

---

# TICASHOP

---

## *Implementación de la solución de software*

NOMBRE: Felipe Silva R, Victoria Carrasco, Matias Diaz

CARRERA: Analista Programador

ASIGNATURA: Proyecto Integrado

PROFESOR: Ivan Riquelme

FECHA ENTREGA: 18 de noviembre, 2025

# Índice

|  |           |
|--|-----------|
| <b>Introducción.....</b>   | <b>3</b>  |
| <b>Metodologías de desarrollo de software seleccionadas.....</b>   | <b>4</b>  |
| <b>Brainstorming (Anexo 1).....</b>  | <b>5</b>  |
| <b>Carta Gantt (Anexo 2).....</b>  | <b>6</b>  |
| <b>Product backlog.....</b>  | <b>7</b>  |
| <b>Plan de sprint.....</b>   | <b>8</b>  |
| <b>Tablero kanban (Anexo 3).....</b>   | <b>9</b>  |
| <b>Paradigma 4+1 (Anexo 3).....</b>  | <b>10</b> |
| Vista de escenarios.....   | 10        |
| Diagrama de casos de uso.....  | 10        |
| Vista lógica   |           |
| Diagrama de clases.....  | 11        |
| Gestión de Usuarios y Empleados.....   | 12        |
| Módulo de Cotizaciones.....  | 12        |
| Gestión de Inventario e Importaciones.....   | 12        |
| Soporte técnico.....   | 12        |
| Diagrama de secuencia: Registrar automáticamente entradas de productos<br>mediante código de barras..... | 13        |
| Diagrama de Secuencias: Sistema de Gestión de Importaciones.....   | 14        |
| Diagrama de secuencia: Generar cotizaciones a partir de órdenes de compra...                             | 15        |
| Diagrama de comunicación.....  | 16        |
| Vista de procesos.....   | 17        |
| Diagramas de actividades.....  | 17        |
| Diagrama de BPMN.....  | 20        |
| Diagrama de paquetes.....  | 21        |
| Diagrama de componentes.....   | 22        |
| Vista física.....  | 23        |
| Diagrama de despliegue.....  | 23        |
| Diagrama de topología.....   | 24        |
| <b>Especificación de tecnologías y herramientas requeridas para la infraestructura.....</b>              | <b>24</b> |
| Frontend.....  | 24        |
| Backend.....   | 25        |
| Base de datos.....   | 25        |
| Autenticación y seguridad.....   | 25        |
| Herramientas de apoyo.....   | 25        |
| Infraestructura local.....   | 26        |
| <b>Políticas de autenticación y autorización.....</b>  | <b>26</b> |
| <b>Gestión de tokens y sesiones.....</b>   | <b>28</b> |
| Responsabilidades por rol.....   | 29        |
| <b>Amenazas comunes y medidas de mitigación.....</b>   | <b>30</b> |

|   |           |
|---|-----------|
| <b>Estándares de codificación segura.....</b>                                 | <b>31</b> |
| <b>Herramientas de Pruebas: Automatizadas y manuales.....</b>                 | <b>32</b> |
| <b>Diseño de Base de Datos (Anexo 3).....</b>                                 | <b>35</b> |
| Roles y Perfiles de Usuario.....  | 36        |
| Perfiles que interactúan con el sistema.....                                  | 36        |
| Roles definidos en el sistema.....  | 36        |
| Roles que intervienen en el módulo de RR.HH. (alcance del mockup).....        | 37        |
| <b>Estándares de Programación Segura con roles y perfiles de usuario.....</b> | <b>38</b> |
| <b>Plan de pruebas (Ver Anexo 5).....</b>                                     | <b>39</b> |
| Tipos de pruebas a realizar.....  | 40        |
| 1. Pruebas funcionales.....   | 40        |
| 2. Pruebas de integración.....  | 40        |
| 3. Pruebas de seguridad.....  | 40        |
| 4. Pruebas de rendimiento.....  | 41        |
| 5. Pruebas de usabilidad.....   | 41        |
| Recomendaciones y marco de referencia OWASP.....                              | 41        |
| <b>Conclusión.....</b>  | <b>42</b> |
| <b>Ev3.....</b>   | <b>42</b> |
| <b>Implementación de software.....</b>  | <b>43</b> |
| <b>Alineación entre la interfaz y las necesidades del negocio.....</b>        | <b>43</b> |
| Capturas de pantalla del sistema/mocks/código funcionando.....                | 44        |
| <b>Coherencia visual y funcional en el diseño de la interfaz.....</b>         | <b>47</b> |
| <b>Estructura de la base de Datos y organización de la información.....</b>   | <b>48</b> |
| <b>Optimización y normalización de la base de datos.....</b>                  | <b>50</b> |
| 1. Normalización de datos.....  | 50        |
| 1.1 Primera Forma Normal (1NF).....   | 50        |
| 1.2 Segunda Forma Normal (2NF).....   | 51        |
| 1.3 Tercera Forma Normal (3NF).....   | 51        |
| 2. Optimización estructural aplicada.....                                     | 51        |
| 2.1 Separación lógica de entidades.....                                       | 51        |
| 2.3 Cálculo automático de campos.....   | 52        |
| 2.4 Columnas opcionales para compatibilidad.....                              | 53        |
| 2.5 Uso de DecimalField para cálculos financieros.....                        | 53        |
| 2.6 Estados normalizados con choices.....                                     | 53        |
| 3. Optimización orientada al rendimiento.....                                 | 54        |
| 3.1 Minimización de redundancia.....  | 54        |
| 3.2 Queries más eficientes.....   | 54        |
| 3.3 Eliminación de datos duplicados.....                                      | 54        |
| <b>Patrones de seguridad aplicados en el módulo de RRHH.....</b>              | <b>55</b> |
| <b>Colaboración en equipo.....</b>  | <b>55</b> |
| Historial de commits del proyecto TICASHOP.....                               | 56        |
| <b>Configuración del entorno de trabajo.....</b>                              | <b>57</b> |
| <b>Documentación de la implementación.....</b>                                | <b>57</b> |

|   |           |
|---|-----------|
| Estructura de archivos del proyecto TiCaShop LATAM en Visual Studio Code... | 59        |
| <b>Cobertura del plan de pruebas.....</b>                                   | <b>59</b> |
| <b>Ejecución del Protocolo de Pruebas.....</b>                              | <b>62</b> |
| <b>Validación de Resultados.....</b>  | <b>62</b> |
| <b>Recomendaciones.....</b>   | <b>63</b> |
| <b>Conclusión.....</b>  | <b>63</b> |
| <b>III. MANUAL DE USUARIO.....</b>  | <b>63</b> |
| Módulo de Recursos Humanos.....   | 64        |
| <b>1. Introducción.....</b>   | <b>64</b> |
| <b>2. Objetivo del Sistema.....</b>   | <b>64</b> |
| <b>3. Requisitos del Sistema.....</b>                                       | <b>65</b> |
| 3.1 Requisitos de Software.....   | 65        |
| 3.2 Librerías de Python.....  | 65        |
| <b>4. Instalación del Sistema.....</b>                                      | <b>65</b> |
| 4.1 Descarga del Proyecto desde GitHub.....                                 | 65        |
| 4.2 Creación de un Entorno Virtual.....                                     | 66        |
| 4.3 Instalación de Dependencias.....  | 66        |
| 4.4 Configuración de la Base de Datos.....                                  | 66        |
| 4.5 Ejecución del Servidor.....   | 67        |
| <b>5. Inicio de Sesión.....</b>   | <b>67</b> |
| <b>6. Roles y Permisos de Usuario.....</b>                                  | <b>68</b> |
| <b>7. Guía de Uso del Sistema.....</b>                                      | <b>68</b> |
| 7.1 Dashboard Principal.....  | 68        |
| 7.2 Módulo de Vacaciones.....   | 69        |
| Solicitar Vacaciones (Empleado).....  | 69        |
| Aprobar o Rechazar (RRHH/ADMIN).....  | 69        |
| 7.3 Módulo de Liquidaciones.....  | 69        |
| Ver Liquidaciones (Empleado/Nómina).....                                    | 69        |
| Administrar Liquidaciones (Nómina).....                                     | 69        |
| 7.4 Módulo de Asistencia.....   | 70        |
| 7.5 Módulo de Comisiones.....   | 70        |
| 7.6 Panel de Administración (ADMIN).....                                    | 70        |
| <b>8. Exportación de Información.....</b>                                   | <b>71</b> |
| <b>9. Seguridad del Sistema.....</b>  | <b>71</b> |
| <b>10. Resolución de Problemas Comunes.....</b>                             | <b>72</b> |
| <b>11. Conclusión.....</b>  | <b>72</b> |
| <b>Bibliografía.....</b>  | <b>72</b> |
| <b>Anexos.....</b>  | <b>72</b> |

# Introducción

La empresa TiCaShop LATAM, dedicada a la comercialización de productos tecnológicos y de telecomunicaciones, enfrenta desafíos operativos derivados del uso de procesos manuales y planillas independientes. Estos métodos generan errores en el control de stock, duplicidad de información, lentitud en la gestión de cotizaciones y escasa trazabilidad en solicitudes internas.

Con el propósito de optimizar sus procesos administrativos y mejorar la eficiencia global de la organización, se desarrolló un sistema ERP (Enterprise Resource Planning) denominado TiCaShop LATAM, orientado a integrar en una sola plataforma digital las áreas de Inventario, Ventas, Finanzas, Recursos Humanos y Soporte Técnico.

El sistema fue construido bajo una arquitectura cliente–servidor, ejecutada en un entorno local (localhost), utilizando Django/Python para el backend, React para el frontend y MariaDB como gestor de base de datos. Esta infraestructura local permite simular un entorno profesional de despliegue, manteniendo compatibilidad con una futura migración hacia plataformas en la nube como AWS o Azure.

El desarrollo se realizó aplicando la metodología ágil Scrum, que favorece la colaboración continua, la entrega incremental de funcionalidades y la mejora progresiva del sistema en cada sprint. Además, se implementaron buenas prácticas de seguridad basadas en OWASP, pruebas funcionales y de rendimiento, y un diseño modular que garantiza escalabilidad y mantenibilidad.

En conjunto, el proyecto TiCaShop LATAM busca transformar los procesos internos de la empresa mediante la automatización, la centralización de datos y la trazabilidad operativa, contribuyendo a una gestión más segura, eficiente y alineada con los estándares actuales de la industria tecnológica.

## **Metodologías de desarrollo de software seleccionadas**

El desarrollo del sistema TiCaShop se realizará bajo el marco de trabajo ágil Scrum, complementado con herramientas visuales de Kanban para el control del flujo de tareas.

Esta metodología fue elegida por las siguientes razones:

- Flexibilidad y adaptabilidad: permite ajustar prioridades y requisitos durante el desarrollo, algo clave en un proyecto de transformación digital en evolución constante
- Entregas incrementales: el software se construye en versiones funcionales (sprints), lo que posibilita validar avances con los usuarios finales y hacer mejoras continuas
- Colaboración constante: fomenta la comunicación entre desarrolladores, analistas y los representantes de TiCaShop, asegurando que el producto final responda a las necesidades reales.
- Gestión visual del progreso: mediante tableros Kanban y reuniones diarias (daily scrum) se mantiene la trazabilidad del trabajo y se evitan cuellos de botella

### **Roles principales:**

Product Owner: encargado de priorizar el backlog (Felipe Silva).

Scrum Master: encargado de guiar al equipo y remover obstáculos (Victoria Carrasco).

Equipo de desarrollo: programación y pruebas (Matías Díaz, Victoria Carrasco, Felipe Silva)

## **Brainstorming (Anexo 1)**

Durante la primera fase del proyecto realizamos un brainstorming colaborativo con el objetivo de identificar las principales áreas funcionales y requerimientos del sistema TiCaShop.

Esta actividad nos permitió organizar las ideas, visualizar los distintos módulos que conformarán la solución y definir los procesos clave que debían ser considerados en el desarrollo. El resultado de este ejercicio se muestra a continuación, donde se representan los componentes centrales del sistema y sus respectivas funciones:



Carta Gantt (Anexo 2)

| Fase                             | Actividades  | Semana 1 | Semana 2 | Semana 3 | Semana 4 | Semana 5 | Semana 6 | Semana 7 | Semana 8 | Semana 9 | Semana 10 | Semana 11 | Semana 12 | Semana 13 | Semana 14 | Semana 15 | Semana 16 | Semana 17 |
|----------------------------------|--|----------|----------|----------|----------|----------|----------|----------|----------|----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| Análisis y diagnóstico           | Levantamiento de información empresarial                                     |          |          |          |          |          |          |          |          |          |           |           |           |           |           |           |           |           |
|                                  | Aplicación de la técnica de los 5 porqués                                    |          |          |          |          |          |          |          |          |          |           |           |           |           |           |           |           |           |
|                                  | Identificación de problemas en ERP Defontana                                 |          |          |          |          |          |          |          |          |          |           |           |           |           |           |           |           |           |
|                                  | Definición de requerimientos funcionales y no funcionales                    |          |          |          |          |          |          |          |          |          |           |           |           |           |           |           |           |           |
| Diseño de la solución            | Historias de usuario y épicos  |          |          |          |          |          |          |          |          |          |           |           |           |           |           |           |           |           |
|                                  | Análisis normativo (Ley 19.628, 21.683, Decreto N°7)                         |          |          |          |          |          |          |          |          |          |           |           |           |           |           |           |           |           |
|                                  | Diseño de arquitectura cliente servidor                                      |          |          |          |          |          |          |          |          |          |           |           |           |           |           |           |           |           |
|                                  | Creación de Carta Gantt parciales y tareas Kanban                            |          |          |          |          |          |          |          |          |          |           |           |           |           |           |           |           |           |
| Implementación de la solución    | Definición de Product Backlog y plan de sprint                               |          |          |          |          |          |          |          |          |          |           |           |           |           |           |           |           |           |
|                                  | Elaboración de diagramas (casos de uso, clases, secuencia, despliegue, BPMN) |          |          |          |          |          |          |          |          |          |           |           |           |           |           |           |           |           |
|                                  | Diseño de Base de datos  |          |          |          |          |          |          |          |          |          |           |           |           |           |           |           |           |           |
|                                  | Diseño de interfaz (mockups en Figma)  |          |          |          |          |          |          |          |          |          |           |           |           |           |           |           |           |           |
| Integración y pruebas finales    | Configuración del entorno de desarrollo (React, Django, MariaDB)             |          |          |          |          |          |          |          |          |          |           |           |           |           |           |           |           |           |
|                                  | Clicado de base de datos y API REST  |          |          |          |          |          |          |          |          |          |           |           |           |           |           |           |           |           |
|                                  | Desarrollo por sprint (Inventario, RRHH, Finanzas, Soporte)                  |          |          |          |          |          |          |          |          |          |           |           |           |           |           |           |           |           |
|                                  | Pruebas unitarias y funcionales por módulo                                   |          |          |          |          |          |          |          |          |          |           |           |           |           |           |           |           |           |
| Despliegue y cierre del proyecto | Control de versiones en GitHub   |          |          |          |          |          |          |          |          |          |           |           |           |           |           |           |           |           |
|                                  | Pruebas de integración (módulos conectados)                                  |          |          |          |          |          |          |          |          |          |           |           |           |           |           |           |           |           |
|                                  | Pruebas de rendimiento, seguridad (OWASP) y trazabilidad                     |          |          |          |          |          |          |          |          |          |           |           |           |           |           |           |           |           |
|                                  | Validación con usuario (cliente ficticio)                                    |          |          |          |          |          |          |          |          |          |           |           |           |           |           |           |           |           |
|                                  | Conexión de logs y optimización de rendimiento                               |          |          |          |          |          |          |          |          |          |           |           |           |           |           |           |           |           |
|                                  | Documentación técnica y manual de usuario                                    |          |          |          |          |          |          |          |          |          |           |           |           |           |           |           |           |           |
|                                  | Presentación final y defensa del proyecto                                    |          |          |          |          |          |          |          |          |          |           |           |           |           |           |           |           |           |
|                                  | Evaluación final del sistema (entorno local)                                 |          |          |          |          |          |          |          |          |          |           |           |           |           |           |           |           |           |
|                                  | Entrega final al docente y contraparte                                       |          |          |          |          |          |          |          |          |          |           |           |           |           |           |           |           |           |
|                                  |  |          |          |          |          |          |          |          |          |          |           |           |           |           |           |           |           |           |

La carta Gantt del proyecto TiCaShop LATAM refleja la planificación completa desde la etapa de análisis hasta la entrega final del sistema, distribuyendo las actividades en un periodo total de 17 semanas.

Durante las primeras tres semanas, el equipo se centró en el análisis y diagnóstico de la empresa, aplicando la técnica de los 5 porqués, identificando problemas en la integración del ERP Defontana y definiendo los requerimientos funcionales, no funcionales y normativos.

En las semanas 4 a 7 se desarrolló la fase de diseño de la solución, donde se definió la arquitectura cliente servidor, se elaboraron diagramas de clases, secuencia y BPMN, además de diseñar las interfaces y planificar el backlog y sprints bajo metodología ágil.

Posteriormente, entre las semanas 8 y 12, se ejecutó la implementación de la solución, enfocándose en la construcción del sistema utilizando React, Django y MariaDB, el desarrollo de

módulos funcionales (Inventario, RRHH, Finanzas y Soporte) y la aplicación de pruebas unitarias y funcionales con control de versiones en GitHub.

Las semanas 13 a 15 correspondieron a la integración y pruebas finales, donde se verificó la interoperabilidad de los módulos, el cumplimiento de las normas de seguridad (OWASP), la trazabilidad y la validación con usuarios simulados, corrigiendo errores y optimizando el rendimiento del sistema.

Finalmente, en las semanas 16 y 17, se realizó la documentación técnica, la presentación y defensa del proyecto, y la entrega final a la contraparte académica, consolidando el trabajo colaborativo y los resultados obtenidos durante todo el proceso.

## Product backlog

| Total Puntos: | 102   |                              |
|---------------|---|------------------------------|
|               |   | Dias                         |
| Id            | Historia de Usuario   | Puntos de Historia Estimados |
| 1             | Como encargado de bodega, quiero registrar automáticamente la entrada de productos con código de barras, para reducir errores en el ingreso de stock. | 20                           |
| 2             | Como administrador, quiero que el sistema calcule automáticamente el costo total de importación, para evitar cálculos manuales y errores.             | 13                           |
| 3             | Como vendedor, quiero generar cotizaciones desde la orden de compra, para agilizar la atención al cliente.  | 8                            |
| 4             | Como empleado, quiero solicitar vacaciones desde el sistema, para evitar trámites manuales y demoras.   | 13                           |
| 5             | Como técnico, quiero recibir tickets de soporte asignados automáticamente, para gestionar incidencias con trazabilidad.                               | 8                            |
| 6             | Como gerente, quiero consultar el stock y ventas desde mi celular, para tomar decisiones rápidas.   | 40                           |

La tabla presenta el Product Backlog del sistema TiCaShop LATAM, compuesto por seis historias de usuario que representan las principales funcionalidades del ERP. Cada historia fue estimada en puntos de historia, totalizando 102 puntos. Estas funcionalidades abarcan los módulos clave del sistema: registro automático de productos mediante código de barras, cálculo del costo total de importación, generación de cotizaciones, solicitud digital de vacaciones, gestión de tickets de soporte y consulta móvil de stock y ventas. Las estimaciones reflejan el nivel de complejidad y prioridad de cada tarea dentro del desarrollo ágil basado en la metodología Scrum.

## Plan de sprint

El plan de sprint define la organización del trabajo ágil durante el desarrollo del sistema TiCaShop, estructurado en tres sprints principales y uno final de integración. Cada sprint tiene un objetivo específico, historias de usuario asociadas y un conjunto de entregables que permiten avanzar de forma incremental en la construcción del sistema.

En el Sprint 1, se desarrolla y prueba el módulo de inventario, incluyendo la funcionalidad de registro mediante código de barras y la visualización móvil del stock.

El Sprint 2 aborda los procesos administrativos y financieros, enfocándose en la automatización del cálculo de costos de importación y la generación de cotizaciones.

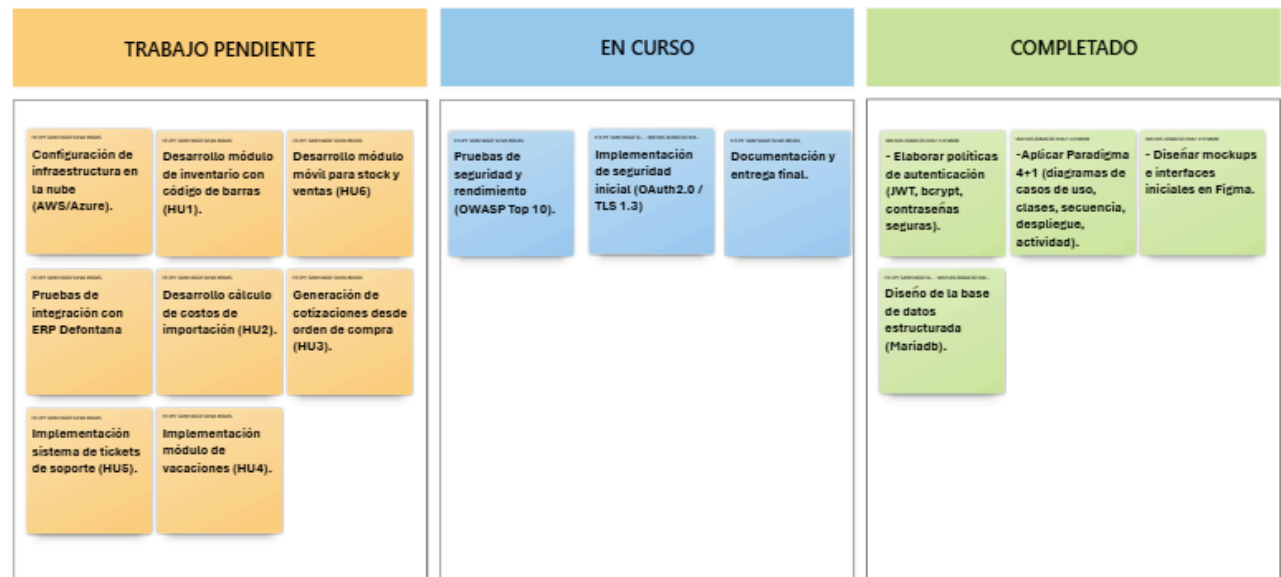
Durante el Sprint 3, se implementan los módulos de Recursos Humanos y Soporte, permitiendo la gestión de vacaciones y la recepción de tickets técnicos.

Finalmente, el Sprint de Integración reúne todos los componentes desarrollados, validando su funcionamiento conjunto y generando la versión final integrada del sistema.

Esta planificación permite mantener un flujo de trabajo iterativo y controlado, facilitando la entrega continua de avances funcionales y la validación temprana con los usuarios.

| Sprint                     | Objetivo   | Historias seleccionadas | Duración  | Entregables                                     |
|----------------------------|--|-------------------------|-----------|---|
| Sprint 1                   | Desarrollar y probar el módulo de inventario           | HU-1, HU-2              | 3 semanas | Módulo inventario y visualización móvil.        |
| Sprint 2                   | Implementar los procesos administrativos y financieros | HU-3, HU-4              | 3 semanas | Módulos de importación y cotización operativos. |
| Sprint 3                   | Automatizar la gestión de RRHH y soporte               | HU-5, HU-6              | 3 semanas | Módulos de vacaciones y tickets.                |
| Sprint Final (integración) | Integrar y probar todos los módulos                    | Todas                   | 2 semanas | Versión final integrada del sistema.            |

# **Tablero kanban (Anexo 3)**



El tablero Kanban muestra la organización y seguimiento del flujo de trabajo del proyecto TiCaShop, dividido en tres columnas: Trabajo Pendiente, En Curso y Completado.

En Trabajo Pendiente se encuentran tareas como la configuración de la infraestructura en la nube, el desarrollo de los módulos de inventario, soporte y las pruebas con el ERP Defontana.

La columna En Curso incluye las actividades activas, entre ellas las pruebas de seguridad y rendimiento, la implementación de autenticación y cifrado (OAuth2.0 / TLS 1.3) y la documentación final.

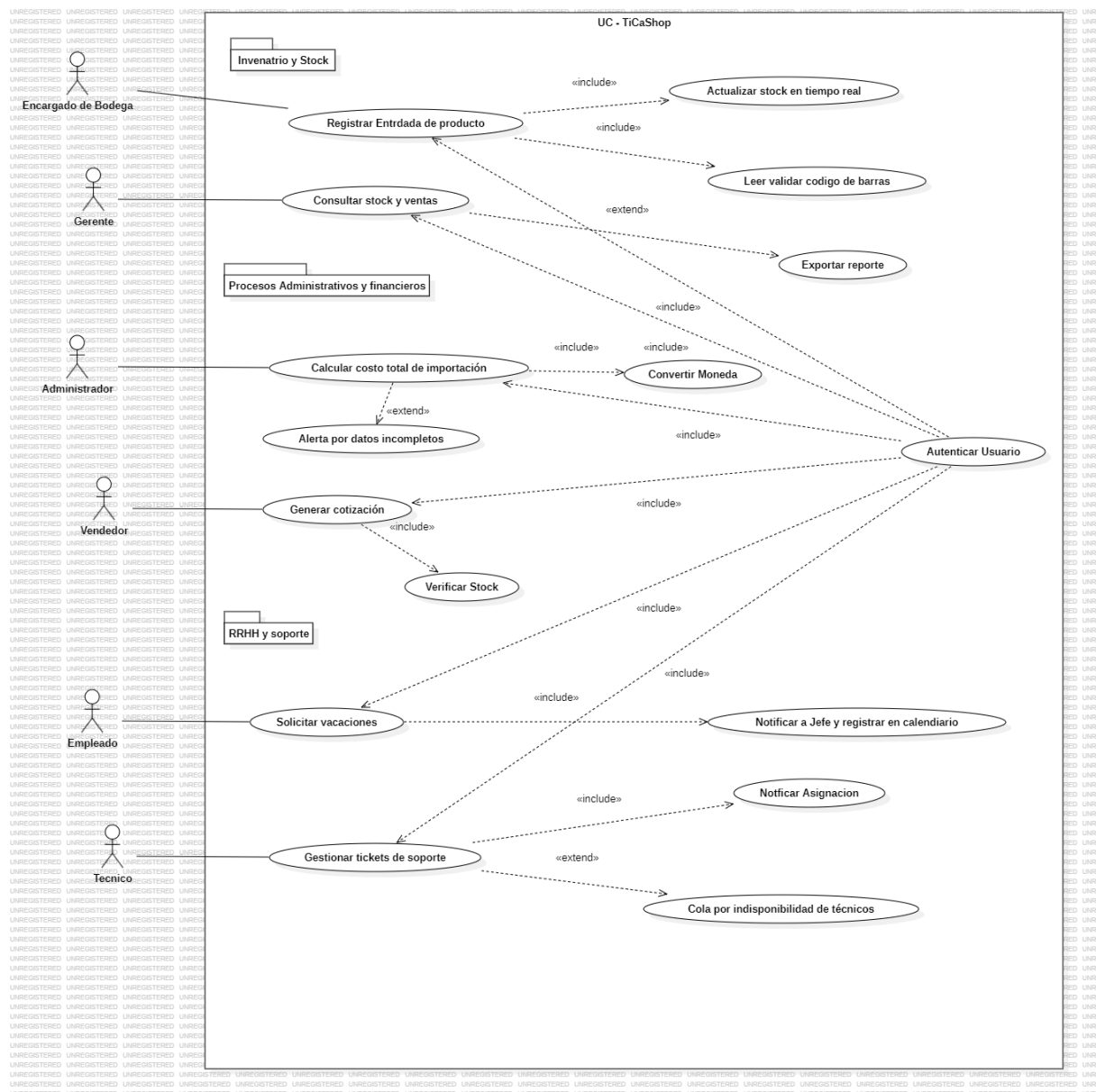
Finalmente, en Completado se registran tareas finalizadas como la aplicación del paradigma 4+1, el diseño de interfaces en Figma y la definición de la base de datos en MariaDB.

Este tablero facilita el control visual del progreso, la colaboración del equipo y la aplicación efectiva de la metodología ágil adoptada en el proyecto.

## Paradigma 4+1 (Anexo 3)

### Vista de escenarios

#### Diagrama de casos de uso

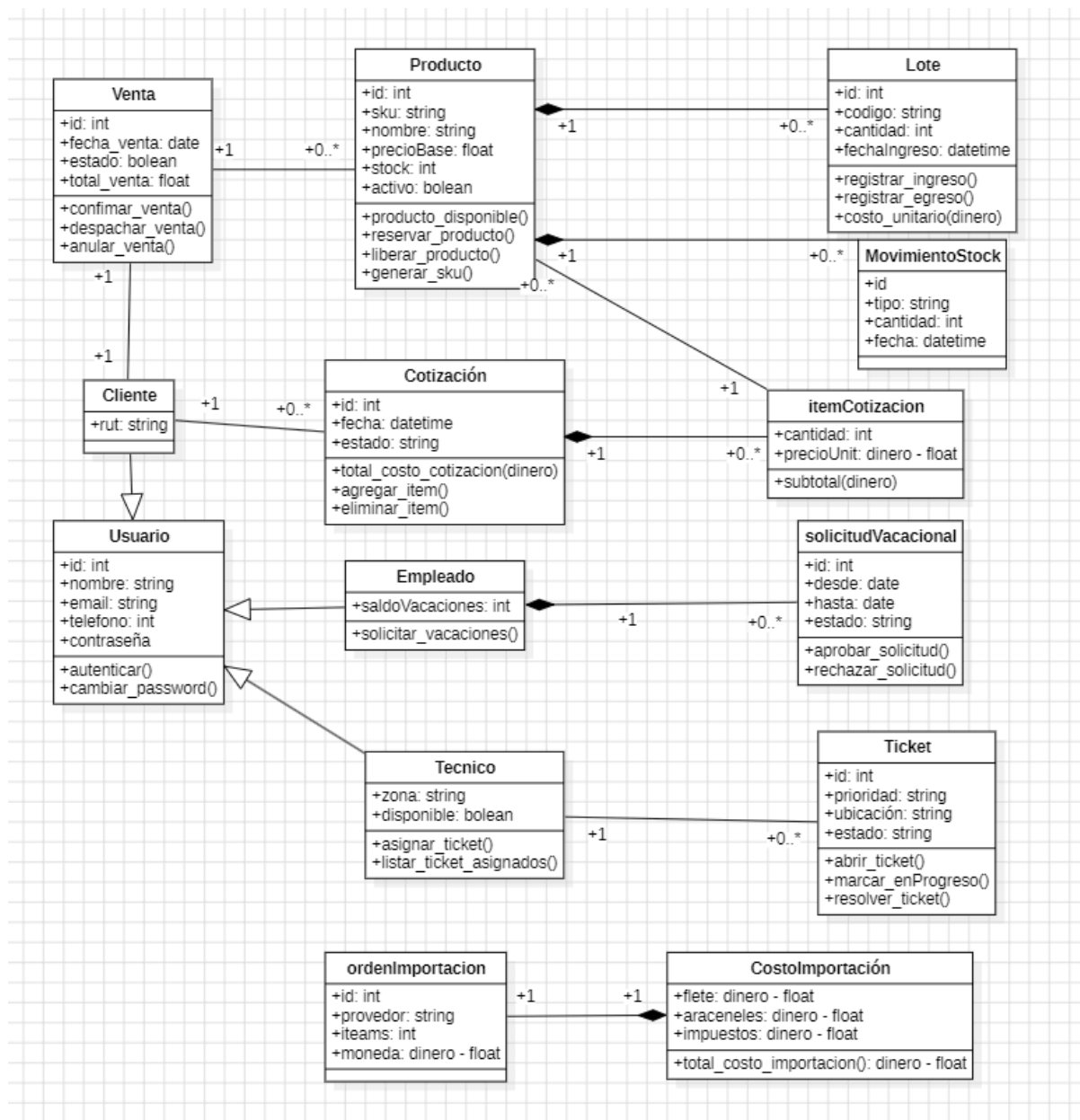


El diagrama muestra la interacción entre los diferentes actores del sistema ERP TiCaShop LATAM y sus principales funciones. Representa los módulos de Inventario, Ventas, Finanzas, Recursos Humanos y Soporte Técnico, evidenciando cómo cada usuario, encargado de bodega, gerente, administrador, vendedor, empleado y técnico, realiza acciones específicas. Además, se incluyen relaciones «include» y «extend» que reflejan procesos obligatorios y opcionales, como la

autenticación de usuarios, la actualización de stock en tiempo real y la gestión de tickets de soporte, asegurando la trazabilidad y seguridad en las operaciones del sistema.

## Vista lógica

### Diagrama de clases



El presente diagrama de clases muestra la estructura general del sistema, que combina los módulos de Recursos Humanos, Cotizaciones, Soporte Técnico e Importaciones.

Cada clase representa una parte importante del funcionamiento del sistema y define sus atributos, métodos principales y relaciones con otras entidades.

## **Gestión de Usuarios y Empleados**

La clase Usuario es la base del sistema y almacena la información esencial de cada persona registrada, como nombre, correo y contraseña.

Desde esta clase se derivan Empleado y Técnico, que heredan sus datos y agregan funciones específicas:

- El Empleado puede solicitar vacaciones y administrar su saldo disponible.
- El técnico gestiona los tickets asignados y actualiza su estado.

Cada empleado puede tener varias Solicitudes Vacacionales, las cuales pueden ser aprobadas o rechazadas mediante los métodos correspondientes.

## **Módulo de Cotizaciones**

El módulo de cotizaciones permite que los Clientes soliciten presupuestos.

Cada Cotización puede incluir varios Productos, lo que se representa con la clase intermedia ItemCotización, donde se detallan la cantidad y el precio unitario.

La clase Producto maneja la información del inventario y cuenta con métodos para revisar disponibilidad y reservar stock.

## **Gestión de Inventario e Importaciones**

El control de inventario se lleva a cabo mediante las clases Producto, Lote y MovimientoStock, que registran ingresos y egresos de productos.

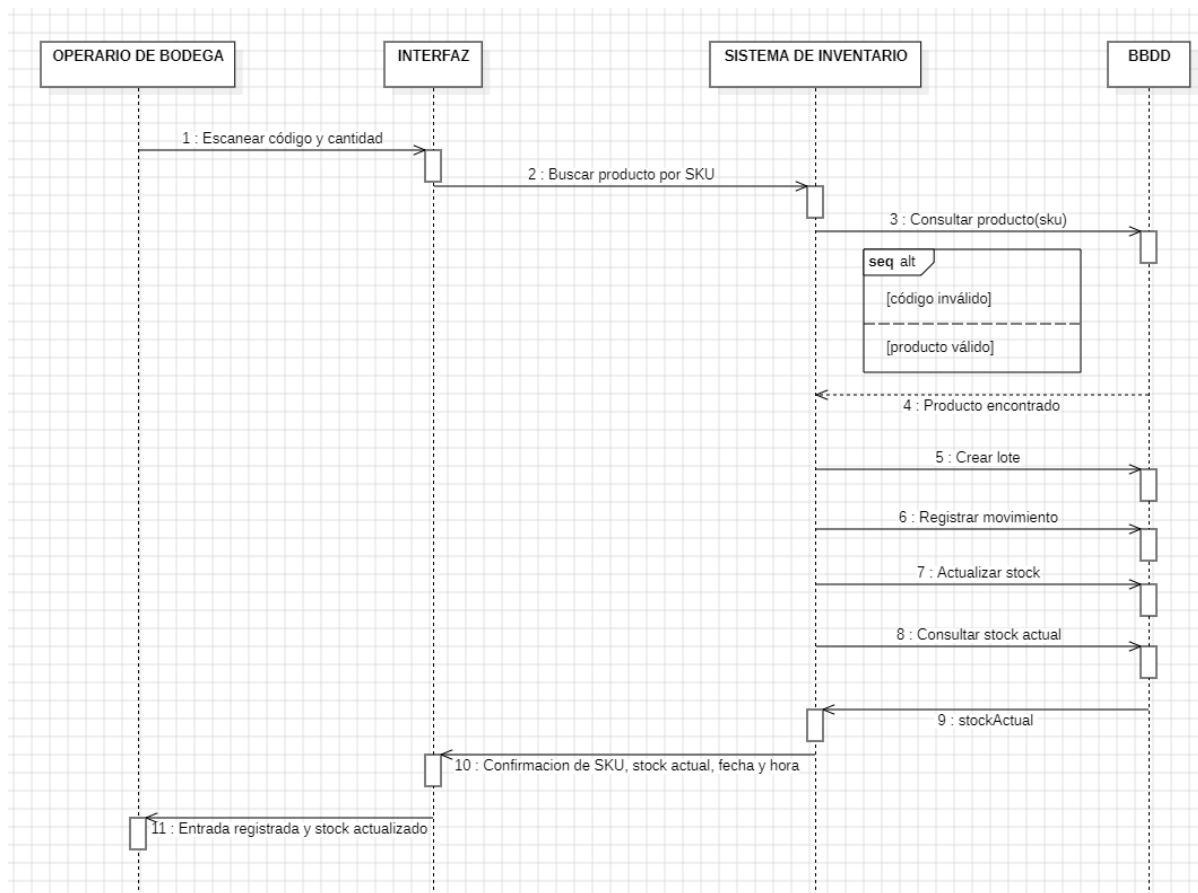
Por otro lado, las clases OrdenImportación y CostoImportación permiten calcular y almacenar los costos asociados a importaciones, como fletes, aranceles e impuestos.

## **Soporte técnico**

El sistema incluye una gestión de Tickets, donde los técnicos pueden recibir, atender y resolver solicitudes.

El diagrama refleja cómo se organizan las distintas áreas del sistema y cómo cada módulo se comunica con los demás. Su estructura modular y el uso de clases reutilizables, como Dinero, facilitan el mantenimiento, la escalabilidad y la comprensión general del modelo.

## Diagrama de secuencia: Registrar automáticamente entradas de productos mediante código de barras

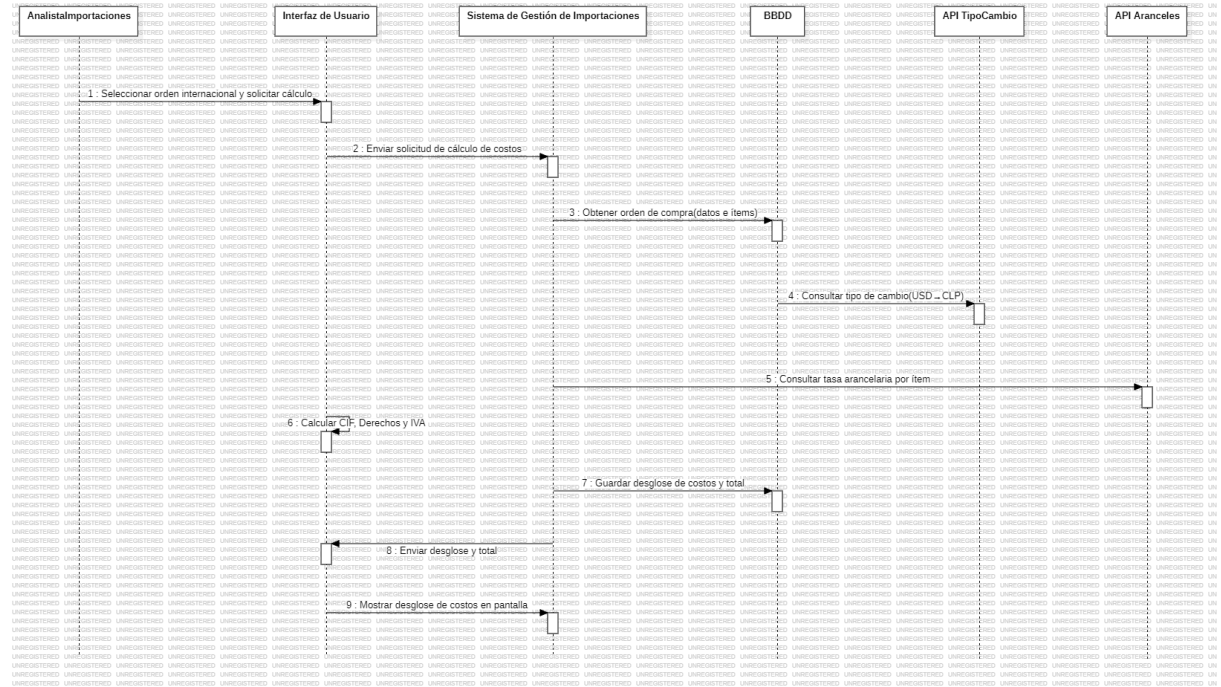


El diagrama muestra cómo el operario de bodega registra un producto escaneando su código de barras

La interfaz del sistema envía la información al módulo de inventario, que busca el producto en la base de datos y valida si ya existe. Si el producto es válido, el sistema crea la entrada, registra el movimiento y actualiza el stock en tiempo real. Finalmente, se confirma el registro al operario, mostrando el SKU, la cantidad y la hora de actualización.

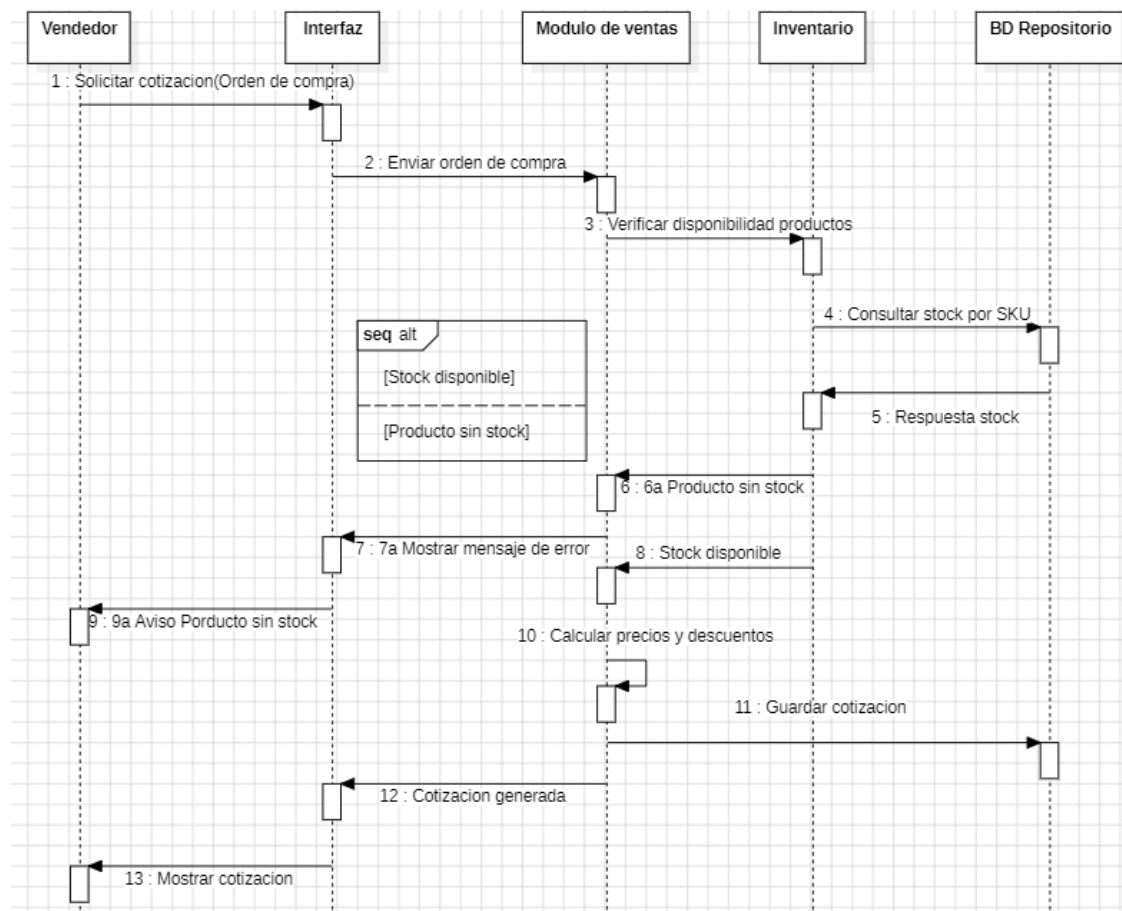
En resumen, el proceso automatiza el control de inventario, evitando errores y asegurando que la información esté siempre al día

## Diagrama de Secuencias: Sistema de Gestión de Importaciones



El diagrama representa la interacción entre el Analista de Importaciones, la Interfaz de Usuario, el Sistema de Gestión de Importaciones, la Base de Datos y las APIs externas (TipoCambio y Aranceles) durante el proceso de cálculo del costo total de importación. El flujo automatiza la obtención de datos, la conversión de moneda y la aplicación de tasas arancelarias, permitiendo generar y mostrar un desglose detallado de costos CIF, derechos e IVA, asegurando precisión, trazabilidad y eficiencia en la gestión de compras internacionales.

## Diagrama de secuencia: Generar cotizaciones a partir de órdenes de compra



El presente diagrama representa la comunicación entre los componentes principales del sistema durante el proceso de generación de una cotización a partir de una orden de compra solicitada por el vendedor.

El flujo comienza cuando el usuario selecciona la opción “Solicitar cotización” en la interfaz desarrollada en React, acción que es procesada por la capa visual y enviada al módulo de ventas a través de la API construida en Django, encargada de gestionar la lógica de negocio.

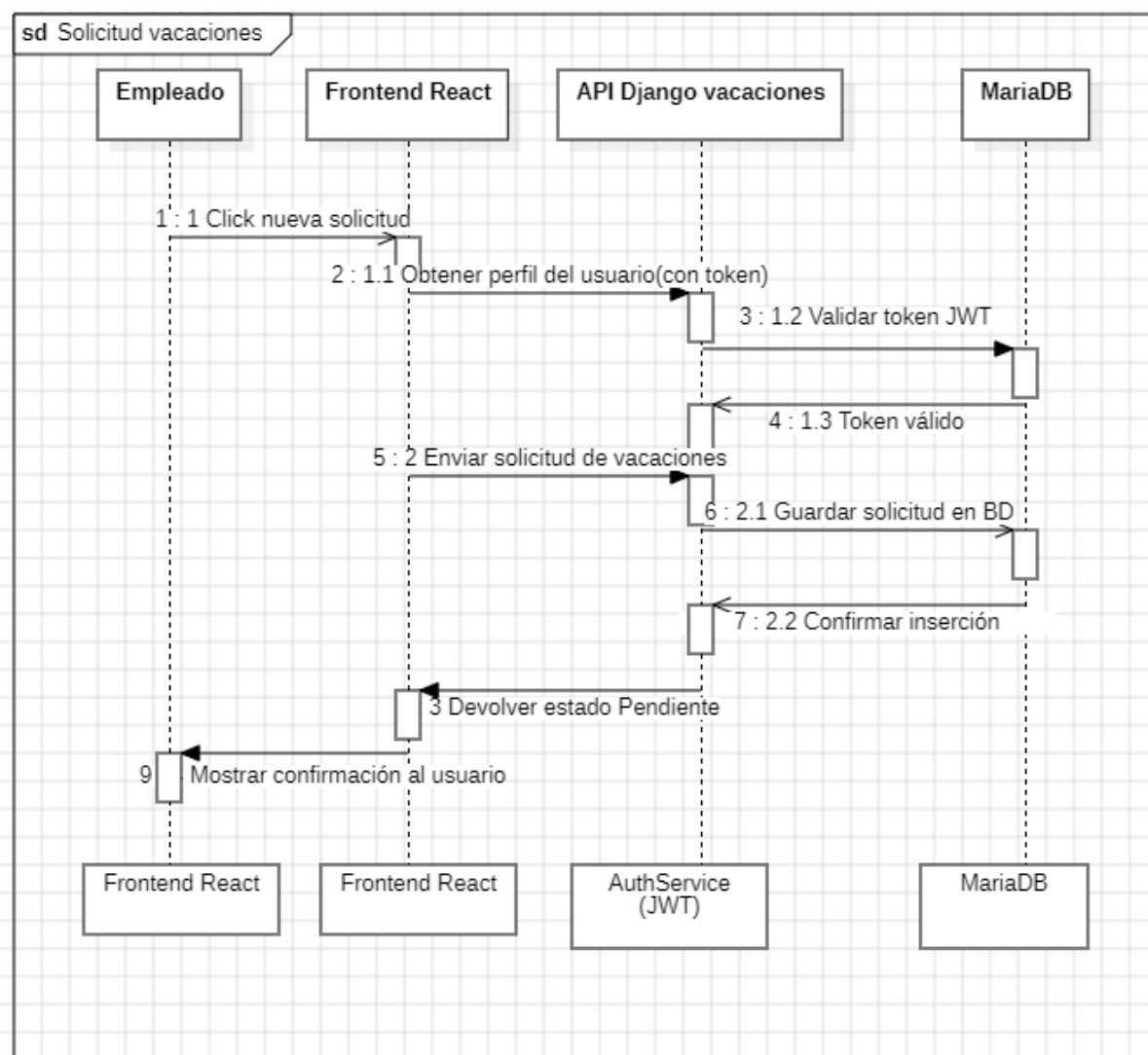
Posteriormente, la API comunica al módulo de inventario la solicitud para verificar la disponibilidad de los productos incluidos en la orden. Este módulo consulta la base de datos MariaDB por medio del código SKU y obtiene una respuesta con el estado actual del stock.

En caso de que los productos no estén disponibles, la API envía una notificación a la interfaz, la cual muestra al vendedor un mensaje de advertencia informando que el producto se encuentra sin stock.

Si los productos están disponibles, el módulo de ventas procede a calcular automáticamente los precios y descuentos correspondientes según las condiciones comerciales registradas. Una vez completado el cálculo, el sistema guarda la cotización en la base de datos, generando un identificador único y almacenando los detalles del pedido.

Finalmente, la API devuelve la respuesta al frontend, donde la interfaz muestra la cotización generada al vendedor, incluyendo la información de precios, productos disponibles y fecha de validez, cerrando así el flujo de manera exitosa.

## Diagrama de comunicación



El presente diagrama representa la comunicación entre los componentes principales del sistema durante el proceso de solicitud de vacaciones por parte de un empleado.

El flujo comienza cuando el usuario selecciona la opción “Nueva Solicitud” en el sistema, acción que es procesada por la interfaz desarrollada en React, la cual se comunica con la API construida en Django para validar la sesión activa y los permisos del usuario.

Posteriormente, la API envía el token al servicio de autenticación AuthService (JWT), que verifica la validez del acceso y responde confirmando la autenticidad del usuario. Una vez validado, la API

registra la solicitud en la base de datos MariaDB, generando un identificador único y marcando el estado inicial como “Pendiente”.

Finalmente, la API devuelve la respuesta al frontend, que muestra al empleado un mensaje de confirmación indicando que su solicitud fue registrada correctamente.

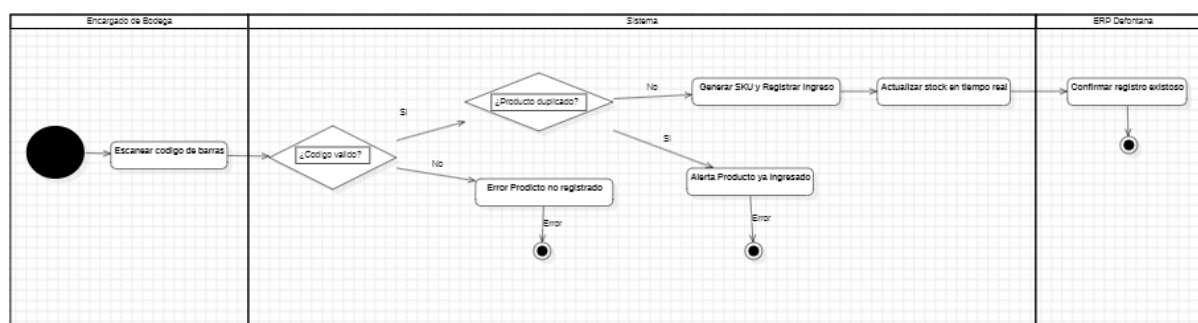
Nota:

Debido a limitaciones de la herramienta (StarUML), el diagrama se elaboró utilizando el formato de diagrama de secuencia, manteniendo la misma lógica de interacción y numeración de mensajes que corresponde a un diagrama de comunicación UML.

## Vista de procesos

### Diagramas de actividades

#### 1. Diagrama de Registro de inventario con código de barras



El presente diagrama representa la secuencia de actividades que se llevan a cabo durante el proceso de registro de productos mediante código de barras en el módulo de inventario.

El flujo comienza cuando el encargado de bodega escanea el código de barras de un producto utilizando la interfaz del sistema. Esta acción inicia una validación automática para comprobar si el código ingresado es válido y existe en la base de datos.

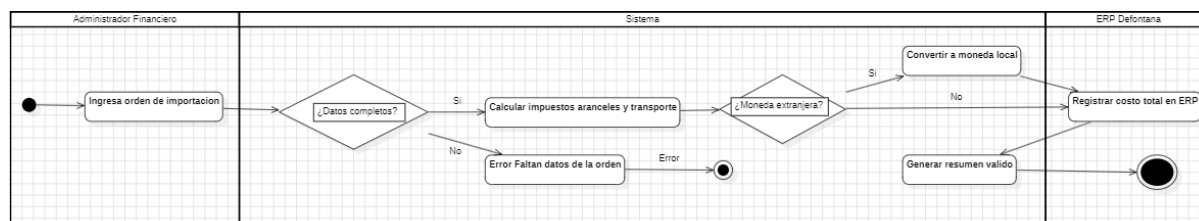
Si el código no es válido, el sistema genera un mensaje de error indicando que el producto no se encuentra registrado, finalizando el proceso. En caso de que el código sea válido, el sistema continúa verificando si el producto ya fue ingresado previamente al inventario.

Si el producto se detecta como duplicado, el sistema emite una alerta de producto ya ingresado y detiene la operación para evitar registros redundantes.

Cuando el producto es nuevo y el código es válido, el sistema procede a generar el SKU correspondiente y registrar el ingreso del producto, actualizando el stock en tiempo real dentro del módulo de inventario. Posteriormente, esta información es enviada a la base de datos principal (ERP Defontana), la cual confirma que el registro ha sido exitosamente almacenado y sincronizado con el sistema central.

Finalmente, el flujo concluye con la confirmación del registro en la base de datos, asegurando la trazabilidad y precisión del inventario, además de reducir errores manuales en la gestión de productos.

## 2. Diagrama de Cálculo automático de costos de importación



El presente diagrama representa la secuencia de actividades que se ejecutan durante el proceso de cálculo automático del costo total de importación en el módulo financiero del sistema.

El flujo comienza cuando el administrador financiero ingresa una nueva orden de importación a través de la interfaz del sistema. Esta acción activa una validación automática que verifica si todos los datos necesarios se encuentran correctamente ingresados.

Si el sistema detecta que faltan valores obligatorios, como costos de transporte, impuestos o aranceles, se genera un mensaje de error indicando que la información de la orden está incompleta, deteniendo el proceso hasta que sea corregido.

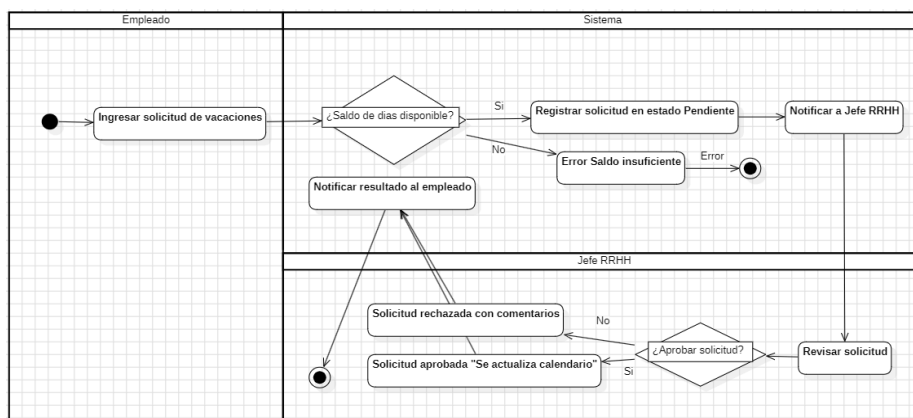
En caso de que la información esté completa, el sistema continúa con el flujo y procede a calcular automáticamente los impuestos, aranceles y gastos de transporte asociados a la importación.

Posteriormente, el sistema valida si los montos se encuentran expresados en moneda extranjera. Si es así, realiza la conversión automática a moneda local (CLP) utilizando los tipos de cambio configurados en el sistema.

Cuando la orden ya está en moneda nacional, se omite este paso y se genera directamente un resumen validado con el total del cálculo.

Finalmente, el sistema registra el costo total de la importación en el ERP Defontana, asegurando la trazabilidad de la operación, la coherencia de los datos contables y la precisión en los registros financieros de TiCaShop.

### 3. Diagrama de Solicitud de vacaciones por sistema



El presente diagrama representa la secuencia de actividades que se llevan a cabo durante el proceso de solicitud de vacaciones por parte de un empleado dentro del sistema TiCaShop.

El flujo comienza cuando el empleado ingresa una solicitud de vacaciones mediante la interfaz del sistema. Una vez enviada, el sistema verifica automáticamente si el trabajador posee días disponibles en su saldo de vacaciones.

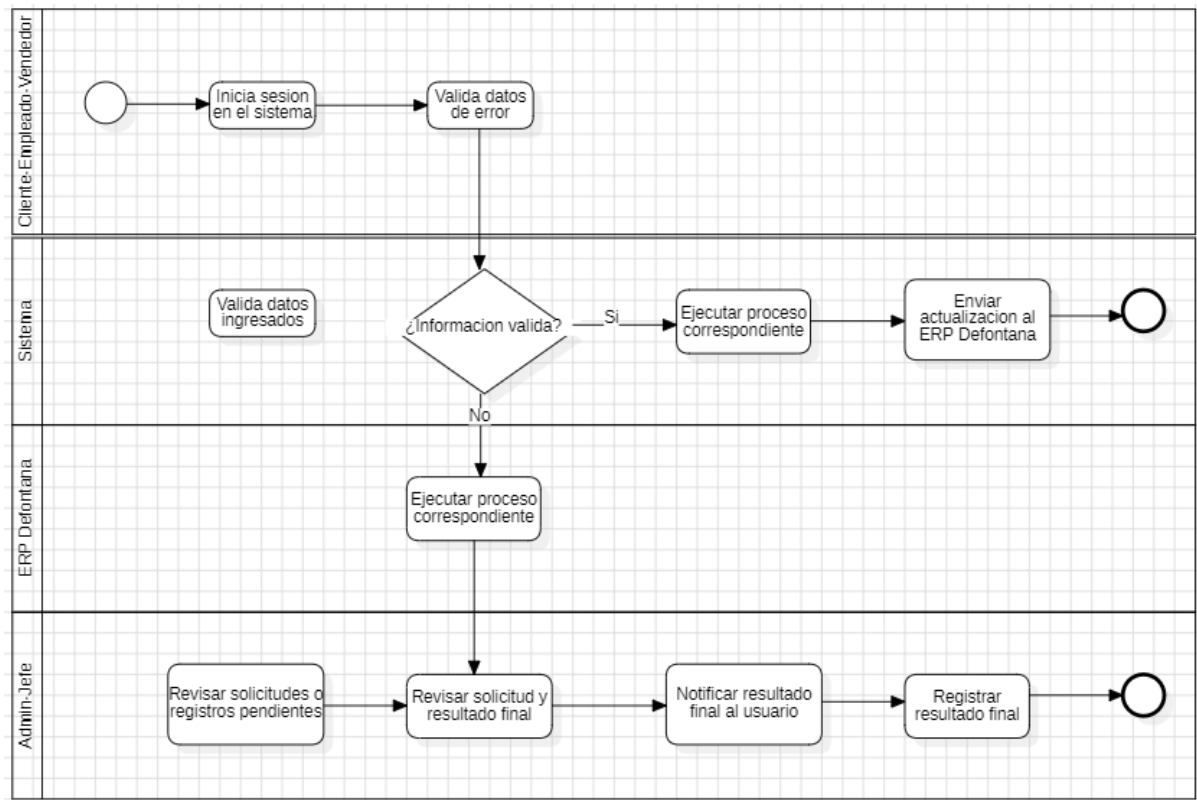
Si el saldo es insuficiente, el sistema genera un mensaje de error informando al empleado que no cuenta con los días necesarios, finalizando el proceso. En caso de que sí existan días disponibles, la solicitud se registra con el estado “Pendiente” y el sistema notifica al jefe de RRHH para su revisión.

El jefe de RRHH recibe la notificación y procede a revisar la solicitud. Luego, decide si aprueba o rechaza la petición. Si la solicitud es rechazada, el sistema registra el resultado con los comentarios correspondientes y notifica al empleado.

Si la solicitud es aprobada, el sistema actualiza automáticamente el calendario de vacaciones, reflejando los días aprobados y enviando una notificación de confirmación al empleado.

Finalmente, el flujo concluye con la notificación del resultado al solicitante, asegurando la trazabilidad del proceso, la transparencia en las decisiones y la correcta gestión de los registros de vacaciones en el sistema.

### Diagrama de BPMN



El presente diagrama representa el flujo general de interacción entre los distintos actores del sistema TiCaShop, mostrando cómo se comunican los módulos internos y los sistemas externos durante la ejecución de un proceso típico dentro de la plataforma.

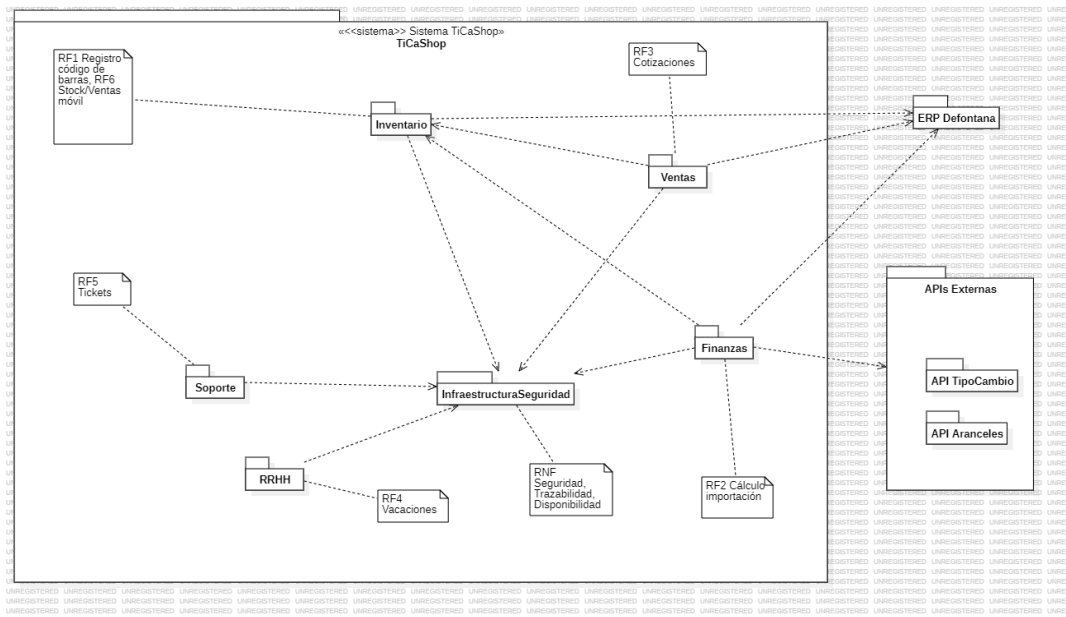
El flujo comienza cuando el cliente, empleado o vendedor inicia sesión en el sistema y selecciona una acción a realizar. El sistema valida los datos ingresados, comprobando que la información sea correcta y cumpla con los requisitos establecidos. En caso de detectar errores, se genera un mensaje de validación que informa al usuario y detiene el proceso. Si los datos son válidos, el sistema ejecuta el proceso correspondiente (como registro de inventario, cotización, importación o solicitud de vacaciones) y envía la información al ERP Defontana para su sincronización contable y operativa.

Posteriormente, el ERP Defontana procesa la información recibida y actualiza sus registros internos, devolviendo una confirmación al sistema TiCaShop. A continuación, el administrador o jefe de área revisa las solicitudes o registros pendientes, valida los resultados y autoriza su aprobación final.

Finalmente, el sistema notifica el resultado al usuario y registra la trazabilidad del proceso, garantizando la coherencia de los datos, la seguridad de la información y la correcta integración entre TiCaShop y los sistemas externos.

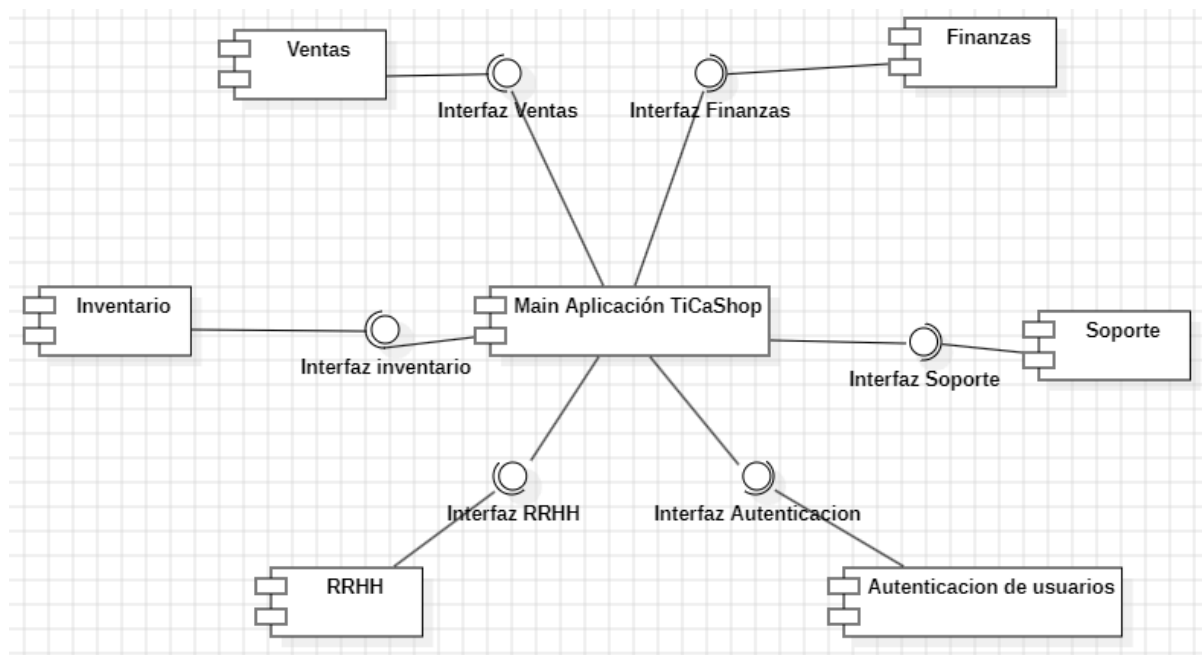
Vista de desarrollo

Diagrama de paquetes



El diagrama muestra la estructura modular del ERP TiCaShop, con los paquetes Inventario, Ventas, Finanzas, RRHH, Soporte e Infraestructura y Seguridad, junto a sus interacciones internas y externas. Se observa la integración con ERP Defontana y las APIs TipoCambio y Aranceles, que soportan el cálculo financiero y la gestión de importaciones. Esta arquitectura modular asegura seguridad, trazabilidad y disponibilidad en todos los procesos del sistema.

## Diagrama de componentes



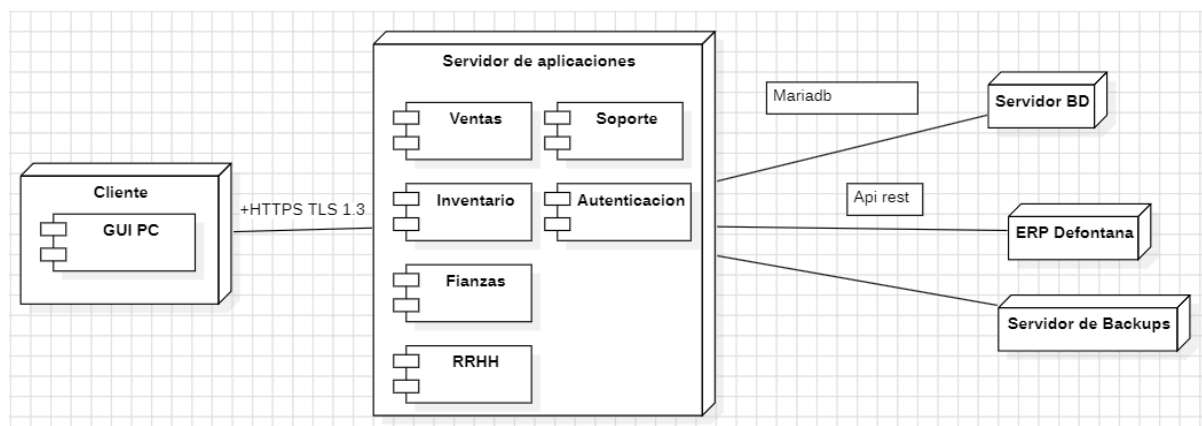
El diagrama muestra cómo la aplicación principal TiCaShop se conecta con los distintos módulos del sistema a través de sus interfaces. Cada módulo (Ventas, Finanzas, Inventario, RRHH, Soporte y Autenticación de usuarios) cumple una función específica y se comunica con la aplicación central mediante su propia interfaz.

De esta forma, el sistema está dividido en partes independientes pero conectadas, lo que facilita el mantenimiento, la seguridad y las futuras actualizaciones.

Cada componente puede mejorar o ampliarse sin afectar a los demás, manteniendo el sistema modular, ordenado y escalable.

## Vista física

### Diagrama de despliegue



El presente diagrama de despliegue representa la vista física del sistema TiCaShop, mostrando cómo se distribuyen los componentes de software y hardware dentro de la arquitectura cliente-servidor.

El flujo comienza con el cliente, que accede a la aplicación a través de la interfaz gráfica (GUI PC). La comunicación entre el cliente y el sistema se realiza mediante el protocolo HTTPS con cifrado TLS 1.3, asegurando la transmisión segura de datos entre el usuario y el servidor.

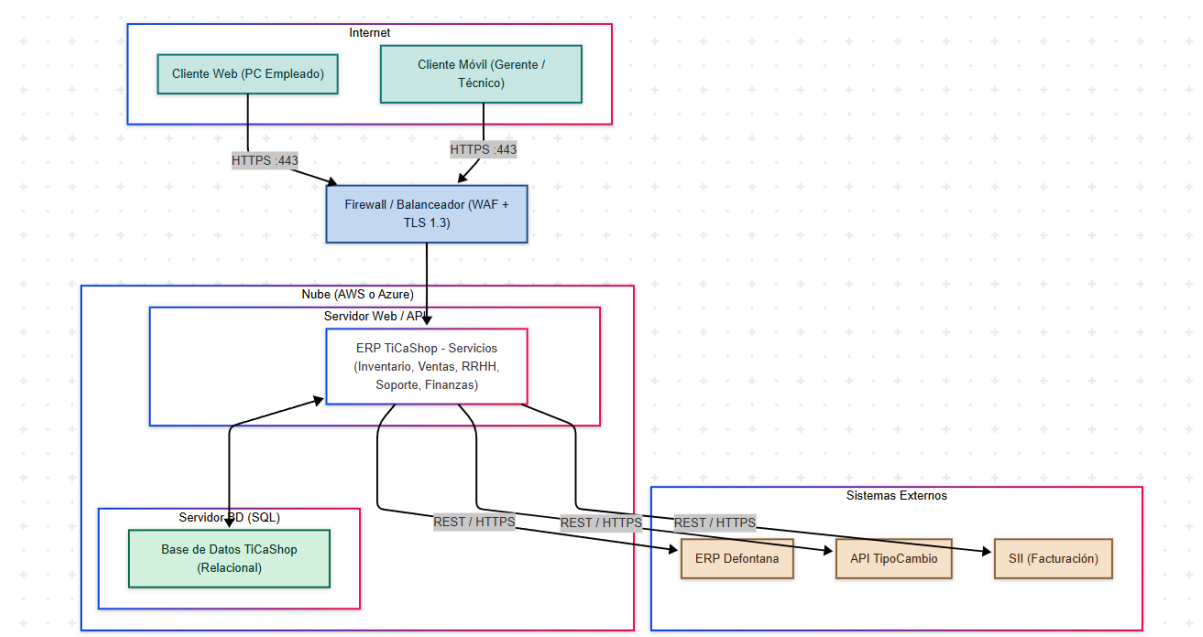
En el servidor de aplicaciones se encuentran implementados los principales módulos del sistema: Ventas, Inventario, Finanzas, Recursos Humanos, Soporte y Autenticación. Cada uno de ellos cumple funciones específicas dentro de la solución ERP de TiCaShop, y se comunican entre sí a través de la API desarrollada en Django, garantizando modularidad y mantenimiento independiente.

El servidor de aplicaciones se conecta con diferentes servicios externos:

- El Servidor de Base de Datos (MariaDB), encargado de almacenar de forma estructurada la información de usuarios, productos, cotizaciones, importaciones y registros de RRHH.
- El ERP Defontana, con el cual el sistema intercambia información contable y de inventario mediante servicios REST seguros, manteniendo la coherencia de los datos.
- El Servidor de Backups, responsable de realizar copias de seguridad automáticas de la base de datos y de los archivos del sistema, asegurando la continuidad operativa ante fallos o contingencias.

De esta forma, el diagrama refleja una infraestructura distribuida y segura, donde cada componente cumple una función específica dentro de un entorno conectado y escalable. Esta estructura garantiza la disponibilidad, trazabilidad y protección de la información, pilares fundamentales del proyecto TiCaShop.

## Diagrama de topología



El sistema adopta una arquitectura cliente–servidor en nube (AWS/Azure). Los usuarios acceden por HTTPS a través de un firewall/balanceador (WAF + TLS 1.3) que protege y distribuye las solicitudes hacia el Servidor Web/API, donde se ejecutan los módulos del ERP (Inventario, Ventas, RRHH, Soporte y Finanzas). Este se comunica con un Servidor de Base de Datos SQL y con sistemas externos como ERP Defontana, API TipoCambio y SII, mediante servicios REST seguros, garantizando disponibilidad, escalabilidad y seguridad en las operaciones

## Especificación de tecnologías y herramientas requeridas para la infraestructura

El sistema se desarrollará bajo una arquitectura cliente servidor, ejecutada completamente en entorno local, con el objetivo de construir un prototipo funcional que permita demostrar las principales características del proyecto. Se seleccionaron herramientas y tecnologías que facilitan la integración, el mantenimiento y la escalabilidad futura del sistema.

### Frontend

Para la interfaz de usuario se utilizará React.js, que es una biblioteca muy popular de Javascript que permite crear aplicaciones dinámicas e interactivas mediante componentes

reutilizables.

El desarrollo visual se realizará con HTML5 y CSS, lo que permitirá tener control total sobre el diseño sin depender de frameworks externos

El objetivo es ofrecer una interfaz clara, moderna y fácil de navegar, en la que cada usuario pueda acceder únicamente a las funciones correspondientes a su rol (Administrador, RRHH, Nómina, etc.)

## **Backend**

El servidor se construirá con Django, un framework de Python que facilita el desarrollo estructurado y seguro de aplicaciones web. Además, se utilizará Django REST Framework para crear la API que gestionará la comunicación entre el frontend y la base de datos, enviando y recibiendo la información de manera eficiente.

Para proteger los datos sensibles de los usuarios, las contraseñas serán encriptadas con la librería bcrypt, evitando que se almacenen en texto plano.

## **Base de datos**

La información del sistema se almacenará en MariaDB, una base de datos relacional que destaca por su rendimiento, estabilidad y compatibilidad con MySQL. En ella se guardarán los datos estructurados del proyecto, como usuarios, roles, vacaciones, liquidaciones y registros de asistencia.

Su integración con Django, a través del ORM (Object Relational Mapper), permite manipular los datos desde el código de forma más sencilla y segura.

## **Autenticación y seguridad**

La autenticación podrá implementarse mediante sesiones o tokens JWT, según se defina en la etapa de desarrollo.

Las contraseñas estarán protegidas con bcrypt, y se aplicará un modelo de roles y permisos que restrinja las acciones de cada usuario según su perfil.

Asimismo, se mantendrán las credenciales y claves del sistema en un archivo de entorno (.env), con el fin de evitar exponer información sensible dentro del proyecto.

## **Herramientas de apoyo**

Durante el desarrollo se emplearán distintas herramientas que facilitan la organización y el control del trabajo:

- Visual Studio Code: entorno principal de programación, con extensiones para Python y React.
- GitHub: repositorio para el control de versiones, seguimiento de cambios y respaldo del código.

- Figma: diseño de la interfaz gráfica y creación de mockups interactivos, que servirán como guía visual para el desarrollo.

### **Infraestructura local**

El sistema se ejecutará completamente de manera local, sin despliegue en la nube.

El backend se levantará en un entorno Python con Django, la base de datos funcionará en un servidor local de MariaDB, y el frontend se ejecutará en React mediante el comando npm start.

La comunicación entre ambos se realizará a través de HTTP en localhost, utilizando distintos puertos.

## **Políticas de autenticación y autorización**

Éstas políticas aplican a todo el sistema (Front end, back end Django + DB MARIADB) y regulan cómo los usuarios acceden y que pueden hacer según su rol.

### **1. Autenticación**

- Método: A través de credenciales como el email/usuario más la contraseña
- Almacenamiento de contraseña: A través de bcrypt
- Protecciones de inicio de sesión: Bloqueo temporal de 15 minutos
- Validación básica anti bot: Captcha simple o pregunta tipo control
- Recuperación de contraseña: Enlace de un solo uso con espiración corta (15 minutos)
- Variables sensibles: Se cargan desde .env (clave JWT, credenciales DB)

### **2. Autorización**

Modelo RBAC ( Role Based Access Control), donde cada usuario pertenece a un rol y el backend valida permisos en cada endpoint

Roles del sistema:

- Administrador
- RRHH
- Nómina (payroll)
- Jefatura
- Supervisor comercial
- Técnico de soporte

- Empleado general

| Módulo                         | Admin | RRHH | Nómina | Jefatura | Sup. Comercial | Técnico | General |
|--------------------------------|-------|------|--------|----------|----------------|---------|---------|
| Vacaciones - Ver global        | ✓     | ✓    |        | ✓        |                |         |         |
| Vacaciones - Aprobar/Rechazar  | ✓     | ✓    |        | ✓        |                |         |         |
| Vacaciones - Solicitar         | ✓     | ✓    |        | ✓        |                | ✓       | ✓       |
| Liquidaciones - Ver global     | ✓     | ✓    | ✓      |          |                |         |         |
| Liquidaciones - Firmar/generar | ✓     | ✓    | ✓      |          |                |         |         |
| Comisiones - Calcular/validar  | ✓     |      |        | ✓        | ✓              |         |         |
| Asistencia - Ver global        | ✓     | ✓    |        | ✓        |                |         |         |
| Asistencia - Ver propia        | ✓     | ✓    |        | ✓        | ✓              | ✓       | ✓       |

|                     |   |   |   |   |   |  |  |
|---------------------|---|---|---|---|---|--|--|
| Gestión de usuarios | ✓ |   |   |   |   |  |  |
| Exportar reportes   | ✓ | ✓ | ✓ | ✓ | ✓ |  |  |

## Gestión de tokens y sesiones

Cuando un usuario inicia sesión, el sistema genera dos tipos de tokens que sirven para mantenerlo autenticado de forma segura:

### 1. Access Token (Token de acceso):

Es el que se usa en cada solicitud al sistema (por ejemplo, cuando el usuario entra al módulo de vacaciones o liquidaciones)

- Tiene una duración corta, de unos 30 minutos, para reducir riesgos si alguien lo llega a robar
- Cada vez que el usuario realiza una acción, el frontend envía este token al backend (Django) para verificar su identidad.

### 2. Refresh Token (token de actualización):

Es un token de respaldo que dura más tiempo (por ejemplo, 7 días) y permite obtener un nuevo access token sin volver a iniciar sesión

- Cuando el token de acceso vence, el sistema usa el refresh para generar uno nuevo automáticamente
- Cada vez que se usa, se crea uno nuevo y el anterior se invalida (rotación).
- Si un usuario cierra sesión, el refresh también se invalida (lista negra), para que nadie más lo pueda usar.

### 3. Cierre de sesión:

Al cerrar sesión, ambos tokens se eliminan y se impide seguir accediendo al sistema. En algunos casos se invalida solo el refresh token del dispositivo actual, para mantener la sesión cerrada de manera segura.

- **Elevación de privilegios:**

En acciones más sensibles (como firmar liquidaciones o aprobar solicitudes masivas), el sistema puede pedir que el usuario vuelva a autenticarse, aunque

ya tenga sesión iniciada  
Esto agrega una capa extra de seguridad.

### Datos mínimos que contiene un JWT (claims)

Cada token JWT almacena algunos datos básicos (no confidenciales) que permiten identificar al usuario y controlar su acceso dentro del sistema:

| Campo  | Significado  |
|--------|--|
| Sub    | Identificador del usuario en el sistema                                    |
| Role   | Rol o perfil del usuario (por ejemplo: Administrador, RRHH, Técnico, etc.) |
| Scopes | Lista de permisos o acciones que el usuario puede ejecutar                 |
| iat    | Fecha y hora en que se generó el token (issued at).                        |
| exp    | Fecha y hora en que el token expira (expiration)                           |

### Reglas de contraseña y cuenta

- Contraseña: Mínimo 10 caracteres, mayúscula, minúscula, número y símbolo
- Cambio forzado si hay sospecha de compromiso.
- Bloqueo tras 5 intentos fallidos (desbloqueo automático a los 15 minutos)

### Responsabilidades por rol

- Administrador: gestión de usuarios/roles, auditoría, configuración general.
- RRHH: vacaciones, asistencia y liquidaciones (operación)
- Nómina: cálculo y firma de liquidaciones (solo módulo de liquidaciones)
- Jefatura: aprobación de vacaciones y vista de asistencia de su equipo.
- Supervisor Comercial: módulo de comisiones (calcular/validar/exportar)
- Técnico / General: solicitar vacaciones y consultar su información personal (liquidaciones/ asistencia)

## Amenazas comunes y medidas de mitigación

Durante el análisis del sistema se identificaron amenazas comunes asociadas a la operación de aplicaciones web y bases de datos en entornos en la nube. Las principales son:

| Amenaza                           | Descripción  | Medida de mitigación   |
|-----------------------------------|--|--|
| Inyección SQL                     | Manipulación de consultas a la base de datos mediante entrada no validada. | Uso de ORM seguro (Django ORM o Sequelize), validación de datos y uso de prepared statements.                    |
| Cross-Site Scripting (XSS)        | Inserción de scripts maliciosos en formularios o URLs.                     | Escapado de contenido en el frontend, sanitización de entradas y uso de políticas CSP (Content Security Policy). |
| Acceso no autorizado              | Uso indebido de credenciales o robo de sesiones.                           | Autenticación mediante JWT/OAuth2, encriptación de contraseñas (bcrypt), políticas de roles y permisos.          |
| Exposición de datos sensibles     | Filtración de información personal o financiera.                           | Cifrado TLS 1.3 en tránsito, cifrado AES-256 en reposo, y control de acceso basado en roles.                     |
| Denegación de servicio (DoS/DDoS) | Saturación del servidor con peticiones falsas.                             | Uso de firewall WAF, balanceador de carga y límites de tasa (rate limiting).                                     |
| Pérdida de disponibilidad         | Fallas del servidor o pérdida de datos.                                    | RespalDOS automáticos diarios, replicación en la nube y monitoreo de uptime.                                     |

## Estándares de codificación segura

El desarrollo del sistema TiCaShop LATAM sigue las buenas prácticas de seguridad recomendadas por OWASP Secure Coding Guidelines, incorporando los siguientes estándares para garantizar la integridad, confidencialidad y disponibilidad de la información:

- Validación de entradas:  
Todas las entradas del usuario son validadas y sanitizadas antes de ser procesadas por el sistema, previniendo ataques de inyección SQL, *Cross-Site Scripting (XSS)* y manipulación de datos. Se emplean formularios seguros, *prepared statements* y filtros de contenido.
- Autenticación y autorización:  
El acceso al sistema se gestiona mediante autenticación con tokens (JWT) y control de roles por niveles. Cada usuario tiene permisos específicos según su perfil (administrador, técnico, empleado, etc.), aplicando el principio de mínimo privilegio en todas las conexiones y operaciones críticas.
- Gestión de errores:  
El sistema maneja las excepciones de forma controlada, evitando exponer información sensible como rutas del servidor, consultas SQL o variables internas. Los errores se registran en logs seguros para auditoría y diagnóstico sin comprometer la seguridad.
- Cifrado y protección de datos:  
Las contraseñas de los usuarios se almacenan mediante hash seguro (bcrypt) y la comunicación entre cliente y servidor se cifra mediante TLS 1.3 (HTTPS). Además, la base de datos utiliza cifrado en reposo para proteger información confidencial.
- Separación de entornos:  
Se mantienen entornos diferenciados de desarrollo, pruebas y producción para evitar fugas de datos o configuraciones inseguras durante el ciclo de vida del software.
- Versionado seguro:  
El código fuente se gestiona en GitHub con control de acceso restringido y revisión por pares (*code review*) para prevenir la introducción de vulnerabilidades.

## Herramientas de Pruebas: Automatizadas y manuales

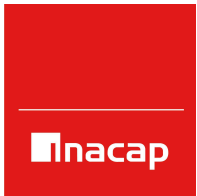
Se contemplaron tanto **herramientas automatizadas** como **manuales** para cubrir el ciclo de calidad:

| Tipo                 | Herramientas contempladas  |
|----------------------|--|
| <b>Automatizadas</b> | <i>SonarQube</i> (análisis estático de código), <i>Postman</i> (pruebas API), <i>JMeter</i> (rendimiento), <i>OWASP ZAP</i> (seguridad). |
| <b>Manuales</b>      | Revisión de código por pares ( <i>code review</i> ), validación de flujos de usuario, y pruebas exploratorias en entorno controlado.     |

### **Interfaz de usuario (Anexo 4 )**

Con el objetivo de representar visualmente la interfaz y flujo funcional del sistema propuesto, se elaboró un mock-up interactivo que refleja los principales módulos y procesos desarrollados durante el proyecto. Este diseño permite visualizar la experiencia del usuario, la disposición de los elementos y la coherencia entre las diferentes vistas, sirviendo como base para la futura implementación del sistema real.

A continuación, se presentan las imágenes correspondientes al mock-up elaborado:



## Interfaz de gestión de vacaciones vista por “Administrador”

Dashboard RRHH  
Recursos Humanos

Admin

Vacaciones

Liquidaciones

Comisiones

Asistencia

Cambiar Rol (Demo)

Administrador

Admin Sistema  
admin@empresa.cl

Gestión de Vacaciones  
Administra las solicitudes de vacaciones del personal

ExportarFirmar digitalmente

+ Nueva

| Empleado        | Fecha Inicio | Fecha Fin  | Días | Estado    | Acciones   |
|-----------------|--------------|------------|------|-----------|--|
| María García    | 2025-10-15   | 2025-10-25 | 10   | Pendiente | Ver <span>✓ Aprobar</span> <span>✗ Rechazar</span> |
| Juan Pérez      | 2025-11-01   | 2025-11-10 | 9    | Aprobada  | Ver  |
| Ana Martínez    | 2025-10-20   | 2025-10-27 | 7    | Rechazada | Ver  |
| Carlos López    | 2025-12-15   | 2025-12-29 | 14   | Pendiente | Ver <span>✓ Aprobar</span> <span>✗ Rechazar</span> |
| Laura Rodríguez | 2025-11-15   | 2025-11-22 | 7    | Aprobada  | Ver  |

## Interfaz de liquidaciones de sueldo vista por “Administrador”

Dashboard RRHH  
Recursos Humanos

Admin

Vacaciones

Liquidaciones

Comisiones

Asistencia

Cambiar Rol (Demo)

Administrador

Admin Sistema  
admin@empresa.cl

Liquidaciones de Sueldo  
Gestión de liquidaciones mensuales del personal

ExportarFirmar digitalmente

+ Generar Liquidación

| Empleado        | Periodo      | Monto Total | Estado          | Acciones          |
|-----------------|--------------|-------------|-----------------|-------------------|
| María García    | Octubre 2025 | \$1.250.000 | Firmada         | Ver Descargar PDF |
| Juan Pérez      | Octubre 2025 | \$1.450.000 | Pendiente Firma | Ver Descargar PDF |
| Ana Martínez    | Octubre 2025 | \$1.180.000 | Firmada         | Ver Descargar PDF |
| Carlos López    | Octubre 2025 | \$1.350.000 | Pendiente Firma | Ver Descargar PDF |
| Laura Rodríguez | Octubre 2025 | \$1.280.000 | Firmada         | Ver Descargar PDF |

### Interfaz de comisión de ventas vista por “Administrador”

Dashboard RRHH  
Recursos Humanos

Admin

Vacaciones

Liquidaciones

Comisiones

Asistencia

Cambiar Rol (Demo)

Administrador

Admin Sistema  
admin@empresa.cl

Comisiones de Ventas

Cálculo y seguimiento de comisiones del equipo comercial

Exportar

Calcular Comisiones

+ Nueva

| Empleado        | Período      | Ventas Totales | Comisión  | Estado    | Acciones    |
|-----------------|--------------|----------------|-----------|-----------|-------------|
| Juan Pérez      | Octubre 2025 | \$45.000.000   | \$675.000 | Calculada | Ver Detalle |
| Ana Martínez    | Octubre 2025 | \$38.500.000   | \$577.500 | Pendiente | Ver Detalle |
| Carlos López    | Octubre 2025 | \$52.000.000   | \$780.000 | Calculada | Ver Detalle |
| Laura Rodríguez | Octubre 2025 | \$41.200.000   | \$618.000 | Pendiente | Ver Detalle |
| Pedro González  | Octubre 2025 | \$48.000.000   | \$720.000 | Calculada | Ver Detalle |

### Interfaz de asistencias vista por “Administrador”

Dashboard RRHH  
Recursos Humanos

Admin

Vacaciones

Liquidaciones

Comisiones

Asistencia

Cambiar Rol (Demo)

Administrador

Admin Sistema  
admin@empresa.cl

Control de Asistencia

Reporte generado desde Talana

Export

Filtros:

Desde: 28/09/2025

Hasta: 29/09/2025

Empleado: Todos

| Empleado        | Fecha      | Entrada | Salida | Horas | Estado   |
|-----------------|------------|---------|--------|-------|----------|
| María García    | 2025-09-29 | 08:45   | 18:30  | 9:45  | Presente |
| María García    | 2025-09-28 | 08:50   | 18:25  | 9:35  | Presente |
| Juan Pérez      | 2025-09-29 | 09:15   | 18:45  | 9:30  | Tardanza |
| Juan Pérez      | 2025-09-28 | 08:30   | 18:20  | 9:50  | Presente |
| Ana Martínez    | 2025-09-29 | -       | -      | -     | Ausente  |
| Ana Martínez    | 2025-09-28 | 08:40   | 18:30  | 9:50  | Presente |
| Carlos López    | 2025-09-29 | 08:35   | 18:15  | 9:40  | Presente |
| Laura Rodríguez | 2025-09-29 | 08:55   | 18:35  | 9:40  | Presente |

Total Registros

8

Presentes

6

Ausencias

1

### Interfaz de gestión de usuarios vista por “Administrador”

Dashboard RRHH

Recursos Humanos

Admin

Vacaciones

Liquidaciones

Comisiones

Asistencia

Usuarios

Cambiar Rol (Demo)

Administrador

AD

Admin. Sistema

admin@empresa.cl

Gestión de Usuarios

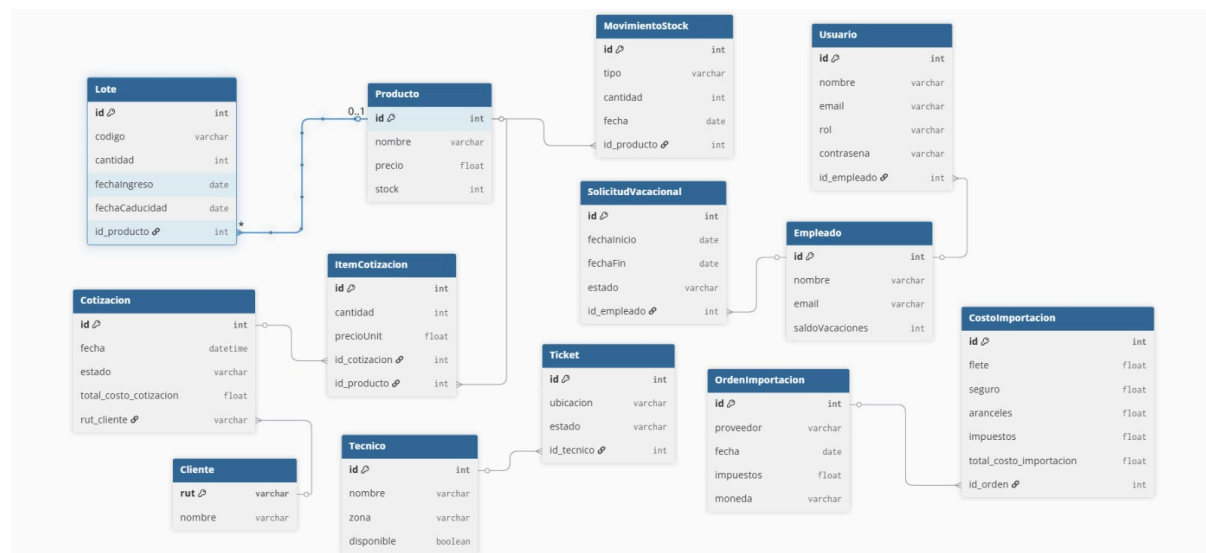
Administra los usuarios y sus roles en el sistema

Exportar

Nuevo Usuario

| Nombre          | Email                      | Rol        | Departamento     | Estado | Acciones   |
|-----------------|----------------------------|------------|------------------|--------|--|
| María García    | maria.garcia@empresa.cl    | Técnico    | Operaciones      | Activo | <div>Ver</div> <div>Editar</div> <div>Eliminar</div> |
| Juan Pérez      | juan.perez@empresa.cl      | General    | Ventas           | Activo | <div>Ver</div> <div>Editar</div> <div>Eliminar</div> |
| Ana Martínez    | ana.martinez@empresa.cl    | RRHH       | Recursos Humanos | Activo | <div>Ver</div> <div>Editar</div> <div>Eliminar</div> |
| Carlos López    | carlos.lopez@empresa.cl    | Supervisor | Ventas           | Activo | <div>Ver</div> <div>Editar</div> <div>Eliminar</div> |
| Laura Rodríguez | laura.rodriguez@empresa.cl | Jefatura   | Operaciones      | Activo | <div>Ver</div> <div>Editar</div> <div>Eliminar</div> |
| Pedro González  | pedro.gonzalez@empresa.cl  | Nómina     | Finanzas         | Activo | <div>Ver</div> <div>Editar</div> <div>Eliminar</div> |

## Diseño de Base de Datos (*Anexo 3*)



El diagrama muestra la estructura relacional de la base de datos del sistema ERP TiCaShop LATAM, organizada por módulos de Inventario, Ventas, Finanzas, Recursos Humanos y Soporte Técnico. Las entidades se vinculan mediante claves primarias y foráneas, garantizando integridad referencial, trazabilidad y consistencia de datos en todos los procesos del sistema.

## Roles y Perfiles de Usuario

El sistema completo contempla distintos módulos funcionales como Inventario, Importaciones, Ventas y Recursos Humanos; sin embargo, esta entrega se centra exclusivamente en el módulo de Recursos Humanos (RR.HH.), ya que fue la parte asignada al equipo. Por este motivo, el mockup interactivo, las pruebas y la interfaz funcional fueron desarrollados en torno a los procesos de vacaciones, liquidaciones, asistencia y tickets internos. No obstante, se definieron todos los roles del sistema para mantener la coherencia del modelo de seguridad (RBAC) y la trazabilidad de permisos entre módulos.

### Perfiles que interactúan con el sistema

- Usuarios no registrados: Sólo pueden acceder a la pantalla de inicio de sesión o recuperación de contraseña
- Clientes externos: ingresan mediante login al portal de clientes para revisar y aceptar cotizaciones emitidas
- Usuarios internos (empleados y staff): acceden al sistema según su rol asignado, aplicando el principio de mínimo privilegio
- Administrador del sistema: gestiona usuarios, roles, seguridad y auditorías globales

Mecanismos de seguridad: autenticación mediante usuario/contraseña (encriptado con *bcrypt*), sesiones con JWT (access 30 min / refresh 7 días) y control de permisos con RBAC (Role-Based Access Control)

### Roles definidos en el sistema

| Rol                      | Descripción y privilegios principales   |
|--------------------------|---|
| Administrador            | Tiene acceso total. Puede crear y modificar usuarios, asignar roles, gestionar auditorías y configurar parámetros del sistema     |
| Encargado de Bodega      | Gestiona productos, lotes y movimientos de stock. Controla entradas, salidas y ajustes de inventario                              |
| Administrador Financiero | Supervisa costos de importación, órdenes de compra e indicadores económicos. Puede consultar cotizaciones y reportes financieros. |

|                               |   |
|-------------------------------|---|
| <b>Vendedor</b>               | Crea y edita cotizaciones, agrega productos y envía presupuestos a clientes. Puede consultar stock, pero no modificarlo.                |
| <b>Empleado (RRHH)</b>        | Solicita vacaciones, visualiza sus liquidaciones y asistencia. Puede abrir tickets de soporte interno                                   |
| <b>Técnico de Soporte</b>     | Atiende y gestiona tickets asignados. También puede solicitar vacaciones y ver su información personal                                  |
| <b>Nómina (Payroll)</b>       | Genera, revisa y firma liquidaciones. Exporta reportes de remuneraciones. Tiene acceso restringido a datos sensibles.                   |
| <b>Supervisor Comercial</b>   | Calcula y valida comisiones del equipo comercial. Supervisa metas y resultados de ventas  |
| <b>RRHH (staff)</b>           | Revisa y aprueba solicitudes de vacaciones, consulta asistencia global y verifica liquidaciones. No interviene en inventario ni ventas. |
| <b>Cliente (externo)</b>      | Accede al portal de clientes para ver, descargar o aceptar cotizaciones emitidas. Sin acceso a módulos internos                         |
| <b>Todos los roles (demo)</b> | Rol técnico de pruebas utilizado para validar las vistas del sistema y simular permisos. No se utiliza en producción                    |

## Roles que intervienen en el módulo de RR.HH. (alcance del mockup)

Los siguientes roles participan directamente en los procesos simulados en el prototipo interactivo:

- **Administrador:** puede visualizar y modificar cualquier información del módulo
- **RRHH (staff):** aprueba o rechaza solicitudes de vacaciones y revisa asistencia general.
- **Nómina:** genera y firma liquidaciones
- **Empleado:** solicita vacaciones, visualiza liquidaciones y asistencia personal
- **Técnico de soporte:** solicita vacaciones y gestiona tickets asignados

- **Jefatura:** aprueba vacaciones de su equipo directo

## Estándares de Programación Segura con roles y perfiles de usuario

La implementación de estándares de programación segura en TiCaShop debe considerar tanto las normativas vigentes en Chile (Ley N.º 19.628 de protección de datos personales, Ley Marco de Ciberseguridad N.º 21.663 y Decreto N.º 7/2023), como las buenas prácticas internacionales (ISO/IEC 27001, 27002, 27005, 29100 y OWASP Top 10).

El objetivo es garantizar la confidencialidad, integridad y disponibilidad de la información, estableciendo medidas diferenciadas según el rol y el nivel de acceso de cada usuario.

La siguiente tabla resume los estándares y controles propuestos para cada perfil de usuario del sistema, asegurando que todos los procesos cumplan con el principio de privilegio mínimo, trazabilidad de acciones y un marco de ciberseguridad alineado a la industria:

| Rol / Perfil             | Estándares y Controles de Seguridad  |
|--------------------------|--|
| Administrador            | - Autenticación multifactor (MFA). - Uso de OAuth2.0/JWT y contraseñas robustas rotadas periódicamente. - Registro y auditoría de todas las acciones críticas. - Principio de privilegio mínimo (RBAC avanzado).                   |
| Encargado de bodega      | - Autenticación con credenciales cifradas para acceso al inventario. - Validación en tiempo real para evitar registros duplicados. - Restricción de acceso únicamente a inventario y trazabilidad de stock.                        |
| Administrador financiero | - Cifrado de datos en tránsito (TLS 1.3) y en reposo (AES-256). - Validación automática de cálculos con logs seguros. - Control de accesos basado en roles (RBAC). - Cumplimiento con ISO/IEC 27001 y 27005 en gestión de riesgos. |

|                             |   |
|-----------------------------|---|
| <b>Vendedor</b>             | - Acceso limitado solo a funciones de cotización. - Enmascaramiento de datos sensibles de clientes en reportes. - Sesiones con expiración automática tras inactividad. - Prevención de vulnerabilidades según OWASP Top 10.   |
| <b>Empleado (RRHH)</b>      | - Acceso restringido únicamente a gestión de vacaciones propias. - Datos personales cifrados y visibles solo a RRHH autorizado. - Registro con trazabilidad de solicitudes y decisiones.  |
| <b>Técnico de soporte</b>   | - Acceso solo a tickets asignados, con visibilidad mínima de datos de cliente. - Registro de cambios de estado en logs inalterables. - Acceso granular según jerarquía o nivel.   |
| <b>Nomina</b>               | - Acceso restringido únicamente a información de remuneraciones y contratos laborales - Autenticación multifactor (MFA) y validación de identidad antes de cada sesión - Cifrado de datos en tránsito (TLS 1.3) y en reposo (AES-256). - Registro de modificaciones y accesos en logs inalterables con trazabilidad completa. - Cumplimiento con Ley 19.628 y políticas internas de confidencialidad laboral. |
| <b>Supervisor comercial</b> | - Acceso autorizado solo a reportes de ventas, desempeño y stock del área. - Validación automática de integridad en datos financieros provenientes del ERP. - Sesiones con expiración por inactividad y control de accesos por rol. - Registro de revisiones, aprobaciones y alertas ante actividad inusual. - Aplicación del principio de privilegio mínimo y trazabilidad de acciones.                      |
| <b>Todos los roles</b>      | - Validación de entradas y salidas (seguridad contra inyecciones SQL, XSS). - Principio de privilegio mínimo. - Monitoreo y auditoría continua. - Pruebas de seguridad periódicas (pentesting, escaneo de vulnerabilidades). - Copias de seguridad automáticas en la nube. - Capacitación en seguridad digital.   |

## **Plan de pruebas (Ver Anexo 5)**

El presente plan de pruebas se elaboró siguiendo las buenas prácticas de la industria del software y las recomendaciones de seguridad establecidas por OWASP Top 10, con el objetivo de validar la confiabilidad, robustez y protección del sistema TiCaShop LATAM antes de su despliegue en producción.

Se contemplaron pruebas funcionales, de integración, seguridad, rendimiento y usabilidad, abarcando todos los módulos críticos del ERP (Inventario, Ventas, RRHH, Soporte y Finanzas).

## **Tipos de pruebas a realizar**

### **1. Pruebas funcionales**

Validan que cada componente individual cumple correctamente con los requerimientos funcionales definidos.

- Verificar registro y actualización de productos.
- Validar cálculo automático del costo de importación.
- Confirmar generación de cotizaciones y control de stock.
- Comprobar solicitudes de vacaciones y asignación de tickets de soporte.  
Resultado esperado: Cada módulo ejecuta correctamente las operaciones para las cuales fue diseñado.

### **2. Pruebas de integración**

Evalúan la correcta comunicación entre los módulos del sistema y con sistemas externos.

- Flujo de ventas integrado con el inventario (descuento automático de stock).
- Conexión entre ERP TiCaShop y ERP Defontana (sincronización contable).
- Comunicación con API TipoCambio y SII mediante servicios REST seguros.  
Resultado esperado: Las interacciones entre módulos se ejecutan sin pérdida de datos ni conflictos de lógica.

### **3. Pruebas de seguridad**

Aplicadas para detectar vulnerabilidades y validar mecanismos de protección contra amenazas comunes.

- Inyección SQL: Validar consultas parametrizadas y uso de ORM seguro.
- XSS (Cross-Site Scripting): Comprobar sanitización de entradas en formularios y campos de texto.
- CSRF (Cross-Site Request Forgery): Confirmar uso de tokens CSRF en peticiones sensibles.
- Control de accesos: Revisar autenticación JWT/OAuth2 y gestión de roles.  
Herramientas: OWASP ZAP, SonarQube, revisiones de código manual.  
Resultado esperado: El sistema resiste ataques básicos sin exposición de datos o degradación del servicio.

#### 4. Pruebas de rendimiento

Permiten medir el desempeño del sistema bajo diferentes cargas de usuarios y transacciones.

- Simulación de 50–100 usuarios concurrentes accediendo al módulo de ventas.
- Evaluación del tiempo de respuesta promedio (< 2 segundos por consulta).
- Medición del uso de CPU, memoria y latencia del servidor de base de datos.  
Herramientas: Apache JMeter, Postman (colecciones automatizadas).  
Resultado esperado: El sistema mantiene un rendimiento óptimo sin afectar la disponibilidad.

#### 5. Pruebas de usabilidad

Evalúan la experiencia del usuario final, asegurando una interfaz intuitiva y fácil de usar.

- Validar la consistencia visual y navegación clara en el panel principal.
- Verificar textos, botones y formularios con terminología clara y accesible.
- Realizar pruebas con usuarios reales (empleados y técnicos).  
Resultado esperado: La solución es comprensible, eficiente y satisface las necesidades del usuario final.

### Recomendaciones y marco de referencia OWASP

El plan de pruebas se sustenta en el marco **OWASP Top 10 (2021)**, enfocándose principalmente en las siguientes categorías de seguridad:

- A01:2021 – Control de acceso roto.
- A03:2021 – Inyección.
- A05:2021 – Configuración de seguridad incorrecta.
- A07:2021 – Identificación y autenticación.
- A08:2021 – Fallos de integridad y deserialización insegura.
- A09:2021 – Registro y monitoreo insuficiente.

Estas recomendaciones permiten reforzar la seguridad desde el desarrollo hasta la fase de despliegue, asegurando que TiCaShop cumpla con estándares internacionales de calidad y protección de datos.

El plan de pruebas garantiza que TiCaShop LATAM sea un sistema seguro, estable y confiable, validando tanto el comportamiento funcional como los aspectos críticos de seguridad y rendimiento bajo un enfoque de calidad total.

## Conclusión

El desarrollo del sistema TICASHOP permitió aplicar de forma práctica los principios de la Ingeniería de Software, utilizando herramientas modernas de programación, diseño UML y metodologías ágiles de gestión. La implementación de un ERP modular permitió integrar en una sola solución los procesos de inventario, ventas, recursos humanos, soporte técnico y finanzas, logrando una operación más eficiente, trazable y segura.

El enfoque en la seguridad de la información y las buenas prácticas OWASP garantizan la protección de los datos, evitando vulnerabilidades comunes como inyección SQL, XSS o accesos no autorizados. Asimismo, la definición de un plan de pruebas funcionales, integrales, de seguridad y rendimiento permitió validar la estabilidad y calidad del sistema antes de su despliegue.

El proyecto demuestra la viabilidad de una solución tecnológica escalable, capaz de integrarse con sistemas externos (como ERP Defontana, SII y APIs de tipo de cambio) y de adaptarse a futuras mejoras. Con ello, TICASHOP se posiciona como una empresa más digital, segura y competitiva, alineada con los desafíos actuales de transformación tecnológica.

## Ev3

El módulo de Recursos Humanos del sistema TiCaShop LATAM fue desarrollado con el objetivo de modernizar y automatizar los procesos internos relacionados con la gestión de personal, tales como vacaciones, asistencia, liquidaciones y administración de usuarios. La implementación se construyó siguiendo lineamientos de ingeniería de software, metodologías ágiles y buenas prácticas de desarrollo seguro, integrando herramientas como Django, motor de base de datos relacional y librerías especializadas para generación de reportes.

Para asegurar que el sistema respondiera adecuadamente a las necesidades reales del negocio, se realizó un trabajo conjunto entre diseño, backend y pruebas funcionales. Se implementó un modelo de datos optimizado, una interfaz consistente y validaciones que permiten mantener la integridad y seguridad de la información. Además, se ejecutó un plan de pruebas que evaluó cada flujo crítico del sistema, verificando comportamiento, rendimiento, restricciones por rol y manejo correcto de errores.

El presente documento reúne la descripción de la implementación, evidencias de funcionamiento, validación de resultados, lineamientos de diseño aplicados y el análisis del proceso de trabajo en equipo, dejando una base sólida para futuras mejoras y escalabilidad del sistema.

## Implementación de software

### Alineación entre la interfaz y las necesidades del negocio

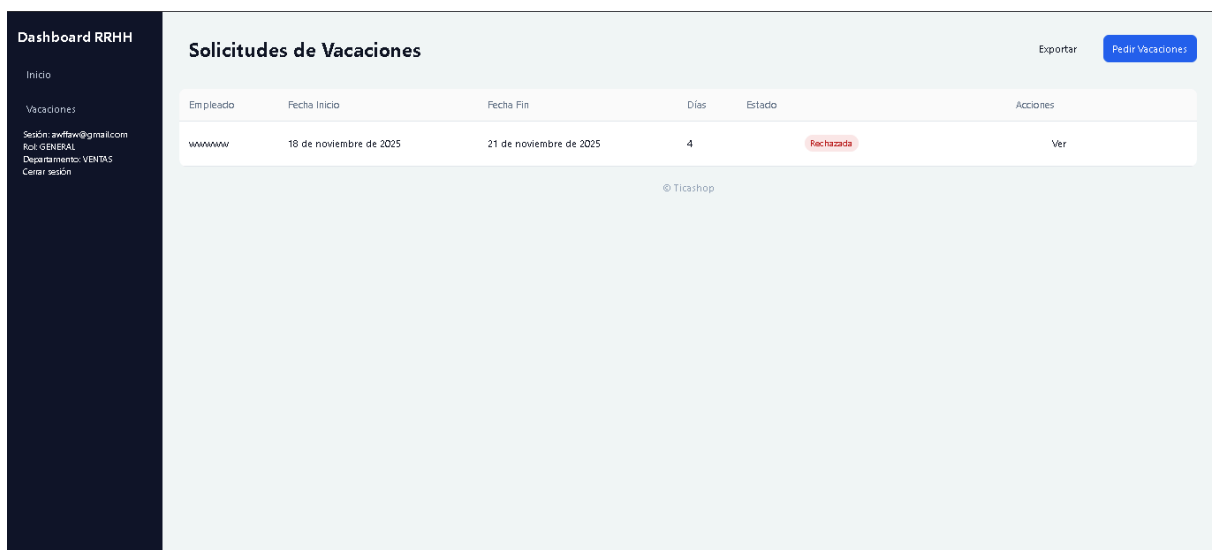
| Necesidad del negocio       | Funcionalidad implementada                                | Pantalla / Vista                            | Estado |
|-----------------------------|---|---|--------|
| Solicitar vacaciones        | Formulario de solicitud con validación y envío automático | Vista “Solicitar Vacaciones (Empleado)”     | 100%   |
| Aprobar/rechazar vacaciones | Panel de revisión y gestión                               | Vista “Vacaciones (RRHH / Jefatura)”        | 100%   |
| Ver liquidaciones           | Acceso seguro y lectura de PDF/registro                   | Vista “Liquidaciones (Empleado / Nómina)”   | 100%   |
| Administrar liquidaciones   | Generar, firmar y publicar liquidaciones                  | Vista “Gestión de Liquidaciones (Nómina)”   | 100%   |
| Consultar asistencia        | Panel con filtros por fecha/usuario                       | Vista “Asistencia (Admin / RRHH)”           | 100%   |
| Gestión de usuarios         | Crear, editar y asignar roles                             | Vista “Gestión de Usuarios (Administrador)” | 100%   |

|              |                                     |                       |      |
|--------------|-------------------------------------|-----------------------|------|
| Trazabilidad | Notificaciones + registro histórico | Módulo RRHH y backend | 100% |
|--------------|-------------------------------------|-----------------------|------|

## Capturas de pantalla del sistema/mocks/código funcionando

Ponlas así:

- **Figura 1: Solicitud de Vacaciones – Vista Vendedor/General**



**Dashboard RRHH**

Inicio

Vacaciones

Sesión: wwwwww@gmail.com  
Rol: GENERAL  
Departamento: VENTAS  
Cerrar sesión

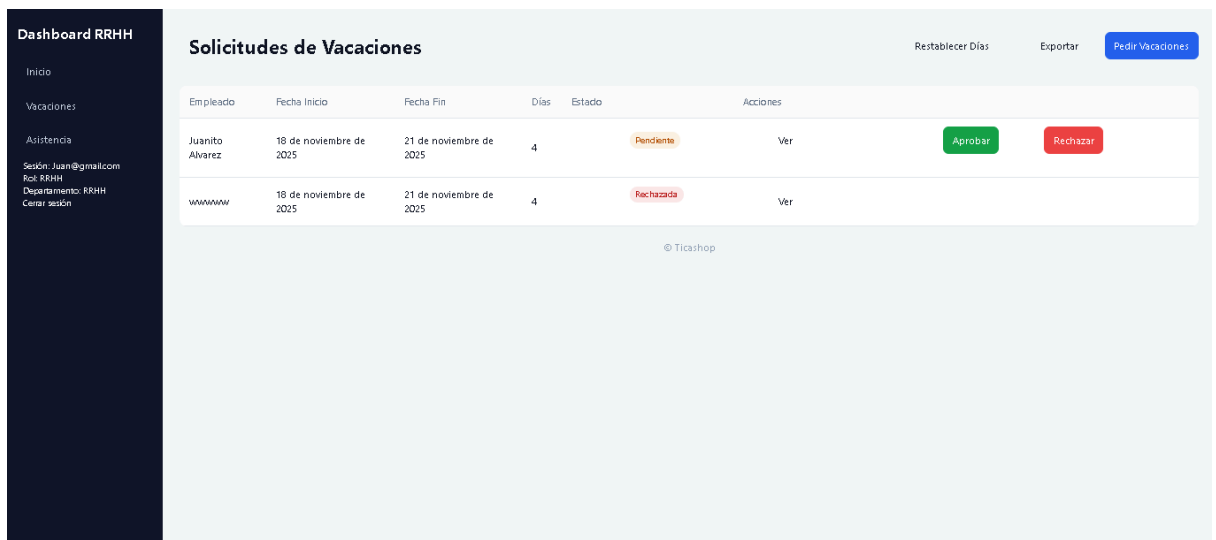
**Solicitudes de Vacaciones**

Exportar [Pedir Vacaciones](#)

| Empleado | Fecha Inicio            | Fecha Fin               | Días | Estado    | Acciones |
|----------|-------------------------|-------------------------|------|-----------|----------|
| wwwwww   | 18 de noviembre de 2025 | 21 de noviembre de 2025 | 4    | Rechazada | Ver      |

© Ticashop

- **Figura 2: Aprobación de Vacaciones – Vista RRHH**



**Dashboard RRHH**

Inicio

Vacaciones

Asistencia

Sesión: Juanito@gmail.com  
Rol: RRHH  
Departamento: RRHH  
Cerrar sesión

**Solicitudes de Vacaciones**

Restablecer Días Exportar [Pedir Vacaciones](#)

| Empleado        | Fecha Inicio            | Fecha Fin               | Días | Estado    | Acciones   |
|-----------------|-------------------------|-------------------------|------|-----------|--|
| Juanito Alvarez | 18 de noviembre de 2025 | 21 de noviembre de 2025 | 4    | Pendiente | Ver <a href="#">Aprobar</a> <a href="#">Rechazar</a> |
| wwwwww          | 18 de noviembre de 2025 | 21 de noviembre de 2025 | 4    | Rechazada | Ver  |

© Ticashop



● **Figura 3: Liquidaciones – Vista Departamento de Nómina**

Dashboard RRHH

Inicio

Liquidaciones

Sesión: basti@gmail.com

Rol: NOMINA

Departamento: FINANZAS

Cerrar sesión

Liquidaciones de Sueldo

+ Generar Liquidación

| Empleado        | Periodo        | Monto Total | Estado          | Acciones |                                |
|-----------------|----------------|-------------|-----------------|----------|--------------------------------|
| Juanito Alvarez | noviembre 2025 | \$300000    | Pendiente Firma | Ver      | <div>Descargar PDFFirmar</div> |
| wwwwww          | noviembre 2025 | \$500000    | Pendiente Firma | Ver      | <div>Descargar PDFFirmar</div> |
| Mr Emetea       | noviembre 2025 | \$500000    | Pendiente Firma | Ver      | <div>Descargar PDFFirmar</div> |
| Victoria        | noviembre 2025 | \$500000    | Pendiente Firma | Ver      | <div>Descargar PDFFirmar</div> |
| Doki            | noviembre 2025 | \$500000    | Pendiente Firma | Ver      | <div>Descargar PDFFirmar</div> |
| Matias Silva    | noviembre 2025 | \$500000    | Pendiente Firma | Ver      | <div>Descargar PDFFirmar</div> |
| Felipe Silva    | noviembre 2025 | \$500000    | Pendiente Firma | Ver      | <div>Descargar PDFFirmar</div> |
| Mariano         | noviembre 2025 | \$450000    | Pendiente Firma | Ver      | <div>Descargar PDFFirmar</div> |

© Ticashop

● **Figura 4: Asistencia – Vista RRHH/Admin**

Dashboard RRHH

Inicio

Vacaciones

Liquidaciones

Comisiones

Asistencia

Administración

Sesión: doki@empresa.cl

Rol: ADMIN

Departamento: RRHH

Cerrar sesión

Control de Asistencia

Desde dd-mm-aaaa

Hasta dd-mm-aaaa

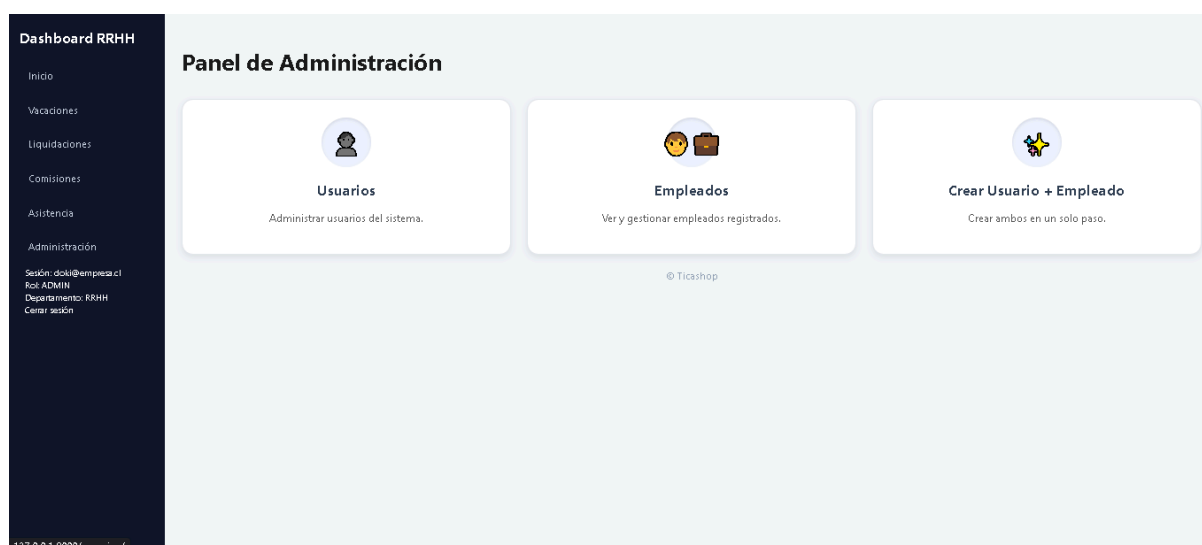
Empleado Nombre o RUT

Filtrar

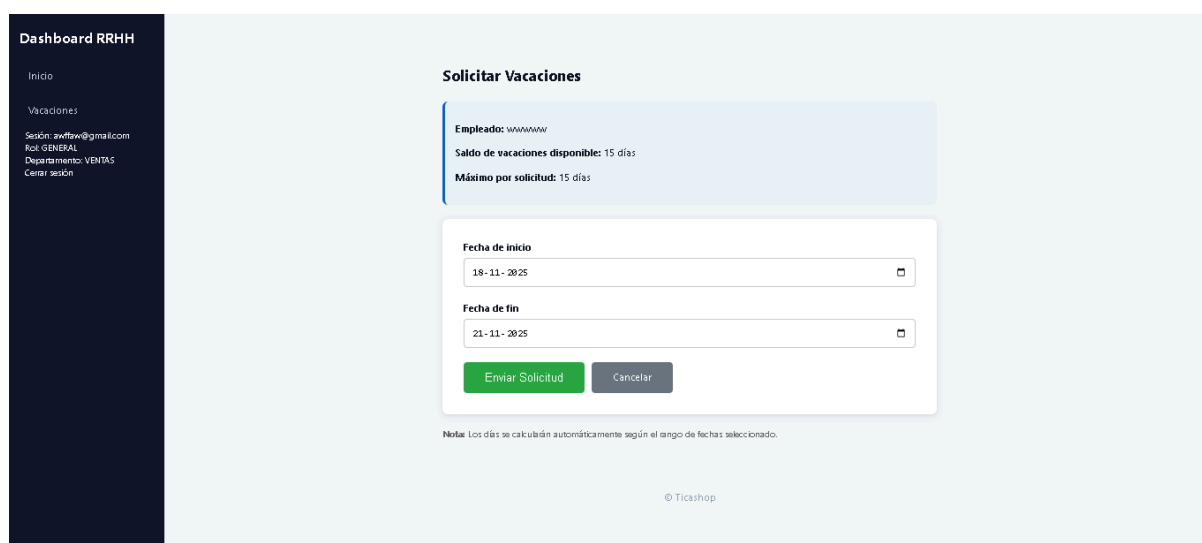
Exportar

| Empleado | Fecha                   | Entrada | Salida | Horas | Estado   |
|----------|-------------------------|---------|--------|-------|----------|
| Mariano  | 30 de noviembre de 2025 | 09:00   | 18:00  | -     | Presente |
| Victoria | 30 de noviembre de 2025 | 09:00   | 18:00  | -     | Presente |
| Mariano  | 29 de noviembre de 2025 | 09:00   | 18:00  | -     | Presente |
| Victoria | 29 de noviembre de 2025 | 09:00   | 18:00  | -     | Presente |
| Mariano  | 28 de noviembre de 2025 | 09:00   | 18:00  | -     | Presente |
| Victoria | 28 de noviembre de 2025 | 09:00   | 18:00  | -     | Presente |
| Mariano  | 27 de noviembre de 2025 | 09:00   | 18:00  | -     | Presente |
| Victoria | 27 de noviembre de 2025 | 09:00   | 18:00  | -     | Presente |
| Mariano  | 26 de noviembre de 2025 | 09:00   | 18:00  | -     | Presente |
| Victoria | 26 de noviembre de 2025 | 09:00   | 18:00  | -     | Presente |
| Mariano  | 25 de noviembre de 2025 | 09:00   | 18:00  | -     | Presente |
| Victoria | 25 de noviembre de 2025 | 09:00   | 18:00  | -     | Presente |

**Figura 5: Gestión de Usuarios – Vista Administrador**



- **Figura 6 – Vista “Solicitar Vacaciones (Empleado)”**



La interfaz de TiCaShop LATAM fue diseñada pensando directamente en las personas que la van a usar día a día. Por eso se buscó un estilo claro, ordenado y coherente, donde cada módulo mantiene la misma forma de navegación y los mismos tipos de botones y encabezados. Esto hace que cualquier usuario, incluso si es su primera vez en el sistema, pueda orientarse rápidamente sin necesidad de instrucciones adicionales.

También se cuidó que la interfaz acompañe la manera real en que trabajan las áreas de RRHH, Nómina y Jefatura. Las acciones más importantes están siempre a la vista, los mensajes del sistema son simples y comprensibles, y los formularios guían al usuario paso a paso para evitar errores o confusiones.

Gracias a esto, el sistema se siente intuitivo y cómodo de usar. Las personas pueden completar sus tareas sin perder tiempo buscando opciones, y todo fluye de manera natural y sin fricciones, lo que mejora la experiencia y facilita la adopción del sistema en la organización.

## **Coherencia visual y funcional en el diseño de la interfaz**

Desde el principio, el diseño de la interfaz de TICASHOP se construyó pensando en las personas que la usarán todos los días. El objetivo fue crear una experiencia visual agradable, moderna y ordenada, que permitiera a cualquier usuario moverse por el sistema sin confundirse ni perder tiempo.

Para lograr esto, se definió una identidad estética consistente: colores equilibrados, tipografías claras y elementos visuales con el mismo estilo en todas las pantallas. Los formularios, botones y tarjetas mantienen una apariencia uniforme, lo que entrega una sensación de continuidad y ayuda a que el usuario sepa exactamente qué esperar en cada módulo. Todo fue diseñado para verse limpio, profesional y fácil de entender a primera vista.

En lo funcional, se cuidaron detalles que hacen que la navegación fluya de manera natural. La barra lateral se mantiene en todas las vistas, las acciones principales están siempre visibles y los mensajes del sistema son directos y comprensibles. Esto permite que cada usuario ya sea administrador, RRHH, nómina o empleado encuentre rápidamente lo que necesita, reciba retroalimentación clara y pueda completar sus tareas sin pasos innecesarios.

Finalmente, esta unión entre estética y funcionalidad hace que el sistema se sienta coherente y familiar. Solicitar vacaciones, revisar liquidaciones, aprobar solicitudes o administrar usuarios se vuelve un proceso sencillo, sin fricciones ni complicaciones. La interfaz combina claridad y simplicidad con una presentación profesional, adaptándose a las necesidades reales del negocio y ofreciendo una experiencia cómoda y confiable para todos los usuarios.

| <u>Lineamiento</u>   | <u>Cómo se cumple</u>                     |
|----------------------|---|
| Consistencia visual  | Colores, tipografías y botones unificados |
| Accesibilidad        | Contraste adecuado, textos claros         |
| Usabilidad           | Navegación lateral, acciones visibles     |
| Simplicidad          | Formularios limpios, pasos guiados        |
| Coherencia funcional | Comportamiento idéntico entre módulos     |

## Estructura de la base de Datos y organización de la información

```
# ===== Usuarios (empleados del sistema) =====
class Empleado(models.Model):
    nombre = models.CharField(max_length=120)
    email = models.EmailField(unique=True)
    rut = models.CharField(max_length=12, unique=True)
    rol = models.CharField(max_length=20, choices=ROLES, default="GENERAL")
    departamento = models.CharField(max_length=20, choices=DEPARTAMENTOS, default="OTRO")
    activo = models.BooleanField(default=True)

    # opcional: saldo simple para vacaciones
    saldo_vacaciones_dias = models.PositiveIntegerField(default=0)

    def __str__(self):
        return f"{self.nombre} · {self.rol}"

# ===== Vacaciones =====
class SolicitudVacacional(models.Model):
    empleado = models.ForeignKey(Empleado, on_delete=models.CASCADE)
    fecha_inicio = models.DateField()
    fecha_fin = models.DateField()
    dias = models.PositiveIntegerField(default=0) # puedes llenarlo a mano o calcular en la vista
    estado = models.CharField(max_length=10, choices=ESTADO_VAC, default="PENDIENTE")
    observacion = models.TextField(blank=True)

    def __str__(self):
        return f"Vacaciones de {self.empleado.nombre} ({self.estado})"
```

```
# ===== Liquidaciones =====
class Liquidacion(models.Model):
    empleado = models.ForeignKey(Empleado, on_delete=models.CASCADE)
    periodo = models.DateField(help_text="Usa el primer día del mes, ej: 2025-11-01")
    monto_total = models.DecimalField(max_digits=14, decimal_places=2)
    estado = models.CharField(max_length=20, choices=ESTADO_LIQ, default="PENDIENTE_FIRMA")
    pdf = models.FileField(upload_to="liquidaciones/", blank=True, null=True)
    comisiones = models.DecimalField(
        max_digits=14,
        decimal_places=2,
        default=0,
        blank=True,
        null=True, # opcional, para que no sea obligatorio
    )

    def __str__(self):
        return f"Liquidación {self.empleado.nombre} · {self.periodo}"
```

```
# ===== Comisiones =====

class ComisionVenta(models.Model):
    empleado = models.ForeignKey('Empleado', on_delete=models.CASCADE)
    periodo = models.DateField()
    ventas_totales = models.DecimalField(max_digits=14, decimal_places=2, default=0)

    # 📌 la columna existe en la BD, pero la vamos a rellenar nosotros
    comision = models.DecimalField(
        max_digits=14,
        decimal_places=2,
        default=0,
        blank=True,
        null=True, # permite que llegue vacía desde phpMyAdmin
    )

    estado = models.CharField(max_length=20, default="CALCULADA", blank=True)

    COMISION_PORCENTAJE = Decimal("0.015") # 1,5%

    def save(self, *args, **kwargs):
        # si no hay ventas_totales, dejamos 0
        if self.ventas_totales is None:
            self.ventas_totales = Decimal("0")

        # 📌 Aquí va tu fórmula:
        # comision = ventas_totales * 0.015
        self.comision = self.ventas_totales * self.COMISION_PORCENTAJE

        if not self.estado:
            self.estado = "CALCULADA"

        super().save(*args, **kwargs)

    def __str__(self):
        return f"{self.empleado} · {self.periodo} · ventas={self.ventas_totales} · com={self.comision}"
```

```
# ===== Asistencia =====
class RegistroAsistencia(models.Model):
    empleado = models.ForeignKey(Empleado, on_delete=models.CASCADE)
    fecha = models.DateField()
    hora_entrada = models.TimeField(blank=True, null=True)
    hora_salida = models.TimeField(blank=True, null=True)
    estado = models.CharField(max_length=10, choices=ESTADO_ASI, default="PRESENTE")

    def clean(self):
        for nombre, t in (("hora_entrada", self.hora_entrada), ("hora_salida", self.hora_salida)):
            if t and not (0 <= t.hour <= 23):
                raise ValidationError({nombre: "La hora debe estar entre 00:00 y 23:59."})

    def __str__(self):
        return f"Asistencia {self.empleado.nombre} · {self.fecha} · {self.estado}"

# ===== Ventas =====

class Venta(models.Model):
    empleado = models.ForeignKey(Empleado, on_delete=models.CASCADE)
    fecha = models.DateField()
    monto = models.DecimalField(max_digits=14, decimal_places=2)
```

La estructura de la base de datos se implementó mediante los modelos de Django (models.py), definiendo explícitamente las claves primarias, foráneas y restricciones de integridad. Cada modelo corresponde a una entidad del negocio (por ejemplo, Empleado, SolicitudVacaciones, Liquidacion, Asistencia y Usuario), lo que facilita mantener el diseño alineado con el dominio funcional y asegura que los cambios se gestionan de forma controlada desde el código

## Optimización y normalización de la base de datos

La base de datos diseñada para el módulo de Recursos Humanos y Ventas fue construida siguiendo principios de normalización, con el objetivo de evitar redundancias, garantizar integridad referencial y optimizar el rendimiento del sistema. A continuación se describen las mejoras aplicadas.

### 1. Normalización de datos

La normalización se aplicó hasta la **Tercera Forma Normal (3NF)** para asegurar una estructura coherente y eficiente.

#### 1.1 Primera Forma Normal (1NF)

- Todas las tablas tienen **campos atómicos**, sin valores repetidos ni listas dentro de una misma columna.
- Ejemplos:

- En *RegistroAsistencia* los campos *hora\_entrada* y *hora\_salida* están correctamente separados.
- En *Liquidación* el campo *comisiones* es independiente y no se mezcla con otros montos.

## 1.2 Segunda Forma Normal (2NF)

- Ninguna tabla depende de claves compuestas; cada entidad tiene una **primary key propia**.
- Todas las columnas dependen directamente de la clave primaria.
  - Ejemplo: En *Venta*, los campos *fecha* y *monto* dependen directamente del ID de la venta y no del empleado.

## 1.3 Tercera Forma Normal (3NF)

- No existen dependencias transitivas.
- Los datos que pertenecen a otras entidades están separados en tablas propias.
  - Ejemplo: Los roles no se guardan duplicados; se almacenan como elección (**choices**) desde la tabla *Empleado*.
  - Las comisiones no se calculan dentro de *Liquidación*, sino en la entidad *ComisionVenta*, evitando mezclar responsabilidades.

# 2. Optimización estructural aplicada

## 2.1 Separación lógica de entidades

Se separaron claramente los módulos según sus responsabilidades:

| Entidad  | Responsabilidad  |
|----------|--|
| Empleado | Datos principales del trabajador y su estado (activo/inactivo) |

|                            |  |
|----------------------------|--|
| <b>SolicitudVacacional</b> | Administración de vacaciones y su estado               |
| <b>RegistroAsistencia</b>  | Control de horas de entrada/salida                     |
| <b>Venta</b>               | Registro de ventas diarias por empleado                |
| <b>ComisionVenta</b>       | Cálculo de comisiones basado en ventas                 |
| <b>Liquidacion</b>         | Resultados finales de sueldo, comisiones y archivo PDF |

Esta separación permite escalabilidad, mantenibilidad y evita duplicación de información.

## 2.2 Optimización de relaciones con ForeignKey

Cada tabla que depende de un empleado usa:

```
empleado = models.ForeignKey(Empleado, on_delete=models.CASCADE)
```

Esto permite:

- Integridad referencial automática.
- Evitar registros “huérfanos”.
- Eliminar toda la información asociada cuando un empleado deja de existir (política lógica del negocio).

## 2.3 Cálculo automático de campos

En *ComisionVenta*, la comisión se calcula en el método `save()`:

```
self.comision = self.ventas_totales * self.COMISION_PORCENTAJE
```

Esto evita:



- Duplicación de lógica en otras partes del código.
- Inconsistencias en comisiones si se modificara el monto.
- Almacenar datos derivados innecesarios.

Optimización aplicada:

- Se calcula una sola vez.
- Minimiza consultas, evitando recomputar en vistas o templates.

## 2.4 Columnas opcionales para compatibilidad

Campos como pdf, comisiones, hora\_entrada y hora\_salida están configurados como:

blank=True, null=True

Esto permite:

- Mayor flexibilidad en el ingreso de datos.
- Evitar errores cuando el registro aún no está completo.
- Adaptarse a cambios del negocio sin romper migraciones.

## 2.5 Uso de DecimalField para cálculos financieros

Los montos usan:

```
models.DecimalField(max_digits=14, decimal_places=2)
```

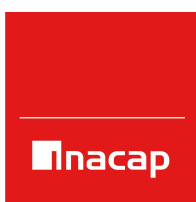
Esto garantiza:

- Exactitud en cálculos monetarios.
- Evita errores de flotantes típicos de Python.
- Permite soportar montos grandes para ventas o comisiones.

## 2.6 Estados normalizados con **choices**

Ejemplos:

```
estado = models.CharField(max_length=20, choices=ESTADO_LIQ)
```



Beneficios:

- Reduce errores humanos (no puede ingresarse "aprovado", "vrificado", etc.).
- Facilita filtros y reportes.
- Mantiene consistencia en cada módulo.

### 3. Optimización orientada al rendimiento

#### 3.1 Minimización de redundancia

Ejemplos:

- La comisión *no* está repetida en Liquidación; se usa desde *ComisionVenta*.
- Los días de vacaciones no están en *Empleado* sino separados en *SolicitudVacacional*, respetando la naturaleza temporal del dato.

#### 3.2 Queries más eficientes

Al separar las entidades:

- Asistencia → solo analiza horas.
- Ventas → solo maneja montos.
- Comisiones → calcula resultados sin cargar Liquidación.
- Liquidación → consume los resultados finales ya procesados.

Esto evita *joins* innecesarios y reduce carga en el sistema.

#### 3.3 Eliminación de datos duplicados

Antes (posible diseño incorrecto):

- Guardar ventas dentro de Liquidación.
- Guardar comisiones dentro de Empleado.

Ahora:

- Cada dato está en su entidad correspondiente.

## Patrones de seguridad aplicados en el módulo de RRHH

La seguridad del módulo de Recursos Humanos fue una prioridad desde el comienzo, ya que maneja información sensible como vacaciones, asistencia y liquidaciones. Por esa razón, se integraron varias medidas que permiten proteger los datos y asegurar que solo las personas autorizadas puedan acceder a ellos.

En primer lugar, el sistema utiliza el mecanismo de autenticación de Django, que guarda las contraseñas de manera hasheada. Esto significa que las contraseñas nunca se almacenan en texto plano, lo que reduce enormemente el riesgo en caso de que alguien intente acceder a la base de datos de forma indebida.

También se implementó control de acceso por roles. Esto asegura que cada usuario vea únicamente lo que necesita según su perfil: RRHH y Nómina pueden acceder a información global del personal, mientras que los empleados solo pueden revisar sus propios datos. Además, las vistas están protegidas para que nadie pueda ingresar sin haber iniciado sesión.

Otro aspecto importante es que los formularios pasan por validaciones desde el servidor antes de guardar cualquier información. Esto ayuda a evitar errores, datos mal ingresados y posibles intentos de manipulación. A esto se suma la protección integrada de Django git mediante el uso del token `{% csrf_token %}` en todos los formularios, lo que previene ataques tipo CSRF y agrega una capa adicional de seguridad en cada acción.

En conjunto, estas medidas permiten que el módulo de RRHH funcione sobre una base sólida y segura, aplicando buenas prácticas que resguardan la información de los trabajadores y mantienen la integridad del sistema.

## Colaboración en equipo

El desarrollo del módulo de Recursos Humanos de TICASHOP fue un trabajo colaborativo que evolucionó bastante a lo largo del proyecto. Al inicio, como equipo, no teníamos del todo claro cómo funcionaba GitHub, por lo que optamos por compartir el código a través de Google Drive. Aunque este método nos permitió

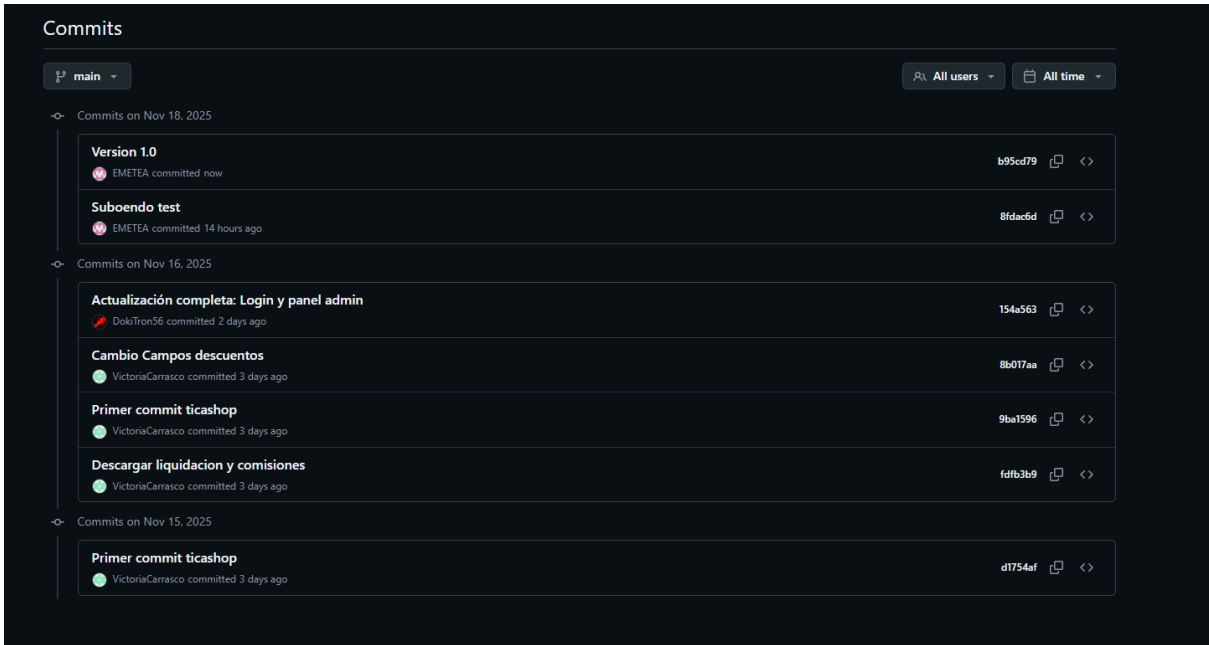
avanzar, pronto nos dimos cuenta de que dificultaba llevar un control ordenado de los cambios y se hacía más complicado trabajar en paralelo.

Con el tiempo y después de buscar información y practicar, empezamos a utilizar GitHub de manera correcta. Al principio hubo algunos errores, archivos duplicados y commits que no subimos bien, pero poco a poco fuimos entendiendo cómo trabajar con ramas, cómo integrar los cambios y cómo mantener el repositorio limpio. Esta transición marcó una diferencia importante, porque desde ese momento pudimos colaborar de forma mucho más fluida y profesional.

Ya usando GitHub, cada integrante realizó sus propios commits, lo que permitió ver claramente quién trabajó en qué parte del módulo y facilitó corregir errores, revisar avances y evitar la pérdida de información. Esto ayudó a mantener una integración ordenada entre las distintas funcionalidades, como el inicio de sesión, la gestión de vacaciones y el panel administrativo.

Además, trabajamos siguiendo principios de metodología ágil. Cada uno tomó tareas del sprint correspondiente y las fue completando progresivamente, lo que permitió avanzar de manera coordinada, resolver dudas rápidamente y tener siempre claridad sobre el estado del proyecto.

**Historial de commits del proyecto TICASHOP**



| Fecha        | Commit                                      | Usuario          | Hash    |
|--------------|---|------------------|---------|
| Nov 18, 2025 | Version 1.0                                 | EMETEA           | b95cd79 |
|              | Suboendo test                               | EMETEA           | 8fdac6d |
| Nov 16, 2025 | Actualización completa: Login y panel admin | DokoTiron56      | 154a563 |
|              | Cambio Campos descuentos                    | VictoriaCarrasco | 8b077aa |
|              | Primer commit ticashop                      | VictoriaCarrasco | 9ba1596 |
|              | Descargar liquidacion y comisiones          | VictoriaCarrasco | fd1b3b9 |
|              | Primer commit ticashop                      | VictoriaCarrasco | d1754af |

Esta experiencia nos permitió comprender la importancia del control de versiones y mejorar nuestra coordinación como equipo para futuros proyectos.

## Configuración del entorno de trabajo

Para el desarrollo del módulo de Recursos Humanos de TICASHOP se preparó un entorno de trabajo simple, pero completamente funcional y replicable. El objetivo fue que cualquier integrante del equipo pudiera levantar el proyecto en su propio computador siguiendo un mismo conjunto de pasos.

Primero se instaló Python 3 y el framework Django, que es la base del proyecto. El trabajo se organizó dentro de un entorno virtual, lo que permite separar las dependencias de este sistema de otras instalaciones que pueda tener el equipo. Sobre esta base se agregaron las librerías necesarias para el funcionamiento del módulo:

- Django: framework principal para el desarrollo web.
- pymysql / mysqlclient: para conectar Django con la base de datos MariaDB/MySQL.
- xhtml2pdf: para generar las liquidaciones en formato PDF a partir de las vistas HTML.
- bcrypt: para el manejo seguro de contraseñas mediante hashing.
- openpyxl: para la exportación de archivos Excel.

Una vez instaladas las dependencias, se configuró la conexión a la base de datos en el archivo settings.py, indicando nombre de la base, usuario, contraseña y host. Luego se ejecutaron las migraciones de Django para crear automáticamente las tablas asociadas a los modelos definidos en el módulo de RRHH.

Para el desarrollo se utilizó Visual Studio Code como editor principal, y el sistema se ejecuta en modo local mediante el comando `python manage.py runserver`, lo que permite probar todas las funcionalidades desde el navegador. Finalmente, el uso de Git y GitHub como control de versiones asegura que la configuración del entorno y el código se mantengan sincronizados entre los distintos integrantes del equipo.

En conjunto, esta configuración deja el entorno de trabajo documentado y fácil de reproducir: basta con instalar las dependencias indicadas, configurar la base de datos y ejecutar las migraciones para tener el proyecto funcionando en cualquier equipo.

## Documentación de la implementación

La implementación del módulo de Recursos Humanos de TICASHOP se desarrolló utilizando Django, organizando el proyecto en una estructura clara de archivos y carpetas que facilita su mantenimiento y comprensión. En la Figura se muestra la estructura principal del proyecto dentro de Visual Studio Code, donde cada componente cumple un rol específico dentro del funcionamiento del sistema.

El proyecto está compuesto por dos partes principales: la configuración general del sistema (ticashop/) y la aplicación que contiene todas las funciones del módulo de RRHH (apptica/). Dentro de la carpeta principal del proyecto se encuentran archivos esenciales como settings.py, donde se definen las configuraciones del sistema; urls.py, que administra las rutas globales; y los archivos wsgi.py y asgi.py, necesarios para el despliegue.

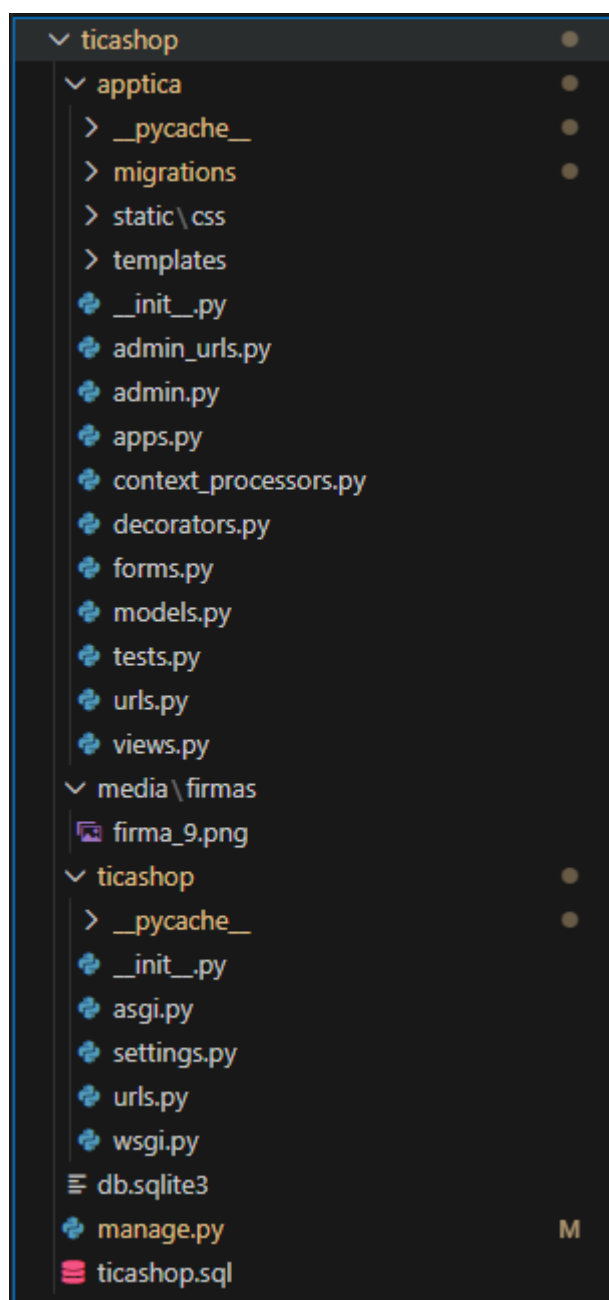
La aplicación appctica concentra la lógica del módulo. Allí se incluyen:

- models.py: define las entidades del sistema, como empleados, solicitudes y liquidaciones, junto con sus relaciones y campos.
- views.py: contiene la lógica que recibe las solicitudes del usuario, procesa datos, valida formularios y retorna la información correspondiente a las plantillas.
- forms.py: gestiona la validación de formularios desde el servidor, asegurando la correcta entrada de datos.
- urls.py: organiza las rutas internas del módulo y conecta cada URL con su vista correspondiente.
- templates/: almacena las plantillas HTML que dan forma visual al módulo, donde se combinan elementos estáticos y datos generados dinámicamente.
- static/: contiene recursos estáticos (CSS, imágenes, scripts) utilizados en la interfaz.

La carpeta migrations/ guarda los archivos generados automáticamente por Django para aplicar cambios en la base de datos, manteniendo un historial ordenado de modificaciones.

Gracias a esta organización, la implementación queda completamente documentada de forma natural: cada archivo y carpeta tiene un propósito claro, lo que permite entender rápidamente el funcionamiento del módulo, repetir la instalación en otro entorno y realizar mejoras futuras sin dificultad.

### *Estructura de archivos del proyecto TiCaShop LATAM en Visual Studio Code.*



## Cobertura del plan de pruebas

Para el módulo de Recursos Humanos se elaboró un plan de pruebas que cubre todas las funcionalidades principales del sistema: login, control de roles, solicitud y aprobación de vacaciones, visualización de liquidaciones, generación de PDF y restricciones de acceso. El plan incluye escenarios normales, casos de error y pruebas de borde, garantizando una evaluación completa del comportamiento esperado del sistema.

Los casos fueron organizados en una tabla donde se identifican el ID de la prueba, la funcionalidad evaluada, los datos de entrada, el resultado esperado y las dependencias cuando corresponden.

|       |                                |                                       |            |                             |                                |                             |                             |                                       |                                      |  | Información para el Seguimiento      |            |                        |  |
|-------|--------------------------------|---------------------------------------|------------|-----------------------------|--------------------------------|-----------------------------|-----------------------------|---------------------------------------|--------------------------------------|--|--------------------------------------|------------|------------------------|--|
| Id    | Caso de Prueba                 | Descripción                           | Fecha      | Área Funcional / Subproceso | Funcionalidad / Característica | Datos / Acciones de Entrada | Resultado Esperado          | Requerimientos de Ambiente de Pruebas | Procedimientos especiales requeridos | Dependencias con otros casos de Prueba | Resultado Obtenido                   | Estado     | Última Fecha de Estado | Observaciones                                  |
| CP001 | Login correcto                 | Usuario ingresa credenciales válidas  | 2025-11-16 | Autenticación               | Login                          | email + contraseña          | Acceso al sistema           | Ambiente QA                           | N/A                                  | N/A                                    | Acceso exitoso                       | Aprobado   | 2025-11-18             | Credenciales válidas, redirección al dashboard |
| CP002 | Login incorrecto               | Usuario ingresa contraseña errónea    | 2025-11-17 | Autenticación               | Login                          | email + clave incorrecta    | Mostrar mensaje de error    | Ambiente QA                           | N/A                                  | N/A                                    | Mensaje de error mostrado            | Aprobado   | 2025-11-18             | Validación correcta de credenciales            |
| CP003 | Login sin estar registrado     | Usuario no existente intenta ingresar | 2025-11-18 | Autenticación               | Login                          | email no registrado         | Mostrar "usuario no existe" | Ambiente QA                           | N/A                                  | N/A                                    | Mensaje "usuario no existe" mostrado | Aprobado   | 2025-11-18             | Manejo adecuado de usuarios no registrados     |
| CP004 | Solicitud de vacaciones válida | Empleado envía solicitud con fechas   | 2025-11-19 | Vacaciones                  | Solicitar                      | fecha inicio/fin            | Solicitud registrada        | Ambiente QA                           | N/A                                  | N/A                                    | Solicitud registrada correctamente   | En Proceso | 2025-11-18             | Fechas válidas, solicitud visible en baso de   |

|       |  | correctas                              |            |               |                   |                    |                             |             |     |     |                                  |            |            | datos  |
|-------|--|--|------------|---------------|-------------------|--------------------|-----------------------------|-------------|-----|-----|----------------------------------|------------|------------|--|
| CP005 | Solicitud con fechas invertidas            | Fecha fin < fecha inicio               | 2025-11-20 | Vacaciones    | Validación        | fechas incorrectas | Mostrar error de validación | Ambiente QA | N/A | N/A | Error de validación mostrado     | En Proceso | 2025-11-18 | Falta validación de fechas                       |
| CP006 | Ver solicitudes de un empleado             | Carga del listado                      | 2025-11-21 | Vacaciones    | Historial         | —                  | Lista visible               | Ambiente QA | N/A | N/A | Listado cargado sin errores      | En Proceso | 2025-11-18 | Algunos roles ingresa                            |
| CP007 | Aprobación de vacaciones                   | Jefatura aprueba una solicitud         | 2025-11-22 | Vacaciones    | Aprobar           | ID solicitud       | Estado cambia a “Aprobado”  | Ambiente QA | N/A | N/A | Estado actualizado a “Aprobado”  | Aprobado   | 2025-11-18 | Acción de RRHH registrada correctamente          |
| CP008 | Rechazo de vacaciones                      | Jefatura rechaza una solicitud         | 2025-11-23 | Vacaciones    | Rechazar          | ID solicitud       | Estado cambia a “Rechazado” | Ambiente QA | N/A | N/A | Estado actualizado a “Rechazado” | Aprobado   | 2025-11-18 | Rechazo registrado y visible en Base de Datos    |
| CP009 | Empleado intenta acceder a vista de Nómina | Prueba de rol                          | 2025-11-24 | Seguridad     | Control de acceso | /liquidaciones     | Acceso denegado             | Ambiente QA | N/A | N/A | Acceso denegado                  | Aprobado   | 2025-11-18 | Control de acceso por rol funciona correctamente |
| CP010 | Generación de liquidación                  | Nómina genera PDF                      | 2025-11-25 | Nómina        | Liquidaciones     | Datos salariales   | PDF generado correctamente  | Ambiente QA | N/A | N/A | PDF generado correctamente       | Aprobado   | 2025-11-18 | Archivo disponible para descarga                 |
| CP011 | Empleado descarga su liquidación           | Prueba de lectura                      | 2025-11-26 | Nómina        | Ver liquidación   | ID liquidación     | Descarga exitosa            | Ambiente QA | N/A | N/A | Descarga exitosa                 | Aprobado   | 2025-11-18 | Archivo descargado sin errores                   |
| CP012 | Acceso sin login                           | Usuario intenta entrar a URL protegida | 2025-11-27 | Seguridad     | Acceso            | URL sin sesión     | Redirige a login            | Ambiente QA | N/A | N/A | Redirección a login              | Aprobado   | 2025-11-18 | Seguridad de sesión validada                     |
| CP013 | Cerrar sesión                              | Usuario cierra sesión                  | 2025-11-28 | Autenticación | Logout            | —                  | Sesión finalizada           | Ambiente QA | N/A | N/A | Sesión finalizada correctamente  | Aprobado   | 2025-11-18 | Usuario redirigido a pantalla de login           |
| CP014 | Carga de muchas solicitudes                | 20+ solicitudes en la                  | 2025-11-29 | Vacaciones    | Listado           | 20 registros       | Lista carga sin errores     | Ambiente QA | N/A | N/A | Lista cargada sin                | Aprobado   | 2025-11-18 | Buen rendimiento con carga                       |

|       |                        |                       |            |         |         |   |                            |             |     |     |                               |          |            |  |
|-------|------------------------|-----------------------|------------|---------|---------|---|----------------------------|-------------|-----|-----|-------------------------------|----------|------------|--|
|       | s (estrés)             | vista                 |            |         |         |   |                            |             |     |     | errores                       | do       |            | alta                                     |
| CP015 | Error en BD o servicio | Inducir error pequeño | 2025-11-30 | Sistema | General | — | Mostrar mensaje controlado | Ambiente QA | N/A | N/A | Mensaje controlado o mostrado | Aprobado | 2025-11-18 | Manejo de errores funciona correctamente |

[Link de la hoja de cálculo](#)

## Ejecución del Protocolo de Pruebas

Las pruebas fueron ejecutadas conforme al protocolo definido en el plan de pruebas. Se abordaron quince casos de prueba que cubren funcionalidades críticas del módulo de Recursos Humanos, incluyendo autenticación, gestión de vacaciones, control de acceso y generación de liquidaciones.

Cada caso fue documentado con su resultado obtenido, estado (aprobado o en proceso) y observaciones relevantes. Durante la ejecución se identificaron inconsistencias en algunos flujos funcionales:

- **Validación de fechas (CP005):** El sistema no mostraba el mensaje de error esperado al ingresar fechas invertidas en la solicitud de vacaciones. Esta validación fue ajustada y el caso se encuentra en proceso de revalidación.
- **Carga de solicitudes por rol (CP006):** Se detectó que algunos roles no acceden correctamente al historial de solicitudes. El comportamiento está siendo revisado.
- **Registro de solicitudes (CP004):** Aunque la solicitud se registra correctamente, se está verificando la persistencia en base de datos.

Los casos con observaciones fueron marcados como “En Proceso” y están siendo reevaluados tras aplicar los ajustes correspondientes.

## Validación de Resultados

Del total de casos ejecutados:

- **12 casos fueron aprobados** en la primera ejecución, cumpliendo con los criterios funcionales establecidos.
- **3 casos se encuentran en proceso**, sujetos a corrección y nueva validación.

Las funcionalidades validadas incluyen:

- **Autenticación:** Login, logout y control de sesión funcionan correctamente, incluyendo el manejo de errores y usuarios no registrados.

- **Gestión de vacaciones:** Solicitudes, aprobaciones, rechazos y visualización de historial operan de forma estable, con excepción de los casos en revisión.
- **Seguridad:** El control de acceso por rol y la redirección en ausencia de sesión están correctamente implementados.
- **Nómina:** La generación y descarga de liquidaciones se realiza sin errores, con archivos disponibles para el usuario.

El sistema demostró estabilidad, coherencia en los flujos funcionales y correcto manejo de datos en el entorno de pruebas.

## Recomendaciones

Como mejora futura se recomienda:

- **Automatizar las pruebas de login y validaciones de formularios**, lo que permitirá incrementar la eficiencia del proceso de pruebas y reducir la intervención manual.
- **Ampliar la cobertura de pruebas de roles y permisos**, para asegurar que todos los perfiles accedan correctamente a las funcionalidades asignadas.
- **Implementar pruebas de carga y estrés periódicas**, especialmente en módulos de solicitudes, para validar el rendimiento ante volúmenes altos de datos.

## Conclusión

El desarrollo del módulo de Recursos Humanos de TiCaShop LATAM permitió construir una solución completa, funcional y alineada con las necesidades operacionales del negocio. El sistema logra automatizar procesos que tradicionalmente se realizaban de forma manual, como solicitudes de vacaciones, control de asistencia, cálculo de comisiones y generación de liquidaciones, mejorando significativamente la eficiencia y precisión en la gestión interna.

Las validaciones realizadas demostraron que las funcionalidades principales cumplen correctamente con los requisitos, manteniendo seguridad, coherencia visual y un flujo de navegación intuitivo para cada tipo de usuario. La estructura de la base de datos, normalizada y optimizada, asegura integridad y escalabilidad, mientras que el control de acceso por roles garantiza que cada colaborador interactúe únicamente con la información que le corresponde.

El uso de Django, GitHub y un entorno de trabajo estructurado permitió desarrollar el proyecto de forma ordenada y colaborativa, fortaleciendo el aprendizaje del equipo y dejando una base sólida para su mantenimiento y crecimiento futuro. Como resultado, el módulo entrega una plataforma robusta, segura y preparada para integrarse con las demás áreas del sistema TiCaShop LATAM.

## III. MANUAL DE USUARIO

# Módulo de Recursos Humanos

**Versión:** 1.0

**Fecha:** Noviembre 2025

**Proyecto:** TiCaShop LATAM

**Elaborado por:** Equipo de Desarrollo – RRHH

## 1. Introducción

El presente Manual de Usuario describe el funcionamiento del módulo de Recursos Humanos del sistema TiCaShop LATAM. Este módulo fue desarrollado con el objetivo de optimizar los procesos internos relacionados con la gestión de empleados, solicitudes de vacaciones, registro de asistencia, cálculo de liquidaciones y comisiones, además de la administración del personal mediante un panel exclusivo para el rol de administrador.

Este documento explica, de manera detallada y paso a paso, cómo instalar el sistema, cómo ejecutarlo en un entorno local y cómo utilizar cada una de sus funcionalidades según el rol asignado al usuario.

## 2. Objetivo del Sistema

El objetivo general del Módulo RRHH de TiCaShop LATAM es proporcionar una plataforma web segura, moderna y fácil de usar para automatizar tareas administrativas, mejorar la trazabilidad del personal y entregar herramientas eficientes a las áreas de Recursos Humanos, Nómina, Supervisión Comercial y Ventas.

Entre los objetivos específicos se encuentran:

- Centralizar la información del personal.
- Facilitar la gestión de vacaciones y asistencia.
- Automatizar el cálculo y la descarga de liquidaciones.
- Garantizar un control de acceso seguro mediante roles.
- Proveer un panel administrativo para gestionar empleados y usuarios.

## 3. Requisitos del Sistema

### 3.1 Requisitos de Software

- Python 3.10 o superior
- Git
- MySQL o MariaDB
- Visual Studio Code (opcional)

### 3.2 Librerías de Python

El sistema requiere las siguientes dependencias:

- Django
- pymysql / mysqlclient
- bcrypt
- xhtml2pdf
- openpyxl

Estas librerías permiten el funcionamiento del backend, la conexión a la base de datos y la generación de documentos PDF o Excel.

## 4. Instalación del Sistema

### 4.1 Descarga del Proyecto desde GitHub

1. Abrir una terminal o PowerShell.
2. Ejecutar el siguiente comando:

```
git clone https://github.com/TU_USUARIO/TU_REPO.git
```



Ejemplo:

```
git clone https://github.com/eltoripa/ticashop_rrhh.git
```

3. Ingresar a la carpeta del proyecto:

```
cd ticashop_rrhh
```

## 4.2 Creación de un Entorno Virtual

Para aislar dependencias del proyecto:

```
python -m venv venv
```

Activación:

### En Windows

```
venv\Scripts\activate
```

### En Linux/Mac

```
source venv/bin/activate
```

---

## 4.3 Instalación de Dependencias

Si el proyecto incluye un archivo requirements.txt:

```
pip install -r requirements.txt
```

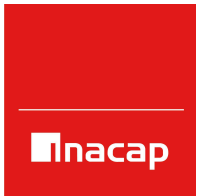
De lo contrario:

```
pip install django pymysql bcrypt xhtml2pdf openpyxl
```

---

## 4.4 Configuración de la Base de Datos

En MySQL ejecutar:



```
CREATE DATABASE ticashop_rrhh;
```

Luego en ticashop/settings.py configurar:

```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.mysql',  
        'NAME': 'ticashop_rrhh',  
        'USER': 'root',  
        'PASSWORD': 'TU_CLAVE',  
        'HOST': 'localhost',  
        'PORT': '3306',  
    }  
}
```

Aplicar migraciones:

```
python manage.py migrate
```

## 4.5 Ejecución del Servidor

Para iniciar el sistema:

```
python manage.py runserver
```

El sistema estará disponible en el navegador en:

<http://127.0.0.1:8000/>

## 5. Inicio de Sesión

El sistema utiliza dos elementos para la autenticación:

- **RUT** del empleado
- **Contraseña** asociada al usuario Django (User)

El formulario contiene:



- Campo de RUT
- Campo de contraseña

Si el usuario es válido, será redirigido al dashboard correspondiente según su rol.

## 6. Roles y Permisos de Usuario

El sistema opera utilizando un modelo de control de acceso basado en roles. Cada rol tiene acceso específico a módulos y acciones.

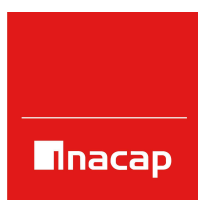
| Rol           | Accesos Permitidos  |
|---------------|---|
| ADMIN         | Acceso total a todas las funciones del sistema                            |
| RRHH          | Aprobar/rechazar vacaciones, ver asistencia, solicitar vacaciones propias |
| NÓMINA        | Liquidaciones, asistencia, vacaciones                                     |
| SUP COMERCIAL | Comisiones, asistencia, vacaciones  |
| VENDEDOR      | Ver sus propias liquidaciones, solicitar vacaciones, ver asistencia       |

## 7. Guía de Uso del Sistema

### 7.1 Dashboard Principal

Una vez iniciada la sesión, el usuario accede al dashboard donde puede visualizar:

- Total de empleados
- Presencia del día
- Solicitudes de vacaciones pendientes
- Liquidaciones pendientes de firma



- Comisiones pendientes

Cada usuario verá únicamente los módulos permitidos por su rol.

## 7.2 Módulo de Vacaciones

### Solicitar Vacaciones (Empleado)

1. Acceder a “Vacaciones”.
2. Ingresar fecha de inicio y fin.
3. Enviar solicitud.
4. El estado aparecerá como “Pendiente”.

### Aprobar o Rechazar (RRHH/ADMIN)

1. Acceder al listado de solicitudes.
2. Revisar solicitudes pendientes.
3. Hacer clic en *Aprobar* o *Rechazar*.
4. El estado cambia inmediatamente.

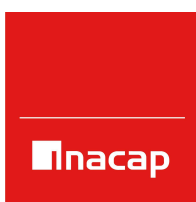
## 7.3 Módulo de Liquidaciones

### Ver Liquidaciones (Empleado/Nómina)

Los empleados pueden ver solamente sus propias liquidaciones.

### Administrar Liquidaciones (Nómina)

1. Entrar al módulo de liquidaciones.
2. Generar nuevas liquidaciones mensuales.
3. Descargar PDF individual.



4. Ver detalles por empleado.

## 7.4 Módulo de Asistencia

Permite:

- Ver asistencia diaria o histórica
- Filtrar por fecha
- Buscar por nombre o RUT

Roles permitidos: ADMIN, RRHH, NOMINA, SUP COM, VENDEDOR.

## 7.5 Módulo de Comisiones

Solo para SUPERVISOR COMERCIAL.

Permite:

- Ver ventas mensuales
- Calcular comisión automáticamente
- Exportar reportes en CSV

## 7.6 Panel de Administración (ADMIN)

Permite:

- Crear empleados
- Crear usuarios Django

- Editar información del personal
- Activar/desactivar usuarios
- Asignar roles y departamentos
- Exportar datos en CSV

## 8. Exportación de Información

El sistema permite exportar:

- Vacaciones
- Liquidaciones
- Comisiones
- Asistencia
- Usuarios

En formato CSV y PDF (según el módulo).

## 9. Seguridad del Sistema

- Contraseñas encriptadas (bcrypt/Django).
- Protección contra CSRF en todos los formularios.
- Control de acceso por roles.
- Validaciones en servidor y cliente.
- Sesiones seguras.

Estas medidas garantizan la integridad y confidencialidad de la información del personal.

## 10. Resolución de Problemas Comunes

| Problema                  | Solución   |
|---------------------------|--|
| Error de conexión a MySQL | Verificar usuario, contraseña y nombre de la base                  |
| Contraseña incorrecta     | Confirmar que el usuario Django coincide con el email del empleado |
| Módulo no visible         | Revisar permisos del rol asignado                                  |
| No carga panel admin      | El usuario debe tener is_staff = True                              |

## 11. Conclusión

El módulo de Recursos Humanos de TiCaShop LATAM ofrece una solución completa para la administración del personal, automatizando procesos clave y proporcionando una interfaz intuitiva y segura para los usuarios. Gracias a su diseño modular y a la implementación de buenas prácticas de desarrollo, el sistema puede ampliarse fácilmente e integrarse con nuevos módulos o áreas de la organización.

## Bibliografía

Schmuller, J. (2003). *Aprendiendo UML en 24 horas*. Prentice Hall.

Fowler, M., & Scott, K. (2000). *UML Gota a Gota: Aprendiendo UML paso a paso*. Addison-Wesley / Pearson Educación.

## Anexos

1. [https://www.canva.com/design/DAGyg0JJSjU/zN1sJpnJVGsQVR\\_9uHqUjg/edit?utm\\_content=DAGyg0JJSjU&utm\\_campaign=designshare&utm\\_medium=link2&utm\\_source=sharebutton](https://www.canva.com/design/DAGyg0JJSjU/zN1sJpnJVGsQVR_9uHqUjg/edit?utm_content=DAGyg0JJSjU&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton)
2. <https://docs.google.com/spreadsheets/d/1H4U-Y8x-P3oMdf6FKwruUGG1Quuavdq0k8Hadxs0Y/edit?usp=sharing>
3. [https://drive.google.com/drive/folders/1yMXXYLapogFtBnJ\\_kDNEXtzQJTAVoNpO?usp=sharing](https://drive.google.com/drive/folders/1yMXXYLapogFtBnJ_kDNEXtzQJTAVoNpO?usp=sharing)
4. <https://bliss-spore-25556855.figma.site/>
5. **Plan de Pruebas – Resumen**

| Tipo de prueba     | Objetivo  | Herramientas / Técnicas   | Resultado esperado  |
|--------------------|---|---|---|
| <b>Funcional</b>   | Verificar que cada módulo del sistema cumpla con los requerimientos definidos (Inventario, Ventas, RRHH, Soporte y Finanzas). | Testing manual, validación de formularios, revisión de flujos CRUD. | Todas las funciones ejecutan correctamente las operaciones definidas. |
| <b>Integración</b> | Validar la comunicación entre módulos internos y sistemas externos (ERP Defontana, API TipoCambio, SII).                      | Postman, revisiones de API REST, pruebas de sincronización.         | Los módulos interactúan sin errores ni pérdida de datos.              |
| <b>Seguridad</b>   | Detectar vulnerabilidades y confirmar protección ante ataques comunes (SQLi, XSS, CSRF).                                      | OWASP ZAP, SonarQube, revisión de código, tokens CSRF, cifrado TLS. | El sistema resiste ataques básicos y mantiene la integridad y         |

|                    |   |   |   |
|--------------------|---|---|---|
|                    |   |   | confidencialidad de los datos.  |
| <b>Rendimiento</b> | Medir tiempos de respuesta, consumo de recursos y estabilidad bajo carga. | Apache JMeter, Postman (colecciones automatizadas).   | Tiempos de respuesta inferiores a 2 s; el sistema mantiene estabilidad con $\geq 50$ usuarios concurrentes. |
| <b>Usabilidad</b>  | Evaluar la facilidad de uso, accesibilidad y claridad de la interfaz.     | Pruebas con usuarios, encuestas, revisión heurística. | Interfaz intuitiva, navegación clara y terminología comprensible.   |