




Secretaría de Finanzas del poder Ejecutivo del Estado

Manual de Administración de la Plataforma

No. de Contrato:	Acrónimo del Proyecto:	Nombre del Proyecto:
SF/DA/019-LPE/2024	AUDITORÍA	Servicios profesionales para el desarrollo e implementación de un sistema para el control y seguimiento de auditoría a impuestos estatales conforme al programa operativo de fiscalización

La información contenida en este documento es para uso interno

	SECRETARÍA DE FINANZAS DEL PODER EJECUTIVO DEL ESTADO
	Coordinación Técnica de Ingresos y Recaudación
	Manual de Administración de la Plataforma
	Asistencia y Soluciones de Proyectos VANNY S.A. de C.V.
No. de Contrato: SF/DA/019-LPE/2024	

Contenido

1. Introducción.....	1
1.1. Objetivo	1
1.2. Datos generales	1
1.3. Definiciones, Abreviaciones y Referencias	2
2. Descripción general de la solución.....	3
2.1. Representación gráfica del proceso del negocio	3
2.2. Representación gráfica de la arquitectura	3
2.3. Justificación para la Arquitectura Seleccionada	4
3. Requisitos de la arquitectura	6
4. Diseño de la Arquitectura	6
4.1. Vista de componentes	6
4.2. Vista de datos	7
5. Patrones de Diseño	7
6. Tecnologías usadas.....	8
6.1. Back.....	8
6.2. Front.....	8
6.3. Extras	9
7. Despliegue.....	9
7.1. Elementos Clave del Diagrama	10
8. Firmas	11



VANNY 	SECRETARÍA DE FINANZAS DEL PODER EJECUTIVO DEL ESTADO
	Coordinación Técnica de Ingresos y Recaudación
	Manual de Administración de la Plataforma
	Asistencia y Soluciones de Proyectos VANNY S.A. de C.V.
No. de Contrato: SF/DA/019-LPE/2024	

Tabla de Versiones y Modificaciones

Versión	Descripción del cambio	Responsable de la Versión	Fecha
0.01	Creación del documento.		
0.02	Actualización		

	SECRETARÍA DE FINANZAS DEL PODER EJECUTIVO DEL ESTADO
	Coordinación Técnica de Ingresos y Recaudación
	Manual de Administración de la Plataforma
	Asistencia y Soluciones de Proyectos VANNY S.A. de C.V.
No. de Contrato: SF/DA/019-LPE/2024	

1. Introducción

1.1. Objetivo

El propósito fundamental de este documento es ofrecer una comprensión profunda y holística de la arquitectura subyacente en el sistema propuesto. A través de una variedad de vistas arquitectónicas, se busca detallar y analizar los diversos componentes, interacciones y decisiones fundamentales que dan forma a la solución tecnológica en cuestión. El enfoque principal radica en la captura y transmisión de las elecciones arquitectónicas clave que han sido adoptadas para establecer las bases del sistema.

La Arquitectura de la Solución Tecnológica desempeña un papel central en la documentación integral del proyecto. Al definir la estructura y el diseño del sistema desde una perspectiva técnica, esta arquitectura establece el camino a seguir para la implementación y el desarrollo coherente de la solución. Cada vista arquitectónica proporciona una lente específica a través de la cual se pueden observar los aspectos esenciales del sistema, incluidas sus capacidades funcionales, su rendimiento, su seguridad y su capacidad de adaptación.

Esta sección también tiene como objetivo presentar una visión general de la estructura del documento en su conjunto. A medida que los lectores avanzan, encontrarán una serie de secciones dedicadas a explorar las distintas vistas arquitectónicas que conforman la solución tecnológica. Estas secciones se han diseñado para ofrecer claridad y coherencia en la presentación de la información, permitiendo a los interesados comprender de manera integral tanto los detalles técnicos como las consideraciones estratégicas que han guiado el diseño del sistema.


La arquitectura propuesta se basa en una estructura modular y escalable que permita la adición de nuevas funcionalidades conforme a las necesidades de la Secretaría. Cada módulo estará interconectado a través de servicios web estandarizados que garantizarán la consistencia de la información y la interoperabilidad con otras plataformas institucionales.

1.2. Datos generales

La Secretaría de Finanzas del Poder Ejecutivo del Estado de Oaxaca requiere un servicio especializado para el desarrollo e implementación de un sistema digital que permita el control y seguimiento de auditorías a impuestos estatales.

El proyecto contempla el desarrollo de un sistema que incluya los siguientes módulos y funcionalidades:

- **Módulo de Registro del Contribuyente:** Desarrollo de interfaces para capturar y administrar la información de los contribuyentes de manera automatizada.
- **Módulo de Autorización del Comité de Programación:** Implementación de funcionalidades para gestionar autorizaciones de forma digital.
- **Módulo del Área Operativa:** Desarrollo de herramientas para monitorear el flujo de auditorías y control operativo.
- **Módulo de Control y Seguimiento:** Creación de dashboards y reportes para el seguimiento del estado de las auditorías.


	SECRETARÍA DE FINANZAS DEL PODER EJECUTIVO DEL ESTADO
	Coordinación Técnica de Ingresos y Recaudación
	Manual de Administración de la Plataforma
	Asistencia y Soluciones de Proyectos VANNY S.A. de C.V.
No. de Contrato: SF/DA/019-LPE/2024	

- **Módulo de Procedimiento a Revisión:** Implementación de flujos de trabajo para el proceso de revisión y validación de auditorías.
- **Módulo de Dictámenes:** Desarrollo de plantillas y generación de dictámenes automáticos.
- **Módulo de Consultas y Reportes:** Creación de funcionalidades para realizar consultas de datos históricos y generar reportes personalizados.
- **Módulo de Administración de Usuarios:** Implementación de un sistema de gestión de permisos y roles para los diferentes usuarios del sistema.
- **Módulo de Administradores:** Desarrollo de paneles de control para la administración del sistema y gestión de configuraciones avanzadas.

El desarrollo de estos módulos deberá alinearse con los estándares tecnológicos establecidos por la Dirección General de Tecnologías e Innovación Digital y considerar la integración con el marco normativo aplicable.

1.3. Definiciones, Abreviaciones y Referencias

Términos/Siglas	Descripción
CONAC	Consejo Nacional de Armonización Contable
SAT	Servicio de Administración Tributaria
Gateway	Punto de entrada o salida entre componentes
Token	Objeto digital para autenticar la identidad de un usuario
OAuth2	Marco de autorización que permite que aplicaciones de terceros accedan a los recursos de un usuario sin exponer sus credenciales
AES-256	Algoritmo de cifrado simétrico que se utiliza para proteger datos confidenciales
APIs REST	Estilo arquitectónico utilizado para crear API que permiten la comunicación entre aplicaciones cliente y servidor
Kubernetes	Plataforma de orquestación de contenedores de código abierto que automatiza la implementación, el escalamiento y la gestión de aplicaciones en contenedores.
Docker Swarm	Herramienta nativa de agrupamiento y orquestación para Docker que permite la gestión y la implementación de aplicaciones en contenedores en un clúster de nodos Docker
branch	Línea independiente de desarrollo que permite trabajar en versiones paralelas de un proyecto sin afectar la rama principal
HTTP	Es el protocolo de comunicación utilizado en la World Wide Web para transferir datos entre un navegador web (cliente) y un servidor

	SECRETARÍA DE FINANZAS DEL PODER EJECUTIVO DEL ESTADO
	Coordinación Técnica de Ingresos y Recaudación
	Manual de Administración de la Plataforma
	Asistencia y Soluciones de Proyectos VANNY S.A. de C.V.
No. de Contrato: SF/DA/019-LPE/2024	

2. Descripción general de la solución

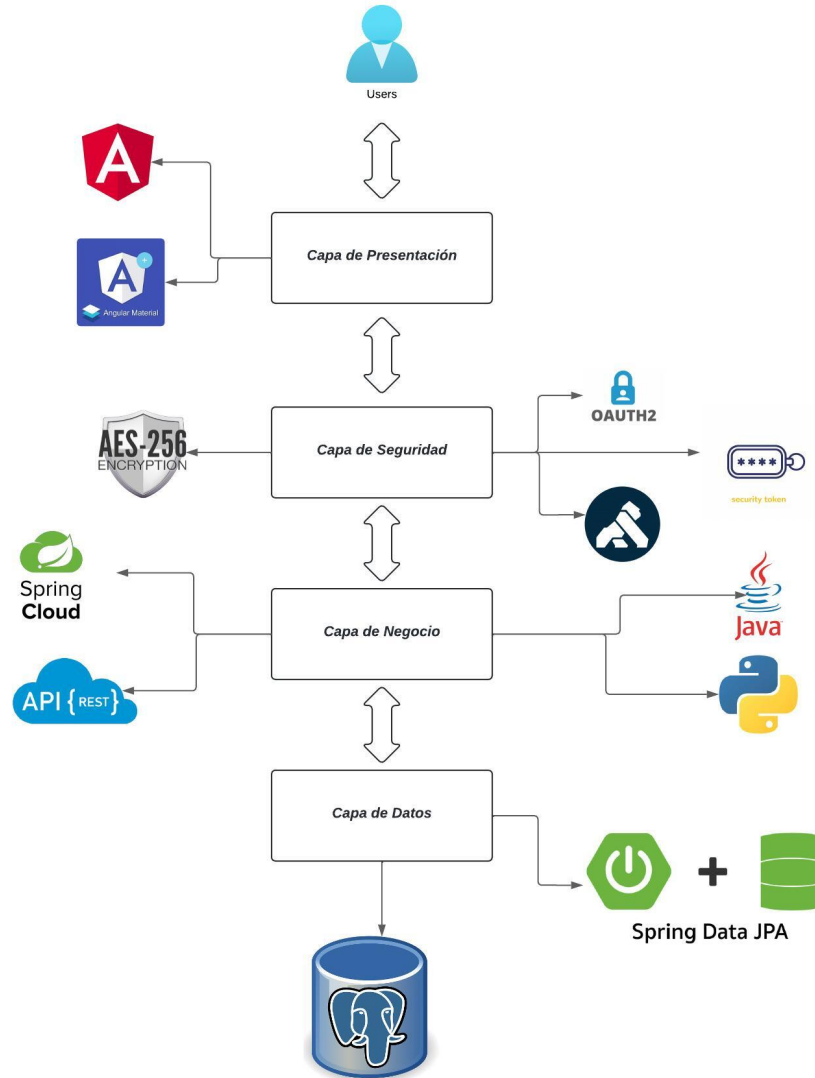
2.1. Representación gráfica del proceso del negocio

2.2. Representación gráfica de la arquitectura

En el siguiente diagrama se proporciona una descripción detallada de la arquitectura funcional de la solución, organizada en capas claramente definidas. Este enfoque permite una estructura modular y desacoplada que facilita la gestión, la escalabilidad y la seguridad del sistema. A continuación, se describe de manera concisa el propósito y los principales componentes tecnológicos presentes en cada capa de la arquitectura:

- **Capa de Presentación:** Esta capa está destinada a gestionar la interacción con el usuario final. En este caso, se implementa utilizando el framework Angular, complementado con Angular Material para el diseño de interfaces de usuario modernas y receptivas. La capa de presentación recibe las solicitudes de los usuarios y las envía a las capas subyacentes, mostrando los resultados de manera intuitiva y visualmente atractiva.
- **Capa de Seguridad:** Esta capa se encarga de todos los aspectos relacionados con la protección de la información y la gestión de accesos. Se utilizan mecanismos de autenticación como OAuth2 para validar las identidades de los usuarios y asegurar el acceso autorizado a las funcionalidades y datos del sistema. Además, se aplica encriptación AES-256 para garantizar la integridad y confidencialidad de los datos sensibles, asegurando un control de acceso robusto y evitando accesos no autorizados.
- **Capa de Negocio:** Aquí reside la lógica principal del sistema, implementada con tecnologías como Java y Python para asegurar un procesamiento eficiente y flexible. Esta capa actúa como el corazón de la aplicación, gestionando el flujo de trabajo y las reglas de negocio que determinan el comportamiento de la aplicación. Además, se integra con microservicios mediante Spring Cloud y expone APIs REST para facilitar la comunicación entre diferentes componentes del sistema y con aplicaciones externas.
- **Capa de Datos:** Esta capa es responsable de la persistencia y gestión de la información. Se apoya en tecnologías como PostgreSQL para el manejo de bases de datos relacionales y utiliza Spring Data JPA para interactuar con los repositorios de datos de manera eficiente. Aquí se realizan operaciones CRUD (Crear, Leer, Actualizar y Eliminar) y se asegura de que la información se almacene y recupere de manera coherente y segura.


Cada una de estas capas está conectada de manera fluida para asegurar que la comunicación entre ellas sea eficiente y segura, proporcionando una arquitectura sólida y adaptable a los requisitos cambiantes del negocio. La implementación de esta estructura multicapa garantiza que la solución pueda evolucionar y escalar conforme a las necesidades del sistema y de los usuarios finales



2.3. Justificación para la Arquitectura Seleccionada

La arquitectura de microservicios facilita la escalabilidad al descomponer una aplicación monolítica en componentes independientes y autónomos (los microservicios). Cada microservicio puede ser escalado individualmente según sea necesario, lo que permite adaptarse dinámicamente a los cambios en la demanda de recursos y mantener un rendimiento óptimo incluso bajo cargas variables. Esto proporciona una mayor flexibilidad y eficiencia en la gestión de recursos, así como la capacidad de escalar de manera más granular y específica según las necesidades del sistema.

La adopción de una arquitectura orientada a microservicios proporciona una serie de beneficios clave que son fundamentales para el desarrollo de sistemas modernos y escalables. A continuación, se justifica la elección de esta arquitectura, teniendo en cuenta los siguientes aspectos:

	SECRETARÍA DE FINANZAS DEL PODER EJECUTIVO DEL ESTADO
	Coordinación Técnica de Ingresos y Recaudación
	Manual de Administración de la Plataforma
	Asistencia y Soluciones de Proyectos VANNY S.A. de C.V.
No. de Contrato: SF/DA/019-LPE/2024	

- **Modularidad y Reusabilidad:** Esto promueve la reusabilidad de componentes y la posibilidad de integrar nuevas funcionalidades o tecnologías sin afectar el resto del sistema. Así, la infraestructura manejada por el cliente puede evolucionar sin comprometer la arquitectura global. Al dividir la aplicación en microservicios, cada uno puede ser desarrollado, desplegado y mantenido de forma independiente, lo que facilita la reutilización de código y la integración con otros sistemas.
- **Escalabilidad:** La escalabilidad en el contexto de la arquitectura orientada a microservicios se refiere a la capacidad del sistema para manejar un aumento en la carga de trabajo de manera eficiente y sin comprometer el rendimiento. En un entorno de microservicios, la escalabilidad se puede lograr de dos formas principales:
 - **Escalabilidad horizontal:** Se refiere a la capacidad de agregar más instancias de un microservicio para distribuir la carga entre ellas. En lugar de aumentar los recursos de una única instancia, se agregan más instancias del mismo microservicio para manejar la demanda creciente. Esto se logra fácilmente mediante la replicación de microservicios en múltiples contenedores o máquinas virtuales, y puede ser gestionado mediante herramientas de orquestación de contenedores como Kubernetes o Docker Swarm.
 - **Escalabilidad vertical:** Se refiere a la capacidad de aumentar los recursos (como CPU, memoria, etc.) de una instancia de microservicio para manejar una mayor carga. Sin embargo, la escalabilidad vertical tiene límites físicos y puede ser más difícil de implementar y gestionar que la escalabilidad horizontal. Además, puede haber un punto en el que aumentar los recursos no ofrezca mejoras significativas en el rendimiento.
- **Mantenibilidad:** Permite realizar modificaciones y actualizaciones de manera más eficiente. Cada microservicio autónomo y puede ser actualizada sin afectar otras partes del sistema, lo que reduce el riesgo de errores y la necesidad de pruebas exhaustivas en todo el sistema. Los microservicios también facilitan la implementación de prácticas de desarrollo ágil, ya que permiten la entrega continua y la rápida iteración de funcionalidades.
- **Experiencia del Usuario:** Al colocar la capa de presentación en el nivel superior, se prioriza la experiencia del usuario. Esta capa se encarga de la interfaz de usuario y la interacción, por lo que es crucial para proporcionar una experiencia fluida y atractiva para el cliente. Los microservicios permiten desarrollar interfaces de usuario altamente interactivas y receptivas, lo que mejora significativamente la experiencia del usuario final.

3. Requisitos de la arquitectura

Servidor	SO	RAM	DD	Procesadores
Ecosistema de ms y front	Linux	64Gb	1TB	16
BD	Linux	64GB	1TB	16

4. Diseño de la Arquitectura

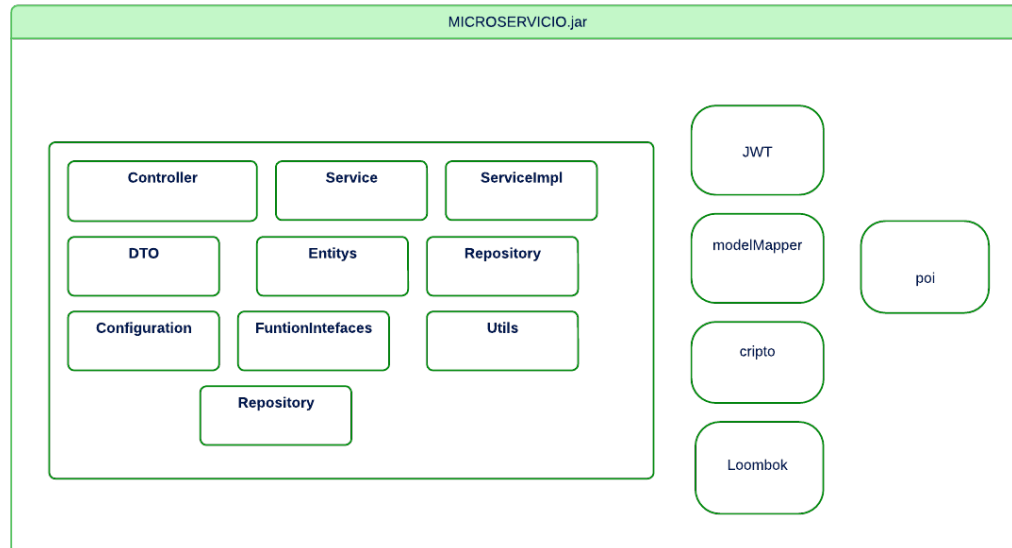
4.1. Vista de componentes

El propósito principal de este diagrama de componentes radica en la provisión de una panorámica de gran amplitud que desentrañe la estructura y las interacciones intrínsecas de los componentes en el sistema. Mediante esta representación visual, se busca facultar a los desarrolladores y arquitectos con una comprensión global y contextualizada de la arquitectura general del software, así como de las interacciones nodales entre los elementos que lo conforman.

De especial importancia es destacar que este diagrama no se sumerge en los pormenores de las implementaciones particulares ni en las especificidades de las tecnologías a emplear. En su lugar, funge como un punto de partida integral que establece los cimientos para un análisis más profundo y detallado. Esta visualización abarca la disposición relacional de los componentes y ofrece una guía inicial para los diálogos y exploraciones futuras, donde se abordarán con minuciosidad los aspectos técnicos concernientes a la implementación, las tecnologías concretas a emplear y las estratificaciones de programación.

Como resultado, este diagrama se erige como un instrumento de comunicación y orientación, permitiendo que los equipos de desarrollo y arquitectura se alineen respecto a la visión general de la estructura del sistema. Además, funge como un catalizador de intercambios sustantivos, proporcionando el contexto necesario para que los detalles técnicos sean debatidos y planificados de manera precisa y coherente. En resumen, este diagrama de componentes desempeña un rol fundamental en la etapa inicial de la planificación y diseño, sentando las bases para la materialización exitosa del sistema en desarrollo.

Por lo tanto, su aplicación trasciende el mero aspecto visual y se erige como un recurso clave en la facilitación de una comprensión compartida, la identificación de direcciones estratégicas y el fomento de un enfoque colaborativo en la creación del software deseado.




4.2. Vista de datos

El presente apartado será abordado en una segunda fase, dado que en el actual momento de la ejecución del proyecto, carecemos aún del nivel de detalle necesario en cuanto a los casos de uso para proceder con la determinación del esquema de la base de datos.

5. Patrones de Diseño

En el proceso de diseño y desarrollo de la intranet, es de suma importancia considerar la aplicación estratégica de patrones de diseño. Estos patrones, que encapsulan soluciones probadas a desafíos específicos, juegan un papel crucial en la creación de una arquitectura que sea robusta, adaptable y sostenible a largo plazo. A continuación, se exponen los patrones de diseño seleccionados y su pertinencia en este contexto particular:

- **Singleton (Singleton):** Es un patrón de diseño creacional que garantiza que una clase tenga solo una instancia y proporciona un punto de acceso global a esa instancia. Es útil cuando solo se necesita una única instancia de una clase en toda la aplicación, como por ejemplo para manejar la conexión a una base de datos o para mantener configuraciones globales.
- **Factory Method (Método de Fábrica):** Es un patrón de diseño creacional que proporciona una interfaz para crear objetos en una superclase, pero permite a las subclases alterar el tipo de objetos que se crean. Esto permite delegar la creación de objetos a las subclases, lo que hace que el código sea más flexible y extensible.
- **Builder (Constructor):** Es un patrón de diseño creacional que se utiliza para construir objetos complejos paso a paso. Permite crear diferentes tipos y representaciones de un objeto utilizando el mismo proceso de construcción. Se utiliza cuando la creación de un objeto requiere muchos pasos o cuando el proceso de construcción es complicado y necesitas encapsularlo.
- **Iterator (Iterador):** Es un patrón de diseño comportamental que proporciona una forma

	SECRETARÍA DE FINANZAS DEL PODER EJECUTIVO DEL ESTADO
	Coordinación Técnica de Ingresos y Recaudación
	Manual de Administración de la Plataforma
	Asistencia y Soluciones de Proyectos VANNY S.A. de C.V.
No. de Contrato: SF/DA/019-LPE/2024	

de acceder secuencialmente a los elementos de una colección sin exponer su representación subyacente. Esto permite recorrer una colección de elementos de manera uniforme sin preocuparse por la estructura interna de la colección.

- **State (Estado):** Es un patrón de diseño comportamental que permite a un objeto cambiar su comportamiento cuando su estado interno cambia. Este patrón se basa en la composición en lugar de la herencia, lo que significa que un objeto puede cambiar de clase a la que está asociado en tiempo de ejecución. Esto hace que sea más flexible y extensible que simplemente cambiar el comportamiento a través de la herencia.

6. Tecnologías usadas


6.1. Back

- Java 11 y java 12
- **Maven:** Es una potente herramienta de gestión de proyectos que se utiliza para gestión de dependencias, como herramienta de compilación e incluso como herramienta de documentación.
- **Swagger:** Es herramientas que nos ayudan a documentar nuestras APIs. De esta manera, podemos realizar documentación que sea realmente útil para las personas que la necesitan. Swagger nos ayuda a crear documentación que todo el mundo entienda.
- **Spring Boot:** Spring Boot es una plataforma de Spring que está orientado al desarrollo de microservicios totalmente, provee de muchas herramientas que hacen el desarrollo más rápido, además existe un amplio soporte en la red.
- **Oauth2:** Es un estándar abierto para la autorización de APIs, que nos permite compartir información entre sitios sin tener que compartir la identidad. Es un mecanismo utilizado hoy en día por grandes compañías como Google, Facebook, Microsoft, Twitter, GitHub o LinkedIn, entre otras muchas.
- **Kong Api:** Es una plataforma de gestión de API de código abierto que actúa como un intermediario entre el cliente y los servicios de backend. Diseñada para facilitar la implementación, seguridad y supervisión de APIs, Kong se utiliza para enrutar, transformar, proteger y gestionar solicitudes HTTP de manera eficiente. Su infraestructura basada en microservicios permite la integración de plugins para agregar funcionalidades como autenticación, control de acceso, balanceo de carga y análisis en tiempo real.
- **Spring Security:** Es un marco de autenticación y control de acceso potente y altamente personalizable. Es el estándar de facto para proteger las aplicaciones basadas en Spring. Spring Security es un marco que se enfoca en proporcionar autenticación y autorización a las aplicaciones Java. Como todos los proyectos de Spring, el verdadero poder de Spring Security se encuentra en la facilidad con la que se puede ampliar para cumplir con los requisitos personalizados.

6.2. Front

- **Angular 18:** Framework de desarrollo de aplicaciones web de código abierto, mantenido principalmente por Google, que permite crear aplicaciones dinámicas y escalables. Está basado en TypeScript y sigue una arquitectura modular que facilita la creación de aplicaciones a través de componentes reutilizables y servicios

Confidencial	Página 8
--------------	----------

	SECRETARÍA DE FINANZAS DEL PODER EJECUTIVO DEL ESTADO
	Coordinación Técnica de Ingresos y Recaudación
	Manual de Administración de la Plataforma
	Asistencia y Soluciones de Proyectos VANNY S.A. de C.V.
No. de Contrato: SF/DA/019-LPE/2024	

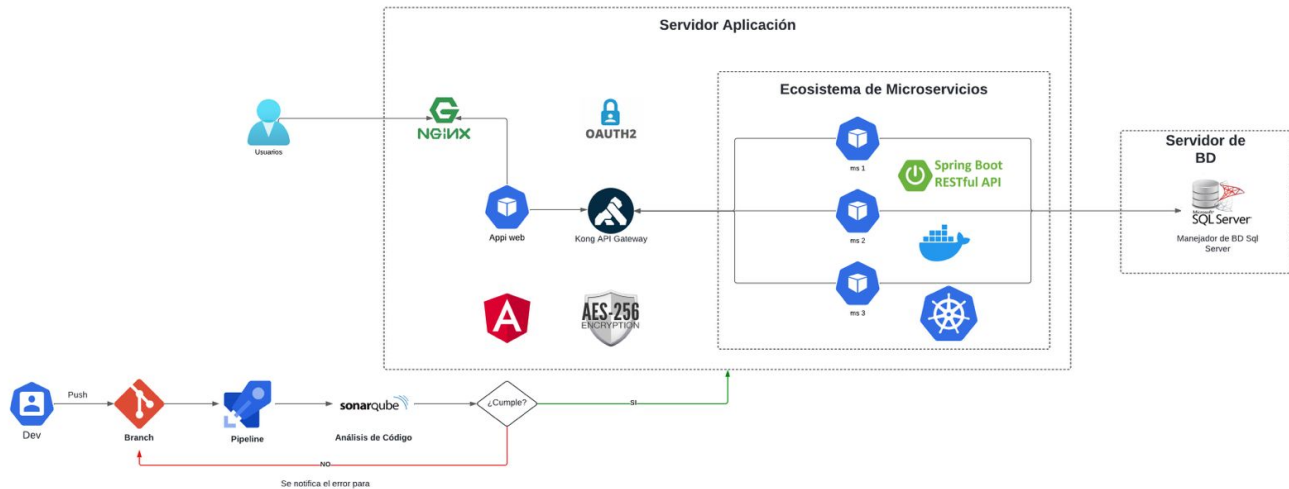
- **Cripto-js:** Es una biblioteca de cifrado de JavaScript que ofrece una amplia variedad de funciones criptográficas. Con Crypto.js, puedes cifrar y descifrar mensajes usando diferentes algoritmos de cifrado como AES, DES, Triple DES, RC4, Rabbit y muchos más.
- **Material Angular:** Es una biblioteca de componentes de interfaz de usuario que implementa el diseño basado en Material Design de Google para aplicaciones construidas con Angular. Ofrece una amplia colección de elementos reutilizables que facilitan la creación de aplicaciones web con una apariencia profesional y con un estilo coherente.
- **Axios:** Es una librería JavaScript que se utiliza para realizar peticiones HTTP desde aplicaciones basadas en navegadores y Node.js. Esta librería simplifica la comunicación entre una aplicación y un servidor al proporcionar una interfaz fácil de usar para enviar y recibir datos a través del protocolo HTTP.

6.3. Extras

- **Nginx:** Es un servidor web que también actúa como proxy de correo electrónico, proxy inverso y balanceador de carga. La estructura del software es asíncrona y controlada por eventos; lo cual permite el procesamiento de muchas solicitudes al mismo tiempo. NGINX también es altamente escalable, lo que significa que sus servicios aumentan a la par con el tráfico de sus clientes. NGINX y Apache son, de hecho, dos de los mejores servidores web del mercado.
- **Docker:** Es una plataforma de software que le permite crear, probar e implementar aplicaciones rápidamente. Docker empaqueta software en unidades estandarizadas llamadas contenedores que incluyen todo lo necesario para que el software se ejecute, incluidas bibliotecas, herramientas de sistema, código y tiempo de ejecución. Con Docker, puede implementar y ajustar la escala de aplicaciones rápidamente en cualquier entorno con la certeza de saber que su código se ejecutará.
- **Docker Compose:** Es una herramienta para definir y ejecutar aplicaciones de Docker de varios contenedores. En Compose, se usa un archivo YAML para configurar los servicios de la aplicación. Después, con un solo comando, se crean y se inician todos los servicios de la configuración.
- **Prácticas de DevOps:** Son un reflejo de la idea de automatización y mejora continuas, y muchas de ellas se centran en una o en varias fases del ciclo de desarrollo.

7. Despliegue

El diagrama muestra una vista de alto nivel de cómo los diferentes componentes del sistema interactúan con los recursos de hardware en un entorno de implementación real. La visualización facilita la comprensión de cómo los elementos del sistema están distribuidos y se relacionan entre sí. Este enfoque es útil para identificar las asignaciones de componentes a nodos de hardware y para delinear las interconexiones clave entre los servicios.




7.1. Elementos Clave del Diagrama

• Usuarios y Entrada al Sistema:

- El flujo comienza con los usuarios, quienes acceden a la aplicación a través de un *frontend web*.
- La solicitud del usuario se enruta a través de un *servidor NGINX*, que actúa como un servidor de balanceo de carga y de gestión de tráfico, dirigiendo las peticiones al ecosistema de microservicios.

• Servidor de Aplicación y Ecosistema de Microservicios:


- El sistema está compuesto por un conjunto de microservicios desplegados en un ecosistema basado en contenedores (indicados por los íconos de Kubernetes y Docker).
- Cada microservicio tiene funciones específicas y están contruidos con Spring Boot para la implementación de APIs RESTful.
- El ecosistema de microservicios está gestionado mediante Kong API Gateway, que actúa como un punto de entrada unificado para las peticiones y facilita la

	SECRETARÍA DE FINANZAS DEL PODER EJECUTIVO DEL ESTADO
	Coordinación Técnica de Ingresos y Recaudación
	Manual de Administración de la Plataforma
	Asistencia y Soluciones de Proyectos VANNY S.A. de C.V.
No. de Contrato: SF/DA/019-LPE/2024	

administración de las API, como la autorización (OAuth2) y la seguridad (AES-256 Encryption).

- Gestión de Base de Datos:**
 - Todos los microservicios, según sea necesario, acceden a un servidor de base de datos independiente que utiliza PostgreSQL como gestor de base de datos.
 - La separación de la base de datos en un nodo distinto garantiza una arquitectura desacoplada y facilita la escalabilidad y la administración independiente de la persistencia de datos.
- Integración Continua y Pipeline de Desarrollo:**
 - El flujo de desarrollo e integración comienza con un desarrollador (Dev) realizando cambios en una rama (branch) del repositorio de código.
 - Cada nuevo cambio se envía a través de un pipeline de CI/CD (Integración y Entrega Continua), donde SonarQube realiza un análisis estático de código para verificar la calidad del mismo.
 - Si el análisis es exitoso, se procede con la compilación y el despliegue de los componentes. En caso de errores, se notifica al desarrollador para realizar correcciones y volver a enviar el código.
- Flujo de Comunicación y Seguridad:**
 - Las conexiones entre los componentes están indicadas mediante líneas que representan la comunicación entre nodos de hardware y componentes de software.
 - El uso de OAuth2 y AES-256 Encryption garantiza que las interacciones estén autenticadas y que los datos estén encriptados, asegurando la integridad y confidencialidad de la información intercambiada.

8. Firmas

Solicitante / Rol / Puesto / Organización	Estatus	Fecha	Firma
C. Javier Jiménez Aquino Apoderado Legal Asistencia y Soluciones de Proyectos VANNY, S.A. de C.V.	Entrega	25/12/2024	
L.C.P. Grimaldo Santiago López Director de Auditoría e Inspección Fiscal Secretaría de Finanzas del Poder Ejecutivo del Estado	Recibe	25/12/2024	