PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR
FACULTAD DE INGENIERÍA



TEMA:

DESARROLLO DE UN PROTOTIPO DE PLATAFORMA SAAS MODULAR PARA LA GESTIÓN EMPRESARIAL DE PEQUEÑAS Y MEDIANAS EMPRESAS EN ECUADOR

AUTOR:

FERNANDO JOSE NAVAS MOLINA

TUTOR:

Dr. NELSON SALGADO

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS DE INFORMACIÓN

QUITO D.M., JUNIO 2026

RESUMEN

Esta tesis refleja el diseño, desarrollo y lanzamiento de PymeFlowEc, una plataforma modular basada en el modelo Software como Servicio (SaaS), que busca atender las necesidades operativas de las pequeñas y medianas empresas (PYME) en Ecuador. El análisis comenzó identificando una contradicción que los datos demuestran de manera irrefutable: es importante destacar que, según información de Movistar Empresas (2023), el noventa por ciento de las PYME en el país planeaban invertir en herramientas para sus operaciones en 2024. Sin embargo, solo una de cada cinco PYME realmente utilizaba tecnologías de la información a un nivel avanzado (Banco Interamericano de Desarrollo [BID], 2025). Esta contradicción revela la existencia de obstáculos reales, tales como costos de implementación que exceden los presupuestos planificados, falta de personal técnico interno y, aún más importante, la insuficiente cantidad de soluciones adaptadas a las realidades operativas y fiscales de Ecuador.

Para abordar este desafío, se seleccionó un conjunto de tecnologías robusto y adecuado para un proyecto de tesis individual. El frontend se desarrolló con Angular 17, implementando el sistema reactivo de Independent Components y Signals, mientras que el backend utiliza Node.js 18 junto con Express 4.x. La gestión de datos se realiza mediante PostgreSQL 15 dentro del marco ERP. La plataforma también opera bajo un enfoque multiusuario, incorporando la columna `company_id`, que actúa como discriminador en todas las tablas operativas, asegurando que la información de cada empresa se mantenga separada. La capa de seguridad integra la autenticación JWT de dos tokens junto con un sistema de control de acceso basado en roles (RBAC) de cinco niveles. Los módulos funcionales abarcan facturación, gestión de inventario con clasificación de productos, gestión de clientes y proveedores, control de caja chica y

gestión de gastos, incluyendo la automatización de procesos recurrentes. Además, la plataforma cuenta con un sistema de diseño propio que utiliza tokens de color OKLch y ofrece soporte nativo para el modo oscuro. Por último, la infraestructura de despliegue se basa en Amazon Web Services, con el backend alojado en AWS Elastic Beanstalk y el frontend en AWS Amplify.

Estos hallazgos respaldan la viabilidad técnica y operativa de adoptar un ERP-SaaS multiusuario en Ecuador, con una API REST extensa que abarca aproximadamente 22 módulos de enrutamiento, junto con su documentación completa en OpenAPI 3.0 y más de 85 endpoints. PymeFlowEc proporciona una sólida base técnica sobre la cual desarrollar diversas soluciones de negocio que impulsan la digitalización del sector de las PYMES en el país.

Palabras clave: SaaS, ERP, PYMES, Ecuador, Angular 17, Node.js, PostgreSQL, multiusuario, JWT, AWS.

ABSTRACT

The objective of this thesis project is to create and develop a modular Software as a Service (SaaS) platform, called PymeFlowEc, designed to improve the operational management of small and medium-sized enterprises (SMEs) in Ecuador. This study is based on a significant inconsistency: while 90% of SMEs in Ecuador expressed their willingness to adopt digital tools by 2024 (Movistar Empresas, 2023), only 19% of these companies actually use advanced information technologies (Inter-American Development Bank [IDB], 2025). This disparity stems from structural obstacles such as implementation costs, which exceed the funds available to SMEs, a lack of trained technical personnel, and, above all, the absence of digital solutions specifically designed to enable SMEs to operate effectively within the current economic and tax framework in Ecuador.

This technology suite is designed for a single-author academic project, ensuring a balance between the robustness of the selected technologies and the practical constraints of working alone. The application's user interface is developed with Angular 17, using standalone components to segment it into more manageable modules. The server is built with Node.js 18 and Express 4.x, which serves as the framework. For data storage, the application uses PostgreSQL version 15 with an ERP schema. Finally, it incorporates a shared architecture through a common discriminating column (company_id) across all operational tables.. Security is handled through a dual-token JWT authentication scheme with a five-level RBAC model. Functional modules cover invoicing, inventory with product categories, customer and supplier management, petty cash, and expense control with automated recurring entries. The platform also incorporates a proprietary design system using OKLch color tokens with native dark mode support. Deployment relies on

Amazon Web Services, with the backend on AWS Elastic Beanstalk and the frontend on AWS Amplify.

ÍNDICE

RESUMEN 2
ABSTRACT 4
ÍNDICE 6
CAPÍTULO I: INTRODUCCIÓN 8
1.1. Planteamiento del Problema 8
1.2. Justificación 9
1.3. Objetivos 11
1.3.1. Objetivo General 11
1.3.2. Objetivos Específicos 11
1.4. Alcance del Proyecto 12
CAPÍTULO II: MARCO TEÓRICO 14
2.1. Software como Servicio (SaaS) 14
2.2. Arquitectura Multi-Tenant 15
2.3. Sistemas ERP (Enterprise Resource Planning) 16
2.4. Arquitectura de Aplicaciones Web Modernas 16
2.4.1. Patrón REST y Arquitectura de Tres Capas 16
2.4.2. Angular 17: Standalone Components y Signals 17
2.4.3. Node.js y Express como Backend API 18
2.4.4. PostgreSQL como Sistema de Gestión de Datos 18
2.5. Seguridad en Aplicaciones Web 19
2.5.1. JSON Web Tokens (JWT) en Contexto Multi-Tenant 19
2.5.2. Control de Acceso Basado en Roles (RBAC) 19
2.6. Metodología de Desarrollo: Scrum Adaptado 20
2.7. Computación en la Nube: AWS 20
CAPÍTULO III: METODOLOGÍA 22
3.1. Tipo de Investigación 22
3.2. Metodología de Desarrollo: Scrum Adaptado 22
3.2.1. Descripción de los Sprints 23
3.3. Herramientas y Tecnologías Utilizadas 24
3.4. Recolección de Requerimientos 25
3.5. Criterios de Validación y Calidad 26
CAPÍTULO IV: DESARROLLO DEL SISTEMA 28
4.1. Descripción General del Sistema 28
4.2. Arquitectura del Sistema 28

4.2.1. Arquitectura General de Tres Capas 28
4.2.2. Stack de Middleware del Backend 29
4.3. Módulos Funcionales del Sistema 31
4.4. Capa de Datos Modelo de Base de Datos 32
4.4.1. Descripción General del Esquema 32
4.4.2. Catálogo de Modelos 33
4.4.3. Triggers y Funciones PL/pgSQL 35
4.5. Capa de Servicios Backend 36
4.5.1. Estructura del Proyecto Backend 36
4.5.2. Sistema de Autenticación JWT de Doble Token 37
4.5.3. Sistema Multi-Tenant y Aislamiento de Datos 38
4.5.4. Sistema de Módulos Dinámicos 38
4.6. Capa de Presentación Frontend Angular 17 39
4.6.1. Arquitectura de la Aplicación Frontend 39
4.6.2. Gestión de Estado con Angular Signals 41
4.6.3. Sistema de Guards e Interceptores 41
4.6.4. Generación de PDFs en el Cliente 42
4.7. Despliegue e Infraestructura en AWS 42
4.7.1. Arquitectura de Despliegue 42
4.7.2. Seguridad Multicapa del Sistema 43
4.7.3. Observabilidad y Auditoría 44
4.8. Documentación de la API 45
CAPÍTULO V: RESULTADOS Y ANÁLISIS 52
5.1. Prototipo Funcional Desarrollado 52
5.2. Cumplimiento de Objetivos Específicos 52
5.3. Características Técnicas del Sistema Construido 53
5.4. Análisis de la Solución Propuesta 54
5.4.1. Pertinencia para el Contexto Ecuatoriano 54
5.4.2. Ventajas Arquitectónicas 55
5.4.3. Limitaciones del Prototipo 58
5.5. Contribución al Conocimiento 59
CONCLUSIONES 60
RECOMENDACIONES 62
BIBLIOGRAFÍA 64

CAPÍTULO I: INTRODUCCIÓN

1.1. Planteamiento del Problema

Las pequeñas y medianas empresas (PYME) son una parte esencial de la economía ecuatoriana, ya que representan más del 90% de las entidades productivas del país y generan un gran número de empleos formales. A pesar de su importancia, esto no ha conllevado una modernización de sus métodos internos. De hecho, la mayoría de estas organizaciones aún utilizan hojas de cálculo, registros en papel o software no interconectado, lo que genera discrepancias en los datos, pérdida de información y una capacidad limitada para tomar decisiones basadas en evidencia.

El estado de la esfera digital en Ecuador presenta una paradoja que requiere atención. La Encuesta de Adopción Digital de Movistar Empresas (2023) reveló que el 91% de las pequeñas y medianas empresas (pymes) del país tenían previsto invertir en digitalización para 2024, lo que representa un aumento de cuatro puntos porcentuales con respecto a 2022 (Telefónica Ecuador, 2024). Sin embargo, la Política Pública para la Transformación Digital 2025-2030 indica que solo el 18,7% de estas mismas empresas están utilizando tecnologías de la información avanzadas o de "alta calidad" (BID, 2025), como se muestra en la Figura 1. La discrepancia entre estas estadísticas se debe a obstáculos estructurales claramente definidos: altos costos, falta de personal técnico cualificado y falta de herramientas adecuadas al contexto tributario local.

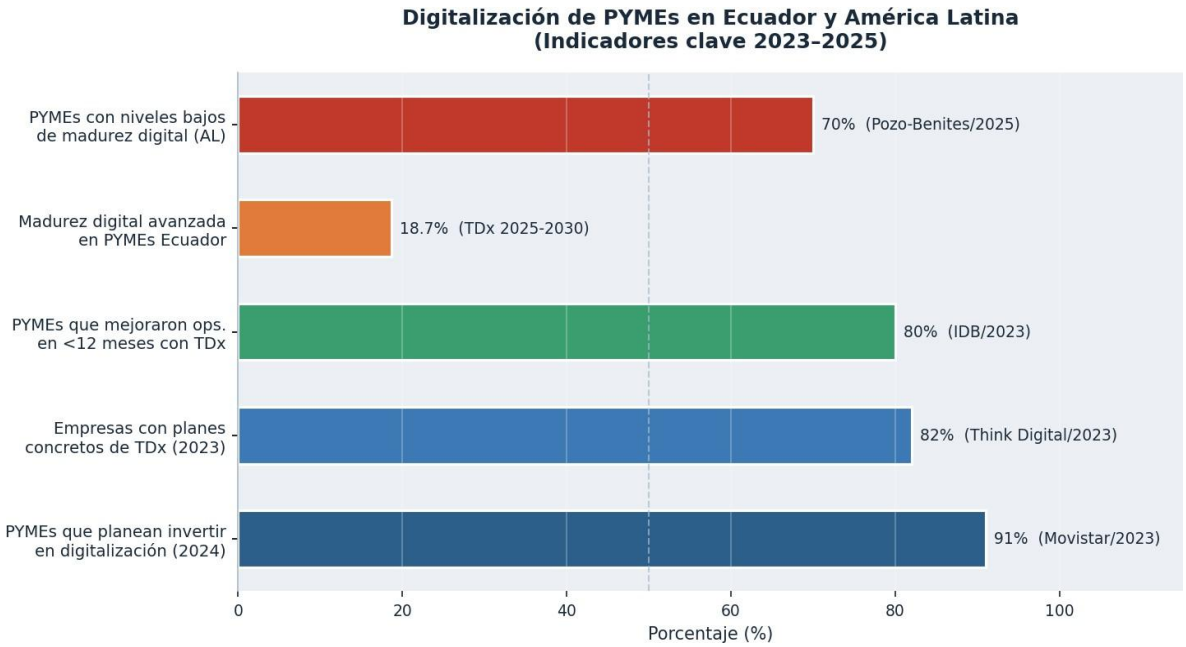


Figura 1. Indicadores de digitalización de PYMES en Ecuador y América Latina (2023–2025)

Pozo-Benites et al. (2025) han registrado que alrededor del setenta por ciento de las pequeñas y medianas empresas en América Latina enfrentan escasa madurez digital, con notorias variaciones entre diferentes naciones. En el caso de Ecuador, las limitaciones en la infraestructura y las capacidades institucionales complican significativamente la integración de tecnología en los procesos productivos. Los investigadores sostienen que en países como Chile, Colombia y Uruguay la cooperación entre el gobierno y el sector privado ha demostrado ser eficaz para cambiar esta realidad.

De esta manera, el modelo tradicional de ERP no es adecuado para las pequeñas y medianas empresas, ya que los costos de infraestructura y licencias son inalcanzables para la mayoría de ellas. Por ello, se hizo crucial crear e implementar una plataforma modular basada en el modelo de Software como Servicio (SaaS) que pueda satisfacer la necesidad de las PYMES de Ecuador de gestionar eficientemente sus procesos de facturación, control de inventarios, finanzas, caja chica y egresos. Esta solución web

debe ser accesible, escalable y estar en conformidad con la Ley Nacional de Régimen Tributario interno.

1.2. Justificación

El uso de herramientas digitales es un factor determinante para la competitividad y la sostenibilidad de las organizaciones. Salazar et al. (2025) reportaron que el 80% de las pymes que adoptaron soluciones tecnológicas mejoraron sus operaciones en menos de doce meses. En el caso de Ecuador, este potencial cobra aún mayor relevancia considerando la considerable brecha existente entre las empresas que requieren digitalización y aquellas que poseen las herramientas para implementarla.

El modelo SaaS se presenta como la opción más lógica para superar esta brecha: elimina la necesidad de costosas infraestructuras locales, reduce el tiempo necesario para su uso efectivo e implementa planes de pago recurrentes adaptados a las capacidades de las pequeñas empresas. Microsoft (s.f.) considera la computación en la nube como un factor de igualdad para organizaciones de distintos tamaños, ya que brinda a las pequeñas empresas acceso a capacidades de análisis y gestión que hasta ahora solo estaban disponibles para las grandes corporaciones.

Desde una perspectiva académica, este proyecto representa la aplicación práctica de áreas que rara vez convergen en una sola tesis: ingeniería de software, arquitectura de sistemas distribuidos, bases de datos avanzadas, ciberseguridad y computación en la nube. Además, utiliza tecnologías con alta demanda en el mercado laboral. El desarrollo de PymeFlowEc demuestra la viabilidad técnica y económica de implementar una plataforma ERP-SaaS multiusuario para el contexto empresarial ecuatoriano,

proporcionando un modelo replicable para futuras iniciativas que transformen la gestión de las pymes.

1.3. Objetivos

1.3.1. Objetivo General

Poner en funcionamiento un prototipo funcional de una plataforma SaaS modular, denominada PymeFlowEc, que permita a las pequeñas y medianas empresas ecuatorianas gestionar sus procesos de facturación, inventario, finanzas, caja chica y gastos, gracias a una arquitectura web escalable y multiusuario, desplegada en los servicios de computación en la nube de AWS.

1.3.2. Objetivos Específicos

a) Diseñar e implementar una arquitectura multi-tenant con aislamiento de datos por empresa mediante el esquema PostgreSQL "erp" y el patrón de identificación por company_id embebido en tokens JWT.

b) Desarrollar un sistema de autenticación y control de acceso basado en roles (RBAC) con cinco perfiles de usuario: PLATFORM_ADMIN, PLATFORM_STAFF, STORE_ADMIN, STORE_SELLER y STORE_WAREHOUSE.

c) Implementar los módulos funcionales de facturación electrónica, gestión de inventario, administración de clientes y proveedores, caja chica y control de egresos con presupuestos y egresos recurrentes.

d) Construir una interfaz de usuario Single-Page Application (SPA) con Angular 17, utilizando Standalone Components y Angular Signals para la gestión reactiva del estado de la aplicación.

e) Desplegar el sistema en servicios de nube de AWS, empleando AWS Elastic Beanstalk para el backend y AWS Amplify para el frontend, con la API de producción accesible en https://api.tesisfernandonavaspuce.es/api.

1.4. Alcance del Proyecto

El presente proyecto abarca el diseño, desarrollo, implementación y despliegue en producción de PymeFlowEc, un prototipo funcional de plataforma ERP-SaaS orientado a la gestión empresarial de las PYMES ecuatorianas. Los procesos cubiertos son: (a) facturación con emisión, cobros parciales y totales, y anulación con reversión automática del stock; (b) inventario con movimientos de entrada, salida y ajuste; (c) administración de clientes y proveedores con validación de documentos tributarios ecuatorianos (cédula y RUC); (d) caja chica con sesiones de apertura y cierre; (e) egresos con categorías, presupuestos y automatización de recurrentes; y (f) administración de plataforma con gestión de empresas, usuarios y habilitación dinámica de módulos.

La plataforma se encuentra operativa en producción sobre infraestructura de AWS, con la API disponible en https://api.tesisfernandonavaspuce.es/api y el frontend en AWS Amplify. El código fuente está disponible en tres repositorios públicos de GitHub: el frontend en github.com/fernandonavas75/pymeflowec-front, el backend en github.com/fernandonavas75/pymeflowec-backend, y la suite de pruebas automatizadas en github.com/fernandonavas75/pymeflowec-tests (Navas, 2026a, 2026b, 2026c).

Quedan fuera del alcance de esta investigación: la integración directa con el SRI para facturación electrónica, la implementación de notificaciones en tiempo real mediante WebSockets, el desarrollo de aplicaciones móviles nativas, la incorporación de módulos

de contabilidad y nómina, y la integración con pasarelas de pago. Estos aspectos constituyen líneas de trabajo futuro del proyecto.

CAPÍTULO II: MARCO TEÓRICO

2.1. Software como Servicio (SaaS)

El modelo SaaS propone una forma diferente de entregar y consumir aplicaciones: en lugar de instalar el software en los equipos del cliente, el proveedor lo aloja en sus propios servidores y lo pone a disposición a través del navegador web. Quien presta el servicio asume la responsabilidad de mantener la infraestructura, aplicar actualizaciones y garantizar la disponibilidad; el cliente paga una suscripción periódica o abona según el consumo real, sin desembolsos iniciales significativos (Sevensquaretech, 2025).

Ideadope (2025) identifica cuatro características que hacen del SaaS una propuesta atractiva en la práctica: acceso desde cualquier dispositivo con internet, escalado de recursos según la demanda real, ausencia de inversión en servidores propios, y actualizaciones que llegan de forma transparente sin intervención del usuario. Para una PYME ecuatoriana que difícilmente puede afrontar los costos de licencias y mantenimiento de un ERP convencional, estas características convierten al SaaS en la vía de acceso más realista a herramientas de gestión empresarial.

PymeFlowEc implementa este modelo de la siguiente manera: múltiples empresas denominadas tenants comparten una única instancia del servidor y una única base de datos, y la separación entre sus datos se garantiza a través de una columna company_id que actúa como discriminador en todas las tablas operativas del sistema.

2.2. Arquitectura Multi-Tenant

La arquitectura multiusuario permite que una única instancia de aplicación atienda a varios clientes simultáneamente, pero los datos de cada cliente permanecen completamente separados. Según Karanja (2025), el mecanismo principal es el siguiente: tras la autenticación del usuario, el sistema asocia el inquilino del usuario con la solicitud; todos los accesos posteriores a la base de datos se validan mediante ese identificador, lo que garantiza que los datos de un cliente sean inaccesibles para otro.

Aftab (2026) establece tres estrategias de aislamiento. La primera asigna una base de datos a cada cliente, lo que proporciona el máximo aislamiento, pero conlleva mayores costos operativos. La segunda utiliza una única base de datos con un esquema independiente para cada inquilino. La tercera Base de datos compartida, esquema compartido almacena todos los registros en las mismas tablas y los distingue mediante una columna discriminadora. PymeFlowEc sigue esta última opción: cada tabla operativa tiene la clave externa `company_id`, que identifica al propietario de cada registro, y la capa de servicio filtra todas las consultas por este campo antes de devolver los resultados.

La práctica más común consiste en incrustar información del inquilino dentro del propio JWT, de modo que no se requiera una consulta adicional a la base de datos con cada solicitud. PymeFlowEc lo hace: el ID de la empresa se incluye en el token, el middleware de autenticación lo recupera y agrega la condición a Sequelize dentro de la consulta. El middleware también se encarga de establecer la variable de sesión de PostgreSQL `app.current_user_id`, que luego es leída por los disparadores de auditoría para registrar el autor de cada operación de escritura.

2.3. Sistemas ERP (Enterprise Resource Planning)

Los sistemas ERP son plataformas tecnológicas diseñadas para integrar los principales procesos operativos y administrativos de una organización bajo un único sistema de gestión. Su característica principal es que los módulos de ventas, compras, inventario, contabilidad y recursos humanos se benefician de una única base de datos, lo que elimina la redundancia y garantiza la coherencia de la información entre los departamentos.

Esta filosofía de integración adoptada por PymeFlowEc se aplica a los procesos más relevantes para las pymes ecuatorianas: facturación mediante una gestión integral del ciclo de cobros; gestión de inventario mediante movimientos de stock estandarizados y categorización de productos, clientes y proveedores con validación de documentos fiscales ecuatorianos; y gestión financiera mediante informes de caja chica y gastos. Esta integración permite, por ejemplo, la reversión automática de la cancelación de una factura (lo que significa que no se requiere reversión manual, ya que se garantiza la consistencia).

2.4. Arquitectura de Aplicaciones Web Modernas

2.4.1. Patrón REST y Arquitectura de Tres Capas

La arquitectura de tres niveles divide el sistema en tres partes: la capa de presentación, la capa de lógica de negocio y la capa de datos. El patrón REST define el protocolo de comunicación entre capas mediante HTTP, utilizando una interfaz uniforme que admite recursos y verbos estándar. Esta separación me permite modificar el frontend sin alterar la lógica del backend o migrar el motor de base de datos sin que la interfaz detecte el cambio; además, permite un escalado diferenciado según la ubicación del cuello de botella.

2.4.2. Angular 17: Standalone Components y Signals

Angular es un framework de código abierto que persiste en su desarrollo por medio de Google para realizar Single-Page Applications mediante el uso del lenguaje TypeScript. La versión que está disponible, que es la 17, que es del tipo Long-Term Support, introduce dos novedades que, además, evidencian de forma palpable y efectiva la nueva forma de escribir código Angular en comparación con versiones anteriores.

Standalone Components: Cada componente define claramente sus propias dependencias sin necesidad de pertenecer a un NgModule. El blog de Angular (2025) concluye que este enfoque ya ha sido adoptado por el 90 % de los desarrolladores. Sus efectos prácticos son dobles: mejora de la eliminación de código no utilizado durante la compilación y eliminación de los NgModules de un solo componente, que solo añadían complejidad estructural al proyecto.

Angular Signals: una novedad que apareció en Angular 16 y se estabilizó en la 17 (Angular, 2025). Un Signal es un contenedor de valor que tiene dos operaciones básicas: lectura de su contenido y la modificación. Cuando un componente accede a un Signal, Angular hace que dicha dependencia quede registrada; si el valor cambia en el Signal, solo se vuelven a renderizar la parte del componente que lo consume. El modelo anterior basado en Zone.js disparaba el ciclo de detección de cambios sobre todo el árbol ante cualquier evento, lo que en aplicaciones de cierta complejidad podía representar un costo computacional significativo.

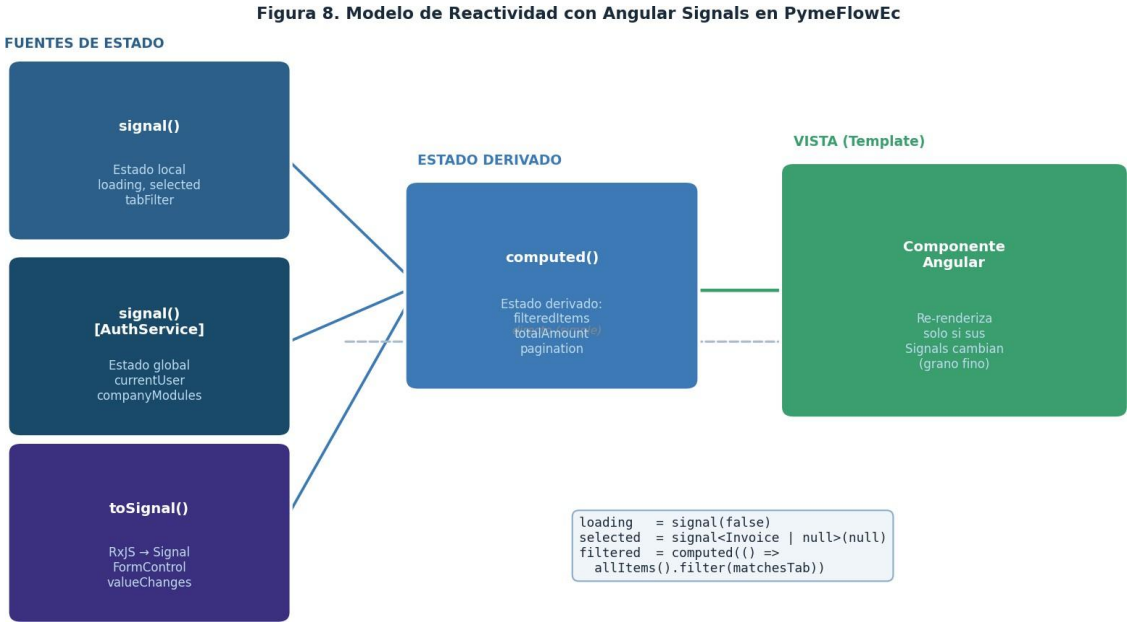


Figura 2. Modelo de reactividad con Angular Signals en PymeFlowEc

2.4.3. Node.js y Express como Backend API

Node.js es un entorno de ejecución de JavaScript del lado del servidor construido sobre el motor V8 de Chrome. Su arquitectura de I/O no bloqueante le permite manejar muchas conexiones concurrentes sin asignar un hilo por cada una, lo que lo hace adecuado para el tráfico típico de una API SaaS donde las peticiones son cortas y pasan la mayor parte del tiempo esperando respuestas de base de datos. Ideadope (2025) recomienda la combinación de Node.js con Express para este tipo de escenario.

2.4.4. PostgreSQL como Sistema de Gestión de Datos

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional de código abierto con más de tres décadas de desarrollo continuo (PostgreSQL Global Development Group, 2024). Su elección para PymeFlowEc responde a cuatro capacidades: el tipo JSONB para almacenar documentos auditables con índices SQL, los triggers y funciones PL/pgSQL para trasladar reglas de negocio al motor, los índices parciales para consultas

selectivas de bajo costo, y las restricciones CHECK/UNIQUE/FOREIGN KEY para garantizar integridad referencial a nivel de disco.

2.5. Seguridad en Aplicaciones Web

2.5.1. JSON Web Tokens (JWT) en Contexto Multi-Tenant

Un JWT es un objeto JSON firmado digitalmente que circula entre cliente y servidor en cada petición autenticada. PymeFlowEc utiliza el algoritmo HS256 (HMAC con SHA-256) de acuerdo con la RFC 7519. En las plataformas SaaS multiusuario, el JWT se ha convertido en el mecanismo estándar para identificar al usuario y la empresa a la que pertenece: el company_id y el rol se incluyen en la carga útil, y el backend resuelve a quién pertenece cada solicitud leyendo directamente ese token, sin consultar la base de datos en cada llamada (Shineos, 2025).

Nicholasdiamond (2024) destaca que una regla de seguridad básica consiste en almacenar los secretos JWT en variables de entorno y no en el código fuente. PymeFlowEc aplica esta regla: todos los secretos se almacenan en variables de entorno configuradas de AWS Elastic Beanstalk; no hay valores confidenciales en el código del repositorio

2.5.2. Control de Acceso Basado en Roles (RBAC)

Establece una capa intermedia entre usuarios y permisos. Los permisos se definen mediante roles que representan funciones empresariales, y a los usuarios se les asignan roles según sus responsabilidades. Esto permite que los cambios en la definición de roles se propaguen a todos los usuarios con ese rol asignado. Ideadope (2025) indica que el RBAC a nivel empresarial es un aspecto arquitectónico esencial en los diseños

multiusuario, ya que permite que cada cliente gestione sus propios usuarios de forma independiente de los demás usuarios.

2.6. Metodología de Desarrollo: Scrum Adaptado

Scrum es un marco de trabajo ágil que organiza el desarrollo en ciclos mínimos, llamados sprints, con roles bien definidos: el Product Owner, el Scrum Master y el Equipo de Desarrollo. También incluye ceremonias periódicas (reuniones, sesiones de planificación, etc.) y la creación de artefactos concretos. Su base epistemológica es el empirismo: las decisiones se toman a partir de la observación de los resultados de cada iteración de trabajo, lo que facilita ajustar la dirección del trabajo en función de los resultados observados y medidos.

Para este proyecto, se adoptó una versión adaptada de Scrum, conservando los principios de iteración e incremento, pero modificando las ceremonias y los roles para ajustarlos a las condiciones de un proyecto de tesis individual. El instructor, cuya función era la de supervisión académica, asumió el rol de Product Owner, validando los entregables al final de cada sprint. El ciclo siguió este orden: diseño del modelo de base de datos, implementación de los modelos de Sequelize, desarrollo de los endpoints REST, construcción de los componentes de Angular, integración y pruebas, y despliegue en AWS.

2.7. Computación en la Nube: AWS

Amazon Web Services es el proveedor líder del mercado de computación en la nube, que ofrece una amplia gama de recursos: computación, almacenamiento, bases de datos, redes y herramientas de implementación, bajo un modelo de pago por uso. Según Ideadope (2025), AWS es una de las mejores alternativas para implementar aplicaciones

SaaS multiusuario, y añade que las plataformas gestionadas como Elastic Beanstalk facilitan el aprovisionamiento y la promoción de nuevas versiones sin el tiempo de inactividad inherente al proceso de implementación.

PymeFlowEc utiliza dos servicios de AWS. El backend se ejecuta en Elastic Beanstalk, que administra las instancias EC2, la configuración del entorno Node.js 18, el balanceo de carga y la promoción de versiones a producción. El frontend utiliza AWS Amplify, que implementa archivos estáticos en la CDN global de Amazon, integra la implementación continua con GitHub y gestiona el concepto de aplicación de una sola página (SPA) redirigiendo las rutas perdidas a index.html.

CAPÍTULO III: METODOLOGÍA

3.1. Tipo de Investigación

Este trabajo se enmarca dentro de la investigación tecnológica aplicada, cuyo objetivo principal es el desarrollo de un software que aborde una situación específica del contexto empresarial ecuatoriano: la escasez de herramientas tecnológicas asequibles y escalables, adaptadas a las necesidades operativas y tributarias de las PYMES. El enfoque de la investigación es metodológico y sigue el paradigma de la ingeniería de software, en el que los criterios de evaluación se verifican mediante la aplicación de pruebas funcionales y de integración.

El enfoque predominante es cuantitativo en la llamada dimensión técnica los indicadores de calidad pueden medirse: tiempos de respuesta, cobertura de endpoints, códigos HTTP como “OK”, pero contiene elementos cualitativos en la fase de análisis de requisitos de la que también se deriva la revisión bibliográfica del contexto PYME ecuatoriano, así como de las mejores prácticas de arquitectura SaaS multi-tenant.

3.2. Metodología de Desarrollo: Scrum Adaptado

Se adoptó Scrum Adaptado como metodología de desarrollo para PymeFlowEc. Scrum Adaptado es una modificación de Scrum para este tipo de proyecto, reconociendo que, si bien las ceremonias y los roles de un Scrum estándar no pueden aplicarse estrictamente en proyectos de tesis individuales, sus principios fundamentales pueden mantenerse: desarrollo iterativo e incremental, entrega continua de funcionalidades verificables y retroalimentación periódica del tutor en el rol de "Product Owner". La estructura de sprint permitió un ritmo constante con entregables concretos al final de cada iteración.

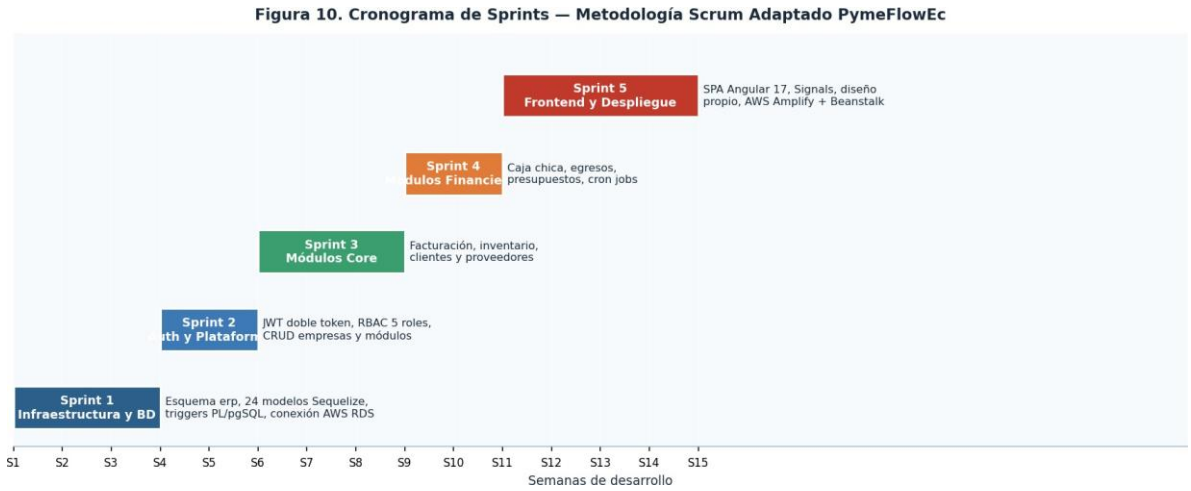


Figura 3. Cronograma de Sprints Metodología Scrum Adaptado en PymeFlowEc

3.2.1. Descripción de los Sprints

Tabla 1. *Organización de sprints del proyecto PymeFlowEc*

Sprint	Semanas	Objetivo	Entregables Principales
Sprint 1	1–3	Infraestructura y Base de Datos	Esquema erp en PostgreSQL 15, 25 modelos Sequelize con asociaciones (incluye ProductCategory), triggers PL/pgSQL de auditoría, conexión a AWS RDS
Sprint 2	4–5	Autenticación y Plataforma	Sistema JWT doble token, RBAC 5 roles, CRUD de empresas, usuarios, módulos y flujo de solicitud/aprobación
Sprint 3	6–8	Módulos Core de Negocio	Facturación con cobros y anulación, inventario con movimientos, clientes y proveedores con validación tributaria ecuatoriana
Sprint 4	9–10	Módulos Financieros	Caja chica con sesiones, egresos con categorías y presupuestos, automatización de egresos recurrentes con cron jobs
Sprint 5	11–14	Frontend y Despliegue	SPA Angular 17 con Standalone Components y Signals, sistema de diseño propio, despliegue AWS Amplify + Elastic Beanstalk

Nota. Elaboración propia. Cada sprint incluyó reuniones de revisión con el tutor al final de la iteración.

3.3. Herramientas y Tecnologías Utilizadas

La selección del stack tecnológico se realizó a partir de cuatro criterios: madurez técnica, disponibilidad de documentación oficial actualizada, nivel de adopción en la industria y compatibilidad con los servicios de AWS. La Tabla 2 detalla las tecnologías empleadas con sus versiones específicas de producción.

Tabla 2. *Stack tecnológico del sistema PymeFlowEc*

Componente	Tecnología / Herramienta	Versión	Rol en el Proyecto
Frontend Framework	Angular	17.3.0	SPA con Standalone Components y Signals
Lenguaje Frontend	TypeScript	5.4.0	Tipado estático en el frontend
UI Components	Angular Material	17.3.0	Componentes MatSnackBar, MatDialog, CDK
Utilidades CSS	Tailwind CSS	3.4.1	Clases utilitarias de diseño
Generación PDF	pdfmake	0.3.7	Facturas PDF generadas en el cliente
Backend Runtime	Node.js	18 LTS	Entorno de ejecución del servidor
Backend Framework	Express	4.18.3	Framework HTTP para la REST API
ORM	Sequelize	6.37.8	Mapeo objeto-relacional para PostgreSQL
Base de Datos	PostgreSQL	15	Motor de base de datos relacional
Cache y Rate Limiting	Redis (ioredis)	5.10.1	Rate limiting distribuido

Componente	Tecnología / Herramienta	Versión	Rol en el Proyecto
Autenticación	jsonwebtoken	9.0.3	Generación y verificación de tokens JWT
Hashing	bcryptjs	3.0.3	Hashing de contraseñas (cost factor 10)
Validación HTTP	express-validator	7.3.1	Validación y sanitización de inputs
Seguridad HTTP	Helmet	8.1.0	Cabeceras de seguridad HTTP
Logging	Winston	3.19.0	Logging estructurado multinivel
Documentación API	swagger-jsdoc + swagger-ui-express	6.2.8 / 4.6.3	OpenAPI 3.0 interactivo
Tareas Programadas	node-cron	4.2.1	Cron jobs para módulos y egresos recurrentes
Deploy Backend	AWS Elastic Beanstalk		Hosting gestionado del backend
Deploy Frontend	AWS Amplify		Hosting estático CDN del frontend
Control de Versiones	Git / GitHub		Versionado del código fuente

Nota. Las versiones corresponden a las utilizadas en el entorno de producción. Fuente: Navas (2026a, 2026b).

3.4. Recolección de Requerimientos

Los requerimientos del sistema se identificaron a través de tres fuentes complementarias: (a) revisión bibliográfica de sistemas ERP-SaaS para PYMES en el contexto latinoamericano y ecuatoriano; (b) análisis del marco tributario y regulatorio del Ecuador validación de cédula, RUC, tasas de IVA del SRI; y (c) revisión de las funcionalidades de plataformas SaaS comerciales similares en el mercado regional.

Los requerimientos se organizaron en el Product Backlog como historias de usuario, priorizados según el valor de negocio para las PYMES ecuatorianas. Los módulos de facturación e inventario se identificaron como los de mayor prioridad, seguidos por la gestión financiera (caja chica y egresos) y la administración de plataforma (empresas, usuarios y módulos dinámicos).

3.5. Criterios de Validación y Calidad

La calidad del prototipo se evaluó a lo largo de cuatro dimensiones complementarias:

a) Funcionalidad: se verificó que cada endpoint de la API y cada componente Angular implementara correctamente la lógica de negocio especificada. Cada endpoint fue ejercitado con casos de éxito y de error a través de la documentación Swagger interactiva en /api-docs.

b) Seguridad: se validó el aislamiento multi-tenant mediante pruebas de acceso cruzado entre empresas, se comprobaron las cabeceras HTTP de Helmet (HSTS, X-Frame-Options, CSP), se verificó la autenticación JWT y se confirmó la validación de documentos tributarios ecuatorianos.

c) Rendimiento: Se verificó que la API de producción operara dentro de plazos aceptables bajo carga normal mediante compresión de datos gzip; la limitación de velocidad impidió que las solicitudes innecesarias utilizaran por completo los recursos del servidor; se utilizó Redis para mantener el estado de la conexión del cliente; y se implementaron índices parciales en PostgreSQL.

d) Mantenibilidad: Se revisó la estructura de carpetas del proyecto, así como la cobertura de las especificaciones OpenAPI 3.0 para los módulos de enrutamiento (22 módulos). También se evaluó el manejo de errores mediante la compresión de los tipos de error, que son gestionados por un manejador de errores global. Este manejador de errores global recibe el objeto `req`, el objeto `res` y el mensaje de error. El error se registra y el caso de error correspondiente se maneja de acuerdo con las especificaciones del manejador de errores global.

CAPÍTULO IV: DESARROLLO DEL SISTEMA

4.1. Descripción General del Sistema

PymeFlowEc es una plataforma de gestión empresarial orientada a las pequeñas y medianas empresas (PYMES) ecuatorianas que explora el modelo SaaS. Los usuarios gestionan sus procesos de negocio a través de cualquier navegador web, sin necesidad de instalar nada adicional ni mantener su propio servidor. El modelo de la plataforma es un patrón ERP modular y multiusuario, ya que varias empresas comparten la misma infraestructura informática y base de datos PostgreSQL. Los límites de los datos se mantienen mediante el esquema, que incluye una columna `company_id` en cada tabla relacionada con los procesos operativos. Sevensquaretech (2025) señala que, en este modelo, la validación del usuario en cada solicitud es la primera línea de defensa, ya que si la validación falla incluso en un solo punto final, se rompe la promesa de aislamiento para todo el sistema.

El sistema se compone de tres repositorios independientes en GitHub. El backend (Navas, 2026b), construido sobre Node.js 18 y Express 4.x, expone una API RESTful con 22 módulos de rutas documentados en OpenAPI 3.0 y es accesible en https://api.tesisfernandonavaspuce.es/api. El frontend (Navas, 2026a), una SPA con Angular 17.3 usando Standalone Components y Angular Signals, está desplegado en AWS Amplify. La suite de pruebas automatizadas (Navas, 2026c) cubre los flujos críticos del backend con 439 casos de prueba.

4.2. Arquitectura del Sistema

4.2.1. Arquitectura General de Tres Capas

La Figura 4 muestra cómo se distribuyen las tres capas de PymeFlowEc entre los servicios de AWS. La capa de presentación es la SPA de Angular 17, alojada en AWS Amplify y distribuida a través de la CDN global de Amazon. Esa SPA se comunica exclusivamente con la API de Node.js y Express en AWS Elastic Beanstalk mediante peticiones HTTPS con el JWT en la cabecera Authorization. La API es el único punto de acceso a PostgreSQL 15 en AWS RDS (SSL obligatorio), con la comunicación canalizada a través de Sequelize ORM y un nodo Redis opcional para rate limiting entre instancias. Ideadope (2025) considera esta separación el cimiento para escalar cada capa de forma independiente.

Figura 2. Arquitectura General de PymeFlowEc (3 Capas)

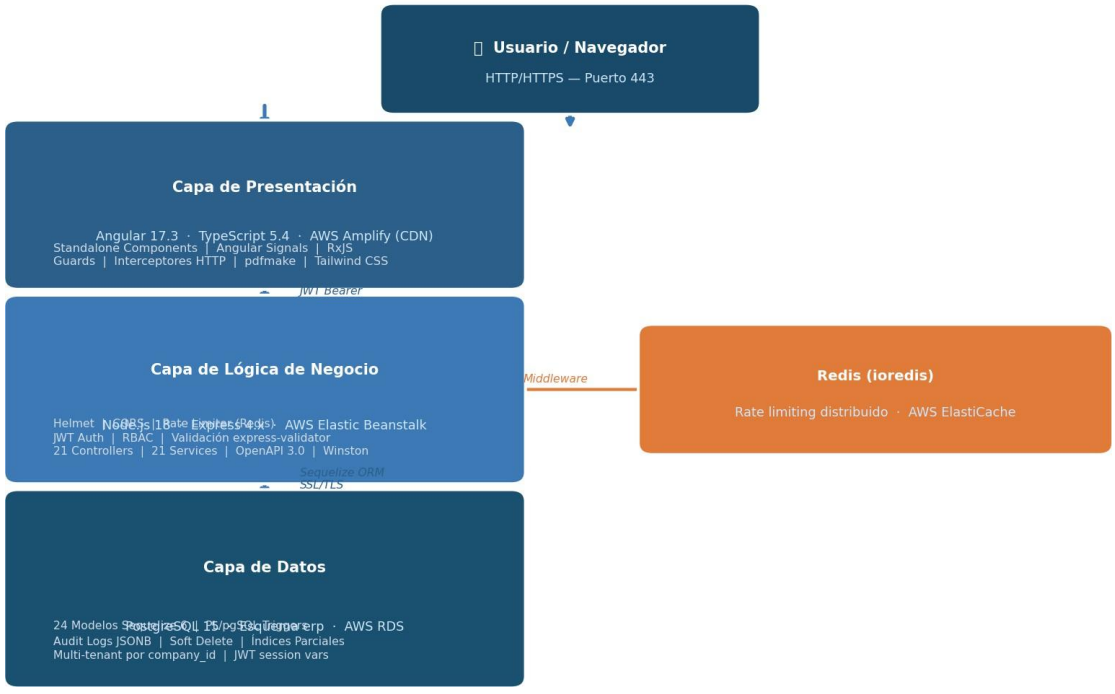


Figura 4. Arquitectura General de PymeFlowEc Tres Capas con Despliegue en AWS

4.2.2. Stack de Middleware del Backend

Antes de que una solicitud HTTP alcance el controlador de negocio, atraviesa una cadena de middleware en orden fijo. Cada componente tiene una responsabilidad acotada cabeceras de seguridad, límite de tasa, traza de auditoría, validación de token y si cualquiera rechaza la solicitud, los siguientes no se ejecutan. La Figura 5 muestra la secuencia completa.

Figura 7. Stack de Middleware del Backend — PymeFlowEc (Orden de Ejecución)

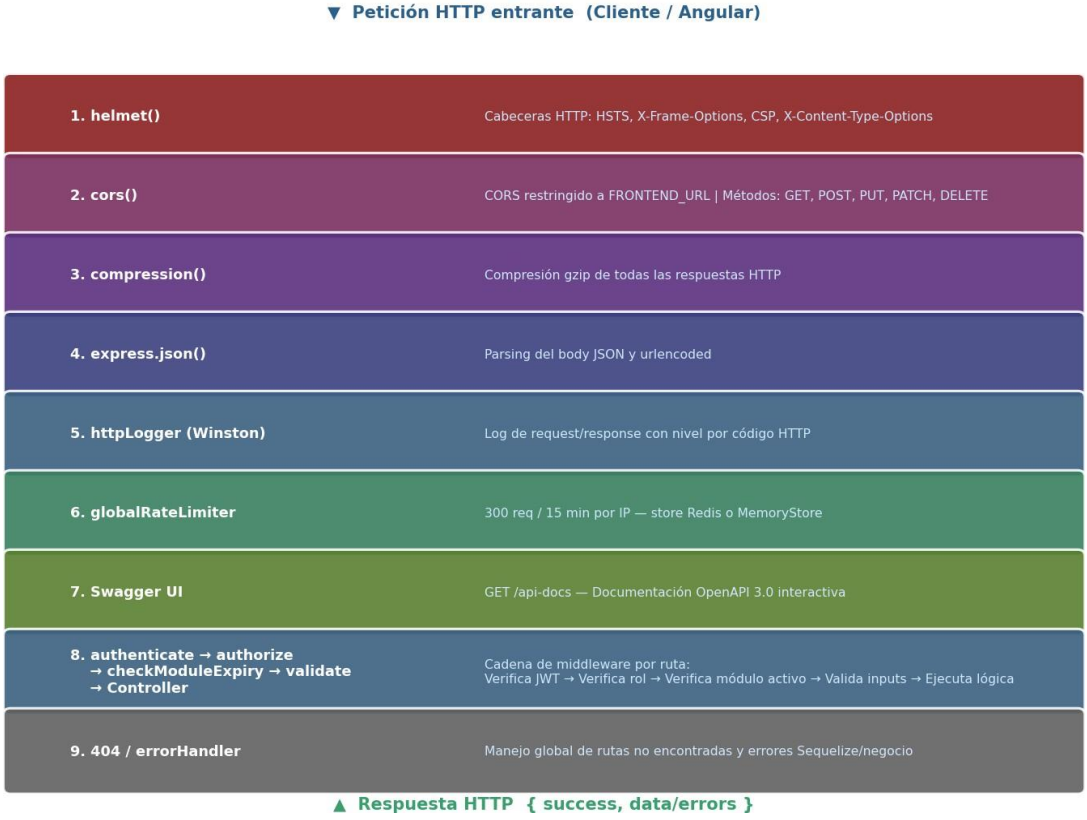


Figura 5. Stack de Middleware del Backend PymeFlowEc Orden de Ejecución

Helmet (v8.1.0) incluye automáticamente encabezados de respuesta HTTP de seguridad. El encabezado `Strict-Transport-Security` con un `max-age` de un año indica al navegador que solo puede acceder a la página web mediante HTTPS. `X-Frame-Options:DENY` indica que no es posible cargar la SPA dentro de un iframe en otras páginas web. `X-Content-Type-Options:nosniff` protege la capacidad del navegador para

detectar tipos de contenido. `Content-Security-Policy` permite aplicar una política restrictiva con respecto al origen de los scripts y estilos. El `globalRateLimiter` impone un límite de 300 solicitudes/counterRequests por dirección IP cada 15 minutos, y el contador se comparte entre Redis si se ejecuta más de una instancia.

4.3. Módulos Funcionales del Sistema

El sistema incluye módulos funcionales estructurados en dos grupos principales. El grupo de ventas abarca facturación, inventario con categorías de productos, clientes y proveedores. El grupo de administración interna incluye gestión financiera (caja chica, gastos presupuestados y recurrentes), configuración de herramientas (empresas, usuarios, módulos dinámicos) y observabilidad (auditorías, registros). Estos grupos se muestran en la Figura 6.

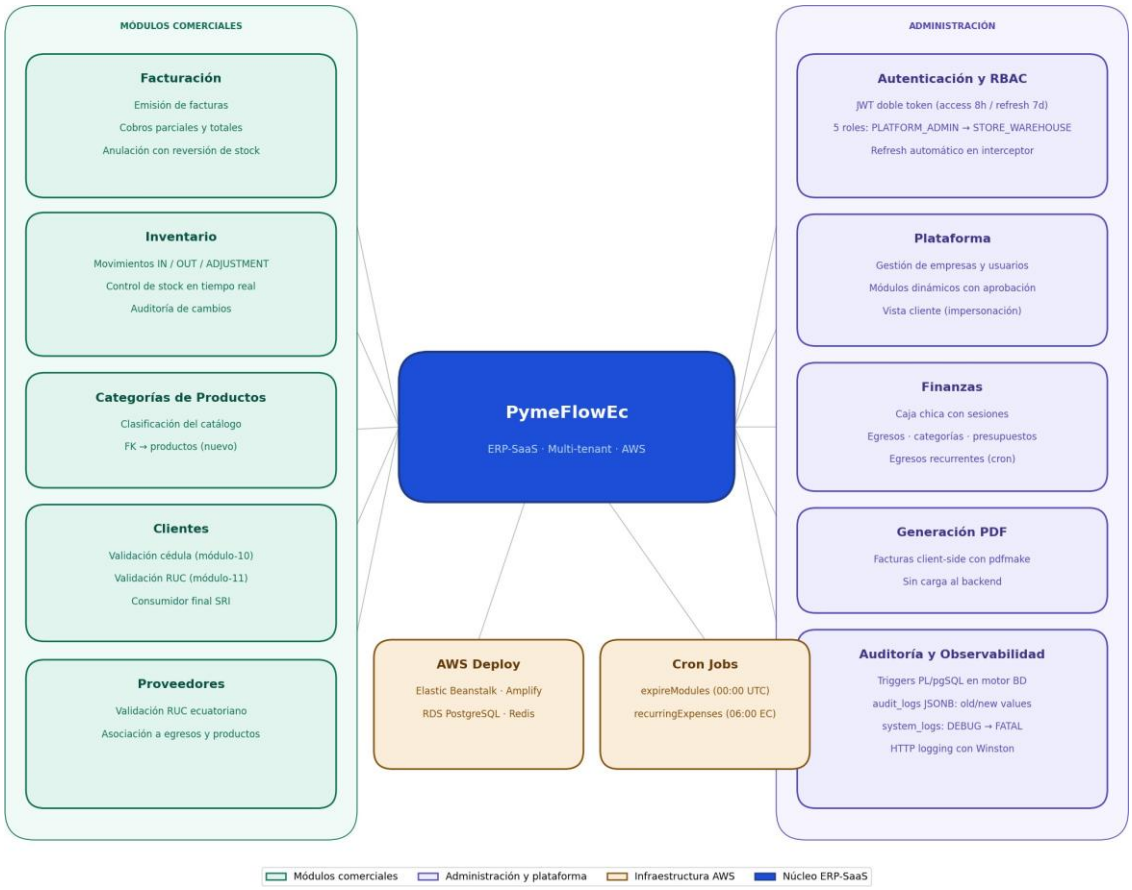


Figura 6. Mapa de Módulos Funcionales del Sistema PymeFlowEc

4.4. Capa de Datos Modelo de Base de Datos

4.4.1. Descripción General del Esquema

La base de datos se ejecuta en PostgreSQL 15, en el esquema erp que contiene 25 modelos Sequelize distribuidos en cuatro dominios funcionales (PostgreSQL Global Development Group, 2024). Los modelos Sequelize asumen tres convenciones globales: 'underscored' convierte los nombres de las columnas de camelCase a snake_case en disco; 'timestamps' mantiene automáticamente actualizados 'created_at' y 'updated_at' sin intervención del programador; y 'paranoid' (que requiere que la tabla sea recuperable) mantiene un registro de una fila eliminada estableciendo 'deleted_at' en lugar de eliminarla, lo que garantiza la trazabilidad. Además de estas convenciones, cada tabla

operativa también tiene 'company_id' (que admite multitenencia) como piedra angular del aislamiento multitenencia.

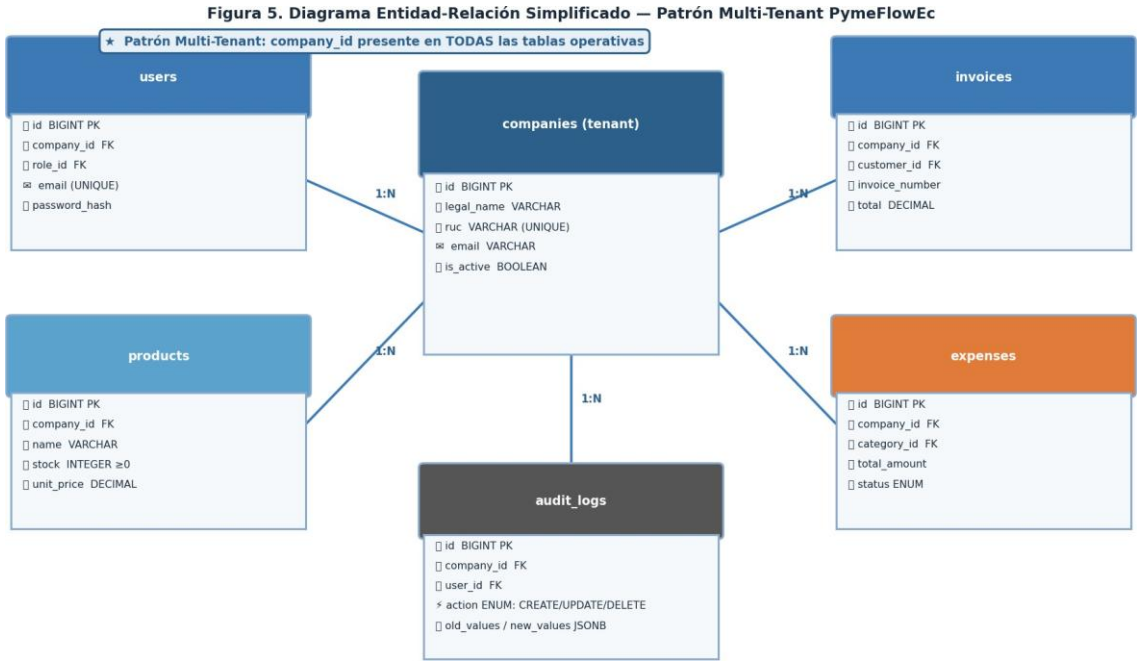


Figura 7. Diagrama Entidad-Relación Simplificado Patrón Multi-Tenant PymeFlowEc

4.4.2. Catálogo de Modelos

La Tabla 3 describe los 25 modelos Sequelize implementados, con sus características técnicas principales.

Tabla 3. *Catálogo de modelos Sequelize del sistema PymeFlowEc*

Modelo	Dominio	Características Técnicas Principales
Role	Identidad	5 roles: PLATFORM_ADMIN, PLATFORM_STAFF, STORE_ADMIN, STORE_SELLER, STORE_WAREHOUSE. ENUM scope: PLATFORM | STORE.
Company	Identidad	Tenant raíz. ruc VARCHAR con validación mod-11, legal_name, trade_name, is_active BOOLEAN. Soft delete.
User	Identidad	FK → Company, Role. password_hash (bcryptjs cost 10). Índice parcial sobre email activo. Soft delete.

Modelo	Dominio	Características Técnicas Principales
Module	Módulos	Catálogo global: code (UNIQUE), name, description, price_monthly, icon_name.
CompanyModule	Módulos	Habilitación de módulo: is_active, activated_at, expires_at. Índice compuesto (company_id, module_id).
CompanyModuleRequest	Módulos	Solicitud: status ENUM PENDING | APPROVED | REJECTED. FK: solicitante y revisor (users).
StoreCustomer	Comercial	Tipos: CEDULA | RUC | CONSUMIDOR_FINAL. Validación cédula (mod-10) y RUC (mod-11). Soft delete.
Supplier	Comercial	Proveedores: ruc VARCHAR, datos de contacto. Soft delete.
Product	Comercial	stock INTEGER (≥0 CHECK), unit_price DECIMAL, tax_rate_id FK, track_inventory BOOLEAN. Soft delete.
TaxRate	Comercial	Tasas de impuesto: name, percentage, is_active. Actualmente IVA 0% e IVA 15%.
ProductCategory	Comercial	Categorías de producto por empresa: name, description. Permite clasificar productos para facilitar la búsqueda y organización del catálogo. FK → companies, hasMany → Products.
Invoice	Operaciones	invoice_number (UNIQUE por empresa), status ENUM: PENDING | PARTIAL | PAID | CANCELLED, subtotal, tax_amount, total.
InvoiceDetail	Operaciones	Línea de factura: unit_price snapshot, quantity, discount_amount, subtotal. FK → Invoice, Product.
InvoicePayment	Operaciones	Cobro: payment_method ENUM (EFECTIVO | TRANSFERENCIA | TARJETA_DEBITO | TARJETA_CREDITO | CHEQUE), amount, reference.
InventoryMovement	Operaciones	Movimiento: type ENUM (IN | OUT | ADJUSTMENT), quantity, unit_cost, notes. Trigger actualiza stock en products.
PettyCash	Finanzas	Sesión: status ENUM (OPEN | CLOSED), opening_amount, current_balance, closing_amount_reported.
PettyCashMovement	Finanzas	Movimiento: type ENUM (EXPENSE | REPLENISH | ADJUSTMENT), amount, balance_after (snapshot acumulado).

Modelo	Dominio	Características Técnicas Principales
Expense	Finanzas	Egreso: status ENUM (PENDING | PARTIALLY_PAID | PAID | CANCELLED), total_amount, paid_amount. FK → ExpenseCategory.
ExpenseCategory	Finanzas	Categorías de egreso por empresa: name, description, is_active.
ExpensePayment	Finanzas	Pago de egreso: amount, payment_method, payment_date. FK → Expense.
ExpenseBudget	Finanzas	Presupuesto mensual por categoría: budget_amount, month, year. spent_amount calculado.
ExpenseRecurring	Finanzas	Plantilla recurrente: frequency ENUM (DAILY | WEEKLY | MONTHLY | YEARLY), next_execution_date.
AuditLog	Auditoría	Registro de escritura: entity_type, entity_id, action (CREATE|UPDATE|DELETE), old_values JSONB, new_values JSONB.
SystemLog	Auditoría	Log del sistema: level (DEBUG a FATAL), source (BACKEND|AUTH|CRON), details JSONB.

Nota. Elaboración propia. Todos los modelos operativos incluyen company_id (FK → companies) para el aislamiento multi-tenant. Fuente: Navas (2026b).

4.4.3. Triggers y Funciones PL/pgSQL

El mecanismo de auditoría de PymeFlowEc opera directamente desde la base de datos. La función `audit_trigger_function()` se ejecuta después de cada operación INSERT, UPDATE o DELETE en tablas operativas y escribe una entrada en `audit_logs` que contiene el tipo de acción, la entidad sobre la que se realiza, los datos anteriores y posteriores en formato JSONB, el usuario responsable y la marca de tiempo. El disparador consulta la variable de sesión de PostgreSQL `app.current_user_id`, que el middleware de autenticación rellena al inicio de cada transacción, para determinar el usuario. Shineos (2025) afirma que ubicar la auditoría en el motor y no en el código del servicio garantiza que no exista ningún endpoint mal escrito que pueda fallar al insertar estos registros.

Autenticación y registro: Las transacciones realizadas dentro de la aplicación en tablas operativas de la base de datos PostgreSQL se instrumentan de forma muy básica, utilizando un altavoz. Esto otorga la máxima responsabilidad a cada uno de los servicios del equipo de desarrollo.

En la misma capa, se implementan dos automatizaciones adicionales: set_updated_at() mantiene el campo updated_at y lo actualiza con los disparadores BEFORE UPDATE, y los disparadores de inventario ajustan el stock de productos, tanto en el caso de confirmar movimientos de ENTRADA/SALIDA como cuando se cancelan facturas, lo que garantiza la coherencia entre módulos independientemente de la ruta del flujo de ejecución del código.

4.5. Capa de Servicios Backend

4.5.1. Estructura del Proyecto Backend

El backend del sistema sigue un patrón MVC extendido con tres carpetas. `controllers/` contiene los 22 archivos que gestionan la capa HTTP: extraen los parámetros de la solicitud, llaman al servicio y devuelven la respuesta. `services/` contiene los 22 archivos que gestionan la lógica de negocio sin depender de `req/res` de Express; por lo tanto, permite realizar pruebas unitarias de la lógica de negocio sin necesidad de iniciar el servidor HTTP. validators/ agrupa 25 archivos basados en express-validator que definen la forma y sanitización de los parámetros de entrada para cada endpoint.

4.5.2. Sistema de Autenticación JWT de Doble Token

El sistema de autenticación emplea el patrón de doble token para resolver el dilema entre seguridad y usabilidad (Sevensquaretech, 2025): si el token de acceso tiene vigencia larga, una sesión comprometida queda activa por horas; si la tiene corta, el usuario debe reautenticarse con frecuencia. Al iniciar sesión se emiten dos tokens: un access token firmado con HS256 y válido durante ocho horas que viaja en la cabecera Authorization de cada petición y un refresh token con la misma firma y validez de siete días, que el cliente guarda aparte para solicitar un nuevo access token al expirar. La Figura 8 presenta el flujo paso a paso.

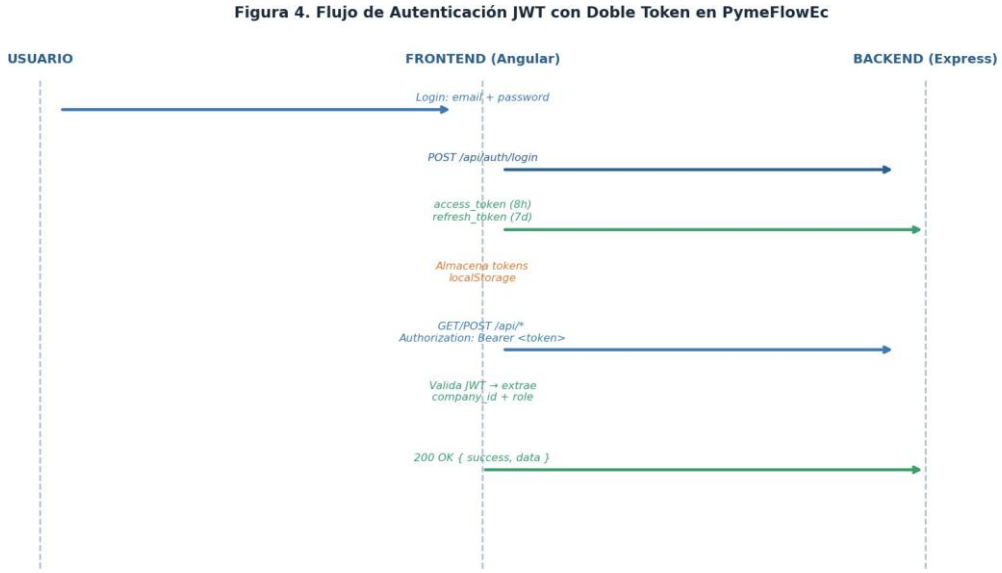


Figura 8. Flujo de Autenticación JWT con Doble Token en PymeFlowEc

El middleware authenticate realiza tres verificaciones por petición: comprueba la firma HS256 del access token, lo descarta si su tiempo de vida expiró y, cuando ambas pasan, extrae el payload user_id, company_id, role, scope para los controladores. Antes de ceder el paso al controlador, ejecuta SET LOCAL app.current_user_id sobre la conexión PostgreSQL de la transacción, permitiendo a los triggers de auditoría identificar

al autor sin que el código de los servicios deba transmitirlo explícitamente (Shineos, 2025).

4.5.3. Sistema Multi-Tenant y Aislamiento de Datos

La separación entre inquilinos se basa en un principio general que abarca toda la capa de servicio: ninguna consulta de Sequelize puede acceder a la base de datos sin incluir un filtro para company_id. Este valor se extrae del JWT mediante el middleware de autenticación y se añade a cada solicitud. Incluso si un usuario de la empresa A puede obtener el ID de un recurso de la empresa B, la consulta no devolverá esa fila, ya que el filtro implícito añadirá company_id como condición de búsqueda adicional (Aftab, 2026).

El sistema también incorpora un modo de vista de cliente correspondiente al rol PLATFORM_ADMIN, destinado a soporte y monitorización, que permite el acceso temporal al contexto de la empresa sin afectar la duración de la sesión. Mediante el clientViewInterceptor del frontend, la intención ?company_id=X se añade automáticamente a todas las solicitudes cuando el modo de vista está activo. El backend solo acepta los parámetros de rol PLATFORM_ADMIN o PLATFORM_STAFF; cualquier otro parámetro genera un error HTTP 403.

4.5.4. Sistema de Módulos Dinámicos

Cada empresa puede activar y desactivar módulos individualmente, en función de sus necesidades y capacidad de pago. Una empresa que utiliza únicamente facturación e inventario no necesita tener habilitado el módulo de egresos recurrentes; el sistema

mantiene esa sección desactivada y bloquea sus endpoints hasta que la empresa lo solicite explícitamente. La Figura 9 muestra el ciclo completo de solicitud y aprobación.

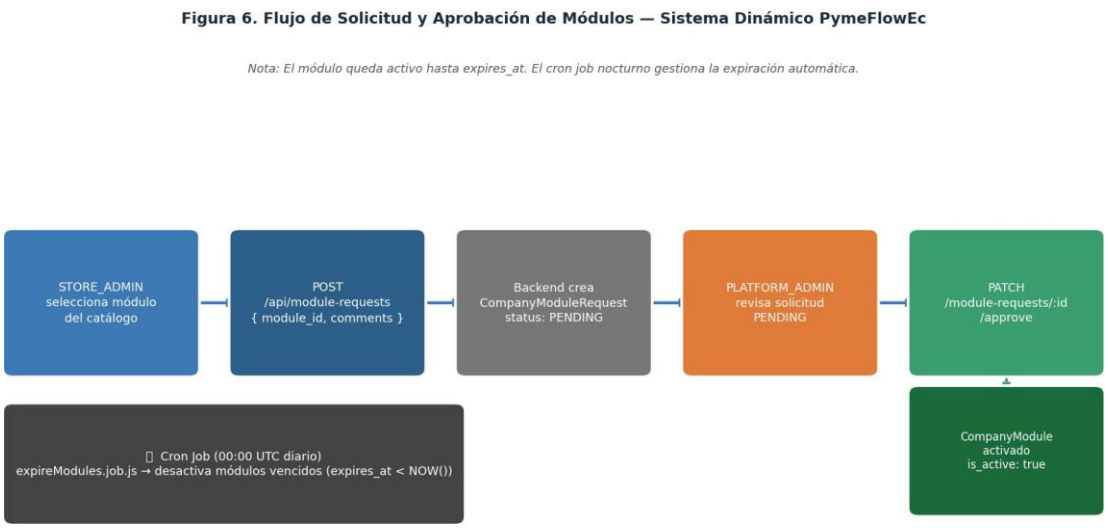


Figura 9. Flujo de Solicitud y Aprobación de Módulos Dinámicos en PymeFlowEc

Una tarea programada denominada expireModules.job.js se ejecuta cada noche a las 00:00 UTC via node-cron y recorre company_modules buscando habilitaciones vencidas; para cada una establece is_active en false y registra la operación en system_logs. El middleware checkModuleExpiry, montado en todas las rutas de negocio, devuelve HTTP 403 si el usuario intenta acceder a un endpoint de un módulo que ya no está activo para su empresa.

4.6. Capa de Presentación Frontend Angular 17

4.6.1. Arquitectura de la Aplicación Frontend

El frontend es una SPA construida con Angular 17.3 bajo el paradigma de Standalone Components. Los quince módulos de funcionalidad auth, landing, dashboard, clients, products (con subcarpeta product-categories), suppliers, invoices, tax-rates, users, finance (con subdivisiones petty-cash, expenses, expense-categories, expense-budgets,

expense-recurring, finance-dashboard), reports, settings, module-requests, companies y platform no se incluyen en el bundle inicial; cada uno se descarga únicamente cuando el usuario navega a una ruta que lo requiere. Para una PYME ecuatoriana que opera con conectividad variable entre regiones, esto acelera la primera pantalla y evita la descarga de código que puede no utilizarse durante toda la sesión (BID, 2025).

La interfaz incluye un sistema de diseño personalizado definido en src/styles.scss (aproximadamente 1700 líneas), que determina por completo el estilo visual de la plataforma sin depender de bibliotecas de diseño externas. Los tokens de color utilizan el espacio perceptivo OKLch, lo que garantiza un contraste consistente entre los elementos al alternar entre los modos claro y oscuro. Los acentos visuales se pueden modificar en tiempo de ejecución mediante el atributo data-accent del elemento raíz. El sistema también define componentes CSS reutilizables .ds-table, .btn, .card, .kpi-grid, .modal-backdrop, .drawer, así como un modo de densidad compacta para pantallas pequeñas.

El código se estructura en tres directorios o carpetas:- core/ contiene los tres protectores de ruta, los tres interceptores HTTP, las interfaces de TypeScript y los más de 30 servicios inyectables para comunicarse con la API.- shared/ contiene componentes reutilizables entre módulos (tablas paginadas, modales, selectores de cliente).- features/ contiene cada módulo funcional y su propia ruta de carga diferida.

El proceso de inicio no pasa por un AppModule convencional: main.ts llama directamente a bootstrapApplication(AppComponent, appConfig), siguiendo la ruta que utiliza el 90 % de los desarrolladores de Angular (Angular, 2025).

4.6.2. Gestión de Estado con Angular Signals

La gestión del estado se basa completamente en las señales nativas de Angular 17, sin necesidad de bibliotecas externas. Cuando se produce una señal, Angular sabe exactamente qué componentes la consumen y provoca que se vuelvan a renderizar, a diferencia del modelo de Zone.js, que activaba la detección de cambios en todo el árbol cada vez que un evento del navegador provocaba un cambio (Angular, 2025).

El estado se comparte en tres niveles: variables locales de componentes, como signal() para datos que solo afectan a ese componente; valores derivados con computed() que se recalculan cada vez que cambia una de sus dependencias; y estado que se comparte en servicios singleton (AuthService, CompanyModulesService) inyectados donde sea necesario.

4.6.3. Sistema de Guards e Interceptores

Tres mecanismos de control funcionales gestionan el acceso a las rutas: auth.guard comprueba que el token sea válido y esté almacenado en localStorage, y que la sesión esté activa (si no se cumple alguna de estas condiciones, redirige a la ruta de inicio de sesión); permission.guard evalúa las restricciones específicas de rol o alcance que se declaran dentro de la ruta; y role.guard gestiona las rutas que se basan en un único rol, que es la lógica más sencilla.

En primer lugar, tokenInterceptor es un interceptor de solicitud y respuesta que incluye autorización en cada solicitud y permite actualizar el token de acceso al recibir

respuestas HTTP 401; en segundo lugar, clientViewInterceptor es un interceptor que agrega ?company_id=X cuando PLATFORM_ADMIN tiene habilitada la opción de vista de cliente; y finalmente, errorInterceptor es un interceptor de respuesta que captura errores HTTP y muestra notificaciones usando MatSnackBar de Angular Material.

4.6.4. Generación de PDFs en el Cliente

La factura en PDF se genera directamente en el navegador del usuario con pdfmake 0.3.7, sin ninguna intervención del servidor. Esta decisión evita cargar el backend con trabajo de maquetado que el cliente puede asumir, y garantiza que la descarga funcione incluso con conectividad inestable. El documento incluye los datos de la empresa emisora con su logotipo, la información del cliente, las líneas de detalle con precio unitario, descuentos y subtotal, el resumen tributario (IVA 0% o 15%) y el historial de cobros parciales con sus métodos de pago.

4.7. Despliegue e Infraestructura en AWS

4.7.1. Arquitectura de Despliegue

PymeFlowEc se despliega en Amazon Web Services utilizando dos servicios gestionados que eliminan la necesidad de administrar manualmente la infraestructura subyacente, como se ilustra en la Figura 10.

Figura 9. Arquitectura de Despliegue en AWS — PymeFlowEc

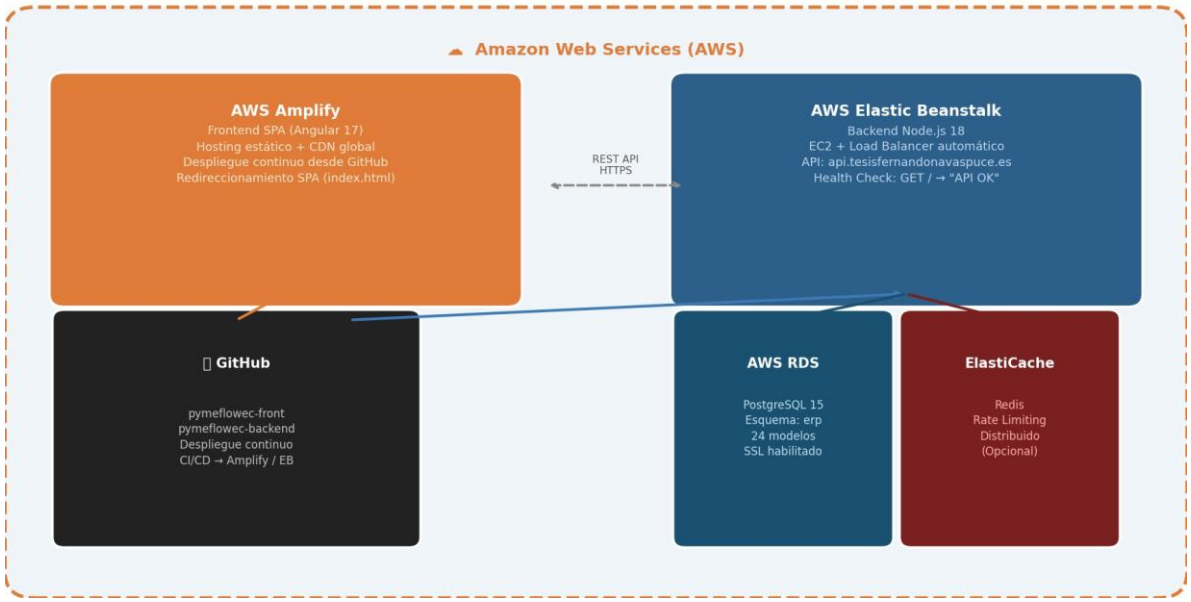


Figura 10. Arquitectura de Despliegue de PymeFlowEc en Amazon Web Services (AWS)

AWS Elastic Beanstalk (Backend): gestiona automáticamente las instancias EC2, configura el entorno Node.js 18, distribuye el tráfico mediante el balanceador de carga de AWS y promueve nuevas versiones a producción con cada build aprobado desde GitHub. La ruta GET / responde sin autenticación para funcionar como health check del balanceador. La API en producción está disponible en https://api.tesisfernandonavaspuce.es/api y la documentación Swagger en /api-docs.

AWS Amplify (Frontend): aloja los archivos estáticos de la SPA y los distribuye globalmente mediante la CDN de Amazon, entregándolos desde el punto de presencia más cercano al usuario. Cada commit en la rama principal dispara un build y despliegue automático, manteniendo el frontend de producción sincronizado con el código fuente. Para sostener el modelo SPA, Amplify redirige hacia index.html cualquier ruta no encontrada (Navas, 2026a).

4.7.2. Seguridad Multicapa del Sistema

La seguridad está conformada a partir de capas que se refuerzan mutuamente. En lo que respecta al transporte, toda comunicación viaja cifrada: HTTPS entre usuario y servidor,; SSL obligatorio en AWS RDS. En la propia aplicación las cabeceras de Helmet HSTS, X-Frame-Options, CSP, X-Content-Type-Options minimizan los vectores del OWASP Top 10 La autenticación de las contraseñas se almacena como hashes bcryptjs con cost factor 10, lo que hace computacionalmente inviable su recuperación incluso con acceso directo a la base de datos (Nicholasdiamond, 2024).

La validación de documentos tributarios es un elemento distintivo que caracteriza al contexto ecuatoriano. El módulo utils/ecuadorId.js realiza la implementación de los algoritmos oficiales del SRI, que son el módulo-10 para la cédula (donde se valida el 9º dígito) y el módulo-11 para el RUC (donde las multiplicaciones de los coeficientes son diferentes según el tipo de contribuyente: persona natural, sociedad privada o entidad pública).

4.7.3. Observabilidad y Auditoría

La plataforma aplica tres niveles de observabilidad que permiten revisar lo sucedido en producción. Primero: Winston registra el método, la URL, el código HTTP devuelto, la duración en milisegundos, la dirección IP de origen y el User-Agent de la solicitud, con los siguientes tipos de registro: "info" (para 2xx y 3xx) y "error" (para 4xx y 5xx). Segundo: los disparadores PL/pgSQL almacenan el estado anterior y posterior de cada operación en audit_logs, con un enlace directo al usuario que la ejecutó.. La tercera: system_logs registra eventos de autenticación, ejecuciones de cron jobs y errores no manejados; el campo details en JSONB permite agregar contexto adicional sin alterar el esquema.

4.8. Documentación de la API

El backend publica documentación interactiva en GET /api-docs, generada por swagger-ui-express 4.6.3 a partir de anotaciones JSDoc en los archivos de rutas. El esquema cumple OpenAPI 3.0 y describe parámetros, request body, respuestas posibles y requisitos de autenticación de cada endpoint. Durante el desarrollo, esta documentación actuó como contrato vivo entre backend y frontend: cuando se implementó cada feature, las firmas de las llamadas a la API se tenían disponibles sin necesidad de coordinaciones adicionales (Swagger.io, 2024).

Tabla 4. *Módulos de rutas de la API REST de PymeFlowEc*

Módulo de Ruta	Prefijo	Endpoints Principales
Autenticación	/api/auth	POST /login, POST /refresh, POST /logout, POST /forgot-password
Empresas	/api/companies	GET, POST, GET/:id, PUT/:id, PATCH/:id/toggle-status
Usuarios	/api/users	GET, POST, GET/:id, PUT/:id, PATCH/:id/change-password
Módulos de Plataforma	/api/platform/modules	GET (catálogo público y privado)
Módulos por Empresa	/api/company-modules	GET, PATCH/:id/activate, PATCH/:id/deactivate
Solicitudes de Módulo	/api/module-requests	GET, POST, GET/all, PATCH/:id/approve, PATCH/:id/reject
Clientes	/api/customers	GET (con búsqueda), POST, GET/:id, PUT/:id, DELETE/:id (soft)
Proveedores	/api/suppliers	GET, POST, GET/:id, PUT/:id, DELETE/:id (soft)
Categorías de Producto	/api/product-categories	GET, POST, GET/:id, PUT/:id, DELETE/:id (soft)
Productos	/api/products	GET, POST, GET/:id, PUT/:id, DELETE/:id (soft)
Tasas de Impuesto	/api/tax-rates	GET, POST, GET/:id, PATCH/:id/toggle

Módulo de Ruta	Prefijo	Endpoints Principales
Facturas	/api/invoices	GET (paginado), POST, GET/:id, PATCH/:id/cancel
Cobros de Factura	/api/invoice-payments	POST, GET/:invoiceId/payments
Inventario	/api/inventory	GET (movimientos), POST (movimiento), GET/stock-summary
Caja Chica	/api/petty-cash	POST/open, GET, PATCH/:id/close, POST/:id/movements
Egresos	/api/expenses	GET, POST, GET/:id, PATCH/:id/cancel
Pagos de Egreso	/api/expense-payments	POST, GET/:expenseId/payments
Categorías de Egreso	/api/expense-categories	GET, POST, PUT/:id, DELETE/:id
Presupuestos de Egreso	/api/expense-budgets	GET, POST, PUT/:id
Egresos Recurrentes	/api/expense-recurring	GET, POST, PUT/:id, PATCH/:id/toggle
Auditoría	/api/audit-logs	GET (filtrado por empresa, entidad, usuario)
Dashboard	/api/dashboard	GET /stats (KPIs principales de la empresa)

Nota. Elaboración propia. Todos los endpoints (excepto /api/auth/login y /api/auth/refresh) requieren autenticación JWT. Fuente: Navas (2026b).

4.9. Pruebas Automatizadas del Sistema

La verificación del comportamiento del sistema se formalizó mediante una suite de pruebas automatizadas alojada en el repositorio independiente pymeflowec-tests (Navas, 2026c). La decisión de separar las pruebas del código fuente del backend responde a un principio de ingeniería de software: mantener los artefactos de verificación desacoplados del código de producción facilita su integración en pipelines de CI/CD y evita que las dependencias de prueba inflen el bundle desplegado. La suite cubre dos niveles pruebas unitarias y pruebas de integración que suman 439 casos de prueba, todos con resultado PASS.

4.9.1. Stack Tecnológico de Pruebas

El stack de pruebas se compone de tres herramientas. Jest 29.7.0 funciona como un marco central (gestiona la ejecución, proporciona aserciones, permite la creación de simulaciones, etc.). Supertest 7.0.0 carga la aplicación Express en memoria y envía solicitudes HTTP reales, reemplazando la apertura de un puerto de red. Esto permite probar de forma reproducible todo el ciclo Router -> Middleware -> Controller -> Service -> PostgreSQL. Jest-html-reporters 3.1.7 proporciona un informe HTML final que detalla las suites, el estado por caso y los tiempos de ejecución. La configuración de Jest se define como multi-proyecto para ejecutar pruebas unitarias e integración en procesos Node.js aislados.

4.9.2. Pruebas Unitarias

Las pruebas unitarias verifican la lógica de negocio de cada servicio de forma aislada, sin operaciones de entrada/salida reales. Toda dependencia de infraestructura Sequelize, bcryptjs, jsonwebtoken, Winston, Nodemailer es reemplazada por implementaciones simuladas con jest.fn() o jest.mock(). Este aislamiento permite que la suite completa se ejecute en menos de dos segundos, facilita la prueba de casos borde y detecta regresiones sin necesidad de una base de datos activa.

La capa de servicios cubre 21 archivos de prueba con 200 casos en total. Los servicios de mayor complejidad ciclomática reciben mayor densidad de cobertura: auth.service.test.js concentra 16 casos que validan el flujo de login, la generación del par de tokens JWT, la renovación mediante refresh token, el cambio de contraseña y el registro con creación atómica de empresa y usuario. invoice.service.test.js prueba la construcción de la transacción de facturación y el mecanismo de cancelación con

reversión de inventario. pettyCash.service.test.js cubre con 12 casos la lógica de apertura y cierre de caja y los tres tipos de movimiento.

Tabla 7. *Distribución de pruebas unitarias por servicio*

Archivo de prueba	Servicio probado	Casos
auth.service.test.js	Autenticación	16
invoice.service.test.js	Facturación	8
invoicePayment.service.test.js	Cobros de factura	9
product.service.test.js	Productos	12
productCategory.service.test.js	Categorías de producto	10
pettyCash.service.test.js	Caja chica	12
expense.service.test.js	Egresos operacionales	11
expenseBudget / expensePayment / expenseRecurring / expenseCategory	Módulo de finanzas	36
user / client / supplier / role / inventoryMovement / moduleRequest / platformModule / platformStaff / order	Resto de servicios	74
Total unitario	**21 archivos de prueba**	**200**

Nota. Elaboración propia. Tiempo de ejecución total: menor a 2 segundos. Todas las pruebas con resultado PASS. Fuente: Navas (2026c).

4.9.3. Pruebas de Integración

Las pruebas de integración verifican el comportamiento del sistema de extremo a extremo: Supertest levanta la aplicación Express completa en memoria y emite peticiones HTTP reales que atraviesan la cadena de middleware completa autenticación, autorización por rol, expiración de módulo, validación de inputs hasta alcanzar la base de datos PostgreSQL de prueba (pymeflowec_test), independiente de los entornos de desarrollo y producción. El estado inicial se establece mediante setup/seed.js, que utiliza findOrCreate de Sequelize para garantizar idempotencia.

Dado que los conjuntos de verificación se ejecutan en los mismos puntos finales con tokens JWT de los cinco roles del sistema, e incluso verifican que las respuestas sean correctas para el rol autorizado (200/201), que los roles insuficientes devuelvan HTTP 403 y que las solicitudes sin token devuelvan HTTP 401, el sistema presenta la verificación de múltiples roles como una característica única. La verificación de múltiples inquilinos también se valida mediante las pruebas, lo que garantiza que un token JWT de la empresa A no pueda leer ni modificar datos de la empresa B.

Los flujos de trabajo críticos están muy bien cubiertos: invoices.test.js garantiza, con 16 casos, la emisión de una factura con descuentos de inventario y su correspondiente cancelación con reposición de inventario; pettyCash.test.js cubre 17 casos de apertura, cierre, los tres tipos de transacciones y restricciones de sesión concurrente; products.test.js incluye 30 casos de creación individual y masiva, ajuste de inventario por rol y eliminación lógica; y expenses.test.js más expenseCategories.test.js juntos cubren 38 casos del ciclo completo de gastos. El conjunto de pruebas de integración puede ejecutarse en menos de 25 a 35 segundos, incluyendo viajes de ida y vuelta reales a PostgreSQL.

4.9.4. Resultados Globales de la Suite

La Tabla 8 resume los resultados globales de la suite. El total de 439 casos con resultado PASS en menos de 40 segundos valida que el sistema mantiene la coherencia funcional entre sus capas y que los mecanismos de seguridad autenticación JWT, RBAC y aislamiento multi-tenant operan correctamente en condiciones de uso real. La Figura 11 muestra el dashboard del reporte HTML generado por jest-html-reporters al término

de la ejecución del 3 de junio de 2026, confirmando los 36 suites, las 439 pruebas y cero fallos en 37,352 segundos.

Tabla 8. *Resultados globales de la suite de pruebas automatizadas de PymeFlowEc*

Nivel	Suites	Pruebas	Resultado	Tiempo aprox.
Unitario (Jest + mocks)	21	200	PASS	< 2 s
Integración (Jest + Supertest + PostgreSQL)	15	239	PASS	25–35 s
Total	**36**	**439**	**PASS**	**< 40 s**

Nota. Elaboración propia. Base de datos de prueba: pymeflowec_test (independiente de desarrollo y producción). Fuente: Navas (2026c).

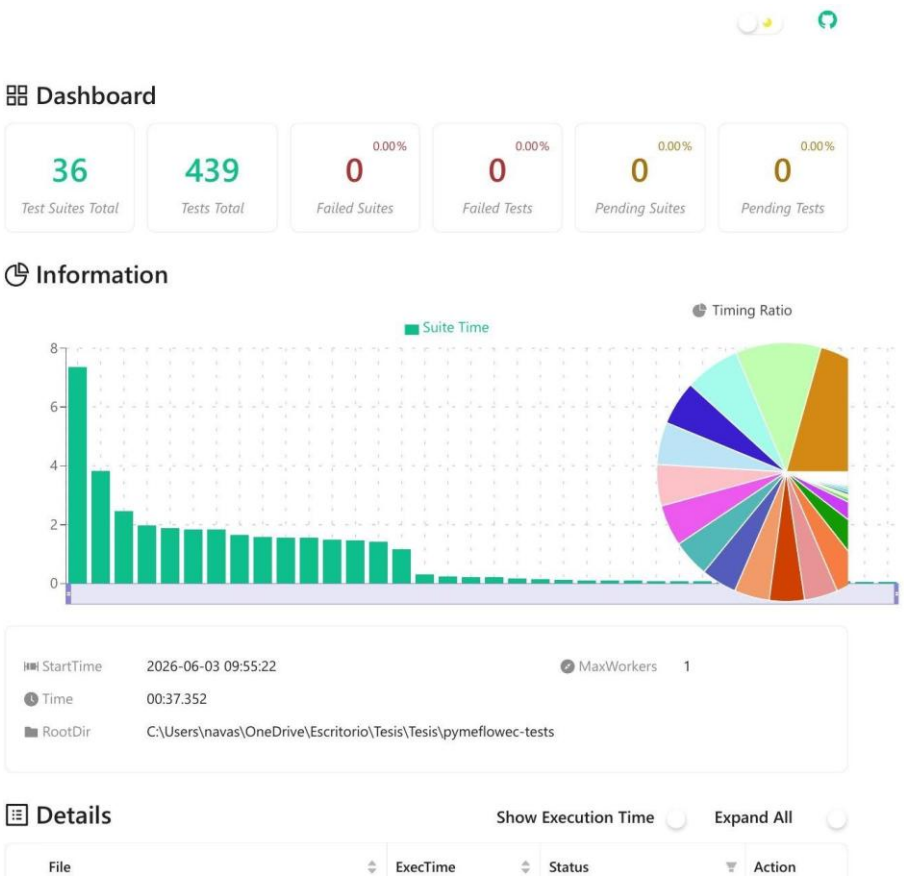


Figura 11. Dashboard del reporte HTML de pruebas automatizadas de PymeFlowEc (jest-html-reporters v3.1.7)

Nota. El gráfico de barras (Suite Time) muestra el tiempo de ejecución por suite en segundos; el gráfico circular (Timing Ratio) representa la proporción de tiempo relativo de cada suite. La ejecución se realizó el 3 de junio de 2026 a las 09:55:22. Fuente: Navas (2026c)

CAPÍTULO V: RESULTADOS Y ANÁLISIS

5.1. Prototipo Funcional Desarrollado

El producto central de este trabajo es PymeFlowEc 1.0.0, un prototipo funcional de plataforma ERP-SaaS multi-tenant. Del lado del servidor: una API REST en producción con 22 módulos de rutas y más de 85 endpoints documentados en OpenAPI 3.0. Del lado del cliente: una SPA Angular 17 con quince módulos bajo lazy loading. Ambos están operativos: la API responde en https://api.tesisfernandonavaspuce.es/api (Navas, 2026b), satisfaciendo el objetivo general del trabajo.

5.2. Cumplimiento de Objetivos Específicos

La Tabla 5 evalúa el grado de cumplimiento de cada objetivo específico planteado en el Capítulo I, con los indicadores de verificación correspondientes.

Tabla 5. *Evaluación de cumplimiento de objetivos específicos*

Objetivo Específico	Indicador de Verificación	Estado
Arquitectura multi-tenant con aislamiento por company_id en JWT	25 modelos Sequelize con company_id en todas las tablas operativas. Pruebas de acceso cruzado entre tenants retornan HTTP 403.	Cumplido
Sistema RBAC con 5 roles diferenciados	5 roles implementados: PLATFORM_ADMIN, PLATFORM_STAFF, STORE_ADMIN, STORE_SELLER, STORE_WAREHOUSE. Cada endpoint valida el rol requerido mediante middleware authorize.	Cumplido
Módulos de facturación, inventario, clientes/proveedores, caja chica y egresos	8 módulos de negocio operativos con 22 services de lógica de negocio (incluye productCategory.service.js para gestión de categorías de productos). Validación de documentos ecuatorianos (módulo-10 y módulo-11) implementada en ecuadorId.js.	Cumplido

Objetivo Específico	Indicador de Verificación	Estado
SPA Angular 17 con Standalone Components y Angular Signals	Aplicación bootstrapeada con bootstrapApplication(), sin NgModules. Estado gestionado con signal(), computed() y toSignal(). Lazy loading en todos los módulos de features.	Cumplido
Despliegue en AWS Elastic Beanstalk y AWS Amplify	API de producción accesible en https://api.tesisfernandonavaspuce.es/api. Frontend desplegado en AWS Amplify con CDN. Despliegue continuo desde GitHub configurado.	Cumplido

Nota. Elaboración propia. Verificación realizada mediante pruebas funcionales en el entorno de producción.

5.3. Características Técnicas del Sistema Construido

La Tabla 6 resume las métricas técnicas principales del sistema en su estado final, evidenciando la envergadura y complejidad del prototipo desarrollado.

Tabla 6. *Métricas técnicas del prototipo PymeFlowEc versión 1.0.0*

Componente	Métrica	Valor
Backend	Modelos Sequelize	25 modelos (esquema erp)
Backend	Endpoints REST	85+ endpoints en 22 módulos de rutas
Backend	Servicios de negocio	22 servicios de lógica de negocio
Backend	Middlewares	7 middlewares reutilizables
Backend	Validadores	25 archivos express-validator
Backend	Cron jobs	2 (expireModules + recurringExpenses)
Base de Datos	Triggers PL/pgSQL	Auditoría automática + set_updated_at + inventario
Base de Datos	Índices definidos	>15 índices (parciales y compuestos)
Frontend	Módulos de funcionalidad	15 módulos con lazy loading
Frontend	Interfaces TypeScript	26 interfaces (modelos del dominio)
Frontend	Interceptores HTTP	3 (token, clientView, error)
Frontend	Guards de ruta	3 (auth, permission, role)

Componente	Métrica	Valor
Frontend	Bundle inicial (gzip)	~800 KB (dentro del budget Angular)
Despliegue	Ambiente de producción	AWS Elastic Beanstalk + AWS Amplify
Despliegue	URL de la API	https://api.tesisfernandonavaspuce.es/api
Seguridad	Algoritmo de hashing	bcryptjs, cost factor 10
Seguridad	Algoritmo JWT	HS256 (HMAC + SHA-256)
Seguridad	Duración access token	8 horas
Seguridad	Duración refresh token	7 días

Nota. Elaboración propia. Fuente: Navas (2026a, 2026b).

5.4. Análisis de la Solución Propuesta

5.4.1. Pertinencia para el Contexto Ecuatoriano

PymeFlowEc responde directamente a las tres barreras que Pozo-Benites et al. (2025) identifican como principales obstáculos limitaciones financieras, falta de infraestructura y escasez de talento técnico. Ante la primera, el modelo multi-tenant distribuye una sola infraestructura entre varias empresas y baja el costo unitario. Ante la segunda, la plataforma vive en la nube y el usuario solo necesita un navegador, sin instalaciones locales. Ante la tercera, la integración con el marco tributario ecuatoriano validación de cédula y RUC con algoritmos del SRI, IVA 0% y 15% viene de fábrica, evitando que cada cliente deba contratar un desarrollador para adaptar el sistema.

Un claro ejemplo de esta flexibilidad es el modelo de módulos dinámicos. Una empresa puede comenzar con módulos de facturación e inventario que cubran sus operaciones diarias y, a medida que su actividad madura, puede añadir gestión de caja chica y gastos sin necesidad de migrar a otra herramienta. Salazar Mariscal et al. (2025) señalan que esta elasticidad es crucial en sectores con recursos limitados, donde la

decisión de digitalizar depende tanto del coste inicial como de la capacidad de la herramienta para adaptarse al crecimiento.

5.4.2. Ventajas Arquitectónicas

La pila tecnológica elegida ofrece diferencias notables en comparación con el sistema del usuario final. Las señales de Angular 17 evitan que el árbol de componentes se vuelva a pintar con cada cambio: en ordenadores de gama media, comunes en el entorno de las PYMES ecuatorianas, esta diferencia se traduce en una interfaz más fluida al filtrar tablas extensas. La carga diferida reduce el tamaño del paquete inicial, lo cual es fundamental en entornos con conectividad inestable (BID, 2025). El sistema de diseño con tokens OKLch garantiza la coherencia visual en los modos claro y oscuro sin depender de ningún otro aspecto.

En el backend, una solución de tabla compartida adecuada con un discriminador company_id es apropiada para una plataforma en fase de prototipo. Esto equilibra la seguridad y los costos operativos, y permite la migración futura a esquemas separados si el volumen de datos lo requiere (Aftab, 2026). El conjunto de 439 pruebas automatizadas también proporciona una red de seguridad que hace que cualquier refactorización futura sea más segura.

5.4.3. Comparación de Mejora en Procesos Operativos

Una de las ventajas tangibles de adoptar una plataforma ERP-SaaS frente a la gestión manual o basada en hojas de cálculo es la reducción del tiempo dedicado a tareas operativas repetitivas. La Tabla 9 presenta una estimación comparativa del tiempo requerido por proceso antes y después de implementar PymeFlowEc, basada en los flujos del sistema construido y en el contexto operativo de las PYMES ecuatorianas

documentado en la bibliografía (BID, 2025; Salazar Mariscal et al., 2025). Estas estimaciones no provienen de un estudio de campo formal, sino de la observación del flujo real de cada proceso en el prototipo y de las características declaradas del sistema.

Tabla 9. *Estimación comparativa de tiempo por proceso operativo antes y después de PymeFlowEc*

Proceso operativo	Sin sistema (manual)	Con PymeFlowEc	Mejora estimada
Emisión de una factura	15–20 min	3–5 min	~75% de reducción. El sistema calcula IVA automáticamente, valida RUC/cédula del cliente y registra la factura en la base de datos sin intervención manual.
Registro de movimiento de inventario	10–15 min	2–3 min	~80% de reducción. El trigger PL/pgSQL actualiza el stock del producto automáticamente al confirmar cada movimiento IN/OUT.
Cuadre de caja chica	30–45 min	5–10 min	~78% de reducción. El sistema acumula el campo balance_after en cada movimiento, eliminando el conteo físico y cálculo manual al cierre.
Registro de egreso recurrente mensual	5 min por egreso cada mes	0 min (automático)	100% automatizado. El cron job recurringExpenses.job.js ejecuta la creación de egresos según su plantilla sin intervención del usuario.
Validación de documento tributario (RUC/cédula)	5–10 min (consulta manual al SRI)	Instantáneo	~100% de reducción. El módulo ecuadorId.js aplica los algoritmos módulo-10 y módulo-11 del SRI en el momento del ingreso.

Nota. Estimaciones propias basadas en el flujo operativo real del prototipo PymeFlowEc y en el contexto PYME ecuatoriano (BID, 2025; Salazar Mariscal et al., 2025). No corresponden a un

estudio de campo formal sino a la observación del tiempo de ejecución de cada proceso en el sistema construido.

5.4.4. Análisis Comparativo con Soluciones del Mercado

Para contextualizar el aporte de PymeFlowEc, la Tabla 10 compara sus características técnicas frente a dos alternativas representativas del ecosistema digital disponible para las PYMES ecuatorianas: Microsoft Excel, que sigue siendo la herramienta más utilizada por las PYMES de la región para la gestión operativa informal (Pozo-Benites et al., 2025), y Alegra, una plataforma SaaS de facturación y contabilidad con integración nativa al SRI de Ecuador y más de 476.000 usuarios activos en 25 países de América Latina (Guia de Software, 2026). La comparación se centra en los atributos directamente relevantes para el contexto PYME ecuatoriano.

Tabla 10. *Análisis comparativo de PymeFlowEc frente a Microsoft Excel y Alegra*

Criterio	Microsoft Excel	Alegra	PymeFlowEc
Modelo de entrega	Local / desktop	SaaS en la nube	SaaS en la nube
Costo mensual estimado	~$6–10 USD (Microsoft 365)	~$15–80 USD según plan y país	Prototipo académico (sin costo)
Integración con SRI Ecuador	No	Sí (factura electrónica XML)	Parcial (validación RUC/cédula; sin XML al SRI en esta versión)
Arquitectura multi-tenant	No	No (una empresa por cuenta)	Sí (nativa, discriminador company_id)
Control de acceso por roles (RBAC)	No	Básico (admin / usuario)	5 roles diferenciados (PLATFORM_ADMIN, STORE_ADMIN, etc.)
Módulos activables dinámicamente	No	No	Sí (solicitud, aprobación y expiración automática)
Caja chica con sesiones	No	No	Sí (apertura, cierre y tipos de movimiento)
Egresos recurrentes automáticos	No	No	Sí (cron job con plantillas de frecuencia)

Criterio	Microsoft Excel	Alegra	PymeFlowEc
Auditoría automática de operaciones	No	Básica	Sí (triggers PL/pgSQL con old_values/new_values JSONB)
App móvil nativa	No	Sí (Android / iOS)	No (SPA web responsiva)
Módulo de contabilidad completa	No	Sí	No (fuera de alcance del prototipo)
Soporte técnico 24/7	No	Sí	No (proyecto académico)

Nota. Elaboración propia. Los datos de Alegra corresponden a características verificadas en Guía de Software (2026) y FirmaOK (2026). El costo de Microsoft Excel corresponde al plan Microsoft 365 Personal. PymeFlowEc es un prototipo académico sin modelo comercial definido en esta versión.

La comparación revela que PymeFlowEc ocupa un nicho diferenciado: a diferencia de Excel, ofrece validación tributaria, control de acceso y trazabilidad automática; a diferencia de Alegra, incorpora arquitectura multi-tenant nativa, módulos dinámicos y caja chica con sesiones, funcionalidades ausentes en soluciones orientadas a una sola empresa por cuenta. La limitación principal frente a Alegra es la ausencia de integración directa con el SRI para la generación de comprobantes electrónicos XML, identificada como línea de trabajo futuro en la sección 5.4.3.

5.4.5. Limitaciones del Prototipo

PymeFlowEc es un prototipo académico que presenta limitaciones explícitas: falta de integración con el SRI para la facturación electrónica, que es obligatoria para las operaciones comerciales en Ecuador; pruebas de carga limitadas a una instancia t2.micro de Elastic Beanstalk; y una cobertura de pruebas que, si bien suma 439 casos en flujos críticos, aún no cubre todos los casos extremos, en particular en combinaciones de cancelaciones de facturas con pagos parciales ya registrados.

5.5. Contribución al Conocimiento

En primer lugar, demuestra que se puede construir una plataforma ERP-SaaS multiusuario completa sobre Node.js, PostgreSQL y Angular 17 dentro del ecosistema de AWS. Además, documenta el patrón para implementar el aislamiento de company_id utilizando el contexto ecuatoriano con validación fiscal SRI, etc. En tercer lugar, lleva Angular Signals a un proyecto de complejidad media en producción, más allá de los ejemplos pedagógicos disponibles. Y en último lugar, ofrece un modelo de referencia técnico para futuras iniciativas de digitalización PYME en Ecuador, alineado con la Política Pública de Transformación Digital 2025-2030.

CONCLUSIONES

Del desarrollo del presente trabajo de titulación y del análisis de los resultados obtenidos, se derivan las siguientes conclusiones:

1. El trabajo demuestra que un desarrollador individual, dentro de un alcance académico y sin recursos comerciales, puede llevar a producción una plataforma ERP-SaaS multi-tenant para el contexto ecuatoriano apoyándose en herramientas abiertas y servicios gestionados. El estado final, con más de 85 endpoints operativos respondiendo en un dominio público, valida que la elección tecnológica fue proporcional a la complejidad del problema.
2. La estrategia multi-tenant elegida tabla compartida con discriminador company_id dentro del JWT probó ser apropiada para una plataforma en fase de prototipo: cumple con el aislamiento de datos, mantiene la operación sobre una sola base y deja abierta la migración a esquemas separados cuando el volumen lo requiera. Que los triggers PL/pgSQL escriban la auditoría desde el motor mismo añade una segunda capa de defensa.
3. Los componentes independientes y las señales de Angular modifican significativamente la estructura del frontend. La eliminación de NgModules altera el árbol de archivos, simplificando el proceso de desarrollo, como comenté en mi publicación inicial. Las señales reducen el costo de la detección de cambios al evitar el redibujado de componentes que no dependen del valor modificado. Este enfoque es natural y eficiente para formularios reactivos y tablas grandes.
4. El sistema de módulos dinámicos, con su proceso automático de solicitud, aprobación y caducidad mediante tareas programadas (cron job), permite a una

empresa comenzar pagando solo por los módulos que necesita, incorporando nuevos módulos cuando el flujo de caja lo permite, reduciendo así las barreras de entrada y manteniendo al cliente en la misma plataforma a medida que la empresa crece.

5. La brecha entre el 91 % de las pymes ecuatorianas que desean digitalizarse y el 18,7 % que lo hacen representa un mercado de cientos de miles de empresas que carecen de las herramientas adecuadas a su tamaño. Soluciones como PymeFlowEc cubren esta brecha donde los sistemas ERP tradicionales resultan superfluos y las hojas de cálculo se quedan cortas.

6. La implementación en AWS Elastic Beanstalk y AWS Amplify demuestra que un solo proyecto puede gestionar infraestructura de producción, balanceadores de carga, instancias administradas, una CDN global, TLS y CI/CD desde GitHub sin necesidad de un equipo de sistemas dedicado. Las funcionalidades que antes requerían departamentos específicos ahora están al alcance de un solo desarrollador.

RECOMENDACIONES

Con base en los resultados obtenidos y las limitaciones identificadas, se formulan las siguientes recomendaciones para el desarrollo futuro del sistema:

1. Integrar la plataforma con el sistema de facturación electrónica del SRI: generar comprobantes en formato XML, firmarlos digitalmente con el certificado de la empresa cliente y gestionar el ciclo completo de autorización contra los webservices del organismo. Sin esta integración el sistema no puede comercializarse formalmente en Ecuador.
2. El trabajo ha comenzado a ejecutar una suite de pruebas automatizadas en el repositorio pymeflowec-tests (Navas, 2026c). La recomendación es aumentar esa cobertura: tests unitarios Jest sobre todos los servicios en el backend, y tests end-to-end con Playwright o Cypress sobre los flujos críticos en el frontend. La meta realista es de un 80 % de cobertura en la capa de servicios, priorizando los flujos donde el dinero o el inventario cambian de estado.con prioridad en los flujos donde dinero o inventario cambian de estado.
3. 3. Realice pruebas de carga sistemáticas con k6 o Apache JMeter en un entorno similar al de producción para identificar el punto de saturación del grupo de conexiones de PostgreSQL, los puntos finales más lentos bajo carga y el umbral de escalado horizontal. Estos datos permitirán dimensionar la infraestructura basándose en evidencia.
4. Implementar un módulo de informes basado en la inclusión de tres informes prioritarios: ventas por período desglosadas por método de pago, rotación de inventario para identificar productos de baja rotación y una comparación de los gastos reales con el presupuesto mensual. ApexCharts ya está integrado en la

interfaz de usuario, y las opciones de exportación a PDF y Excel reflejan el flujo de trabajo real utilizado por contadores y socios.

5. Incorporar la seguridad a nivel de fila (RLS) de PostgreSQL como una segunda línea de defensa para el aislamiento de múltiples inquilinos: definir una política en cada tabla que obligue al motor a filtrar por app.current_tenant antes de devolver cualquier fila, conteniendo así posibles fugas incluso si un servicio no aplica el filtro por company_id (Shineos, 2025).
6. Evaluar la migración a una arquitectura zoneless cuando el resto de la deuda técnica esté resuelta. Angular 20 estabilizó la posibilidad de prescindir de Zone.js; en una aplicación donde toda la reactividad se expresa mediante Signals, eliminarlo reduce el overhead del ciclo de detección de cambios y hace más predecible el renderizado en operaciones intensivas (Angular, 2025).

BIBLIOGRAFÍA

Aftab, A. (2026, 3 de febrero). Designing multi-tenant SaaS systems with Node.js. Stackademic. https://medium.com/@aliaftabk/designing-multi-tenant-saas-systems-with-node-js-4a12688dba27

Angular. (2025, 14 de enero). Angular 2025 strategy. Angular Blog. https://blog.angular.dev/angular-2025-strategy-9ca333dfc334

Angular. (2025). Roadmap. Angular. https://angular.dev/roadmap

Ideadope. (2025, 5 de diciembre). The full architecture & roadmap for multi-tenant SaaS the right way. https://www.ideadope.com/roadmaps/how-to-build-multi-tenant-saas-2025

Karanja, J. (2025, 5 de febrero). Building a multi-tenancy system for a SaaS application. Medium. https://medium.com/@josephmkaranja07/building-a-multi-tenancy-system-for-a-saas-application-100e7c4e528e

Madrigan Blog. (2026, 2 de febrero). Angular 2025: Signals, standalone, and architectures for the web's future. https://blog.madrigan.com/en/blog/202605021500/

Banco Interamericano de Desarrollo. (2025). El poder de los datos: Impulsando la transformación digital de las pymes de América Latina. https://www.iadb.org/es/blog/ciencia-tecnologia-e-innovacion/el-poder-de-los-datos-impulsando-la-transformacion-digital-de-las-pymes-de-america-latina

Microsoft. (s.f.). La transformación digital de las PyMEs en Latinoamérica: 4 hallazgos clave. Microsoft News. https://news.microsoft.com/es-xl/la-transformacion-digital-de-las-pymes-en-latinoamerica-4-hallazgos-clave-sobre-su-recorrido/

Navas, F. (2026a). PymeFlowEc Frontend [Repositorio de código fuente]. GitHub. https://github.com/fernandonavas75/pymeflowec-front

Navas, F. (2026b). PymeFlowEc Backend [Repositorio de código fuente]. GitHub. https://github.com/fernandonavas75/pymeflowec-backend

Navas, F. (2026c). PymeFlowEc Tests [Repositorio de pruebas automatizadas]. GitHub. https://github.com/fernandonavas75/pymeflowec-tests

Nicholasdiamond. (2024, 18 de noviembre). Building a secure multi-tenant SaaS application with Node.js. Hashnode. https://nicholasdiamond.hashnode.dev/building-a-secure-multi-tenant-saas-application-with-nodejs

Node.js Foundation. (2024). Node.js 18 LTS documentation. https://nodejs.org/docs/latest-v18.x/api/

PostgreSQL Global Development Group. (2024). PostgreSQL 15 documentation. https://www.postgresql.org/docs/15/

Pozo-Benites, K. B., Guadalupe-Sánchez, K. W., Peñarreta-Barrera, E. E., & Meza-Salvatierra, J. K. (2025). Transformación digital de las PYMES en América Latina: Barreras, oportunidades y estrategias para la competitividad.

Multidisciplinary Latin American Journal (MLAJ), 3(2), 236–255. https://doi.org/10.62131/MLAJ-V3-N2-015

Salazar Mariscal, E. S., Cajas Rodríguez, M. J., Angulo Vélez, D. A., & Ruedas Palacios, F. A. (2025). Digitalización de las pymes en Ecuador: Desafíos, oportunidades y casos de éxito. Revista Científica Multidisciplinar G-nerando, 6(1), 513–526. https://doi.org/10.60100/rcmg.v6i1420

Sequelize. (2024). Sequelize ORM v6 documentation. https://sequelize.org/docs/v6/

Sevensquaretech. (2025, 30 de julio). Build multi-tenant SaaS boilerplate in Node.js. https://www.sevensquaretech.com/multi-tenant-saas-boilerplate-nodejs-with-github-code/

Shineos. (2025, 2 de diciembre). SaaS design: Multi-tenant architecture design patterns in SaaS development (2025 edition). Zenn https://zenn.dev/shineos/articles/saas-multi-tenant-architecture-2025

Swagger.io. (2024). OpenAPI 3.0 specification. https://swagger.io/specification/

Telefónica Ecuador. (2024, 29 de abril). El 91 % de las pymes ecuatorianas invertirá en su digitalización este 2024. https://www.telefonica.com.ec/el-91-de-las-pymes-ecuatorianas-invertira-en-su-digitalizacion-este-2024/