Proyecto rutinas mantenimiento

**2019**

Héctor Josué Ardón Morga

Corporación Centroamericana de Servicios de Navegación Aérea

22/11/2019



Contenido

[***Introducción*** 2](#_Toc25698431)

[***Base de datos MySQL*** 3](#_Toc25698432)

[***Datos almacenados en las tablas MySQL*** 3](#_Toc25698433)

[***Base de datos Mongo DB*** 17](#_Toc25698434)

[***Datos almacenados en base de datos Mongo DB*** 23](#_Toc25698435)

[***Estructura del proyecto*** 27](#_Toc25698436)

[***Carpeta controladores*** 27](#_Toc25698437)

[***Carpeta routes*** 28](#_Toc25698438)

[***Carpeta public*** 30](#_Toc25698439)

[***Carpeta views*** 31](#_Toc25698440)

[***Casos de uso*** 31](#_Toc25698441)

[***Propuesta diseño del Front End*** 32](#_Toc25698442)

[***Glosario de términos*** 34](#_Toc25698443)

# Introducción

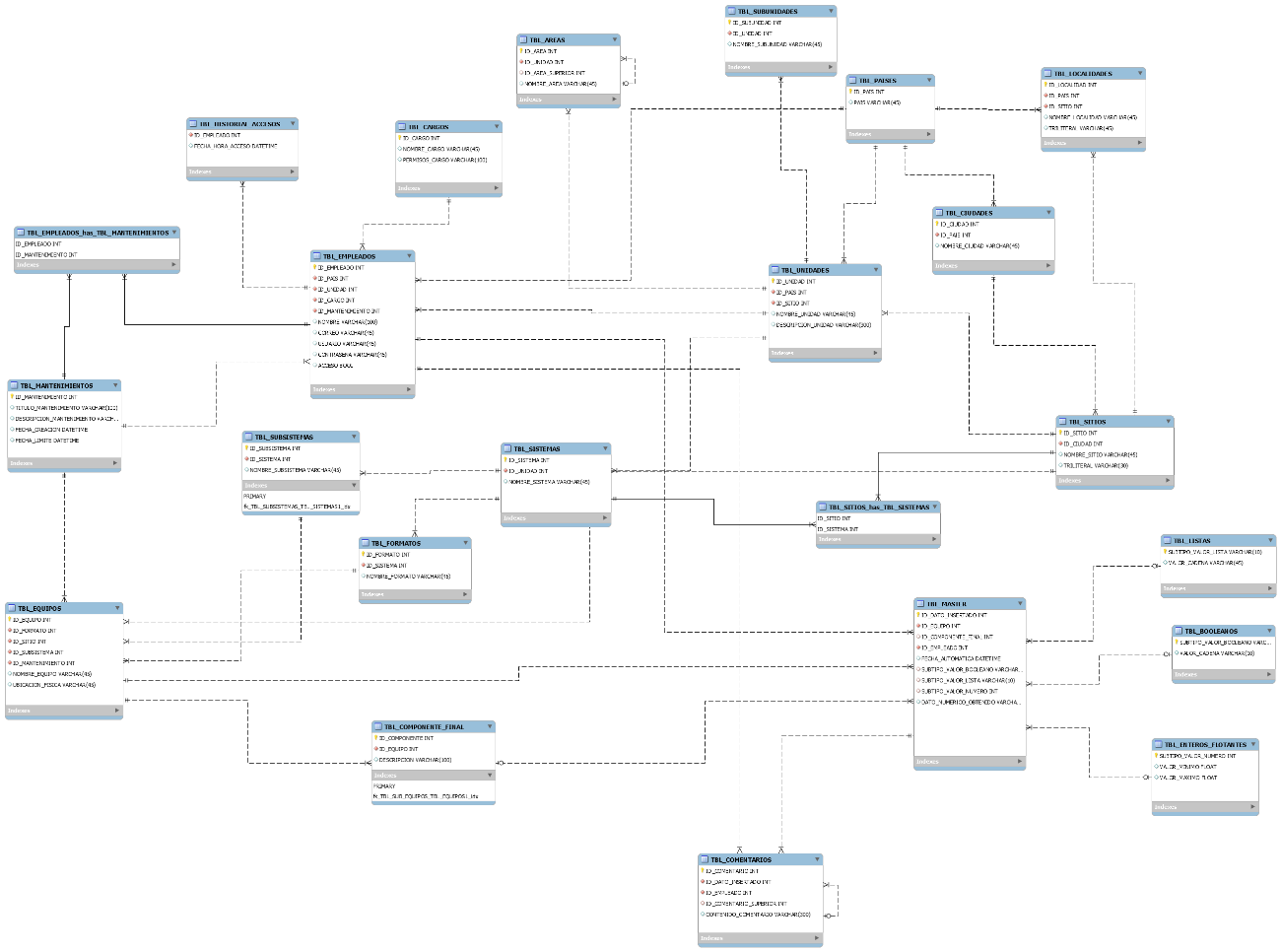
El presente proyecto describe algunas características utilizadas para el desarrollo de este proyecto, el cual es para la empresa COCESNA. El desarrollo de este proyecto describe algunas cosas elaboradas hasta la fecha, el cual nos ha servido para comenzar a evolucionar aspectos que son importantes de implementar en la empresa, el hecho de comenzar a automatizar procesos y disponiendo de herramientas que contribuyen al desarrollo de la misma y del país.

Se han utilizado herramientas de desarrollo web para implementar algunas necesidades en la empresa, cabe destacar que un proyecto de programación, no está 100% terminado, los proyectos deben ir escalando a medida que pasa el tiempo, pero dentro de lo respectivo se va a tratar de explicar aspectos importantes del mismo.

Se han hecho visitas a los centros de mantenimiento recolectando datos útiles para la lógica de la base de datos

# Base de datos MySQL

Nos hemos visto en la necesidad de evolucionar la forma en la que se desarrollan los mantenimientos, elaborándolos en una plataforma digital, para esto hemos elaborado el siguiente modelo de base de datos My SQL.



# Datos almacenados en las tablas MySQL

Las tablas que considero más importantes que cubren las verdaderas funcionalidades del proyecto son las ***tablas de mantenimientos, equipos, componente\_final y master*,** a continuación, muestro datos almacenados en cada tabla de la base de datos, cabe destacar que los datos almacenados se encuentran en un archivo llamado **DML\_V2**

**Tabla de cargos**

El objetivo de esta tabla es enumerar la lista de cargos que hay en la empresa, pueden haber muchos empleados que tengan ese cargo, pero un empleado exclusivamente tiene asignado un rol o cargo en la empresa.

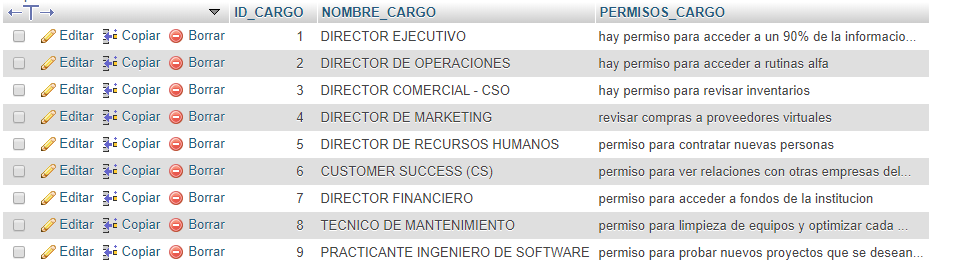


Tabla países

En esta tabla almacenamos cada uno de los países con los cuales nuestra empresa COCESNA tiene relaciones.

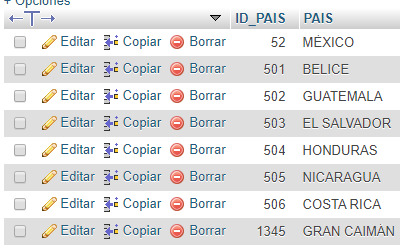


Tabla ciudades

Cada país puede tener muchas ciudades, pero una ciudad le pertenece a un país



Tabla sitios

Cada ciudad puede tener muchos sitios, y un sitio le pertenece a una ciudad.

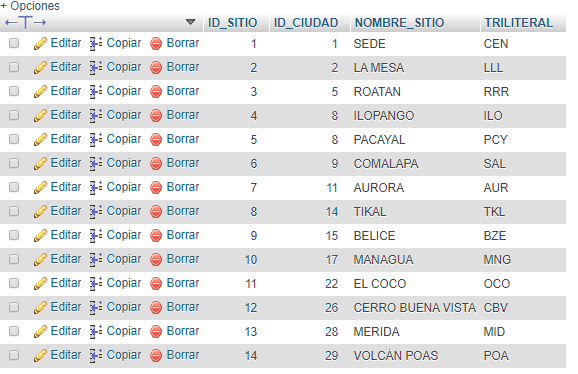
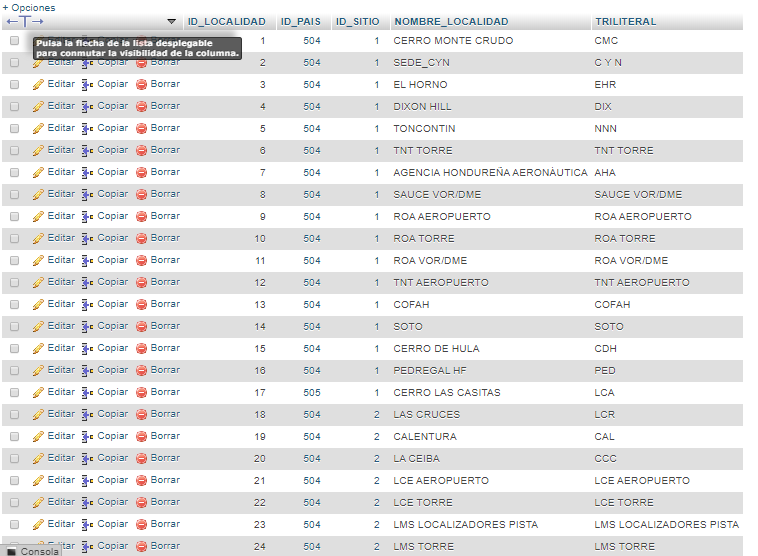


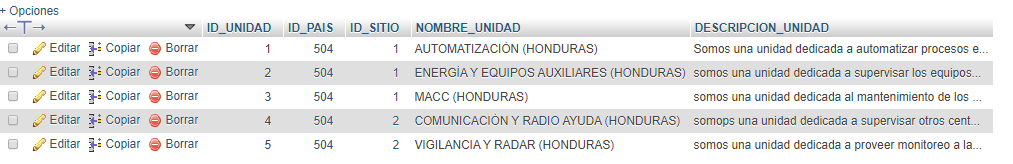
Tabla localidades



Hay más localidades disponibles, la razón por la que dejo capturas de la forma en la que se distribuyen los datos, es para que el usuario no tenga que correr todo el script, sino que aquí en la documentación vea las explicaciones, y al momento de correr el script DML, ya tener un conocimiento previo de la lógica de la base de datos.

Tabla unidades

En esta tabla se almacenan unidades, las unidades dependen de un país, ciudad y sitio, en este caso deje solamente la relación entre la tabla unidades y la tabla sitios,



si se desea saber a qué ciudad pertenece esa unidad, y a qué país pertenece, se debe utilizar una consulta como la siguiente.

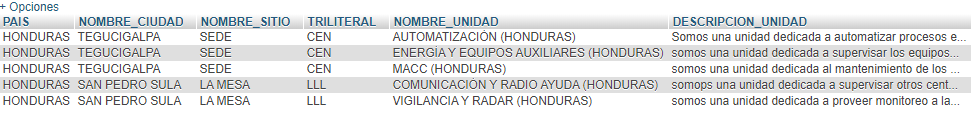
*SELECT A.PAIS, B.NOMBRE\_CIUDAD, C.NOMBRE\_SITIO, C.TRILITERAL, D.NOMBRE\_UNIDAD, D.DESCRIPCION\_UNIDAD*

*FROM TBL\_PAISES A INNER JOIN TBL\_CIUDADES B ON A.ID\_PAIS = B.ID\_PAIS*

*INNER JOIN TBL\_SITIOS C ON B.ID\_CIUDAD = C.ID\_CIUDAD*

*INNER JOIN TBL\_UNIDADES D ON C.ID\_SITIO= D.ID\_SITIO*

Esta consulta nos devuelve la siguiente información



Nos devuelve el país, la ciudad y el sitio al cual pertenecen esas unidades, en este caso solamente hay unidades para el país Honduras.

**Tabla áreas**

Esta tabla es porque una unidad puede tener áreas, incluso puede haber subáreas, por eso la tabla tiene relación consigo misma.

Los datos se observan de la siguiente forma



Tabla mantenimientos

Esta tabla la considero muy importante, en esta tabla se almacenan los títulos de los mantenimientos, un mantenimiento como tal se le puede hacer a muchos equipos, y cada equipo tiene diferentes formatos, ya que no todos los equipos ocupan un mantenimiento diario, algunos ocupan mantenimientos semanales, otros mensuales y por último trimestrales.

Lo que se almacena en esta tabla es lo siguiente



Tabla empleados

En esta tabla se almacena a las personas que están autorizadas para hacer estos mantenimientos, cada empleado está asignado a una unidad, una forma si desean saber ¿Dónde pertenece ese empleado?, es utilizando la siguiente consulta, la cual detalla el país, ciudad y sitio al que pertenece ese empleado.

*SELECT A.PAIS, B.NOMBRE\_CIUDAD, C.NOMBRE\_SITIO, C.TRILITERAL, D.NOMBRE\_UNIDAD, E.NOMBRE, E.CORREO*

*INNER JOIN TBL\_CIUDADES B ON A.ID\_PAIS = B.ID\_PAIS*

*INNER JOIN TBL\_SITIOS C ON B.ID\_CIUDAD = C.ID\_CIUDAD*

*INNER JOIN TBL\_UNIDADES D ON C.ID\_SITIO = D.ID\_SITIO*

*INNER JOIN TBL\_EMPLEADOS E ON D.ID\_UNIDAD = E.ID\_UNIDAD*

Esta consulta devuelve el siguiente resultado

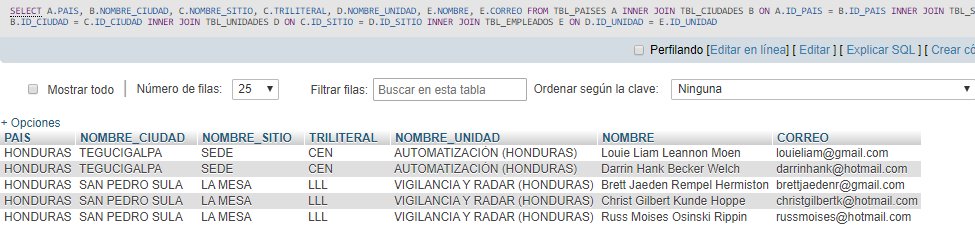


Tabla sistemas

En esta tabla se almacenan ejemplos de sistemas que son de acuerdo a los requerimientos planteados, la tabla almacena datos como estos.



Tabla sitios\_has\_tbl\_sistemas

Un sitio puede tener muchos sistemas y un sistema le puede pertenecer a muchos sitos, por ejemplo el sitio de la SEDE puede tener los sistemas AIRCON, VSAT y VOR/DME, y estos sistemas pueden ser administrador por más sitios, por ejemplo el AIRCON puede ser administrado por la SEDE, LA MESA e ILOPANGO por decirlo así.

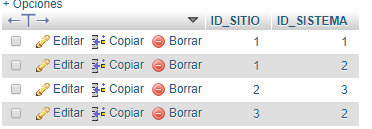
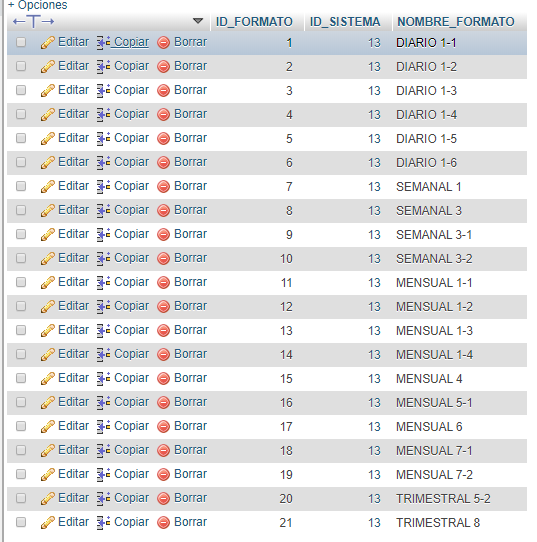


Tabla formatos

En esta tabla se almacenan cada uno de los formatos con los que son administrados los equipos, los formatos son diarios, semanales, mensuales y trimestrales, la razón por la que hay más de cada uno es para hacer un poco más fácil el manejo, por ejemplo del formato diario hay 6 paginas diferentes por decirlo así y cada pagina tiene diferentes equipos, de igual forma con el resto de formatos.



En este caso se dice que todos esos formatos le pertenecen al sistema AIRCON, que es con el sistema que trabajamos en la bitácora.

Tabla subsistemas

Cada equipo pertenece a un equipo superior por decirlo así, por ejemplo un subsistema es SDD, y sus equipos son SDD1, SDD2, SDD3,…,SDD1, son 11 equipos que tiene el subsistema SDD.

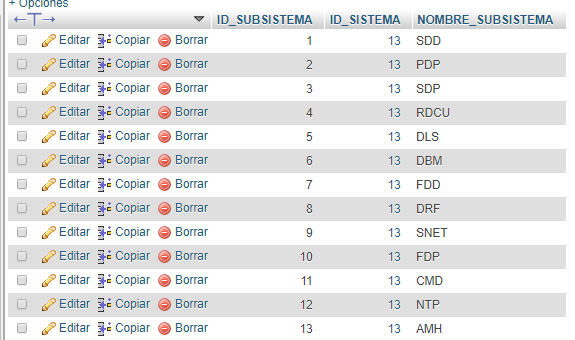


Tabla equipos

Esta tabla es fundamental, porque lo que se almacena en esta tabla es a lo que se desea hacerle el mantenimiento, son bastantes equipos, así que solo mostraré una parte de ellos, la parte inicial de los equipos que se insertaron.

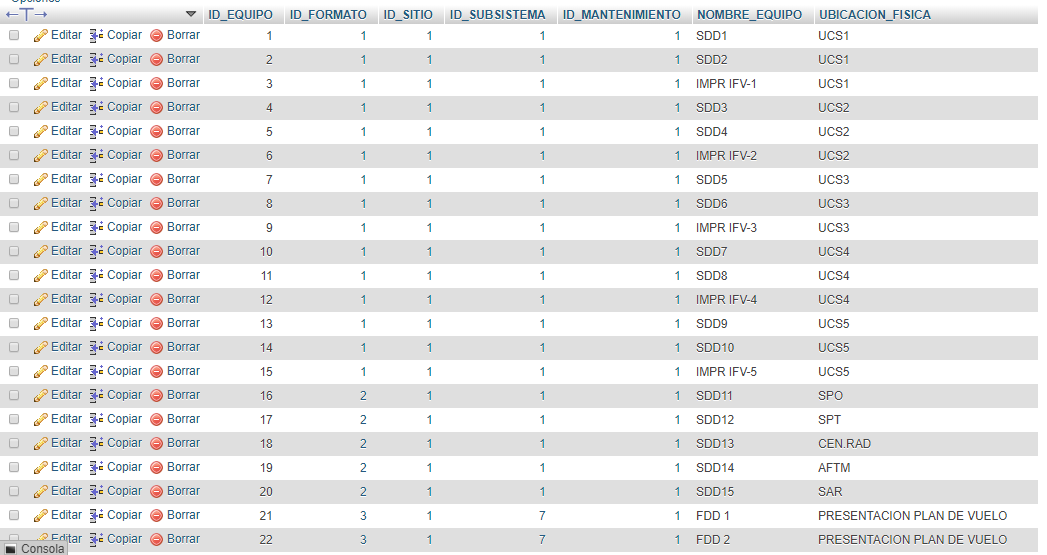
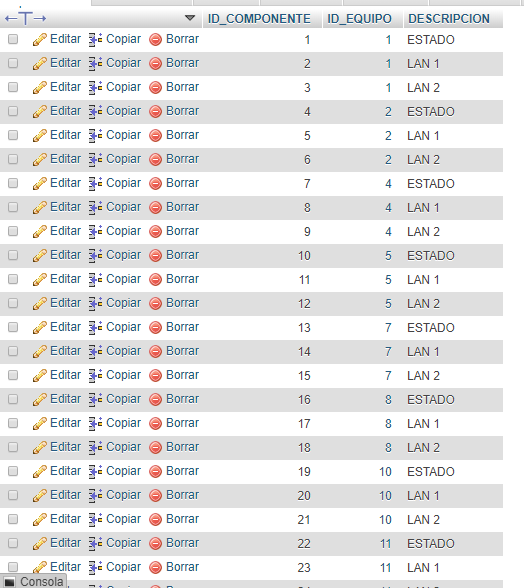


Tabla componente\_final

Cada equipo se puede descomponer en equipos más pequeños, se eligio el nombre de componente final, por ejemplo, la SDD1 tiene como componentes finales, lo que es estado, lan1, y lan2 y así hay muchos equipos más, de hecho esta es la tabla más larga de toda la base de datos.



Quizás es cierto haya mucha repetición, lo que pasa es que pertenecen a equipos diferentes, y pueden haber muchos equipos que tengan las mismas componentes, por ejemplo pueden haber muchas computadoras portátiles y como todos sabemos todas las computadoras tienen pantalla, teclado, mouse y otros elementos, entonces por eso es que hay mucha repetición aquí, porque esos componentes le pertenecen a equipos distintos.

Tabla booleanos

En esta tabla se almacenan valores booleanos, que ya deben estar disponibles al hacer el mantenimiento sobre un equipo, se toma la clave primaria y se usa como foránea en la tabla master para saber cual dato booleano se insertó, se usa notación **SUBTIPO-VALOR**, tal como el nombre de la columna, se indica que numero de booleano es, y que numero de valor es, como sabemos los valores booleanos solo tienen dos posibles estados, eso se muestra en la siguiente imagen.



Para encontrar de forma fácil a algún valor booleano, entre los miles que pueden haber, se planteó la siguiente consulta.

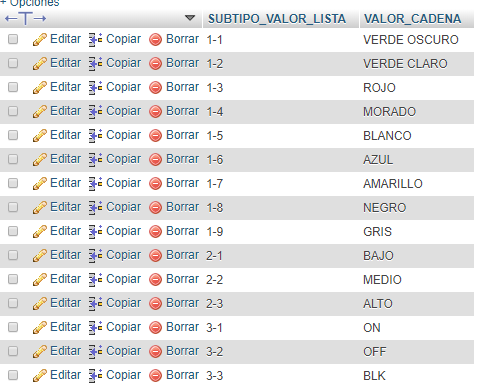
SELECT \* FROM tbl\_booleanos WHERE SUBTIPO\_VALOR\_BOOLEANO LIKE "1%"

Cambiar el valor de 1, por el numero de booleano a buscar, por ejemplo si se escribe el 1, va a retornar esto.



Tablas listas

En esta tabla se almacenan listas que se necesitan en diferentes inputs o entradas del sistema, actualmente se constituye de esta forma.



De la misma forma si desea buscar alguna lista, entre los miles que pueden haber se usa esta consulta.

SELECT \* FROM TBL\_LISTAS

WHERE SUBTIPO\_VALOR\_LISTA LIKE "2%"

En este caso buscamos la lista numero 2 y nos retorna lo siguiente



Tabla enteros\_flotantes

En esta tabla se almacenan valores numéricos prácticamente, cada valor tiene un campo primario que lo identifica de forma única, un valor mínimo y máximo, a continuación se muestra el ejemplo.



Tabla tbl\_master

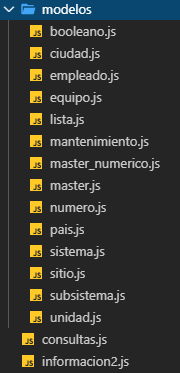
Esta tabla reúne datos de varias tablas, en esta tabla indicamos a que equipo le hacemos el mantenimiento, son los valores que se le insertan a ese equipo en esa fecha, equipo que su mantenimiento puede ser diario, semanal, mensual o trimestral, a continuación se muestra el ejemplo.



Se indica el campo primario de ese registro, el equipo sobre el que se hace el mantenimiento, hay equipos que no necesariamente tienen un componente final, asi que ese campo se dejo como opcional, la fecha es automática (aún no está en formato UTC), y cada uno de los valores que puede aceptar ese equipo, incluso se pueden hacer combinaciones entre valores booleanos, listas, números, y el ultimo campo es el dato obtenido por el usuario, los valores numéricos son los únicos que el usuario va a tener que digitar algo, los otros valores ya van a estar almacenados previamente en la base de datos.

# Base de datos Mongo DB

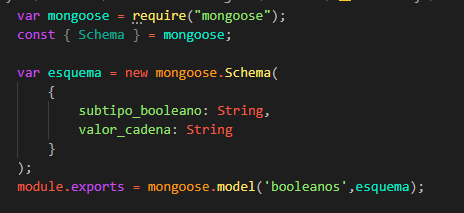
También se elaboró parte de este modelo usando el gestor Mongo DB, el cual es un gestor mucho más moderno y actualmente se encuentra agarrando más fuerza en cuanto a las bases de datos no relacionales.



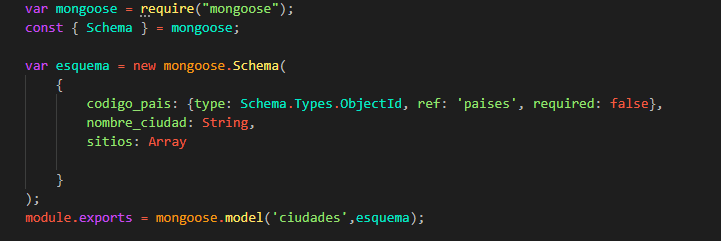
Los modelos se encuentran en archivos con extensión .js, en base a estos modelos es que se guía la aplicación, como si fuera el esquema que estamos acostumbrados a hacer en My SQL Work Bench.

A continuación, se exponen lo que son las colecciones de esta base de datos

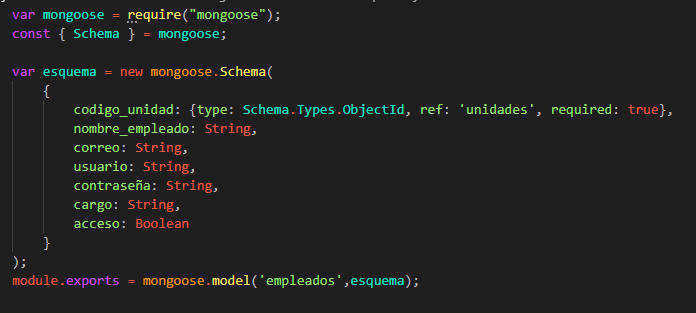
1. Colección de booleanos



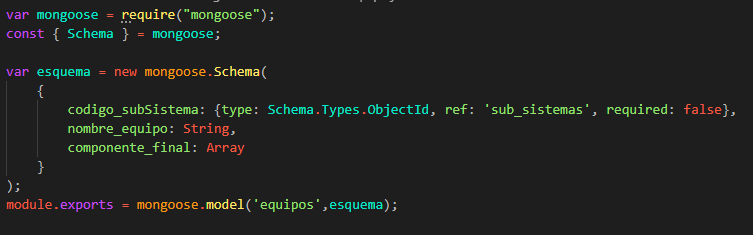
1. Colección de ciudades



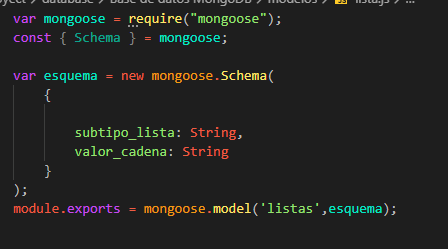
1. Colección de empleados



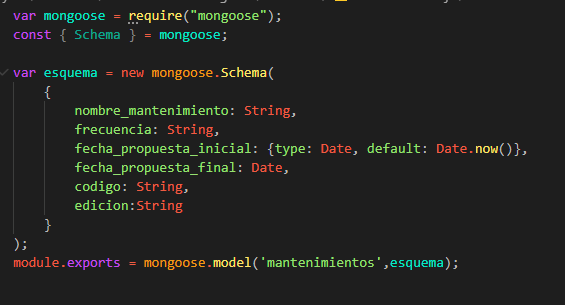
1. Colección de equipos



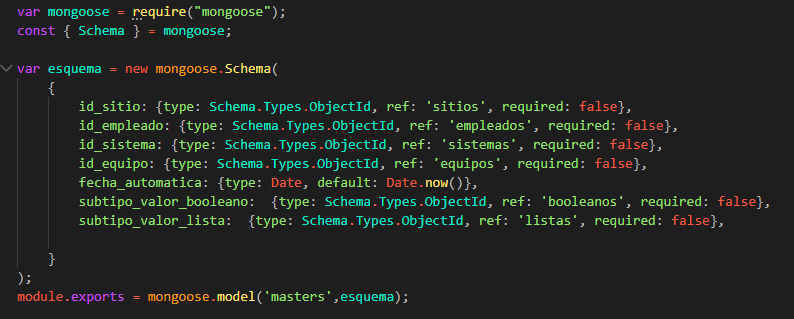
1. Colección de listas



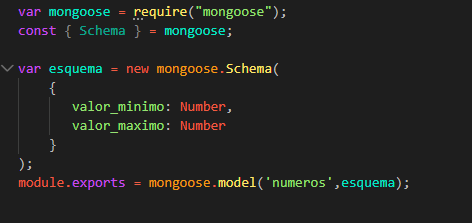
1. Colección de mantenimientos



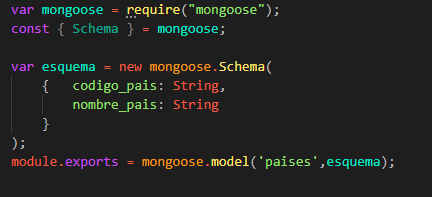
1. Colección master



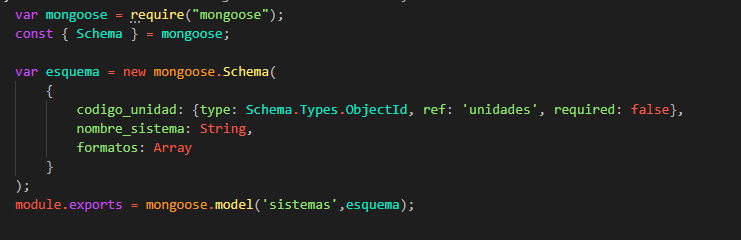
1. Colección de datos numéricos



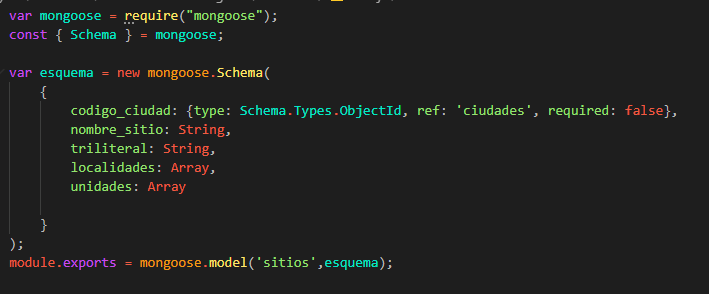
1. Colección de países



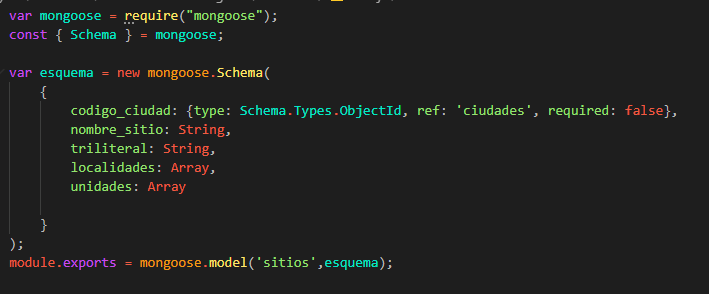
1. Colección de sistemas



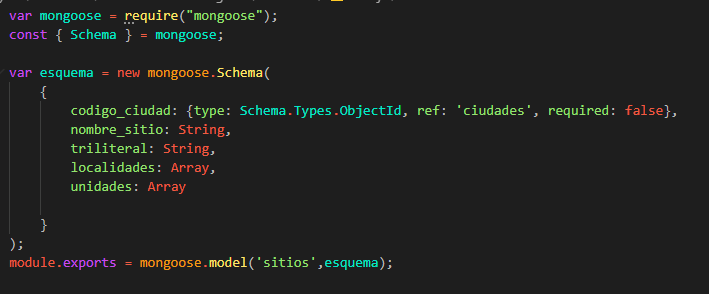
1. Colección de sitios



1. Colección de subsistemas



1. Colección de unidades

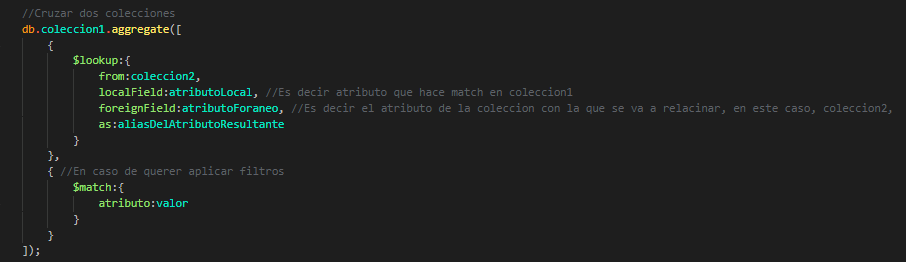


Cabe destacar que la forma que se hacen los cruces aquí, así como en My SQL usamos sentencias multitabla, ejemplos INNER JOIN, LEFT JOIN, RIGHT JOIN, etc,

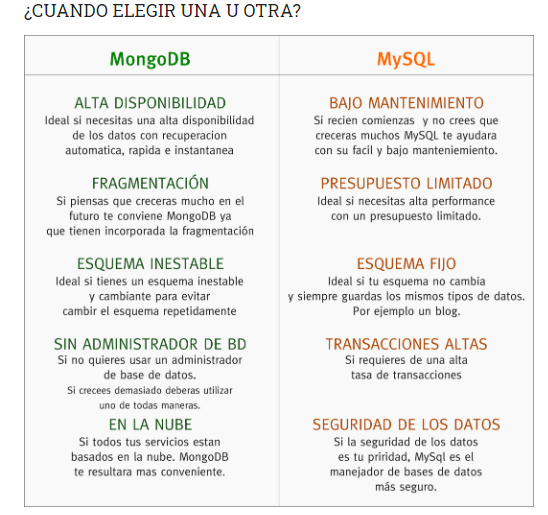
Aquí los parámetros que tienen el tipo type: Schema.Types.ObjectId, indican claves foráneas, y para entender esto mejor, pongamos un ejemplo:

La colección o tabla países es independiente, todas las colecciones independientes no van a tener el dato Schema.Types. ObjectId, pero las colecciones dependientes por ejemplo ciudades, las ciudades dependen de un país, por lo tanto la colección ciudades debe tener un dato con tipo Schema.Types.ObjectId, el cual almacene el \_id del país al que pertenece, \_id es el equivalente a una clave primaria de las que se usan en SQL.

Una forma de hacer cruces en Mongo, equivalente a las operaciones multitabla que se usan en My SQL, con INNER JOIN, es de esta forma.



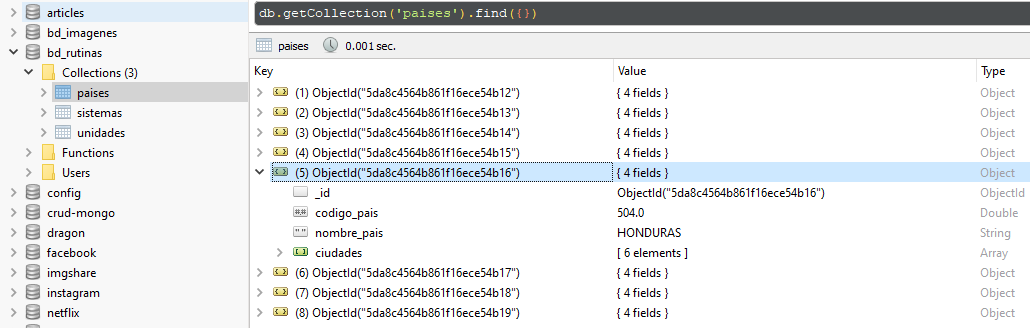
En **colección1** se escribe el nombre de la colección dependiente, es decir la colección que tiene el atributo con el tipo Schema.Types.ObjectID, en **colección2** se escribe el nombre de la colección independiente, por ejemplo si miramos países y ciudades, los países son independientes, pero las ciudades no, las ciudades dependen de un país, por lo tanto la colección de ciudades debe tener un atributo con tipo Schema.Types.ObjectId, el cual haga referencia al país que pertenece esa ciudad, el atributo **localField** lo que se escribe ahí es el nombre del campo con el que va a hacer el cruce el campo que tiene tipo Schema.Types.ObjectId, y en el atributo **foreignField**, por lo general se escribe \_id, porque ese atributo es un JSON, un JSON es un objeto JavaScript que se escribe entre llaves así, { }. Y por último el **$match,** es para que únicamente nos devuelva un registro en especifico de la base de datos, cabe destacar que aquí a los registros se les llaman documentos, u objetos JSON.



# Datos almacenados en base de datos Mongo DB

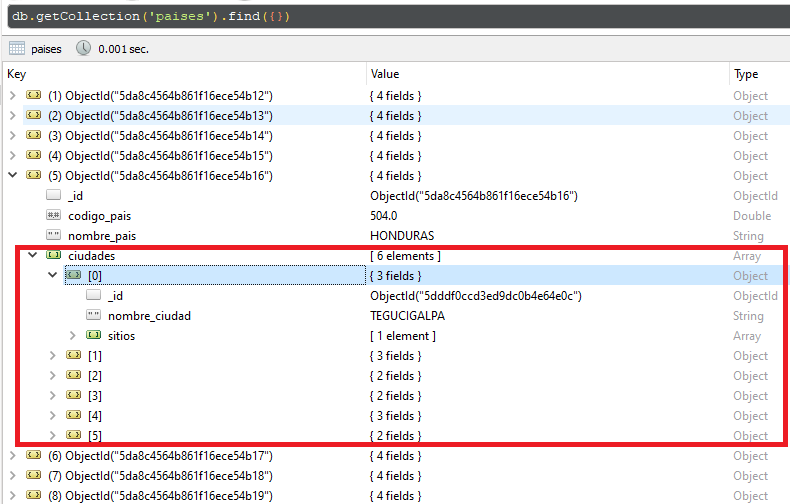
Esta base de datos no era el objetivo principal, aun así, es un aporte extra de mi parte, se crearon 3 colecciones, países, unidades y sistemas, cada colección tiene arreglos con objetos JSONs anidados.

Las colecciones en Mongo DB siguen una estructura jerárquica muy similar a una estructura como un árbol, se utilizaron dos herramientas las cuales son, Mongo DB Compass y Robo 3T, explicaré en Robo 3T ya que a mi en lo personal me agrada más.

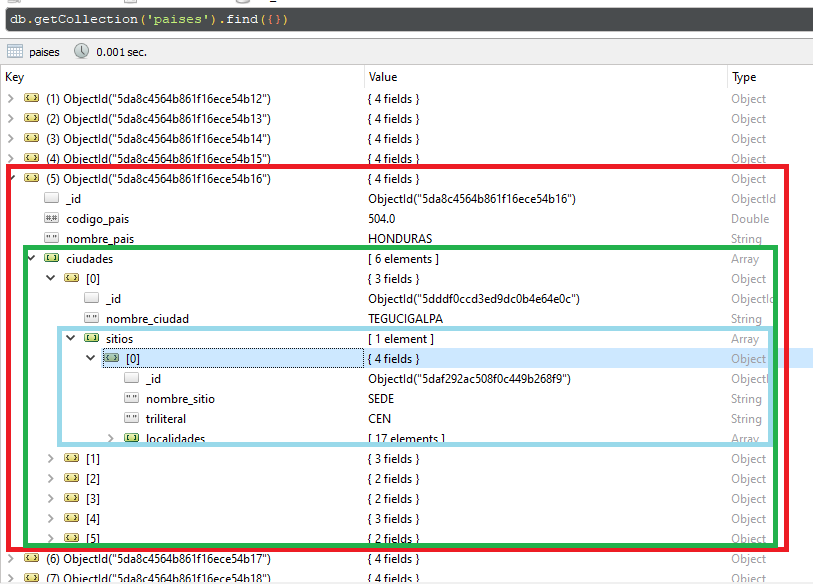


En la parte de la derecha se ven todas las bases de datos que tengamos, al pulsar click sobre alguna base de datos podemos ver sus colecciones, las colecciones son equivalentes a las tablas como decíamos antes.

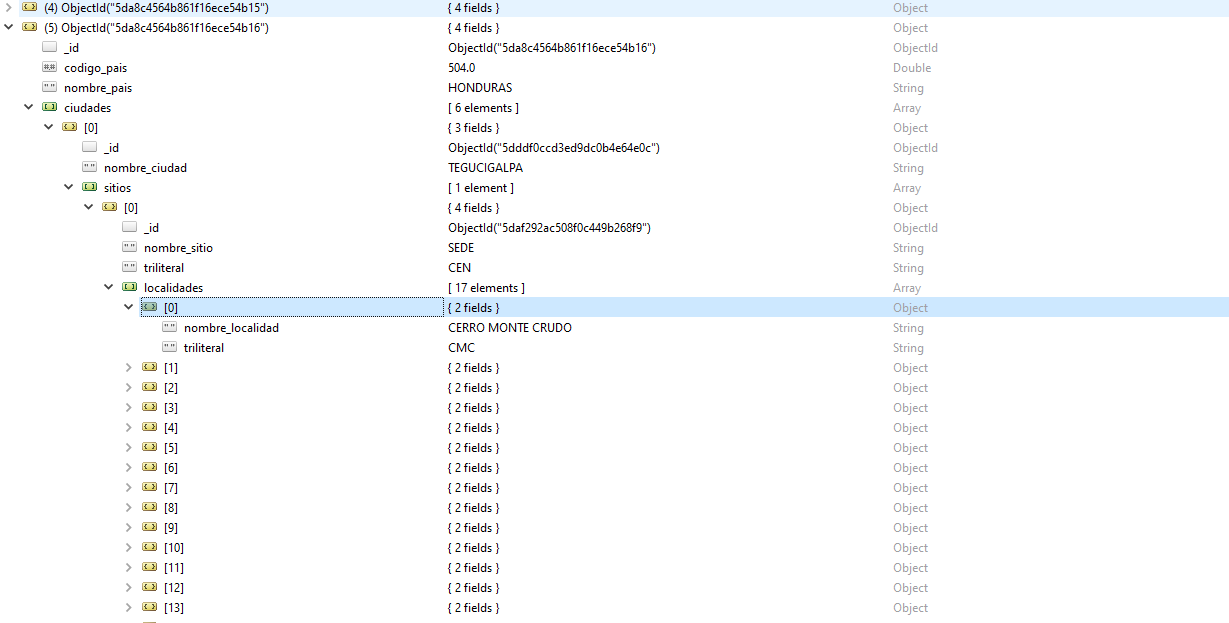
En la imagen mostrada arriba podemos ver JSON del país Honduras, que tiene un arreglo de ciudades, si damos click vamos a poder ver las ciudades de Honduras.



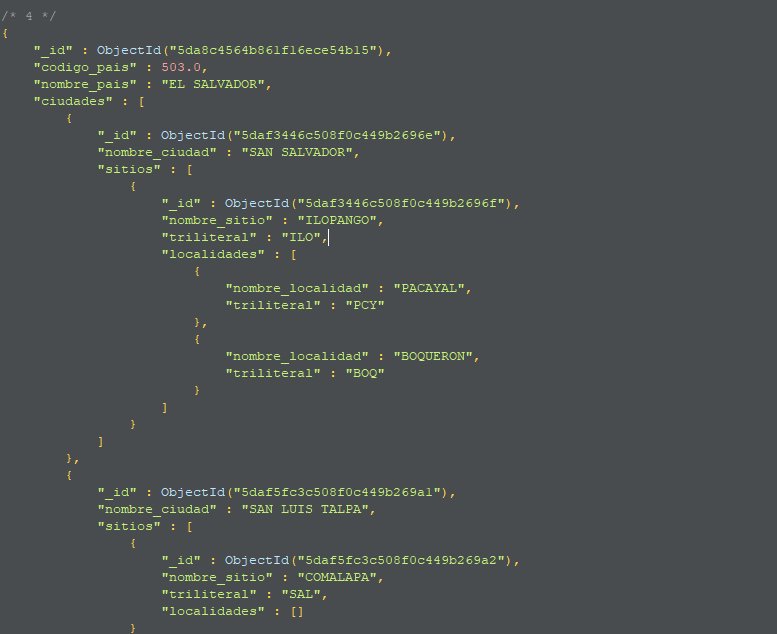
En ese caso podemos ver que Honduras tiene 6 ciudades, (Con la posibilidad de agregar más ciudades si se desea), al darle click en el primer ítem vemos que es de Tegucigalpa, ese JSON tiene otro arreglo que es de sitios, lo que se quiere decir, es que esos sitios pertenecen a la ciudad de Tegucigalpa.



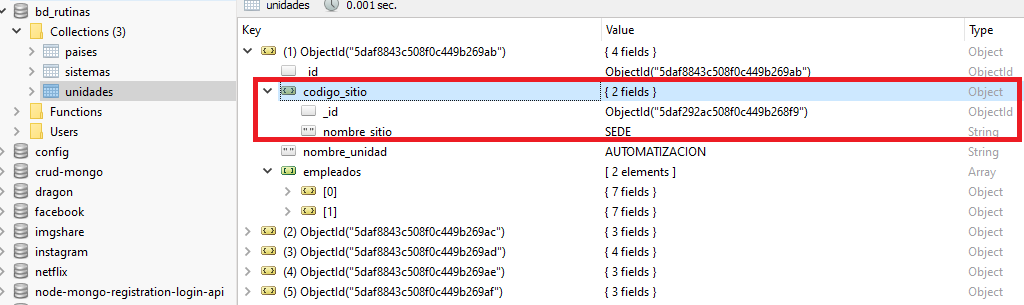
Seleccionamos en el cuadro azul el sitio de la SEDE, que pertenece a la ciudad de Tegucigalpa, y ese sitio puede tener localidades, al pulsar en el arreglo de localidades, se pueden observar.



Incluso se puede observar de esta otra forma



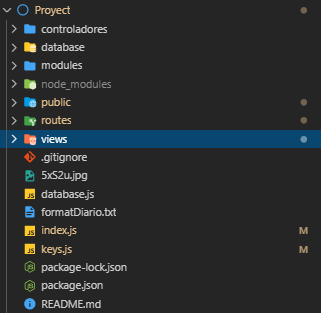
Y para hacer un cruce como hacemos en My SQL, se puede hacer de esta forma



En ese caso está diciendo que la unidad automatización le pertenece al sitio de la SEDE, como es un campo de tipo Schema.Types.ObjectId, entonces puede tener el \_id del sitio al que pertenece esa unidad, los sitios están en la colección de países.

# Estructura del proyecto

El proyecto se encuentra estructurado de esta forma



El proyecto de forma original se conecta a la base de datos My SQL, pues la idea para hacerlo era usando este gestor, el lenguaje de programación utilizado es JavaScript, se usa el entorno Node JS del lado del servidor, al decir este término del lado del servidor, es del lado de la base de datos o sea lo que se le llama el back end, que es lo que le hace que cada componente del Front End, (HTML, CSS, Java Script), pueda hacer la función para la cual haya sido establecido

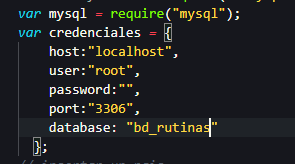
# Carpeta controladores

Esa carpeta contiene archivos que contienen funciones, esas funciones se encargan de hacer las tareas que tengan que hacer los routers, routers son las rutas que el servidor responde cuando el cliente hace una petición.

Una aplicación CRUD, es una aplicación básica de un sistema de información, CRUD significa *CREATE READ UPDATE DELETE,* Create se refiere a la operación de insertar en la base de datos, lo cual se usa la sentencia INSERT INTO.

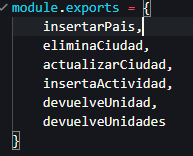
Read es lo que se le llama leer, para esta operación se usan consultas de tipo SELECT

Update es la operación para actualizar un elemento en la base de datos, y Delete es la operación de eliminar un dato que ya está insertado en la base de datos, cabe destacar en esta carpeta se encuentra la conexión a la base de datos My SQL, la conexión a la base de datos la hace con el siguiente código



En este caso el host, está ejecutándose en la misma computadora que se desarrolla la app, el usuario indica que es root, contraseña no tiene, el puerto por defecto es el 3306, y la ultima línea es muy importante, en esa línea indica el nombre de la base de datos que se va a conectar.

Cada archivo contiene funciones que exporta a la carpeta routes, las funciones las exporta con el siguiente código.



Cada función es posible utilizarla en los archivos de la carpeta routes en los cuales sea invocado el archivo en el que se encuentran dichas funciones, la forma de la cual se invocan archivos en Expressjs (Node JS) se hace de esta forma.



# Carpeta routes

La carpeta routes se conecta a la carpeta llamada controladores, un ejemplo de la forma que un archivo dentro de esta carpeta, se conecta a un archivo ubicado en la carpeta controladores se hace con esta línea de código.



El módulo router indica que se define la dirección de una ruta al servidor, algo que el cliente solicita y se le brinda una respuesta.



Para seguir un orden de las rutas se ha elaborado un archivo llamado índex.routes, lo que hace es concatenar un prefijo a los archivos que se encuentran en esa misma carpeta, el objetivo de añadir un prefijo es indicar que tipo de rutas son, y para entenderle mejor, veamos la siguiente imagen.



Como se puede observar hay líneas comentadas, esas líneas se han dejado como propuesta, aun no se han utilizado, hay rutas para autenticación de usuarios, rutas para probar cosas, hay rutas para renderizar vistas, este termino renderizar significa que lanza una vista en pantalla de acuerdo a la petición que haga el cliente desde la aplicación real.

Si observamos la ruta que tiene el prefijo /test, esa ruta se conecta con un archivo en la misma carpeta, el cual se llama test. routes, a continuación, analizamos ese archivo.



Ahí se encuentran algunas rutas de ensayo, cada ruta llama a una función que se encuentra en este caso en el archivo ctrl-rutas-post, dentro de la carpeta controladores, y cada ruta le añade un prefijo extra que se llama *api\_mantenimiento* el cual es el nombre de la aplicación, ese prefijo se lo añade en el archivo index.js de la carpeta raíz del proyecto, ese archivo es el que se encarga de levantar el servidor y de invocar todas las rutas que se encuentren en el archivo index.routes.

Esta es una explicación básica de la lógica que pretendo llevar al desarrollar esta aplicación.

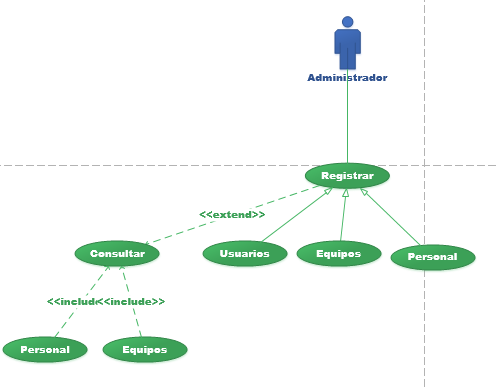
# Carpeta public

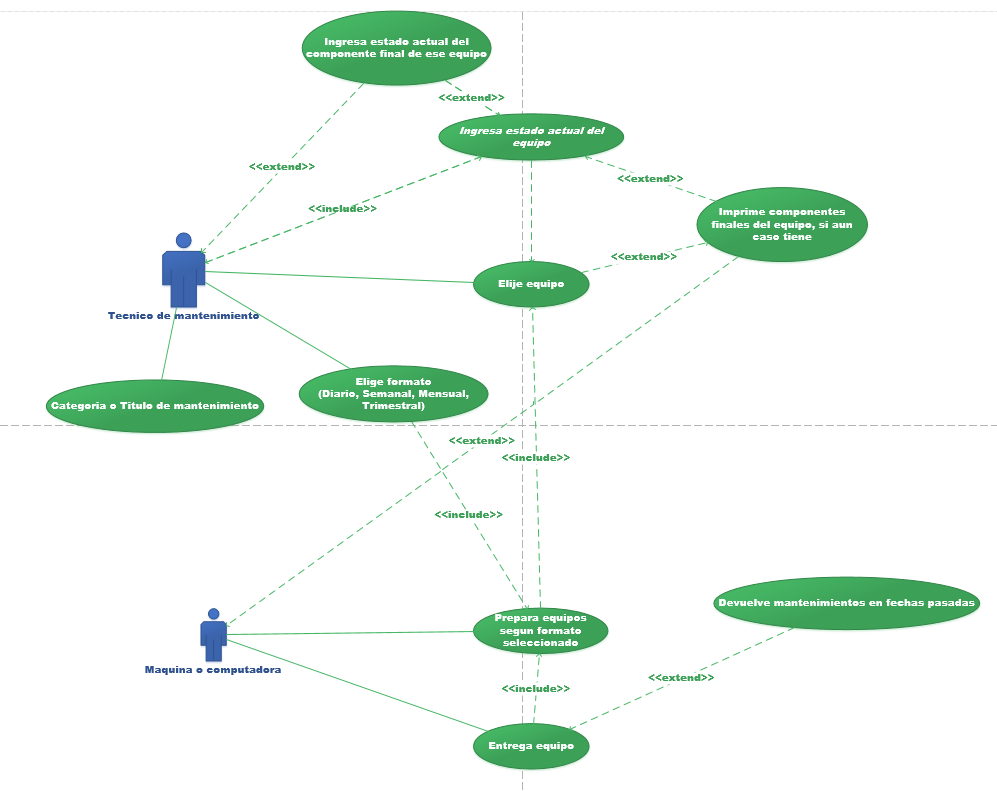
En esta carpeta se almacenan literalmente como dice archivos públicos, archivos públicos son estilos CSS, archivos Javascript que permitan añadir funcionalidades como validaciones antes de enviar al servidor los datos, además de conectar las vistas con el modelo, a esto se le llama el controlador, lo que conecta las vistas con el modelo es decir lo que se planea almacenar en la base de datos, o lo que se piensa obtener.

# Carpeta views

En esta carpeta se han guardado vistas de la aplicación utilizando HTML, CSS, Bootstrapt, jQuery y otros plugins adicionales, se ha pensado utilizar el motor de vistas EJS para generar HTML dinamico, cabe destacar que el uso del motor de plantillas no es necesario del todo, estas cosas se pueden hacer con archivos con extensión .html, lo que pasa es que cuando se cambia de una pantalla a otra genera un poco de confusión, mientras que con el motor de plantillas, dependiendo de la acción que realice el usuario en la aplicación, se le va a renderizar o mostrar la vista, de acuerdo a la petición que solicite al servidor, especialmente si es una petición de tipo GET, que son las peticiones que devuelven datos del servidor.

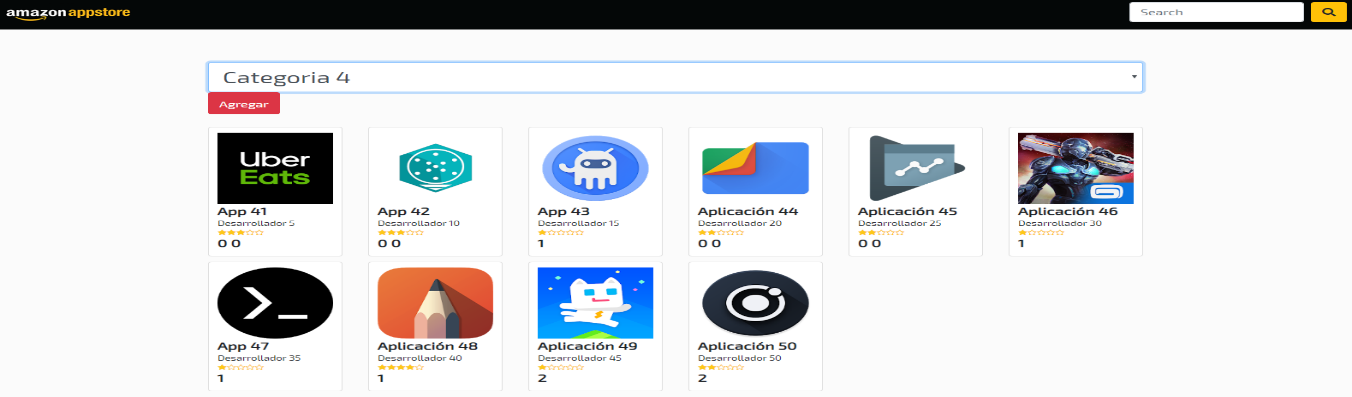
# Casos de uso





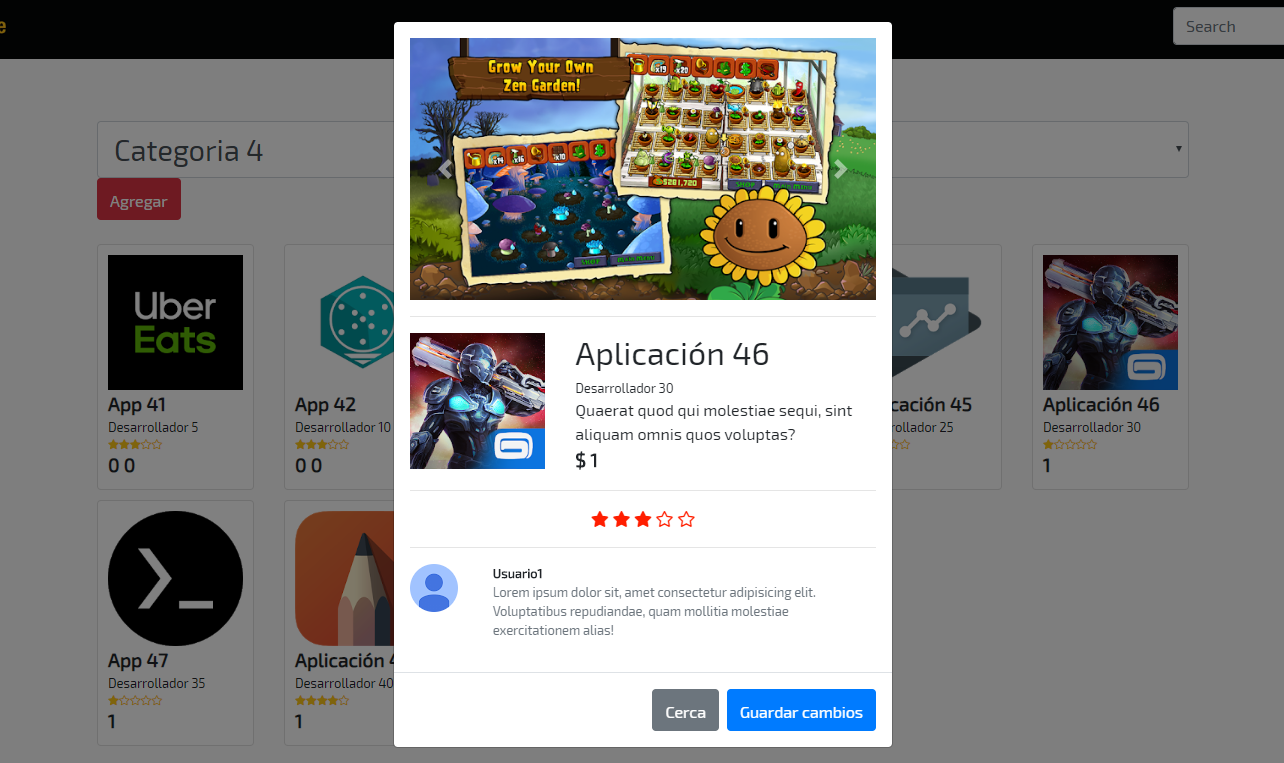
# Propuesta diseño del Front End

Para el diseño actual de las pantallas tengo esta idea en mente



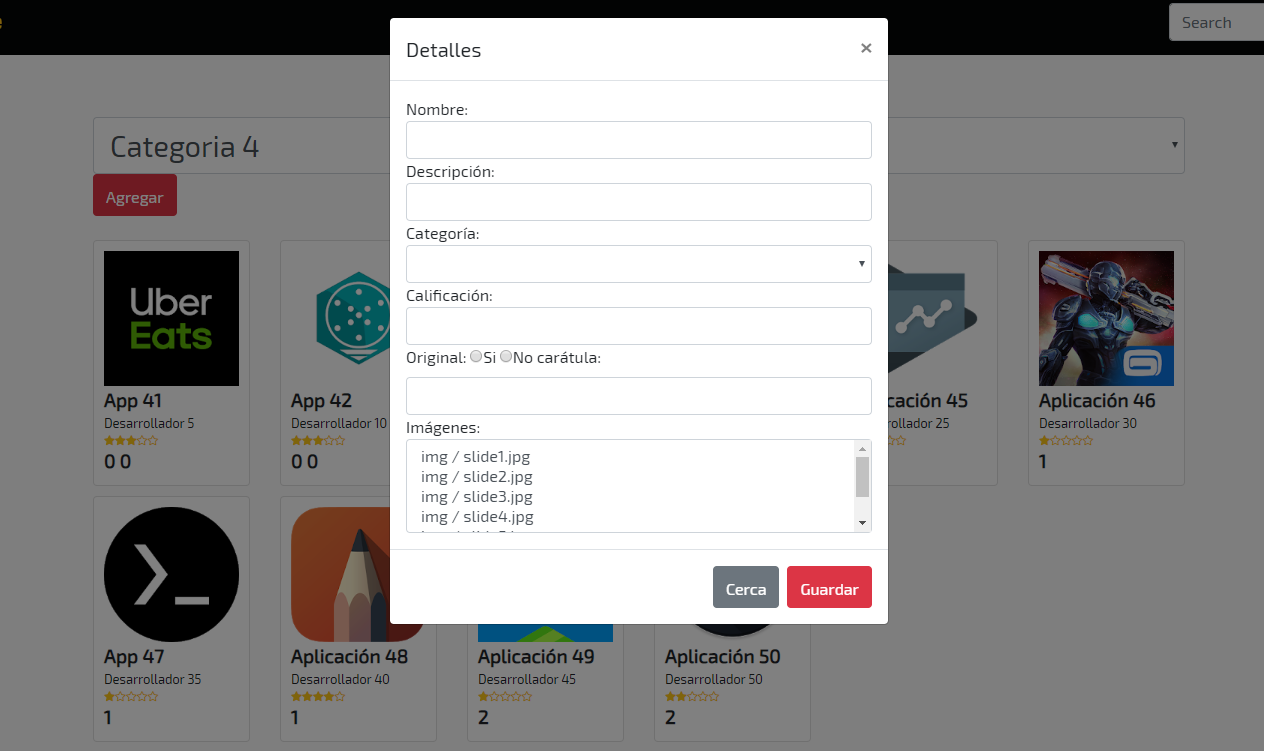
La idea es que al seleccionar en el select list o comboBox un equipo, abajo se muestren las componentes de ese equipo, si el equipo no tienen ningún componente final, se muestra un mensaje de que el equipo no tiene eso, o simplemente no se muestra nada.

Al mostrar cada componente final, el mantenimiento que se le va a hacer, la idea es así, al darle click a cada componente desplegar un formulario modal, en el cual se puedan ingresar los datos que le corresponden a ese equipo en esa fecha, algo así.



Esto solamente son ejemplos de cómo puede ir la aplicación, no es la aplicación real todavía.

Hay ciertos equipos que no tienen componente final, pero que de igual forma requieren algún mantenimiento, al presionar el botón que dice agregar abajo del select list, ese botón es para hacer mantenimiento a ese equipo, si el equipo tiene componentes finales, ese botón podría desaparecer ya que el mantenimiento de ese equipo se divide en múltiples componentes haciendo un equipo completo, ese botón podría desplegar un formulario como el siguiente.



Se han dejado controles que podría llevar la aplicación real, hay controles de texto, controles de combo Box o select list con opción de ingresar un elemento o de ingresar múltiples elementos, hay de radio botones para los valores booleanos y también controles numéricos, para datos numéricos, valga la redundancia.

# Glosario de términos

**MySQL:** Sistema de gestión de base de datos relacional desarrollado bajo licencia dual por Oracle Corporation y está considerada como la base de datos de código abierto más popular del mundo y una de las más populares en general junto a Oracle y SQL server sobre todo para entornos de desarrollo web.

**Mongo DB:** Es un sistema de base de datos No SQL orientado a documentos de código abierto, tiene la capacidad de realizar consultas utilizando JavaScript, haciendo que estas sean enviadas directamente a la base de datos para ser ejecutadas.

**JavaScript:** Es un lenguaje de programación interpretado, dialecto del estándar ECMAScript, se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

**Node JS:** Es una forma de ejecutar JavaScript en el servidor, además de mucho más, es un entorno JavaScript del lado del servidor basado en eventos.

**Express. Js:** Es un framework rápido, minimalista y flexible de Node JS. Permite crear APIs y aplicaciones web fácilmente, provee un conjunto de características como manejo de rutas (direccionamiento), archivos estáticos, uso de motor de plantillas, integración con bases de datos, manejo de errores, middlewares entre otras.

**jQuery:** Es una biblioteca multiplataforma de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

**AJAX:** Es un acrónimo de JavaScript asíncrono y XML, esta tecnología nos ayuda a cargar datos desde el servidor sin una actualización de la página del navegador. JQuery es una gran herramienta que proporciona un amplio conjunto de métodos AJAX para desarrollar aplicaciones web de próxima generación.

**EJS:** Es un lenguaje de plantillas simple que le permite generar marcado HTML con JavaScript simple.

**CRUD:** CRUD es la parte esencial que todo sistema de información debe tener, la forma en que interactua con la base de datos sus operaciones esenciales son

Lectura -> SELECT

Escritura -> INSERT

Actualización -> UPDATE

Eliminación -> DELETE