Manual

Sistema de seguridad NodeJs – SailsJs

Héctor Agustín Charry Osorio

agustincharry@gmail.com

Manizales, Colombia 2018

Tabla de contenido

Contenido

Introducción – Descripción

El sistema presentado es un conjunto de módulos especializados para formar las bases de futuros proyectos de software web en los que se necesite un sistema de manejo de usuarios, roles y permisos, con formulario de inicio de sesión y toda la seguridad que se debe tener en un software que maneje información privada como lo es la gestión de usuarios en un sitio web.

El software presenta todo un panel de administración de "súper usuario" que permite gestionar toda la seguridad del proyecto que se esté desarrollando sobre él.

Todo el sistema está desarrollado en NodeJs con el framework SailsJs el cual cuenta con un ORM (Waterline) el cual gestiona toda la creación de tablas y consultas a la base de datos de forma automática, lo cual permite utilizar el sistema desde diferentes tipos de base de datos tanto relacionales, como no relacionales.

El software cuenta con una capa de seguridad para cada url visitada por el usuario en el sistema web y lo permite acceder o no a ciertas páginas según sus permisos en la misma.

Cuenta con un módulo de encriptación de contraseñas para intensificar la seguridad y cumplir con las normativas del país.

Además, permite gestionar y modificar todo lo referente a los menús, urls, permisos, usuarios y roles desde un panel de administrador usando el usuario “superUser”.

Pre-requisitos

Programas requeridos

- NodeJS 9.6 o superior.

  Nota: Para saber cuál es la versión instalada, escribir en la consola del sistema operativo: `node -v`

- MongoDB.

  Nota: Para el manual se usará ésta base de datos, pero ésta puede ser cambiada fácilmente.

Programas opcionales:

- Robo 3T.

Instalación del sistema

1) Iniciar la base de datos, MongoDB.

   A. Buscar la ubicación de instalación del ManoDB y abrir una consola del sistema operativo allí.
   Nota: Generalmente la ruta de instalación en Windows es:

   ```
   C:\Program Files\MongoDB\Server\3.6\bin
   ```

   B. En la consola escribir:

   ```
   .\mongod.exe
   ```

   Esto iniciará el servidor de MongoDB, haciendo que quede a la escucha de nuevas conexiones.

   Nota: No cerrar esa consola una vez se ha iniciado.

2) Configurar el sistema para que inicialice la base de datos.

   A. En las carpetas del proyecto, abrir el archivo ./config/bootstrap.js
   B. Buscar la línea `await startDB()`;
   C. Des-comentar la línea, y guardar.
   OJO: Una vez se inicie el sistema por primera vez, VOLVER A COMENTAR LA LÍNEA.

3) Instalar SailsJS

   A. Abrir una consola del sistema operativo y escribir:

   ```
   npm install sails –g
   ```

4) En la carpeta del proyecto, abrir una consola del sistema operativo y escribir:

   ```
   npm install
   ```

Correr el sistema

1) Si el servidor de MongoDB no está corriendo, iniciarlo:

   A. Buscar la ubicación de instalación del ManoDB y abrir una consola del sistema operativo allí.

      Nota: Generalmente la ruta de instalación en Windows es:

      C:\Program Files\MongoDB\Server\3.6\bin

   B. En la consola escribir:

      .\mongod.exe

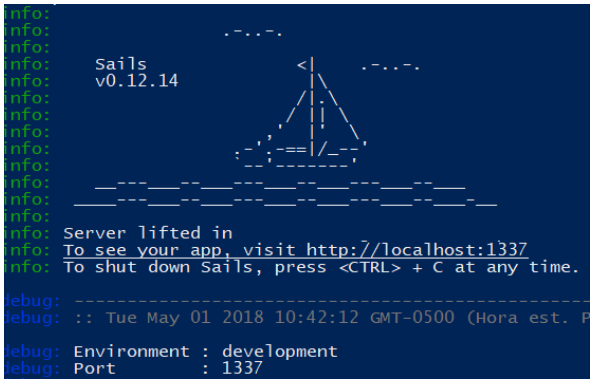
      Esto iniciará el servidor de MongoDB, haciendo que quede a la escucha de nuevas conexiones.

      Nota: No cerrar esa consola una vez se ha iniciado.

2) En la carpeta del proyecto, abrir una consola del sistema operativo y escribir:

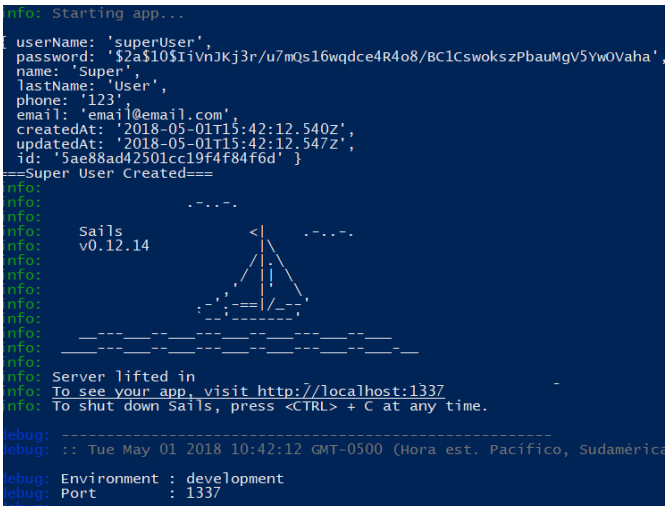
   sails lift

3) Se observará en la consola algo como eso:



   Lo cual indica que la aplicación está corriendo exitosamente.

Nota: Si se ha configurado el ./config/bootstrap.js para que inicialice la base de datos, se mostrará algo como esto:



4) Iniciar un navegador web e ir a la url:

http://localhost:1337

Nota: el usuario por defecto (administrador) tiene las siguientes credenciales:

UserName: superUser

Password: user951

5) En caso de ser pertinente, abrir el archivo ./config/bootstrap.js y comentar la línea `await startDB();`

Utilización

Toda url que se vaya a usar debe ser registrada y asociada con ciertos permisos a un rol de un usuario.

En el menú de “Security” en el sistema se puede ver información relacionada con:

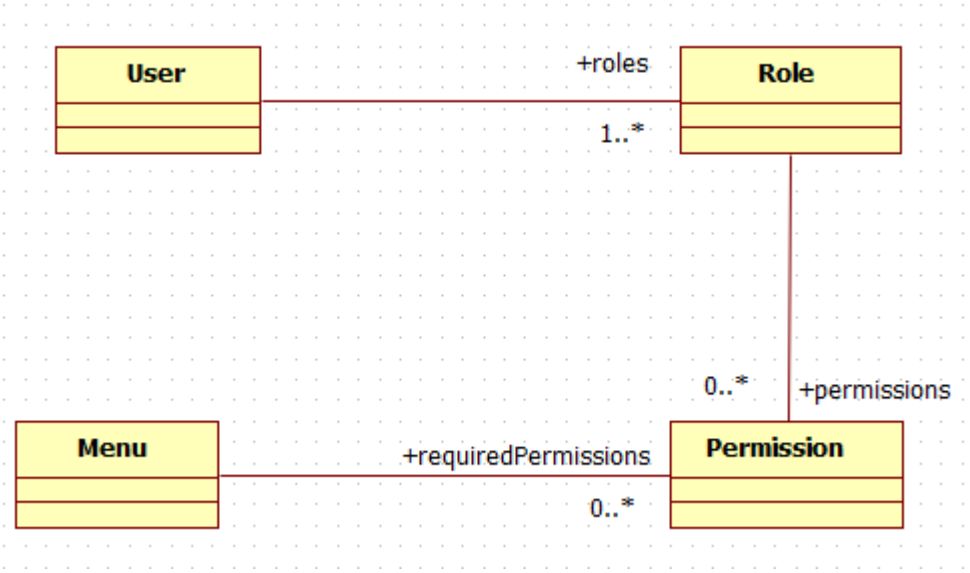
Permiso: Son usados para definir restricciones para ciertas urls.

Menú: Son usados para definir cada url que maneja el sistema y los diferentes menús que aparecen en la barra de navegación una vez se ha logueado un usuario.
Cada menú cuenta con una lista de permisos que son requeridos para para que se pueda acceder a dicha url.

Rol: Se usa para clasificar los diferentes tipos de usuarios que manejará el sistema, cada rol tiene una lista de permisos sobre el sistema.

Usuario: Son las entidades que pueden iniciar sesión en el sistema.

Un diagrama de clases **(Muy simplificado)** para explicar lo anterior sería:



Ejemplo de utilización

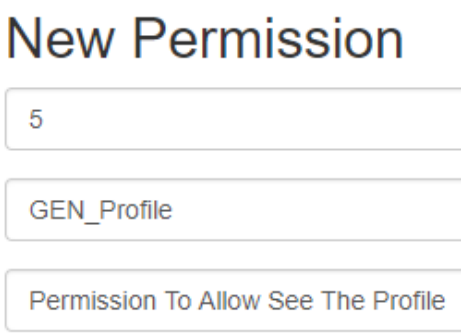
Como ejemplo se usará una vista de “Perfil” previamente programada, pero sin estar registrada la url en el sistema de seguridad, por lo que **es inaccesible a cualquier usuario.**

Si se intenta acceder a la url http://localhost:1337/profile saldrá una imagen como ésta, la cual indica que no existen permisos para acceder a dicha url.

Para tener acceso a la url, se tendrán que seguir los siguientes pasos:

1) Crear permiso

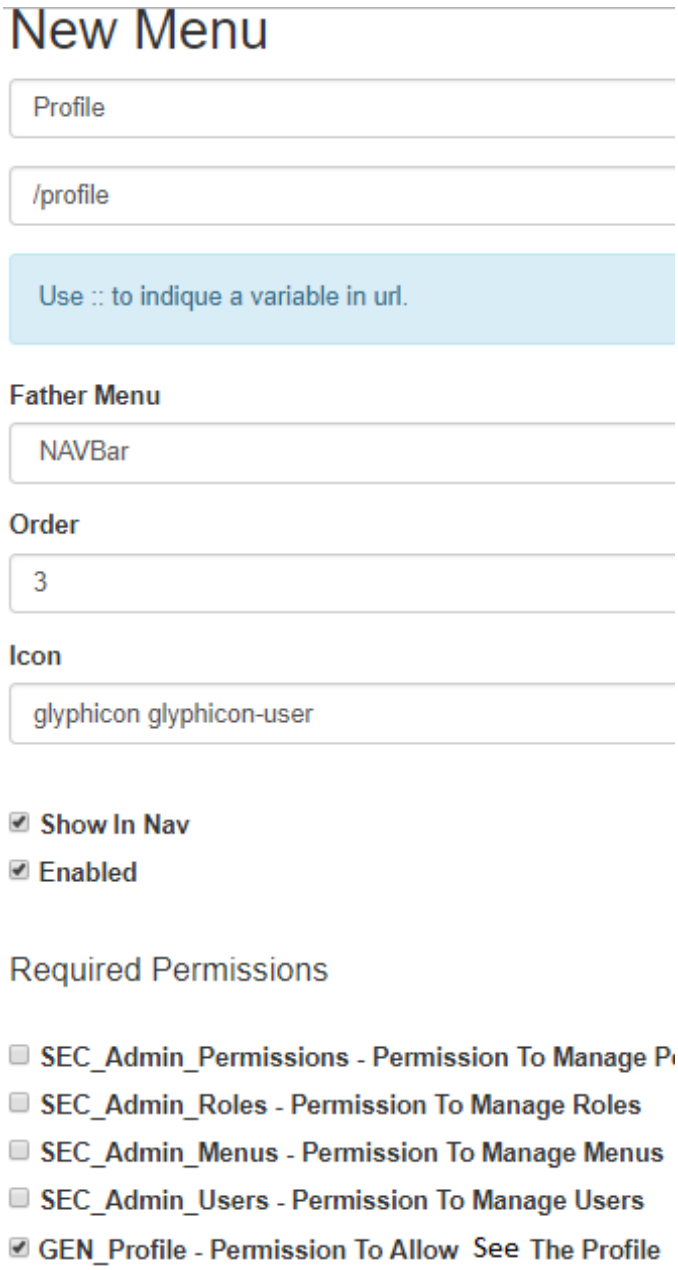
    a. Ingresar a http://localhost:1337/newPermission
    b. Llenar el formulario como éste:



ID: 5
Name: GEN_Profile
Desciption: Permission To Allow See The Profile

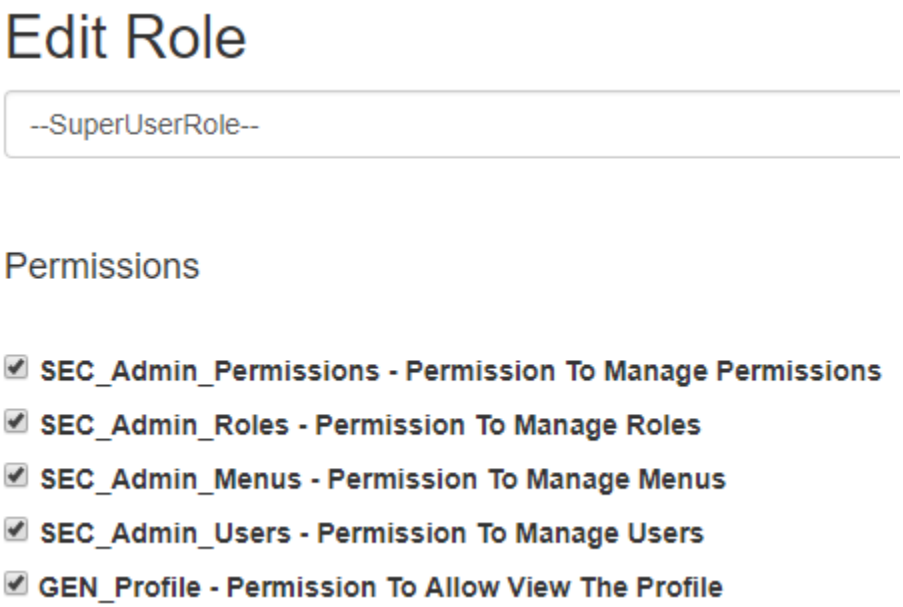
2) Crear Menú

a. Ingresar a http://localhost:1337/newMenu
b. Llenar el formulario como éste:



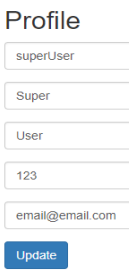
3) Asignar permiso al rol

   a. Ingresar a http://localhost:1337/listRoles
   b. Buscar el rol al que se le desea asignar el nuevo permiso, en este caso será el --SuperUserRole--
   c. Dar click en “edit”
   d. Asignar el nuevo permiso.



4) Una vez terminado esto se podrá observar un nuevo menú en la barra de navegación

Y la url http://localhost:1337/profile ya es accesible.



Descripción de archivos y carpetas importantes

1) Todo archivo y carpeta que su nombre comience con “SEC_” pertenecen al módulo de seguridad y **no deben ser editados**.
2) Carpeta “api” contiene el núcleo del proyecto.
    a. Carpeta “api/controllers” contiene los controladores del proyecto.

        En este sistema de seguridad, el archivo “api/controllers/PageController.js” corresponde al controlador de todas las vistas que **NO** requieren inicio de sesión.

        En este sistema de seguridad, el archivo “api/controllers/AppController.js” corresponde al controlador de todas las vistas que **requieren** inicio de sesión.

3) Carpeta “assets” contiene todos los archivos que pueden ser accedidos públicamente desde el navegador, como archivos css, js, jpg, etc
4) Archivo “config/bootstrap.js” contiene el código que se ejecuta antes de que empiece a correr la aplicación.

    En este sistema de seguridad se utiliza para inicializar la base de datos si se está corriendo el sistema por primera vez, des-comentando la línea “await startDB();”

5) Archivo “config/routes.js” contiene las rutas o urls que utiliza la aplicación y el controlador que responde a cada una de ellas.

    Ejemplo:

    '/profile': 'AppController.profile'

    Cuando se entre en el navegador web a la ruta “localhost:1337/profile” la aplicación ejecutará el código que se encuentre en el controlador “AppController” en la función “profile”

6) Carpeta “views” contiene todas las vistas (HTML) en el formato del motor de vistas de SailsJS… ejs.

    En este sistema de seguridad se estipuló que todo lo que esté en la carpeta “views/App” requiere que el usuario haya iniciado sesión, lo que esté por fuera no lo requiere.

Creación de una vista desde cero

A continuación, se creará una vista que será accedida solo por ciertos usuarios.

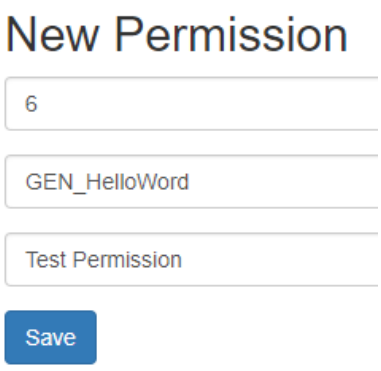
1) Crear la vista HTML (.ejs)
    a. En la carpeta “views/App” crear el archivo “holaMundo.ejs”
    b. Escribir dentro de él “<h4>Hola Mundo, soy una prueba</h4>”
2) En el archivo “api/controllers/AppController.js” crear una nueva función como esta:

```
holaMundo: async function(req, res){

        return res.view('App/holaMundo');

}
```

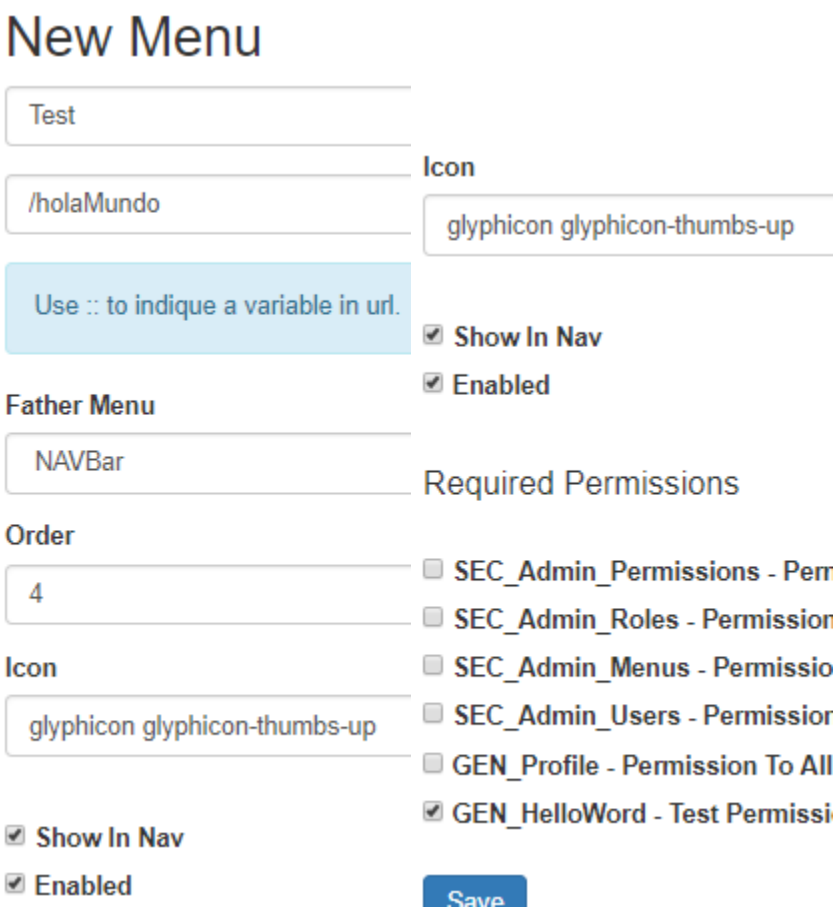
3) En el archivo “config/routes.js” ingresar la ruta a la nueva vista:

```
'/holaMundo': 'AppController.holaMundo'
```

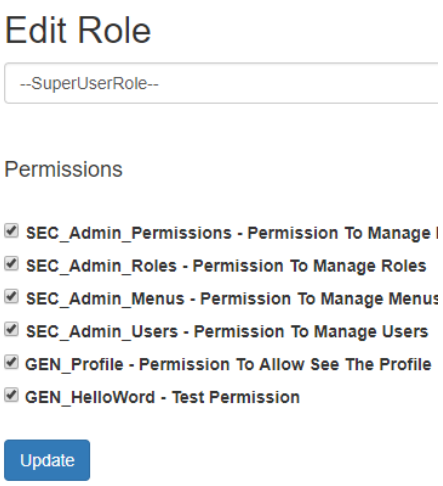
4) Correr la aplicación e iniciar sesión como súper administrador.
5) Crear un nuevo permiso para la nueva vista.



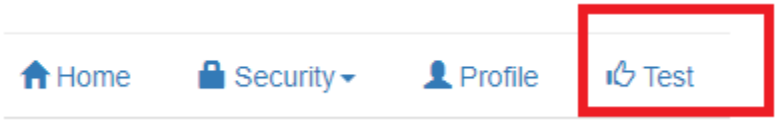
6) Crear el nuevo menú y asociar el nuevo permiso:



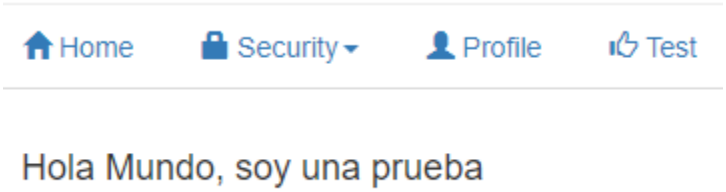
7) Asociar el permiso al rol:



En la barra de navegación debió haber aparecido un nuevo ítem.



Al ingresar a dicho menú debe salir, el mensaje programado anteriormente:



Funciones/Servicios importantes

1) SEC_FlashService.success(req, message);
   SEC_FlashService.warning(req, message);
   SEC_FlashService.error(req, message);

   Son funciones globales que pueden ser accedidas desde cualquier lado de la aplicación que se utilizan para enviar mensajes al usuario.

2) SEC_SecurityService.verifyPermissionsToPath(res, url)

   Es una función global que puede ser accedida desde cualquier lado de la aplicación que se utilizan para verificar si el usuario en sesión tiene permiso de acceder a una url.