**PRUEBA DE ORINA 5-HIAA: DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE INFORMACIÓN PARA EL REGISTRO Y CONSULTORÍA DE RESULTADOS (MANUAL DEL PERSONAL MÉDICO).**

**JAVIER ALEJANDRO QUINTERO RODRIGUEZ.**

**UNIVERSIDAD SURCOLOMBIANA**

**FACULTAD DE INGENIERÍA**

**PROGRAMA DE INGENIERÍA ELECTRÓNICA**

**NEIVA, COLOMBIA**

**2023**

Contenido

[**1.** **Archivo *server.js*** 4](#_Toc115194527)

[**1.1 Paquetes** 4](#_Toc115194528)

[**1.2 Inicialización** 4](#_Toc115194529)

[**1.3 Ajustes** 4](#_Toc115194530)

[**1.4 *Middlewares*** 5](#_Toc115194531)

[**1.5 Variables globales** 6](#_Toc115194532)

[**1.6 Enrutamiento** 6](#_Toc115194533)

[**2.** **Archivo *index.js*** 6](#_Toc115194534)

[**3.** **Archivo *database.js*** 7](#_Toc115194535)

[**4.** **Modelos de la base de datos** 8](#_Toc115194536)

[**5.** **Rutas** 9](#_Toc115194537)

[**6.** **Controladores** 10](#_Toc115194538)

[**7.** **Carpeta *Public*** 12](#_Toc115194539)

[**8.** **Carpeta *views*** 12](#_Toc115194540)

**Manual del programador**

En su versión de entrega, el aplicativo web contiene 7 carpetas: *config, controllers, helpers, models, public, routes, views* (después serán explicadas detalladamente) y 3 archivos individuales: *database.js, index.js, server.js*. También se instalaron paquetes y dependencias, los cuales son mostrados en la Tabla 1.

Tabla 1. Dependencias y paquetes utilizados en el aplicativo web

|  |  |
| --- | --- |
| **Módulo** | **Descripción** |
| Express | Permite la integración con motores de renderización de “vistas” para generar respuestas mediante la introducción de datos en plantillas y establece qué puerto se va a usar. |
| connect-flash | Paquete para Express que permite la comunicación entre vistas y muestra mensajes o alertas. |
| bcrypt.js | Es una función criptográfica de hash, que transforma cualquier bloque arbitrario de datos en una nueva serie de caracteres con una longitud fija, independientemente de la longitud de entrada. |
| Handlebars | Es un lenguaje de plantillas simple. Utiliza una plantilla y un objeto de entrada para generar HTML u otros formatos de texto. |
| Express-session | Almacena los datos de sesión en el servidor; sólo guarda el ID de sesión en la propia cookie, no los datos de sesión. |
| Method-override | Crea una nueva función de middleware para anular la propiedad req.method con un nuevo valor, gracias a este módulo es posible usar el método PUT para editar o eliminar documentos de la base de datos. |
| Mongoose | Biblioteca de programación orientada a objetos de JavaScript que crea una conexión entre MongoDB y el entorno de tiempo de ejecución de JavaScript de Node.js. |
| Passport | Middleware de autenticación para Node.js flexible y modular, se puede colocar discretamente en cualquier aplicación web basada en Express. |
| Nodemon | Herramienta que ayuda a desarrollar aplicaciones basadas en Node.js al reiniciar automáticamente la aplicación del nodo cuando se detectan cambios en los archivos del directorio. |

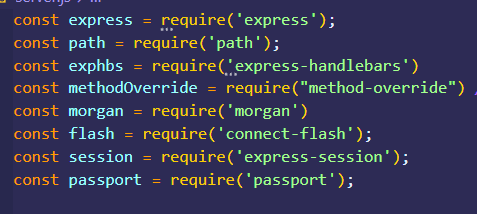
# **Archivo *server.js***

El archivo *server.js* se divide en 7 secciones: los paquetes que son requeridos para ser usados en el proyecto, las inicializaciones, los ajustes, los middlewares, variables globales, rutas y los archivos estáticos.

## **1.1 Paquetes**

En esta sección son llamados los paquetes ya mostrados en la Tabla *1*, En la Figura 1 se puede observar cómo son importados para su posterior uso.

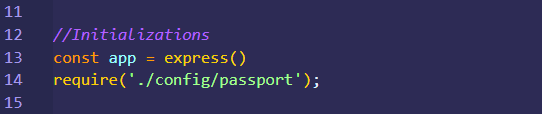
Figura 1. Importación de paquetes y dependencias.



## **1.2 Inicialización**

En la Figura 2 Se muestra la inicialización del servidor *Express* y el requerimiento del *middleware* *Passport* (línea 14), gracias a *Passport* es posible iniciar sesión y la autenticación del tipo de usuario.

Figura 2. Inicialización del servidor de Express



## **1.3 Ajustes**

En la Figura 3 se observan los ajustes del servidor, en la línea 17 se ajusta el puerto en el que trabajará el aplicativo, si todos están ocupados, el puerto por defecto con el que se trabajará será el puerto 3000. De la línea 18 a la línea 25 se establece la ruta en donde está la carpeta con las vistas del aplicativo y se establece que se trabajará con *Handlebars* y su correspondiente extensión ‘.hbs’.

Figura 3. Ajustes del servidor



## **1.4 *Middlewares***

En la Figura 4 se puede observar los *middlewares* usados en el desarrollo del aplicativo web, se usa Morgan en su versión desarrollador (“dev”), este *middleware* le indica al desarrollador qué tipo de error se está presentando en cierta solicitud. En la línea 31 se establece que express devuelva las peticiones y consultas en formato JSON, en la línea 32 declara el método *Override* para que se abra la posibilidad de editar o eliminar documentos dentro de las colecciones de la base de datos. En la línea 33 se declara que los archivos estáticos (hojas de estilo en cascada e imágenes) se encuentran dentro de la carpeta “*Public*”. Finalmente se declara que se usará el *middleware* *Passport* para el inicio de sesión y *flash* para mostrar notificaciones y alertas al usuario.

Figura 4. Declaración de Middlewares

Texto

Descripción generada automáticamente

## **1.5 Variables globales**

En las variables globales se estipularon los mensajes de alerta y notificación del aplicativo, tal como se puede observar de la línea 44 a la línea 50 de la Figura 5.

Figura 5. Definición de variables globales del aplicativo

Texto

Descripción generada automáticamente

## **1.6 Enrutamiento**

Lo último presente dentro del archivo del servidor (*express.js*)es colocar la dirección de los archivos que contienen las rutas de la aplicativo, tal como se aprecia en la Figura 6, son 3 archivos: *index.routes, users.routes* y *exams.routes.*

Figura 6. Requerimiento de los archivos que contienen las rutas del aplicativo

Texto

Descripción generada automáticamente

# **Archivo *index.js***

El archivo index.js se encarga inicializar el aplicativo, se hace el requerimiento del archivo del servidor (*server.js*) previamente descrito y del archivo de la base de datos que será explicado en la próxima sección. De la línea 6 a la línea 8 de la Figura 7 se observa la solicitud del puerto en donde está corriendo la plataforma, de la línea 1 a la 3 se observan los archivos requeridos.

Figura 7. Configuración del archivo index.js

Texto

Descripción generada automáticamente

# **Archivo database.js**

El archivo database.js contiene las credenciales de conexión a la base de datos (Observe la línea 4 y línea 9 de la Figura 8), por seguridad, la dirección y el nombre de la base de datos se declaran en las variables globales PSA\_APP\_HOST y PSA\_APP\_DATABASE, respectivamente, en el archivo *.gitignore* (observe la Figura 9)*,* ya que, cuando se suba a un servidor que no sea global, el archivo *.gitignore* no será visible.

Figura 8. Conexión a la base de datos

Captura de pantalla con letras

Descripción generada automáticamente

Figura 9. Credenciales para acceder a la base de datos

Texto

Descripción generada automáticamente

# **Modelos de la base de datos**

Los modelos de la base de datos se encuentran en la carpeta “models”. Hay dos modelos de base de datos: *users* y *exams*. el primero es el modelo de los usuarios, en donde se almacena la información de todos los usuarios sin importar el tipo de usuario (medico, paciente, empleado o administrador). Las siguientes propiedades de tipo *string*: *identification\_type, identification, name, lastname, sec\_lastname, date\_of\_birth, gender, blood\_type, rh, marital\_status, EPS, home\_phone, mobile\_phone, work\_phone, address, city, role, email, password, contact\_name, contact\_lastName, contact\_sec\_lastName, contact\_relationship* y *contact\_phone.* La única propiedad que es distinta a *string*, es *last\_login\_date* y el tipo de dato es date.

Dentro del modelo *users* se realiza la encriptación de la contraseña, tal como se muestra en la Figura 10 de la línea 122 a 129, también se realiza la exportación del modelo (línea 131).

Figura 10. encriptación de contraseña y exportación del modelo

Captura de pantalla de un celular

Descripción generada automáticamente

El segundo es el modelo de los exámenes, en donde se almacena tanto información de los exámenes, como información del paciente que lo está tomando. En la Tabla 2 se pueden observar las propiedades del esquema de los exámenes.

Tabla 2. Propiedades y descripción del modelo exams

|  |  |  |
| --- | --- | --- |
| **Propiedades** | **Tipo de dato** | **Descripción** |
| *Id* | ObjectId | Por defecto, Mongoose añade un ID único a cada documento. |
| *Result* | Number | El resultado una vez se ha tomado el examen. |
| *Description* | String | Breve análisis con base al resultado. |
| *date\_of\_exam* | String | Fecha en la que se programó el examen. |
| *Dx* | String | Dictamen por el cual se programa el examen. |
| *hour\_of\_exam* | String | Hora en la que se programa el examen. |
| *status* | String | Estado en el que se encuentre la orden (pendiente, cancelado, completo). |
| *whyCancel* | String | Razón por la que se cancela el examen. |
| *nivel* | String | El nivel en el que se encuentra el examen (bajo, normal, alto). |
| *user* | String | ObjectId del paciente. |
| *Patient* | Array | Contiene información de la paciente extraída de la colección “users”, guarda las propiedades “*name*”, “*lastName*”, “*identification*”, “*city*” y “*date\_of\_birth*”. |
| *Age* | Number | Edad del paciente. |
| *Group* | String | Grupo según rango de edad del paciente. |

# **Rutas**

La carpeta “routes” contiene tres archivos: *exams.routes.js, index.routes.js* y *users.routes.js*, como su nombre lo indica, son las rutas del aplicativo web. En la Tabla 3 se encuentran las rutas disponibles en el archivo exams.routes.js y en la Tabla 4 se observan las rutas disponibles en el archivo users.routes.js. En el archivo index.routes.js solo se encuentra disponible la ruta “/” que redirige al usuario a la página de inicio del aplicativo móvil. Cada una de las rutas mencionadas tienen su correspondiente controlador, o *controller*, que serán expuestos en la siguiente sección.

Tabla 3. Rutas disponibles en exams.routes.js

|  |
| --- |
| **Rutas** |
| */exams/findUserByIdentification* |
| */staff/orders* |
| */exams/edit/:id* |
| */exams-done* |
| */exams/exam/:id* |
| */exams-canceled* |
| */exams/cancelOrder/:id* |
| */exams/updateLastResult/:user* |
| */exams-patient* |
| */staff/delete/orders* |
| */exams/stadistics/user* |
| */exams/stadistics/examsPerCity* |
| /exams/stadistics/examsPerAge |
| /exams/stadistics/general |

Tabla 4. Rutas disponibles en users.routes.js

|  |
| --- |
| **Rutas** |
| */users/signInForm* |
| */users/signInError* |
| */users/started* |
| */users/logout* |
| */users/createUserForm* |
| */users/createAdminForm* |
| */users/seeAllPatientsAdmin* |
| */users/seeAllPhysicianAdmin* |
| */users/seeAllLabStaffAdmin* |
| */users/seeAllAdmin* |
| */users/seeAllUsersAdmin* |
| */users/findUserByIdentification* |
| */users/findUserByEmail* |
| */users/seeUserAdminForm/:id* |
| */users/editUserFormAdmin/:id* |
| */users/myProfile* |
| */users/changePasswd* |
| */users/forgotPasswdForm* |

# **Controladores**

Como se mencionó en la sección anterior, cada ruta tiene su correspondiente controlador, que se encargará ya sea de renderizar vistas, ejecutar el método Override, subir algún formulario a través del método POST, realizar consulta a las bases de datos, etc. Los archivos de los controladores se encuentran dentro de la carpeta “controllers” y se dividen, al igual que las rutas, en controladores de usuarios, controladores de exámenes y el controlador del índex. Para mirar el controlador correspondiente a cada ruta se selecciona el nombre del controlador y se presiona f12, esto redirigirá al desarrollador a la sección del archivo en donde se declara y desarrolla el ya mencionado controller. Desde la Tabla *5* hasta la Tabla *8* se puede encontrar la descripción de cada uno de los controladores utilizados en el aplicativo web, el programador puede utilizar esta información como un diccionario que describe la función que está realizando para aplicar algún cambio o mejor.

Tabla 5. Descripción de los controladores de las funciones del médico

|  |  |
| --- | --- |
| **Controlador** | **Descripción** |
| *findUserByIdentificationForm* | Renderiza el formulario para buscar pacientes con su número de cédula. |
| *findUserByIdentification* | Realiza la busqueda del número de cédula a través del método post. |
| *CreateNewExam* | Crea una nueva de orden de examen |
| *renderDoneExam* | Hace una consulta a la base de datos y renderiza los examenes con estado realizado (“Done”). |
| *renderDoneSingleExam* | Renderiza la información del examen que ya fue tomado, muestra el diagnostico diferencial, el resultado, comentarios adicionale, nombre del paciente, documento, entre otros. |

Tabla 6. Handlers para las rutas estadísticas

|  |  |
| --- | --- |
| **Handler** | **Descripción** |
| *renderStadisticsPerUser* | Consulta a la colección Exams y busca los examenes cuyo estado sea completado (status: done) y con el método count cuenta el total de documentos por cada nivel. Renderiza en un gráfico el total de resultados por niveles (bajo, normal, alto). |
| *renderStadisticsPerCity* | Realiza una busqueda en la colección Exams de los documentos cuyo estado sea completado (status: done) y con el método count entrega el total de examanes realizados por cada municipio.Realiza una busqueda en la colección Exams de los documentos cuyo nivel sea bajo (nivel: bajo) y con el método count entrega el total de documentos de esta busqueda.  Renderiza un gráfico de barras del total de examenes realizados por municipio y el total de examenes con resultados fuera del rango normal. |
| *renderStadisticsPerAge* | Realiza una busqueda en la colección Exams de los documentos cuyo estado sea completado (status: done) y con el método count entrega el total de examanes realizados por la propiedad group.  Renderiza un gráfico circular en donde se muestra el total de examenes realizados por grupos de edad. |

Tabla 7. Descripción de los controladores de las funciones del administrador

|  |  |
| --- | --- |
| **Controlador** | **Descripción** |
| *createUserForm* | Renderiza el formulario para la creación de un nuevo usario. |
| *createUser* | Crea un nuevo usuario usando el metodo Post. |
| *seeAllPatientsAdmin* | Consulta a la base de datos para encontrar a los usuarios con el rol “Paciente”, renderiza una vista y muestra en una lista la consulta. |
| *seeAllPhysicianAdmin* | Consulta a la base de datos para encontrar a los usuarios con el rol “Medico”, renderiza una vista y muestra en una lista la consulta. |
| *seeAllAdmin* | Consulta a la base de datos para encontrar a los usuarios con el rol “Admin”, renderiza una vista y muestra en una lista la consulta. |
| *seeAllUsersAdmin* | Consulta a la base de datos para encontrar a todos los uuarios, sin importar el rol, renderiza una vista y muestra en una lista la consulta. |
| *findUserByIdentificationForm* | Renderiza el formulario para la busqueda por número de identificación. |
| *findUserByIdentification* | Usa el método post para realiza la consulta por identificación. |
| *findUserByEmailForm* | Renderiza el formulario para la busqueda por correo electronico. |
| *findUserByEmail* | Usa el método post para realiza la consulta por correo electronico. |
| *editUserFormAdmin* | Renderizar el formulario para la actualización de usarios. |
| *editUserAdmin* | Actualiza los usuarios usando el metodo Put. |
| *deletUserAdmin* | Elimina usuarios. |
| *DeleteORdersList* | Renderiza el listado de las ordenes pendientes con la opción “eliminar”, solo el administrador tiene acceso a este listado. |
| *deleteExam* | Elimina la orden seleccionada a través del metodo delete. |

Tabla 8. Descripción de los controladores de las funciones del empleado

|  |  |
| --- | --- |
| **Controlador** | **Descripción** |
| *renderCreatedOrders* | Realiza una consulta a la base de datos y renderiza el total de examenes para el día actual, su respectivo listado y un listado de los examenes pendientes posteriores. |
| *renderEditForm* | Renderiza el formulario para agregar los resultados correspondiente a la orden. |
| *updateExam* | Actualiza el documento correspondiente a la orden de la que se tomó el examen. |
| *renderCanceledOrders* | Realiza una consulta a la base de datos y renderiza un listado de las ordenes que fueron canceladas por los empleados. |
| *cancelOrderForm* | Renderiza el formulario para la cancelación de examenes. |
| *cancelOrder* | Cambia el estado de la orden creada a cancelada a través del metodo update. |

# **Carpeta *Public***

Dentro de esta carpeta se encuentran dos carpetas más, la carpeta CSS que contiene los estilos del aplicativo web y la carpeta img, en donde se encuentra el contenido multimedia.

# **Carpeta *views***

En esta carpeta se encuentran todas las vistas de la aplicación web, están agrupadas en cuatro carpetas más:

* *Exams*: contiene las vistas correspondientes a los exámenes, aquí se pueden encontrar otras dos carpetas donde se agrupan las vistas de los pacientes y del personal médico, tal como se puede observar en la Figura 11.

Figura 11. Vistas de la carpeta exams

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* *Partials*: en esta carpeta se encuentran las vistas parciales de la plataforma, como el archivo *head*, *footer,* la barra de navegación *navbar y* la sección de JavaScript *script*, la lista completa de los archivos de esta carpeta se pueden observar en la Figura 12.

Figura 12. Archivos de la carpeta Partials

Texto

Descripción generada automáticamente con confianza baja

* *Users*: En esta carpeta se encuentran las vistas de los formularios para la creación, actualización y eliminado de perfiles.
* *Layouts*: Dentro de esta carpeta solo se encuentra el archivo *Main.hbs,* que es el encargado de renderizar los archivos y las vistas parciales mencionadas anteriormente.