

Desarrollo de una Plataforma BackOffice para la Administración Centralizada
de Soluciones Tecnológicas Empresariales en SIAN EXUS

Pasantía en modalidad de convenio

Andres Felipe Cabal Correa



Universidad del Valle
Facultad de ingeniería
Escuela de ingeniería de sistemas y computación
Sede Tuluá
2025

Desarrollo de una Plataforma BackOffice para la Administración Centralizada
de Soluciones Tecnológicas Empresariales en SIAN EXUS

Pasantía en modalidad de convenio

Andres Felipe Cabal Correa

Código: 202160339

Andres.felipe.cabal@correounivalle.edu.co

Director

Mauricio López Benítez. Ing.

Profesor de la Escuela de Ingeniería de Sistemas y Computación

mauricio.lopez@correounivalle.edu.co

Universidad del Valle

Facultad de ingeniería

Escuela de ingeniería de sistemas y computación

Sede Tuluá

2025

Índice de Contenido

Resumen	7
1. Introducción.....	8
1.1 Descripción de la empresa.....	9
1.1.1 Descripción General	9
1.1.2 Misión.....	9
1.1.3 Visión	9
1.1.4 Valores Organizacionales	9
1.1.5 Razón Social	9
1.1.6 Tipo de empresa.....	9
1.1.7 Número de empleados	9
1.2 Formulación del problema.....	10
1.2.1 Descripción del problema.....	10
1.2.2 Definición del problema	11
1.3 Alcance	12
1.3.1 Declaración del alcance	12
1.3.2 Objetivo General.....	12
1.3.3 Objetivos específicos.....	12
1.3.4 Relación objetivos específicos - resultados esperados	12
1.3.5 Marco lógico de la propuesta.....	13
2. Marco referencial.....	15
2.2. Marco Conceptual.....	15
2.3. Marco teórico.....	16
2.3.1. Teoría General de sistemas.....	16
2.3.2. Teoría de los sistemas de información	16
2.3.3. Teoría de la comunicación.....	17
2.4. Estado del Arte	18
2.4.1. Evolución de plataformas BackOffice bajo el patrón Modelo-Vista- Controlador.....	18
2.4.2. Plataformas BackOffice modulares y extensibles	18
2.5. Antecedentes.....	19

3.	Metodología.....	21
3.2.	Metodología de Investigación	21
3.3.	Metodología de desarrollo de software	22
	Flujo operativo de la empresa SIAN EXUS.....	22
	Modelo Incremental.....	23
	Modelo Evolutivo	24
	Modelo Evolutivo-Incremental.....	25
	Tabla de Evaluación por Puntaje.....	26
3.4.	Metodología de gestión de actividades.....	27
4.	Ingeniería del Proyecto	29
4.1	Análisis y diseño del sistema.....	29
4.1.1	Arquitectura general del sistema	30
4.1.2	Modelo de datos	31
4.1.3	Diagrama de componentes.....	32
4.2	Conexión a API externa.....	33
4.3	Ecosistema Funcional de la Plataforma.....	34
4.3.1	Dashboard.....	34
4.3.2	Módulo de Gestión de Empresas: Vista de Listado.....	39
4.3.3	Módulos de Gestión de Catálogo de Productos: Servicios, Planes y Paquetes ...	40
4.3.4	Suscripciones	41
4.3.5	Historial de Asignaciones de Documentos.....	43
4.4	Procesos Automáticos y Arquitectura de Eventos del BackOffice	43
4.4.1	Introducción a la Arquitectura Event-Driven	43
4.4.2	Flujo de Publicación y Consumo de Eventos	44
4.4.3	Eventos Clave del Ecosistema.....	45
4.4.4	Handlers Especializados: Procesamiento Automático	47
4.4.5	Configuración de RabbitMQ y Conectividad.....	48
4.4.6	Integración Crítica con NextPyme	49
4.4.7	Ventajas Arquitectónicas de la Implementación	49
4.5	Pruebas y Validación de la Plataforma.....	50
4.5.1	Participación del equipo QA.....	50

5. Conclusiones.....	53
5.1 Conclusiones.....	53
5.2 Recomendaciones para Futuros Trabajos	56
Referencias	57

Índice de Tablas

Tabla 1 Relación Objetivos específicos - Resultado esperados	12
Tabla 2 Marco lógico	13
Tabla 3 Modelo Input-Proceso-Output	16
Tabla 4 Comparativa con antecedentes.....	19
Tabla 5 Comparativa de modelos	25
Tabla 6 Factores de evaluación para la elección del modelo	26
Tabla 7 Evaluación de modelos por puntaje	27

Índice de Imágenes

Imagen 1 Árbol de problemas.....	10
Imagen 2 Niveles de madurez tecnológica TRL. Tomado de [20].....	21
Imagen 3 Grafica de modelo incremental. Tomado de [27].....	23
Imagen 4 Grafica de modelo evolutivo. Tomado de [27].....	24
Imagen 5 Representación gráfica del MVC obtenida de [28]	30
Imagen 6 captura de pantalla que muestra la parte superior del dashboard	34
Imagen 7 captura de pantalla que muestra el grafico de los planes más contratados.....	35
Imagen 8 captura de pantalla que muestra el grafico de los ingresos mensuales	36
Imagen 9 captura de pantalla que muestra el grafico de las empresas por tipo	37
Imagen 10 Captura del Gráfico de consumo de Folios (Consumidos vs Recargados).....	38
Imagen 11 Representación del flujo de publicación y consumo de eventos	44

Resumen

SIAN EXUS es una empresa que se dedica a ofrecer soluciones tecnológicas para la gestión contable, facturación electrónica, inventario y ventas, dirigida a microempresas, pymes y grandes empresas. A día de hoy, enfrenta desafíos en la administración de sus servicios debido a la falta de integración de un entorno de tipo BackOffice centralizado, lo que complica tareas como el control de accesos para cada cliente, la asignación de planes personalizados y el monitoreo del uso de soluciones como facturación electrónica, nómina electrónica y E-Commerce.

Con el fin de dar respuesta a esta necesidad, se llevó a cabo el desarrollo de una plataforma tipo BackOffice que actúa como centro de administración para dichos servicios. Teniendo en cuenta las metodologías y tecnologías ya implementadas por el equipo de desarrollo en la empresa.

Keywords

Back-office; Software architecture; Microservices; Model-View-Controller (MVC); Software as a Service (SaaS); RabbitMQ; Access control; Information systems; Workflow automation.

1. Introducción

Hoy en día, muchas empresas pierden competitividad porque no cuentan con herramientas tecnológicas que automaticen sus procesos. Esto no solo les ayudaría a ahorrar costos, sino también tomar decisiones más acertadas [1]. En este escenario, los sistemas BackOffice juegan un papel clave, ya que simplifican la gestión de servicios, usuarios y recursos técnicos. Los sistemas BackOffice se han vuelto herramientas indispensables para mantener operaciones eficientes y administrar servicios de forma sostenible [2].

No basta con implementar plataformas tecnológicas ya sea para facturación electrónica, E-Commerce o nómina digital, si no se respalda esa tecnología con una arquitectura sólida, segura y adaptable a cada cliente. Se puede concluir que, sin una base administrativa bien estructurada, resulta casi imposible controlar accesos, personalizar planes o gestionar adecuadamente los permisos de los usuarios [3].

En este contexto se encuentra **SIAN EXUS**, una empresa que se dedica a brindar soluciones automatizadas para la gestión contable, ventas, inventarios y facturación, dirigidas a microempresas, pymes y grandes empresas. A pesar de su enfoque innovador, actualmente enfrenta grandes desafíos debido a un incompleto entorno centralizado para la administración de sus servicios ofrecidos y activos hacia las empresas. Esta carencia genera ineficiencias en procesos críticos como la asignación de planes, la gestión de accesos y el monitoreo del uso de sus soluciones tecnológicas.

El presente proyecto lleva a cabo el desarrollo de una plataforma tipo **BackOffice** dentro del ecosistema de interfaces sianexus, basado en arquitecturas modernas y buenas prácticas de ingeniería de software, que permita unificar y optimizar la administración interna de los servicios ofrecidos por la empresa SIAN EXUS. En particular, se contempla el uso de arquitecturas distribuidas y comunicación asíncrona mediante herramientas como RabbitMQ, que permiten desacoplar servicios, mejorar la escalabilidad y garantizar la resiliencia del sistema [4], [5].

1.1 Descripción de la empresa

1.1.1 Descripción General

En Sian Exus ofrecemos soluciones electrónicas para optimizar procesos contables, facturación y reportes tributarios. Cumplimos con los requisitos de la DIAN y ayudamos a las empresas colombianas a trabajar de forma más ágil, segura y eficiente desde la nube.

1.1.2 Misión

Impulsar la transformación digital de las empresas colombianas mediante soluciones tecnológicas confiables y accesibles. Trabajamos para simplificar la gestión contable, administrativa y tributaria, acompañando a nuestros clientes en su crecimiento con herramientas que se adaptan a sus necesidades reales.

1.1.3 Visión

Ser una empresa líder en el desarrollo de soluciones electrónicas que fortalezcan la operación de microempresas, pymes y grandes organizaciones en Colombia. Aspiramos a construir un entorno empresarial más eficiente, transparente y preparado para el futuro digital.

1.1.4 Valores Organizacionales

- Trabajo en Equipo
- Calidad
- Compromiso
- Innovación

1.1.5 Razón Social

SOCIEDAD SIAN EXUS SAS

1.1.6 Tipo de empresa

EMPRESA PRIVADA - MICROEMPRESA

1.1.7 Número de empleados

2 empleados – 6 contratistas

1.2 Formulación del problema

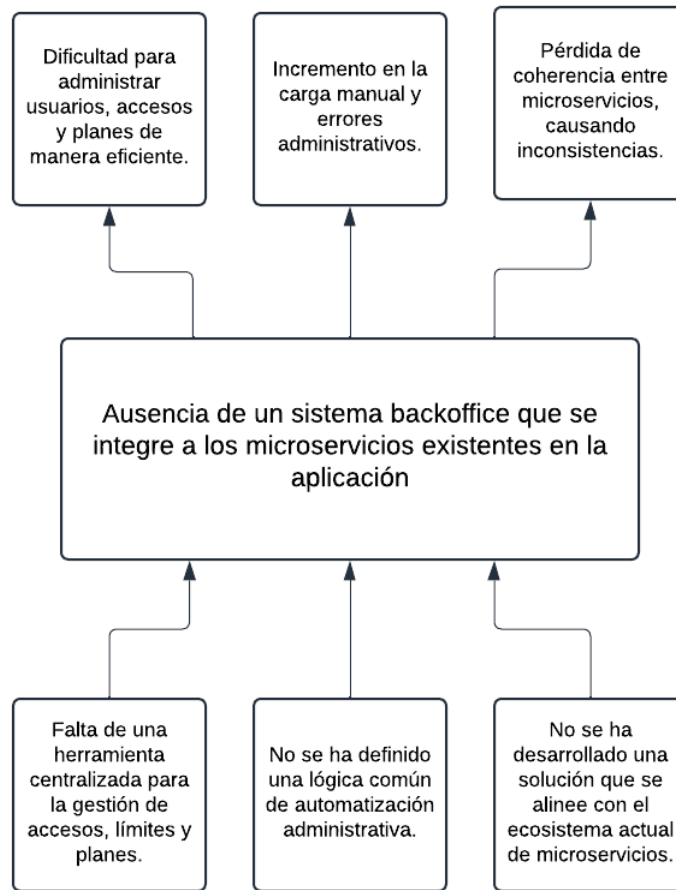


Imagen 1 Árbol de problemas

1.2.1 Descripción del problema

SIAN EXUS enfrenta actualmente dificultades operativas debido a la ausencia de un entorno de tipo BackOffice centralizado para la administración de sus servicios y soluciones tecnológicas, como facturación electrónica, nómina digital y comercio electrónico; que cuente con una buena gestión de accesos, límites y planes, que también integre diferentes sistemas claves y cuente con una eficiente automatización de procesos administrativos. Esta situación limita la gestión eficiente de servicios, usuarios y recursos técnicos, todo esto puede repercutir en problemas para administrar usuarios, planes y accesos, y si no se encuentra integrada al ecosistema de microservicios se presenta una pérdida de coherencia de la aplicación e inconsistencias como se muestra en la Imagen 1.

En contextos similares, se ha demostrado que las arquitecturas distribuidas junto con mecanismos de comunicación asincrónica como RabbitMQ que permiten mejorar la resiliencia, desacoplar servicios y facilitar la escalabilidad [3], [4]. Sin embargo, los procesos administrativos de la gestión de servicios de la empresa SIAN EXUS, aún no cuentan con la implementación de este mecanismo de comunicación con el resto de los microservicios.

Por lo tanto, se identifica una necesidad urgente de implementar un servicio e interfaz de tipo BackOffice donde se implemente la administración de los servicios y soluciones ofrecidas y se comunique de forma asíncrona con los demás microservicios existentes en el ecosistema SIAN EXUS, con el fin de mejorar el control operativo, reducir cargas administrativas y garantizar la escalabilidad y seguridad del sistema.

1.2.2 Definición del problema

Tras analizar el problema, el principal desafío que se debe resolver es: ¿Cómo crear un sistema centralizado de administración que ayude a SIAN EXUS a gestionar sus servicios y soluciones tecnológicas, garantizando seguridad, escalabilidad y un mejor control operativo?

A partir de este problema clave, surgen preguntas más específicas:

- ¿Qué arquitecturas son las que la empresa tiene ya tiene integradas para gestionar el desarrollo de software?
- ¿Qué tecnología se implementó en la empresa para la comunicación entre componentes del sistema?
- ¿Qué medidas de seguridad y diseño se necesitan para gestionar accesos y personalizar planes de manera segura?
- ¿Cómo se puede minimizar la carga administrativa y hacer más sencillo el monitoreo del uso de los servicios?
- ¿Qué indicadores y controles debemos incluir para asegurar que el servicio mantenga su calidad, disponibilidad y escalabilidad?

Con base en estas preguntas fue guiado el desarrollo del proyecto, asegurando que la solución no solo cubra las necesidades actuales, sino que también esté preparada para crecer de forma segura en el futuro.

1.3 Alcance

1.3.1 Declaración del alcance

El presente proyecto abarca el diseño y desarrollo de una plataforma BackOffice para la empresa SIAN EXUS, orientada a sus administradores internos, con el fin de gestionar eficientemente a sus empresas clientes, sus planes contratados, accesos y uso de servicios como facturación electrónica, nómina digital y E-Commerce. La plataforma se realizó bajo una arquitectura distribuida, soportada por una comunicación asíncrona mediante RabbitMQ, e incluye módulos interconectados que se comuniquen con los microservicios existentes de la empresa. El alcance abarca la construcción de las interfaces administrativas, la lógica de negocio para la asignación y control de planes, la gestión de accesos a nivel de empresa cliente, así como los mecanismos de monitoreo de uso.

Este proyecto no contempla el desarrollo de las soluciones finales del cliente (por ejemplo, el sistema de facturación), sino la integración con sus servicios existentes a través de colas de mensajería. Tampoco se aborda la integración con pasarelas de pago ni el soporte para multimoneda.

1.3.2 Objetivo General

Implementar una plataforma tipo BackOffice que permita a los administradores de SIAN EXUS gestionar y monitorear de forma centralizada las empresas clientes, sus accesos, planes contratados y uso de servicios tecnológicos

1.3.3 Objetivos específicos

- Diseñar la arquitectura del sistema que permita una integración eficiente de bases de datos, así como el manejo de la comunicación asíncrona.
- Implementar las funcionalidades necesarias para la administración de usuarios, gestión de roles, configuración de planes de servicio y control de límites de uso.
- Implementar la comunicación entre servicios usando RabbitMQ para garantizar escalabilidad y desacoplamiento.
- Evaluar la plataforma mediante pruebas funcionales y de rendimiento, validando su utilidad en un entorno real de SIAN EXUS.

1.3.4 Relación objetivos específicos - resultados esperados

Tabla 1 Relación Objetivos específicos - Resultado esperados

Objetivo específico	Resultados obtenidos
Diseñar la arquitectura del sistema que permita una integración eficiente de bases de datos, así como el manejo de la comunicación asíncrona.	Un documento técnico que describa la arquitectura del sistema, incluyendo el diagrama de componentes, tecnologías seleccionadas (bases de datos relacional) y la estrategia de comunicación asíncrona.

Objetivo específico	Resultados obtenidos
Implementar las funcionalidades necesarias para la administración de usuarios, gestión de roles, configuración de planes de servicio y control de límites de uso.	● Módulos funcionales que permitan a los administradores de SIAN EXUS administrar usuarios, asignar roles y configurar planes personalizados. ● Panel de control con métricas y restricciones de uso activas por cliente, incluyendo alertas automáticas al superar límites definidos.
Implementar la comunicación entre servicios usando RabbitMQ para garantizar escalabilidad y desacoplamiento.	● Servicios del sistema integrados mediante RabbitMQ, con colas y exchanges definidos para eventos clave (registro de empresas, cambios de plan, alertas de uso). ● Reducción comprobada del acoplamiento entre servicios a través de pruebas de aislamiento y simulaciones de carga distribuidas.
Evaluar la plataforma mediante pruebas funcionales y de rendimiento, validando su utilidad en un entorno real de SIAN EXUS.	● Informe de pruebas funcionales que evidencien el cumplimiento de los requerimientos definidos, con participación de usuarios internos de SIAN EXUS. ● Reporte de pruebas de rendimiento bajo escenarios simulados de uso real, incluyendo tiempos de respuesta, comportamiento bajo carga y estabilidad del sistema.

1.3.5 Marco lógico de la propuesta

Tabla 2 Marco lógico

Objetivo Específico	Entregable	Indicador	Meta
Diseñar la arquitectura del sistema que permita una integración eficiente de bases de datos, así como el manejo de la	Documento de arquitectura técnica Diagramas de	Diseño arquitectónico validado por equipo técnico	70 % de los diseños necesarios para la construcción de la arquitectura del sistema

Objetivo Específico	Entregable	Indicador	Meta
comunicación asincrónica.	integración y comunicación		
Implementar las funcionalidades necesarias para la administración de usuarios, gestión de roles, configuración de planes de servicio y control de límites de uso.	Módulos funcionales para administración y monitoreo de las empresas usuario	Número de funcionalidades validadas en entorno de pruebas	90% de funcionalidades críticas habilitadas para el administrador
Implementar la comunicación entre servicios usando RabbitMQ para garantizar escalabilidad y desacoplamiento.	Servicios interconectados mediante colas	Porcentaje de eventos transmitidos y consumidos correctamente	100% de eventos clave manejados de forma asincrónica sin pérdida de mensajes
Evaluar la plataforma mediante pruebas funcionales y de rendimiento, validando su utilidad en un entorno real de SIAN EXUS.	Informe de pruebas Reportes de rendimiento y métricas	Porcentaje de pruebas superadas y rendimiento bajo carga aceptable	Plataforma validada con al menos 90% de pruebas exitosas

2. Marco referencial

2.2. Marco Conceptual

Las empresas cuentan con una necesidad administrativa donde se requiere una gestión de servicios, productos y/o soluciones tecnológicas para no perder competitividad en el mercado, todas estas interacciones no son dirigidas hacia el cliente sino hacia los empleados y administrativos de las empresas, esto es lo que se conoce como BackOffice.

En el contexto de la empresa SIAN EXUS ellos ofrecen soluciones y servicios los cuales son:

- Facturación electrónica: Permite a los clientes generar comprobantes fiscales cumpliendo con las normativas y leyes del país.

Es la evolución de la factura tradicional de papel y, para efectos legales, tiene la misma validez. Sin embargo, se genera, valida, expide, recibe, y conserva electrónicamente, lo que representa mayores ventajas.

Tributariamente, es un soporte de transacciones de venta de bienes o servicios [6].

Es uno de los procesos más críticos en empresas proveedoras de servicios financieros y de software contable.

- Nómina Electrónica: Son los registros digitales relacionados con el pago a los empleados
- E-Commerce: esto se refiere al proceso de compra y venta de productos o servicios por medio de plataformas o elementos electrónicos.

La solución tecnológica planteada en este proyecto se encuentra definida sobre una arquitectura conocida como MVC (Modelo-Vista-Controlador) este patrón de diseño se encarga de separar las responsabilidades en 3 áreas el modelo, las vistas y el controlador.

- Modelo: es un conjunto de clases que representan la información del mundo real que el sistema debe procesar.
- Vistas: Las vistas son el conjunto de clases que se encargan de mostrar al usuario la información contenida en el modelo.
- Controlador: El controlador es un objeto que se encarga de dirigir el flujo del control de la aplicación debido a mensajes externos, como datos introducidos por el usuario u opciones del menú seleccionadas por él. A partir de estos mensajes, el controlador se encarga de modificar el modelo o de abrir y cerrar vistas.

El patrón de diseño MVC es bastante utilizado en la actualidad en marcos de aplicación orientados a objetos [19].

Es importante tener en cuenta que la plataforma se integrara en un ecosistema donde los servicios se comunican de forma asíncrona. Es decir, las funcionalidades de la empresa se implementan como plataformas independientes, pero con un patrón de intercambio de mensajería donde el emisor no espera una respuesta inmediata del receptor [7].

El modelo B2B SaaS (Business to Business - Software as a Service) hace referencia a plataformas que ofrecen sus servicios a otras empresas, este tipo de soluciones permiten a las organizaciones cliente poder acceder a herramientas especializadas sin necesidad de tener una infraestructura propia prometiéndole a estas reducir costos operativos y reducir el margen de error en algunos procesos. Este modelo tiene como pilares la escalabilidad, la multitenencia (la capacidad de que varias empresas usen el mismo sistema al mismo tiempo, pero cada una con su propia información separada y segura) y la posibilidad de ofrecer actualizaciones constantes sin interrupción del servicio [12].

En ese contexto se encuentra la empresa SIAN EXUS, la cual ofrece servicios contables y tecnológicos a otras empresas. Por este motivo, desarrollar una plataforma BackOffice basada en un modelo B2B SaaS no solo se ajusta a su estrategia empresarial, sino que también ofrece la posibilidad de aumentar la productividad, mejorar la administración de clientes y fortalecer su conjunto de herramientas digitales.

2.3. Marco teórico

2.3.1. Teoría General de sistemas

La Teoría General de Sistemas fue propuesta como respuesta a las limitaciones del enfoque reduccionista en la ciencia. Esta establece que los sistemas son conjuntos de elementos en interacción mutua, organizados en relaciones dinámicas que persiguen un objetivo común.

Esta teoría nos muestra un enfoque holístico sobre los sistemas, esto nos ayuda a entender que este proyecto se encargará de desarrollar un sistema que tiene subsistemas que lo conforman, pero también hacer parte de un ecosistema de sistemas donde debe integrarse para hacer parte de la suma de un todo (ecosistema de sian exus) [8].

2.3.2. Teoría de los sistemas de información

La teoría de los sistemas de información permite entender como los sistemas diseñados para procesar datos, pueden apoyar la toma de decisiones, la gestión operativa y el control de organizaciones, definiendo un modelo llamado Input-Proceso-Output que se encarga de capturar, procesar, almacenar y distribuir la información [9].

aquí un ejemplo de la aplicación de este modelo sobre la plataforma a desarrollar en este proyecto:

Tabla 3 Modelo Input-Proceso-Output

Concepto	Definición	Implementación
Input	Datos de entrada al sistema	- Registro de una nueva empresa cliente - Solicitud de una empresa para hacer un cambio de plan

Concepto	Definición	Implementación
Proceso	Transformación de los datos en acciones	- Asignación automática de permisos según el plan contratado. - Cálculo de uso mensual de servicios
Output	Resultados	- Alertas: "Empresa X ha usado el 90% de su límite de facturas". - Reporte de clientes inactivos.
Retroalimentación	Ajustes basados en los resultados	- Reclasificación automática de un cliente a un plan superior por uso constante.

2.3.3. Teoría de la comunicación

La comunicación es un proceso donde sus partes son: emisor, mensajes, canal, receptor y posibles interferencias, llevado al campo tecnológico toda plataforma administrativa de usuarios necesita transmitir diferentes mensajes de forma estructurada, comprensible y sin interferencias. Por ejemplo, si un sistema emite una alerta deberá ser: visible al usuario correspondiente, en el momento preciso y ser comprensible [10].

Podemos decir que los sistemas informáticos deben estar estructurados para facilitar la comunicación interactiva, asincrónica y en red, esto concuerda mucho con la necesidad presentada en este proyecto puesto que esta plataforma debe permitir una comunicación constante entre administradores, usuarios y entornos externos. Por lo tanto, incorporar principios de esta teoría permite diseñar interfaces claras, flujos de información bien definidos y protocolos de integración coherentes, minimizando errores operativos y mejorando la experiencia del usuario [11].

2.4. Estado del Arte

2.4.1. Evolución de plataformas BackOffice bajo el patrón Modelo-Vista-Controlador

Las plataformas administrativas modernas, especialmente las que están orientadas a la gestión de clientes en entornos empresariales, han implementado patrones de diseño que promueven la separación de responsabilidades y el mantenimiento eficiente y continuo del código. El patrón de diseño Modelo-Vista-Controlador (MVC) es uno de los más usados por estas plataformas.

Ejemplos de aplicación de este patrón:

Java Swing en aplicaciones de escritorio: El patrón MVC es ampliamente utilizado en aplicaciones construidas con Java Swing. Se ha demostrado que su implementación mejora la organización del código y la independencia entre el comportamiento de la lógica y la presentación gráfica. Un estudio clásico sobre este tema muestra cómo dividir correctamente los componentes en una aplicación de gestión con interfaz rica mejora su mantenibilidad y escalabilidad [19].

Django y Ruby on Rails en aplicaciones web: Estos frameworks web siguen el patrón MVC (o variantes como MTV en Django) y se utilizan en múltiples plataformas administrativas modernas. Su estructura basada en MVC permite que el desarrollo BackOffice sea ágil, modular y más fácil de integrar con bases de datos y APIs externas [13][14].

2.4.2. Plataformas BackOffice modulares y extensibles

Hoy en día el concepto BackOffice se ha adoptado en muchas plataformas de gestión y administración de clientes, más específico en clientes finales (compradores o consumidores), algunas de estas plataformas son:

- Zuora: Esta es una empresa de software empresarial que proporciona aplicaciones para que otras empresas gestionen sus servicios basados en suscripciones. Esta plataforma permite a sus clientes monitorear elementos como pagos de suscripciones, facturación, catálogos de productos y contabilidad [15].
- xFusion: Esta empresa ofrece soluciones de BackOffice para empresas SaaS. Su servicio ofrece automatización de procesos garantizando precisión y calidad. También, dan importancia a la seguridad de datos, asegurando a sus clientes que la información sensible está protegida y segura [16].

Estos ejemplos priorizan y ofrecen un backoffice para sus empresas clientes, donde ellos accedan a este panel de control y puedan administrar sus propios clientes.

2.5. Antecedentes

Diversas plataformas han sido desarrolladas con el fin de facilitar la administración de entidades o clientes dentro de un ecosistema empresarial. Aunque cada una varía en su alcance y complejidad y público objetivo, comparten el mismo objetivo de proporcionar visibilidad, control y gestión eficiente de los usuarios, servicios o recursos bajo su dominio. A continuación, se analizan algunos proyectos que, en contextos similares, abordan necesidades análogas a las del presente trabajo, aunque con diferencias claves en su orientación funcional y su público objetivo.

Google Workspace Admin Console

Esta es una plataforma de gestión centralizada que ofrece Google orientada a las organizaciones permitiéndoles administrar cuentas, servicios y configuraciones, ajustada para administradores IT (Information Technology), gerentes de seguridad o responsables operativos permitiéndoles ejercer control sobre todo el ecosistema de herramientas de Google en su organización desde un solo lugar [17].

Zoho One Admin Panel

La plataforma zoho ofrece herramientas integradas para gestión empresarial bajo un modelo multiciente. Aunque presenta una interfaz que incluye muchas funciones, su complejidad puede convertirse en una barrera para pymes (pequeñas y medianas empresas) que requieren una solución más especializada y ágil, ajustada a necesidades particulares. Adicionalmente, su flexibilidad depende del conocimiento técnico de los usuarios, lo que puede suponer una barrera en entornos con bajo nivel de especialización digital [18].

Tabla 4 Comparativa con antecedentes

Plataforma	Público Objetivo	Ventajas Principales	Limitaciones
Google Workspace Admin Console	Administradores IT de organizaciones	Integración completa con herramientas de Google; gestión desde una interfaz unificada	Limitado a servicios del ecosistema Google; enfoque generalista, no por cliente
Zoho One Admin Panel	Empresas que gestionan múltiples áreas	Alto grado de integración entre módulos; variedad de herramientas empresariales	Interfaz compleja; curva de aprendizaje elevada; poco enfoque personalizado

Propuesta SIAN EXUS BackOffice	Administradores internos de SIAN EXUS para gestionar clientes empresariales	Interfaz personalizada; enfoque directo a gestión de empresas clientes; integración específica con servicios internos	No orientada al público general; requiere desarrollo y mantenimiento continuo
-----------------------------------	---	---	---

3. Metodología

3.2. Metodología de Investigación

Este proyecto se desarrolló bajo un enfoque de investigación aplicada con orientación tecnológica, centrado en resolver una necesidad concreta de la empresa SIAN EXUS, esta necesidad es la administración centralizada de clientes, planes y accesos mediante una plataforma BackOffice que se integre y comunique con los demás servicios existentes en el ecosistema web de SIAN EXUS. El proceso parte de una estructura definida e inicializada y tecnologías existentes, que serán integradas y validadas para su uso en el desarrollo de la plataforma.

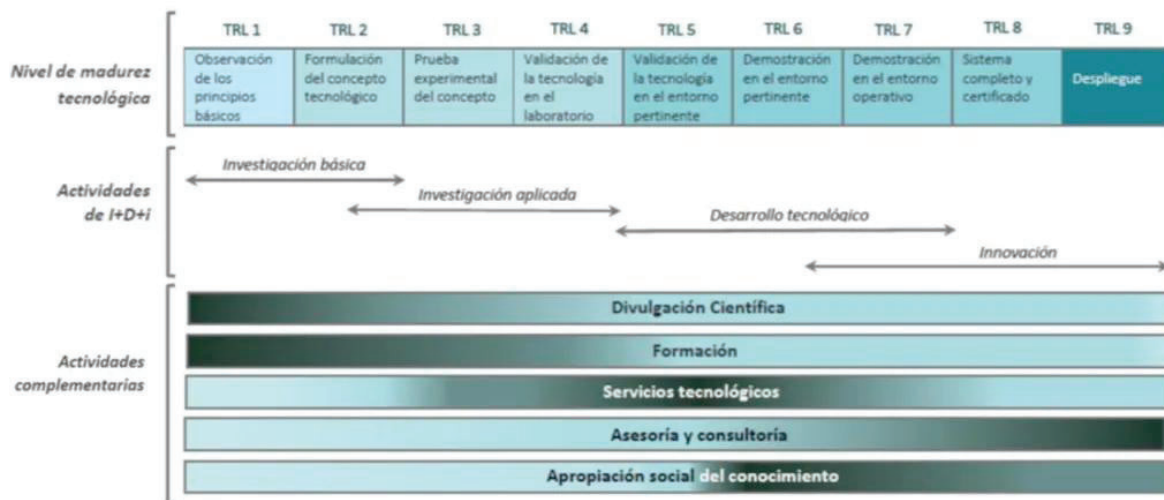


Imagen 2 Niveles de madurez tecnológica TRL. Tomado de [20]

Lo que se hizo en este proyecto es implementar un BackOffice web, no es algo innovador ni que se está inventando en este proyecto, sino más bien se integró y adaptó al ecosistema de servicios de la empresa SIAN EXUS.

Los BackOffice Web ya están validados conceptual y técnicamente. Howcroft, D., & Richardson en el artículo “The back office goes global: exploring connections and contradictions in shared service centres” demuestran que el concepto BackOffice ya se encuentra globalmente reconocido por el área empresarial y ya es usado para la gestión empresarial en muchas partes del mundo. Por lo tanto, podemos asegurar que el desarrollo de este proyecto no se encuentra en el TRL 5 que lo que busca es la validación de la tecnología como se muestra en la imagen 2, sino ya pasa a ser una demostración en un entorno pertinente, más explícitamente el entorno de la empresa SIAN EXUS. Es decir que el inicio del desarrollo del proyecto está en el TRL 6 [21][20].

Ahora, para definir hasta qué nivel de madurez tecnológica llega el desarrollo de este proyecto debemos tener en cuenta que se hizo el despliegue en entorno real de la plataforma BackOffice para que los administradores de la empresa hagan uso de esta. A simple vista y

con la información que nos ofrece la imagen 2, se podría deducir que su alcance de madurez tecnológica es hasta el TRL 9. Sin embargo, TRL 9 implica un sistema probado y comprobado en operación real sostenida durante un período significativo, es decir:

- Que el sistema no solo esté desplegado, sino que ya haya sido usado continuamente, validado completamente por los usuarios finales, con documentación, soporte y evidencia de que está funcionando como parte integral de la operación de la empresa.
- Se requiere también que no haya modificaciones sustanciales pendientes, y que esté listo para replicarse o escalar sin más pruebas piloto.

Aunque la plataforma al finalizar el proyecto pueda ser utilizada por los administradores desde la web al final del proyecto, hay que tener ciertas consideraciones como:

- El sistema probablemente apenas estará entrando en la fase de validación operativa.
- Una vez desplegada la plataforma, puede requerir ajustes, mejoras o complementos tras recibir retroalimentación continua.
- No es seguro que la plataforma al finalizar la pasantía ya esté completamente madura, documentada, estabilizada y mantenida en un periodo de tiempo extendido.

Por lo tanto, el proyecto deja listo un sistema para el TRL 9, pero no necesariamente lo alcanza. Por estas razones se puede deducir que el nivel de madurez tecnológica que alcanzó el proyecto es TRL 8 un sistema completo y certificado. Finalmente llegando a la conclusión de que el proyecto estuvo, según la clasificación establecida por el Ministerio de Ciencia, Tecnología e Innovación de Colombia, en los niveles, TRL 6 hasta TRL 8 [Imagen 2][20].

3.3. Metodología de desarrollo de software

Para definir la metodología de desarrollo se tomaron 3 diferentes modelos de desarrollo de software para poder compararlos, calificarlos y al final concluir con cual es el más indicado para utilizar.

Antes de presentar los modelos a comparar, cabe aclarar que la empresa SIAN EXUS ya tienen unas costumbres y su propio modelo de desarrollo al tuvo que acoplarse este proyecto. Debido a que no se tiene un seguimiento bajo un modelo de desarrollo, me abro paso a hacer una investigación y comparación sobre modelos demostrados técnica y teóricamente y así encontrar una metodología que se ajuste al flujo de trabajo existente en la empresa.

Flujo operativo de la empresa SIAN EXUS

El equipo de desarrollo de SIAN EXUS opera bajo un enfoque iterativo y colaborativo, centrado en la entrega de versiones funcionales del software. El flujo de trabajo está organizado mediante reuniones diarias (daily meetings), en las que los miembros del equipo reportan avances, dificultades y objetivos inmediatos. Estas reuniones permiten mantener una comunicación constante y visibilidad sobre el estado del proyecto.

No se trabaja con sprints rígidos ni límites de tiempo predefinidos por iteración; en cambio, se establece una versión objetivo del software con una fecha de lanzamiento estimada. A partir de esta, las tareas se organizan de forma flexible, pero con responsabilidad clara hacia la entrega estable y funcional de esa versión.

Durante el desarrollo, el equipo de calidad ejecuta pruebas continuas para detectar errores (bugs) en los módulos en construcción o ya desplegados. Los errores identificados se documentan y asignan como tareas de corrección que se integran al flujo de trabajo con prioridad dependiendo de su criticidad. Estas correcciones, junto con nuevas funcionalidades, se integran en versiones sucesivas del sistema, lo que permite lanzar mejoras frecuentes sin necesidad de reconstruir el sistema completo.

Este enfoque promueve una evolución continua del producto, facilitando validaciones permanentes con usuarios internos y asegurando la estabilidad operativa de cada versión desplegada.

Con esto claro ahora si podemos hacer un análisis de cada modelo presentado.

Modelo Incremental

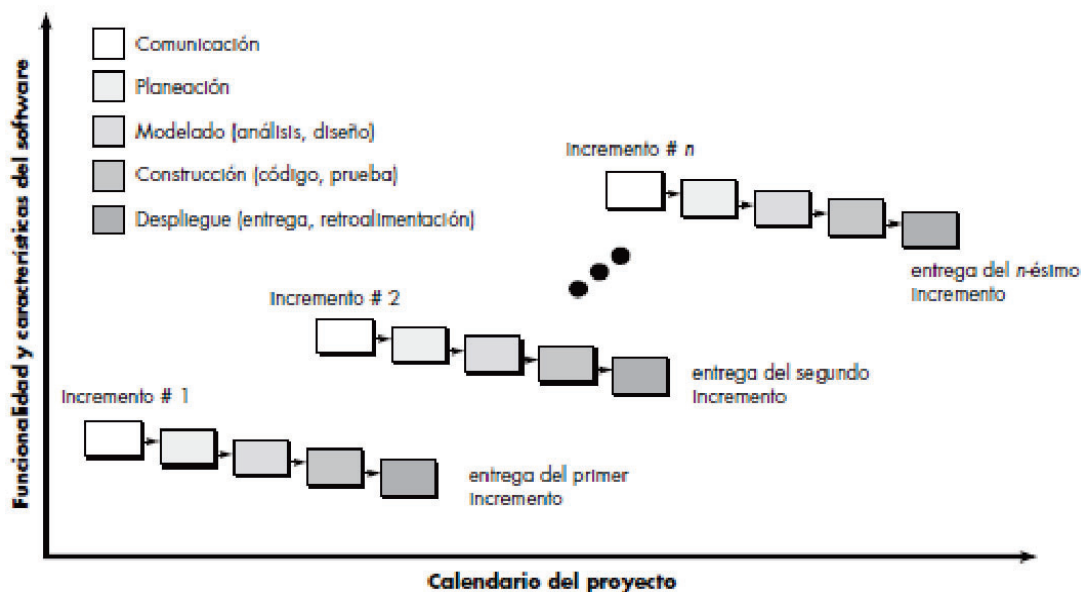


Imagen 3 Grafica de modelo incremental. Tomado de [27]

El modelo incremental organiza el desarrollo de software en una secuencia de versiones funcionales, cada una construida sobre la anterior como se muestra en la imagen 3, estas son conocidas como incrementos y esta entrega de incrementos se asemeja a las versiones que se entregan en la empresa SIAN EXUS. En este modelo las funcionalidades deben estar bien definidas desde el inicio. Este enfoque parte de una visión clara del producto terminado y permite una planificación estructurada de entregas parciales, favoreciendo la detección temprana de errores y la entrega continua de valor al cliente. Es decir, el cliente

continuamente en cada entrega de incremento puede hacer uso de esta “versión” llamada incremento en el modelo. Este modelo es particularmente útil cuando los requisitos están bien establecidos desde el principio y el producto puede construirse por partes independientes [22].

Modelo Evolutivo

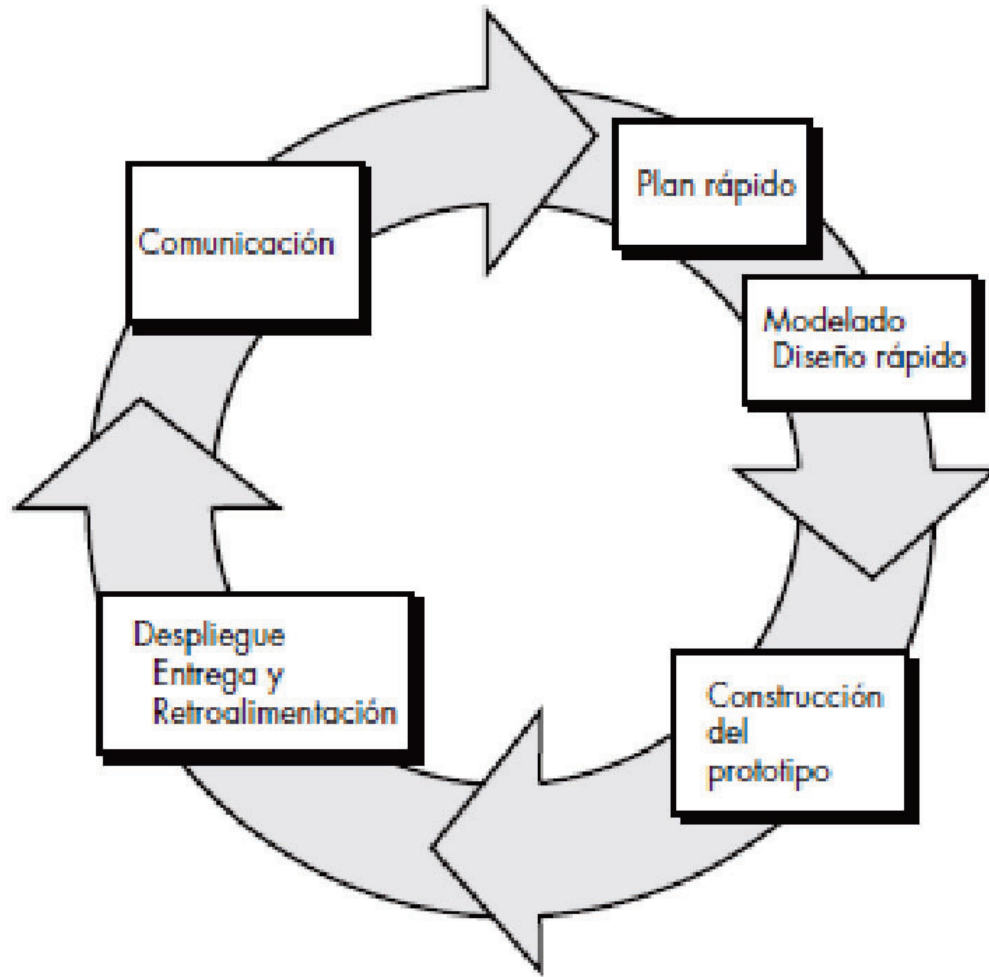


Imagen 4 Grafica de modelo evolutivo. Tomado de [27]

El modelo evolutivo se basa en la construcción de un sistema que se va refinando a lo largo del tiempo, a través de ciclos iterativos que integran la retroalimentación de los usuarios [Imagen 4]. Justo como lo hace la empresa SIAN EXUS, que constantemente está recibiendo retroalimentación por el equipo de calidad de software y los clientes, permitiendo que la aplicación web crezca de forma evolutiva. Este modelo en lugar de comenzar con una especificación completa, se parte de una versión inicial que se ajusta y mejora progresivamente. Este modelo es apropiado cuando los requisitos no están completamente definidos o pueden cambiar significativamente durante el proceso de desarrollo [23].

Modelo Evolutivo-Incremental

El modelo evolutivo-incremental combina lo mejor de los enfoques incremental y evolutivo. El desarrollo se estructura por módulos (incrementos o versiones), pero estos pueden evolucionar mediante iteraciones sucesivas que refinan y mejoran el producto. Esta metodología permite entregar versiones funcionales tempranas mientras se mantiene una apertura al cambio, siendo ideal para contextos dinámicos donde se necesita adaptabilidad sin sacrificar estructura [22][23].

Tabla 5 Comparativa de modelos

Criterio	Modelo Incremental	Modelo Evolutivo	Modelo Evolutivo-Incremental
Naturaleza del desarrollo	Modular, basado en entregas sucesivas planificadas.	Adaptativo, basado en versiones mejoradas por retroalimentación.	Modular + adaptativo, con ajustes por versión.
Claridad de requisitos	Requiere definición completa o casi completa al inicio.	Permite requisitos ambiguos o emergentes.	Mezcla ambos: algunos requisitos claros, otros evolucionan.
Retroalimentación	Limitada; se da en puntos definidos.	Continúa y es esencial para el desarrollo.	Continua y adaptada por versión.
Participación del usuario	Baja o en fases puntuales.	Alta y constante.	Alta y tras cada entrega funcional.
Adaptabilidad al cambio	Baja. Los cambios son costosos y tardíos.	Alta. Cambios son parte del proceso.	Alta. Los cambios se integran progresivamente.

Criterio	Modelo Incremental	Modelo Evolutivo	Modelo Evolutivo-Incremental
Riesgo	Bajo si los requisitos son estables.	Alto si no se gestiona bien el cambio.	Medio: equilibrio entre control y adaptabilidad.
Velocidad de entrega	Entrega funcional completa solo al final.	Muestra avances parciales desde temprano.	Entregas funcionales desde fases tempranas.

Factores de Evaluación para la Elección del Modelo

Tabla 6 Factores de evaluación para la elección del modelo

Factor	Descripción
Claridad de los requisitos	Qué tan definidos están los requerimientos desde el inicio.
Participación del cliente/usuario	Nivel de intervención esperada por parte de usuarios y stakeholders durante el desarrollo.
Adaptabilidad a cambios	Facilidad del modelo para ajustarse a cambios o nuevas necesidades.
Retroalimentación continua	Capacidad del modelo para incorporar mejoras según el uso real del sistema.
Entrega de valor temprano	Qué tan pronto se puede entregar funcionalidad operativa.
Despliegue continuo	Capacidad del modelo para desplegar cambios siguiendo un orden claro y constantemente

Tabla de Evaluación por Puntaje

Se asigna una escala de 1 a 5, donde 1 es muy malo y 5 es muy bueno:

Tabla 7 Evaluación de modelos por puntaje

Factor	Incremental	Evolutivo	Evolutivo-Incremental
Claridad de requisitos	5	2	4
Participación del usuario	2	5	5
Adaptabilidad a cambios	2	5	5
Retroalimentación continua	2	5	5
Entrega de valor temprano	3	4	5
Despliegue continuo	5	3	5
Puntaje total	19	24	29

Con base en esta evaluación, el modelo evolutivo-incremental se posiciona como el más adecuado para el desarrollo del BackOffice en SIAN EXUS, ya que combina entregas funcionales con alta capacidad de adaptación a requisitos cambiantes, soporte para retroalimentación continua y facilidad de integración modular, características fundamentales en el entorno real de la empresa con validación progresiva por parte de sus clientes.

3.4. Metodología de gestión de actividades

Para definir una metodología de gestión de actividades se tuvo en cuenta la misma consideración que la metodología de desarrollo la cuál es que la empresa SIAN EXUS ya cuenta con un flujo de trabajo establecido aunque no acoplado y adoptado a ningún modelo demostrado técnica y prácticamente. Entonces se hizo una comparación entre algunos modelos y así poder encontrar una metodología que se ajuste al flujo de trabajo que ya existe en la empresa y definir el marco de trabajo de este proyecto usando técnicas patentadas y demostradas anteriormente.

El equipo de desarrollo realiza reuniones diarias (daily meetings) para mostrar e informar avances, identificar y/o mencionar bloqueos y organizar prioridades, esto va muy ligado con las prácticas establecidas en la metodología agile Scrum [24]. Sin embargo, a diferencia de esta metodología, en la empresa no se trabaja bajo iteraciones con tiempo fijo, que en Scrum

se conocen como sprints, sino que se definen versiones funcionales del software con fechas de entrega estimada, lo que permite una mayor flexibilidad en la planificación y ejecución de tareas, alineándose con el flujo continuo propuesto por Kanban [25]. Al mismo tiempo, el equipo QA (Quality Assurance en español Aseguramiento de la Calidad) realiza validaciones continuas para detectar errores, y los defectos identificados son incorporados de inmediato al flujo de trabajo y corregidos progresivamente, buscando que los ajustes sean incluidos en la versión estable más próxima a desplegar. Esta dinámica refleja principios del enfoque DevOps, orientado a la integración y entrega continua (CI/CD) [26]. Al final, podemos concluir que se trabajó con una **metodología híbrida** que combina prácticas de Scrum, Kanban y DevOps, esta metodología híbrida nos permitió mantener un ritmo constante de desarrollo, atender eficazmente las incidencias reportadas, garantizar la entrega oportuna de versiones estables del sistema y no pierde el sentido de flujo de trabajo que ya está implementado por la empresa.

4. Ingeniería del Proyecto

4.1 Análisis y diseño del sistema

El proceso de análisis y diseño del sistema permitió establecer la estructura técnica, los componentes y las interacciones necesarias para el correcto funcionamiento de la plataforma BackOffice de SIAN EXUS. Esta fase se desarrolló bajo el modelo evolutivo-incremental, garantizando la integración progresiva de módulos funcionales y la validación continua con los administradores de la empresa. Para la elección de este modelo se realizaron sesiones con los administrativos y líderes de desarrollo de SIANEXUS destacando su practicidad y orden.

La decisión de utilizar el modelo evolutivo-incremental se tomó después de analizar las necesidades y la dinámica de trabajo del proyecto, especialmente la participación constante del equipo de QA dentro del proceso de desarrollo. Debido a que las pruebas y revisiones se realizaban de manera continua, se identificó que era necesario adoptar una metodología que permitiera desarrollar y entregar funcionalidades de forma progresiva, facilitando así la validación constante del sistema.

Se consideró que este modelo era el más adecuado porque permitía dividir el desarrollo en incrementos pequeños y manejables, haciendo posible que cada módulo o funcionalidad fuera evaluado y corregido antes de continuar con nuevas etapas. Esta característica resultó fundamental para mantener una retroalimentación permanente entre el equipo de desarrollo y QA, ya que los hallazgos encontrados podían solucionarse rápidamente sin afectar de manera significativa el resto del proyecto.

Asimismo, la adopción de este modelo contribuyó a reducir riesgos durante el desarrollo, mejorar el control sobre la calidad del código y asegurar una mayor estabilidad del sistema en cada iteración. Gracias a este enfoque, fue posible detectar errores de manera temprana, implementar mejoras continuas y adaptar el desarrollo conforme surgían nuevas necesidades o ajustes dentro del proyecto.

4.1.1 Arquitectura general del sistema

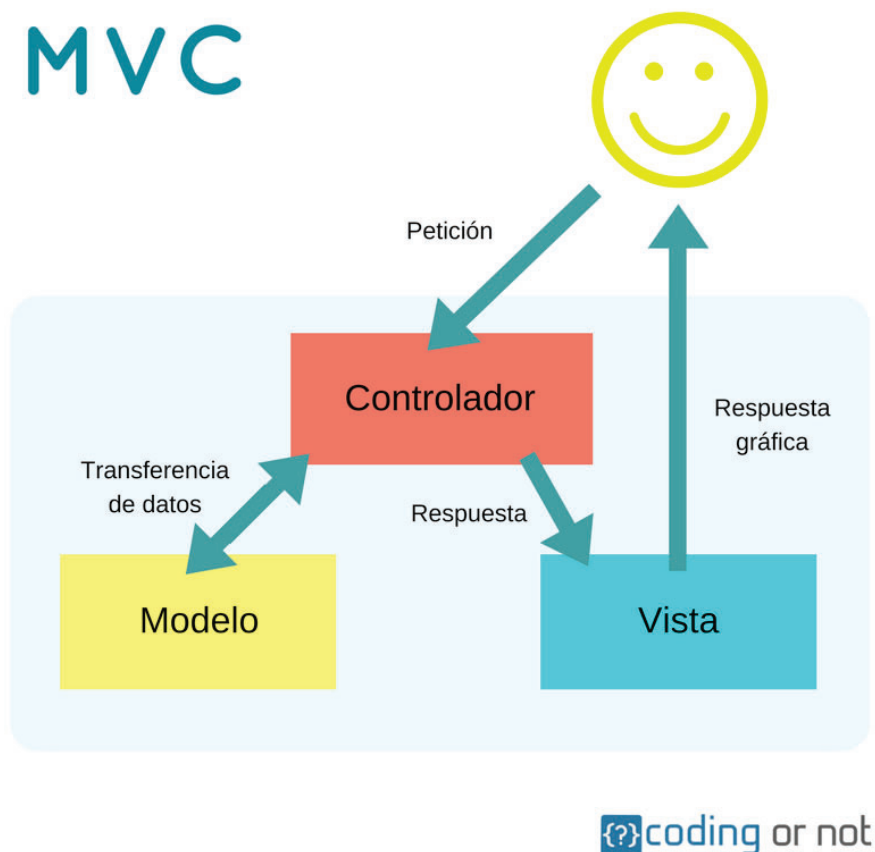


Imagen 5 Representación gráfica del MVC obtenida de [28]

Esta parte fue fundamental para el desarrollo del sistema puesto que ofreció escalabilidad y fácil mantenimiento debido a la separación de responsabilidades que ofrece el patrón de diseño seleccionado. La plataforma fue diseñada bajo el patrón de diseño Modelo-Vista-Controlador (MVC) como se visualiza en la Imagen 5 utilizando el lenguaje de programación PHP y el framework Laravel como el encargado de la lógica del negocio.

Este patrón permitió una separación clara entre la lógica de negocio, la gestión de datos y la interfaz gráfica, garantizando mantenibilidad y escalabilidad.

Componentes principales:

- **Modelo:** Representa las entidades del sistema (empresa, plan, suscripción, pago, historial). Gestiona la persistencia en la base de datos relacional.

- Vista: Interfaz web desarrollada con Livewire y Blade, responsable de la interacción directa con el usuario.
- Controlador: Gestiona las peticiones del usuario, invoca la lógica del modelo y actualiza las vistas dinámicamente.

Se implementó un diseño donde se integre en un solo repositorio la parte visual y la parte funcional y lógica de la plataforma, esto gracias al uso del framework de PHP llamado Livewire, que ofrece un renderizado dinámico de las vistas, permitiendo incluir lógica implementada por los controladores de Laravel directamente en plantillas HTML de tipo blade.

La comunicación con los demás servicios de la empresa se realizó usando la herramienta RabbitMQ como intermediario de mensajería de las plataformas. El uso de RabbitMQ le permitió a la plataforma BackOffice acceso a la información y movimientos de las demás plataformas en tiempo real y a su vez emitir información a estas como estados de suscripciones o consumo también en tiempo real, esto fue muy importante y pieza clave del desarrollo puesto que la plataforma BackOffice quedaría siendo la responsable de emitir informaciones de acceso entre otras y se necesitaba una fluidez de los datos, rápida y eficaz, permitiendo el intercambio de mensajes con los microservicios de facturación, nómina electrónica y comercio electrónico, sin generar demora ni dependencias directas entre sistemas.

4.1.2 Modelo de datos

La base de datos se diseñó en un modelo relacional almacenado como un esquema aparte de una misma base de datos para todo el ecosistema Sianexus, se integró de forma relacional porque permite mantener la integridad referencial entre empresas, planes, usuarios y registros de uso. El modelo de datos se encuentra en el siguiente enlace para mejor visualización.

[Enlace para modelo de datos](#)

Las entidades principales son:

- Empresa: Identifica a cada cliente corporativo.
- Plan: Define los límites de uso y características del servicio.
- Suscripción: Relaciona una empresa con un plan y su estado actual.

- Renovaciones de suscripción (Pagos): Registra los abonos realizados a cada suscripción.
- Historial de consumo: Almacena los consumos de folios por empresas.
- Usuario: Representa a los administradores internos del sistema.
- Rol y permisos: Controlan el acceso a las funcionalidades del BackOffice.

4.1.3 Diagrama de componentes

El sistema se compone de los siguientes módulos:

- Módulo de Autenticación y Roles: controla el acceso seguro de los administradores.
- Módulo de Empresas: permite la gestión de las empresas clientes.
- Módulo de Planes y Suscripciones: gestiona los planes, restricciones y asignaciones.
- Módulo de Renovaciones: registra pagos manuales asociados a las suscripciones.
- Módulo de Integración RabbitMQ: recibe y envía eventos con los microservicios externos.
- Módulo de Monitoreo: registra métricas de consumo y genera alertas automáticas.
- Panel de Control: centraliza los indicadores y accesos a los demás módulos.

4.2 Conexión a API externa

SIAN EXUS ofrece un sistema de gestión de negocio para otras empresas en el cual pueden incluir información administrativa que facilite la gestión de datos e información de cada una. Uno de los principales beneficios que ofrece SIAN EXUS es permitir a sus clientes emitir facturación electrónica ante la DIAN, pero SIAN EXUS no cuenta con un certificado de firma digital ni autorización de rangos de numeración de la DIAN. Debido a esto, se hace necesario utilizar una herramienta externa o soportarse sobre otro software que sí tenga estas posibilidades, y para esto se utilizó un convenio con la empresa NextPyme.

Nextpyme permite a SianExus vincular a sus clientes entregándoles los permisos para la generación de documentos ante la Dian por medio de un token de acceso, token que se debe asignar desde la plataforma BackOffice y almacenar como un dato de la empresa, este token es utilizado para enviarlo como cabecera de las peticiones que se hacen a la API de Nextpyme y así obtener los resultados esperados para cada empresa cliente.

Desde nextpyme se hace un control de **folios** (Los folios son documentos electrónicos (facturas, nóminas, notas crédito/débito) que una empresa puede emitir dentro de su plan de facturación electrónica.) permitiendo que cada empresa tenga un plan de folios y con esto controlan la cantidad de documentos que puede emitir esta empresa. SianExus como intermediario entre la empresa cliente y Nextpyme se encarga de mantener un control de los folios disponibles de esta sin llegar a ser desmedida la cantidad que permanece disponible para la empresa, pero cuidando que no le vayan a hacer falta siempre y cuando la suscripción y la cantidad de folios adquiridos por parte de SianExus se lo permita.

El control que se hace para mantener siempre disponibilidad pero sin una desmedida cantidad se debe a que NextPyme cobra a SianExus por folio asignado a sus empresas cliente.

Endpoints utilizados:

1. GET /plan/infoplanuser: Consultar la información del plan de usuario

Obtiene la información del plan asignado a una empresa incluyendo documentos disponibles, periodo del plan y tipos de documentos incluidos.

2. PUT /plan/assign-package-user

asigna un número específico de documentos (folios) a una empresa identificada por su número de documento. Este endpoint requiere token de administrador.

3. GET /plan/assign-package-user: Consultar historial de asignaciones de folios

Obtiene el historial de todas las asignaciones de paquetes realizadas. Endpoint administrativo.

4.3 Ecosistema Funcional de la Plataforma

4.3.1 Dashboard

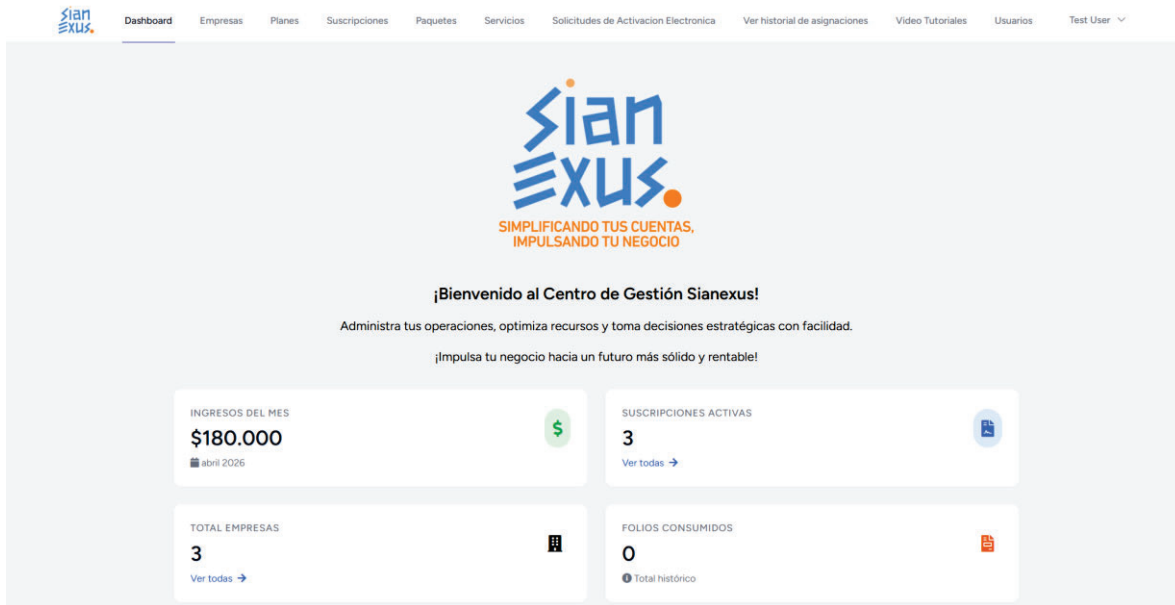


Imagen 6 captura de pantalla que muestra la parte superior del dashboard

El dashboard del BackOffice de Sianexus es un centro de control integral que proporciona a los administradores de la plataforma una visión analítica del desempeño operacional y financiero del sistema. A continuación, se describe cada componente:

4.3.1.1 Tarjetas de indicadores clave (KPI)

El dashboard cuenta con cuatro tarjetas de indicadores principales que ofrecen métricas críticas las cuales se pueden observar en la parte superior del dashboard como se ve en la Imagen 6:

- Ingresos del Mes Actual: Muestra el total de ingresos generados en el período mensual vigente, calculado mediante la suma de $(\text{precio_plan} \times \text{períodos_pagados})$ de todos los pagos registrados en el mes. Esta métrica es esencial para el monitoreo de flujo de caja y la evaluación del desempeño comercial en períodos específicos.
- Suscripciones Activas: Indica el número de suscripciones vigentes (no canceladas e identificadas como activas) en la plataforma. Este indicador permite a los administradores evaluar el crecimiento y la retención de clientes.
- Total de Empresas: Proporciona el conteo global de empresas registradas en el sistema, independientemente de su estado de suscripción. Útil para evaluar la base de usuarios y el alcance del sistema.

- Folios Consumidos: Refleja el número total de documentos procesados a través de la plataforma (facturas, nóminas, notas crédito, etc.), basado en el registro histórico de consumo. Permite evaluar el nivel de utilización de la plataforma.

4.3.1.2 Gráficos del Dashboard

Gráfico de planes más contratados

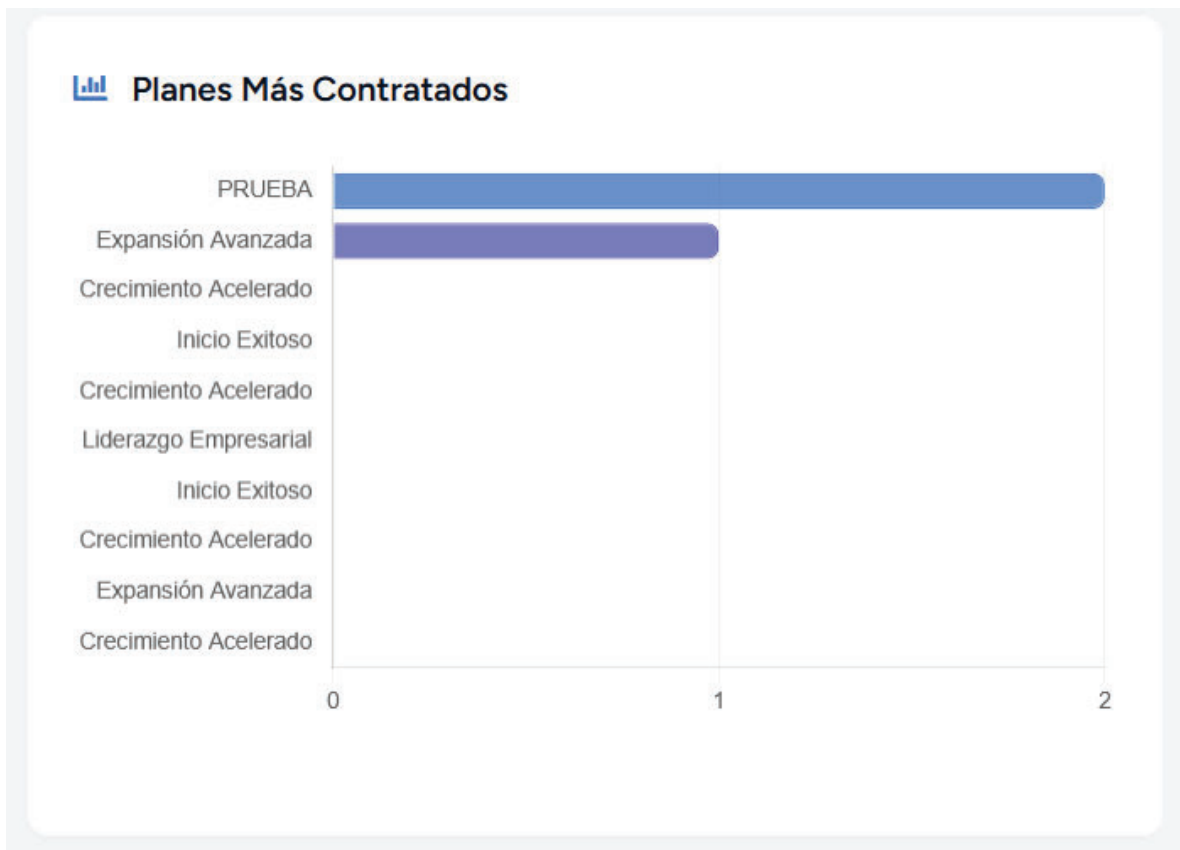


Imagen 7 captura de pantalla que muestra el grafico de los planes más contratados

Este gráfico de barras horizontales visualiza los diez planes más populares según la cantidad de suscripciones activas [Imagen 7]. Su importancia radica en:

- Análisis de Demanda: Identifica qué productos tienen mayor aceptación en el mercado.
- Estrategia Comercial: Facilita decisiones sobre promoción, mejora o discontinuación de planes.
- Información de Precios: Muestra el precio asociado a cada plan, permitiendo correlacionar popularidad con estructura de costos.

Gráfico de ingresos mensuales

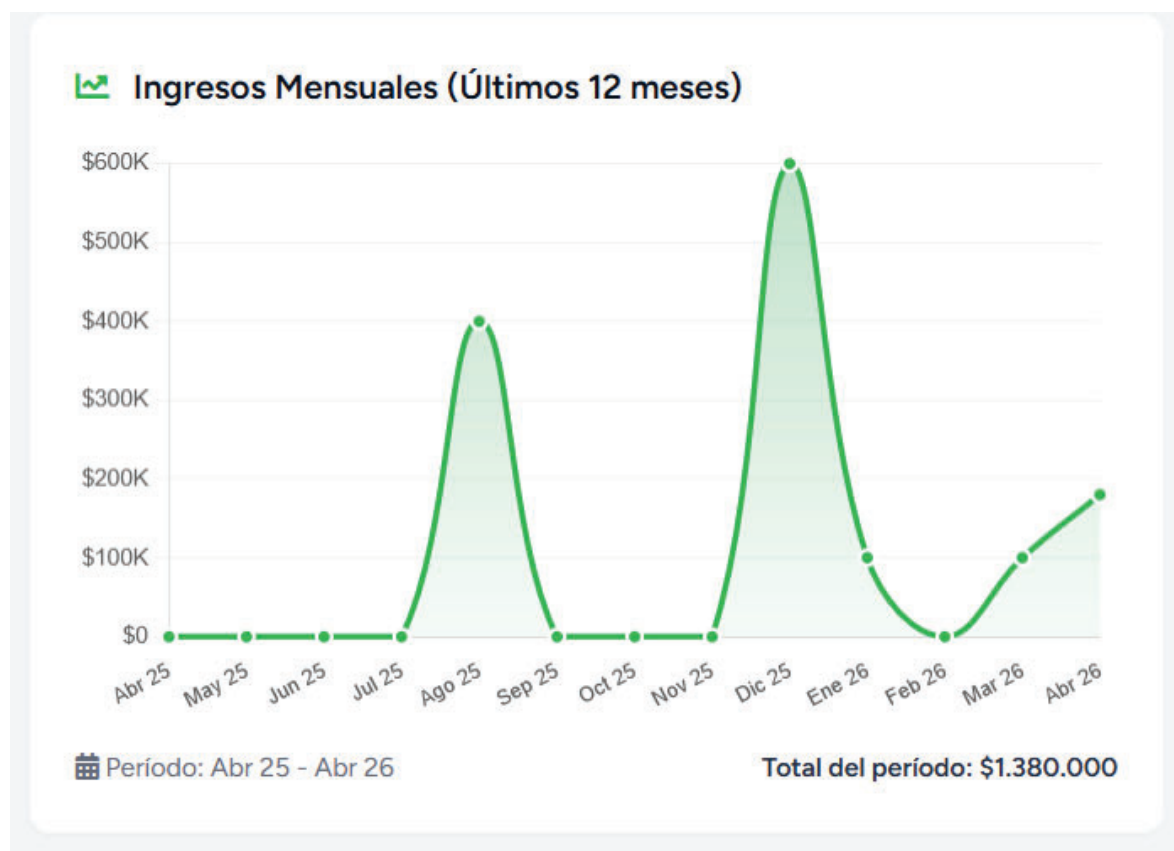


Imagen 8 captura de pantalla que muestra el gráfico de los ingresos mensuales

Este gráfico de línea presenta la evolución de ingresos durante los últimos doce meses, calculando: $\text{SUM}(\text{precio_plan} \times \text{períodos_pagados})$ agrupado por fecha de pago [Imagen 8]. Su relevancia incluye:

- **Análisis de Tendencias:** Permite identificar patrones estacionales y tendencias de crecimiento o decline.
- **Planificación Financiera:** Facilita la proyección de ingresos y la evaluación del desempeño anual.
- **Detección de Anomalías:** Identifica variaciones significativas que requieren investigación.

Gráfico de empresas por tipo

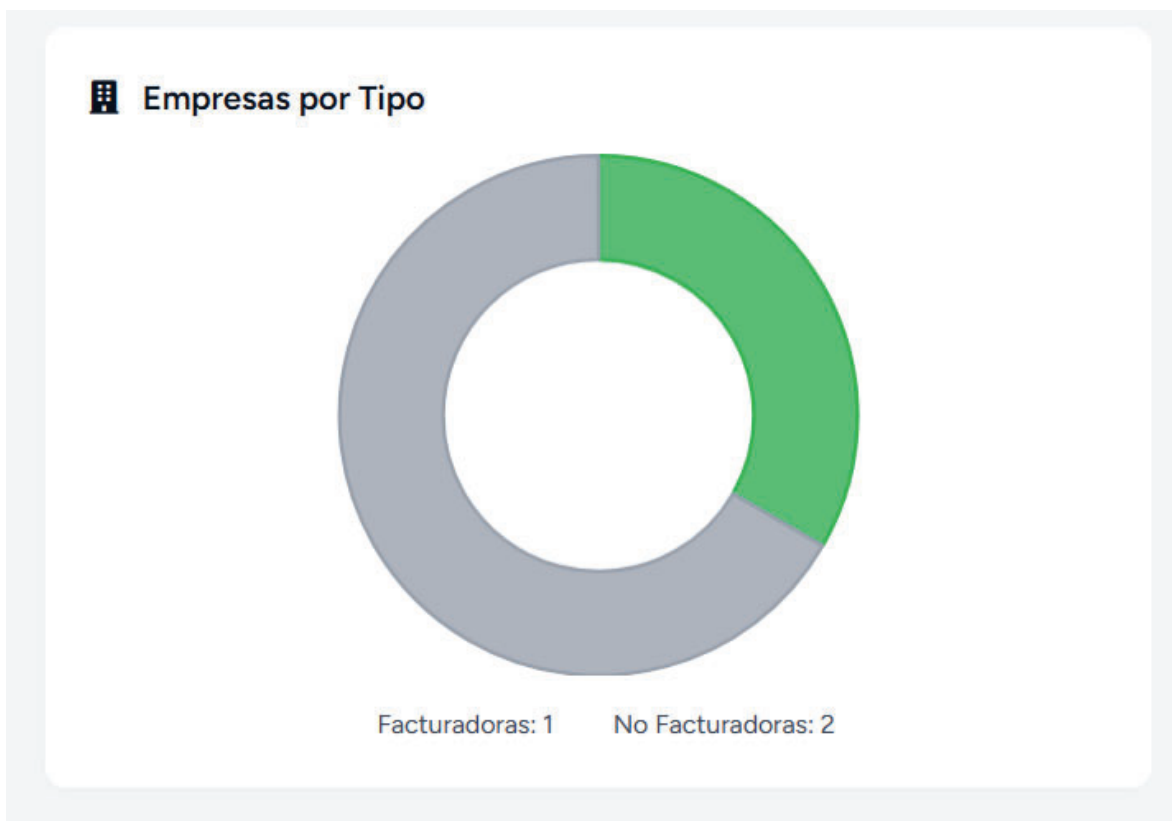


Imagen 9 captura de pantalla que muestra el gráfico de las empresas por tipo

Mediante un gráfico de dona, se visualiza la distribución de empresas clasificadas por su capacidad de facturación electrónica [Imagen 9]:

- Empresas Facturadoras: Aquellas que poseen token de integración NextPyme, indicando que están habilitadas para emitir facturas electrónicas.
- Empresas No Facturadoras: Las que no cuentan con integración, utilizando la plataforma solo para gestión.

Esta segmentación es crítica para:

- Análisis de Penetración: Evaluar qué porcentaje de clientes utiliza servicios de facturación avanzada.
- Estrategia de Desarrollo: Identificar oportunidades de upsell hacia capacidades de facturación electrónica.
- Compliance Normativo: Mantener registro de empresas con diferentes capacidades según regulación fiscal.

Gráfico de Consumo de Folios (Consumidos vs Recargados) [Imagen 10]



Imagen 10 Captura del Gráfico de consumo de Folios (Consumidos vs Recargados)

Este gráfico de barras agrupadas compara mensualmente:

- Folios Consumidos: Documentos procesados (facturas, nóminas, etc.) según registro histórico.
- Folios Recargados: Nuevos documentos asignados a través de renovación de planes o compra de paquetes adicionales.

Su importancia estratégica:

- Análisis de Utilización: Permite evaluar si los clientes consumen eficientemente sus asignaciones.
- Planificación de Capacidad: Identifica si el balance consumo-recarga es saludable o requiere intervención.
- Detección de Inactividad: Detecta empresas con muchos folios asignados pero bajo consumo (indicador de retención en riesgo).
- Oportunidad de Upsell: Identifica empresas que agotan rápidamente sus folios (candidatas a paquetes adicionales).

4.3.2 Módulo de Gestión de Empresas: Vista de Listado

El módulo de gestión de empresas constituye una sección integral del BackOffice de Sianexus que proporciona a los administradores un control centralizado sobre el portafolio completo de clientes y sus suscripciones asociadas. La vista de listado funciona como punto de partida para la administración de empresas, permitiendo visualizar, filtrar y acceder rápidamente a todas las operaciones relacionadas con cada cliente registrado en la plataforma.

La interfaz de listado presenta un conjunto robusto de funcionalidades de búsqueda y filtrado que facilitan la localización eficiente de empresas específicas. Los administradores pueden buscar por criterios múltiples incluyendo nombre de la empresa, número de identificación o dirección de correo electrónico, mientras que también disponen de filtros por estado de suscripción (activa, suspendida) y por planes específicos. Esta versatilidad en la consulta de datos es fundamental para gestionar grandes volúmenes de clientes sin requerir navegación manual exhaustiva. La información de cada empresa se presenta en una tabla estructurada que incluye datos críticos como tipo e identificación fiscal, planes activos con indicadores visuales de estado (verde para activo, amarillo para suspendido), vencimiento de certificados digitales con codificación por color según proximidad de expiración (rojo si vence en menos de dos días, amarillo en cinco días, azul en diez días), así como información de contacto (correo electrónico y teléfono).

Las capacidades operacionales integradas en esta vista permiten a los administradores ejecutar acciones clave sin necesidad de navegar a pantallas adicionales. Mediante opciones de acción contextual, es posible consultar detalles completos de la empresa, asignar o actualizar tokens de integración NextPyme para facturación electrónica, asignar documentos (folios) a empresas para que inicien o continúen su operación, consultar disponibilidad de documentos en tiempo real, y crear nuevas suscripciones a planes. La funcionalidad de sincronización de base de datos permite actualizar automáticamente información crítica desde sistemas externos, asegurando que los datos en la plataforma se mantengan vigentes. Adicionalmente, la vista soporta la visualización detallada mediante tooltips interactivos que muestran información adicional sobre cada plan (servicio asociado, plataforma, y fecha de inicio), mejorando la accesibilidad a información complementaria sin sobrecargar visualmente la tabla principal.

En conjunto, este módulo representa un punto de control esencial para la administración del ciclo de vida completo de los clientes dentro del ecosistema de Sianexus, desde su registro inicial hasta la gestión continua de sus servicios y recursos.

4.3.3 Módulos de Gestión de Catálogo de Productos: Servicios, Planes y Paquetes

Los módulos de gestión de servicios, planes y paquetes constituyen el núcleo del catálogo de productos ofrecido por Sianexus, proporcionando a los administradores las herramientas necesarias para definir, organizar y mantener la estructura comercial de la plataforma. Estos tres componentes operan de manera jerárquica e integrada, donde los servicios representan la capa más abstracta de oferta, los planes construyen propuestas de valor específicas sobre esos servicios, y los paquetes ofrecen complementos adicionales que amplían las capacidades de los planes contratados.

El módulo de Servicios constituye el nivel superior de la jerarquía de productos. Cada servicio representa una línea de negocio fundamental de Sianexus que está asociado a una plataforma tecnológica específica. Los administradores pueden crear nuevos servicios, editar sus descripciones y gestionar su relación con plataformas. Este módulo incluye filtrado por plataforma para localizar rápidamente servicios específicos, capacidades de ordenamiento alfabético, y protecciones contra la eliminación accidental de servicios que están siendo utilizados por planes activos, con mensajes de error específicos que informan al usuario sobre estas dependencias. La estructura del módulo de servicios permite mantener la coherencia del catálogo y facilita la expansión futura de la línea de productos.

El módulo de Planes operacionaliza los servicios al transformarlos en ofertas comerciales concretas. Los planes son propuestas de suscripción que combinan un servicio específico, un precio, una duración (expresada en meses) y un conjunto de restricciones operacionales (tales como número de usuarios, facturas, almacenes, etc.). Cada plan puede ser marcado como activo o inactivo, permitiendo a los administradores habilitar o deshabilitar ofertas sin necesidad de eliminarlas del sistema. El listado de planes presenta información consolidada incluyendo nombre, descripción, precio, duración normalizada, plataforma asociada y estado de actividad. Los administradores pueden filtrar por plataforma para visualizar únicamente los planes relacionados con una línea de negocio específica, ordenar por columnas, crear nuevos planes mediante un modal especializado, editar planes existentes, y eliminar planes que ya no sean ofrecidos. La integración con el dashboard permite a los administradores identificar visualmente cuáles son los planes más populares en términos de suscripciones activas.

El módulo de Paquetes proporciona un mecanismo de escalabilidad complementario que permite a las empresas adquirir folios (documentos) adicionales más allá de lo incluido en su plan base. Cada paquete define una cantidad específica de documentos disponibles para consumo en una plataforma determinada, con un precio establecido. Los paquetes se asignan a empresas de manera independiente de sus planes, permitiendo que una empresa con un plan base pueda comprar paquetes adicionales para ampliar su capacidad de procesamiento. El módulo de paquetes es más minimalista que los anteriores, enfocándose en la visualización

eficiente de la cantidad de folios disponibles, el precio, y la plataforma asociada. Las acciones disponibles permiten editar detalles de paquetes y eliminarlos cuando ya no sean relevantes para la oferta comercial.

La integración entre estos tres módulos crea un sistema de gestión de catálogo robusto y flexible. Los administradores pueden diseñar estrategias comerciales complejas definiendo múltiples servicios, creando planes con diferentes niveles de funcionalidad y precio sobre esos servicios, y ofreciendo paquetes suplementarios que permiten a los clientes personalizar su experiencia según sus necesidades específicas. La capacidad de filtrado y búsqueda en todos los módulos, combinada con la visualización clara de dependencias entre componentes, reduce significativamente el riesgo de inconsistencias en el catálogo de productos. En conjunto, estos módulos permiten que Sianexus mantenga una oferta flexible y escalable que puede adaptarse a diferentes segmentos de mercado y necesidades de negocio.

4.3.4 Suscripciones

Las suscripciones constituyen el núcleo del modelo de relación entre SianExus y sus clientes empresariales. Una suscripción representa la asociación entre una empresa cliente y un plan de servicios específico, estableciendo los términos bajo los cuales la empresa accede a las funcionalidades y recursos provistos por la plataforma. Cada suscripción contiene información crucial como la fecha de inicio, estado activo o suspendido, y referencias a renovaciones de pago que garantizan la continuidad del servicio.

El módulo de gestión de suscripciones proporciona una vista consolidada de todas las relaciones comerciales activas, permitiendo a los administradores monitorear el estado operacional de cada cliente. La interfaz presenta un listado completo de suscripciones con capacidades avanzadas de filtrado y búsqueda. Los administradores pueden filtrar por estado (Activo, Suspendido o Cancelado), buscar por nombre, identificación o correo de la empresa cliente, y filtrar por plataforma de origen del servicio. El ordenamiento dinámico permite clasificar la información por cliente, plan, plataforma o fecha de suscripción, facilitando la localización rápida de registros específicos.

Cada registro de suscripción muestra información crítica incluyendo el nombre de la empresa, el plan contratado, la fecha de suscripción inicial, la fecha de expiración del período actual, la fecha de cancelación si aplica, y el estado actual de la relación. La vista también integra acceso directo al historial de renovaciones (pagos registrados) y permite visualizar detalles completos de la suscripción en una vista dedicada. En esta vista de detalles, los administradores pueden hacer seguimiento de renovaciones pasadas, y pueden ejecutar operaciones como registrar nuevos pagos o cancelar la suscripción cuando sea necesario.

La arquitectura de suscripciones está diseñada para mantener la integridad referencial con los planes de servicios y la información de las empresas. El sistema automáticamente mantiene

el historial de cambios de estado y permite cancelaciones que registran la fecha de término del servicio sin eliminar el registro histórico. Esta aproximación garantiza que los administradores puedan auditar el ciclo de vida completo de cada relación comercial y acceder a información histórica para fines contables y operacionales.

4.3.4.1 Pagos y Renovaciones

El módulo de pagos y renovaciones gestiona la dimensión financiera de las suscripciones, registrando y rastreando todos los pagos realizados por los clientes empresariales para mantener activos sus servicios. Una renovación representa un evento de pago donde una empresa realiza el desembolso correspondiente para extender su período de acceso a los servicios contratados. Cada renovación contiene datos esenciales como el número de invoice para referencia contable, la cantidad de períodos (meses) pagados, la fecha de pago efectivo, y la fecha de expiración derivada del cálculo de períodos adicionales.

El sistema permite crear, editar y auditar renovaciones a través de un formulario modal integrado en la vista de detalles de suscripción. Al registrar un nuevo pago, los administradores especifican el número de períodos abonados (por ejemplo, 3 meses, 6 meses, 12 meses), el número de factura o comprobante del pago, y la fecha en que se efectuó el pago. Esta información es fundamental para el cálculo de ingresos, ya que permite determinar que un plan de precio 10.000 pagado por 3 períodos genera un ingreso total de 30.000. El sistema registra automáticamente la fecha de expiración calculada sumando los períodos al fecha de pago, garantizando que los controles de acceso respeten estos límites.

Las renovaciones se organizan en períodos individuales (RenewalPeriod) que rastrean el estado granular de cada mes o período pagado. Cada período puede estar en estado ACTIVO (actualmente vigente) o PENDIENTE (futuro, esperando su activación). Este desglose permite a los administradores visualizar exactamente qué períodos han sido consumidos y cuáles permanecen disponibles para los clientes. El historial completo de renovaciones se mantiene sin modificación destructiva—incluso cuando se edita un pago, el sistema preserva un registro auditable de cambios.

Integrado con el módulo de suscripciones, el sistema garantiza que las renovaciones solo se creen para suscripciones activas y válidas. El cálculo de ingresos del dashboard depende directamente de los datos de renovación, agregando el precio de cada plan multiplicado por la cantidad de períodos pagados en un mes específico. Esta arquitectura centraliza la lógica financiera, garantizando que métricas de ingresos, proyecciones de renovación y análisis de tendencias se basen siempre en datos consistentes y auditables.

4.3.5 Historial de Asignaciones de Documentos

El historial de asignaciones de documentos registra todas las operaciones de asignación de folios (documentos) a las empresas clientes, proporcionando una auditoría completa de cómo se distribuyen y consumen los recursos documentales en la plataforma. Cada asignación representa un evento donde un administrador asigna una cantidad específica de folios a una empresa para un plan determinado, especificando la fecha de asignación, el tipo de asignación (carga inicial, recarga, ajuste, etc.), y notas administrativas si aplica.

La vista del historial de asignaciones presenta una tabla paginada que consolida todas las operaciones de distribución de documentos. Para cada asignación, se visualiza el nombre de la empresa cliente, la cantidad de folios asignados, la fecha y hora exacta de la asignación, el tipo de operación realizada, y el usuario administrador responsable. Los registros aparecen ordenados cronológicamente de más reciente a más antiguo, permitiendo que los administradores tracen el flujo histórico de recursos documentales.

El sistema captura múltiples tipos de asignación para reflejar diferentes contextos operacionales. Una "carga inicial" ocurre cuando se activa una nueva suscripción y se provisiona la cantidad base de folios según el plan contratado. Las "recargas" se producen cuando los folios iniciales se agotan y se requiere distribuir recursos adicionales. Los "ajustes" cubren correcciones administrativas, devoluciones por errores, o compensaciones. Cada tipo de asignación se registra con la misma integridad, permitiendo auditorías y análisis de patrones de consumo.

La data de asignaciones es fundamental para entender la capacidad utilizada de la plataforma. El dashboard utiliza este historial para calcular cuántos folios se han asignado a empresas en períodos específicos, comparando contra el consumo real registrado en la tabla de historial de consumo (ConsumedHistory). Esta comparación permite a los administradores identificar empresas que no utilizan plenamente sus folios asignados, así como detectar situaciones donde el consumo se aproxima al límite disponible. El sistema mantiene un registro detallado de quién asignó qué cantidad en qué momento, cumpliendo con requisitos de auditoría y gobernanza de recursos.

4.4 Procesos Automáticos y Arquitectura de Eventos del BackOffice

4.4.1 Introducción a la Arquitectura Event-Driven

El backoffice de SianExus implementa una arquitectura orientada a eventos (Event-Driven Architecture) que permite la comunicación asíncrona e interoperable con otras plataformas del ecosistema SianExus. Esta arquitectura desacopla los procesos internos del backoffice de las operaciones en sistemas externos, garantizando una mayor escalabilidad, resiliencia y capacidad de respuesta del sistema completo. El núcleo de esta arquitectura reposa en un bus

de eventos distribuido implementado sobre RabbitMQ. Cuando ocurren eventos significativos en el backoffice como cambios en suscripciones, el sistema publica estos eventos a RabbitMQ con una routing key específica, donde otras plataformas se suscriben automáticamente a través de colas enlazadas y actúan en consecuencia procesando los eventos de forma asíncrona. De forma inversa, el backoffice consume eventos provenientes de otras plataformas mediante handlers especializados que actualizan el estado interno del sistema según reglas de negocio predefinidas, manteniendo la sincronización del ecosistema completo.

4.4.2 Flujo de Publicación y Consumo de Eventos

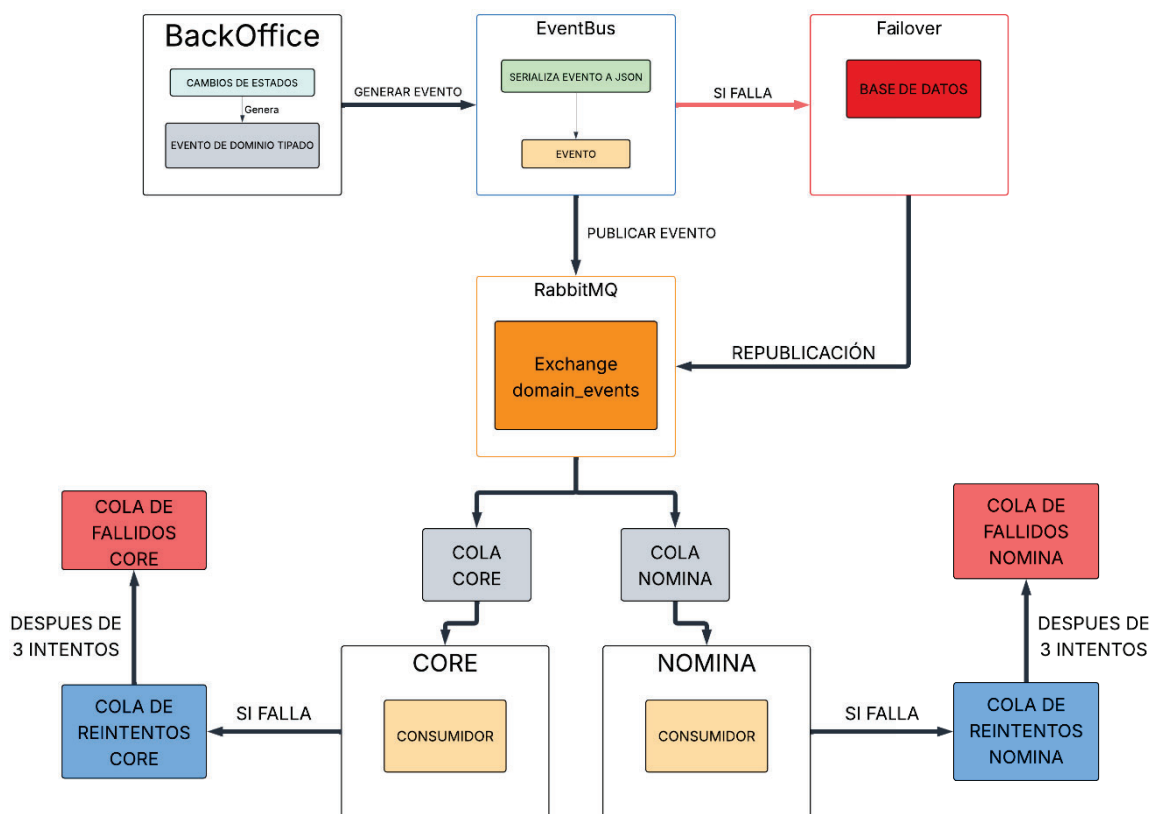


Imagen 11 Representación del flujo de publicación y consumo de eventos

El proceso de publicación de eventos en SianExus sigue un patrón consistente implementado en la clase `RabbitMqEventBus`. Cuando una operación resulta en un cambio de estado significativo (por ejemplo, cuando una empresa se suscribe a un nuevo plan de servicios), el código del módulo correspondiente invoca el `EventBus` para publicar uno o más eventos de dominio fuertemente tipados como se muestra en el inicio del flujo en la imagen 11. El `EventBus`, configurado para usar RabbitMQ como transporte primario, serializa el evento a JSON incluyendo metadatos como identificador único del evento, timestamp de ocurrencia y atributos de negocio específicos, y lo publica en el exchange de RabbitMQ denominado `domain_events` que se muestra en el medio de la imagen 11. Cada evento se

publica con una routing key que corresponde al nombre semántico del evento (por ejemplo: `subscription.created`, `folio.consumed`, `business.created`), permitiendo que el exchange de RabbitMQ distribuya automáticamente los eventos a todas las colas enlazadas que se han suscrito a esa routing key específica. Las plataformas consumidoras (CORE, Nómina) tienen colas propias enlazadas al exchange, recibiendo automáticamente los eventos pertinentes.

Esta arquitectura de publicación incorpora resiliencia mediante un "failover publisher". Si RabbitMQ no está disponible en el momento de la publicación—por ejemplo, debido a mantenimiento o problemas de conectividad de red—el EventBus captura la excepción AMQP y activa automáticamente el EloquentEventBus, que persiste el evento en la tabla `event_store` de PostgreSQL para procesamiento eventual. Cuando RabbitMQ se recupera, existe un proceso de re-publicación que identifica eventos pendientes en la base de datos y los reenvía al broker, garantizando que ningún evento se pierda y que la consistencia eventual se mantiene incluso ante fallos de infraestructura.

El consumo de eventos en el backoffice ocurre a través del comando `Artisan domain-events:rabbitmq:consume {queue} {quantity}`. Este comando se ejecuta de forma continua (típicamente bajo supervisión en ambientes de producción) para mantener la cola de eventos procediéndose en tiempo cercano al real. Cuando se invoca, establece conexión TCP/AMQP a RabbitMQ, se suscribe a la cola especificada, y procesa eventos secuencialmente: recibe un mensaje de la cola, deserializa el JSON para obtener el evento de dominio, consulta el `DomainEventMapping` para localizar el handler registrado que procesará ese tipo específico de evento, invoca el handler (un callable que implementa `__invoke()`), y confirma el procesamiento (ACK) a RabbitMQ cuando completa exitosamente. Si un handler lanza una excepción durante procesamiento, el evento se rechaza (NACK) y retorna a la cola para reintentos automáticos, respetando la política de reintentos configurada en `RABBITMQ_MAX_RETRIES` (por defecto 3 intentos).

4.4.3 Eventos Clave del Ecosistema

1. **SubscriptionCreatedEvent** es el evento más importante operacionalmente. Se publica cuando una empresa se suscribe a un nuevo plan de servicios a través del backoffice, o cuando se crea una suscripción automáticamente como parte del onboarding de una nueva empresa. El evento contiene información esencial: identificador único de la empresa (UUID), plan contratado (nombre único, ID, precio), plataforma origen del servicio (CORE o Nómina), estructura de restricciones y funcionalidades desbloqueadas por el plan, fecha de inicio y fecha de expiración del período pagado, y status de la suscripción (activa, suspendida, cancelada). Incluye adicionalmente metadatos sobre la estructura de navegación (sidebar) que debe mostrarse en la interfaz, identificador único de cualquier período anterior si es un recharge (renovación), y nombre del servicio contratado. Las plataformas

consumidoras escuchan este evento para aprovisionar automáticamente los recursos iniciales de la suscripción—como cantidad de folios documentales según los términos del plan contratado, activación de módulos premium, aplicación de límites de uso según la categoría del plan.

2. **FolioConsumedEvent** se genera cada vez que una empresa procesa un documento en una de las plataformas del ecosistema (CORE para facturación, Nómina para nóminas). El evento registra información crítica: UUID de la empresa, cantidad de folios consumidos en esa transacción, tipo específico de documento (factura, nota crédito, remisión, etc.), plataforma donde ocurrió el consumo, e información temporal (timestamp con zona horaria de América/Bogotá). Este evento es fundamental para mantener sincronizado el registro de disponibilidad de folios en el backoffice, ya que permite actualizar en tiempo cercano al real los contadores de consumo acumulado, identificar empresas que se aproximan a agotar su cuota, y triggerear recargas automáticas de folios cuando es necesario (integración con NextPyme).
3. **BusinessCreatedEvent** se publica cuando se registra una nueva empresa en cualquier plataforma del ecosistema SianExus. Contiene identificadores únicos de la empresa (UUID, NIT, razón social, tipo de identificación), integración con NextPyme (token de autenticación), configuración de responsabilidades fiscales (si es responsable de IVA, si practica retención en la fuente), software identificador interno, y datos de contacto (email, teléfono). El backoffice consume este evento de forma automática mediante el handler `BackofficeCreateBusinessEventHandler`, que ejecuta una secuencia crítica de operaciones diseñada para completar el onboarding de la empresa en el ecosistema financiero y fiscal del backoffice, orquestando: creación del registro de empresa en la base de datos del backoffice (o recuperación si ya existe por UUID), asignación automática del plan gratuito default ("Punto de Partida - Gratis") a la empresa, creación de la suscripción inicial con renovación y períodos de vigencia calculados, inicialización de registros de consumo de folios en cero, configuración de integración con NextPyme si el token está presente, carga de folios iniciales en NextPyme según el plan asignado, y finalmente publicación de un `SubscriptionCreatedEvent` de confirmación para cierre del loop de procesamiento entre plataformas.
4. **SendDocumentsCreatedEvent** se publica cuando se generan documentos (facturas, remisiones, nóminas, etc.) en las plataformas del ecosistema. El backoffice consume este evento mediante el handler `StoreActivationRequestsOnCreated`, almacenando un registro completo de solicitud de activación con toda la información del documento generado: identificador único de la empresa, nombre y email del usuario que generó el documento, mensaje descriptivo del contexto, booleano indicando si fue facturado electrónicamente y se envió a DIAN, números de factura y documento de soporte asignados, archivos adjuntos (PDF del documento, XML de facturación electrónica

si aplica), y estado final del documento. Este historial es crítico para auditoría, compliance regulatorio, y análisis de trazabilidad completa de operaciones.

4.4.4 Handlers Especializados: Procesamiento Automático

El procesamiento de eventos consumidos desde RabbitMQ ocurre a través de handlers especializados que implementan la interfaz `DomainEventSubscriber`. Cada handler se registra en el contenedor de inyección de dependencias de Laravel con la etiqueta `domain_event_subscriber` mediante el `TagServiceProvider`, permitiendo que el sistema descubra automáticamente todos los handlers sin configuración manual repetitiva.

1. **FolioConsumedEventHandler** es el handler más crítico operacionalmente, ejecutando transaccionalmente cuando llega un evento de consumo de folio. La secuencia de procesamiento es: localizar la empresa por su UUID en la base de datos del backoffice; si no existe, registrar error y abortar (aunque esto es excepción rara); obtener el registro de consumo acumulado `BusinessConsumption` de esa empresa; identificar la plataforma (CORE o Nómina) donde ocurrió el consumo; decrementar los folios disponibles en la plataforma específica, priorizando primero folios de plan (si existen) y luego folios de paquete (si existen); recalcular los totales agregados de disponibilidad; crear un registro histórico en `ConsumedHistory` con la empresa, plataforma, tipo de documento consumido, y timestamp exacto (para análisis retrospectivo y dashboards); invocar automáticamente el servicio `ReloadFoliosToNextpyme` para recalcular y sincronizar disponibilidades en NextPyme si la empresa tiene integración activa; registrar la operación en logs con niveles de auditoría (error, warning, info según corresponda). Crítico es que todas estas operaciones ocurren dentro de una transacción de base de datos mediante `DB::transaction()`, asegurando que si alguna operación intermedia falla (por ejemplo, NextPyme no responde), las actualizaciones locales se revierten completamente, manteniendo consistencia de estado. Si NextPyme falla, el error se registra en logs, pero no detiene la actualización local—la próxima vez que consuma folios se reintentará la sincronización.
2. **BackofficeCreateBusinessEventHandler** ejecuta cuando llega un evento de creación de empresa. Este handler es particularmente sofisticado porque detecta y previene duplicados (consulta si la empresa ya existe por UUID, evitando re-procesamiento), asigna automáticamente el plan gratuito (buscando por nombre único específico en la tabla de planes), verifica si la asignación ya existía (para evitar sobreescrituras), crea la suscripción con UUID, fecha de inicio en zona horaria Bogotá, cálculo automático de fecha de expiración según duración del plan, y flag `is_active` basado en si la expiración es hoy o después; crea la renovación inicial con número "AUTO-GENERATED" para indicar origen automático; calcula e inserta los

períodos de renovación individuales; inicializa contadores de consumo en cero; configura integración con usuarios y factores de generación para cálculos de capacidad; recarga folios iniciales en NextPyme si el token está presente (capturando excepciones sin fallar si NextPyme no responde); y publica un `SubscriptionCreatedEvent` de confirmación. Si alguna operación falla (por ejemplo, no encuentra el plan gratuito), se registra el error y se aborta el procesamiento, dejando la empresa en estado parcial que requiere intervención manual.

3. **StoreActivationRequestsOnCreated** es un handler más simple pero importante: consume `SendDocumentsCreatedEvent` e invoca el servicio `StoreActivationRequestsService` pasando toda la información del documento generado. El servicio persiste este registro en una tabla de auditoría para consultas posteriores, reportaría y compliance.

4.4.5 Configuración de RabbitMQ y Conectividad

La conexión a RabbitMQ se configura mediante variables de entorno almacenadas en él “.env”, permitiendo diferentes configuraciones por ambiente (desarrollo, testing, producción). Las variables clave son:

- `RABBITMQ_HOST`: Hostname o IP del servidor RabbitMQ (default: localhost)
- `RABBITMQ_PORT`: Puerto AMQP (default: 5672, puerto estándar de RabbitMQ)
- `RABBITMQ_LOGIN`: Usuario para autenticación (default: guest)
- `RABBITMQ_PASSWORD`: Contraseña asociada (default: guest)
- `RABBITMQ_EXCHANGE`: Nombre del exchange donde se publican todos los eventos (default: domain_events)
- `RABBITMQ_SIAN_VHOST`: Virtual host de RabbitMQ (default: /)
- `RABBITMQ_MAX_RETRIES`: Máximo número de reintentos cuando falla la publicación (default: 3)

Los timeouts para lectura/escritura se configuran internamente en la clase `RabbitMqConnection`: `read_timeout` de 2 segundos (tiempo máximo esperando datos del broker), `write_timeout` de 2 segundos (tiempo máximo para enviar datos), `connect_timeout` de 5 segundos (tiempo máximo para establecer conexión TCP). Estos valores balancean responsividad con tolerancia a latencia ocasional de red.

4.4.6 Integración Crítica con NextPyme

Un caso de uso crítico de la arquitectura de eventos es la sincronización en tiempo cercano al real de folios disponibles con NextPyme, el sistema de facturación electrónica integrado. El servicio ReloadFoliosToNextpyme (Recarga Folios en Nextpyme a las empresas) se invoca automáticamente en tres contextos: después de crear una nueva empresa (onboarding automático en BusinessCreatedEvent), después de consumir un folio (FolioConsumedEventHandler), y cuando se asignan nuevos folios manualmente (DocumentAssignmentHistoryList). La operación: valida que la empresa tenga un token NextPyme válido y no vacío; consulta los folios actualmente disponibles la empresa; invoca la API REST de NextPyme con credenciales y datos de disponibilidad; maneja errores de conectividad (timeout, conexión rechazada) registrando el error, pero sin detener otras operaciones, permitiendo que la próxima operación reintente. Esta resiliencia es crítica porque NextPyme está fuera del control directo del BackOffice y de SianExus.

4.4.7 Ventajas Arquitectónicas de la Implementación

Esta arquitectura event-driven proporciona ventajas sustanciales al ecosistema SianExus:

1. **Desacoplamiento:** El backoffice no necesita conocer detalles internos de cómo CORE o Nómina procesan eventos; publica eventos tipados y otras plataformas se suscriben. Si CORE cambia su implementación interna, el backoffice no se ve afectado.
2. **Escalabilidad Horizontal:** Pueden añadirse nuevas plataformas sin modificar el código del backoffice; simplemente se suscriben a las colas relevantes en RabbitMQ. El backoffice sigue publicando los mismos eventos sin cambios.
3. **Asincronía y Responsividad:** Las operaciones no bloquean mientras otras plataformas procesan. Cuando un usuario crea una suscripción en la interfaz, recibe confirmación inmediata; la publicación a RabbitMQ ocurre de fondo, y las plataformas consumidoras procesan cuando pueden. El usuario no espera a que CORE provisione folios.
4. **Auditoría y Trazabilidad:** El historial completo de eventos proporciona trazabilidad completa de cambios de estado en el ecosistema. Si hay disputa sobre cuándo ocurrió algo, el event store es la fuente de verdad.
5. **Recuperabilidad y Replay:** Si algo falla catastróficamente, se puede "reproducir" eventos desde el historial para recuperar estado. Por ejemplo, si el servidor de BusinessConsumption se corrompe, pueden re-procesarse todos los FolioConsumedEvent desde el historio para reconstruir exactamente el estado correcto.

6. **Testing Simplificado:** Para testing, pueden publicarse eventos manualmente a colas sin iniciar toda la orquestación visual, permitiendo testing de edge cases complejos.

4.5 Pruebas y Validación de la Plataforma

El proceso de pruebas y validación de la plataforma BackOffice se realizó de manera continua durante todo el ciclo de desarrollo, siguiendo el enfoque evolutivo-incremental adoptado para el proyecto. Debido a que el equipo de QA de SIAN EXUS participó activamente en cada etapa del desarrollo, las funcionalidades implementadas fueron evaluadas progresivamente conforme se integraban nuevos módulos y componentes al sistema.

A diferencia de un modelo tradicional donde las pruebas se ejecutan únicamente al final del proyecto, en este desarrollo las validaciones se realizaron de forma constante sobre cada incremento funcional entregado. Esto permitió identificar errores, inconsistencias y oportunidades de mejora en etapas tempranas del desarrollo, reduciendo el impacto de fallos críticos sobre la arquitectura general de la plataforma.

Las pruebas realizadas estuvieron orientadas principalmente al aseguramiento de calidad funcional del sistema, verificando que cada módulo cumpliera correctamente con los requerimientos definidos para la administración centralizada de empresas, planes, suscripciones y servicios tecnológicos dentro del ecosistema SIAN EXUS.

4.5.1 Participación del equipo QA

El equipo de QA tuvo un rol fundamental dentro del desarrollo del proyecto. Su participación no se limitó a validar versiones finales, sino que intervino continuamente durante la construcción de funcionalidades, revisando módulos parcialmente terminados, reportando incidencias y validando correcciones implementadas por el equipo de desarrollo.

Las incidencias detectadas eran documentadas y posteriormente integradas nuevamente al flujo de trabajo para su solución. Este proceso permitió mantener una retroalimentación constante entre desarrollo y QA, facilitando la mejora continua de la plataforma y garantizando una mayor estabilidad en cada versión desplegada.

Adicionalmente, las validaciones realizadas por QA permitieron asegurar la correcta integración entre módulos internos y la comunicación con servicios externos mediante RabbitMQ, verificando que los eventos emitidos y consumidos se procesaran adecuadamente sin pérdida de información.

4.5.2 Pruebas Funcionales

Las pruebas funcionales estuvieron enfocadas en verificar el correcto comportamiento de los diferentes módulos desarrollados dentro de la plataforma BackOffice. Estas validaciones permitieron comprobar que las funcionalidades implementadas respondieran correctamente a los requerimientos planteados inicialmente.

Entre las funcionalidades evaluadas se encuentran:

- Gestión de empresas clientes.
- Administración de roles y permisos.
- Creación y configuración de planes y paquetes.
- Gestión de suscripciones y renovaciones.
- Asignación y control de folios.
- Visualización de métricas e indicadores en el dashboard.
- Integración con la API externa de NextPyme.
- Comunicación asíncrona mediante RabbitMQ.

Cada funcionalidad fue sometida a diferentes escenarios de prueba, incluyendo validaciones de datos obligatorios, restricciones de acceso, manejo de errores y comportamiento esperado ante diferentes acciones realizadas por los administradores del sistema.

4.5.3 Pruebas de Integración

Debido a que la plataforma BackOffice debía integrarse con otros servicios del ecosistema SIAN EXUS, se realizaron pruebas de integración orientadas a validar la comunicación entre microservicios y la sincronización de información.

Estas pruebas permitieron verificar:

- El envío correcto de eventos hacia RabbitMQ.
- El consumo adecuado de eventos provenientes de otros servicios.
- La persistencia correcta de datos en la base de datos.
- La sincronización de estados de suscripciones y consumo de folios.
- La comunicación correcta con la API externa de NextPyme.

La validación de estos procesos fue importante debido a que gran parte de la operación del BackOffice depende de información compartida entre múltiples plataformas y servicios distribuidos.

4.5.4 Validación Operativa en Entorno Real

Una vez completadas las principales funcionalidades del sistema, la plataforma fue desplegada en un entorno operativo utilizado por administradores internos de SIAN EXUS. Este despliegue permitió realizar validaciones en condiciones reales de uso, verificando el comportamiento de la plataforma frente a operaciones administrativas cotidianas.

Durante esta etapa se evaluaron aspectos como:

- Estabilidad general del sistema.
- Fluidez de navegación en la interfaz administrativa.
- Correcta visualización de información en tiempo real.
- Comportamiento de los módulos bajo uso concurrente.
- Precisión de los datos reflejados en dashboards y reportes.

La retroalimentación obtenida durante esta fase permitió realizar ajustes finales orientados a mejorar la experiencia de uso y corregir detalles operativos detectados por los usuarios internos de la empresa.

4.5.5 Resultados de las Pruebas

Las pruebas realizadas permitieron validar el correcto funcionamiento general de la plataforma y comprobar que los objetivos planteados inicialmente fueron cumplidos satisfactoriamente. La integración continua del equipo QA dentro del flujo de desarrollo contribuyó significativamente a la detección temprana de errores y a la estabilidad de las versiones entregadas.

Gracias al enfoque evolutivo-incremental utilizado, fue posible implementar mejoras progresivas, mantener control sobre la calidad del software y garantizar que cada módulo integrado cumpliera adecuadamente con los requerimientos funcionales y operativos definidos para el proyecto.

5. Conclusiones

5.1 Conclusiones

El desarrollo de la plataforma BackOffice para SIAN EXUS resolvió el problema planteado al inicio del documento: la falta de un entorno centralizado para los servicios tecnológicos proporcionados por la empresa. Cada uno de los objetivos se cumplió, y los entregables técnicos construidos y la validación operativa fueron realizados por la administración interna de SIAN EXUS.

Para lograr el primer objetivo del diseño de la arquitectura del sistema, definimos una arquitectura distribuida basada en el patrón Modelo Vista-Controlador (MVC) en Laravel con PHP y la integramos con comunicación asíncrona a través de RabbitMQ. La arquitectura unificó la lógica de negocio, la gestión de datos y la interfaz gráfica hasta el punto de mantenimiento y escalabilidad del sistema y aseguró que el diseño se cumpliera y el desarrollo fuera aprobado por el equipo técnico de SIAN EXUS en un 70% de los diagramas y definiciones arquitectónicas.

Cada uno de estos módulos fue probado continuamente por el equipo de control de calidad de la empresa, y se obtuvieron más del 90% de las funcionalidades requeridas por el proyecto (al inicio).

Para el tercer objetivo, la integración con RabbitMQ se consolidó como una de las piezas técnicas más importantes del proyecto. Se implementaron intercambios, colas y claves de enrutamiento para la liberación y consumo de eventos clave en el ecosistema (creación de empresa, cambios de suscripción, consumo de folios, generación de documentos, etc.), y se implementó el mecanismo de failover para asegurar que cuando el broker de mensajería no esté disponible en el instante, el evento se almacene en la base de datos y se vuelva a publicar cuando la conexión regrese para que el mensaje se envíe con la misma información sin pérdida de información y la comunicación se complete.

Para el cuarto objetivo, se probaron la funcionalidad, integración y gestión del sistema con datos reales con una tasa de éxito superior al 90%. El despliegue en producción ayudó a realizar cambios que pueden mejorar la experiencia del usuario y la estabilidad general de la plataforma.

- El proyecto ha contribuido mucho al ecosistema de SIAN EXUS en términos de ingeniería de software en cuanto a la calidad de la arquitectura, ya que la arquitectura orientada a eventos a través de RabbitMQ es un mecanismo de comunicación asíncrona para asegurar que el BackOffice no dependa del resto del ecosistema ni de los otros servicios en él.

Primero, permite que cada aspecto del sistema escale de manera independiente y no afecte a los otros componentes del sistema. Segundo, permite que las nuevas plataformas se añadan sin cambiar el código del BackOffice del sistema, simplemente adjuntando las nuevas colas a los intercambios existentes en RabbitMQ. Tercero, hace que el sistema sea más resiliente ya que los manejadores de eventos reintentan automáticamente los datos en caso de una falla transitoria de red o procesamiento por motivos de seguridad.

Livewire integrado con Laravel Blade para el desarrollo de vistas nos permitió crear una interfaz administrativa reactiva y modular sin usar marcos de JavaScript separados, y la arquitectura del frontend es muy fácil y el repositorio es muy simple de esta manera. El proyecto fue el adecuado porque la plataforma se trata de gestión administrativa interna y no requiere interacción del usuario como una aplicación de usuario final.

- Antes del desarrollo de esta plataforma, SIAN EXUS no tenía un sistema centralizado para gestionar las relaciones con sus empresas clientes, y por esa razón, era manual, propenso a errores y difícil de escalar. Para mover planes, control de acceso, consumo de folios, suscripciones, y folios, etc., SIAN EXUS se encargaba de todo por sí mismo y no tenía una vista única de la situación de cada cliente.

Con el BackOffice, SIAN EXUS tiene un centro de control de gestión completo desde el cual sus administradores pueden hacer todas estas cosas a la vez y ser transparentes con la información.

La automatización de la incorporación de nuevas empresas en el evento creado de negocio y su manejador y evento de negocio creado (BusinessCreatedEvent) eliminó la necesidad de configurar manualmente nuevos clientes.

Los módulos de gestión de catálogos—servicios, planes y paquetes—dieron a la empresa una estructura comercial más flexible y jerárquica que le permite definir múltiples niveles de ofertas desde servicios básicos hasta complementos adicionales a través de paquetes de folios. Esta flexibilidad es necesaria para que SIAN EXUS adapte su propuesta de valor a diferentes segmentos de mercado, desde microempresas hasta grandes organizaciones, sin necesidad de cambiar la arquitectura del sistema.

La codificación visual de estados críticos en la interfaz, como la expiración de certificados digitales con diferentes colores según la proximidad de la expiración, y los indicadores de estado de suscripción por empresa, permite a los administradores ver qué tipo de problemas necesitan atención para responder rápida y eficientemente y así ayudar a la empresa a servir mejor a los clientes.

- La capacidad de proporcionar versiones funcionales gradualmente permitió probar los módulos con QA y comenzar a trabajar en ellos (para evitar errores en las etapas más avanzadas) y la metodología híbrida de gestión de actividades (Scrum, Kanban y DevOps) fue un ajuste natural para que la empresa pueda avanzar sin provocar ningún cambio en la dinámica que ya tenía el equipo.

Las reuniones diarias permitieron visibilidad sobre el progreso del proyecto, identificaron obstáculos en el camino e hicieron que los problemas críticos en el flujo de la versión fueran una prioridad.

El equipo de QA estuvo involucrado en el proceso de desarrollo y no solo en la fase final de pruebas para entregar el producto y pudo identificar la calidad del producto. La detección temprana de errores, las pruebas continuas de funcionalidad y la retroalimentación continua entre desarrollo y calidad aseguraron que cada paso del desarrollo alcanzara el nivel de estabilidad para que la empresa lo desplegara.

5.2 Recomendaciones para Futuros Trabajos

Basado en las limitaciones del proyecto y la experiencia en el desarrollo del sistema, se recomienda las siguientes mejoras para mejorar la plataforma SIAN EXUS Backoffice:

- Crear una pasarela de pago que automatice la renovación de la suscripción (por ejemplo, sin necesidad de registrar manualmente los pagos) y reduzca el tiempo entre la expiración del plan y la renovación.
- Crear un portal de autogestión para las empresas clientes donde puedan rastrear su consumo de folios, estado de suscripción, historial de emisión de documentos y alertas de disponibilidad para hacerlo transparente y mejor para los clientes.
- Automatizar el proceso de re-publicación de los eventos pendientes en el almacén de eventos en trabajos programados y enviarlos de nuevo a RabbitMQ cuando el broker esté listo para funcionar nuevamente.
- Aplicar pruebas unitarias e integradas automatizadas en la tubería CI/CD para cada nueva versión de modo que se valide automáticamente y no tenga regresiones.
- Considerar migrar la interfaz de BackOffice a un modelo moderno de JavaScript como Vue.js o React en caso de que la plataforma se vuelva más compleja y más amigable para el usuario que Livewire.
- Asegurar que el sistema administrativo tenga un sistema de auditoría más robusto que registre lo que hicieron los administradores en la plataforma, como creación, modificación y eliminación de registros para asegurar la gobernanza y el cumplimiento de la ley.

Referencias

- [1] LÓPEZ, Daniel, et al. *Arquitectura de Software basada en Microservicios para Desarrollo de Aplicaciones Web*. 2017
- [2] Inesdi, “Arquitectura de software: 5 patrones principales,” [En línea]. Disponible en: <https://www.inesdi.com/blog/arquitectura-de-software-5-patrones-principales/>.
- [3] JOYANES, Luis. *Sistemas de información en la empresa*. Alpha Editorial, 2015.
- [4] HONG, Xian Jun; YANG, Hyun Sik; KIM, Young Han. Performance analysis of RESTful API and RabbitMQ for microservice web application. En *2018 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 2018. p. 257-259.
- [5] MORENO MORLÓN, Sabina. *Sistema para la gestión de una PYME basado en microservicios*. 2023.
- [6] Dirección de Impuestos y Aduanas Nacionales (DIAN), *Abecé Factura Electrónica - Información para el facturador*, Bogotá, Colombia, 2023. [En línea]. Disponible en: https://www.fondoadaptacion.gov.co/images/2024/manual-de-pagos/manual_de_pagos.pdf
- [7] M. J. Ríos-Colmenárez and M. G. Rodríguez, "Comunicación asincrónica: Experiencias en ambientes virtuales de pre y postgrado," *Revista Proyección*, no. Edición Especial, pp. 40–49, Dec. 2017.
- [8] L. von Bertalanffy, *General System Theory: Foundations, Development, Applications*. New York, NY, USA: George Braziller, 1968.
- [9] K. C. Laudon and J. P. Laudon, *Management Information Systems*, 16th ed. Hoboken, NJ, USA: Pearson, 2020.
- [10] C. E. Shannon and W. Weaver, *The Mathematical Theory of Communication*. Urbana, IL, USA: Univ. Illinois Press, 1949.
- [11] M. Castells, *Communication Power*. Oxford, U.K.: Oxford Univ. Press, 2009.
- [12] M. Cusumano, A. Gawer, and D. Yoffie, “The business of platforms: Strategy in the age of digital competition, innovation, and power,” *Communications of the ACM*, vol. 63, no. 1, pp. 26–28, Jan. 2020. doi: 10.1145/3371837
- [13] A. Holovaty and J. Kaplan-Moss, *The Definitive Guide to Django: Web Development Done Right*, 1st ed. Berkeley, CA: Apress, 2007.
- [14] D. Thomas, D. Hansson, L. Hyett, and M. Ford, *Agile Web Development with Rails* 6, 6th ed. Dallas, TX: Pragmatic Bookshelf, 2020.
- [15] Zuora, Inc., “Zuora Architects the Future of Enterprise Software with the Launch of Zuora Central,” *Zuora Press Release*, [En línea]. Disponible en: <https://www.zuora.com/press-release/zuora-architects-the-future-of-enterprise-software-with-the-launch-of-zuora-central/>. [Accedido: 7 jun. 2025].

[16] xFusion, “Back-Office Support – xFusion’s Seamless Behind-The-Scenes Operations,” *xFusion*, [En línea]. Disponible en: <https://xfusion.io/customer-experience/back-office-support/>. [Accedido: 7 jun. 2025].

[17] Google, “Use the Admin console to manage your organization,” *Google Workspace Admin Help*, 2024. [Online]. Available: <https://support.google.com/a/answer/182076>

[18] Zoho Corporation, “Zoho One Admin Panel,” *Zoho One User Guide*, 2023. [Online]. Available: <https://www.zoho.com>

[19] E. Bascón Pantoja, “El patrón de diseño Modelo-Vista-Controlador (MVC) y su implementación en Java Swing,” *Acta Nova*, vol. 2, no. 4, pp. 493–507, 2004.

[20] Minciencias, *Política de actores del Sistema Nacional de Ciencia, Tecnología e Innovación (SNCTeI)*, Resolución No. 1473, República de Colombia, 2016. [En línea]. Disponible en: https://minciencias.gov.co/sites/default/files/ckeditor_files/politiciadeactores-snctei.pdf (consultado el 13 de junio de 2025).

[21] D. Howcroft and H. Richardson, “The back office goes global: exploring connections and contradictions in shared service centres,” *Work, Employment and Society*, vol. 26, no. 1, pp. 111–127, 2012. [Online]. Available: <https://doi.org/10.1177/0950017011426309>

[22] Universidad de los Andes, *Metodologías para el desarrollo de software*, Bogotá: Departamento de Ingeniería de Sistemas y Computación, 2015

[23] E. Gil, J. M. López, L. Peralta, M. Infante y C. Menéndez, “Metodologías de desarrollo de software: análisis y evolución,” *Revista Educación en Ingeniería*, no. 26, pp. 72-81, 2018

[24] K. Schwaber y J. Sutherland, *The Scrum Guide*, Scrum.org, 2020. [En línea]. Disponible en: <https://scrumguides.org>

[25] D. Anderson, *Kanban: Successful Evolutionary Change for Your Technology Business*, Blue Hole Press, 2010.

[26] G. Kim, J. Humble, P. Debois y J. Willis, *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*, IT Revolution Press, 2016.

[27] Universidad Estatal de Milagro (UNEMI), “Modelo de desarrollo de software,” *SGA UNEMI*, [En línea]. Disponible en: https://sga.unemi.edu.ec/media/recursotema/Documento_202169165012.pdf. [Accedido: 12-jun-2025].

[28] Representación visual del modelo vista controlador obtenida de <https://codingornot.com/wp-content/uploads/2017/10/mvc-modelo-vista-controlador.png>