TESIS UANCV
VICERRECTORADO DE INVESTIGACIÓN
"OFICINA DE INVESTIGACIÓN"

UNIVERSIDAD ANDINA

NÉSTOR CÁCERES VELÁSQUEZ

FACULTAD DE INGENIERÍA DE SISTEMAS

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

DESARROLLO DE UN SISTEMA DE INVENTARIO PARA OPTIMIZAR LA GESTIÓN DE ALMACÉN EN LA EMPRESA INTIQA HOTEL PUNO 2025

TESIS PRESENTADA POR:

Bach. ABELARDO IDME MAMANI

PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERO DE SISTEMAS

JULIACA- PERÚ

2026

OFICINA DE INVESTIGACIÓN
Tesis Publicada con autorización del autor
http://repositorio.uancv.edu.pe/

TESIS UANCV
UNIVERSIDAD ANDINA
VICERRECTORADO DE INVESTIGACIÓN
"OFICINA DE INVESTIGACIÓN"

NÉSTOR CÁCERES VELÁSQUEZ

FACULTAD DE INGENIERÍA DE SISTEMAS

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

DESARROLLO DE UN SISTEMA DE INVENTARIO PARA OPTIMIZAR LA GESTIÓN DE ALMACÉN EN LA EMPRESA INTIQA HOTEL PUNO 2025

TESIS PRESENTADA POR:

Bach. ABELARDO IDME MAMANI

PARA OPTAR EL TÍTULO PROFESIONAL DE:

INGENIERO DE SISTEMAS

APROBADA POR EL JURADO REVISOR:

PRESIDENTE : _____
Dr. JUAN CARLOS HERRERA MIRANDA

PRIMER MIEMBRO : _____
M.Sc JESUS ESTEBAN CASTILLO MACHACA

SEGUNDO MIEMBRO : _____
Ing. ADWAR RANULFO SANCHEZ CARREON

ASESOR DE TESIS : _____
Dr. PAUL MAMANI TISNADO

LÍNEA DE INVESTIGACIÓN : CIENCIA DE LOS ORDENADORES – P24

OFICINA DE INVESTIGACIÓN
Tesis Publicada con autorización del autor
http://repositorio.uancv.edu.pe/

TESIS UANCV

RESOLUCIÓN DECANAL Nº 283-2026-D-FIS-UANCV

Juliaca, 02 de julio de 2026

VISTOS; el Expediente Nº 2026-CU-005581, presentado por el (a) señor (a) **IDME MAMANI ABELARDO**, quien solicita REVISION DEL INFORME FINAL DE LA INVESTIGACIÒN (borrador de tesis), **el proveído Nº 146-2026-UI-FIS-UANCV/J**, del Director de la Unidad de Investigación de la Facultad de Ingeniería de Sistemas.

CONSIDERANDO:

Que, el (la) señor (a) **IDME MAMANI ABELARDO**, ha presentado su informe final de la investigación (borrador de tesis), titulado: **DESARROLLO DE UN SISTEMA DE INVENTARIO PARA OPTIMIZAR LA GESTIÓN DE ALMACÉN EN LA EMPRESA INTIQA HOTEL PUNO 2025**, para optar el título profesional de **INGENIERO DE SISTEMAS.**

Que, al haberse cumplido con los requisitos exigidos por el Reglamento Interno de Trabajo de Investigación conducente a Grados y Títulos con fines de obtención de Grados Académicos y Títulos, el Director de la Unidad de Investigación de la Facultad de Ingeniería de Sistemas emitió opinión favorable al Expediente 2026-CU-005581, del informe final (borrador de tesis), titulado: **DESARROLLO DE UN SISTEMA DE INVENTARIO PARA OPTIMIZAR LA GESTIÓN DE ALMACÉN EN LA EMPRESA INTIQA HOTEL PUNO 2025**, correspondiente a la línea de investigación, **CIENCIAS DE LOS ORDENADORES.**

Que al haberse cumplido con los requisitos exigidos por el reglamento interno de trabajos de investigación conducentes a grados y títulos mediante resolución Nº 0294-2023-UANCV-CU-R, y estando a la opinión favorable de la Unidad de Investigación respecto al informe final de la investigación (borrador de tesis).

Estando, con la opinión favorable de la propuesta de investigación del Comité de investigación de la Facultad de Ingeniería de Sistemas y en concordancia con el reglamento interno de trabajos de investigación conducente a grados y títulos, aprobado con resolución Nº 0294-2023-UANCV-CU-R y en mérito al art. 25 del reglamento, con fines de obtención de obtención de Grados Académicos y Títulos Profesionales y en uso a las atribuciones que le concede la ley Universitaria Nº 30220, Ley de creación de la UANCV Nº 23738 y modificatoria Nº P 24661, y el estatuto de la UANCV, el Decano y Director de la Facultad de Ingeniería de Sistemas.

RESUELVE:

ARTÍCULO PRIMERO.- APROBAR, el INFORME FINAL DE LA INVESTIGACION (BORRADOR DE TESIS), para la REVISION DE SIMILITUD TURNITIN, presentado por el (la) señor (a): **IDME MAMANI ABELARDO**, para optar el Título Profesional de **INGENIERO DE SISTEMAS** con el tema titulado: **DESARROLLO DE UN SISTEMA DE INVENTARIO PARA OPTIMIZAR LA GESTIÓN DE ALMACÉN EN LA EMPRESA INTIQA HOTEL PUNO 2025**, correspondiente a la línea de investigación, **CIENCIAS DE LOS ORDENADORES**, en virtud a los considerandos expuestos.

ARTÍCULO SEGUNDO. - RATIFICAR como **ASESOR DE INVESTIGACION** al (a la) **Dr. PAUL MAMANI TISNADO.**

ARTÍCULO TERCERO. - DISPONER que, la Unidad de Investigación, Responsables del Comité de Investigación de la Facultad de Ingeniería de Sistemas y el Director de la Escuela profesional, **INGENIERÍA DE SISTEMAS**, quedan encargados del cumplimiento de la presente Resolución.

Regístrese, Comuníquese y Archívese.

UNIVERSIDAD ANDINA "NÉSTOR CÁCERES VELÁSQUEZ" FACULTAD DE INGENIERÍA DE SISTEMAS - JULIACA - DECANO

UNIVERSIDAD ANDINA
"NÉSTOR CÁCERES VELÁSQUEZ"

Dr. Juan Carlos Herrera Miranda
DECANO

C.c.
Arch 2026
interesado
JCHM/e.h.r.

OFICINA DE INVESTIGACIÓN
Tesis Publicada con autorización del autor
http://repositorio.uancv.edu.pe/

TESIS UANCV

VICERRECTORADO DE INVESTIGACIÓN
"OFICINA DE INVESTIGACIÓN"

UNIVERSIDAD ANDINA "NÉSTOR CÁCERES VELÁSQUEZ"

RESOLUCIÓN DECANAL Nº 299-2026-D-FIS-UANCV

Juliaca, 17 de julio de 2026

VISTOS; el Expediente Nº 2026-CU-006578, presentado por el señor (a) **IDME MAMANI ABELARDO,** egresado de la Escuela Profesional de **INGENIERÍA DE SISTEMAS,** de la Facultad de Ingeniería de Sistemas, quien solicita **NOMINACIÓN DE JURADOS Y PROGRAMACIÓN DE FECHA Y HORA DE SUSTENTACIÓN.**

CONSIDERANDO:

Que, el (la) Bachiller **IDME MAMANI ABELARDO,** quien solicita **NOMINACIÓN DE JURADOS Y PROGRAMACIÓN DE FECHA Y HORA DE SUSTENTACIÓN** de la tesis titulada: **DESARROLLO DE UN SISTEMA DE INVENTARIO PARA OPTIMIZAR LA GESTIÓN DE ALMACÉN EN LA EMPRESA INTIQA HOTEL PUNO 2025,** la misma que pertenece a la línea de investigación, **CIENCIAS DE LOS ORDENADORES,** para optar el título profesional de **INGENIERO DE SISTEMAS.**

Que, al haberse cumplido con los requisitos exigidos por el Reglamento Interno de Trabajo de Investigación conducente a Grados y Títulos mediante resolución 0294-2023-UANCV-CU-R y en concordancia con el dictamen de similitud.

De conformidad al Reglamento Interno de Trabajos de Investigación conducente a grados y títulos, aprobado con resolución Nº 0294-2023-UANCV-CU-R y en mérito al art. 24, Art. 28 del reglamento, con fines de obtención de obtención de Grados Académicos y Títulos Profesionales y en uso a las atribuciones que le concede la ley Universitaria Nº 30220, Ley de creación de la UANCV Nº 23738 y modificatoria Nº 24661, y el estatuto de la UANCV, el Decano y Director de la Unidad de Investigación de la Facultad de Ingeniería de Sistemas.

RESUELVE:

ARTÍCULO PRIMERO. - APROBAR la **NOMINACION DE JURADOS,** integrado por los siguientes docentes:

Presidente	**: Dr. JUAN CARLOS HERRERA MIRANDA**
1er miembro	**: M.Sc. JESUS ESTEBAN CASTILLO MACHACA**
2do miembro	**: Ing. ADWAR RANULFO SANCHEZ CARREON**

ARTÍCULO SEGUNDO. - RECONOCER como asesor de la investigación (tesis) de la Facultad de Ingeniería de Sistemas, al (a la) docente, **Dr. Paul Mamani Tisnado**

ARTÍCULO TERCERO. - APROBAR la **FECHA Y HORA DE SUSTENTACION DE LA TESIS** del (la) Bachiller: **IDME MAMANI ABELARDO,** del informe final de la investigación (tesis) titulado: **DESARROLLO DE UN SISTEMA DE INVENTARIO PARA OPTIMIZAR LA GESTIÓN DE ALMACÉN EN LA EMPRESA INTIQA HOTEL PUNO 2025,** para optar el título profesional de **INGENIERO DE SISTEMAS,** de acuerdo al siguiente detalle:

FECHA : Viernes, 17 de julio de 2026
HORA : 04:00 p.m.
LUGAR : Sala de Docentes-FIS

ARTÍCULO CUARTO. - DISPONER que, la Unidad de Investigación, Responsables del Comité de Investigación de la Facultad de Ingeniería de Sistemas y el Director de la Escuela profesional, **INGENIERÍA DE SISTEMAS,** quedan encargados del cumplimiento de la presente Resolución.

Regístrese, Comuníquese y Archívese.

UNIVERSIDAD ANDINA "NÉSTOR CÁCERES VELÁSQUEZ"
DECANO

Dr. Juan Carlos Herrera Miranda
DECANO

Cc.
Arch 2026
Interesado
JCHM/c.h.r.

OFICINA DE INVESTIGACIÓN
Tesis Publicada con autorización del autor
http://repositorio.uancv.edu.pe/

TESIS UANCV
VICERRECTORADO DE INVESTIGACIÓN
"OFICINA DE INVESTIGACIÓN"

UNIVERSIDAD ANDINA "NÉSTOR CÁCERES VELÁSQUEZ"

RESOLUCIÓN DECANAL Nº 0028-2025-D-FIS-UANCV

Juliaca, 09 de septiembre de 2025

VISTOS; el Expediente Nº **2025-CU-7516**, presentado por el señor (a) **ABELARDO IDME MAMANI** solicitando **APROBACIÓN DE LA PROPUESTA DE INVESTIGACION**, el PROVEIDO Nº **004-2024-UI-FIS-UANCV/J** y la **OPINIÓN DE LA PROPUESTA DE INVESTIGACIÓN** formato ANEXO Nº 01 del Director de la Unidad de Investigación de la Escuela Profesional de INGENIERIA DE SISTEMAS de la Facultad de Ingeniería de Sistemas, según el reglamento interno de trabajo de investigación conducente a grados y títulos .

CONSIDERANDO:

Que, el (la) señor (a) **ABELARDO IDME MAMANI**, ha presentado su propuesta de investigación titulado: **DESARROLLO DE UN SISTEMA DE INVENTARIO PARA OPTIMIZAR LA GESTIÒN DE ALMACÈN EN LA EMPRESA INTIQA HOTEL PUNO 2025**, para optar el Título Profesional de **INGENIERO DE SISTEMAS.**

Que, al haberse cumplido con los requisitos exigidos por el Reglamento Interno de Trabajo de Investigación conducente a Grados y Títulos con fines de obtención de Grados Académicos y Títulos, el Director de la Unidad de Investigación de la Facultad de Ingeniería de Sistemas emitió opinión favorable de la propuesta de investigación; Expediente 2025-CU-7516, aprobando la propuesta de investigación titulada: **DESARROLLO DE UN SISTEMA DE INVENTARIO PARA OPTIMIZAR LA GESTIÒN DE ALMACÈN EN LA EMPRESA INTIQA HOTEL PUNO 2025.**

Que es requisito indispensable contar con un asesor docente ordinario y/o contratado de la facultad de Ingeniería de Sistemas, con mínimo de cinco años de docencia, grado de doctor o magister y experiencia en la línea a investigar, o deberá estar acreditado por resolución Nº 0989-2022-UANCV-CU-R, quien asumirá como asesor de la propuesta de investigación según el área de grado.

Estando, con la opinión favorable de la propuesta de investigación del Comité de investigación de la Facultad de Ingeniería de Sistemas y en concordancia con el reglamento interno de trabajos de investigación conducente a grados y títulos, aprobado con resolución Nº 0294-2023-UANCV-CU-R y en mérito al art. 25 del reglamento, con fines de obtención de obtención de Grados Académicos y Títulos Profesionales y en uso a las atribuciones que le concede la ley Universitaria Nº 30220, Ley de creación de la UANCV Nº 23738 y modificatoria Nº 24661, y el estatuto de la UANCV, el Decano y Director de la Facultad de Ingeniería de Sistemas.

RESUELVE:

ARTÍCULO PRIMERO.- APROBAR, la **PROPUESTA DE INVESTIGACION**, presentada por el (la) Bachiller: **ABELARDO IDME MAMANI**, para optar el Título Profesional de **INGENIERO DE SISTEMAS** con el tema titulado: **DESARROLLO DE UN SISTEMA DE INVENTARIO PARA OPTIMIZAR LA GESTIÒN DE ALMACÈN EN LA EMPRESA INTIQA HOTEL PUNO 2025**, correspondiente a la línea de investigación, **CIENCIA DE LOS ORDENADORES.**

La misma que deberá proceder con la ejecución de la propuesta de investigación aprobado de acuerdo a los establecido en el Reglamento Interno de Trabajos de Investigación Conducente a Grados y Títulos, con fines de obtención de Grados Académicos y Títulos Profesionales

ARTÍCULO SEGUNDO. - RECONOCER como **ASESOR DE INVESTIGACION** al (a la) docente **Dr. Paul Mamani Tisnado.**

ARTÍCULO TERCERO. - DISPONER que, la Unidad de Investigación, Responsables del Comité de Investigación de la Facultad de Ingeniería de Sistemas y el Director de la Escuela profesional, **INGENIERIA DE SISTEMAS**, quedan encargados del cumplimiento de la presente Resolución.

Regístrese, Comuníquese y Archívese.

UNIVERSIDAD ANDINA "NESTOR CACERES VELASQUEZ"
DECANO
Dr. Rodolfo Fredy Arpasi Chuna
DECANO (e)

Cc.
Arch 2025

OFICINA DE INVESTIGACIÓN
Tesis Publicada con autorización del autor
Central Telefónica: (051) 321142 - Juliaca - Puno - Perú - Pag. Web: www.uancv.edu.pe
http://repositorio.uancv.edu.pe/

TESIS UANCV
VICERRECTORADO DE INVESTIGACIÓN
"OFICINA DE INVESTIGACIÓN"

ABELARDO IDME MAMANI

DESARROLLO DE UN SISTEMA DE INVENTARIO PARA OPTIMIZAR LA GESTIÓN DE ALMACÉN EN LA EMPRESA INTI...

Trabajos de investigación para Repositorio

Detalles del documento

Identificador de la entrega
trn:oid:::13016:617576666

Fecha de entrega
26 ago 2026, 23:20 GMT-5

Fecha de descarga
26 ago 2026, 23:34 GMT-5

Nombre del archivo
T036_42169500_T.docx

Tamaño del archivo
56.5 MB

118 páginas

21.222 palabras

144.011 caracteres

turnitin Página 1 de 131 - Portada Identificador de la entrega trn:oid:::13016:617576666
OFICINA DE INVESTIGACIÓN
Tesis Publicada con autorización del autor
http://repositorio.uancv.edu.pe/

TESIS UANCV

VICERRECTORADO DE INVESTIGACIÓN
"OFICINA DE INVESTIGACIÓN"

21% Similitud general

El total combinado de todas las coincidencias, incluidas las fuentes superpuestas, para ca...

Fuentes principales

13% Fuentes de Internet
9% Publicaciones
17% Trabajos entregados (trabajos del estudiante)

Marcas de integridad

N.° de alertas de integridad para revisión

Los algoritmos de nuestro sistema analizan un documento en profundidad para buscar inconsistencias que permitirían distinguirlo de una entrega normal. Si advertimos algo extraño, lo marcamos como una alerta para que pueda revisarlo.

Una marca de alerta no es necesariamente un indicador de problemas. Sin embargo, recomendamos que preste atención y la revise.

OFICINA DE INVESTIGACIÓN
Tesis Publicada con autorización del autor
http://repositorio.uancv.edu.pe/

TESIS UANCV

VICERRECTORADO DE INVESTIGACIÓN
"OFICINA DE INVESTIGACIÓN"

Metadatos complementarios

TÍTULO DE LA TESIS	
DESARROLLO DE UN SISTEMA DE INVENTARIO PARA OPTIMIZAR LA GESTIÓN DE ALMACÉN EN LA EMPRESA INTIQA HOTEL PUNO 2025	
Datos de autor	
Nombres y apellidos	ABELARDO IDME MAMANI
Tipo de documento de identidad	DNI
Número de documento de identidad	42169500
URL de ORCID	https://orcid.org/0009-0009-4980-3297
Datos de asesor	
Nombres y apellidos	PAUL MAMANI TISNADO
Tipo de documento de identidad	DNI
Número de documento de identidad	01314987
URL de ORCID	https://orcid.org/0000-0002-0287-7143
Datos del jurado	
Presidente del jurado	
Nombres y apellidos	JUAN CARLOS HERRERA MIRANDA
Tipo de documento	DNI
Número de documento de identidad	29606930
Miembro del jurado 1	
Nombres y apellidos	JESUS ESTEBAN CASTILLO MACHACA
Tipo de documento	DNI
Número de documento de identidad	01323821
Miembro del jurado 2	
Nombres y apellidos	ADWAR RANULFO SANCHEZ CARREON
Tipo de documento	DNI
Número de documento de identidad	02064066

OFICINA DE INVESTIGACIÓN
Tesis Publicada con autorización del autor
http://repositorio.uancv.edu.pe/

Datos de investigación	
Línea de investigación	CIENCIAS DE LOS ORDENADORES - P24
Grupo de investigación	No aplica.
Agencia de financiamiento	Sin financiamiento.
Ubicación geográfica de la investigación	País: Perú Departamento: Puno Provincia: Puno Distrito: Puno Coordenadas: Latitud: -15.473452 Longitud: -70.1284417 https://maps.app.goo.gl/4n56FRemmrL9XQb48 
Año o rango de años en que se realizó la investigación	Septiembre 2025 – Julio 2026
URL de disciplinas OCDE https://concytec-pe.github.io/Peru-CRIS/vocabularios/ocde_ford.html	**Ingeniería de sistemas y comunicaciones** https://purl.org/pe-repo/ocde/ford#2.02.04 **Telecomunicaciones** https://purl.org/pe-repo/ocde/ford#2.02.05

UNIVERSIDAD ANDINA "NÉSTOR CÁCERES VELÁSQUEZ"
FACULTAD DE INGENIERÍA DE SISTEMAS - JULIACA
DIRECCIÓN

Dr. Juan Carlos Herrera Miranda
DIRECTOR (e)
Unidad de Investigación FIS

TESIS UANCV
TESIS UANCV
VICERRECTORADO DE INVESTIGACIÓN
"OFICINA DE INVESTIGACIÓN"

DECLARACIÓN DE AUTENTICIDAD Y RESPONSABILIDAD

Yo ABELARDO IDME MAMANI, identificado con DNI Nro. 42169500 en mi condición de egresado de:

☒ **Escuela Profesional**

☐ **Programa de Segunda Especialidad,**

☐ **Programa de Maestría o Doctorado**

INGENIERÍA DE SISTEMAS,

informo que he elaborado el/la ☒ **Tesis** o ☐ **Trabajo de Investigación,** ☐ **Trabajo Académico** denominada:

DESARROLLO DE UN SISTEMA DE INVENTARIO PARA OPTIMIZAR LA GESTIÓN DE ALMACÉN EN LA EMPRESA INTIQA HOTEL PUNO 2025

Asesorado por: Dr. PAUL MAMANI TISNADO

Es un tema original.

Declaro que el presente trabajo de tesis es elaborado por mi persona y **no existe plagio/copia** de ninguna naturaleza, en especial de otro documento de investigación (tesis, revista, texto, congreso, o similar) presentado por persona natural o jurídica alguna ante instituciones académicas, profesionales, de investigación o similares, en el país o en el extranjero.

Dejo constancia que las citas de otros autores han sido debidamente identificadas en el trabajo de investigación, por lo que no asumiré como suyas las opiniones vertidas por terceros, ya sea de fuentes encontradas en medios escritos, digitales o Internet.

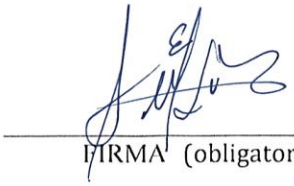
Asimismo, ratifico que soy plenamente consciente de todo el contenido de la tesis y asumo la responsabilidad de cualquier error u omisión en el documento, así como de las connotaciones éticas y legales involucradas.

El incumplimiento de lo declarado da lugar a responsabilidad del declarante, en consecuencia; a través del presente documento asumo frente a terceros, la Universidad Andina Néstor Cáceres Velásquez y/o la Administración Pública toda responsabilidad que pueda derivarse por el trabajo final presentado. Lo señalado incluye responsabilidad pecuniaria incluido el pago de multas u otros por los daños y perjuicios que se ocasionen.

Juliaca 21 de JULIO del 2026



Firma del Asesor
(obligatoria)



FIRMA (obligatoria)



Huella

OFICINA DE INVESTIGACIÓN
Tesis Publicada con autorización del autor
http://repositorio.uancv.edu.pe/
http://repositorio.uancv.edu.pe/

TESIS UANCV
VICERRECTORADO DE INVESTIGACIÓN
"OFICINA DE INVESTIGACIÓN"

DEDICATORIA

A Dios; a mi adorable madre, Francisca por su amor incondicional, su fortaleza y su ejemplo de perseverancia, a mi hijo Andrei por su dulzura y a la memoria de mi hermano y padre Ruben, Andres; a toda mi familia, Romulo, Juan, Lisandro, José, quienes me enseñaron que los sueños se alcanzan con esfuerzo y fe, Este logro también son suyos.

OFICINA DE INVESTIGACIÓN
Tesis Publicada con autorización del autor
http://repositorio.uancv.edu.pe/

TESIS UANCV
VICERRECTORADO DE INVESTIGACIÓN
"OFICINA DE INVESTIGACIÓN"

AGRADECIMIENTO

Agradezco profundamente al Dr. Paul Mamani Tisnado, mi asesor, por su orientación académica, su compromiso y su constante apoyo durante el desarrollo de esta tesis. Su experiencia y guía fueron esenciales para culminar con éxito este proyecto.

OFICINA DE INVESTIGACIÓN
Tesis Publicada con autorización del autor
http://repositorio.uancv.edu.pe/

TESIS UANCV
VICERRECTORADO DE INVESTIGACIÓN
"OFICINA DE INVESTIGACIÓN"

INDICE GENERAL

DEDICATORIA ..... **iii**

AGRADECIMIENTO ..... **iv**

INDICE GENERAL ..... **v**

INDICE DE TABLAS ..... **ix**

INDICE DE FIGURAS ..... **ix**

RESUMEN ..... **x**

ABSTRACT ..... **xi**

INTRODUCCIÓN ..... **xii**

CAPÍTULO I

EL PROBLEMA DE INVESTIGACIÓN

1.1. Análisis De La Situación Problemática 1

1.2. Formulación del Problema 3

1.2.1. Problema General 3

1.3.1. Objetivo General 4

OFICINA DE INVESTIGACIÓN
Tesis Publicada con autorización del autor
http://repositorio.uancv.edu.pe/

1.3.2. Objetivos Específicos 4

1.4. Hipótesis de la Investigación 4

1.4.1. Hipótesis General 4

1.5. Justificación de la Investigación 5

1.5.1. Justificación Teórica 5

1.5.2. Justificación Práctica 5

1.5.3. Justificación Metodológica 6

1.6. Operacionalización de Variables 7

CAPÍTULO II

MARCO TEÓRICO

2.1. Antecedentes de la Investigación 9

2.1.1. Antecedentes Internacionales 9

2.1.2. Antecedentes Nacionales 12

2.1.3. Antecedentes Locales 12

2.2. Bases teóricas de la Variable Independiente Sistema de Inventario en Python 14

2.2.1. Arquitectura de Software Local vs. Sistemas ERP 14

2.2.2. Control de Acceso Basado en Roles (RBAC) y Seguridad de la Información 16

2.2.3. El Método Kardex Como Motor Transaccional 17

2.2.4. Visualización de Datos Para la Toma de Decisiones (Dashboards) 18

2.3. Bases teóricas de la Variable Dependiente Gestión de Almacén 20

2.3.1. Gestión de Almacenes en el Sector Hotelero..... 20

2.3.2. Trazabilidad y Asignación por Centros de Costos..... 21

2.4. Marco Conceptual..... 22

CAPÍTULO III

METODOLOGÍA DE LA INVESTIGACIÓN

3.1. Enfoque de la Investigación 25

3.2. Tipo de Investigación..... 25

3.3. Nivel de Investigación 26

3.4. Diseño de la Investigación 26

3.5. Método de Investigación..... 26

3.6. Población y Muestra 27

3.6.1. Población..... 27

3.6.2. Muestra 27

3.8. Validez y Confiabilidad del Instrumento..... 30

CAPÍTULO IV

DESARROLLO DEL SISTEMA Y ANÁLISIS DE RESULTADOS

4.1. Metodología de Desarrollo de Software Aplicada 32

4.2. Arquitectura y Diseño de la Solución Tecnológica 33

4.3. Diseño Lógico de la Base de Datos 34

4.4. Evidencias de Implementación del Sistema 35

4.4.1. Módulo de Seguridad y Control de Acceso (RBAC) 35

4.4.2. Módulo de Trazabilidad por Centros de Costos 37

4.4.3. Módulo de Visualización de Datos (Dashboards Gerenciales) 38

4.4.4. Módulo de Alertamiento Temprano (Stock Mínimo) 39

4.4.5. Módulo de Registro Kardex Permanente 41

4.5. Análisis e Interpretación de Resultados de la Encuesta 42

4.5.1. Contraste de Métricas de Eficiencia Operativa (Evaluación Técnica del Sistema) 45

4.6. Discusión de Resultados 47

4.7. Contrastación de Hipótesis 48

CAPÍTULO V

CONCLUSIONES, RECOMENDACIONES Y TRABAJO FUTURO

5.1. Conclusiones 51

5.2. Recomendaciones 53

5.3. Trabajo Futuro 54

REFERENCIAS BIBLIOGRÁFICAS..... 55

INDICE DE TABLAS

Tabla 1. Operacionalización de las variables de investigación..... 7

Tabla 2. Distribución de frecuencia de la evaluación del sistema de inventario 42

Tabla 3. Comparativo de indicadores de eficiencia operativa (Pre y Post Implementación). 45

INDICE DE FIGURAS

Figura 1. Modelo Entidad-Relación (MER) de la base de datos Kardex.db. 35

Figura 2. Interfaz de autenticación de ususarios meiante hash criptográfico. 36

Figura 3. Interfaz principal del sistema mostrando el control de acceso basado en roles 37

Figura 4. Módulo de egreso de inventario mostrado la asignación obligatoria por centros. 38

Figura 5. Dashboard de inteligencia de negocios mostrando distribución proporciona 39

Figura 6. Interfaz del módulo de alertamiento temprano comparando saldos reales 40

Figura 7. Motor transaccional desplegando el registro histórico kardex con cálculo..... 41

RESUMEN

En el sector hotelero, el almacén no es un simple depósito, sino el pilar financiero que sostiene la calidad del servicio al huésped. Sin embargo, en la mayoría de pequeñas y medianas empresas de hospedaje, la gestión logística sigue anclada en procesos manuales. El presente estudio tuvo como objetivo desarrollar un sistema de inventario en Python, basado en trazabilidad por centros de costos, para optimizar la gestión de almacén y mitigar las pérdidas de insumos en Intiqa Hotel Puno. La investigación abordó un enfoque cuantitativo, de tipo aplicado y diseño no experimental, trabajando con la totalidad de la población operativa del almacén mediante un censo. El sistema desarrollado bajo una arquitectura local (Python, Tkinter y SQLite) integró funcionalidades críticas como control de acceso por roles (RBAC), un motor transaccional Kardex y cuadros de mando (Dashboards) para visualizar costos por área. Los resultados demostraron una reducción del 99% en los tiempos de consulta, la eliminación total de errores matemáticos manuales y una percepción de satisfacción del usuario de 4.8 sobre 5.0. Se concluye que implementar soluciones tecnológicas a medida, de bajo costo y sin dependencia de internet, es una alternativa viable y superior para blindar la operatividad hotelera frente a los costosos sistemas ERP comerciales.

Palabras clave: Sistema de inventario, gestión de almacén, trazabilidad, centros de costos, Python, inteligencia de negocios.

TESIS UANCV
VICERRECTORADO DE INVESTIGACIÓN
"OFICINA DE INVESTIGACIÓN"

ABSTRACT

In the hospitality sector, a warehouse is not just a storage room, but the financial pillar that sustains guest service quality. However, in most small and medium-sized lodging businesses, logistics management remains anchored in manual processes. This study aimed to develop a Python-based inventory system, centered on cost center traceability, to optimize warehouse management and mitigate inventory losses at Intiqa Hotel Puno. The research adopted a quantitative, applied approach with a non-experimental design, working with the entire warehouse operational population through a census. The system, developed under a local architecture (Python, Tkinter, and SQLite), integrated critical features such as Role-Based Access Control (RBAC), a Kardex transactional engine, and dashboards to visualize costs per area. The results demonstrated a 99% reduction in query times, the total elimination of manual mathematical errors, and a user satisfaction perception of 4.8 out of 5.0. It is concluded that implementing tailored, low-cost technological solutions without internet dependency is a viable and superior alternative to safeguard hotel operations compared to expensive commercial ERP systems.

Keywords: Inventory system, warehouse management, traceability, cost centers, Python, business intelligence.

OFICINA DE INVESTIGACIÓN
Tesis Publicada con autorización del autor
http://repositorio.uancv.edu.pe/

INTRODUCCIÓN

Para cualquier establecimiento de hospedaje, el correcto abastecimiento de insumos es tan vital como la atención al cliente. Cuando un huésped pide una cerveza en el bar o un café en la mañana, rara vez se pregunta de dónde salió ese producto, pero detrás de esa operación existe una compleja red logística que, si falla, genera pérdidas económicas silenciosas. En el caso de Intiqa Hotel Puno, esta realidad era evidente, la administración del almacén dependía de cuadernos y hojas de Excel que generaban "islas de información". El problema más crítico no era solo la lentitud, sino la Invisibilidad logística; es decir, cuando faltaba stock, nadie sabía exactamente si el desperdicio se originaba en la cocina, en la barra o en el área de pisos. Esta falta de trazabilidad creaba lo que en la logística se conoce como "Pérdidas de dinero que no se podían rastrear", afectando directamente la rentabilidad del negocio.

Frente a este diagnóstico, la solución del mercado tradicional apunta a adquirir sistemas ERP (como SAP u Oracle). No obstante, para una PYME hotelera regional, esta opción representa un costo de capital prohibitivo y exige una infraestructura tecnológica que no siempre es accesible. Es aquí donde nace la pregunta que guía esta investigación: ¿Es posible lograr un control de inventario de clase mundial, con auditoría y trazabilidad por áreas, sin depender de internet y sin pagar licencias onerosas?

La respuesta demostrada en este trabajo es un rotundo sí. A través de la ingeniería de software, se diseñó e implementó un sistema en Python que Permite que un negocio pequeño tenga un control de inventario avanzado sin pagar costos de grandes empresas. Utilizando tecnologías de código abierto (Tkinter para la interfaz y SQLite para la base de datos), se logró crear una

herramienta autónoma que no requiere internet. El sistema obliga a registrar a qué área va cada insumo, quién lo retiró y a qué costo, Convirtiendo los registros del almacén en gráficos fáciles de entender para el gerente que permiten a la gerencia tomar decisiones proactivas en segundos.

El presente trabajo se estructura en cinco capítulos. El primero contextualiza el problema; el segundo sustenta teóricamente la propuesta; el tercero detalla la metodología aplicada; el cuarto presenta el desarrollo del software, las evidencias de implementación y la contrastación empírica de las hipótesis; y, finalmente, el quinto presenta las conclusiones, recomendaciones y las futuras líneas de evolución de este sistema.

OFICINA DE INVESTIGACIÓN
Tesis Publicada con autorización del autor
http://repositorio.uancv.edu.pe/

CAPÍTULO I

EL PROBLEMA DE INVESTIGACIÓN

1.1. Análisis de la situación problemática

Para cualquier hotel, el almacén no es solo un depósito; es un pilar crítico para sostener las finanzas del negocio. Al diagnosticar a Intiqa Hotel en Puno, se hizo evidente que sus procesos logísticos tenían fallas estructurales. Estas deficiencias impactaban de forma directa en la calidad del servicio.

Actualmente, la administración del almacén se sustenta en métodos analógicos, centralizados principalmente en el registro manual en cuadernos y en el uso de herramientas ofimáticas no integradas (hojas de cálculo Excel). Esta dependencia de registros no estandarizados hace que los datos queden separados, impidiendo que el almacén tenga una vista unificada y actualizada de sus productos.

El problema más grave que se identificó es la falta de trazabilidad por áreas de consumo (centros de costos). Es decir, no se sabe si los insumos que salen van a cocina, barra o pisos. Como los egresos de bienes de alta rotación (cervezas, aguas, leche, café, yogurt, jamón, queso) no se anotaban a nombre de un área o un responsable, Se permitieron errores y pérdidas sin que nadie se diera cuenta. Todo esto conlleva a la pérdida de activos sin justificación ni registro, conocidas comúnmente como mermas. Al no tener claro qué área operativa consume más recursos, la gerencia se ve limitada a la hora de diseñar estrategias para controlar o reducir sus costos.

Este tipo de descontrol no es un caso aislado dentro del sector hotelero y gastronómico del Perú, (Da Silva Rodriguez, 2020) en una investigación sobre la gestión de un almacén en un restaurante de Lima, detalló cómo el hecho de no rotar adecuadamente los insumos perecederos y la falta de un registro minucioso derivaban en mermas graves y pérdidas directas de dinero. En su estudio se pudo notar que, a falta de un método de control estricto, los trabajadores terminaban entregando insumos nuevos mientras los antiguos se vencían en las estanterías. Esta falla de trazabilidad se relaciona directamente con lo que ocurre en Intiqa Hotel, ya que, al no existir un sistema digital que respalde las salidas, resulta imposible detectar qué área de la operación está generando la mayor cantidad de desperdicios.

Esta situación de descontrol logístico no es exclusiva de los restaurantes, sino un problema generalizado en el hospedaje regional. En el ámbito hotelero peruano, (Peralta Ranilla y Ranilla Alvarez, 2022) analizaron el control de inventarios en el Hotel La Posada de Ligarte en Arequipa, donde evidenciaron que la falta de estrategias y registros adecuados generaba incongruencias entre el stock físico y los registros contables. Este antecedente demuestra que los hoteles de escala pequeña y mediana en el país siguen enfrentando dificultades para proteger sus insumos, lo cual refuerza la necesidad de que Intiqa Hotel cuente con una herramienta tecnológica que automatice y asegure estos procesos.

Visto desde la auditoría operativa, no tener un sistema digital para respaldar las transacciones Impide saber quién se llevó los productos. Y si a esto le sumamos que no existían contraseñas ni roles de usuario, era imposible saber quién autorizaba un movimiento y quién lo ejecutaba. La responsabilidad se diluía por completo.

Finalmente, la carencia de Alertas automáticas de stock bajo (como notificaciones de stock mínimo) y la imposibilidad de emitir reportes gerenciales dinámicos obligan a que la toma de

decisiones se realice de manera reactiva y empírica, generando retrasos operativos (como la búsqueda manual de insumos que demanda varios minutos) y fricciones administrativas ante la falta de visibilidad del estado real del almacén.

En este contexto, se evidencia la necesidad imperativa de desarrollar un sistema de información que automatice, integre y trace los procesos de gestión de almacén, transformando el modelo actual reactivo y descontrolado en un sistema ordenado, donde se registra todo y se evitan pérdidas de recursos hoteleros.

1.2. Formulación del problema

1.2.1. Problema General

¿De qué manera el desarrollo e implementación de un sistema de inventario en Python, basado en trazabilidad por centros de costos, contribuirá a optimizar la gestión de almacén y mitigar las pérdidas no registradas de insumos en Intiqa Hotel Puno, durante el año 2025?

1.2.2. Problemas Específicos

- PE1. ¿Cuál es el nivel de incidencia que tiene la actual ausencia de trazabilidad y de registros desagregados por áreas de consumo (cocina, barra, pisos) en la pérdida de bienes de alta rotación (bebidas y alimentos) de Intiqa Hotel Puno?
- PE2. ¿Qué características funcionales y de seguridad (control de acceso por roles, vinculación de vales de salida con facturas de ingreso y alertas de stock mínimo) debe integrar el sistema de información en Python para garantizar la auditoría y el control transaccional del almacén?
- PE3. ¿De qué manera la automatización de los reportes estadísticos por área de consumo y la centralización de la información en el sistema propuesto mejorarán la eficiencia en los tiempos de consulta y la toma de decisiones gerenciales en Intiqa Hotel Puno?

1.3. Objetivos de la investigación

1.3.1. Objetivo General

Desarrollar e implementar un sistema de inventario en Python, basado en trazabilidad por centros de costos, para optimizar la gestión de almacén y mitigar las pérdidas no registradas de insumos en Intiqa Hotel Puno, durante el año 2025.

1.3.2. Objetivos Específicos

- OE1. Determinar el nivel de incidencia que tiene la actual ausencia de trazabilidad y de registros desagregados por áreas de consumo (cocina, barra, pisos) en la pérdida de bienes de alta rotación (bebidas y alimentos) de Intiqa Hotel Puno.
- OE2. Diseñar e implementar el sistema de información en Python que integre control de acceso por roles, vinculación de vales de salida con facturas de ingreso y alertas de stock mínimo, para garantizar la auditoría y el control transaccional del almacén.
- OE3. Evaluar la mejora en la eficiencia de los tiempos de consulta y la toma de decisiones gerenciales, a partir de la automatización de reportes estadísticos por área de consumo y la centralización de la información en el sistema propuesto.

1.4. Hipótesis de la investigación

1.4.1. Hipótesis General

El desarrollo e implementación de un sistema de inventario en Python, estructurado bajo el enfoque de trazabilidad por centros de costos, se relaciona positivamente con la optimización de la gestión de almacén y la mitigación de las pérdidas no registradas de insumos en Intiqa Hotel Puno, durante el año 2025.

1.4.2. Hipótesis Específicas

- HE1. La ausencia actual de trazabilidad por áreas de consumo se asocia de manera directa con la pérdida injustificada de bienes de alta rotación (bebidas y alimentos) en Intiqa Hotel Puno.
- HE2. El sistema de información en Python, al integrar control de acceso por roles y la vinculación digital de vales de salida con facturas de ingreso, permite dejar un registro claro de quién hizo cada movimiento que favorece la auditoría y el control transaccional del almacén.
- HE3. La centralización de la información y la generación automatizada de reportes estadísticos desagregados por áreas de consumo se relacionan con una mayor eficiencia operativa (reducción de tiempos de consulta) y facilitan la toma de decisiones gerenciales en Intiqa Hotel Puno.

1.5. Justificación de la investigación

1.5.1. Justificación Teórica

El presente estudio aporta al cuerpo de conocimientos de la Ingeniería de Software y la logística operativa al demostrar que conceptos avanzados de la cadena de suministro, como la "trazabilidad por centros de costos", no son exclusivos de corporaciones transnacionales. A nivel teórico, este trabajo comprueba que no se requieren grandes inversiones para aplicar una gestión de almacenes moderna. El uso de herramientas de código abierto y arquitecturas ligeras, específicamente Python y SQLite, resultó suficiente para llevar a la práctica estos principios. En este sentido, el proyecto evidencia cómo la teoría de operaciones puede adaptarse sin complicaciones al entorno tecnológico y económico de los pequeños hoteles en el Perú. Así, la tesis aporta una base documental que respalda el desarrollo de software local como una alternativa real y efectiva para optimizar este tipo de procesos.

1.5.2. Justificación Práctica

Esta investigación tiene un impacto directo y palpable en las ganancias de Intiqa Hotel Puno. Al poner en marcha el software, se reduce al instante la pérdida de productos sin justificación, como bebidas, alimentos y otros insumos clave. Esto se logra porque la herramienta crea un registro seguro que permite identificar con precisión qué área de consumo gasta más.

Ahora bien, el mayor aporte práctico de este trabajo es demostrar que la tecnología es económicamente viable para estos negocios. El proyecto deja claro que los hoteles pequeños y medianos no necesitan gastar grandes sumas de dinero en licencias de sistemas privados (como SAP u Oracle) para lograr un control de almacén sólido. El programa creado en Python ofrece las mismas funciones indispensables, trazabilidad, control de usuarios por roles y reportes para la gerencia. Todo esto con un costo de implementación casi nulo, ajustado a las necesidades reales del hotel y logrando que las consultas de inventario que antes tomaban varios minutos ahora se resuelvan en cuestión de segundos.

1.5.3. Justificación Metodológica

Desde el punto de vista metodológico, el proceso seguido en esta investigación sirve como un modelo replicable para otros proyectos. El estudio comenzó identificando las fallas de los procesos manuales, luego se pasó a diseñar una arquitectura de software segura y, en la etapa final, se validaron los resultados utilizando indicadores de eficiencia operativa. El presente trabajo servirá como guía empírica y modelo de referencia para futuros ingenieros de sistemas que deban intervenir en la realidad logística de otras organizaciones del sector turismo y hospedaje en la región de Puno. Además, demuestra que la Investigación Aplicada en sistemas de información no depende de herramientas comerciales costosas, sino del análisis ingenieril del problema para proponer soluciones tecnológicas oportunas, escalables y de bajo costo.

1.6. Operacionalización de variables

Tabla 1

Operacionalización de las variables de investigación

VARIABLE	DIMENSIÓN	INDICADORES	ESCALA / ÍNDICE	INSTRUMENTO DE MEDICIÓN
Variable Independiente: Sistema de Inventario en Python	D1. Control de Acceso y Seguridad	1. Cantidad de perfiles de usuario (Administrador/Operador) configurados. 2. Nivel de restricción de acceso a módulos críticos.	Nominal / Categórica	Ficha de observación del sistema / Ficha de validación técnica.
	D2. Gestión Transaccional y Trazabilidad	1. Porcentaje de vales de salida vinculados a un área de consumo (centro de costo). 2. Porcentaje de ingresos registrados con número de factura de respaldo.	Razón (Porcentaje %)	Revisión de base de datos (SQLite) / Ficha de observación del sistema.
	D3. Automatización e Inteligencia de Datos	1. Tiempo promedio (en segundos) de generación de reportes estadísticos por área. 2. Número de alertas de stock mínimo emitidas por el sistema.	Razón / Intervalo	Cronometraje de tareas (Registros previos vs. registros del sistema actual)

TESIS UANCV
VICERRECTORADO DE INVESTIGACIÓN
"OFICINA DE INVESTIGACIÓN"

VARIABLE	DIMENSIÓN	INDICADORES	ESCALA / ÍNDICE	INSTRUMENTO DE MEDICIÓN
Variable Dependiente: Gestión de Almacén	D1. Control de Stock y Mitigación de Pérdidas	1. Variación porcentual de mermas no registradas en bienes de alta rotación (bebidas/alimentos). 2. Nivel de coincidencia entre el stock físico y el stock del sistema.	Razón (Porcentaje %)	Comparativo documentario de inventarios (Registros históricos vs. estado actual del sistema)/ Encuesta.
	D2. Eficiencia Operativa	1. Tiempo promedio (en segundos) de consulta de un insumo específico. 2. Tiempo total (en horas) empleado en el cuadre de inventario mensual.	Razón (Segundos / Horas)	Análisis documentario y cronometraje (Registros manuales previos vs. registros del sistema actual)
	D3. Toma de Decisiones Gerenciales	1. Frecuencia de uso de los reportes de gastos por área para toma de decisiones. 2. Nivel de percepción de fiabilidad de la información por parte de la gerencia.	Likert (1 a 5)	Cuestionario de encuesta aplicado a la gerencia / personal de almacén.

Nota. Elaboración propia con base a los objetivos e hipótesis de la investigación (2025).

OFICINA DE INVESTIGACIÓN
Tesis Publicada con autorización del autor
http://repositorio.uancv.edu.pe/

CAPÍTULO II

MARCO TEÓRICO

2.1. Antecedentes de la Investigación

2.1.1. Antecedentes Internacionales

A nivel global, la gestión de inventarios en el sector hotelero se enfrenta a un problema oculto, las pérdidas de insumos no se detectan a tiempo porque no hay una separación clara de gastos por áreas. Sobre este punto, Christopher (2023) advierte en su estudio sobre cadenas de suministro que la mera digitalización de un almacén no sirve de mucho si la información no se segmenta. Según su perspectiva, si una empresa no logra desagregar sus costos logísticos por centros de costo (por ejemplo, saber exactamente cuánto le cuesta la operación de la barra frente a la cocina), opera bajo un modelo donde las pérdidas se diluyen y es imposible determinar qué área está generando ineficiencia. Esta premisa teórica es la base de nuestra investigación.

Al revisar cómo otros han intentado solucionar esto a nivel técnico, vemos dos caminos distintos. Por un lado está Dueñas Feuerbach (2022), quien trabajó con una empresa industrial en España. Él logró solucionar el problema de la información desordenada creando tableros visuales (dashboards) para que la gerencia pudiera ver los datos de forma rápida. Lo que vale la pena destacar es que comprobó que la visualización gráfica agiliza las decisiones. La desventaja de su

propuesta es que requería servidores en la nube y licencias con un costo elevado. Ante esto, surge una duda lógica para las pequeñas empresas de la región: ¿de qué manera se pueden obtener estos mismos beneficios si no se cuenta con el presupuesto para pagar infraestructuras complejas?

Esa limitación fue justamente lo que intentó resolver Andi Tapuy (2025) en Ecuador. Su caso de estudio se centró en una tienda ubicada en una zona con una mala conexión a internet. La autora solucionó este inconveniente desarrollando un sistema basado en SQLite, una base de datos local que funciona directamente en la computadora sin necesidad de estar conectada a la red. Los resultados que obtuvo fueron positivos. Con esta experiencia, demostró que no es indispensable ser una gran corporación para implementar un control de inventario eficiente, lo cual resulta de gran utilidad en lugares donde la señal de internet es deficiente o nula.

Nuestra investigación toma estos dos casos internacionales como base para justificar el sistema de Intiqa Hotel. Tomamos la filosofía de Christopher para obligar a separar los costos, tomamos la parte positiva de Dueñas (los dashboards para visualizar gastos) y la parte práctica de Andi (el SQLite offline para no depender de internet), pero fuimos un paso más allá. La gran diferencia es que nosotros sí logramos cerrar el ciclo, el sistema obliga a registrar a qué área va cada insumo, resolviendo esa "invisibilidad logística" que ambos autores dejaron pendiente.

De manera complementaria, en el contexto iberoamericano, Mejia Nuñez (2025) desarrolló un proyecto enfocado en la optimización de la gestión de inventarios mediante análisis de ventas y modelado predictivo con Python, utilizando datos de Amazon. La autora concluyó que el uso de librerías como Matplotlib permite predecir patrones de diferencia de demanda y transformar

grandes volúmenes de datos crudos en información estratégica para la toma de decisiones, reduciendo problemas de desabastecimiento.

Relación con el estudio: La presente investigación y el trabajo de Mejia Nuñez (2025) comparten la elección del ecosistema Python y el uso de Matplotlib para diseñar los Dashboards. Esto confirma que dichas herramientas resultan muy adecuadas para el manejo de inventarios. No obstante, la diferencia principal radica en el enfoque utilizado. Mientras que la autora colombiana se centró en un modelo predictivo (Machine Learning) trabajando con grandes volúmenes de datos en la nube, este proyecto se enfoca en el control transaccional y la auditoría (Kardex y RBAC) en un entorno físico local. De este modo, se demuestra que la combinación de Python y SQLite es suficiente para optimizar un almacén hotelero sin depender de una conexión a internet.

Desde la perspectiva de la ingeniería de software ágil, Lorenzo Guerrero (2025), desarrolló un sistema de gestión hotelero en la UNAM. Su propuesta se basó en tecnologías web (Python, HTML/CSS y MySQL) aplicando la metodología Scrum. En su investigación, la autora concluye que el uso de estas arquitecturas web permite mejorar las operaciones, proteger la información y agilizar la toma de decisiones por parte de la gerencia.

Relación con el estudio: Al igual que Lorenzo Guerrero (2025) esta tesis eligió Python como base lógica del programa, lo que demuestra la versatilidad de este lenguaje para resolver problemas de gestión hotelera a nivel internacional. Sin embargo, hay diferencias notables en la tecnología empleada. La autora mexicana decidió usar una arquitectura web (Python y MySQL) bajo el marco Scrum, lo cual implica pagar por servidores (hosting) y depender de internet. Este

proyecto, por el contrario, plantea una solución distinta: para un hotel pequeño en la región, resulta más viable usar una arquitectura local (Python, Tkinter y SQLite) con la metodología RAD. De esta forma, se logra el mismo nivel de optimización y seguridad (RBAC), pero sin generar gastos de implementación, convirtiéndola en una alternativa mucho más económica.

2.1.2. Antecedentes Nacionales

Si miramos lo que pasa en otros hoteles del país, vemos que Intiqa Hotel no es un caso alineado. Peralta Ranilla y Ranilla Alvarez (2022) evaluaron el caso del Hotel La Posada de Ligarte en la ciudad de Arequipa. Le aplicaron un cuestionario a 20 trabajadores para saber cómo manejaban el almacén. El dato que dio el cuestionario fue preocupante, el 80% admitió que su control de inventario es "regular" y cometían errores al recibir insumos.

Por su parte, Alarcon Turpo y Quispe Cubas (2023) llevaron el método Kardex a una minera en Puno. Lograron medir el impacto real, aumentaron la productividad un 20.17% y bajaron los tiempos de proceso en un 28.39%.

Sin embargo, ninguno de los dos autores logró separar los insumos por áreas de consumo (como cocina, barra o pisos). Ellos demostraron que el método Kardex funciona, pero su aplicación se dio en un entorno minero, no en la hotelería. Esa es la falla estructural que el presente proyecto buscó resolver. Se tomó el modelo Kardex que ellos validaron, se adaptó a la dinámica de un hotel y se configuró el sistema para que calcule el gasto financiero de cada área. Esta mejora le da soporte a la teoría de Christopher (2023) si los costos no se separan por áreas, el dinero simplemente desaparece sin dejar rastro.

2.1.3. Antecedentes Locales

Fuera de los ejemplos internacionales y nacionales, la región de Puno también ofrece evidencia directa sobre la utilidad del software en el control logístico. En la Universidad Andina. Ticona Vargas (2025), presentó una tesis donde desarrolló un sistema web para centralizar el inventario de una empresa en Juliaca. Mediante un diseño preexperimental, el autor logró demostrar que juntar la información en un solo lugar ayuda a corregir las diferencias de stock de manera automática, lo cual agiliza la toma de decisiones. Este antecedente es de gran ayuda como respaldo institucional. Al comprobar que un egresado de la misma universidad pudo optimizar inventarios usando Python, el sistema de arquitectura local planteado en esta tesis se presenta como una evolución lógica para el entorno empresarial de la zona.

En otra investigación local, Parisuaña Cusilayme (2024), analizó el caso de una planta procesadora de lácteos en Puno. El propósito de su estudio, realizado desde la Universidad Nacional del Altiplano, fue comprobar si el uso de tecnología lograba disminuir las pérdidas por despacho o caducidad. Los resultados que obtuvo fueron claros: al estructurar la automatización, las mermas operativas se eliminan casi por completo. Estos hallazgos complementan el presente trabajo, ya que prueban que la automatización de los flujos de datos es útil para cualquier negocio de la región que trabaje con productos perecederos, y no solo para el sector hotelero.

Por último, Aguilar Moya (2024), quien también pertenece a la Universidad Andina, diseñó un aplicativo web para una empresa en Juliaca con el fin de reemplazar el conteo manual de mercancía. En su investigación quedó demostrado que al sistematizar las salidas de productos mediante una interfaz digital, se logran erradicar los errores humanos que suelen ocurrir durante el inventario físico. La importancia de este antecedente radica en que prueba empíricamente, en

nuestra misma ciudad, que la transición del papel al software ya no es una idea teórica, sino una necesidad operativa para no perder dinero. Los tres estudios demuestran que la automatización de los inventarios mitiga las mermas invisibles, validando la premisa central de nuestra tesis: si no sabes quién saca el producto, no puedes saber quién se lo llevó.

2.2. Bases teóricas de la Variable Independiente Sistema de Inventario en Python

2.2.1. Arquitectura de Software Local vs. Sistemas ERP

La elección de la arquitectura tecnológica es una de las decisiones más importantes en el desarrollo de un sistema, ya que de esto depende cuánto costará mantenerlo, si será escalable y qué tan autónomo será el negocio al usarlo. En el rubro de la gestión de almacenes, las organizaciones suelen evaluar dos caminos, implementar sistemas ERP (Planificación de Recursos Empresariales) comerciales o desarrollar un software local a medida.

Los sistemas ERP (como SAP, Oracle u Odoo) son plataformas que integran múltiples departamentos (recursos humanos, finanzas, ventas, logística) en tiempo real, generalmente a través de servidores o en la nube. Según Christopher (2023), aunque los ERP ofrecen una visión general de la empresa, su implementación requiere una inversión elevada en licencias, infraestructura de servidores y mantenimiento constante. Además, necesitan una conexión a internet estable y personal técnico dedicado, condiciones que suelen estar fuera del alcance para la mayoría de las Pequeñas y Medianas Empresas (PYMES) hoteleras regionales.

Por otro lado, la arquitectura de software local (también conocida como standalone) consiste en aplicaciones que se instalan y ejecutan directamente en las computadoras de trabajo, utilizando bases de datos que se guardan en el mismo equipo. Slack et al. (2022) sostienen que, para resolver problemas operativos específicos como el control de un solo departamento, estos

sistemas locales ofrecen un rendimiento muy bueno en términos de velocidad, precisamente porque no dependen de la velocidad del internet ni de servidores externos.

Para el caso específico de Intiqa Hotel Puno, se optó por una arquitectura local utilizando Python lenguaje que ha sido ampliamente reconocido en los últimos años por su eficiencia y facilidad para crear interfaces gráficas (Hortigüela Alcazar, 2025), Tkinter y SQLite por tres razones principales:

- **Independencia de internet (Offline):** Al no depender de servicios web, el sistema garantiza que el registro de ingresos y egresos de insumos nunca se detenga por caídas de red. En la operación diaria de un almacén hotelero personalizado, no poder registrar un egreso por falta de internet es un problema crítico que esta arquitectura elimina.
- **Cero costos de licencias (Open Source):** Tanto Python como Tkinter y SQLite son herramientas de código abierto y gratuitas. Esto significa que el hotel no tiene que pagar licencias mensuales ni anuales, lo cual representa una gran ventaja económica frente a un ERP comercial.
- **Adaptación exacta a los procesos del hotel:** Mientras que un ERP obliga al hotel a cambiar la forma en que trabaja para adaptarse al software, el sistema local permitió diseñar la pantalla exactamente conforme a la lógica del hotel. Por ejemplo, se logró vincular directamente un "Vale de salida" con un "Área de consumo" en un solo paso, tal como lo requiere la operativa real de Intiqa Hotel.

En conclusión, elegir una arquitectura local no fue una limitación técnica, sino la solución más práctica y funcional. Permitió dotar al Intiqa Hotel de capacidades importantes de trazabilidad y auditoría (como el control por roles y el registro Kardex), pero con la agilidad, el bajo costo y la independencia que el tamaño del negocio exigía.

.

2.2.2. Control de Acceso Basado en Roles (RBAC) y Seguridad de la Información

En la arquitectura de sistemas de información, la seguridad lógica es tan importante como la funcionalidad del software. Para que un sistema de inventario genere confianza en la gerencia, debe garantizar la no repudiación (la imposibilidad de que un usuario niegue haber realizado una acción) y el registro claro y seguro de los datos. Para lograr esto, el presente sistema fue diseñado bajo el estándar de ingeniería conocido como RBAC (Role-Based Access Control o Control de Acceso Basado en Roles).

Según los principios de la seguridad de la información, el modelo RBAC se aleja de la asignación de permisos directos a individuos y, en su lugar, asigna privilegios a perfiles funcionales predefinidos (roles). La aplicación de este modelo en el sistema desarrollado se materializa a través de dos mecanismos técnicos fundamentales:

- El Principio de Menor Privilegio (PoLP): Este principio dicta que un usuario solo debe tener los permisos estrictamente necesarios para realizar su tarea laboral, y nada más. En el contexto del Intiqa Hotel, el sistema implementó una bipartición de roles: el Rol Administrador (con control total sobre el sistema) y el Rol Usuario Operador (con acceso restringido únicamente a sus transacciones operativas). Si un operador intenta acceder a módulos de administración, el sistema deniega el acceso de manera preventiva.
- Protección de la identidad mediante hash criptográfico: Para que el control de roles funcione correctamente, los datos de acceso deben estar bien protegidos. El software no guarda las contraseñas en formato de texto visible. En su lugar, aplica el algoritmo SHA-256, que funciona como un método matemático unidireccional. Al crear un usuario o al

ingresar al sistema, la clave que escribe la persona se transforma en un código de letras y números imposible de revertir a la palabra original.

Impacto en la gestión del almacén: La unión de este control de roles con la encriptación soluciona de raíz el problema de pérdida de productos que se vio en Intiqa Hotel. Cada vez que un empleado saca bebidas o alimentos del depósito, debe entrar al sistema con su propia cuenta. De este modo, la salida física del bien queda registrada digitalmente junto al nombre del responsable. Si llegaran a faltar productos en el inventario, la gerencia tiene a la mano un historial que no se puede alterar. Allí se detalla con exactitud quién fue el autor del movimiento, lo que ayuda a acabar con la falta de responsabilidad en el manejo de los insumos.

2.2.3. El Método Kardex Como Motor Transaccional

Para la administración de operaciones, el control de un almacén va mucho más allá de contar productos físicos. Se trata más bien de un proceso que requiere anotar cada movimiento de manera ordenada. El método Kardex, que en sus inicios se basaba en tarjetas de cartón, hoy en día es la herramienta más utilizada para registrar lo que entra y sale del depósito. Heizer et al. (2020) señalan que usar un sistema de inventario permanente como este resulta clave para asegurar que siempre haya insumos disponibles, evitar que falten productos y mejorar el manejo de los costos. Esto se debe a que permite saber cuánta mercancía queda en tiempo real, sin necesidad de detener las labores diarias para hacer un recuento físico.

Al llevar este concepto al desarrollo de software, el Kardex deja de ser un simple formato en papel para convertirse en un motor de transacciones. Esto significa que cada ingreso o egreso no es simplemente un "apunte", sino una transacción atómica que altera el estado del inventario y deja un rastro auditor (log). El sistema desarrollado implementa esta lógica a través de un modelo de Inventario Perpetuo Automatizado, el cual opera bajo la siguiente premisa matemática y lógica:

- Registro de entradas (Ingresos): Cuando se compran bienes, el sistema registra la transacción. De manera obligatoria, vincula la cantidad de productos con el precio por unidad y el número de factura del proveedor.
- Registro de salidas (Egresos): Al entregar mercancía, el software descuenta el stock. Para que el movimiento se complete, se debe conectar la salida con un área específica (centro de costos) y con el nombre del empleado responsable.
- Cálculo dinámico de saldos: El programa suma los ingresos y resta los egresos de manera automática para cada producto. Esto asegura que la cantidad de mercancía que se ve en la pantalla sea exacta, evitando por completo los errores matemáticos que se cometen a mano.

Impacto en la operatividad de Intiqa Hotel: Cambiar el antiguo método manual (que usaba cuadernos y archivos de Excel desordenados) por este sistema digital soluciona dos problemas graves. Por un lado, elimina la posibilidad de que el personal se equivoque al calcular los saldos. Por otro lado, permite aplicar métodos de valoración, como el Promedio Ponderado, y crear reportes financieros en formato PDF en muy pocos segundos, lo que ayuda a cumplir con las normas de trazabilidad que exige la contabilidad.

2.2.4. Visualización de Datos Para la Toma de Decisiones (Dashboards)

Tener un sistema de inventario que solo guarde números en una base de datos no es de gran ayuda. El verdadero beneficio aparece cuando el software puede mostrar esos datos de tal forma que el gerente los entienda rápido y pueda actuar. Según Chopra y Meindl (2013), mencionan que las empresas suelen cometer el error de juntar mucha información, pero al no tener cómo analizarla en el momento, terminan demorando sus decisiones.

En ese sentido, una plataforma de inventarios no debe limitarse a guardar datos en tablas. Su aporte estratégico está en la capacidad de procesar esa información y mostrarla a través de

gráficos visuales. Estas pantallas ayudan a que la gerencia interprete lo que ocurre con el almacén de manera mucho más sencilla, mejorando así la rapidez con la que se toman decisiones. Al respecto, Humbatov (2024) sostiene que un problema recurrente en la gestión logística empresarial es la acumulación pasiva de grandes volúmenes de información sin herramientas oportunas de visualización y diagnóstico, un factor crítico que ralentiza la capacidad de respuesta operativa ante variaciones del stock.

Para solucionar esto, el sistema utiliza gráficos visuales, conocidos como Dashboards o cuadros de mando. Estos cuadros de mando han demostrado ser una herramienta muy útil en el sector empresarial para que los gerentes interpreten datos complejos de forma rápida (Dueñas Feuerbach, 2022). La idea es sencilla: en lugar de hacer que el gerente lea tablas llenas de números, el sistema le muestra la información en gráficos automáticos. En este caso se implementaron dos tipos de gráficos principales:

- **Gráficos Circulares (o de torta):** Permiten ver de un vistazo cómo se distribuye el porcentaje del gasto entre las diferentes áreas del hotel (cocina, barra, pisos). Esto ayuda a identificar rápidamente qué área consume la mayor parte del presupuesto.
- **Gráficos de Barras:** Permiten comparar los costos exactos (en soles) entre las áreas durante un periodo de tiempo. Esto es muy útil para que el gerente note anomalías, por ejemplo, si en un mes específico el gasto de la barra subió drásticamente en comparación con los meses anteriores.

Impacto en la operatividad de Intiqa Hotel, antes de este sistema, si la gerencia quería saber qué área gastaba más, tenía que pedir los cuadernos del almacén y hacer cálculos manuales. Esto tomaba tiempo y generalmente se hacía solo cuando ya se notaba una falta de dinero en el negocio.

Con la implementación de estos dashboards, el sistema suma los costos por área de forma automática y muestra los gráficos al instante. Esto permite que el gerente revise los gastos todos los días sin esfuerzo. Al poder ver la información en tiempo real, se pasa de reaccionar cuando ya hay un problema económico, a prevenir el problema desde el momento en que un área empieza a solicitar más insumos de lo normal.

2.3. Bases teóricas de la Variable Dependiente Gestión de Almacén

2.3.1. Gestión de Almacenes en el Sector Hotelero

La gestión de almacenes en el sector de hotelería presenta una peculiaridad estructural fundamental frente a la industria manufacturera, el producto final que se ofrece al huésped es de naturaleza intangible (la experiencia de hospedaje), pero la viabilidad de esa experiencia depende enteramente de la disponibilidad oportuna de bienes físicos tangibles.

Según Kotler et al. (2021), la satisfacción del cliente en el sector turismo y hotelería está directamente condicionada a la optimización de los procesos de almacenamiento interno. Los autores sostienen que un establecimiento de hospedaje no puede cumplir su promesa de marca si el suministro oportuno de alimentos, bebidas, lencería y productos de amenidad falla. En este sentido, el almacén hotelero deja de ser un simple depósito para convertirse en un nodo logístico crítico.

A esta situación se le conoce como "invisibilidad logística". Las salidas de productos se realizan en el Back of House (las áreas operativas de atrás), fuera de la vista del cliente. No obstante, si algo falla en esa zona, el problema se nota de inmediato en el Front of House, como en el restaurante o la barra. Chopra y Meindl (2013) explican que, en el sector servicios, la cadena

de suministro es eficiente solo si el almacén logra entregar los productos a las áreas internas sin demoras ni contratiempos.

Impacto en la operatividad de Intiqa Hotel Puno: Entender este punto resulta clave para el presente proyecto. Ayuda a explicar por qué que falten cervezas, o que no haya control sobre el café y la leche, no debe verse como una "pérdida menor". En realidad, son fallas logísticas que ponen en riesgo el servicio principal del hotel. Si la cocina pide insumos y el almacén no lleva un control claro separado por áreas, se forma un cuello de botella. Al final, ese problema termina afectando de forma directa la experiencia del huésped.

2.3.2. Trazabilidad y Asignación por Centros de Costos

En el ámbito financiero y logístico de un hotel, las pérdidas en el almacén no siempre se dan por robos. Muchas veces, el problema principal es la merma operativa, es decir, el deterioro o el mal uso de los productos. Para hacerle frente a esto, una buena alternativa es usar centros de costos. La idea es tratar cada área que pide insumos (como pisos, barra o cocina) como un consumidor independiente. Christopher (2023) explica que, si la empresa no separa los gastos por cada una de estas áreas, ocurre una dilución del dinero. En pocas palabras, las pérdidas se esconden dentro del presupuesto general y los administradores no logran detectar de dónde viene la fuga.

Para solucionar esto, la ingeniería de sistemas aplica la trazabilidad en el suministro interno. Chopra y Meindl (2013) señalan que es fundamental tener visibilidad sobre cómo se mueve el inventario, ya que esto ayuda a reducir los errores en los registros y a mejorar el servicio. Esto significa que se debe rastrear cada producto desde el momento en que entra al depósito hasta que un departamento específico lo consume.

Controlar esto se vuelve más complicado en el sector hotelero porque se trabaja con muchos bienes perecederos. Kotler et al. (2021) mencionan que, si no se hace un buen seguimiento de los productos que se vencen, las ganancias del negocio bajan de forma directa. Un ejemplo claro ocurre si se compran yogures, pero el software no exige vincular su salida con el área de cocina. En ese caso, se crea un punto ciego. Cuando el personal encuentra productos vencidos al cerrar el mes, la administración no sabe si la pérdida fue porque la cocina pidió de más o porque el almacén no hizo rotar bien la mercancía.

Para evitar que esto siga ocurriendo, el sistema creado para Intiqa Hotel Puno incorporó una regla estricta: todo egreso debe estar vinculado a un área de consumo. El programa está diseñado para que el trabajador elija el destino de los insumos antes de registrar la salida. Si no selecciona el área, el sistema simplemente no permite completar el movimiento. De esta forma, se crea una relación directa donde cada retiro de mercancía afecta la cuenta de la sección que lo pidió. Así, cuando un departamento saca más recursos de la cuenta y los desperdicia, el reporte de gastos muestra de inmediato ese consumo excesivo.

2.4. Marco conceptual

A continuación, se detallan los conceptos principales que sustentan el desarrollo de esta investigación:

1. **Sistema de Inventario en Python**: Es el software creado con herramientas de uso libre que se utilizó para anotar los movimientos del almacén en el momento en que ocurren. La decisión de emplear este lenguaje de programación no fue casual; Sánchez Alberca (2020) explica que Python se destaca por lo fácil que resulta leer y escribir su código. Esa misma

claridad técnica fue la que permitió ajustar el programa a las reglas de Intiqa Hotel sin causar problemas, logrando conectar las pantallas y los reportes de manera directa.

2. **Gestión de Almacén:** Se refiere a la forma en que se planifican, organizan y controlan las actividades de recibir, guardar y entregar los insumos. Christopher (2023) indica que cuando esta área se maneja bien, la empresa siempre cuenta con los materiales que necesita a tiempo y logra reducir sus gastos logísticos.
3. **Método Kardex:** También conocido como inventario perpetuo. Es una forma de llevar los registros donde se anota en orden de fecha todo lo que entra, lo que sale y lo que queda en el depósito. Heizer et al. (2020) señalan que usar este método es indispensable para evitar quedarse sin productos y para saber cuánta mercancía hay sin tener que parar el trabajo para contarla a mano.
4. **Trazabilidad:** Es la posibilidad de seguir el recorrido completo de un producto desde que ingresa al hotel hasta que alguien lo consume. Chopra y Meindl (2013) mencionan que llevar este control ayuda a disminuir los errores en los registros y hace que la operación diaria sea mucho más ágil.
5. **Centro de Costos:** Es cualquier sección del hotel que pide productos del almacén, como por ejemplo la cocina o la barra. Christopher (2023) advierte que si la empresa no separa los gastos por cada una de estas áreas, será imposible saber cuál de ellas está gastando más dinero del debido.
6. **Mermas Operativas:** Es la pérdida de mercancía o de dinero que ocurre por causas internas, ya sea porque los productos se vencen, se rompen o se pierden. Kotler et al. (2021) indican que este tipo de pérdidas afecta de forma directa a la satisfacción de los clientes en el rubro hotelero.

7. **Control de Acceso Basado en Roles (RBAC):** Paradigma de seguridad informática que limita los permisos del usuario según su rol organizacional, aplicando de esta manera el principio de menor privilegio (Chapple y Seidl, 2024).
8. **Arquitectura de Software Local (Standalone):** Modelo de sistema que se ejecuta directamente en los equipos de trabajo sin depender de internet. Slack et al. (2022)sostienen que estas arquitecturas ligeras ofrecen un rendimiento superior al no depender de latencias de red.
9. **Sistemas ERP:** Planificación de Recursos Empresariales (como SAP u Oracle). Christopher (2023) señala que, aunque ofrecen visión transversal, su implementación conlleva un gasto elevado desproporcionado para PYMES.
10. **Acumulación de datos sin interpretar:** Problema común en las empresas donde se juntan muchos registros en el sistema, pero no existe una forma sencilla de analizarlos para tomar decisiones rápidas. Chopra y Meindl (2013) identifican esta barrera como la principal causa de retrasos en las empresas.
11. **Alertamiento Temprano (Stock Mínimo):** Es una función del sistema que avisa con anticipación cuando la cantidad de un producto está por agotarse. Slack et al. (2022) contar con este tipo de alertas es lo que marca la diferencia entre un manejo de almacén que solo reacciona a los problemas y otro que logra anticiparse a ellos.
12. **Dashboards Gerenciales (Cuadros de Mando):** Son pantallas gráficas que toman los datos operativos del sistema y los presentan de forma visual para facilitar su comprensión. Debido a que las personas pueden analizar gráficos mucho más rápido que tablas llenas de números, usar estos cuadros de mando ayuda a que la gerencia entienda la situación del almacén y tome decisiones con mayor agilidad (Dueñas Feuerbach, 2022).

CAPÍTULO III

METODOLOGÍA DE LA INVESTIGACIÓN

3.1. Enfoque de la Investigación

Para el desarrollo de este estudio, se optó por un enfoque cuantitativo bajo el paradigma positivista. Según explican Hernández Sampieri et al. (2014), este tipo de enfoque es el indicado cuando se necesita recolectar datos numéricos para someter a prueba las hipótesis planteadas. En la práctica, esto significó medir aspectos operativos concretos, como cuánto se redujeron los tiempos de consulta y la cantidad de errores en los registros. Además, se evaluó la percepción de los usuarios mediante la escala de Likert. La combinación de estas mediciones ayudó a garantizar que los resultados obtenidos fueran objetivos y posibles de comprobar en futuros estudios.

3.2. Tipo de Investigación

La investigación fue de tipo Aplicado. Al respecto, Hernández Sampieri et al. (2014) establece que los estudios aplicados se caracterizan por buscar la solución de problemas prácticos e inmediatos, utilizando los conocimientos científicos y tecnológicos para intervenir en la realidad. En concordancia con esta premisa, el presente estudio no tuvo como finalidad la creación de nueva teoría abstracta, sino la utilización directa de la ingeniería de software (desarrollo en Python) para intervenir y resolver la deficiencia estructural de trazabilidad y control de inventarios que afectaba operativamente a Intiqa Hotel Puno.

OFICINA DE INVESTIGACIÓN
Tesis Publicada con autorización del autor
http://repositorio.uancv.edu.pe/

3.3. Nivel de Investigación

El estudio alcanzó un nivel Descriptivo-Explicativo. En su nivel descriptivo, porque buscó detallar y caracterizar la situación actual del almacén previa a la intervención, así como las funcionalidades específicas del sistema implementado (RBAC, Kardex transaccional, Dashboards por centro de costos). En su nivel explicativo, porque fue más allá de la simple descripción al buscar establecer la relación de asociación entre la Variable Independente (implementación del sistema Python) y la Variable Dependiente (optimización de la gestión de almacén), explicando por qué y cómo la trazabilidad por áreas redujo las mermas operativas y mejoró la toma de decisiones gerenciales.

3.4. Diseño de la Investigación

La investigación se sustentó en un diseño No experimental, de corte transversal. Se denominó no experimental porque no se manipularon las variables de manera deliberada en un entorno de laboratorio; más bien, se observó el fenómeno en su contexto natural (en Intiqa Hotel) para analizar el efecto del sistema ya implementado. Fue de corte transversal porque la recolección de datos y la evaluación de los indicadores de gestión se realizaron en un momento y espacio únicos (durante el periodo de ejecución del año 2025), obteniendo una "fotografía" del impacto del sistema en ese punto específico del tiempo (Hernández Sampieri et al., 2014).

3.5. Método de Investigación

El desarrollo de la investigación se estructuró bajo el rigor del método científico, adoptando específicamente la lógica deductiva e hipotético-deductiva propia del enfoque cuantitativo. Al respecto, Hernández Sampieri et al. (2014) señala que este proceso exige partir de una observación empírica para formular hipótesis que posteriormente deben ser evaluadas mediante la recolección de datos. En concordancia con este planteamiento, el estudio inició con el análisis del problema

logístico (pérdidas de insumos sin trazabilidad), prosiguió con la formulación de la hipótesis (el sistema en Python optimizaría la gestión del almacén) y culminó con su contrastación. Dicha validación se logró aplicando instrumentos de recolección de datos a los usuarios operativos, lo que permitió comprobar la viabilidad y el impacto de la propuesta tecnológica en el entorno real de Intiqa Hotel Puno.

3.6. Población y Muestra

3.6.1. Población

La población de estudio estuvo confo

rmada por la totalidad del personal directamente involucrado en las operaciones logísticas de Intiqa Hotel Puno, específicamente el área de almacén y compras, la cual estuvo compuesta por 5 trabajadores.

3.6.2. Muestra

Para determinar la muestra representativa de la población de 5 trabajadores, fue necesario aplicar un cálculo acorde a la magnitud del universo. Al respecto, Hernández Sampieri et al. (2014) establece que, cuando se conoce el tamaño exacto de la población y esta es reducida, no es metodológicamente correcto aplicar fórmulas diseñadas para muestras infinitas.

En concordancia con este principio, se aplicó la teoría de muestreo para poblaciones finitas desarrollada por Cochran (1977), la cual ajusta matemáticamente el tamaño de la muestra cuando el universo es pequeño. La ecuación es la siguiente:

$$n = \frac{z^2 pqN}{(N-1)e^2 + z^2\, pq}$$

Donde:

n = Tamaño de la muestra.

N = Tamaño de la población (5 trabajadores).

p = Probabilidad de éxito (0.5 para máxima variabilidad).

q = Probabilidad de fracaso (0.5).

Z = Nivel de confianza (1.96 para un 95%).

e = Margen de error (0.05 o 5%).

Sustituyendo los valores en la ecuación:

$$n = \frac{1.96^2 * 0.5 * 0.5 * 5}{(5-1)00.5^2 + 196^2 * 0.5 * 0.5}$$

$$n = \frac{3.8416 * 0.5 * 0.5 * 5}{4 * 0.0025 + 3.8416 * 0.25}$$

$$n = \frac{4.802}{0.9604}$$

$$n = 4.948$$

Dado que la muestra matemática resultante es de 4.948 sujetos, y en estadística aplicada no es posible evaluar fracciones de persona, se redondea al entero superior inmediato, resultando en **n = 5.**

Como el resultado de la fórmula arroja que se debe evaluar a 5 personas, lo cual equivale al 100% de la población total (N), la metodología dicta realizar un censo, evaluando a la totalidad de los trabajadores del área de almacén y compras de Intiqa Hotel Puno. Esto elimina cualquier margen de error muestral y garantiza la máxima representatividad de los datos.

3.7. Técnicas e Instrumentos de Recolección de Datos

Para la contrastación de las hipótesis y la evaluación del sistema, se emplearon las siguientes técnicas e instrumentos, alineados estrictamente a las dimensiones e indicadores de la tabla de operacionalización de variables (Capítulo I):

- **Técnica de la Encuesta:** Se utilizó para recoger la percepción cualitativa convertida en datos cuantitativos de los usuarios respecto a la usabilidad, seguridad (RBAC) y utilidad de los módulos del sistema. Instrumento, cuestionario estructurado. Diseñado con 10 ítems de opción múltiple, empleando una escala de medición tipo Likert de 5 puntos (1 = Totalmente en desacuerdo, 5 = Totalmente de Acuerdo), aplicado de forma personalizada a la muestra seleccionada.
- **Técnica de la Observación Directa:** Se utilizó para comprobar en el sitio cómo funcionaba el programa y para medir aspectos técnicos que no dependen de la opinión de las personas, como los tiempos de respuesta. Para esto, se empleó como instrumento una ficha de observación de sistemas. Esta guía sirvió para revisar si el software cumplía con las funciones requeridas durante el uso real en el hotel. Vale la pena señalar que obtener esta información fue más sencillo debido a que el investigador labora en el mismo hotel, en el turno de auditoría nocturna. Estar en ese puesto permitió hacer una observación participante. Esto resultó clave para notar la diferencia que existía entre lo que decían los cuadernos y lo que realmente pasaba en el almacén, como la pérdida de insumos sin darse

cuenta o la salida de bebidas sin ningún registro. Estos hechos demostraron de forma clara por qué era tan necesario desarrollar este sistema.

3.8. Validez y Confiabilidad del Instrumento

Para asegurar que el cuestionario midiera de forma correcta lo que se pretendía evaluar, el primer paso fue analizar su validez de contenido. El instrumento fue entregado a tres especialistas para su revisión: el asesor de la investigación, el Dr. Paul Mamani Tisnado; un ingeniero de sistemas, el Mg. Ing. Jose Daniel Zamalloa Nuñez; y un administrador con experiencia en el área logística, el Mg. Sc. Guido Vilmar Mamani Lujano. Estos profesionales se encargaron de revisar la redacción de las preguntas para comprobar su claridad y su relación con los objetivos del proyecto. Como criterio de filtro, en el cuestionario final solo se incluyeron las preguntas en las que los tres lograron un nivel de acuerdo superior al 80%.

Al llegar al punto de la confiabilidad, la lógica metodológica tradicional suele pedir aplicar el Alfa de Cronbach. Sin embargo, hacer eso en nuestro caso tenía un problema de fondo, nuestra muestra no es una fracción de una población grande, es un censo directo de solo 5 personas. Desde el punto de vista estadístico, calcular el Alfa de Cronbach con apenas cinco personas no entrega un resultado confiable, ya que los cálculos se ven muy afectados por el tamaño del grupo.

Al momento de evaluar la consistencia interna del cuestionario, la metodología tradicional casi siempre indica usar el coeficiente Alfa de Cronbach. Sin embargo, aplicar esta fórmula en el presente caso implicaba un problema de base. El grupo de análisis no es una pequeña porción extraída de una población más grande, sino un censo completo de cinco trabajadores (n=5). Sobre este tema, Field (2018) explica que la confiabilidad de los datos estadísticos depende en gran

medida del tamaño de la muestra. Cuando se trabaja con un número tan reducido de participantes, las varianzas y el error estándar sufren alteraciones severas, lo que genera un sesgo que le quita validez al examen. Por consiguiente, la estimación del Alfa de Cronbach bajo un marco de n=5 distorsionaría la realidad empírica, ofreciendo un indicador artificialmente forzado y desprovisto de rigor estadístico.

Ante esta limitación, y con el fin de no forzar un cálculo matemático que no aportaría datos reales, la confiabilidad del cuestionario se verificó usando un enfoque empírico. Esta alternativa resulta mucho más práctica y directa para el campo de la ingeniería de sistemas. Lo que hicimos fue tomar las percepciones que los 5 operadores marcaron en el papel (su encuesta) y cruzarlas directamente con los hechos técnicos reales que arrojó el sistema Python ya operando (los tiempos y porcentajes de la Tabla 3). Si el papel dice que el sistema es rápido y la Tabla 3 demuestra que el tiempo de consulta cayó de minutos a segundos, entonces el instrumento es confiable, porque la realidad operativa le está dando la razón a lo que la gente respondió.

CAPÍTULO IV

DESARROLLO DEL SISTEMA Y ANÁLISIS DE RESULTADOS

4.1. Metodología de Desarrollo de Software Aplicada

Para la construcción del sistema de información, se seleccionó la metodología de Desarrollo Rápido de Aplicaciones (RAD, por sus siglas en inglés). Diversos estudios recientes han demostrado que esta metodología sigue siendo muy útil para construir sistemas de información de manera ágil y ordenada (Adaile Benítez et al., 2025), complementada con un enfoque de Prototipado Evolutivo.

A diferencia de metodologías pesadas como el modelo en Cascada (Waterfall) que exige documentación exhaustiva previa o marcos ágiles rígidos como Scrum que requieren ritualizar reuniones diarias (Daily stand-ups) y ciclos de desarrollo fragmentados en Sprints, el modelo RAD se fundamenta en la construcción iterativa del software y en la participación activa del usuario final. Esta selección de ingeniería se justificó por tres motivos fundamentales:

- El entorno de Intiqa Hotel exigía una solución operativa y funcional en el corto plazo, priorizando el valor operativo sobre la burocracia del proceso.
- El uso del lenguaje Python junto con la librería nativa Tkinter permite generar interfaces gráficas visuales de manera casi inmediata, siendo un lenguaje muy eficiente para este tipo de desarrollos (Hortigüela Alcazar, 2025), resultando idóneo para el prototipado rápido.

- Debido a que el autor de esta tesis trabaja directamente en la operación del hotel, se pudo llevar a cabo una observación participante. Esto fue una ventaja, ya que permitió mantener una comunicación constante entre el desarrollador y los usuarios. Gracias a esto, el programa se fue modificando sobre la marcha según las necesidades reales del almacén, hasta llegar a la versión final lista para usarse, todo esto sin tener que esperar largos tiempos de aprobación.

4.2. Arquitectura y Diseño de la Solución Tecnológica

Después de elegir la metodología y de definir qué funciones debía cumplir el programa según el diagnóstico del hotel, se pasó a la etapa de construcción del software. Como se mencionó en la sección 2.2.1, se decidió crear un sistema que funcione de forma local, sin depender de una conexión a internet. Esta arquitectura independiente se organizó en tres capas lógicas:

- **Capa de Presentación (Front-end):** Se diseñó usando la librería Tkinter de Python. La ventaja de esto es que el programa resulta bastante ligero y permite que se ejecute en las computadoras del hotel sin tener que instalar complementos extra.
- **Capa de Lógica de Negocio (Back-end):** Programada en Python puro, encapsulando las reglas de negocio críticas (cálculo de saldos mediante Kardex permanente, validación de RBAC, verificación de umbrales de stock mínimo).
- **Capa de Persistencia de Datos:** Gestionada mediante el motor de base de datos relacional SQLite (archivo kardex.db), el cual es ampliamente utilizado por su ligereza (Mercado Pavón, 2023) y permite almacenar las transacciones de forma segura, ágil y sin requerir un servidor de base de datos dedicado.

A continuación, se presentan las evidencias de implementación de los módulos que materializan las variables de estudio, contrastadas con las bases teóricas del Capítulo II.

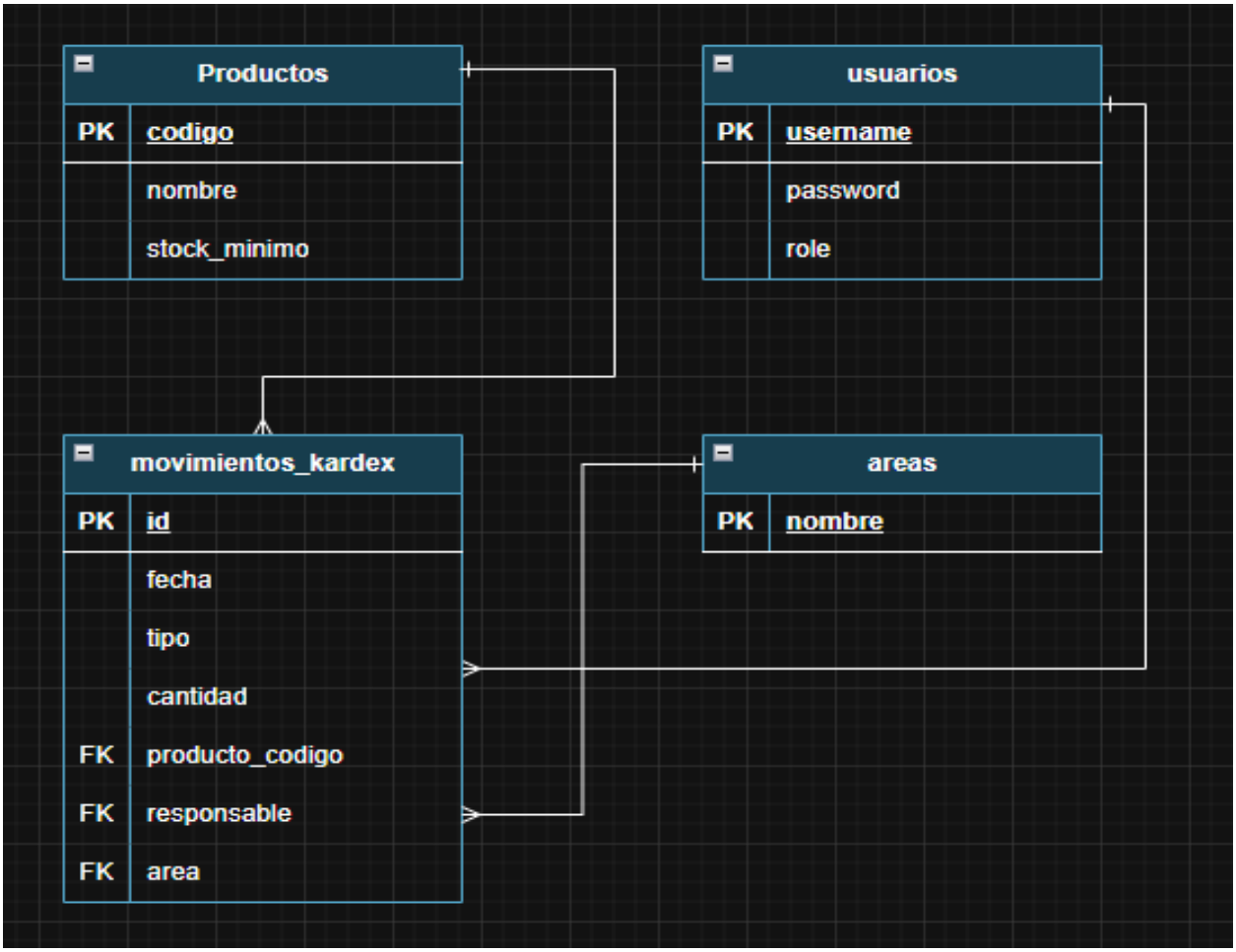
4.3. Diseño Lógico de la Base de Datos

Para garantizar la persistencia de la información y la integridad referencial de los datos de la Capa de Persistencia, se diseñó un modelo relacional alojado en SQLite. El diseño se estructuró en tres entidades principales y una entidad auxiliar, normalizadas hasta la Tercera Forma Normal (3FN):

- Entidad **usuarios**: Almacena las credenciales y el perfil RBAC.
- Entidad **productos**: Registra el catálogo maestro de insumos y umbrales de alerta.
- Entidad **movimientos_kardex**: Tabla transaccional (Hecho) que almacena cada entrada o salida.
- Entidad **areas**: Catálogo de los centros de costos del hotel.

Figura 1

Modelo Entidad-Relación (MER) de la base de datos kardex.db.



Nota. Elaboración propia diseñada mediante la plataforma Diagrams.net, basada en la arquitectura lógica del sistema (2025).

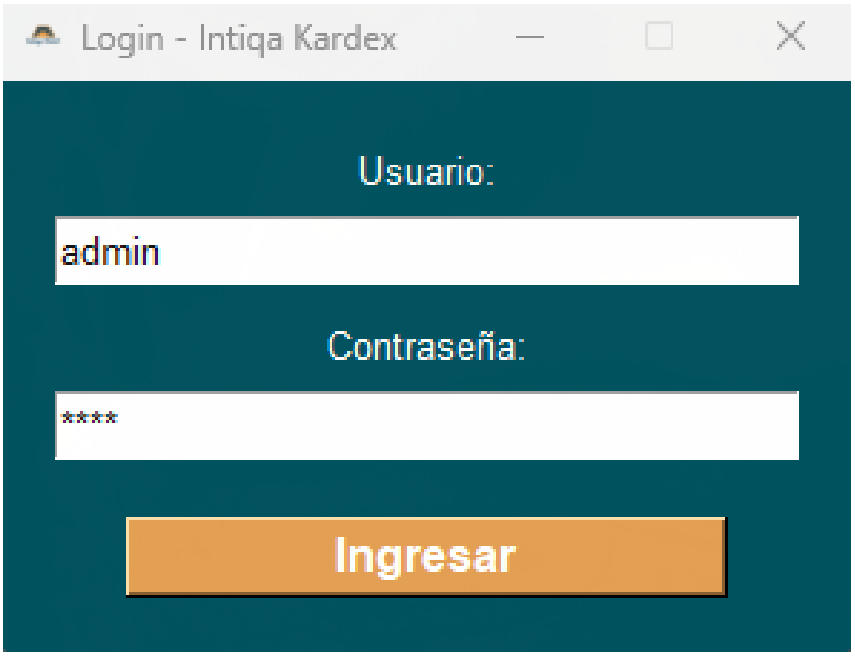
4.4. Evidencias de Implementación del Sistema

4.4.1. Módulo de Seguridad y Control de Acceso (RBAC)

Como se teorizó en el apartado 2.2.2, la seguridad lógica es fundamental para la no repudiación. El sistema implementa una pantalla de autenticación que bloquea el acceso a usuarios no autorizados mediante la validación de credenciales hasheadas (SHA-256).

Figura 2.

Interfaz de autenticación de usuarios mediante hash criptográfico.

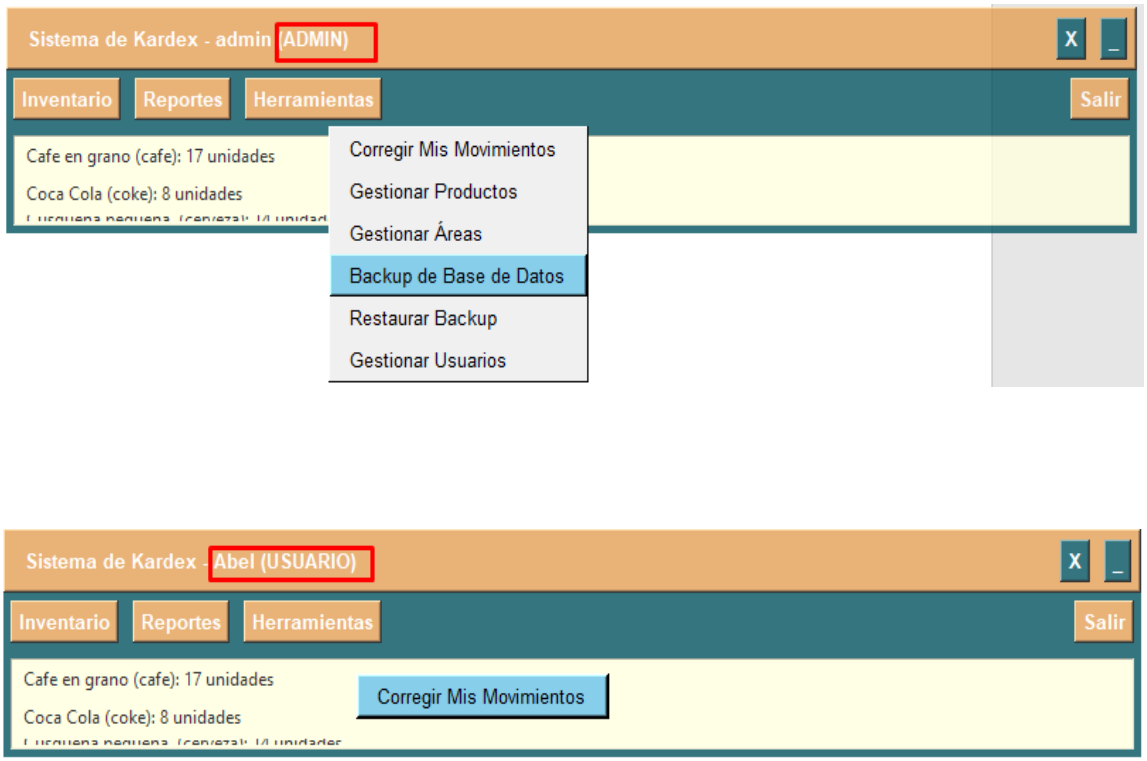


Nota: Elaboración propia a partir del sistema desarrollado en Python (2025).

Una vez autenticado, el sistema aplica el Principio de Menor Privilegio (PoLP). La Figura 3 demuestra cómo el sistema distingue entre un administrador (con acceso total a herramientas de gestión) y un usuario operador, restringiendo botones críticos según el rol asignado en la base de datos.

Figura 3

Interfaz principal del sistema mostrando el control de acceso basado en roles (RBAC).



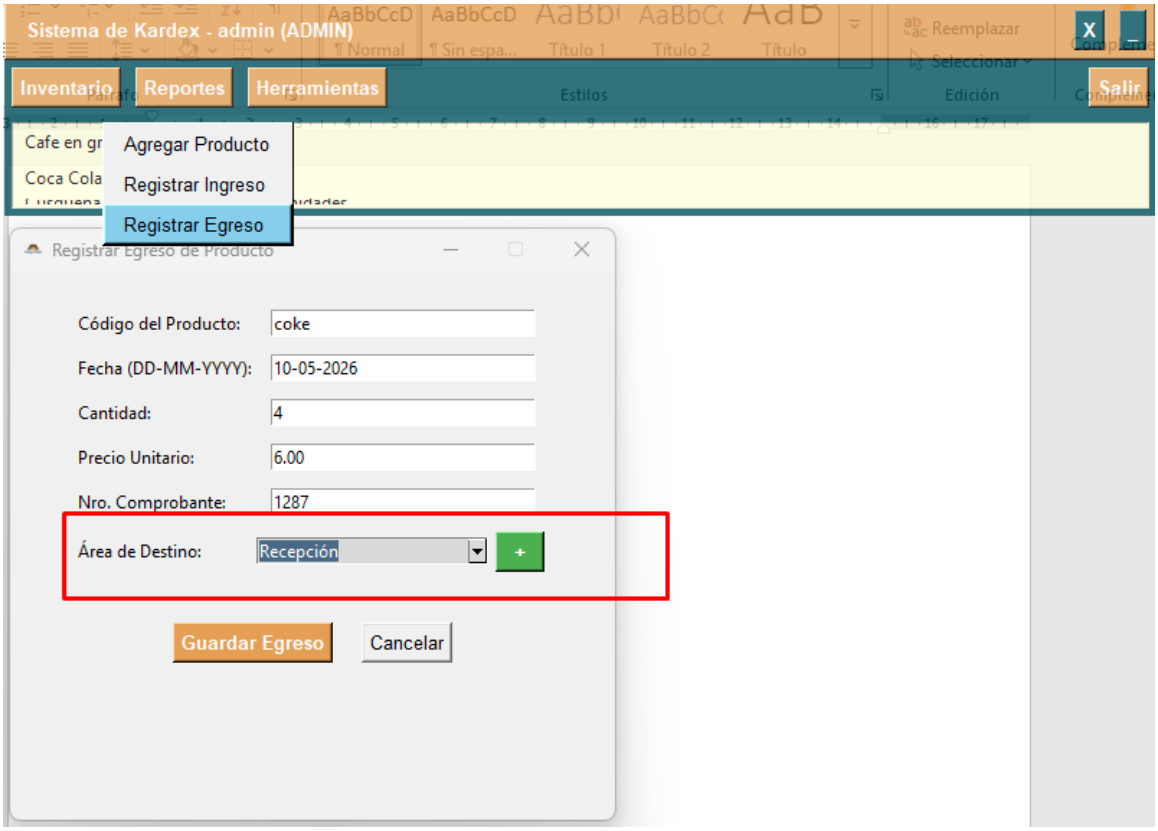
Nota: Elaboración propia a partir del sistema desarrollado en Python (2025).

4.4.2. Módulo de Trazabilidad por Centros de Costos

El núcleo de la Variable Dependiente (Gestión de Almacén) se resolvió mediante la trazabilidad. En la Figura 3, se observa el módulo de "Registrar Egreso", el cual exige de manera obligatoria la selección del "Área de Destino" (Centro de Costos). Esta funcionalidad conecta directamente el egreso físico del insumo con el área consumidora (ej. Cocina, Barra), resolviendo el problema de las mermas operativas y donde no se sabía qué área gastaba más, teorizados en el apartado 2.3.2.

Figura 4.

Módulo de egreso de inventario mostrando la asignación obligatoria por centros de costos.



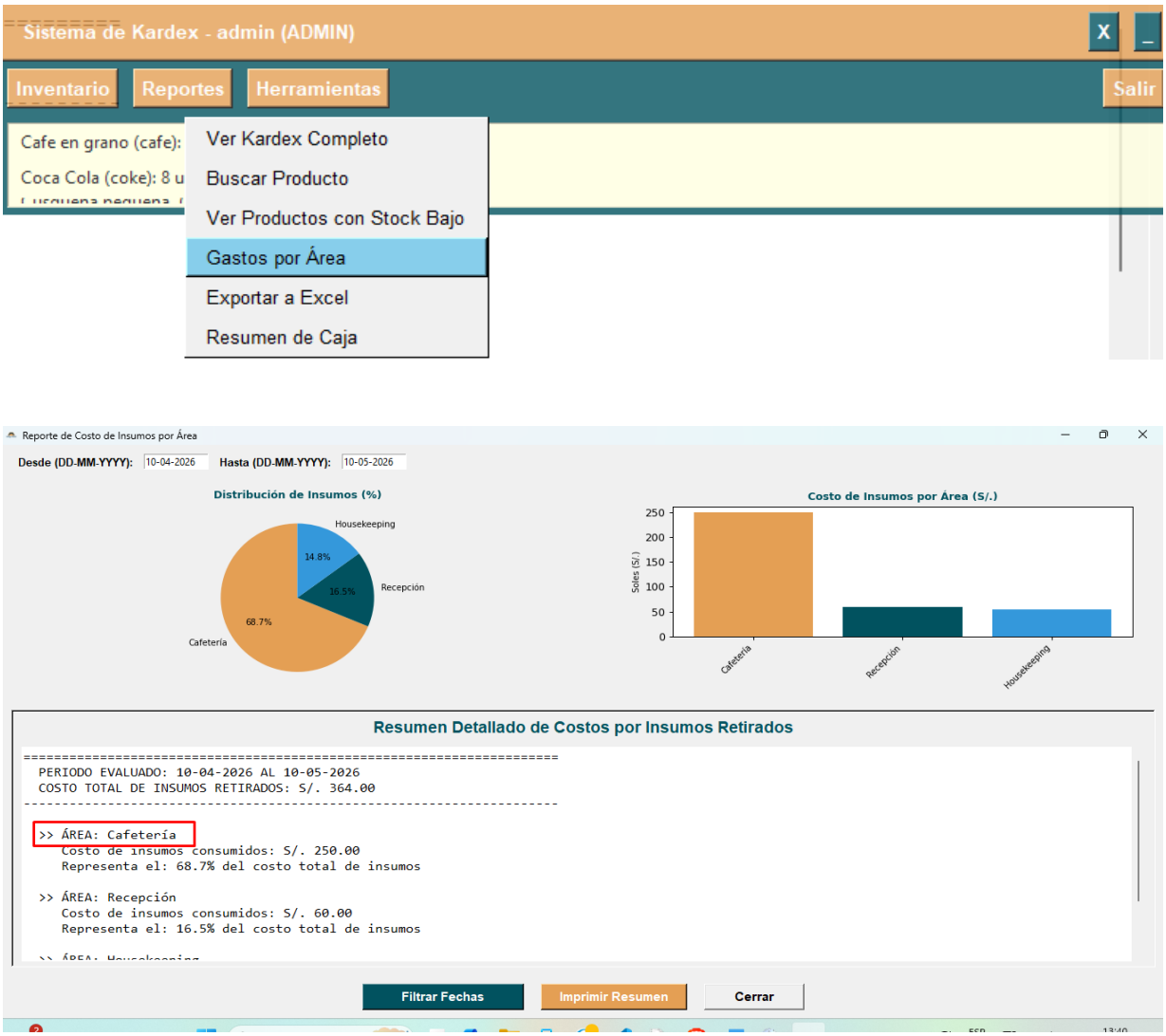
Nota: Elaboración propia a partir del sistema desarrollado en Python (2025).

4.4.3. Módulo de Visualización de Datos (Dashboards Gerenciales)

Como se explicó en la teoría (apartado 2.2.4), tener mucha información guardada no sirve de mucho si no se puede interpretar fácilmente. Por eso, el sistema utiliza la librería Matplotlib para generar los Dashboards. La Figura 4 muestra el reporte de "Gastos por Área", el cual permite a la gerencia del Intiqa Hotel ver de forma rápida y visual en qué área se está gastando el mayor porcentaje del presupuesto (en soles). Esto ayuda a que los encargados tomen decisiones a tiempo y eviten que los costos se descontrolen en una sola área.

Figura 5.

Dashboard de inteligencia de negocios mostrando distribución proporcional de costos por área de consumo.



Nota: Elaboración propia a partir del sistema desarrollado en Python (2025).

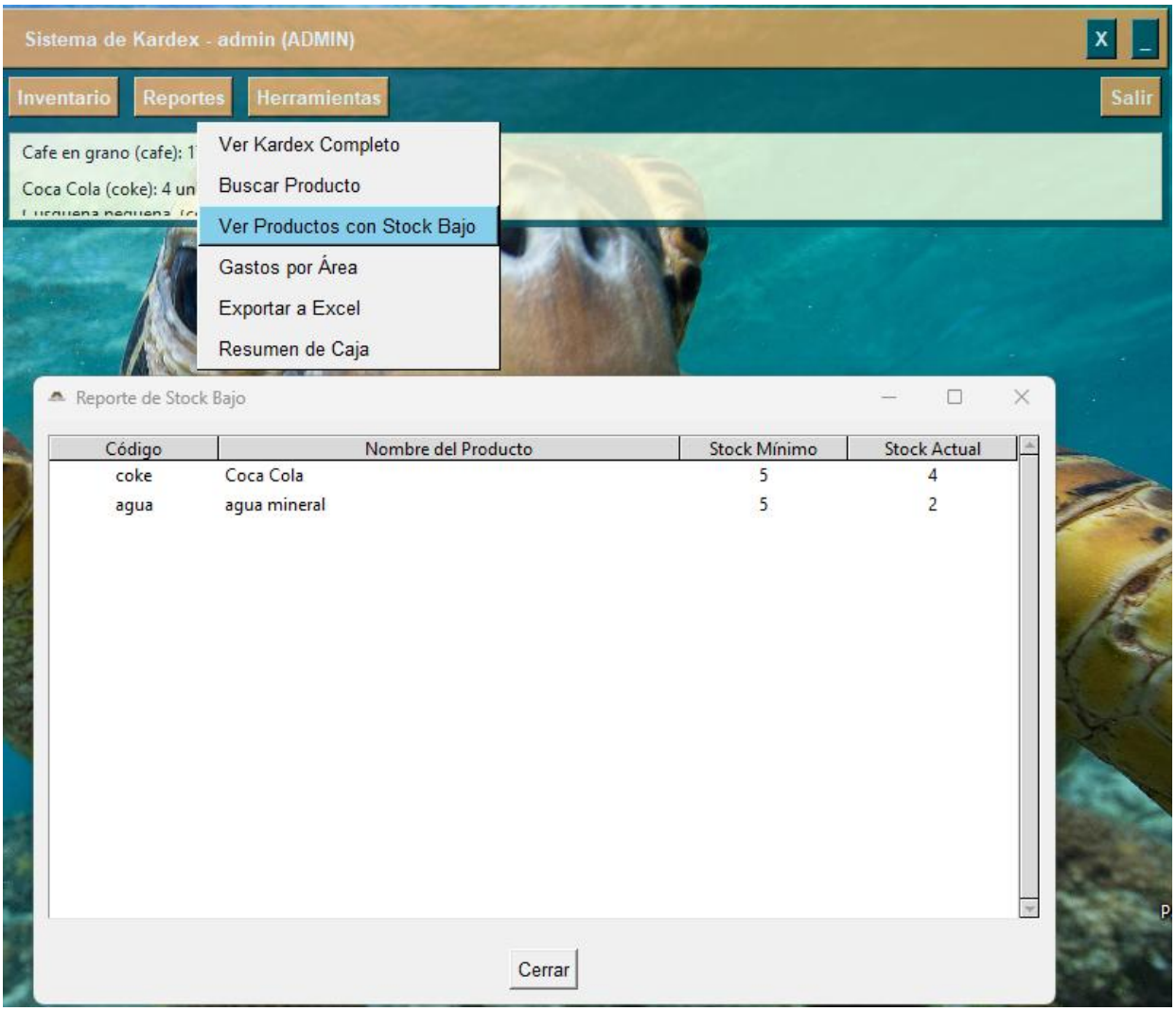
4.4.4. Módulo de Alertamiento Temprano (Stock Mínimo)

Tal como se fundamentó en el apartado 2.3.3, la diferencia entre una gestión reactiva y una proactiva radica en los sistemas de alertamiento temprano. La Figura 5 muestra el módulo de "Reporte de Stock Bajo", el cual cruza automáticamente el saldo actual de cada producto contra el

umbral mínimo configurado. Esto permite a los responsables del almacén prever rupturas de stock (ej. que no falten cervezas en el bar) y optimizar el flujo de caja al realizar compras exactas.

Figura 6.

Interfaz del módulo de alertamiento temprano comparando saldos reales vs. umbrales mínimos.



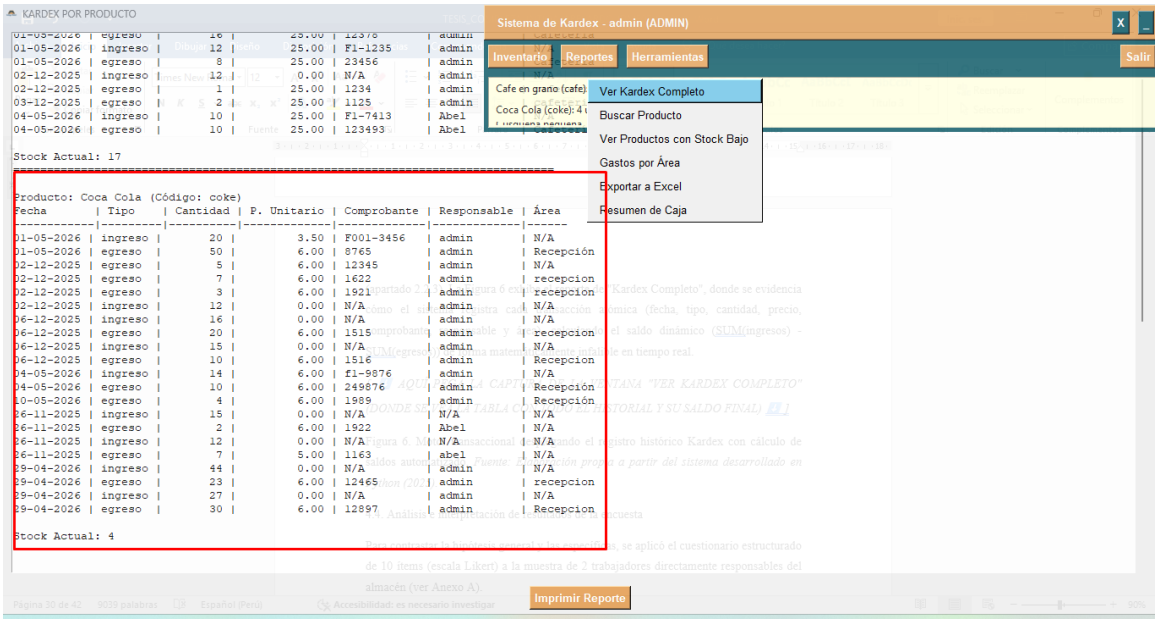
Nota: Captura de pantalla del software creado por el autor en Python (2025).

4.4.5. Módulo de Registro Kardex Permanente

Con este módulo, el software evita los errores que cometía el personal al hacer los cálculos a mano en los cuadernos. Esto permite aplicar en la práctica lo explicado sobre el Inventario en el apartado 2.2.3. La Figura 6 muestra el reporte del "Kardex Completo". En esta pantalla se observa cómo el programa registra cada movimiento de forma detallada, guardando datos como la fecha, el tipo, la cantidad, el precio, el comprobante, el responsable y el área de destino. Para saber cuánto queda de cada producto, el sistema resta de manera automática las salidas de las entradas. Esta operación hace que el saldo que ve el usuario en la pantalla sea totalmente exacto y se actualice al instante.

Figura 7.

Motor transaccional desplegando el registro histórico Kardex con cálculo de saldos automatizado.



Nota: Elaboración propia a partir del sistema desarrollado en Python (2025).

4.5. Análisis e interpretación de resultados de la encuesta

Para contrastar la hipótesis general y las específicas, se aplicó el cuestionario estructurado de 10 ítems (escala Likert) a la muestra de 5 trabajadores directamente responsables del almacén (ver Anexos).

A continuación, se presenta la tabulación general de los resultados obtenidos:

Tabla 2

Distribución de frecuencias de la evaluación del sistema de inventario

Dimensión / Ítem Evaluado	TD (1)	ED (2)	N (3)	DA (4)	TA (5)	Promedio
1. Antes había pérdidas por falta de control de áreas.	0	0	0	0	5	5.0
2. Ahora sé qué área gasta más insumos.	0	0	0	0	5	5.0
D2: Seguridad (RBAC)						
3. El Login me hace sentir responsable.	0	0	0	2	3	4.6
4. El bloqueo de opciones evita alteraciones.	0	0	0	0	5	5.0
D3: Automatización (Dashboards)						
5. Los gráficos ayudan a entender los gastos.	0	0	0	2	3	4.6

Dimensión / Ítem Evaluado	TD (1)	ED (2)	N (3)	DA (4)	TA (5)	Promedio
6. Las alertas de Stock Bajo son útiles.	0	0	0	0	5	5.0
7. Generar PDF/Excel es más rápido y seguro.	0	0	0	2	3	4.6
D4: Eficiencia Operativa						
8. El tiempo de búsqueda de stock se redujo.	0	0	0	0	5	5.0
9. Se eliminaron los errores matemáticos a fin de mes.	0	0	0	2	3	4.6
D5: Toma de Decisiones						
10. La información es confiable para la gerencia.	0	0	0	2	3	4.6
TOTAL GENERAL	0	0	0	10	40	4.8

Nota. TD = Totalmente en Desacuerdo; ED = En Desacuerdo; N = Neutral; DA = De Acuerdo; TA = Totalmente de Acuerdo.

Fuente: Elaboración propia a partir de los datos recolectados (2025).

Interpretación: El análisis de la Tabla 2 revela que el 100% de las respuestas se concentran en las categorías de "De Acuerdo" (50%) y "Totalmente de Acuerdo" (50%), logrando un promedio general ponderado de 4.8 sobre 5.0. Desde el punto de vista estadístico descriptivo, un valor superior a 4.0 en una escala Likert indica un nivel de aceptación y satisfacción "Alto/Muy Alto".

Esto permite deducir empíricamente que la implementación del sistema de inventario en Python optimizó significativamente la gestión de almacén, al validar que:

- El problema inicial de las mermas por falta de trazabilidad fue erradicado (Promedio 5.0 en D1).
- Los tiempos operativos de consulta y cuadre se redujeron drásticamente (Promedio 4.8 en D4).
- Los dashboards y alertas cumplieron su función de inteligencia de negocios (Promedio 4.73 en D3).

Sin embargo, la percepción subjetiva del usuario requiere ser complementada con evidencia objetiva y cuantificable. Por ello, para validar el impacto real del Sistema de Inventario en Python (Variable Independiente) sobre la Gestión de Almacén (Variable Dependiente), se estableció un comparativo de indicadores de rendimiento operativo. Se evaluaron los procesos logísticos previos a la implementación (gestión manual con registros en cuadernos y hojas de Excel no integradas) frente a la operativa actual con el sistema automatizado. Los resultados empíricos se detallan en la siguiente tabla:

4.5.1. Contraste de métricas de eficiencia operativa (Evaluación Técnica del Sistema)

Tabla 3.

Comparativo de indicadores de eficiencia operativa (Pre y Post implementación)

Indicador de Gestión (Dimensión a evaluar)	Método Manual (Antes)	Sistema Python - Intiqa Hotel (Después)	Impacto / Variación
Tiempo de consulta de stock (Ej. buscar saldo de aguas minerales o coca colas)	5 a 8 minutos por producto (búsqueda física en cuadernos o filtrado manual en Excel desordenado).	3 segundos (búsqueda automatizada en la interfaz por código o nombre).	Reducción del 99% en tiempos de consulta. (Eficiencia Operativa)
Tiempo de cuadre de inventario mensual	4 horas (suma manual, errores de calculadora, discrepancias entre hojas).	10 minutos (generación del reporte Kardex PDF con cálculo dinámico automático).	Reducción del 95% en tiempo de cierre. (Eficiencia Operativa)
Trazabilidad de salidas (Centros de costo)	0% (Los egresos se anotaban sin detallar si era para Cocina, Barra o Pisos).	100% (El sistema bloquea el egreso si no se selecciona el área de consumo).	Visibilidad total del gasto por área. (Trazabilidad)
Identificación de responsable del egreso	Nula (No existían usuarios ni contraseñas, cualquier persona podía modificar el archivo).	100% Auditado (Inicio de sesión con hash SHA-256, registro de usuario que ejecuta la transacción).	Registro claro y seguro de quién hizo el retiro. (Seguridad RBAC)

Indicador de Gestión (Dimensión a evaluar)	Método Manual (Antes)	Sistema Python - Intiqa Hotel (Después)	Impacto / Variación
Control de mermas y vencimientos (Ej. Yogures)	Reactiva (Se descubrían los productos vencidos cuando ya emanaban mal olor o al querer usarlos, generando pérdida directa).	Proactiva (Visualización inmediata del stock disponible y su rotación por área, permitiendo consumo FIFO).	Mitigación de pérdidas por descuido. (Control de Stock)
Alertas de desabastecimiento	Inexistentes (El almacén se quedaba sin productos sin previo aviso, afectando el servicio al huésped).	Automatizado (El sistema emite alertas visuales de "Stock Mínimo" al ingresar al módulo).	Eliminación de rupturas de stock. (Toma de Decisiones)

Nota: Elaboración propia basada en la evaluación de los procesos de almacén del Intiqa Hotel Puno (2025). Cabe precisar que los datos expuestos en la columna "Método Manual (Antes)" no son aproximaciones teóricas. Como se mencionó en la sección de metodología, el investigador ejercía labores de auditoría en el turno nocturno, lo que permitió cronometrar y tomar tiempos reales de las operaciones previas al sistema: desde los largos minutos invertidos en revisar cuadernos físicos hasta las horas destinadas al cierre mensual en hojas de cálculo. Además de cronometrar los tiempos, trabajar en ese puesto resultó clave para registrar los problemas reales que ocasionaba la falta de trazabilidad. Se pudieron documentar situaciones concretas, como la pérdida de bebidas de alta salida o el vencimiento de lácteos por no haberlos rotado bien. Estos hechos sirvieron como punto de partida para evaluar después el impacto del nuevo software.

Interpretación técnica: Tal como se observa en la Tabla 3, dejar atrás los métodos manuales para usar un sistema en Python no consistió únicamente en pasar la información a la computadora.

En realidad, este cambio mejoró por completo la forma de controlar el almacén. Antes no se tenía claro qué sección del hotel gastaba más dinero, pero ahora los cuadros de mando permiten ver esos costos de forma clara y dividida por áreas. Por otro lado, aplicar el control de acceso por roles (RBAC) solucionó de fondo el problema de la pérdida de bebidas y otros insumos que se piden con frecuencia. Debido a que cada movimiento queda registrado, la gerencia puede saber exactamente qué empleado sacó el producto y para qué área lo pidió.

4.6. Discusión de Resultados

Al comparar los datos de las Tablas 2 y 3 con la teoría y los estudios previos, se notan resultados que coinciden y se diferencian de lo encontrado por otros autores. Esto sirve para confirmar que la tecnología propuesta funciona.

Hay un punto en común muy claro con el trabajo de Alarcon Turpo y Quispe Cubas (2023) quienes analizaron el sector minero en Puno y lograron reducir los tiempos operativos usando el método Kardex. En este proyecto, los datos de la Tabla 3 muestran que el sistema creado en Python bajó el tiempo de cuadre mensual de 4 horas a solo 10 minutos. La diferencia principal es que ellos usaron un Kardex general, mientras que aquí se mejoró el método al sumarle el control por centros de costos para adaptarlo a la hotelería.

Por otro lado, la propuesta difiere de lo hecho por Acero Pilco (2024), a nivel local. Ese autor optó por un sistema web que necesita servidores para funcionar. Esta tesis plantea lo

contrario: como se ve en la Tabla 3, usar un programa local sin conexión a internet permitió acortar el tiempo de consulta de 5-8 minutos a 3 segundos, dejando de lado la necesidad de tener red.

Sobre el control de usuarios (RBAC), los resultados confirman que aplicar el Principio de Menor Privilegio fue una buena decisión. La Tabla 3 muestra cómo se pasó de no tener ningún control de identidad a tener un sistema 100% auditado con encriptación. Esto solucionó el problema de los movimientos anónimos, pero sin tener que pagar lo que cuesta un software corporativo.

Finalmente, en cuanto a la gestión del almacén, los hallazgos se alinean con lo que dice Christopher (2023) sobre la importancia de ver los gastos con claridad. La Tabla 3 detalla que el sistema aumentó la trazabilidad de 0% a 100%, lo que casi eliminó por completo las pérdidas ocultas. Además, el uso de alertas automáticas pone en práctica la idea de alertamiento temprano de Slack et al. (2022), Gracias a esto, el hotel dejó de reaccionar a último momento y ahora puede prever los problemas, lo que ayuda a cuidar las áreas internas (Back of House) y respalda lo que Kotler et al. (2021) dicen sobre la sostenibilidad del servicio hotelero.

4.7. Contrastación de Hipótesis

Para comprobar las hipótesis de esta investigación, se tomaron en cuenta dos elementos: los datos operativos de la Tabla 3 y la opinión de los usuarios reflejada en la escala Likert de la Tabla 2, donde se logró un promedio de 4.8 sobre 5.

Sin embargo, antes de revisar cada hipótesis, es necesario aclarar un punto importante sobre la metodología. El uso de la palabra relación en las hipótesis podría llevar a pensar obligatoriamente en pruebas tradicionales como Pearson o Spearman. No obstante, esas

herramientas estadísticas son inadecuadas para una población censal tan reducida (n=5). En la práctica real de la ingeniería de sistemas, la forma de comprobar que una tecnología impacta en un proceso no es solo a través de fórmulas de correlación, sino evaluando si el software cumplió su objetivo técnico (Tabla 3) y si el usuario final lo percibe como útil (Tabla 2). Es esta doble comprobación empírica la que fundamenta la decisión de aceptar las hipótesis a continuación.

- **Contrastación de la Hipótesis General (HG):** El desarrollo e implementación de un sistema de inventario en Python, estructurado bajo el enfoque de trazabilidad por centros de costos, se relaciona positivamente con la optimización de la gestión de almacén y la mitigación de las pérdidas no registradas de insumos en Intiqa Hotel Puno. Decisión: SE ACEPTA. Los datos duros de la Tabla 3 demuestran una reducción del 99% en tiempos de consulta y el paso de un 0% a un 100% de trazabilidad. Sumado a la percepción altamente positiva de los operadores (4.8/5.0 en la Tabla 2), se comprueba que existe una relación positiva entre la implementación del sistema y la optimización de la gestión lograda.
- **Contrastación de la HE1:** La ausencia actual de trazabilidad por áreas de consumo se asocia de manera directa con la pérdida injustificada de bienes de alta rotación (bebidas y alimentos) en Intiqa Hotel Puno. Decisión: SE ACEPTA. El diagnóstico previo (Tabla 3, Método Manual) evidenció la pérdida de bienes y mermas debido a la falta de asignación por áreas (0% de trazabilidad). La implementación del sistema, al hacer obligatoria la selección del centro de costos (100% de trazabilidad), confirma que al eliminar la ausencia de registros desagregados, se mitiga la pérdida injustificada de insumos.

- **Contrastación de la HE2:** El sistema de información en Python, al integrar control de acceso por roles y la vinculación digital de vales de salida con facturas de ingreso, permite dejar un registro claro de quién hizo cada movimiento que favorece la auditoría y el control transaccional del almacén. Decisión: SE ACEPTA. La evaluación técnica (Tabla 3) evidencia el paso de una identificación "Nula" a "100% Auditada" mediante el modelo RBAC con hash criptográfico (SHA-256), lo que valida que el sistema efectivamente permite establecer dicho Registro claro y seguro, resolviendo la problemática de registros anónimos.
- **Contrastación de la HE3:** La centralización de la información y la generación automatizada de reportes estadísticos desagregados por áreas de consumo se relacionan con una mayor eficiencia operativa (reducción de tiempos de consulta) y facilitan la toma de decisiones gerenciales en Intiqa Hotel Puno. Decisión: SE ACEPTA. La generación de dashboards en tiempo real permitió visualizar los costos por área (Cocina, Barra, Pisos), transformando la toma de decisiones de empírica a basada en datos. Esto se comprueba objetivamente con la reducción del tiempo de cuadre mensual de 4 horas a 10 minutos (Tabla 3) y la percepción favorable en la dimensión de eficiencia operativa por parte de los usuarios (Tabla 2).

CAPÍTULO V

CONCLUSIONES, RECOMENDACIONES Y TRABAJO FUTURO

5.1. Conclusiones

Luego de haber analizado los resultados obtenidos mediante la evaluación de métricas operativas de eficiencia y la validación empírica del sistema en el entorno real de Intiqa Hotel Puno, se formulan las siguientes conclusiones, guardando correspondencia directa con los objetivos e hipótesis planteados en la investigación:

Conclusión General: La investigación demostró empíricamente que el desarrollo e implementación del sistema de inventario en Python logró mejorar significativamente la gestión de almacén en Intiqa Hotel Puno durante el año 2025. El sistema logró transformar un modelo logístico reactivo, basado en registros manuales descentralizados que generaban donde no se sabía qué área gastaba más y mermas operativas (como el vencimiento de lácteos o la pérdida de bebidas), en un modelo proactivo, auditado y trazable, validando la hipótesis general de la investigación. Esta optimización se materializó mediante la reducción del 99% en los tiempos de consulta de stock (de minutos a segundos) y del 95% en los tiempos de cuadre mensual (de 4 horas a 10 minutos), sumado a una percepción de satisfacción del 4.8/5.0 por parte de los operadores. Todo esto se consiguió usando una arquitectura de software local. Esta opción demostró ser una alternativa viable y gratuita en comparación con los caros sistemas ERP comerciales.

Conclusiones Específicas:

1. **Sobre el diagnóstico y la trazabilidad (OE1 / HE1):** Se comprobó que la principal causa de la pérdida de insumos era la falta de un control que rastreara los productos por cada área de consumo. Al obligar al sistema a registrar el área de destino (Cocina, Barra, Pisos) en el 100% de las salidas, se logró identificar con exactitud qué sección del hotel gasta más dinero. Esto ayudó a reducir las mermas, acabó con la falta de responsabilidad sobre los productos y permitió que la gerencia tomara decisiones basándose en datos reales.
2. **Sobre el diseño e implementación técnica (OE2 / HE2):** Se desarrolló un programa sólido que reúne varias herramientas: un motor Kardex para inventario perpetuo, alertas de stock mínimo y un control de usuarios por roles (RBAC) usando encriptación SHA-256. También se integraron librerías como Matplotlib para crear los cuadros de mando (Dashboards), los cuales resumen la información del almacén. Gracias a estas funciones, la auditoría interna se vuelve mucho más sencilla y todo funciona sin necesidad de tener conexión a internet.
3. **Sobre la eficiencia y toma de decisiones (OE3 / HE3):** La evaluación técnica y la percepción de los usuarios reflejaron un nivel de aceptación "Muy Alto" (promedio Likert de 4.8/5.0), evidenciando que la automatización eliminó los errores de cálculo manual a fin de mes, y proporcionó a la gerencia reportes confiables en PDF y Excel que transformaron la toma de decisiones de un modelo empírico a uno basado en datos, mejorando sustancialmente la velocidad y calidad en la gestión gerencial.

5.2. Recomendaciones

Con base en los hallazgos y la experiencia adquirida durante el desarrollo de la presente investigación, se emiten las siguientes recomendaciones:

• Para la Gerencia del Intiqa Hotel Puno:

- Institucionalizar el uso exclusivo del sistema para toda transacción de almacén, prohibiendo el uso paralelo de cuadernos o hojas de cálculo no oficiales, para evitar la dispersión de la información.
- Utilizar mensualmente el reporte de "Gastos por Área" (Dashboard) como base para evaluar el presupuesto asignado a cada centro de costos (ej. reducir el cupo de pedido si un área genera excesivo desperdicio).

• Para los Operarios del Almacén:

- Respetar estrictamente el principio de seguridad RBAC: No compartir credenciales de acceso entre personal, ya que esto destruye el registro claro y seguro que garantiza el sistema.
- Realizar copias de seguridad (Backup) de la base de datos (kardex.db) de forma semanal mediante la herramienta integrada en el sistema, guardando el archivo en un pendrive externo para prevenir pérdida de información por fallas de hardware.

• Para futuros Tesistas de Ingeniería de Sistemas:

- Considerar la metodología de arquitectura local (Python + SQLite) como una primera viable y de altísimo impacto para PYMES del sector servicios que no cuentan con presupuesto para infraestructura en la nube.

5.3. Trabajo Futuro

Si bien el sistema desarrollado cumple a cabalidad con los objetivos planteados y resuelve el problema inmediato del hotel, la evolución constante de la ingeniería de software permite vislumbrar las siguientes líneas de mejora a futuro para escalar la solución:

1. Migración a Arquitectura Web y en la Nube: mejorar la interfaz actual por un diseño web moderno de escritorio (Tkinter) hacia un framework web moderno (como React o Vue.js) conectado a un backend en FastAPI o Django, y migrar la base de datos de SQLite a PostgreSQL en un servicio en la nube (AWS o Google Cloud). Esto permitiría que la gerencia consulte los dashboards desde el celular en tiempo real, sin estar físicamente en el hotel.
2. Integración de Inteligencia Artificial (Machine Learning): Utilizar los datos históricos acumulados en el motor Kardex para desarrollar modelos matemáticos que permitan adivinar la demanda (usando librerías como Scikit-Learn de Python). El sistema no solo alertaría cuando el stock está bajo, sino que predeciría con anticipación cuántos insumos de cada tipo se necesitarán la próxima semana en base a la tasa de ocupación histórica del hotel, optimizando así las compras a proveedores.
3. Internet de las Cosas (IoT) para trazabilidad automática: Integrar lectores de código de barras o etiquetas RFID en los puntos de egreso, de modo que el operador solo tenga que escanear el producto, y el sistema asigne automáticamente el área de consumo y descuente el stock, reduciendo el tiempo de registro a milisegundos y eliminando el error de tipeo manual.

REFERENCIAS BIBLIOGRÁFICAS

Acero Pilco, L. A. (2024). *Desarrollo de un sistema web para mejorar el control de inventarios de la Botica Sud Puno 2024* [Tesis Repositorio, Universidad Andina Néstor Cáceres Velásquez]. https://repositorio.uancv.edu.pe/items/cb8bfa1c-dd67-4247-a6fb-4fcf9e999b4d

Adaile Benítez, N. T. y Camacho Vázquez, M. R. y Cárdenas Castellanos, J. y Pérez Méndez, J. y Martínez García, J. E. (2025). Plataforma web para la gestión eficiente de eventos académicos utilizando la metodología RAD. *Revista Multidisciplinaria Voces de América y El Caribe*, *2*(1), 585–616. https://doi.org/10.69821/remuvac.v2i1.145

Aguilar Moya, R. C. (2024). *Mejora del sistema de almacén ventas en la empresa Prisontec Juliaca mediante un sistema web 2024* [Tesis Pregrado, Universidad Andina Néstor Cáceres Velásquez].

Alarcon Turpo, J. M. y Quispe Cubas, M. A. (2023). *Gestión de inventarios para mejorar la productividad en un almacén del sector minero, Puno 2023* [Tesis Repositorio, Universidad César Vallejo]. https://repositorio.ucv.edu.pe/bitstream/handle/20.500.12692/139785/Alarcon_TJM-Quispe_CMA-SD.pdf?sequence=1&isAllowed=y

Andi Tapuy, S. P. (2025). *Desarrollo de una Aplicación Móvil con Base de Datos Local Para la Gestión de Ventas e Inventario de la Tienda Génesis*. [Tesis pregrado, Instituto Superior Tecnológico Tena].

Chapple, M. y Seidl, D. (2024). *ISC2 ® CISSP ® Certified Information Systems Security Professional Official Practice Tests Fourth Edition* (4th ed.) Inc. John Wiley y Sons.

Chopra, S. y Meindl, P. (2013). *Administración de la cadena de suministro Estrategia, planeación y operación* (5th ed.) Pearson Education Limited.

Christopher, M. (2023). *Logistics and Supply Chain Management* (6th ed)Pearson Education Limited, https://issuu.com/education_library/docs/20787

Cochran, W. G. (1977). *Sampling techniques.* (Inc. John Wiley y Sons, Ed.; 3rd ed.). Wiley.

Da Silva Rodriguez, S. A. (2020). *Facultad de Administración Hotelera, Turismo y Gastronomía para optimizar la rotación de insumos perecibles* [Tesis de Pregrado, Universidad San Ignacio de Loyola]. https://repositorio.usil.edu.pe/entities/publication/9c54da8c-ecdc-4bae-9ba0-fd7a6be3d21e

Dueñas Feuerbach, M. (2022). *Uso de dashboards para la explotación de perspectivas para la toma de decisiones comerciales en una empresa industrial* [Repositorio, Escola Tècnica Superior d'Enginyeria Industrial de Barcelona].

Field, A. (2018). *Discovering Statistics Using IBM SPSS Statistics* (SAGE Publications., Ed.).

Heizer, J. y Render, B. y Munson, C. (2020). *Principles of Operations Management Sustainability and Supply Chain Management* (11th ed.). Pearson. www.pearson.com/mylab/operations-management

Hernández Sampieri, R. y Fernández Collado, C. y Baptista Lucio, M. del P. (2014). *Metodología de la Investigación Sexta Edición* (McGraw-Hill Education, Ed.; 6ta ed.). https://www.esup.edu.pe/wp-content/uploads/2020/12/2.%20Hernandez,%20Fernandez%20y%20Baptista-metodolog%C3%ADa%20Investigacion%20Cientifica%206ta%20ed.pdf

Hortigüela Alcazar, G. (2025). *Desarrollo de una interfaz gráfica de usuario en Python para ejecutar tareas de forma síncrona y asíncrona* [Memoria Repositorio, Escola Tècnica Superior d'Enginyeria Industrial de Barcelona]. https://upcommons.upc.edu/entities/publication/21419dcd-db24-4dfe-9f34-584da1770fa9

Humbatov, J. (2024). In-depth Analysis of Global Supply Chain Dynamics and its Core Components. *IMAQ Journal A Canadian Trans-disciplinary Research Journal*, *1*, 9–16.

Kotler, P. y Bowen, J. T. y Baloglu, S. (2021). *Marketing for hospitality and tourism* (Pearson Education Limited, Ed.; 8th ed.). Pearson. https://api.pageplace.de/preview/DT0400.9781292363523_A42098572/preview-9781292363523_A42098572.pdf

Lorenzo Guerrero, C. (2025). *Diseño e Implementación de un Sistema de Gestión Hotelero con Base en la Metodología de scrum*. [Tesis de pregrado, Universidad Nacional Autónoma de México].

Mejia Nuñez, A. M. (2025). *Optimización de la Gestión de Inventarios Mediante Análisis de Ventas y Modelado Predictivo con Python Utilizando una Base de Datos de Amazon del año 2022*. [Tesis de especialización,Universidad Nacional Abierta y a Distancia UNAD].

Mercado Pavón, R. D. (2023). Bases de datos en aplicaciones móviles para sistema operativo Android. *ARANDU-UTIC – Revista Científica Internacional*, *10*, 17–37. https://www.uticvirtual.edu.py/revista.ojs/index.php/revistas/article/view/154/254

Parisuaña Cusilayme, W. (2024). *Sistema web de ventas para la gestión del control e inventario en la planta procesadora de lácteos Grupo Yaguno SAC 2021* [Tesis Pregrado, Universidad Nacional del Altiplano].

Peralta Ranilla, L. L. y Ranilla Alvarez, Y. (2022). *Estrategias del control de inventarios en el Hotel La Posada de Ligarte - Arequipa, 2022* [Tesis Pregrado, Universidad María Auxiliadora]. https://orcid.org/0000-0002-3807-4369

Sánchez Alberca, A. (2020). *Aprende Phyton*. https://aprendeconalf.es/docencia/python/

Slack, N. y Brandon-Jones, A. y Burgess, N. (2022). *Operations Management* (10th ed.). Pearson.

Ticona Vargas, E. J. (2025). *Implementación de un sistema web para mejorar la gestión de inventarios de Newtex Juliaca 2024* [Tesis Pregrado. Universidad Andina Néstor Cáceres Velásquez].

ANEXO 1: INSTRUMENTO DE RECOLECCIÓN DE DATOS

CUESTIONARIO DE EVALUACIÓN DEL SISTEMA DE INVENTARIO INTIQA HOTEL

I. Datos Generales

- Empresa: Intiqa Hotel Puno
- Área de evaluación: Almacén / Logística
- Objetivo: Evaluar el impacto del sistema de inventario en Python en la optimización de la gestión de almacén.

II. Instrucciones

Estimado colaborador, el presente cuestionario tiene como objetivo recoger información técnica sobre la operatividad del nuevo Sistema de Inventario. Le pedimos leer detenidamente cada afirmación y marcar con una "X" en la casilla que mejor represente su experiencia real, considerando la siguiente escala:

(1) Totalmente en desacuerdo

(2) En desacuerdo

(3) Neutral

(4) De acuerdo

(5) Totalmente de acuerdo

III. Cuerpo del Cuestionario

N°	Ítem (Pregunta)	1	2	3	4	5
DIMENSIÓN: Control de Stock y Mitigación de Pérdidas						
1	Antes de implementar el sistema, existían pérdidas de insumos (bebidas, lácteos) porque no se sabía qué área los retiraba.					

N°	Ítem (Pregunta)	1	2	3	4	5
2	Actualmente, el registro de "Áreas" (centros de costos) al sacar un producto permite saber exactamente qué sector gasta más.					
DIMENSIÓN: Control de Acceso y Seguridad (RBAC)						
3	El uso de usuarios y contraseñas (Login) hace que yo me sienta responsable de los movimientos que registro.					
4	El hecho de que el sistema bloquee opciones según el rol (Admin/Usuario) evita que se borre o modifique información sin permiso.					
DIMENSIÓN: Automatización e Inteligencia de Datos						
5	La visualización de los gráficos (torta/barras) me ayuda a entender rápidamente qué área consume más insumos.					
6	Las alertas de "Stock Bajo" son útiles para pedir productos a tiempo y evitar que se acaben en el hotel.					

N°	Ítem (Pregunta)	1	2	3	4	5
7	Generar reportes en PDF o Excel desde el sistema es más rápido y seguro que hacerlo en cuadernos.					
DIMENSIÓN: Eficiencia Operativa						
8	El tiempo para buscar un producto o consultar el stock actual se ha reducido drásticamente con el sistema.					
9	El sistema ha permitido reducir los errores matemáticos al momento de cuadrar el inventario a fin de mes.					
DIMENSIÓN: Toma de Decisiones Gerenciales						
10	La información que muestra el sistema es confiable para que la gerencia tome decisiones sobre las compras del hotel.					

ANEXO 2: CÓDIGO FUENTE DEL SISTEMA

Especificaciones Técnicas del Entorno de Desarrollo

Lenguaje de Programación: Python (Versión 3.14)

Interfaz Gráfica: Tkinter (Librería estándar nativa)

Motor de Base de Datos: SQLite 3 (Integrado en Python)

Librerías Externas Utilizadas:

- **openpyxl** (Para exportación a Microsoft Excel).
- **plyer** (Para notificaciones de escritorio nativas).
- **pystray** y **PIL** (Para minimización a la bandeja del sistema).
- **reportlab** (Para la generación de reportes en formato PDF).
- **matplotlib** (Para el renderizado de Dashboards gráficos).
- **hashlib** (Para el cifrado SHA-256 de contraseñas).

Script Principal (sistema_kardex_intiqa.py)

```
import tkinter as tk
from tkinter import messagebox, simpledialog, filedialog
import sys
import os
import json
import tempfile
import sqlite3
import openpyxl
from plyer import notification
import pystray
from pystray import MenuItem as item
from PIL import Image, ImageTk
import threading
from tkinter import ttk
from reportlab.pdfgen import canvas
from reportlab.lib.pagesizes import letter
from reportlab.lib.units import inch
import shutil
from datetime import datetime, timedelta
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.figure import Figure
import hashlib

# ---------------- CONFIGURACIÓN DE BASE DE DATOS ----------------
DB_FILE = "kardex.db"
JSON_FILE = "kardex_data.json"

def hash_password(password):
    return hashlib.sha256(password.encode('utf-8')).hexdigest()

def conectar_db():
```

```
    conn = sqlite3.connect(DB_FILE)
    conn.row_factory = sqlite3.Row
    return conn

def crear_tablas_si_no_existen():
    conn = conectar_db()
    cursor = conn.cursor()
    cursor.execute('''CREATE TABLE IF NOT EXISTS usuarios (username TEXT PRIMARY
KEY, password TEXT NOT NULL, role TEXT NOT NULL)''')
    cursor.execute('''CREATE TABLE IF NOT EXISTS productos (codigo TEXT PRIMARY
KEY, nombre TEXT NOT NULL, stock_minimo INTEGER NOT NULL)''')
    cursor.execute('''CREATE TABLE IF NOT EXISTS movimientos_kardex (id INTEGER
PRIMARY KEY AUTOINCREMENT, producto_codigo TEXT NOT NULL, fecha TEXT NOT NULL, tipo
TEXT NOT NULL, cantidad INTEGER NOT NULL, precio_unitario REAL, comprobante TEXT,
responsable TEXT, area TEXT, FOREIGN KEY (producto_codigo) REFERENCES productos
(codigo))''')
    cursor.execute('''CREATE TABLE IF NOT EXISTS areas (nombre TEXT PRIMARY
KEY)''')

    cursor.execute("SELECT COUNT(*) FROM areas")
    if cursor.fetchone()[0] == 0:
        areas_iniciales = ["Administración", "Logística", "Ventas", "Producción",
"TI", "RRHH", "Marketing", "Mantenimiento", "Cafetería", "Limpieza", "Otro"]
        for area in areas_iniciales:
            cursor.execute("INSERT OR IGNORE INTO areas (nombre) VALUES (?)",
(area,))

    cursor.execute("SELECT COUNT(*) FROM usuarios")
    if cursor.fetchone()[0] == 0:
        cursor.execute("INSERT INTO usuarios (username, password, role) VALUES (?,
?, ?)",
                       ('admin', hash_password('admin'), 'admin'))

    try:
        cursor.execute("ALTER TABLE movimientos_kardex ADD COLUMN area TEXT")
    except sqlite3.OperationalError:
        pass
    conn.commit()
    conn.close()

def migrar_datos_desde_json():
    if not os.path.exists(JSON_FILE):
        return
    print("Iniciando migración de datos desde JSON...")
    try:
        with open(JSON_FILE, 'r', encoding='utf-8') as f:
            datos_antiguos = json.load(f)
        conn = conectar_db()
        cursor = conn.cursor()
        usuarios_antiguos = datos_antiguos.get("usuarios", {})
        for user, passw in usuarios_antiguos.items():
            role = "admin" if user == "admin" else "usuario"
            cursor.execute("INSERT OR IGNORE INTO usuarios (username, password,
role) VALUES (?, ?, ?)",
                           (user, hash_password(passw), role))
        productos_antiguos = datos_antiguos.get("productos", {})
        stock_minimo_antiguo = datos_antiguos.get("stock_minimo", {})
        for codigo, nombre in productos_antiguos.items():
            minimo = stock_minimo_antiguo.get(codigo, 0)
            cursor.execute("INSERT OR IGNORE INTO productos (codigo, nombre,
stock_minimo) VALUES (?, ?, ?)", (codigo, nombre, minimo))
        kardex_antiguo = datos_antiguos.get("kardex", {})
        for codigo, movimientos in kardex_antiguo.items():
```

```
            for mov in movimientos:
                fecha_orig = mov.get("fecha")
                fecha_db = formatear_fecha_almacenar(fecha_orig) if fecha_orig else
datetime.now().strftime('%Y-%m-%d')
                cursor.execute('''INSERT INTO movimientos_kardex (producto_codigo,
fecha, tipo, cantidad, precio_unitario, comprobante, responsable) VALUES (?, ?, ?,
?, ?, ?, ?)''',
                               (codigo, fecha_db, mov.get("tipo"),
mov.get("cantidad"), mov.get("precio_unitario"), mov.get("comprobante"),
mov.get("responsable")))
        conn.commit()
        conn.close()
        os.rename(JSON_FILE, f"{JSON_FILE}.migrated")
        print("Migración completada con éxito.")
    except Exception as e:
        messagebox.showerror("Error de Migración", f"No se pudo migrar los datos
desde el archivo JSON:\n{e}")

# ---------------- VARIABLES GLOBALES ----------------
rol_usuario = None
nombre_usuario = None
icon_tray = None
root = None
frame_saldos = None

# ---------------- FUNCIONES AUXILIARES ----------------
def resource_path(relative_path):
    try: base_path = sys._MEIPASS
    except Exception: base_path = os.path.abspath(".")
    return os.path.join(base_path, relative_path)

icono_path = resource_path("img/logo_intiqa__rMV_icon.ico")
if not os.path.exists(icono_path):
    print(f"Advertencia: Icono no encontrado en {icono_path}.")
    icono_path = None

def notificar(titulo, mensaje):
    try:
        notification.notify(title=titulo, message=mensaje, app_icon=icono_path,
timeout=5)
    except Exception as e:
        print(f"No se pudo mostrar la notificación: {e}")

def formatear_fecha_almacenar(fecha_str):
    try:
        partes = fecha_str.split('-')
        if len(partes) == 3 and len(partes[2]) == 4:
            return f"{partes[2]}-{partes[1]}-{partes[0]}"
        return None
    except Exception:
        return None

def formatear_fecha_mostrar(fecha_str):
    try:
        partes = fecha_str.split('-')
        if len(partes) == 3 and len(partes[0]) == 4:
            return f"{partes[2]}-{partes[1]}-{partes[0]}"
        return fecha_str
    except Exception:
        return fecha_str

def validar_fecha(fecha):
    try:
```

```
            partes = fecha.split('-')
            if len(partes) != 3: return False
            dia, mes, anio = int(partes[0]), int(partes[1]), int(partes[2])
            return (1 <= mes <= 12) and (1 <= dia <= 31) and (len(str(anio)) == 4)
        except Exception:
            return False

def centrar_ventana(ventana, modal=False):
    ventana.update_idletasks()
    ancho_pantalla = ventana.winfo_screenwidth()
    alto_pantalla = ventana.winfo_screenheight()
    ancho_ventana = ventana.winfo_width()
    alto_ventana = ventana.winfo_height()
    x = (ancho_pantalla // 2) - (ancho_ventana // 2)
    y = (alto_pantalla // 2) - (alto_ventana // 2)
    ventana.geometry(f"+{x}+{y}")
    if modal:
        ventana.transient(root)
        ventana.grab_set()

# --- SPLASH SCREEN (EFECTO WORD) ---
def mostrar_splash_screen(parent, iniciar_sistema_callback):
    splash = tk.Toplevel(parent) # Se adhiere a la ventana principal para no crear
'tk' fantasma
    splash.overrideredirect(True)
    splash.attributes("-topmost", True)
    splash.configure(bg="#01525f")

    ancho_splash = 500
    alto_splash = 300
    x = (splash.winfo_screenwidth() // 2) - (ancho_splash // 2)
    y = (splash.winfo_screenheight() // 2) - (alto_splash // 2)
    splash.geometry(f"{ancho_splash}x{alto_splash}+{x}+{y}")

    try:
        if icono_path:
            img = Image.open(icono_path)
            img = img.resize((120, 120), Image.Resampling.LANCZOS)
            photo = ImageTk.PhotoImage(img)
            lbl_img = tk.Label(splash, image=photo, bg="#01525f")
            lbl_img.image = photo
            lbl_img.pack(pady=(40, 10))
    except Exception:
        pass

    tk.Label(splash, text="SISTEMA KARDEX INTIQA", font=("Arial", 22, "bold"),
fg="white", bg="#01525f").pack()
    tk.Label(splash, text="Cargando módulos, por favor espere...", font=("Arial",
11), fg="#e49f54", bg="#01525f").pack(pady=(5, 20))

    estilo = ttk.Style()
    estilo.theme_use('default')
    estilo.configure("Verde.Horizontal.TProgressbar", troughcolor='#014251',
background='#e49f54', thickness=15)

    progress = ttk.Progressbar(splash, orient="horizontal", length=400,
mode="indeterminate", style="Verde.Horizontal.TProgressbar")
    progress.pack(pady=10)
    progress.start(15)

    def cerrar_splash():
        progress.stop()
        splash.destroy()
```

```
        iniciar_sistema_callback() # Llama a la función que muestra la ventana
principal

    splash.after(3000, cerrar_splash)

# ---------------- FUNCIONES DE GESTIÓN DE ÁREAS ----------------
def obtener_areas_bd():
    try:
        with sqlite3.connect(DB_FILE) as conn:
            conn.row_factory = sqlite3.Row
            cursor = conn.cursor()
            cursor.execute("SELECT nombre FROM areas ORDER BY nombre")
            areas = [fila[0] for fila in cursor.fetchall()]
            return areas if areas else []
    except Exception as e:
        print(f"Error al obtener áreas: {e}")
        return []

def agregar_area_rapida(combobox_widget, parent_window):
    nueva_area = simpledialog.askstring("Nueva Área", "Ingrese el nombre de la
nueva área:", parent=parent_window)
    if nueva_area and nueva_area.strip():
        nueva_area = nueva_area.strip().title()
        try:
            with sqlite3.connect(DB_FILE) as conn:
                cursor = conn.cursor()
                cursor.execute("INSERT OR IGNORE INTO areas (nombre) VALUES (?)",
(nueva_area,))
                if cursor.rowcount > 0:
                    combobox_widget['values'] = obtener_areas_bd()
                    combobox_widget.set(nueva_area)
                    messagebox.showinfo("Éxito", f"Área '{nueva_area}' creada y
seleccionada.", parent=parent_window)
                else:
                    messagebox.showwarning("Ya existe", f"El área '{nueva_area}' ya
existe. Seleccionándola.", parent=parent_window)
                    combobox_widget.set(nueva_area)
        except Exception as e:
            messagebox.showerror("Error", f"No se pudo guardar el área: {e}",
parent=parent_window)

def gestionar_areas_gui():
    if rol_usuario != "admin":
        messagebox.showerror("Acceso denegado", "Solo el administrador puede
gestionar las áreas.")
        return

    ventana_areas = tk.Toplevel(root)
    ventana_areas.title("Gestionar Áreas del Sistema")
    ventana_areas.geometry("400x450")
    ventana_areas.resizable(False, False)
    if icono_path: ventana_areas.iconbitmap(icono_path)
    ventana_areas.configure(bg="#f0f0f0")
    centrar_ventana(ventana_areas)

    frame_form = tk.Frame(ventana_areas, bg="#f0f0f0")
    frame_form.pack(padx=20, pady=20, fill=tk.X)

    tk.Label(frame_form, text="Nueva Área:", bg="#f0f0f0", font=("Arial", 11,
"bold")).grid(row=0, column=0, sticky="w", pady=5)
    entry_nueva_area = tk.Entry(frame_form, width=25, font=("Arial", 11))
    entry_nueva_area.grid(row=0, column=1, pady=5, padx=5)
```

```
    frame_lista = tk.Frame(ventana_areas, bg="#f0f0f0")
    frame_lista.pack(fill=tk.BOTH, expand=True, padx=20)

    listbox_areas = tk.Listbox(frame_lista, font=("Arial", 11), bg="white")
    listbox_areas.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)
    scrollbar = tk.Scrollbar(frame_lista, command=listbox_areas.yview)
    listbox_areas.config(yscrollcommand=scrollbar.set)
    scrollbar.pack(side=tk.RIGHT, fill=tk.Y)

    def actualizar_lista():
        listbox_areas.delete(0, tk.END)
        for area in obtener_areas_bd():
            listbox_areas.insert(tk.END, area)

    def agregar_area():
        nueva = entry_nueva_area.get().strip().title()
        if not nueva:
            messagebox.showwarning("Vacío", "Ingrese un nombre para el área.",
parent=ventana_areas)
            return
        try:
            with sqlite3.connect(DB_FILE) as conn:
                cursor = conn.cursor()
                cursor.execute("INSERT INTO areas (nombre) VALUES (?)", (nueva,))
            entry_nueva_area.delete(0, tk.END)
            actualizar_lista()
            messagebox.showinfo("Éxito", f"Área '{nueva}' agregada correctamente.",
parent=ventana_areas)
        except sqlite3.IntegrityError:
            messagebox.showerror("Error", f"El área '{nueva}' ya existe.",
parent=ventana_areas)

    def eliminar_area():
        seleccion = listbox_areas.curselection()
        if not seleccion:
            messagebox.showwarning("Sin selección", "Seleccione un área para
eliminar.", parent=ventana_areas)
            return
        area_a_borrar = listbox_areas.get(seleccion[0])
        if messagebox.askyesno("Confirmar", f"¿Está seguro de eliminar el área
'{area_a_borrar}'?", parent=ventana_areas):
            with sqlite3.connect(DB_FILE) as conn:
                cursor = conn.cursor()
                cursor.execute("DELETE FROM areas WHERE nombre = ?",
(area_a_borrar,))
            actualizar_lista()

    frame_botones = tk.Frame(ventana_areas, bg="#f0f0f0")
    frame_botones.pack(pady=15)
    tk.Button(frame_botones, text="Agregar", command=agregar_area, bg="#4CAF50",
fg="white", font=("Arial", 10, "bold"), width=10).pack(side=tk.LEFT, padx=5)
    tk.Button(frame_botones, text="Eliminar", command=eliminar_area, bg="#ff4d4d",
fg="white", font=("Arial", 10, "bold"), width=10).pack(side=tk.LEFT, padx=5)
    tk.Button(frame_botones, text="Cerrar", command=ventana_areas.destroy,
font=("Arial", 10, "bold"), width=10).pack(side=tk.LEFT, padx=5)

    actualizar_lista()

# ---------------- FUNCIONES DEL SISTEMA ----------------
def agregar_producto_gui():
    if rol_usuario != "admin":
        messagebox.showerror("Acceso denegado", "Solo el administrador puede
agregar productos.")
```

```
        return

    ventana_agregar = tk.Toplevel(root)
    ventana_agregar.title("Agregar Nuevo Producto")
    ventana_agregar.geometry("400x250")
    ventana_agregar.resizable(False, False)
    if icono_path: ventana_agregar.iconbitmap(icono_path)
    ventana_agregar.configure(bg="#f0f0f0")
    centrar_ventana(ventana_agregar)

    frame_form = tk.Frame(ventana_agregar, bg="#f0f0f0")
    frame_form.pack(padx=20, pady=20)

    tk.Label(frame_form, text="Código del Producto:", bg="#f0f0f0").grid(row=0,
column=0, sticky="w", pady=5)
    entry_codigo = tk.Entry(frame_form, width=30)
    entry_codigo.grid(row=0, column=1, pady=5)

    tk.Label(frame_form, text="Nombre del Producto:", bg="#f0f0f0").grid(row=1,
column=0, sticky="w", pady=5)
    entry_nombre = tk.Entry(frame_form, width=30)
    entry_nombre.grid(row=1, column=1, pady=5)

    tk.Label(frame_form, text="Stock Mínimo:", bg="#f0f0f0").grid(row=2, column=0,
sticky="w", pady=5)
    entry_minimo = tk.Entry(frame_form, width=30)
    entry_minimo.grid(row=2, column=1, pady=5)

    def guardar_producto():
        codigo = entry_codigo.get().strip()
        nombre = entry_nombre.get().strip()
        minimo_str = entry_minimo.get().strip()

        if not all([codigo, nombre, minimo_str]):
            messagebox.showerror("Error", "Todos los campos son obligatorios.",
parent=ventana_agregar)
            return

        try:
            minimo = int(minimo_str)
        except ValueError:
            messagebox.showerror("Error", "El stock mínimo debe ser un número
entero.", parent=ventana_agregar)
            return

        try:
            with sqlite3.connect(DB_FILE) as conn:
                cursor = conn.cursor()
                cursor.execute("INSERT INTO productos (codigo, nombre,
stock_minimo) VALUES (?, ?, ?)", (codigo, nombre, minimo))
            messagebox.showinfo("Éxito", f"Producto '{nombre}' agregado con
éxito.", parent=ventana_agregar)
            notificar("Producto agregado", f"'{nombre}' agregado correctamente.")
            actualizar_saldos()
            ventana_agregar.destroy()
        except sqlite3.IntegrityError:
            messagebox.showerror("Error", f"Ya existe un producto con el código
'{codigo}'.", parent=ventana_agregar)
        except Exception as e:
            messagebox.showerror("Error de Base de Datos", f"No se pudo agregar el
producto.\nError: {e}", parent=ventana_agregar)

    frame_botones = tk.Frame(ventana_agregar, bg="#f0f0f0")
```

```
    frame_botones.pack(pady=10)
    tk.Button(frame_botones, text="Guardar Producto", command=guardar_producto,
bg="#e49f54", fg="white", font=("Arial", 10, "bold")).pack(side=tk.LEFT, padx=10)
    tk.Button(frame_botones, text="Cancelar", command=ventana_agregar.destroy,
font=("Arial", 10)).pack(side=tk.LEFT, padx=10)

def gestionar_productos_gui():
    if rol_usuario != "admin":
        messagebox.showerror("Acceso denegado", "Solo el administrador puede
gestionar productos.")
        return

    ventana = tk.Toplevel(root)
    ventana.title("Gestionar Productos")
    ventana.geometry("700x450")
    if icono_path: ventana.iconbitmap(icono_path)
    centrar_ventana(ventana)

    frame = tk.Frame(ventana)
    frame.pack(fill=tk.BOTH, expand=True, padx=10, pady=10)

    tree = ttk.Treeview(frame, columns=('codigo', 'nombre', 'minimo', 'stock'),
show='headings')
    tree.heading('codigo', text='Código'); tree.heading('nombre', text='Nombre')
    tree.heading('minimo', text='Stock Mínimo'); tree.heading('stock', text='Stock
Actual')
    tree.column('codigo', width=100, anchor=tk.CENTER); tree.column('nombre',
width=300, anchor=tk.W)
    tree.column('minimo', width=100, anchor=tk.CENTER); tree.column('stock',
width=100, anchor=tk.CENTER)

    scrollbar = ttk.Scrollbar(frame, orient=tk.VERTICAL, command=tree.yview)
    tree.configure(yscroll=scrollbar.set)
    tree.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)
    scrollbar.pack(side=tk.RIGHT, fill=tk.Y)

    def actualizar_lista():
        for i in tree.get_children(): tree.delete(i)
        with sqlite3.connect(DB_FILE) as conn:
            conn.row_factory = sqlite3.Row
            cursor = conn.cursor()
            cursor.execute("SELECT * FROM productos ORDER BY nombre")
            for prod in cursor.fetchall():
                stock = calcular_saldo(prod['codigo'])
                tree.insert('', tk.END, values=(prod['codigo'], prod['nombre'],
prod['stock_minimo'], stock))

    def editar_producto():
        sel = tree.focus()
        if not sel:
            messagebox.showwarning("Sin selección", "Seleccione un producto.",
parent=ventana)
            return
        vals = tree.item(sel)['values']
        old_codigo, old_nombre, old_minimo = vals[0], vals[1], vals[2]

        edit_win = tk.Toplevel(ventana)
        edit_win.title(f"Editar Producto: {old_codigo}")
        edit_win.geometry("400x200")
        edit_win.configure(bg="#f0f0f0")
        centrar_ventana(edit_win)
```

```
        tk.Label(edit_win, text="Nombre:", bg="#f0f0f0").grid(row=0, column=0,
padx=10, pady=10, sticky="w")
        e_nombre = tk.Entry(edit_win, width=30); e_nombre.insert(0, old_nombre);
e_nombre.grid(row=0, column=1)

        tk.Label(edit_win, text="Stock Mínimo:", bg="#f0f0f0").grid(row=1,
column=0, padx=10, pady=10, sticky="w")
        e_minimo = tk.Entry(edit_win, width=30); e_minimo.insert(0, old_minimo);
e_minimo.grid(row=1, column=1)

        def guardar():
            nn = e_nombre.get().strip(); nm = e_minimo.get().strip()
            if not nn or not nm:
                messagebox.showerror("Error", "Campos vacíos.", parent=edit_win);
return
            try:
                with sqlite3.connect(DB_FILE) as conn:
                    cursor = conn.cursor()
                    cursor.execute("UPDATE productos SET nombre=?, stock_minimo=?
WHERE codigo=?", (nn, int(nm), old_codigo))
                messagebox.showinfo("Éxito", "Actualizado.", parent=edit_win)
                edit_win.destroy(); actualizar_lista(); actualizar_saldos()
            except Exception as e:
                messagebox.showerror("Error", str(e), parent=edit_win)

        tk.Button(edit_win, text="Guardar", command=guardar, bg="#2196F3",
fg="white", font=("Arial", 10, "bold")).grid(row=2, column=0, columnspan=2,
pady=20)

    def eliminar_producto():
        sel = tree.focus()
        if not sel:
            messagebox.showwarning("Sin selección", "Seleccione un producto.",
parent=ventana)
            return
        codigo = tree.item(sel)['values'][0]
        nombre = tree.item(sel)['values'][1]

        with sqlite3.connect(DB_FILE) as conn:
            cursor = conn.cursor()
            cursor.execute("SELECT COUNT(*) FROM movimientos_kardex WHERE
producto_codigo = ?", (codigo,))
            movimientos = cursor.fetchone()[0]

        if movimientos > 0:
            messagebox.showerror("Error", f"No se puede eliminar '{nombre}' porque
tiene {movimientos} movimientos.", parent=ventana)
            return

        if messagebox.askyesno("Confirmar", f"¿Eliminar '{nombre}'?",
parent=ventana):
            with sqlite3.connect(DB_FILE) as conn:
                cursor = conn.cursor()
                cursor.execute("DELETE FROM productos WHERE codigo = ?", (codigo,))
            actualizar_lista(); actualizar_saldos()

    frame_btn = tk.Frame(ventana)
    frame_btn.pack(pady=10)
    tk.Button(frame_btn, text="Editar", command=editar_producto, bg="#2196F3",
fg="white", font=("Arial", 10, "bold"), width=12).pack(side=tk.LEFT, padx=5)
    tk.Button(frame_btn, text="Eliminar", command=eliminar_producto, bg="#ff4d4d",
fg="white", font=("Arial", 10, "bold"), width=12).pack(side=tk.LEFT, padx=5)
```

```
    tk.Button(frame_btn, text="Cerrar", command=ventana.destroy, font=("Arial", 10,
"bold"), width=12).pack(side=tk.LEFT, padx=5)

    actualizar_lista()

def agregar_usuario_gui():
    if rol_usuario != "admin":
        messagebox.showerror("Acceso denegado", "Solo el administrador puede
agregar nuevos usuarios.")
        return

    ventana_usuario = tk.Toplevel(root)
    ventana_usuario.title("Agregar Nuevo Usuario")
    ventana_usuario.geometry("400x320")
    ventana_usuario.resizable(False, False)
    if icono_path:
        ventana_usuario.iconbitmap(icono_path)
    ventana_usuario.configure(bg="#f0f0f0")
    centrar_ventana(ventana_usuario)

    frame_form = tk.Frame(ventana_usuario, bg="#f0f0f0")
    frame_form.pack(padx=20, pady=20)

    tk.Label(frame_form, text="Nuevo Usuario (Username):", bg="#f0f0f0",
font=("Arial", 10)).grid(row=0, column=0, sticky="w", pady=5)
    entry_user = tk.Entry(frame_form, width=30, font=("Arial", 10))
    entry_user.grid(row=0, column=1, pady=5)

    tk.Label(frame_form, text="Contraseña:", bg="#f0f0f0", font=("Arial",
10)).grid(row=1, column=0, sticky="w", pady=5)
    entry_pass = tk.Entry(frame_form, width=30, show="*", font=("Arial", 10))
    entry_pass.grid(row=1, column=1, pady=5)

    tk.Label(frame_form, text="Rol del Usuario:", bg="#f0f0f0", font=("Arial",
10)).grid(row=2, column=0, sticky="w", pady=5)
    rol_var = tk.StringVar(value="usuario")
    frame_radio = tk.Frame(frame_form, bg="#f0f0f0")
    frame_radio.grid(row=2, column=1, pady=5, sticky="w")
    tk.Radiobutton(frame_radio, text="Usuario estándar", variable=rol_var,
value="usuario", bg="#f0f0f0", font=("Arial", 10)).pack(anchor="w")
    tk.Radiobutton(frame_radio, text="Administrador", variable=rol_var,
value="admin", bg="#f0f0f0", font=("Arial", 10)).pack(anchor="w")

    def guardar_usuario():
        nuevo_usuario = entry_user.get().strip()
        nueva_contraseña = entry_pass.get().strip()
        rol_asignado = rol_var.get()

        if not nuevo_usuario or not nueva_contraseña:
            messagebox.showerror("Error", "El usuario y la contraseña son
obligatorios.", parent=ventana_usuario)
            return

        try:
            with sqlite3.connect(DB_FILE) as conn:
                cursor = conn.cursor()
                cursor.execute("SELECT username FROM usuarios WHERE username = ?",
(nuevo_usuario,))
                if cursor.fetchone():
                    messagebox.showerror("Error de Usuario", f"El usuario
'{nuevo_usuario}' ya existe.", parent=ventana_usuario)
                    return
```

```
                cursor.execute("INSERT INTO usuarios (username, password, role)
VALUES (?, ?, ?)",
                               (nuevo_usuario, hash_password(nueva_contraseña),
rol_asignado))

            messagebox.showinfo("Éxito", f"Usuario '{nuevo_usuario}' agregado
correctamente con rol de '{rol_asignado}'.", parent=ventana_usuario)
            notificar("Nuevo Usuario", f"Se ha creado el usuario
'{nuevo_usuario}'.")
            ventana_usuario.destroy()
        except Exception as e:
            messagebox.showerror("Error de Base de Datos", f"No se pudo agregar el
usuario.\nError: {e}", parent=ventana_usuario)

    frame_botones = tk.Frame(ventana_usuario, bg="#f0f0f0")
    frame_botones.pack(pady=10)
    tk.Button(frame_botones, text="Guardar Usuario", command=guardar_usuario,
bg="#e49f54", fg="white", font=("Arial", 10, "bold")).pack(side=tk.LEFT, padx=10)
    tk.Button(frame_botones, text="Cancelar", command=ventana_usuario.destroy,
font=("Arial", 10)).pack(side=tk.LEFT, padx=10)

def registrar_ingreso_gui():
    ventana_ingreso = tk.Toplevel(root)
    ventana_ingreso.title("Registrar Ingreso de Producto")
    ventana_ingreso.geometry("400x300")
    ventana_ingreso.resizable(False, False)
    if icono_path: ventana_ingreso.iconbitmap(icono_path)
    ventana_ingreso.configure(bg="#f0f0f0")
    centrar_ventana(ventana_ingreso)

    frame_form = tk.Frame(ventana_ingreso, bg="#f0f0f0")
    frame_form.pack(padx=20, pady=20)

    tk.Label(frame_form, text="Código del Producto:", bg="#f0f0f0").grid(row=0,
column=0, sticky="w", pady=5)
    entry_codigo = tk.Entry(frame_form, width=30)
    entry_codigo.grid(row=0, column=1, pady=5)

    tk.Label(frame_form, text="Fecha (DD-MM-YYYY):", bg="#f0f0f0").grid(row=1,
column=0, sticky="w", pady=5)
    entry_fecha = tk.Entry(frame_form, width=30)
    entry_fecha.grid(row=1, column=1, pady=5)
    entry_fecha.insert(0, datetime.now().strftime('%d-%m-%Y'))

    tk.Label(frame_form, text="Cantidad:", bg="#f0f0f0").grid(row=2, column=0,
sticky="w", pady=5)
    entry_cantidad = tk.Entry(frame_form, width=30)
    entry_cantidad.grid(row=2, column=1, pady=5)

    tk.Label(frame_form, text="Precio Unitario Proveedor:",
bg="#f0f0f0").grid(row=3, column=0, sticky="w", pady=5)
    entry_precio = tk.Entry(frame_form, width=30)
    entry_precio.grid(row=3, column=1, pady=5)

    tk.Label(frame_form, text="Nro. Comprobante:", bg="#f0f0f0").grid(row=4,
column=0, sticky="w", pady=5)
    entry_comprobante = tk.Entry(frame_form, width=30)
    entry_comprobante.grid(row=4, column=1, pady=5)

    def guardar_ingreso():
        codigo = entry_codigo.get().strip()
        fecha_ui = entry_fecha.get().strip()
        cantidad_str = entry_cantidad.get().strip()
```

```
        precio_str = entry_precio.get().strip()
        comprobante = entry_comprobante.get().strip()

        if not all([codigo, fecha_ui, cantidad_str]):
            messagebox.showerror("Error", "Código, Fecha y Cantidad son
obligatorios.", parent=ventana_ingreso)
            return

        try:
            cantidad = int(cantidad_str)
            precio = float(precio_str) if precio_str else 0.00
        except ValueError:
            messagebox.showerror("Error", "La cantidad debe ser entera y el precio
un número (ej. 15.50).", parent=ventana_ingreso)
            return

        if not validar_fecha(fecha_ui):
            messagebox.showerror("Error", "Formato de fecha inválido. Use DD-MM-
YYYY.", parent=ventana_ingreso)
            return

        fecha_db = formatear_fecha_almacenar(fecha_ui)

        with sqlite3.connect(DB_FILE) as conn:
            conn.row_factory = sqlite3.Row
            cursor = conn.cursor()
            cursor.execute("SELECT nombre FROM productos WHERE codigo = ?",
(codigo,))
            producto = cursor.fetchone()
            if not producto:
                messagebox.showerror("Error", "Producto no encontrado.",
parent=ventana_ingreso)
                return

            cursor.execute('''INSERT INTO movimientos_kardex (producto_codigo,
fecha, tipo, cantidad, precio_unitario, comprobante, responsable) VALUES (?, ?,
'ingreso', ?, ?, ?, ?)''',
                           (codigo, fecha_db, cantidad, precio, comprobante,
nombre_usuario))

        messagebox.showinfo("Éxito", f"Ingreso de {cantidad} unidades de
'{producto['nombre']}' registrado.", parent=ventana_ingreso)
        notificar("Ingreso registrado", f"Ingreso de {cantidad} unidades de
{producto['nombre']}.")
        actualizar_saldos()
        verificar_stock_bajo()
        ventana_ingreso.destroy()

    frame_botones = tk.Frame(ventana_ingreso, bg="#f0f0f0")
    frame_botones.pack(pady=10)
    tk.Button(frame_botones, text="Guardar Ingreso", command=guardar_ingreso,
bg="#e49f54", fg="white", font=("Arial", 10, "bold")).pack(side=tk.LEFT, padx=10)
    tk.Button(frame_botones, text="Cancelar", command=ventana_ingreso.destroy,
font=("Arial", 10)).pack(side=tk.LEFT, padx=10)

def registrar_egreso_gui():
    ventana_egreso = tk.Toplevel(root)
    ventana_egreso.title("Registrar Egreso de Producto")
    ventana_egreso.geometry("420x380")
    ventana_egreso.resizable(False, False)
    if icono_path: ventana_egreso.iconbitmap(icono_path)
    ventana_egreso.configure(bg="#f0f0f0")
    centrar_ventana(ventana_egreso)
```

```
    frame_form = tk.Frame(ventana_egreso, bg="#f0f0f0")
    frame_form.pack(padx=20, pady=20)

    tk.Label(frame_form, text="Código del Producto:", bg="#f0f0f0").grid(row=0,
column=0, sticky="w", pady=5)
    entry_codigo = tk.Entry(frame_form, width=30)
    entry_codigo.grid(row=0, column=1, pady=5)

    tk.Label(frame_form, text="Fecha (DD-MM-YYYY):", bg="#f0f0f0").grid(row=1,
column=0, sticky="w", pady=5)
    entry_fecha = tk.Entry(frame_form, width=30)
    entry_fecha.grid(row=1, column=1, pady=5)
    entry_fecha.insert(0, datetime.now().strftime('%d-%m-%Y'))

    tk.Label(frame_form, text="Cantidad:", bg="#f0f0f0").grid(row=2, column=0,
sticky="w", pady=5)
    entry_cantidad = tk.Entry(frame_form, width=30)
    entry_cantidad.grid(row=2, column=1, pady=5)

    tk.Label(frame_form, text="Precio Unitario:", bg="#f0f0f0").grid(row=3,
column=0, sticky="w", pady=5)
    entry_precio = tk.Entry(frame_form, width=30)
    entry_precio.grid(row=3, column=1, pady=5)

    tk.Label(frame_form, text="Nro. Comprobante:", bg="#f0f0f0").grid(row=4,
column=0, sticky="w", pady=5)
    entry_comprobante = tk.Entry(frame_form, width=30)
    entry_comprobante.grid(row=4, column=1, pady=5)

    tk.Label(frame_form, text="Área de Destino:", bg="#f0f0f0").grid(row=5,
column=0, sticky="w", pady=5)
    frame_area = tk.Frame(frame_form, bg="#f0f0f0")
    frame_area.grid(row=5, column=1, pady=5, sticky="w")
    combo_area = ttk.Combobox(frame_area, values=obtener_areas_bd(), width=24,
state="readonly")
    combo_area.pack(side=tk.LEFT)
    if len(obtener_areas_bd()) > 0: combo_area.current(0)
    tk.Button(frame_area, text="+", width=3, font=("Arial", 10, "bold"),
bg="#4CAF50", fg="white",
              command=lambda: agregar_area_rapida(combo_area,
ventana_egreso)).pack(side=tk.LEFT, padx=5)

    def guardar_egreso():
        codigo = entry_codigo.get().strip()
        fecha_ui = entry_fecha.get().strip()
        cantidad_str = entry_cantidad.get().strip()
        precio_str = entry_precio.get().strip()
        comprobante = entry_comprobante.get().strip()
        area = combo_area.get().strip()

        if not all([codigo, fecha_ui, cantidad_str, precio_str]):
            messagebox.showerror("Error", "Los campos Código, Fecha, Cantidad y
Precio son obligatorios.", parent=ventana_egreso)
            return

        try:
            cantidad = int(cantidad_str)
            precio = float(precio_str)
        except ValueError:
            messagebox.showerror("Error", "La cantidad debe ser un entero y el
precio un número decimal.", parent=ventana_egreso)
            return
```

```
        if not validar_fecha(fecha_ui):
            messagebox.showerror("Error", "Formato de fecha inválido. Use DD-MM-
YYYY.", parent=ventana_egreso)
            return

        fecha_db = formatear_fecha_almacenar(fecha_ui)

        with sqlite3.connect(DB_FILE) as conn:
            conn.row_factory = sqlite3.Row
            cursor = conn.cursor()
            cursor.execute("SELECT nombre FROM productos WHERE codigo = ?",
(codigo,))
            producto = cursor.fetchone()
            if not producto:
                messagebox.showerror("Error", "Producto no encontrado.",
parent=ventana_egreso)
                return

            stock_actual = calcular_saldo(codigo)
            if cantidad > stock_actual:
                messagebox.showerror("Error", f"Stock insuficiente. Stock actual:
{stock_actual}.", parent=ventana_egreso)
                return

            cursor.execute('''INSERT INTO movimientos_kardex (producto_codigo,
fecha, tipo, cantidad, precio_unitario, comprobante, responsable, area) VALUES (?,
?, 'egreso', ?, ?, ?, ?, ?)''',
                           (codigo, fecha_db, cantidad, precio, comprobante,
nombre_usuario, area))

        messagebox.showinfo("Éxito", f"Egreso de {cantidad} unidades de
'{producto['nombre']}' registrado.", parent=ventana_egreso)
        notificar("Egreso registrado", f"Egreso de {cantidad} unidades de
{producto['nombre']}.")
        actualizar_saldos()
        verificar_stock_bajo()
        ventana_egreso.destroy()

    frame_botones = tk.Frame(ventana_egreso, bg="#f0f0f0")
    frame_botones.pack(pady=10)
    tk.Button(frame_botones, text="Guardar Egreso", command=guardar_egreso,
bg="#e49f54", fg="white", font=("Arial", 10, "bold")).pack(side=tk.LEFT, padx=10)
    tk.Button(frame_botones, text="Cancelar", command=ventana_egreso.destroy,
font=("Arial", 10)).pack(side=tk.LEFT, padx=10)

def calcular_saldo(codigo):
    try:
        with sqlite3.connect(DB_FILE) as conn:
            conn.row_factory = sqlite3.Row
            cursor = conn.cursor()
            cursor.execute('''SELECT SUM(CASE WHEN tipo = 'ingreso' THEN cantidad
ELSE 0 END) - SUM(CASE WHEN tipo = 'egreso' THEN cantidad ELSE 0 END) AS stock FROM
movimientos_kardex WHERE producto_codigo = ?''', (codigo,))
            resultado = cursor.fetchone()
            return resultado['stock'] if resultado and resultado['stock'] is not
None else 0
    except Exception:
        return 0

def imprimir_desde_texto(text_widget):
    contenido = text_widget.get(1.0, tk.END).strip()
    if not contenido:
```

```
            messagebox.showwarning("Impresión", "No hay contenido para imprimir.")
            return
        pdf_file_path = tempfile.mktemp(".pdf")
        try:
            c = canvas.Canvas(pdf_file_path, pagesize=letter)
            width, height = letter
            c.setFont("Courier", 10)
            margin = inch
            y_position = height - margin
            lines = contenido.split('\n')
            for line in lines:
                while c.stringWidth(line, "Courier", 10) > (width - 2 * margin):
                    max_index = 0
                    for i, char in enumerate(line):
                        if c.stringWidth(line[:i+1], "Courier", 10) > (width - 2 *
margin):
                            break
                        max_index = i
                    c.drawString(margin, y_position, line[:max_index])
                    line = line[max_index:].lstrip()
                    y_position -= 12
                c.drawString(margin, y_position, line)
                y_position -= 12
                if y_position < margin:
                    c.showPage()
                    y_position = height - margin
                    c.setFont("Courier", 10)
            c.save()
            if os.path.exists(pdf_file_path):
                os.startfile(pdf_file_path)
                messagebox.showinfo("Reporte Generado", "El reporte se ha generado como
PDF y se abrirá en su visor.\n\nPuede imprimirlo desde allí.")
        except Exception as e:
            messagebox.showerror("Error de Impresión", f"No se pudo generar el reporte
PDF.\nError: {e}")

def mostrar_kardex_gui():
    ventana = tk.Toplevel(root)
    ventana.title("KARDEX POR PRODUCTO")
    ventana.attributes('-alpha', 0.9)
    if icono_path: ventana.iconbitmap(icono_path)
    centrar_ventana(ventana)

    frame_texto = tk.Frame(ventana)
    frame_texto.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)
    texto = tk.Text(frame_texto, width=90, height=25)
    texto.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)
    scrollbar = tk.Scrollbar(frame_texto, command=texto.yview)
    scrollbar.pack(side=tk.RIGHT, fill=tk.Y)
    texto.config(yscrollcommand=scrollbar.set)

    contenido = ""
    with sqlite3.connect(DB_FILE) as conn:
        conn.row_factory = sqlite3.Row
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM productos ORDER BY nombre")
        productos_db = cursor.fetchall()

        for producto in productos_db:
            codigo = producto['codigo']
            nombre = producto['nombre']
            contenido += f"\nProducto: {nombre} (Código: {codigo})\n"
```

```
            contenido += "Fecha       | Tipo     | Cantidad | P. Unitario |
Comprobante | Responsable | Área\n"
            contenido += "------------|---------|----------|-------------|---------
----|-------------|------\n"

            cursor.execute('''SELECT fecha, tipo, cantidad, precio_unitario,
comprobante, responsable, area FROM movimientos_kardex WHERE producto_codigo = ?
ORDER BY fecha''', (codigo,))
            movimientos = cursor.fetchall()

            for mov in movimientos:
                fecha_ui = formatear_fecha_mostrar(mov['fecha'])
                contenido += f"{fecha_ui} | {mov['tipo']:7} | {mov['cantidad']:8} |
{(mov['precio_unitario'] or 0.00):12.2f} | {(mov['comprobante'] or 'N/A'):11} |
{(mov['responsable'] or 'N/A'):11} | {(mov['area'] or 'N/A')}\n"

            stock_actual = calcular_saldo(codigo)
            contenido += f"\nStock Actual: {stock_actual}\n"
            contenido += "="*80 + "\n"

    texto.insert(tk.END, contenido)
    texto.config(state=tk.DISABLED)

    frame_botones = tk.Frame(ventana)
    frame_botones.pack(pady=5, fill=tk.X)
    tk.Button(frame_botones, text="Imprimir Reporte", command=lambda:
imprimir_desde_texto(texto), bg="#e49f54", fg="white", font=("Arial", 10,
"bold")).pack()

def exportar_excel():
    with sqlite3.connect(DB_FILE) as conn:
        conn.row_factory = sqlite3.Row
        cursor = conn.cursor()
        cursor.execute('''SELECT p.codigo, p.nombre, m.fecha, m.tipo, m.cantidad,
m.precio_unitario, m.comprobante, m.responsable, m.area FROM movimientos_kardex m
JOIN productos p ON m.producto_codigo = p.codigo ORDER BY p.nombre, m.fecha''')
        datos = cursor.fetchall()

    if not datos:
        messagebox.showwarning("Sin Datos", "No hay movimientos en el kardex para
exportar.")
        return
    ruta = filedialog.asksaveasfilename(title="Guardar archivo de Excel",
defaultextension=".xlsx", filetypes=[("Archivos de Excel", "*.xlsx"), ("Todos los
archivos", "*.*")])
    if not ruta: return
    try:
        wb = openpyxl.Workbook()
        ws = wb.active
        ws.title = "Reporte de Kardex"
        headers = ["Código", "Nombre", "Fecha", "Tipo", "Cantidad", "Precio
Unitario", "Comprobante", "Responsable", "Área"]
        ws.append(headers)
        for fila in datos:
            fila_dict = dict(fila)
            fila_dict['fecha'] = formatear_fecha_mostrar(fila_dict['fecha'])
            ws.append(list(fila_dict.values()))
        wb.save(ruta)
        messagebox.showinfo("Exportación Exitosa", f"Los datos han sido exportados
correctamente a:\n{ruta}")
    except PermissionError:
        messagebox.showerror("Error de Permiso", f"No se pudo guardar el archivo.
Asegúrate de que el archivo '{os.path.basename(ruta)}' no esté abierto.")
```

```
    except Exception as e:
        messagebox.showerror("Error al Exportar", f"Ocurrió un error inesperado al
exportar a Excel:\n{e}")

def buscar_producto():
    ventana_buscar = tk.Toplevel(root)
    ventana_buscar.title("Buscar Producto")
    ventana_buscar.geometry("400x150")
    ventana_buscar.resizable(False, False)
    if icono_path: ventana_buscar.iconbitmap(icono_path)
    ventana_buscar.configure(bg="#f0f0f0")
    centrar_ventana(ventana_buscar)

    tk.Label(ventana_buscar, text="Ingrese nombre o código:", bg="#f0f0f0",
font=("Arial", 11)).pack(pady=(20, 5))
    entry_buscar = tk.Entry(ventana_buscar, font=("Arial", 12), width=35)
    entry_buscar.pack(pady=5)
    entry_buscar.focus_set()

    def ejecutar_busqueda():
        criterio = entry_buscar.get().strip()
        if not criterio:
            messagebox.showwarning("Vacío", "Ingrese un criterio de búsqueda.",
parent=ventana_buscar)
            return

        with sqlite3.connect(DB_FILE) as conn:
            conn.row_factory = sqlite3.Row
            cursor = conn.cursor()
            cursor.execute('''SELECT p.codigo, p.nombre, m.fecha, m.tipo,
m.cantidad, m.precio_unitario, m.comprobante FROM movimientos_kardex m JOIN
productos p ON m.producto_codigo = p.codigo WHERE p.codigo LIKE ? OR p.nombre LIKE
? ORDER BY p.nombre, m.fecha''', (f'%{criterio}%', f'%{criterio}%'))
            resultados = cursor.fetchall()

        ventana_buscar.destroy()

        ventana = tk.Toplevel(root)
        ventana.title(f"Resultado de búsqueda: {criterio}")
        if icono_path: ventana.iconbitmap(icono_path)
        centrar_ventana(ventana)

        frame_texto = tk.Frame(ventana)
        frame_texto.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)
        texto = tk.Text(frame_texto, width=80, height=20)
        texto.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)
        scrollbar = tk.Scrollbar(frame_texto, command=texto.yview)
        scrollbar.pack(side=tk.RIGHT, fill=tk.Y)
        texto.config(yscrollcommand=scrollbar.set)

        contenido = ""
        if not resultados:
            contenido = "No se encontraron coincidencias."
        else:
            for fila in resultados:
                fecha_ui = formatear_fecha_mostrar(fila['fecha'])
                contenido += f"\nProducto: {fila['nombre']} (Código:
{fila['codigo']})\n"
                contenido += f"Fecha: {fecha_ui}, Tipo: {fila['tipo']}, Cantidad:
{fila['cantidad']}, Precio: {fila['precio_unitario']}, Comp:
{fila['comprobante']}\n"
                contenido += "-"*40 + "\n"
        texto.insert(tk.END, contenido)
```

```
        texto.config(state=tk.DISABLED)

        frame_botones = tk.Frame(ventana)
        frame_botones.pack(pady=5, fill=tk.X)
        tk.Button(frame_botones, text="Imprimir Búsqueda", command=lambda:
imprimir_desde_texto(texto), bg="#e49f54", fg="white", font=("Arial", 10,
"bold")).pack()

    entry_buscar.bind('<Return>', lambda event: ejecutar_busqueda())

    frame_btn = tk.Frame(ventana_buscar, bg="#f0f0f0")
    frame_btn.pack(pady=10)
    tk.Button(frame_btn, text="Buscar", command=ejecutar_busqueda, bg="#01525f",
fg="white", font=("Arial", 10, "bold"), width=15).pack(side=tk.LEFT, padx=5)
    tk.Button(frame_btn, text="Cancelar", command=ventana_buscar.destroy,
font=("Arial", 10), width=15).pack(side=tk.LEFT, padx=5)

def mostrar_resumen_caja_gui():
    with sqlite3.connect(DB_FILE) as conn:
        cursor = conn.cursor()
        cursor.execute("SELECT SUM(cantidad * precio_unitario) FROM
movimientos_kardex WHERE tipo = 'egreso'")
        resultado = cursor.fetchone()
    total = resultado[0] if resultado and resultado[0] is not None else 0.0

    ventana = tk.Toplevel(root)
    ventana.title("Resumen de Caja")
    if icono_path: ventana.iconbitmap(icono_path)
    ventana.attributes('-alpha', 0.9)
    centrar_ventana(ventana)
    tk.Label(ventana, text=f"Total costo de egresos: S/. {total:.2f}",
font=("Arial", 14)).pack(pady=20)

def obtener_productos_bajo_stock_optimizado():
    try:
        with sqlite3.connect(DB_FILE) as conn:
            conn.row_factory = sqlite3.Row
            cursor = conn.cursor()
            query = '''
                SELECT p.codigo, p.nombre, p.stock_minimo,
                    COALESCE(SUM(CASE WHEN m.tipo = 'ingreso' THEN m.cantidad ELSE
0 END), 0) -
                    COALESCE(SUM(CASE WHEN m.tipo = 'egreso' THEN m.cantidad ELSE 0
END), 0) AS stock_actual
                FROM productos p
                LEFT JOIN movimientos_kardex m ON p.codigo = m.producto_codigo
                GROUP BY p.codigo, p.nombre, p.stock_minimo
                HAVING stock_actual < p.stock_minimo
                ORDER BY p.nombre
            '''
            cursor.execute(query)
            return cursor.fetchall()
    except Exception:
        return []

def verificar_stock_bajo():
    productos_bajo_stock = obtener_productos_bajo_stock_optimizado()
    for producto in productos_bajo_stock:
        nombre = producto['nombre']
        codigo = producto['codigo']
        stock_actual = producto['stock_actual']
        mensaje = f"¡Alerta! El stock del producto '{nombre}' (Código: {codigo}) es
bajo: {stock_actual} unidades."
```

```
            notificar("Stock Bajo", mensaje)

def mostrar_reporte_stock_bajo_gui():
    productos_bajo_stock = obtener_productos_bajo_stock_optimizado()
    ventana_reporte = tk.Toplevel(root)
    ventana_reporte.title("Reporte de Stock Bajo")
    ventana_reporte.geometry("700x400")
    if icono_path: ventana_reporte.iconbitmap(icono_path)
    centrar_ventana(ventana_reporte)

    frame = tk.Frame(ventana_reporte)
    frame.pack(fill=tk.BOTH, expand=True, padx=10, pady=10)
    tree = ttk.Treeview(frame, columns=('codigo', 'nombre', 'minimo', 'actual'),
show='headings')
    tree.heading('codigo', text='Código'); tree.heading('nombre', text='Nombre del
Producto')
    tree.heading('minimo', text='Stock Mínimo'); tree.heading('actual', text='Stock
Actual')
    tree.column('codigo', width=100, anchor=tk.CENTER); tree.column('nombre',
width=300, anchor=tk.W)
    tree.column('minimo', width=100, anchor=tk.CENTER); tree.column('actual',
width=100, anchor=tk.CENTER)
    tree.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)
    scrollbar = ttk.Scrollbar(frame, orient=tk.VERTICAL, command=tree.yview)
    tree.configure(yscroll=scrollbar.set)
    scrollbar.pack(side=tk.RIGHT, fill=tk.Y)

    if not productos_bajo_stock:
        tree.insert('', tk.END, values=('N/A', 'No hay productos con stock bajo',
'N/A', 'N/A'))
    else:
        for item in productos_bajo_stock:
            tree.insert('', tk.END, values=(item['codigo'], item['nombre'],
item['stock_minimo'], item['stock_actual']))
    tk.Button(ventana_reporte, text="Cerrar", command=ventana_reporte.destroy,
font=("Arial", 10)).pack(pady=10)

def obtener_gastos_por_area(fecha_inicio=None, fecha_fin=None):
    try:
        with sqlite3.connect(DB_FILE) as conn:
            conn.row_factory = sqlite3.Row
            cursor = conn.cursor()
            query = '''
                SELECT COALESCE(area, 'Sin Área Asignada') AS area_nombre,
                    SUM(cantidad * precio_unitario) AS monto_total
                FROM movimientos_kardex
                WHERE tipo = 'egreso'
            '''
            params = []
            if fecha_inicio and fecha_fin:
                query += " AND fecha BETWEEN ? AND ?"
                params.extend([fecha_inicio, fecha_fin])
            query += " GROUP BY UPPER(TRIM(COALESCE(area, 'Sin Área Asignada')))
ORDER BY monto_total DESC"
            cursor.execute(query, params)
            return cursor.fetchall()
    except Exception as e:
        print(f"Error gastos por area: {e}")
        return []

def mostrar_reporte_gastos_area_gui():
    ventana_reporte = tk.Toplevel(root)
    ventana_reporte.title("Reporte de Costo de Insumos por Área")
```

```
    ventana_reporte.geometry("1100x850")
    if icono_path: ventana_reporte.iconbitmap(icono_path)
    centrar_ventana(ventana_reporte)

    widget_texto_resumen = None

    frame_botones = tk.Frame(ventana_reporte, bg="#f0f0f0")
    frame_botones.pack(side=tk.BOTTOM, pady=10)

    tk.Button(frame_botones, text="Filtrar Fechas", command=lambda:
generar_reporte(), bg="#01525f", fg="white", font=("Arial", 11, "bold"),
width=20).pack(side=tk.LEFT, padx=10)

    def imprimir_resumen():
        if widget_texto_resumen is not None:
            imprimir_desde_texto(widget_texto_resumen)
        else:
            messagebox.showwarning("Sin Datos", "No hay un reporte generado para
imprimir.", parent=ventana_reporte)

    tk.Button(frame_botones, text="Imprimir Resumen", command=imprimir_resumen,
bg="#e49f54", fg="white", font=("Arial", 11, "bold"), width=18).pack(side=tk.LEFT,
padx=10)
    tk.Button(frame_botones, text="Cerrar", command=ventana_reporte.destroy,
font=("Arial", 11, "bold"), width=12).pack(side=tk.LEFT, padx=10)

    frame_fechas = tk.Frame(ventana_reporte, bg="#f0f0f0")
    frame_fechas.pack(fill=tk.X, padx=10, pady=10)

    tk.Label(frame_fechas, text="Desde (DD-MM-YYYY):", bg="#f0f0f0", font=("Arial",
10, "bold")).pack(side=tk.LEFT, padx=5)
    entry_desde = tk.Entry(frame_fechas, width=12)
    entry_desde.pack(side=tk.LEFT, padx=5)

    tk.Label(frame_fechas, text="Hasta (DD-MM-YYYY):", bg="#f0f0f0", font=("Arial",
10, "bold")).pack(side=tk.LEFT, padx=5)
    entry_hasta = tk.Entry(frame_fechas, width=12)
    entry_hasta.pack(side=tk.LEFT, padx=5)

    desde_defecto = (datetime.now() - timedelta(days=30)).strftime('%d-%m-%Y')
    hasta_defecto = datetime.now().strftime('%d-%m-%Y')
    entry_desde.insert(0, desde_defecto)
    entry_hasta.insert(0, hasta_defecto)

    frame_principal = tk.Frame(ventana_reporte, bg="#f0f0f0")
    frame_principal.pack(fill=tk.BOTH, expand=True, padx=10, pady=(0, 10))

    def generar_reporte():
        nonlocal widget_texto_resumen
        for widget in frame_principal.winfo_children():
            widget.destroy()

        fecha_desde_ui = entry_desde.get().strip()
        fecha_hasta_ui = entry_hasta.get().strip()

        if not validar_fecha(fecha_desde_ui) or not validar_fecha(fecha_hasta_ui):
            messagebox.showerror("Error", "Formato de fecha inválido.",
parent=ventana_reporte)
            return

        fecha_desde_db = formatear_fecha_almacenar(fecha_desde_ui)
        fecha_hasta_db = formatear_fecha_almacenar(fecha_hasta_ui)
```

```
        datos = obtener_gastos_por_area(fecha_desde_db, fecha_hasta_db)

        if not datos or (len(datos) == 1 and datos[0]['area_nombre'] == 'Sin Área
Asignada'):
            tk.Label(frame_principal, text="No hay egresos (insumos retirados) en
ese rango de fechas.", bg="#f0f0f0", fg="red", font=("Arial",
12)).pack(expand=True)
            widget_texto_resumen = None
            return

        frame_graficos = tk.Frame(frame_principal, bg="#f0f0f0")
        frame_graficos.pack(fill=tk.X, pady=(0, 10))

        colores_base = ['#e49f54', '#01525f', '#3498db', '#2ecc71', '#f1c40f',
'#e67e22', '#9b59b6', '#1abc9c']
        labels = [item['area_nombre'] for item in datos]
        sizes = [item['monto_total'] for item in datos]

        plt.close('all')

        fig_pie = Figure(figsize=(4.5, 3), dpi=90, facecolor='#f0f0f0')
        ax_pie = fig_pie.add_subplot(111)
        ax_pie.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90,
colors=colores_base[:len(labels)], textprops={'fontsize': 8})
        ax_pie.set_title('Distribución de Insumos (%)', fontsize=10,
color='#01525f', weight='bold')
        fig_pie.tight_layout()
        canvas_pie = FigureCanvasTkAgg(fig_pie, master=frame_graficos)
        canvas_pie.draw()
        canvas_pie.get_tk_widget().pack(side=tk.LEFT, fill=tk.BOTH, expand=True,
padx=5)

        fig_bar = Figure(figsize=(4.5, 3), dpi=90, facecolor='#f0f0f0')
        ax_bar = fig_bar.add_subplot(111)
        ax_bar.bar(labels, sizes, color=colores_base[:len(labels)])
        ax_bar.set_title('Costo de Insumos por Área (S/.)', fontsize=10,
color='#01525f', weight='bold')
        ax_bar.set_ylabel('Soles (S/.)', fontsize=8)
        ax_bar.set_xticks(range(len(labels)))
        ax_bar.set_xticklabels(labels, rotation=45, ha="right", fontsize=8)
        fig_bar.tight_layout()
        canvas_bar = FigureCanvasTkAgg(fig_bar, master=frame_graficos)
        canvas_bar.draw()
        canvas_bar.get_tk_widget().pack(side=tk.RIGHT, fill=tk.BOTH, expand=True,
padx=5)

        frame_resumen = tk.Frame(frame_principal, bg="#f0f0f0", relief=tk.SUNKEN,
bd=2)
        frame_resumen.pack(fill=tk.BOTH, expand=True)

        tk.Label(frame_resumen, text="Resumen Detallado de Costos por Insumos
Retirados", font=("Arial", 14, "bold"), bg="#f0f0f0", fg="#01525f").pack(pady=5)

        frame_texto_resumen = tk.Frame(frame_resumen, bg="#f0f0f0")
        frame_texto_resumen.pack(fill=tk.BOTH, padx=10, pady=5)

        texto_resumen = tk.Text(frame_texto_resumen, wrap=tk.WORD,
font=("Consolas", 12), bg="white", relief=tk.FLAT, height=15)
        scrollbar_resumen = tk.Scrollbar(frame_texto_resumen,
command=texto_resumen.yview)
        texto_resumen.config(yscrollcommand=scrollbar_resumen.set)
        scrollbar_resumen.pack(side=tk.RIGHT, fill=tk.Y)
        texto_resumen.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)
```

```
        widget_texto_resumen = texto_resumen

        total_general = sum(item['monto_total'] for item in datos)

        contenido_resumen = f"{'='*70}\n"
        contenido_resumen += f"  PERIODO EVALUADO: {fecha_desde_ui} AL
{fecha_hasta_ui}\n"
        contenido_resumen += f"  COSTO TOTAL DE INSUMOS RETIRADOS: S/.
{total_general:,.2f}\n"
        contenido_resumen += f"{'-'*70}\n\n"

        for item in datos:
            area_nombre = item['area_nombre']
            monto = item['monto_total']
            porcentaje = (monto / total_general) * 100 if total_general > 0 else 0

            contenido_resumen += f"  >> ÁREA: {area_nombre}\n"
            contenido_resumen += f"     Costo de insumos consumidos: S/.
{monto:,.2f}\n"
            contenido_resumen += f"     Representa el: {porcentaje:.1f}% del costo
total de insumos\n\n"

        texto_resumen.insert(tk.END, contenido_resumen)
        texto_resumen.config(state=tk.DISABLED)

    generar_reporte()

def actualizar_saldos():
    for w in frame_saldos.winfo_children():
        w.destroy()
    with sqlite3.connect(DB_FILE) as conn:
        conn.row_factory = sqlite3.Row
        cursor = conn.cursor()
        cursor.execute('''SELECT p.codigo, p.nombre, COALESCE(SUM(CASE WHEN m.tipo
= 'ingreso' THEN m.cantidad ELSE 0 END), 0) - COALESCE(SUM(CASE WHEN m.tipo =
'egreso' THEN m.cantidad ELSE 0 END), 0) AS stock_actual FROM productos p LEFT JOIN
movimientos_kardex m ON p.codigo = m.producto_codigo GROUP BY p.codigo, p.nombre
ORDER BY p.nombre''')
        saldos = cursor.fetchall()

    if not saldos:
        tk.Label(frame_saldos, text="No hay productos registrados.",
bg="lightyellow", anchor="w").pack(fill=tk.X, padx=5)
        return
    for item in saldos:
        stock = item['stock_actual']
        nombre = item['nombre']
        codigo = item['codigo']
        tk.Label(frame_saldos, text=f"{nombre} ({codigo}): {stock} unidades",
bg="lightyellow", anchor="w").pack(fill=tk.X, padx=5, pady=2)

def corregir_movimientos_gui():
    ventana_correccion = tk.Toplevel(root)
    if rol_usuario == "admin":
        ventana_correccion.title("Corregir Todos los Movimientos")
    else:
        ventana_correccion.title(f"Corregir Mis Movimientos ({nombre_usuario})")
    ventana_correccion.geometry("1100x500")
    if icono_path: ventana_correccion.iconbitmap(icono_path)
    centrar_ventana(ventana_correccion)

    frame = tk.Frame(ventana_correccion)
```

```
    frame.pack(fill=tk.BOTH, expand=True, padx=10, pady=10)

    columnas = ('id', 'producto', 'fecha', 'tipo', 'cantidad', 'precio',
'comprobante', 'responsable', 'area')
    tree = ttk.Treeview(frame, columns=columnas, show='headings')
    tree.heading('id', text='ID'); tree.heading('producto', text='Producto');
tree.heading('fecha', text='Fecha'); tree.heading('tipo', text='Tipo')
    tree.heading('cantidad', text='Cantidad'); tree.heading('precio', text='P.
Unitario'); tree.heading('comprobante', text='Comprobante');
tree.heading('responsable', text='Responsable')
    tree.heading('area', text='Área')
    tree.column('id', width=40, anchor=tk.CENTER); tree.column('producto',
width=150); tree.column('fecha', width=100, anchor=tk.CENTER)
    tree.column('tipo', width=80, anchor=tk.CENTER); tree.column('cantidad',
width=80, anchor=tk.CENTER); tree.column('precio', width=80, anchor=tk.E)
    tree.column('comprobante', width=100); tree.column('responsable', width=120);
tree.column('area', width=100)

    scrollbar = ttk.Scrollbar(frame, orient=tk.VERTICAL, command=tree.yview)
    tree.configure(yscroll=scrollbar.set)
    tree.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)
    scrollbar.pack(side=tk.RIGHT, fill=tk.Y)

    def actualizar_treeview():
        for i in tree.get_children(): tree.delete(i)
        with sqlite3.connect(DB_FILE) as conn:
            conn.row_factory = sqlite3.Row
            cursor = conn.cursor()
            if rol_usuario == "admin":
                cursor.execute('''SELECT m.id, p.nombre, m.fecha, m.tipo,
m.cantidad, m.precio_unitario, m.comprobante, m.responsable, m.area FROM
movimientos_kardex m JOIN productos p ON m.producto_codigo = p.codigo ORDER BY
m.fecha DESC''')
            else:
                cursor.execute('''SELECT m.id, p.nombre, m.fecha, m.tipo,
m.cantidad, m.precio_unitario, m.comprobante, m.responsable, m.area FROM
movimientos_kardex m JOIN productos p ON m.producto_codigo = p.codigo WHERE
m.responsable = ? ORDER BY m.fecha DESC''', (nombre_usuario,))
            movimientos = cursor.fetchall()

        for mov in movimientos:
            fecha_ui = formatear_fecha_mostrar(mov['fecha'])
            precio_formateado = f"{mov['precio_unitario']:.2f}" if
mov['precio_unitario'] else "0.00"
            tree.insert('', tk.END, values=(mov['id'], mov['nombre'], fecha_ui,
mov['tipo'], mov['cantidad'], precio_formateado, mov['comprobante'],
mov['responsable'], mov['area'] or 'N/A'))

    def eliminar_movimiento():
        selected_item = tree.focus()
        if not selected_item:
            messagebox.showwarning("Selección", "Por favor, selecciona un
movimiento para eliminar.")
            return
        item_id = tree.item(selected_item)['values'][0]
        if messagebox.askyesno("Confirmar Eliminación", f"¿Estás seguro de que
quieres eliminar el movimiento con ID {item_id}?"):
            try:
                with sqlite3.connect(DB_FILE) as conn:
                    cursor = conn.cursor()
                    cursor.execute("DELETE FROM movimientos_kardex WHERE id = ?",
(item_id,))
                messagebox.showinfo("Éxito", "Movimiento eliminado correctamente.")
```

```
                actualizar_treeview()
                actualizar_saldos()
            except Exception as e:
                messagebox.showerror("Error", f"No se pudo eliminar el
movimiento.\nError: {e}")

    def editar_movimiento():
        selected_item = tree.focus()
        if not selected_item:
            messagebox.showwarning("Selección", "Por favor, selecciona un
movimiento para editar.")
            return
        item_id = tree.item(selected_item)['values'][0]

        with sqlite3.connect(DB_FILE) as conn:
            conn.row_factory = sqlite3.Row
            cursor = conn.cursor()
            cursor.execute("SELECT * FROM movimientos_kardex WHERE id = ?",
(item_id,))
            movimiento = cursor.fetchone()

        if not movimiento:
            messagebox.showerror("Error", "No se pudo encontrar el movimiento en la
base de datos.")
            return

        edit_win = tk.Toplevel(ventana_correccion)
        edit_win.title(f"Editar Movimiento ID: {item_id}")
        edit_win.geometry("420x400")
        if icono_path: edit_win.iconbitmap(icono_path)
        centrar_ventana(edit_win)

        fecha_ui = formatear_fecha_mostrar(movimiento['fecha'])

        tk.Label(edit_win, text="Fecha (DD-MM-YYYY):").grid(row=0, column=0,
padx=10, pady=5, sticky="w")
        entry_fecha = tk.Entry(edit_win)
        entry_fecha.insert(0, fecha_ui)
        entry_fecha.grid(row=0, column=1, padx=10, pady=5)

        tk.Label(edit_win, text="Cantidad:").grid(row=1, column=0, padx=10, pady=5,
sticky="w")
        entry_cantidad = tk.Entry(edit_win)
        entry_cantidad.insert(0, movimiento['cantidad'])
        entry_cantidad.grid(row=1, column=1, padx=10, pady=5)

        tk.Label(edit_win, text="Precio Unitario:").grid(row=2, column=0, padx=10,
pady=5, sticky="w")
        entry_precio = tk.Entry(edit_win)
        entry_precio.insert(0, movimiento['precio_unitario'] or 0.00)
        entry_precio.grid(row=2, column=1, padx=10, pady=5)

        tk.Label(edit_win, text="Comprobante:").grid(row=3, column=0, padx=10,
pady=5, sticky="w")
        entry_comprobante = tk.Entry(edit_win)
        entry_comprobante.insert(0, movimiento['comprobante'] or "")
        entry_comprobante.grid(row=3, column=1, padx=10, pady=5)

        tk.Label(edit_win, text="Responsable:").grid(row=4, column=0, padx=10,
pady=5, sticky="w")
        entry_responsable = tk.Entry(edit_win)
        entry_responsable.insert(0, movimiento['responsable'] or "")
        entry_responsable.grid(row=4, column=1, padx=10, pady=5)
```

```
        tk.Label(edit_win, text="Área:").grid(row=5, column=0, padx=10, pady=5,
sticky="w")
        frame_area_edit = tk.Frame(edit_win)
        frame_area_edit.grid(row=5, column=1, padx=10, pady=5, sticky="w")
        combo_area_edit = ttk.Combobox(frame_area_edit, values=obtener_areas_bd(),
width=24, state="readonly")
        combo_area_edit.pack(side=tk.LEFT)
        combo_area_edit.set(movimiento['area'] or "")
        tk.Button(frame_area_edit, text="+", width=3, font=("Arial", 10, "bold"),
bg="#4CAF50", fg="white",
                  command=lambda: agregar_area_rapida(combo_area_edit,
edit_win)).pack(side=tk.LEFT, padx=5)

        def guardar_cambios():
            nueva_fecha_ui = entry_fecha.get()
            nueva_cantidad = entry_cantidad.get()
            nuevo_precio = entry_precio.get()
            nuevo_comprobante = entry_comprobante.get()
            nuevo_responsable = entry_responsable.get()
            nueva_area = combo_area_edit.get()

            if not nueva_fecha_ui or not nueva_cantidad:
                messagebox.showerror("Error", "La fecha y la cantidad son
obligatorias.")
                return
            if not validar_fecha(nueva_fecha_ui):
                messagebox.showerror("Error", "Formato de fecha incorrecto (DD-MM-
YYYY).")
                return
            try:
                int(nueva_cantidad)
                float(nuevo_precio)
            except ValueError:
                messagebox.showerror("Error", "La cantidad y el precio deben ser
números válidos.")
                return

            nueva_fecha_db = formatear_fecha_almacenar(nueva_fecha_ui)

            try:
                with sqlite3.connect(DB_FILE) as conn:
                    cursor = conn.cursor()
                    cursor.execute('''UPDATE movimientos_kardex SET fecha = ?,
cantidad = ?, precio_unitario = ?, comprobante = ?, responsable = ?, area = ? WHERE
id = ?''',
                                   (nueva_fecha_db, int(nueva_cantidad),
float(nuevo_precio), nuevo_comprobante, nuevo_responsable, nueva_area, item_id))
                messagebox.showinfo("Éxito", "Movimiento actualizado
correctamente.")
                edit_win.destroy()
                actualizar_treeview()
                actualizar_saldos()
            except Exception as e:
                messagebox.showerror("Error", f"No se pudo actualizar el
movimiento.\nError: {e}")

        tk.Button(edit_win, text="Guardar Cambios", command=guardar_cambios,
bg="#4CAF50", fg="white", font=("Arial", 10, "bold")).grid(row=6, column=0,
columnspan=2, pady=20)

    frame_botones = tk.Frame(ventana_correccion)
    frame_botones.pack(pady=10, fill=tk.X)
```

```
    tk.Button(frame_botones, text="Editar Movimiento Seleccionado",
command=editar_movimiento, bg="#2196F3", fg="white", font=("Arial", 10,
"bold")).pack(side=tk.LEFT, padx=5)
    tk.Button(frame_botones, text="Eliminar Movimiento Seleccionado",
command=eliminar_movimiento, bg="#ff4d4d", fg="white", font=("Arial", 10,
"bold")).pack(side=tk.LEFT, padx=5)
    actualizar_treeview()

def realizar_backup_gui():
    fecha_actual = datetime.now().strftime("%Y-%m-%d")
    nombre_archivo_sugerido = f"kardex_backup_{fecha_actual}.db"
    ruta = filedialog.asksaveasfilename(title="Guardar Copia de Seguridad de la
Base de Datos", initialfile=nombre_archivo_sugerido, defaultextension=".db",
filetypes=[("Archivos de Base de Datos", "*.db"), ("Todos los archivos", "*.*")])
    if not ruta: return
    try:
        shutil.copy2(DB_FILE, ruta)
        messagebox.showinfo("Backup Exitoso", f"La copia de seguridad se ha
guardado correctamente en:\n{ruta}")
        notificar("Backup Realizado", "La base de datos ha sido respaldada con
éxito.")
    except Exception as e:
        messagebox.showerror("Error de Backup", f"No se pudo crear la copia de
seguridad.\nError: {e}")

def restaurar_backup_gui():
    ruta = filedialog.askopenfilename(title="Seleccionar Copia de Seguridad",
filetypes=[("Archivos de Base de Datos", "*.db"), ("Todos los archivos", "*.*")])
    if not ruta: return

    if messagebox.askyesno("⚠️ CONFIRMACIÓN CRÍTICA", "¿Está ABSOLUTAMENTE
seguro?\n\nEsto reemplazará TODOS los datos actuales por los del archivo de
respaldo seleccionado."):
        try:
            backup_previo = f"{DB_FILE}.pre_restore"
            if os.path.exists(DB_FILE):
                shutil.copy2(DB_FILE, backup_previo)

            shutil.copy2(ruta, DB_FILE)

            with sqlite3.connect(DB_FILE) as conn:
                conn.execute("SELECT name FROM sqlite_master WHERE type='table'")

            messagebox.showinfo("Éxito", "Base de datos restaurada
correctamente.\nEl sistema se cerrará para aplicar los cambios. Vuelva a
iniciarlo.")
            if root: root.destroy()

        except Exception as e:
            if os.path.exists(backup_previo):
                shutil.copy2(backup_previo, DB_FILE)
            messagebox.showerror("Error Crítico", f"No se pudo restaurar y se
revirtió el cambio.\nError: {e}")

# ---------------- BANDEJA DEL SISTEMA ----------------
def quit_application(icon, item):
    if root: root.after(0, root.destroy)
    icon.stop()
def show_window(icon, item):
    if root:
        root.after(0, root.deiconify)
        root.after(0, root.lift)
    icon.notify("Aplicación restaurada.")
```

```
def setup_tray_icon():
    global icon_tray
    if not icono_path or not os.path.exists(icono_path):
        print("No se puede crear el ícono de la bandeja del sistema porque el
archivo de imagen no existe.")
        return
    image = Image.open(icono_path)
    menu = pystray.Menu(item('Mostrar', show_window, default=True), item('Salir',
quit_application))
    icon_tray = pystray.Icon("intiqa_kardex", image, "Sistema de Kardex Intiqa",
menu)
    icon_tray.run()

# ---------------- CREAR VENTANA PRINCIPAL ----------------
def abrir_sistema():
    global root, frame_saldos

    root = tk.Tk()
    root.overrideredirect(True)
    root.attributes("-topmost", True)
    root.attributes("-alpha", 0.80)

    screen_width = root.winfo_screenwidth()
    window_width = 800
    window_height = 150
    x_position = screen_width - window_width
    y_position = 50
    root.geometry(f'{window_width}x{window_height}+{x_position}+{y_position}')

    root.configure(bg="#01525f")
    if icono_path: root.iconbitmap(icono_path)

    # ---> NUEVO: Ocultamos la ventana principal mientras carga el splash
    root.withdraw()

    def start_move(event): root.x = event.x; root.y = event.y
    def stop_move(event): root.x = None; root.y = None
    def on_move(event):
        deltax = event.x - root.x; deltay = event.y - root.y
        x = root.winfo_x() + deltax; y = root.winfo_y() + deltay
        root.geometry(f"+{x}+{y}")
    def on_closing():
        root.withdraw()
        if icon_tray: icon_tray.notify("Aplicación minimizada a la bandeja del
sistema.")

    barra = tk.Frame(root, bg="#e49f54", bd=2, relief=tk.RAISED)
    barra.pack(fill=tk.X, side=tk.TOP)
    barra.bind("<ButtonPress-1>", start_move)
    barra.bind("<ButtonRelease-1>", stop_move)
    barra.bind("<B1-Motion>", on_move)
    tk.Label(barra, text=f"Sistema de Kardex - {nombre_usuario}
({rol_usuario.upper()})", bg="#e49f54", fg="white", font=("Arial", 10,
"bold")).pack(side=tk.LEFT, padx=10, pady=5)
    tk.Button(barra, text="_", command=on_closing, bg="#01525f", fg="white",
font=("Arial", 10, "bold")).pack(side=tk.RIGHT, padx=5, pady=5)
    tk.Button(barra, text="X", command=root.destroy, bg="#01525f", fg="white",
font=("Arial", 10, "bold")).pack(side=tk.RIGHT, padx=5, pady=5)

    def mostrar_menu_opciones(event, opciones):
        menu_win = tk.Toplevel(root)
        menu_win.overrideredirect(True)
        menu_win.attributes("-topmost", True)
```

```
        botones_menu = []
        for texto, comando in opciones:
            btn = tk.Button(menu_win, text=texto, command=lambda c=comando:
[menu_win.destroy(), c()], font=("Arial", 10), relief=tk.FLAT, anchor="w", padx=10)
            btn.pack(fill=tk.X)
            botones_menu.append(btn)
        for btn in botones_menu:
            btn.bind("<Enter>", lambda e, b=btn: b.config(bg='#87CEEB',
relief=tk.RAISED))
            btn.bind("<Leave>", lambda e, b=btn: b.config(bg='SystemButtonFace',
relief=tk.FLAT))
        menu_win.update_idletasks()
        x = root.winfo_x() + event.x_root - root.winfo_rootx()
        y = root.winfo_y() + event.y_root - root.winfo_rooty() + 25
        ancho_pantalla = menu_win.winfo_screenwidth()
        ancho_menu = menu_win.winfo_width()
        if x + ancho_menu > ancho_pantalla:
            x = ancho_pantalla - ancho_menu
        menu_win.geometry(f"+{int(x)}+{int(y)}")
        menu_win.configure(bg="lightgrey", bd=1, relief=tk.RAISED)
        menu_win.bind("<FocusOut>", lambda e: menu_win.destroy())
        menu_win.focus_set()

    frame_menu_bar = tk.Frame(root, bg="#01525f")
    frame_menu_bar.pack(fill=tk.X, side=tk.TOP, pady=5)

    opciones_inventario = [("Registrar Ingreso", registrar_ingreso_gui),
("Registrar Egreso", registrar_egreso_gui)]
    if rol_usuario == "admin":
        opciones_inventario.insert(0, ("Agregar Producto", agregar_producto_gui))

    opciones_reportes = [
        ("Ver Kardex Completo", mostrar_kardex_gui),
        ("Buscar Producto", buscar_producto),
        ("Ver Productos con Stock Bajo", mostrar_reporte_stock_bajo_gui),
        ("Gastos por Área", mostrar_reporte_gastos_area_gui)
    ]
    if rol_usuario == "admin":
        opciones_reportes.extend([("Exportar a Excel", exportar_excel), ("Resumen
de Caja", mostrar_resumen_caja_gui)])

    opciones_herramientas = [("Corregir Mis Movimientos",
corregir_movimientos_gui)]
    if rol_usuario == "admin":
        opciones_herramientas.extend([
            ("Gestionar Productos", gestionar_productos_gui),
            ("Gestionar Áreas", gestionar_areas_gui),
            ("Backup de Base de Datos", realizar_backup_gui),
            ("Restaurar Backup", restaurar_backup_gui),
            ("Gestionar Usuarios", agregar_usuario_gui)
        ])

    opciones_salir = [("Minimizar", on_closing), ("Cerrar Programa", root.destroy)]

    boton_inventario = tk.Button(frame_menu_bar, text="Inventario", bg="#e49f54",
fg="white", font=("Arial", 10, "bold"), relief=tk.RAISED)
    boton_inventario.pack(side=tk.LEFT, padx=5)
    boton_inventario.bind("<Button-1>", lambda e: mostrar_menu_opciones(e,
opciones_inventario))

    boton_reportes = tk.Button(frame_menu_bar, text="Reportes", bg="#e49f54",
fg="white", font=("Arial", 10, "bold"), relief=tk.RAISED)
    boton_reportes.pack(side=tk.LEFT, padx=5)
```

```
    boton_reportes.bind("<Button-1>", lambda e: mostrar_menu_opciones(e,
opciones_reportes))

    boton_herramientas = tk.Button(frame_menu_bar, text="Herramientas",
bg="#e49f54", fg="white", font=("Arial", 10, "bold"), relief=tk.RAISED)
    boton_herramientas.pack(side=tk.LEFT, padx=5)
    boton_herramientas.bind("<Button-1>", lambda e: mostrar_menu_opciones(e,
opciones_herramientas))

    boton_salir = tk.Button(frame_menu_bar, text="Salir", bg="#e49f54", fg="white",
font=("Arial", 10, "bold"), relief=tk.RAISED)
    boton_salir.pack(side=tk.RIGHT, padx=5)
    boton_salir.bind("<Button-1>", lambda e: mostrar_menu_opciones(e,
opciones_salir))

    frame_saldos = tk.Frame(root, bg="lightyellow", bd=1, relief=tk.SUNKEN)
    frame_saldos.pack(fill=tk.BOTH, expand=True, padx=5, pady=5, side=tk.BOTTOM)

    actualizar_saldos()

    if icono_path and os.path.exists(icono_path):
        tray_thread = threading.Thread(target=setup_tray_icon, daemon=True)
        tray_thread.start()

    # ---> NUEVO: Función que se ejecutará cuando termine el Splash Screen
    def mostrar_ventana_principal():
        root.deiconify() # Muestra la ventana principal
        root.lift()

    # ---> NUEVO: Llamamos al splash pasándole la raíz y qué hacer al terminar
    mostrar_splash_screen(root, mostrar_ventana_principal)

    root.mainloop()

# ---------------- LOGIN ----------------
def login():
    def verificar():
        nonlocal login_win
        usuario = entry_user.get()
        clave = entry_pass.get()
        global rol_usuario, nombre_usuario

        with sqlite3.connect(DB_FILE) as conn:
            conn.row_factory = sqlite3.Row
            cursor = conn.cursor()

            cursor.execute("SELECT role FROM usuarios WHERE username = ? AND
password = ?", (usuario, hash_password(clave)))
            resultado = cursor.fetchone()

            if not resultado:
                cursor.execute("SELECT role FROM usuarios WHERE username = ? AND
password = ?", (usuario, clave))
                resultado = cursor.fetchone()

                if resultado:
                    cursor.execute("UPDATE usuarios SET password = ? WHERE username
= ?", (hash_password(clave), usuario))
                    conn.commit()

        if resultado:
            rol_usuario = resultado['role']
            nombre_usuario = usuario
```

```
                login_win.destroy()
                abrir_sistema() # ---> Se eliminó la llamada al splash de aquí
            else:
                messagebox.showerror("Error", "Credenciales incorrectas.")

    login_win = tk.Tk()
    login_win.title("Login - Intiqa Kardex")
    login_win.configure(bg="#01525f")
    login_win.attributes('-alpha', 0.92)
    login_win.resizable(False, False)
    if icono_path: login_win.iconbitmap(icono_path)
    screen_width = login_win.winfo_screenwidth()
    screen_height = login_win.winfo_screenheight()
    window_width = 300; window_height = 200
    x = (screen_width / 2) - (window_width / 2)
    y = (screen_height / 2) - (window_height / 2)
    login_win.geometry(f'{window_width}x{window_height}+{int(x)}+{int(y)}')

    frame_login = tk.Frame(login_win, bg="#01525f")
    frame_login.pack(expand=True, fill=tk.BOTH, padx=20, pady=20)
    tk.Label(frame_login, text="Usuario:", bg="#01525f", fg="white", font=("Arial",
10)).pack(pady=(0, 5))
    entry_user = tk.Entry(frame_login, font=("Arial", 10))
    entry_user.pack(fill=tk.X, ipady=2)
    tk.Label(frame_login, text="Contraseña:", bg="#01525f", fg="white",
font=("Arial", 10)).pack(pady=(10, 5))
    entry_pass = tk.Entry(frame_login, show="*", font=("Arial", 10))
    entry_pass.pack(fill=tk.X, ipady=2)
    entry_pass.bind('<Return>', lambda event: verificar())
    tk.Button(frame_login, text="Ingresar", command=verificar, bg="#e49f54",
fg="white", font=("Arial", 12, "bold"), width=20).pack(pady=(20, 0), ipady=20)
    login_win.mainloop()

# ---------------- EJECUTAR ----------------
crear_tablas_si_no_existen()
migrar_datos_desde_json()
login()
```

TESIS UANCV
VICERRECTORADO DE INVESTIGACIÓN
"OFICINA DE INVESTIGACIÓN"

ANEXO 03: MATRIZ DE CONSISTENCIA

ELEMENTOS	PROBLEMAS	OBJETIVOS	HIPÓTESIS	VARIABLES	MÉTODO, DISEÑO, TIPO, TÉCNICA E INSTRUMENTO
GENERAL	¿De qué manera el desarrollo e implementación de un sistema de inventario en python, basado en trazabilidad por centros de costos, contribuirá a optimizar la gestión de almacén y mitigar las pérdidas no registradas de insumos en el Intiqa Hotel Puno, durante el año 2025?	Desarrollar e implementar un sistema de inventario en python, basado en trazabilidad por centros de costos, para optimizar la gestión de almacén y mitigar las pérdidas no registradas de insumos en el Intiqa Hotel Puno, durante el año 2025.	El desarrollo e implementación de un sistema de inventario en Python, estructurado bajo el enfoque de trazabilidad por centros de costos, se relaciona positivamente con la optimización de la gestión de almacén y la mitigación de las pérdidas no registradas de insumos en el Intiqa Hotel Puno, durante el año 2025.	VARIABLE INDEPENDIENTE: Sistema de Inventario En Python.	Enfoque: cuantitativo. Tipo: aplicado. Nivel: descriptivo-explicativo. Diseño: no experimental, de corte transversal. Método: científico, deductivo e hipotético-deductivo. Muestra: censo (100% de la población) - sustentado en Hernández Sampieri (2014).
ESPECÍFICO 1	¿Cuál es el nivel de incidencia que tiene la actual ausencia de trazabilidad y de registros desagregados por áreas de consumo (cocina, barra, pisos) en la pérdida de bienes de alta rotación (bebidas y alimentos) del Intiqa Hotel Puno?	Determinar el nivel de incidencia que tiene la actual ausencia de trazabilidad y de registros desagregados por áreas de consumo (cocina, barra, pisos) en la pérdida de bienes de alta rotación (bebidas y alimentos) del Intiqa Hotel Puno.	La ausencia actual de trazabilidad por áreas de consumo se asocia de manera directa con la pérdida injustificada de bienes de alta rotación (bebidas y alimentos) en el Intiqa Hotel Puno.		Técnica: observación directa. Instrumento: ficha de observación del sistema.
ESPECÍFICO 2	¿Qué características funcionales y de seguridad (control de acceso por roles, vinculación de vales de salida con facturas de ingreso y alertas de stock mínimo) debe integrar el sistema de información en python para garantizar la auditoría y el control transaccional del almacén?	Diseñar e implementar el sistema de información en python que integre control de acceso por roles, vinculación de vales de salida con facturas de ingreso y alertas de stock mínimo, para garantizar la auditoría y el control transaccional del almacén.	El sistema de información en Python, al integrar control de acceso por roles y la vinculación digital de vales de salida con facturas de ingreso, permite dejar un registro claro de quién hizo cada movimiento que favorece la auditoría y el control transaccional del almacén.	VARIABLE DEPENDIENTE: Gestión de Almacén.	Técnica: observación directa. Instrumento: ficha de observación de sistemas.
ESPECÍFICO 3	¿De qué manera la automatización de los reportes estadísticos por área de consumo y la centralización de la información en el sistema propuesto mejorarán la eficiencia en los tiempos de consulta y la toma de decisiones gerenciales en el Intiqa Hotel Puno?	Evaluar la mejora en la eficiencia de los tiempos de consulta y la toma de decisiones gerenciales, a partir de la automatización de reportes estadísticos por área de consumo y la centralización de la información en el sistema propuesto.	La centralización de la información y la generación automatizada de reportes estadísticos desagregados por áreas de consumo se relacionan con una mayor eficiencia operativa (reducción de tiempos de consulta) y facilitan la toma de decisiones gerenciales en el Intiqa Hotel Puno.		Técnica: encuesta. Instrumento: cuestionario estructurado de 10 ítems, empleando una escala de medición tipo likert de 5 puntos.

OFICINA DE INVESTIGACIÓN
Tesis Publicada con autorización del autor
http://repositorio.uancv.edu.pe/

VALIDACIÓN DE INSTRUMENTO POR JUICIO DE EXPERTO

Perfil: Ingeniería de Sistemas

1.1. Objetivo de la validación

Determinar la validez de contenido del cuestionario estructurado de 10 ítems, diseñado para evaluar el sistema de inventario en Python (Variable Independiente), mediante el juicio de un experto en Ingeniería de Sistemas, garantizando que las preguntas midan adecuadamente los estándares técnicos, de seguridad lógica y de usabilidad del software desarrollado.

1.2. Identificación del experto validador

Nombre: Jose Daniel Zamalloa Nuñez

Grado Académico: Maestro en Análisis y Visualización de Datos Masivos

Especialidad: Ingeniería de Sistemas e Ingeniería de Datos

Institución: INDRA Perú

1.3. Matriz de Juicio de Experto

Escala: 1 = Totalmente en desacuerdo, 2 = En desacuerdo, 3 = Indiferente, 4 = De acuerdo, 5 = Totalmente de acuerdo.

Ítem	Criterio a Evaluar	Dimensión que valida	Puntaje
1	¿El ítem mide adecuadamente la funcionalidad técnica del sistema?	Funcionalidad Técnica	5
2	¿El ítem es claro y comprensible para el usuario final?	Claridad Lingüística	5
3	¿El ítem es coherente con los principios de seguridad informática (RBAC y Hash)?	Coherencia Teórica (Seguridad)	5
4	¿El ítem evalúa correctamente la interfaz gráfica y experiencia de usuario (UX)?	Usabilidad (UX)	5
5	¿El ítem es pertinente para validar un software de esta naturaleza?	Pertinencia	5
6	¿El ítem es coherente con la arquitectura de software propuesta?	Coherencia Arquitectónica	5
7	¿El ítem mapea de forma precisa la integración de las bases de datos?	Funcionalidad Técnica	5
8	¿El ítem es claro en su redacción?	Claridad Lingüística	5

Ítem	Criterio a Evaluar	Dimensión que valida	Puntaje
9	¿El ítem evalúa la eficiencia técnica del motor transaccional (Kardex)?	Coherencia Teórica (Kardex)	5
10	¿El ítem es pertinente para el perfil del sistema desarrollado?	Pertinencia	5
PROMEDIO			5.0

1.4. Conclusión

El instrumento de medición (Cuestionario de 10 ítems) obtuvo una puntuación perfecta en la evaluación del experto en Ingeniería de Sistemas, demostrando que las preguntas poseen total claridad, pertinencia y coherencia técnica. Por lo tanto, se concluye que el instrumento es válido desde la perspectiva del desarrollo de software.

1.5. Firma del Validador

Nombre y Firma: _____

Mg. Ing. Jose Daniel Zamalloa Nuñez
INGENIERO DE SISTEMAS
C.I.P. N.° 216174

VALIDACIÓN DE INSTRUMENTO POR JUICIO DE EXPERTO

Perfil: Profesional experto en Gestión Logística y Operaciones, con grado de Magíster en Educación Superior por la Universidad Nacional del Altiplano con amplia trayectoria en el sector turismo (Hotelería) Administrador Intiqa Hotel Puno, Con Experiencia Laboral Como Jefe de Recepción Hotel Royal Inn 4 Estrellas, Auditor Hotelero Casa Andina Nessus Hoteles Perú S.A., Jefe de recepción ECO Inn Hotel 4 estrellas,

2.1. Objetivo de la validación

Determinar la validez de contenido del cuestionario estructurado de 10 ítems, diseñado para medir la optimización de la gestión de almacén (Variable Dependiente), mediante el juicio de un docente experto en gestión logística y operaciones, garantizando que las preguntas midan adecuadamente la trazabilidad por centros de costos, la reducción de mermas y la toma de decisiones gerenciales.

2.2. Identificación del experto validador

Nombre: Guido Vilmar Mamani Lujano

Grado Académico: Mg. En Educación Superior por la UNAP

Especialidad: Administrador Hotelero

Institución: Administrador Intiqa Hotel Puno, Con Experiencia Laboral Como Jefe de Recepción Hotel Royal Inn 4 Estrellas, Auditor Hotelero Casa Andina Nessus Hoteles Perú S.A., Jefe de recepción ECO Inn Hotel 4 estrellas,

2.3. Matriz de Juicio de Experto

Escala: 1 = Totalmente en desacuerdo, 2 = En desacuerdo, 3 = Indiferente, 4 = De acuerdo, 5 = Totalmente de acuerdo.

Ítem	Criterio a Evaluar	Dimensión que valida	Puntaje
1	¿El ítem mide adecuadamente la incidencia de la trazabilidad por áreas?	Trazabilidad por Centros de Costos	5
2	¿El ítem es pertinente para medir la mitigación de mermas operativas?	Mitigación de Mermas Operativas	5
3	¿El ítem es coherente con la teoría de asignación por centros de costos?	Coherencia Teórica (Centros de Costos)	5
4	¿El ítem es claro y comprensible para un gerente o logístico?	Claridad Gerencial	5
5	¿El ítem es coherente con las teorías de alertamiento temprano y stock mínimo?	Coherencia Teórica (Alertamiento Temprano)	5.

Ítem	Criterio a Evaluar	Dimensión que valida	Puntaje
6	¿El ítem evalúa correctamente la toma de decisiones basada en datos?	Toma de Decisiones Basada en Datos	5
7	¿El ítem es pertinente para evaluar la eficiencia operativa hotelera?	Eficiencia Operativa Logística	5
8	¿El ítem es claro en su redacción?	Claridad Lingüística	4
9	¿El ítem mide de forma precisa la cadena de custodia lógica (auditoría)?	Auditoría y Cadena de Custodia	5
10	¿El ítem es coherente con los indicadores de rendimiento logístico?	Indicadores de Rendimiento Logístico	5
PROMEDIO			4.9

2.4. Conclusión

El instrumento de medición obtuvo una puntuación altísima en la evaluación del experto en Gestión Logística, demostrando que las preguntas poseen total pertinencia y coherencia teórica con la realidad del almacén hotelero. El experto confirmó que el instrumento aborda los KPIs críticos de la logística moderna. Por lo tanto, se concluye que el instrumento es válido desde la perspectiva de la gestión operativa hotelera.

2.5. Firma del Validador

Nombre y Firma: _____

M.Sc Guido V. Mamani Lujano
ADMINISTRADOR

TESIS UANCV
TESIS UANCV
VICERRECTORADO DE INVESTIGACIÓN
"OFICINA DE INVESTIGACIÓN"

ANEXO 1
FORMULARIO DE AUTORIZACIÓN

AUTORIZACIÓN PARA LA INCORPORACIÓN DE LOS TRABAJOS DE INVESTIGACIÓN EN EL REPOSITORIO INSTITUCIONAL UANCV

Formato digital [X] Fecha de entrega: 21 - 07 - 2026

1. Datos del autor (es):

Nombres y Apellidos: ABELARDO IDME MAMANI
Dirección: Jr. Zarumilla Nro. 518
DNI/Carné de Extranjería/Pasaporte N°: 42169500
Teléfono: 950 349 092 email: idmema1@gmail.com

Nombres y Apellidos:
Dirección:
DNI/Carné de Extranjería/Pasaporte N°:
Teléfono: email:

Facultad y/o Escuela de Posgrado: FACULTAD DE INGENIERÍA DE SISTEMAS
Escuela Profesional o Mención: ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS
Título o Grado Académico a optar: INGENIERO DE SISTEMAS
Asesor: Dr. PAUL MAMANI TISNADO
Esta obra se encuentra dentro de las siguientes denominaciones:
Trabajo de Investigación [ ] Tesis [X] Trabajo de Suficiencia Profesional [ ] Trabajo Académico [ ]
Título: DESARROLLO DE UN SISTEMA DE INVENTARIO PARA OPTIMIZAR LA GESTIÓN DE ALMACÉN EN LA EMPRESA INTIQA HOTEL PUNO 2025

Palabras claves, (3 a 5 términos): Sistema de inventario, gestión de almacén, trazabilidad, centros de costos, Python.

¿Esta obra se desarrolló en la UANCV [1,2]?

1

[1] Indicar si su producción intelectual ha empleado recursos tales como, instalaciones, laboratorios, insumos, equipos, bases de datos, asesoría técnica por parte del personal de la UANCV, financiamiento, entré otros relacionados.
[2] Si su producción intelectual se desarrolló en la UANCV totalmente o parcialmente, deberá autorizar el depósito en el Repositorio de manera obligatoria.

OFICINA DE INVESTIGACIÓN
Tesis Publicada con autorización del autor
http://repositorio.uancv.edu.pe/

TESIS UANCV
TESIS UANCV
VICERRECTORADO DE INVESTIGACIÓN
"OFICINA DE INVESTIGACIÓN"

2. Referencia de tesis:

☐ Bachiller ☒ Titulo ☐ 2da Especialidad ☐ Maestría ☐ Doctorado

3. Licencias:

a) Licencia estándar:

Bajo los siguientes términos, autorizo el depósito de mi tesis en el Repositorio Digital de la UANCV.

Con la autorización de depósito de mi producción Intelectual, otorgo a la Universidad Andina "Néstor Cáceres Velásquez" una licencia no exclusiva para reproducir, distribuir, comunicar al público, transformar (únicamente mediante su traducción a otros idiomas) y poner a disposición del público mi producción intelectual (incluido el resumen), en formato físico o digital, en cualquier medio, conocido o por conocerse, a través de los diversos servicios por la Universidad, creados o por crearse, tales como el Repositorio Digital de tesis UANCV, colección de producción intelectual, entre otros, en el Perú y en el extranjero por el tiempo y veces que considere necesarias, y libres de remuneraciones.

En virtud de dicha licencia, la Universidad Andina "Néstor Cáceres Velásquez" podrá reproducir mi producción intelectual en cualquier tipo de soporte y en más de un ejemplar, sin modificar su contenido, solo con propósitos de seguridad, respaldo y preservación.

Declaro que la producción intelectual es una creación de mi autoría y exclusiva titularidad, coautoría con titularidad compartida, y me encuentro facultado a conceder la presente licencia y, asimismo, garantizo que dicha producción intelectual no infringe derechos de autor de terceras personas.

La Universidad Andina "Néstor Cáceres Velásquez" consignará el nombre del y/o los autor(es) de la producción intelectual, y no le hará ninguna modificación más que la permitida en la licencia.

Autorizo su publicación (marque con una X)

☒ Sí, autorizo que se deposite inmediatamente.

☐ Sí, autorizo que se deposite a partir de la fecha (d/m/a): _____

☐ No autorizo.

b) Licencia CREATIVE COMMONS 4.0 INTERNACIONAL:

Si usted concede una licencia CREATIVE COMMONS sobre su producción intelectual, mantiene la titularidad de los derechos de autor de esta y, a la vez, permite que otras personas puedan reproducirla, comunicarla al público y distribuir ejemplares de esta, bajo las condiciones siguientes:

¿Quiere permitir usos comérciales de su producción intelectual?

Sí: significa que usted permite la reproducción, distribución y comunicación pública de la producción intelectual incluso con fines comerciales.

No: significa que usted permite la reproducción, y comunicación pública de la producción intelectual, pero sin fines comerciales.

☐ Sí autorizo

☒ No autorizo

OFICINA DE INVESTIGACIÓN
Tesis Publicada con autorización del autor
http://repositorio.uancv.edu.pe/

TESIS UANCV
VICERRECTORADO DE INVESTIGACIÓN
"OFICINA DE INVESTIGACIÓN"

Jurisdicción de su Licencia

Todas las licencias CREATIVE COMMONS son de ámbito mundial, sin embargo, usted puede elegir entre la opción "internacional" o una adaptada a su jurisdicción, como para el caso peruano.

La opción "internacional" emplea el lenguaje y la terminología de los tratados internacionales; en cambio, la adaptada a su jurisdicción, recoge las particularidades de la legislación peruana.

En consecuencia, **la opción "internacional" goza de una mayor eficacia a nivel mundial, gracias a que tiene jurisdicción neutral**. Mientras que la opción adaptada a la jurisdicción del Perú goza de una mayor eficacia ante los tribunales peruanos.

☐ Internacional

☒ Nacional

Línea de investigación: CIENCIA DE LOS ORDENADORES – P24

UNIVERSIDAD ANDINA NESTOR CACERES JULIACA

Firma de Autor | huella digital | 21 – 07 - 2026 Fecha

OFICINA DE INVESTIGACIÓN
Tesis Publicada con autorización del autor
http://repositorio.uancv.edu.pe/