Implementación del backend de una aplicación web para el mantenimiento de infraestructura y equipo en el ITSM

Ing. Rogelio Ramírez Silva
Estudiante de Maestría en Sistemas Computacionales programa PNPC
Tecnológico Nacional de México / IT de Acapulco
Acapulco Gro., México
rrasilva18@gmail.com

M.C. Francisco Javier Gutiérrez Mata
Docente de Maestría en Sistema Computacionales
Tecnológico Nacional de México / IT de Acapulco
Acapulco Gro., México
fcmata84@hotmail.com

M.I.D.S. Alma Delia De Jesús Islao
Docente de Maestría en Sistema Computacionales
Tecnológico Nacional de México / IT de Acapulco
Acapulco Gro., México
alma.islao.ita@gmail.com

M.C. Eloy Cadena Mendoza
Docente de Maestría en Sistema Computacionales
Tecnológico Nacional de México / IT de Acapulco
Acapulco Gro., México
eloy_cadena@yahoo.com

Ing. José Raúl López Morales
Estudiante de Maestría en Sistemas Computacionales programa PNPC
Tecnológico Nacional de México / IT de Acapulco
Acapulco Gro., México
jraul_lopz@hotmail.com

***Resumen*— En el presente artículo, se plasma el trabajo interdisciplinario de la Maestría en Sistemas Computacionales con apoyo del CONACyT, impartida en el Instituto Tecnológico de Acapulco. El objeto de este artículo es, presentar la implementación de la herramienta informática SISCMIE, la cual es un sistema informático destinado a dar seguimiento y control del mantenimiento de infraestructura y equipo en el Instituto Tecnológico de San Marcos (ITSM), programada la parte del backend en NodeJS en una versión web que permite a los usuarios tener acceso desde cualquier punto geográfico con acceso a internet. Este artículo forma parte de una serie de trabajos siendo el segundo de estos, el cual da seguimiento a un artículo previo titulado: Propuesta de diseño de un sistema de información web para el seguimiento y control del mantenimiento de infraestructura y equipo [4].**

Palabras clave*— *NodeJS, Backend, Mantenimiento, MVC.

I. Introducción

Las tecnologías para el desarrollo de aplicaciones web, móviles y de escritorio han aumentado considerablemente, surgiendo herramientas que ayudan a la industria del desarrollo de software profesional, actualmente las tecnologías para el desarrollo de aplicaciones web se dividen en dos grandes intereses, los cuales son, Backend y Frontend.

En el backend se ejecutan las tecnologías que por lo general realizan el trabajo más pesado como el de procesar la información y persistirla en la base de datos, algunas de las tecnologías backend son, por mencionar algunas; Asp.Net Core, Ruby, Java y JavaScript. Las tecnologías ejecutadas del lado del Frontend son herramientas que se utilizan para renderizar la información almacenada en las bases de datos y que ayudan al usuario final a tener una experiencia agradable al introducir o extraer información.

Para la implementación del backend de la herramienta mencionada en este artículo se hará uso del framework NodeJS, el cual es un entorno JavaScript de código abierto que le permite ejecutar JavaScript en el lado del servidor. Esto permite utilizar el mismo lenguaje para ejecutar código tanto en el cliente como ahora en el servidor. NodeJS utiliza una colección de módulos. Los módulos conforman la funcionalidad central de NodeJS y le permiten hacer cosas como trabajar con el sistema de archivos, con protocolos de red (HTTP, TCP, DNS), datos binarios y la capacidad de comunicarse con la base de datos, es importante mencionar que fue diseñado para construir aplicaciones en red escalables [1].

El lenguaje que se utiliza para el desarrollo de la implementación es JavaScript el cual es un lenguaje de script interpretado que utilizando NodeJS se obtiene la capacidad de hacer cosas como hacer solicitudes desde una base de datos y responder a solicitudes HTTP y crear archivos. Otra característica muy importante por mencionar del lenguaje es que en tiempo de ejecución es multiplataforma, asíncrono, con E/S de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google [2].

La base de datos utilizada para persistir la información del sistema es mongodb, el cual almacena datos en documentos flexibles similares a JSON, haciendo uso de este tipo de bases de datos NoSQL que rompen los paradigmas de las bases de datos relacionales las cuales dictan ciertas características que

Consejo Nacional de Ciencia y Tecnología

debe cumplir una base de datos, es importante resaltar que la información almacenada en las bases de datos NoSQL se recuperan mucho más rápido que una base de datos relacional [3].

II. Objetivo General

El objetivo principal que se pretende lograr con la implementación del software SISCMIE es gestionar la información de las anomalías y fallos que presentan los equipos e infraestructuras del ITSM.

III. Implementación del Backend de los Módulos de SISCMIE

A partir de esta sección, se presenta la implementación de los módulos del sistema que trata este artículo, en la figura 1 se observa el diagrama de clases, el cual da soporte al desarrollo de SISCMIE haciendo uso de la tecnología NodeJS. Las clases en las cuales se basó el desarrollo de la implementación son las siguientes: Catalogo, Articulo, Ticket, DetalleTicket, Plan.

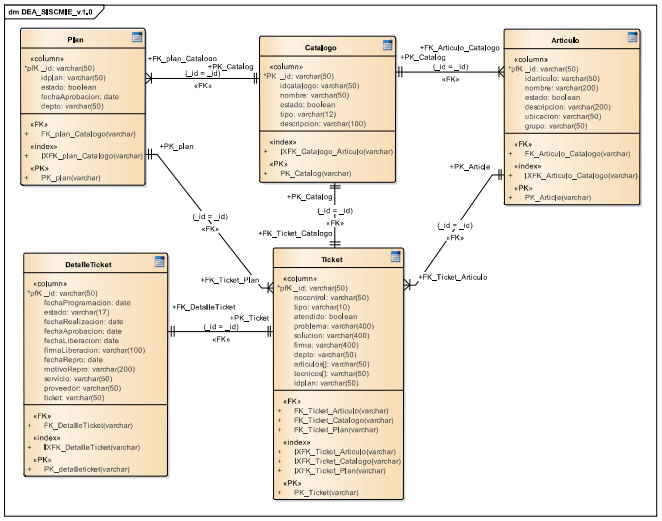

Fig. 1. Diagrama de clases de SISCMIE.

La primera funcionalidad que se implementó en el API REST (Aplicación ejecutada en el servidor) de SISCMIE con NodeJS fue la de dar de alta departamentos, servicios, proveedores y grupos que se utiliza para clasificar los equipos e infraestructuras por tipo.

A. *Módulo artículo*

Este módulo gestiona la parte de los artículos que el sistema administra. A continuación, se presenta la implementación del método para agregar un artículo al sistema, siendo este uno de los más importantes a mencionar debido a que si un artículo no existe en la base de datos no se puede realizar ninguna otra operación sobre sus datos. Las demás funcionalidades que permiten actualizar, buscar y eliminar, si fueron implementadas, aunque en el sistema no se eliminan artículos, más bien cambian de estado, los cuales pueden ser dos, activo o inactivo y al realizar esta operación se utiliza el método para actualizar. En las siguientes líneas de código se muestran la funcionalidad de agregar.

En forma local se envían los datos a la siguiente dirección http://localhost/article, los datos son enviados utilizando la aplicación postman para simular un cliente ingresando datos al sistema.

```
app.post('/article', (req, res) => {
    var body = req.body;

    var articulo = new Articulo({
        idarticulo: body.idarticulo,
        nombre: body.nombre,
        estado: body.estado,
        descripcion: body.descripcion,
        ubicacion: body.ubicacion,
        grupo: body.grupo
    });
    articulo.save((err, articuloGuardado) => {
       if(err) {
          return res.status(400).json({
            ok: false,
            mensaje: 'Error creando artículo',
            errors: err
            });
       }
       res.status(201).json({
            ok: true,
            articulo: articuloGuardado
       });
    });
});
```

Como se observa en el código anterior se obtiene a través de la URL un objeto de tipo JSON en el mismo orden que lo dicta el Objeto de tipo Articulo, los datos son recibidos a través del protocolo HTTP y almacenados en la base de datos, si hubiera un error este sería tratado y enviado al cliente que consume el backend, avisando del error ocurrido.

B. *Módulo ticket*

El módulo de ticket permite gestionar las solicitudes de mantenimiento correctivo que los jefes de cada departamento del ITSM realizan, también es importante mencionar que este mismo módulo se utiliza para realizar el procedimiento de verificación de infraestructuras que se menciona en el primer artículo [4].

Para este módulo se implementaron cuatro métodos que corresponden a un CRUD (Create, Read, Update y Delete) que se utilizan para crear tickets, obtener tickets de la base de datos, actualizar tickets y eliminar tickets. Los cuales se pueden acceder a través de los endpoint o métodos que brindan la funcionalidad dependiendo del verbo (post, put, get y delete)

que brinda el API REST. El endpoint para acceder a estas funcionalidades ejecutando la aplicación de forma local es http://localhost:3000/ticket.

```
app.post('/ticket', (req, res) => {
    var body = req.body;

    var ticket = new Ticket({
    });
    ticket.save((err, ticketGuardado) => {
        if(err) {
            return res.status(400).json({
               ok: false,
               mensaje:'Error creando ticket',
               errors: err
            });
        }
        res.status(201).json({
            ok: true,
            ticket: ticketGuardado
        });
    });
});
```

En las líneas de código anterior muestra el endpoint para crear tickets, el cual recibe a través del protocolo HTTP el cuerpo de tipo JSON con la información a persistir en la base de datos los datos son obtenidos del cuerpo de la petición y mapeados al objeto de tipo Ticket y posteriormente con el método save se guarda la información, en caso de existir un error al guardar la información se envía un error de tipo 400 (se utiliza para indicar al usuario que no está enviando la información correctamente) con un mensaje indicando que ocurrió un error al guardar el ticket y si la persistencia de la información se realiza con éxito, se envía la información que se almacenó en formato JSON para que la persona que realizó la operación realice la tarea que sea necesaria.

Cuando se realiza una verificación que es un procedimiento que se lleva a cabo en la revisión de infraestructuras y equipo del cual se habla en el primer artículo [4], se obtienen un conjunto de datos los cuales son iguales a si realizaras una solicitud de mantenimiento correctivo, es por eso que se optó por utilizar el método de ticket para realizar este procedimiento y almacenarlo en la misma tabla, la única diferencia es que, si se realiza una verificación, el campo tipo, almacena un valor indicando que es una verificación y si es una solicitud se almacena el valor solicitud, posteriormente se reprograma la fecha en que se realizará dicho trabajo y es ahí donde se utiliza el método que se llama DetalleTicket el cual me permite almacenar un detalle de dicha solicitud o verificación.

```
app.post('/', (req, res) => {
   var body = req.body;

   var detalleticket = new Ticket({
      fechaProgramada: body.fechaProgramada,
      estado: body.estado,
    fechaRealizacion: body.fechaRealizacion,
    fechaAprobacion: body.fechaAprobacion,
    firmaAprobacion: body.firmaAprobacion,
    fechaLiberacion: body.fechaLiberacion,
    firmaLiberacion: body.firmaLiberacion,
    fechaReprogramada: body.fechaReprogramada,
  motivoReprogramada: body.motivoReprogramada,
  servicio: body.servicio,
  proveedor: body.proveedor,
  ticket: body.ticket
 });

detalleticket.save((err,detalleticketGuardado)
 => {
    if(err) {
     return res.status(400).json({
      ok: false,
      mensaje: 'Error al crear detalleticket',
      errors: err
    });
    }

     res.status(201).json({
         ok: true,
         detalleticket: detalleticketGuardado
        });
    });
});
```

El código mostrado anteriormente corresponde a la funcionalidad que permite agregar detalle a las solicitudes o verificaciones de infraestructuras, los datos que envía el cliente son la fecha en que se programa la atención de la anomalía encontrada o mantenimiento correctivo a resolver, así como el servicio que se realizará, el proveedor que atenderá el servicio si es externo o interno y el identificador único de la solicitud que le corresponde dicho detalle.

La información es recibida por el servidor y almacenada en caso de ser correcta, en caso de que suceda un error al intentar almacenar la información, el servidor enviará un mensaje indicando el error al cliente que realiza la operación en el API REST NodeJS.

C. Módulo plan

La implementación del módulo plan consiste en gestionar información de la base de datos, como estado del plan, los cuales pueden ser en proceso o cerrado, fecha de aprobación del plan y departamento que creó el plan, esta información está relacionada con el módulo de solicitudes, debido a que en el proceso de verificación se hacen solicitudes de los mantenimientos que se realizarán como parte del plan de mantenimiento preventivo que se realizarán durante el semestre en curso en el ITSM. La finalidad de este módulo es asignarle solicitudes de mantenimiento preventivo que posteriormente se le asignarán fecha para ser realizadas.

Los métodos implementados para este módulo son: creación, obtención, actualización y eliminación. La dirección en modo local para acceder a dichos métodos es http://localhost:3000/plan, haciendo uso de esta dirección se puede manipular la información de los planes que son creados y están en proceso.

El código mostrado a continuación permite crear un plan, donde los datos se obtienen a través del protocolo HTTP enviados con el verbo post desde el cliente, la información se obtiene del lado del API REST en NodeJS y se almacena si no existe ningún error y si existe algún problema se notifica al cliente que está realizando la operación y se envía un error 400 el cual indica que los datos no se están enviando correctamente.

```
app.post('/plan', (req, res) => {
    var body = req.body;

    var plan = new Plan({
        idplan: body.idplan,
        estado: body.estado,
        fechaAprobacion: body.fechaAprobacion,
        depto: body.depto
    });
    plan.save((err, planGuardado) => {
        if(err) {
            return res.status(400).json({
                ok: false,
                mensaje: 'Error creando plan',
                errors: err
            });
        }
        res.status(201).json({
            ok: true,
            ticket: planGuardado
        });
    });
});
```

IV. Resultados

En este apartado se presenta el resultado de la implementación del API REST en NodeJS.


```
{
  "ok": true,
  "articulo": {
       "estado": false,
       "_id": "5e73fbc7e27120352c4ab6e8",
       "idarticulo": "54321",
       "nombre": "Impresora Epson",
       "descripcion": "Color Negro, Modelo ep-
       23, Monocromático",
       "ubicacion":"5e6440733fac904658196b63",
       "grupo":"5e6a6d2362444faa94aaef2b"
  }
}
```


Para hacer las peticiones al Backend se utilizó la aplicación postman para no tener que desarrollar un cliente que consuma los datos por el momento. Como se observa en el código mostrado anteriormente, al crear un artículo el servidor devuelve los datos que se almacenaron en la base de datos, los datos mostrados ya están listos para ser utilizados por cualquier cliente que quiera hacer uso de ellos, es decir, desarrolladores para una aplicación móvil u otra aplicación web que requiera información de los equipos almacenados.


```
{
 "ok": true,
 "ticket": {
   "estado": "ABIERTA",
   "articulos": ["5e583f4898f23361a414bfa4"],
   "_id": "5e655b94f8501f75d0f6e582",
   "idticket": "1",
   "tecnico": "5b3440083fbc905458326b5f",
   "problema": "Reparación de memoria RAM",
   "deptoDestino": "5e6440083fac904658196b5f",
   "deptoOrigen": "5e64403f3fac904658196b60",
   "Aprobacion": "2020-03-08T20:54:44.361Z",
   "createdAt": "2020-03-08T20:54:44.390Z",
   "updatedAt": "2020-03-08T20:54:44.390Z",
  }
}
```


La información en formato JSON que se muestran a continuación, es el resultado devuelta por el servidor SISCMIE al crear un plan, a partir de ese momento la información puede ser mostrada en tablas o manipula de la manera que mejor convenga.


```
{
  "ok": true,
  "plan": {
     "estado": false,
     "_id": "5e66d1890f950d4d60aeb27f",
     "periodo": "ENE-JUN2020",
     "createdAt": "2020-03-09T23:30:17.820Z",
     "updatedAt": "2020-03-19T23:12:53.176Z",
  }
}
```


Las líneas en formato JSON mostradas a continuación, corresponde a los datos devueltos cuando se ingresan información al catálogo general del sistema, en este catálogo se registran departamentos, proveedores, grupos y servicios, en este caso se registró un departamento.


```
{
  "ok": true,
  "catalogo": {
     "estado": true,
     "tipo": "DEPARTAMENTO",
     "_id": "5e66d1680f951f4f60gfb265f",
     "idcatalogo": depto-20,
     "nombre": "Recursos Materiales",
     "descripcion": "Jefatura de materiales",
     "createdAt": "2020-03-09T22:30:17.920Z",
     "updatedAt": "2020-03-19T22:12:53.156Z",
  }
}
```

Los campos de los departamentos, proveedores, grupos y servicios son los mismos, lo único que cambia en cada uno de ellos son los datos que se registran, es por eso que se optó por implementarlo de esta forma, es decir, diferenciarlos por un campo llamado tipo.

V. CONCLUSIONES

En este artículo, se presentó la implementación del backend de un sistema de información web para el seguimiento y control del mantenimiento de infraestructura y equipo, que se encuentra en su etapa de desarrollo, haciendo uso de las tecnologías JavaScript y NodeJS, los cuales son tecnologías fáciles de utilizar y de fácil acceso, pero, sobre todo, tecnologías de uso libre, es decir, con licencias que permiten desarrollar aplicaciones que no te penalizan por hacer uso personal o comercial con productos finales desarrollados con dichas herramientas. NodeJS es una tecnología con la que se pueden desarrollar servicios web rápidos y fáciles, es decir, levantar un servicio se realiza en un tercio del tiempo que llevaría realizarlo con otras tecnologías como Asp.Net Core o Java. Es importante recalcar que NodeJS es una tecnología que tiene buena seguridad y que es respaldada por NPM una biblioteca de librerías online que da soporte a dicho framework.

Hasta este punto la información ya se puede utilizar para hacer pruebas o en su defecto también se puede hacer uso de dicha información para consumirla con una aplicación móvil y desarrollar aplicativos que realicen otras funciones como consultar información de solicitudes pendientes o notificar que no se ha realizado una solicitud que está atrasada, a partir de este punto ya se puede comenzar a utilizar el sistema y darle el rumbo que el cliente considere apropiado. Es importante mencionar que toda la información almacena está protegida haciendo uso de web tokens, es decir, si alguna otra persona ajena el ITSM quisiera consumir la información del sistema esta debe proporcionarles la clave con la que se desencripta el token que se envía en cada petición que se realiza para realizan cualquier operación sobre el API REST como Crear, Actualizar, Obtener o Eliminar datos de dicho sistema.

REFERENCIAS

[1] F. Russ, Beginning JavaScript. The Ultimate Guide to Modern JavaScript Development*: Apress 2019*.

[2] K. Manuel. ayllusolar. [En línea] 29 de 1 de 2015. [Citado el: 3 de 3 de 2020.] http://ayllusolar.cl/wp-content/uploads/2016/08/node_js_Guia_Principiantes.pdf.

[3] Zorrilla, Marta. Universidad de Cantabria. [En línea] 3 de 2017.[Citado el: 3 de 3 de 2020.] https://ocw.unican.es/pluginfile.php/2396/course/section/2473/NoSQL_Tema2_MongoDB.pdf.

[4] R. Ramírez Silva, F. J. Gutiérrez Mata, E. De la Cruz Gámez y E. Cadena Mendoza, «Propuesta de diseño de un sistema de información web para el seguimiento y control del mantenimiento de infraestructura y equipo» vol. 11, n°2, 2019.