# Aviary Cocteles

Juan González, Sergio González, Luis Martinez, Maria Suarez, Zeuddy Segovia

Entrega de Informe

Facultad de Ingeniería, Universidad “Valle del Momboy”

Front End I

Ing. Freddy Ramirez

# Desarrollo

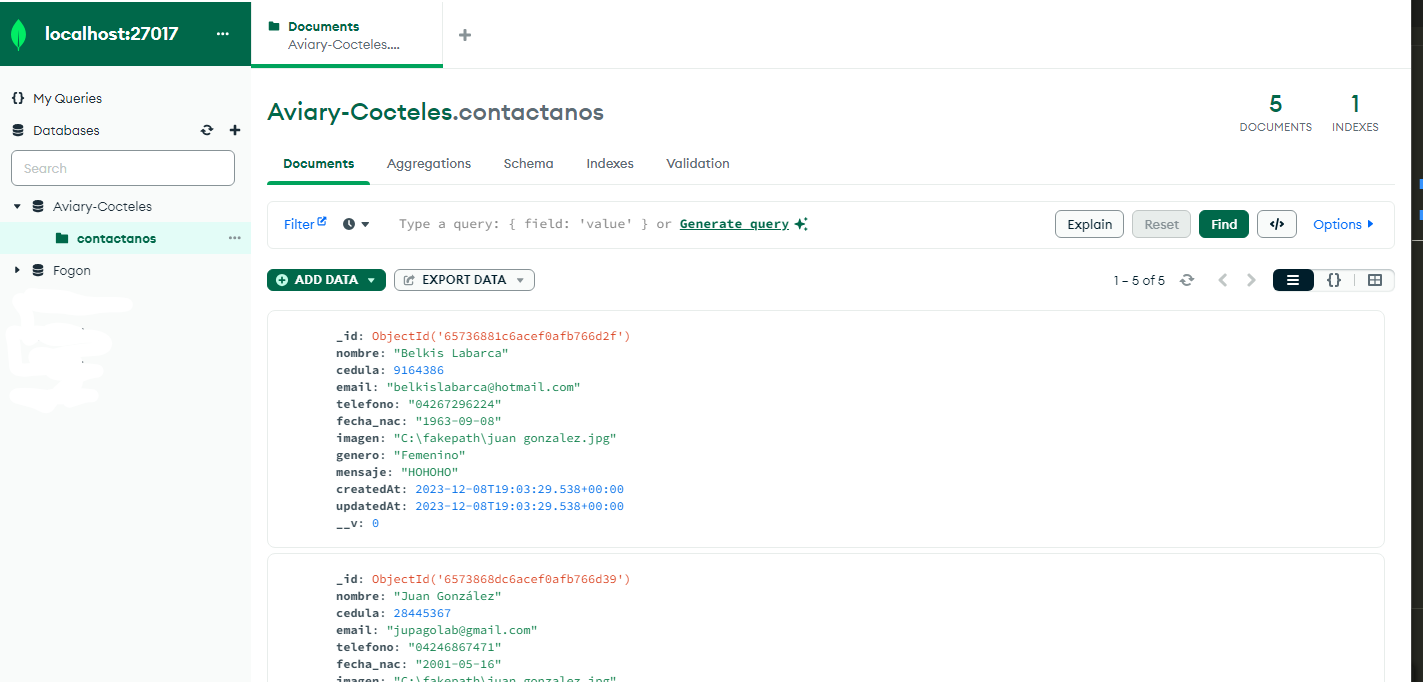
Diseño Propuesto: La Rest-API permite a los usuarios acceder a información de preparación y detalles como ingredientes y medidas de las diferentes diferentes variedades de cocteles, consumidas desde la api pública The CocktailDB, que inicialmente brinda su información en inglés, pero al mismo tiempo requerimos el consumo de otra api para traducir dicha información obtenido con la api Text Translator 2. La APP 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 HTML y CSS para proporcionar una experiencia de usuario agradable.

# Creación de la Base de Datos

La base de datos "Avairy-Cocteles" almacena información referente a los mensajes que envían los visitantes de la página, en la sección de contáctanos.

La colección "contactanos" de la base de datos " Avairy-Cocteles " almacena dichos mensajes que son enviados, en caso de querer comunicarse con los desarrolladores, por soporte o cualquier otro motivo. Cada mensaje se identifica de forma única mediante un campo "\_id". Las respuestas a cada campo del formulario se almacenan en un array llamado "respuestas".

***Base de Datos***



# Dependencias del proyecto

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

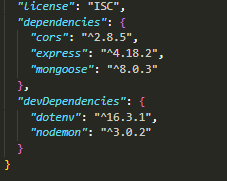
mongoose: Una librería para interactuar con bases de datos MongoDB de manera más fácil y con menos código.

cors: es un paquete de Node.js que facilita la gestión de la política de mismo origen (CORS) en aplicaciones web.

**Dependencias de desarrollo (BackEnd):**

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

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



**Dependencias de Desarrollo (FrontEnd)**

@babel/core: Núcleo de Babel.

@babel/plugin-transform-runtime: Plugin para reducir duplicación de código.

@babel/preset-env: Conjunto de configuraciones para utilizar características de JavaScript moderno.

babel-loader: Cargador para integrar Babel en Webpack.

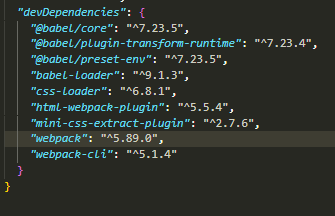
css-loader: Cargador para importar archivos CSS en la aplicación.

html-webpack-plugin: Plugin para generar archivos HTML automáticamente.

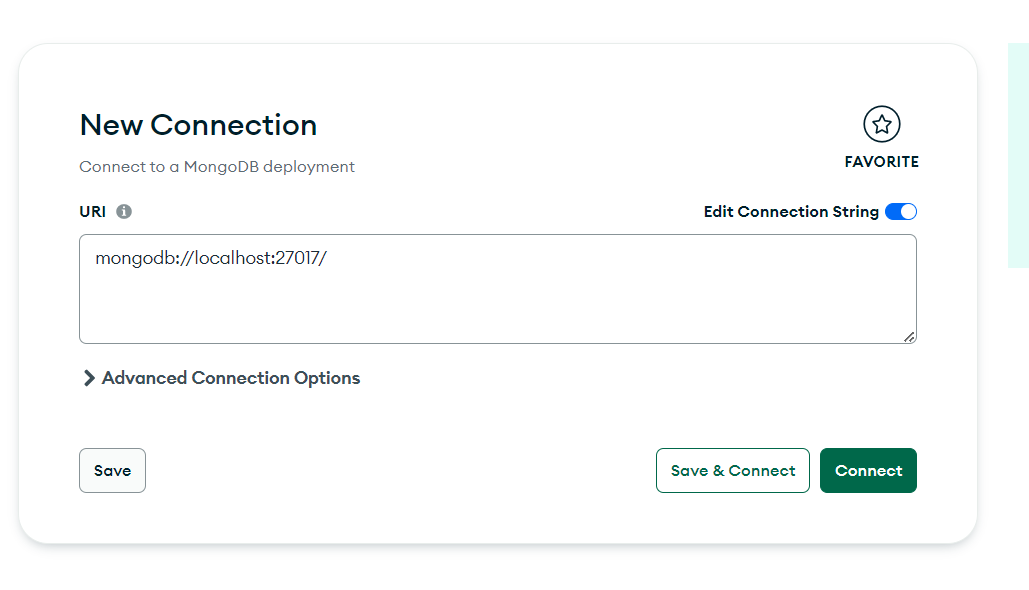
mini-css-extract-plugin: Plugin para extraer código CSS en archivos separados.

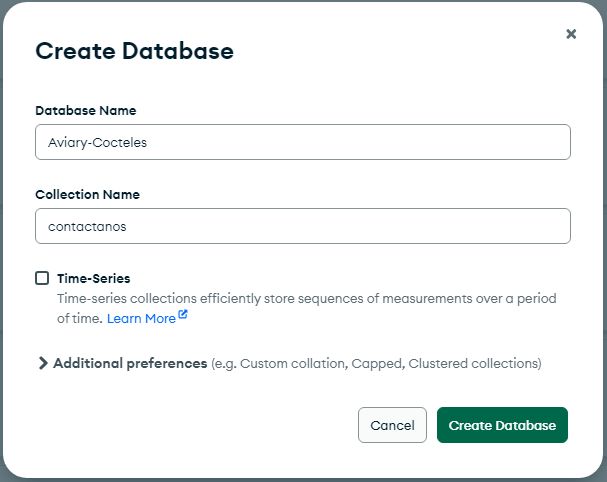
webpack: Empaquetador de módulos.

webpack-cli: Interfaz de línea de comandos para Webpack.



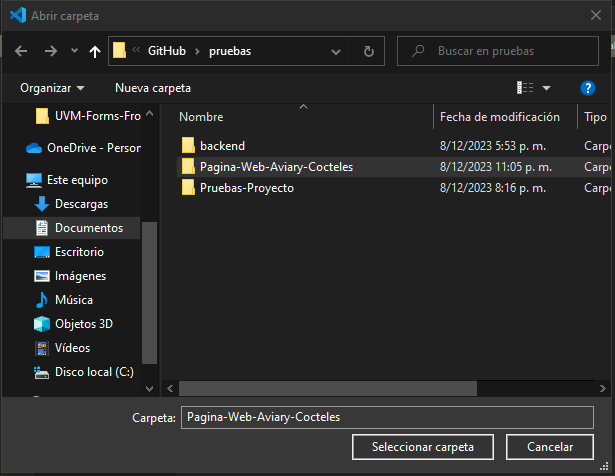
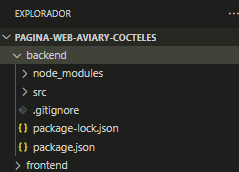
# Instalación

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 clonar el repositorio: <https://github.com/SergioG19/Pagina-Web-Aviary-Cocteles.git>
3. Luego de descargarlo, instalamos las dependencias necesarias de cada carpeta (backend y frontend)
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(Aviary-Cocteles) y la creación de una de la colección utilizada en este proyecto, en este caso ‘contactanos’



Nos debería quedar como fue mencionado anteriormente en el documento.

1. Ahora, abrimos Visual Studio Code y seleccionamos en Archivo/Abrir Carpeta, luego de esto seleccionan la carpeta de nuestro repositorio clonado, de esta manera podrán ver el código del proyecto y podrán inicializarlo.

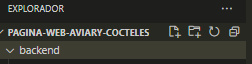
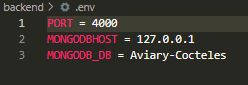
 

1. Si nos percatamos en la carpeta backend no se encuentra el archivo “. env”, eso es

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

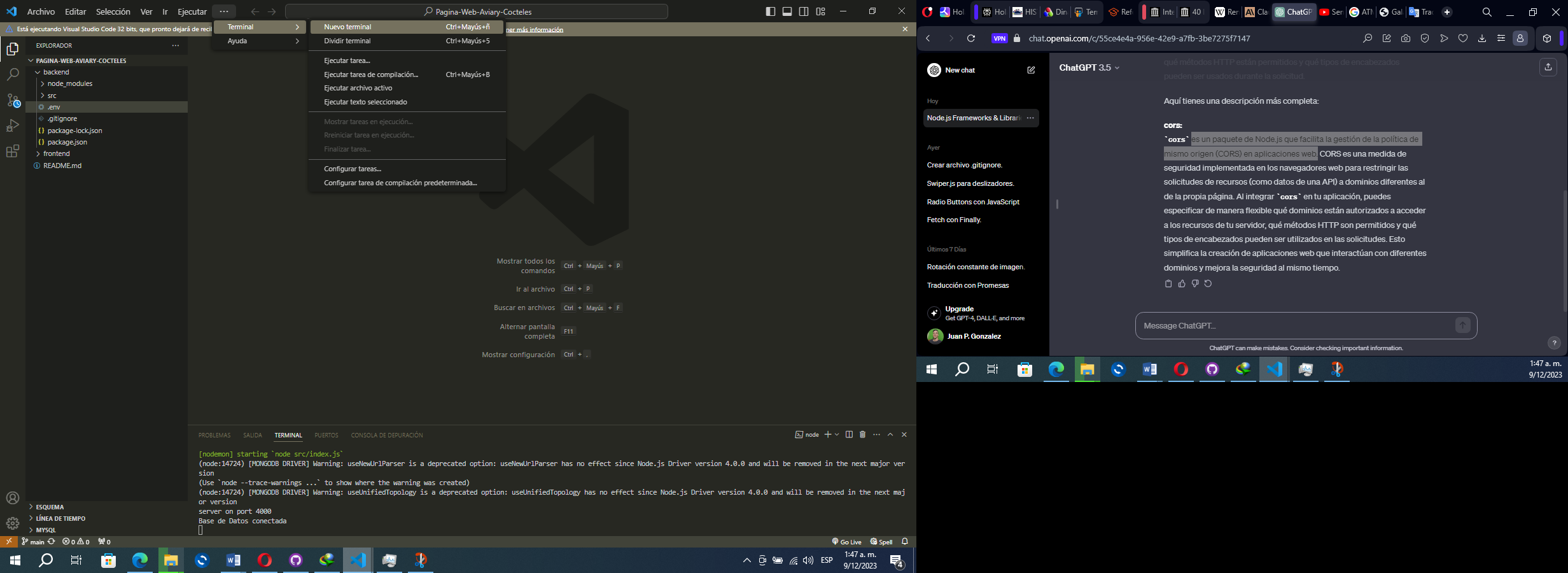
eso simplemente creamos un archivo llamado “. env” (se puede realizar con el Icono de

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

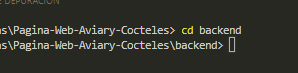
 

1. Lo siguiente es crear una terminal desde Visual Studio Code, con las teclas

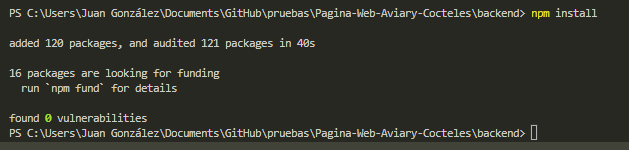
CTRL+Mayus+Ñ o desde Terminal/Nuevo Terminal



1. Cambiamos el directorio del terminal a la carpeta backend con el comando ‘cd backend’



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



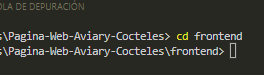
1. Ahora usamos npm run dev y si tenemos todo correcto nos saldrá lo siguiente



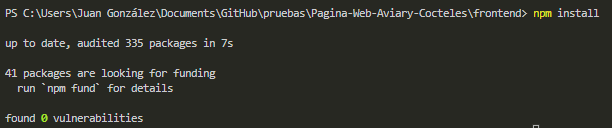
1. Ahora creamos otra terminal con el botón de símbolo de suma



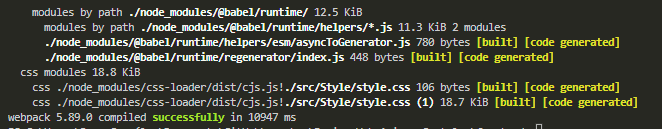
1. Cambiamos el directorio del terminal a la carpeta frontend con el comando ‘cd frontend’



1. Colocaremos npm install en la terminal para instalar todas las dependencias, en este caso de webpack.



1. Luego ejecutamos el comando npm run dev para así compilar y obtener detalles de errores (en caso de existir). Si compila de forma correcta, quedaría algo así:



1. Por ultimo para ir a la *home page* abrimos el archivo index.html que se encuentra en la carpeta ‘dist’, que a su vez está alojada en la carpeta raíz del frontend.