Web API para análisis de datos

Gabriel Romero Luna[1], Yuliana Casanova López [2], Carlos David Aquino Reyes[3]
Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias de la Computación
Servicio Social
gabriel.romerolu@alumno.buap.mx, yuliana.casanovalo@alumno.buap.mx,
carlos.aquinore@alumno.buap.mx

Resumen. El proyecto consistió en el diseño e implementación de una arquitectura de software orientada a ofrecer una Web API que facilite el análisis de datos fiscales. El sistema está dirigido principalmente a contribuyentes que requieran analizar información de sus CFDI emitidos, ya sea en intervalos de tiempo definidos o mediante filtros específicos, con el fin de optimizar la interpretación y aprovechamiento de su información fiscal.

Palabras clave: Web API, CFDI, microservicios, frameworks, desarrollo web, análisis de datos, webhook, machine learning, arquitectura de software, grPC.

1 Introducción

1.1 Objetivo general

Diseño e implementación de una arquitectura de software orientada a proveer una Web API útil en el proceso de análisis de datos de tipo fiscal.

1.2 Alcance

Se obtendrá como resultado el modelo o diagrama de una arquitectura de software que cumpla con el objetivo general, así como una versión beta de la implementación de dicho modelo, acotada a las siguientes operaciones:

- De visualización (agregación y desagregación)
- De conjuntos (unión, intersección, etc.)
- Medidas de tendencia central (promedio, mediana y moda)
- Estadísticas básicas (rango, varianza, desviación estándar, coeficiente de variación)

1.3 Objetivos específicos

- Diseñar un modelo arquitectónico de software que permita la integración y análisis de datos fiscales provenientes de CFDI.
- Implementar una versión beta de la Web API que cumpla con las funcionalidades establecidas.
- Desarrollar módulos para la visualización de datos mediante procesos de agregación y desagregación.
- Incorporar operaciones de conjuntos (unión, intersección, etc.) para el tratamiento de datos CFDI.
- Implementar cálculos de medidas de tendencia central (promedio, mediana y moda) y estadísticas básicas (rango, varianza, desviación estándar y coeficiente de variación).
- Optimizar la consulta y filtrado de información bajo criterios definidos por el usuario.

2 Estado del arte

2.1 Antecedentes

Antes de comenzar con el desarrollo del proyecto, el equipo ya contaba con conocimientos prácticos en el uso de distintos frameworks, tecnologías, diagramas UML y bases de datos aplicados al desarrollo de aplicaciones y servicios web, adquiridos en cursos previos. Entre ellos, se tenía experiencia en frameworks como Flask, Slim y FastAPI, así como en el manejo de Python como lenguaje principal de

programación. Estas experiencias iniciales permitieron comprender cómo exponer información mediante endpoints y cómo estructurar proyectos orientados a servicios, sentando así una base sólida para la construcción de la solución propuesta.
No obstante, el proyecto requería un enfoque más especializado. Para ello fue necesario investigar a fondo la estructura de los CFDI, comprendiendo que estos documentos fiscales se encuentran en formato XML, lo cual los ubica dentro de la categoría de datos semiestructurados. El análisis detallado permitió identificar que cada CFDI contiene nodos jerárquicos bien definidos, tales como emisor, receptor, conceptos, impuestos y complementos, que debían ser procesados para su análisis posterior.

De igual manera, se estudió con mayor profundidad el modelo de comunicación REST, entendiendo sus principios arquitectónicos (identificación de recursos mediante URLs, uso de métodos HTTP como GET, POST, PUT y DELETE, y transferencia de representaciones en JSON o XML). Este conocimiento fue esencial para diseñar una API que pudiera operar bajo estándares de interoperabilidad y escalabilidad.

Finalmente, se exploraron distintos frameworks modernos y se seleccionó FastAPI por sus ventajas frente a otros entornos: alto rendimiento, validación automática de datos mediante Pydantic, soporte para programación asíncrona y generación automática de documentación. Aunque no formaba parte del conocimiento previo del equipo, su adopción fue estratégica para cumplir con los objetivos del proyecto.

2.2 Panorama actual

Actualmente, en el mercado existen diversas aplicaciones que gestionan documentos CFDI, como Factura.com, Facturama, Aspel y Contpaqi, entre otras. Estas soluciones suelen enfocarse principalmente en el timbrado, almacenamiento y descarga de comprobantes, así como en la generación de reportes básicos para la administración contable. Si bien cumplen con las necesidades de emisión y resguardo de facturas electrónicas, presentan limitaciones en cuanto al análisis estadístico, la visualización avanzada de la información y la integración con procesos de analítica de datos.

En contraste, la solución propuesta en este proyecto busca ir más allá de la simple gestión documental, al implementar una arquitectura de software basada en una Web API que habilite el análisis de datos fiscales. Entre las funcionalidades contempladas en el prototipo destacan:

- Operaciones de visualización: agregación y desagregación de información.
- Operaciones de conjuntos: unión, intersección y diferencia de datos.
- Medidas estadísticas: promedio, mediana, moda, rango, varianza, desviación estándar y coeficiente de variación.
- Mecanismos de filtrado dinámico: consultas por fechas, rangos o categorías.
- Soporte para exportación en formatos como JSON, XML, CSV y Excel.
- Autenticación segura mediante tokens JWT y claves de API, así como autorización basada en roles (RBAC).
- Capacidades de integración avanzada, como modelos de machine learning.

De esta forma, el avance principal de la propuesta consiste en ofrecer una herramienta orientada al análisis que complemente y supere a las aplicaciones existentes en cuanto a flexibilidad, profundidad y aprovechamiento de la información contenida en los CFDI.

3 Marco teórico

3.1 Datos semiestructurados (XML)

Los datos semiestructurados son aquellos que no siguen un modelo rígido como el relacional, pero poseen elementos de organización que facilitan su interpretación mediante etiquetas, claves y jerarquías. Según Pal (2016), este tipo de datos surge como una alternativa flexible para representar información que cambia con frecuencia y que no se ajusta fácilmente a esquemas estáticos [1].

En particular, el formato XML (eXtensible Markup Language) ha sido ampliamente adoptado por organismos internacionales como estándar para el intercambio de información entre sistemas diversos. XML permite definir esquemas personalizados mediante el uso de DTD o XSD, lo que refuerza la integridad y consistencia de los datos, y facilita su interoperabilidad. Además, las bases de datos nativas en XML proporcionan mecanismos útiles como XPath, XQuery y XSLT, que permiten explorar, transformar y presentar la información de forma eficiente. Esto lo convierte en una excelente opción para almacenar documentos complejos con múltiples niveles jerárquicos (CFDI), sin tener que fragmentar o normalizar la información en múltiples tablas, tal como ocurre en los enfoques tradicionales basados en bases de datos relacionales [2].

Estructura de un documento XML

Los documentos XML representan datos organizados como nodos en un árbol jerárquico, donde cada elemento puede tener subelementos (hijos) o texto.

- Declaración XML: Todo documento XML debe comenzar con una etiqueta indicando versión y codificación del archivo.
- Elemento raíz: El documento debe contener un único elemento raíz que engloba todos los demás.
- Elementos e hijos: Etiquetas anidadas que representan secciones de datos o la estructura del documento.
- Atributos: Pares nombre-valor dentro de una etiqueta, que añaden información descriptiva.
- Contenido: Texto o datos dentro de un elemento.
- Comentarios: Notas no procesadas [3].

Control estructural: DTD y XML Schema (XSD)

- DTD (Document Type Definition): Define los elementos permitidos, su orden, anidación y atributos mediante una sintaxis específica.
- XML Schema (XSD): Estándar del W3C que permite definir tipos de datos, cardinalidad, espacios de nombres y restricciones detalladas, ofreciendo mayor control que DTD [3].

Impacto en el desarrollo

En el proyecto, comprender la naturaleza de los datos semiestructurados fue esencial, ya que los CFDI (Comprobantes Fiscales Digitales por Internet) están codificados en XML. Cada CFDI incluye nodos específicos como:

- <cfdi:Emisor> → datos de la empresa emisora.
- <cfdi:Receptor> → información del receptor.
- <cfdi:Conceptos> → lista de bienes o servicios.
- <cfdi:Impuestos> → desglose de retenciones y traslados.

El manejo correcto de estos nodos y atributos permite realizar operaciones de análisis como agregación, filtrado y cálculo de estadísticas básicas. Así, la elección de trabajar con XML no fue opcional, sino una condición técnica impuesta por la normativa fiscal mexicana, y dominarlo fue indispensable para la implementación de la Web API propuesta.

3.2 Web APIs (diseño e implementación)

Una Web API (Web Application Programming Interface) es una interfaz que permite que dos aplicaciones se comuniquen e intercambien datos a través de protocolos y estándares web, generalmente usando HTTP como medio de transporte. Estas interfaces exponen un conjunto de operaciones o recursos que pueden ser consumidos por otros sistemas, facilitando la interoperabilidad y la integración entre aplicaciones heterogéneas, ya sea en una misma organización o entre diferentes entornos.

Las Web APIs pueden implementarse siguiendo diversos estilos arquitectónicos, siendo REST (Representational State Transfer) uno de los más populares debido a su

simplicidad, escalabilidad y compatibilidad con los principios de la web. En este modelo, los recursos se identifican mediante URLs únicas, se manipulan a través de métodos HTTP (GET, POST, PUT, DELETE) y se intercambian representaciones en formatos como JSON o XML [4].

Api REST

El modelo más común en la construcción de Web APIs es el estilo arquitectónico REST, proporcionan una forma ligera de crear API web y se utilizan comúnmente para facilitar el intercambio de datos entre aplicaciones, servicios web y bases de datos, y para conectar componentes en arquitecturas de microservicios. REST se basa en principios que permiten que los sistemas sean escalables, simples y estandarizados, entre los cuales destacan:

- Identificación de recursos: cada recurso (usuarios, facturas, CFDI, etc.) se identifica mediante una URI única.
- Operaciones uniformes: los recursos se manipulan siempre de la misma manera usando métodos estándar de HTTP.
- Interfaz uniforme: facilita la interoperabilidad al mantener un diseño consistente y predecible.
- Statelessness: cada petición al servidor debe contener toda la información necesaria, sin depender de estados previos [5].

Verbos HTTP

Las API REST se comunican a través de solicitudes HTTP para realizar funciones estándar de bases de datos, como crear, leer, actualizar y eliminar registros (también conocido como CRUD) dentro de un recurso. Los encabezados y parámetros de solicitud también son importantes en las llamadas a la API REST, porque incluyen información importante del identificador, como metadatos, autorizaciones, identificadores de recursos uniformes (URI), almacenamiento en caché, cookies y más. Los encabezados de solicitud y los encabezados de respuesta, junto con los códigos de estado HTTP convencionales, se utilizan dentro de las API REST bien diseñadas.

- GET: Consultar un recurso o colección de recursos (ejemplo: obtener todos los CFDI de un cliente).
- POST: Crear un nuevo recurso (ejemplo: registrar un nuevo usuario en el sistema).
- PUT: Actualizar un recurso existente (ejemplo: modificar los datos de un CFDI).
- DELETE: Eliminar un recurso (ejemplo: borrar un comprobante almacenado).
- PATCH: Actualizar parcialmente un recurso (ejemplo: cambiar solo un atributo del CFDI) [5].

Uso de JSON y XML

En cuanto al formato de representación de datos, las Web APIs REST suelen trabajar con:

- JSON (JavaScript Object Notation): formato ligero, basado en pares clave-valor, fácil de procesar por aplicaciones web y móviles. Su sintaxis sencilla lo ha convertido en el estándar más usado en la actualidad.
- XML (Extensible Markup Language): aunque más verboso, sigue siendo ampliamente utilizado en entornos empresariales y en sistemas fiscales, ya que permite representar jerarquías complejas y es compatible con validación mediante esquemas [5].

Impacto en el desarrollo

El proyecto se fundamenta en la construcción de una Web API siguiendo los principios REST, con el framework FastAPI. El sistema define recursos como CFDI, usuarios y consultas, los cuales se manipulan a través de endpoints bien estructurados que utilizan métodos HTTP estándar. La API entrega sus respuestas en JSON, por su

ligereza y fácil consumo, pero también mantiene compatibilidad con XML, asegurando coherencia con el formato original de los comprobantes fiscales. El uso de estos formatos asegura la interoperabilidad de la API, ya que pueden ser consumidos por sistemas heterogéneos.
De esta forma, el diseño REST, el uso de verbos HTTP y la correcta elección de formatos de datos permitieron que la solución desarrollada fuera escalable, interoperable y alineada a los estándares modernos de servicios web.

3.3 Análisis de datos

El análisis de datos es el proceso de examinar, limpiar, transformar e interpretar conjuntos de datos con el objetivo de extraer información útil, apoyar la toma de decisiones y generar conocimiento. Este proceso puede ser descriptivo, predictivo o prescriptivo, dependiendo de si se limita a resumir datos pasados, anticipar comportamientos futuros o proponer acciones óptimas [6].

El análisis de datos se apoya en conceptos y técnicas provenientes de la estadística, la minería de datos y la inteligencia de negocios. Entre los fundamentos más relevantes se encuentran:

- Extracción y limpieza de datos: preparación de los conjuntos eliminando duplicados, errores o inconsistencias.
- Transformación y normalización: adaptación de los datos a formatos estructurados (por ejemplo, conversión de CFDI en XML a tablas relacionales o JSON para análisis automático).
- Exploración de datos (EDA): uso de estadísticas descriptivas y visualizaciones para detectar patrones, tendencias y anomalías.
- Técnicas de modelado: desde algoritmos de regresión y clasificación hasta métodos de clustering o análisis predictivo.
- Visualización de resultados: uso de gráficos y tableros que permitan una comprensión rápida por parte de usuarios no técnicos.

En el contexto de servicios web y APIs, el análisis de datos cobra relevancia al integrar procesos de captura y transformación en tiempo real, permitiendo que aplicaciones externas consuman información procesada sin necesidad de acceder directamente a la base de datos original [7].

Impacto en el desarrollo

En este proyecto, el análisis de datos se orienta a procesar la información contenida en los CFDI para identificar montos, emisores, receptores y otros campos relevantes, mediante la API desarrollada, se facilita la extracción de información semiestructurada en XML, su transformación a JSON/tablas, y posteriormente su análisis.
Esto permite a los usuarios finales generar reportes, visualizar tendencias y realizar consultas específicas sin interactuar manualmente con cada documento fiscal. El proyecto no solo automatiza la lectura y conversión de CFDI, sino que sienta las bases para análisis posteriores como control de gastos o reportes financieros automatizados.

3.4 CFDI

El CFDI (Comprobante Fiscal Digital por Internet) es un documento electrónico estandarizado que sirve para comprobar la realización de una operación fiscal, como una venta, un ingreso, un egreso o el pago de una nómina. Está avalado por el SAT y debe cumplir con un formato XML específico que contiene la información fiscal necesaria tanto del emisor como del receptor. El uso del CFDI no solo es obligatorio para la mayoría de las operaciones económicas en México, sino que también forma parte del modelo de fiscalización electrónica, permitiendo una trazabilidad completa de las transacciones. Comprender su estructura, funcionamiento y principales componentes es esencial para empresas, profesionistas y estudiantes relacionados con áreas contables, fiscales o tecnológicas [8].

Estructura CFDI 4.0

El Comprobante Fiscal Digital por Internet (CFDI) en su versión 4.0 está estructurado en un formato XML que organiza la información fiscal en distintos nodos. Esta estructura permite que el SAT y los Proveedores Autorizados de Certificación (PAC) puedan validar cada comprobante de manera automática. La comprensión de estos componentes es clave para interpretar correctamente un CFDI y entender su papel dentro del ecosistema tributario mexicano [9].

Componentes principales

Nodo XML	Descripción funcional
cfdi:Comprobante	Es el nodo raíz que contiene datos generales del comprobante: versión, fecha, serie, folio, moneda, método y forma de pago, tipo de comprobante, condiciones de pago, subtotal, descuentos, total, entre otros.
cfdi:Emisor	Contiene los datos del emisor del CFDI: RFC, nombre o razón social, y régimen fiscal (obligatorio).
cfdi:Receptor	Incluye información del receptor: RFC, nombre, uso del CFDI, código postal del domicilio y régimen fiscales
cfdi:Conceptos	Representa los productos o servicios amparados por el comprobante, cada concepto incluye: clave del producto/servicio, cantidad, unidad, descripción, valor unitario, importe, impuestos asociados y descuentos si los hay.
cfdi:Impuestos	Resume los impuestos trasladados y retenidos, incluyendo tipo de impuesto (IVA, ISR, IEPS), tasa o cuota, y monto. Puede aparecer a nivel global o por concepto.
cfdi:Complemento	Sección reservada para añadir información específica mediante complementos oficiales, como nómina, pagos, carta porte, etc. Se integra como subnodo dentro del XML.
cfdi:Addenda	Sección opcional. Permite incluir información no fiscal relevante para acuerdos comerciales (como número de orden de compra, condiciones de entrega, etc.). No es validada por el SAT.
tfd:TimbreFiscalDigital	Nodo que contiene el sello digital del SAT, el UUID (folio fiscal), la fecha de certificación, y el sello del CFDI. Es generado por el PAC y autentifica el CFDI ante el SAT.

Formatos del CFDI: XML y PDF

El CFDI tiene dos formas de representación, que son las siguientes:

- XML: Es la representación digital oficial del CFDI, tiene una estructura específica validada por el SAT, con nodos y atributos codificados en lenguaje XML como: contiene el sello digital, UUID y certificados que garantizan su autenticidad. Es el archivo que se firma digitalmente y se envía al SAT mediante un PAC (Proveedor Autorizado de Certificación).

- PDF: Es una representación impresa o visual del CFDI, se usa para facilitar su lectura y entrega física o digital al cliente. Incluye los datos del XML, pero no tiene validez por sí sola [9].

Tipos de CFDI

El SAT define distintos tipos de CFDI, según la naturaleza de la operación que se esté documentando y se insertan dentro del nodo TipoDeComprobante del CFDI [10]. A continuación, se describen los tipos:

Tipo	Clave XML	Uso principal
Ingreso	I	Para ventas, cobros o cualquier entrada de dinero.
Egreso	E	Para devoluciones, descuentos o cancelaciones, se utiliza para corregir un CFDI de ingreso.
Traslado	T	Para documentar el traslado de mercancías sin valor comercial (por ejemplo, movimiento entre almacenes).
Nómina	N	Para pagar y documentar sueldos y salarios, prestaciones, ISR retenido, etc.
Pago	P	Para registrar pagos en parcialidades o diferidos (Complemento para Recepción de Pagos - REP), se emite cuando el pago no es en una sola exhibición.

Complementos del CFDI

Los complementos son extensiones que se integran al CFDI para añadir información adicional y específica, dependiendo del tipo de operación que se esté registrando. Si bien el CFDI por sí solo incluye los datos generales de una transacción, en muchos casos es necesario proporcionar detalles que no se contemplan en la estructura base del comprobante. Cada complemento está diseñado para atender una necesidad fiscal o administrativa particular, y su uso es obligatorio solo cuando la operación lo requiera, según las disposiciones del SAT. Por ejemplo, si se emite un CFDI para el pago de una nómina, es obligatorio incluir el complemento de nómina, ya que contiene información detallada sobre percepciones, deducciones, impuestos retenidos, entre otros datos laborales [11].

Tipos de complementos CFDI

Complemento	Clave XML	Uso principal
Nómina	nomina12	Para documentar sueldos, salarios, percepciones, deducciones, ISR retenido, subsidios, etc. Obligatorio en CFDI tipo “Nómina”.
Pagos (REP)	pago20	Para registrar pagos en parcialidades o diferidos, detalla montos pagados y facturas relacionadas. Obligatorio en CFDI tipo “Pago”.
Comercio Exterior	cce11	Para exportaciones definitivas, incluye información aduanal, fracción arancelaria y datos del comprador extranjero.
Carta Porte	cartaporte20	Para documentar el transporte de mercancías por medios propios o contratados, contiene datos de rutas, unidades, operadores y productos transportados.

Donatarias	donat11	Para indicar que el CFDI corresponde a una donación deducible, emitido por organizaciones autorizadas por el SAT.
Educativo (colegiaturas)	educativo10	Para registrar pagos de colegiaturas y permitir su deducción en la declaración anual de personas físicas.

Timbrado CFDI

El timbrado es el proceso mediante el cual un CFDI es certificado digitalmente por un Proveedor Autorizado de Certificación (PAC). Durante este proceso, el PAC valida la estructura del comprobante, verifica los datos fiscales del emisor y receptor, y añade un Timbre Fiscal Digital (TFD), que contiene:

- El UUID (folio fiscal), que funciona como un identificador único del CFDI.
- La fecha y hora de certificación.
- El sello digital del SAT, que garantiza la autenticidad.
- El sello del emisor, previamente generado mediante su certificado de sello digital.

Una vez timbrado, el CFDI se considera legal y puede ser enviado al receptor, almacenado por el emisor y reportado automáticamente al SAT [11].

Roles del PAC (Proveedor Autorizado de Certificación)

Los PAC son entidades autorizadas por el SAT para validar, certificar y timbrar los CFDI, su función es fundamental dentro del ecosistema de facturación electrónica. Entre sus principales responsabilidades se encuentran:

- Validar la sintaxis del XML de acuerdo con los esquemas establecidos por el SAT.
- Verificar el certificado de sello digital (CSD) del emisor.
- Certificar el CFDI mediante el timbre fiscal digital.
- Enviar el CFDI timbrado al SAT y al emisor.
- Garantizar disponibilidad y seguridad de los servicios de timbrado las 24 horas.
- Asignar el folio fiscal único (UUID).

El PAC actúa como un intermediario técnico confiable entre el contribuyente y el SAT, agilizando el proceso sin comprometer la validez fiscal [11].

4 Diseño

4.1 Requerimientos

Requerimientos funcionales

1. Gestión de autenticación y autorización

ID	Descripción del requerimiento	Prioridad	Responsable
RF01	La API debe soportar autenticación mediante tokens (JWT/OAuth) y claves de API (API Keys)	Alta	Gabriel
RF02	Permitir autorización basada en roles (RBAC) para controlar el acceso a datos sensibles.	Alta	Gabriel
RF03	Soporte para autenticación multi-tenant en caso de sistemas SaaS.	Baja	Gabriel

2. Consulta y filtrado de datos

ID	Descripción del requerimiento	Prioridad	Responsable
RF04	Permitir consultas con filtros dinámicos (por fechas, rangos, categorías, etc.).	Alta	Yuliana
RF05	Soporte para paginación y ordenamiento de resultados.	Alta	Yuliana
RF06	Capacidad de consultar datos en tiempo real y en modo batch (lotes).	Media	Carlos

3. Procesamiento y transformación de datos

ID	Descripción del requerimiento	Prioridad	Responsable
RF07	Permitir aplicar transformaciones básicas (agregaciones, promedios, sumatorias) desde la API.	Alta	Yuliana
RF08	Capacidad de combinar múltiples fuentes de datos en una sola respuesta (joins virtuales).	Alta	Gabriel
RF09	Soporte para ejecución de scripts personalizados (Python, R, SQL) para análisis básico, medio o avanzado.	Media	Gabriel

4. Exportación e integración de datos

ID	Descripción del requerimiento	Prioridad	Responsable
RF10	Soporte para múltiples formatos de respuesta (JSON, XML, CSV, Excel).	Alta	Yuliana
RF11	Permitir la descarga de informes generados (PDF, XLSX, etc.).	Media	Gabriel
RF12	Webhooks o notificaciones en tiempo real cuando un análisis esté listo	Alta	Carlos

5. Soporte para analítica avanzada

ID	Descripción del requerimiento	Prioridad	Responsable
RF13	Generación de gráficos y visualizaciones (en formato JSON para frontends o como imágenes).	Alta	Carlos
RF14	Análisis históricos y comparativos (YoY, MoM, tendencias).	Baja	Gabriel
RF15	Integración con modelos de machine learning (predicciones, clasificaciones).	Baja	Carlos

Requerimientos no funcionales

ID	Descripción del requerimiento	Prioridad	Responsable
RNF01	Cifrado de comunicaciones con TLS y uso seguro de secretos (sin claves en código).	Alta	Gabriel
RNF02	Autenticación con JWT/OAuth2 integrada en FastAPI.	Alta	Gabriel
RNF03	Autorización por roles en endpoints.	Alta	Yuliana
RNF04	CORS restringido por lista blanca de dominios.	Alta	Yuliana
RNF05	Validación estricta de datos con Pydantic/JSON Schema.	Alta	Carlos
RNF06	Operaciones críticas usando	Alta	Gabriel

	transacciones ACID en PostgreSQL.		
RNF07	Migraciones de base de datos controladas con prisma.	Alta	Yuliana
RNF08	Documentación de API con OpenAPI y ejemplos de uso	Media	Todos

Restricciones y consideraciones

Restricciones técnicas

- Limitaciones de infraestructura en la versión beta: uso de un solo servidor o clúster pequeño para pruebas.
- Soporte limitado para scripts personalizados (RF09) en la beta, restringido a Python y SQL.
- Solo se permitirá el uso de los formatos de respuesta JSON, XML, CSV, Excel y PDF especificados en los requerimientos funcionales (RF10).

Restricciones de negocio

- Cumplimiento con regulaciones fiscales mexicanas (SAT) y normativas de protección de datos (LFPDPPP).
- La API debe ser accesible para usuarios con conocimientos técnicos limitados (interfaz simplificada para contribuyentes).
- No se permitirá análisis predictivo en la primera versión: funcionalidades de Machine Learning (RF15) estarán deshabilitadas o limitadas.
- En la etapa inicial, el acceso se limitará a usuarios dentro de México.

Consideraciones

- Priorizar la seguridad (RF01, RF02, RF26, RF27, RF28) debido a la sensibilidad de los datos fiscales.
- Usar tecnologías adecuadas para el desarrollo
- Diseñar la API con un enfoque RESTful y documentar con OpenAPI para facilitar integraciones (RF23, RF24).

4.2 Arquitectura de software

Dado el objetivo y los requerimientos, se propone una arquitectura con los siguientes componentes:

1. API Gateway:
    - Maneja autenticación (JWT/OAuth, API Keys) y autorización (RBAC).

2. Autenticación (RF01, RF02, RF03):
    - Gestiona tokens, roles, y multi-tenancy.
    - Base de datos: Redis para sesiones, PostgreSQL para usuarios y roles.

3. Consulta y Filtrado (RF04, RF05, RF06):
    - Procesa consultas dinámicas, paginación, y ordenamiento.
    - Soporta modo batch y tiempo real.
    - Base de datos: PostgreSQL.

4. Procesamiento de Datos (RF07, RF08, RF09):
    - Realiza agregaciones, estadísticas (promedio, mediana, moda, varianza, etc.), y joins virtuales.
    - Soporta scripts personalizados (Python con Pandas/NumPy, SQL).
    - Tecnología: FastAPI para endpoints

5. Microservicio de Exportación (RF10, RF11, RF12):
    - Genera respuestas en múltiples formatos (JSON, CSV, Excel, XML).
    - Crea informes descargables (PDF, XLSX).
    - Implementa webhooks para notificaciones.
    - Tecnología: Python con bibliotecas como openpyxl y reportlab.

6. Sistema de Logging y Auditoría (RF21, RF22):
    - Registra consultas, cambios, y uso de la API.

- Almacena logs en Elasticsearch o una base de datos dedicada.
- Genera reportes de uso con herramientas como Kibana.

7. Infraestructura:
    - Despliegue en la nube (AWS, Azure, o GCP) o local con Docker/Kubernetes.
    - Colas de mensajes (Kafka o RabbitMQ) para procesamiento asíncrono (RF19).
    - Cifrado en tránsito (HTTPS/TLS) y en reposo (RF26).

4.3 Arquitectura middleware

El API Gateway es el punto de entrada para todas las solicitudes y actúa como middleware global, manejando tareas comunes antes de distribuirlas a los microservicios.

Funciones:
- Autenticación: Valida JWT Bearer (RF01).
- Autorización: Verifica roles y permisos (RF02).
- Rate Limiting: Limita el número de solicitudes por usuario (RF20).
- Enrutamiento: Redirige solicitudes a los microservicios correspondientes.
- Transformación de Solicitudes: Normaliza formatos de entrada/salida (RF10).
- Logging: Registra todas las solicitudes para auditoría (RF21).
- Cifrado: Asegura HTTPS/TLS (RF26).

Tecnologías:
- FastAPI con middleware personalizado.

Versión beta

API Gateway: Usa FastAPI con middleware básico (autenticación, logging).

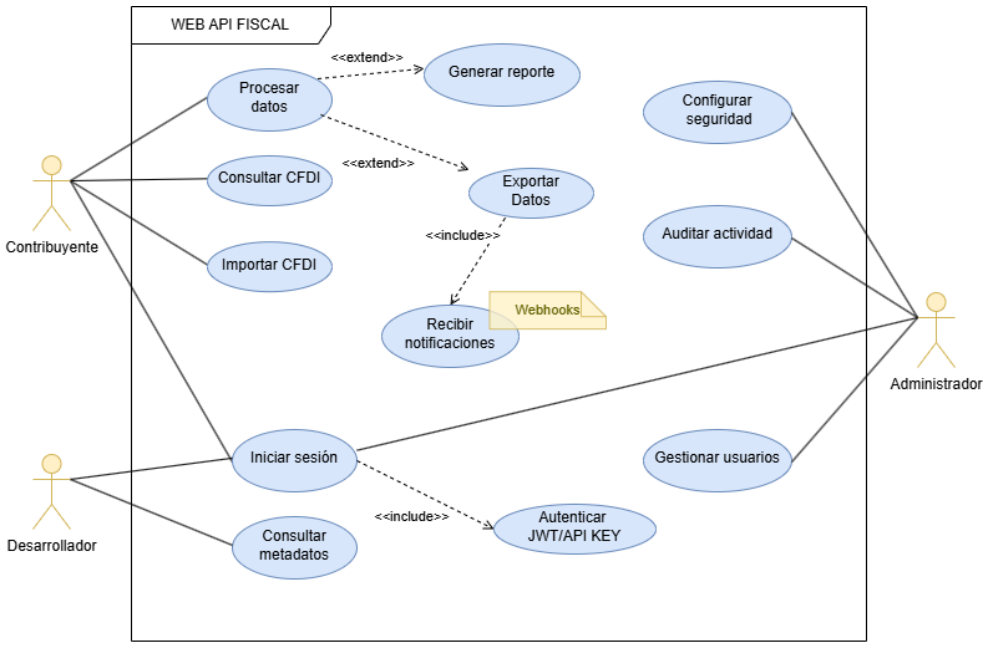
Microservicios Clave:
- Autenticación: Soporte JWT y roles.
- Consulta: Filtros dinámicos y caché.
- Procesamiento: Cálculos estadísticos con Celery.
- Exportación: Formatos JSON/CSV, etc y webhooks.

Tecnologías Iniciales: FastAPI, Redis, Celery, PostgreSQL.

Pruebas: Válida autenticación, consultas, y notificaciones con herramientas como Postman.

4.4 Diagramas

Casos de uso



Especificación casos de uso

Caso de uso 1	Iniciar sesión		
Descripción	Permite a un usuario autenticarse en la Web API mediante JWT o API Keys para acceder a las funcionalidades del sistema. (RF01)		
Actores	Contribuyente, Desarrollador, Administrador		
Pre condiciones	- El usuario debe tener credenciales válidas (usuario/contraseña o API Key). - El sistema debe estar disponible y configurado para autenticación		
Post condiciones	- El usuario recibe un token JWT o API Key válido. - El usuario puede acceder a los recursos autorizados según su rol.		
Secuencia Normal	#		
	1	El usuario envía sus credenciales (usuario/contraseña o API Key) a través del endpoint /auth/login.	
	2	El sistema valida las credenciales contra la base de datos.	
	3	El sistema genera un token JWT con los datos del usuario (ID, rol, tenant).	
	4	El sistema devuelve el token al usuario.	
Excepciones	#		
	1	- Acción: Credenciales inválidas	Respuesta: El sistema devuelve un error 401 (No autorizado) con el mensaje "Incorrect RFC or password ".

	2	- Acción: Sistema no disponible.	Respuesta: Error 503 (Servicio no disponible).
Rendimiento	El sistema deberá realizar las acciones descritas en el paso 1 al 4, en un máximo de 1.2s		
Frecuencia	Alta (cada vez que un usuario necesita acceder al sistema).		
Importancia	Alta (es el punto de entrada para todas las operaciones).		
Urgencia	Inmediatamente		
Comentarios	- El token tiene una duración limitada (1 día) y puede ser renovado. - Soporte para multi-tenancy (RF03) implica incluir el tenant ID en el token.		

Caso de uso 2	Consulta CFDI		
Descripción	Permite consultar los CFDI del usuario con filtros dinámicos (fechas, categorías, etc.), soportando paginación y ordenamiento. (RF04, RF05, RF06)		
Actores	Contribuyente, Desarrollador, Sistema Externo		
Pre condiciones	- El usuario debe estar autenticado (token válido). - Los CFDI deben estar disponibles en la base de datos.		
Post condiciones	- El sistema devuelve una lista de CFDI que cumplen con los filtros, paginada y ordenada según lo solicitado.		
Secuencia Normal	#		
	1	El usuario envía una solicitud GET a los endpoints de consulta	
	2	El sistema valida el token y los parámetros de la solicitud.	
	3	El sistema consulta la base de datos con los filtros especificados.	
	4	El sistema aplica paginación y ordenamiento.	
	5	El sistema devuelve los resultados en formato JSON	
Excepciones	#		
	1	- Acción: Filtros inválidos (e.g., formato de fecha incorrecto).	Respuesta: Error 400 (Solicitud incorrecta) con mensaje "Filtros inválidos".
	2	- Acción: Token expirado.	Respuesta: Error 401 (No autorizado).
Rendimiento	El sistema deberá realizar las acciones descritas en el paso 1 al 5, en un máximo de 2 s		
Frecuencia	Alta (los usuarios consultan frecuentemente sus CFDI).		
Importancia	Alta (es una funcionalidad central para los contribuyentes).		

Urgencia	Inmediatamente
Comentarios	- Soporte para modo batch (RF06) puede implicar procesar grandes volúmenes de datos asíncronamente. - Caché (RF18) puede aplicarse para mejorar el rendimiento.

Caso de uso 3	Procesar Datos	
Descripción	Permite al usuario realizar agregaciones, cálculos estadísticos básicos (promedio, mediana, moda, suma, conteo, etc.), y combinar datos de múltiples fuentes (e.g., CFDI, conceptos, complementos) mediante joins virtuales. Los resultados se devuelven en el formato solicitado (JSON, CSV, XML, Excel, PDF) y, opcionalmente, se guardan como reportes con notificaciones en tiempo real vía webhook (RF07, RF08, RF12).	
Actores	Contribuyente, Desarrollador	
Pre condiciones	El usuario debe estar autenticado con un token válido (Authorization: Bearer) y tener el permiso join:execute. Los datos (CFDI) deben estar disponibles para procesamiento. Para consultas con filtros específicos (e.g., folio, start_date, end_date), los datos deben cumplir con los criterios de filtrado Si se solicita guardar el reporte (save_report=True), el usuario debe tener permisos para crear reportes (report:create). Para notificaciones vía webhook (RF12), el usuario debe tener un webhook_url configurado en la tabla User.	
Post condiciones	El sistema devuelve los resultados procesados en el formato solicitado (JSON, CSV, XML, Excel, PDF), incluyendo información de paginación (page, page_size, total_pages). Si save_report=True, el reporte se guarda en la tabla Report con metadatos (nombre, descripción, filtros, operación, etc.). Si save_report=True y el usuario tiene un webhook_url configurado, se crea un registro en la tabla Notification y se envía una notificación HTTP POST al webhook con el report_id, formato, y timestamp. Los resultados incluyen agregaciones o estadísticas según la consulta seleccionada (e.g., promedio de gastos, conteo de conceptos, suma de impuestos).	
Secuencia Normal	#	
	1	El usuario envía una solicitud POST a los endpoints establecidos con los datos a procesar
	2	El sistema valida el token y los datos de entrada.
	3	El sistema realiza los cálculos (e.g., promedio, mediana, varianza, etc).
	4	Si se solicita un join virtual, el sistema combina datos de múltiples fuentes, (consultas ya predefinidas).
	5	El sistema devuelve los resultados en formato solicitado

Excepciones	#		
	1	Token de autenticación inválido o faltante	Devuelve HTTP 401 Unauthorized con mensaje "Invalid or missing token".
	2	page_size igual a 0	Devuelve HTTP 422 Unprocessable Entity con mensaje "page_size must be greater than 0".
	3	Filtros requeridos faltantes (e.g., start_date y end_date)	Devuelve HTTP 400 Bad Request con mensaje "start_date and end_date are required for summary queries".
	4	Error al generar el reporte (e.g., formato inválido	Devuelve HTTP 500 Internal Server Error con mensaje "Failed to generate report: ".
	5	El usuario no tiene el permiso (RBAC)	Devuelve HTTP 403 Forbidden con mensaje "Permission denied".
Rendimiento	El sistema debe completar los pasos 1 al 5 en un máximo de 10 segundos para consultas con hasta 1000 registros (page_size=1000). Para grandes volúmenes de datos, las consultas pueden encolar asíncronamente (RF19) para cumplir con el tiempo de respuesta.		
Frecuencia	Media (depende de la frecuencia con la que los usuarios necesiten análisis estadísticos).		
Importancia	Alta (es una funcionalidad clave para análisis fiscal).		
Urgencia	Inmediatamente		
Comentarios	- Scripts personalizados (RF09) se implementarán en fases futuras. - Los cálculos asíncronos (RF19) se implementarán para grandes volúmenes de datos en fases futuras. - Los resultados en formatos binarios (Excel, PDF) se codifican en Base64 para evitar errores como UnicodeDecodeError.		

Caso de uso 4	Exportar Datos
Descripción	Permite exportar los resultados de consultas o análisis en múltiples formatos (JSON, CSV, Excel, PDF). (RF10, RF11). Los datos exportados pueden devolverse directamente en la respuesta o guardarse como un reporte en la base de datos con notificaciones opcionales vía webhook si save_report=True o o descargarse desde un reporte previamente guardado (GET /api/v1/download/{report_id}). Los formatos binarios (Excel, PDF) se entregan como archivos descargables codificados en Base64, mientras que otros formatos (JSON, CSV, XML) se devuelven como texto.
Actores	Contribuyente

Pre condiciones	El usuario debe estar autenticado con un token válido (Authorization: Bearer) y tener el permiso join:execute o reports:generate (para descargar reportes). Los datos a exportar (e.g., resultados de una consulta predefinida o análisis) deben estar disponibles en la base de datos (**src.database.db**). Para descargar reportes guardados (GET /api/v1/download/{report_id}), el reporte debe existir en la tabla Report y pertenecer al usuario autenticado (user_id = user["rfc"]) Si save_report=True, el usuario debe tener permisos para crear reportes (report:create). Si se solicita notificación vía webhook (RF12), el usuario debe tener un webhook_url configurado en la tabla User. La consulta predefinida (join_id) debe ser válida (1-16) y los filtros (e.g., start_date, end_date, folio) deben cumplir con los requisitos de la consulta.		
Post condiciones	- El sistema genera y devuelve los datos exportados en el formato solicitado (JSON, CSV, XML, Excel, PDF). Para Excel y PDF, se devuelve un archivo binario con el encabezado Content-Disposition para descarga. Para JSON, CSV, y XML, se devuelve el contenido como texto en el cuerpo de la respuesta. - Si save_report=True, el reporte se guarda en la tabla Report con metadatos (e.g., name, description, filters, operation, join_name, file_content en Base64). - Si save_report=True y el usuario tiene un webhook_url, se crea un registro en la tabla Notification con type="analysis_ready", status="sent" o "failed", y se envía una notificación HTTP POST al webhook_url con el payload {"event_type": "analysis_ready", "payload": {"report_id": <ID>, "format": <format>}, "timestamp": <ISO>}. - Los logs del sistema registran la ejecución, incluyendo el tiempo de procesamiento y posibles errores. Para GET /api/v1/download/{report_id}: El reporte se descarga en el formato almacenado (JSON, CSV, XML, Excel), con el contenido decodificado si es necesario (e.g., Base64 para Excel). El encabezado Content-Disposition asegura que el archivo se descargue con un nombre adecuado (e.g., report_<report_id>.<format>).		
Secuencia Normal	#		
	1	Opción A: Generar y exportar un nuevo reporte El usuario autenticado envía una solicitud POST a /api/v1/join/predefined con parámetros: join_id (1-16), filters (e.g., start_date, end_date, folio, format), page, page_size, y opcionalmente save_report, name, y description. Opción B: Descargar un reporte existente El usuario autenticado envía una solicitud GET a /api/v1/download/{report_id} con el report_id del reporte previamente guardado.	
	2	Opción A: El sistema valida los parámetros: join_id (1-16), page (≥1), page_size (1-1000), formato (json, csv, xml, excel, pdf), y filtros requeridos (e.g., start_date, end_date). Opción B: El sistema verifica que el reporte	

		exista en la tabla Report (id=report_id) y que pertenezca al usuario autenticado (user_id=user["rfc"]).	
	3	Opción A: El sistema ejecuta la consulta predefinida usando JoinService.execute_predefined_join, combinando datos de tablas como CFDI, Issuer, Concept, o Payment, y aplicando filtros y paginación. Opción B: El sistema recupera el reporte de la tabla Report usando db.report.find_unique(where={"id": report_id}) y valida la propiedad del usuario	
	4	Opción A: Los resultados se procesan con export_join_service.generate_report_from_data , generando el archivo en el formato solicitado. Si save_report=True, el reporte se guarda en la tabla Report con un report_id y metadatos. Excel y PDF se codifican en Base64; JSON, CSV, y XML se devuelven como texto. Opción B: El sistema decodifica el file_content (Base64 para Excel) y determina el content_type según el formato del reporte (e.g., application/vnd.openxmlformats-officedocument.spreadsheetml.sheet para Excel).	
	5	Opción A: Si save_report=True y el usuario tiene un webhook_url, se crea una notificación en la tabla Notification con estado "pending" y se envía un POST al webhook_url usando NotificationService.notify_webhook. El estado se actualiza a "sent" o "failed". El sistema devuelve el archivo (Excel/PDF) o el contenido (JSON/CSV/XML) con los encabezados adecuados. Opción B: El sistema devuelve el contenido del reporte como un archivo descargable con el encabezado Content-Disposition (e.g., attachment; filename=report_.)	
Excepciones	#		
	1	Token de autenticación inválido o faltante	Devuelve HTTP 401 Unauthorized con mensaje "Invalid or missing token".
	2	El usuario no tiene el permiso join:execute (Opción A) o reports:generate (Opción B)	Devuelve HTTP 403 Forbidden con mensaje "Permission denied".
	3	Opción A: page_size igual a 0	Devuelve HTTP 422 Unprocessable

			Entity con mensaje "page_size must be greater than 0".
		Opción A: Formato no válido (e.g., format="txt")	Devuelve HTTP 400 Bad Request con mensaje "Invalid format. Must be one of: json, csv, xml, excel, pdf".
		Opción A: Filtros requeridos faltantes (e.g., start_date)	Devuelve HTTP 400 Bad Request con mensaje "Missing required filters".
		Opción B: Reporte no encontrado (report_id inválido)	Devuelve HTTP 404 Not Found con mensaje "Report not found".
		Opción A: Folio especificado en filters.folio no encontrado	Devuelve HTTP 404 Not Found con mensaje "Folio not found".
		Opción A: Error al generar el reporte (e.g., datos inválidos para Excel)	Devuelve HTTP 500 Internal Server Error con mensaje "Failed to generate report: ".
		Opción B: Error al decodificar file_content (e.g., Base64 inválido)	Opción B: Error al decodificar file_content (e.g., Base64 inválido)
		Opción A: Usuario sin webhook_url configurado	Crea un registro en Notification con status="failed" y no envía el webhook.
		Opción A: Error al enviar el webhook (e.g., HTTP 500 o timeout)	Actualiza el registro en Notification a

			status="failed" y registra el error en los logs.
Rendimiento	El sistema deberá realizar las acciones descritas en el paso 1 al 5, en un máximo de 4 s		
Frecuencia	Media (los usuarios exportan datos para informes periódicos).		
Importancia	Alta (es esencial para generar informes fiscales).		
Urgencia	Inmediatamente		
Comentarios	Los formatos soportados (JSON, CSV, XML, Excel, PDF) se generan con export_join_service.generate_report_from_data (POST /api/v1/join/predefined) o se recuperan de la tabla Report (GET /api/v1/download/{report_id}). Los archivos binarios (Excel, PDF) se codifican en Base64 en la tabla Report (file_content) y se decodifican al descargar. Las notificaciones vía webhook (RF12) se envían cuando save_report=True, usando el webhook_url del usuario. El procesamiento asíncrono (RF19) para formatos pesados está planificado para fases futuras. Los enlaces de descarga en GET /api/v1/download/{report_id} están protegidos por autenticación y permisos, pero podrían mejorarse con tokens firmados de tiempo limitado en el futuro. La paginación (page, page_size, total_pages) asegura un manejo eficiente de grandes volúmenes de datos.		

Caso de uso 5	Recibir Notificaciones	
Descripción	Permite registrar y recibir notificaciones en tiempo real mediante webhooks cuando un análisis o exportación está listo. (RF12)	
Actores	Desarrollador	
Pre condiciones	- El usuario debe estar autenticado. - El usuario debe haber registrado un webhook válido mediante el endpoint /webhooks/register.	
Post condiciones	- El sistema envía una notificación al webhook registrado con los detalles del evento (e.g., enlace de descarga).	
Secuencia Normal	#	
	1	El usuario registra un webhook enviando una solicitud POST a /webhooks/register con la URL del webhook.
	2	El sistema valida el token y almacena la URL del webhook.
	3	Cuando un evento ocurre (e.g., exportación completada), el sistema encola una tarea para enviar la notificación.
	4	El sistema envía una solicitud POST al webhook con los datos del evento.

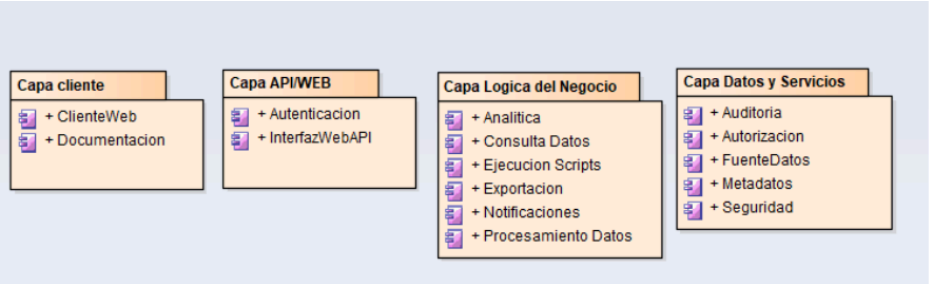
Excepciones	#		
	1	- Acción: Webhook no responde.	Respuesta: El sistema reintenta 3 veces; si falla, registra el error en logs (RF21).
	2	- Acción: URL de webhook inválida.	Respuesta: Error 400 con mensaje "URL inválida".
Rendimiento	El sistema deberá realizar las acciones descritas en el paso 1 al 5, en un máximo de 2 s		
Frecuencia	Media (depende de la frecuencia de análisis/exportaciones).		
Importancia	Alta (esencial para integraciones automatizadas).		
Urgencia	Inmediatamente		
Comentarios	- Usa un sistema de colas (Kafka, RabbitMQ) para garantizar que las notificaciones se envíen de forma fiable. - Los datos enviados al webhook deben estar cifrados (RF26).		

Caso de uso 6	Importar CFDI		
Descripción	Permite al contribuyente importar CFDI desde archivos XML para su procesamiento y almacenamiento en el sistema.		
Actores	Contribuyente		
Pre condiciones	- El usuario debe estar autenticado. - El archivo XML debe cumplir con el estándar del SAT (versión 4.0).		
Post condiciones	- El CFDI se almacena en la base de datos con sus datos asociados (conceptos, impuestos, adjuntos). - Se devuelve un mensaje de éxito con el UUID del CFDI.		
Secuencia Normal	#		
	1	El usuario autenticado envía una solicitud POST con un archivo XML.	
	2	El sistema valida el formato y contenido del XML.	
	3	El sistema extrae y procesa los datos del CFDI (emisor, receptor, conceptos, etc.).	
	4	El sistema almacena el CFDI y su adjunto (codificado en Base64) en la base de datos.	
Excepciones	#		
	1	Si el XML es inválido, se devuelve un error 400.	
	2	Si ocurre un error de base de datos, se devuelve un error 500.	
Rendimiento	El sistema deberá realizar las acciones descritas en los pasos 1 al 4 en un máximo de 5 segundos.		

Frecuencia	Alta (los contribuyentes importan CFDI regularmente para su gestión fiscal).
Importancia	Alta (esencial para la funcionalidad principal del sistema).
Urgencia	Inmediatamente
Comentarios	- El archivo XML se almacena como Base64 para compatibilidad con la base de datos. - En futuras mejoras se podrían incluir validación contra el SAT en tiempo real.

Diagrama de componentes

División por capas



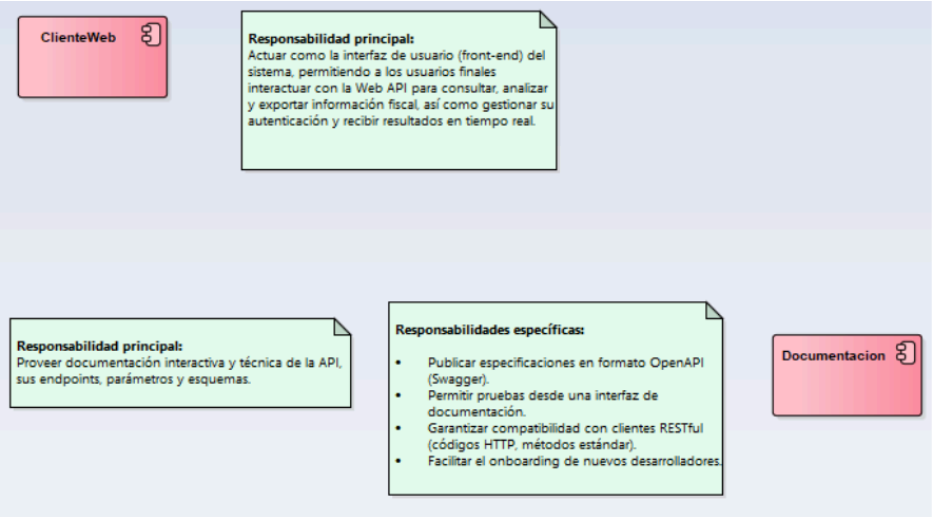
Capa cliente

Proveer una interfaz gráfica de usuario para permitir que los usuarios interactúen con la Web API del sistema, es responsable de la experiencia de usuario, navegación, entrada de datos y visualización de resultados.

Responsabilidades detalladas

- Proporcionar menús, formularios, filtros y vistas intuitivas para interactuar con el sistema.
- Capturar y enviar credenciales para autenticación. Gestionar sesiones con tokens JWT o API Keys.
- Enviar solicitudes HTTP (GET, POST, etc.) hacia la InterfazWebAPI, incluyendo headers, tokens y parámetros.
- Mostrar resultados obtenidos de consultas, exportaciones o análisis en forma de tablas, gráficos o dashboards.
- Validar datos antes de enviarlos a la API (fechas, filtros, parámetros) y prevenir errores de usuario.
- Mostrar alertas, mensajes o confirmaciones (por ejemplo: “Análisis listo”, “Exportación descargada”).
- Facilitar al usuario el acceso a la documentación técnica (por ejemplo, vía Swagger UI).

Componentes capa cliente



Componente ClienteWeb

Responsabilidades especificas

Interacción con usuario

- Capturar credenciales del usuario
- Envió de datos de login al componente autenticación a través de la interfaz web api
- Almacenamiento y envió de tokens (JWT/API KRY) para sesiones autenticadas

Solicitudes de análisis de datos

- Solicitar datos filtrados (fechas, categorías, etc.)
- Pedir cálculos estadísticos, tendencias o conjuntos desde el componente consulta datos o analítica

Comunicación con servicios de la API

- Enviar peticiones HTTP a la interfazwebAPI
- Mostrar resultados obtenidos de la API al usuario en forma visual (tablas, graficas, dashboards)

Exportación de resultados

- Permitir al usuario solicitar y descargar archivos generados por el componenteExportacion (Excel, CSV, PDF)

Recepción de notificaciones

- Mostrar al usuario alertas o notificaciones provenientes del componente notificaciones

Acceso a la documentación

- Ofrecer acceso a la documentación técnica de la API en interfaces como Swagger UI

Navegación y experiencia de usuario

- Proveer menús, formularios, filtros y paneles que guíen al usuario la interacción con el sistema
- Validar datos de entrada antes de enviarlos a la API

Componente Documentación

Responsabilidades especificas

- Publicar especificaciones en formato OPENAPI (swagger)
- Permitir pruebas desde una interfaz de documentación
- Garantizar compatibilidad con clientes RESTful (códigos HTPP, métodos estandar).

- Facilitar el onboarding de nuevos desarrolladores

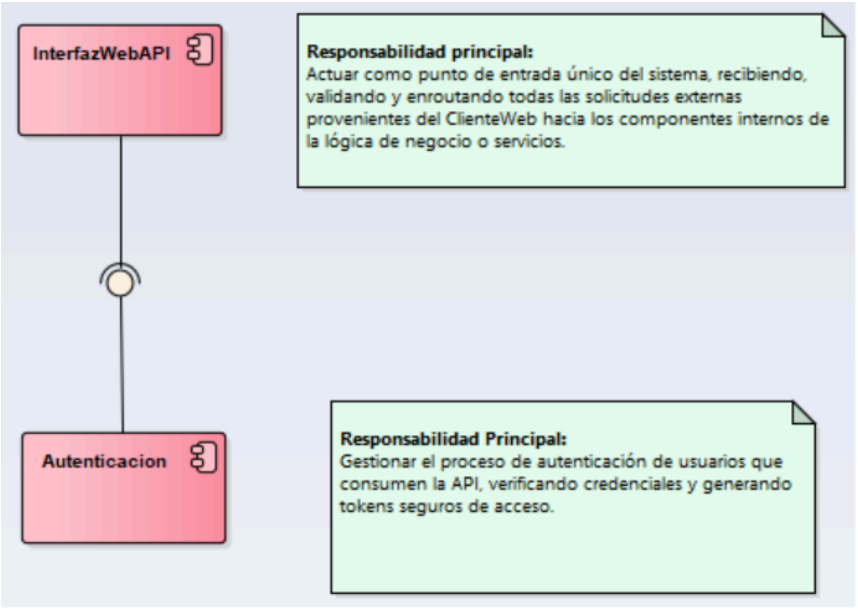
Capa API/Web

Servir como puerta de entrada oficial al sistema, esta capa expone los servicios del backend a través de interfaces web (REST), validando, enrutando y controlando el acceso a cada operación solicitada.

Responsabilidades detalladas

- Exponer los endpoints RESTful que los clientes pueden consumir (/login, /consultar, /exportar, etc.).
- Validar estructura y formato de peticiones (JSON, parámetros, headers).
- Verificar autenticación (tokens, claves) y consultar con autorización antes de permitir el acceso a recursos.
- Enrutar solicitudes hacia los componentes internos adecuados (ConsultaDatos, Exportación, etc.).
- Estandarizar y enviar respuestas con los códigos de estado HTTP apropiados (200 OK, 401 Unauthorized, etc.).
- Publicar especificaciones OpenAPI o Swagger como referencia para desarrolladores externos.
- Gestionar diferentes versiones de la API sin afectar a clientes existentes (/v1/, /v2/, etc.).
- Actuar como capa de protección entre el cliente y la lógica interna, evitando el acceso directo a los servicios.

Componentes capa API/Web



Componente InterfazWebAPI

Responsabilidades especificas

Recepción de solicitudes externas

- Expones HTPP para acceso a funcionalidades del sistema (HET, POST, PUT DELETE)
- Recibir peticiones del ClienteWeb u otros consumidores

Validación Inicial

- Validar la estructura y formato de solicitudes (parámetros, cuerpos JSON, etc)
- Verificar autenticidad y validez de los tokens (JWT, API KEYS)

Orquestación de seguridad

- Redirigir la autenticación al componente Componente Autenticacion
- Consultar con componente Autorización si el usuario tiene permiso para operación

Ruteo de solicitudes a componentes internos
- Redirigir cada endponint a su componente responsable (consultaDatos, Exportacion, Analitica, etc)
- Transformar la solicitud si es necesario para el componente destino

Entrega de respuesta
- Formatear la respuesta en formatos estándar (JSON, XML)
- Manejar códigos de estado

Interacción con documentación
- Exponer especificación OpenAPI/Swagger para que el componente documentación pueda generar documentación interactiva

Versionado y mantenimiento
- Sopotar versiones de la API

Manejo de seguridad transversal
- Evitar acceso a endoponits sensibles sin autorización
- Prevenir ataques como inyecciones, accesos inválidos y denegaciones de servicio

Componente Autenticación

Responsabilidades específicas

- Validar credenciales de usuarios mediante usuario/contraseña, API Keys o tokens JWT.
- Generar tokens de sesión autenticados con tiempo de expiración.
- Verificar la validez de tokens incluidos en peticiones.
- Rechazar accesos no autorizados antes de llegar a otros servicios.

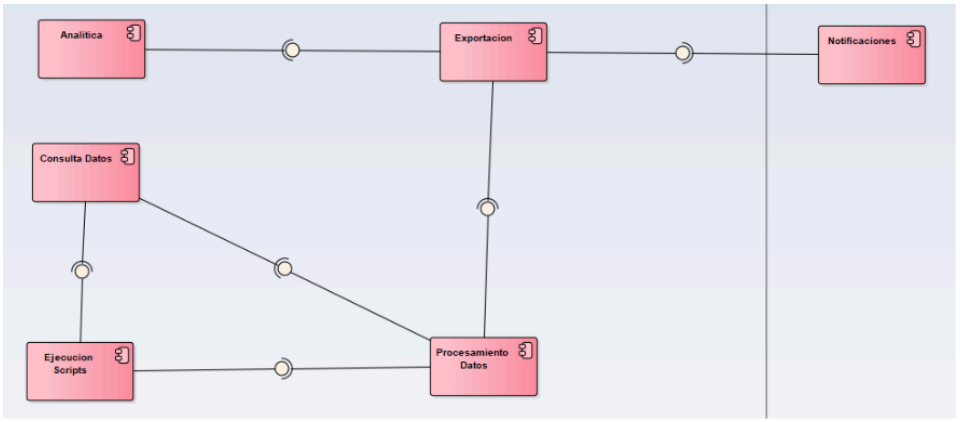
Capa lógica del negocio

Representa el núcleo funcional del sistema, en ella se concentra la implementación de las reglas, operaciones y procesos que definen el comportamiento específico de la aplicación, esta capa actúa como intermediaria entre las solicitudes recibidas desde la interfaz de programación (API/Web) y las operaciones de acceso a datos o servicios internos, asegurando que todas las interacciones sigan las reglas y restricciones definidas por el dominio del sistema.

Responsabilidades detalladas

- Ejecutar operaciones funcionales solicitadas por los usuarios, tales como procesamiento, análisis, filtrado o transformación de datos.
- Aplicar reglas de negocio específicas que determinan cómo se deben manipular los datos antes de entregarlos o almacenarlos.
- Controlar el flujo de información que proviene desde la capa de presentación, validando las instrucciones recibidas.
- Encapsular los algoritmos necesarios para generar resultados calculados, estadísticas, reportes o análisis estructurados.
- Coordinar tareas entre diferentes áreas del sistema, garantizando la consistencia y coherencia de los resultados entregados.
- Administrar procesos asincrónicos o por lotes cuando se requiera un tratamiento diferido de la información.

Componentes de la capa lógica del negocio



Componente ConsultaDatos

Permitir a los usuarios recuperar datos fiscales mediante filtros, rangos de fechas, paginación y ordenamiento.

Responsabilidades específicas

- Ejecutar consultas dinámicas basadas en filtros definidos por el usuario.
- Soportar ordenamiento de resultados.
- Implementar paginación eficiente.
- Diferenciar entre consultas en tiempo real y en lotes.
- Validar permisos antes de exponer datos (consulta solo si autorizado).
- Interactuar con el componente de procesamiento si se requiere transformación previa.

Componente ProcesamientoDatos

Aplicar transformaciones a los datos solicitados, tales como agregaciones, cálculos estadísticos y operaciones sobre conjuntos.

Responsabilidades específicas

- Realizar cálculos como suma, promedio, mediana, moda, etc.
- Unir o cruzar datos de diferentes fuentes (joins virtuales).
- Procesar datos para ser enviados en formatos agregados o resumidos.
- Trabajar en conjunto con consultas y exportación.
- Garantizar que las operaciones se realicen sobre datos autorizados.

Componente Ejecución Scripts

Permitir la ejecución controlada de scripts definidos por el usuario (en Python, R, SQL) para análisis avanzados.

Responsabilidades específicas

- Interpretar y ejecutar scripts enviados por el usuario.
- Asegurar aislamiento y seguridad en la ejecución de código.
- Validar sintaxis y tipo de análisis solicitado.
- Integrarse con los datos internos o cargados por el usuario.
- Devolver resultados de forma estructurada a otros componentes.

Componente Notificaciones

Informar al usuario en tiempo real sobre el estado de sus análisis o solicitudes de datos.

Responsabilidades específicas

- Enviar notificaciones mediante Webhooks, correo u otro canal configurado.
- Detectar eventos como "análisis completado", "exportación disponible", "error en ejecución".
- Integrarse con componentes de procesamiento y exportación.
- Gestionar colas de eventos o suscripciones según el usuario.

Componente Analítica

Realizar análisis avanzados sobre los datos, incluyendo generación de gráficos, comparativas y predicciones.

Responsabilidades específicas

- Generar visualizaciones en formatos JSON o imagen.
- Calcular indicadores históricos como tendencias, variaciones YoY/MoM.
- Integrarse con modelos de machine learning.
- Proveer resultados al frontend o exportación.
- Permitir configuración de parámetros de análisis por parte del usuario.

Componente Exportación

Permitir a los usuarios exportar resultados de análisis o consultas en múltiples formatos de archivo.

Responsabilidades específicas

- Convertir respuestas en formatos como JSON, XML, CSV, Excel, PDF.
- Generar informes descargables de manera automática o bajo demanda.
- Integrarse con el componente de notificaciones para informar al usuario cuando el archivo esté listo.
- Garantizar que solo se exporten datos a los que el usuario tiene acceso autorizado.

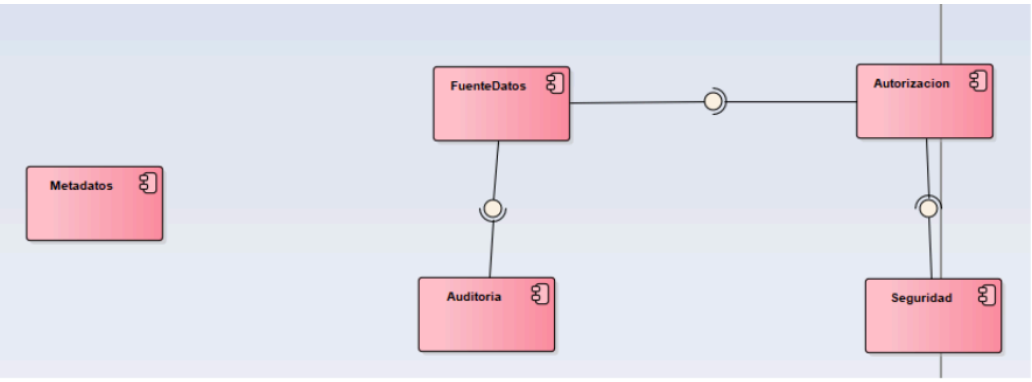
Capa de datos y servicios

Tiene como propósito fundamental proveer infraestructura de soporte, acceso a fuentes de información y gestión de servicios transversales del sistema, su diseño garantiza que los recursos críticos como bases de datos, seguridad, trazabilidad y cumplimiento normativo sean utilizados de forma segura, controlada y eficiente

Responsabilidades detalladas

- Facilitar el acceso estructurado y seguro a las fuentes de datos, asegurando consistencia, disponibilidad y control de concurrencia.
- Implementar políticas de seguridad técnica que protejan la integridad, confidencialidad y validez de la información en tránsito y en reposo.
- Aplicar mecanismos de validación técnica sobre los datos de entrada y salida para prevenir vulnerabilidades como inyecciones o accesos indebidos.
- Registrar eventos significativos del sistema, como autenticaciones, transacciones, errores o acciones críticas, para fines de auditoría y monitoreo.
- Proveer soporte a los procesos de cumplimiento normativo, asegurando que el sistema se adhiera a estándares legales o industriales cuando corresponda.
- Atender funciones que requieren persistencia, verificación de identidad, control de acceso y otras capacidades de base para el funcionamiento confiable del sistema.

Componentes de la capa de datos y servicios



Componente Autorización

Determinar si un usuario autenticado tiene permiso para realizar una operación específica o acceder a ciertos datos.

Responsabilidades específicas

- Implementar control de acceso basado en roles (RBAC).
- Consultar y mantener la matriz de permisos por usuario/rol.
- Validar acciones específicas contra el rol del usuario.
- Integrarse con el componente de autenticación para obtener información del usuario autenticado.
- Proporcionar decisiones de acceso a otros componentes (consulta, procesamiento, exportación).

Componente Metadatos

Administrar y exponer la estructura y características de los conjuntos de datos disponibles para análisis.

Responsabilidades específicas

- Mostrar tipos de datos, nombres de campos, descripciones.
- Facilitar la exploración de catálogos de datos.
- Implementar búsqueda semántica por campo o significado.

Componente FuenteDatos

Servir como capa de acceso a las fuentes de datos internas o externas (ej. base de datos fiscal, archivos CFDI).

Responsabilidades específicas

- Ejecutar queries y devolver resultados estructurados.
- Gestionar la conexión y seguridad de acceso a la base de datos.
- Adaptar datos para uso por otros componentes.
- Soportar acceso concurrente y consultas complejas.

Componente Seguridad

Aplicar medidas técnicas de protección, validación y cumplimiento de normativas legales.

Responsabilidades específicas

- Cifrar datos en tránsito (TLS/HTTPS) y en reposo si aplica.
- Validar los datos de entrada para prevenir inyecciones SQL/NoSQL.
- Apoyar cumplimiento con GDPR, HIPAA u otras normativas del sector.
- Colaborar con los componentes de autenticación y autorización.

Componente Auditoría

Registrar toda la actividad relevante del sistema para fines de trazabilidad, monitoreo y cumplimiento.

Responsabilidades específicas

- Guardar logs de consultas, errores, exportaciones, autenticaciones.
- Asociar cada acción a un usuario, rol y hora.
- Generar reportes de uso por usuario u organización.
- Integrarse con sistemas externos de monitoreo si aplica.

Diagrama general de componentes – Black Box

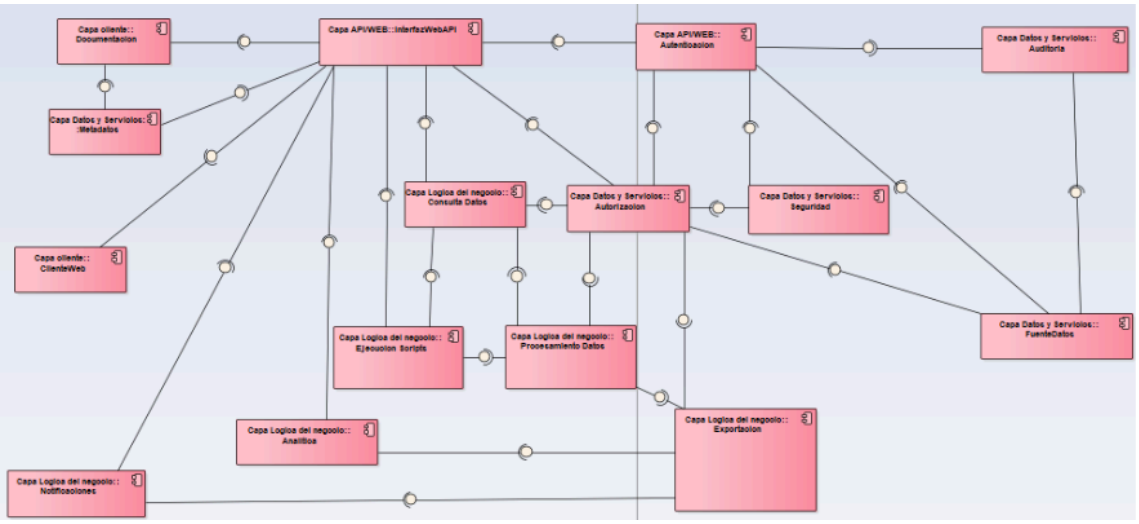


Diagrama White Box

Componente ConsultaDatos

Es responsable de ofrecer capacidades de consulta estructurada sobre datos fiscales, permitiendo filtrar información por fechas, categorías o rangos, aplicando paginación y entregando resultados listos para ser procesados o exportados.

Interfaces

Interfaz proporcionada

IConsultaDatos

Expone las operaciones de consulta que pueden ser invocadas por otros componentes como InterfazWebAPI.

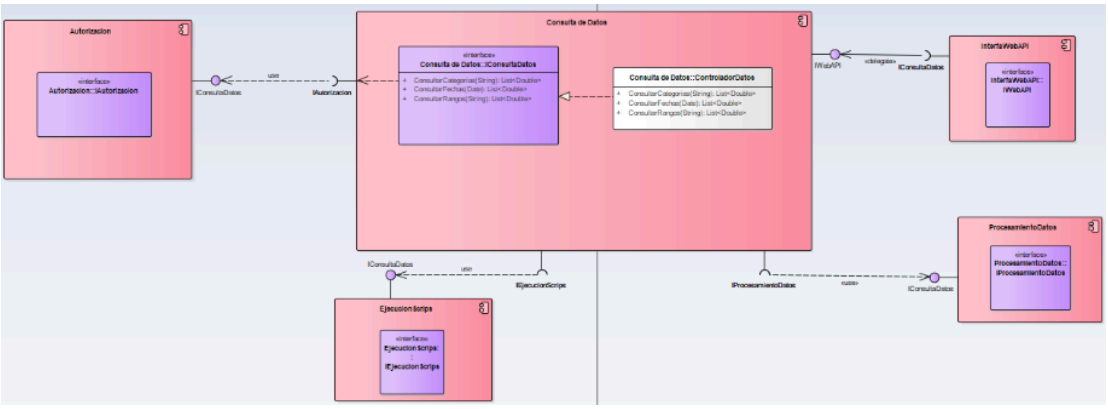
Operaciones expuestas

consultarPorFechas(Date): List<Double>
consultarPorCategoria(String): List<Double>
consultarPorRango(String): List<Double>

Interfaz requerida	Proveedor	Propósito funcional
IAutorización	Componente Autorización	Validar que el usuario tenga permisos para consultar los datos solicitados.
IEjecuciónScripts	Componente	Permitir la ejecución

	EjecuciónScripts	controlada de scripts definidos por el usuario (en Python, R, SQL) para análisis avanzados
IProcesamientoDatos	Componente ProcesamientoDatos	Aplicar transformaciones o filtros especiales sobre los resultados obtenidos.
IWebAPI	Componente InterfazWebAPI	Indica que esta interfaz es consumida por la API del sistema.

Diagrama



Componente ProcesamientoDatos

Es responsable de aplicar transformaciones sobre los datos obtenidos a través de consultas, incluyendo agregaciones, cálculos estadísticos y operaciones sobre conjuntos. Funciona como capa intermedia entre los datos crudos y los resultados entregados al usuario final.

Interfaces

Interfaz proporcionada

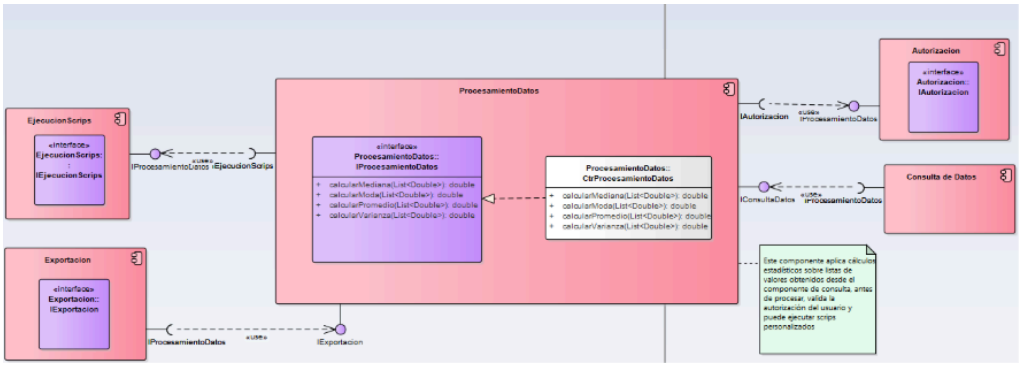
IProcesamientoDatos

- Expone las operaciones de procesamiento disponibles para otros componentes.

Operaciones expuestas

- calcularPromedio(List<Double>): Double
- calcularMediana(List<Double>): Double
- calcularModa(List<Double>): Double
- calcularVarianza(List<Double>): Double

Diagrama



Componente Analítica

Es responsable de realizar operaciones de análisis avanzadas sobre datos fiscales consultados previamente, esto incluye cálculos de tendencias, análisis comparativos entre periodos, generación de visualizaciones y, eventualmente, integración con modelos predictivos de aprendizaje automático.

Interfaces

Interfaz proporcionada

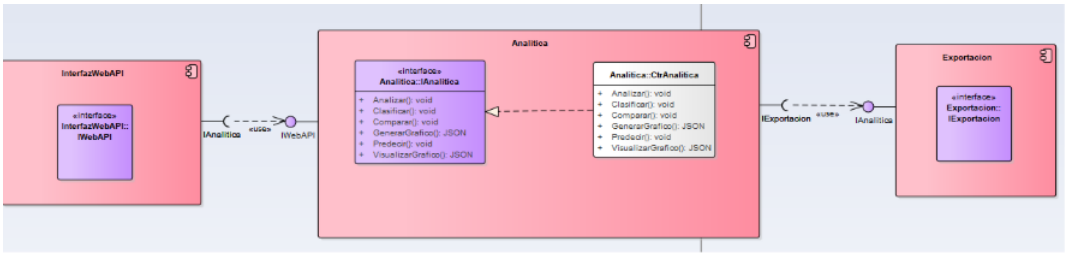
IAnalitica

- Expone operaciones analíticas para otros componentes (como Exportación o InterfazWebAPI).

Operaciones expuestas

- Analizar():void
- Claificar():void
- Comparar():void
- GenerarGrafico():JSON
- Predecir():void
- VisualizarGrafico():JSON

Interfaz requerida	Proveedor esperado	Propósito
IConsultaDatos	Componente ConsultaDatos	Obtener datos sobre los que se aplicarán los cálculos
IAutorización	Componente Autorización	Verificar que el usuario tiene permisos para realizar ciertos análisis
IExportacion	Componente Exportacion	Permitir la descarga de informes generados (PDF, XLSX, etc.).
IEjecuciónScripts (opcional)	Componente EjecuciónScripts	Ejecutar cálculos avanzados definidos por el usuario



Diagrama

Componente Exportación

Permite generar archivos con resultados de análisis o consultas realizados por el usuario, ofrece formatos como PDF, Excel (XLSX), CSV o JSON, y gestiona la preparación, formato y entrega de dichos archivos.

Interfaces

Interfaz proporcionada

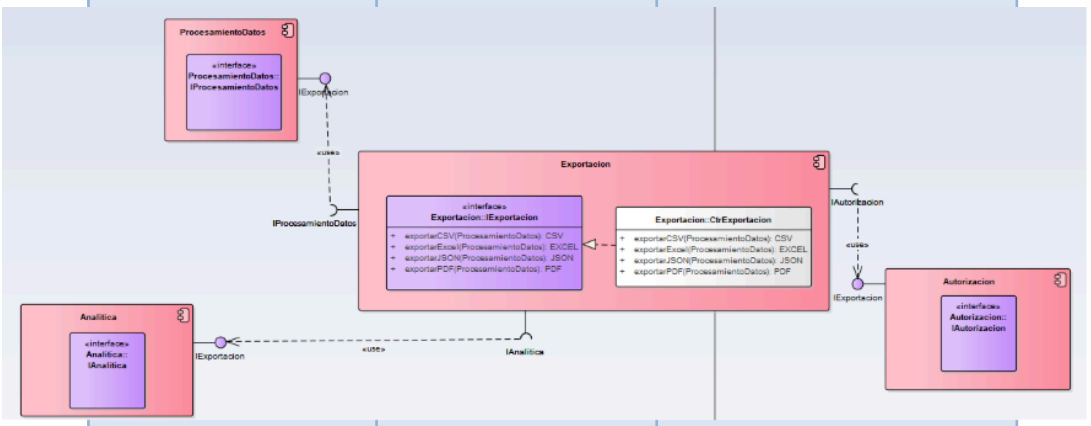
IExportacion

- Expone métodos que permiten exportar datos o resultados analíticos en diversos formatos.

Operaciones expuestas

- exportarCSV(ProcesamientoDatos): CSV
- exportarExcel(ProcesamientoDatos)XLSX
- exportarPDF(ProcesamientoDatos): PDF
- exportarJSON(ProcesamientoDatos): JSON

Interfaz requerida	Proveedor	Propósito
IConsultaDatos	ConsultaDatos	Obtener datos sin procesar para exportar
IProcesamientoDatos	ProcesamientoDatos	Obtener datos ya transformados o agregados
IAnalitica	Analítica	Exportar resultados gráficos o comparativos
IAutorización	Autorización	Verificar si el usuario puede exportar ciertos datos



Diagrama

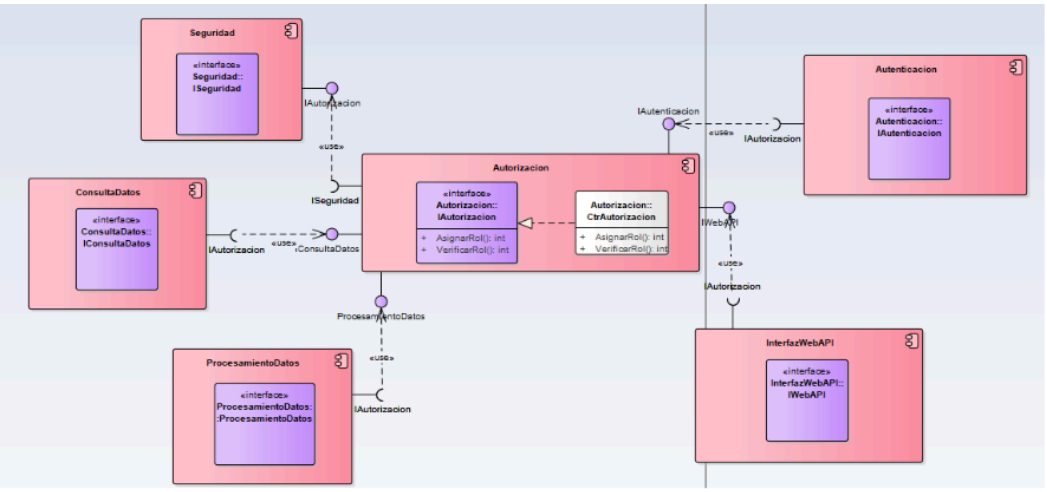
Componente Autorización

Se encarga de verificar que un usuario autenticado tenga permisos suficientes para acceder a recursos específicos del sistema, implementa lógica de control de acceso basada en roles (RBAC) y políticas, y se consulta desde otros componentes antes de ejecutar operaciones sensibles.

Interfaces

Interfaz proporcionada
IAutorizacion

- Permite consultar los permisos o roles de un usuario sobre una acción o



recurso.

Operaciones expuestas

- AsignarRol():boolean
- VerificarRol():boolean

Diagrama

Componente Autenticación

Se encarga de verificar la identidad del usuario, validando credenciales (usuario/contraseña, token, API Key, etc.), generando tokens de sesión y garantizando que las solicitudes provienen de entidades válidas, también puede gestionar sesiones y tokens con tiempo de expiración.

Interfaces

Interfaz proporcionada

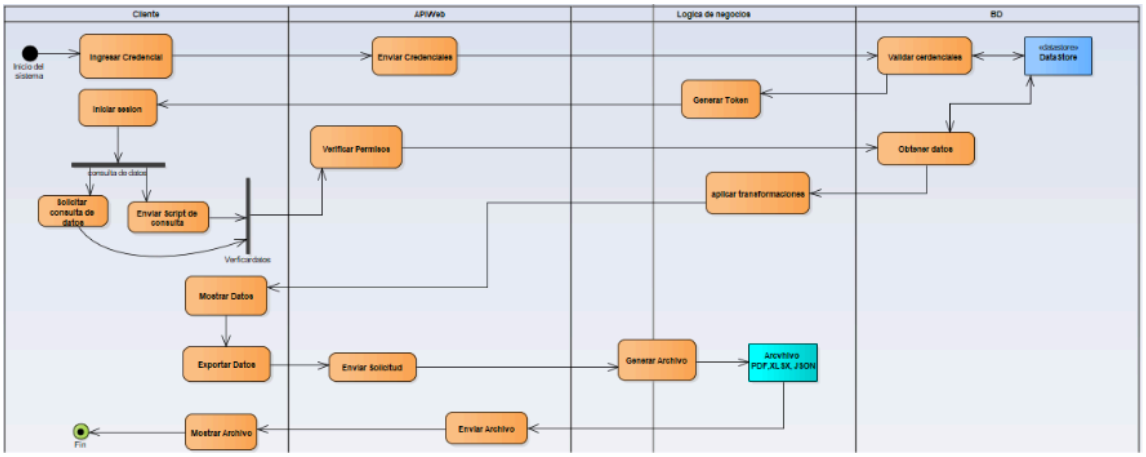
IAutenticacion

- Ofrece servicios de login, verificación de tokens y obtención de identidad de usuario autenticado.

Operaciones expuestas

- ObtenerToken():void
- ValidarToken():void

Diagrama



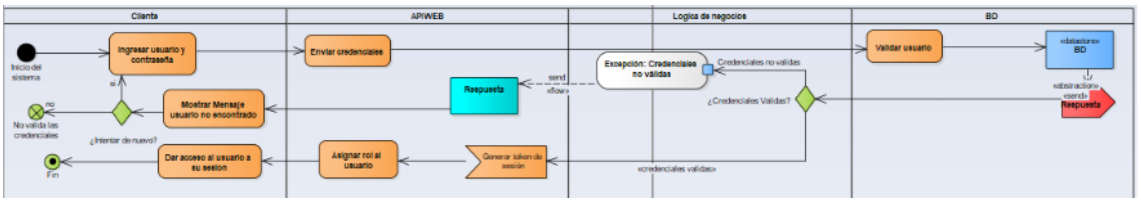


Diagrama de actividades

Diagrama global

Diagrama Iniciar sesión

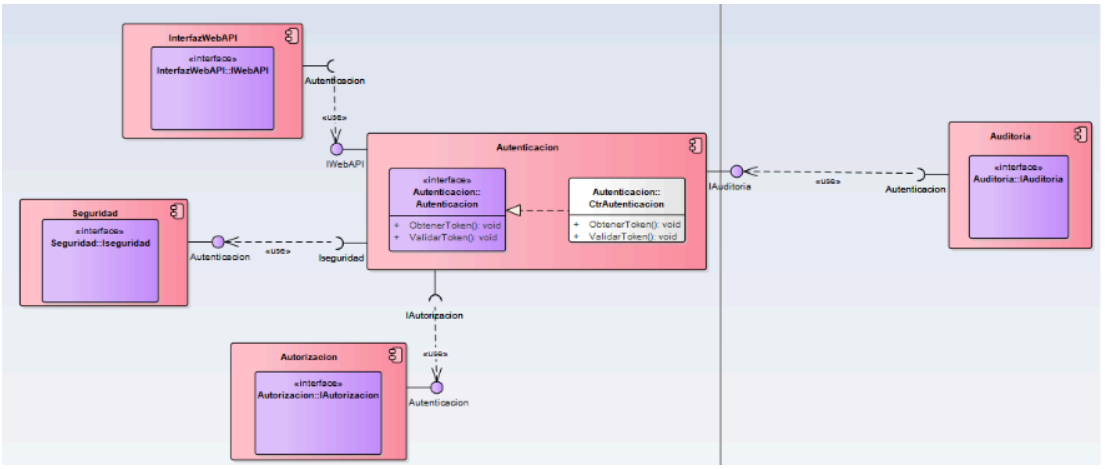
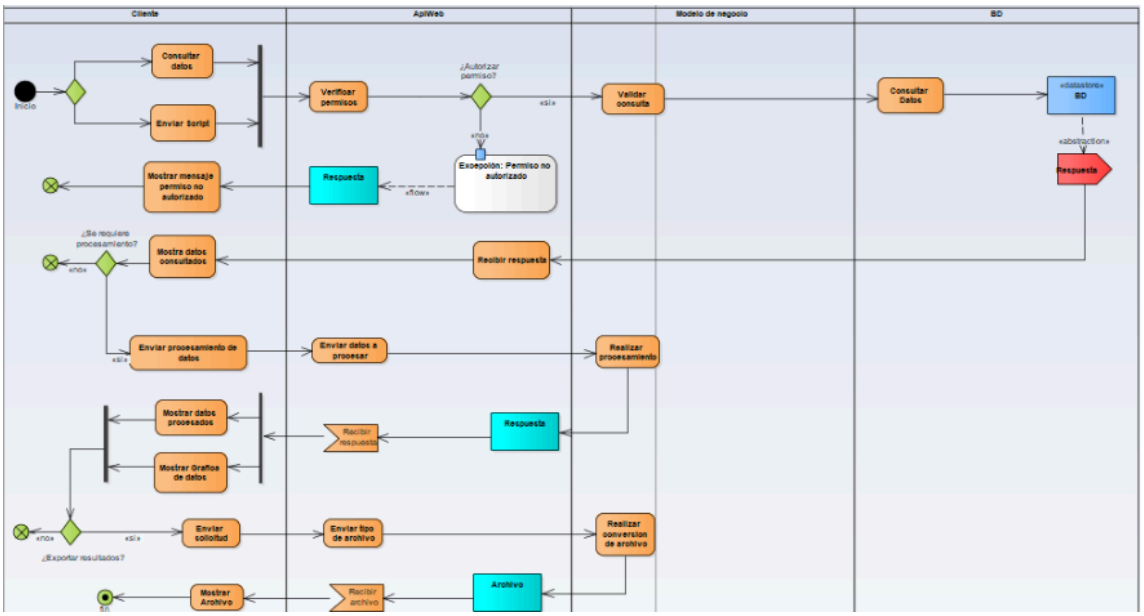
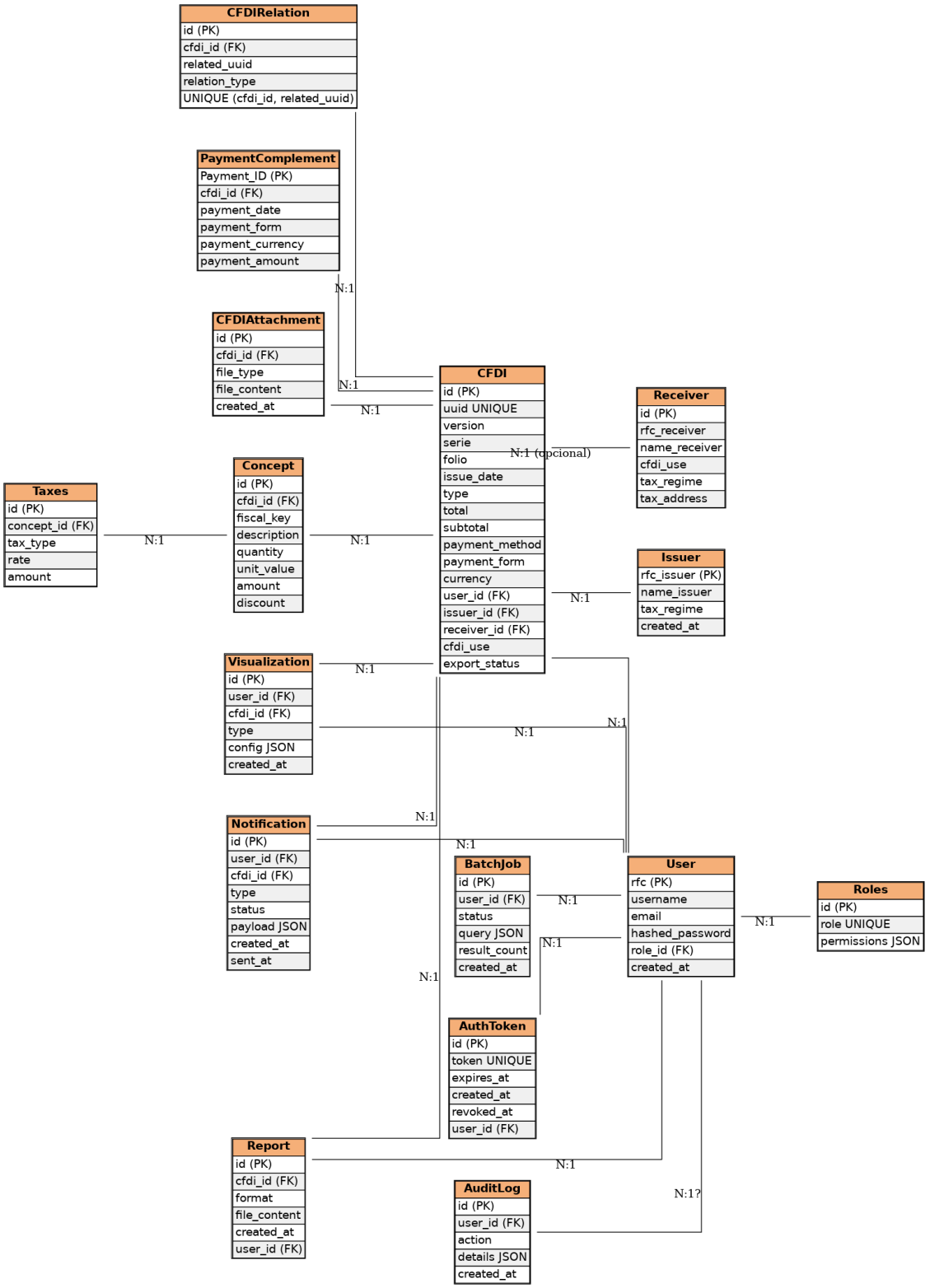


Diagrama Procesamiento de datos

4.5 Base de datos

Diagrama ER





Especificación de diagrama ER

1) Roles (Roles)
Propósito: catálogo de roles para permisos en la plataforma.

- id: Identificador único del rol (clave primaria).
- role: Nombre del rol (p. ej., *admin*, *user*); se usa en la autorización y visibilidad de funciones.
- permissions: Conjunto de permisos/opciones extra en formato estructurado; sirve para activar/desactivar capacidades específicas del rol.

2) Usuario (User)
Propósito: personas que usan la plataforma (contribuyentes/operadores).

- rfc: Identificador fiscal y clave primaria del usuario; también sirve para relacionarlo con sus CFDI y acciones.
- username: Nombre visible del usuario en interfaces, reportes y auditoría.
- email: Correo de contacto y recuperación; puede emplearse para notificaciones.
- hashed_password: Contraseña almacenada de forma segura; se utiliza en el inicio de sesión.
- role_id: Referencia al rol asignado al usuario; determina permisos y alcance.
- created_at: Fecha de alta del usuario; útil para auditoría y métricas de adopción.

3) Token de Autenticación (AuthToken)
Propósito: control y auditoría de tokens emitidos (sesiones/API).

- id: Identificador único del token (clave primaria).
- token: Cadena del token emitido; se utiliza para validar acceso mientras esté vigente.
- expires_at: Momento en que el token deja de ser válido; soporta seguridad temporal.
- created_at: Cuándo se generó el token; para auditoría y diagnóstico.
- revoked_at: Si el token fue invalidado antes de expirar; permite cierre de sesión forzado.
- user_id: Usuario al que pertenece el token; relaciona sesiones con cuentas.

4) Emisor (Issuer)
Propósito: entidad que emite el CFDI.

- rfc_issuer: Identificador fiscal del emisor y clave primaria; enlaza emisores con CFDI.
- name_issuer: Denominación o razón social del emisor; aparece en reportes y verificaciones.
- tax_regime: Régimen fiscal con el que tributa el emisor; requerido para validez del comprobante.
- created_at: Registro de cuándo se dio de alta el emisor; útil para control.

5) Receptor ("Receiver")
Propósito: destinatario del CFDI.

- id: Identificador interno del receptor (clave primaria).
- rfc_receiver: Identificador fiscal del receptor; se usa para búsquedas y validaciones.
- name_receiver: Denominación o razón social del receptor.
- cfdi_use: Clave de uso que el receptor dará al CFDI (catálogo SAT).
- tax_regime: Régimen fiscal del receptor; necesario para la emisión correcta.
- tax_address: Domicilio fiscal (por lo general código postal); requerido en CFDI 4.0.

6) CFDI ("CFDI")
Propósito: comprobante fiscal con su información principal.

- id: Identificador interno del comprobante (clave primaria).
- uuid: Folio fiscal único asignado por el SAT; base para validación y relaciones entre CFDI.
- version: Versión del estándar del comprobante (p. ej., 4.0); indica el esquema aplicado.
- serie: Serie del comprobante definida por el emisor; ayuda a orden interno.
- folio: Folio interno del comprobante dentro de la serie.
- issue_date: Fecha y hora de expedición; clave para filtros por periodo y vigencia.

- seal: Sello digital del comprobante; prueba de integridad.
- certificate_number: Número de certificado del emisor; identifica el CSD usado.
- certificate: Certificado público asociado; respaldo para validaciones.
- place_of_issue: Lugar de expedición (clave de CP o localidad) declarado por el emisor.
- type: Tipo de comprobante (I/E/T…); determina su naturaleza (ingreso, egreso, traslado).
- total: Importe total del CFDI, ya con impuestos y descuentos.
- subtotal: Importe antes de impuestos y descuentos; base de cálculo de totales.
- payment_method: Medio con el que se realizó o realizará el pago (efectivo, tarjeta, etc.).
- payment_form: Esquema de pago (pago en una exhibición, parcialidades/diferido).
- currency: Moneda del comprobante; permite conversiones/reportes multimoneda.
- user_id: Usuario "propietario" lógico del CFDI en la plataforma (quien lo gestiona).
- issuer_id: Emisor al que pertenece el comprobante; vincula con datos fiscales del emisor.
- receiver_id: Receptor del comprobante; puede ser nulo si aún no se relaciona.
- cfdi_use: Uso del CFDI declarado (cuando aplique).
- export_status: Indicador de exportación o estatus aduanal, si corresponde.

7) Archivo de CFDI (CFDIAttachment)
Propósito: archivos asociados al CFDI (XML original, PDF, otros).
- id: Identificador del archivo (clave primaria).
- cfdi_id: Referencia al CFDI al que pertenece el archivo.
- file_type: Tipo de archivo almacenado (p. ej., xml, pdf).
- file_content: Contenido del archivo; sirve para descarga y re-procesamiento.
- created_at: Cuándo se adjuntó; útil para trazabilidad de cargas.

8) Concepto (Concept)
Propósito: partidas o líneas de producto/servicio del CFDI.
- id: Identificador del concepto (clave primaria).
- cfdi_id: Referencia al CFDI al que pertenece esta línea.
- fiscal_key: Clave SAT del producto/servicio (catálogo); estandariza clasificaciones.
- description: Descripción del bien o servicio vendido.
- quantity: Cantidad de unidades en la línea.
- unit_value: Precio por unidad antes de impuestos y descuentos.
- amount: Importe de la línea (base antes de impuestos/descuentos).
- discount: Descuento aplicado a esta línea; afecta el cálculo del total.

9) Impuesto (Taxes)
Propósito: impuestos aplicados por cada concepto.
- id: Identificador del impuesto (clave primaria).
- concept_id: Concepto al que se aplica este impuesto.
- tax_type: Tipo/mecanismo del impuesto (traslado/retención u otros).
- rate: Tasa o cuota aplicada al concepto (porcentaje o factor).
- amount: Importe monetario resultante del impuesto para esa línea.

10) Relación de CFDIs (CFDIRelation)
Propósito: vincular un CFDI con otro por motivos fiscales (p. ej., sustitución, notas de crédito).
- id: Identificador de la relación (clave primaria).
- cfdi_id: CFDI base desde el cual se declara la relación.
- related_uuid: Folio fiscal del CFDI relacionado; permite rastrear el documento contraparte.
- relation_type: Clave de tipo de relación según catálogo (p. ej., sustitución, devolución).

11) Reporte (Report)
Propósito: reportes generados desde la plataforma a partir de un CFDI.

- id: Identificador del reporte (clave primaria).
- cfdi_id: CFDI del cual se generó el reporte.
- format: Formato del reporte (p. ej., PDF, XLSX, CSV).
- file_content: Contenido generado; se usa para descarga/consulta.
- created_at: Fecha de generación del reporte.
- user_id: Usuario que solicitó o generó el reporte; para auditoría.

12) Visualización (Visualization)
Propósito: configuraciones de vistas o dashboards guardados por usuarios.

- id: Identificador de la visualización (clave primaria).
- user_id: Usuario propietario de la visualización.
- cfdi_id: CFDI asociado si la visualización depende de un documento concreto (puede ser nulo).
- type: Tipo de visualización (tabla, gráfica, etc.); guía el render en el frontend.
- config: Configuración/estado de la vista (filtros, campos incluidos, parámetros).
- created_at: Momento de creación; útil para ordenar o limpiar vistas obsoletas.

13) Notificación (Notification)
Propósito: notificaciones o eventos enviados a usuarios (correo, webhook, UI).

- id: Identificador de la notificación (clave primaria).
- user_id: Usuario destinatario o relacionado con el evento.
- cfdi_id: CFDI al que se refiere la notificación (si aplica).
- type: Tipo de notificación (p. ej., *análisis listo*, *error*, *recordatorio*).
- status: Estado del envío o procesamiento (pendiente, enviado, fallido, leído).
- payload: Contenido estructurado del evento (datos del envío o contexto).
- created_at: Momento en que se creó la notificación.
- sent_at: Momento en que se envió (si ya fue despachada).

14) Bitácora de Auditoría (AuditLog)
Propósito: rastrear acciones y cambios en el sistema.

- id: Identificador del evento de auditoría (clave primaria).
- user_id: Usuario que realizó la acción (puede quedar vacío si fue del sistema).
- action: Acción realizada (p. ej., *login*, *descarga XML*, *actualización de rol*).
- details: Datos adicionales del evento en formato estructurado (qué entidad cambió, valores, etc.).
- created_at: Momento en que se registró la acción.

15) Trabajo por Lotes (BatchJob)
Propósito: ejecuciones en segundo plano y consultas masivas.

- id: Identificador del trabajo (clave primaria).
- user_id: Usuario que solicitó el procesamiento.
- status: Estado actual (pendiente, en proceso, completado, falló).
- query: Parámetros o definición de lo que se procesará (filtros, rangos, agregaciones).
- result_count: Cantidad de resultados producidos; útil para métricas y paginación de salidas.
- created_at: Registro de cuándo se creó el trabajo.

16) Complemento de Pago ("PaymentComplement")
Propósito: pagos asociados a CFDI (especialmente PPD).

- Payment_ID: Identificador del complemento de pago (clave primaria).
- cfdi_id: CFDI al que se vincula el pago.
- payment_date: Fecha en que se realizó el pago; base para conciliación.
- payment_form: Medio con el que se efectuó el pago (transferencia, efectivo, etc.).
- payment_currency: Moneda en la que se pagó; útil para conversiones.
- payment_amount: Monto pagado aplicado a ese CFDI.

5 Desarrollo

5.1 Entorno de desarrollo

Durante el desarrollo de la aplicación se integraron diversas tecnologías que permitieron construir un sistema seguro, escalable y accesible:

Base de Datos – PostgreSQL, Prisma, Docker y Railway

En la etapa inicial se trabajó con PostgreSQL en contenedores Docker, lo que permitió estandarizar el entorno de desarrollo en local. Para gestionar la base de datos y sus migraciones se utilizó Prisma ORM, facilitando la definición de modelos y la interacción con PostgreSQL. Posteriormente, la infraestructura se migró a Railway, donde la base de datos quedó desplegada en la nube, permitiendo a todo el equipo acceder de forma centralizada y trabajar sobre un entorno compartido y más ágil.

Backend – FastAPI (Python)

El servidor fue implementado con FastAPI por su rapidez y simplicidad. Se utilizó Swagger/OpenAPI para documentar de manera automática las APIs, lo que facilitó las pruebas, validación e integración.

Frontend – HTML5, Bootstrap y AJAX

Aunque el desarrollo del frontend no era el objetivo principal, se implementaron pantallas básicas con HTML5 y Bootstrap, utilizando AJAX para consumir de forma dinámica los servicios de la API.

Mensajería – RabbitMQ con CloudAMQP

Se utilizaron para la gestión de procesos en segundo plano y comunicación asincrónica, se integró RabbitMQ, usando el servicio CloudAMQP como broker en la nube.

En conjunto, estas herramientas permitieron que el sistema evolucionara de un prototipo local a una aplicación colaborativa en la nube, con APIs documentadas, un backend robusto y soporte para tareas asincrónicas.

5.2 EndPoints

5.3 Pruebas

Referencias

1. Pal, R. (2016). *Semi-structured data models and query languages*. International Journal of Computer Applications, 141(7), 20–25.
2. OCW UC3M. (s. f.). *DTD y XML Schema* [PDF]. Universidad Carlos III de Madrid. Recuperado de https://ocw.uc3m.es/pluginfile.php/1582/mod_page/content/18/DTD_XML_esquema.pdf
3. Luján-Mora, S. (2011). *XML Schema: estructura del documento y elementos* [PDF]. Universidad de Alicante.
4. Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures* (Doctoral dissertation, University of California, Irvine).
5. Ibm. (2025, 18 julio). API REST. *IBM*. https://www.ibm.com/mx-es/think/topics/rest-apis
6. Han, J., Pei, J., & Kamber, M. (2011). *Data Mining: Concepts and Techniques* (3rd ed.). Elsevier.
7. Provost, F., & Fawcett, T. (2013). *Data Science for Business: What you need to know about data mining and data-analytic thinking*. O'Reilly Media.
8. Universidad Nacional Autónoma de México (UNAM). *La factura electrónica en México*.https://www.fca.unam.mx/docs/pubrev/21_2/la-factura-electronica-en-mexico.pdf
9. Instituto Politécnico Nacional (IPN). *Estructura XML del CFDI 4.0*. https://repositorio.ipn.mx/handle/123456789/5432
10. Instituto Politécnico Nacional (IPN). El Comprobante Fiscal Digital por Internet (CFDI) como herramienta para la fiscalización electrónica. https://www.revistacontadurianegocios.ipn.mx/cfdi_fiscalizacion.pdf
11. SAT. *Esquemas y validaciones técnicas de los complementos del CFDI*. https://www.sat.gob.mx/informacion_fiscal/complementos