# UVM Forms

Luis Monsalve, José Escalona, Maikel Villegas

Entrega de Informe

Facultad de Ingeniería, Universidad “Valle del Momboy”

Front End I

Brian Santeliz

# Desarrollo

Diseño Propuesto: La Rest-API permite a los usuarios registrarse e iniciar sesión para crear, modificar y eliminar sus propios quizzes. También pueden responder a los quizzes creados por otros usuarios y ver las respuestas de otros usuarios a sus propios quizzes. Los usuarios pueden buscar quizzes por categoría o por palabra clave, así como obtener una lista de los quizzes más populares. La API utiliza Node.js y Express para manejar las solicitudes HTTP entrantes y MongoDB para almacenar los datos. Los datos se presentan en un sitio web implementado con EJS y CSS para proporcionar una experiencia de usuario agradable. Además, la API cuenta con medidas de seguridad para proteger los datos de los usuarios y evitar la exposición de información confidencial con medidas de seguridad para proteger los datos de los usuarios y evitar la exposición de información confidencial.

# Creación de la Base de Datos

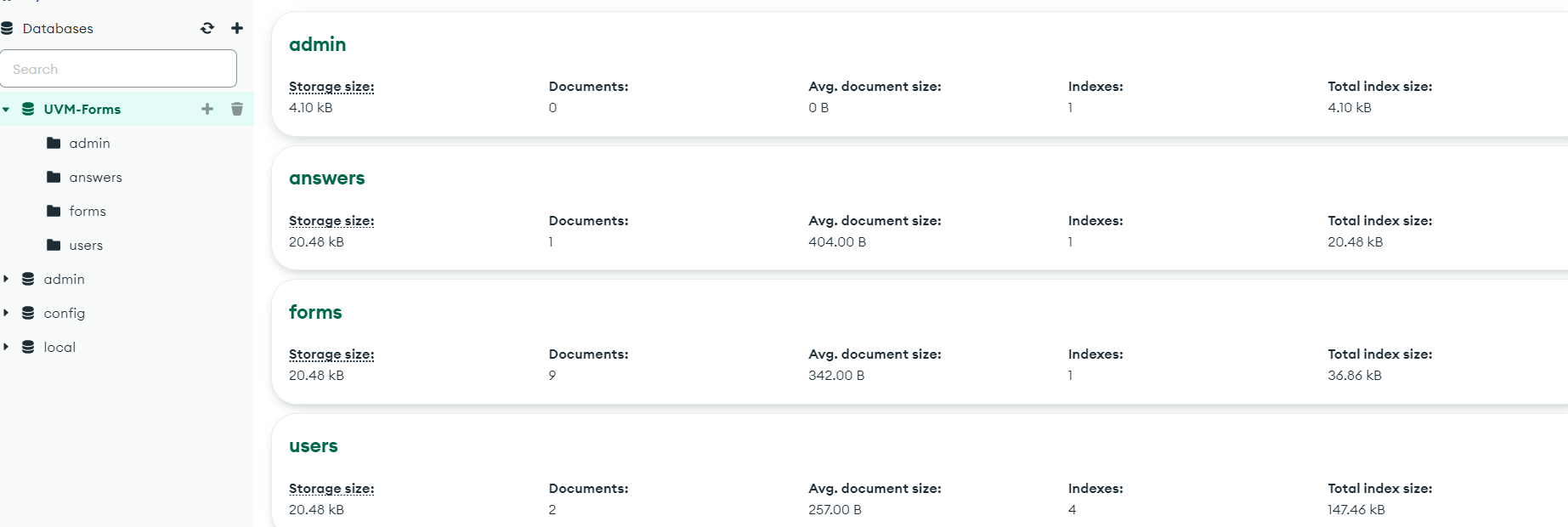
La base de datos "UVM-Forms" almacena información relacionada con formularios y encuestas.

La colección "answers" de la base de datos "UVM-Forms" almacena las respuestas de los estudiantes a las encuestas y formularios. Cada respuesta se identifica de forma única mediante un campo "\_id". Además, cada respuesta está asociada a un formulario específico mediante un campo "id\_encuesta" y a un estudiante específico mediante un campo "id\_User". Las respuestas a cada campo del formulario se almacenan en un array llamado "respuestas". El campo "respuestas" contiene un objeto por cada campo del formulario, y cada objeto contiene la respuesta del estudiante a ese campo en particular. Cada objeto de respuesta contiene información como el tipo de campo (por ejemplo, texto, opción múltiple, selección única), el valor de la respuesta y cualquier otra información relevante para ese tipo de campo. La colección "answers" también contiene campos de fecha de creación y modificación para cada respuesta.

La colección "forms" de la base de datos "UVM-Forms" almacena información sobre los formularios y encuestas creados por los usuarios. Cada formulario se identifica de forma única mediante un campo "\_id" y contiene información como el título, la descripción y el autor del formulario. Los campos del formulario se almacenan en un array llamado "preguntas". Cada pregunta del formulario contiene información como el tipo de campo, el texto de la pregunta y cualquier otra información relevante para ese tipo de campo. La colección "forms" también contiene campos de fecha de creación y modificación para cada formulario.

La colección "usuarios" de la base de datos "UVM-Forms" almacena información sobre los usuarios del sistema. Cada usuario se identifica de forma única mediante un campo "\_id" y contiene información como el nombre, la cédula, el correo electrónico, el nombre de usuario y la contraseña del usuario. El campo "rol" indica el rol del usuario en el sistema (por ejemplo, "administrador" o "usuario"). La colección "usuarios" también contiene campos de fecha de creación y modificación para cada usuario.

***Base de Datos***



***Diagrama propuesto***

# Dependencias de Desarrollo

bcryptjs: Una librería para el hashing de contraseñas.

connect-flash: Un middleware de Express para mostrar mensajes flash en la aplicación.

cookie-parser: Un middleware de Express para manejar cookies en la aplicación.

ejs: Un motor de plantillas para generar HTML dinámico en el servidor.

express: Un framework de Node.js para construir aplicaciones web.

express-handlebars: Un motor de plantillas para generar HTML dinámico en el servidor con soporte para Handlebars.

express-session: Un middleware de Express para manejar sesiones en la aplicación.

jsonwebtoken: Una librería para la autenticación basada en tokens JWT.

method-override: Un middleware de Express para habilitar HTTP PUT y DELETE en navegadores que no los soportan.

mongoose: Una librería para interactuar con bases de datos MongoDB de manera más fácil y con menos código.

passport: Un framework de autenticación para Node.js.

passport-local: Una estrategia de autenticación para Passport que utiliza nombre de usuario y contraseña.

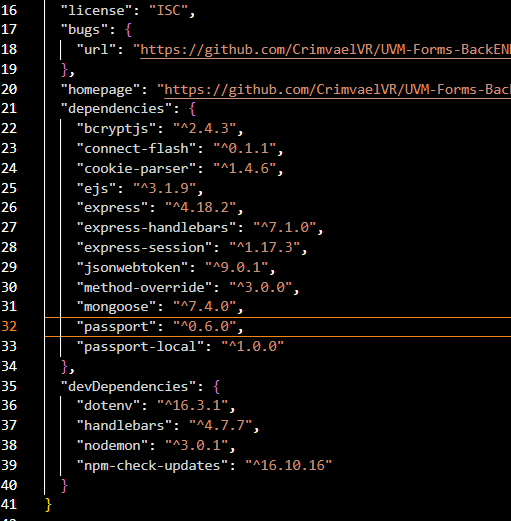
En cuanto a las dependencias de desarrollo:

dotenv: Una librería para cargar variables de entorno desde un archivo .env.

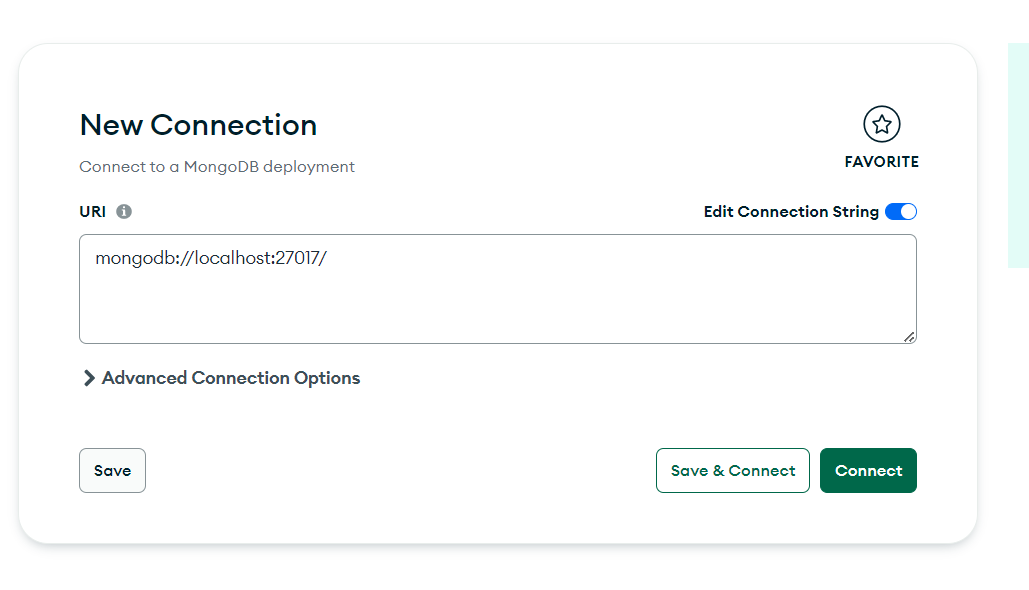
handlebars: Un motor de plantillas para generar HTML dinámico en el servidor con soporte para Handlebars.

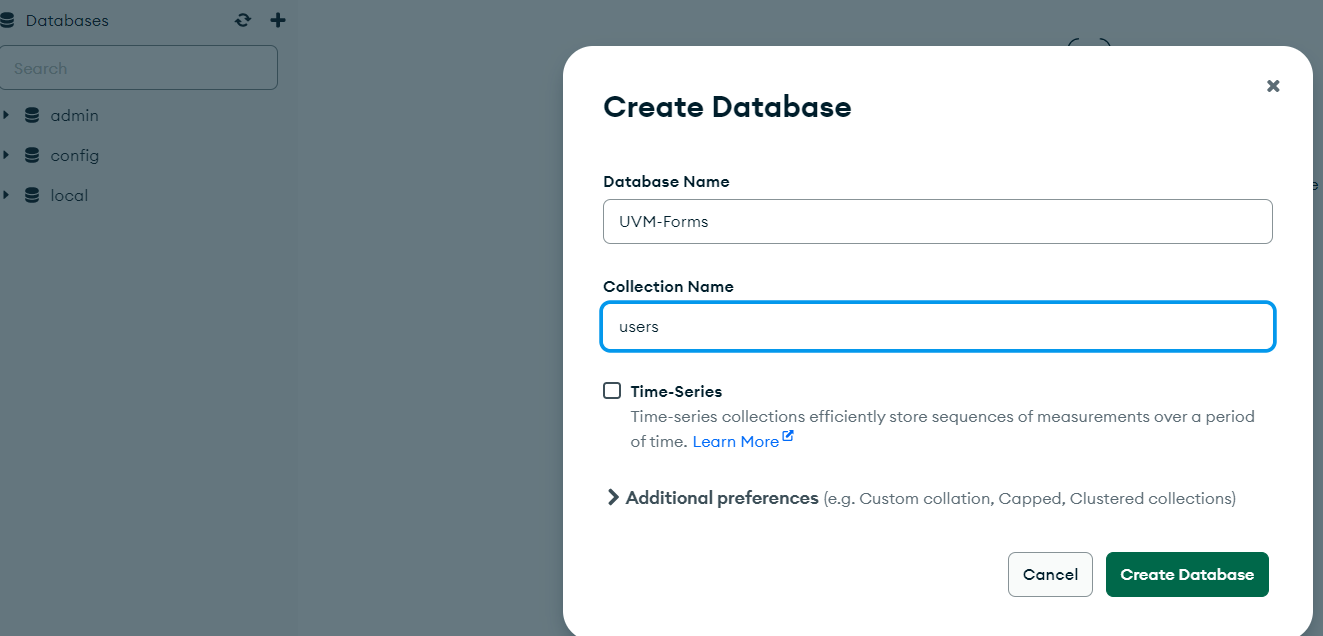
nodemon: Una herramienta que reinicia automáticamente la aplicación cuando se detectan cambios en el código fuente.

npm-check-updates: Una herramienta para actualizar las dependencias del proyecto a sus últimas versiones compatibles.

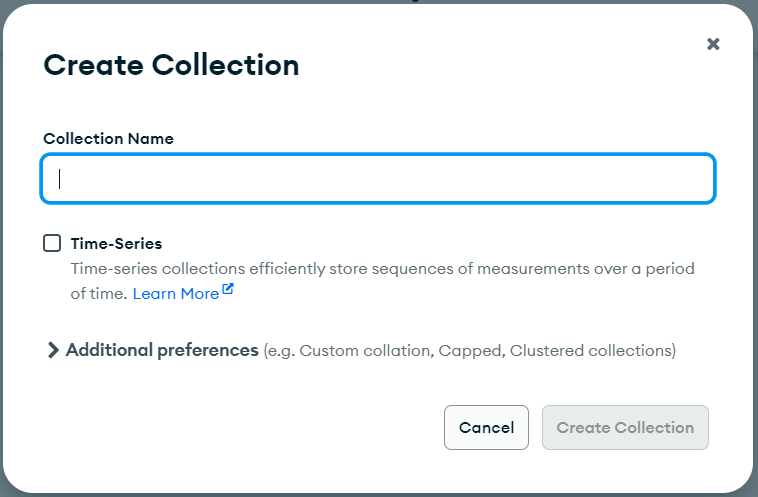


# Instalación

1. Antes de instalar requerirás los siguientes programas y componentes: Visual Studio Code, Node JS, MongoDB, MongoDB Server y MongoDB Compass
2. Lo primero es descargar ambos repositorios: <https://github.com/CrimvaelVR/UVM-Forms-FrontEND> (El front) y <https://github.com/CrimvaelVR/UVM-Forms-BackEND>
3. Luego de descargarlos vamos a pasar el contenido de la carpeta de Front-End al src del Back-End
4. Lo siguiente es hacer la DB a partir de MongoDB Compass, vamos a abrir el programa y conectaremos hacia la siguiente dirección: 
5. Crearemos una base de datos desde el signo de suma de arriba a la izquierda y asignaremos el nombre de la base de datos que necesitamos(UVM-Forms) y la creación de una de las colecciones necesarias



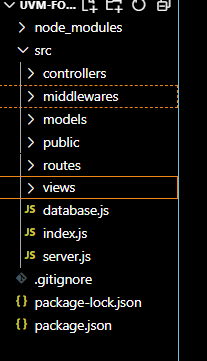
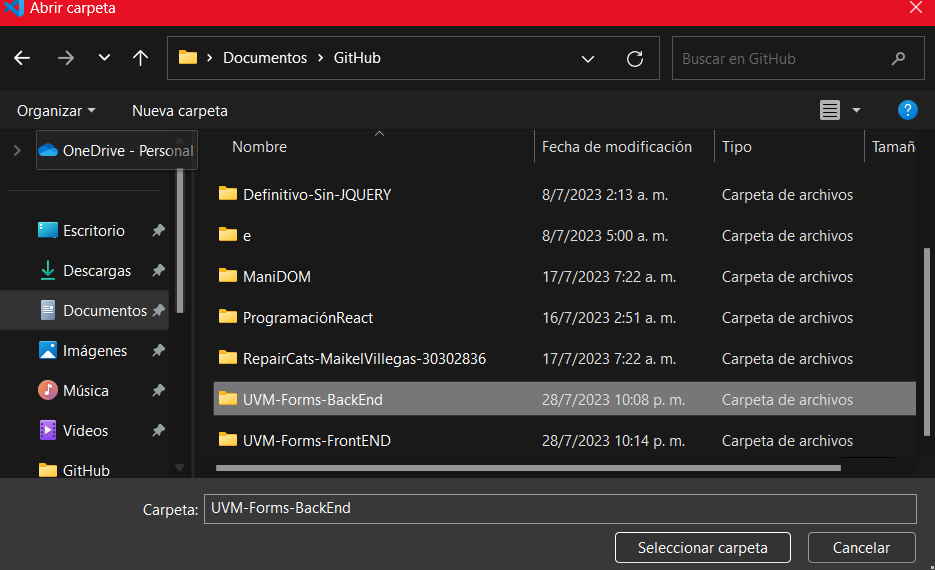
1. Crearemos las otras dos colecciones(answers y forms) necesarias con el signo de agregar que se encuentra al lado de la BD



1. Nos debería quedar como fue mencionado anteriormente en el documento
2. Ahora, abrimos Visual Studio Code y seleccionamos en Archivo/Abrir Carpeta, luego de

esto seleccionan la carpeta de App\_Backend, de esta manera podrán ver el código del

proyecto y podrán inicializarlo

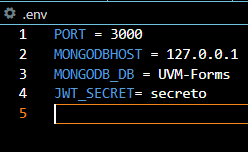


1. Como se percatarán en la carpeta descargada no se encuentra el archivo “. env”, eso es

porque debemos crearlo para poder acceder a la base de datos en nuestro servidor, para

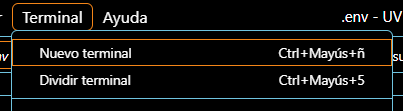
eso simplemente creamos un archivo llamado “. env” (se puede realizar con el Icono de

Archivo+), y colocaremos las siguientes variables, también llamadas variables de entorno

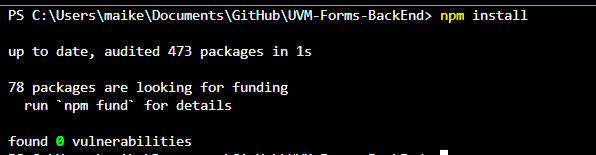


1. Lo siguiente es crear una terminal desde Visual Studio Code, con las teclas

CTRL+Mayus+Ñ o desde Terminal/Nuevo Terminal



1. Antes de inicializar el servidor, colocaremos un npm install en la terminal para instalar todas las dependencias



1. Ahora usamos npm run dev y si tenemos todo correcto nos saldrá lo siguiente



1. Ahora por ejemplo, si vamos a localhost:3000 podremos ver la landing page