Universidad Católica de Santa María

Facultad de Ciencias e Ingenierías Físicas y Formales

Escuela Profesional de Ingeniería de Sistemas

Implementación del módulo de control presupuestal y logístico para el pago de comprobantes en una universidad privada utilizando Scrum

Tesis presentada por el bachiller:

Alosilla Sanchez Moreno, Guillermo

ORCID: 0009-0005-2453-1855

para optar al Título Profesional de Ingeniero de Sistemas

Asesor:

Dr. Torres Gamarra, Nestor

ORCID: 0009-0007-5406-5482

Arequipa - Perú

2024

UCSM-ERP

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

INGENIERIA DE SISTEMAS

TITULACIÓN CON TESIS

DICTAMEN APROBACIÓN DE BORRADOR

Arequipa, 16 de Septiembre del 2024

Dictamen: 012754-C-EPIS-2024

Visto el borrador del expediente 012754, presentado por:

2018200191 - ALOSILLA SANCHEZ MORENO GUILLERMO

Titulado:

IMPLEMENTACIÓN DEL MÓDULO DE CONTROL PRESUPUESTAL Y LOGÍSTICO PARA EL PAGO DE COMPROBANTES EN UNA UNIVERSIDAD PRIVADA UTILIZANDO SCRUM

Nuestro dictamen es:

APROBADO

Titulo Profesional/Titulo de Segunda Especialidad/Grado Académico a optar:

INGENIERO DE SISTEMAS

29302116 - DELGADO DELGADO FREDY RAMIRO
DICTAMINADOR

29244573 - PAREDES MARCHENA FERNANDO GERMAN
DICTAMINADOR

29612305 - SULLA TORRES JOSE ALFREDO
DICTAMINADOR

Implementación del módulo de control presupuestal y logístico para el pago de comprobantes en una universidad privada utilizando Scrum

INFORME DE ORIGINALIDAD

16%	14%	3%	5%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

1	hdl.handle.net Fuente de Internet	1%
2	www.coursehero.com Fuente de Internet	1%
3	Submitted to Corporación Universitaria Minuto de Dios,UNIMINUTO Trabajo del estudiante	1%
4	repositorio.unas.edu.pe Fuente de Internet	<1%
5	Submitted to Ilerna Online Trabajo del estudiante	<1%
6	Submitted to Universidad TecMilenio Trabajo del estudiante	<1%
7	repositorio.utp.edu.pe Fuente de Internet	<1%
8	repositorio.ug.edu.ec Fuente de Internet	<1%

DEDICATORIA

Dedicado a mi madre, Martha, por acompañarme en los momentos más difíciles, guiarme y siempre estar a mi lado; a mi padre, Javier, por su motivación y su intención de asegurarme un mejor futuro; a mi hermana, Karen, quien es mi mayor fuente de inspiración; y a mi abuela, Elsa, quien siempre me brindó mucho amor.

AGRADECIMIENTOS

Quiero expresar mi agradecimiento a mi equipo de trabajo, quienes, además de ser grandes profesionales, se convirtieron en mis amigos y me apoyaron desde el primer día para la realización de este proyecto. A mi familia, por enseñarme a nunca rendirme pese a las adversidades y por ser una fuente constante de amor. Y finalmente, a todas esas personas que, a lo largo de mi vida, me han dejado lecciones valiosas y que han sido fundamentales para mi crecimiento personal y profesional.

RESUMEN

En la actualidad, la digitalización de procesos administrativos y financieros en instituciones educativas está avanzando significativamente debido a la adopción de nuevas tecnologías y

metodologías ágiles. Este progreso ha permitido a las distintas instituciones privadas mejorar la eficiencia y precisión en la gestión de sus operaciones internas.

El tema del presente trabajo de investigación es el desarrollo e implementación de un módulo de control presupuestal y logístico para el pago de comprobantes en una universidad privada, utilizando el *framework* ágil Scrum. Este módulo se enfoca en la digitalización del proceso de contabilización de comprobantes entre una empresa asociada y la universidad, empleando archivos XML para la extracción de datos y microservicios para la integración con el sistema ERP existente.

A partir del análisis de sistemas de información y estudios previos sobre digitalización de procesos contables, se ha desarrollado una solución que permite reducir el tiempo de procesamiento y la carga administrativa, mejorando la precisión y eficiencia en la gestión de comprobantes. Los resultados obtenidos han demostrado que el uso de Scrum como *framework* de desarrollo facilita la iteración y la adaptación continua del sistema, asegurando que se cumplan las necesidades de los usuarios y se optimice el flujo de trabajo.

El presente trabajo de investigación, considerando la importancia de la precisión en la gestión financiera, ha sido sometido a rigurosas pruebas y evaluaciones, cuyos resultados se reflejan en la documentación técnica de los procesos desarrollados. No obstante, debido a la naturaleza dinámica de las actividades institucionales, el sistema puede estar sujeto a ajustes y mejoras continuas para mantener su eficacia y eficiencia en la prestación de servicios a los usuarios y beneficios para la institución.

Palabras clave: XML, digitalización, microservicios, Scrum.

ABSTRACT

Currently, the digitization of administrative and financial processes in educational institutions is advancing significantly due to the adoption of new technologies and agile methodologies. This progress has allowed various private institutions to improve the efficiency and accuracy in the management of their internal operations.

The topic of this research work is the development and implementation of a budgetary and logistical control module for the payment of invoices in a private university, using the agile framework Scrum. This module focuses on the digitization of the invoicing process between an associated company and the university, employing XML files for data extraction and microservices for integration with the existing ERP system.

Based on the analysis of information systems and previous studies on the digitization of accounting processes, a solution has been developed that reduces processing time and administrative workload, improving the accuracy and efficiency in invoice management. The results obtained have demonstrated that using Scrum as a development framework facilitates the iteration and continuous adaptation of the system, ensuring that user needs are met and the workflow is optimized.

This research work, considering the importance of accuracy in financial management, has undergone rigorous tests and evaluations, with results reflected in the technical documentation of the developed processes. However, due to the dynamic nature of institutional activities, the system may be subject to continuous adjustments and improvements to maintain its effectiveness and efficiency in providing services to users and benefits to the institution.

Keywords: XML, digitization, microservices, Scrum.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Las instituciones educativas privadas hoy en día no se limitan únicamente a la enseñanza, sino que también han adoptado un enfoque administrativo y financiero avanzado para mejorar la calidad de los servicios que ofrecen. La necesidad de una gestión administrativa más eficiente ha impulsado la adopción de nuevas tecnologías y metodologías ágiles, lo que ha permitido una mayor precisión y eficiencia en sus operaciones internas.

El sistema burocrático que aún persiste en algunas universidades privadas demanda un cambio radical en la gestión de comprobantes y el control presupuestal. La implementación de un módulo de control presupuestal y logístico para el pago de comprobantes se presenta como una solución imprescindible para digitalizar y optimizar estos procesos. Este proyecto de investigación se centra en el desarrollo de dicho módulo utilizando el *framework* ágil Scrum, con el objetivo de mejorar la precisión y eficiencia en la gestión de comprobantes entre una empresa asociada y la universidad privada.

Para lograrlo, se utilizarán archivos XML para la extracción de datos y microservicios para la integración con el sistema ERP existente. La digitalización del proceso de contabilización de comprobantes no solo reducirá el tiempo de procesamiento y la carga administrativa, sino que también mejorará la precisión y transparencia de los datos financieros.

El uso de Scrum como *framework* de desarrollo facilita la iteración y la adaptación continua del sistema, asegurando que se cumplan las necesidades de los usuarios y se optimice el flujo de trabajo. Además, la incorporación de microservicios permitirá desarrollar aplicaciones mediante un conjunto de servicios independientes y escalables, que podrán evolucionar y adaptarse a las necesidades del sistema de manera efectiva.

El presente trabajo de investigación tiene por objetivo principal implementar un módulo de control presupuestal y logístico para el pago de comprobantes en una universidad privada, utilizando el *framework* Scrum. Y como objetivos específicos se tiene los siguientes:

- Reducir el tiempo de procesamiento y contabilización de comprobantes en la universidad privada mediante la digitalización del control presupuestal y logístico del pago de comprobantes.

- Implementar las funcionalidades necesarias en el nuevo módulo como la extracción de datos desde archivos XML y la generación de asientos contables utilizando microservicios.
- Agilizar el flujo de trabajo de aprobación de comprobantes en el área de autoridades correspondiente.

Este trabajo de investigación está compuesto por cuatro capítulos cuyo contenido está organizado de la siguiente manera:

- Capítulo I: Descripción del proyecto. Incluye toda la información base del proyecto desarrollado y el marco teórico.
- Capítulo II: Metodología. Explica las metodologías utilizadas y cómo se implementaron, así como el desarrollo del marco de trabajo.
- Capítulo III: Implementación del módulo de control presupuestal y logístico en una universidad privada.
- Capítulo IV: Resultados y discusión. Presenta los resultados de las pruebas obtenidas de los servicios implementados y la discusión de los resultados.

CAPÍTULO I: DESCRIPCIÓN DEL PROYECTO

1. Descripción y Objetivos del Proyecto

1.1 Descripción del Problema

En la universidad privada al que hace referencia este proyecto, el control presupuestal y logístico para el pago de comprobantes entre empresas asociadas y el área de contabilidad de esta universidad ha sido históricamente efectivo pero complejo, involucrando múltiples interesados y etapas administrativas. Este procedimiento requería que el área usuaria (empresas asociadas) registrara su requerimiento en el módulo de requerimientos del ERP propio de la universidad, subiera la información de su comprobante y sea enviado al área de logística. A continuación, la autoridad administrativa debía aprobar el requerimiento, tras lo cual el área de logística generaba la orden de compra/servicio y la enviaba al área de presupuesto. Una vez que el área de presupuesto realizaba la afectación presupuestal, el documento retornaba a logística para ser firmado por el jefe de logística y la autoridad administrativa. Solo entonces, logística generaba una orden interna de logística (OL) y la enviaba al área de contabilidad, donde finalmente se generaban los asientos contables y se realizaba el desembolso a las empresas asociadas.

Este proceso, aunque eficaz, implicaba la participación de numerosos actores y varias etapas, lo que podía dar lugar a demoras y a una alta carga administrativa. Cada etapa del proceso requería la intervención de distintas áreas, desde la solicitud inicial por parte de la empresa asociada, pasando por la aprobación de la autoridad administrativa, hasta la generación y firma de órdenes de compra y su posterior gestión por parte del área de presupuesto. Este flujo de trabajo, al depender de la intervención manual en múltiples puntos, era susceptible a retrasos debido a la disponibilidad de los responsables y la carga de trabajo de cada área.

Además, la centralización de la aprobación en la autoridad administrativa limitaba la flexibilidad y agilidad del sistema. Al depender de una sola persona para la aprobación final, cualquier atraso o indisponibilidad de esta figura podía causar demoras significativas. Este enfoque también limitaba la capacidad de las autoridades correspondientes no administrativas (académicas y de investigación) para gestionar y aprobar sus propios consumos, lo que podía generar descoordinación y falta de autonomía. La necesidad de múltiples firmas y aprobaciones en diferentes etapas también aumentaba el riesgo de errores administrativos y duplicación de esfuerzos, lo que a su vez incrementaba la carga administrativa sobre el personal involucrado.

La complejidad del proceso manual también dificultaba la trazabilidad y seguimiento de los comprobantes, haciendo más complicado identificar y resolver rápidamente cualquier discrepancia o problema que pudiera surgir. La dependencia en procesos manuales, además de ser menos eficientes, aumentaba la probabilidad de errores humanos, como la pérdida de documentos, errores de transcripción o retrasos en la comunicación entre las áreas. Esta situación no solo impactaba negativamente en la eficiencia operativa, sino que también podía afectar la precisión y confiabilidad de la información financiera, crucial para la toma de decisiones estratégicas en la universidad privada.

1.2 Objetivos del Proyecto

1.2.1 Objetivo General

Implementar un módulo de control presupuestal y logístico para el pago de comprobantes que digitalice la contabilización de comprobantes entre la empresa asociada y la universidad privada, utilizando el *framework* de Scrum.

1.2.2 Objetivos Específicos

- Reducir el tiempo de procesamiento y contabilización de comprobantes en la universidad privada mediante la digitalización del control presupuestal y logístico del pago de comprobantes.
- Implementar las funcionalidades necesarias en el nuevo módulo, como la extracción de datos desde archivos XML y la generación de asientos contables utilizando microservicios.
- Agilizar el flujo de trabajo de aprobación de comprobantes en el área de autoridades correspondiente.

1.3 Alcances y Limitaciones

1.3.1 Alcances

El alcance del proyecto está enfocado en el desarrollo e implementación de un módulo de control presupuestal y logístico para el pago de comprobantes en una universidad privada, utilizando el *framework* ágil Scrum. Este módulo permitirá la digitalización del proceso de

contabilización de comprobantes entre una empresa asociada y la universidad, empleando archivos XML para la extracción de datos y microservicios para la integración con el sistema ERP propio de la universidad privada.

El proyecto busca reducir el tiempo de procesamiento y la carga administrativa, implementando funcionalidades clave como la extracción automática de datos desde archivos XML emitidos por la SUNAT (Superintendencia Nacional de Aduanas y de Administración Tributaria) y la generación de asientos contables de manera eficiente. La implementación se realizará utilizando tecnologías ya disponibles en la universidad como Python y PHP, evitando costos elevados adicionales.

En un contexto en donde la agilidad y precisión en la gestión financiera son esenciales, este módulo proporcionará una solución innovadora y práctica. Además, al implementar un enfoque basado en Scrum, se asegurará que el desarrollo sea iterativo y colaborativo, permitiendo ajustes y mejoras continuas basadas en el *feedback* de los usuarios.

El módulo contribuirá a la digitalización y mejora de los procesos administrativos y financieros de la universidad privada, proporcionando una mayor seguridad y control en la gestión de los comprobantes, y asegurando que los datos sean precisos y fiables. Este alcance incluye la implementación completa del módulo, pruebas rigurosas y evaluaciones para garantizar su efectividad y eficiencia en el entorno institucional.

1.3.2 Limitaciones

- El presente proyecto desarrolla la digitalización del proceso de contabilización de comprobantes específicamente en una universidad privada, optimizando la gestión entre una empresa asociada y el área de contabilidad de la universidad.
- El desarrollo del módulo de control presupuestal y logístico sutilizará tecnologías como Python y PHP para la extracción de datos desde archivos XML y la implementación de microservicios.

- La extracción y procesamiento de datos se realizará utilizando archivos XML emitidos por la SUNAT, asegurando que los datos sean precisos y fiables para su integración con el sistema ERP de la universidad.
- El proyecto se desarrollará en el contexto de una universidad privada ubicada en la ciudad de Arequipa, Perú, durante el año 2024.
- La implementación del módulo utilizará tecnologías y recursos disponibles en la universidad, evitando costos adicionales elevados. El proyecto se beneficiará del uso del *framework* ágil Scrum, pero se limitará a las herramientas y metodologías que ya son parte del entorno tecnológico de la universidad.

1.4 Estado del Arte

Existen distintos enfoques con respecto a este proyecto, la mayoría se centra en el desarrollo de sistemas de información o sistemas web, y también la automatización de procesos proporcionando una base sólida para la implementación de sistemas eficientes y precisos.

El estudio de (Ortiz, 2022), se centra en el desarrollo de una herramienta que automatiza la recepción y procesamiento de facturas electrónicas utilizando archivos XML, reduciendo significativamente el trabajo administrativo y minimizando errores humanos.

(Huarcaya, 2022) aborda la optimización del proceso de aprobación de facturas, destacando la importancia de mejorar la comunicación y la eficiencia en la evaluación de facturas negociables. Sus resultados proporcionan valiosos *insights* sobre la implementación de procesos eficientes y la modernización de sistemas financieros, alineándose con los objetivos del presente proyecto.

En el campo de la automatización de sistemas contables, (Leon & Villagaray, 2023), presentan una propuesta centrada en la automatización del registro de compras, utilizando plantillas en Excel para facilitar la carga de datos y reducir errores manuales. Similarmente, el presente proyecto busca un control presupuestal y logístico para el pago de comprobantes y

generar automáticamente asientos contables en el sistema financiero de universidad privada, compartiendo el objetivo de mejorar la eficiencia y precisión del procesamiento de datos contables.

La tesis de (Parraga & Barreto, 2023) se enfoca en la mejora y automatización del sistema de gestión de facturas para garantizar la integridad de los datos y reducir errores en el registro contable. Este estudio proporciona un marco de referencia sólido para la implementación del presente proyecto, destacando la importancia de la integridad y precisión de los datos.

Finalmente, (Alvarado, 2020) desarrolla un sistema de información web para optimizar la gestión administrativa y contable, mejorando la eficiencia operativa y reduciendo el tiempo dedicado a tareas manuales. De manera similar, el presente proyecto busca digitalizar la gestión de comprobantes mediante la extracción de datos XML y la generación automática de asientos contables, resaltando la importancia de integrar soluciones tecnológicas para mejorar la precisión y eficiencia en la gestión de datos

1.5 Fundamentos Teóricos

1.5.1 Scrum

Scrum se define como un marco de trabajo en el que las personas pueden abordar problemas complejos, mientras mantienen una alta productividad en la entrega de productos de alta calidad (Zayat & Senvar, 2020).

1.5.2 Archivos XML (Extensible Markup Language)

Un archivo XML es un archivo de lenguaje de marcado extensible y se utiliza para estructurar datos para su almacenamiento y transporte. En un archivo XML, hay etiquetas y texto. Las etiquetas proporcionan la estructura a los datos. El texto del archivo que desea almacenar está rodeado por estas etiquetas, que se adhieren a pautas de sintaxis específicas. En esencia, un archivo XML es un archivo de texto estándar que utiliza etiquetas personalizadas

para describir la estructura del documento y cómo debe almacenarse y transportarse (Martel Solis, 2021).

1.5.3 Comprobante electrónico

Regulado por SUNAT, es usado en la venta de bienes y/o prestación de servicios. Contiene información del emisor electrónico, usuario o adquirente, número, tipo de operación, fecha y hora de emisión, ítems, operaciones gravadas y/o inafectas y/o exoneradas y/o gratuitas, valor de venta total, importe de venta total, entre otras (Bravo, 2022).

1.5.4 *Enterprise Resource Planning* (ERP)

Un sistema ERP es aquel software que se usa en las organizaciones para gestionar las actividades del día a día que forman parte del negocio, como las referentes a contabilidad, compras, logística, entre otras (Bobadilla, 2022).

1.5.5 Microservicios

Los microservicios se conciben como un estilo arquitectónico enfocado a desarrollar aplicaciones mediante un conjunto de servicios, independientes, escalables, colaborativos, evolutivos, capaces de autoadaptarse a ecosistemas complejos (Mamani, Del Pino, & Gonzales, 2020).

1.6 Técnicas y Herramientas

1.6.1 Técnicas

a) Revision Documental

Esta técnica permitió reunir datos relevantes sobre la digitalización de procesos administrativos y financieros, la utilización de archivos XML para la extracción de datos, y el uso de microservicios.

1.6.2 Herramientas

a) Smarty

Smarty es un motor de plantillas para PHP que facilita el trabajo de diseñadores y desarrolladores, permitiendo que colaboren de manera más autónoma y eficaz.

b) HTML (*HyperText Markup Language*)

HTML es el lenguaje estándar utilizado para construir páginas web. Establece la estructura del contenido web a través del uso de elementos y etiquetas.

c) CSS

Es un lenguaje de estilos que se utiliza para definir la apariencia de un documento HTML. Permite gestionar el diseño y el formato visual de las páginas web.

d) JavaScript

Es un lenguaje de programación que facilita la creación de contenido interactivo en las páginas web, siendo ampliamente utilizado para mejorar la experiencia del usuario.

e) Bootstrap 5

Es un *framework* de diseño web que simplifica el desarrollo de sitios y aplicaciones web adaptables a diferentes dispositivos.

f) JQuery

Es una biblioteca de JavaScript que facilita la manipulación del DOM, la gestión de eventos, la animación y las solicitudes AJAX.

g) AJAX

Es una técnica de desarrollo web que permite actualizar contenido de manera asíncrona sin necesidad de recargar toda la página.

h) PHP

Es un lenguaje de programación de propósito general, lo que significa que está diseñado para ser utilizado en una amplia variedad de aplicaciones y contextos, que se utiliza ampliamente para el desarrollo web.

i) Python

Es un lenguaje de programación diseñado para múltiples propósitos. Como lenguaje de alto nivel, su sintaxis es cercana al lenguaje humano, lo que facilita la comprensión y escritura del código. Es ampliamente reconocido por su claridad y por contar con una extensa biblioteca estándar.

j) ElementTree

Es una biblioteca de Python utilizada para manipular y procesar archivos XML. Facilita la creación, modificación y exploración de documentos XML de forma sencilla.

k) FastAPI

Es un *framework* web moderno y de alto rendimiento para Python, es conocido por su rapidez y eficacia en la creación de APIs.

l) Uvicorn

Es un servidor ASGI (*Asynchronous Server Gateway Interface)* para Python, creado para proporcionar alto rendimiento y bajo consumo de recursos, ideal para aplicaciones asíncronas como las desarrolladas con FastAPI.

m) PostgreSQL

Es un sistema de gestión de bases de datos relacional de código abierto, reconocido por su estabilidad, capacidad de extensión y adherencia al estándar SQL.

n) PgAdmin4

Herramienta de código abierto que permite la administración de bases de datos en PostgreSQL, facilitando su administración.

o) VisualStudioCode

Es un editor de código fuente que soporta muchos lenguajes de programación y ofrece funciones como depuración, control de versiones y extensiones.

p) Jira

Herramienta de gestión de proyectos, ampliamente utilizada en equipos de desarrollo de software para implementar *frameworks* ágiles como Scrum.

CAPÍTULO II: METODOLOGÍA

En el presente capítulo se detalla la elección de las metodologías y tecnologías que sustentan el desarrollo del módulo de control presupuestal y logístico. Mediante cuadros comparativos, se explican las razones detrás de la selección de archivos XML como formato de datos de los comprobantes de pago y del *framework* Scrum como metodología ágil de desarrollo.

Además, se destaca la importancia de la ingeniería de requerimientos en el proceso de desarrollo, asegurando que las necesidades de los *stakeholders* sean capturadas, analizadas y gestionadas de manera efectiva. La herramienta Jira se utilizó para gestionar estas tareas y requerimientos, permitiendo una integración más eficiente con el sistema ERP existente y asegurando un flujo de trabajo organizado y transparente.

Para finalizar el capítulo, se detalla la estructura del marco de trabajo, explicando el *framework* Scrum y describiendo los roles, eventos y artefactos utilizados durante el desarrollo del proyecto.

2. Justificación y aporte del proyecto

2.1 Contribución y valor del proyecto

2.1.1 Justificación Teórica

El desarrollo de un módulo de control presupuestal y logístico para el pago de comprobantes en una universidad privada utilizando el *framework* ágil Scrum aportó significativos beneficios a la gestión administrativa y financiera de la institución. Este módulo permitió digitalizar el proceso de contabilización de comprobantes, optimizando el tiempo de procesamiento y reduciendo la carga administrativa.

Este proyecto no solo benefició a la universidad privada en cuestión, sino que también sirvió como un modelo de innovación tecnológica en la gestión administrativa y financiera en el ámbito educativo. Al permitir una mejor gestión de los comprobantes y una mayor transparencia en los procesos contables, el módulo contribuirá a una administración más eficiente y confiable.

2.1.2 Justificación Técnica

El proyecto se sustenta técnicamente en la elección de un conjunto de herramientas y metodologías que garantizan la eficiencia, escalabilidad y seguridad del sistema a desarrollar.

Centrándose en el tema de la tecnología de la información, Scrum es uno de los marcos innovadores para desarrollar productos y servicios (Pramitya & Eko, 2020). El uso del *framework* Scrum está justificado por su capacidad para gestionar proyectos complejos de manera iterativa y adaptable, lo que asegura que el desarrollo del módulo pueda responder a cambios y mejoras continuas basadas en el *feedback* del usuario. La herramienta Jira será utilizada para la implementación de Scrum, lo que permitirá una planificación y seguimiento detallado del progreso del proyecto, facilitando la coordinación del equipo y asegurando que se cumplan los plazos y objetivos establecidos.

En cuanto al desarrollo del *frontend*, la elección de Smarty, HTML, CSS y JavaScript se fundamenta en su capacidad para crear interfaces de usuario dinámicas y responsivas, esenciales para ofrecer una experiencia de usuario fluida. Herramientas adicionales como Bootstrap 5, jQuery y AJAX son cruciales para garantizar que la interfaz se adapte a diferentes dispositivos y que las interacciones sean rápidas y eficientes, mejorando la usabilidad y la accesibilidad del sistema.

Para el *backend* y los microservicios, la combinación de PHP y Python está justificada por su fiabilidad y robustez en la creación de aplicaciones web escalables. La librería ElementTree de Python es especialmente adecuada para la manipulación de datos XML, una tarea crítica en la gestión de comprobantes, mientras que FastAPI y Uvicorn proporcionan un entorno de microservicios ágil y eficiente, asegurando una comunicación segura y de alto rendimiento entre los distintos componentes del sistema.

El uso de PostgreSQL como sistema de gestión de bases de datos se basa en su capacidad para manejar grandes volúmenes de datos de manera estable y eficiente, garantizando la integridad y seguridad de la información de los comprobantes. pgAdmin4 será el gestor utilizado para administrar la base de datos, permitiendo un control detallado y una administración eficaz de los datos almacenados.

Finalmente, la elección de Visual Studio Code como entorno de desarrollo integrado (IDE) se justifica por su flexibilidad y capacidad para soportar tanto el desarrollo *frontend* como *backend*, proporcionando herramientas avanzadas que facilitan la escritura y depuración de código, lo que contribuye a la calidad y eficiencia del desarrollo del proyecto.

2.1.3 Justificación Operativa

- Digitalización de Procesos: El módulo agilizará la gestión de pagos y comprobantes en la universidad, reduciendo tiempos y errores administrativos.

- Extracción Automática de Datos: Utilizar archivos XML con ElementTree permitirá una extracción precisa y eficiente de datos de comprobantes.
- Integración Eficiente: Los microservicios con FastAPI asegurarán una comunicación segura y escalable entre los componentes del sistema ERP.
- Tecnologías Modernas: PostgreSQL gestionado con pgAdmin4 y Visual Studio Code facilitarán el manejo robusto de datos y el desarrollo efectivo del proyecto.
- Desarrollo Ágil: Scrum permitirá un desarrollo iterativo, adaptativo y colaborativo, cumpliendo con las necesidades específicas de los usuarios.

2.1.4 Importancia e Impacto

La importancia e impacto del desarrollo de un módulo de control presupuestal y logístico para el pago de comprobantes en una universidad privada radica en los beneficios que traerá para la gestión administrativa y financiera de la universidad privada. Este módulo permitirá una mayor eficiencia y precisión en la contabilización de comprobantes, optimizando los tiempos de procesamiento y reduciendo la carga administrativa.

Para ello, se plantea el desarrollo de un sistema que utilice archivos XML para la extracción de datos y microservicios para la integración con el sistema ERP existente, implementando Scrum. Este *framework*, conocida por su enfoque iterativo y colaborativo, facilitará la adaptación continua del sistema a las necesidades de los usuarios, mejorando así la calidad del proceso.

Los principales beneficiados de este proyecto serán las autoridades y el personal administrativo de la universidad, quienes experimentarán una notable mejora en la gestión de comprobantes y un aumento en la precisión de los datos financieros. Además, la digitalización del proceso contribuirá a una mayor transparencia y confiabilidad en la toma de decisiones estratégicas dentro de la institución.

2.2 Comparativa entre formato de datos

El presente proyecto utilizó el formato XML para la extracción automática de datos de los comprobantes electrónicos que emite la empresa asociada a la universidad privada, a continuación, se muestra un cuadro comparativo donde se muestran las principales diferencias con otros formatos de emisión que permite la SUNAT.

Tabla 1
Cuadro comparativo de formato de datos

Característica	XML	PDF
Estructura	Altamente estructurado con etiquetas anidadas	Menos estructurado, presenta el documento completo visualmente.
Legibilidad	Principalmente legible por máquinas, legible por humanos con herramientas específicas	Altamente legible para humanos, sin necesidad de herramientas adicionales.
Estándar	Amplio uso para transmisión de datos estructurados, soportado por la SUNAT para comprobantes electrónicos	Ampliamente utilizado para presentación visual, consistencia en la apariencia
Validación	Soporta validación mediante DTD y XML Schema, asegurando integridad de datos	No soporta validación estructural, pero puede contener firmas digitales para autenticidad
Ventajas	Facilita la extracción y procesamiento de datos	Fácilmente legible y accesible para cualquier usuario
Desventajas	No es fácilmente legible para usuarios sin herramientas específicas	Dificultad para la extracción automática de datos

Compatibilidad	Alta, especialmente con aplicaciones de gestión financiera.	Baja, principalmente utilizado para visualización

Nota. Actualmente la SUNAT maneja estos 2 formatos de emisión.

2.3 Comparativa entre metodologías de desarrollo

El presente proyecto utilizó Scrum como *framework* de desarrollo ágil, a continuación, se muestra un cuadro comparativo donde se muestran las principales diferencias con otras metodologías de desarrollo.

Tabla 2
Cuadro comparativo Scrum vs Cascada

Característica	Scrum	Cascada
Estructura	Iterativa e incremental, permite cambios y mejoras continuas	Lineal y secuencial, cada frase debe completarse antes de pasar a la siguiente
Flexibilidad	Alta, se adapta a los cambios y necesidades del proyecto durante el ciclo de desarrollo	Baja, los cambios son difíciles de implementar una vez iniciada la fase siguiente
Fases del Proyecto	Dividido en *Sprints*, incluye planificación, revisión y retrospectiva	Fases claramente definidas: requisitos, diseño, implementación, verificación y mantenimiento
Planificación	Flexible, se puede ajustar según las necesidades y el *feedback* de los usuarios	Rígida, toda la planificación se realiza al inicio del proyecto
Entrega de Producto	Entrega continua de incrementos del producto	Entrega al final del ciclo de desarrollo
Retroalimentación	Continua, permite ajustes y mejoras basadas en el *feedback*	Limitada, se obtiene al final de cada fase

2.4 Objetivos

A continuación, se mencionan nuevamente los objetivos del proyecto y su correspondiente validación en el trascurso del desarrollo de este.

Tabla 3
Validación de Objetivos

Objetivo General	
Objetivo	Validación
Implementar un módulo de control presupuestal y logístico para el pago de comprobantes que digitalice la contabilización de comprobantes entre la empresa asociada y la universidad privada, utilizando el *framework* de Scrum.	El objetivo se valida en el Capítulo III, donde se presentará la implementación del módulo en su totalidad.
Objetivos Específicos	
Objetivo	Validación
Reducir el tiempo de procesamiento y contabilización de comprobantes en la universidad privada mediante la digitalización del control presupuestal y logístico del pago de comprobantes.	El objetivo se valida en el Capítulo IV, donde se menciona el tiempo empleado antes y después de la implementación del módulo.
Implementar las funcionalidades necesarias en el nuevo módulo, como la extracción de datos desde archivos XML y la generación de asientos contables utilizando microservicios.	El objetivo se valida en el Capítulo III, donde se muestra como se obtiene información crítica de los comprobantes en archivos XML a través de solicitudes JSON
Agilizar el flujo de trabajo de aprobación de comprobantes en el área de autoridades correspondiente.	El objetivo se valida en el Capítulo IV, donde se hace referencia al proceso de negocio antes y después de la implementación del módulo.

2.5 Scrum y la Ingeniería de requerimientos

La ingeniería de requerimientos juega un papel importante en el éxito del ciclo de vida de desarrollo de software. Scrum como un *framework* ágil ganó gran atención debido a su capacidad para afrontar entornos de desarrollo cambiante beneficiándose de la ingeniería de requerimientos (Ramadan & Megahed, 2016).

En el marco de Scrum, la gestión de requerimientos se lleva a cabo de manera continua y flexible, lo que significa que los requerimientos no se establecen por completo al inicio del proyecto, sino que se desarrollan y refinan a lo largo del ciclo de vida del producto. Este enfoque permite que los requerimientos sean revisados y ajustados continuamente, adaptándose a nuevas demandas y cambios en el entorno del proyecto.

La ingeniería de requerimientos en Scrum se gestiona a través del *Product Backlog*, cuyo encargado es el *Product Owner* quien se encarga de mantener y priorizar este backlog, asegurando que refleje las necesidades de los interesados. En Scrum, los requerimientos suelen capturarse mediante historias de usuario, que describen las funcionalidades desde la perspectiva del usuario final, facilitando una comprensión clara y compartida de las expectativas. Estas historias se priorizan según su valor para el negocio, el costo estimado, el riesgo y su relación con otras tareas.

2.6 Estructura de Scrum

Cabe recalcar que Scrum está definido en su totalidad en la Guía de Scrum, la cual se mantiene de manera independiente a cualquier empresa o tecnología específica (Schwaber & Sutherland, Scrum.org, 2024). Por lo tanto, solo funciona de manera efectiva cuando se implementa siguiendo los principios, roles, eventos y artefactos descritos en dicha guía, sin modificaciones que comprometan su esencia y enfoque en la mejora continua y el trabajo en equipo. Scrum no es simplemente un conjunto de procesos o ceremonias, sino un marco de trabajo que requiere la adopción plena de sus roles, eventos y artefactos para maximizar la

eficiencia y efectividad del equipo de desarrollo. La disciplina en la implementación de Scrum, incluyendo la correcta realización de las reuniones diarias, la planificación de *sprints*, las revisiones y retrospectivas, es esencial para garantizar que se alcancen los objetivos del proyecto y se obtenga el valor máximo del trabajo realizado. Sin esta adherencia estricta, los beneficios potenciales de Scrum pueden verse comprometidos, resultando en una menor calidad del producto y una pérdida de la colaboración y auto-organización del equipo.

2.7 Roles en Scrum

Scrum define tres roles principales: el Product Owner, el Scrum Master y el Equipo de Desarrollo. Dado el contexto del presente trabajo de investigación, mi persona tomó las responsabilidades del Product Owner y formó parte del Equipo de Desarrollo.

a) Product Owner

Responsabilidades:

- Gestión del Product Backlog: Esto incluye la creación, ordenación y clarificación de los elementos del backlog.
- Definición de prioridades: Priorizar los elementos de Product Backlog para maximizar el valor del trabajo realizado por el equipo de desarrollo.
- Comunicación con *Stakeholders*: Actuar como el punto principal de contacto entre el equipo Scrum y los *Stakeholders* (en este caso los *Stakeholders* son el área de contabilidad de la empresa asociada encargada de emitir las facturas, el área de contabilidad de la universidad privada y las principales autoridades académicas, administrativas y de investigación de la universidad).

b) Scrum Master

Responsabilidades:

- Facilitación del Scrum: Ayuda a todos a entender y adoptar Scrum, asegurando que el equipo Scrum siga los principios y prácticas del *framework*

- Eliminación de Impedimentos: Trabaja para eliminar cualquier impedimento que pueda obstaculizar el progreso del equipo de desarrollo.
- Coaching y Soporte: Proporciona coaching al equipo de desarrollo para que se autoorganicen y alcancen el máximo rendimiento.
- Protección del Equipo: Protege al equipo de interrupciones externas y facilita las reuniones Scrum

c) Equipo de Desarrollo

Responsabilidades:

- Auto-organización: El equipo de desarrollo se auto-organiza para completar el trabajo del sprint sin necesidad de supervisión directa
- Multifuncionalidad: Posee todas las habilidades necesarias para crear un incremento de producto funcional al final de cada sprint.
- Responsabilidad Colectiva: Todos los miembros del equipo son colectivamente responsables de cumplir con los objetivos del sprint y entregar un incremento de producto de alta calidad.

2.8 Ceremonias de Scrum

a) Sprint Planning

- **Propósito:** Planificar el trabajo que se realizará durante el sprint
- **Participantes:** El equipo Scrum
- **Salida:** Sprint Goal, Sprint Backlog.

b) Daily Scrum

- **Propósito:** Revisar el progreso hacia el Sprint Goal y ajustar el plan de trabajo.
- **Participantes:** Equipo de Desarrollo, facilitado por el Scrum Master
- **Formato:** Reunión de 15 minutos, realizada diariamente.

c) Sprint Review

- **Propósito:** Revisar el trabajo completado y obtener *feedback* de los *stakeholders.*
- **Participantes:** Equipo Scrum y *stakeholders*
- **Salida:** Incremento de producto potencialmente entregable, *feedback* para mejoras.

d) Sprint Retrospective

- **Propósito:** Reflexionar sobre el sprint pasado y planificar mejoras para el próximo Sprint.
- **Participantes:** Equipo Scrum.
- **Salida:** Plan de acción para mejoras continuas

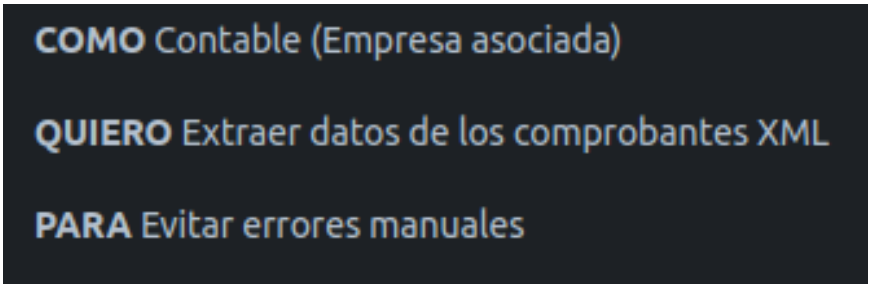
2.9 Historias de Usuario

El enfoque de la metodología ágil está en la satisfacción del cliente a través de la entrega temprana y continua de software valioso. En un entorno ágil, comúnmente conocido como Agile, los requerimientos se representan en forma de historias de usuario (Pokharel & Vaidyam, 2020). Las historias de usuario son una herramienta de comunicación para describir funcionalidades de software desde la perspectiva del usuario final.

2.9.1 Formato de Historias de Usuario

Las historias de usuario en el presente proyecto utilizaron una notación específica que sigue el formato: **Como [rol], QUIERO [acción], PARA [beneficio].** Este formato asegura que se identifique claramente quien es el usuario, qué necesita hacer y por qué es importante.

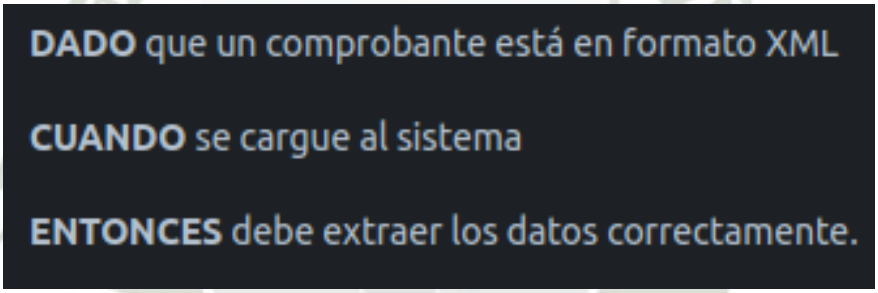
Figura 1
Ejemplo de Historia de Usuario utilizado en el proyecto



Para una mayor completitud, las historias de usuario debieron cumplir condiciones específicas las cuáles son nombradas criterios de aceptación. Estas se redactaron usando la notación **DADO [contexto], CUANDO [acción], ENTONCES [resultado]**.

Figura 2

Ejemplo de Criterios de aceptación utilizado en el proyecto



Nota. Ejemplo de Criterios de aceptación utilizado en el proyecto

2.9.2 Relación entre historias de usuario y el Product Backlog

El Product Backlog es una lista priorizada del trabajo a realizar en el proyecto. Esta lista incluye todas las historias de usuario, así como otros tipos de trabajo como *bugs* y potenciales mejoras. Las historias de usuario en el Product Backlog se ordenan según la prioridad determinada por el valor que aportan al usuario, la urgencia y el esfuerzo requerido.

Figura 3

Parte del Product Backlog del proyecto

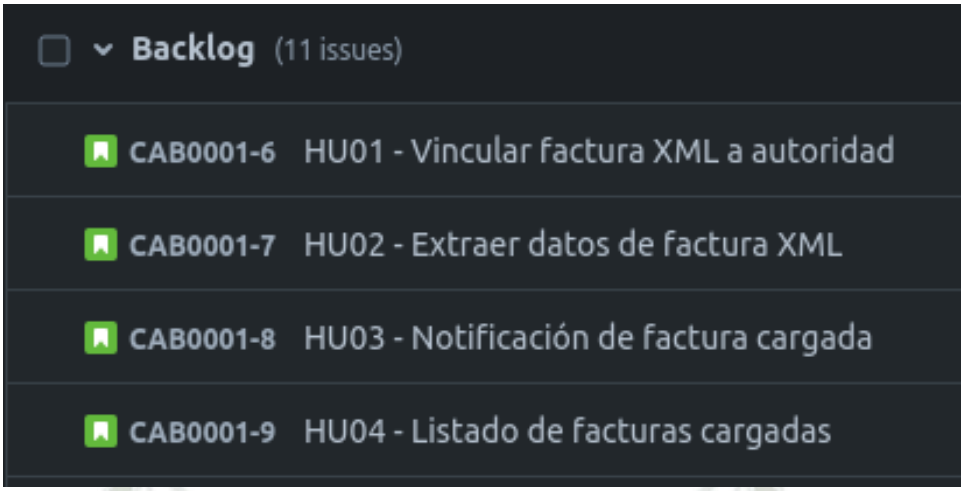


Figura 3 Parte del Product Backlog del proyecto

2.10 Proceso de Desarrollo con Scrum

2.10.1 Planificación del Sprint (Sprint Planning)

La planificación del *sprint* consistió en la colaboración del equipo Scrum en su totalidad para definir el Sprint Goal y seleccionar los elementos del Product Backlog que se moverán al Sprint Backlog.

Basándose en la complejidad y el esfuerzo estimado para cada historia de usuario, se planificaron cuatro *sprints* de dos semanas cada uno:

a) Sprint 1: Configuración Inicial y Funcionalidades Básicas

- **Sprint Goal:** Implementar la infraestructura básica del sistema y permitir la carga y vinculación de facturas en formato XML para el área de contabilidad de la empresa asociada.
- **Sprint Backlog:**
  - Desarrollo de la funcionalidad para ingresar el DNI de la autoridad y vincular la factura XML.
  - Implementación de la extracción de datos de las facturas XML cargadas.
  - Validación de la estructura del archivo XML y aseguramiento de la integridad de los datos.

b) Sprint 2*:* Notificaciones y Visualización de comprobantes

- **Sprint Goal:** Implementar las notificaciones y las funcionalidades de visualización de comprobantes cargados para las autoridades correspondientes de la universidad privada.
- **Sprint Backlog:**
  - Implementación del sistema de notificaciones por correo para facturas cargadas.
  - Desarrollo de la interfaz para listar las facturas cargadas a cada autoridad.
  - Implementación de la funcionalidad para ver los detalles específicos de una factura cargada.
  - Aseguramiento de la funcionalidad de filtrado y búsqueda en la lista de facturas.

c) Sprint 3: Aprobación de consumos y gestión de facturas aprobadas

- **Sprint Goal:** Permitir la aprobación y denegación de facturas, así como la visualización y gestión de facturas aprobadas.
- **Sprint Backlog:**
  - Desarrollo de la funcionalidad para que las autoridades aprueben o denieguen el consumo de facturas.
  - Implementación de la lista de facturas aprobadas y denegadas, con opciones de filtrado y búsqueda.
  - Actualización automática del saldo de la actividad correspondiente tras la aprobación o denegación de un comprobante.

d) Sprint 4: Generación y Guardado de Asientos Contables

- **Sprint Goal:** Implementar la generación y gestión de asientos contables, y actualizar el estado de los comprobantes para enviar a pago.

- **Sprint Backlog:**
    - Implementación de la funcionalidad para generar asientos contables automáticamente a partir de los comprobantes aprobados.
    - Desarrollo de la interfaz para editar y ajustar los detalles del asiento contable.
    - Implementación de la funcionalidad para guardar los asientos contables.

El Capítulo II ofreció una descripción exhaustiva del uso del *framework* Scrum en la implementación del módulo de control presupuestal y logístico destinado al pago de comprobantes en una universidad privada. Al identificar roles, ceremonias y artefactos, se estableció un marco enriquecido para la gestión del proyecto, que permite responder rápidamente a los cambios y mantiene un enfoque centrado en las necesidades del cliente. La planificación de los *sprints*, junto con la utilización de historias de usuario, promovió un desarrollo iterativo que garantiza que cada incremento del producto aporte valor tangible a la universidad. Además, los criterios de aceptación definidos para cada historia han sido esenciales para asegurar que las expectativas del equipo estén alineadas con los requisitos del proyecto, asegurando que cada entrega cumpla con los estándares de calidad deseados. Este enfoque sistemático no solo ha optimizado la eficiencia y la transparencia del proceso de desarrollo, sino que también ha mejorado la colaboración entre los miembros del equipo Scrum.

CAPÍTULO III: IMPLEMENTACIÓN DEL PROYECTO

En el presente capítulo se describe el proceso de implementación del módulo de control presupuestal y logístico para el pago de comprobantes utilizando el *framework* ágil de Scrum en una universidad privada. La implementación se llevó a cabo mediante un enfoque metódico que incorpora una arquitectura de sistema robusta, la integración de tecnologías avanzadas, y un enfoque en la interoperabilidad con el sistema ERP propio de la universidad privada.

Se exploran las tecnologías utilizadas, incluyendo el uso de lenguajes de programación como PHP y Python, y *frameworks* como FastAPI. Además, se abordan las funcionalidades clave del sistema, desde la vinculación de facturas XML hasta la generación de asientos contables, detallando cómo cada componente fue implementado para cumplir con los requisitos del proyecto.

Finalmente, se examinan las pruebas realizadas para asegurar el correcto funcionamiento del módulo, garantizando que se maximicen los beneficios operativos para la universidad.

3. Arquitectura del Sistema

La arquitectura del sistema para el módulo de control presupuestal y logístico para el pago de comprobantes en una universidad privada ha sido diseñada para ser robusta, escalable y eficiente, asegurando una integración efectiva con el sistema ERP propio de la universidad privada. Basada en el patrón de diseño Modelo-Vista-Controlador (MVC), la arquitectura proporciona una separación clara de responsabilidades.

3.1 Visión General de la Arquitectura del Sistema

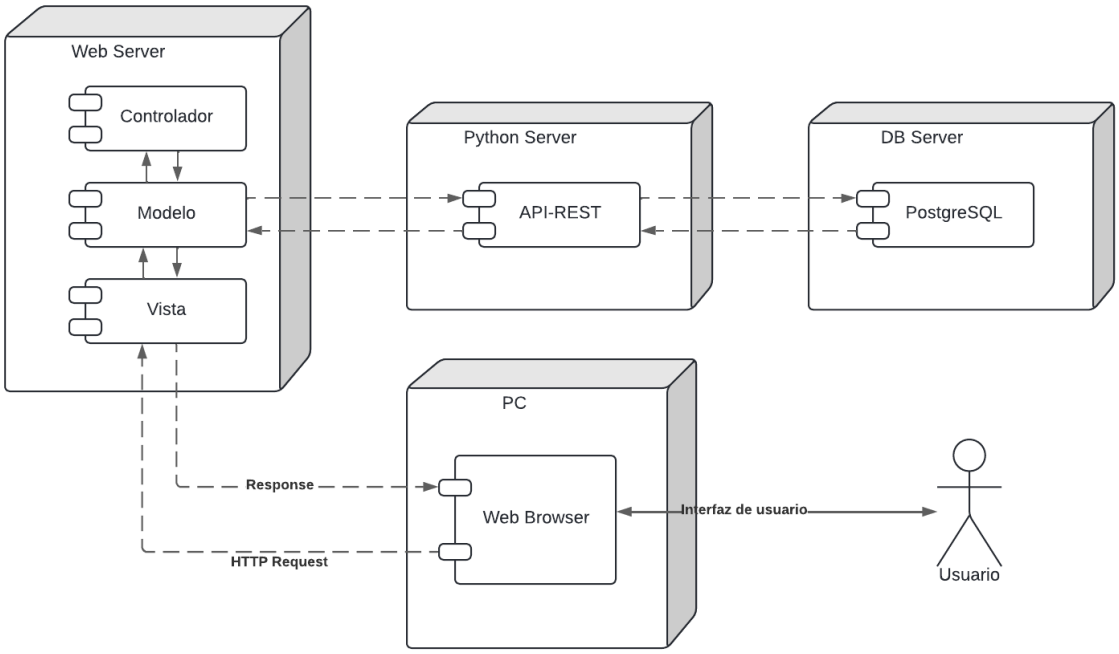
El sistema se organiza en tres capas principales:

- **Vista:** Encargada de la presentación y la interacción con el usuario final, la capa de vista utiliza tecnologías web modernas como HTML, CSS, JavaScript, Bootstrap y el motor de plantillas Smarty. Esta capa es responsable de renderizar las interfaces de usuario de manera dinámica y responsiva, permitiendo a los usuarios interactuar con el sistema de forma intuitiva.
- **Controlador:** Implementado principalmente en PHP, el controlador gestiona la lógica de negocio y actúa como intermediario entre la vista y el modelo. Este componente procesa las solicitudes de los usuarios, ejecuta la lógica necesaria y decide qué vistas mostrar.
- **Modelo:** El modelo encapsula la lógica de negocio y las operaciones de acceso a datos. En este sistema, el modelo está implementado en PHP y es responsable de gestionar las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) y la lógica de negocio relacionada con los datos de las facturas. El modelo también interactúa con la base de datos PostgreSQL para asegurar la persistencia y consistencia de la información. Además, el modelo se comunica con microservicios en Python para delegar ciertas tareas que requieren un procesamiento más especializado.

3.2 Diagrama de Despliegue

En el contexto actual, el diagrama de despliegue proporciona una representación clara de cómo se organizan los elementos del sistema en el entorno de ejecución como se muestra en la Figura 4.

Figura 4
Diagrama de despliegue de la arquitectura del sistema.



la arquitectura del sistema

El Servidor Web alberga tres componentes esenciales: la Vista, el Controlador y el Modelo. La Vista es la capa que gestiona la interacción con el usuario final, generando interfaces gráficas utilizando tecnologías como HTML, CSS, JavaScript, y Smarty. Este componente está diseñado para proporcionar una experiencia de usuario intuitiva y responsiva, permitiendo a los usuarios acceder fácilmente a las funcionalidades del sistema.

El Controlador es el núcleo de la lógica de negocio, actuando como intermediario entre la Vista y el Modelo. Implementado en PHP, el controlador procesa las solicitudes del usuario, ejecuta las reglas de negocio necesarias, y coordina el flujo de información entre las distintas capas del sistema. Este componente también es responsable de interactuar con microservicios

externos a través de llamadas cURL, permitiendo la delegación de tareas específicas que requieren procesamiento avanzado.

El Modelo maneja las operaciones de datos y la lógica de negocio asociada. También está implementado en PHP y se conecta a la base de datos PostgreSQL para realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar). Además, el modelo se comunica con microservicios alojados en un servidor Python para realizar tareas complejas que son mejor manejadas por servicios especializados.

El Servidor de Microservicios funciona como una extensión del sistema principal, proporcionando APIs RESTful que manejan procesos especializados. Estas APIs, desarrolladas en Python usando frameworks como FastAPI, permiten que el sistema ejecute operaciones complejas de manera eficiente y escalable. Al separar estas funcionalidades en microservicios, el sistema puede escalar de manera independiente según la demanda, optimizando los recursos disponibles.

La Base de Datos PostgreSQL, ubicada en el Servidor de Base de Datos, almacena toda la información crítica del sistema, garantizando la integridad y seguridad de los datos. PostgreSQL es elegido por su capacidad para manejar grandes volúmenes de datos y su robustez en el mantenimiento de la consistencia de la información.

Finalmente, la PC del Usuario representa el punto de interacción para los usuarios finales. A través de un navegador web, los usuarios acceden al sistema, donde pueden cargar y consultar información de facturas, gestionar comprobantes, y realizar otras operaciones financieras. El navegador envía solicitudes al servidor web y recibe respuestas que se presentan en forma de páginas web dinámicas.

3.3 Modelado de Procesos de Negocio

En este apartado se describe cómo los diagramas BPMN (*Business Process Management Notation*) han sido esenciales para modelar, analizar y optimizar los procesos de

negocio involucrados en el módulo de control presupuestal y logístico para el pago de comprobantes en una universidad privada. El uso de BPMN permite visualizar claramente los flujos de trabajo, y comunicar estos procesos de manera efectiva a todos los *stakeholders* del proyecto.

3.3.1 Importancia de los Diagramas BPMN en el Proyecto

En el contexto de este proyecto, BPMN ha sido fundamental para:

- Visualización de Procesos Complejos: Permitir una representación visual de los procesos relacionados con la gestión de comprobantes y presupuestos, haciendo más fácil identificar ineficiencias.
- Iteración y Adaptación: Permite al equipo Scrum iterar rápidamente sobre los procesos modelados, adaptándose al cambio en los requisitos y en la retroalimentación de los usuarios.
- Comunicación efectiva: Sirve como un lenguaje común entre el *Product Owner*, el *Scrum Master* y el equipo de desarrollo, mejorando la comunicación y colaboración.

3.3.2 Integración de BPMN con el Ciclo de *Sprints*

Durante la planificación de sprints, BPMN ayuda a definir claramente los procesos de negocio que deben ser desarrollados o mejorados. Los equipos pueden priorizar las historias de usuario que impactan directamente.

a) Proceso de Carga y Validación de Comprobantes XML

A continuación, se presenta el diagrama BPMN que ilustra el proceso de carga y validación de comprobantes XML. Este diagrama destaca cómo los procesos clave se alinean con las historias de usuario y se integran en el ciclo de *sprints*.

Este diagrama representa las interacciones entre el área de contabilidad de la empresa asociada a la universidad privada y el sistema ERP de la universidad.

Figura 5

Diagrama BPMN del Proceso de Carga y Validación de Comprobantes XML

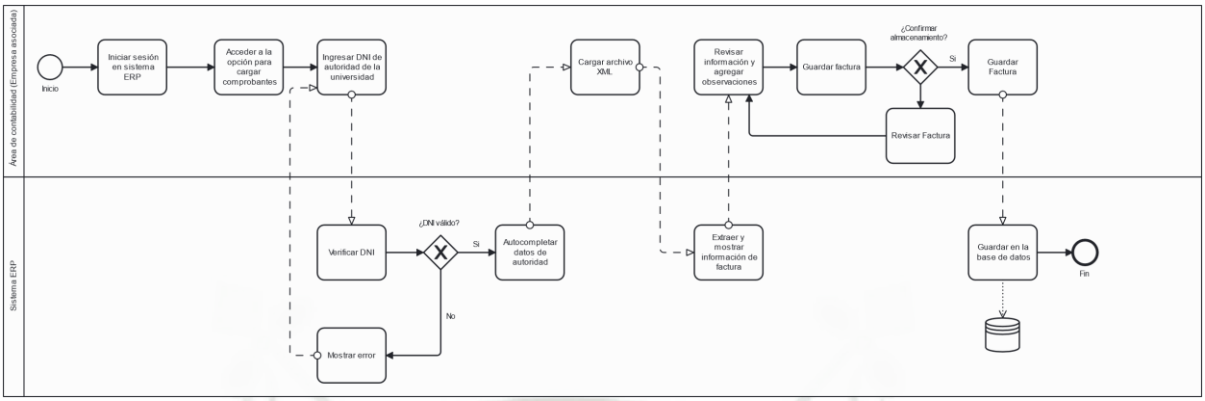


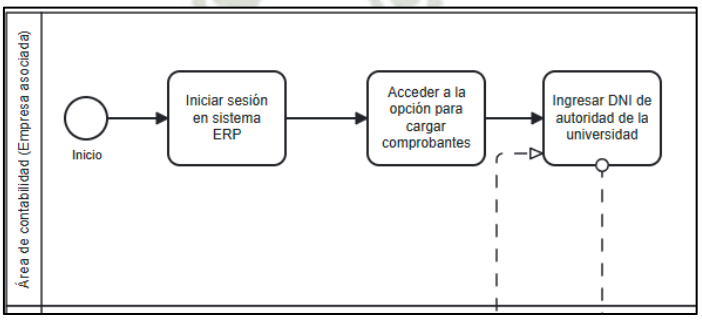
Diagrama BPMN del Proceso de Carga y Validación

El diagrama BPMN del proceso de carga y validación de comprobantes XML representa detalladamente el flujo de trabajo desde el inicio de sesión del usuario en el sistema ERP de la universidad hasta el almacenamiento del comprobante en la base de datos.

El proceso comienza con el inicio de sesión del usuario contable en el ERP, seguido de la selección de la opción "Subir Comprobante XML" en el menú específico para el área de contabilidad de la empresa asociada. El usuario ingresa el DNI de la autoridad universitaria, y el sistema verifica la validez del DNI.

Figura 5.1

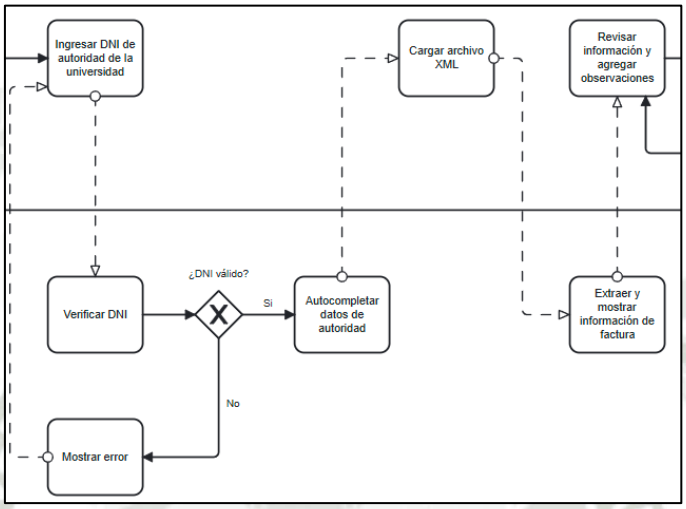
Fragmento de la Figura 5 relacionado con el acceso al sistema ERP y el ingreso del DNI de la autoridad universitaria.



Si el DNI es válido, el proceso continúa, permitiendo al usuario cargar el archivo XML del comprobante. El sistema ERP extrae automáticamente la información relevante del

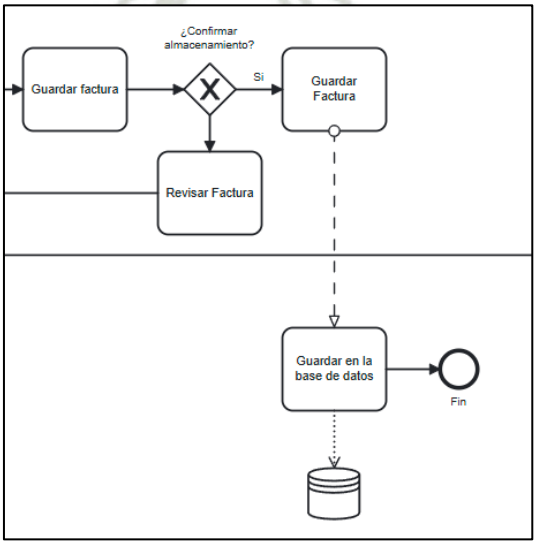
comprobante XML y la presenta al usuario para su revisión. El usuario puede añadir observaciones antes de proceder a guardar la información.

Figura 5.2
Fragmento de la Figura 5 relacionado con la validación de DNI de la autoridad y la carga del comprobante en formato XML.



Tras una confirmación final, el sistema almacena de forma segura la información del comprobante en la base de datos del ERP. Este flujo de trabajo asegura que solo se procesen y almacenen comprobantes válidos, mejorando la eficiencia y precisión del manejo de datos dentro del sistema universitario.

Figura 5.3
Fragmento de la Figura 5 relacionado con el almacenamiento del comprobante.

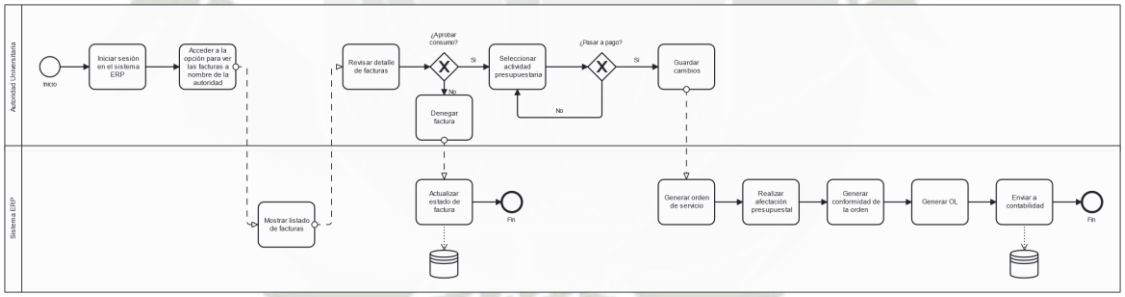


b) Proceso de Revisión de Autoridad correspondiente y afectación presupuestal

A continuación, se presenta el diagrama BPMN que ilustra el proceso de aprobación de facturas por parte de las autoridades universitarias. Este diagrama resalta cómo las decisiones tomadas por las autoridades se integran en el sistema ERP, asegurando una gestión eficiente y controlada de los presupuestos asignados. El diagrama destaca el flujo de trabajo desde la revisión de facturas hasta la afectación presupuestal y la generación de órdenes de servicio. Representa las interacciones entre las autoridades universitarias y el sistema ERP de la universidad, subrayando la importancia de la transparencia y precisión en la gestión de los recursos financieros.

Figura 6

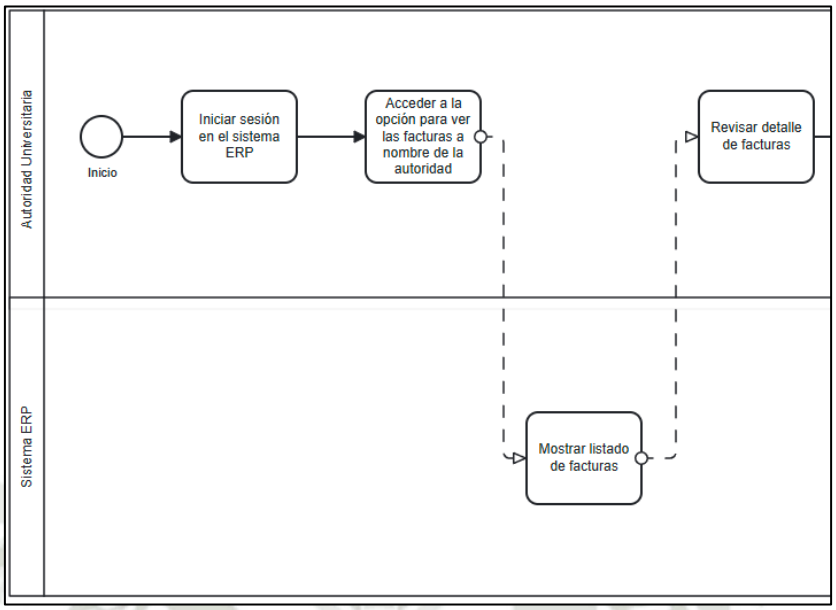
Diagrama BPMN del Proceso de Revisión de Autoridad correspondiente, control logístico y afectación presupuestal.



En esta segunda parte del proceso, las autoridades universitarias acceden al sistema ERP para revisar un listado de comprobantes cargados a su nombre. Pueden visualizar el detalle de cada comprobante, incluyendo montos y observaciones, antes de tomar una decisión de aprobación.

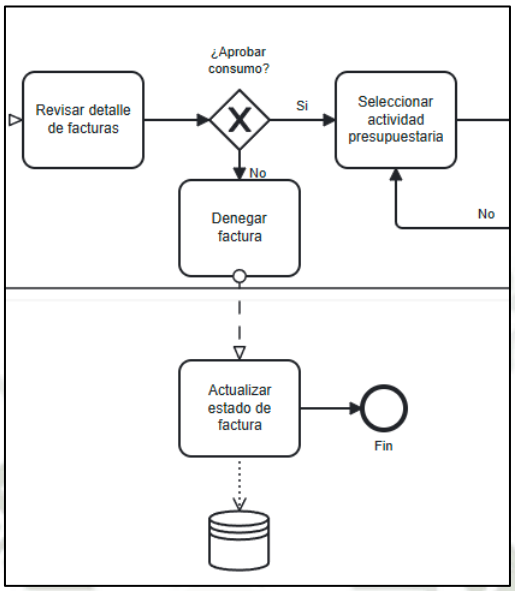
Figura 6.1

Fragmento de la Figura 6 relacionado con el acceso al sistema ERP y el listado de comprobantes dirigidos a una autoridad.



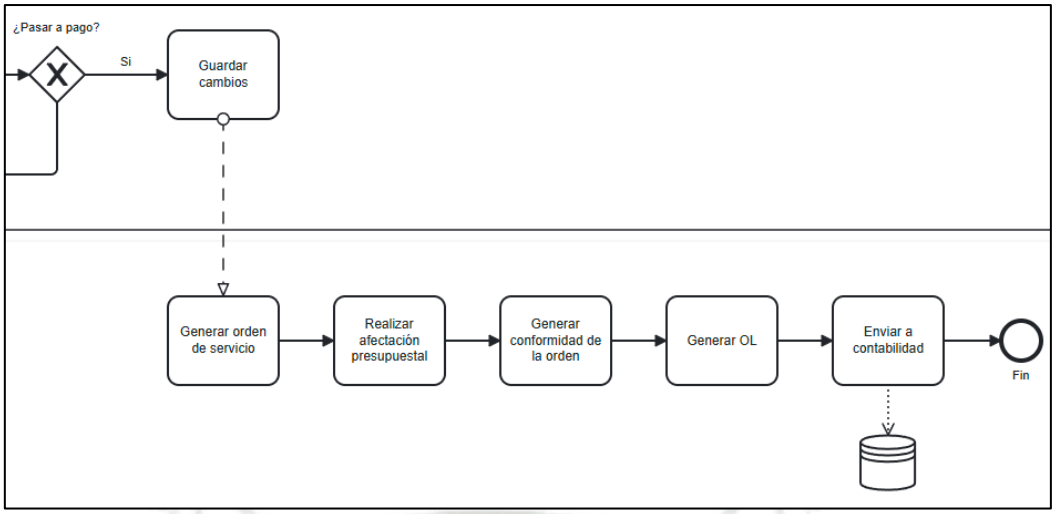
Una vez que un comprobante es revisado, la autoridad tiene la opción de aprobar o denegar el consumo. Si aprueba el comprobante, el sistema presenta una lista de actividades presupuestarias del área correspondiente a las que se ha asignado un presupuesto específico por la universidad. La autoridad selecciona la actividad correspondiente, lo que permite al sistema ajustar automáticamente el saldo disponible para esa actividad en función del monto de la factura. Además, este proceso de selección asegura que los gastos se distribuyan correctamente entre las diferentes partidas presupuestarias asignadas.

Figura 6.2
Fragmento de la Figura 6 relacionado con la aprobación de consumos y actividades presupuestarias.



Posteriormente, el sistema genera una orden de servicio y notifica al área de logística sobre el ajuste presupuestario mientras que el área de contabilidad es notificada sobre la aprobación. Este flujo no solo garantiza un control eficaz de los gastos, asegurando que los consumos se alineen con las asignaciones presupuestarias de la universidad, sino que también mejora la gestión del control logístico.

Figura 6.3
Fragmento de la Figura 6 relacionado con la afectación presupuestal y la generación de la orden de servicio.



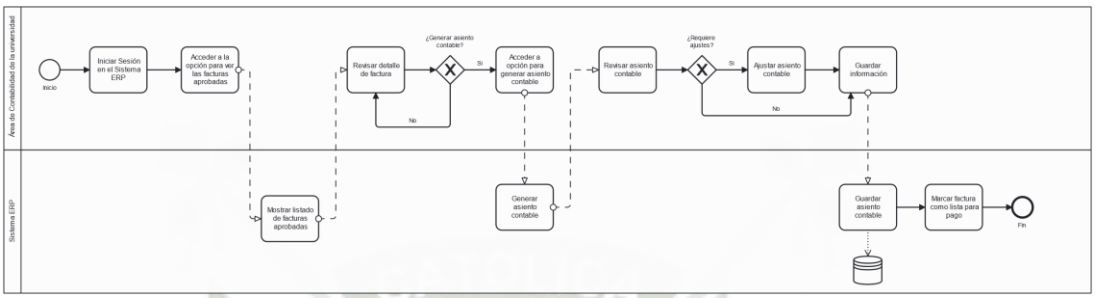
Al integrar la gestión financiera con el control logístico, el sistema permite una planificación más precisa de las necesidades de recursos, facilitando la coordinación entre las áreas responsables de las adquisiciones y las operaciones. Esto asegura que los recursos se utilicen de manera eficiente, minimizando desperdicios y optimizando la cadena de suministro interna. El proceso no solo mejora la eficiencia y transparencia en la gestión financiera, sino que también permite a las autoridades gestionar de manera proactiva sus presupuestos y necesidades logísticas, adaptándose a las demandas cambiantes de su área.

c) Proceso Contable de Comprobantes Aprobados

A continuación, se presenta el diagrama BPMN que ilustra el proceso contable de gestión de comprobantes aprobados por el área de contabilidad de la universidad. Este diagrama resalta cómo los comprobantes, una vez aprobados por las autoridades, son manejados dentro del sistema ERP para asegurar un registro contable preciso y preparado para el pago. El diagrama destaca el flujo de trabajo desde la revisión detallada de cada comprobante hasta la generación y ajuste de asientos contables, mostrando cómo estos pasos se integran en el sistema financiero de la universidad. Representa las interacciones entre el personal contable y el sistema ERP, subrayando la importancia de la precisión y eficiencia en el manejo de las transacciones financieras. Además, el proceso asegura que cada factura cumpla con los

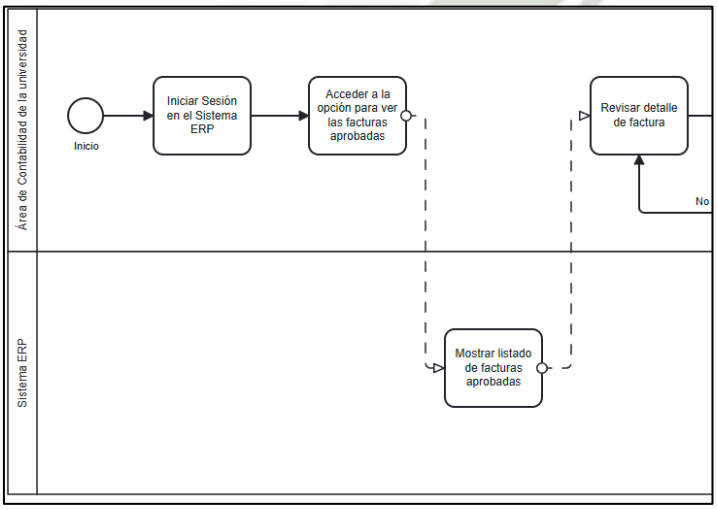
requisitos administrativos y contables antes de ser marcada como lista para pago, garantizando así un control robusto sobre los flujos de caja institucionales.

Figura 7
Diagrama BPMN del Proceso Contable de Comprobantes Aprobados



En esta fase final del proceso, el área de contabilidad de la universidad accede al sistema ERP para gestionar los comprobantes que han sido previamente aprobados por las autoridades. Este acceso les permite visualizar el detalle completo de cada comprobante aprobado, incluyendo información crítica.

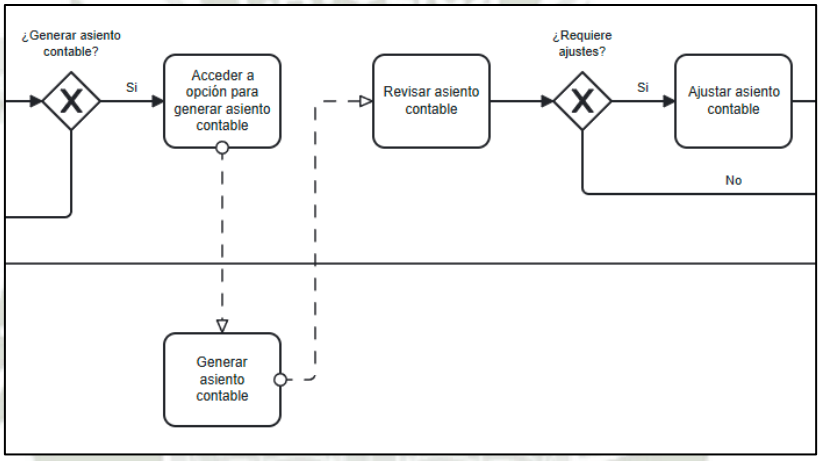
Figura 7.1
Fragmento de la Figura 7 relacionado con el acceso al sistema ERP y el listado de los comprobantes aprobados.



Además, el ERP proporciona la opción de generar automáticamente un asiento contable para cada comprobante, lo que facilita el registro contable de la transacción. Una vez generado

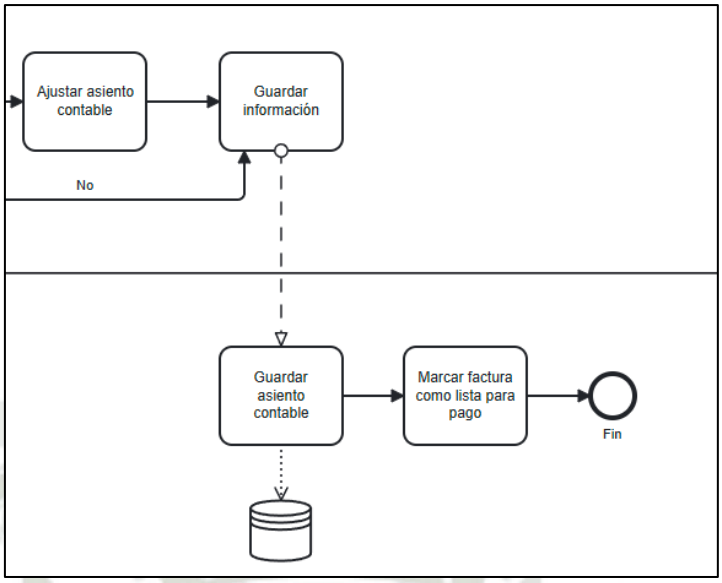
el asiento contable, los contadores pueden revisarlo minuciosamente para asegurarse de que todos los detalles sean correctos y cumplan con las normativas contables de la universidad. Tienen la posibilidad de ajustar los asientos según sea necesario, asegurando que reflejen con precisión las transacciones económicas y los impactos presupuestarios correspondientes. Este ajuste es crucial para mantener la integridad de los registros financieros y asegurar que los informes contables sean precisos y confiables.

Figura 7.2
Fragmento de la Figura 7 relacionado con la generación de asiento contable.



Tras realizar las revisiones y ajustes pertinentes, el personal contable guarda la información actualizada en el sistema. Una vez que el comprobante y su asiento contable asociado han sido validados y guardados, se marca el comprobante como listo para el envío a pago. Este estado indica que el comprobante ha cumplido con todos los requisitos administrativos y contables y está preparado para proceder con el desembolso correspondiente.

Figura 7.3
Fragmento de la Figura 7 relacionado con el almacenamiento de asiento contable.



3.4 Base de Datos

La base de datos es el corazón del sistema ERP de la universidad, y su diseño juega un papel fundamental en la integración y funcionamiento de todas las operaciones administrativas y financieras. Utilizando PostgreSQL como el motor de base de datos, se asegura un manejo robusto y eficiente de grandes volúmenes de datos, lo que es esencial para satisfacer las necesidades de una institución educativa de gran escala.

PostgreSQL, reconocido por su capacidad para gestionar transacciones complejas y mantener la integridad de los datos, permite implementar un modelo relacional que se alinea perfectamente con las exigencias del sistema ERP de la universidad. Además, la utilización de pgAdmin como herramienta de gestión proporciona una interfaz amigable para la administración y mantenimiento de la base de datos, facilitando tareas como la creación de tablas, la ejecución de consultas SQL y la gestión de usuarios y permisos.

3.4.1 Diseño de la Base de Datos

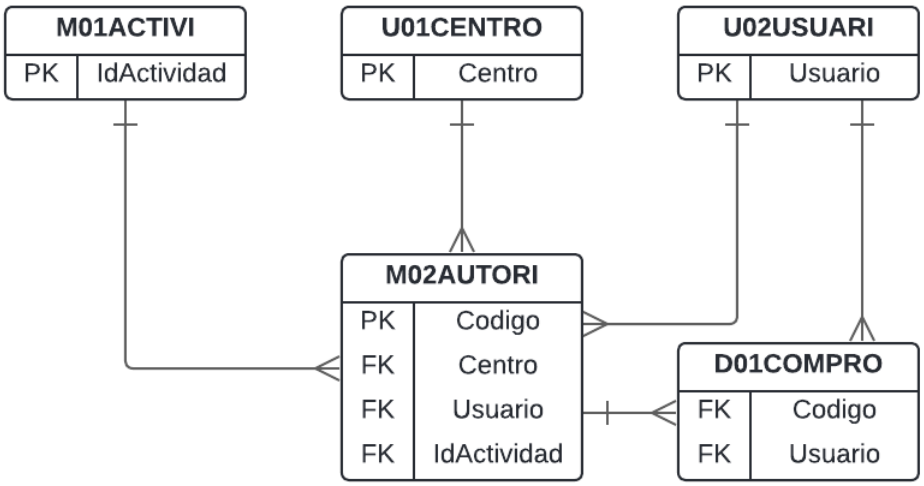
El diseño propuesto para la base de datos se integra completamente con el núcleo del sistema ERP de la universidad, permitiendo una interoperabilidad sin fisuras entre los diferentes módulos del ERP. Esta integración asegura que los procesos de carga, validación y almacenamiento de facturas operen de manera coordinada con las funciones existentes del

ERP, optimizando así el flujo de trabajo administrativo y garantizando que todas las áreas de la universidad trabajen sobre una base de datos unificada y coherente.

3.4.2 Diagrama Relacional

El diagrama relacional es una representación visual que muestra la estructura lógica de la base de datos, ilustrando cómo las tablas están interrelacionadas. A continuación, se presenta el diagrama relacional que describe el diseño de la base de datos utilizada en este proyecto:

Figura 8
Diagrama Relacional del Sistema



3.4.3 Descripción de las Entidades y Atributos

a) **M01ACTIVI (Actividades de determinadas áreas de la universidad):**
   - **IdActividad (PK):**
     - **Tipo de dato:** VARCHAR(10)
     - **Descripción:** Identificador único de cada actividad. Sirve para distinguir de manera exclusiva cada actividad asociada a un área registrada en el sistema.
     - **Restricciones:** Clave primaria (PK) que asegura la unicidad de cada entrada.

Figura 9

Entidad M01ACTIVI

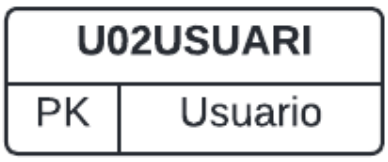
b) **U01CENTRO (Centros operativos de la universidad):**
   - **Centro (PK):**
     - **Tipo de dato:** VARCHAR(10)
     - **Descripción:** Código único que identifica cada centro operativo dentro del sistema.
     - **Restricciones:** Clave primaria (PK) que garantiza la unicidad.

Figura 10
Entidad U01CENTRO

Figura 10 Entidad U01CENTRO

c) **U02USUARI (Usuarios registrados en la universidad):**
   - **Usuario (PK):**
     - **Tipo de dato:** VARCHAR(10)
     - **Descripción:** Código único de usuario que identifica a los usuarios registrados en el sistema.
     - **Restricciones:** Clave primaria (PK) que asegura que cada usuario es único.

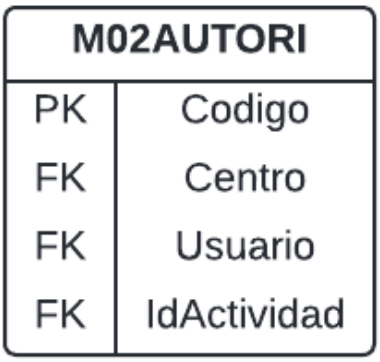
Figura 11
Entidad U01USUARI



d) **M02AUTORI (Autoridades que pueden hacer uso del módulo al que hacer referencia el proyecto):**

- **Codigo (PK):**
  - **Tipo de dato:** VARCHAR(10)
  - **Descripción:** Código único de autoridad que permite identificar a las autoridades que pueden realizar consumos.
  - **Restricciones:** Clave primaria (PK).
- **Centro (FK):**
  - **Tipo de dato:** VARCHAR(10)
  - **Descripción:** Código que refiere al centro operativo al cual está asociada la autoridad.
  - **Restricciones:** Clave foránea (FK) relacionada con U01CENTRO.
- **Usuario (FK):**
  - **Tipo de dato:** VARCHAR(10)
  - **Descripción:** Código de usuario de la autoridad en la universidad.
  - **Restricciones:** Clave foránea (FK) que se refiere a U02USUARI.
- **IdActividad (FK):**
  - **Tipo de dato:** VARCHAR(10)
  - **Descripción:** Identificador de la actividad asociada a la autoridad.
  - **Restricciones:** Clave foránea (FK) que se relaciona con M01ACTIVI.

Figura 12
Entidad M02AUTORI



e) D01COMPRO (Comprobantes que emite la empresa asociada):

- **Codigo (FK):**
  - **Tipo de dato:** VARCHAR(10)
  - **Descripción:** Código de la autorización asociada, estableciendo un vínculo con M02AUTORI.
  - **Restricciones:** Clave foránea (FK) que se refiere a M02AUTORI.
- **Usuario (FK):**
  - **Tipo de dato:** VARCHAR(10)
  - **Descripción:** Código de usuario asociado con el registro de autorización.
  - **Restricciones:** Clave foránea (FK) relacionada con U02USUARI.

Figura 13
Entidad D01COMPRO

Figura 13 Entidad D01COMPRO

3.4.4 Relaciones entre entidades

Relación entre M01ACTIVI y M02AUTORI:

En relación a la Figura 14, la entidad M02AUTORI tiene una clave foránea IdActividad que se relaciona con la clave primaria IdActividad de M01ACTIVI. Esta relación indica que cada autoridad de la universidad está asociada con una actividad específica.

Figura 14
Relación entre M01ACTIVI y M02AUTORI

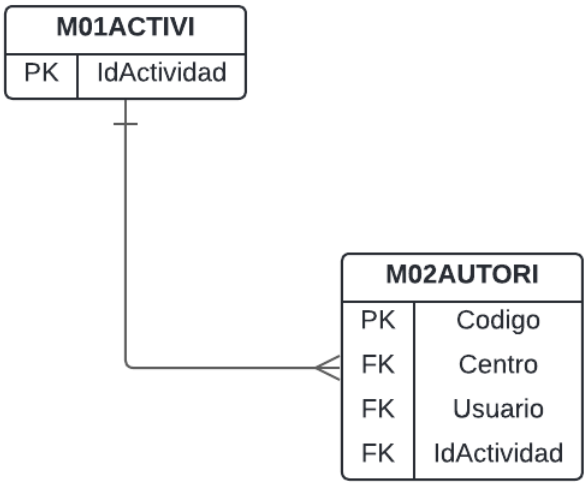


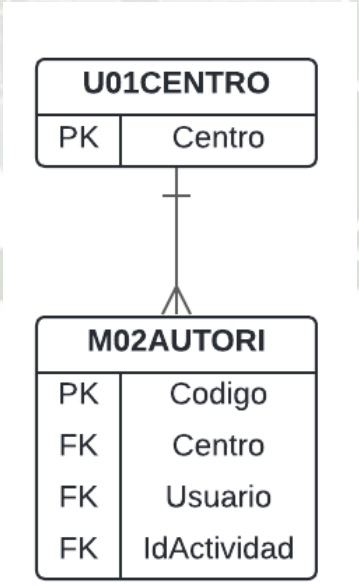
Figura 14 Relación entre M01ACTIVI y M02

Relación entre U01CENTRO y M02AUTORI:

En relación con la Figura 15, la clave foránea Centro en M02AUTORI se refiere a la clave primaria Centro en U01CENTRO. Esto significa que cada autoridad de la universidad está vinculada a un centro operativo específico.

Figura 15

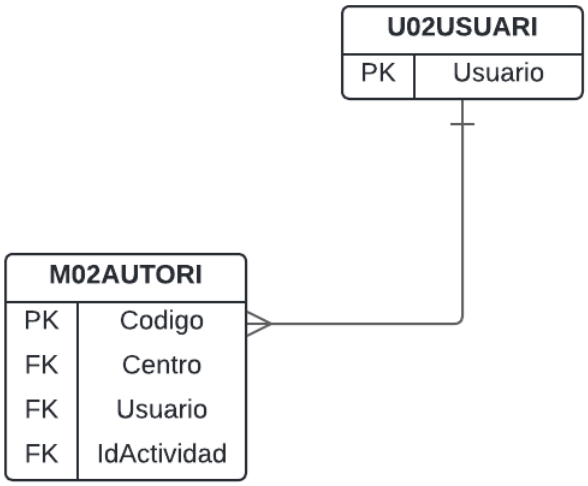
Relación entre U01CENTRO y M02AUTORI



Relación entre U02USUARI y M02AUTORI:

En relación con la Figura 16, la clave foránea Usuario en M02AUTORI se refiere a la clave primaria Usuario en U02USUARI. Esta relación asocia las autoridades con usuarios existentes.

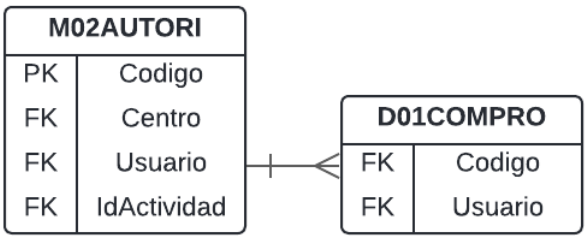
Figura 16
Relación entre U02USUARI y M02AUTORI



Relación entre M02AUTORI y D01COMPRO:

En relación con la Figura 17, D01COMPRO tiene una clave foránea Codigo que se refiere a Codigo en M02AUTORI. Esto establece una asociación directa entre las autoridades y los comprobantes registrados, permitiendo vincular datos adicionales con cada autoridad.

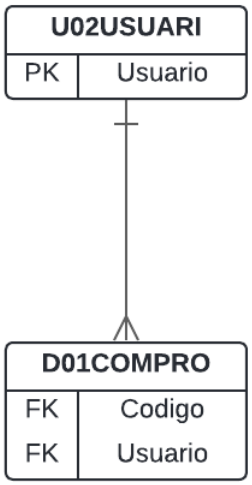
Figura 17
Relación entre M02AUTORI y D01COMPRO



Relación entre U02USUARI y D01COMPRO:

En relación con la Figura 18, D01COMPRO también tiene una clave foránea Usuario que se refiere a Usuario en U02USUARI. Esta relación indica que los comprobantes están vinculadas a usuarios existentes.

Figura 18
Relación entre M02AUTORI y D01COMPRO



3.4.5 Diccionario de Datos

Tabla 4
Diccionario de datos de la tabla M02AUTORI

Atributo	Tipo de Dato	Descripción	Restricción
Codigo	CHARACTER (5)	Código único identificador	PK
Estado	CHARACTER (1)	Estado: • A: Activo • I: Inactivo	
Tipo	CHARACTER (2)	Tipo (Empresa asociada EA)	
Usuario	CHARACTER (5)	Código de usuario autorizado	FK
Centro	CHARACTER (3)	Centro operativo	FK

Tabla 5
Diccionario de datos de la tabla D01COMPRO

Atributo	Tipo de Dato	Descripción	Restricción
Serial	SERIAL	Serial correlativo	
Codigo	CHARACTER (4)	Código único M02AUTORI	FK
Estado	CHARACTER (1)	Estado: • R: Registrado • A: Aprobado • D: Denegado • C: Contabilizado • P: Pagado	

		• X: Anulado	
Comprobante	CHARACTER (13)	Número de comprobante	
Datos	TEXT	Datos del comprobante	
Generacion	TIMESTAMP	Fecha y hora de generación	
CodigoUsu	CHARACTER (5)	Código que aprueba el comprobante U02USUARI	FK

3.5 Microservicios

Los microservicios son componentes esenciales en la arquitectura del sistema, permitiendo una implementación escalable de funciones específicas. En el presente proyecto, se desarrollaron dos microservicios clave que facilitan la automatización de procesos críticos: la extracción de datos de comprobantes en formato XML y la generación de asientos contables automáticos. Ambos microservicios están desarrollados en Python y se integran con el sistema ERP a través de solicitudes cURL desde PHP. Además, recibe solicitudes en formato JSON que contiene el archivo XML, después de procesarlo y extraer los datos necesarios, la respuesta será también en formato JSON.

3.5.1 Extracción de Datos de Comprobantes XML

Este microservicio se encarga de procesar archivos XML de comprobantes, extrayendo información relevante que posteriormente será almacenada en la base de datos.

La información relevante incluye:

Cabecera

- Número de comprobante
- Fecha de emisión
- Fecha de vencimiento
- RUC del Cliente
- Forma de pago

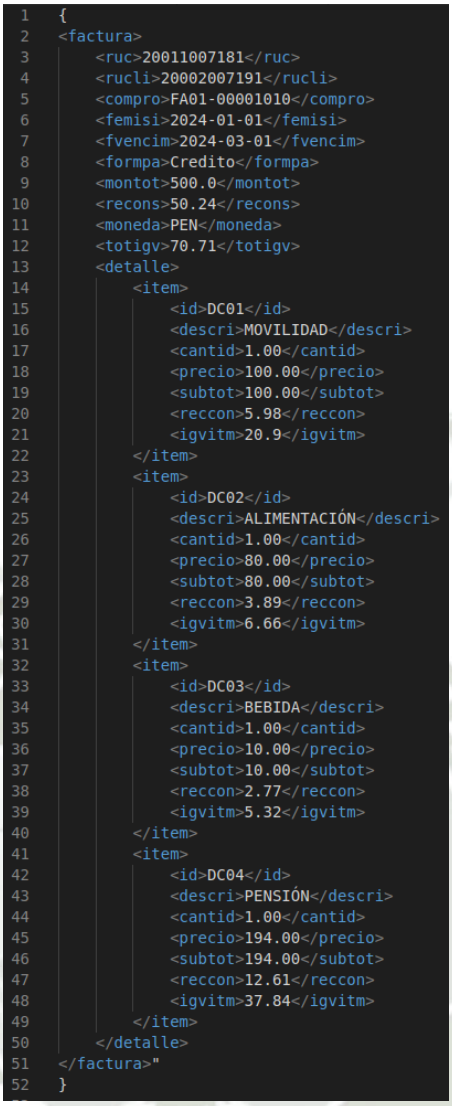
- Moneda
- Monto Total
- Monto IGV Total
- Recargo al consumo Total

Detalle

- ID
- Cantidad de item
- Descripción de item
- Precio por item
- Subtotal por item
- IGV por item
- Recargo al consumo por item

Para ilustrar cómo se envía y recibe la información a través del microservicio, a continuación, se presenta un ejemplo del formato JSON utilizado para la extracción de datos de un comprobante en formato XML.

Figura 19
Solicitud JSON enviada al Microservicio



```
{
<factura>
    <ruc>20011007181</ruc>
    <rucli>20002007191</rucli>
    <compro>FA01-00001010</compro>
    <femisi>2024-01-01</femisi>
    <fvencim>2024-03-01</fvencim>
    <formpa>Credito</formpa>
    <montot>500.0</montot>
    <recons>50.24</recons>
    <moneda>PEN</moneda>
    <totigv>70.71</totigv>
    <detalle>
        <item>
            <id>DC01</id>
            <descri>MOVILIDAD</descri>
            <cantid>1.00</cantid>
            <precio>100.00</precio>
            <subtot>100.00</subtot>
            <reccon>5.98</reccon>
            <igvitm>20.9</igvitm>
        </item>
        <item>
            <id>DC02</id>
            <descri>ALIMENTACIÓN</descri>
            <cantid>1.00</cantid>
            <precio>80.00</precio>
            <subtot>80.00</subtot>
            <reccon>3.89</reccon>
            <igvitm>6.66</igvitm>
        </item>
        <item>
            <id>DC03</id>
            <descri>BEBIDA</descri>
            <cantid>1.00</cantid>
            <precio>10.00</precio>
            <subtot>10.00</subtot>
            <reccon>2.77</reccon>
            <igvitm>5.32</igvitm>
        </item>
        <item>
            <id>DC04</id>
            <descri>PENSIÓN</descri>
            <cantid>1.00</cantid>
            <precio>194.00</precio>
            <subtot>194.00</subtot>
            <reccon>12.61</reccon>
            <igvitm>37.84</igvitm>
        </item>
    </detalle>
</factura>"
}
```

Solicitud JSON enviada al Microservicio

En la Figura 19, el contenido XML de la factura incluye campos como ruc (Registro Único de Contribuyentes del proveedor), rucli (Registro Único de Contribuyentes del cliente), compro (Comprobante), femisi (Fecha de emisión), fvencim (Fecha de vencimiento), formpa (Forma de pago), montot (Monto total), recons (Recargo de consumo), moneda (Moneda), totigv (Total IGV), y un detalle que incluye varios ítems con sus respectivas descripciones y montos. La extracción de esta información se realiza utilizando la biblioteca ElementTree de Python, que permite un análisis eficiente del contenido XML.

Figura 20

Respuesta JSON devuelta por el Microservicio

```
{
    "COMPRO": "FA01-00001010",
    "RUCCLI": "20002007191",
    "RUCPRO": "20011007181",
    "FEMISI": "2024-01-01",
    "FVENCIM": "2024-03-01",
    "FORMPA": "Credito",
    "MONTOT": 500.0,
    "RECONS": 50.24,
    "MONEDA": "PEN",
    "TOTIGV": 70.71,
    "DETALL": [
        {
            "ID": "DC01",
            "DESCRI": "MOVILIDAD",
            "CANTID": "1.00",
            "PRECIO": 100.00,
            "SUBTOT": 100.00,
            "RECCON": 5.98,
            "IGVITM": 20.9
        },
        {
            "ID": "DC02",
            "DESCRI": "ALIMENTACIÓN",
            "CANTID": "1.00",
            "PRECIO": 80.00,
            "SUBTOT": 80.00,
            "RECCON": 3.89,
            "IGVITM": 6.66
        },
        {
            "ID": "DC03",
            "DESCRI": "BEBIDA",
            "CANTID": "1.00",
            "PRECIO": 10.00,
            "SUBTOT": 10.00,
            "RECCON": 2.77,
            "IGVITM": 5.32
        },
        {
            "ID": "DC04",
            "DESCRI": "PENSIÓN",
            "CANTID": "1.00",
            "PRECIO": 194.00,
            "SUBTOT": 194.00,
            "RECCON": 12.61,
            "IGVITM": 37.84
        }
    ]
}
```


Como se aprecia en la Figura 20, el microservicio procesa el contenido XML y devuelve un JSON con los datos extraídos: comprobante, RUC del cliente y proveedor, fechas, forma de pago, montos y detalles de los ítems.

3.5.2 Generación de Asientos Contables

El microservicio de generación automática de asientos contables permite agilizar y asegurar la precisión del proceso contable, reduciendo carga manual y minimizando errores humanos.

Para ilustrar cómo se realiza la solicitud se tomó como referencia la Figura 20, ya que esta información se guardó en la base de datos y se utilizará para generar los asientos contables.

Figura 21
Respuesta JSON devuelta por el Microservicio


```json
{
    "EMISIO": "2024-01-01",
    "VENCIM": "2024-03-01",
    "TIPREG": "XML",
    "COMPRO": "FA01-00001010",
    "CODIGO": "0001",
    "0": {
        "CTACNT": "0501",
        "DESCNT": "PENSION",
        "DEBE": 194.00,
        "HABER": 0,
        "OPERAC": "A"
    },
    "1": {
        "CTACNT": "0501",
        "DESCNT": "ALIMENTACION",
        "DEBE": 80.00,
        "HABER": 0,
        "OPERAC": "A"
    },
    "2": {
        "CTACNT": "0502",
        "DESCNT": "TOTAL",
        "DEBE": 0,
        "HABER": 2210,
        "OPERAC": "T"
    }
}
```

Como muestra la Figura 21, el microservicio procesa la solicitud y devuelve un JSON con los datos del asiento contable generado, incluyendo detalles como las cuentas contables, descripciones, montos en el debe y haber, y otros detalles necesarios.

3.6 Pruebas y Validación

En el proyecto de desarrollo del módulo de control presupuestal y logístico para el pago de comprobantes en una universidad privada, se llevaron a cabo diversas pruebas para garantizar la calidad, fiabilidad y rendimiento del sistema.

El enfoque de pruebas abarcó múltiples niveles, desde pruebas unitarias de los microservicios individuales hasta pruebas de integración para verificar la correcta interacción entre los componentes del sistema. Además, se implementaron pruebas de aceptación del usuario (UAT) para asegurar que el sistema cumpla con las necesidades y expectativas del usuario final en un entorno de prueba que simula las condiciones reales de operación.

3.6.1 Pruebas Unitarias de Microservicios

Las pruebas unitarias se realizaron para validar la funcionalidad de cada microservicio de manera aislada. Estas pruebas verifican que cada función o método produzca la salida esperada para un conjunto dado de entradas.

a) Herramientas Utilizadas

- Pytest: Es un framework de pruebas para Python que facilita la creación de pruebas unitarias
- Unittest: Módulo estándar de pruebas unitarias en Python

Figura 22

Validación de casos de éxito y error para el microservicio de extracción de datos

```python
import unittest
from fastapi.testclient import TestClient
from CXmlFactura import app

class TestExtraerXml(unittest.TestCase):
    def f_setUp(self):
        self.client = TestClient(app)

    def f_TestExitoExtraerXml(self):
        response = self.client.post("/extract_data/", json={
            "xml_content": "<factura><ruc>20011007181</ruc><rucli>20002007191</rucli><cor
        })
        self.assertEqual(response.status_code, 200)
        self.assertIn("COMPRO", response.json())

    def f_TestValidarXml(self):
        response = self.client.post("/extract_data/", json={
            "xml_content": "<factura><ruc>20011007181</ruc><rucli>20002007191</ruc>"
        })
        self.assertEqual(response.status_code, 400)
        self.assertIn("El XML proporcionado no es válido.", response.json()["detail"])

if __name__ == "__main__":
    unittest.main()
```

En la Figura 22 se muestra cómo se validan los casos de éxito y error para el microservicio de extracción de datos.

3.6.2 Pruebas de Integración del Sistema

En el proyecto de desarrollo del módulo de control presupuestal y logístico para el pago de comprobantes en una universidad privada, las pruebas de integración aseguraron que la comunicación y la interacción entre los componentes del sistema fueran fluidas y sin errores.

a) Herramientas Utilizadas

- Postman: Para realizar pruebas de API y verificar la respuesta de los endpoints.
- Pytest y requests (Python): Para automatizar las pruebas de integración de los microservicios.

Figura 23
Pruebas de integración del sistema

```python
import unittest
import requests

class TestIntegration(unittest.TestCase):
    def f_TestIntegracion(self):
        # URL del microservicio de extracción de datos
        url_extraer = "http://localhost:8000/extraer_xml/"
        # URL del microservicio de generación de asientos contables
        url_generar = "http://localhost:8000/generar_asiento/"

        # Datos de prueba (archivo XML)
        xml_data = {
            "xml_content": "<factura><ruc>20011007181</ruc><rucli>20002007191</rucli><compr
        }

        # Realizar la solicitud al microservicio de extracción de datos
        response_extraer = requests.post(url_extraer, json=xml_data)
        self.assertEqual(response_extraer.status_code, 200)

        # Datos extraídos del XML (simulación de la respuesta del microservicio)
        extraer_data = response_extraer.json()

        # Realizar la solicitud al microservicio de generación de asientos contables
        response_generar = requests.post(url_generar, json=extraer_data)
        self.assertEqual(response_generar.status_code, 200)

        # Verificar la respuesta del microservicio de generación de asientos contables
        asiento_generado = response_generar.json()
        self.assertEqual(asiento_generado["total_debe"], asiento_generado["total_haber"])

if __name__ == "__main__":
    unittest.main()
```

Figura 23 Pruebas de integración del sistema

En la Figura 23 se aprecia lo siguiente, se prepara el contenido XML del comprobante para la prueba. Luego, se realiza una solicitud POST al endpoint del microservicio de extracción de datos y se verifica que la respuesta sea exitosa y se toma la respuesta JSON del microservicio de extracción de datos como entrada para el siguiente microservicio. Además, se realiza una solicitud POST al endpoint del microservicio de generación de asientos contables utilizando los datos extraídos y se verifica que la respuesta sea exitosa. Finalmente, se comprueba que el total del debe y el haber en el asiento contable generado sean iguales.

3.6.3 Pruebas de Aceptación del Usuario

Las pruebas de aceptación del usuario garantizaron que el sistema cumpla con las expectativas y necesidades de los usuarios finales. Estas pruebas se llevaron a cabo

en un entorno que simula el entorno de producción para verificar que el sistema funcione según los requisitos especificados.

El principal objetivo es validar que el sistema es apto para su uso y satisface los criterios de aceptación.

a) Definición de Casos de Prueba

Los casos de prueba se definieron en base a las historias de usuario y los criterios de aceptación documentados durante el desarrollo del proyecto.

b) Configuración del Entorno de Prueba

Se configuró un entorno de pruebas que replica el entorno de producción. Esto incluye la configuración necesaria de los componentes necesarios para el funcionamiento del sistema.

c) Ejecución de pruebas

Los usuarios finales, generalmente representados por miembros del personal de contabilidad y autoridades de la universidad, ejecutaron los casos de prueba definidos.

d) Documentación de resultados

Los resultados de las pruebas se documentaron cuidadosamente, registrando cualquier problema encontrado.

e) Ejemplo de Pruebas de Aceptación del Usuario

Se tomará de ejemplo la Figura 24 que hace referencia a la historia de usuario de aprobación o denegación de consumo.

Figura 24

Historia de usuario de aprobación o denegación de consumo.

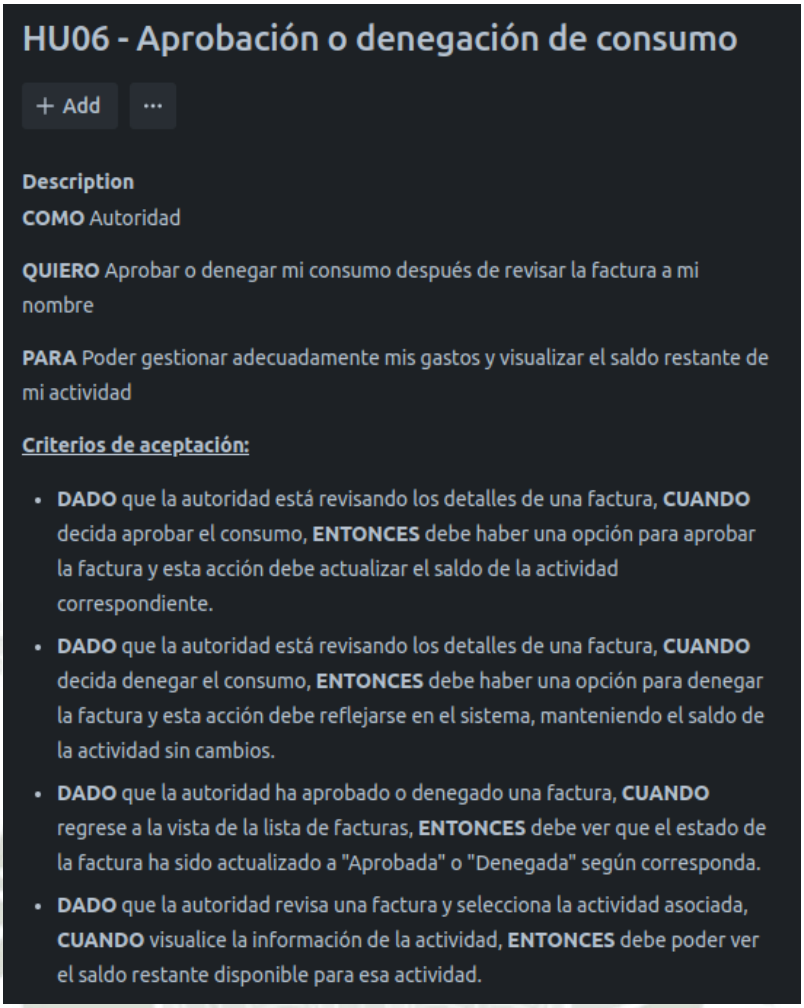


Figura 24 Historia de usuario de aprobación o denegación de con

Caso de Prueba:

a) Descripción: Validar que la autoridad pueda aprobar o denegar un comprobante después de revisar los detalles y que el saldo de la actividad se actualice correctamente.

b) Precondiciones: La autoridad ha iniciado sesión y tiene acceso a la lista de facturas cargadas a su nombre.

c) Pasos:

1. Navegar a la lista de facturas cargadas.
2. Seleccionar una factura para ver sus detalles.
3. Aprobar la factura.
4. Verificar que el saldo de la actividad se actualice.

5. Denegar otra factura.
6. Verificar que el estado de la factura se actualice a "Denegada".

d) Resultados Esperados:

- La opción para aprobar la factura está disponible y funcional.
- El saldo de la actividad se actualiza correctamente después de aprobar la factura.
- La opción para denegar la factura está disponible y funcional.
- El estado de la factura se actualiza correctamente a "Denegada".

3.7 Seguridad

La seguridad fue un aspecto fundamental en el diseño e implementación del módulo de control presupuestal y logístico para el pago de comprobantes en una universidad privada. Asegurar la integridad, confidencialidad y disponibilidad de los datos es crucial para mantener la confianza de los usuarios y cumplir con las normativas y estándares de seguridad. La naturaleza sensible de los datos manejados, que incluye información financiera y personal, requiere la implementación de medidas de seguridad que protejan contra una amplia gama de amenazas y vulnerabilidades.

a) Integridad de los Datos:

La integridad de los datos garantizó que la información no sea alterada de manera no autorizada durante su almacenamiento, procesamiento o transmisión. Para asegurar la integridad de los datos, se utilizaron técnicas como la validación de entradas y la implementación de controles de acceso basados en roles. Estos métodos aseguran que solo las personas y sistemas autorizados puedan modificar la información y que cualquier cambio no autorizado sea detectado y revertido.

b) Disponibilidad de los Datos:

La disponibilidad asegura que los datos y los sistemas estén accesibles para los usuarios autorizados cuando los necesiten. Para mantener la disponibilidad, se implementaron medidas como la realización de copias de seguridad regulares. Esto aseguró que el sistema pueda recuperarse rápidamente de fallos y que los usuarios puedan acceder a los servicios sin interrupciones significativas.

CAPÍTULO IV: RESULTADOS Y DISCUSIÓN

En este capítulo se presentan y analizan los resultados obtenidos durante el desarrollo del módulo de control presupuestal y logístico para el pago de comprobantes en una universidad privada. La discusión se estructura en función de los objetivos planteados al inicio del proyecto, evaluando en qué medida se han y cómo estos resultados contribuyen al propósito general del sistema.

El capítulo comienza con una revisión de los resultados alcanzados en relación con cada objetivo específico del proyecto. A continuación, se expone una discusión detallada sobre el desempeño del sistema, la efectividad de las metodologías utilizadas, y la validación del sistema a través de pruebas técnicas y de aceptación por parte de los usuarios.

Finalmente, se interpretan los resultados obtenidos en el contexto de su impacto en la institución y su relevancia para el campo académico y profesional.

4. Resultados en Relación con los Objetivos

En esta sección, se presentarán y analizarán los resultados obtenidos durante el desarrollo e implementación del proyecto, evaluando cómo estos resultados se alinean con el objetivo general y los objetivos específicos. Esto permitirá determinar en qué medida se han cumplido los objetivos, destacando los logros alcanzados.

4.1 Objetivo General: Implementar un Módulo de Control Presupuestal y Logístico

El objetivo general de este proyecto fue desarrollar e implementar un módulo de control presupuestal y logístico para una universidad privada, que digitalizara el proceso de contabilización de comprobantes emitidos por una empresa asociada a la universidad. El sistema debía integrarse con el ERP existente en la universidad, facilitando una gestión más eficiente y precisa de los comprobantes y otros documentos financieros. Para validar que este objetivo se cumplió, se realizaron Pruebas de Aceptación del Usuario (UAT) con usuarios

finales, como el personal de contabilidad de la empresa asociada y las autoridades de la universidad.

Estas pruebas fueron diseñadas para simular escenarios de uso real del sistema y asegurar que cumpliera con las expectativas funcionales y operativas. Las principales pruebas de aceptación son las que se detallan en la Tabla 6.

Tabla 6
Pruebas de aceptación realizadas

Objetivo de la prueba	Resultados
Validar la capacidad del sistema para gestionar todo el proceso de contabilización de un comprobante, desde la carga del archivo XML hasta la aprobación final.	Los usuarios pudieron completar todas las etapas del proceso sin dificultades, desde la carga del archivo XML hasta la aprobación de los comprobantes, confirmando que el sistema digitaliza el flujo de trabajo.
Asegurar que el sistema pueda cargar y validar correctamente los archivos XML de los comprobantes.	El sistema validó correctamente los archivos XML cargados, aceptando los que cumplían con el formato requerido y proporcionando *feedback* en caso de errores, lo que demuestra su capacidad para manejar la entrada de datos de manera confiable.
Validar que el módulo se integra correctamente con el ERP de la universidad, sincronizando los datos.	La integración fue exitosa, permitiendo que la información del comprobante se sincronizará correctamente con el ERP, lo que garantiza la coherencia y accesibilidad de los datos en todo el sistema institucional.
Validar que las autoridades puedan revisar y aprobar o rechazar los comprobantes de manera eficiente.	Las autoridades pudieron revisar y aprobar los comprobantes sin complicaciones, y el sistema actualizó correctamente el estado de los comprobantes, demostrando su eficacia en la gestión de aprobaciones.
Evaluar la facilidad de uso del sistema para los usuarios finales.	Los usuarios reportaron una experiencia positiva, destacando la claridad de la interfaz y la facilidad de navegación.

Discusión:

El análisis de los resultados obtenidos a través de las pruebas de aceptación del usuario permitió validar que el sistema desarrollado cumplió de manera efectiva con el objetivo general planteado.

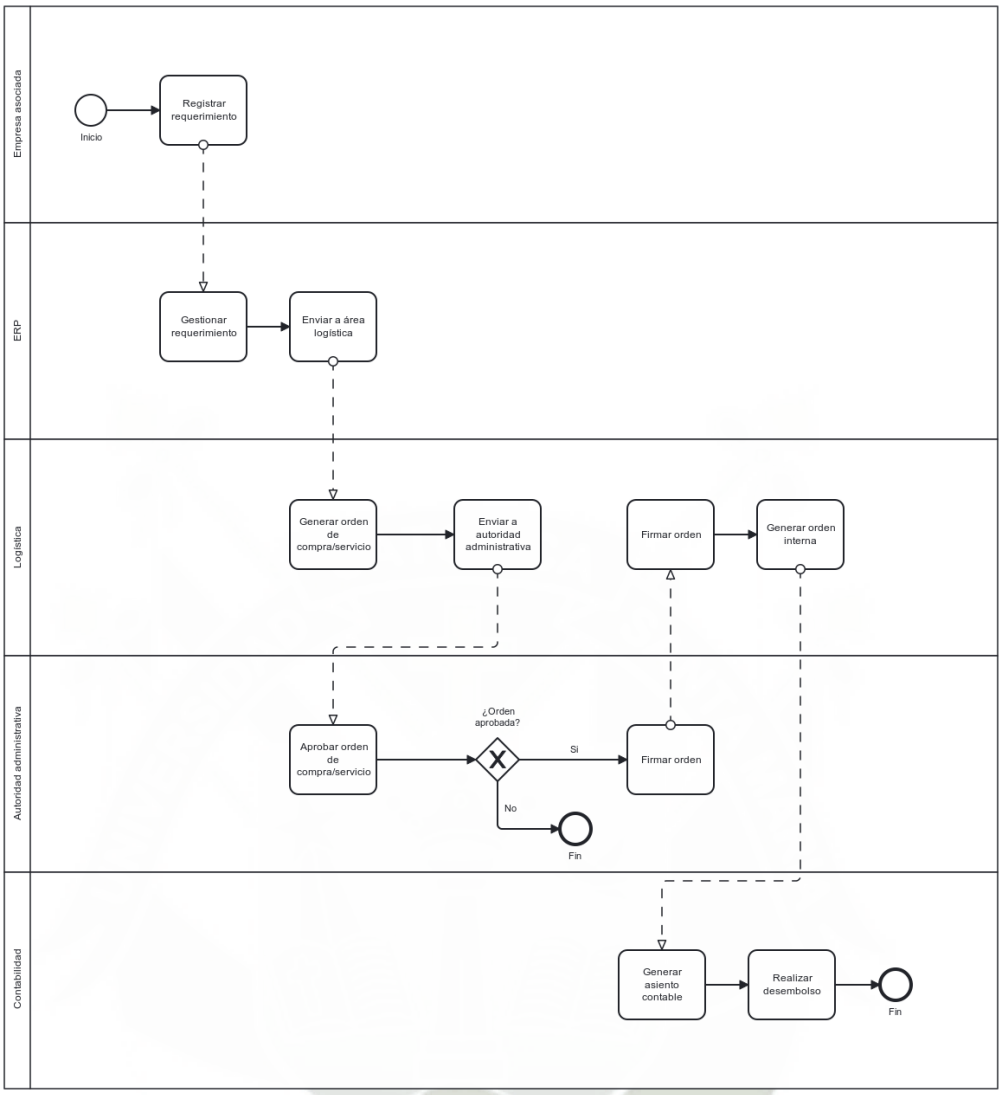
La integración del módulo con el sistema ERP de la universidad fue importante, y las pruebas demostraron que los datos se sincronizaron adecuadamente. Esto garantizó que la información estuviera disponible y fuera coherente en toda la plataforma institucional, facilitando la transparencia y control financiero.

Las pruebas de aprobación de comprobantes y generación de asientos contables confirmaron que el sistema ofrecía flexibilidad a los usuarios para revisar los comprobantes y gestionar el saldo de sus actividades; y a los usuarios contables, la flexibilidad para revisar y ajustar los asientos contables antes de su registro final.

4.2 Objetivo Específico 1: Reducir el tiempo de procesamiento y contabilización de comprobantes

Uno de los principales objetivos específicos del proyecto fue reducir significativamente el tiempo necesario para procesar y contabilizar los comprobantes emitidos por la empresa asociada a la universidad.

Figura 25
Proceso anterior a la implementación del sistema



Como se observa en la Figura 25, antes de la implementación del nuevo módulo, el procesamiento de comprobantes comprendía muchas tareas manuales que podían demorar hasta 10 días, ya que los comprobantes pasaban por diversas áreas, cada una de las cuales contribuía al tiempo total del proceso, con la implementación del nuevo módulo se esperaba reducir este tiempo, digitalizando varias etapas clave, como la extracción de datos de los comprobantes y la generación automática de los asientos contables, además de prescindir de ciertas áreas en donde el proceso podría tardar.

Para validar el cumplimiento de este objetivo, se utilizó un enfoque principal:

a) Pruebas de Validación del Tiempo de Procesamiento (*Before-and-After Analysis)*

Se realizó un análisis comparativo entre los tiempos de procesamiento de comprobantes antes y después de la implementación del sistema.

Para ello, se recolectaron datos históricos del tiempo promedio que tomaba procesar una factura por área, desde la generación del comprobante hasta su pago.

Tabla 7
Tiempo de procesamiento de comprobantes por área antes de la implementación del sistema

Área o agente	Actividades relacionadas	Promedio de días requeridos
Contabilidad (Empresa Asociada)	Registro de requerimiento de comprobante	1
Autoridad administrativa	Aprobación del comprobante	2
Logística	Generación de orden de compra/servicio Firma del jefe de área Generación de orden interna	3
Presupuesto	Afectación presupuestal	1
Contabilidad	Generación de asientos contables Desembolso	3
Total		10

La Tabla 7 ilustra claramente el tiempo de procesamiento de comprobantes antes de la implementación del nuevo sistema, en donde se puede inferir lo siguiente:

- El proceso completo de contabilización y pago de un comprobante requería un promedio total de 10 días, lo que refleja la complejidad y la cantidad de pasos involucrados, así como la participación de múltiples áreas dentro de la universidad
- El área de logística contribuye con un tiempo significativo de 3 días. Esto sugiere que la generación de órdenes en esta área es un cuello de botella que el nuevo sistema tuvo que abordar.
- La necesidad de intervención de varias áreas incrementa la duración del proceso. Cada área añade su propio tiempo de espera, lo que extiende el ciclo total de procesamiento.
- La aprobación del comprobante por la autoridad administrativa, que toma un promedio de 2 días, también es un factor crítico en el tiempo total de procesamiento y se beneficiaría de mayor eficiencia.

Tabla 8
Tiempo de procesamiento de comprobantes por área después de la implementación del sistema

Área o agente	Actividades relacionadas	Promedio de días requeridos
Contabilidad (Empresa Asociada)	Registro de comprobante	1
Autoridad de la universidad	Aprobación del comprobante	2
ERP	Generación de orden de compra/servicio Generación de orden interna Afectación presupuestal	-
Contabilidad	Generación de asientos contables Desembolso	2

Total	5

La Tabla 8 muestra el tiempo de procesamiento de comprobantes por área después de la implementación del sistema en donde se puede inferir lo siguiente:

- El tiempo total de procesamiento de comprobantes se ha reducido de 10 días a 5 días, lo que representa una disminución del 50% en el tiempo total. Esta reducción es un claro indicador de la eficiencia que el nuevo sistema ha aportado al proceso.
- Las tareas que anteriormente se realizaban en el área de Logística y de Presupuesto ahora se centralizaron y digitalizaron dentro del ERP de la universidad, eliminando el tiempo de espera previamente requerido en estas áreas. Esto no solo simplifica el flujo de trabajo, sino que también reduce la complejidad del proceso al consolidar varias tareas en un sistema centralizado.
- El área de Contabilidad de la universidad presentó una reducción de tiempo, la generación de asientos contables automáticos redujo el tiempo de 3 días a 2 días. Esto sugiere que la digitalización de estos procesos ha logrado una mayor eficiencia y precisión.
- Aunque el tiempo requerido para la aprobación de comprobantes por parte de la autoridad universitaria se mantuvo en 2 días, el sistema mejoró la gestión de estos comprobantes al permitir que cada autoridad de la universidad, no solo la administrativa, pudiera gestionar eficientemente el saldo de sus actividades.

4.3 Objetivo Específico 2: Implementar microservicios para la extracción de datos y generación de asientos contables

El segundo objetivo específico del proyecto fue la implementación de microservicios que facilitaran la extracción de datos desde los comprobantes en formato XML y la generación automática de asientos contables.

a) Extracción de Datos de Comprobantes en Formato XML

Se desarrolló un microservicio especializado en la extracción de datos desde archivos XML de comprobantes. Este microservicio se encargó de procesar la estructura del archivo XML para extraer campos clave como el número de comprobante, los datos del emisor y receptor, montos totales, detalles de los ítems, impuestos y otros datos relevantes.

Tabla 9

Proceso y resultados del microservicio de extracción de datos de comprobante XML

Proceso	Resultados
Una vez que el área de contabilidad de la empresa asociada subía un archivo XML, el microservicio de extracción de datos se activaba para analizar el archivo y extraer la información necesaria. Este proceso aseguraba que todos los datos importantes fueran capturados y almacenados correctamente en la base de datos del ERP, reduciendo la posibilidad de errores manuales y garantizando la integridad de la información.	El microservicio fue validado mediante pruebas exhaustivas, confirmando que extraía los datos correctamente de una variedad de comprobantes en formato XML. Además, se observó una mejora significativa en la velocidad y precisión de la extracción de datos en comparación con los métodos manuales anteriormente utilizados.

La Tabla 9 muestra en qué parte del proceso se aplica el microservicio, además la implementación de este microservicio utilizó la librería ElementTree en Python, que permite una manipulación eficiente y precisa de archivos XML. Para mayor detalle de los microservicios, consultar el Capítulo III que hace referencia a la implementación del sistema.

b) Generación automática de asientos contables

El segundo microservicio implementado fue diseñado para automatizar la generación de asientos contables basados en los datos extraídos del XML. Este microservicio procesaba la información obtenida del primer microservicio y generaba los asientos contables correspondientes, asignando los montos a las cuentas contables de la universidad.

Tabla 10
Proceso y resultados del microservicio de generación automática de asientos contables

Proceso	Resultados
Después de que los datos del comprobante fueran extraídos y validados, el microservicio de generación de asientos contables creaba automáticamente las entradas contables necesarias. Este microservicio también permitía a los contables revisar y ajustar los asientos antes de su registro final.	Las pruebas realizadas sobre este microservicio mostraron que no solo generaba asientos contables con alta precisión, sino que también permitía una considerable reducción del tiempo que el personal de contabilidad dedicaba a esta tarea. La capacidad de generar asientos automáticamente y la opción de revisión y ajuste previo a su registro final proporcionaron un equilibrio óptimo entre automatización y control humano.

La Tabla 10 muestra en qué parte del proceso se aplica el microservicio y se infiere la eficiencia operativa, la reducción de errores y la precisión contable. Para más información consulte el Capítulo III en donde se habla de la implementación del sistema.

Discusión:

La implementación de estos microservicios cumplió con el objetivo específico de automatizar los procesos de extracción de datos y generación de asientos contables, dos actividades que anteriormente requerían una considerable cantidad de tiempo y esfuerzo manual. Con la utilización de estos microservicios, el sistema no solo mejoró la precisión de los datos y la eficiencia operativa, sino que también proporcionó una base sólida para futuras integraciones y mejoras dentro del ERP de la universidad.

El microservicio de extracción de datos demostró ser una herramienta crítica, capaz de manejar una variedad de formatos XML y extraer información crítica de manera rápida y precisa. Este enfoque no solo reduce el tiempo necesario para procesar cada comprobante, sino que también minimiza el riesgo de errores humanos, que son comunes en los procesos de entrada manual de datos.

Por otro lado, el microservicio de generación automática de asientos contables elevó significativamente la eficiencia del proceso contable. La automatización de esta tarea crítica permite que el personal contable se enfoque en actividades de mayor valor añadido, como la planificación financiera y el análisis contable, en lugar de tareas repetitivas y sujetas a errores.

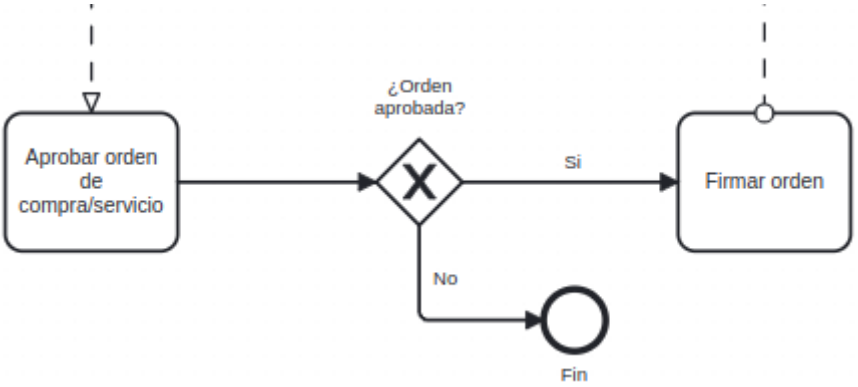
Además, la capacidad de estos microservicios para integrarse perfectamente con otros sistemas dentro del ERP proporciona una plataforma escalable que puede adaptarse a las necesidades futuras de la universidad. La modularidad de los microservicios también facilita su mantenimiento y actualización, asegurando que el sistema pueda evolucionar a medida que cambian los requisitos institucionales.

4.4 Objetivo Específico 3: Agilizar el Flujo de Trabajo de Aprobación de Comprobantes

El tercer objetivo específico del proyecto se centró en agilizar el flujo de trabajo relacionado con la aprobación de comprobantes Previamente, este proceso era altamente burocrático y centralizado, donde solo la autoridad administrativa tenía la responsabilidad de gestionar y aprobar todos los comprobantes de pago. Este enfoque centralizado no solo ralentizaba el proceso, sino que también generaba una carga administrativa significativa y aumentaba el riesgo de retrasos y errores en la gestión de los pagos.

a) Análisis del Proceso Anterior

Figura 25.1
Fragmento de la Figura 25 donde la autoridad administrativa gestiona y aprueba los comprobantes.



Como se ve en la Figura 25.1, en el proceso anterior, todo el flujo de aprobación de órdenes de servicio/compra de comprobantes estaba concentrado en una sola autoridad administrativa. Esta centralización resultaba en una serie de ineficiencias, ya que todos los comprobantes, independientemente de su origen, debían pasar por una única persona para ser aprobados. Esto no solo demoraba el proceso de aprobación, sino que también generaba cuellos de botella, especialmente cuando la autoridad administrativa no estaba disponible.

Las diferentes áreas de la universidad no tenían autonomía para gestionar sus propios comprobantes y gastos, lo que dificultaba un control adecuado a sus presupuestos y actividades. Esta dependencia de la autoridad centralizada limitaba la capacidad de respuesta y la eficiencia en la gestión financiera de cada área.

b) Implementación del nuevo sistema

El nuevo sistema fue diseñado para descentralizar el proceso de aprobación de comprobantes, permitiendo que cada autoridad dentro de la universidad gestione de manera autónoma los comprobantes asociados a sus actividades y presupuesto. Este cambió significó que, en lugar de depender de una única autoridad administrativa, cada área y facultad ahora tiene la capacidad de gestionar sus propios gastos en la empresa asociada.

El sistema permite que cada autoridad universitaria gestione directamente los comprobantes relacionados con sus actividades específicas y los presupuestos asignados. Esto incluye la revisión y aprobación de comprobantes, así como el monitoreo continuo de los saldos

disponibles en sus presupuestos. Esta funcionalidad no solo agiliza el proceso de aprobación, sino que también mejora la precisión y responsabilidad en la gestión financiera de cada área.

Aunque la gestión se ha descentralizado en términos de responsabilidades, el proceso en sí ha sido centralizado dentro del sistema ERP. Esto significa que, aunque múltiples autoridades ahora pueden aprobar comprobantes, todas las acciones se registran y gestionan dentro de un sistema centralizado, lo que mejora la transparencia, el control y la trazabilidad del proceso.

La descentralización de la aprobación, combinada con la centralización de la gestión en el ERP, ha resultado en una aceleración significativa del proceso de aprobación. Al eliminar la dependencia de una única autoridad y permitir que cada facultad o departamento gestione sus propios comprobantes, se han reducido los tiempos de espera y se ha mejorado la eficiencia del flujo de trabajo.

c) Resultados

- La carga administrativa de la autoridad central se ha reducido considerablemente, ya que ahora las responsabilidades de aprobación están distribuidas entre las distintas áreas de la universidad. Esto ha permitido una gestión más equilibrada y ha liberado tiempo para que la autoridad central se concentre en tareas más estratégicas.
- Las autoridades universitarias ahora tienen un control directo sobre los comprobantes y el presupuesto de sus respectivas áreas, lo que ha mejorado la responsabilidad financiera y ha facilitado una gestión más proactiva de los recursos.
- La centralización en el ERP ha mejorado la transparencia del proceso, permitiendo un seguimiento claro y preciso de todas las aprobaciones y pagos

realizados. Cada acción es registrada en tiempo real, lo que facilita auditorías y revisiones cuando sea necesario.

- La implementación de este sistema ha sido validada a través de encuestas y análisis de los tiempos de procesamiento de comprobantes antes y después de la implementación. Los resultados muestran una reducción significativa en el tiempo de aprobación, así como una mejora en la satisfacción de las autoridades que ahora tienen un mayor control sobre sus áreas de responsabilidad.

d) Discusión

La descentralización del proceso de aprobación de comprobantes ha demostrado ser una estrategia efectiva para mejorar la eficiencia y la responsabilidad en la gestión financiera de la universidad. Al permitir que cada autoridad universitaria gestione directamente sus propios comprobantes y presupuesto, se ha logrado no solo una aceleración del proceso de aprobación, sino también una mejora significativa en la calidad de la gestión administrativa.

Este nuevo enfoque ha eliminado los cuellos de botella que existían en el sistema anterior, reduciendo el tiempo total de procesamiento y aprobando comprobantes de manera más ágil y efectiva. La centralización en el ERP ha garantizado que, a pesar de la descentralización en la responsabilidad, el proceso siga siendo controlado y transparente, lo que es crucial para la rendición de cuentas y la eficiencia operativa de la universidad.

4.5 Discusión General

El proyecto ha logrado transformar significativamente los procesos administrativos y financieros en la universidad privada, específicamente en la gestión y aprobación de comprobantes. La digitalización ha sido clave para optimizar tareas que anteriormente eran manuales, lentas y propensas a errores. Este cambio ha mejorado la eficiencia operativa, permitiendo que los recursos tecnológicos se utilicen de manera más efectiva y que las

autoridades universitarias gestionen sus responsabilidades de manera más autónoma y con mayor precisión.

La descentralización del proceso de aprobación ha permitido que las distintas áreas y facultades de la universidad tengan un control más directo sobre sus presupuestos y actividades. Este enfoque no solo ha agilizado el flujo de trabajo, sino que también ha fomentado una mayor responsabilidad financiera entre las autoridades.

La implementación del sistema ha demostrado el valor de la digitalización en la gestión administrativa. Al eliminar las tareas repetitivas y propensas a errores, el sistema no solo ha mejorado la precisión de los datos y la eficiencia del proceso, sino que también ha aumentado la transparencia y trazabilidad de las operaciones. Cada acción realizada dentro del sistema es registrada y puede ser auditada, lo que refuerza la confianza en la gestión financiera de la universidad y facilita el cumplimiento de normativas y estándares de calidad.

Además, la capacidad del sistema para integrarse con otros componentes del ERP de la universidad asegura que la solución sea escalable y pueda adaptarse a futuras necesidades. Esta flexibilidad es crucial en un entorno universitario donde los procesos y requerimientos pueden cambiar rápidamente.

Conclusiones

- Se desarrolló un sistema digitalizado de control presupuestal y logístico para el pago de comprobantes en una universidad privada, utilizando un enfoque basado en microservicios. El sistema permitió descentralizar la gestión de comprobantes, otorgando a las diferentes autoridades universitarias la capacidad de gestionar y aprobar sus propios gastos por actividad y presupuesto. Esto redujo significativamente los tiempos de procesamiento, optimizando el flujo de trabajo en un 50% y eliminando los cuellos de botella asociados con la gestión centralizada.

- Se implementaron microservicios para la extracción de datos de comprobantes en formato XML y para la generación automática de asientos contables. Estos microservicios garantizaron la precisión en la captura y procesamiento de los datos financieros, integrándose de manera efectiva con el sistema ERP de la universidad. La digitalización de estos procesos no solo mejoró la exactitud de las operaciones, sino que también facilitó la auditoría y trazabilidad de las transacciones, lo que es esencial para mantener la integridad de la gestión financiera.
- El uso del *framework* ágil Scrum facilitó un desarrollo iterativo y centrado en el cliente, permitiendo la adaptación rápida a las necesidades cambiantes de los usuarios. Las historias de usuario y los criterios de aceptación definidos aseguraron que cada incremento del producto cumpliera con los estándares de calidad establecidos. Las ceremonias de Scrum, como las revisiones y retrospectivas, fueron cruciales para identificar y resolver problemas de manera oportuna, manteniendo el proyecto alineado con los objetivos del negocio.
- Se llevaron a cabo pruebas exhaustivas de integración, funcionales y de carga para validar la eficiencia y precisión del sistema. Las pruebas funcionales confirmaron que todos los componentes del sistema funcionaban correctamente, mientras que las pruebas de carga mostraron la capacidad del sistema para manejar un volumen creciente de transacciones sin pérdida de rendimiento. Estas pruebas fueron fundamentales para asegurar la fiabilidad del sistema en un entorno real.

Recomendaciones y trabajos futuros

- Aunque el sistema actual ya implementa medidas de seguridad, la incorporación de un token digital podría mejorar significativamente la protección de los datos sensibles y asegurar que solo usuarios autorizados puedan acceder y modificar la información.
- Optimizar las solicitudes a los microservicios reduciendo la cantidad de datos innecesarios transmitidos. Esto no solo mejoraría el rendimiento del sistema al disminuir el tiempo de respuesta del servidor, sino que también podría reducir la carga sobre la infraestructura de red, haciendo el sistema más eficiente y ágil.
- Ampliar el uso del sistema a otros procesos administrativos dentro de la universidad. El éxito del sistema en la gestión de comprobantes sugiere que podría ser adaptado para

optimizar otros flujos de trabajo, como la gestión de inventarios, la asignación de recursos, o la planificación académica.

- Realizar estudios de impacto a largo plazo para evaluar el efecto del sistema en la eficiencia operativa y financiera de la universidad. Esto podría incluir encuestas periódicas a los usuarios, análisis de costos y beneficios, y auditorías de desempeño para asegurar que el sistema continúe cumpliendo con los objetivos establecidos y se mantenga alineado con las necesidades de la institución.

REFERENCIAS BIBLIOGRÁFICAS

Alvarado, A. (2020). Sistema de Información Web desarrollado en Lenguaje de Programación PHP, para Controlar y Mejorar los Procesos Administrativos y la Gestión de los Pacientes del laboratorio “Cortes Buitrago S.A.S”. *Universidad Nacional Abierta y a Distancia*, 96.

Bobadilla, S. (2022). Desarrollo de un framework con una arquitectura basada en microservicios para el upgrade del sistema de planificación de recursos empresariales de la UCSM. *Universidad Católica de Santa María*, 90.

Bravo, H. (2022). Usabilidad de los servicios web de SUNAT para la emisión de la factura electrónica. *Universidad Nacional Agraria de la Selva*.

Huarcaya, F. (2022). Análisis y mejora del proceso de aprobación de facturas negociables en los bancos del Perú. *Universidad del Pacífico*, 72.

Leon, E., & Villagaray, S. (2023). Automatización del Sistema Contable y sus Matrices para la Eficiencia del Registro de Compras en la Empresa del Sector Deportivo. *Universidad Peruana de Ciencias Aplicadas*, 33.

Mamani, Z., Del Pino, L., & Gonzales, J. (2020). Arquitectura basada en Microservicios y DevOps para una. *Revista Industrial Data*, 10.

Martel Solis, L. (2021). Automatización de procesos para el registro de. *Universidad Nacional Mayor de San Marcos* , 53.

Ortiz, E. (2022). Automatización del proceso de recepción de las facturas electrónicas de las pymes en Colombia. *Universidad EAN*, 36.

Parraga, B., & Barreto, C. (2023). Sistema de gestión de facturas en el diseño de la integridad en los datos de las comisiones diferidas en el sector financiero. *Universidad Peruana de Ciencias Aplicadas*, 88.

Pokharel, P., & Vaidyam, P. (2020). A Study of User Story in Practice. *IEEE Xplore*, 5.

Pramitya, A., & Eko, B. (2020). Evaluation of Scrum Practice Matuiry in Software Development of Mobile Communication Application. *IEEE Xplore*, 6.

Ramadan, N., & Megahed, S. (2016). Requirements Engineering in Scrum Framework. *International Journal of Computer Applications*, 6.

Schwaber, K., & Sutherland, J. (18 de Noviembre de 2020). *Scrum.org*. Obtenido de La Guía Scrum 2020: https://www.scrum.org/resources/blog/la-guia-scrum-2020-scrum-guide-2020

Schwaber, K., & Sutherland, J. (2024). *Scrum.org*. Obtenido de The Scrum Guide : https://www.scrum.org/resources/scrum-guide

Smarty. (2024). *smarty Template Engine*.

Zayat, W., & Senvar, O. (2020). Framework Study for Agile Software Development Via Scrum and Kanban. *International Journal of Innovation and Technology Management*, 24.

ANEXOS

Figura 26
Historia de usuario 01 – Vincular factura XML a autoridad

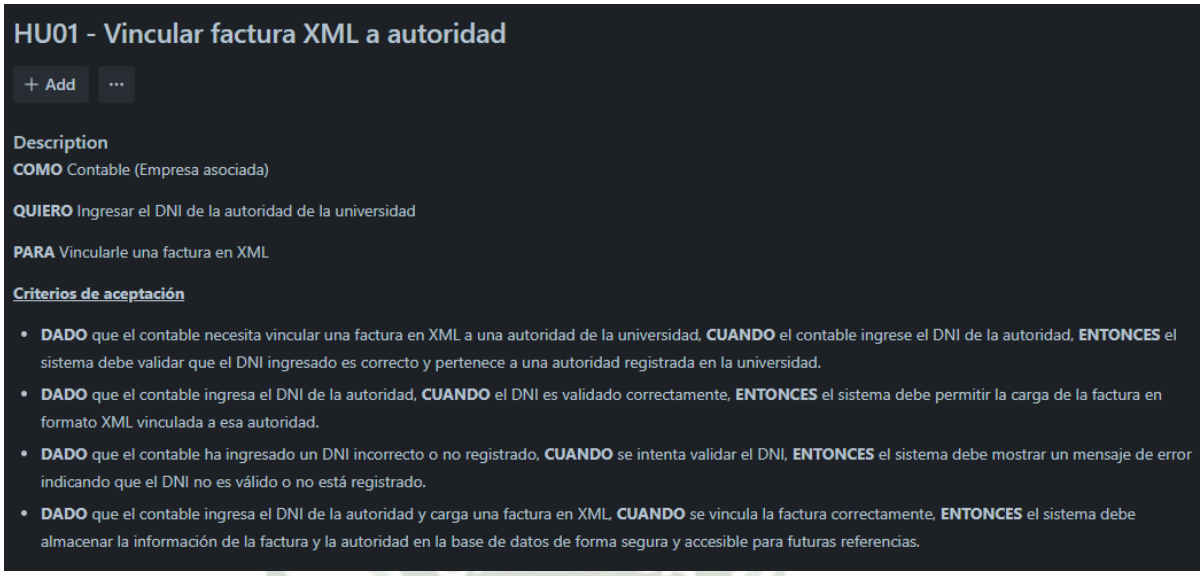


Figura 27

Historia de usuario 02 – Extraer datos de factura XML

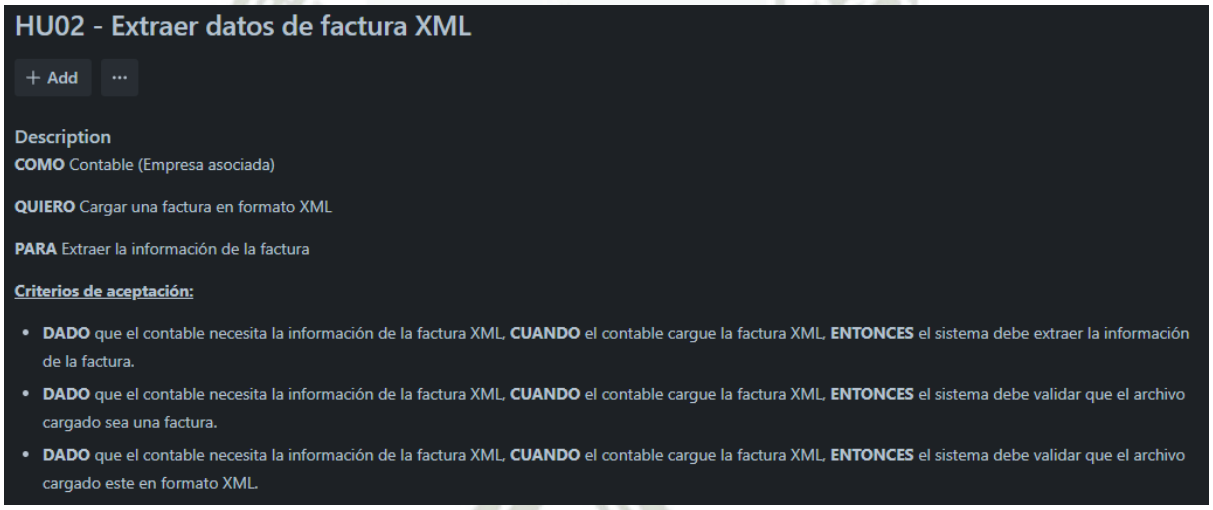


Figura 28

Historia de usuario 03 – Notificación de factura cargada

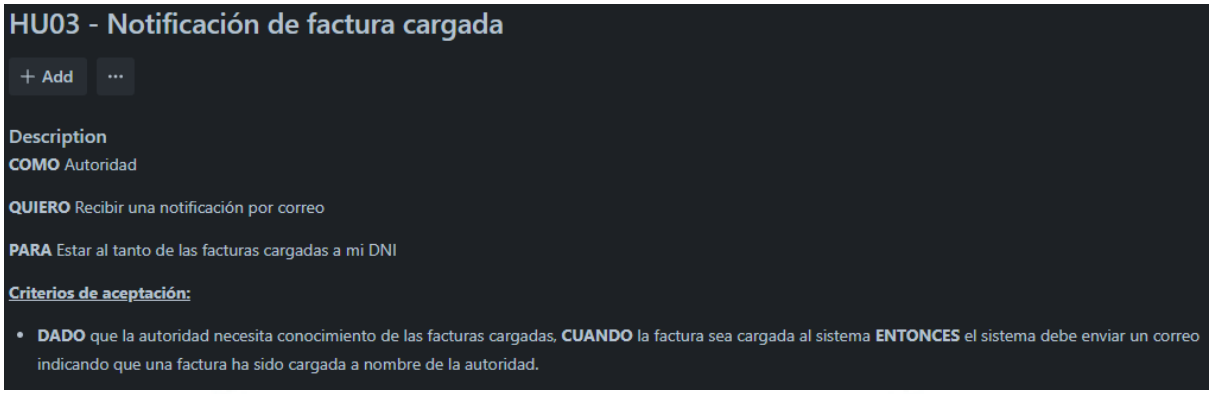


Figura 29

Historia de usuario 04 – Listado de facturas cargadas

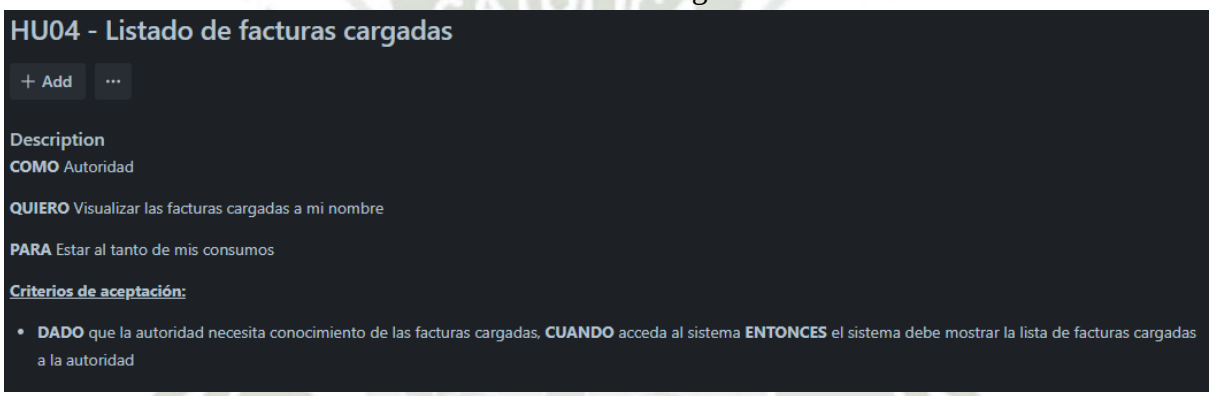


Figura 30

Historia de usuario 05 – Detalle de facturas cargadas

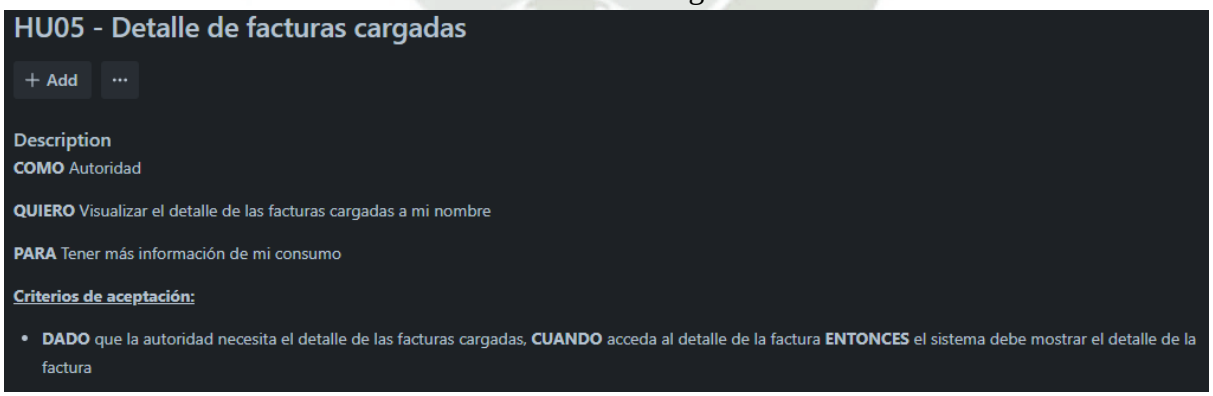


Figura 31

Historia de usuario 06 – Aprobación o denegación de consumo

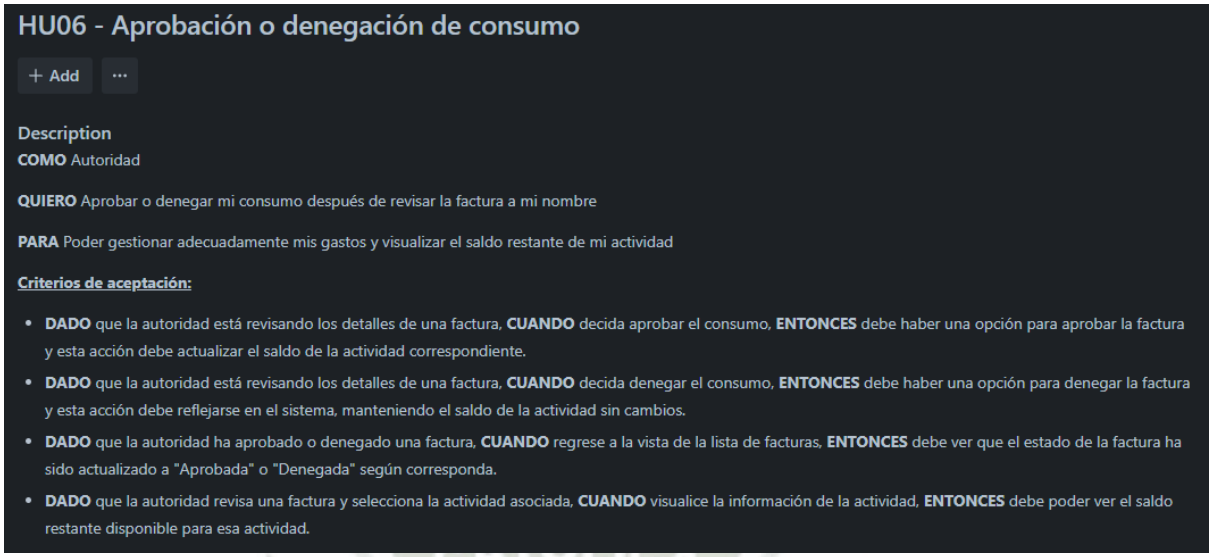


Figura 32
Historia de usuario 07 – Listado de facturas aprobadas

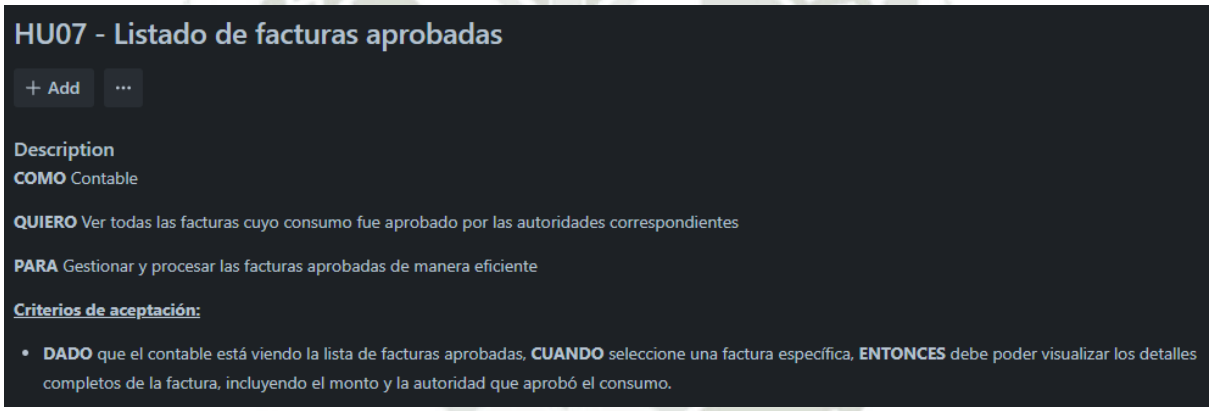


Figura 33
Historia de usuario 08 – Generación de asiento contable

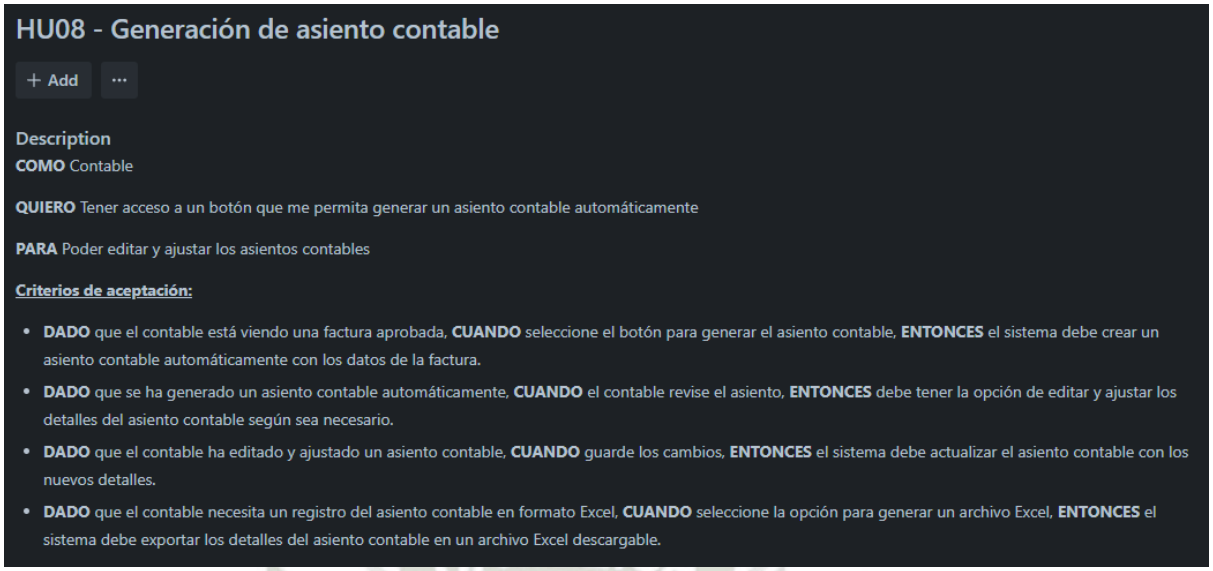


Figura 34

Historia de usuario 09 – Guardar la información de asiento contable contable