al., 2017). Generalmente las capas de agrupación son usadas inmediatamente después de las capas convolucionales. Lo que hacen estas capas de agrupación es simplificar la información en la salida de la capa convolucional.

La técnica de *pooling* máximo es la técnica más usada ya que esta produce mejores resultados. Esta técnica lo que hace es tomar el valor máximo en cada ventana. Lo cual ayuda a optimizar el tamaño del mapa de características, asegurando que se conserve la información clave sobre la imagen.

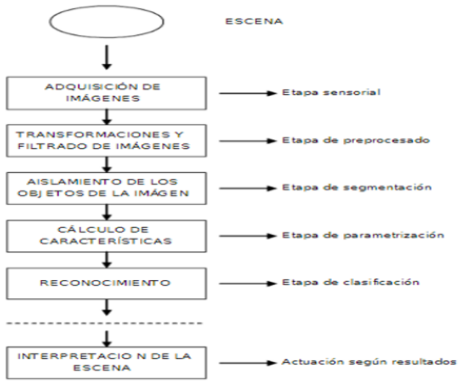
5.2.3.3 Capa totalmente conectada (Fully connected layer). Al final de una CNN, la salida de la última capa de agrupación actúa como entrada a la llamada capa totalmente conectada. Una capa totalmente conectada es una red neuronal con todas las unidades neuronales conectadas entre sí. Puede haber una o más de estas capas. Las capas totalmente conectadas realizan una clasificación basada en las características extraídas por las capas anteriores. Por lo general, esta capa es un ANN tradicional que contiene una función de activación softmax, que genera una probabilidad (un número que va de 0 a 1) para cada una de las etiquetas de clasificación que el modelo intenta predecir (Deep Learning Series, P2: Understanding Convolutional Neural Networks – Towards Machine Learning, 2018).

5.2.4 Transfer learning. Es un método de visión artificial que permite construir modelos precisos, esto permite que, en vez de iniciar un modelo de entrenamiento desde cero, se comienza haciendo uso de modelos pre-entrenados. El aprendizaje de transferencia utiliza el conocimiento adquirido mientras se resuelve un problema y se aplica a otro diferente pero relacionado. Por ejemplo, el conocimiento adquirido mientras se aprende a reconocer automóviles se puede utilizar en cierta medida para reconocer camiones (Radhakrishnan, 2017).

5.2.5 Visión Computacional. La visión artificial es la ciencia y tecnología que permite que las máquinas puedan ver, extrayendo la información de las imágenes digitales, y entendiendo la escena que están observando (García, 2002). los pasos fundamentales en la visión artificial son (i)adquirí la imagen digital, una vez que la imagen ha sido obtenida el (ii) pre procesamiento de la imagen del cual su objetivo es mejorar la imagen,

(iii) la segmentación, su objetivo es dividir la imagen en las partes que la constituyen o los objetos que la forman. (iv) la parametrización, extraer rasgos que producen alguna información cuantitativa de interés o rasgos que son básicos para diferenciar una clase de objetos de otra. Y por último (v) el reconocimiento es el proceso que asigna una etiqueta a un objeto basada en la información que proporcionan los descriptores (clasificación). Los pasos fundamentales de la visión computacional anteriormente descrita se representan en la Figura 2.

Figura 2 Pasos de la Visión Computacional.



Nota. Diagrama de bloques de las etapas de un sistema de visión artificial. Fuente: García, (2002).

5.2.5.1 Detección de bordes. La detección de bordes es una operación de bajo nivel utilizada en la etapa de procesamiento de imágenes. El objetivo principal de la detección de bordes es localizar e identificar discontinuidades agudas en una imagen. Asimismo, reduce significativamente la cantidad de datos y filtra información no deseada y da la información significativa en una imagen. En esta sección contiene información sobre de las técnicas de borde utilizada para extraer y filtrar bordes de una imagen. El filtrado gaussiano y el sobel.

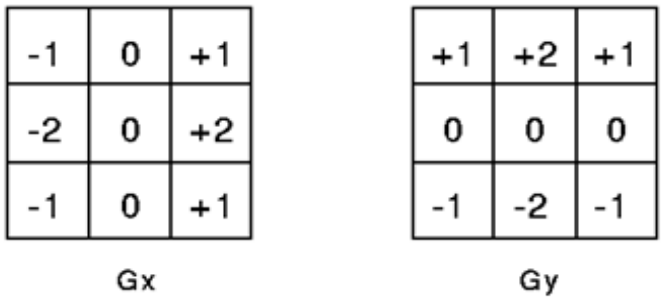
5.2.5.2 Filtrado gaussiano. El filtro gaussiano es utilizado para emborronar imágenes y eliminar el ruido. Gaussian contiene un parámetro que se puede ajustar para lograr resultados requeridos. También en la forma en cómo se construye el núcleo, toma más en consideración los píxeles centrales que el exterior. Algunas de sus ventajas son es separable esto quiere decir que, en vez de realizarse una convolución bidimensional, se pueden realizar dos convoluciones unidimensionales una en sentido horizontal y otra en sentido vertical y así lograr cierto rendimiento. Y otra de sus ventajas es que produce un suavizado más uniforme.

Simulan una distribución gaussiana bivariante. El valor máximo aparece en el pixel central y disminuye hacia los extremos tanto más rápido cuanto menor sea el parámetro de desviación típica s. El resultado será un conjunto de valores entre 0 y 1.

Para transformar la matriz a una matriz de números enteros se divide toda la matriz por el menor de los valores obtenidos.

5.2.5.3 Operadores de Sobel. Para mejorar la detección de orillas se podría utilizar un preprocesamiento para eliminar altas frecuencias o ruido. El detector de orillas Sobel tienen la tarea de suavizar la imagen de tal manera que se elimine un poco el ruido de la imagen si este lo tiene. Los operadores de Sobel parten de los operadores de Prewitt adicionando ciertos pesos en la máscara que aproximan a un suavizamiento Gaussiano (Sucar & Gómez, 2011).

Figura 3 Operador de Sobel.



Nota. Detección de bordes con el Operador de Sobel.
Fuente: Ashish, (2018).

El operador sobel es una aproximación a una derivada de una imagen. Está separado en las direcciones y y x. Si observamos la dirección x, el gradiente de una imagen en la dirección x es igual a este operador aquí. Utilizamos un núcleo de 3 por 3 matriz, uno para cada dirección x e y. El gradiente para la dirección x tiene números negativos en el lado izquierdo y números positivos en el lado derecho y estamos preservando un poco de los píxeles centrales como se puede observar en la Figura 3. Del mismo modo, el gradiente para la dirección tiene números negativos en la parte inferior y números positivos en la parte superior y aquí estamos conservando un poco en los píxeles de la fila central (Ashish, 2018).

El operador de sobel puede ser utilizado con gran éxito para segmentar una imagen. Además, de que es muy rápido de ejecutar. Dado que produce la misma salida cada vez que se ejecuta sobre una imagen, el operador de sobel es una técnica de detección de bordes estable para segmentación de imágenes (Ashish, 2018).

5.2.5.4 Detector de bordes Canny. Es un operador de detección de bordes que utiliza un algoritmo de un algoritmo de etapas múltiples para detectar una amplia gama de bordes en las imágenes. El filtro Canny es un detector de borde de etapas múltiples. Utiliza un filtro basado en la derivada de un gaussiano para calcular la intensidad de los

gradientes. El gaussiano reduce el efecto del ruido presente en la imagen. Luego, los bordes potenciales se reducen a curvas de 1 píxel al eliminar píxeles no máximos de la magnitud del gradiente. Finalmente, los píxeles de borde se mantienen o eliminan usando el umbral de histéresis en la magnitud del gradiente (Maitra Satyajit, 2019).

El filtro de Canny maneja tres parámetros ajustables: el ancho del gaussiano (cuanto más ruidosa es la imagen, mayor es el ancho) y el umbral bajo y alto para el umbral de histéresis.

Los criterios generales para la detección de bordes incluyen:

1. Detección de bordes con baja tasa de error, lo que significa que la detección debe capturar con precisión tantos bordes mostrados en la imagen como sea posible
2. El punto de borde detectado por el operador debe ubicarse con precisión en el centro del borde.
3. Un borde dado en la imagen solo debe marcarse una vez y, cuando sea posible, el ruido de la imagen no debe crear bordes falsos (Maitra Satyajit, 2019).

5.2.6 Algoritmos de detección de objetos.

La detección de objetos en imágenes implica, no solamente identificar de qué tipo de objeto se trata, sino también localizarlo dentro de la imagen (obtener las coordenadas de la caja que lo contiene).

5.2.7 R-CNN. Región-CNN (R-CNN) es uno de los enfoques de detección de objetos de aprendizaje profundo basados en CNN de última generación. En base a esto, hay R-CNN rápido y R-CNN más rápido para la detección de objetos a mayor velocidad. para cada imagen, hay una ventana deslizante, para encontrar cada posición en una imagen, ya sea diferentes tipos de objetos o un mismo objeto puede tener diferentes relaciones de aspectos y tamaños dependiendo de los tamaños de imagen también afectan el tamaño efectivo de la ventana. Si se usa una CNN para la clasificación de imágenes en cada ubicación este proceso sería lento (Remanan Surya, 2019).

1. R-CNN utiliza la búsqueda selectiva para generar aproximadamente 2000 propuestas de región, es decir, cuadros delimitadores para la clasificación de imágenes.
2. Luego, para cada cuadro delimitador, la clasificación de imágenes se realiza a través de CNN.
3. Finalmente, cada cuadro delimitador se puede refinar mediante regresión.

5.2.8 R-CNN rápido. Fast R-CNN es similar al algoritmo R-CNN. Pero, en lugar de alimentar las propuestas de la región a la CNN, alimentamos la imagen de entrada a la CNN para generar un mapa de características convolucional. A partir del mapa de características convolucionales, identifica la región de las propuestas y las deforma en cuadrados y, mediante el uso de una capa de agrupación RoI, se rediseña en un tamaño

fijo para que se pueda alimentar a una capa totalmente conectada. Desde el vector de características RoI, se usa una capa softmax para predecir la clase de la región propuesta y también los valores de desplazamiento para el cuadro delimitador (Remanan Surya, 2019).

5.2.9 YOLO. YOLO -You Only Look Once- utiliza deep learning y CNN para detectar objetos. Para realizar la detección primero divide la imagen en una cuadrícula de SxS como se muestra en la Figura 4 (imagen de la izquierda). En cada una de las celdas predice N posibles de bounding boxes y calcula el nivel de probabilidad de cada una de ellas (imagen del centro), es decir, se calculan SxSxN diferentes cajas, la gran mayoría de ellas con un nivel de probabilidad muy bajo. Después de obtener estas predicciones se procede a eliminar las cajas que estén por debajo de un límite. A las cajas restantes se les aplica un paso de non-max suppression que sirve para eliminar posibles objetos que fueron detectados por duplicado y así dejar únicamente el más exacto de ellos (imagen de la derecha) (Remanan Surya, 2019).

Figura 4 Detección de objetos.

Nota. Detección y división en una cuadrícula SxS con el algortimo YOLO. Fuente: Remanan Surya, (2019

La detección de objetos como un único problema de regresión, directamente desde los píxeles de la imagen a las coordenadas de los bounding box y a las probabilidades de clase. Usando nuestro sistema, you only look once (YOLO) a una imagen para predecir qué objetos están presentes y dónde están. Cada celda de la cuadrícula predice las coordenadas de los cuadros delimitadores B y la confianza (Redmon, Joseph, Santosh Divvala & Farhadi, 2016).

5.2.10 Ingeniería de Software. Es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software; es decir, la aplicación de la ingeniería de software (Pressman, 2002). Los métodos de la ingeniería de software facilitan técnicas para elaborar software. Incluyen una serie de tareas, como comunicación, análisis de los requerimientos, modelación del diseño, construcción del

programa, pruebas y apoyo. Los métodos de la ingeniería de software se basan en un conjunto de principios fundamentales que gobiernan cada área de la tecnología e incluyen actividades de modelación y otras técnicas descriptivas (Pressman, 2002).

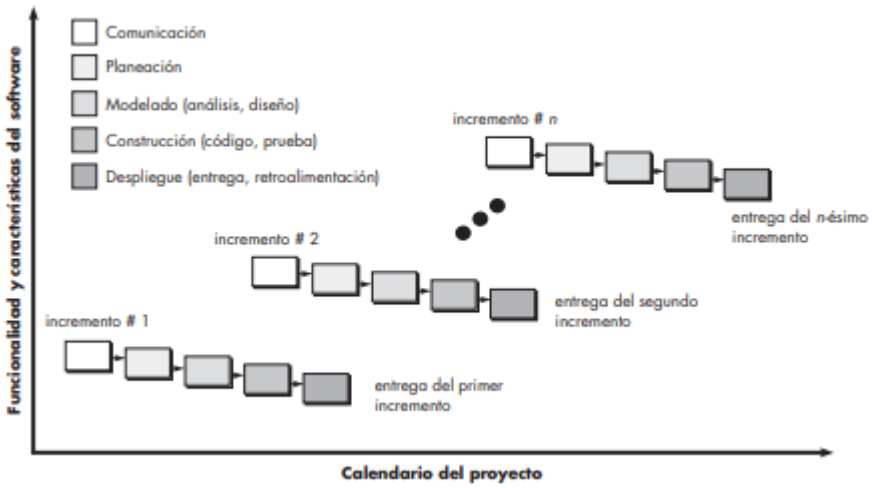
El proceso de software incorpora cinco actividades estructurales: comunicación, planeación, modelado, construcción y despliegue que son aplicables a todos los proyectos de software (Pressman, 2002).

- **Comunicación**: es la comunicación con el cliente para para reunir los requerimientos que permitan definir las características y funciones del software.
- **Planeación**: definición de las tareas técnicas a realizar, los riesgos probables, los recursos a utilizar, los productos del trabajo a obtener y la programación de las actividades a desarrollar.
- **Modelado**: diseño del modelo para entender mejor los requerimientos del software y que logre cumplir estos mismos.
- **Construcción**: Es la combinación de generación de código y las pruebas necesarias para descubrir y corregir errores en el software.
- **Despliegue**: Entrega del software al cliente para que lo evalúe para su respectiva retroalimentación.

5.2.10.1 Modelo de Gestión Incremental. El modelo incrementar tiene como objetivo un crecimiento progresivo de la funcionalidad. Es decir, el producto va evolucionando con cada una de las entregas previstas hasta que se adapta a lo requerido por el cliente (*Características y fases del modelo incremental* | *OBS Business School*, 2016).

El modelo de gestión incremental es fácil de adaptarse a las diferentes características de cualquier software. Este modelo aplica secuencias lineales de forma escalonada a medida que se avanza en el calendario de las actividades. Cada secuencia lineal es un producto del software. Cada entrega del software se denomina un incremento del cual se construye sobre un incremento que ya ha sido entregado (Pressman, 2002).

Figura 5 Fases del Modelo Incremental.



Nota. Proceso del modelo Incremental con respecto a la funcionalidad y el cronograma del proyecto. Fuente: Pressman (2002).

Fases del modelo incremental:

1. Requerimientos: son los objetivos generales y específicos que persigue el proyecto.
2. Definición de las tareas y las iteraciones: con base a los objetivos del proyecto, se hace una lista de tareas agrupadas en las iteraciones.
3. Diseño de los incrementos: establecidas las iteraciones, se define cuál será la evolución del producto en cada incremento. Cada iteración debe superar al anterior.
4. Desarrollo del incremento: se realizan las tareas previstas y se desarrollan los incrementos establecidos en la etapa anterior.
5. Validación de incrementos: al término de cada iteración, se valida si el incremento cumple los resultados esperados o si ha presentado algún retroceso, en dado caso, se busca las causas de ello.
6. Integración de incrementos: después de validados, los incrementos dan forma a lo que se denomina línea incremental o evolución del proyecto en su conjunto. Cada incremento ha contribuido al resultado final.
7. Entrega del producto: cuando el producto en su conjunto ha sido validado y se confirma que cumpla con los requerimientos, se procede a su entrega final (*Características y fases del modelo incremental* | *OBS Business School*, 2016).

5.2.10.2 Metodología CRISP-DM. CRISP-DM por sus siglas en inglés significa *CRoss-Industry Standard Process for Data Mining*, es una metodología utilizada para la orientación de proyectos de minería. Además, cubre las fases del proyecto, las tareas repetitivas y la relación entre estas mismas tareas. La metodología CRISP-DM contempla el proceso de análisis de datos como un proyecto profesional, estableciendo así un contexto mucho más rico que influye en la elaboración de los modelos (Villena Román, 2016).

CRISP-DM surge a finales de la década de 1990, cuando un grupo de consorcios importantes de empresas europeas, comenzaron por proponer las primeras ideas a partir de diferentes versiones de *Knowledge Discovery in Databases* – KDD, el desarrollo de una guía de referencia de libre distribución denominada *Cross Industry Standard Process for Data Mining* – CRISP-DM (Gallardo Arancibia, 2013).

Figura 6 Modelo CRISP-DM.

Nota. Las 6 Fases del modelo CRISP-DM.
Fuente: Pete er al (2000).

La metodología CRISP-DM se dividen en seis fases Figura 6, a continuación, se describe cada una de estas fases:

1. **Fase de comprensión del negocio**: En esta fase inicial se centra en la compresión de los objetivos y los requerimientos del proyecto desde el punto de vista empresarial, para después, convertir este conocimiento en una definición del problema de minería de datos, definir los criterios de éxito y definición de los objetivos (Azevedo & Filipe Santos, 2008; Gallardo Arancibia, 2013; Pete et al., 2000).

2. **Fase de comprensión de los datos**: La fase de entendimiento de datos comienza con la recolección de datos y continúa con las actividades que permiten familiarizarse con los datos, identificar los problemas de calidad, descubrir conocimiento preliminar sobre los datos, y/o descubrir subconjuntos interesantes para formar hipótesis en cuanto a la información oculta (Pete et al., 2000; Villena Román, 2016).

3. **Fase de preparación de los datos**: La fase de preparación de datos cubre todas las actividades necesarias para construir el conjunto final de datos, los cuales se utilizarán en las herramientas de modelado, a partir de los datos en bruto iniciales (Azevedo & Filipe Santos, 2008; Villena Román, 2016).

4. **Modelado**: En esta fase, se seleccionan las técnicas de modelado más apropiadas para el proyecto y se calibran sus parámetros a valores óptimos. Algunas técnicas tienen requisitos específicos en cuanto a la forma de los datos. Por lo tanto, a menudo es necesario volver a la fase de preparación de los datos (Azevedo & Filipe Santos, 2008; Pete et al., 2000; Villena Román, 2016).

5. **Fase de evaluación**: En esta fase se evalúa el modelo, teniendo en cuenta el cumplimiento de los criterios de éxito del problema (Villena Román, 2016). Antes del despliegue final del modelo, es importante evaluarlo, revisar el proceso y validar que haya cumplido con los objetivos (Pete et al., 2000).

6. **Despliegue**: Generalmente, la creación del modelo no es el final del proyecto. Incluso si el objetivo del modelo es aumentar el conocimiento de los datos, el conocimiento obtenido tendrá que organizarse y presentarse para que el cliente pueda usarlo (Villena Román, 2016).

5.2.1.1 Interfaz Gráfica de Usuario (GUI)

La Interfaz Gráfica de Usuario es un conjunto de programas informáticos que permite a una persona comunicarse con una computadora mediante el uso de símbolos, metáforas visuales y dispositivos señaladores (Levy Steven, 2015). Una interfaz de usuario es lo que permite al usuario controlar una aplicación de software o dispositivo de hardware.

Elementos de una interfaz gráfica

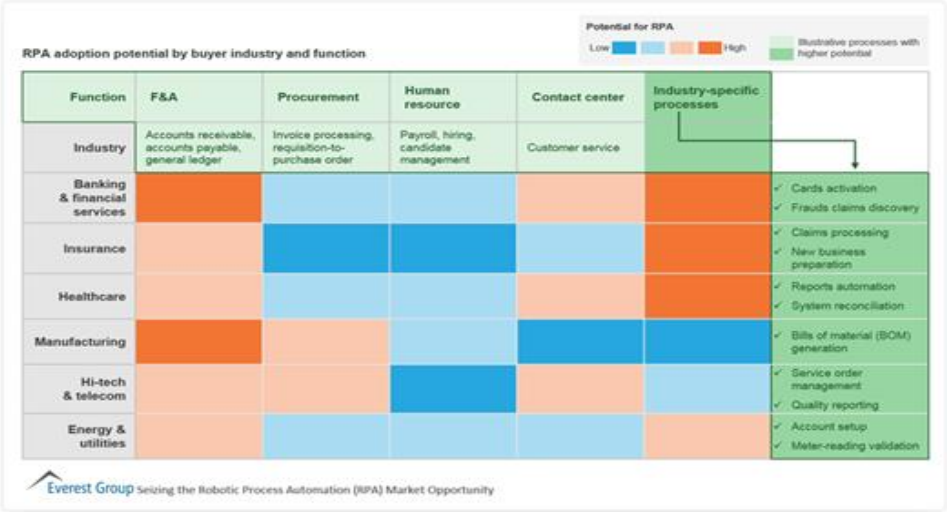
- **Barra de desplazamiento:** Elemento de la interfaz gráfica de usuario que permite visualizar el contenido de una ventana o área de trabajo (*EcuRed*, 2015).
- **Barra de herramientas:** Una barra de herramientas es una tira de controles que permite el acceso conveniente a funciones usadas comúnmente (*GNOME developer*, 2014).
- **Barras de menú:** Una barra de menú incorpora una tira de menús desplegables (*GNOME developer*, 2014).
- **Botones:** Los botones son uno de los elementos básicos de interfaz de usuario más conocidos. Los botones pueden usarse para realizar acciones, activar opciones o vistas, activar herramientas o mostrar diálogos, cuadros emergentes u otros elementos de la interfaz (*GNOME developer*, 2014).
- **Campos de texto:** Un campo de entrada de texto es un elemento de la interfaz para introducir o editar texto (*GNOME developer*, 2014).
- **Casillas:** Las casillas se usan para mostrar o cambiar una configuración. Tienen dos estados, activada y desactivada, y se distinguen por la presencia de una marca en la casilla etiquetada (*GNOME developer*, 2014).

- **Listas desplegables:** Una lista desplegable es un elemento de la interfaz de usuario que le permite seleccionar de una lista opciones mutuamente excluyentes. Aparece como un botón que, cuando se pulsa, muestra una lista (*GNOME developer*, 2014).
- **Pestañas:** Las pestañas ofrecen una manera de dividir una ventana en varias vistas. Hay dos tipos principales: fijas y dinámicas. Las pestañas fijas proporcionan un conjunto inmutable de vistas predefinidas, y se usan principalmente en ventanas de diálogo. Las pestañas dinámicas permiten que una ventana contenga una selección modificable de elementos de contenido, tales como páginas, documentos o imágenes (*GNOME developer*, 2014).

5.2.12 RPA (Automatización de Procesos Robótica)

La *Automatización de Procesos Robótica*, es la tecnología orientada al uso de software del cual su principal objetivo es disminuir la intervención humana, en tareas repetitivas (AuraPortal, 2018). Los robots emplean la interfaz de usuario para capturar datos y manipular aplicaciones existentes del mismo modo que los humanos. Realizan interpretaciones, activan respuestas y se comunican con otros sistemas para operar las tareas repetitivas (UiPath, 2017).

Figura 7 Automatización de Procesos de Robótica.



Nota. Potencial de adopción de RPA por industria y función del comprador.
Fuente: UiPath, (2017).

Robotics, engloba tres términos: softwares de Robotic Process Automation –RPA-, Tecnología Cognitiva e Inteligencia Artificial. Tres conceptos están escalando posiciones imparablemente como fuente de ventaja competitiva (Deloitte, 2017).

El robot Dell software de RPA funciona en la interfaz de usuario (IU), realizando acciones: como mover el mouse, utilizar el teclado, buscar en el monitor. Después de entrenarlo

para entender un proceso en específico, la RPA puede ejecutar de forma automática los procesos que se le asignaron (Deloitte, 2017).

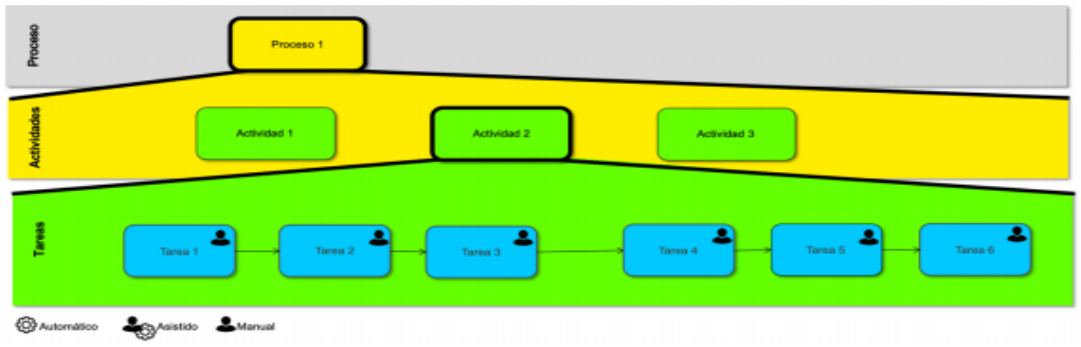
Una condición indispensable para el uso de RPA es su aplicación en procesos estructurados, con inputs y outputs basados en datos existentes. Además, deben ser entrenados para ejecutar las tareas basados en reglas establecidas y flujos de trabajos. Las RPA no tienen la capacidad de aprender de su experiencia, razón por la cual no están capacitados para el manejo de excepciones que llegue a encontrar en la ejecución de un proceso (Deloitte, 2017).

5.2.12.1 Estructura de una RPA

Se compone de 6 fases, basadas en las capas que intervienen en el diseño de la arquitectura de la solución RPA: lineamientos de proceso, lineamientos de aplicación, lineamientos de robots, lineamientos de datos y lineamientos de integración (Gonzales, 2019).

Lineamientos de Proceso. definición del diagrama de procesos, a nivel de proceso macro, actividades y tareas. En la Figura 8 se plantea el diagrama funcional con el proceso, las actividades y las tareas que lo componen.

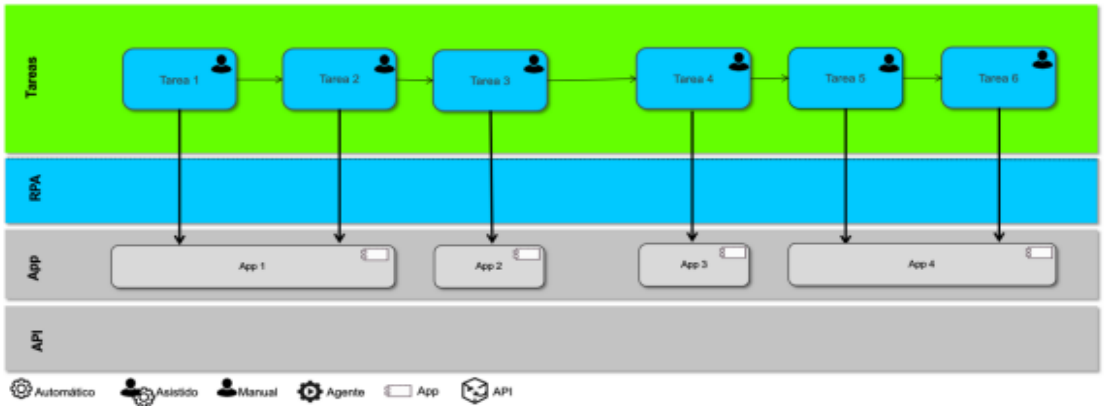
Figura 8 Diagrama de Procesos.



Nota. Diagrama funcional de los procesos de aplicación. Fuente: González, (2019).

Lineamientos de aplicación. Definición de las aplicaciones que se utilizan en cada una de las tareas, para así determinar la factibilidad de interacción. En la Figura 9 se muestra el diagrama de aplicación (Gonzales, 2019).

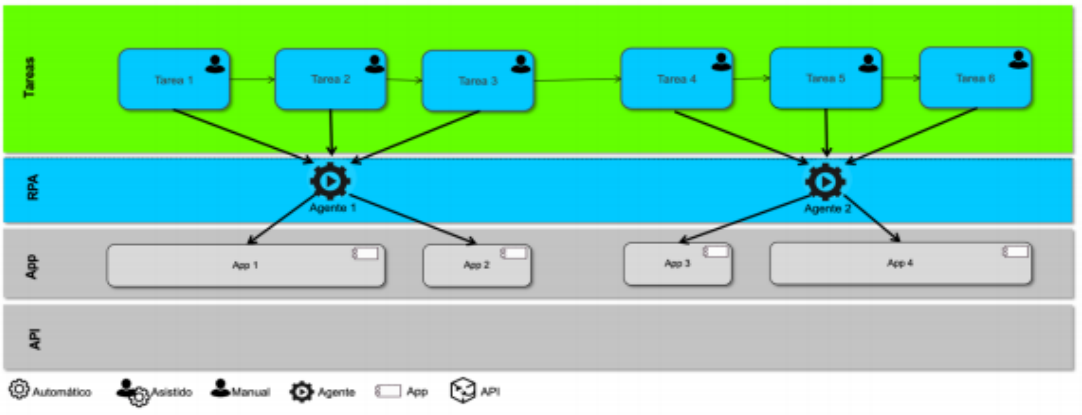
Figura 9 Diagrama de aplicación.



Fuente: González, (2019).

Lineamientos de Robots (Agentes). Diseño de la solución incluyendo la capa de Robots (Agentes). Los Robots pueden interactuar con las aplicaciones a través de Screen Scraping en sus diferentes técnicas de extracción de datos para aplicaciones Desktop, Web entre otras (Gonzales, 2019).

Figura 10 Diagrama de Robot.

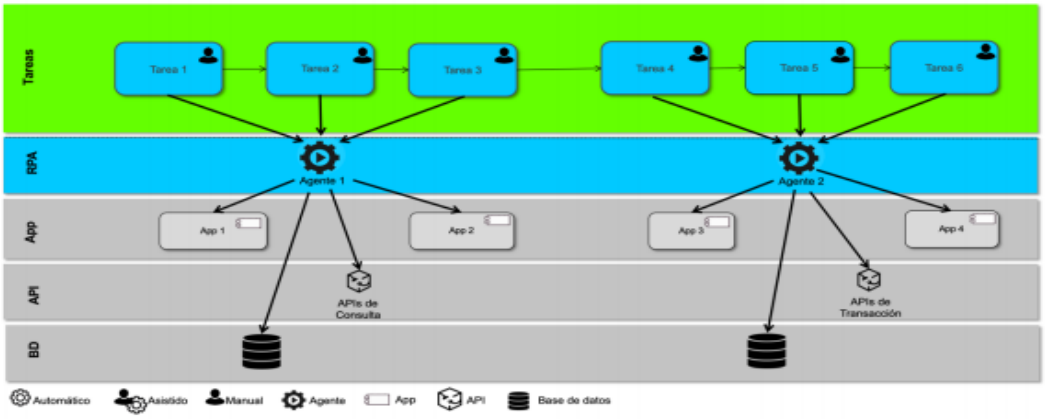


Fuente: González, (2019).

Lineamientos de Datos. la RPA accede solo a su propio esquema de bases de datos.

Lineamientos de Integración e Interacción. La RPA puede integrarse a otras aplicaciones solo a través de APIs. También podrá interactuar con otras aplicaciones a través de *Scrapping* (Gonzales, 2019).

Figura 11 Diagrama de interacción.



Fuente: González, (2019).

5.2.13 Herramientas para desarrollo de RPA

Robot Framework. Es un marco genérico de automatización de código abierto. Se puede utilizar para la automatización de pruebas y la automatización de procesos robóticos (RPA). Robot Framework es abierto y extensible y se puede integrar con prácticamente cualquier otra herramienta para crear soluciones de automatización potentes y flexibles. Sus capacidades pueden ampliarse mediante bibliotecas implementadas con Python o Java. Es un software de código abierto y permite algunas características interesantes. El marco tiene un ecosistema rico a su alrededor, que consiste en bibliotecas y herramientas que se desarrollan como proyectos separados (Capocchi et al., 2013; Robot Framework, 2008).

Python. Es un lenguaje de programación que está orientado a objetos y es muy sencillo de utilizar al mismo tiempo que guarda toda su potencia. Es de código abierto, algunas de sus características son: sencillez, exactitud en la síntesis, buena legibilidad, poder programar algoritmos complejos en pocas líneas y también usar los dialectos, unas variantes adaptables a otros lenguajes (Sevilla, 2018).

5.3 ESTADO DEL ARTE

El objetivo general de esta subsección es conocer qué métodos y tecnologías existen para el reconocimiento de interfaces de usuario usando visión computacional. En la Tabla 1 se presenta el protocolo de búsqueda del estado del arte. Inicialmente se recuperaron 30 documentos, y luego de una lectura y análisis de su pertinencia para el proyecto, se seleccionaron 11 documentos para el desarrollo del estado del arte de la presente investigación.

Tabla 1 Criterios de búsqueda

Revisión de la literatura	
Palabras Clave	*UI testing, computer visión, test automation, automatic code generation*
Bases de datos Consultadas	*IEEE Xplore, Google, Google Scholar*
Cantidad de Referencias Recuperadas	11 documentos recuperados
Criterios de Búsqueda	Rango de Fecha de la Búsqueda: entre 2014 y 2020.
	Pertinencia con el tema de investigación: *UI testing.* *Computer vision.* *Test automation,* *Automatic code generation*
	Documentos en inglés, publicados en Revistas Científicas, libros o Actas de Conferencia.

Fuente: Autoras, (2020).

En la Tabla 2 se presentan los trabajos recopilados que tienen mayor relevancia con el proyecto de investigación para el análisis del estado del arte. Estos trabajos se presentan en orden cronológico.

Tabla 2 Trabajos recopilados.

Autores	**Título**	**Año**	**Cita**
Ki, Jang-geun Kwon, Kee-young	Detection of GUI Elements on Sketch Images Using Object Detector Based on Deep Neural Networks	2019	Ki & Kwon, (2019)
Mihir Mistry, Ameya Apte, Varad Ghodake(&), and S. B. Mane	Machine Learning Based User Interface Generation	2019	Mihir Mistry, Ameya Apte, Varad Ghodake(&), (2019)

Narayana, M. Raghu Ram Reddy, N. Hyndavi Reddy, N.	High speed script execution for GUI Automation using Computer Vision	2019	Narayana et al., (2019)
Uskenbayeva, R., Kalpeyeva, Z., Satybaldiyeva, R., Moldagulova, A., & Kassymova, A.	Applying of RPA in Administrative Processes of Public Administration.	2019	Uskenbayeva et al., (2019)
Beltramelli, Tony	Pix2code: Generating code from a graphical user interface screenshot	2018	Beltramelli, (2018)
Kim, Bada Park, Sangmin Kim, Bongjae	Deep - Learning Based Web UI Automatic Programming	2018	Kim et al., (2018)
Torres, Luis Guillermo	Generador de Código para Arquitectura Microsoft. NET a partir de modelos ISML	2018	Torres, (2018)
Yun, Young-sun Park, Jisu	Automatic Mobile Screen Translation Using Object Detection Approach Based on Deep Neural Networks	2018	Yun & Park, (2018)
Garcia, Alejo	Automatización de pruebas de interfaz gráfica en herramientas de tesorería	2016	Garcia, (2016)
Huang, Ruozi Long, Yonghao Chen, Xiangping	Automatically generating web page from a mockup	2016	Huang et al., (2016)
Nguyen, Tuan Anh Csallner, Christoph	Reverse engineering mobile application user interfaces with REMAUI	2016	Nguyen & Csallner, (2016)

Fuente: Autoras, (2020).

A continuación, es presentada una síntesis de los documentos estudiados, los cuales fueron revisados en su totalidad por los autores del proyecto.

Ki y Kwon (2019) presentan el desarrollo de conversión de código semiautomática y construcción de elementos GUI. Inicialmente, los autores proponen la detección de elementos GUI basado en redes neuronales (DNN) para describir objetos jerárquicamente. Además, en los estudios presentes en el documento, la mayoría se basan en el conocimiento específico del dominio o generaban códigos de computadora directamente usando una combinación de redes neuronales recurrentes y convolucionales (CNN y RNN). Sin embargo, los autores sugieren el método de reconocimiento basado en el detector de objetos de elementos GUI. Por otro lado, presenta un sistema automático de generación de código a partir de una imagen de boceto dada con un modelo jerárquico de descripción de objetos. Por consiguiente, para la preparación de datos usa *Labellmg* como herramienta de anotación implementado con python y Qt, para encontrar los objetos gráficos emplean la técnica de detección de objetos en tiempo real de YOLO. Por último, después de encontrar los objetos gráficos,

la anotación que incluye el cuadro delimitador y la información de clase se procede a transformarla en una estructura semántica jerárquica. Finalmente, el resultado fue el diseño de sistema de generación automática de código para componentes GUI que se dibujan manualmente con lápiz en el papel.

Mihir Mistry, Ameya Apte, Varad Ghodake (2019) proponen un sistema para la generación de código HTML desde la imagen de un boceto, implementado visión por computadora y usando métodos como el reconocimiento óptico de caracteres (OCR), las redes neuronales de convolución y sus mejoras en específico en los algoritmos RCNN y YOLO, la región de interés (ROI). Los autores encontraron que la capa de salida no era constante, porque la imagen puede tener un número variable de ocurrencia de objetos en una imagen. Por lo tanto, para la solucionar el problema de encontrar y clasificar los objetos encontraron que algoritmos como R-CNN, Faster-RCNN y YOLO son útiles para resolver ese problema. Con los resultados obtenidos de los experimentos realizados en el entrenamiento de estos algoritmos generaron que necesitaban de una gran cantidad de datos de entrenamientos para que fueran precisos. El preprocesamiento de las imágenes mejoró la detección de las ROI mediante métodos openCV, lo que aumentó en gran medida la precisión del conjunto de datos en comparación con los algoritmos de detección de objetos. Después de obtener un archivo JSON con que describe el contenido de los bocetos utilizando la técnica de las redes adversas generativas GAN para generación de códigos.

Narayana et al. (2019) presentan el desarrollo de un prototipo de herramienta de automatización basado en visión computacional utilizando C++ y OpenCV, para automatizar aplicaciones de windows, aplicaciones web, sitios web flash, aplicaciones basadas en Citrix. El modelo propuesto, usa la coincidencia de plantilla, la cual es una técnica de la visión computacional para encontrar el área de imagen que coincide con la imagen de las plantillas. Por otra parte, el teclado se maneja simulando las pulsaciones de teclas usando códigos de teclas virtuales de varios botones en el teclado, con lo cual, se pueden realizar varias acciones. El manejo del mouse, se genera una vez después de localizar el objeto utilizando el algoritmo de coincidencia de plantilla basado en las entradas del usuario. Asimismo, para algunos escenarios es probable que haya múltiples objetos de plantillas iguales. Para este caso, el objeto deseado se puede identificar según la posición en la imagen de origen. Finalmente, el modelo propuesto se compara con la herramienta de código abierto Sikuli, según los resultados el tiempo de ejecución del modelo propuesto fue menor con 406.04 segundos comparado con 425.09 segundos de la ejecución de los scripts automáticos de Sikuli, es decir, el tiempo de ejecución se pudo minimizar.

Uskenbayeva *et al* (2019) aplicaron la tecnología RPA en los procesos de administración pública, y presentan criterios y fases de ejecución de RPA, como son: repetir las operaciones y volumen de datos procesados. Asimismo, mencionan que la aplicación efectiva de RPA, incluye: análisis general de los procesos de negocio, selección de procesos para la robotización, el desarrollo, implementación y escalamiento de la RPA. Por otro lado, cabe señalar el uso de metodologías de gestión de procesos de negocio –

BPM centrado en la automatización de procesos de negocio existentes y aumentar la existencia de métodos y sistemas de gestión y operación de procesos de negocio.

Beltramelli (2018) intenta abordar el problema de la generación de código de interfaz de usuario a partir de entradas visuales aprovechando el aprendizaje automático para aprender variables latentes en lugar de diseñar heurísticas complejas. El autor presenta pix2code, un método para generar código de computadora dada una sola imagen GUI como entrada. Asimismo, utiliza la Red Neuronal Convolucional (CNN) para realizar el aprendizaje de características no supervisadas mediante el mapeo de una imagen de entrada a un vector aprendido de longitud fija; actuando como un codificador. Por otro lado, presenta el diseño de un DSL para describir las GUI, este modelo puede realizar un modelado de lenguaje a nivel de token (elemento) con una entrada discreta mediante el uso de vectores codificados en caliente; eliminando la necesidad de técnicas de incrustación de palabras. Además, al usar un modelo CNN, el token de entrada se codifica por un modelo de lenguaje basado en LSTM (Memoria larga a corto plazo) permitiendo que el modelo se centre más en ciertos tokens y menos en otros.

Kim et al. (2018), su principal propósito en su estudio era en el desarrollo de un Sistema automatizado. Una tecnología de programación que generará una página web HTML real del boceto de una interfaz de usuario. Implementan un algoritmo de visión por computadora para la identificación de las características de componentes simples. Asimismo, lo combina con el algoritmo de aprendizaje profundo Faster R-CNN para el reconocimiento de múltiples objetos GUI dentro de una imagen.

Torres (2018) construye un generador de código denominado DontNeyGenerator. Este generador permite generar código de una fuente de Aplicación Web ASP.NET MCV, el código fuente puede ser abierto si usa la herramienta Visual Studio Community 2017 el cual permite reducir esfuerzo en costo y tiempo invertido en las generaciones de componentes de una aplicación Web. Este generador, permite convertir un modelo ISML (*Information Systems Modeling Language*) a código en Arquitectura Microsoft .NET.

Yun y Park (2018) proponen en su investigación la detección de elementos GUI basado en la estrategia de detección de objetos utilizando redes neuronales profundas (DNN). La detección de objetos con DNN es el enfoque que integra las técnicas de localización y clasificación. A partir del resultado experimental, los resultados obtenidos se utilizarían para la generación automática de código a partir de la captura de una imagen. Para extraer los objetos gráficos, utilizaron características básicas GUI usando estos elementos como información jerárquica. El método utilizado al principio por los autores para detección de objeto utilizó red neuronal convolucional y una red neuronal convolucional basada en la región evaluaron que el método de RCNN tenían una desventaja que era lento. Para mejorar la velocidad de detección, el algoritmo de búsqueda de selección se reemplaza con una red neuronal. R-CNN más rápido. Concluyeron que a la integral el proceso de detección de objetos y el proceso de clasificación mostraban un rendimiento del 87%, considerando que para obtener un buen

rendimiento se debe seleccionar un modelo apropiado de acuerdo con los datos de entrenamiento y la información de la clase de objeto.

Garcia (2016) presenta el desarrollo de una serie de funcionalidades para automatizar la fase de pruebas GUI en aplicaciones del sector de la tesorería. Asimismo, en el área de banca y tesorería se han desarrollado herramientas que automatizan los procesos y tareas de una aplicación. Una de ellas se llama *Test Case Executor* (TCE), implementada sobre Calypso. Por tanto, el autor realiza un estudio de las herramientas más prominentes y de esta manera seleccionar la más apropiada para implementarla en funcionalidades de TCE sobre Calypso. Sin embargo, TCE solo muestra si la funcionalidad va bien o no, pero no muestra un soporte del error en caso tal de que suceda. Por esta razón, es necesario añadir funcionalidades para añadir evidencia, con base en ello, dispone de una herramienta de automatización de pruebas de GUI que permitirá asegurar el correcto funcionamiento de la interfaz de la aplicación. Basado en el estudio que el autor realizó, las herramientas más usadas para la automatización de testing GUI son Selenium, Sikuli, Automa y AutoIt. Sin embargo, determinó que Sikuli es la más apropiada para las funcionalidades a desarrollar en el TCE sobre Calypso. Esto se debe a que Selenium es una herramienta orientada a aplicaciones web y necesita el acceso al código de la aplicación, pues reconoce los distintos elementos mediante los id que les son asignados en los ficheros HTML. Automa y AutoIt son herramientas muy potentes, pero están diseñadas para la automatización de pruebas en aplicaciones del sistema operativo de Windows. A diferencia de ellas, Sikuli es una herramienta que automatiza las pruebas de testing GUI de cualquier tipo de aplicación sin necesitar acceder a su código fuente, pues distingue los elementos mediante el reconocimiento visual. Esto se debe a que se basa en la herramienta de OpenCV.

Huang et al. (2016) presentan un método para generar un borrador de página web a partir de una maqueta. Los autores proponen un algoritmo para extraer los elementos de la maqueta en función de la estructura metálica de la maqueta. Asimismo, el proceso para realizar el borrador de página web está estructurado en los siguientes pasos: diseñar una estructura metálica para mostrar las funciones y estructuras básicas de la página, la maqueta puede estimarse como un *screenshot* del diseño y tema de colores de la página web, extraer los elementos de la maqueta y selección de etiquetas para estos; y finalmente, generar la estructura jerárquica de los elementos y escribir códigos HTML y CSS para generar un prototipo de la página web. A partir de esto, los autores notaron que el proceso de transformación de la maqueta al prototipo de página web tiene el potencial de ser automatizado, luego de aplicar el algoritmo para extraer los elementos pudieron obtener un árbol jerárquico de las estructuras de los elementos. Dado que los elementos tienen un uso uniforme en las páginas web, los autores proponen un algoritmo de generación de etiquetas de abajo hacia arriba para elegir etiquetas para cada elemento. Este algoritmo se basa en el método del bosque aleatorio. Finalmente, la definición de estos elementos genera un prototipo de página web.

Nguyen y Csallner (2016) presentan una propuesta llamada REMAUI, que tiene como objetivo realizar ingeniería inversa GUI de aplicaciones móviles. Este sistema utiliza la

combinación de *Optical Character Recognition* (OCR), visión computacional y heurística para detectar componentes y generar una aplicación estática. Las técnicas de visión computacional utilizadas en REMAUI son muy eficaces ya que los resultados presentados en los experimentos realizados de capturas de pantallas de aplicaciones móviles, la IU generadas por REMAUI eran similares a las originales.

Análisis y conclusiones del Estado del Arte

En los documentos estudiados en este estado del arte, se identificó que la mayoría de las investigaciones que estuvieron relacionadas en el reconocimiento de la interfaz gráfica implementaron visión computacional. Estos estudios llegaron a la conclusión que, algoritmos como redes neuronales de convolución combinadas, R-CNN, Faster-RCNN y YOLO permiten mejorar y solucionar los problemas para encontrar y clasificar los objetos.

En la revisión de la literatura se identificó que el tema de visión computacional para el desarrollo de RPA está poco abordado en la literatura. Respecto a la tecnología de visión computacional sí en se desarrolla la implementación de esta tecnología en diferentes áreas como lo es en empresas industriales, empresas financieras, de logística y el área contable. Asimismo, en base a esos casos de estudios se puede ver las ventajas que estas tecnologías logran en las empresas para la reducción de costos, el mejoramiento en el servicio que prestan y la alta productividad.

5.4 MARCO LEGAL

En el marco legal del proyecto se tuvieron en consideración normas y recomendaciones nacionales e internacionales relacionadas con el tema de este proyecto.

Norma ISO 27001. de Sistemas de Gestión de Riesgos y Seguridad, la cual permite el aseguramiento, la confidencialidad e integridad de los datos y de la información, así como de los sistemas que la procesan. El estándar ISO 27001:2013 para los Sistemas Gestión de la Seguridad de la Información permite a las organizaciones la evaluación del riesgo y la aplicación de los controles necesarios para mitigarlos o eliminarlos (ISO, 2009)

Norma ISO/IEC 25000. Calidad del producto, conocida como SQuaRE (System and Software Quality Requirements and Evaluation) la cual constituye una serie de normas basadas en ISO/IEC 9126 y en ISO/IEC 14598 cuyo objetivo principal es guiar el desarrollo de los productos de software mediante la especificación de requisitos y evaluación de características de calidad (ISO 25000, 2016)

Norma ISO/IEC 2500n. – División de Gestión de Calidad, formada por una serie de normas que definen todos los modelos, términos y definiciones comunes referenciados por todas las otras normas de la familia 25000 (Iso25000, 2018).

Norma ISO/IEC 25030 - Quality requirements: provee de un conjunto de recomendaciones para realizar la especificación de los requisitos de calidad del producto software (Iso25000, 2018).

6. METODOLOGÍA

Para el desarrollo de este proyecto de Trabajo de Grado, se definieron cinco etapas, relacionadas estrechamente con los objetivos específicos del proyecto. A continuación, se describen cada una de las fases, y en la Tabla 1 se presentan las actividades asociadas con cada una de las etapas y sus respectivos entregables.

Fase 1: Caracterización del proceso contable

En esta fase inicial, se diseñará un instrumento de recolección de información que será aplicado al personal contable de organizaciones del sector público y privado, esto con el fin de reunir características genéricas de dicho proceso, sin importar el sector económico o tamaño de la organización. Una vez recopilada y analizada la información, se procederá a realizar un levantamiento de requerimientos para el prototipo software a desarrollar.

Fase 2: Selección de tecnologías y métodos

En esta fase se realizará una revisión de literatura sobre tecnologías que hay para el desarrollo de RPA y los modelos de visión computacional. A partir de esta revisión, se seleccionará las tecnologías y métodos para el desarrollo del prototipo software. Además, se revisará y seleccionará el conjunto de datos que se utilizará para el entrenamiento del modelo de visión computacional que se seleccione.

Fase 3: Construcción del componente de visión computacional

En esta fase, se construirá el componente de visión computacional que se integrará al prototipo para identificar los componentes de interfaz gráfica del software contable. Una vez construido, se realizará el modelado para el componente de visión computacional, se realizará el entrenamiento del modelo con sus respectivas pruebas. Finalmente, se contrastará los resultados obtenidos con los esperados, y se realizarán los ajustes pertinentes, y se realizará el despliegue del modelo para la siguiente fase del proyecto.

Fase 4: Desarrollo de prototipo software

En esta fase se implementará la metodología de Modelo de Gestión Incremental. Se seleccionó esta metodología debido a que su objetivo principal es el crecimiento progresivo de la funcionalidad. Es decir, el producto va a ir evolucionando con cada una de las entregas previstas hasta que este mismo logre ajustarse a lo requerido por el usuario. Esta metodología nos permitiría una fácil administración de las tareas, es un modelo propicio a los cambios o modificaciones y se adapta a las necesidades que se presenten. Una vez desarrollado el prototipo, se diseñarán unos escenarios para probar y validar el funcionamiento del prototipo.

Fase 5: Evaluación de desempeño del prototipo software.

En esta fase se realizará la evaluación del prototipo software con respecto a aspectos sobre la influencia del software en los procesos contables, específicamente en métricas, indicadores de mejora, evaluación de rendimientos y comparativa de procesos con software y sin software.

En la Tabla 3 se presentan las actividades asociadas con cada una de las etapas y sus respectivos entregables.

Tabla 3 Actividades y entregables del proyecto.

Fase	Actividad	Entregable
Fase 1: Caracterización del proceso contable	1.1 Diseño de instrumento de recolección de información	Instrumento de recolección de información
	1.2 Recolección de datos al personal contable de organizaciones públicas y privadas	Características del proceso contable y herramientas tecnológicas usadas
	1.3 Levantamiento de requerimientos del prototipo de software a partir de la caracterización realizada	Requerimientos funcionales y de calidad del prototipo de software
Fase 2: Selección de tecnologías y métodos	2.1 Revisión documental sobre tecnologías y métodos para el desarrollo del prototipo	Lista de posibles tecnologías y métodos a usar en el proyecto
	2.2 Evaluación de tecnologías y métodos	Proceso de evaluación de las tecnologías y métodos a usar en el proyecto
	2.3 Selección de tecnologías y métodos	Lista tecnologías y métodos a usar en el proyecto
Fase 3: Construcción del componente de visión computacional (CRISP-DM)	3.1 Entendimiento de datos	Conocer cómo se manejan los datos en el proceso contable
	3.2 Preparación de datos	Conjunto de datos a utilizar en el proyecto
	3.3 Selección de características	Análisis de datos y características seleccionadas
	3.4 Modelado	Modelo preliminar
	3.5 Evaluación	Modelo ajustado
	3.6 Despliegue	Modelo versión 1.0
Fase 4: Desarrollo del prototipo software.	4.1 Definición de las tareas	Lista de tareas para el desarrollo del prototipo.
	4.2 Desarrollo del prototipo de software.	Proceso de desarrollo del componente de visión computacional y proceso de desarrollo del *chatbot.*
	4.3 Pruebas del prototipo de Software.	Pruebas realizadas al componente de visión computacional conectado al *chatbot*.
	4.4 Resultado de pruebas del prototipo de software.	Videos del resultado final de la conexión entre el componente

		de visión computacional y el *chatbot.*
Fase 5: Evaluación de desempeño del software RPA en un ambiente simulado.	5.1 Definición de métricas usadas por el software	Lista de métricas usadas por el software
	5.2 Indicadores de mejora de procesos	Documento con mejora de procesos usando el software
	5.3 Evaluación de rendimientos	
	5.4 Comparativa de procesos con software y sin software.	Informe comparativo del rendimiento de procesos con software y sin software.

Fuente: Autoras, (2020).

7. RESULTADOS

Con el desarrollo del Proyecto de Investigación, se espera obtener los siguientes resultados:

7.1 CARACTERIZACIÓN

Consiste en identificar condiciones y/o elementos que hacen parte del proceso contable para saber cómo se hace y que se requiere para su funcionamiento.

Esta sección está dada en dos subsecciones. En primer lugar, la sección 7.1.1 que presenta el diseño del instrumento de recolección de información con sus respectivos resultados. Asimismo, muestra el proceso contable de manera manual y automatizada. En segundo lugar, la sección 7.1.2 que presenta el levantamiento de los requerimientos tanto funcionales como no funcionales para el prototipo software, a partir, de los resultados obtenidos en la sección 7.1.1.

7.1.1 Caracterización del proceso contable y las tecnologías usadas en el mismo. Se presenta la caracterización del proceso contable, su diseño de instrumento de recolección de información, con los respectivos resultados y la descripción del proceso contable de manera manual y automático con sus diagramas de proceso.

7.1.1.1 Diseño de instrumento de recolección de información. La recolección de la información para el levantamiento de requerimientos del prototipo de software de creación de RPA se realizó por medio de una encuesta. Esta medida se tomó dado por las circunstancias que se presentaron por la contingencia causada por la pandemia del COVID-19. Las restricciones generadas por la pandemia dificultaron encontrar una empresa pública que permitiera hacer un pilotaje del proceso, levantar los requerimientos y realizar las pruebas del prototipo. El objetivo de la encuesta es caracterizar el proceso contable y las tecnologías usadas en dicho proceso.

Como instrumento de recolección de información se diseñó una encuesta de 10 preguntas que se realizaron a contadores de empresas públicas y privadas. Para ello, se utilizó la herramienta Google formularios y con base a los resultados se obtienen los requerimientos para el desarrollo de la RPA.

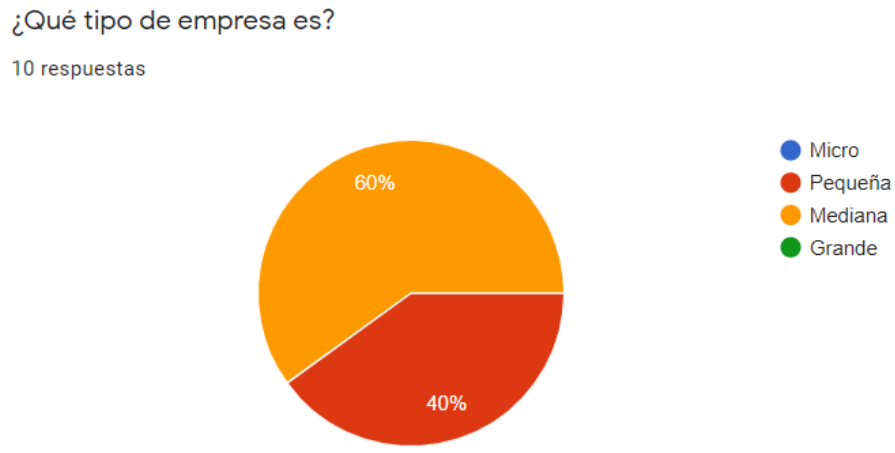
7.1.1.2 Presentación de los resultados. Se aplicaron 10 encuestas a empresas públicas y privadas en la ciudad de Bucaramanga. A continuación, se presentan los resultados por cada pregunta realizada.

Figura 12 Sector Empresarial en el que labora.

Fuente: Autoras, (2020).

A partir de la Figura 12 podemos observar que, del total de encuestados, el 70% pertenece a empresas públicas.

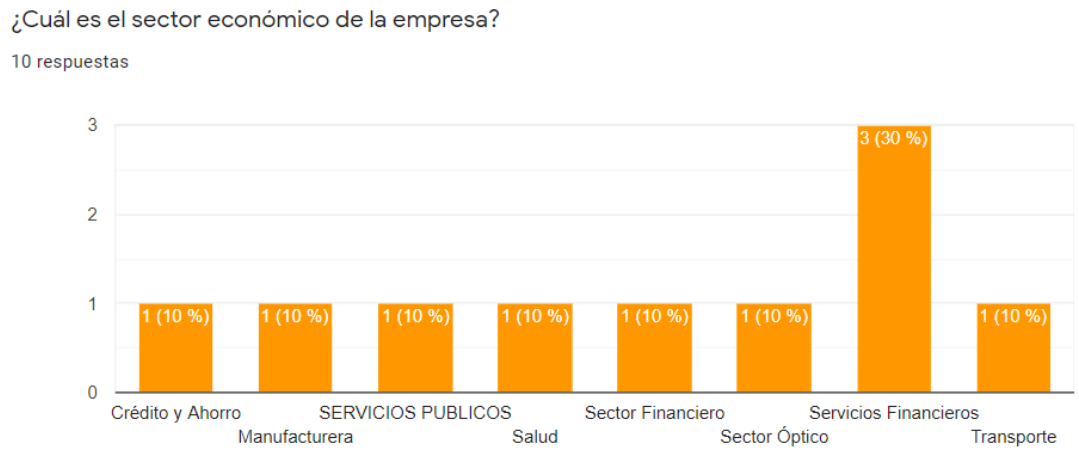
Figura 13 Tipo de Empresa.



Fuente: Autoras, (2020).

De acuerdo con la Figura 13, el 60% pertenece a mediana empresa y el 40% a pequeña empresa.

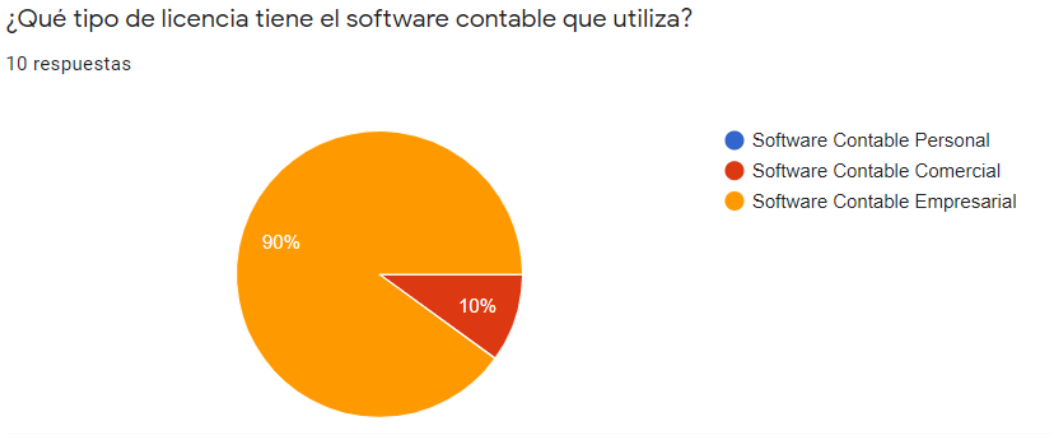
Figura 14 Sector económico de la empresa.



Fuente: Autoras, (2020).

Según Figura 14, el sector económico que tuvo mayor uso de software contable fue el de servicios financieros con el 30%.

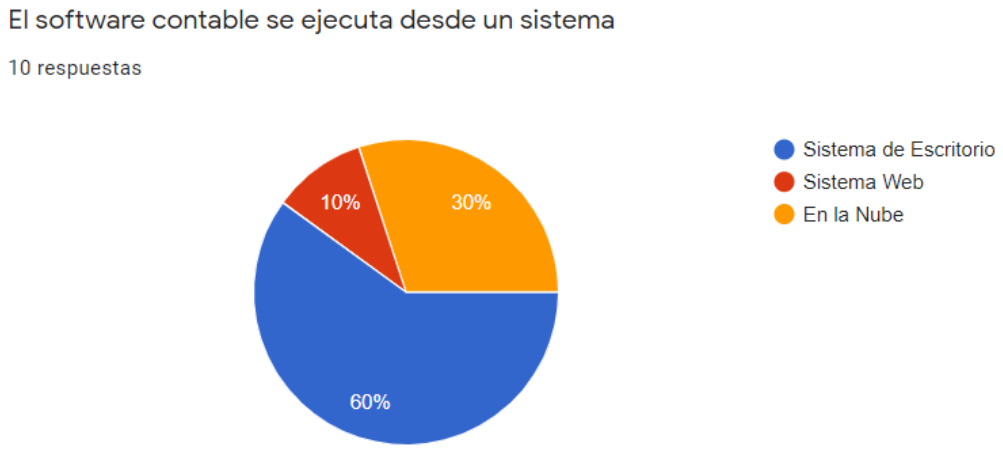
Figura 15 Tipo de licencia del software.



Fuente: Autoras, (2020).

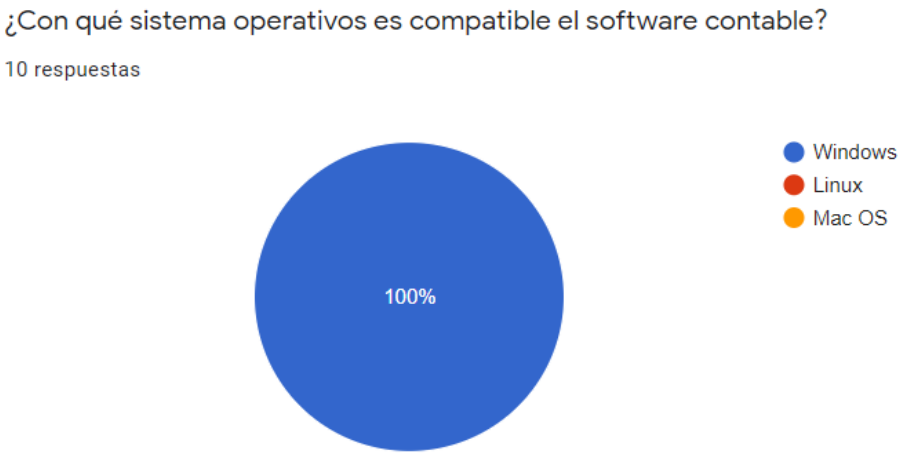
Según con la Figura 15, se puede observar que el 90% de los encuestados usa software contable Empresarial y el 10% restante software comercial.

Figura 16 Sistema de ejecución del software contable.



Fuente: Autoras, (2020).

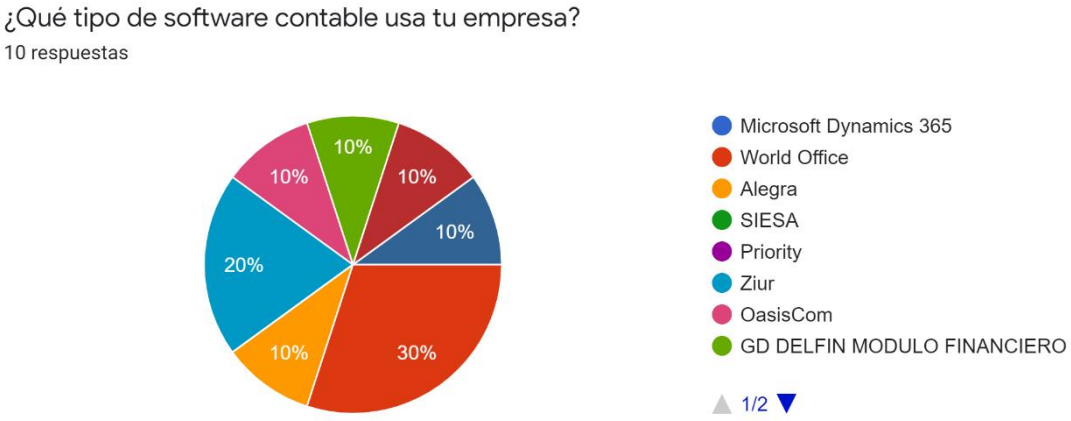
Figura 17 Sistema operativo compatible al software contable.



Fuente: Autoras, (2020).

La Figura 17 evidencia el sistema operativo utilizado por las empresas encuestadas, en el que se evidencia que el 100% de los softwares contables nombrados en la encuesta son compatibles con el sistema operativo Windows.

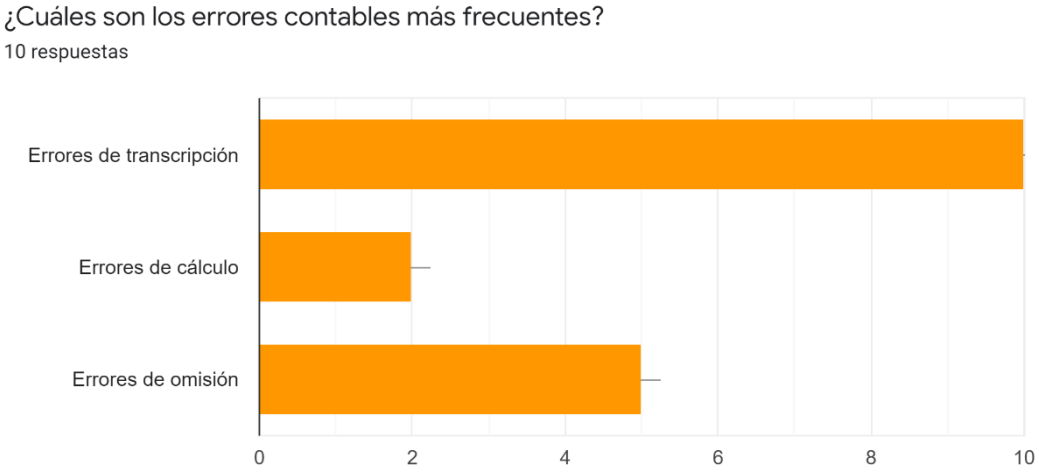
Figura 18 Software contable de la empresa.



Fuente: Autoras, (2020).

Según la Figura 18, el 30 % de los encuestados usa el *Software Contable World Office*, el 20% Ziur, el 10% *Microsoft Dynamics* 365, el 10% GD Delfin Modulo Financiero, el 10% SIESA y el 10 % Foxconn.

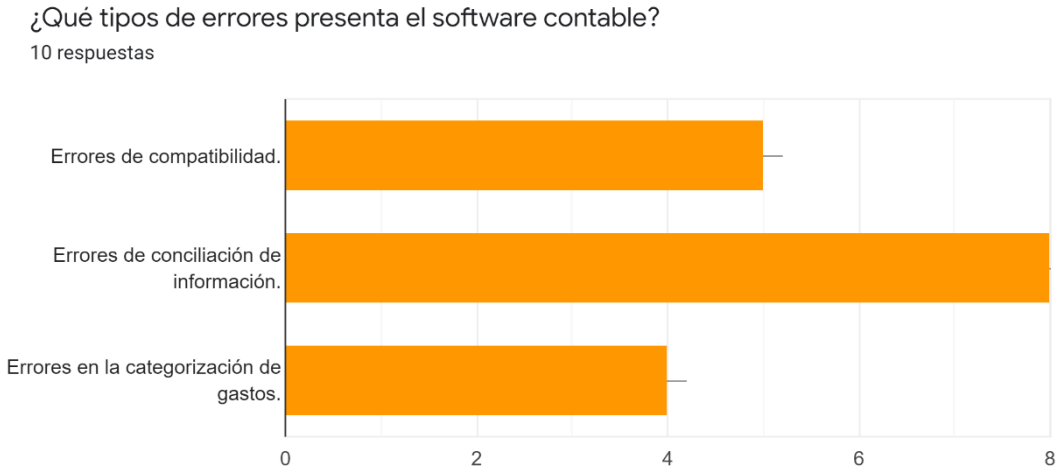
Figura 19 Errores Contables.



Fuente: Autoras, (2020).

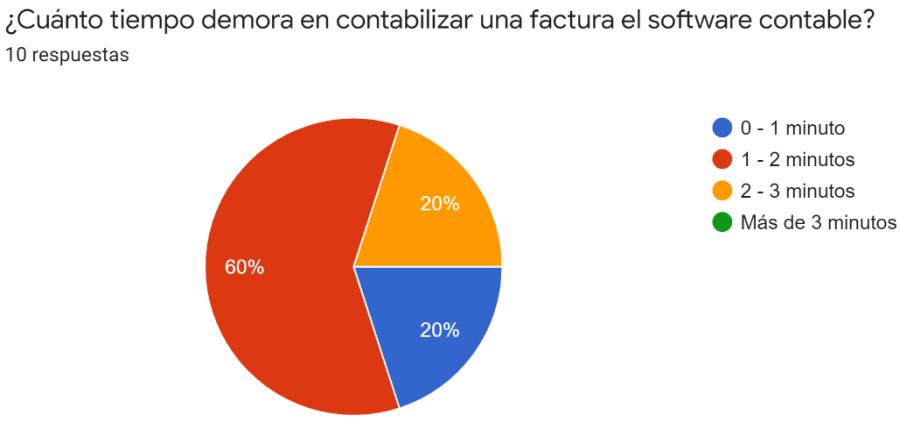
Según la Figura 19, los errores contables más comunes son errores de transcripción y errores de omisión.

Figura 20 Tipos de errores contables.



Fuente: Autoras, (2020).

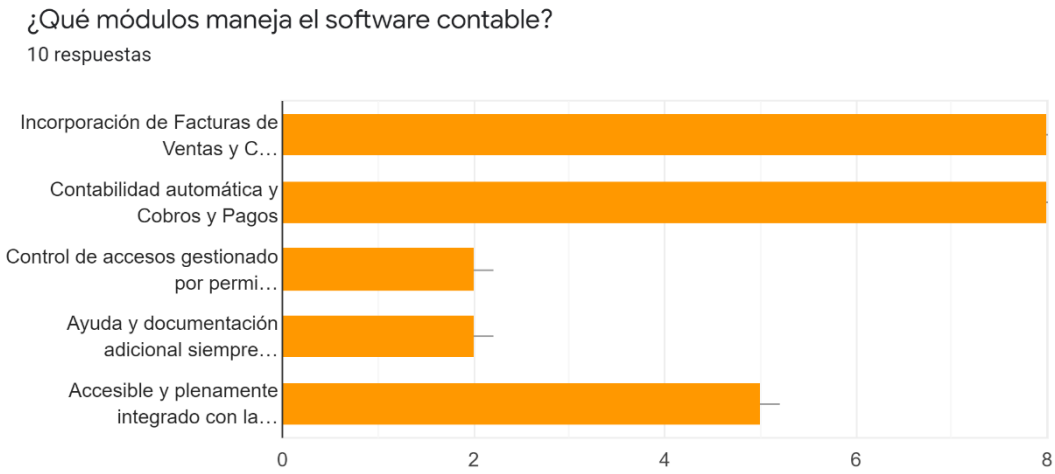
Figura 21 Tiempo de contabilización de una factura.



Fuente: Autoras, (2020).

A partir de la Figura 21, se puede evidenciar que el 60% a la hora de contabilizar una factura demora de 1-2 minutos, el 20% 0-1 minuto y el 20% de 2-3 minutos.

Figura 22 Módulos del software contable.



Fuente: Autoras, (2020).

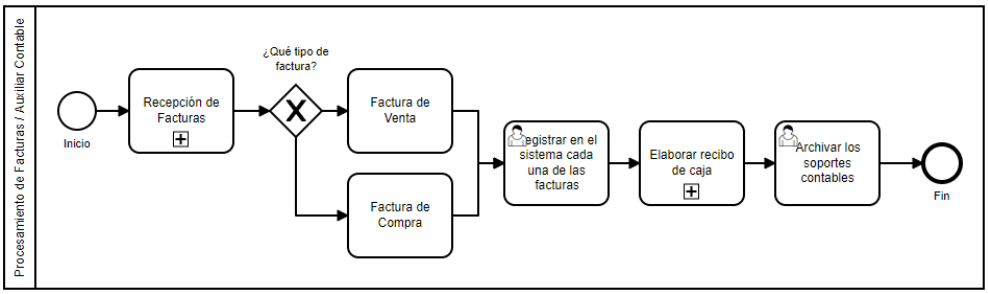
Los módulos presentes en los softwares contables nombrados son incorporación de facturas de ventas y compras, contabilidad automática y cobros y pagos, control de acceso gestionado por permisos, ayuda y documentación adicional. Por último, accesible y plenamente integrado con Word y Excel.

7.1.1 **Descripción del proceso contable.** A continuación, se presenta la descripción del proceso contable desde dos perspectivas: el proceso manual y el proceso automatizado del software contable con una factura de venta.

- **Proceso Manual**

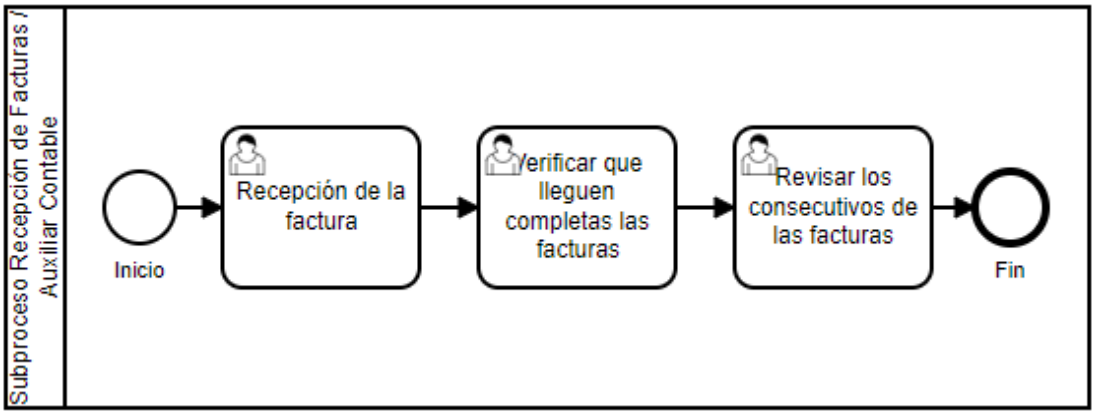
En la Figura 23 se presenta el flujo de trabajo del procesamiento de las facturas, realizada por el auxiliar contable dentro de una organización de manera manual y subprocesos presentes en la Figura 24, recepción de facturas y Figura 25, elaboración de recibo de caja.

Figura 23 Diagrama de proceso de facturas.



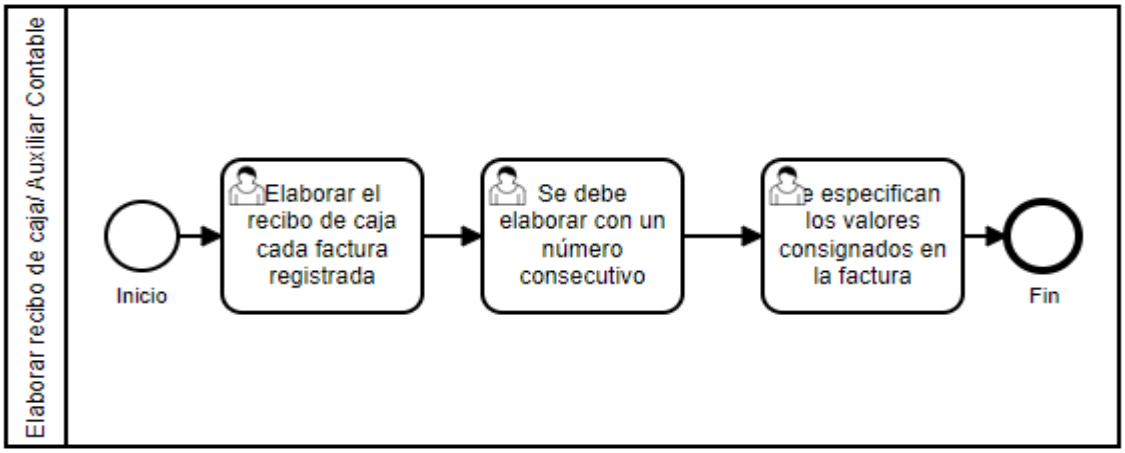
Fuente: Autoras, (2020).

Figura 24 Diagrama subproceso recepción de facturas.



Fuente: Autoras, (2020).

Figura 25 Diagrama Subproceso creación recibo de caja.

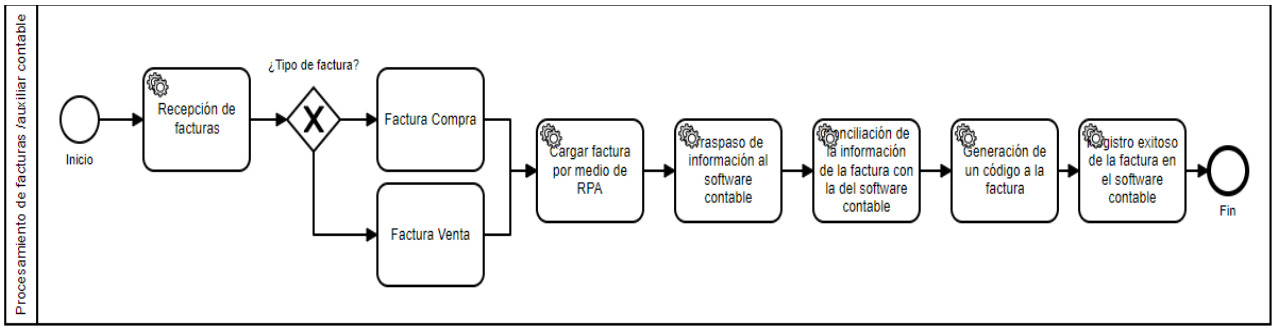


Fuente: Autoras, (2020).

- **Proceso Automatizado**

En la Figura 26, se presenta el flujo de trabajo del procesamiento de las facturas, realizada por el auxiliar contable dentro de una organización de manera automatizada.

Figura 26 Diagrama de procesamiento de facturas automatizado.



Fuente: Autoras, (2020).

7.1.2 Levantamiento de Requerimientos. A continuación, se presentan los requerimientos funcionales y no funcionales del prototipo de software a desarrollar.

7.1.2.1 Requerimientos funcionales (RF).

- RF01: El componente de visión computacional debe identificar los elementos de interfaz de usuario del sistema contable a partir de un video.

- RF02: El componente de visión computacional debe detectar los elementos de la interfaz de usuario del sistema contable por medio de la técnica de reconocimiento de color.

- RF03: El componente de visión computacional debe reconocer el elemento de interfaz de usuario y extraer el texto que lo acompaña (por ejemplo, detectar el botón y extraer el nombre de este). Una vez realizado este proceso se debe generar un archivo de etiquetado en formato de texto plano (.txt) para los elementos de interfaz de usuario identificados.

- RF04: El sistema debe generar el código de la RPA a partir del archivo de etiquetado en formato de texto plano (.txt) generado por el componente de visión computacional.
- RF05: El sistema debe permitir ejecutar el código de la RPA de interfaz gráfica generado para validar su funcionamiento.

- RF06: El usuario interactúa con el sistema a través de un chatbot usando lenguaje natural para realizar acciones del sistema como generar y ejecutar la RPA.

7.1.2.2 Requerimientos no funcionales (RNF)

- RNF01: El sistema se debe ejecutar en sistemas operativos Windows para computadores.

- RNF02: La interacción del usuario con el sistema debe ser amigable.

7.2 SELECCIÓN DE TECNOLOGÍAS Y MÉTODOS

Esta sección, permite seleccionar la tecnología y métodos adecuados para llevar a cabo el trabajo dentro de una organización y garantizará la compatibilidad de estás en el prototipo a realizar.

7.2.1 Revisión documental sobre tecnologías y métodos para el desarrollo del prototipo. A continuación, se presenta la revisión documental sobre las tecnologías y métodos más adecuados para el desarrollo del prototipo de software clasificados de la siguiente forma: algoritmos de detección de objetos y localización de estos, *frameworks* enfocados al aprendizaje automático, *framework* para el desarrollo de la RPA (Automatización Robótica de Procesos), lenguajes de programación que se pueden llegar a usar en el desarrollo del prototipo software, softwares de desarrollo de RPA (Automatización Robótica de Procesos), entornos de desarrollo para el prototipo software, herramientas y métodos para la transformación del lenguaje natural, y herramientas de bases de datos SQL para el desarrollo del prototipo software.

7.2.1.1 Algoritmos de detección de objetos. La detección de objetos combina la clasificación de objetos y la localización de objetos, lo cual lo convierte en uno de los temas más desafiante en el campo de la visión por computadora. El principal objetivo de estos modelos es determinar la ubicación de varios elementos en una determinada imagen y clasificarlos. A continuación, se nombra algunos de los algoritmos para la detección de objetos más conocidos.

Fast R-CNN

Es un modelo escrito en Python y C++ (*Caffe*) de código abierto, la arquitectura de una red *Fas*t R-CNN toma la entrada de una imagen completa y un conjunto de regiones de interés (RoI) se ingresan en una red totalmente convolucional (Choudhury Ambika, 2020). Cada región de interés se agrupa en un mapa de características de tamaño fijo y luego se asigna a un vector de características mediante capas completamente conectadas (FC). La red tiene dos vectores de salida por RoI: probabilidades *softmax* y compensaciones de regresión de cuadro delimitador por clase. La arquitectura se entrena de un extremo a otro con una pérdida de múltiples tareas(Girshick, 2015). Este algoritmo corrige las desventajas de R-CNN y SPPnet, al tiempo que mejora su velocidad y precisión. Además, es un modelo rápido de entrenar y de probar (Choudhury Ambika, 2020).

Sus principales ventajas son:

- Mayor calidad de detección (mAP)
- El entrenamiento es de una sola etapa, usando una pérdida de múltiples tareas.
- La formación puede actualizar todas las capas de la red.
- No se requiere almacenamiento en disco para el almacenamiento en caché de funciones.

Faster R-CNN

Es un algoritmo de detección de objetos similar a R-CNN. Es un algoritmo basado en redes convolucionales, que utiliza una red de detección de objetos y una de propuestas regionales –RPN– lo cual toma una imagen de cualquier tamaño como entrada y genera un conjunto de propuestas de objetos rectangulares, cada una con una puntuación de objetividad en cada posición (Ren, Shaoqing, Kaiming He, Ross Girshick, 2017). La arquitectura se entrena de extremo a extremo para generar propuestas de región de alta calidad, que luego son utilizadas por Fast R -CNN para detección de objetos (Choudhury Ambika, 2020).

Histogram of Oriented Gradients (HOG)

El histograma de gradientes orientados – HOG, es un descriptor de características que se utiliza para detectar objetos en la visión por computadora y el procesamiento de imágenes. La técnica del descriptor HOG cuenta las ocurrencias de orientación de gradiente en porciones localizadas de una ventana de detección de imágenes o región de interés –ROI. Es un algoritmo simple y de fácil compresión de la información que contiene (*HOG Descriptor*, 2020).

Region-based Convolutional Neural Networks (R-CNN)

El método de red convolucional basada en regiones – RCNN, es una combinación de propuestas regionales con redes neuronales convolucionales –CNN. R-CNN ayuda a localizar objetos con una red profunda y a entrenar un modelo de alta capacidad con solo una pequeña cantidad de datos de detección anotados (Choudhury Ambika, 2020).

El sistema toma una imagen, extrae un conjunto de alrededor de 2000 propuestas de regiones ascendentes, calcula las características de cada propuesta usando una red neuronal convolucional y luego clasifica región usando SVM lineales específicas de clase R-CNN (Girshick et al., 2014):

Region-based Fully Convolutional Network (R-FCN)

Es un algoritmo de detección de objetos basados en regiones que, mediante redes profundas y totalmente convolucionales, permite la detección de objetos de una forma precisa y eficiente. R-FCN puede adoptar de forma nativa potentes redes troncales de clasificadores de imágenes totalmente convolucionales, como ResNets, para la detección de objetos (Choudhury Ambika, 2020). la arquitectura R-FCN está diseñada para clasificar los RoI en categorías de objetos y antecedentes. En R-FCN, todas las capas de peso que se pueden aprender son convolucionales y se calculan en la imágen completa (Jifeng Dai, Yi Li, Kaiming He, 2016).

Single Shot Detector (SSD)

El detector de disparo único (SSD) es un método para detectar objetos en imágenes utilizando una única red neuronal profunda (Choudhury Ambika, 2020). El enfoque SSD se basa en una red convolucional de retroalimentación que produce una colección de tamaño fijo de cuadros delimitadores y puntuaciones para la presencia de instancias de clase de objeto en esos cuadros, seguido de un paso de supresión no máxima para producir las detecciones finales. Las primeras capas de la red se basan en una arquitectura estándar utilizada para la clasificación de imágenes de alta calidad (Liu et al., 2015), fácil de entrenar y sencillo de integrar en sistemas que requieren un componente de detección

YOLO (You Only Look Once)

Es uno de los algoritmos de detección de objetos más utilizado, ya que su arquitectura unificada es extremadamente rápida. El modelo base de YOLO procesa imágenes en tiempo real a 45 cuadros por segundo, mientras que la versión más pequeña de la red, *Fast* YOLO procesa una cantidad de 155 cuadros por segundo mientras logra el doble de mAP que otros detectores en tiempo real. Este algoritmo supera a los otros métodos de detección, incluidos DPM y R-CNN (Redmon, 2016).

El modelo de YOLO v3 es más rápido y preciso. Puede intercambiar fácilmente entre velocidad y precisión simplemente cambiando el tamaño del modelo, además que no se requiere reentrenamiento (Redmon joseph, n.d.) .

7.2.1.3 Framework para Desarrollo de RPA

Robot Framework

Robot Framework es una solución de automatización de procesos robóticos (RPA) de código abierto que se utiliza para automatizar procesos comerciales. su arquitectura modular que se basa en librerías de extensión. Estas librerías brindan funcionalidades como reconocimiento óptico de imágenes, acceso a bases de datos, API, HTTP, compatibilidad con aplicaciones iOS y Android y ejecución remota. Además, permite crear

nuevas librerías a partir de código Python o Java que permite a cualquier usuario integrar nuevas funciones en Robot Framework (Robot Framework, 2008).

Robot Framework es independiente del sistema operativo y la aplicación. El marco principal se implementa con Python y también se ejecuta en Jython (JVM) e IronPython (.NET).

7.2.1.4 Software de desarrollo de RPA

Existen diferentes proveedores en el mercado en RPA, entre los principales proveedores líderes están Automation Anywhere, UiPath y Blue Prism.

Automation Anywhere: Es un proveedor de software de automatización de procesos robóticos (RPA). Esta solución de nivel empresarial combina sofisticadas tecnologías RPA, IA y analíticas integradas para crear bots de software que automaticen y administren tareas de front y back office (Capterra, 2020).

Su implementación puede ser en la nube, SaaS, web, instalado en sistema operativo Windows y móvil (Capterra, 2020), es compatible para integrarse con sistemas de terceros, facturación, ERP, Analytics. Y con la capacidad de extraer datos de diferentes tipos de archivos y fuentes su licencia es pago pero cuenta con una licencia gratis de 30 días (Becerra et al., 2019).

UiPath: Es una plataforma de alto nivel dedicada a proporcionar una automatización perfecta de la entrada de datos en cualquier formulario web y aplicación de escritorio. Es compatible con Excel y proporciona integración con SAP y Citrix. Se puede implementar en la nube, SaaS, web e instalación en sistemas operativos Windows, iOS y Android. (UiPath, 2019) está bajo una licencia de pago pero cuenta con una licencia gratis por 60 días.

Blue Prism: Es un proveedor líder de modelado de procesos, documentación y soluciones de optimización para organizaciones empresariales (capterra.co, 2020). Este software se puede implementar en la nube, SaaS, web y en sistemas operativos de Mac y Windows. Su licencia es de pago y no cuenta con licencias de pruebas o versiones gratis.

7.2.1.5 Lenguaje de Programación

Python

Es un lenguaje de programación que está orientado a objetos y es muy sencillo de utilizar al mismo tiempo que guarda toda su potencia. Es de código abierto, algunas de sus características son: sencillez, exactitud en la síntesis, buena legibilidad, poder programar

algoritmos complejos en pocas líneas y también usar los dialectos, unas variantes adaptables a otros lenguajes (Sevilla, 2018).

Jython (JVM)

Jython es una implementación del lenguaje de scripts Python, escrito en el lenguaje Java e integrado con la plataforma Java. Jython proporciona las características de productividad de un lenguaje de script maduro y, a diferencia de Python, se ejecuta en cualquier entorno que soporte una máquina virtual Java -JVM (*Jython*, n.d.).

IronPython (.NET)

IronPython es una implementación de código abierto del lenguaje de programación Python que está estrechamente integrado con .NET Framework. IronPython puede usar las bibliotecas .NET Framework y Python, y otros lenguajes .NET pueden usar el código Python con la misma facilidad (*IronPython.Net* , n.d.).

El lenguaje de programación e intérprete que se seleccionaría para trabajar en Robot Framework se selecciona el lenguaje de programación de Python, al ser un lenguaje sencillo de trabajar, de entender y de implementar.

7.2.1.6 Entorno de Desarrollo

Anaconda Individual Edition Python

Anaconda Individual Edition es un administrador de paquetes, administrador de entorno y distribución de Python gratuito y fácil de instalar con una colección de más de 1,500 paquetes de código abierto. Esta Suite es multiplataforma y se puede utilizar para Windows, Linux y Macintosh (*Anaconda Documentation*, n.d.).

Google Collaboratory

Google Collaboratory es un entorno de máquinas virtuales basado en Jupyter Notebooks. Permite ejecutar y programar en Python en el navegador, no requiere configuración, además permite el acceso gratuito a GPUs. Permite ejecutar en tres entornos CPU, GPU y TPU. El ambiente de trabajo ya viene con librerías instaladas listas para utilizar. Sin embargo, presenta algunas restricciones, por ejemplo, una sesión tiene una duración de 12 horas, pasado ese tiempo se restablece el ambiente, lo cual se pierde las variables y archivos que se hayan almacenado (Martinez Paula, 2019).

7.2.1.7 Procesamiento de Lenguaje Natural (NLP)

Generación de lenguaje natural: se basa en la conversión de los datos almacenados en una base de datos, o indentaciones semánticas, a un lenguaje legible para los humanos (J. Bagnato, 2018)

- Entendimiento de lenguaje natural: consiste en convertir segmentos de texto en representaciones tales como la lógica de primer orden.
- Reconocimiento de texto en imágenes (OCR): esta aplicación busca obtener texto a partir del texto que pudiese contener una imagen.
- Respuesta de preguntas: consiste en que la computadora pueda responder a preguntas formuladas en lenguaje humano.
- Reconocimiento de implicación textual: determina la relación entre dos fragmentos de texto en función de si el hecho de que una sea cierta podría resultar en que sea falsa la otra o viceversa (J. Bagnato, 2018).

Pix2Code

Se enfocan en la transformación de un *screenshot* a código por medio de aprendizaje profundo usando Pix2code basado en redes Neuronales recurrentes y convolucionales generando tokens de computadora. Lo anterior, permitió la generación de código para varias plataformas tales como interfaces nativas de iOS y Android, e interfaces web basadas en múltiples plataformas HTML/CSS (Beltramelli, 2018).

Librería NLTK

Usa la librería de Python NLTK para NLP (Procesamiento de Lenguaje Natural) junto con funciones y análisis tradicionales para transformar por medio de webscraping el blog de cuentos de Hernan Casciari, creando archivos de texto (.txt) para pasarlo a una estructura de *Dataframe* de Pandas para seguir con su uso. Después, se realiza una limpieza de estos datos para tratar el texto, es decir, pasar el texto a minúsculas, sin signos de puntuación, espacios extra, tabulaciones, etc. Tiene un diseño modular que lo hace ideal para adaptarse a distintas problemáticas por medio de la composición de módulos. NLTK ha ayudado a resolver varios problemas asociados a diferentes aspectos del procesamiento de lenguaje natural. Existen libros y guías que sirven como ayuda para poder utilizar este módulo. Involucrarse con NLTK sin leer estas guías es casi una tarea imposible; este módulo tiene una curva de aprendizaje compleja y posee varias limitaciones internas. Este último aspecto constituye su principal desventaja (Ana & Di, 2019)**.**

CoreNLP

Este módulo fue desarrollado por el grupo Stanford. Está escrito en Java y es conocido por su excelente velocidad. Ofrece funciones tales como *Part-of-Speech* (POS), etiquetado, aprendizaje de patrones y reconocimiento de entidades entre otras. A pesar de estar escrita en Java, posee *wrappers* para utilizar otros lenguajes tales como Python. Es una librería ampliamente usada en ambientes de producción y su implementación ha sido mejorada con el pasar de los Años (Ana & Di, 2019)**.**

TextBlob

Esta librería no puede considerarse como una librería NLP por sí misma. En su lugar, se trata de una interfaz montada sobre NLTK y que agrega componentes tales como analizadores de sentimiento. Su uso permite resolver parte de la complejidad de NLTK, al mismo tiempo que permite crear rápidamente prototipos (Ana & Di, 2019)**.**

Gensim

Este módulo presenta limitaciones en cuanto a la versatilidad que tienen los enumerados anteriormente. Sin embargo, su punto fuerte radica en la comparación entre distintos documentos. Gensim permite realizar un análisis que determine las similitudes entre dos documentos (Ana & Di, 2019)**.**

SpaCy

Esta librería open source está escrita en Cython. Mientras que NLTK ofrece más de 50 variantes para abordar la solución de un problema, SpaCy solo ofrece una. Si bien esto puede parecer contraproducente, se compensa porque en lugar de tener que encontrar el camino óptimo, en la mayoría de los casos la solución provista por SpaCy es la mejor. En términos de rendimiento, se puede considerar que esta librería tiene una performance similar al de una librería escrita en C (Ana & Di, 2019)**.**

Dado lo anterior, la tecnología más adecuadas para usar son SpaCy y NLTK, están basadas en Python, resulta más sencillo que el resto del desarrollo también sea hecho en Python. Cabe resaltar que en curva de aprendizaje Spacy es mucho más eficiente que otras librerías, además el tokenizador de Spacy puede ser completamente personalizado. La arquitectura de esta librería está basada en Cython, factor que permite ofrecer una API sencilla en Python con un rendimiento similar al de un programa escrito en C. Por lo antes expuesto, la performance puede ser comparable a la de CoreNLP pero este no es Open Source. La desventaja que presenta Spacy frente a otras librerías es que solo ofrece una variante para solucionar el problema a diferencia de NLTK que presenta 50.

7.2.2 EVALUACIÓN DE TECNOLOGÍAS Y MÉTODOS. Dada la revisión documental presente en la sección 7.2.1, se realiza la evaluación de las tecnologías y métodos que son más compatibles con cada una de las herramientas a usar, para el buen funcionamiento y desarrollo del prototipo software.

Lista de tecnologías y métodos seleccionados para el desarrollo del prototipo software. Para la selección del algoritmo de detección de objetos a utilizar en el módulo de visión artificial, a partir de las características que presenta cada uno los algoritmos

mencionados. En la Tabla 4 se muestra las características principales que se tuvieron en cuenta para su selección.

Tabla 4 Criterios para selección de algoritmo de detección de objetos.

Características / Modelo	**Fast R-CNN**	**Faster R-CNN**	**HOG**	**R-CNN**	**R-FCN**	**SSD**	**YOLO**
Modelo rápido y fácil de entrenar	✔					✔	✔
Velocidad y precisión	✔	✔			✔	✔	✔
Algoritmo simple y de fácil compresión	✔	✔	✔	✔	✔	✔	✔
Sencillo de integrar en sistemas	✔	✔	✔	✔	✔	✔	✔
Detección de objetos en videos	✔	✔	✔	✔	✔	✔	✔
Código abierto	✔	✔	✔	✔	✔	✔	✔
Windows y Linux			✔	✔	✔	✔	✔
Requisitos de Hardware mínimas			✔	✔		✔	✔

Fuente: Autoras, (2020).

Para la selección de las herramientas a trabajar a partir de las características ya sea en el *framework* de RPA o en el sistema de software de RPA anteriormente mencionados, se realizó una matriz de calificación en donde se muestra las principales características que se tuvieron en cuenta. Determinando una ponderación de 0,2 de importancia para cada aspecto evaluándose en 1 no cumple, 2 cumple parcialmente y 3 cumple.

Tabla 5 Criterios de selección para framework de RPA.

Característica / framework o sist software de RPA	Ponderación	Robot Framework	Automation anywhere	Ulpath	Blueprism
Compatibilidad en S.O Windows	0,2	3	3	3	3
Compatibilidad con lenguaje Python	0,2	3	1	2	1
Reconocimiento óptico de imágenes	0,2	3	1	3	1
Licencia open source o licencia de prueba gratis	0,2	3	2	2	1
Fácil implementación en desarrollo de código	0,2	2	1	2	1
Total	1	2,8	1,6	2,4	1,4

Fuente: Autoras, (2020).

En la selección de la herramienta a trabajar para el entorno de desarrollo en la que se desarrollará el entrenamiento del algoritmo de visión. Se realizó la Tabla 6, en donde se muestra las principales características que se tuvieron en cuenta. Determinando una ponderación de 0,2 de importancia para cada aspecto evaluándose en 1 no cumple, 2 cumple parcialmente y 3 cumplen.

Tabla 6 Criterios para selección entorno de desarrollo.

Características / Herramienta	Ponderación	Anaconda	Google Collaboratory
Compatibilidad con lenguaje de Python	0,2	3	3
Multiplataforma	0,2	3	3
Fácil configuración de librerías	0,2	3	3
Open source	0,2	3	2
Sin restriccion de tiempo en las sesiones	0,2	3	2
Total	1	3	2,6

Fuente: Autoras, (2020).

7.2.3 SELECCIÓN DE TECNOLOGÍAS Y MÉTODOS. En la Tabla 7, se puede observar el proceso, características y tecnologías seleccionadas para el desarrollo del prototipo software.

Tabla 7 Tecnologías y métodos seleccionados para el desarrollo del prototipo software.

Proceso	Característica	Tecnologías usadas
Detección de Objetos	Detección de una región determinada con coordenadas y su clasificación.	YOLO[1]
Framework de RPA	Marco de trabajo para la creación de la RPA	Robot Framework
Lenguaje de Programación	Lenguaje que se usó que se adapte a las tecnologías a usar	Python
IDE	Entorno de desarrollo	Anaconda
Procesamiento de Lenguaje Natural NLP	Campo para la interacción entre máquina y lenguaje humano.	Librería NLTK

Fuente: Autoras, (2020).

7.3 COMPONENTE DE VISIÓN COMPUTACIONAL

En este capítulo, se describe el proceso de construcción del componente de visión computacional que se implementó en el prototipo software. A continuación, se presentan las actividades realizadas.

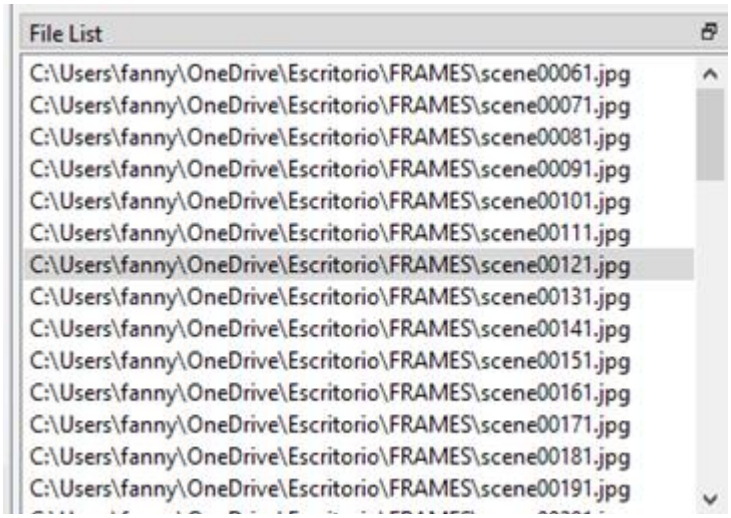
En la sección 7.3.1, se encuentra el proceso de entendimiento de datos. En la sección 7.3.2, se muestra la preparación de datos y sus características. En la sección 7.3.3, se realiza el análisis de los datos y a partir de los resultados del análisis se realiza un cambio en la tecnología a usar. En la sección 7.3.4, se define la tecnología para el desarrollo del componente de visión computacional. En la sección 7.3.5, se desarrolla el componente de visión computacional.

7.3.1 Entendimiento de datos. En esta sección, se presenta cómo funcionan los datos con el algoritmo YOLO para detectar los objetos y, por medio de *LabelImg*, obtener y etiquetar las imágenes. A partir de ello, se genera un archivo de etiquetado por imagen en formato de texto. Después, se realiza el entrenamiento del algoritmo YOLO con los archivos generados en formato de texto del etiquetado de las imágenes.

7.3.2 Preparación de datos y selección de características. En esta sección, se muestra el paso a paso para la preparación de los datos, el conjunto de imágenes de interfaz de usuarios de software de escritorio para el entrenamiento del algoritmo YOLO se construyó con imágenes obtenidas en videos y de internet. Lo anterior, debido a que no se encontró un *dataset* de GUI de escritorio. Para la preparación de las imágenes se realizó lo siguientes pasos:

1. las imágenes se obtuvieron en la web y en videos.
2. las imágenes seleccionadas fueron las que cumplían con las características necesarias.
3. Se tomaron *frames* de diferentes softwares contables para realizar el etiquetado de las imágenes, como se aprecia en la Figura 27.

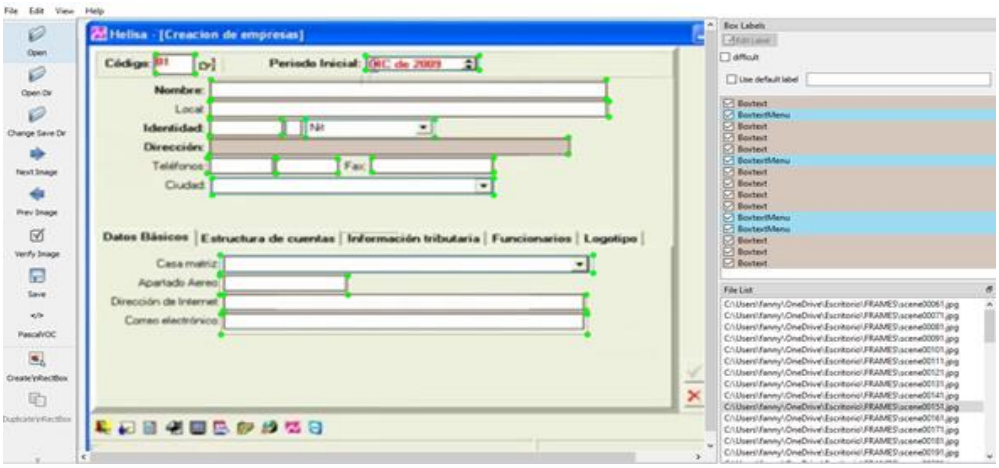
Figura 27 Lista de Frames.



Fuente: Autoras, (2020).

4. En la Figura 28, se muestra el etiquetado *LabelImg* y se carga la carpeta de *frames*.

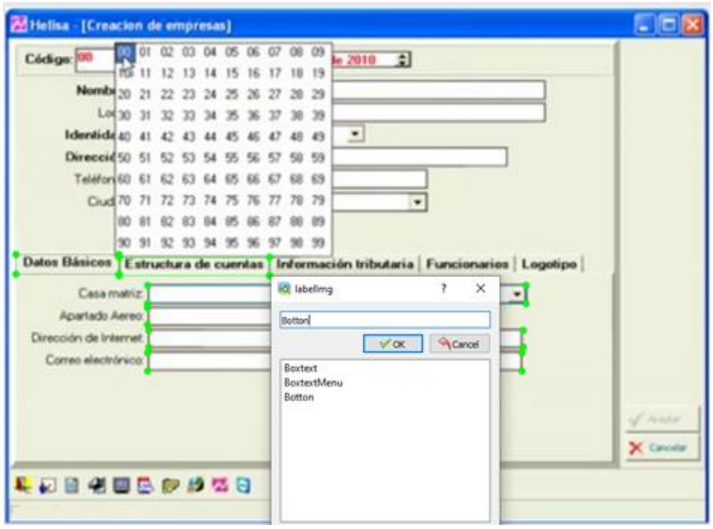
Figura 28 Etiquetador LabelIng.



Fuente: Autoras, (2020).

5. En la Figura 29, se selecciona la región donde se encuentra el control de GUI (Interfaz gráfica de usuario) y se le asigna un nombre al etiquetador.

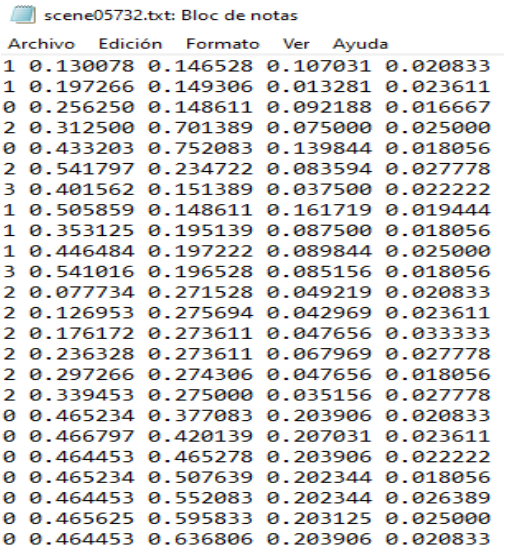
Figura 29 Selección de región de imágenes.



Fuente: Autoras, (2020).

6. Luego de terminar el etiquetado, las coordenadas de cada una de las regiones seleccionadas en cada imagen se guardan los archivos generados en formato de archivo plano (.txt). En Figura 30 se muestra un ejemplo de cómo se visualizan las coordenadas de una imagen.

Figura 30 Coordenadas de un frame.



Fuente: Autoras, (2020).

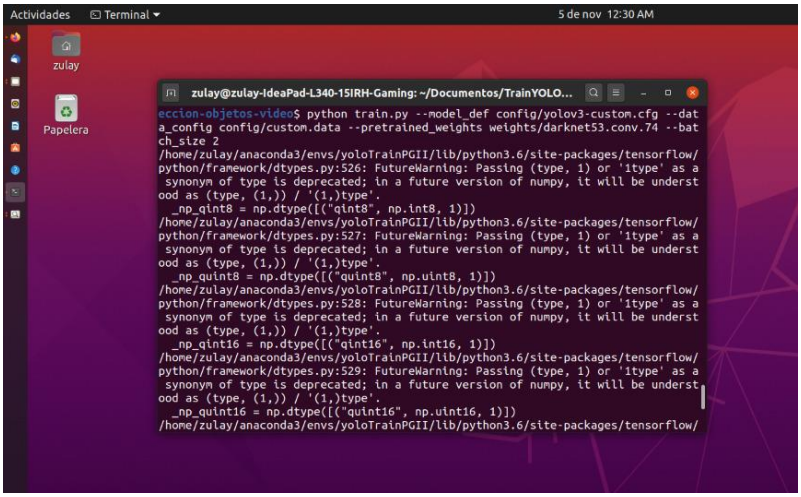
7. Una vez se han etiquetado todas las imágenes, se realiza el entrenamiento del algoritmo YOLO con el conjunto de imágenes construido con sus respectivas etiquetas.

7.3.4 Modelado. El modelo seleccionado para el desarrollo del componente de visión. Se seleccionó el modelo de YOLO v3. Este modelo está pre entrenado para detectar 80 diferentes clases de objetos.

En este caso se entrenó el algoritmo con las clases definidas a nuestras necesidades que son los siguientes *textbox*, *botton*, *BoxtextMenu* y *Checkbox* de las cuales son elementos que se encuentra en la GUI. Después de realizar el etiquetado de las imágenes en formato VOC se realiza el entrenamiento con las imágenes con su determinada metadata. Se divide en dos bloques, un bloque para entrenamiento y otro para validación durante el entrenamiento.

7.3.5 Evaluación. El algoritmo presentado inicialmente fue pensado para desarrollarse con la técnica de YOLO ya que este es un modelo que está optimizado para la detección de objetos a una velocidad alta. lo cual permite una mejor detección de objetos en videos. Razón por la cual se seleccionó para lograr detectar los diferentes elementos GUI que se encontrara en un video para así obtener las coordenadas de las posiciones en que se encontrara estos elementos. Pero este camino no se pudo seguir debido a que no se encontró un *dataset* disponible de imágenes de interfaz gráfica de escritorio. Así que se creó un *dataset* con imágenes obtenidas de la red y videos. De lo cual, al momento del entrenamiento parte de las imágenes se presentaba dañadas y la cantidad de imágenes que se pueden entrenar como se observa en la Figura 31 no es suficiente para lograr buenos resultados. Además de esto, se observó que se presentaba otra limitación con el algoritmo de detección YOLO, ya que, los objetos que reconocería este algoritmo debían ser todos los elementos GUI que encontrara en el video y no los elementos que el usuario directamente utiliza.

Figura 31 Entrenamiento del algoritmo YOLO.



Fuente: Autoras, (2020).

A partir de los resultados obtenidos con el entrenamiento del algoritmo de detección de objetos YOLO, se llegó a la conclusión que los resultados obtenidos no son los esperados. Por este motivo, optamos por usar la técnica de visión de reconocimiento de color para la detección de objetos. Esta técnica permitió obtener las coordenadas GUI utilizadas por el usuario.

La técnica de detección de color permite detectar el color y la forma de un objeto dentro de una imagen. Esta técnica, realiza la captura de una imagen la cual se utiliza para buscar la marca de los cuatro cuadros que esta deja donde se encuentra el objeto. Asimismo, se aplican los filtros a la imagen. En primer lugar, se convierte la imagen en escala de grises. En segundo lugar, se reduce la resolución a la mitad y se vuelve a

ampliar la resolución a la original para eliminar el ruido. Por último, se suaviza la imagen y se elimina el mayor ruido posible (Ernesto Arévalo-Vázquez et al., 2015).

A partir de lo anterior, se procede a la segmentación de la imagen, es decir, la imagen se filtra por medio de un algoritmo llamado *Canny*, el cual, busca reducir los datos de la imagen de interés con la información que sea de nuestro interés. Después, se realiza la extracción de características por medio de los contornos, del cual se obtienen coordenadas de cada uno de los cuadros encontrados. Para ello, se dibuja un recuadro verde sobre la imagen original y se guardan las coordenadas obtenidas. De esta forma, la coordenada guardada se usa para identificar el objeto en fases posteriores (Ernesto Arévalo-Vázquez et al., 2015).

7.3.6 Despliegue. El componente de visión se verifica y confirma si cumple con los requisitos correspondientes del prototipo de software antes de la implementación. A partir de ello, el prototipo realiza las siguientes funciones:

- Realizar el etiquetado de los elementos de la GUI, tomando un video como dato de entrada.
- Tomar cada fotograma del video y busca los controles que están encerrados en color amarillo para obtener las coordenadas donde se encuentra ubicado el objeto.
- Realizar una segmentación de la imagen, para eliminar posibles duplicados.
- Realizar la extracción de las características, en este caso de las coordenadas de la región seleccionada.
- Realizar los recortes de cada región con las coordenadas obtenidas anteriormente.
- Realizar un barrido de texto a cada uno de los recortes.
- Generar un archivo en formato (*.csv*) de cada recorte y definir si es un controlador de texto o un botón.

7.4 PROTOTIPO SOFTWARE FUNCIONAL

Esta sección se desarrolla de la siguiente forma. Primero, en la sección 7.4.1 se realiza la definición de las tareas para el correcto funcionamiento del prototipo. Segundo, en la sección 7.4.2 se presenta el desarrollo del prototipo de software, dividido en el desarrollo del componente de visión computacional y el desarrollo del *chatbot*. Tercero, en la sección 7.4.3 se muestran las pruebas realizadas al componente junto con el software contable *Quickbook.* Por último, en la sección 7.4.4 se presentan los resultados del módulo de visión computacional conectado al *chatbot.*

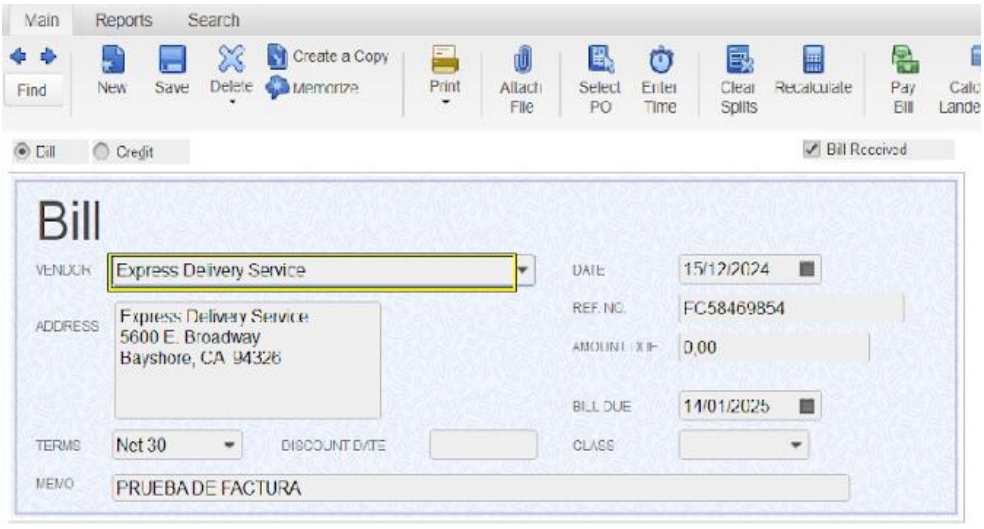
7.4.1 Definición de tareas. Para el correcto funcionamiento del prototipo de software se presentan las siguientes tareas que abarca el proceso del componente de visión computacional y la RPA.

- Leer el archivo plano generado por el componente de visión con el etiquetado de cada elemento de GUI encontrado en el video.
- Obtener cada una de las imágenes de las coordenadas encontradas, y eliminar coordenadas replicadas.
- Realizar barrido de texto de cada una de las imágenes para clasificar cada uno de los elementos GUI guardando la información en un archivo .csv
- Crear estructura de conversación y reglas del *chatbo*t
- Conectar el componente de visión con el *chatbot,* obtener datos del archivo .csv generar el código de la RPA
- Obtener el archivo en formato *(.robot)* generado y ejecutar RPA
- Implementar la interfaz gráfica del *chatbot* y pruebas del funcionamiento del prototipo.

7.4.2 Desarrollo del prototipo de software. Está dada en dos partes. Primero, la sección 7.4.2.1 que presenta el proceso del componente de visión computacional para la detección de objetos por color. Segundo, la sección 7.4.2.2 que presenta el desarrollo del *chatbot* con una estructura de conversación básica para la interacción de usuario y máquina.

7.4.2.1 Desarrollo del componente de visión computacional. El componente de visión recibe como dato de entrada, un video para obtener cuales eran los elementos que se estaban utilizando de la GUI. Asimismo, se utilizó la herramienta *inspect* para encerrar en un cuadro de color amarillo los elementos como se muestra en la Figura 32.

Figura 32 Uso de inspect para encerrar elementos en un cuadro de color amarillo.



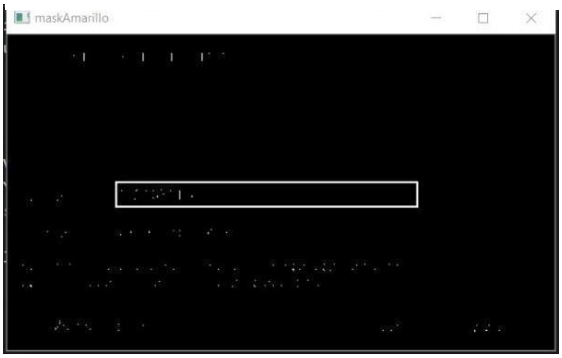
Fuente: Autoras, (2020).

Usando la técnica de visión de reconocimiento de color. Primero, se obtiene un *frame* que será utilizado para realizar los recortes de las diferentes regiones. Después, se obtienen los fotogramas del video transformándolo de BGR a HSV *(Hue, Saturation, Value* / Matiz, Saturación, Brillo). Luego, se determinan los rangos en donde se encuentra el color a detectar, en este caso, el color amarillo como se muestra en la Figura 33. Con base a estos rangos se obtienen las imágenes, que son binarias las cuales se muestran de color blanco, la ubicación que corresponde al color amarillo y el negro en donde no se encuentran estos rangos de color.

Figura 33 Conexión a escritorio remoto.

Fuente: Autoras, (2020).

Figura 34 Imagen binaria obtenida después de aplicar detección de color amarillo.

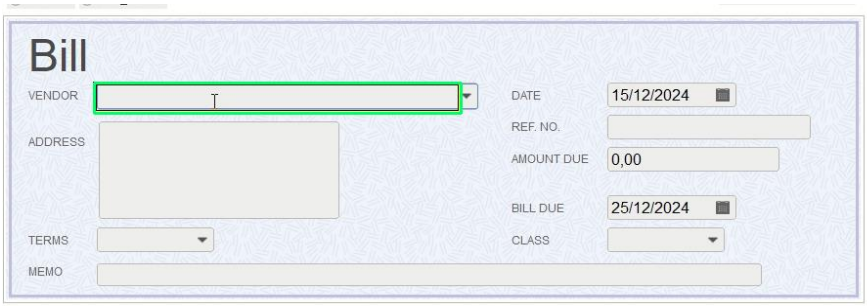


Fuente: Autoras, (2020).

Después de detectar en qué lugar se encuentra el color amarillo dentro la imagen. Se encierran las regiones en donde se encuentra el color amarillo y se descartan las pequeñas áreas que no son importantes. Luego se detectan las coordenadas del color detectado. Al final de este proceso, se dibuja un rectángulo de color verde

correspondiente a la región en donde se encuentra el color como se muestra en la Figura 35.

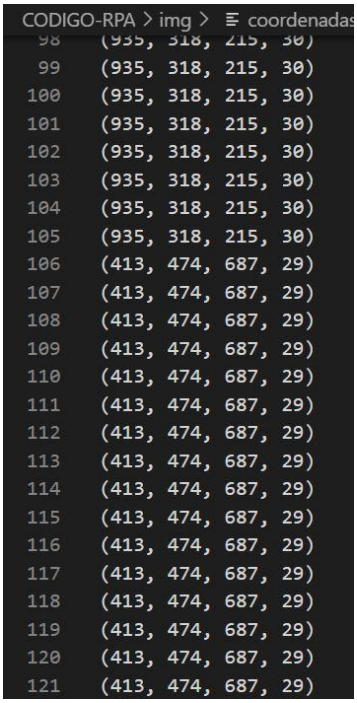
Figura 35 Detección de color verde.



Fuente: Autoras, (2020).

Cada coordenada detectada se guarda en un archivo de etiquetado en formato .txt como se muestra en la Figura 36.

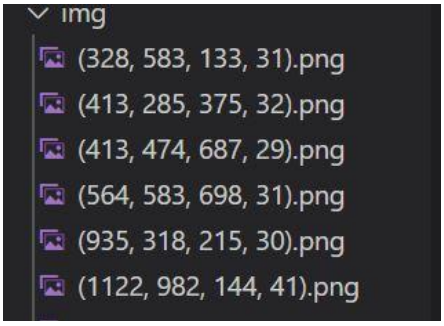
Figura 36 Coordenadas detectadas.



Fuente: Autoras, (2020).

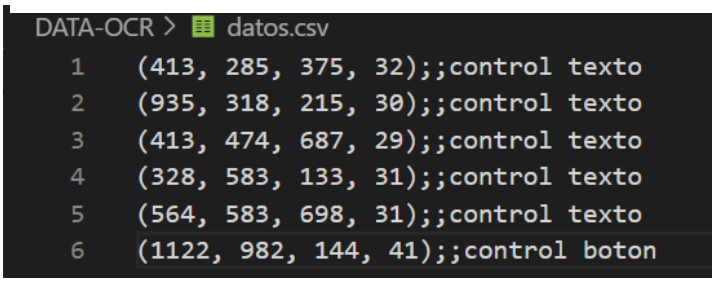
Después de obtener cada una de las coordenadas de los elementos en cada uno de los fotogramas del video, se filtran los datos y se eliminan los datos duplicados. Luego se pasa a realizar el recorte de cada región dentro de la imagen que se había obtenido del video al inicio. Después de obtener el recorte de cada una de las imágenes de las coordenadas como se observa en la Figura 37, se realiza un barrido de texto utilizando la librería de *pytesseract* de *python* y se obtiene el texto que contenga, luego, se va validando si es una caja de texto, botón o botón desplegable, estos datos se guardan en un archivo en formato .csv con el nombre de imagen, texto obtenido y clase de elemento que se puede ver en la Figura 38.

Figura 37 Recortes de las imágenes.



Fuente: Autoras, (2020).

Figura 38 Datos obtenidos al realizar el barrido con la librería pytesseracy de python en formato .csv.



Fuente: Autoras, (2020).

7.4.2.2 Desarrollo del Chatbot. En esta sección se muestra el desarrollo de un chatbot con una estructura de conversación muy básica utilizando la librería de lenguaje natural NLTK, para realizar la conexión del componente de visión, la generación del código de la RPA en el lenguaje de *robot framework* y la ejecución de este.

En la Figura 39, se muestra la estructura de conversación del *chatbo*t, en donde se encuentra las posibles entradas de texto del usuario con las respuestas del *chatbot* de las cuales algunas de las respuestas irán acompañadas con los métodos para ejecutar cada función que debe realizar el *bot*, que son ejecutar el código de visión, extraer y validar los elementos de las imágenes, generar código y ejecutar RPA.

Figura 39 Estructura de conversación del chatbot.

```
pairs = [
    ["hola", ["Saludos! Mi nombre es  Chatbot-PGII, Quien eres?."], None],
    ["soy(.*)", ["hola %1, ¿que desea hacer?. \n 1-Crear RPA  2-Ejecutar RPA"], None],
    ["1|Crear RPA|crear rpa", ["Desea generar el codigo..."], metodos_RPA.run],
    ["2|Ejecutar RPA|ejecutar rpa", ["Ejecutando RPA..."], metodos_RPA.Run_RPA],
    ["Seleccionar", ["Ejecutando RPA..."], metodos_RPA.Run_RPA],
    ["si", ["leyendo datos..."], metodos_RPA.datos_csvw],
    ["RPA generada", ["¿Desea ejecutar la RPA?"], None],
    ["control texto", ["creando metodo texto..."], None],
    ["control boton desplegable", ["creando metodo..."], None],
    ["control boton", ["creando boton..."], None],
    ["rpa generada", ["se genero codigo de rpa"], None],
    ["chao|no|NO", ["chao"], metodos_RPA.exit_chat],
]
```

Fuente: Autoras, (2020).

Figura 40 Imagen del script generado de la RPA en el lenguaje de robot framework desde el chatbot.

```
CODIGO-RPA > quickbooksPrueba.robot > Open Windows
*** Settings ***
Documentation     RPA Demo
Test Setup        Add Needed Image Path
Test Teardown     Stop Remote Server
Library           SikuliLibrary

*** Variables ***
${IMAGE_DIR}      ${CURDIR}\\img

*** Test Cases ***
Windows Software Contable
    Open Windows

*** Keywords ***
Add Needed Image Path
    Add Image Path    ${IMAGE_DIR}

Open Windows
    Double Click    2.png
    Sleep   3s
    Input Text    (413, 285, 375, 32).png  empresa abc
    Sleep   2s
    Input Text    (935, 318, 215, 30).png  FC58469854
    Sleep   2s
    Input Text    (413, 474, 687, 29).png  PRUEBA DE FACTURA
    Sleep   2s
    Input Text    (328, 583, 133, 31).png  54200
    Sleep   2s
    Input Text    (564, 583, 698, 31).png  PRUEBA DE FACTURA
    Sleep   2s
```

Fuente: Autoras, (2020).

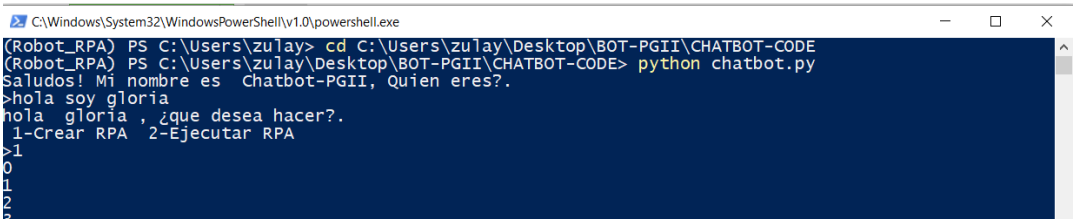
A continuación, se muestra el diseño de la interfaz gráfica del *chatbot*, desarrollado con la librería *Tkinter* de python.

Figura 41 Diseño de la interfaz gráfica del chatbot.

Fuente: Autoras, (2020).

7.4.3 Pruebas del prototipo de software. En esta sección se muestran las pruebas realizadas al prototipo, las pruebas se realizaron en el software contable de *Quickbook* de licencia de prueba de 30 días. Asimismo, se seleccionó este software ya que era fácil de entender y de utilizar. Además, este software trae una base de datos de ejemplo de una empresa. Luego, se grabaron diferentes videos ingresando datos en el software. Estos mismos, fueron los datos de entrada para el módulo de visión. A continuación, se muestra en la Figura 42 el *chatbot* en su fase inicial y en su fase final.

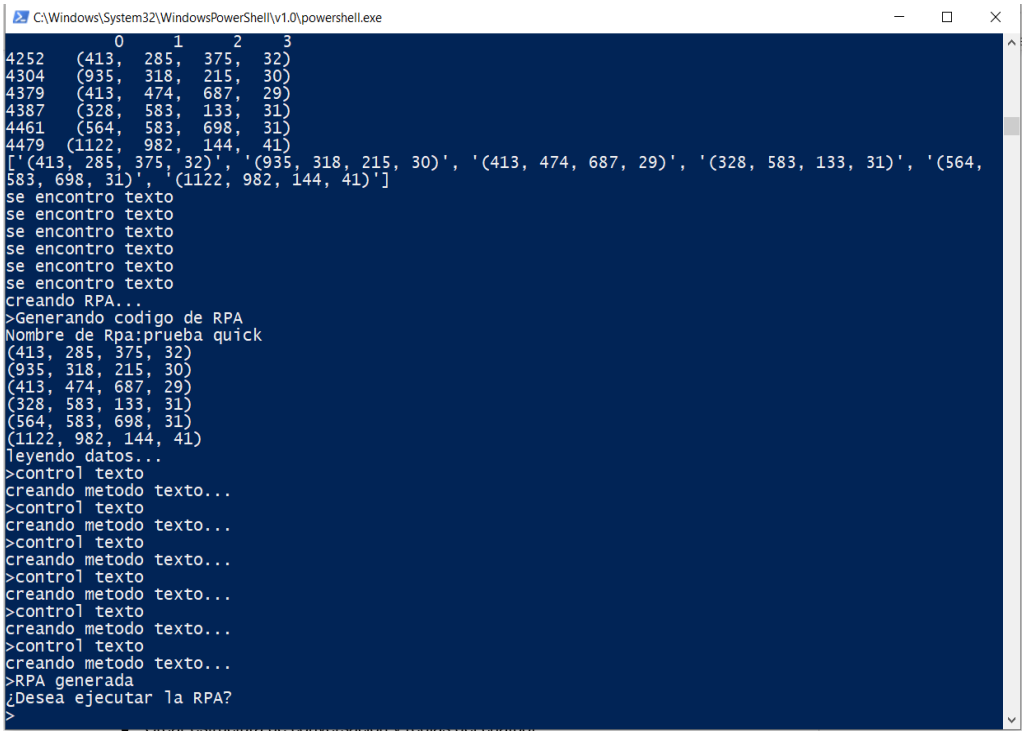
Figura 42 Inicio de conversación con el chatbot.



Fuente: Autoras, (2020).

En la Figura 43 se muestra la eliminación de las coordenadas duplicadas y la validación de los elementos GUI encontrados en el video con reconocimientos de texto, recorte de imagen de cada región y la generación del código de RPA.

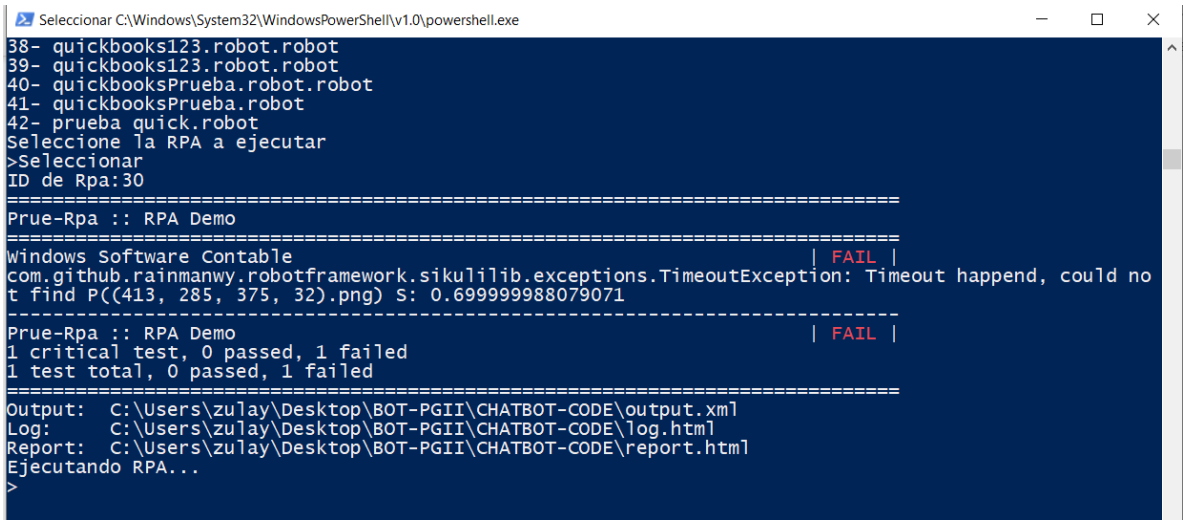
Figura 43 Eliminación de coordenadas duplicadas y validación de los elementos GUI.



Fuente: Autoras, (2020).

En la Figura 44, se muestra el lanzamiento de la RPA creada anteriormente

Figura 44 Lanzamiento de la RPA.



Fuente: Autoras, (2020).

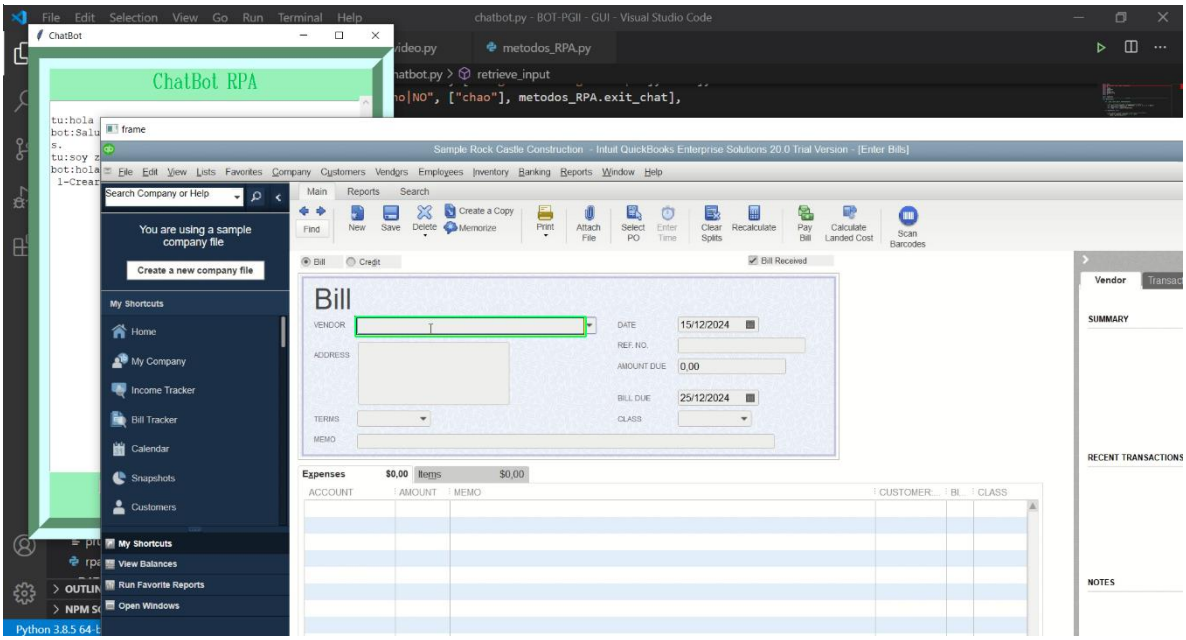
Fase final. A continuación, se muestra el *chatbot* con la interfaz gráfica de usuario.

Figura 45 Chatbot con Interfaz gráfica de usuario final.

Fuente: Autoras, (2020).

En la Figura 46 muestra el *chatbot* abriendo el archivo de video, realizando el reconocimiento de color amarillo y el encierro de la ubicación de este mismo para obtener las coordenadas de cada región.

Figura 46 Reconocimiento de color amarillo y encierro de la ubicación del elemento.

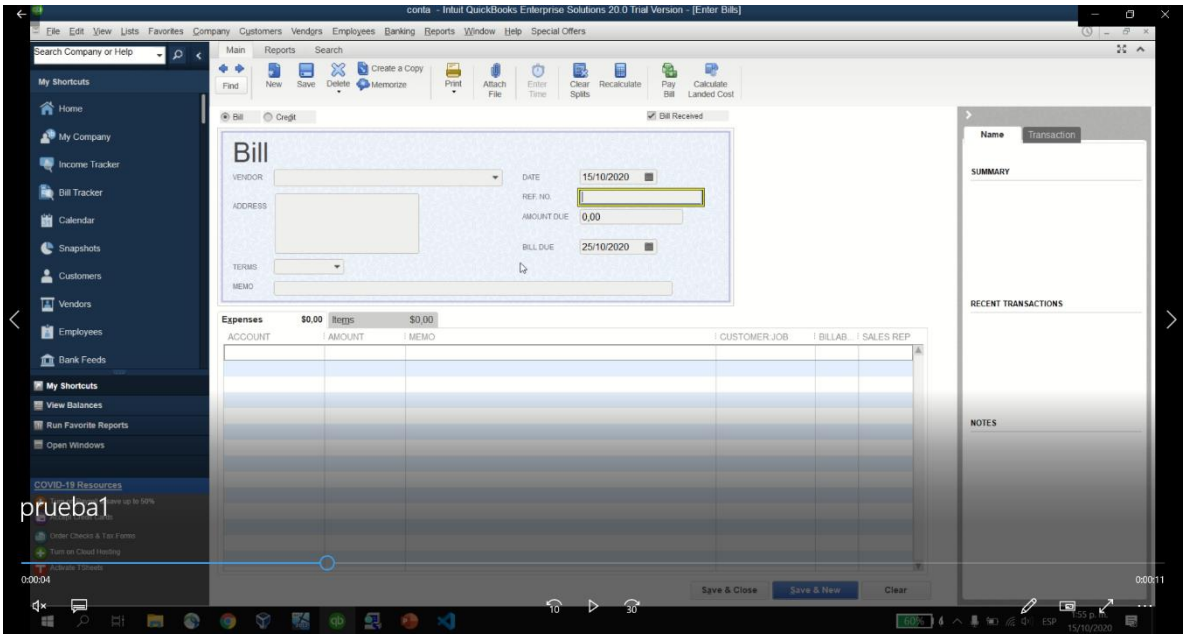


Fuente: Autoras, (2020).

7.4.4 Resultados de las pruebas. Para las pruebas del módulo de visión computacional conectado al *chatbot* se grabaron 4 videos sobre el software contable de *Quickbook*.

Video1. El video se grabó seleccionando los elementos GUI (Interfaz Gráfica de Usuario) a utilizar del software contable sin ingresar datos, como se observa en la Figura 47.

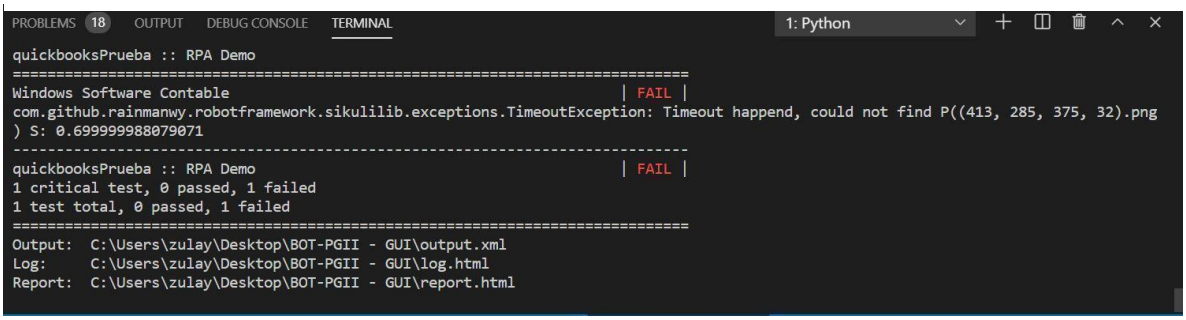
Figura 47 Seleccionando elementos GUI.



Fuente: Autoras, (2020).

En la Figura 48, la prueba presentó errores en la RPA cuando accedía a los controles para ingresar los datos. Ya que, algunas de las coordenadas eran de un mismo control, entonces, cuando la RPA accedía al control e ingresaba el dato, la RPA buscaba la siguiente imagen, pero con coordenada diferente, es decir, se encontraba que era la misma imagen del anterior por lo cual no lo encontraba ya que se había ingresado el dato. Este error se logró solucionar, al obtener cada coordenada y validar si estas eran iguales en sus puntos x,y y w, ya que, estas eran las que se mantenían en un valor fijo, mientras que su coordenada h era la cual cambiaba de valor.

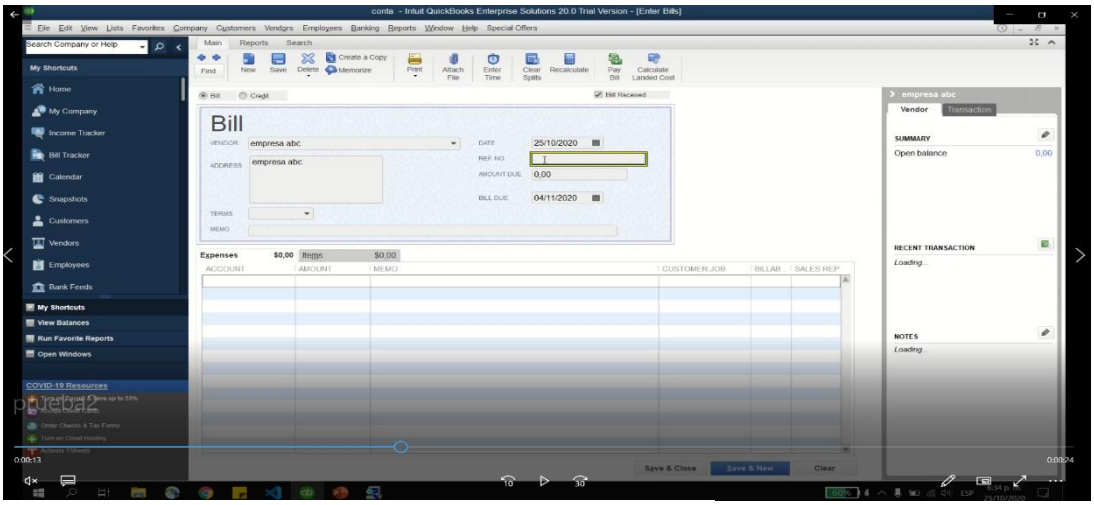
Figura 48 Errores de la RPA al momento de acceder a los controles para ingresar los datos.



Fuente: Autoras, (2020).

Video 2. En este video se grabó ingresando datos en el software, tratando de ingresarlos de una forma rápida como se observa en la Figura 49.

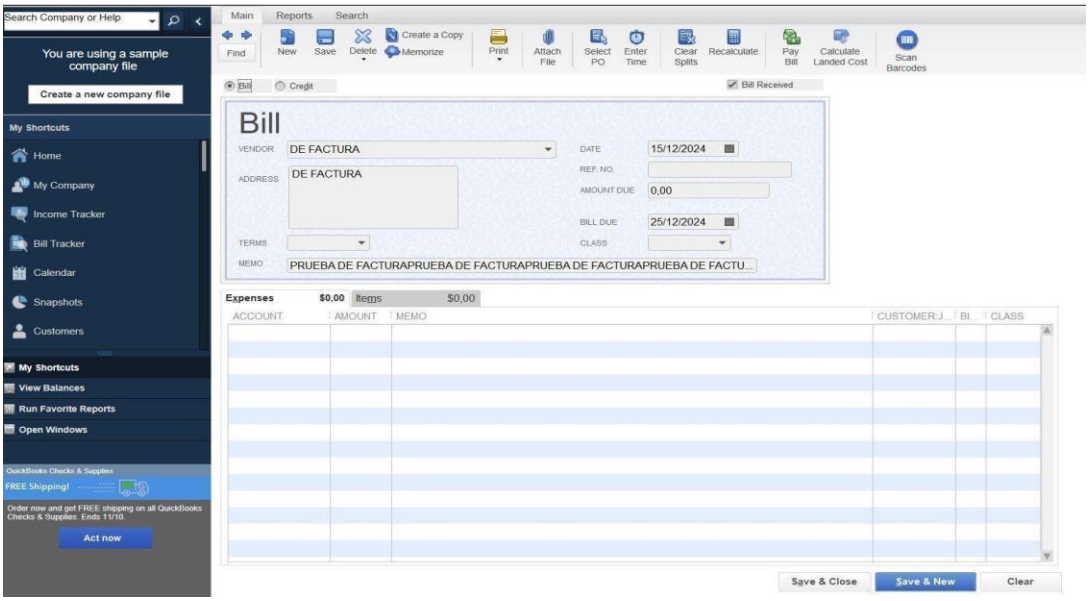
Figura 49 Ingresando datos al software.



Fuente: Autoras, (2020).

En la Figura 50, la prueba accedida a los controles de texto, pero no los seleccionados. En esta prueba la RPA ingresaba los datos en un mismo control y el dato que no era.

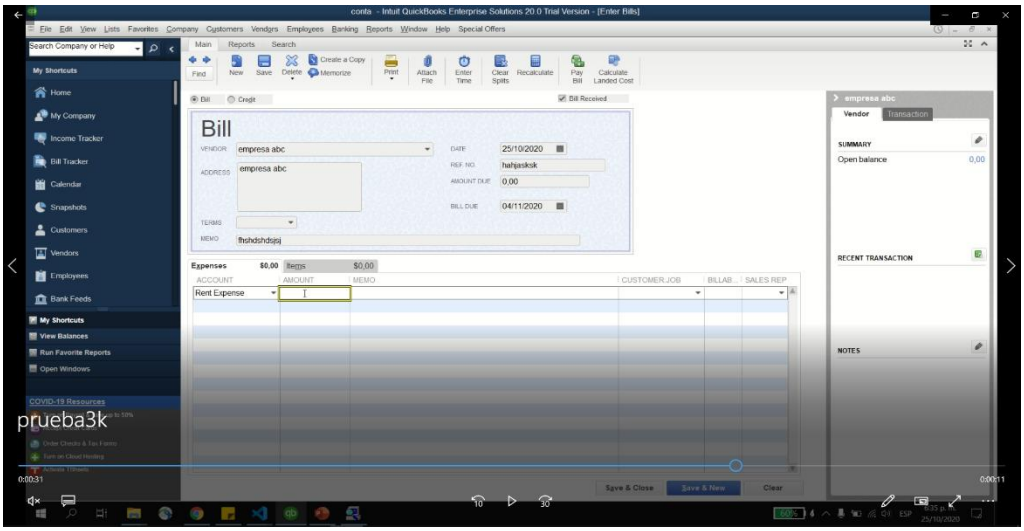
Figura 50 Ingreso de datos en el mismo control y datos erróneos.



Fuente: Autoras, (2020).

Video 3. En este video se ingresaron los datos al software de manera que en cada control que se ingresaba el dato, el usuario se demorara un tiempo máximo 8 a 10 segundos como se puede ver en la Figura 51.

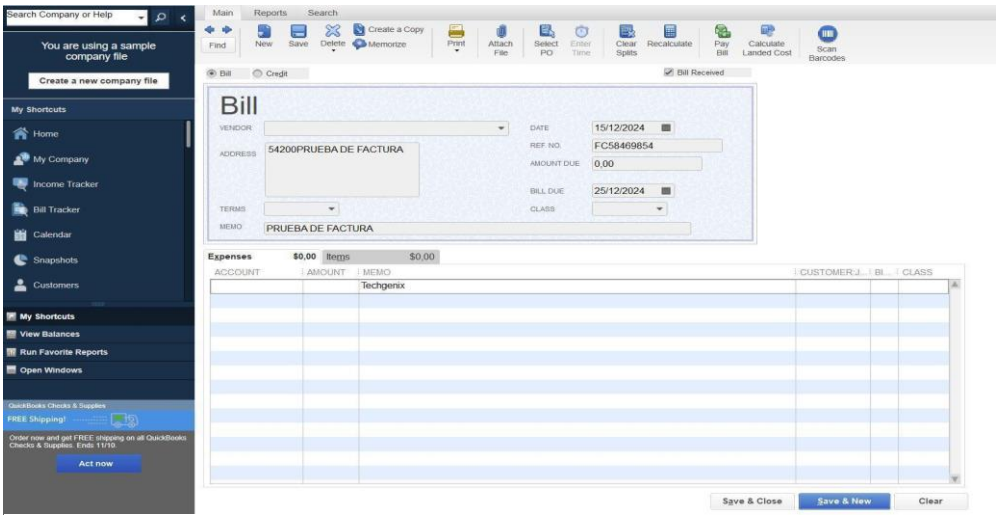
Figura 51 Ingresando datos en cada control.



Fuente: Autoras, (2020).

En esta prueba se ingresaron algunos de los datos en los controles seleccionado, pero algunos datos los ingresaba en un mismo control o en otro control, presente en la Figura 52

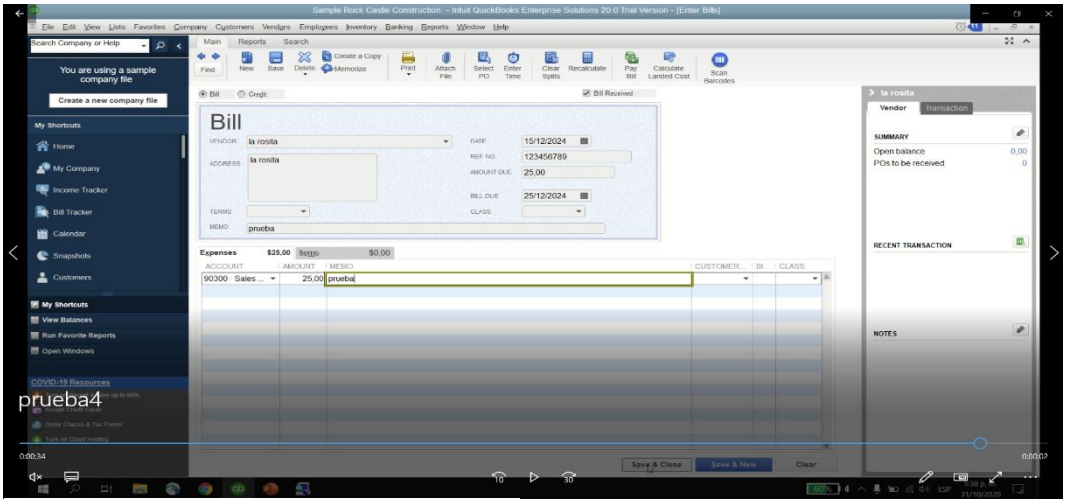
Figura 52 Prueba, falla de ingreso de datos en controles seleccionados.



Fuente: Autoras, (2020).

Video 4. En este video se ingresaron todos los datos en el software contable. Realizando una demora en cada control después de ingresar el dato mínimo de 3 a 4 segundos como se puede ver en la Figura 53.

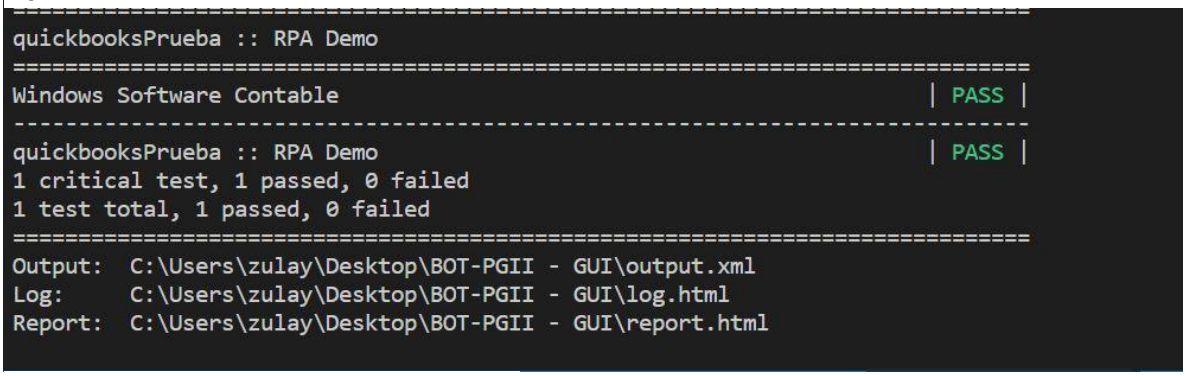
Figura 53 Se ingresan todos los datos al software contable.



Fuente: Autoras, (2020).

En la Figura 54, los resultados de la prueba fueron mejores. la RPA seleccionó los controles utilizados, aunque solo en un control no tomó el dato de la coordenada y realizó el ingreso de cada uno de los datos en los controles respectivos. Para solucionar que tomara los datos de todos los controles encerrado en amarillo. Este error se logró solucionar, tomando el área encontrada y comparándolo con 1700 pixeles, para así lograr que solo pasaran los contornos que superen dicho valor, por lo tanto, los valores más pequeños se descartan.

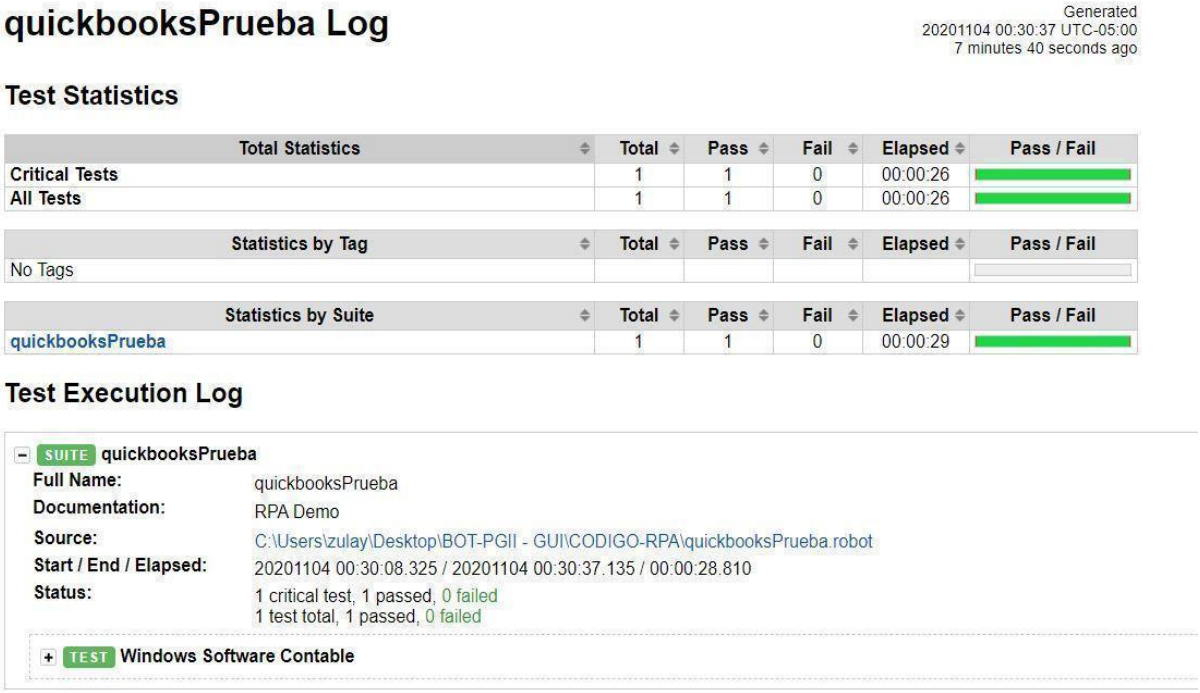
Figura 54 La RPA seleccionó los controles utilizados.



Fuente: Autoras, (2020).

A continuación, se muestra la prueba generada por *robot framework*, en la Figura 55 se muestra las estadísticas de las pruebas y si realizo el proceso. En la Figura 56, se muestra cuáles fueron los controles que accedió el dato que ingresó y el tiempo en que se demoró en acceder a cada uno.

Figura 55 Estadísticas de las pruebas y estado del proceso.



quickbooksPrueba Log

Generated
20201104 00:30:37 UTC-05:00
7 minutes 40 seconds ago

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	1	1	0	00:00:26	
All Tests	1	1	0	00:00:26	

Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
No Tags					

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
quickbooksPrueba	1	1	0	00:00:29	

Test Execution Log

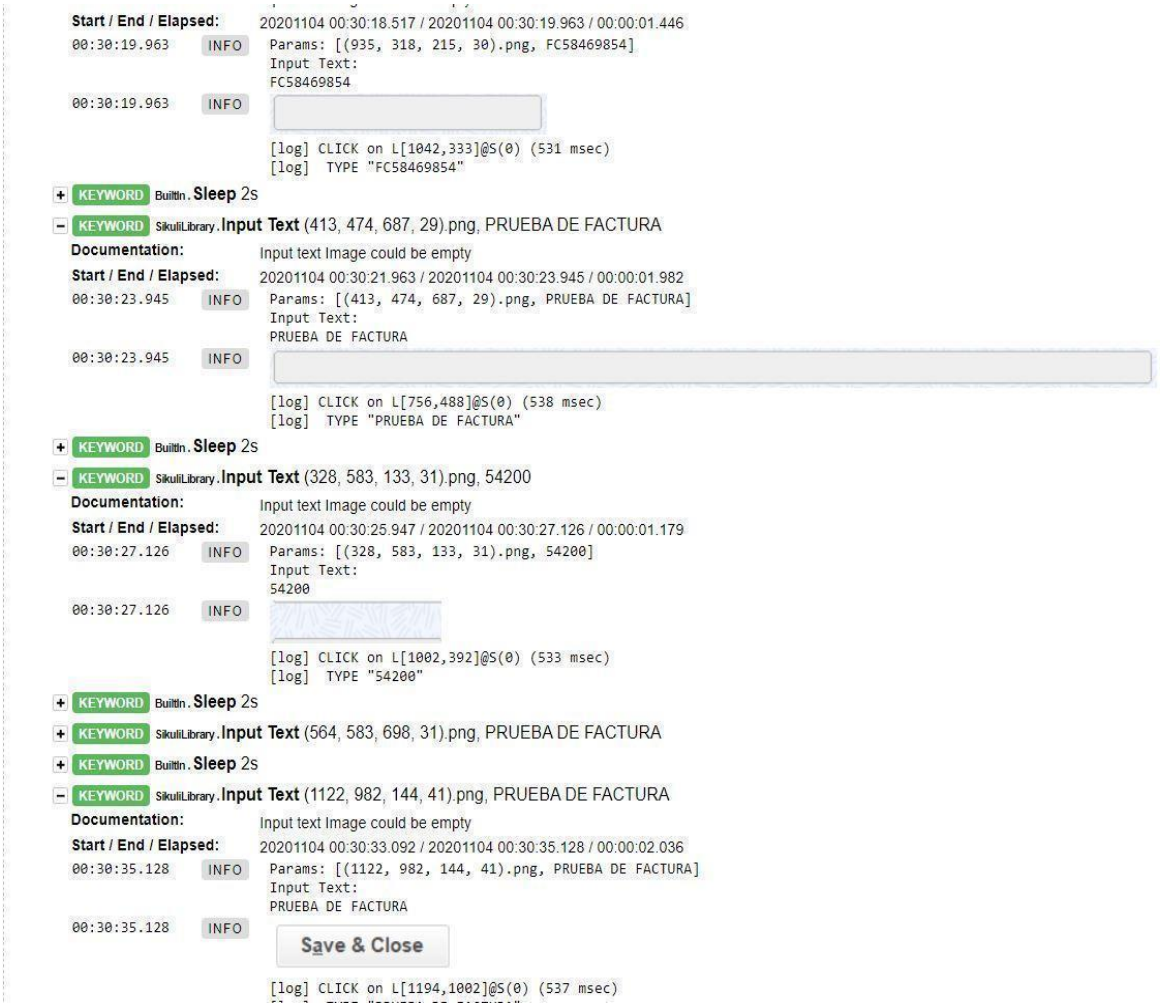
SUITE quickbooksPrueba

Full Name:	quickbooksPrueba
Documentation:	RPA Demo
Source:	C:\Users\zulay\Desktop\BOT-PGII - GUI\CODIGO-RPA\quickbooksPrueba.robot
Start / End / Elapsed:	20201104 00:30:08.325 / 20201104 00:30:37.135 / 00:00:28.810
Status:	1 critical test, 1 passed, 0 failed 1 test total, 1 passed, 0 failed

TEST Windows Software Contable

Fuente: Autoras, (2020).

Figura 56 Controles a los que accedió los datos que ingreso.



Fuente: Autoras, (2020).

7.5 INFORME COMPARATIVO DEL RENDIMIENTO DE PROCESOS CON SOFTWARE Y SIN SOFTWARE.

Esta sección está dada en tres partes. La 7.5.1, define el escenario de pruebas que está dividida en escenario de creación de RPA y escenario de registro de facturas. La 7.5.2, evalúa el rendimiento del escenario de pruebas y la 7.5.3 presenta la comparativa de procesos con software y sin software.

7.5.1 Definición de escenario de pruebas. Se presentan dos escenarios, por un lado, el desarrollo de la RPA y por el otro el registro de facturas en el software *Quickboock* de manera manual y automatizada.

7.5.1.1 Escenario creación de RPA. En esta sección se describe el proceso de desarrollo de una RPA manualmente y una RPA generada automáticamente.

Proceso de creación de RPA de manera manual. Para el desarrollo de una RPA, primero se realiza un análisis y entendimiento del proceso a automatizar para capturar cada actividad que debe realizar la RPA para realizar el proceso, se verifica que tipo de software se va a automatizar de escritorio o web y se verifica si es un proceso simple o complejo. Después de realizar el paso anterior se inicia la codificación de la RPA con base a las actividades captadas del proceso. Después se realizan pruebas para validar su funcionamiento y que cumpla en cada una de las actividades del proceso. Este proceso según la complejidad del proceso a automatizar puede ser en semanas o días si es un proceso simple.

Proceso de generación de RPA automáticamente. El proceso de generación de RPA automáticamente se realiza de la siguiente forma. El usuario en este caso que es el contador graba en un video todo el proceso que realiza en el registro de una factura, en el video grabado los diferentes elementos GUI que vaya el usuario utilizando este se muestra encerrado en amarillo y lo cual queda captado en el video. Este video se envía como dato al módulo de visión y se extrae las coordenadas de los elementos GUI utilizadas por el usuario y se clasifica cada control de texto, botón, botón desplegables etc. Ya obteniendo esa información se toma y se genera automáticamente el código con base a la información obtenida del video. Todo este proceso el usuario lo realiza interactuando con un *chatbot* de lo cual lo va guiando en el proceso en la generación de la RPA y ejecución de la misma.

7.5.1.2 Escenario proceso de registro de las facturas. Describe el proceso de registro de las facturas de manera manual y automatizada con el software contable *Quickbook.*

Proceso de registro de facturas de manera manual. Al momento de realizar el registro de las facturas, se debe abrir el software contable *Quickbook* e ingresar a la sección de facturas de venta. Luego, se empieza a ingresar los datos de la factura como lo es, el

ingreso del nombre del cliente, el número de radicado, el ítem que pertenece al producto, la descripción del producto y el valor de la factura. Posteriormente, se le da guardar y con ello finaliza el proceso de subir la información de la factura al software contable.

Proceso de registro de facturas de manera automatizada. Al momento de registrar la facturas, el usuario se conecta a un *chatbot*. el usuario le comunica al *chatbot* que desea ejecutar la RPA que anteriormente creó para realizar determinado proceso en este caso registrar una factura de venta**.** la RPA automáticamente se inicia y abre el software contable de *Quickbook* la sección de factura de venta y accede a los controles determinados e ingresa cada dato de la factura como lo son nombre de la empresa, número de facturas del ítem del producto a registrar, descripción del producto, valor de la factura y finaliza guardando la factura.

7.5.2 Evaluación del rendimiento y comparativa de procesos con software y sin software. En esta sección se presentan la evaluación del rendimiento y la comparación de los procesos con software y sin software en los dos escenarios propuestos en la sección 7.5.1. dado de la siguiente forma:

7.5.2.1 Escenario de creación de la RPA. La evaluación del rendimiento de la creación de la RPA de manera manual y automatizada está dada de la siguiente forma:

Creación de RPA de manera manual. Para la creación de una RPA de manera manual, se estima un tiempo de demora entre 1 día como mínimo a 2 días como máximo en un proceso simple, lo que equivale entre 86.400 seg a 172.800 seg.

Generación de RPA automáticamente. El tiempo de la RPA en generarse automáticamente está dado en un rango de 30 a 35 seg.

Comparación del escenario del desarrollo de la RPA de manera manual y la generación de la RPA automáticamente. Los tiempos tomados en cada uno de los escenarios se puede observar en la Tabla 8.

Tabla 8 Intervalo de tiempo de desarrollo de la RPA manual y automatizada.

	Desarrollo de RPA manualmente (seg)	**Generación de RPA automáticamente (seg)**	**Diferencia (seg)**
Tiempo	[86400-172800]	[30-35]	[86370-172765]

Fuente: Autoras, (2020).

En el escenario de creación de RPA se refleja una diferencia dada entre 86.370 seg a 172.765 seg entre la creación de una RPA de manera manual a una generada automáticamente. Por ello, crear la RPA de manera automática le permite al usuario ahorrar tiempo y dinero en el desarrollo de esta. Además, de que no necesita tener un conocimiento previo en programación

7.5.2.2 Escenario de registro de facturas de manera manual. Se contó con 5 personas para llenar 10 facturas de venta al software Quickboock cada uno cuenta con un solo producto. La factura consta de 5 ítems importantes: cliente, número de radicado, número del ítem, descripción del producto y valor del producto. A partir del proceso de registro, obtuvimos los siguientes resultados:

Tabla 9 Llenado de facturas manualmente.

Factura	Número	Tiempo (seg)	Errores
Aware industrial	05	116	Se eliminó la información del sistema al escoger el *customer job*.
Aware industrial	06	90	No hubo problema.
Colchones Spring	08	45	No hubo problema.
Colchones Spring	07	70	No hubo problema.
Namaste Textil	02	184	Se eliminó la información al darle *enter.*
Namaste Textil	01	80	No hubo problema.
Shindaiwa	09	132	No tomaba el ítem.
Shindaiwa	10	75	No hubo problema.
Safari Textil	03	54	No hubo problema.
Safari Textil	04	55	No hubo problema.

Fuente: Autoras, (2020).

Escenario de llenado de facturas de manera automatizada. A continuación, en la Tabla 10 se presenta el tiempo que demora cada factura en subir su información de manera automática al sistema contable *Quickbook*.

Tabla 10 Llenado de facturas de manera automática.

Factura	Tiempo (seg)
1	27
2	27
3	34
4	26
5	33
6	29
7	28
8	29
9	32
10	28

Fuente: Autoras, (2020).

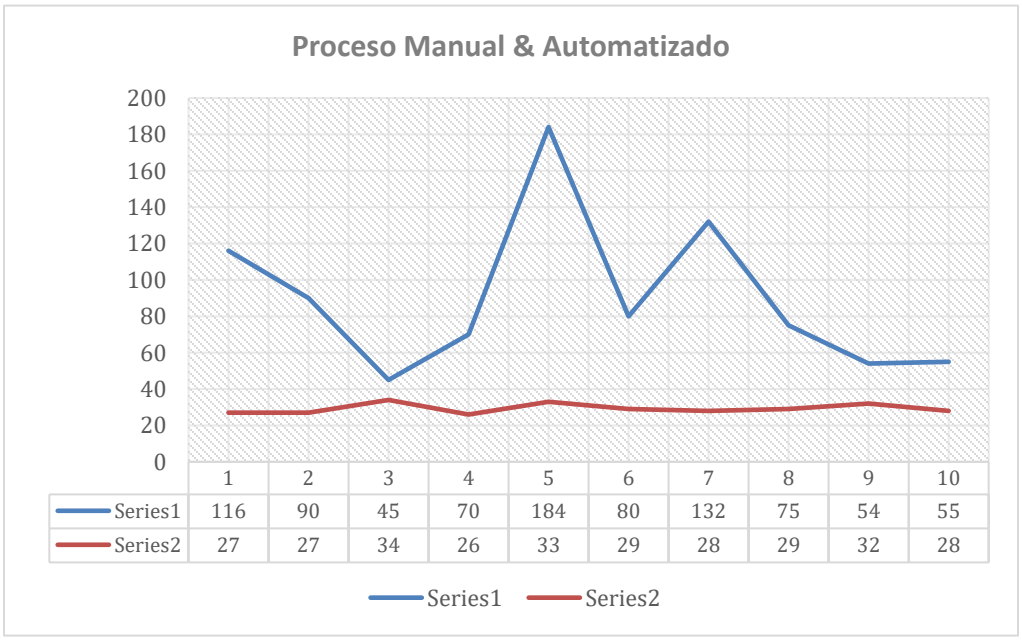
Comparación del escenario de registro manual y el automatizado. En la Tabla 11, se muestran los tiempos de registro de las facturas de manera manual y automatizada.

Tabla 11 Tiempo de llenado de factura de manera manual y automatizada.

Factura	Manual (seg)	Automático (seg)	Diferencia (seg)
1	116	27	89
2	90	27	63
3	45	34	11
4	70	26	44
5	184	33	151
6	80	29	51
7	132	28	104
8	75	29	46
9	54	32	22
10	55	28	27
Promedio	**90,1**	**29,3**	**60,8**

Fuente: Autoras, (2020).

Figura 57 Gráfica proceso de llenado de facturas manual y automatizado.



Fuente: Autoras, (2020).

En el escenario de proceso de registro de las 10 facturas de manera manual, el tiempo usado para subir la información al software tiene un promedio de 90.1 seg, mientras que el proceso de llenado de manera automática tiene un promedio de 29.3 seg como se puede observar en la Tabla 10. De lo anterior, se puede decir que la diferencia entre el registro de la información entre los dos escenarios es de 60.8 seg. Por tanto, el registro de las facturas de manera automática ahorra tiempo y les permite a los usuarios realizar otras tareas mientras el software realiza la tarea.

8. CONCLUSIONES Y TRABAJO A FUTURO

En este proyecto se presenta el desarrollo de un Prototipo de Software para la creación de RPA orientadas a Software Contables para Instituciones del Sector Público. Con base a la metodología planteada para el desarrollo del prototipo de las cuales están estructurada de la siguiente forma: fase 1, caracterización del proceso contable, recolección y análisis de información sobre las características generales del proceso contable y levantamiento de requerimientos para el prototipo a desarrollar. Fase 2, selección de tecnologías y métodos, revisión de literatura de las diferentes tecnologías en modelo de visión y desarrollo de RPA y selección de las tecnologías a utilizar. Fase 3, construcción del componente de visión computacional. En esta fase, se construyó el componente de visión computacional que se integró al prototipo para identificar los componentes de interfaz gráfica del software contable. Fase 4, desarrollo del prototipo de software. implementando la metodología de gestión incremental, lo cual permitió una adecuada administración de las tareas y adaptación de cambios o modificaciones en cada entrega hasta ajustarse a lo requerido. Fase 5, Evaluación de desempeño del prototipo software. Evaluando el rendimiento del software y comparando el proceso con el software y sin el software.

En la primera fase se realizó una caracterización del proceso contable. Para ello, se aplicó una encuesta a 10 contadores de diferentes empresas, con el fin de identificar las características principales de los softwares contables que usan en las empresas. Esta encuesta evidenció que: la mayor parte del software contable usado se ejecuta en sistemas de escritorio compatible con el sistema operativo Windows. Además, se evidencio que los errores más frecuentes en el proceso contable son los errores de transcripción y de omisión. También se evidencio que para el proceso de registro de facturas se tarda de 1 a 2 minutos.

De la selección de las tecnologías y métodos, se construyó un ecosistema basado *python* ya que es un lenguaje de programación multiparadigma, lo que no fuerza a adoptar un estilo particular a la hora de programar, además de su fácil uso y versatilidad.

La construcción del componente de visión computacional se realizó por medio de la metodología CRISP-DM, ya que proporciona una descripción normalizada del ciclo de vida de un proyecto estándar de análisis de datos. Además, se optó por utilizar la técnica de detección de objetos por color ya que permitió tomar los elementos que el usuario intervenía y no se necesitaba el uso de un *dataset.* Ya que, con el modelo de YOLO se presentó dificultad al momento de procesar las imágenes del *dataset* y el entrenamiento de estas. Además, YOLO no permitió obtener los datos de los elementos que intervenía el usuario.

Para el desarrollo del *chatbot* se usó la librería de lenguaje natural NLTK ya que facilitó la interacción de usuario y máquina. Además, el *chatbot* le da la posibilidad al usuario de ejecutar el código de visión computacional y ejecutar la RPA a través de la interfaz gráfica de usuario.

Para validar el prototipo de software se diseñó una prueba en dos escenarios, en un escenario se realizó el registro de forma manual y el otro utilizando el software desde la generación de la RPA hasta el registro de la factura con esta misma. Realizando la comparativa de los procesos, se pudo evidenciar que los tiempos de la creación de la RPA de manera manual y automatizada difieren en un rango de 86.400 seg a 172.765 seg. Por otro lado, el registro de la información de manera manual y automatizada presenta una diferencia de 60.8 seg. Según las pruebas realizadas, refleja que la manera más eficaz y que conlleva menos inyección de recursos y tiempo en los dos casos es la automatizada.

A partir de lo anterior, se puede evidenciar los tiempos de los procesos de carga de datos de manera manual y automatizada y se puede evidenciar que hay una optimización de 60.8 seg. Además, el uso de visión computacional para realizar la creación de RPA'S comprobó que es más eficiente y económicamente factible que realizar este proceso de manera manual.

Como trabajo a futuro, se plantea que el software pueda manejar diferentes ventanas del software contable. Además, que el prototipo de software al momento de realizar el registro de la información pueda subir más de un ítem por factura y que la cantidad de facturas que se tenga en contabilidad, se puedan subir de manera automática para agilizar el proceso.

En el *chatbot*, se espera escalar a unas respuestas más personalizadas y con la conexión con una API de inteligencia artificial poder implementar y generar los códigos de RPA con un nivel de mayor dificultad. Así mismo, mejorar su interfaz gráfica. Por otro lado, fusionar el algoritmo de detección de objetos YOLO con la técnica por detección de color para mejorar la precisión del componente de visión computacional al momento de realizar el reconocimiento de diferentes ventanas del software contable y de los controladores con los que interactúa el usuario.

9. REFERENCIAS

Alaña C., Solórzano S., T. P., & , Sayonara, S. (2015). Procesos contables básicos para no contadores. In *Espol* (Machala :).

Álarcon, G. (2014). El Proceso Contable: Análisis E Interpretación De La Información Contable En Las Organizaciones Actuales. *Méthodos*, *12*(12), 92–101. http://www.ucipfg.com/Repositorio/MAP/MAPD-02/UNIDADES_DE_APRENDIZAJE/UNIDAD_1/LECTURAS/Vision_y_mision_de_una_empresa.pdf

Alpaydin, E. (2014). Introduction to Machine Learning Ethem Alpaydin. *Introduction to Machine Learning, Third Edition*.

Amodeo, E. (2010). *¿Qué son los DSL (Domain Specific Languages)?* https://eamodeorubio.wordpress.com/2010/09/13/¿que-son-los-dsl-domain-specific-languages/

Ashish. (2018). *Understanding Edge Detection (Sobel Operator) - Data Driven Investor - Medium*. https://medium.com/datadriveninvestor/understanding-edge-detection-sobel-operator-2aada303b900

AuraPortal. (2018, June 7). *RPA: Robotic Process Automation - Qué es y cómo nos ayuda • AuraPortal*. https://www.auraportal.com/es/rpa-robotic-process-automation-que-es/

Automation Anywhere. (2020). *Casos de estudio de clientes | Automation Anywhere*. https://www.automationanywhere.com/la/customers/case-studies

Azevedo, A., & Filipe Santos, M. (2008, January). *(PDF) KDD, semma and CRISP-DM: A parallel overview*. https://www.researchgate.net/publication/220969845_KDD_semma_and_CRISP-DM_A_parallel_overview

Bagnato, J. I. (2018a). *Convolutional Neural Networks: La Teoría explicada en Español | Aprende Machine Learning*. https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/

Bagnato, J. I. (2018b, November 29). *Convolutional Neural Networks: La Teoría explicada en Español | Aprende Machine Learning*. https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/

Barchard, K. A., & Pace, L. A. (2011). Preventing human error: The impact of data entry methods on data accuracy and statistical results. *Computers in Human Behavior*, *27*(5), 1834–1839. https://doi.org/10.1016/j.chb.2011.04.004

Beltramelli, T. (2018). pix2code: Generating code from a graphical user interface screenshot. *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems, EICS 2018*, 1–9. https://doi.org/10.1145/3220134.3220135

BMind Licencias. (2019). *IBM RPA (Robotic Process Automation) - BMind Licencias*. https://bmind.com/licencias/ibm-rpa

Burke, B., Cearley, D., Litan, A., Groombridge, D., & Mahdi, D. (2020). Top 10 Strategic Technology Trends for 2020: Practical Blockchain. *Gartner*, *October 2019*, 1–13.

Calvo, D. (2019). *Aprendizaje no supervisado - Diego Calvo*. Diegocalvo.Es. http://www.diegocalvo.es/aprendizaje-no-supervisado/

Capocchi, L., Santucci, J. F., & Ville, T. (2013). Software test automation using DEVSimPy environment. *SIGSIM-PADS 2013 - Proceedings of the 2013 ACM SIGSIM Principles of Advanced Discrete Simulation*, 343–348. https://doi.org/10.1145/2486092.2486137

Capterra. (2019). *UiPath Robotic Process Automation - Opiniones, precios, y características - Capterra España 2020*. https://www.capterra.es/software/135186/uipath-robotic-process-automation

CGN, C. G. de la N. (2014). *Doctrina Contable Pública Compilada Actualizada Del 2 de enero al 31 de diciembre de 2014*. 1–1391. http://www.contaduria.gov.co/wps/wcm/connect/9903da6e-11e6-44a5-a1f0-effa8cac282c/DOCTRINA+contablePublicaDic312013.pdf?MOD=AJPERES&CACHEID=9903da6e-11e6-44a5-a1f0-effa8cac282c

Chang, T. H., Yeh, T., & Miller, R. C. (2010). GUI testing using computer vision. *Conference on Human Factors in Computing Systems - Proceedings*, *3*(Figure 1), 1535–1544. https://doi.org/10.1145/1753326.1753555

Chollet, F. (2018). Deep Learning with Phyton. In *Manning*.

Christensson, P. (2009). *User Interface Definition*. https://techterms.com/definition/user_interface

Congreso de Colombia. (2012). *Ley 1575 de 2012 "Por medio de la cual se establece la Ley General de Bomberos de Colombia."*

Cooper, L. A., Holderness, D. K., Sorensen, T. L., & Wood, D. A. (2019). Robotic process automation in public accounting. *Accounting Horizons*, *33*(4), 15–35. https://doi.org/10.2308/acch-52466

Cowley, J. (2018). Redes neuronales convolucionales. *Ibm*, 1. https://www.ibm.com/developerworks/ssa/library/cc-convolutional-neural-network-vision-recognition/index.html

DANE. (2012). *Revisión 4 adaptada CIIU Rev . 4 A . C .* 496. https://www.dane.gov.co/files/nomenclaturas/CIIU_Rev4ac.pdf

DataWow. (2018). *Interns Explain CNN - Data Wow*. https://blog.datawow.io/interns-explain-cnn-8a669d053f8b

Deloitte. (2017, May 25). *¿Qué es Robotic Process Automation? | Deloitte España*. https://www2.deloitte.com/es/es/pages/operations/articles/que-es-robotic-process-automation.html

Deloitte. (2020). Tech Trends 2020. *Deloitte Insights*, 1–130. https://www2.deloitte.com/us/en/insights/focus/tech-trends.html

Dhakal, V., Feit, A. M., Kristensson, P. O., & Oulasvirta, A. (2018). Observations on typing from 136 million keystrokes. *Conference on Human Factors in Computing Systems - Proceedings*, *2018-April*. https://doi.org/10.1145/3173574.3174220

Díaz Moreno, H. (2006). Contabilidad general: enfoque práctico con aplicaciones informáticas. In *Editorial Mc Graw Hill Interamericana SA*. https://www.biblionline.pearson.com/Pages/BookRead.aspx

EcuRed. (2015). *EcuRed*. https://www.ecured.cu/Barra_de_desplazamiento

Ernesto Arévalo-Vázquez, E., Zúñiga-López, A., Villegas-Cortez, J., & Avilés-Cruz, C. (2015). Implementación de reconocimiento de objetos por color y forma en un robot móvil. In 21 Research in Computing Science (Vol. 91).

FAEDIS. (2018, September 10). FAEDIS. http://virtual.umng.edu.co/distancia/ecosistema/odin/odin_desktop.php?path=Li4vb3Zhcy9hZG1pbmlzdHJhY2lvbl9lbXByZXNhcy9jb250YWJpbGlkYWRfZ2VuZXJhbC91bmlkYWRfMS8=#slide_5.2

Fernando F. Coelho. (2019). *Introducción a Selenium: Cómo funciona, Características y Opciones*. https://www.digital55.com/desarrollo-tecnologia/herramientas-testing-introduccion-selenium/

Fisher, R., Perkins, S., Walker, A., & Wolfart, E. (2003). *Feature Detectors - Canny Edge Detector*. http://homepages.inf.ed.ac.uk/rbf/HIPR2/sobel.htm

Gallardo Arancibia, J. A. (2013). *Metodología para el Desarrollo de Proyectos en Minería de Datos CRISP-DM*. *84*, 487–492. http://ir.obihiro.ac.jp/dspace/handle/10322/3933

Garcia, A. (2016). *Automatización de pruebas de interfaz gráfica en herramientas de tesorería*. https://www.iit.comillas.edu/pfc/resumenes/578e702f6cafb.pdf

García, E. M. i. (2002). Visión Artificial. In *Inteligencia Artificial*.

glosarios@servidor-alicante.com. (2015). Eficiencia (Contabilidad de gestión). *Glosarios@servidor-Alicante.Com*. https://glosarios.servidor-alicante.com/contabilidad-de-gestion/eficiencia

GNOME developer. (2014). *GNOME developer*. https://developer.gnome.org/hig/stable/toolbars.html.es

Gollapudi, S. (2019). *Learn computer vision using OpenCV : with deep learning CNNs and RNNs*.

Gonzales, R. (2019). *Fundamentos para diseñar una Arquitectura de Solución con RPA*.

Guru99. (n.d.). *Clasificación de imágenes de TensorFlow: CNN (Red Neural Convolucional) - Guru99*. Retrieved April 24, 2020, from https://guru99.es/convnet-tensorflow-image-classification/#2

helpsystems. (2020). *Software de automatización GUI*. https://www.helpsystems.com/es/productos/automate/software-de-automatizacion-gui-macros

Huang, R., Long, Y., & Chen, X. (2016). Automaticly generating web page from a mockup. *Proceedings of the International Conference on Software Engineering and Knowledge Engineering, SEKE*, *2016-Janua*, 589–594. https://doi.org/10.18293/SEKE2016-231

Hureño, O. (2010). *Contabilidad Básica Colección Didáctica Ciencias Económicas Y Administrativas.* https://www.sanmateo.edu.co/documentos/publicacion-contabilidad-basica.pdf

IBM, I. B. M. (2012). Manual CRISP-DM de IBM SPSS Modeler. *IBM Corporation*, 56. http://www.ibm.com/spss.%0Aftp://public.dhe.ibm.com/software/analytics/spss/documentation/modeler/15.0/es/CRISP-DM.pdf

IBM Robotic Process Automation. (2020). *Robotic Process Automation with Automation Anywhere - Colombia | IBM*. IBM Robotic Process Automation.

(n.d.). Robotic Process Automation with Automation Anywhere - Colombia | IBM. Retrieved April 1, 2020, from Https://Www.Ibm.Com/Co-Es/Products/Robotic-Process-Automation. https://www.ibm.com/co-es/products/robotic-process-automation

Identigate. (2018). *Manual Data Entry: The weak link in automated Systems – Identigate: Web and Mobile Identity Management Solutions*. http://www.identigate.co.ke/2018/04/14/manual-data-entry-the-weak-link-in-automated-systems/

ISO. (2009). *ISO 9001 - Software ISO 9001 de Sistemas de Gestión ISO*. ISOTools Excellence. https://www.isotools.org/normas/calidad/iso-9001?__hstc=268265809.657f678a4e6ad8c124f59cda1704dff7.1588472668048.1588472668048.1588472668048.1&__hssc=268265809.2.1588472668048&__hsfp=1312440609

ISO 25000. (2016). *ISO 25000 Portal*. https://iso25000.com/

Iso25000. (2018). *NORMAS ISO 25000*. ISO 25000. https://iso25000.com/index.php/normas-iso-25000?limit=4&start=4

ISOL. (2019). *RPA (Robotic Process Automation) Beneficios | ISOL*. https://isol.mx/rpa-robotic-process-automation-beneficios/

Ki, J., & Kwon, K. (2019a). Proceedings of the Sixth International Conference on Green and Human Information Technology. In *Proceedings of the Sixth International Conference on Green and Human Information Technology. ICGHIT 2018* (Vol. 502). Springer Singapore. https://doi.org/10.1007/978-981-13-0311-1

Ki, J., & Kwon, K. (2019b). Proceedings of the Sixth International Conference on Green and Human Information Technology. *Proceedings of the Sixth International Conference on Green and Human Information Technology. ICGHIT 2018*, *502*, 10–13. https://doi.org/10.1007/978-981-13-0311-1

Kim, B., Park, S., & Kim, B. (2018). *Deep - Learning Based Web UI Automatic Programming*. 2–3.

Kokina, J., & Blanchette, S. (2019). Early evidence of digital labor in accounting: Innovation with Robotic Process Automation. *International Journal of Accounting Information Systems*, *35*, 100431. https://doi.org/10.1016/j.accinf.2019.100431

Lemley, J., Bazrafkan, S., & Corcoran, P. (2017). Deep Learning for Consumer Devices and Services: Pushing the limits for machine learning, artificial intelligence, and computer vision. *IEEE Consumer Electronics Magazine*, *6*(2), 48–56. https://doi.org/10.1109/MCE.2016.2640698

Levy Steven. (2015). *Graphical user interface | computing | Britannica*. https://www.britannica.com/technology/graphical-user-interface

Luenendonk, M. (2017, October 20). *Accounting Errors*. https://www.cleverism.com/lexicon/accounting-errors/

Maitra Satyajit. (2019, February 24). *What Canny Edge Detection algorithm is all about? - SATYAJIT MAITRA - Medium*. https://medium.com/@ssatyajitmaitra/what-canny-edge-detection-algorithm-is-all-about-103d94553d21

Mays, J. A., & Mathias, P. C. (2019). Measuring the rate of manual transcription error in outpatient point-of-care testing. *Journal of the American Medical Informatics Association*, *26*(3), 269–272. https://doi.org/10.1093/jamia/ocy170

Mihir Mistry, Ameya Apte, Varad Ghodake(&), and S. B. M. (2019). Machine Learning Based User Interface Generation. In *Robotics and Autonomous Systems* (Vol. 7, Issues 2–3). https://doi.org/10.1016/0921-8890(91)90033-H

Moreno, A. (2017). *¿Qué es el procesamiento de lenguaje natural?* Procesamiento Del Lenguaje Natural, ¿qué Es? https://www.sas.com/es_co/insights/analytics/what-is-natural-language-processing-nlp.html

Narayana, M., Raghu Ram Reddy, N., & Hyndavi Reddy, N. (2019). High speed script execution for GUI Automation using Computer Vision. *International Journal of Electrical and Computer Engineering*, *9*(1), 231–236. https://doi.org/10.11591/ijece.v9i1.pp231-236

Nguyen, T. A., & Csallner, C. (2016). Reverse engineering mobile application user interfaces with REMAUI. *Proceedings - 2015 30th IEEE/ACM International Conference on Automated Software Engineering, ASE 2015*, 248–259. https://doi.org/10.1109/ASE.2015.32

OBS. (2020). *Características y fases del modelo incremental*. OBS Business School. https://obsbusiness.school/int/blog-project-management/metodologias-agiles/caracteristicas-y-fases-del-modelo-incremental

Organizaci, P. D. E. L. A., Iv, J., & Vicepresidente, O. (2018). *Aprovechar la automatización inteligente de procesos : El 1300 % de retorno de la inversión genera una mayor satisfacción de los clientes y USD 7 millones en nuevas fuentes de ingresos*.

Pete, C., Julian, C., Randy, K., Thomas, K., Thomas, R., Colin, S., & Wirth, R. (2000). Crisp-Dm 1.0. *CRISP-DM Consortium*, 76.

Pressman, R. (2002a). *Ingeniería del Software. Un enfoque práctico.* http://cotana.informatica.edu.bo/downloads/ld-Ingenieria.de.software.enfoque.practico.7ed.Pressman.PDF

Pressman, R. (2002b). *Ingeniería del Software. Un enfoque práctico.*

R., A. (2011). *La MISION DE UNA EMPRESA*. 1–6. http://www.crecenegocios.com/la-mision-de-una-empresa/el

Radhakrishnan, P. (2017, November 17). *What is Transfer Learning? - Towards Data Science*. https://towardsdatascience.com/what-is-transfer-learning-8b1a0fa42b4

Ray, S., Tornbohm, C., Miers, D., & Kerremans, M. (2019). *Magic Quadrant for Robotic Process Automation Software*. *July*, 1–40.

Redmon, Joseph, Santosh Divvala, R. G., & Farhadi, A. (2016). *You Only Look Once: Unified, Real-Time Object Detection*.

Remanan Surya. (2019, April 28). *Beginner's Guide to Object Detection Algorithms - Analytics Vidhya - Medium*. https://medium.com/analytics-vidhya/beginners-guide-to-object-detection-algorithms-6620fb31c375

Robot Framework. (2008, August 21). *Robot Framework*. https://robotframework.org/

Rouse, M. (2017). *¿Qué es Aprendizaje profundo (deep learning)? - Definición en WhatIs.com*. Abril 2017. https://searchdatacenter.techtarget.com/es/definicion/Aprendizaje-profundo-deep-learning

scannmore. (2018, July 15). *The Biggest Disadvantages of Manual Data Entry*. https://scannmore.com/manual-data-entry-disadvantages/

Leyes desde 1992 - Vigencia expresa y control de constitucionalidad [DECRETO_2811_1974], (2006). http://www.secretariasenado.gov.co/senado/basedoc/constitucion_politica_1991_pr011.html#354

Sevilla, P. (2018, July 7). *Lenguaje de programación Python: qué es, utilidades y ventajas*. https://initiumsoft.com/blog/que-es-el-lenguaje-de-programacion-python-y-para-que-sirve/

Standardization, F. O. R., & Normalisation, D. E. (1987). *International Standard Iso*. *1987*.

Sucar, L. E., & Gómez, G. (2011). Vision Computacional. *Instituto Nacional de Astrofísica, Óptica y Electrónica*, 185. http://ccc.inaoep.mx/~esucar/Libros/vision-sucar-gomez.pdf

The MathWorks, I. (2019). *¿Qué es una red neuronal? - MATLAB & Simulink*. 2019. https://la.mathworks.com/discovery/neural-network.html

ThinkAutomation. (2018, September 19). *Everything wrong with manual data entry - ThinkAutomation*. https://www.thinkautomation.com/productivity/everything-wrong-with-manual-data-entry/

Torres, L. G. (2018). *PA181-3-DotNetGen DotNetGenerator : Generador de Código para Arquitectura Microsoft . NET a partir de modelos ISML DotNetGenerator : Generador de Código para Arquitectura Microsoft . NET a partir de modelos ISML*.

Towards Machine Learning. (2018). *Deep Learning Series, P2: Understanding Convolutional Neural Networks – Towards Machine Learning*. https://towardsml.com/2018/10/16/deep-learning-series-p2-understanding-convolutional-neural-networks/

UiPath. (2017). *¿Qué es RPA (Automatización Robótica de Procesos)? | UiPath®*. https://www.uipath.com/es/rpa/automatizacion-robotica-de-procesos

UiPath. (2019). Capterra. Obtenido de https://www.capterra.co/software/135186/uipath-robotic-process-automation

Uskenbayeva, R., Kalpeyeva, Z., Satybaldiyeva, R., Moldagulova, A., & Kassymova, A. (2019). Applying of RPA in Administrative Processes of Public Administration. Proceedings - 21st IEEE Conference on Business Informatics, CBI 2019, 2, 9–12. https://doi.org/10.1109/CBI.2019.10089

Villena Román, J. (2016, August 2). *CRISP-DM: La metodología para poner orden en los proyectos - Sngular*. https://www.sngular.com/es/data-science-crisp-dm-metodologia/

Yun, Y., & Park, J. (2018). *Automatic Mobile Screen Translation Using Object Detection Approach Based on Deep Neural Networks*. *21*(11), 1305–1316.