	UNIVERSIDAD SURCOLOMBIANA GESTIÓN DE BIBLIOTECAS						
	CARTA DE AUTORIZACIÓN						
CÓDIGO	AP-BIB-FO-06	VERSIÓN	1	VIGENCIA	2014	PÁGINA	1 de 1

Neiva, Huila

Señores

CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN

UNIVERSIDAD SURCOLOMBIANA

El (Los) suscrito(s):

Sebastián Cruz Dussan, con C.C. No.1081159725, Andres Felipe Aristizabal Morales, con C.C. No 1075298280.,

Autor(es) de la tesis y/o trabajo de grado titulado DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA PARA FACILITAR LA DATIFICACIÓN DE TEXTO Y GESTIÓN EMPRESARIAL USANDO VISIÓN POR COMPUTADORA Y MACHINE LEARNING presentado y aprobado en el año 2024 como requisito para optar al título de ingeniero electrónico; Autorizo(amos) al CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN de la Universidad Surcolombiana para que, con fines académicos, muestre al país y el exterior la producción intelectual de la Universidad Surcolombiana, a través de la visibilidad de su contenido de la siguiente manera:

- Los usuarios puedan consultar el contenido de este trabajo de grado en los sitios web que administra la Universidad, en bases de datos, repositorio digital, catálogos y en otros sitios web, redes y sistemas de información nacionales e internacionales “open access” y en las redes de información con las cuales tenga convenio la Institución.

- Permita la consulta, la reproducción y préstamo a los usuarios interesados en el contenido de este trabajo, para todos los usos que tengan finalidad académica, ya sea en formato Cd-Rom o digital desde internet, intranet, etc., y en general para cualquier formato conocido o por conocer, dentro de los términos establecidos en la Ley 23 de 1982, Ley 44 de 1993, Decisión Andina 351 de 1993, Decreto 460 de 1995 y demás normas generales sobre la materia.

- Continúo conservando los correspondientes derechos sin modificación o restricción alguna; puesto que, de acuerdo con la legislación colombiana aplicable, el presente es un acuerdo jurídico que en ningún caso conlleva la enajenación del derecho de autor y sus conexos.

De conformidad con lo establecido en el artículo 30 de la Ley 23 de 1982 y el artículo 11 de la Decisión Andina 351 de 1993, “Los derechos morales sobre el trabajo son propiedad de los autores” , los cuales son irrenunciables, imprescriptibles, inembargables e inalienables.

EL AUTOR/ESTUDIANTE:

Sebastian Camilo Cruz Dussan

firma:

EL AUTOR/ESTUDIANTE:

Andres Felipe Aristizabal Morales

firma:

TÍTULO COMPLETO DEL TRABAJO: DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA PARA FACILITAR LA DATIFICACIÓN DE TEXTO Y GESTIÓN EMPRESARIAL USANDO VISIÓN POR COMPUTADORA Y MACHINE LEARNING

AUTOR O AUTORES:

Primero y Segundo Apellido	Primero y Segundo Nombre
cruz dussan	sebastian camilo

Primero y Segundo Apellido	Primero y Segundo Nombre
aristizabal morales	andres felipe

DIRECTOR Y CODIRECTOR TESIS:

Primero y Segundo Apellido	Primero y Segundo Nombre
Mosquera Cerquera	Vladimir

ASESOR (ES):

Primero y Segundo Apellido	Primero y Segundo Nombre
salgado patron	jose de jesus

Primero y Segundo Apellido	Primero y Segundo Nombre
díaz franco	fernand

PARA OPTAR AL TÍTULO DE: ingeniero electrónico

FACULTAD: **ingenieria**

PROGRAMA O POSGRADO: ingeniería electrónica

CIUDAD: neiva AÑO DE PRESENTACIÓN: 2024 NÚMERO DE PÁGINAS: 86

TIPO DE ILUSTRACIONES (Marcar con una **X**)**:**

Diagramas x Fotografías x Grabaciones en discos ___ Ilustraciones en general x Grabados ___ Láminas ___ Litografías ___ Mapas ___ Música impresa ___ Planos ___ Retratos ___ Sin ilustraciones ___ Tablas o Cuadros x

PALABRAS CLAVES EN ESPAÑOL E INGLÉS:

	Español	Inglés		Español	Inglés
1.	DIGITALIZACIÓN	DIGITIZATION	6.	GOOGLE CLOUD VISION	GOOGLE CLOUD VISION
2.	REACT	REACT			
3.	SOFTWARE LIBRE	OPEN SOURCE			
4.	API REST	REST API			
5.	ODOO	ODOO			

RESUMEN DEL CONTENIDO: (Máximo 250 palabras)

Este trabajo corresponde al diseño e implementación de un aplicativo de usuario final para la digitalización de venta para control de inventario en una empresa piloto en un ambiente no controlado. El aplicativo consta de la implementación del servidor y la interfaz de usuario. El servidor se implementó en una plataforma open-source para la autenticación de usuario y la lógica de control del inventario, y para la interfaz de usuario se realizó una página web construida en React, que se comunica con el servidor mediante un api Rest. Para la digitalización del registro de venta se creó un módulo en Odoo, que implementa el api de los servicios de Google en la nube de visión por computadora para actualizar el inventario.

ABSTRACT: (Máximo 250 palabras)

This work corresponds to the design and implementation of an end-user application for the digitization of sales for inventory control in a pilot company in an uncontrolled environment. The application consists of the server implementation and the user interface. The server was implemented on an open-source platform for user authentication and inventory control logic, and a web page built in React was created for the user interface, which communicates with the server through a Rest api. For the digitization of the sales record, a module was created in Odoo, which implements the api of Google's services in the computer vision cloud to update the inventory.

Nombre Presidente Jurado: Vladimir Hosquera Cerquera

Firma: Vladimir Hosquera.

Nombre Jurado: José Fernando Barrera Compo

Firma: José Fernando Barrera Compo

Nombre Jurado: José de Jesus Salgado Patoi

Firma:

UNIVERSIDAD SURCOLOMBIANA GESTIÓN DE BIBLIOTECAS							
DESCRIPCIÓN DE LA TESIS Y/O TRABAJOS DE GRADO							
CÓDIGO	AP-BIB-FO-07	VERSIÓN	1	VIGENCIA	2014	PÁGINA	3 de 3

This work corresponds to the design and implementation of an end-user application for the digitization of sales for inventory control in a pilot company in an uncontrolled environment. The application consists of the server implementation and the user interface. The server was implemented on an open-source platform for user authentication and inventory control logic, and a web page built in React was created for the user interface, which communicates with the server through a Rest api. For the digitization of the sales record, a module was created in Odoo, which implements the api of Google's services in the computer vision cloud to update the inventory.

APROBACION DE LA TESIS

Nombre Presidente Jurado: Vladimir Mosquera Cerquera

Firma: Vladimir Mosquera.

Nombre Jurado: José Fernando Berrera Campo

Firma: José Fernando Berrera Campo

Nombre Jurado: Jairo de Jesus Salgado Patiño

Firma:

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA PARA FACILITAR LA DATIFICACIÓN DE TEXTO Y GESTIÓN EMPRESARIAL USANDO VISIÓN POR COMPUTADORA Y MACHINE LEARNING

SEBASTIAN CAMILO CRUZ DUSSAN
ANDRES FELIPE ARISTIZABAL MORALES

UNIVERSIDAD SURCOLOMBIANA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA ELECTRÓNICA
NEIVA, HUILA
2023

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA PARA FACILITAR LA DATIFICACIÓN DE TEXTO Y GESTIÓN EMPRESARIAL USANDO VISIÓN POR COMPUTADORA Y MACHINE LEARNING

SEBASTIAN CAMILO CRUZ DUSSAN
ANDRÉS FELIPE ARISTIZABAL MORALES

PROYECTO DE GRADO PARA OPTAR AL TÍTULO DE INGENIERO ELECTRÓNICO

Director
VLADIMIR MOSQUERA CERQUERA
Magíster en Ingeniería Electrónica

UNIVERSIDAD SURCOLOMBIANA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA Electrónica
NEIVA, HUILA
2023

Nota de aceptación:

_____

_____

_____

_____

_____

_____

_____

_____

Director

_____

Firma del Jurado

_____

Firma del Jurado

DEDICATORIA.

Dedico este trabajo de grado a mis padres, ellos son las personas más importantes en mi vida, que me han ayudado de forma moral y económica. También dedico este trabajo a mi familia y amigos cercanos, quienes han estado a mi lado, durante el tiempo que he trabajo en este proyecto. Ellos son quienes me han apoyado y prestaron su ayuda, y un especial agradecimiento a nuestro director que a pesar de estar en estos tiempos difíciles de pandemia nos ayudó con su tiempo y conocimiento.

-Sebastian Camilo Cruz Dussan.

Dedico este trabajo a Dios que ha llenado de bendiciones y oportunidades mi existencia logrando superar los retos de la vida diaria. También dedico este trabajo a mis padres por brindarme el apoyo necesario para poder culminar mis estudios brindándome apoyo moral y económico. Por último y no menos importante, una dedicación especial a mi hija Antonella, mi motivación para seguir adelante.

-Andrés Felipe Aristizábal Morales.

AGRADECIMIENTO

De manera inicial queremos agradecer al ingeniero Vladimir por darnos su apoyo condicional para poder culminar el trabajo de grado, a todos los ingenieros que nos brindaron los conocimientos en nuestra carrera universitaria y a cada uno de nuestros compañeros por brindarnos un ambiente compañerismo, respeto y apoyo haciendo la carrera más agradable.

GLOSARIO

ODOO: es un conjunto de aplicaciones de código abierto que sirven de gestión empresarial que incluye una gama de herramientas de fácil uso para optimizar y rentabilizar los negocios.

REACT: es una librería JavaScript diseñada para el desarrollo de interfaces de usuario optimizando el código.

OPEN-SOURCE: capacidad de ejercer cierto grado de independencia, es decir tomar decisiones sin necesidad de la opinión de un tercero.

ALGORITMOS: secuencia de instrucciones finitas que permite solucionar un problema y/o llevar a cabo una serie de tareas.

REDES NEURONALES CONVOLUCIONALES: es un tipo de redes neuronales artificiales donde las neuronas corresponden a campos receptivos, este tipo de red es una variación de un perceptrón multicapa, este tipo de red es muy utilizado para trabajo de visión artificial.

PROCESAMIENTO DE IMÁGENES: grupo de métodos para el tratamiento de imágenes en busca de mejorar la calidad o facilitar búsqueda de información sobre la misma.

SENSORES: dispositivos encargados de reaccionar a estímulos y acciones externas, como consecuencia se obtiene una respuesta, su principio es transformar magnitudes físicas o químicas en magnitudes eléctricas.

PYTHON: lenguaje de programación dinámico, versátil y multiplataforma, frecuentemente utilizado por su fácil manejo, además de ser robusto de tal manera que puede soportar varios procesos o paradigmas de desarrollo.

TABLA DE CONTENIDO

pág.

LISTA DE ILUSTRACIONES

pág.

LISTA DE TABLAS

pág.

RESUMEN

Este trabajo corresponde al diseño e implementación de un aplicativo de usuario final para la digitalización de venta para control de inventario en una empresa piloto en un ambiente no controlado. El aplicativo consta de la implementación del servidor y la interfaz de usuario. El servidor se implementó en una plataforma open-source para la autenticación de usuario y la lógica de control del inventario, y para la interfaz de usuario se realizó una página web construida en React, que se comunica con el servidor mediante un api Rest. Para la digitalización del registro de venta se creó un módulo en Odoo, que implementa el api de los servicios de Google en la nube de visión por computadora para actualizar el inventario.

DIGITALIZACIÓN, REACT, OPEN SOURCE, API REST, ODOO, GOOGLE CLOUD VISION.

ABSTRACT

This work corresponds to the design and implementation of an end-user application for the digitization of sales for inventory control in a pilot company in an uncontrolled environment. The application consists of the server implementation and the user interface. The server was implemented on an open-source platform for user authentication and inventory control logic, and a web page built in React was created for the user interface, which communicates with the server through a Rest api. For the digitization of the sales record, a module was created in Odoo, which implements the api of Google's services in the computer vision cloud to update the inventory.

DIGITIZATION, REACT, OPEN SOURCE, REST API, ODOO, GOOGLE CLOUD VISION.

INTRODUCCIÓN

La globalización y conectividad es un tema que va creciendo y tocando varios sectores de la sociedad, en su mayor medida al sector empresarial. El aprovechamiento de los datos que se producen en la realización de múltiples actividades ha sido fundamental para aumentar el crecimiento de las empresas, por tal motivo se tiene que trabajar de manera eficaz y objetiva con los datos adquiridos para ser más competentes.

A pesar de esto la digitalización de las microempresas en Colombia está en una situación muy precaria, debido a que los procesos que se realizan en los documentos e información para gestión administrativa de forma productiva son en ambientes controlados que consumen mucho tiempo y recurso que la mayoría de las microempresas no están dispuestas o en la capacidad de usar. Afortunadamente, los avances recientes en la visión por computadora permiten abarcar la detección de texto con otros enfoques más permisivos, que son en entornos no controlados.

En este orden de ideas, este proyecto de grado realiza un aplicativo de usuario final que permite la digitalización de las ventas con el nuevo enfoque, el reconocimiento de texto en imágenes en entornos no controlados, el cual se aleja al de documentos impresos y se acerca más a la detección de objetos o escritos a mano alzada, con mayor tolerancia al cambio de atributos naturales como luminosidad, sombras, estilos, fuentes y fondos. Con este enfoque se permite que la empresa piloto digitalice sus ventas con una herramienta no invasiva a sus procesos, haciéndola más precisa en la gestión administrativa.

1. PLANTEAMIENTO DEL PROBLEMA

Las empresas tienen un papel fundamental en el desarrollo social y económico de un país, por eso es de suma importancia su adaptabilidad a cambios, ya que con el pasar de los años la sociedad comienza a desarrollar nuevas herramientas con el fin de facilitar sus actividades cotidianas. Una de estas actividades es la administración empresarial, actividad muy compleja debido a la gran cantidad de datos inherentes con las que se tiene que trabajar de manera eficaz y objetiva. Una herramienta actual que optimizará el trabajo administrativo aprovechando la información obtenida de los datos, es la digitalización de la gestión documental y venta física de forma inteligente con tecnologías avanzadas como el Big data y la inteligencia artificial.

La digitalización de las empresas en Colombia está en una situación muy precaria, según fuentes como la Revista Dinero informa que "según el Departamento Nacional de Planeación, Colombia tiene un indicador de aprovechamiento de los datos en un 9.3%; Las entidades públicas se encuentran digitalizadas en un 50% y además solo el 4% de las entidades cuentan con las condiciones para la implementación de Big data y aprovechamiento de los datos" [1] y cifras del observatorio de economía digital, "el porcentaje de implementación de Big data es 3.2%, 16.8%, 4%, 1.3% y en Inteligencia artificial es del 1.6%, 9.7%, 2.4%, 0.7% en entidades nacionales, grandes empresas, pymes y micros respectivamente"[2].

La digitalización de documentos para la gestión administrativa se lleva a cabo generalmente en ambientes controlados, lo que puede ser costoso en términos de tiempo y recursos. Sin embargo, muchas organizaciones no tienen la disposición o capacidad para utilizar estos entornos controlados, lo que los lleva a realizar este proceso en ambientes no controlados. El reconocimiento de texto en estos escenarios presenta desafíos distintos a los del reconocimiento de documentos convencionales, ya que se asemeja más a la identificación de objetos o escritura a mano alzada. Esto se debe a la presencia de características naturales como luminosidad, sombras, estilos, fuentes y fondos, que pueden afectar la precisión del reconocimiento de texto en dichos entornos.

En este orden de ideas, ¿Cómo contribuir desde la formación recibida en la Universidad Surcolombiana al desarrollo e implementación de un sistema que

[1] Colombia, el primer país latino en tener política pública de datos. (2018, 19 de abril). Semana.com Últimas Noticias de Colombia y el Mundo. https://www.semana.com/pais/articulo/colombia-ya-tiene-politica-publica-de-big-data/257479/

[2] MinTIC revela los primeros resultados del Observatorio de Economía (2017,6 de diciembre). https://mintic.gov.co/portal/inicio/Sala- deprensa/Noticias/61929:MinTICrevela-los-primeros-resultados-del-Observatorio-deEconomiaDigital#:~:text=Los%20resultados%20muestran%20que%20si,para%20mejorar%20los%20procesos%20productivos

permite extraer información textual en entornos real y contribuya en la digitalización de las empresas?

2. JUSTIFICACIÓN

En la actualidad la datificación es un método que está revolucionando los sistemas tradicionales de obtener y guardar información de actividades empresariales que generan datos primordiales de una organización o compañía; esta nueva tecnología incentiva el crecimiento de una empresa facilitando labores administrativas. El aprovechamiento de los datos obtenidos mediante el método de datificación beneficia en cuanto al aumento económico, progreso social, solución a problemas políticos que se puedan ocasionar en el ámbito empresarial y facilita la innovación de ideas que permitan la adaptabilidad de una sociedad por los patrones sociales obtenidos mediante la información compilada.

Considerando que la cuarta revolución industrial ha generado la transformación del mundo real a datos procesables y cuantificables, donde dichos procesos generan cada vez mayor cantidad de datos resultado del uso de los avances tecnológicos. Según el DNP (Departamento de Nacional de Planeación) “el crecimiento de datos pasó de 1.2 billones de gigabytes en 2010 a 16.1 billones en 2016 y se espera que para 2025 esta cifra se multiplique diez veces, es decir, que sean creados 163 billones de gigabytes”[3].

El reconocimiento de texto en entornos no controlados, es una importante tecnología en visión por computadora debido al aumento de dispositivos móviles con cámara de gran calidad y la importancia de la datificación de las fuentes de información físicas en escenarios no controlados, pero presenta grandes retos desde el punto de vista técnico porque el tamaño de la fuente puede ser muy diverso, pueden estar alineadas en muchas direcciones y con distintas distorsiones geométricas, su color en ocasiones es similar a el fondo, además los cambios en el ángulo de la cámara, dan distorsión de perspectiva que disminuyen el desempeño. Dar soluciones a estos problemas genera conocimiento con gran demanda en la actualidad por su importancia.

Finalmente lo que se busca con este proyecto de grado es afianzar los conocimientos y habilidades en el diseño e implementación de sistemas de adquisición y procesamiento de datos desde un ambiente no controlado mediante algoritmos de visión por computadora, fortalecer el carácter investigativo de los estudiantes desarrollando, afianzando conocimientos y habilidades requeridas en el perfil investigativo, necesario para ser profesionales de alto desempeño y calidad; y además se obtienen nuevos conocimientos en temas como machine learning y sistemas inteligentes y desarrollo web.

[3] Colombia primer país en Latinoamérica con política pública para la explotación de datos Big data. (2018, 19 de abril). https://2022.dnp.gov.co/Paginas/Colombia-primer-pa%C3%ADs-en-Latinoam%C3%A9rica-con-una-pol%C3%ADtica-p%C 3 %BAblica-para-la-explotaci%C3%B3n-de-datos-Big-Data.aspx

3. OBJETIVOS

3.1 OBJETIVO GENERAL

Diseñar e implementar un sistema para datificar fuentes de información física(texto) en entorno real.

3.2 OBJETIVOS ESPECÍFICOS

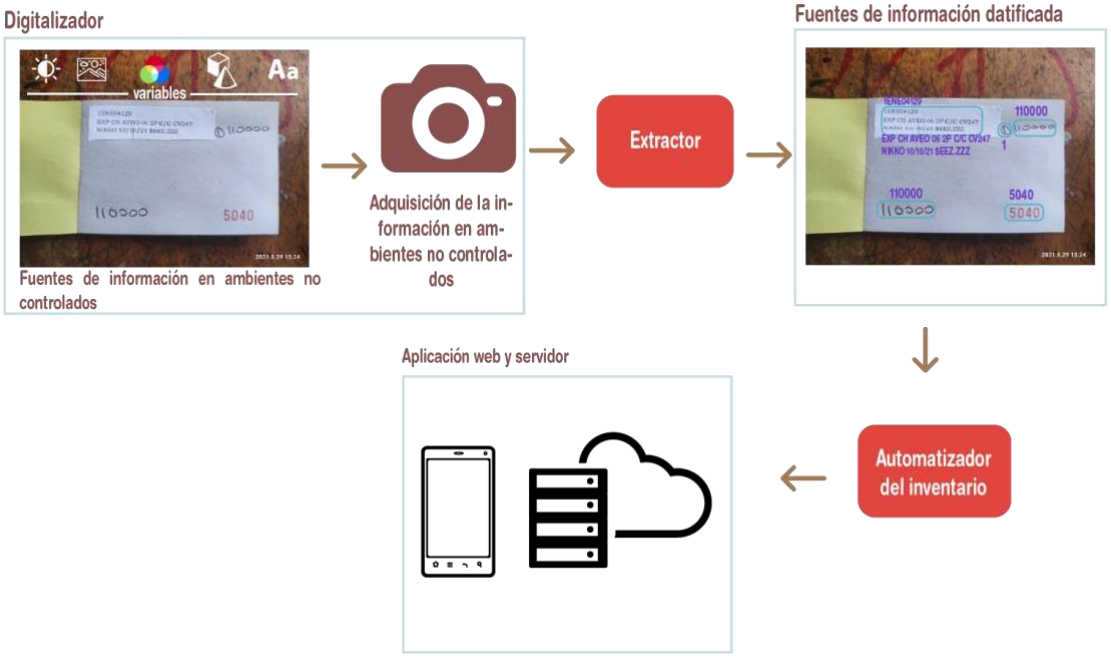
- Desarrollar o implementar un algoritmo que permita la localización del texto y reconocimiento óptico de caracteres en una escena natural.
- Integrar y evaluar los algoritmos establecidos.
- Desarrollar una página web que permita implementar el aplicativo para la prueba piloto.
- Verificar la funcionalidad del sistema con una empresa piloto.

4. MARCO TEÓRICO.

Este capítulo desarrolla los conceptos básicos necesarios para la comprensión del desarrollo de este trabajo de tesis.

La Ilustración 1, muestra la relación entre los componentes y los procesos esenciales del sistema de digitalización de texto desarrollado en esta tesis. El componente Digitalizador, está conformado por la fuente de información en ambientes no controlados y el sistema de captura de la imagen, la cual es procesada por el proceso Extractor para obtener el componente fuente de información datificada (el texto relevante para el negocio). Finalmente, el proceso Almacenamiento guarda la información datificada con el objetivo de ser aprovechada por el componente Aplicación Web y Servidor.

Ilustración 1. Relación entre los componentes y procesos del Sistema de Digitalización de Texto.



Fuente: Imagen realizada por los tesistas

4.1. DIGITALIZADOR

El digitalizador tiene un desarrollo de gran importancia en el funcionamiento de este proyecto de grado. Su principal función es captar la fuente de información en ambientes no controlados, luego la representa en una matriz numérica la cual puede ser leída por la computadora para sus respectivos procesos.

La fuente de información es la persona, imagen u objeto de los que se obtienen datos para ser analizados. El dato es el valor de una variable o de una constante, proporciona información sobre los procesos comerciales (ingreso y venta de un producto) y sirve de base para el análisis estadístico.

Las fuentes de información en ambientes no controlados son aquellas que se encuentran en un entorno donde sus condiciones son variables y se debe tener las condiciones mínimas lumínicas que permitan la visualización de esta. La imagen, es una fuente de información en ambientes no controlados ya digitalizada. Como se observa en la Ilustración 1, la fuente de información se encuentra en un ambiente comercial, donde el fondo, la iluminación, colores y tipos de fuentes del texto son variables.

El sistema empleado para transferir la información de un medio físico a un medio digital es un Dispositivo inteligente, dispositivo capturador de imagen que es sensible a un intervalo específico del espectro electromagnético. El sensor ha interactuado con la radiación y la ha registrado, conviene especificar que, después de la digitalización de la imagen (teniendo en cuenta la resolución y la tesselation), la interpretación de la imagen obtenida se puede expresar mediante una función matemática bidimensional donde su magnitud es la intensidad en una coordenada específica. Se obtienen características importantes de la imagen con el fin de realizar procesos computacionales para trabajar posteriormente con ella.

4.2 EXTRACTOR

En este proceso se extrae la información relevante de la fuente de información, ubicando y reconociendo texto de la imagen ya digitalizada. La literatura propone dos enfoques: el enfoque de extremo a extremo (end to end) que localiza y reconoce el texto en un solo algoritmo, y el enfoque que localiza mediante un algoritmo y reconoce texto mediante otro.

4.2.1 Método de extremo a extremo (end to end): Google cloud visión

La API de Cloud Visión permite integrar con facilidad la función de detección de reconocimiento óptico de caracteres (OCR). Esto permite el filtrado de información del taco digitalizado (la fuente de información en ambiente no controlado ya digitalizada). La API devuelve el texto extraído en una estructura jerarquizada, organizada en Páginas, Bloques, Párrafos, Palabras y Símbolos.

La herramienta de Cloud Visión se divide en dos funciones de anotación que permiten implementar el reconocimiento de texto de Google: TEXT_DETECTION y DOCUMENT_TEXT_DETECTION; ambas detectan texto en una imagen, la única diferencia es que la segunda es más eficiente para textos y documentos densos.

4.3 FUENTE DE INFORMACIÓN DATIFICADA

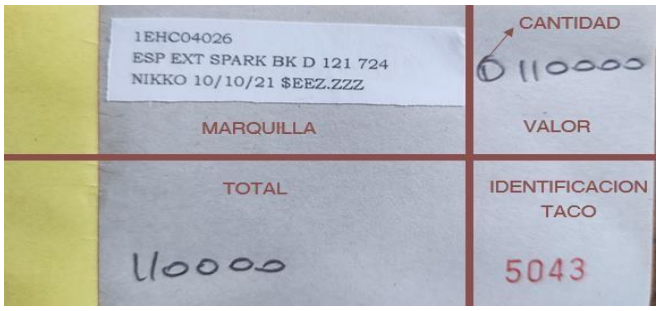
La Ilustración 2 representa al taco, este es una fuente de información física que permite registrar los datos de la venta de un producto en la empresa piloto. Por tal motivo este dará la información pertinente para digitalizar la venta realizada, modificando el inventario y obteniendo información de flujo de caja.

4.3.1 Documento a datificar (taco)

El "taco" es un documento crucial en la empresa Lujos y Eléctricos Cruz, ya que registra detalladamente las ventas de sus productos. Actualmente, la digitalización de este documento se efectúa de manera manual. Sin embargo, como parte de este trabajo de grado, nos enfocaremos en automatizar tanto la digitalización como la actualización del inventario. Por esta razón, resulta fundamental proporcionar una explicación acerca de su estructura.

Como se muestra la Ilustración 2, el taco se encuentra dividido en cuatro regiones; en la región superior izquierda se encuentra la marquilla, en la región superior derecha está ubicado el número de artículos vendidos y el precio, la parte inferior derecha del taco se localiza el identificador de frecuencia de los tacos y en la parte inferior izquierda se encuentra el total de venta.

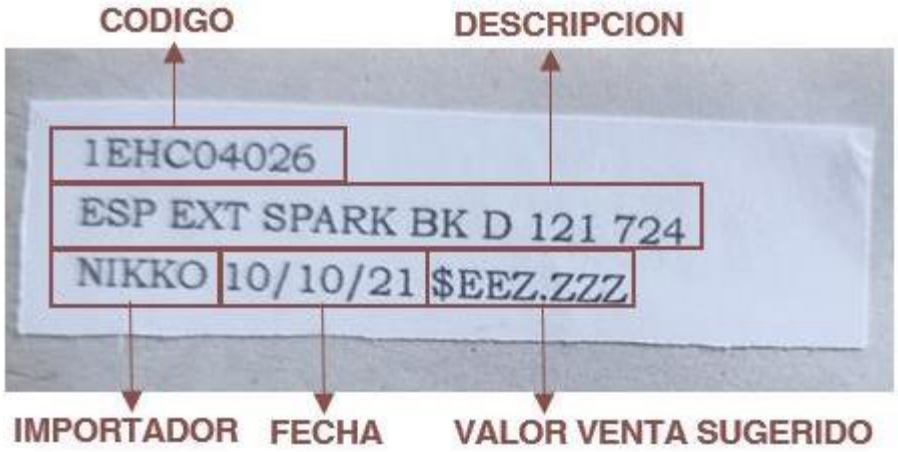
Ilustración 2. Imagen a Datificar (taco).



4.3.2 Marquilla

Como muestra la Ilustración 3, en la marquilla se halla la referencia del producto, el nombre del producto, el distribuidor y la fecha de compra del artículo.

Ilustración 3. Marquilla.



4.4. AUTOMATIZADO DE INVENTARIO

El sistema diseñado automatiza el proceso actualización de inventariado del local comercial (empresa piloto), donde el ingreso y venta de cada producto se actualiza de manera automática y en tiempo real.

Para la base de datos se utilizó *ODOO,* un software de planificación de recursos empresariales (ERP) integrado de código abierto, que incluye, sitio web para

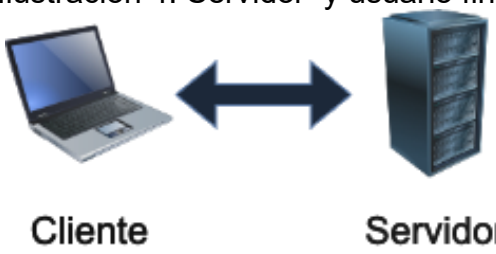
comercio electrónico, facturación, contabilidad, fabricación, gestión de almacenes y proyectos, e inventario entre otros. Cuenta con una versión "comunitaria" de código abierto bajo licencia LGPLv3.

Para la interfaz de usuario se usó *REACT*. La página oficial de React la define como "una biblioteca Javascript de código abierto diseñada para crear interfaces de usuario con el objetivo de facilitar el desarrollo de aplicaciones en una sola página. Es mantenido por Facebook y la comunidad de software libre"[4].

4.5 APLICACIÓN WEB Y SERVIDOR

En este trabajo de grado, se ha desarrollado una aplicación web que sirve como interfaz de usuario final, construida en React. Esta aplicación se conecta a una base de datos, como se muestra en la Ilustración 4, y establece una comunicación con el servidor implementado con ODOO. En dicho servidor, se almacena el inventario para una interacción constante con el usuario. A través de esta aplicación, los usuarios pueden capturar imágenes deseadas y enviar la información al servidor para su posterior procesamiento de imágenes.

Ilustración 4. Servidor y usuario final.

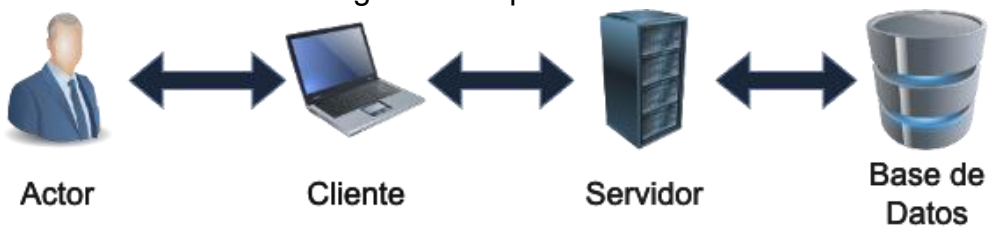


[4] ¿Qué es React?. (s.f). https://es.react.dev/

5. DISEÑO.

En este capítulo, se presenta el diseño del prototipo para el sistema de datificación de texto y gestión empresarial.

Ilustración 2. Diseño del prototipo para el sistema de amplificación de texto y gestión empresarial.



En la estructura mostrada en la Ilustración 5, el Actor interacciona con el sistema mediante la página web (Cliente) la cual hace las operaciones de usuario final. Mediante una interfaz de programación el cliente se comunica con el servidor el cual decide qué órdenes envía a la base de datos para recuperar o actualizar la información del inventario. Con la información final se toma la decisión que observa el actor en el cliente.

5.1 DIAGRAMA DE CASOS DE USO.

Durante la fase inicial de modelado, los casos de uso desempeñan un papel fundamental al ayudar a comprender de manera precisa los requisitos del sistema, permitiendo un enfoque estructurado en el desarrollo del proyecto.

5.1.1 Identificación de actores y casos de uso.

Los actores del sistema son explicados a continuación:

Actor 1: Administrador:

Actor encargado de regir el sistema para establecer y gestionar el actor vendedor y actualizar el inventario.

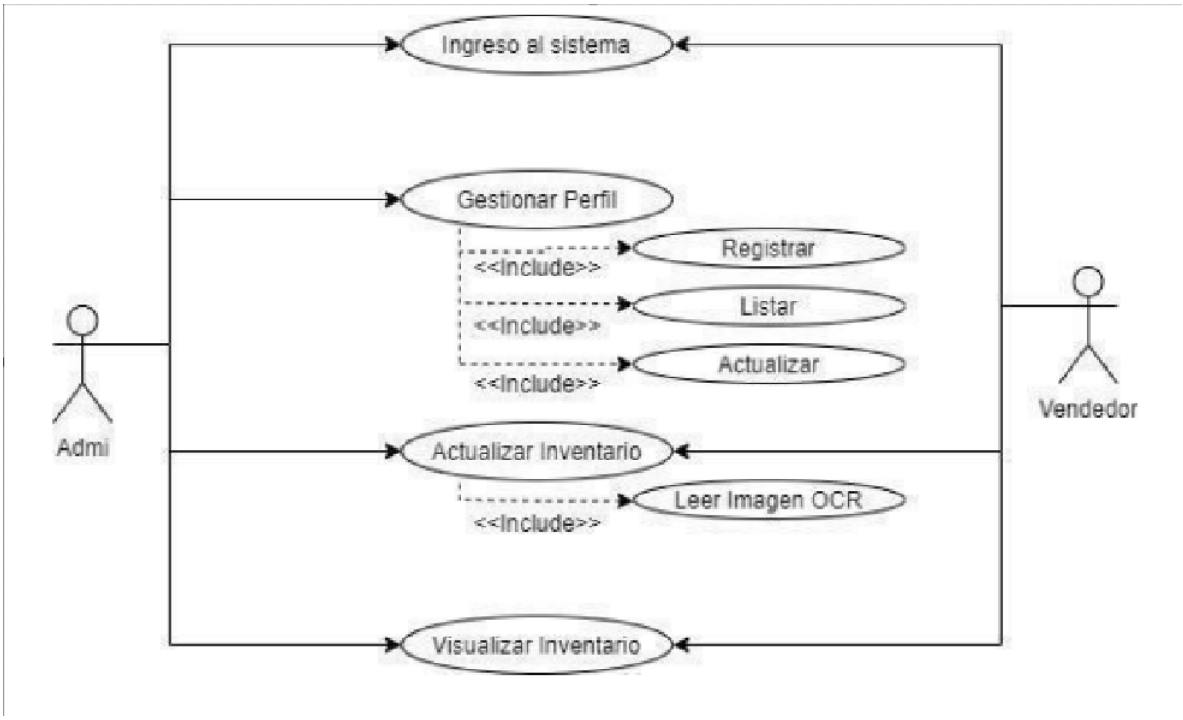
Actor 2: Vendedor:

Actor que es ingresado al sistema mediante el administrador. puede modificar el inventario mediante la venta realizada.

5.1.2 Modelo de casos de uso

A través de la Ilustración 6, se identifican los actores y sus roles en el sistema, lo que permite obtener una lista de los casos de uso de interacción correspondientes.

Ilustración 3. Diagrama de casos de uso.



- Caso de uso 1*: Ingresar vendedor al sistema.* Proceso encargado de registrar al actor vendedor en el sistema. Después de iniciar sesión correctamente el administrador tendrás la opción de agregar o eliminar los perfiles de vendedor, digitalizar tacos y revisar o modificar inventario.

- Caso de uso 2: *Gestionar perfil.* Proceso encargado de gestionar, listar y actualizar el actor vendedor y administrador.

- Caso de uso 3: *Actualizar inventario.* Proceso encargado de actualizar el inventario mediante la realización de una venta. Se lee la marquilla que contiene información que actualiza el inventario.

- Caso de uso 4: *Visualizar inventario*. Proceso encargado de visualizar el inventario para buscar un producto para la realización de una cotización o una venta.

5.1.4 Modelo de diagrama de clases

Inicialmente se encuentra la sección de ingreso al sistema “login”, donde el administrador debe ingresar los datos correspondientes (cédula y contraseña). La sección de perfil está compuesta por el método Listar (), con esta clase se pueden realizar consultas sobre la información de los vendedores existentes en la base de datos; en donde se encuentran los atributos (cédula, nombre, edad, cargo). La sección siguiente es Administrador que está conformada por los métodos Crear () y Actualizar (); aquí se agrega o modifica la información de los vendedores y las variables implementadas son iguales a las presentes en la sección perfil adicionando contraseña. cabe resaltar que el administrador hereda las características de perfil.

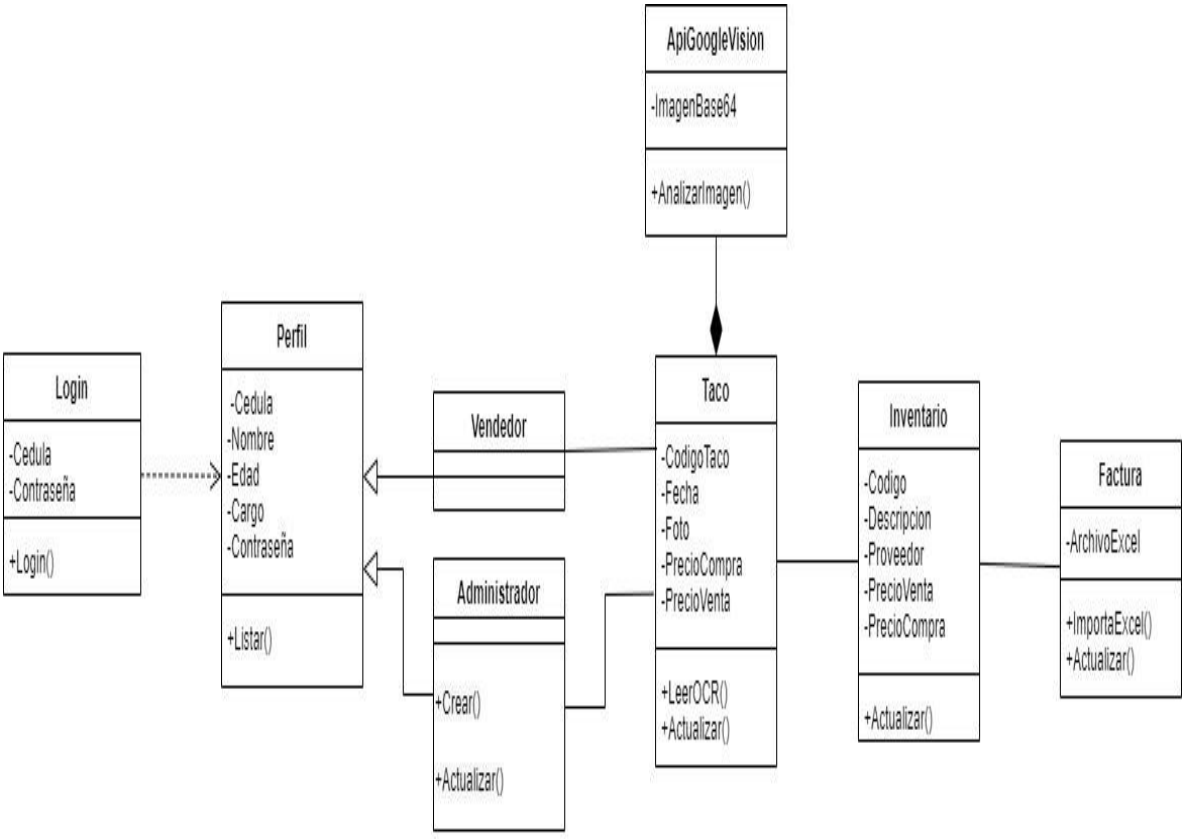
Taco es la sección encargada de la venta, es decir, registra la ventana realizada o la salida de una unidad existente en el inventario. Leer OCR () y Actualizar () son los métodos que pertenecen a esta sección, la primera permite leer el taco de la venta que se realiza y en Actualizar() se actualiza el inventario con las cantidades existentes posteriores a la venta. Las variables utilizadas en esta sección son (codigoTaco, Fecha, Foto, PrecioCompra y PrecioVenta). Es importante tener en cuenta que sección taco se implementa el ApiGoogleVision, en esta sección se envían los servicios de Google y se recibe la información de texto jerarquizada como se explica en el marco teórico.

En la sección inventario se localizan todas las unidades disponibles en el momento, esta se puede visualizar de manera óptima gracias a los filtros incorporados, en ella se utiliza el método Lista () y las variables código, Descripción, Proveedor, PrecioVenta y FechaCompra.

Por último, se presenta la sección Factura, la cual realiza el ingreso de nuevas unidades a la base de datos, cuyo proceso se lleva a cabo mediante la importación de un archivo Excel donde están las nuevas unidades y cantidades a ingresar. Una vez importado se actualiza el inventario existente. Esta sección está conformada por dos métodos Importar Excel (), Actualizar () además de una variable (Archivo Excel).

5.1.5 Elaboración del modelo de diagrama de clases

Ilustración 7. Diagrama de clases.



El diagrama de clases representa un sistema de gestión del local lujos y eléctricos cruz en donde se implementa el taco. El sistema tiene cuatro clases principales como se puede dar a conocer en la Ilustración 7:

- ApiGoogleVision: Es una clase que proporciona acceso a la API de Google Vision.
- Perfil: Representa un perfil de usuario.
- Vendedor: Representa un vendedor de tacos.
- Administrador: Representa un administrador de la tienda.

Las relaciones entre las clases son las siguientes:

- La clase ApiGoogleVision es utilizada por la clase Perfil: La relación entre las clases ApiGoogleVision y Perfil es de dependencia. La clase Perfil necesita la clase ApiGoogleVision para realizar el análisis de imágenes.

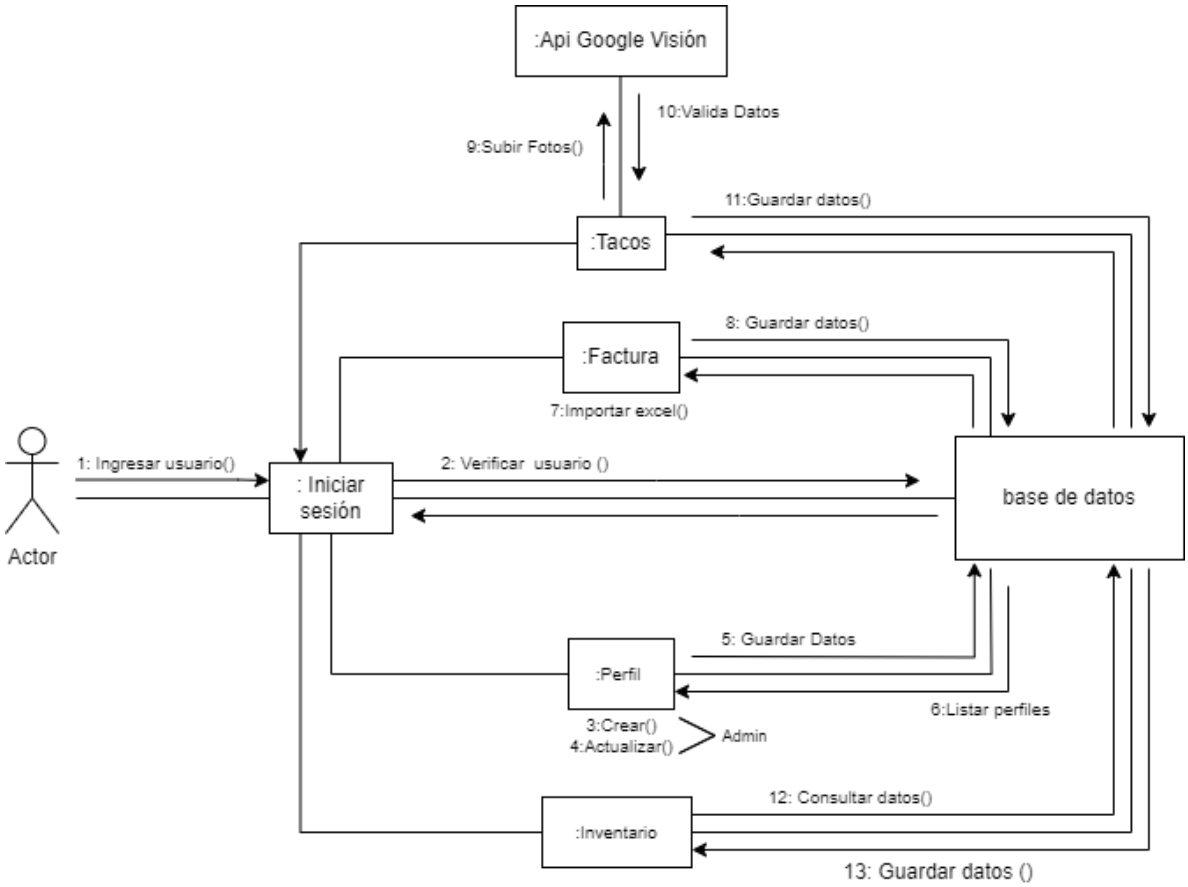
- Un vendedor puede tener varios perfiles: La relación entre las clases Vendedor y Perfil es de asociación. Un vendedor puede tener varios perfiles asociados, pero un perfil solo puede tener un vendedor asociado.

- Un administrador puede tener varios perfiles: La relación entre las clases Administrador y Perfil es de asociación. Un administrador puede tener varios perfiles asociados, pero un perfil solo puede tener un administrador asociado.

5.1.6 Modelo de colaboración

Como se puede observar en la Ilustración 8, el administrador inicia el proceso a través de la sección Login, una vez ingresa los datos correspondientes la información es verificada contra la almacenada en la base de datos; si no se encuentran coincidencias el administrador debe registrarse o actualizar datos, realizado estos procesos se guardan si no ha sido registrado y si la información coincide procederá a las actividades que desee realizar.

Ilustración 8. Diagrama de colaboración.



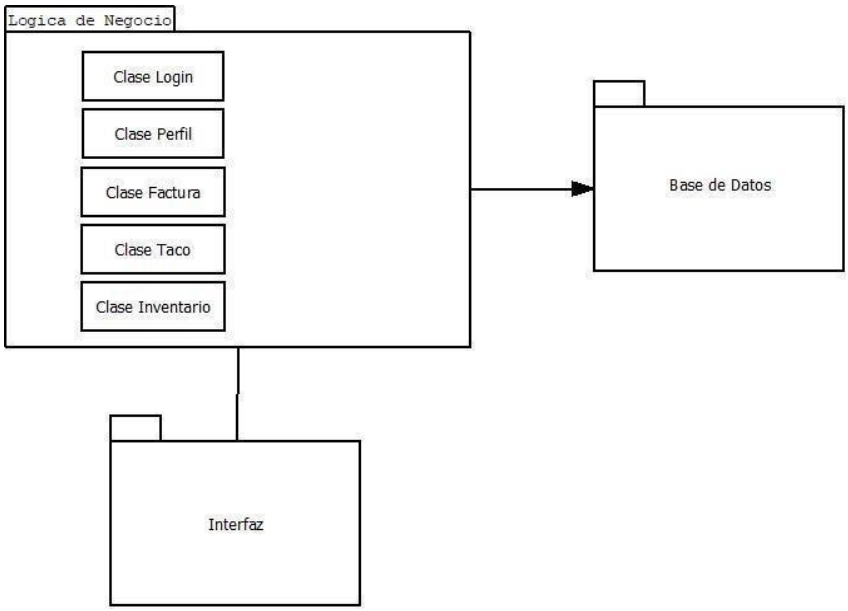
las instancias Factura o Taco, la primera corresponde a el ingreso de nuevos productos y el registro de entrada y salida del almacenamiento en la base de datos, la segunda es la venta de un producto del inventario, aquí se envía una foto a la ApigoogleVision donde se validan los datos existentes en la base de datos, finalmente, se actualiza el inventario ya sea por ingreso de nuevas cantidades o productos o salida de estos.

Perfil e Inventario, ya cuando el administrador ya es aceptado por login procede a acceder a estas dos secciones: En la primera puede crear, editar o eliminar perfiles de vendedores lo cual la información obtenida mediante estos procesos se actualiza en la base de datos. En la segunda se puede visualizar lo que se encuentra en el inventario donde se obtendrá la información pertinente a la cantidad de productos vendidos o almacenados.

5.1.8 Modelo de paquetes

Mediante la Ilustración 9, se puede observar que en el sistema se encuentran tres secciones, interfaz, lógica del negocio y base de datos. La interfaz es la sección donde el usuario interactúa con el sistema. La lógica de negocio en esta compuesta por las clases que permite modificar y actualizar la base de datos. El usuario es capaz de seleccionar el caso de interés, mediante la interfaz realizando ventas, creando perfiles o actualizando inventario, generando una información que es aprovechada por la base de datos

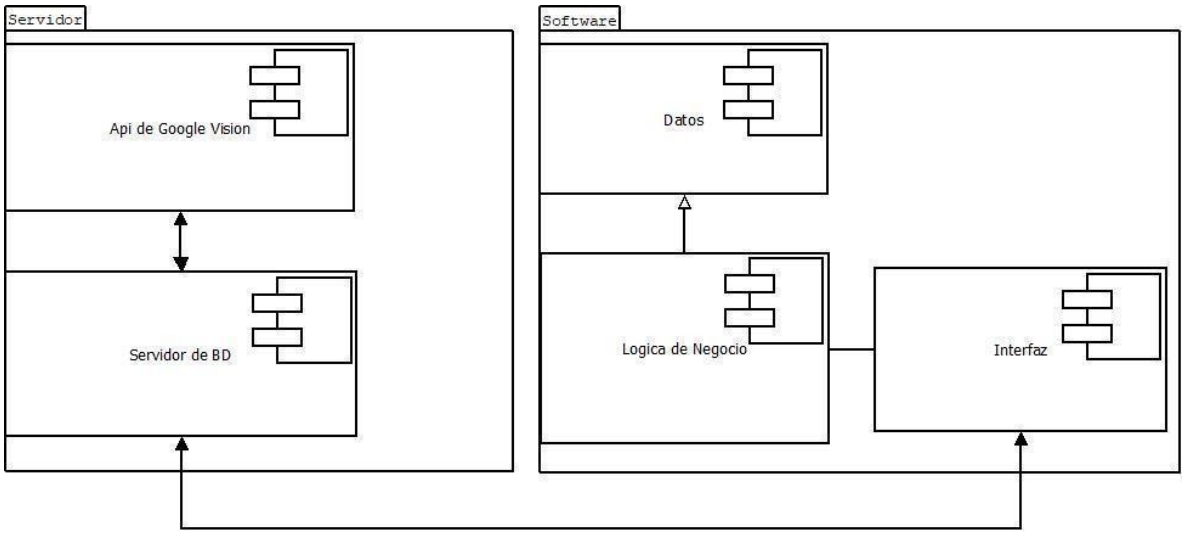
Ilustración 9. Diagrama de paquetes.



5.1.10 Modelo de componentes

Como muestra la Ilustración 10, los componentes se encuentran divididos en dos paquetes software y servidor dependientes uno del otro mediante los componentes interfaz y servidor bd (base de datos).

Ilustración 10. Diagrama de componentes.



El paquete servidor cuenta con dos componentes; api de Google Vision y servidor de base de datos. El primer componente de este paquete es el encargado de detectar texto en la imagen y filtrar el texto, el segundo se encarga de almacenar y organizar la información pertinente a la base de datos.

El paquete software incluye los componentes necesarios para que el proyecto se lleve a cabo mediante el componente interfaz que genera dependencia a la lógica del negocio ya que el usuario mediante la interfaz podrá realizar acciones en el sistema. La componente lógica de negocio posee una relación binaria con base de datos, donde algunas subclases de lógica de negocio tienen relación con el componente base de datos.

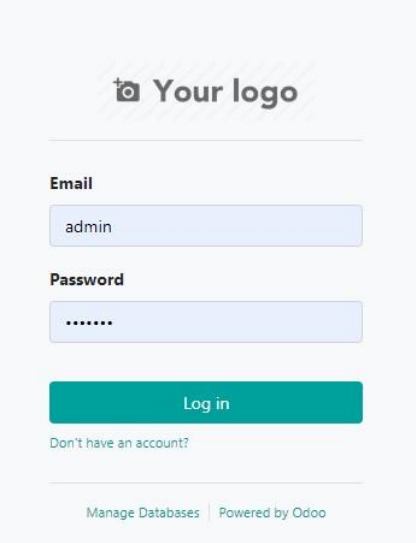
6. IMPLEMENTACIÓN

En el presente capítulo presenta con detalle la implementación del proyecto de grado.

6.1 ACTORES ADMINISTRADOR Y VENDEDOR

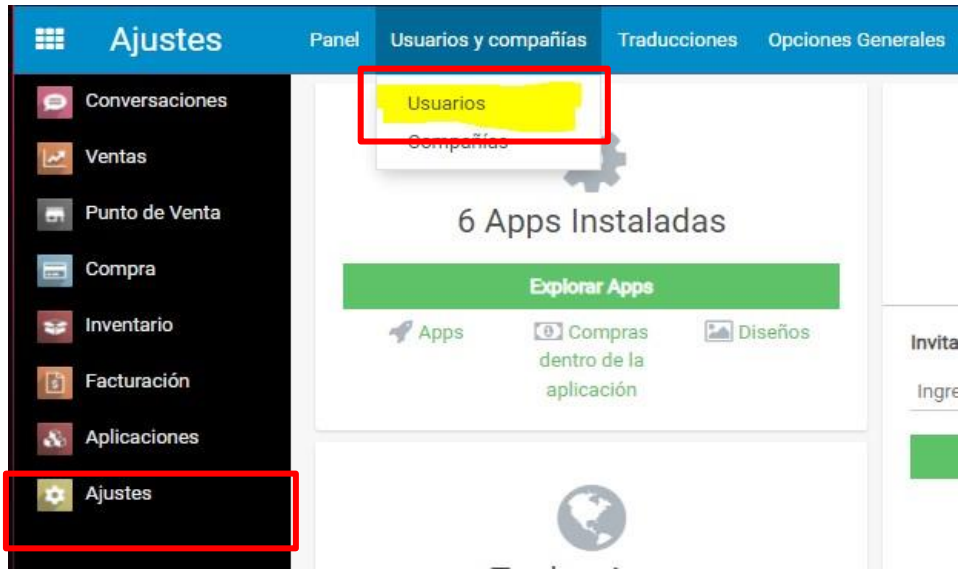
En la implementación de software se accede a los servicios de administrador de la base de datos a través de una Api creada con Odoo quien se encarga del backend. La forma de acceder es por medio de la página de Odoo ya teniendo creada la base de datos, donde se inicia sesión para acceder a las configuraciones del backend. En la Ilustración 11 se observa la interfaz de inicio de sesión que proporciona Odoo.

Ilustración 11. Inicio de sesión del administrador.



El acceder como administrador tendrá la opción de revisar los usuarios que pueden tener permiso como administrador o vendedor, cambiando o agregando casillas como edad, cédula, nombre. De ser necesario algún dato adicional podrá ser creada dicha casilla para alimentar el frontend. Estas configuraciones se pueden realizar como se muestra en la Ilustración 12 en la pestaña “usuarios” que se encuentra en “usuarios y compañías” proveniente de desplegar “ajustes”.

Ilustración 12. Interfaz del administrador.



.

El frontend se alimenta del backend gracias a un enlace que redirige los datos del api que están almacenados en la nube por un servidor que provee Digital Ocean. Se utiliza la biblioteca React de JavaScript para construir la interfaz de usuario. Para crear el acceso a dos datos de usuario se hizo uso de una función flecha con nombre doLogin como se puede ver en la Ilustración 13 que se alimenta de los datos extraídos del api, esto se realizó a través de un método POST dentro de un método global fetch. La función doLogin tiene una estructura correspondiente a los datos a solicitar (usuario, contraseña) donde reconocerá si está accediendo un administrador o un vendedor.

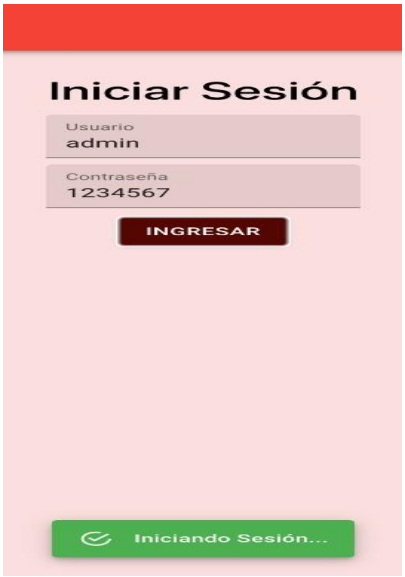
Los servicios que se configuraron son un modelo para saber a qué tabla apunta de la base de datos y un método para buscar, crear o actualizar con opciones de búsqueda avanzados.

Ilustración 13. Código de enlace frontend con el backend.

```
doLogin = () => {

    fetch('http://142.93.62.149:8080/api/login/', {
        method: 'POST',
        headers: {
            Accept: 'application/json',
            'Content-Type': 'application/json',
        },
        body: JSON.stringify(
            {
                "host": "127.0.0.1",
                "port": 9000,
                "database": "lujosec",
                "username": this.state.cedula,
                "password":  this.state.pass,
                "model": "res.users",
                "method": "search_read",
                "options": {
                    "fields":  [],
                    "domain": []
                }
```

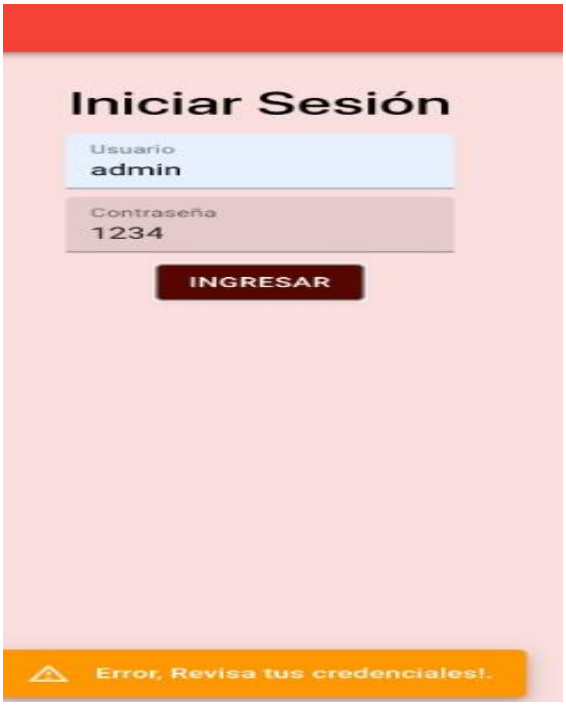
Cuando se envía la petición de la función explicada en el párrafo anterior al servidor el responderá con un mensaje interno “200” correspondiente a un inicio de sesión positivo, como se muestra en la Ilustración 14, permitirá visualizar en pantalla que está iniciando sesión con el recuadro verde indicando el mensaje positivo.

Ilustración 14. Inicio de sesión positivo.



En la Ilustración 15 se obtiene un mensaje de error que permite visualizar un “warning” informando que es necesario revisar las credenciales.

Ilustración 15. Inicio de sesión negativo.



El código que se utiliza en este caso corresponde a la respuesta de una petición creando un objeto “response”. En la Ilustración 16 se puede observar las cualidades de este objeto el cual está estructurado con un “if” que responde un mensaje “200” cuando se cumple correctamente la petición, y un “else” que responde un mensaje el “warning”.

Ilustración 16. Código de objetivo response.

```
.then(response => response.json())
.then(responseJson => {

    if (responseJson.message.includes('200')) {

        localStorage.setItem('user', JSON.stringify(responseJson))

        this.setState({ open: true, message: 'Iniciando Sesión...', severity: 'success' })
        setTimeout(() => {
            this.setState({ open: false })
            this.props.history.push('/perfiles')
        }, 1500);
    } else {
        this.setState({ open: true, message: 'Error,  Revisa tus credenciales!.', severity: 'warning' })

        setTimeout(() => {
            this.setState({ open: false })
        }, 3000);
    }
})
.catch(error => {
    console.log(error);
});
```

6.2. IMPLEMENTACIÓN DE LOS CASOS DE USO:

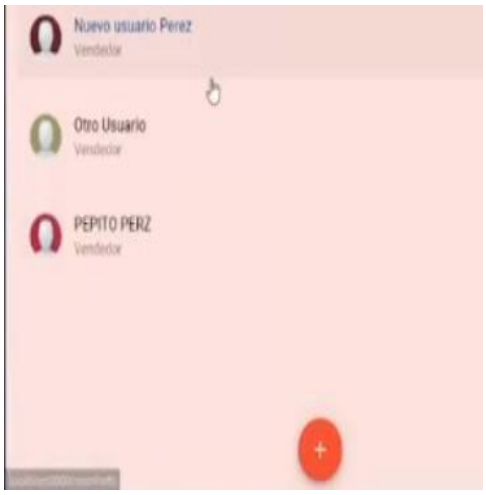
6.2.1 Caso de uso 1: ingresar vendedor al sistema

Proceso encargado de registrar al actor vendedor en el sistema. Después de iniciar sesión correctamente el administrador tendrá las opciones vistas en la Ilustración 17 que en este caso corresponden a Administrar perfiles, digitalizar tacos y revisar o modificar inventario.

Ilustración 17. Opciones de administrador.

Ingresando en el apartado “Administrar perfiles” será redireccionado a donde se encuentran los perfiles ya creados como indica la Ilustración 18, donde puede ingresar a cada uno de ellos para verificar sus datos, eliminar o crear uno nuevo de ser necesario. Para crear el perfil nuevo es necesario cliquear “+” visible en la parte baja de la Ilustración que corresponde a un botón.

Ilustración 18. Ingresar perfiles.



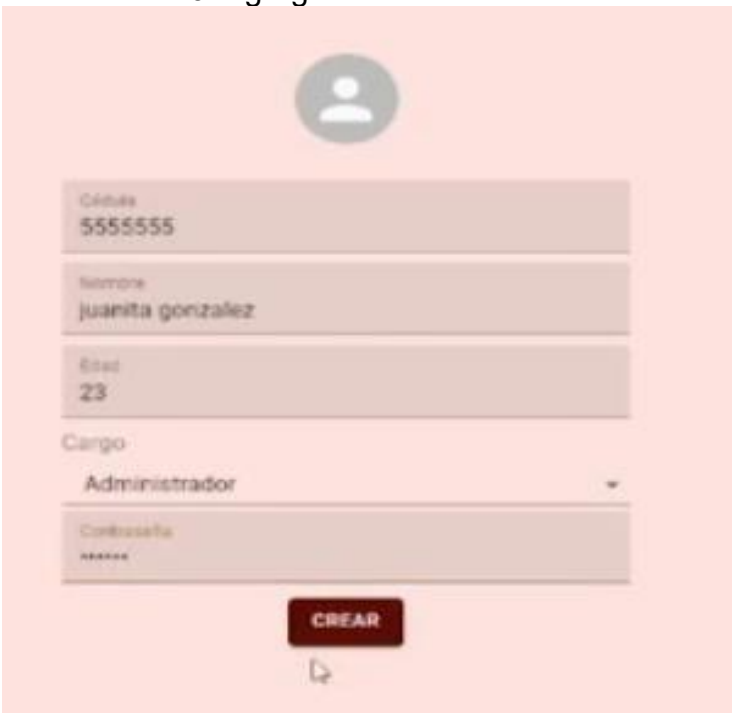
Como se comentó, en la Ilustración 18 se encuentra el botón “agregar” que es creado dentro de una función render. En la Ilustración 19 se ve el código correspondiente a este botón que se encarga de realizar las redirecciones, en este caso para agregar los datos del nuevo perfil que corresponde a una clase llamada “crearPerfil”.

Ilustración 19. Código agregar nuevo perfil.

```
    {/*BOTON DE AGREGAR   */}
    <div className='col-12 d-flex fixed-b'>
        <Fab color="primary" aria-label="add" className=" mx-auto" component={Link} to={{
            pathname: '/crearPerfil',
            aboutProps: {
                data: null
            }
        }}>
            <AddIcon />
        </Fab>
    </div>
</div>
```

Seguidamente se ingresaron los datos de nuevo vendedor o administrador y se cliquea en “CREAR”. Estos datos se pueden observar en la Ilustración 20 donde se realiza un ejemplo de cómo se crea un nuevo perfil.

Ilustración 20. Agregar datos del nuevo vendedor.



Para realizar correctamente la creación de un perfil nuevo se agregó una clase encargada de capturar los datos de las casillas correspondientes para almacenarlas en variables. Esta clase es la mencionada antes “crearPerfil” y sus atributos y métodos corresponden a la Ilustración 21, las variables almacenadas tienen diferentes propiedades y deben ser llenadas en totalidad.

Ilustración 21. código de la clase para capturar datos.

```
class CrearPerfil extends React.Component {

    static propTypes = {
        match: PropTypes.object.isRequired,
        location: PropTypes.object.isRequired,
        history: PropTypes.object.isRequired
    };

    constructor(props) {
        super(props);
        // Initial state is defined
        console.log('props', props.location.aboutProps);
        this.dataProp = props.location.aboutProps ? props.location.aboutProps.data : null



        this.state = {
            id: '',
            cedula: '',
            nombre: '',
            edad: '',
            cargo: '',
            password: '',

            open: false,
            message: '',
            severity: '',
            props: this.dataProp,
            showPassword: false,

        };
        this._isMounted = false;

        this.createUsers = this.createUsers.bind(this)
        this.updateUser = this.updateUser.bind(this)
```

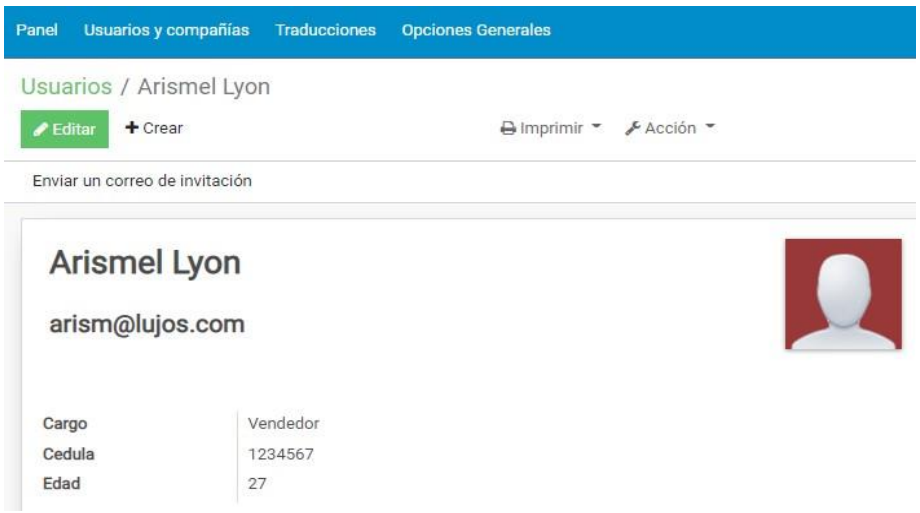
Estas variables corresponden a las mismas que se tienen en la base de datos para no tener errores. Son enviadas por medio de la conexión con el api como se aprecia en la Ilustración 22. Posee un método “POST” para actualizar la base de datos con la información del nuevo perfil. En esta ilustración se puede ver la estructura del cuerpo del mensaje a enviar que debe cumplir con las características de formato json para ser recibida y actualizada sin errores.

Ilustración 22. Código de conexión de las variables con el api.

```
fetch('http://142.93.62.149:8080/api/call_kw/', {
    method: 'POST',
    headers: {
        Accept: 'application/json',
        'Content-Type': 'application/json',
    },
    body: JSON.stringify(
        {
            "host": "127.0.0.1",
            "port": 9000,
            "database": "lujosec",
            "username": "admin",
            "password": "1234567",
            "model": "res.users",
            "method": "create",
            "options":
            {
                "login": this.state.nombre.slice(0, 5).toLocaleLowerCase() + '@lujos.com',
                "name": this.state.nombre,
                "cargo": this.state.cargo,
                "edad": this.state.edad,
                "cedula": this.state.cedula,
                "password": this.state.password
            }
```

Estas acciones pueden ser corroboradas como muestra la Ilustración 23, entrando directamente a la base de datos y revisando los perfiles creados donde también pueden agregarse manualmente con la opción “+Crear”.

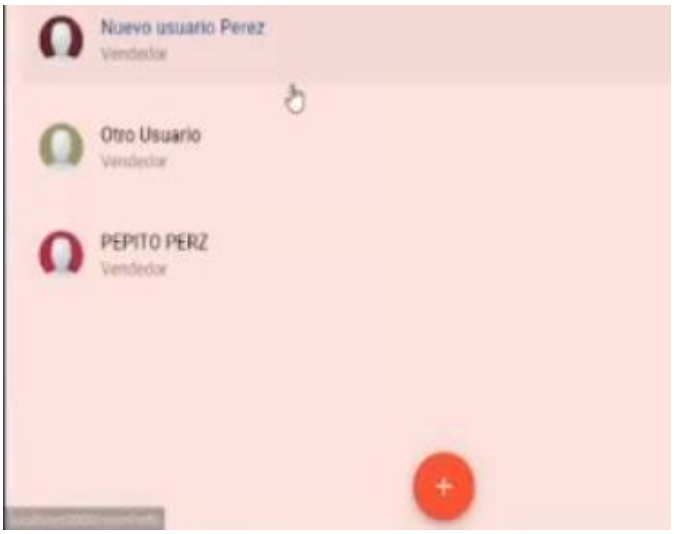
Ilustración 23. Verificación del perfil creado en la base de datos.



6.2.2 Caso de uso 2: Gestionar perfil

Para este caso igualmente es necesario iniciar sesión como administrador e ingresar en administrar perfiles donde podrá visualizar los perfiles como aparecen en la Ilustración 24 que corresponden a los perfiles existentes en la base de datos, al clicar en alguno de ellos se puede acceder a su información para ser modificada o visualizados.

Ilustración 24. Lista de perfiles en la base de datos.



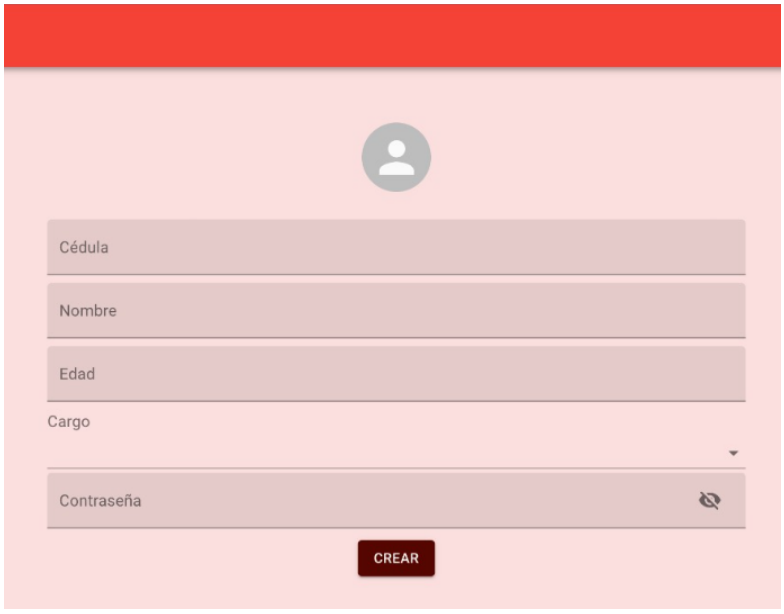
Los perfiles son listados para ser visualizados mediante una función llamada “getUser” las propiedades y codificación de esta función corresponden a la Ilustración 25 que interactúa directamente con la base de datos para traer y organizar cada uno de los perfiles mediante un método “POST” en formato json, las propiedades de este método son únicamente para visualizar y buscar, no es posible modificar directamente algún perfil listado.

Ilustración 25. Código función getUser, para la organización de los perfiles.

```
getUsers() {

    fetch('http://142.93.62.149:8080/api/call_kw/', {
        method: 'POST',
        headers: {
            Accept: 'application/json',
            'Content-Type': 'application/json',
        },
        body: JSON.stringify({
            host: '127.0.0.1',
            port: 9000,
            database: 'lujosec',
            username: 'admin',
            password: '1234567',
            model: 'res.users',
            method: 'search_read',
            options: {
                "fields": ["id", "login", "name", 'image_small', 'cargo', 'cedula', 'edad' , 'password'],
                "domain": []
            },
```

Para modificar información una vez listados los usuarios, se ingresa al perfil que se actualizará. En la Ilustración 26 se tiene un ejemplo de los campos disponibles para modificar en cada perfil, simplemente se cambian los datos que se quieren actualizar o modificar en caso de que estén erróneos y se clica el botón “ACTUALIZAR”.

Ilustración 26. Actualización de datos



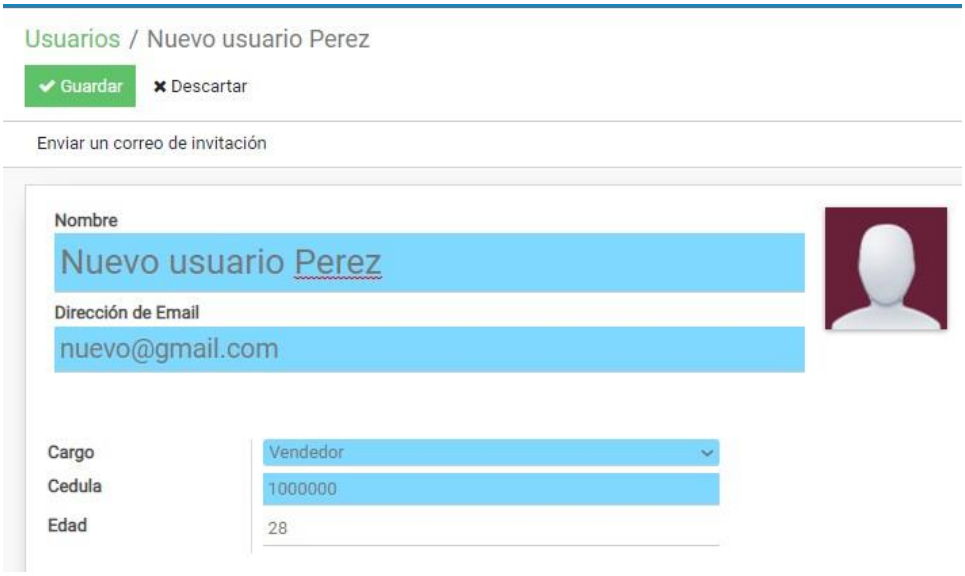
Esta opción almacena en variables como muestra la Ilustración 27, al ser declarada con la palabra reservada “let” son bloqueadas para ser usadas solo en la función updateUser. Como se aprecia es requerido nuevamente un método “POST” que realiza la petición de actualizar la base de datos cuando es autenticado el inicio de sesión como administrador.

Ilustración 27. Código solicitud backend.

```
updateUser = () => {

    let data = {
        "id": this.state.id ? this.state.id : (this.dataProp != null ? this.dataProp.id : ''),
        "name": this.state.nombre ? this.state.nombre : (this.dataProp != null ? this.dataProp.name : ''),
        "cargo": this.state.cargo ? this.state.cargo : (this.dataProp != null ? this.dataProp.cargo : ''),
        "edad": this.state.edad ? this.state.edad : (this.dataProp != null ? this.dataProp.edad : ''),
        "cedula": this.state.cedula ? this.state.cedula : (this.dataProp != null ? this.dataProp.cedula : ''),
        "password": this.state.password ? this.state.password : (this.dataProp != null ? this.dataProp.password : '')
    }
    if (data.cargo.includes('admin')) {
        delete data.password
    }

    fetch('http://142.93.62.149:8080/api/call_kw/', {
        method: 'POST',
        headers: {
            Accept: 'application/json',
            'Content-Type': 'application/json',
        },
        body: JSON.stringify(
            {
                "host": "127.0.0.1",
                "port": 9000,
                "database": "lujosec",
                "username": "admin",
                "password": "1234567",
                "model": "res.users",
                "method": "write",
                "options":
                {
                    "fields": data,
                    "ids": [data.id]
                }
            }
```

Esta modificación también puede ser visualizada en la Ilustración 28, que muestra la manera para realizar directamente las modificaciones en Odoo, entrando en los perfiles que están en la base de datos y clicando “Editar” y posteriormente “Guardar” después de terminar las correcciones.

Ilustración 28. Modificación directa de perfiles en Odoo.



6.2.3 Caso de uso 3: actualizar inventario

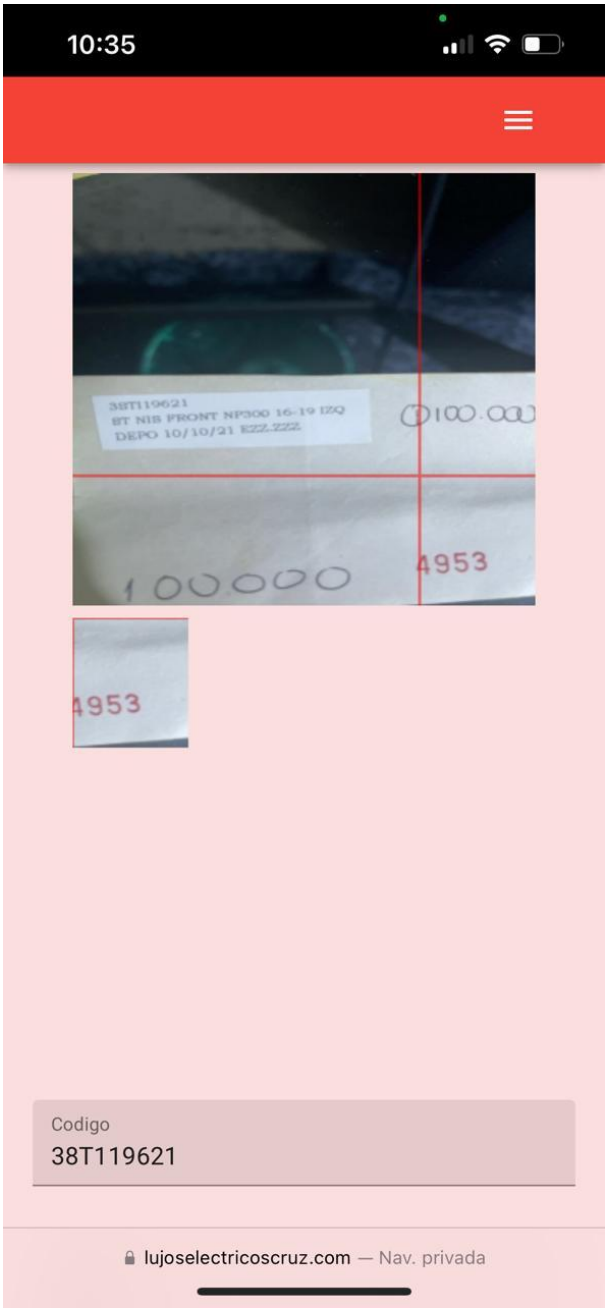
Proceso encargado de actualizar el inventario mediante la realización de una venta. Se lee la marquilla que contiene información que actualiza el inventario.

En la lectura del taco de venta, se digitaliza la imagen de cada venta realizada por administrador o vendedor. Teniendo en cuenta que cada taco tiene un patrón, es necesario dividir la imagen en 4 secciones y hacer uso de tres de estas que contienen la información, para procesar cada una de ellas se hizo uso de una herramienta de Google que permite extraer texto en imágenes de forma muy precisa. Para realizar ese proceso se accede a “Digitalizar Tacos”, esta opción se despliega como se mencionó antes después de iniciar sesión, en la Ilustración 29 se puede apreciar la opción donde se debe ingresar.

Ilustración 29. Acceso a la opción 'Digitalizar Tacos.

Ahora se abre la pantalla de captura con acceso a la cámara, donde se toma la foto del taco como se muestra en la Ilustración 30, aparecen los datos correspondientes para descargar de la base de datos. Cabe resaltar que cada proceso de autenticación o validación en la base de datos tiene que ser exacto, de no existir el número de taco, referencia de productos o si la digitalización de la imagen arroja un número erróneo se verá un mensaje de error “No se puede actualizar”, entonces se podrá comparar y corregir manualmente si es el caso. En la imagen se puede visualizar lo mencionado anteriormente, la imagen es dividida en fracciones para analizarla como imágenes diferentes y conformar un solo tramo de información. Esto será explicado con claridad en este documento.

Ilustración 30. Captura de imagen con acceso a la cámara.



Para lograr la lectura correcta del taco de venta se creó una clase llamada “Taco”, la estructura de esta clase se puede apreciar en la Ilustración 31 que tiene un constructor donde se crearon las variables a utilizar correspondientes a la base de datos, las variables serán utilizadas dentro de la clase en los diferentes métodos que fueron necesarios para la digitalización de la imagen.

Ilustración 31. Constructor de variables para la base de datos en React.

```
class Taco extends React.Component {


    constructor(props) {
        super(props);
        // Initial state is defined
        this.state = {
            img: '',
            codigo: '',
            proveedor: '',
            fecha: '',
            precioVenta: '',
            precioCompra: '',
            codigoRojo: '',
            open: false,
            message: '',
            severety: '',

        };
```

Dentro de la clase principal fue necesario realizar las operaciones pertinentes para preparar la imagen y para decodificar la información del taco de venta.

Después se tiene capturada la imagen para tener las tres divisiones que se utilizaron, como todos los tacos de venta tienen las mismas proporciones se creó un código para demarcar las tres áreas a transformar en texto. Mediante el método “updateCanvas” presentado en la Ilustración 32, se crea un boceto de los límites a demarcar para crear la nueva imagen.

Ilustración 32. Creación de boceto para demarcar áreas de transformación imagen.

```
updateCanvas(dataUri) {

    const canvas1Ele = this.canvas1Ref.current;
    this.ctx1 = canvas1Ele.getContext("2d");
    this.ctx1.beginPath();

    const image = new Image();
    //parte 1
    let hy1 = canvas1Ele.height - canvas1Ele.height * 30 / 100
    let wx1 = canvas1Ele.width - canvas1Ele.width * 25 / 100
    let offsetX1 = 0
    let offsetY1 = 0
    //parte2
    let hy2 = canvas1Ele.height - canvas1Ele.height * 30 / 100
    let wx2 = canvas1Ele.width - canvas1Ele.width * 25 / 100
    let offsetX2 = 0
    let offsetY2 = canvas1Ele.height - canvas1Ele.height * 30 / 100

    //parte3
    let hy3 = canvas1Ele.height - canvas1Ele.height * 30 / 100
    let wx3 = canvas1Ele.width - canvas1Ele.width * 25 / 100
    let offsetX3 = canvas1Ele.width - canvas1Ele.width * 25 / 100
    let offsetY3 = canvas1Ele.height - canvas1Ele.height * 30 / 100
```

Una vez demarcada la imagen esta es trazada con una línea horizontal y vertical, creada con la función flecha “image. onload” presentada en la Ilustración 33 para dividirla y pintarla sobre la imagen original según las coordenadas demarcadas anteriormente, estas son visibles cuando se toma la foto y queda capturada en pantalla, la Ilustración 29 muestra un ejemplo del resultado visible por el usuario.

Ilustración 33. División de imagen mediante línea horizontal y vertical.

```
image.onload = () => {
    this.ctx1.drawImage(image, 0, 0, canvas1Ele.width, canvas1Ele.height)
    this.ctx1.beginPath();

    //LINEA HORIZONTAL
    //          |       x , y
    this.ctx1.moveTo(0, hy1)
    this.ctx1.lineTo(canvas1Ele.width, hy1);

    // LINEA VERTICAL
    this.ctx1.moveTo(wx1, 0)
    this.ctx1.lineTo(wx1, canvas1Ele.height);


    this.ctx1.strokeStyle = 'red'
    this.ctx1.stroke();
}
```

Fue necesario tener la imagen en base 64 como el requisito del api de visión “googleapis” que recibe Google para interpretar la imagen. Para tener la imagen en base 64 se creó una función llamada “getArea” y se puede ver en la Ilustración 34, esta función recibe los canvas, y coordenadas para realizar la conversión de la imagen y envíasela al api de Google mediante una función interna llamada “getImageData”, donde será interpretada y digitalizada.

Ilustración 34. Conversión de imagen a base 64

```
getArea(canvas, offsetX, offsetY, width, height) {


    const canvas2Ele = this.canvas2Ref.current;
    this.ctx2 = canvas2Ele.getContext("2d");
    this.ctx2.beginPath();

    const image = new Image();

    image.onload = () => {
        //this.ctx2.drawImage(image, 10, 10, width, height)
        var imgData = this.ctx1.getImageData(offsetX, offsetY, width, height);
        this.ctx2.putImageData(imgData, 0, 0);
    }


    let read = new Promise((resolve, reject) => {

        setTimeout(() => {
            //console.log(canvas.toDataURL());
            image.src = canvas.toDataURL()
            setTimeout(() => {
                //console.log(canvas2Ele.toDataURL());
                (async () => {
                    let res = await this.doSendGoogleApi(canvas2Ele.toDataURL())
                    console.log(res)
                    if (res.data.responses[0]) {
                        resolve(res.data.responses[0])
                    }
                })()

            }, 500);

        }, 500);

    })
```

Ahora la Ilustración 35 corresponde a la función mencionada anteriormente “getImageData” a su vez representa una solicitud de respuesta conformada por el objeto creado con las variables de la imagen ahora en formato “PNG”, la envía con un tipo de petición “POST” al url de “googleapis”.

Ilustración 35. Función para enviar imagen a API de Google Fuente: Captura tomada por los autores.

```
async doSendGoogleApi(dataUri) {
    //console.log(dataUri);
    var data = {
        requests: [
            {
                image: {
                    content: dataUri.includes('png') ? dataUri.slice(22) : dataUri.slice(23),
                },
                features: [{
                    type: "TEXT_DETECTION",
                    maxResults: 5
                }]
            }
        ]
    }
    return await axios({
        method: 'post',
        url: 'https://vision.googleapis.com/v1/images:annotate?key=AIzaSyAqc1gUL7iNXSwdBWGPJN
        data,
        config: { headers: { 'Content-Type': 'application/json' } }
    })

}
```

Teniendo ya el área demarcada y las funciones que se aplicaran, se implementaron en cada parte del taco para dividir la información y tener un mensaje de error de cada tramo al no ser interpretado.

En la Ilustración 36 se tiene la parte 1 y 2 del taco, en cada una de ellas se crea un objeto para dividir la imagen con la información de cada tramo, se aplicó la función “getArea” para verificar si tiene información legible o no.

Ilustración 36. División de información en tramos y verificación de legibilidad.

```
part1.then((res1 => {
    let part2 = new Promise((resolve, reject) => {
        this.getArea(canvas1Ele, offsetX2, offsetY2, wx2, hy2).then((res) => {
            //console.log(res.fullTextAnnotation.text);
            if (res.fullTextAnnotation) {
                resolve(res.fullTextAnnotation.text)
            } else {
                resolve("texto no encontrado")
            }
        })

    });

    part2.then(res2 => {
        let part3 = new Promise((resolve, reject) => {
            this.getArea(canvas1Ele, offsetX3, offsetY3, wx3, hy3).then((res) => {
                // console.log(res.fullTextAnnotation.text);

                if (res.fullTextAnnotation) {
                    resolve(res.fullTextAnnotation.text)
                } else {
                    resolve("texto no encontrado")
                }
            })

        });
```

Se tiene una tercera que contiene más información correspondiente a código especial, precio de compra, proveedor, código de producto y fecha de adquisición, en la Ilustración 37 se tiene esta tercera parte, tiene un tratamiento más profundo para la adquisición de estos datos, cada dato obtenido de la imagen es almacenada en las variables locales correspondientes a cada parte de la imagen.

Ilustración 37. Procesamiento de datos adicionales en la tercera parte del taco.

```
part3.then(res3 => {
    console.log(res1.text);
    let codigoProducto = res1 ? this.splitLines(res1.text)[0] : ''

    let x = res1 ? this.splitLines(res1.text)[2] : ''


    let proveedor = x ? this.splitSpaces(x)[0] : ''
    let fechaAdquisicion = x ? this.splitSpaces(x)[1] : ''
    let precioVenta = x ? this.splitSpaces(x)[2] : ''
    console.log(x.length ? this.splitSpaces(x) : '');
    let codigoRojo = res3.length ? this.splitLines(res3)[0] : ''
    let precio = res2.length ? this.splitLines(res2)[0] : ''
    precio = precio.replace('-', '.')

    console.log('codigoRojo', codigoRojo);
    console.log('precioCompra', precio);

    console.log('proveedor', proveedor);
    console.log('codigoProducto', codigoProducto);
    console.log('fechaAdquisicion', fechaAdquisicion);
    console.log(this.decrypPrecio(precioVenta));

    this.setState({
        codigo: codigoProducto,
        proveedor: proveedor,
        fecha: fechaAdquisicion,
        precioVenta: this.decrypPrecio(precioVenta),
        precioCompra: precio,
        codigoRojo: codigoRojo,
    });
```

La función encargada de enviar los datos al api de Google es llamada “doSendGoogleApi”, en la Ilustración 38 se puede apreciar la estructura de esta función, que consta de una variable que envía una petición con la imagen mediante un método con protocolo “POST” retornando como respuesta los datos requeridos.

Ilustración 38. Estructura de la función 'doSendGoogleApi' para enviar datos al API de Google.

```
async doSendGoogleApi(dataUri) {
    //console.log(dataUri);
    var data = {
        requests: [
            {
                image: {
                    content: dataUri.includes('png') ? dataUri.slice(22) : dataUri.slice(23),
                },
                features: [{
                    type: "TEXT_DETECTION",
                    maxResults: 5
                }]
            }
        ]
    }
    return await axios({
        method: 'post',
        url: 'https://vision.googleapis.com/v1/images:annotate?key=AIzaSyAqc1gUL7iNXSwdBWGPJNz
        data,
        config: { headers: { 'Content-Type': 'application/json' } }
    })
}
```

La respuesta que es recibida del api de Google necesita recibir un mapeo para obtener el texto requerido antes de ser visto en pantalla o enviado a la base de datos, este mapeo fue realizado en la Ilustración 39 que corresponde a los pasos de código para la separación de caracteres, principalmente la tercera parte que contiene más información, cada campo de datos de esta parte es separada con un carácter especial “/” y contiene un código secreto para los clientes, pero debe ser visualizado y descargado de inventario codificado pues así se encuentra en la base de datos.

El método “decrypPrecio” de la Ilustración 39 se encarga para desencriptar el código secreto, se mapea el resultado del api separando cada parte del texto para tener la información de cada campo, luego se visualiza en pantalla para corroborar los datos del taco por el vendedor o administrador quienes podrán enviar el requerimiento a la base de datos.

Ilustración39. Pasos para separar caracteres y obtener información de la tercera parte del taco.

```
splitLines(t) {
    if (t) {
        return t.split(/\r\n|\r|\n/);
    }
}
splitSpaces(t) {
    if (t) {
        return t.split(' ');
    }
}
decrypPrecio(v) {
    console.log(v);
    if (v) {
        v.replace('0', 'O')
        v.replace('00', 'OO')
        // E R A S M O C L U Z
        // 1 2 3 4 5 6 7 8 9 0
        let r = v.replace(/S/g, '$').replace(/E/g, '1').replace(/R/g, '2').replace(/A/g, '3')
        .replace(/S/g, '4').replace(/M/g, '5').replace(/O/g, '6').replace(/C/g, '7')
        .replace(/L/g, '8').replace(/U/g, '9').replace(/Z/g, '0')
        return r;
    }
}
```

Cuando ya está verificada la información a enviar se procede a utilizar la función “sendInfoTaco” vista en la Ilustración 40, esta función se encarga de realizar la comunicación con la base de datos e inventario para descargar el producto referente a la información del taco de venta. Si la información es correcta y corresponde a un producto en la base de datos esta se enviará sin problema, si algún dato es incorrecto no podrá ser descargado y tendrá un mensaje de error para verificar la información. Se tiene un mensaje de error si algún campo está vacío, pues es necesario comparar todos los campos correspondientes a enviar.

Ilustración 40. Función para enviar información a la base de datos e inventario autores.

```
sendInfoTaco = () => {
    const { codigo, proveedor, fecha, precioVenta, precioCompra, codigoRojo } = this.state;
    if (codigo &&
        proveedor &&
        fecha &&
        precioVenta &&
        precioCompra &&
        codigoRojo) {
        //ajax1234567
        console.log('se puede enviar');

        fetch('http://142.93.62.149:8080/api/call_kw/', {
            method: 'POST',
            headers: {
                Accept: 'application/json',
                'Content-Type': 'application/json',
            },
            body: JSON.stringify(
                {
                    "host": "127.0.0.1",
                    "port": 9000,
                    "database": "lujosec",
                    "username": "admin",
                    "password": "1234567",
                    "model": "sale_taco",
                    "method": "create",
                    "options":
                    {
                        "codigo": /*codigo*/ '2LDT15831',
                        "proveedor": proveedor,
                        //  "fecha": new Date().getFullYear() + '/' + (new Date().getMonth()+1) + '/'+ new Date().getDate(),
                        "fecha": fecha,
                        "precio_compra": precioCompra.replace(/\D/gm, ""), //dejar solo numeros
                        "precio_venta": precioVenta.replace(/\D/gm, ""),  //dejar solo numeros
                        "codigo_taco": codigoRojo
                    }
```

6.2.4 Caso de uso 4: analizar inventario

Proceso encargado de visualizar el inventario para buscar un producto para la realización de una cotización o una venta. También es posible ingresar productos en la base de datos a partir de un archivo de Excel.

El apartado inventario se puede acceder como se resalta en la Ilustración 41 para visualizar y buscar productos que se encuentran actualmente activos o en venta.

Ilustración 41. Acceso al apartado de inventario.

En la ventana que se accede se listaran los productos como se ve en la Ilustración 42, permitiendo visualizar su código, nombre, cantidad en stock, fecha de compra y precio. Tiene en cabecera un espacio de búsqueda donde se puede realizar búsqueda personalizada de cada variable.

Ilustración 42. Vista de la ventana de inventario con lista de productos y espacio de búsqueda.

Esto se construye creando una clase llamada inventario esta clase tiene la estructura vista en la Ilustración 43, esta tendrá los métodos necesarios para llamar el inventario del api y realizar la búsqueda mediante un constructor para inicializar un estado local asignando un objeto al “this. state”.

Ilustración 43. Estructura de la clase Inventario.

```
class Inventario extends React.Component {



  constructor(props) {
      super(props);
      // Initial state is defined
      this.state = {
          isLoaded: false,
          buscando: false,
          error: false,
          allProductos: [],
          productos: [],
      };

  }
```

La función que conecta la base de datos para extraer y visualizar la información en inventario es vista en la Ilustración 44 y llamada “getInventario”. Esta función tiene en su estructura principal un fetch que acciona un método “POST”, también arroja un mensaje de error mediante unos métodos “response y catch” cuando no es posible la conexión con la base de datos. El método json () de la interfaz “response” toma el flujo de “response” y lo lee hasta completarlo. Devuelve una promesa que se resuelve con el resultado de analizar el cuerpo del texto como JSON, a pesar de que el método se llama json (), el resultado no es JSON sino que es el resultado de tomar JSON como entrada y analizarlo para producir un objeto JavaScript.

Ilustración 44. Estructura de la función getInventario.

```
getInventario() {
    fetch('http://142.93.62.149:8080/api/call_kw/', {
        method: 'POST',
        headers: {
            Accept: 'application/json',
            'Content-Type': 'application/json',
        },
        body: JSON.stringify({
            "host": "127.0.0.1",
            "port": 9000,
            "database": "lujosec",
            "username": "admin",
            "password": "1234567",
            "model": "product.product",
            "method": "search_read",
            "options": {
                "fields": ["id", "default_code", "name",
                           "standard_price", "qty_available",
                           "image_small", "write_date"],
                "domain": []
            }
        }),
    })
        .then(response => response.json())
        .then(responseJson => {


            this.setState({
                isLoaded: true,
                allProductos: responseJson.response,
                productos: responseJson.response
            });
        })
        .catch(error => {
            console.log(error);
            this.setState({
                error: true,
                isLoaded: true,
```

Para mostrar en pantalla el inventario se usó la función correspondiente a la Ilustración 45, llamada “componentDidMount”. Esta encapsula la función “getInventario” en un método “setTimeout” que permite temporizar su ejecución.

Ilustración 45. Función para mostrar inventario en pantalla.

```
componentDidMount() {
    localStorage.title = 'Inventario'

    setTimeout(() => {
        this.getInventario()
    }, 500);
}
```

Teniendo el inventario listo y visualizado, en la Ilustración 46 se encuentra el código que permite realizar las búsquedas necesarias, estas se realizan por nombre de producto, esta opción es realizada por una función “render” que compara lo que se está buscando con lo que tiene en pantalla filtrando y quitando lo que no corresponda a la búsqueda. Lo realiza con un tiempo de 100 milisegundos con mensajes en pantalla indicando “filtrando” o “reseteando”.

Ilustración 46. Función de búsqueda de productos en el inventario.

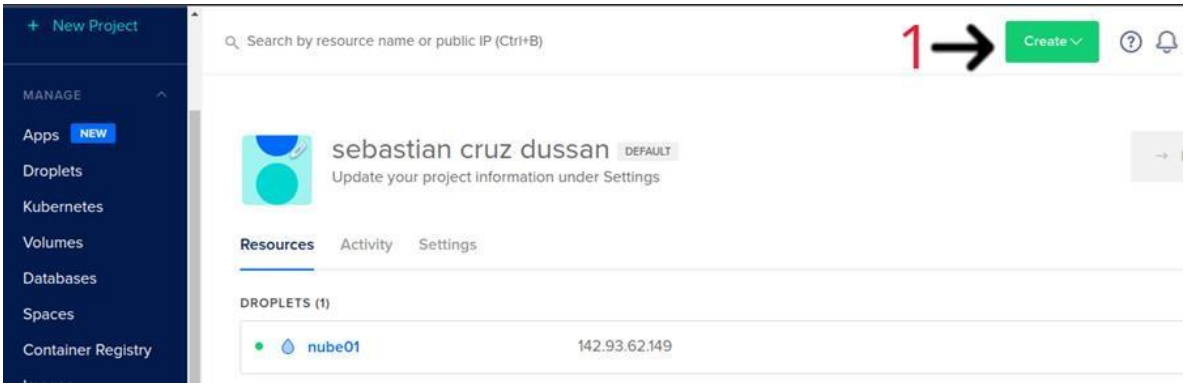
```
render() {

    const { productos } = this.state;
    const { buscando } = this.state;


    const handleBuscar = (e) => {

        let text = e.target.value;
        console.log(text);
        if (text.length > 2) {

            console.log(text.length, 'filtrando');
            var lowSearch = text.toLowerCase();
            let filtered = this.state.allProductos.filter(function (currentElement) {
                if ((currentElement.name.toLowerCase().includes(lowSearch) ||
                currentElement.default_code.includes(lowSearch) )) {
                    return currentElement
                }
            });
            // console.log(filtered);
            setTimeout(() => {
                this.setState({
                    productos: filtered
                });

            }, 100);


        } else {
            console.log( 'reseteando');

                this.setState({
                    productos: this.state.allProductos
                });
        }

    }
```

6.3 PREPARACIÓN DEL VPS

Para la configuración del VPS se tuvo en cuenta las características necesarias para contener el sistema en producción realizando la siguiente configuración.

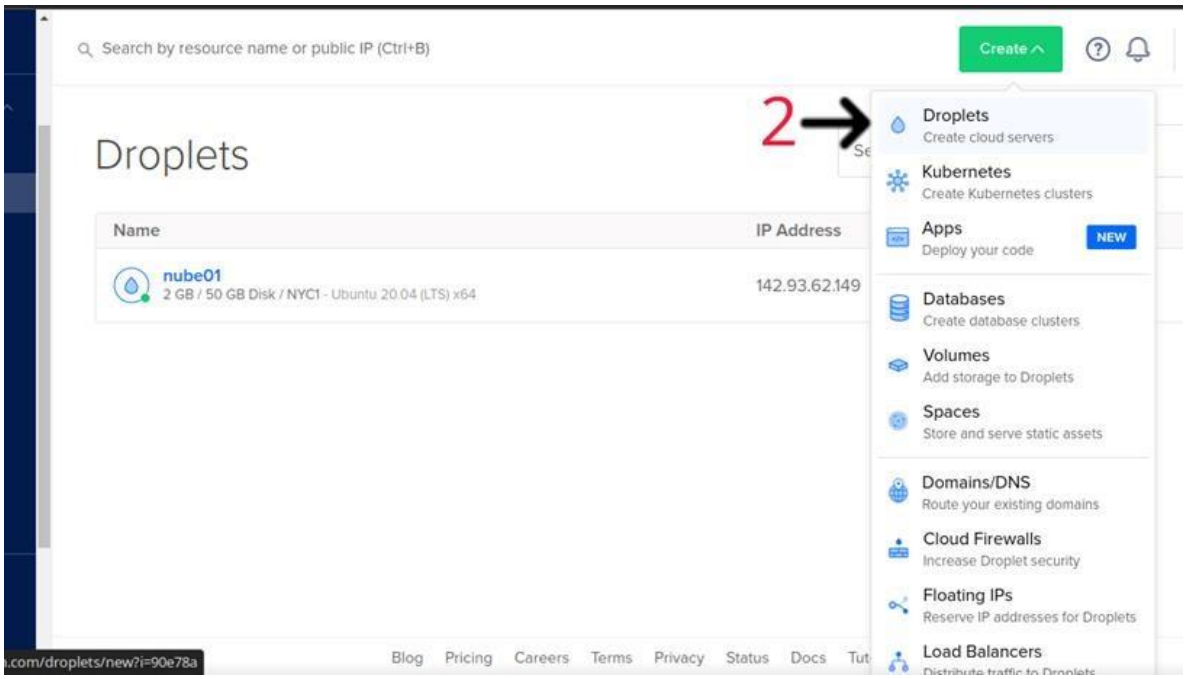
Especificaciones:
Proveedor: Digital Ocean
Sistema Operativo: Ubuntu 20.04
Memoria RAM: 2GB
Disco/Capacidad: SSD/50GB
Con el VPS adquirido se configuró realizando los siguientes pasos.

Ilustración 47. Inicio de crear servidor en Digital Ocean.



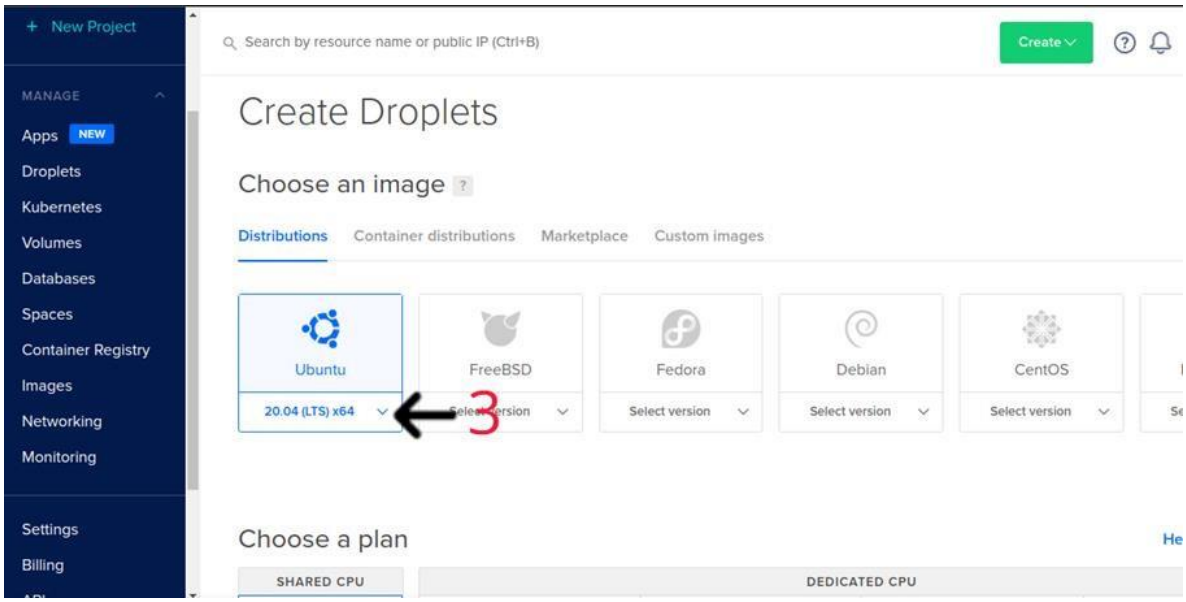
Teniendo la cuenta Digital Ocean, inicialmente se da clic en el botón “crear” como muestra la Ilustración 47.

Ilustración 48. Selección de servidor privado virtual.



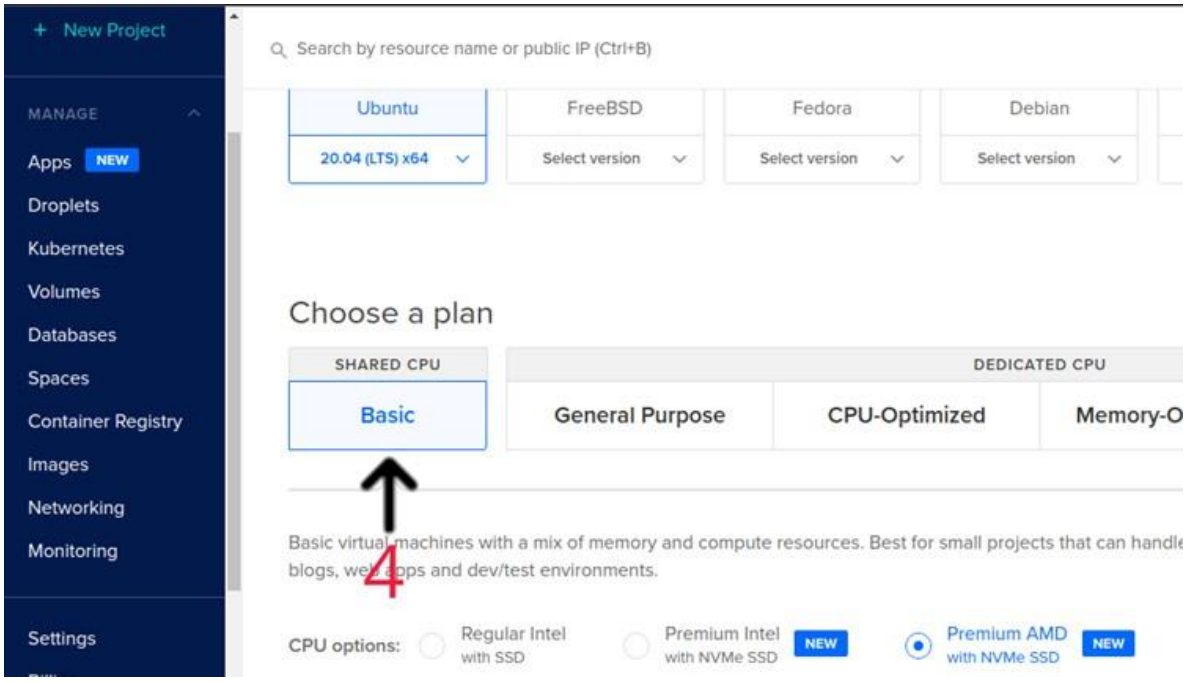
Se selecciona Droplet como indica la Ilustración 48, que es la forma en la que Digital Ocean llama a sus servidores privados virtuales o VPS

Ilustración 49. Selección de sistema instalado en el servidor.



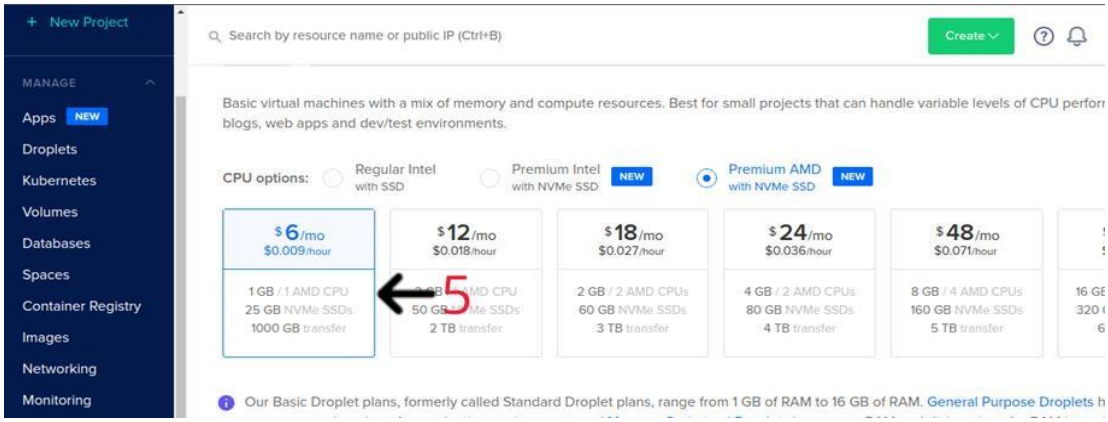
Como requerían las especificaciones se seleccionó Ubuntu 20.04, tal como la Ilustración 49.

Ilustración 50. Tipo de potencia de CPU según requerimiento de proyecto.



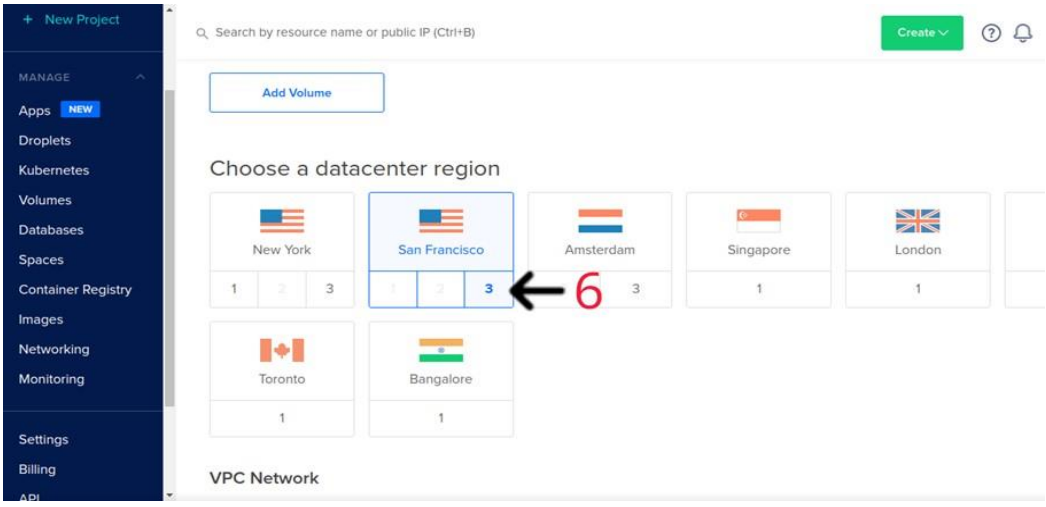
Considerando que este VPS únicamente está dedicado a mantener la aplicación web en funcionamiento y que no tendrá un número elevado de usuarios por ser dedicada para una empresa y para el trabajo de tesis, es seleccionado un tipo de procesamiento de CPU básico como se indica en la ilustración 50.

Ilustración 51. Características de almacenamiento elegidas.



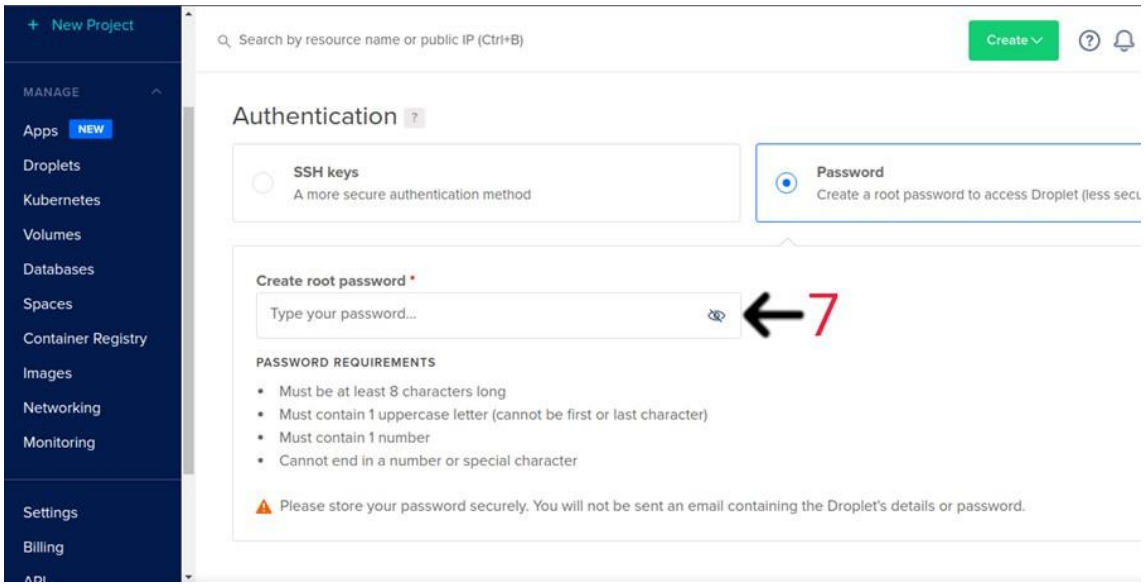
En la Ilustración 51 se denota la capacidad de almacenamiento elegida, calculada considerando la poca información que sara necesaria guardad en el VPS y considerando que los archivos compilados del proyecto no requieren mucha capacidad.

Ilustración 52. Ubicación del servidor.



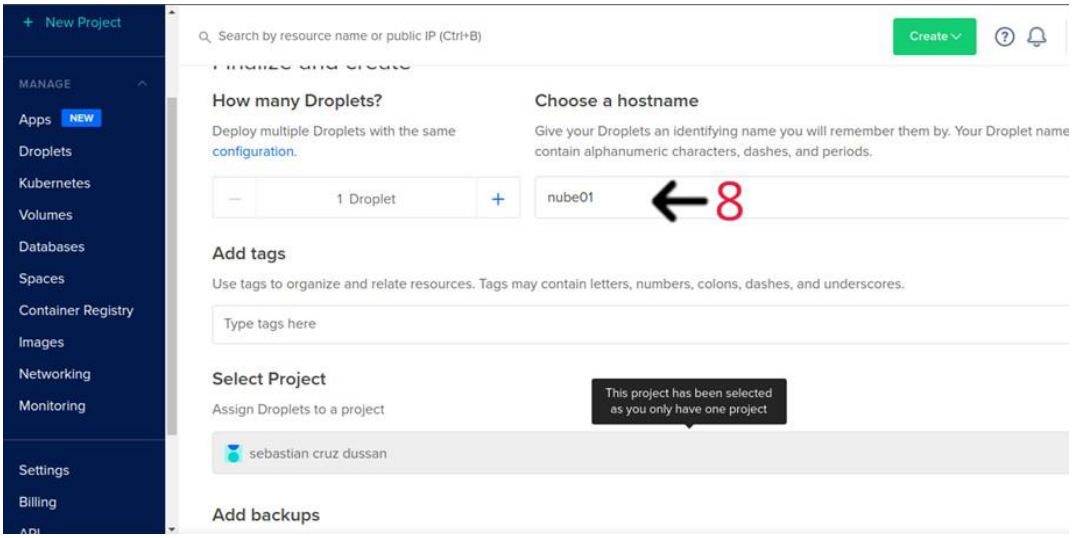
Después fue seleccionada la ubicación del datacenter como muestra la ilustración 52 en este caso se busca tener el más cercano a Colombia para tener mejor comunicación y menos probabilidad de fallos.

Ilustración 53. Tipo de autenticación para acceder al servidor.



Para la autenticación se tienen dos opciones, configurar por SSH para tener control de validación en equipos autenticados para acceder y configurar el VPS y password que como muestra la Ilustración 53 es la seleccionada y se creó una contraseña para administrar las configuraciones.

Ilustración 54. Nombre del servidor.



El último paso para terminar la configuración del VPS es escribir un nombre como indica la Ilustración 54, este nombre será el visible únicamente dentro del VPS. Con estos pasos se creó el VPS.

Estando dentro del VPS se realizaron una lista de instrucciones para terminar la configuración de inicial conFigurando puertos, actualizando paquetes e instalando Docker entre otras mostradas a continuación en su respectivo orden:

sudo apt update # Actualizar las referencias a los repositorios de Ubuntu sudo apt upgrade -y # Actualizar todos los paquetes instalados de Ubuntu sudo apt install -y docker-compose # Instala Docker Compose y sus dependencias.

sudo apt upgrade -y # Actualizar todos los paquetes instalados de Ubuntu sudo apt install -y docker-compose # Instalar Docker Compose y sus dependencias.

sudo ufw allow 22 # Firewall, se abre el puerto 22, para ingresar al vps mediante ssh. sudo ufw enable # Se habilita el firewall.

sudo ufw status # Permite ver el estado del firewall y sus puertos abiertos y protocolos permitidos.

sudo apt **install** -y certbot # Se instala la aplicación para instalar certificados SSL.

mkdir dbshared # Se crea directorio para persistencia de la base de datos.

Postgresql **mkdir** lujosEC # Se crea directorio para almacenar archivos de Odoo.

cd lujosEC # Se ingresa al directorio

mkdir config addons tmp # Se crean subdirectorios en lujosEC para distribuir los archivos de Odoo.

vdir # Lista el contenido de un directorio.

cd # Sale del directorio actual.
pwd # Muestra la ruta del directorio actual.

cd dbshared**/** # Ingresar al directorio dbshared.

sudo docker run -d -e POSTGRES_USER=odoo POSTGRES_PASSWORD=eD9Hr2BsInWXcDPSobyupsA5b --restart unless stopped --volume **~/**dbshared:**/**var**/**lib**/**postgresql**/**data --name db postgres:10.0 # Se crea el contenedor para la base de datos Postgresql.

sudo docker **ps** # Lista los contenedores que están corriendo actualmente.

sudo docker run -d -p 9000:8069 --name lujosec --restart unless-stopped -v **/**home**/**soporte**/**lujosEC**/**addons:**/**mnt**/**extra-addons -v **/**home**/**soporte**/**lujosEC**/**config:**/**etc**/**odoo --link db:db -t odoo:12.0 # Se crea el

contenedor de Odoo versión 12 con enlace hacia el contenedor de Postgresql. Escucha en el puerto 9000 del host.

Estos pasos realizados inicializan la conFiguración del VPS donde es necesario instalar un servidor web, para este caso es utilizado nginx pues Odoo está adaptado para trabajar con mayor fluidez con este servidor web en producción.

Para instalar nginx en el VPS se instala mediante el sistema de paquetes apt.
Este paso se realiza utilizando las siguientes líneas de código:

sudo apt update ==> actualiza referencias de repositorios sudo apt install nginx ==> instala nginx como servidor web sudo ufw allow 'Nginx Full' ==> Se abren los puertos 80 y 443 para acceso de aplicaciones web.

Tras aceptar el procedimiento, apt instalará Nginx y cualquier dependencia necesaria en el servidor. Pero antes de probar Nginx, se deben aplicar ajustes al software del firewall para permitir el acceso al servicio. Nginx se registra de forma automática como un servicio con ufw tras la instalación, lo que hace que sea sencillo permitir el acceso de Nginx utilizando “sudo ufw app list”.

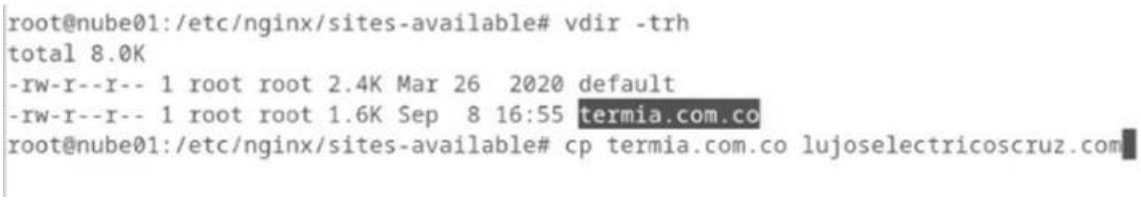
Al final del proceso de instalación, Ubuntu 20.04 inicia Nginx. El servidor web ya queda activo, esto se verificó pegando la ip del VPS en un navegador como muestra la Ilustración 55 obteniendo como respuesta un error 404 correspondiente a tener el servidor web vacío.

Ilustración 55. Servidor web creado y vacío.

6.3.1 Configuración de odoo en vps.

Para realizar la conFiguración de Odoo se modificó el archivo nginx en la ruta del VPS (/etc/nginx/sites-available) para que las peticiones a esta ip paren a ser las peticiones de Odoo, por defecto el crea uno llamado termia.com.co este fue remplazado por la url del proyecto.

Ilustración 56. Copia de la estructura del proyecto.



Como indica la Ilustración 56 se usó el comando "cp" para copiar la estructura del archivo "termia" con un nuevo nombre que corresponde a la dirección del proyecto. Luego, se utiliza el comando "rm -f" para eliminar ese archivo.

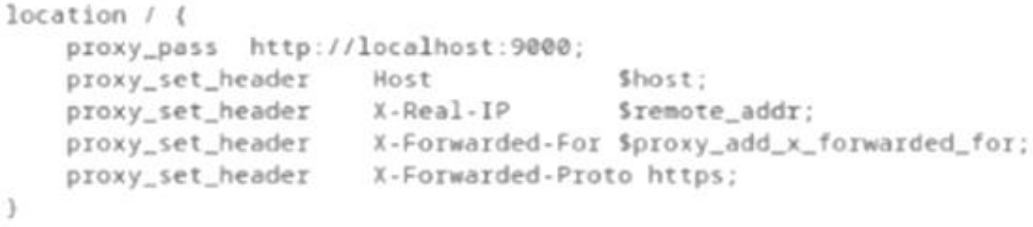
La estructura copiada de un boceto del funcionamiento para poner Odoo en producción, pero es necesario cambiar hacia dónde apuntan unas direcciones.

Ilustración 57. Cambio en la plantilla desde comando.

En la Ilustración 57 se denota el primer cambio en la plantilla, cambiando el “server_name” al requerido por el proyecto.

Otro paso importante es indicar hacia dónde va a direccionar la aplicación Odoo que está escuchando por el puerto 9000 configurado por docker

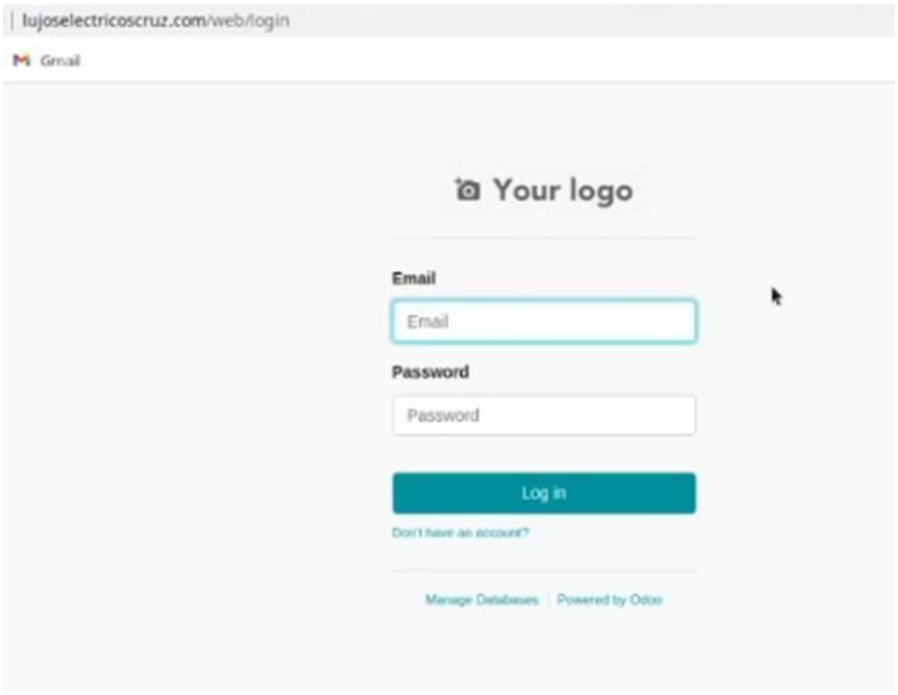
Ilustración 58. Resultado de revisión de sintaxis.



La configuración fue realizada dentro de la misma plantilla indicando en “location” el puerto local del VPS controlado por Docker y de este modo cuando se realiza la petición a “lujoselectricoscruz.com el direccione la petición al puerto local 9000 que es donde está funcionando Docker.

Desde el mismo servidor nginx permite revisiones de sintaxis que fue realizada utilizando la línea de código “syntax is ok”, obteniendo una respuesta “test is successful”, esto corrobora el funcionamiento de la configuración de nginx con Odoo, cabe resaltar que para aplicar estos cambios fue necesario también reiniciar el servidor con el comando “restart nginx.service” los resultados del est son vistos en la Ilustración 58.

Ilustración 59. Ingreso a la dirección desde la web.



La Ilustración 59 muestra el resultado de ingresar a “lujoselectricoscruz.com” con Odoo configurado en el servidor nginx.

6.3.2 Configuración react en vps

Inicialmente dentro del VPS es necesario instalar las dependencias de React para su funcionamiento con Odoo esto se realizó con la línea de código “npm install reactnative-odoo” en la consola con permisos de administrador.

Una vez terminado el proyecto React se realizó la compilación de este teniendo como resultado unos archivos planos muy comprimidos que fueron agregados en las rutas del VPS que se encuentran en “var/www/lujoselectricoscruz.com/html”, una vez teniendo los archivos de la página listos en el servidor se configuro el archivo “lujoselectricoscruz.com” que se encuentra en “etc/nginx/sites-available”, ahora para apuntar estas peticiones desde el front-end.

Ilustración 60. Configuración del front-end en la ruta y dirección del servidor.

```
server {

        root /var/www/lujoselectricoscruz.com/html;
        index index.html index.htm index.nginx-debian.html;

        server_name lujoselectricoscruz.com www.lujoselectricoscruz.com;

        location / {
           #     try_files $uri $uri/ =404;
           add_header 'Access-Control-Allow-Origin' '$http_origin' always;
           add_header 'Access-Control-Allow-Methods' 'GET, POST, OPTIONS';
           add_header 'Access-Control-Allow-Headers' 'DNT,User-Agent,X-Requested-With,I
           add_header 'Access-Control-Expose-Headers' 'Content-Length,Content-Range';

           if ($request_method = 'OPTIONS') {
               add_header 'Access-Control-Max-Age' 1728000;
               add_header 'Content-Type' 'text/plain; charset=utf-8';
               add_header 'Content-Length' 0;
```

La Ilustración 60 es la captura de la configuración del frontend donde se visualiza como parte primordial la ruta correspondiente a la dirección que almacenan los .bild y el index, también la dirección para acceder desde el navegador en “server_name” y los cors de validación y autenticación en “location”.

Ilustración 61. Manipulando de certificados SSL para asegurar la página web.

```
listen [::]:443 ssl ipv6only=on; # managed by Certbot
listen 443 ssl; # managed by Certbot
ssl_certificate /etc/letsencrypt/live/lujoselectricoscruz.com/fullchain.pem; # managed by Certbot
ssl_certificate_key /etc/letsencrypt/live/lujoselectricoscruz.com/privkey.pem; # managed by Certbot
include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot
```

La aplicación Certbot se encarga de los certificados SSL para validar el funcionamiento de HTTPS, lo que permite asegurar la página web. Certbot maneja y crea automáticamente esta parte, como se puede observar en la Ilustración 61.

Ilustración 62. Subdominio del API.

```
server {
    server_name api.lujoselectricoscruz.com www.api.lujoselectricoscruz.com;

    access_log  /var/log/nginx/odoo.access.log;
    error_log   /var/log/nginx/odoo.error.log;

    proxy_buffers 16 64k;
    proxy_buffer_size 128k;
```

La Ilustración 62 muestra el subdominio del API, que es una parte importante de la configuración de este archivo. El API es la aplicación que proporciona datos almacenados en Odoo para alimentar el front-end.

Ilustración 63. Configuración de Nginx como servidor inverso.

```
location / {
    proxy_pass http://localhost:8080;
    proxy_set_header    Host            $host;
    proxy_set_header    X-Real-IP       $remote_addr;
    proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header    X-Forwarded-Proto https;
}
```

En este caso, como la aplicación contiene información distinta a la del front-end, se necesita un servidor separado para la misma. Nginx actúa como servidor inverso y realiza un proxy en el puerto 8080, como se muestra en la Ilustración 63. Para establecer la relación entre el API de Odoo y la parte visual de la página (front-end), se requiere la instalación de ciertas dependencias especiales. Dado que se están manejando tres capas de servicio en un solo VPS distribuidas por el servidor, se ha agregado un archivo .py suministrado desde un repositorio de GitHub que corresponde a React Native Odoo Proxy.

Ilustración 64. Archivos alojada dentro del VPS.

```
soporte@nube01:~$ vdir -trh
total 664K
drwxrwxr-x  3 soporte          soporte 4.0K Aug  9  2021 odoo
drwxrwxr-x  5 soporte          soporte 4.0K Aug  9  2021 lujosEC
-rw-rw-r--  1 soporte          soporte 3.1K Aug 27  2021 history.txt
-rw-rw-r--  1 soporte          soporte 1.1K Dec 10  2021 base_custom.tar.bz2
-rw-rw-r--  1 soporte          soporte 1.3K Dec 11  2021 sale_tacos.tar.bz2
drwx------ 19 systemd-coredump soporte 4.0K May 20 20:09 dbshared
drwxrwxr-x  4 soporte          soporte 4.0K Jun 14 04:03 tmp
-rw-rw-r--  1 soporte          soporte 629K Jun 14 04:21 build_junio.rar
drwxrwxr-x  6 soporte          soporte 4.0K Jun 14 05:39 react-native-odoo-proxy
soporte@nube01:~$ cd
```

La carpeta que contiene el archivo queda alojada dentro del VPS, tal como se indica en la Ilustración 64. Dentro de esta carpeta se encuentra el archivo app.py, el cual es donde se configura la capa intermedia del servidor.

Ilustración 65. Configuración principal de la aplicación Flask con permisos CORS y puertos configurados.

```
import os
import logging
from flask import Flask, request, jsonify, make_response, render_template
from auth import check_odoo_alive, check_odoo_login
from dotenv import load_dotenv
import sys

load_dotenv()

app = Flask(__name__)
app.config['SHOW_LOGS'] = os.getenv("SHOW_LOGS")

if app.config['SHOW_LOGS'] == 'True':
    proxy_log_file = os.getenv("PROXY_LOG_FILE")
    logging.basicConfig(filename=proxy_log_file, level=logging.DEBUG,
                        format=f'%(asctime)s %(levelname)s %(name)s %(threadName)s : %(message)s
')
app.py
```

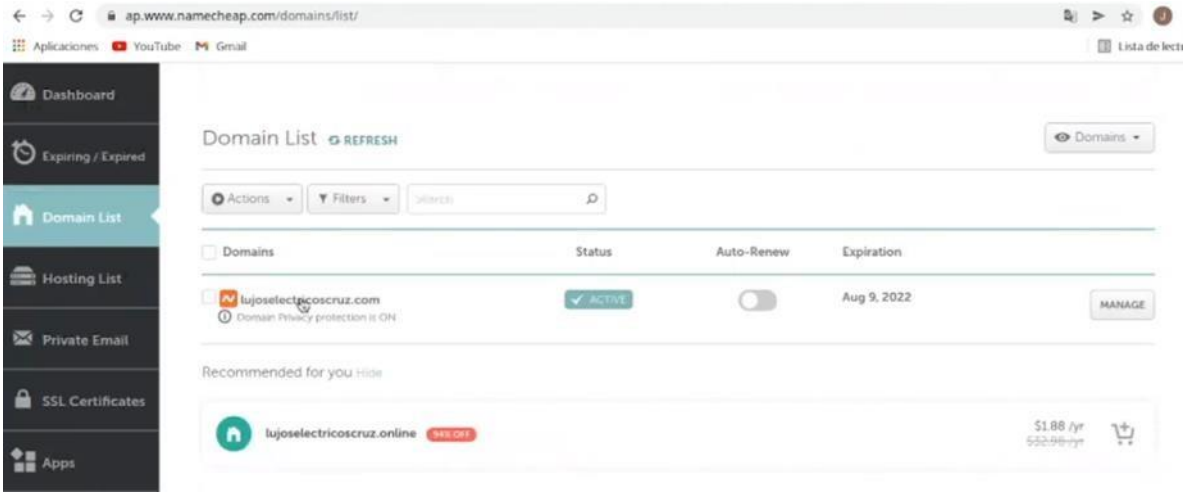
La Ilustración 65 muestra la configuración principal de la aplicación app.py, la cual ha sido codificada con Flask. Como solo se necesitaba una aplicación web sencilla para otorgar permisos para los CORS y configurar los puertos entre las capas primarias, se utiliza una aplicación llamada "pm2" para mantenerla en

funcionamiento continuo. Esta aplicación garantiza que la aplicación Python esté disponible todo el tiempo.

6.4 CONFIGURACIÓN DE DOMINIOS

El proveedor del dominio, "lujoselectricoscruz.com", es Namecheap, mientras que el proveedor de la aplicación es DigitalOcean. La aplicación tiene una dirección IP única, que es 142.93.62.149, y es necesario apuntarla a la dirección web. Para lograr esto, se realizaron algunos pasos de configuración.

Ilustración *66*. Dominio adquirido con acceso a configuración de servidor.



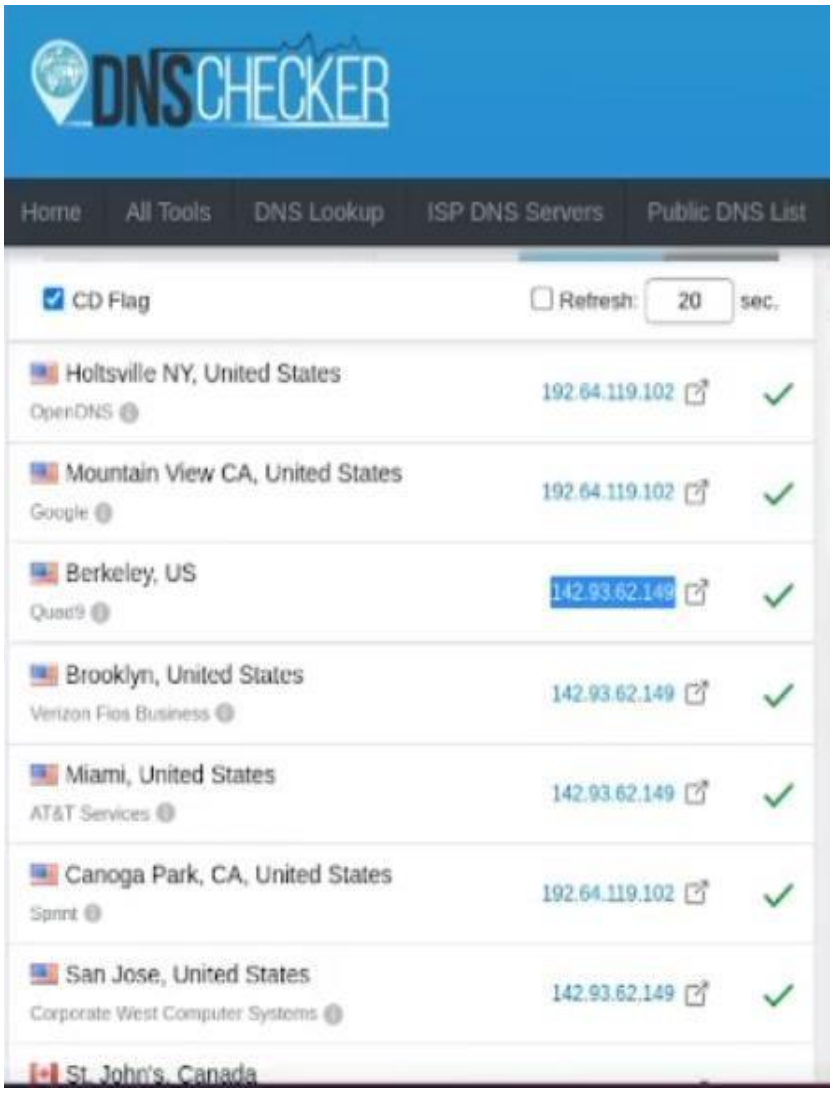
En la Ilustración 66 se puede observar el acceso a la página del proveedor de dominio donde se muestra el dominio adquirido. Al hacer clic en el botón "MANAGE", se accede a la configuración de los servidores de nombres.

Ilustración 67. Configuración DNS en DigitalOcean.

La configuración de los DNS en DigitalOcean se puede observar en la Ilustración 67. Una vez que se ha completado esta configuración, el servidor de dominio realiza

la propagación del dominio, lo que apunta la dirección web a la ruta del servidor en Digitalocean.

Ilustración 68. Herramienta de verificación de propagación del servidor.



En la Ilustración 68 se muestra una herramienta utilizada para verificar la propagación del servidor. Como parte del proceso de configuración, se verificó que el dominio adquirido apuntara al servidor adecuado, ya que podría haber tenido un enrutado diferente en el pasado. En este caso, se verificó que la dirección web correspondiente a "192.64.119.102" se haya actualizado a la dirección IP del servidor actual. Este proceso tardó aproximadamente 24 horas en propagarse por todo el mundo y quedar completamente en línea.

Una vez que el sitio web estuvo en funcionamiento, se realizaron validaciones de seguridad para garantizar que se tratara de una página segura. Esto se logró mediante el uso de certificados de seguridad generados por la herramienta cerbot de nginx en el servidor web.

Ilustración 69. Proceso de certificación SSL.

```
root@nube01:/etc/nginx/sites-available#
root@nube01:/etc/nginx/sites-available# certbot --nginx -d lujoselectricoscruz.com -d www.lujose
lectricoscruz.com
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Plugins selected: Authenticator nginx, Installer nginx
Obtaining a new certificate
Performing the following challenges:
http-01 challenge for lujoselectricoscruz.com
http-01 challenge for www.lujoselectricoscruz.com
Waiting for verification...
Cleaning up challenges
```

La Ilustración 69 muestra la línea de código principal para realizar la certificación SSL, la cual ejecuta una serie de pruebas que redirige todas las solicitudes hacia el puerto seguro. Como resultado, se obtiene un sitio seguro que cifra los datos para prevenir que alguien intercepte la información.

7. ANÁLISIS DE RESULTADOS

En el marco de esta tesis, se ha concebido un sistema de administración de inventario fundamentado en inteligencia artificial, con la capacidad de detectar y extraer información textual en ambientes no controlados. Para lograrlo, se ha desarrollado una aplicación de usuario final que simplifica la digitalización de datos. Una serie de pruebas exhaustivas se ha llevado a cabo, empleando diversas imágenes de transacciones capturadas en el registro de tacos. Estas pruebas sirvieron para contrastar el desempeño de la aplicación con el proceso manual de registro, que es el enfoque tradicional.

Los datos obtenidos mediante la digitalización de los tacos del método tradicional como del método propuesto del establecimiento comercial "Lujos y Eléctricos Cruz", ubicado en la ciudad de Neiva, en el departamento de Huila se encuentran en la tabla 1. La obtención de los datos fue gracias a la colaboración con el personal de dicho establecimiento, quienes diariamente realizan la labor de registro manual de ventas. El enfoque desarrollado en este trabajo será implementado en el mismo establecimiento, con la finalidad de recopilar datos sobre el tiempo necesario y la tasa de error asociada al proceso de digitalización de registros de transacciones de productos.

Tabla 1. Datos obtenidos del método tradicional y del método propuesto.

Numero de tacos	Tacos correctamente digitalizados método tradicional	Tacos correctamente digitalizados método propuesto
100	95	97
100	94	98
200	170	194

La cuantificación del tiempo requerido para digitalizar lotes de 100 tacos se realizó a través del cronómetro de un dispositivo móvil. Ambos enfoques, el tradicional y el propuesto, se sometieron a esta medición. En el método tradicional, el cronómetro se activaba al empezar la introducción manual de los datos de cada taco en el ordenador, y se detenía cuando se completaba la digitalización de todos los tacos del lote.

Por otro lado, en el método propuesto, se iniciaba el cronometraje al abrir la aplicación y capturar la primera imagen del lote de 100 tacos. La medición se detenía cuando todos los datos de los tacos estaban completamente digitalizados en la aplicación.

Para evaluar el margen de error, se contrastaron los lotes de 100 tacos en ambos métodos. Mientras que en el método tradicional todos los errores eran de origen humano, el método propuesto permitía corregir directamente los errores generados por la aplicación, además de obtener datos a partir de las imágenes. Esto contribuye a prevenir los errores humanos asociados al cansancio o la fatiga visual.

Esta metodología de medición posibilitó una captura precisa del tiempo requerido para completar cada proceso de digitalización en ambos enfoques, proporcionando una base cuantitativa sólida para comparar la eficiencia entre el método tradicional y la propuesta innovadora.

Para obtener el porcentaje de error de ambos métodos se implementó la siguiente ecuación:

$$\%E = \left| \left(\frac{Aciertos - exacto}{exacto} \right) \right| * 100 \;\; ec(1)$$

Los resultados presentados en la Tabla 2 reflejan de manera inequívoca que la aplicación de digitalización implementada es capaz de extraer los datos de las transacciones de manera considerablemente más rápida que el método tradicional. Esta mejora en la eficiencia se debe en gran medida a la eliminación de factores humanos que afectan el proceso manual, como la fatiga, problemas de visión y errores de interpretación de imágenes, cabe destacar que, al aumentar número de tacos a digitalizar, el proceso tradicional incrementa el triple de los errores humanos pasando del 5% al 15% en errores de los tacos digitalizados. La Ilustración 70 proporciona una representación visual de cómo el sistema clasifica de manera precisa los datos relevantes de cada transacción. Asimismo, en la Ilustración 71 se presentan los datos de la transacción una vez que han sido exitosamente digitalizados y procesados por el sistema.

Tabla 2. Comparativa del método tradicional contra el método digital.

Número de tacos	Tiempo de ejecución (Método tradicional)	Porcentaje de error (método tradicional)	Tiempo de ejecución (sistema propuesto)	Porcentaje de error (sistema propuesto)
100	2.30hr	5%	1.15hr	3%

100	2.35hr	6%	1.20hr	2%
200	5.0hr	15%	3.30hr	3%

Ilustración 70. Captación de la información del taco a 8 cm-30 grados.

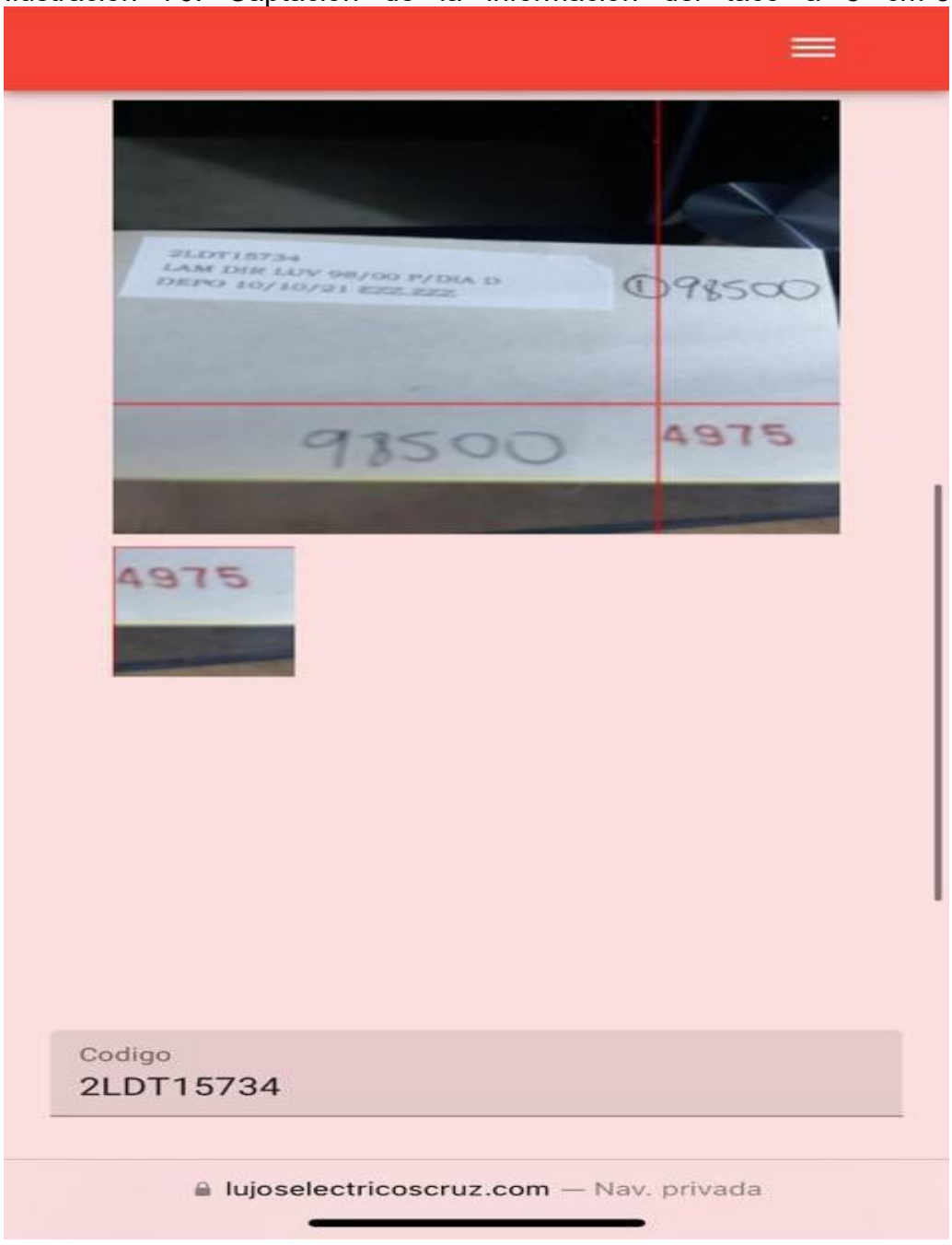


Ilustración 71. Datos obtenidos con el sistema.

Para determinar el ángulo de inclinación y la distancia óptima entre el celular y el taco, se llevará a cabo la captura de 600 fotografías. Estas imágenes se agruparán en conjuntos de 100, cada uno variando la distancia o el ángulo de inclinación durante la captura. El sistema se considera preciso cuando logra identificar y digitalizar correctamente tanto el código como el identificador del taco.

El conjunto de datos se compone de múltiples grupos de 100 imágenes, cada uno representando una configuración específica de ángulo y distancia entre el celular y el taco. El propósito es determinar cuál de estas configuraciones resulta en el mayor índice de aciertos en la identificación y digitalización. En la Tabla 3 se presenta el porcentaje de aciertos obtenidos de la muestra de 100 imágenes capturadas de los tacos a través de la aplicación de usuario final del sistema. Estas imágenes fueron tomadas utilizando un dispositivo móvil de referencia iPhone 13.

Este enfoque experimental busca proporcionar una visión precisa sobre cómo las diferentes condiciones de captura, en términos de ángulo y distancia, impactan en la eficiencia y precisión del sistema de detección y digitalización. Los resultados de esta evaluación ofrecerán información valiosa para establecer las configuraciones óptimas y, al final, mejorar el desempeño general del sistema.

Tabla 3. Aciertos de los códigos marquilla e identificación del taco.

Distancia de la cámara (cm) - ángulo de inclinación en grados.	Porcentaje de aciertos del código de marquilla.	Porcentaje de aciertos del código identificador del taco.
7 cm – 0 grados	60%	20%
8 cm - 0 grados	40%	50%
8 cm 30 grados	60%	100%
10 cm - 0 grados	70%	50%
10 cm - 30 grados	30%	70%
11 cm – 0 grados	20%	30%

En la Tabla 3 se detalla el desempeño en la prueba de validación utilizando el teléfono celular designado. Los resultados muestran que el porcentaje de aciertos es más alto cuando la captura de imagen se realiza con un ángulo de inclinación de 30 grados. Esto se debe al enfoque de la cámara en relación con la posición del identificador de los tacos. Respecto a la distancia de captura, se observa que la mejor elección es a 10 cm, ya que es la distancia óptima para la lente del celular y, por fin, proporciona imágenes nítidas y bien enfocadas.

En concordancia con lo anterior, se concluye que la configuración más efectiva para la captura de imágenes es a una distancia de 8 cm con un ángulo de inclinación de 30 grados. Esta combinación resulta en los mayores porcentajes de aciertos tanto en la identificación de la marca como en el identificador del taco. Estos hallazgos contribuyen significativamente a mejorar la precisión y eficiencia general del sistema de detección y digitalización.

Ilustración 72. Aciertos de código de marquilla.

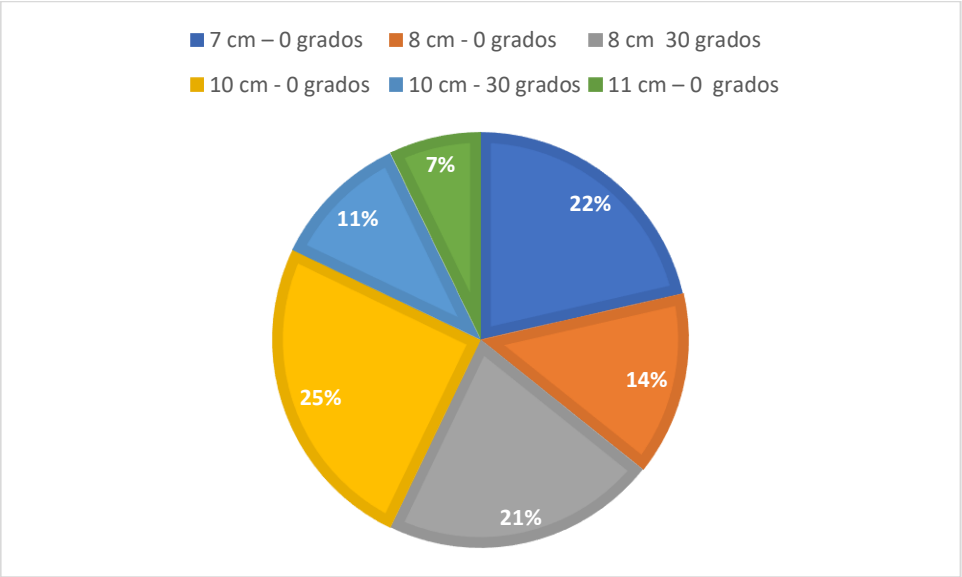
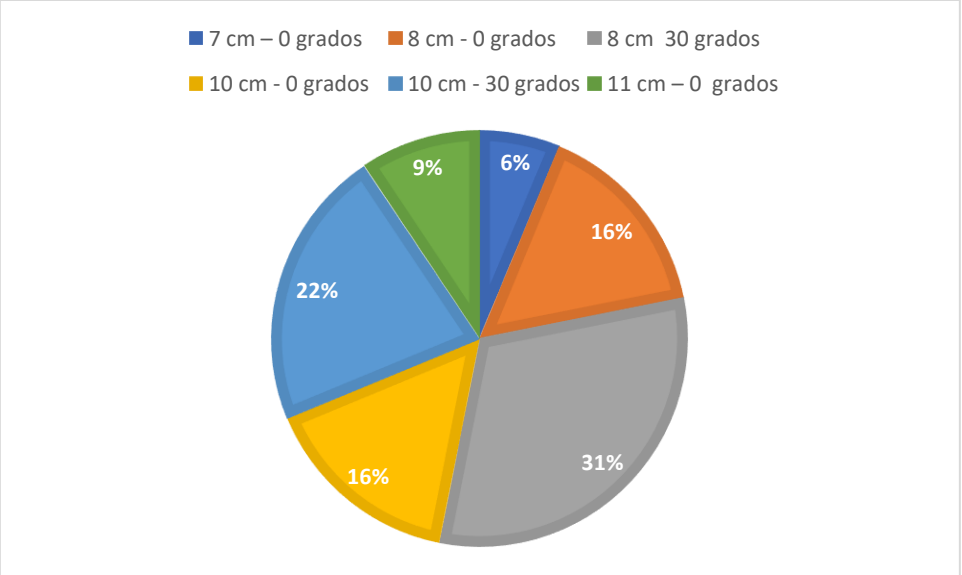


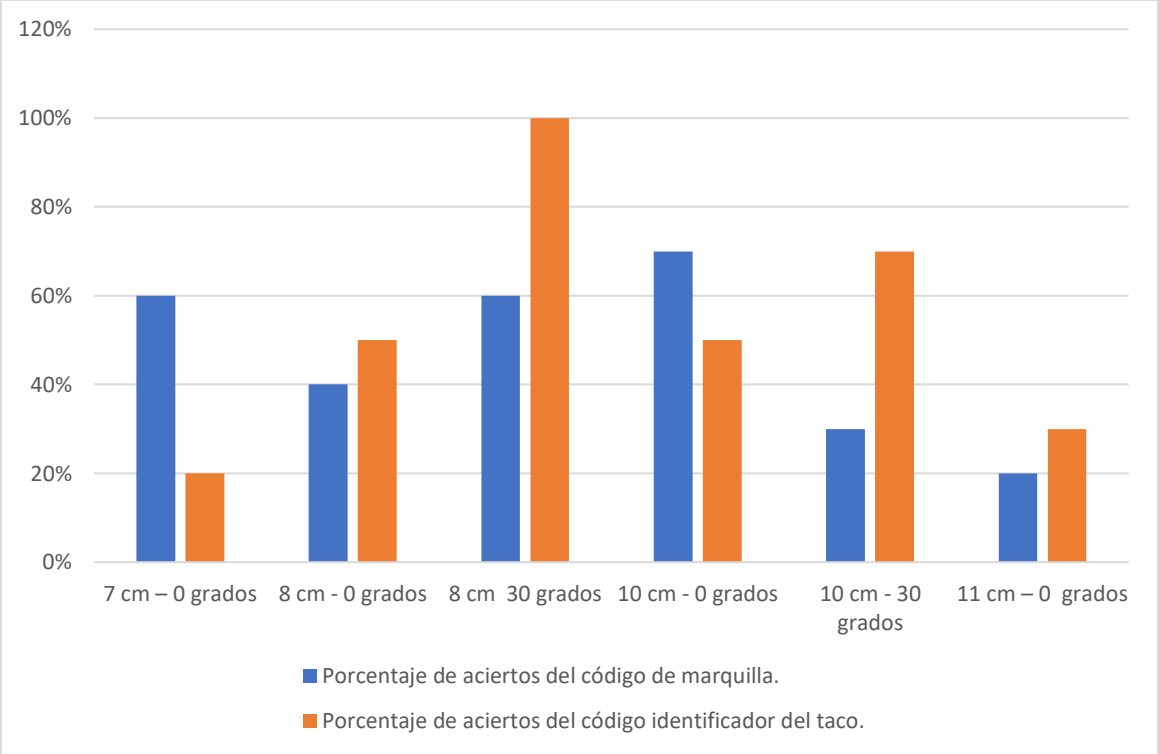
Ilustración 73. Acierto del código identificador del taco.



A través de la Ilustración 72, se puede apreciar que las distancias que presentan menor precisión en la captura de la marquilla entre la cámara y la escena son 11 cm y 7 cm desde la lente hasta la imagen. Esta discrepancia se origina debido a que la imagen resulta estar fuera de los parámetros de estructuración predeterminados para el taco. Es crucial que el usuario final se asegure de posicionar la imagen de manera adecuada con respecto a la cuadrícula de datos, tal como se muestra en la

Ilustración 2. En esta ilustración, la marquilla, el precio del producto y el identificador del taco se están dispuestos en posiciones cartesianas. Este alineamiento es esencial para lograr una captura precisa y coherente de la información relevante.

Ilustración 74. Porcentaje de aciertos a diferentes ángulos de inclinación y distancia de la cámara.



La Ilustración 74 revela que el grado de precisión en la detección del código de la marca y la identificación del taco experimentó mejoras cuando se incrementó la distancia de la cámara y se redujo el ángulo de inclinación. Un ejemplo destacado se observa cuando se analiza una distancia de 10 cm junto a un ángulo de inclinación de 0 grados. En esta configuración, se obtuvo un acierto del 70% en la detección del código de la marca, y un 50% en la identificación del taco.

Una observación aún más notable se desprende de una distancia de 8 cm junto a un ángulo de inclinación de 30 grados. En esta combinación, se alcanzó un acierto del 60% en la detección del código de la marca, mientras que la identificación del taco obtuvo un impresionante 100% de aciertos.

La mejora en la precisión al adoptar un ángulo de inclinación de 30 grados se debe a una alineación más directa y adecuada entre la lente de la cámara y el identificador del taco. Esto permite que el sistema capte con mayor claridad la información

relevante, aumentando así la tasa de aciertos en la detección. En conjunto, estos hallazgos destacan la importancia de considerar tanto la distancia como el ángulo de inclinación en la optimización del proceso de detección y digitalización de datos.

Se mejora de manera significativa los costos laborales a la hora de implementar el sistema propuesto:

Se procederá a calcular los costos tanto del sistema tradicional como del sistema propuesto, basándonos en los datos obtenidos de la Tabla 1. Para ello, determinaremos la cantidad promedio de tacos digitalizados por hora utilizando ambos métodos, considerando una jornada laboral diaria de 8 horas, que es el estándar en una jornada laboral.

Mediante la siguiente ecuación se realiza la tabla 4 y 5. La cual muestra los tacos digitalizados del método tradicional y el método propuesto.

$$\frac{\sum_{i=1}^{n} tacos\ cada\ hora_n}{n} \quad ec(2)$$

Tabla 4. Cantidad de Tacos Digitalizados por Hora en Cada Sesión del método tradicional

Número de tacos	Tiempo de Ejecución	Tacos por Hora
100	2.3	43,47
100	2.35	42,55
200	5.0	20

Promedio utilizando el método tradicional: 35.4 tacos por hora.

Tabla 5. Cantidad de Tacos Digitalizados por Hora en Cada Sesión de Experimento utilizando el Sistema Propuesto.

Número de tacos	Tiempo de Ejecución	Tacos por Hora
100	1.15	86,95
100	1.2	83.3
200	3.3	60

Promedio utilizando el sistema propuesto: 76.75 tacos por hora.

En la Tabla 6, se realiza una comparativa de los costos entre el método tradicional y el método propuesto.

Tabla 6. Comparativa de Costos entre el Método Tradicional y el Método Propuesto

Método	Tiempo de Ejecución (h)	Total, de Tacos Digitalizados	Costo en Tiempo Empleado (COP)	Costo en Servidor (USD)
Tradicional	192	6797	1400000	-
propuesta	88,56	6797	768600	69.0

El costo por tiempo empleado en el método tradicional se obtuvo considerando el salario mínimo y aproximadamente 7292 pesos por hora. En el caso del método propuesto, se suma el costo del servidor, que equivale a 18 dólares por mes, y 1,5 dólares por cada mil tacos.

En conjunto, estos resultados demuestran que el método propuesto permite una mayor eficiencia en términos de tacos digitalizados por hora, lo que a su vez se traduce en un menor costo operativo en comparación con el método tradicional. La optimización en la velocidad de digitalización y la eficacia en el uso de recursos tecnológicos hacen que el sistema propuesto sea una opción atractiva y rentable en entornos de captura y procesamiento de información.

8. CONCLUSIONES

La implementación del sistema diseñado en este trabajo de grado en la empresa Lujos y Eléctricos Cruz ha demostrado ser una herramienta valiosa para mejorar la eficiencia y velocidad del proceso de digitalización del taco. La utilización de un dispositivo móvil permitió una mayor precisión y rapidez en la recolección de datos, comparado con el método anteriormente utilizado, que consistía en la realización manual por parte de un usuario. Además, el uso del sistema contribuyó a reducir los costos operativos en esta área de la empresa. En resumen, la implementación de este sistema ha mejorado significativamente el proceso de digitalización del taco en Lujos y Eléctricos Cruz.

El análisis exhaustivo de los datos obtenidos de las pruebas comparativas entre el método tradicional y el sistema propuesto ha proporcionado resultados de notable relevancia. Al evaluar la cantidad de tacos digitalizados por hora, se constata de manera inequívoca que el sistema propuesto presenta un desempeño superior en términos de eficiencia. Con un promedio de 76.75 tacos digitalizados por hora en contraposición a los 35.4 tacos del método tradicional, este sistema no solo duplica la productividad, sino que también reduce considerablemente los costos operativos.

Además de su efecto en la eficiencia, el sistema propuesto también presenta ventajas financieras notables. La comparativa de gastos refleja que, incluso al considerar el costo del servidor y otros gastos adicionales, el sistema propuesto sigue siendo económicamente más ventajoso que el método tradicional. Este hallazgo demuestra el potencial de la ingeniería y la tecnología para generar un impacto positivo en la gestión de costos y la rentabilidad.

En conjunto, estos resultados refuerzan la noción de que la innovación tecnológica respaldada por sólidos fundamentos de ingeniería puede desempeñar un papel clave en la transformación empresarial. La optimización de procesos, la reducción de errores humanos y el aumento en la velocidad de digitalización se alinean con los objetivos de las organizaciones modernas en busca de una mayor eficiencia y competitividad. Esta tesis no solo destaca la efectividad del sistema propuesto, sino que también resalta el valor esencial de la ingeniería en la búsqueda de soluciones pragmáticas y efectivas para los desafíos empresariales actuales.

9. BIBLIOGRAFÍA.

[1] Dinero. (19 de abril de 2018). Colombia ya tiene política pública de Big Data - Dinero. Recuperado de https://www.dinero.com/pais/articulo/colombia-ya-tienepolitica-publica-de-big-data/257479

[2] MinTIC. (6 de diciembre de 2017). MinTIC revela los primeros resultados del Observatorio de Economía.... Recuperado de https://www.mintic.gov.co/portal/604/w3-article-61929.html

[3] MinTIC. (6 de diciembre de 2017). MinTIC revela los primeros resultados del Observatorio de Economía.... Recuperado de https://www.mintic.gov.co/portal/604/w3-article-61929.html

[4] DNP. (17 de abril de 2018). Colombia primer país en Latinoamérica con una política... - DNP. Recuperado de https://www.dnp.gov.co/Paginas/Colombia-primerpa%C3%ADs-en-Latinoam%C3%A9rica-con-una-pol%C3%ADticap%C3%BAblica-para-la-explotaci%C3%B3n-de-datos-Big-Data.aspx

[5] DNP. (17 de abril de 2018). Colombia primer país en Latinoamérica con una política... - DNP. Recuperado de https://www.dnp.gov.co/Paginas/Colombia-primerpa%C3%ADs-en-Latinoam%C3%A9rica-con-una-pol%C3%ADticap%C3%BAblica-para-la-explotaci%C3%B3n-de-datos-Big-Data.aspx

[6] Sucar Enrique, & Gómez, Giovani. (2016). Visión computacional. Instituto Nacional de Astrofísica, Óptica y Electrónica.

[7] Zhou, X., Yao, C., Wen, H., Wang, Y., Zhou, S., He, W., & Liang, J. (2017). EAST: An Efficient and Accurate Scene Text Detection. Megvii Technology Inc., Beijing, China.

[8] Tellez, Alberto. (2005). Extracción de información con algoritmo de clasificación (tesis de maestría). Tonantzintla, Pue.

[9] López, Rafael. (2016). Detección de texto en entornos no controlados (trabajo de grado). Universidad Politécnica de Valencia, Valencia, España.

[10] Jaderberg, M. (2014). Deep Learning for text spotting (D.Phil Thesis). Universidad de Oxford, Michaelmas Term.

[11] Google Cloud. (s.f.). Cloud Vision API: Reconocimiento óptico de caracteres (OCR). Recuperado de https://cloud.google.com/vision/docs/ocr?hl=es-419.

[12] Odoo. (s.f.). Que es odoo. Recuperado de https://www.odoo.com/es_ES/page/about-us.

[13] Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Media.

[14] Chollet, F. (2017). Deep Learning with Python. Manning Publications.

[15] Gómez-Verdejo, V. (2019). Deep Learning. Aprende a construir redes neuronales desde cero. Marcombo.

[16] Molina, P., Vargas, D., Prieto, A., & Montalvo, A. (2019). Aprendizaje profundo con Python: la guía completa para desarrollar proyectos de inteligencia artificial. Alfaomega.