Humanidades, Tecnología y Ciencia, del Instituto Politécnico Nacional
Certificado de Reserva de Derechos al Uso Exclusivo del Título: 04-2010-0326124-14000-203 ISSN: 2007-1957

CREACIÓN DE API PARA LA GESTIÓN AUTOMATIZADA DE TICKETS DE INCIDENTES

Verónica Agustín Domínguez
Instituto Politécnico Nacional
vagustin@ipn.mx

Axel Ernesto Moreno Cervantes
Instituto Politécnico Nacional
amorenoc@ipn.mx

Armando Tonatiuh Ávalos Bravo
Escuela Superior de Ingeniería Química e Industrias Extractivas
avalosarma@hotmail.com

Abstract

La gestión de incidentes y solicitudes de soporte técnico representa un desafío crítico para las organizaciones modernas, donde la falta de integración entre sistemas internos y plataformas de gestión de boletos de atención de incidentes genera ineficiencias operativas, errores humanos y retrasos en la atención. Este artículo presenta el diseño y desarrollo de una API REST que actúa como intermediario entre los sistemas de los clientes y la plataforma TOPdesk, permitiendo la creación, consulta y gestión automatizada de tickets mediante peticiones HTTP. La solución fue implementada con Node.js y Express.js, contenerizada con Docker para garantizar portabilidad y escalabilidad, y validada mediante pruebas de rendimiento que demostraron tiempos de respuesta inferiores a 700 ms en todos los endpoints. Los resultados confirman que la API constituye una alternativa viable y eficiente para automatizar la gestión de incidentes, reduciendo la carga operativa y mejorando los tiempos de respuesta en entornos empresariales.

*Palabras clave: A*utomatización, API REST, gestión de tickets, Node.js, Docker, TOPdesk

En México, una gran cantidad de empresas y organizaciones enfrentan una creciente necesidad de optimizar la gestión de incidentes en su área de TI y solicitudes de soporte técnico. Un incidente se define como cualquier interrupción no planificada o falla en un servicio de TI que afecta la operación normal de un sistema o proceso. Muchas organizaciones optan por soluciones manuales o por el uso directo de las interfaces nativas de plataformas como *TOPdesk*, *ServiceNow* o

Certificado de Reserva de Derechos al Uso Exclusivo del Título: 04-2010-0326124-14000-203 ISSN: 2007-1957

Jira Service Management, las cuales pueden resultar poco accesibles o limitadas en términos de personalización para usuarios no técnicos. Esta situación conduce a:

- Largos periodos de capacitación y adaptación para los operadores.
- Falta de centralización en la creación y seguimiento de incidentes.
- Dificultades para adaptar los flujos de trabajo a las necesidades específicas de cada empresa.
- Incremento en los tiempos de respuesta y reducción de la calidad del servicio.

Ante esta problemática, se propone el desarrollo de una API que actúe como intermediario flexible entre los sistemas de los clientes y la plataforma TOPdesk.

Un API (Interfaz de Programación de Aplicación) de acuerdo con (ORACLE, 2025) es un conjunto de reglas y protocolos que actúan como intermediarios entre dos aplicaciones y permiten que éstas se comuniquen sin necesidad de conocer una el funcionamiento interno de la otra. Existen diferentes tipos de API, de acuerdo con (POSTMAN, 2025), se tienen las siguientes tipos de API:

- REST: Basadas en peticiones HTTP, tienen buena integración con servicios web.
- SOAP: Basadas en XML, altamente estructuradas, aunque generan más metadatos que usar REST.
- GraphQL: Basadas en JSON, ampliamente usadas para apps móviles.
- gRPC: basada en mecanismos de serialización (buffers), son eficientes, aunque no son del todo independientes del lenguaje de programación.

Dada la necesidad de alta integración con la plataforma TOPDesk se decidió crear un API de tipo REST.

Esta API permitirá a las empresas crear y consultar tickets mediante solicitudes HTTP, personalizando los parámetros según sus necesidades sin depender de la interfaz nativa de TOPdesk. La solución está diseñada para:

- Automatizar la creación de tickets, reduciendo la intervención manual.
- Centralizar la gestión de incidentes desde un único punto de acceso.
- Personalizar los flujos de trabajo según los requerimientos de cada cliente.
- Integrarse de manera sencilla con sistemas internos existentes.

Un API (Interfaz de Programación de Aplicación)

Metodología

El proyecto se desarrolló siguiendo el modelo de desarrollo en cascada, éste de acuerdo con (Delgado Olivera, 2021) es un modelo de gestión de proyectos que divide el desarrollo en fases secuenciales claramente definidas. Este enfoque resultó adecuado debido a que los requisitos del sistema estaban claramente establecidos desde el inicio, incluyendo las funcionalidades a implementar, las plataformas con las que se integraría (TOPdesk y sistemas del cliente) y los flujos de trabajo esperados.

Las fases del desarrollo fueron:

1. Análisis de requisitos: identificación de requerimientos funcionales y no funcionales.
2. Diseño del sistema: definición de la arquitectura, modelos de datos y flujos de información.
3. Implementación: codificación de la API, controladores, servicios y rutas.

Humanidades, Tecnología y Ciencia, del Instituto Politécnico Nacional
Certificado de Reserva de Derechos al Uso Exclusivo del Título: 04-2010-0326124-14000-203 ISSN: 2007-1957

4. Pruebas: verificación de endpoints, autorización, rendimiento y despliegue.
5. Documentación: elaboración de la documentación técnica y de usuario.

Análisis de requisitos

Se definieron requerimientos funcionales tomando como guía a (Pressman & Maxim, 2021) que incluyeron la creación de tickets mediante solicitudes HTTP (POST), la consulta de tickets existentes (GET), la validación de datos de entrada, el envío de tickets a TOPdesk y la autorización de usuarios mediante credenciales. Adicionalmente, se establecieron requerimientos no funcionales orientados a garantizar rendimiento, disponibilidad, portabilidad, fiabilidad y mantenibilidad del sistema.

Reglas de negocio

Se definieron once reglas de negocio que rigen el comportamiento del sistema considerando lo que dice (Leonardi, Sampaio, & Rossi, 1998), entre las que destacan:

- RN1: Todo ticket debe incluir título, creador y rama de pertenencia; de lo contrario, se rechaza con error 400.
- RN3: El envío a TOPdesk solo procede si la validación de datos es exitosa.
- RN7: Las credenciales de autorización son obligatorias; si son inválidas, se retorna error 401.
- RN11: El formato de los datos debe ser JSON; formatos incorrectos generan error 400

Plan de implementación

La implementación consistió en desarrollar un servidor en Node.js utilizando Express.js, con middlewares para el manejo de JSON, registro de solicitudes en archivos CSV y habilitación de CORS (Intercambio de Recursos de Origen Cruzado) para restringir las peticiones hacia la API (solo GET y POST). La API se desplegó dentro de un contenedor Docker definido mediante Docker Compose, lo que permite reflejar cambios automáticamente y exponer puertos configurables a través de variables de entorno.

Arquitectura

El sistema consta de los siguientes módulos:

- server.js: configuración principal del servidor, establece el puerto, tiempos de espera y middlewares.
- Modelos (ticket.js, TicketV2.js, TicketV3.js): definen la estructura de los tickets y encapsulan la información que se enviará a TOPdesk.
- Controladores (topdesk.js): contienen la lógica para manejar las solicitudes HTTP, incluyendo funciones para crear tickets (create, createV2, createV3) y consultarlos (getIncident, getAllIncidents).
- Servicios (topdesk.js): actúan como puente entre los controladores y la API externa de TOPdesk, gestionando el envío y recepción de información.
- Utilidades (topdesk.js): responsables de realizar las peticiones HTTP a TOPdesk utilizando autenticación Basic Auth.
- Rutas (tickets.js): definen los endpoints de la API y los enlazan con las funciones del controlador.

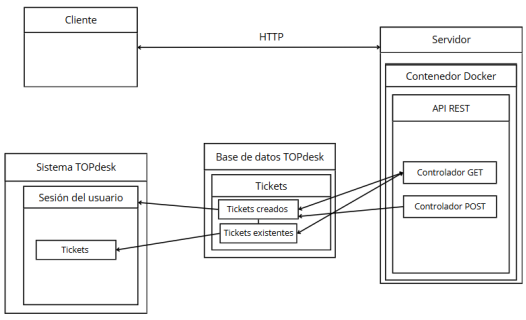

Figura 1. Arquitectura del sistema

Humanidades, Tecnología y Ciencia, del Instituto Politécnico Nacional
Certificado de Reserva de Derechos al Uso Exclusivo del Título: 04-2010-0326124-14000-203 ISSN: 2007-1957

La arquitectura sigue el estilo REST (Representational State Transfer), que se basa en los siguientes principios:

- Cliente-Servidor: separación entre la interfaz de usuario y la lógica de negocio.
- Stateless (sin estado): cada solicitud HTTP contiene toda la información necesaria para su procesamiento.
- Recursos bien definidos: cada entidad (ej. un ticket) se representa con una URL única.
- Formato estándar: uso de JSON para el intercambio de datos. La API está contenerizada dentro de un contenedor Docker, lo que permite su despliegue eficiente y portable en cualquier servidor. La arquitectura incluye al cliente, que realiza solicitudes POST o GET hacia diferentes controladores de la API, ofreciendo flexibilidad en los datos enviados.

Pruebas

Pruebas de endpoints

Se verificaron los cinco endpoints principales de la API:

- POST /v1/create: creación de tickets con parámetros básicos. Respuesta exitosa con código 200.
- POST /v2/createV2: creación con 11 parámetros predefinidos. Respuesta 200.
- POST /v3/createV3: creación con parámetros opcionales; si no se envían, se toman valores por defecto.
- GET /v1/incidents: obtención de todos los tickets existentes. Respuesta 200 con datos en JSON.
- GET /v1/incident/:id: consulta de un ticket específico por su número. Respuesta 200.

Pruebas de autorización y seguridad

Se comprobó que el sistema bloquea solicitudes sin credenciales válidas:

- Solicitud sin credenciales: retorna código 401 Unauthorized.
- Solicitud con usuario erróneo: retorna 401 Unauthorized.

Pruebas de flujo completo

Se realizó una prueba integral que incluyó:

1. Ingreso de credenciales de autorización en Postman.
2. Envío de una solicitud POST al endpoint createV2 con todos los parámetros requeridos.
3. Recepción de respuesta 200 confirmando la creación del ticket.
4. Verificación en el portal de TOPdesk de que el ticket fue registrado correctamente.

Pruebas de despliegue con Docker

Se verificó el correcto levantamiento del contenedor mediante el comando docker compose up. El comando docker ps -a confirmó que el contenedor topdesk-api se encuentra activo.

Pruebas de rendimiento

Se midieron los tiempos de respuesta de cada endpoint en condiciones normales de operación:

Tabla 1. Tiempos de respuesta por endpoint (elaboración propia).

Endpoint	Tiempo de respuesta (ms)	Evaluación
/createV1	589	Bueno/Aceptable
/createV2	673	Bueno/Aceptable
/createV3	452	Excelente
/tickets	613	Bueno/Aceptable
/tickets/{id}	338	Excelente

Resultados

Todos los tiempos se mantuvieron por debajo de 1 segundo, lo cual cumple con los criterios de rendimiento establecidos por AWS, Microsoft y OWASP para servicios que involucran operaciones externas o comunicación con otros sistemas.

La API implementa tres versiones de endpoints (v1, v2, v3) que ofrecen distintos niveles de configuración. Además, los tickets creados se reflejan correctamente en la plataforma TOPDesk.

Conclusiones

El uso de Docker y Docker Compose hizo posible que la API pueda desplegarse en cualquier entorno de manera consistente, facilitando su escalabilidad horizontal según la demanda. La solución propuesta basada en servicios REST sienta las bases para futuras expansiones, como la integración con otras plataformas de ticketing como *ServiceNow* o *Jira Service Management.*

Referencias

Delgado Olivera, L. D. (2021). Modelos de Desarrollo de Software. *Revista Cubana de Ciencias Informáticas*, 37-51. Obtenido de https://www.redalyc.org/journal/3783/378366538003/378366538003.pdf

IBM. (02 de 03 de 2021). *Reglas de negocio.* Obtenido de https://www.ibm.com/docs/es/bpm/8.5.6?topic=types-business-rules

Leonardi, M., Sampaio, J. C., & Rossi, G. (1998). Estrategias para la identificación de Reglas de Negocio. *Simpósio Brasileiro de Engenharia de Software*, 53-67. Obtenido de https://www-di.inf.puc-rio.br/~julio/leo-reg-98.pdf

ORACLE. (24 de Febrero de 2025). *¿Qué es una API (interfaz de programación de aplicaciones)?* Obtenido de https://www.oracle.com/latam/cloud/cloud-native/api-management/what-is-api/

POSTMAN. (9 de Noviembre de 2025). *Tipos de API: Una guía completa sobre arquitecturas de API.* Obtenido de https://blog.postman.com/different-types-of-apis/

Pressman, R. S., & Maxim, B. R. (2021). *Ingeniería de software.* McGraw-Hill Interamericana de España S.L.