

Sistema de Administración de Fallas

API-REST Versión 1.0

Manual Técnico

Contenido

| | |
|---------------------------------------|---|
| Requerimientos de Hardware y Software | 2 |
| Arquitectura de la aplicación | 3 |
| Servicios implementados | 4 |
| Status de la API | |
| Agregar un nuevo ticket | |
| Obtener un ticket | |
| Modificar datos de ticket | |
| Eliminar un ticket | |
| Matriz de pruebas | 9 |

Requerimientos de Hardware y Software

Para poder ejecutar correctamente la API-REST de SAF se necesitan cubrir como mínimo los siguientes requisitos:

- Hardware

- Memoria RAM de 4gb.
- Espacio disponible en disco duro de 50mb.
- Conexión a internet estable.

- Software

- NodeJS v4.16.0+

Se puede descargar del sitio oficial (<https://nodejs.org/es/>).

- NPM v6.14.11+

Actualmente ya viene incluido en la instalación de NodeJS.

- MySQL v8

Se puede utilizar un paquete integrado como wamp o xampp (recomendado) los cuales ya incluyen la instalación automática de MySQL o bien, instalar manualmente desde su sitio oficial (<https://www.mysql.com/>).

- Código fuente de la API

Se encuentra disponible en un repositorio de GitHub, se puede realizar un clonado de la rama máster, o bien, se puede descargar el código fuente en un archivo ZIP y posteriormente descomprimir.

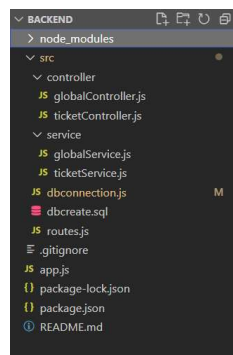
Ubicación: <https://github.com/carloscarcano/saf-backend>

Arquitectura de la aplicación

Cuando se ejecuta la API primeramente se establece por defecto el puerto **8085** para escuchar peticiones y se determinan las rutas y verbos HTTP que tendrán los servicios a través del archivo **/src/routes.js**, posteriormente, se inicializa una conexión a MySQL en el archivo **/src/dbconnection.js** para que esta sea utilizada por los servicios cuando sea requerido.

Cuando se solicita un servicio a la API este es analizado de acuerdo al archivo de rutas para entonces direccionar el flujo de ejecución al controlador correspondiente. En el controlador se realizan validaciones de los parámetros de entrada y si todo está correcto se pasa el control a la capa de servicios.

En la capa de servicios se realiza la lógica de establecer comunicación con la base de datos (haciendo uso de la conexión a MySQL que se inicializa al levantar la API), se prepara la llamada y se obtiene la respuesta de la base de datos y se prepara la información de respuesta la cual se devuelve al controlador. Finalmente, de acuerdo a lo que se haya obtenido en la capa de servicios, el controlador prepara la respuesta para el cliente y se establece el código de estado según la operación realizada (por ejemplo 500 si ocurrió algún error durante el proceso o 200 si todo se resolvió correctamente).



Estructura de carpetas del código fuente

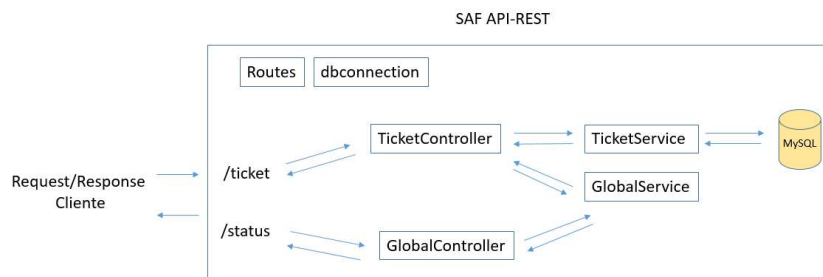


Diagrama de funcionamiento

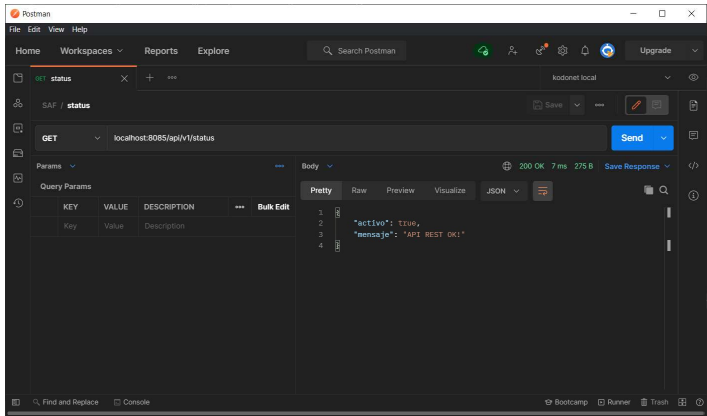
Servicios implementados

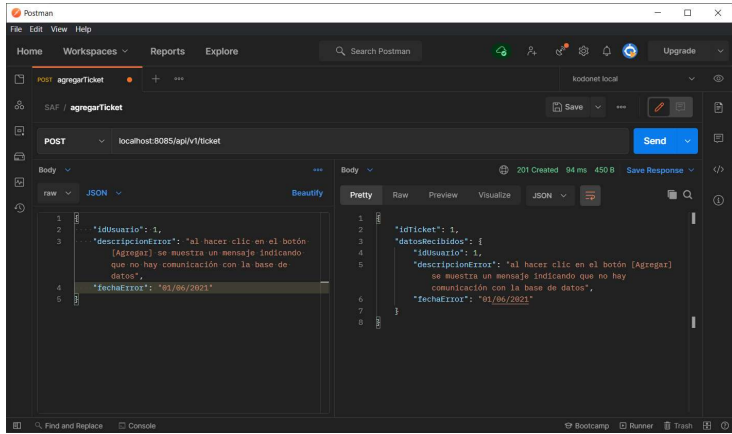
Resumen:

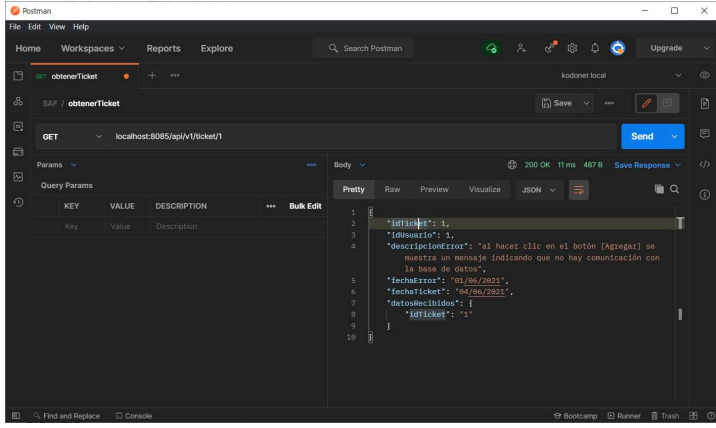
| | |
|----------------------------------|--|
| GET /api/v1/status | Saber si se encuentra en funcionamiento la API |
| POST /api/v1/ticket | Agregar un nuevo ticket |
| GET /api/v1/ticket/[idticket] | Obtener un ticket |
| PATCH /api/v1/ticket | Modificar datos de un ticket |
| DELETE /api/v1/ticket/[idticket] | Eliminar un ticket |

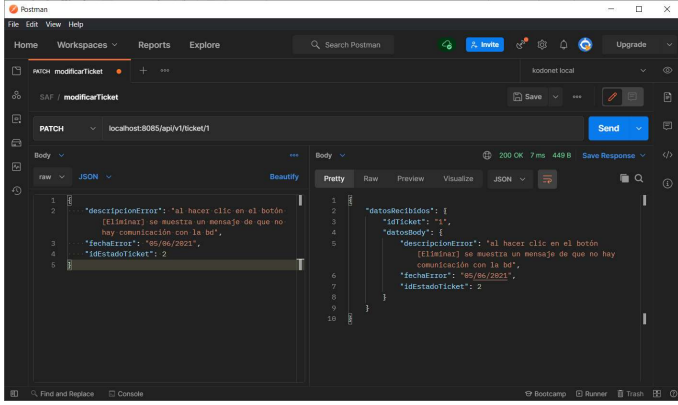
La información que se envía en el cuerpo de la petición es en formato JSON (para los casos de POST y PATCH) así como la que devuelven los servicios igualmente es en JSON.

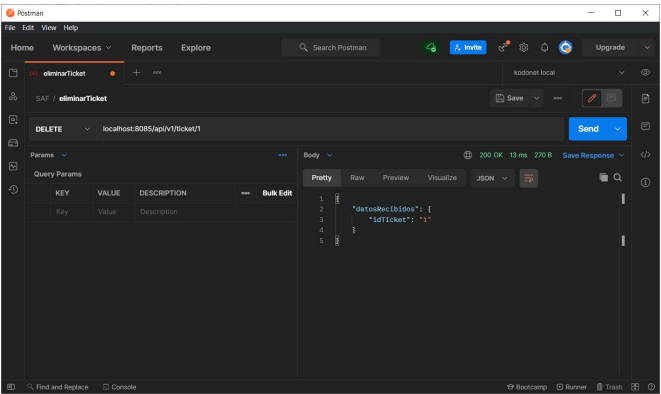
Información detallada de cada servicio:

| | |
|---------------------------|---|
| GET /api/v1/status | Saber si se encuentra en funcionamiento la API |
| Datos enviados en la URL | Ninguno. |
| Datos enviados en el Body | Ninguno. |
| Respuestas | <p>ECONNREFUSED: no se encuentra activo el servicio.</p> <p>Código 200: se encuentra escuchando peticiones.</p> <pre>{ "activo": true, "mensaje": "API REST OK!" }</pre> <p>Código 500: se encuentra activo el servicio pero hay un error en la aplicación.</p> |
| Ejemplo |  |

| | |
|----------------------------|--|
| POST /api/v1/ticket | Agregar un nuevo ticket |
| Datos enviados en la URL | Ninguno. |
| Datos enviados en el Body | <pre>{ "idUsuario": 1, "descripcionError": "descripción del error", "fechaError": "05/02/2021" }</pre> <p>idUsuario: número entero. descripcionError: cadena de caracteres, máximo 500. fechaError: dd/mm/yyyy</p> |
| Respuestas | <p>Código 201: el registro se agregó correctamente.</p> <pre>{ "idTicket": 1, "datosRecibidos": { "idUsuario": 1, "descripcionError": "al hacer clic en el botón [Agregar] se muestra un mensaje indicando que no hay comunicación con la base de datos", "fechaError": "01/06/2021" } }</pre> <p>idTicket: número asignado en la tabla. datosRecibidos: datos enviados en la url y/o body.</p> <p>Código 500: si se encuentra activo el servicio pero hay un error en la aplicación.</p> |
| Ejemplo |  |

| | |
|--------------------------------------|---|
| GET /api/v1/ticket/[idticket] | Obtener un ticket |
| Datos enviados en la URL | El ID del ticket que se quiere obtener. |
| Datos enviados en el Body | Ninguno. |
| Respuestas | <p>Código 200: se encontró el ticket solicitado.</p> <pre>{ "idTicket": 1, "idUser": 1, "descripcionError": "al hacer clic en el botón [Agregar] se muestra un mensaje indicando que no hay comunicación con la base de datos", "fechaError": "01/06/2021", "fechaTicket": "04/06/2021", "datosRecibidos": { "idTicket": "1" } }</pre> <p>Código 404: no existe el ticket con el id especificado.</p> <pre>{ "datosRecibidos": { "idTicket": "18" } }</pre> <p>Código 500: ocurrió un error inesperado.</p> |
| Ejemplo |  |

| PATCH /api/v1/ticket/[idticket] | Modificar datos de un ticket |
|---------------------------------|---|
| Datos enviados en la URL | El ID del ticket que se quiere obtener. |
| Datos enviados en el Body | <pre data-bbox="676 371 1356 526">{ "descripcionError": "al hacer clic en el botón [Eliminar] se muestra un mensaje de que no hay comunicación con la bd", "fechaError": "05/06/2021", "idEstadoTicket": 2 }</pre> <p data-bbox="676 548 1356 616">Solo permite modificar tres datos: descripcionError, fechaError e idEstadoTicket.</p> <p data-bbox="676 660 1356 694">Los datos que se envíen son los que se modifican.</p> <p data-bbox="676 739 1356 772">descripcionError: cadena de caracteres, máximo 500.</p> <p data-bbox="676 779 1356 813">fechaError: dd/mm/aaaa.</p> <p data-bbox="676 819 1356 920">idEstadoTicket: número entero entre 1 y 4 (1 = pendiente, 2 = en proceso, 3 = atendido, 4 = descartado).</p> |
| Respuestas | <p data-bbox="676 931 1356 999">Código 200: se modificaron correctamente los datos que se enviaron.</p> <pre data-bbox="676 1021 1356 1310">{ "datosRecibidos": { "idTicket": "1", "datosBody": { "descripcionError": "al hacer clic en el botón [Eliminar] se muestra un mensaje de que no hay comunicación con la bd", "fechaError": "05/06/2021", "idEstadoTicket": 2 } } }</pre> <p data-bbox="676 1344 1356 1379">Código 500: ocurrió un error inesperado.</p> |
| Ejemplo |  |

| | |
|---|--|
| DELETE /api/v1/ticket/[idticket] | Eliminar un ticket |
| Datos enviados en la URL | El ID del ticket que se quiere obtener. |
| Datos enviados en el Body | Ninguno. |
| Respuestas | <p>Código 200: se eliminó correctamente el ticket.</p> <pre>{ "datosRecibidos": { "idTicket": "1" } }</pre> <p>Código 404: no existe un ticket con el id especificado.</p> <p>Código 500: ocurrió un error inesperado.</p> |
| Ejemplo |  <p>The screenshot shows the Postman interface for a DELETE request. The URL is localhost:8085/api/v1/ticket/1. The response status is 200 OK, and the response body is a JSON object: {"datosRecibidos": {"idTicket": "1"}}.</p> |