**Universidad Valle del Momboy**

**Vicerrectorado académico**

**Facultad de Ingeniería**

**Escuela Ingenieria de Computacion**

**Informe Técnico:**

**Red Social UVM**

**Alumno:** Daniel Fernandez.

**Alumno:** Luis Orlando.  
**Tutor:** Prof. Roberto Di Michelle

**INTRODUCCION.**

Este informe técnico detalla la arquitectura y funcionalidad de un proyecto de programación backend desarrollado con JavaScript utilizando Express Generator y Node.js. Este sistema está diseñado para ofrecer una API RESTful eficiente, escalable y fácil de mantener, utilizando MongoDB como base de datos.

**Estructura del Proyecto**

La estructura del proyecto se organiza de la siguiente manera:

/proyecto-backend

│

├── /bin

│ └── www

│

├── /src

│ ├── /controllers

│ ├── /database

│ ├── /jwt

│ ├── /routes

│ ├── /models

│ ├── /views

│ └── /public

│

├── .env

├── app.js

├── package.json

├── .gitignore

└── package-lock.json

**Descripción de Carpetas y Archivos**

1. Carpeta bin  
     
   • www: Este archivo es el punto de entrada de la aplicación. Contiene la configuración inicial del servidor, incluyendo la creación de la instancia de Express, la configuración del puerto y el manejo de errores. Es responsable de iniciar la aplicación y escuchar las solicitudes entrantes. Se encarga de establecer el entorno de ejecución y puede incluir lógica para manejar diferentes entornos (desarrollo, producción).
2. **Carpeta src:**  
     
   Esta carpeta contiene el código fuente principal de la aplicación, organizado en subcarpetas que agrupan funcionalidades específicas:  
     
   • /controllers: Esta carpeta alberga los controladores que gestionan la lógica de negocio de la aplicación. Cada controlador se encarga de recibir las solicitudes HTTP, procesar los datos y devolver las respuestas adecuadas. Los controladores actúan como intermediarios entre los modelos y las rutas, facilitando la separación de preocupaciones en el diseño del software.  
     
   • /database: Aquí se encuentran los archivos relacionados con la conexión a la base de datos MongoDB. Esto incluye la configuración para establecer la conexión, así como scripts para inicializar la base de datos o realizar migraciones. Esta carpeta es crucial para facilitar la interacción entre la aplicación y MongoDB, permitiendo operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sobre los documentos.  
     
   • /jwt: Esta carpeta contiene la lógica relacionada con JSON Web Tokens (JWT). Incluye funciones para generar y verificar tokens, así como middleware para proteger rutas que requieren autenticación. La implementación de JWT permite gestionar sesiones de usuario de forma segura y sin estado.  
     
   • /routes: En esta carpeta se definen las rutas de la aplicación. Cada archivo en esta carpeta representa un conjunto de rutas relacionadas, que asignan URLs específicas a sus respectivos controladores. Permite organizar las rutas de manera modular y facilita la gestión del enrutamiento en la aplicación.  
     
   • /models: Esta carpeta contiene los modelos de datos que representan las entidades en MongoDB. Utilizando Mongoose (una biblioteca ODM para MongoDB), los modelos definen la estructura de los documentos, así como métodos para interactuar con ellos. Esto incluye validaciones, métodos de búsqueda y lógica relacionada con los datos.  
     
   • /views: Aunque este es un proyecto backend centrado en una API RESTful, si se utiliza algún motor de plantillas (como EJS o Pug), esta carpeta contendría las vistas que se renderizan en el lado del servidor. Sin embargo, en un enfoque puramente API, esta carpeta puede no ser necesaria.  
     
   • /public: Aquí se almacenan los archivos estáticos que se sirven directamente al cliente, como imágenes, estilos CSS y scripts JavaScript del lado del cliente. Estos archivos son accesibles sin necesidad de pasar por el controlador, facilitando la entrega rápida de contenido estático.
3. **Archivos en la raíz del proyecto:**

• .env: Este archivo se utiliza para almacenar variables de entorno sensibles, como credenciales de acceso a MongoDB o claves API. Permite mantener configuraciones específicas fuera del código fuente y facilita el manejo de diferentes entornos (desarrollo, prueba, producción). Este archivo no debe ser compartido públicamente.  
  
• app.js: Este archivo es el núcleo de la aplicación. Aquí se configuran middlewares (como body-parser, cors), se establecen las rutas principales y se inicia el servidor. Es el punto donde se orquesta toda la funcionalidad del backend, integrando todos los componentes necesarios para el funcionamiento adecuado del sistema.  
  
• package.json: Este archivo contiene metadatos sobre el proyecto, incluyendo su nombre, versión, descripción, dependencias y scripts. Es esencial para gestionar las bibliotecas necesarias y las configuraciones del proyecto. Permite a npm (Node Package Manager) instalar todas las dependencias requeridas para el funcionamiento del proyecto.  
  
• .gitignore: Este archivo especifica qué archivos o carpetas deben ser ignorados por Git al realizar un seguimiento de los cambios. Generalmente incluye archivos temporales, dependencias instaladas y credenciales sensibles que no deben ser compartidas. Esto ayuda a mantener el repositorio limpio y seguro.  
  
• package-lock.json: Este archivo se genera automáticamente cuando se instalan las dependencias del proyecto. Proporciona un registro detallado de las versiones exactas de cada paquete instalado, asegurando que el entorno sea reproducible en diferentes máquinas. Facilita la gestión de dependencias y previene problemas derivados de cambios inesperados en las bibliotecas utilizadas.

**Detalle de Peticiones HTTP**

**1) Peticiones POST**  
  
Las siguientes son las peticiones POST que permiten la creación de recursos o la ejecución de acciones en el sistema:  
  
| URL | Body |  
|-------------------------------------------------|-------------------------------------------------------------------------------------------------------|  
| http://localhost:3060/users/register          | { "username": "", "email": "", "rol": "", "password": "", "passwordConfirm": "" }                |  
| http://localhost:3060/posts/add/:usuario-que-publica | { "token": "", "title": "", "description": "", "url\_media": "" }                                  |  
| http://localhost:3060/comments/add/:usuario-que-comenta/:id-publicacion | { "token": "", "comment": "" }                                                                     |  
| http://localhost:3060/requests/send/:usuario-que-envia/:usuario-que-recibe | { "token": "" }                                                                                     |  
| http://localhost:3060/requests/accept/:usuario-que-acepta/:usuario-que-envio | { "token": "" }                                                                                     |  
| http://localhost:3060/users/login             | { "username": "", "password": "" }                                                                |

**2) Peticiones GET**  
  
Las siguientes son las peticiones GET que permiten la recuperación de datos del sistema:  
  
| URL | Body |  
|-------------------------------------------------|-------------------------------------------------------------------------------------------------------|  
| http://localhost:3060/users/posts/:usuario    | { "token": "" }                                                                                     |  
| http://localhost:3060/users/feed/:usuario     | { "token": "" }                                                                                     |  
| http://localhost:3060/posts/comments/:id-de-la-publicacion | { "token": "" }                                                                                     |  
| http://localhost:3060/requests/listFriends     | { "token": "", "username": "" }                                                                    |  
| http://localhost:3060/requests/listRequest     | { "token": "", "username": "" }                                                                    |  
  
**3) Peticiones PUT**  
  
Las siguientes son las peticiones PUT que permiten la actualización de recursos existentes en el sistema:  
  
| URL | Body |

|-------------------------------------------------|-------------------------------------------------------------------------------------------------------|  
| http://localhost:3060/users/update/:id-usuario | { "token": "", "email": "", "password": "", "passwordConfirm": "" }                               |  
| http://localhost:3060/posts/update/:id-publicacion | { "title": "", "description": "", "url\_media": "" }                                              |  
  
4) **Peticiones DELETE**  
  
Las siguientes son las peticiones DELETE que permiten la eliminación de recursos en el sistema:  
  
| URL | Body |  
|-------------------------------------------------|-------------------------------------------------------------------------------------------------------|  
| http://localhost:3060/users/delete/:id-usuario | { "token": "" }                                                                                     |  
| http://localhost:3060/posts/delete/:id-publicacion | { "token": "" }                                                                                     |

CONCLUSIONES.

El proyecto backend desarrollado con Express Generator y Node.js presenta una arquitectura bien definida que facilita el desarrollo y mantenimiento a largo plazo. La separación clara entre controladores, modelos y rutas permite una fácil escalabilidad y adaptabilidad a futuras necesidades. La implementación adecuada de MongoDB como base de datos junto con JWT para autenticación asegura que el sistema sea robusto y eficiente en el manejo de solicitudes HTTP. Esta estructura modular no solo mejora la legibilidad del código, sino que también promueve buenas prácticas en el desarrollo de software.