Proyecto rutinas mantenimiento

**2019**

Héctor Josué Ardón Morga

Corporación Centroamericana de Servicios de Navegación Aérea

22/11/2019



Contenido

[***Introducción*** 2](#_Toc25323202)

[***Identificación del problema*** 3](#_Toc25323203)

[***Estructura del proyecto*** 9](#_Toc25323204)

[***Carpeta controladores*** 10](#_Toc25323205)

[***Carpeta routes*** 11](#_Toc25323206)

[***Casos de uso*** 14](#_Toc25323207)

[***Propuesta diseño del Front End*** 15](#_Toc25323208)

# Introducción

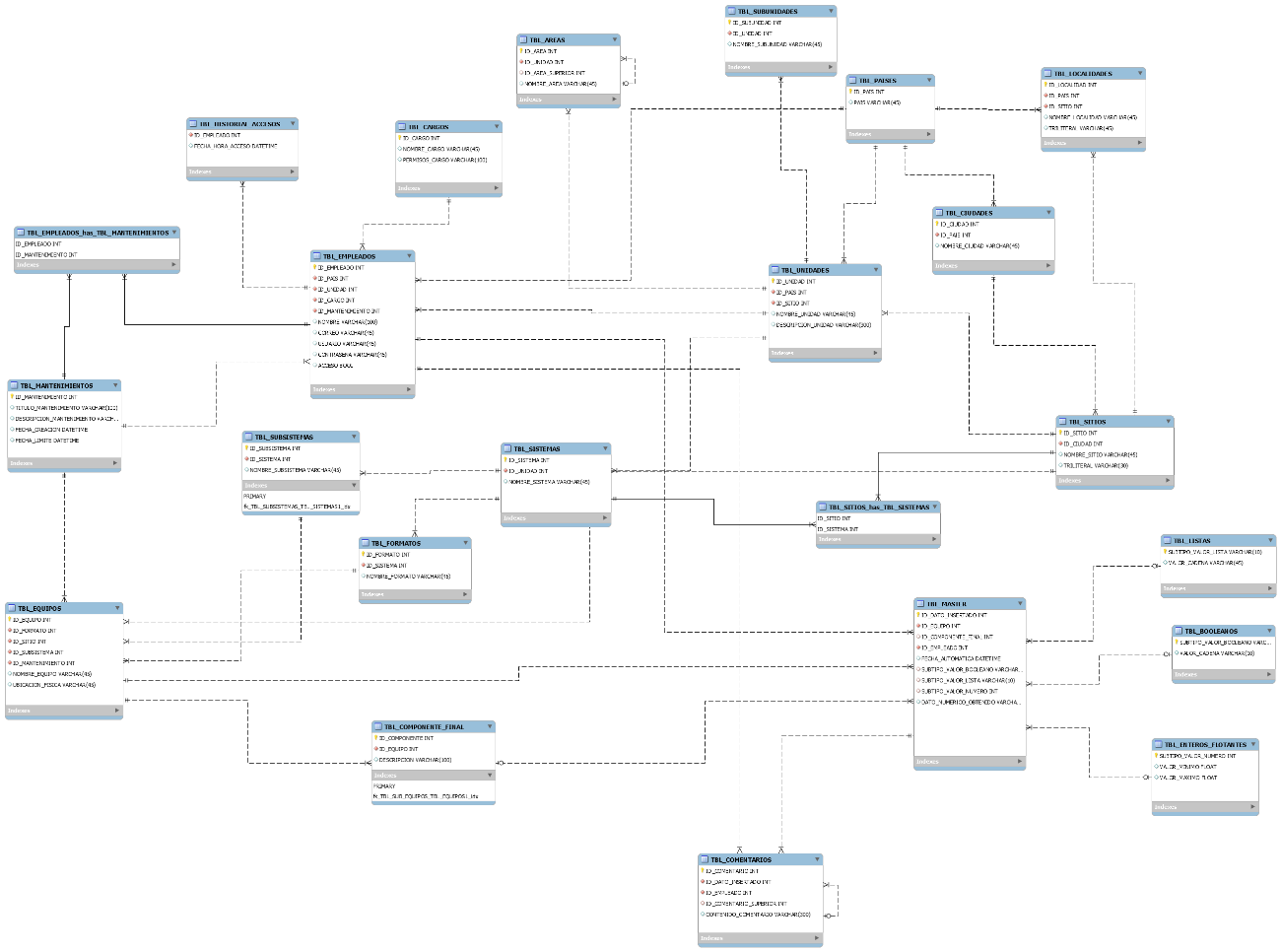
El presente proyecto describe algunas características utilizadas para el desarrollo de este proyecto, el cual es para la empresa COCESNA. El desarrollo de este proyecto describe algunas cosas elaboradas hasta la fecha, el cual nos ha servido para comenzar a evolucionar aspectos que son importantes de 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.

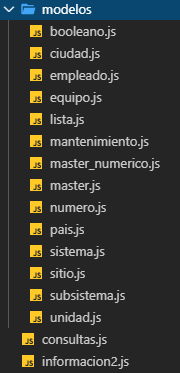
Se han hecho visitas a los centros de mantenimiento recolectando datos útiles para la lógica de la base de datos

# Identificación del problema

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.



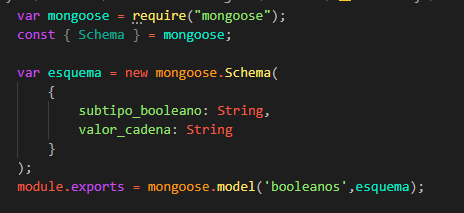
También se elaboró parte de este modelo usando el gestor Mongo DB, el cual es un gestor mucho más moderno y actualmente se encuentra agarrando más fuerza en cuanto a las bases de datos no relacionales.



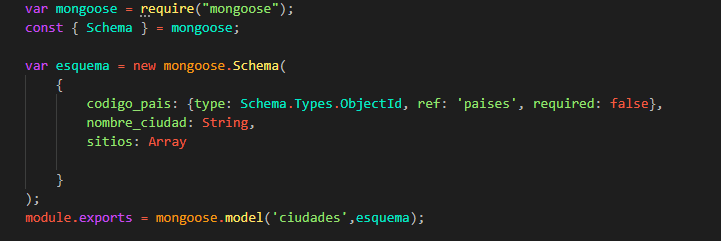
Los modelos se encuentran en archivos con extensión .js, en base a estos modelos es que se guía la aplicación, como si fuera el esquema que estamos acostumbrados a hacer en My SQL Work Bench.

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

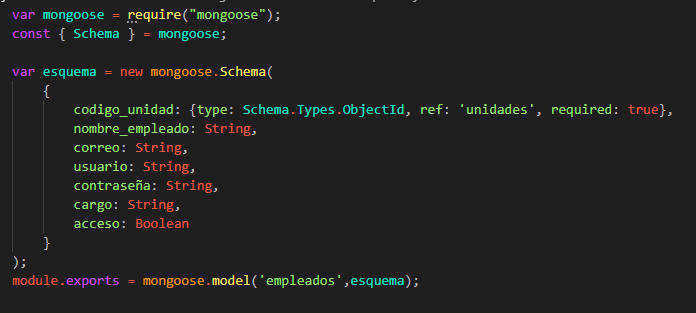
1. Colección de booleanos



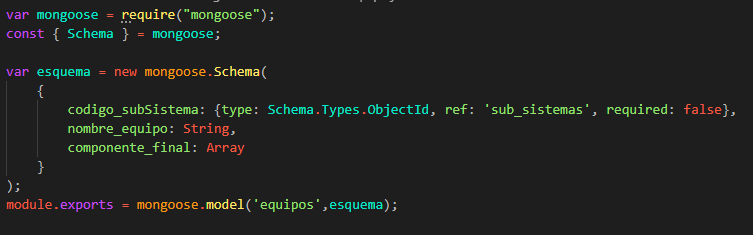
1. Colección de ciudades



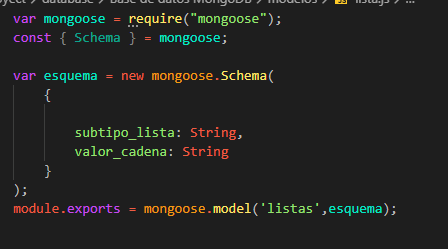
1. Colección de empleados



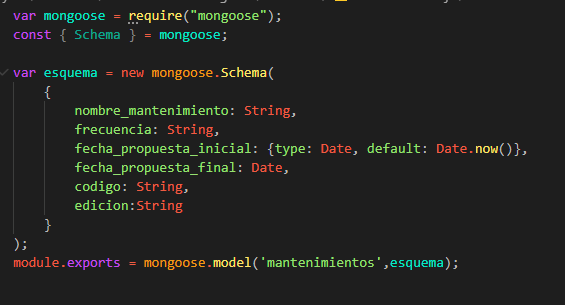
1. Colección de equipos



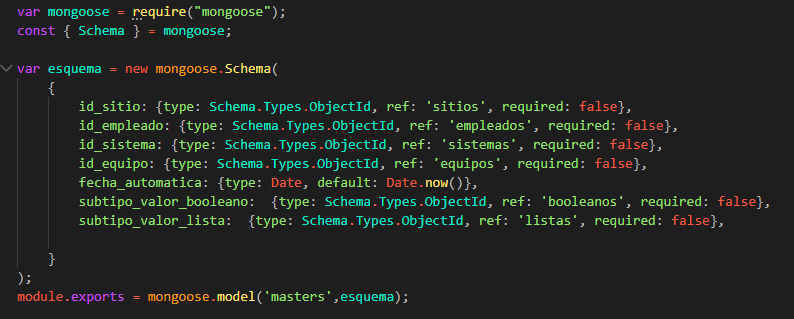
1. Colección de listas



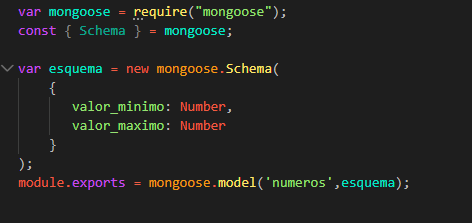
1. Colección de mantenimientos



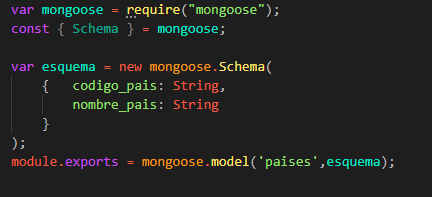
1. Colección master



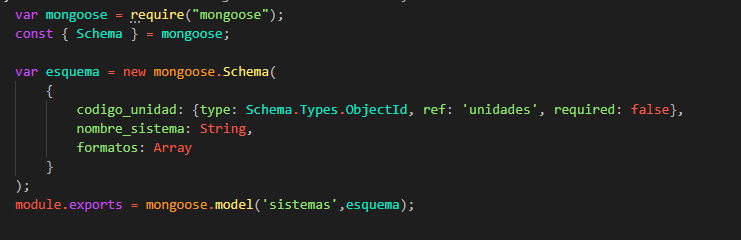
1. Colección de datos numéricos



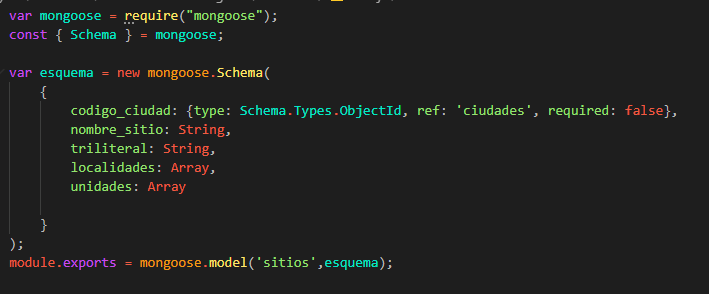
1. Colección de países



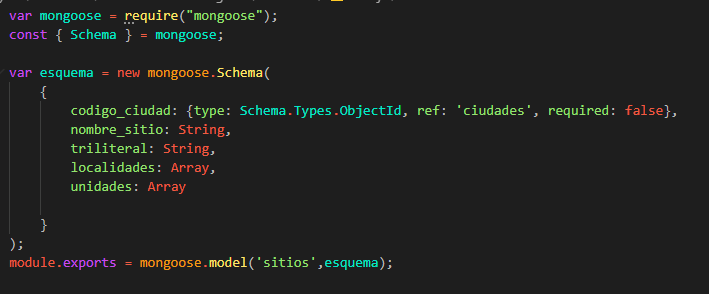
1. Colección de sistemas



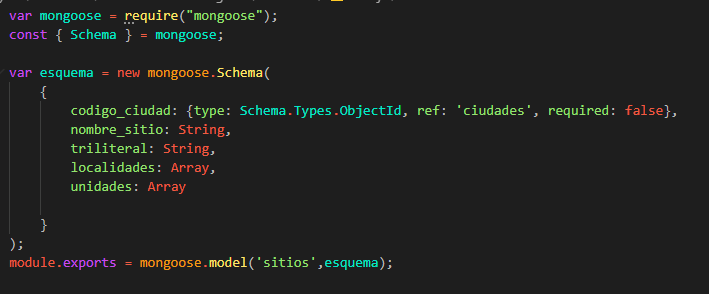
1. Colección de sitios



1. Colección de subsistemas



1. Colección de unidades

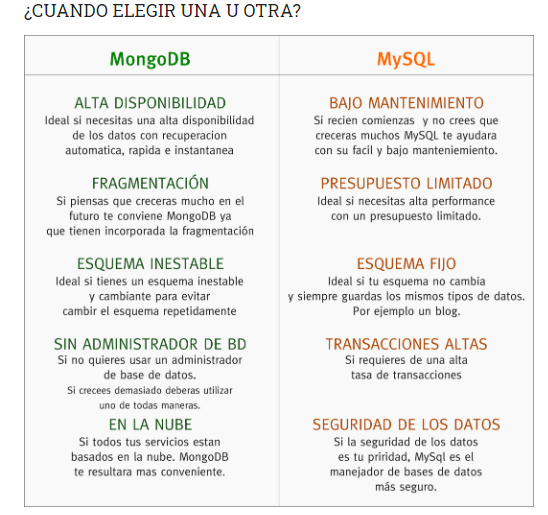


Cabe destacar que la forma que se hacen los cruces aquí, así como en My SQL usamos sentencias multitabla, ejemplos INNER JOIN, LEFT JOIN, RIGHT JOIN, etc,

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

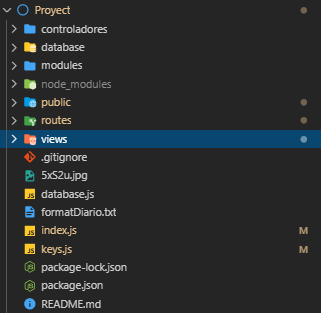
La colección o tabla países es independiente, todas las colecciones independientes no van a tener el dato Schema.Types.ObjectId, pero las colecciones dependientes por ejemplo ciudades, las ciudades dependen de un país, por lo tanto la colección ciudades debe tener un dato con tipo Schema.Types.ObjectId, 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.

Me tomo un poco más el tiempo de explicar sobre este gestor, porque SQL es bastante popular, ya muchas consultas y sentencias se conocen desde hace muchos años, y la documentación es incluso mucho más extensa.



# 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

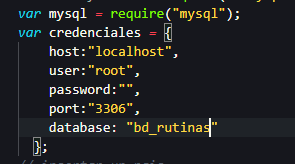
# Carpeta controladores

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.

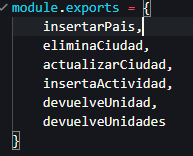
Read es lo que se le llama leer, para esta operación se usan consultas de tipo SELECT

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.

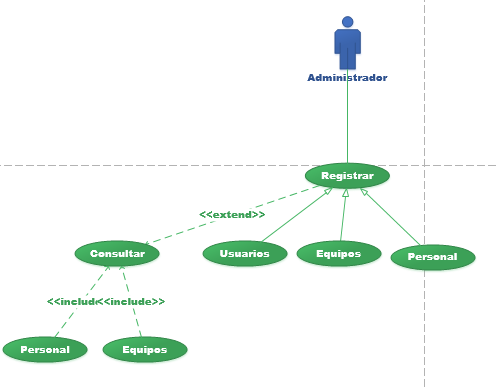
Si observamos la ruta que tiene el prefijo /test, esa ruta se conecta con un archivo en la misma carpeta, el cual se llama test. routes, a continuación, analizamos ese archivo.

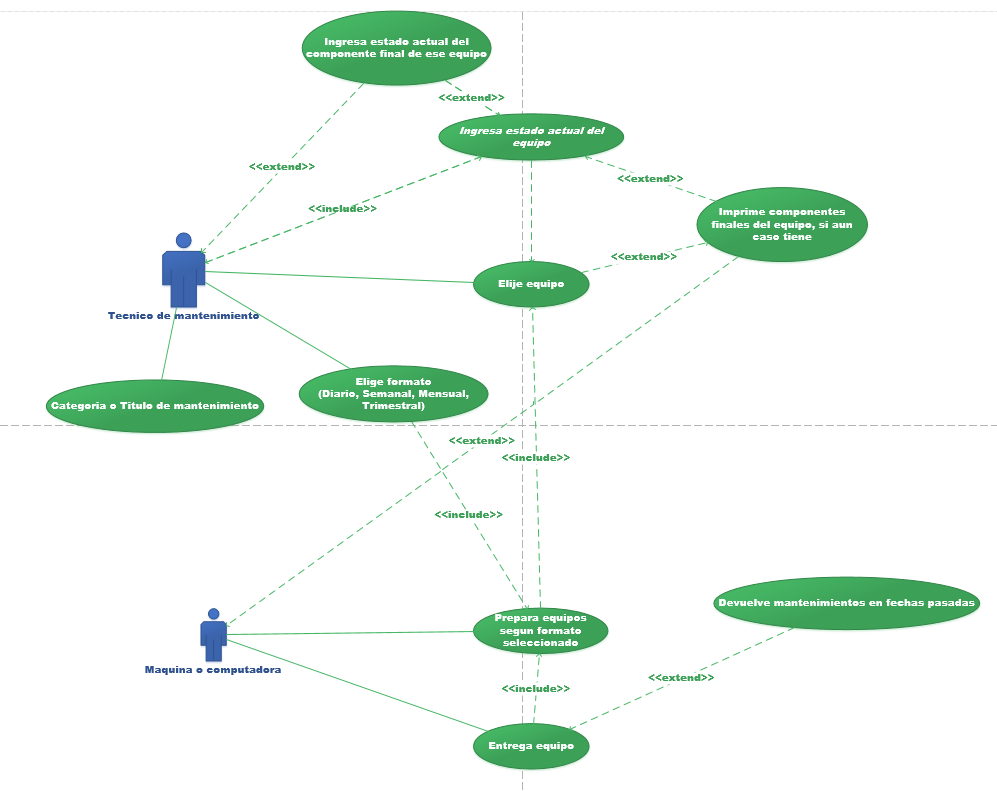


Ahí se encuentran algunas rutas de ensayo, cada ruta llama a una función que se encuentra en este caso en el archivo ctrl-rutas-post, dentro de la carpeta controladores, y cada ruta le añade un prefijo extra que se llama *api\_mantenimiento* el cual es el nombre de la aplicación, ese prefijo se lo añade 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.

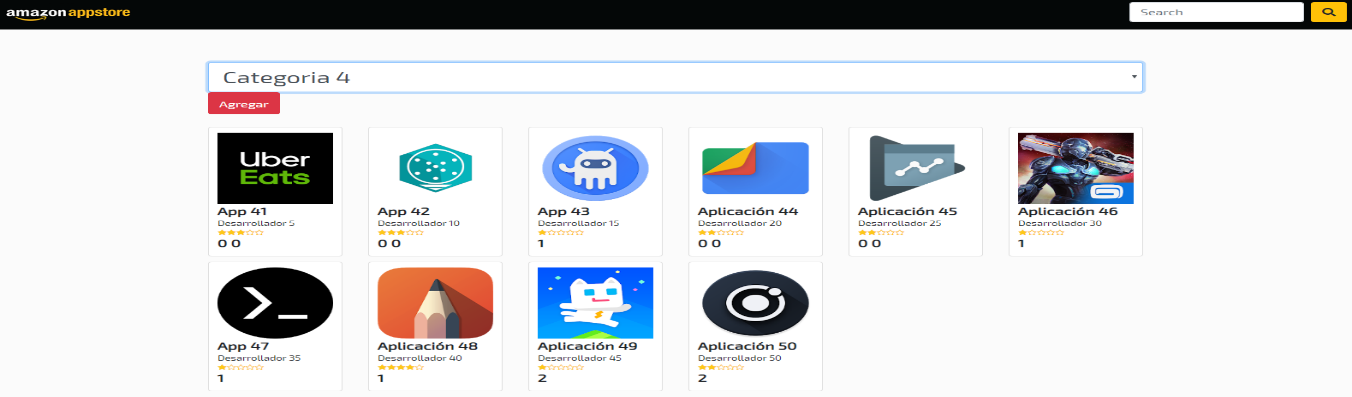
Esta es una explicación básica de la lógica que pretendo llevar al desarrollar esta aplicación.

# Casos de uso



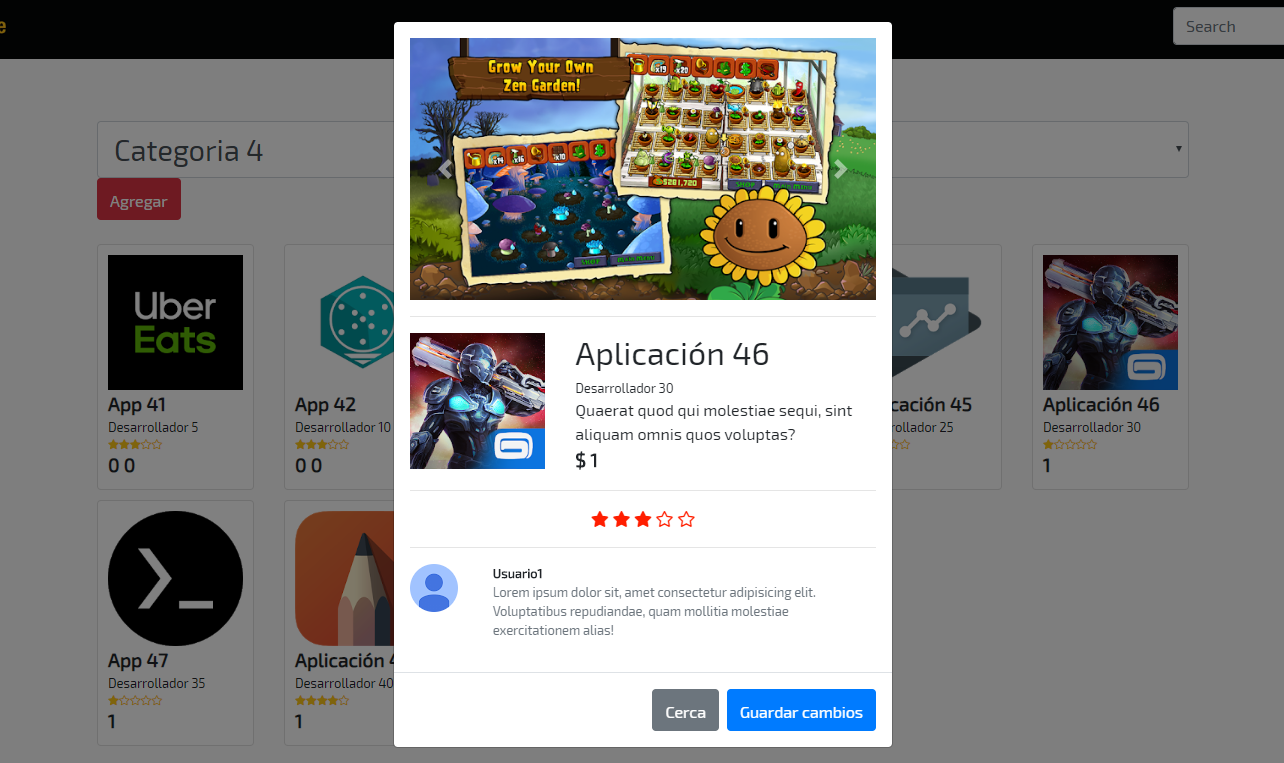


# Propuesta diseño del Front End

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

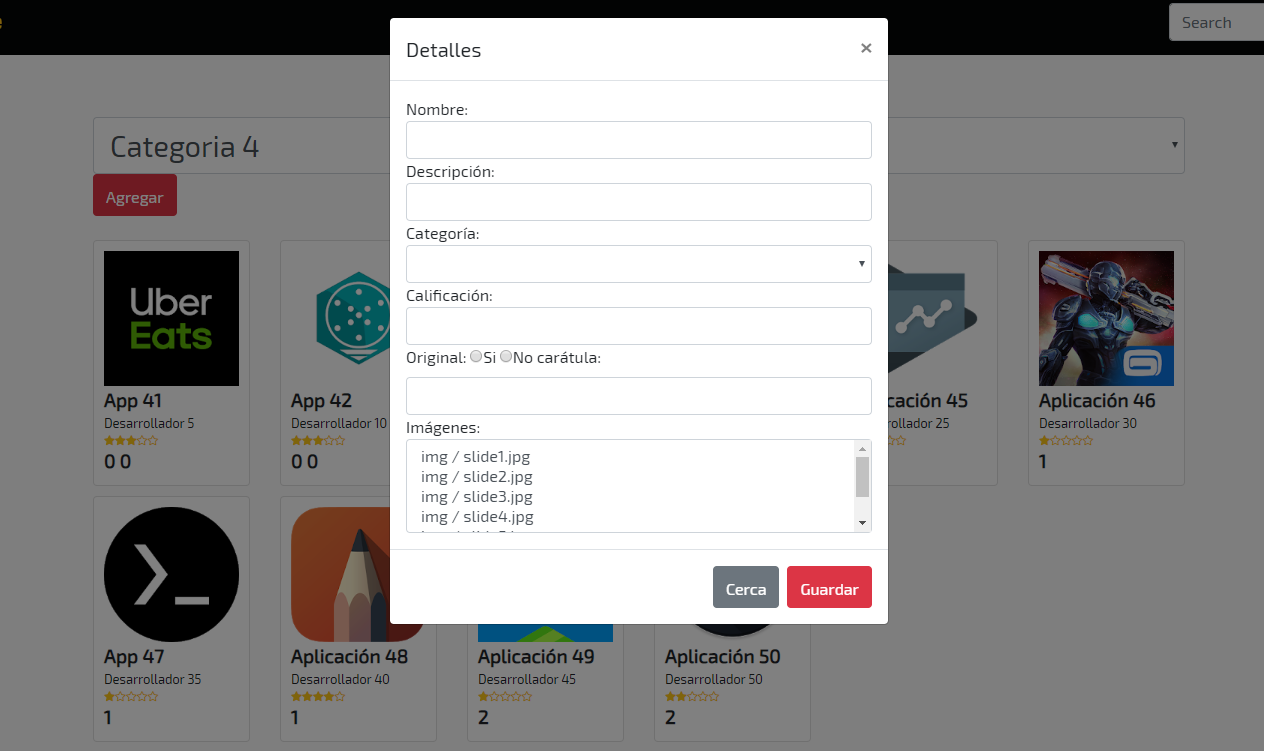
La idea es que al seleccionar en el select list o comboBox un equipo, abajo se muestren las componentes de ese equipo, si el equipo no tienen ningún componente final, se muestra un mensaje de que el equipo no tiene eso, o simplemente no se muestra nada.

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.