SPRINT 3. CREACIÓN DE PIPELINES Y PRUEBAS UNITARIAS

Marcela Peñaranda Ortega – C.C: 1091670131

Brandom Mosquera Cadena– C.C: 1013674995

Sergio González Betin -C.C: 92529506

Miguel González Rodríguez– C.C: 1010005478

Alejandra Rosero Maya – C.C: 1233194367

MISION TIC 2022 – DESARROLLO WEB

UNIVERSIDAD DE ANTIOQUIA

26 DE SEPTIEMBRE DE 2021

**Objetivo Grupal**

* Crear las pipelines correspondientes a nuestro proyecto empleando aws e implementando las pruebas unitarias para los elementos de la base de datos.

**Proceso para alcanzar los objetivos y requerimientos del sprint.**

Para poder llevar a cabo la ejecución del Sprint 3, cada uno de los miembros del equipo escogieron una tarea específica con el propósito de cumplir la entrega, pues mientras algunos desarrollaron las pruebas unitarias, otros se encargaron de la gestión completa de los pipelines en el servidor.

Con el propósito de garantizar la trazabilidad del proyecto se implementó el uso de GitHub y Trello, pues en Github se puede registrar los cambios del proyecto (como agregación del proyecto y pruebas), mientras que en Trello se especifican las tareas de cada miembro:

1. Dentro de Trello, se gestionaron las tareas del sprint 3 para cada miembro, en donde se incluían el desarrollo de las pruebas unitarias usando mocha, chai y mongoose.
2. Para la gestión de la actualización del proyecto, cada uno de los miembros se encargó de subir los archivos desarrollados con el propósito de hacer un pull request una vez estén completados.
3. Para crear las pipelines se hizo seguimiento de la tutoría dada en clase a fin de garantizar un buen desarrollo de las mismas.

De nuevo, para actualizar el repositorio de Git se hizo uso de las ramas iniciales y se modificaron las carpetas de fuentes y bases de datos para indicar las instrucciones que se utilizaron para crear las pipelines y pruebas unitarias.

**Rutas**

**GitHub:**   <https://github.com/AlejandraRo/ciclo4-proyectofinal>

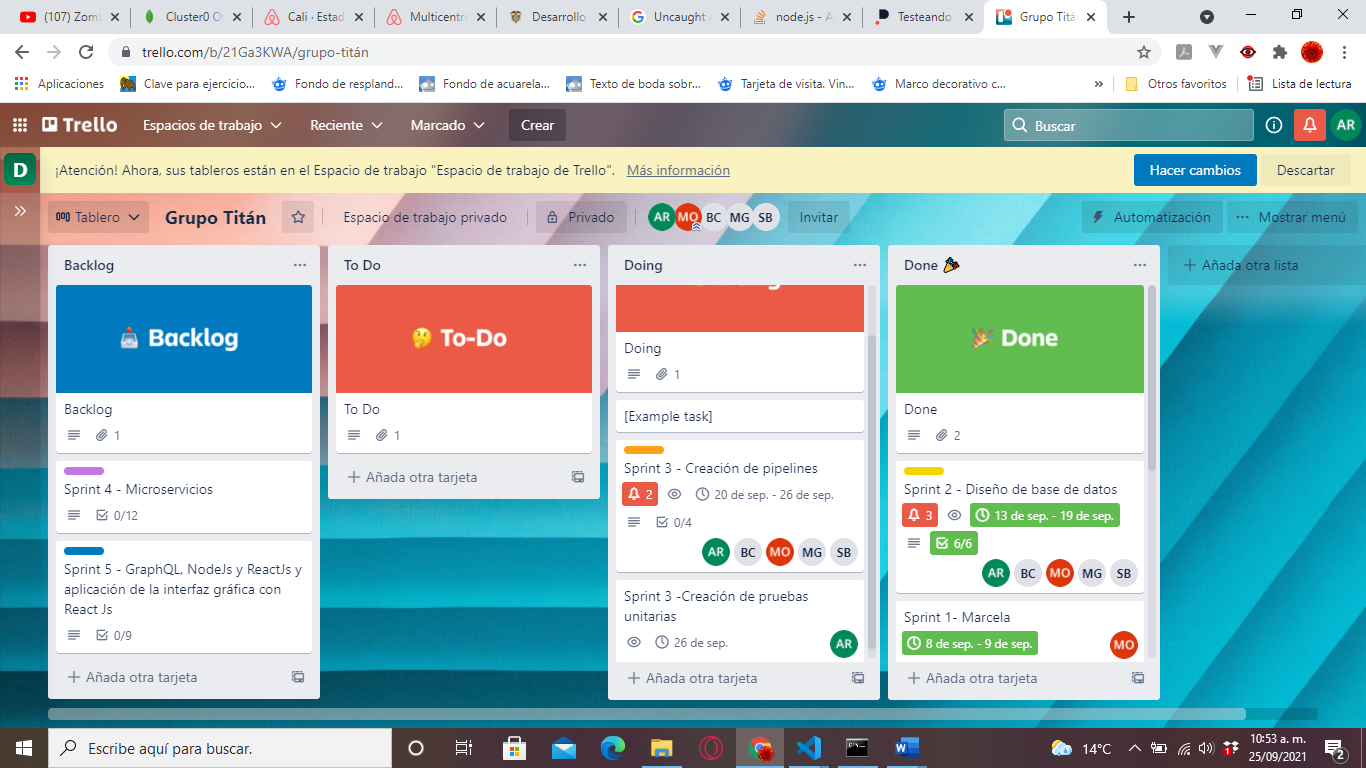
(Dentro de la rama Development)

**Trello:** <https://trello.com/b/21Ga3KWA/grupo-tit%C3%A1n>

**MongoDB:** <https://cloud.mongodb.com/v2/613e16830a992558f04f17cc#access>

**AWS:** <https://dev.azure.com/ssgonzalezb/manage_proyects/_build>

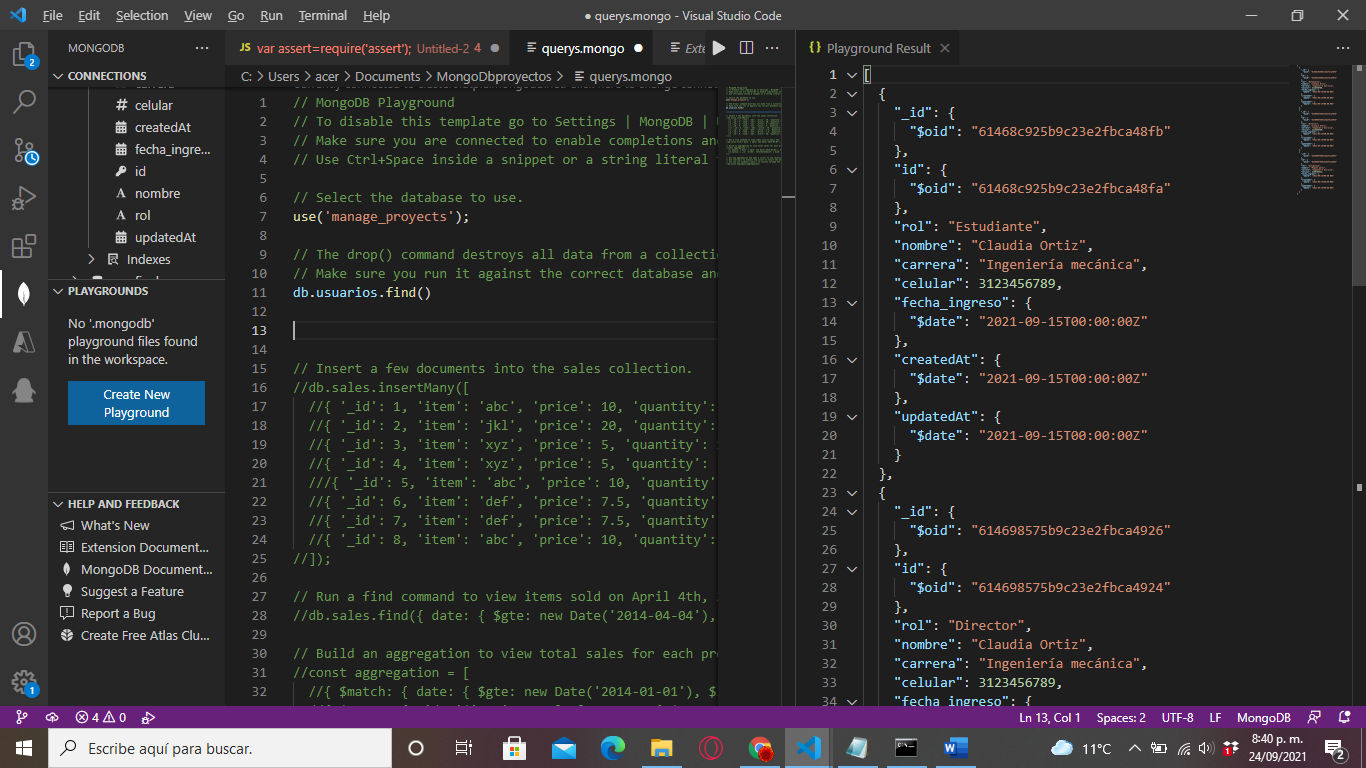
**EVIDENCIAS**



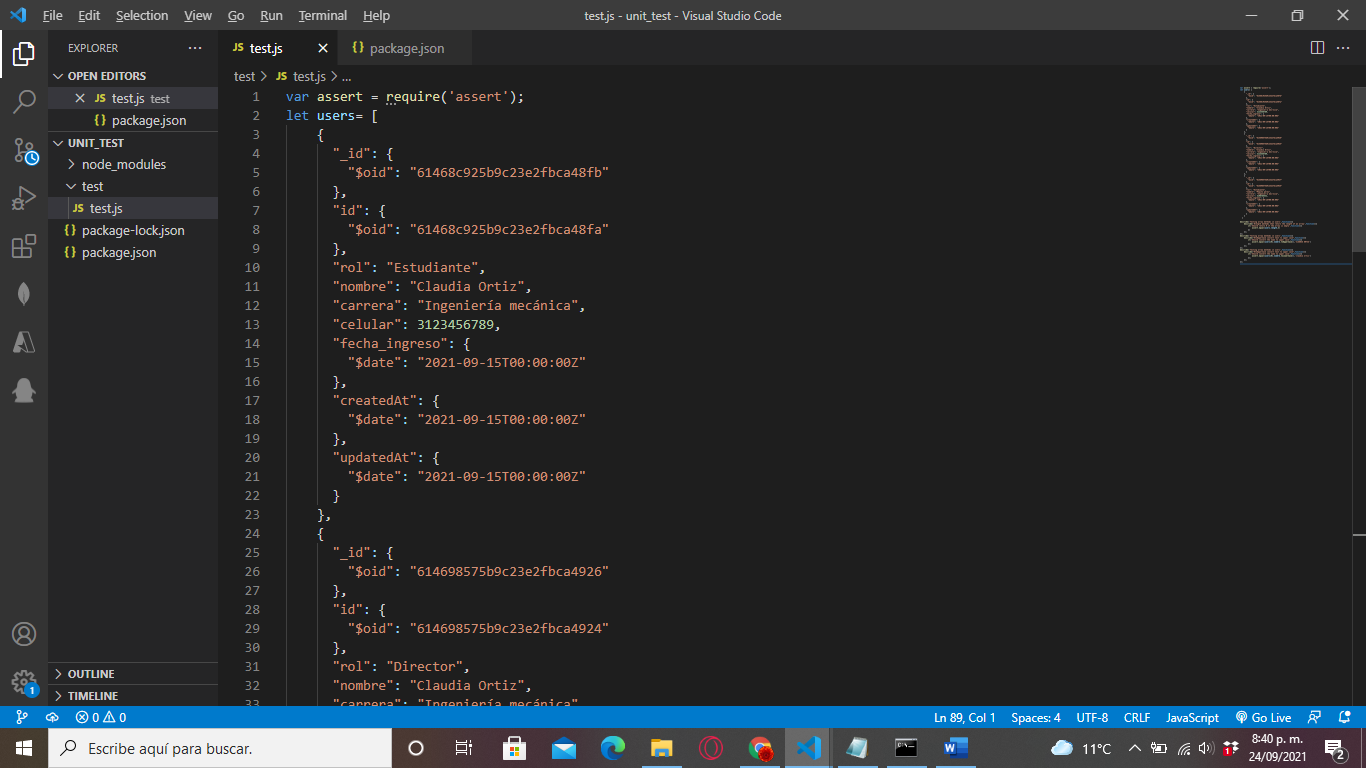
**DESARROLLO HISTORIAS DE USUARIO**

HU\_013: Configuración de Jobs en las pruebas unitarias.

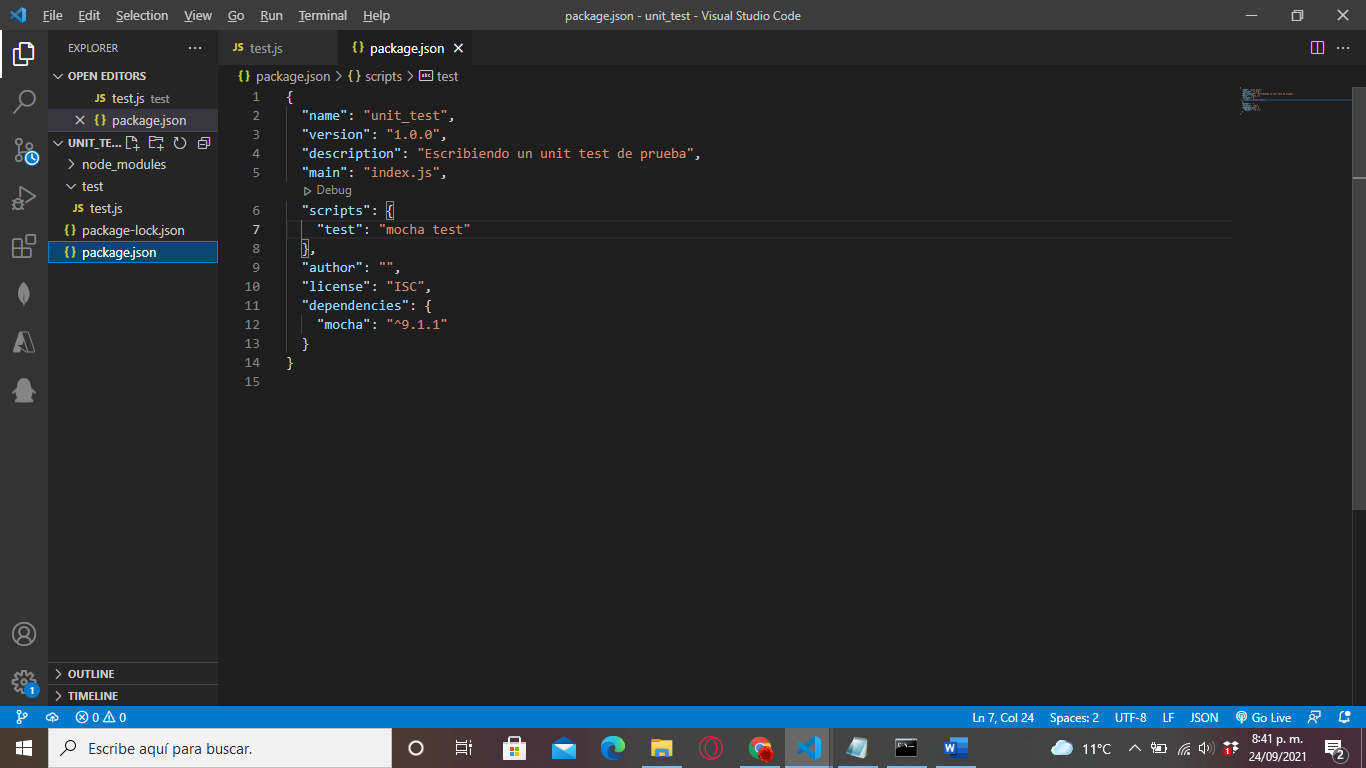
A través de una extensión de MongoDB en Visual Studio Code se realizó la conexión a la base de datos del proyecto a fin de obtener los datos de las colecciones para el manejo de pruebas



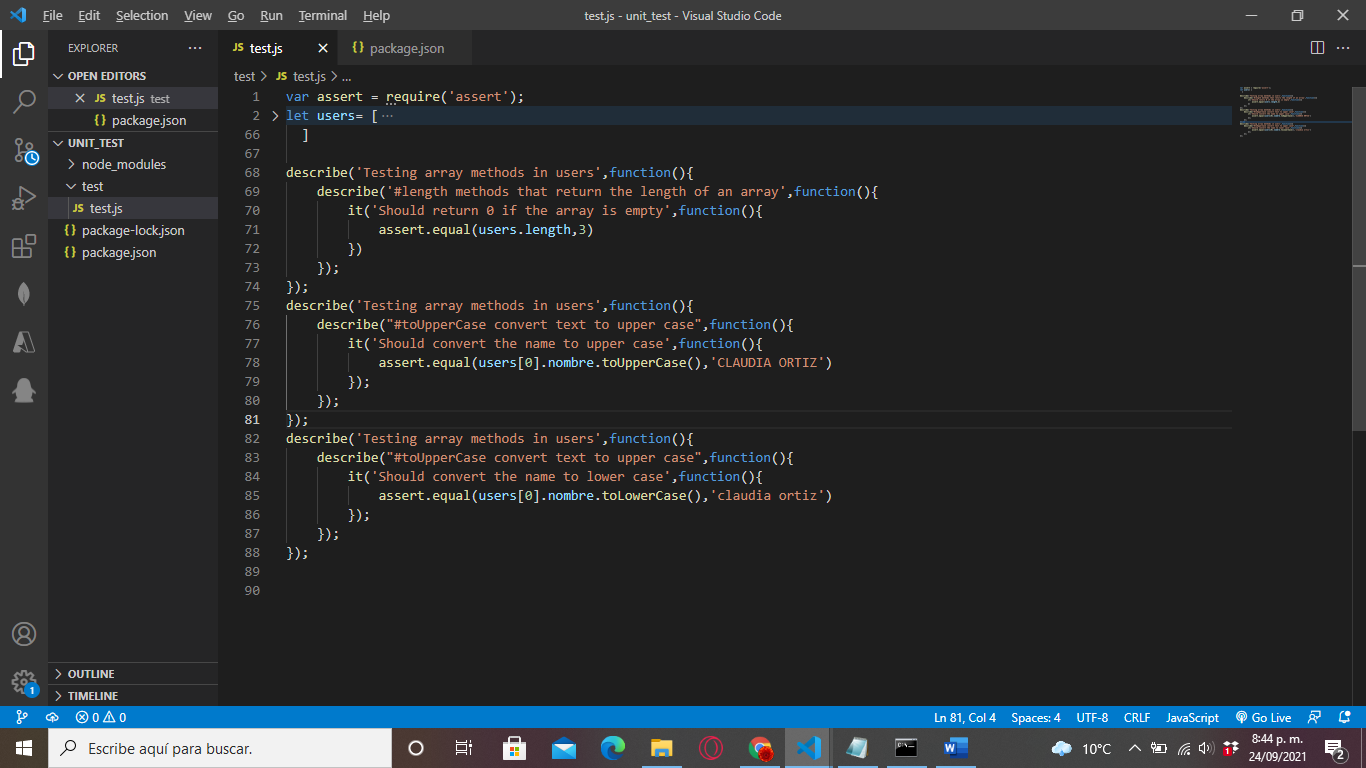
Con este resultado en un nuevo archivo de test se creó la variable users



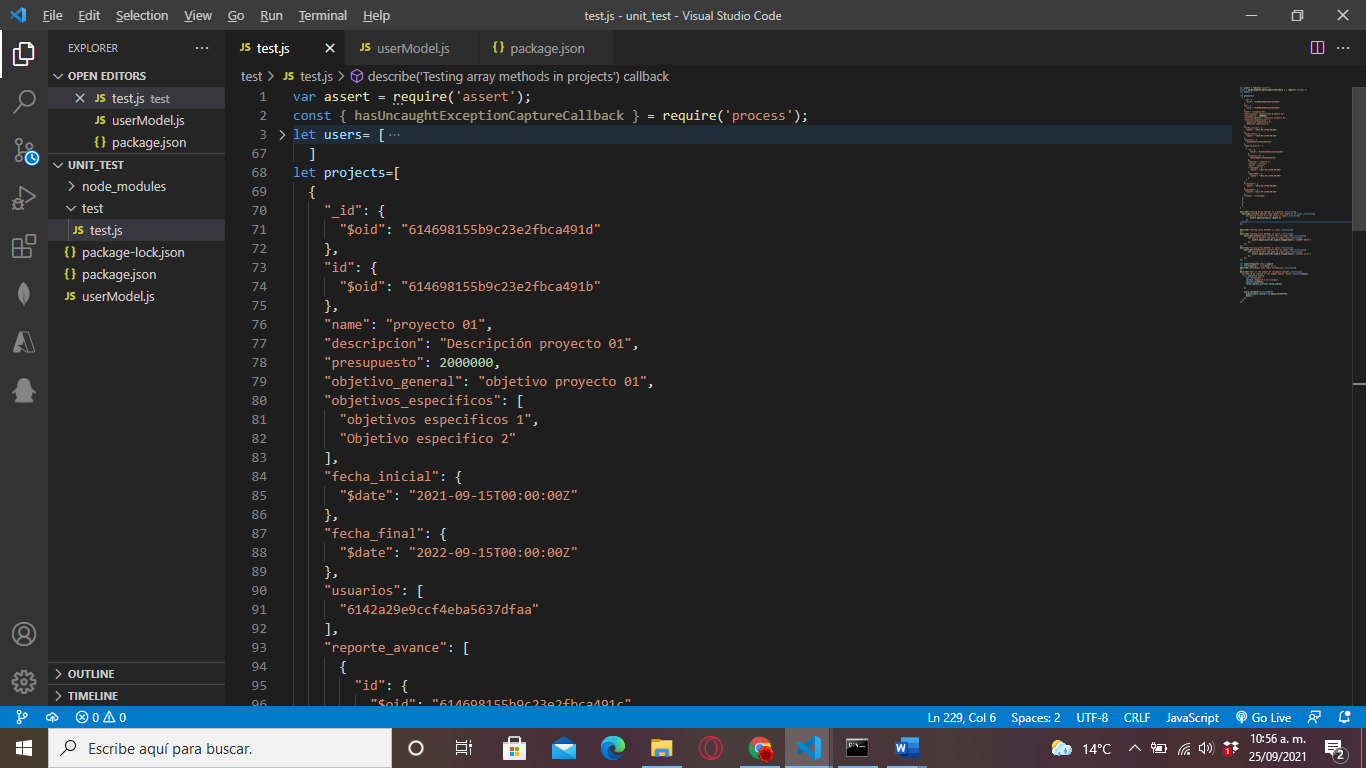
Además, después de instalar mocha se hizo necesario modificar el comando del test en el package.json para poder ejecutar las pruebas porque en Windows no se reconoce el comando mocha en cmd



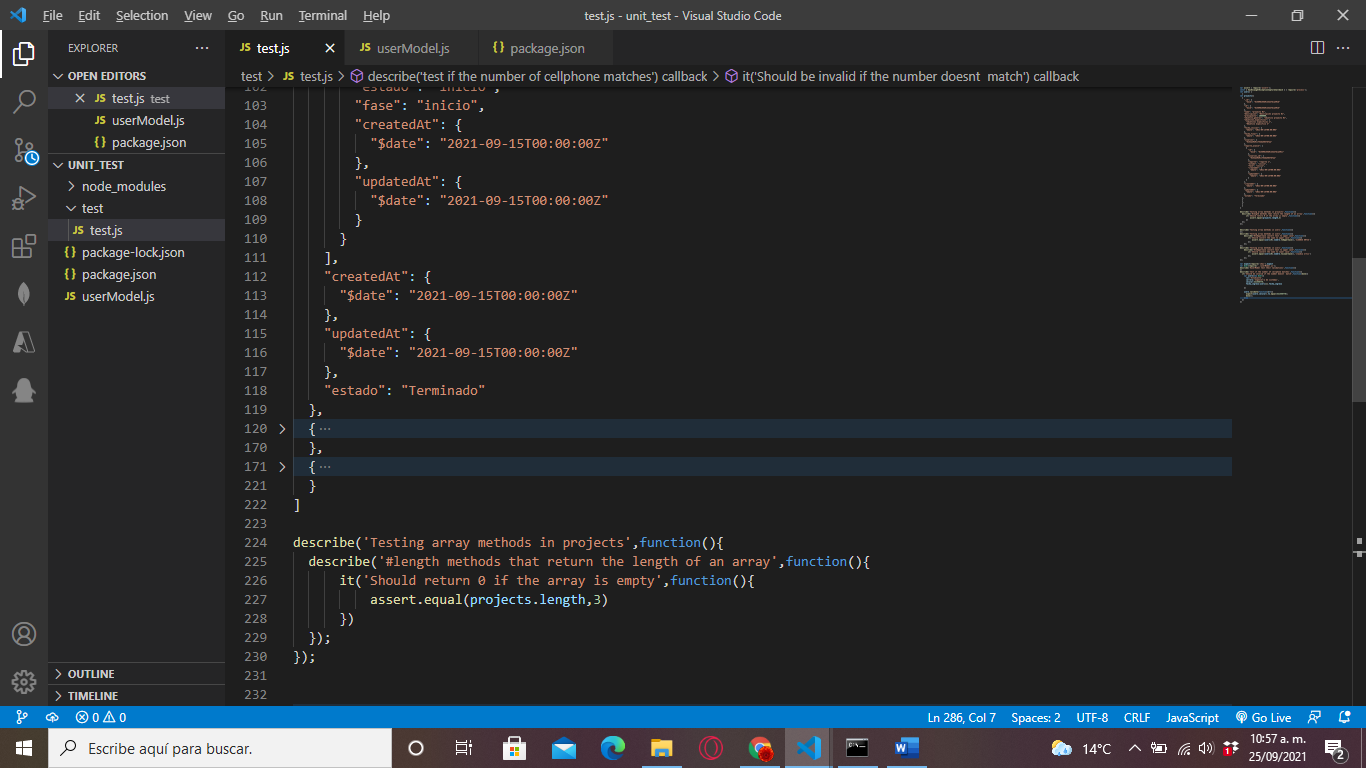
Se construyeron tres métodos para validar que efectivamente los datos concuerdan con lo declarado inicialmente, entonces se realizó la prueba del largo del array que corresponde a 3 usuarios, un método para UpperCase aplicado sobre el nombre de un usuario y un método LowerCase sobre este mismo dato.



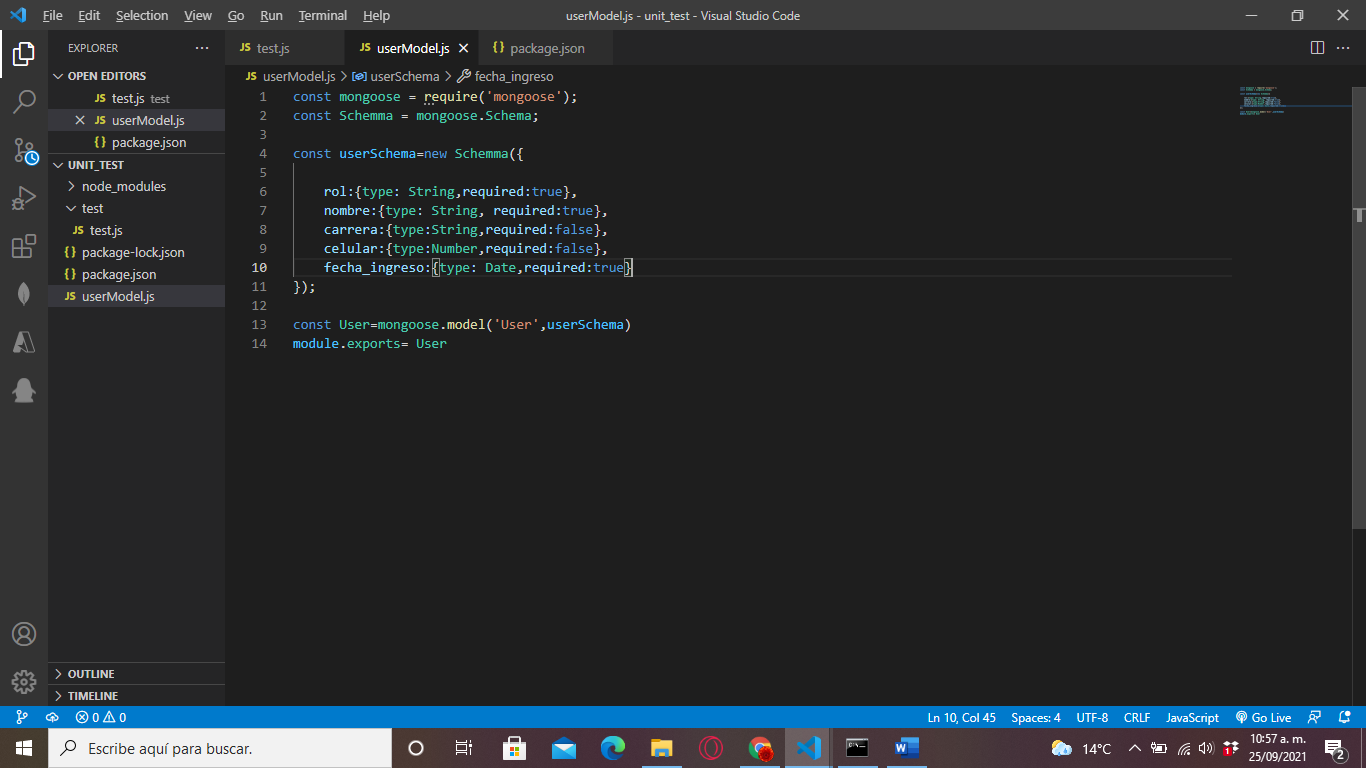
A través de otra consulta se hizo la creación del array proyectos



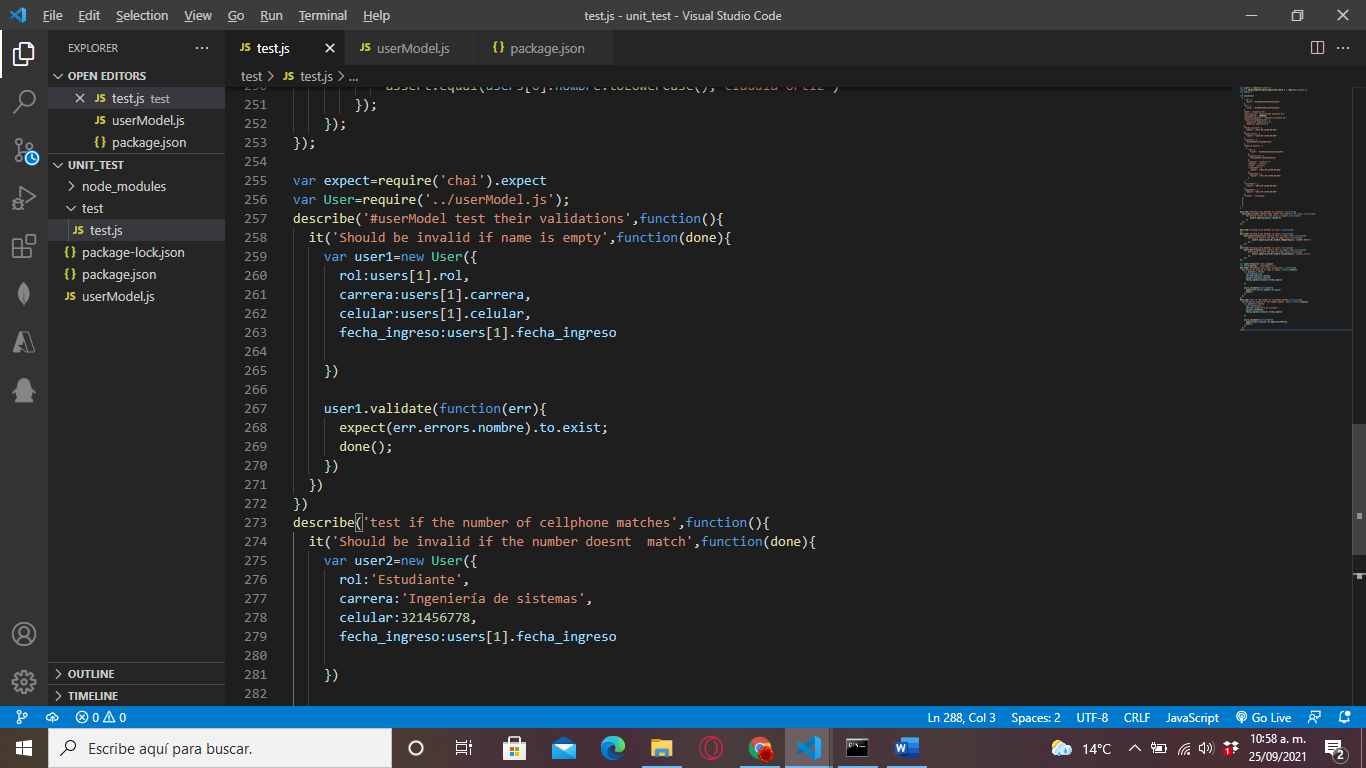
Se hizo la validación del largo del array



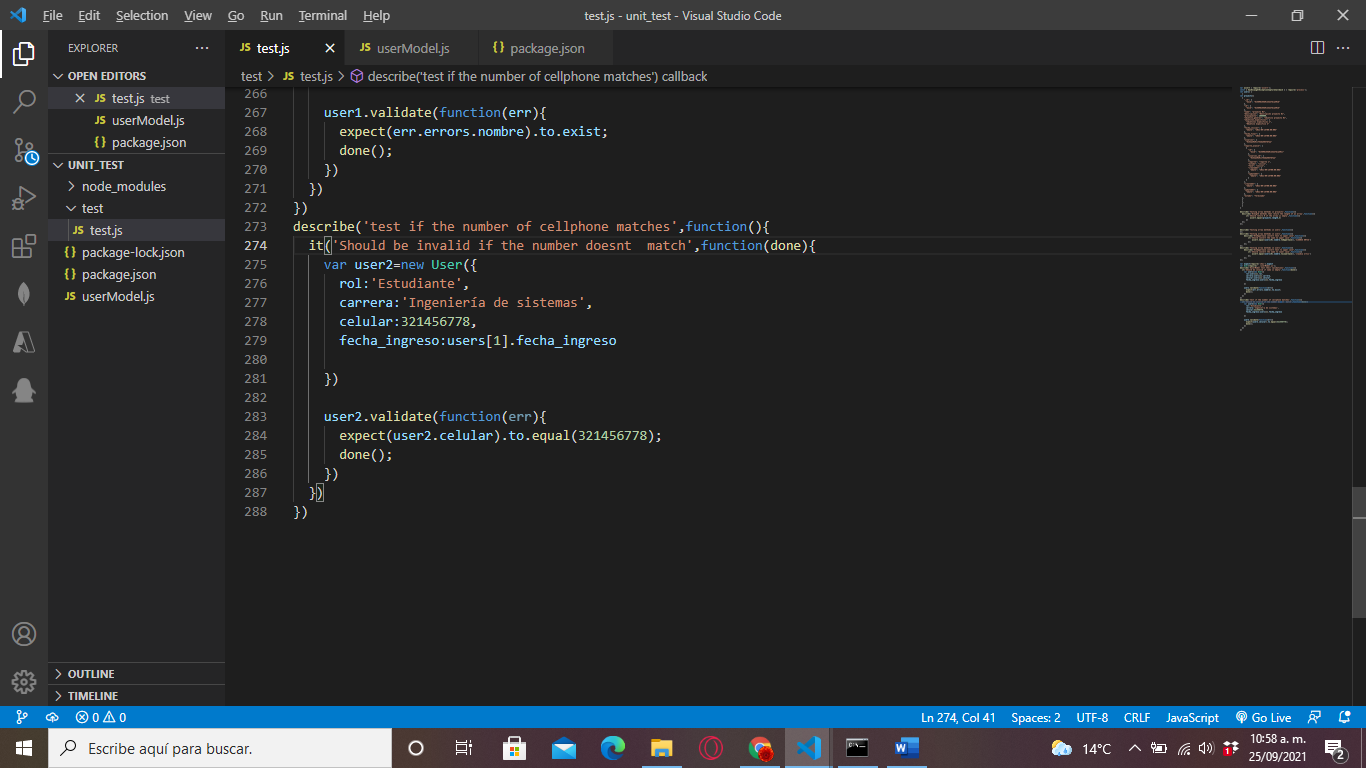
Para la validación de modelos, se creó el modelo de usuario que corresponde al especificado en la base de datos



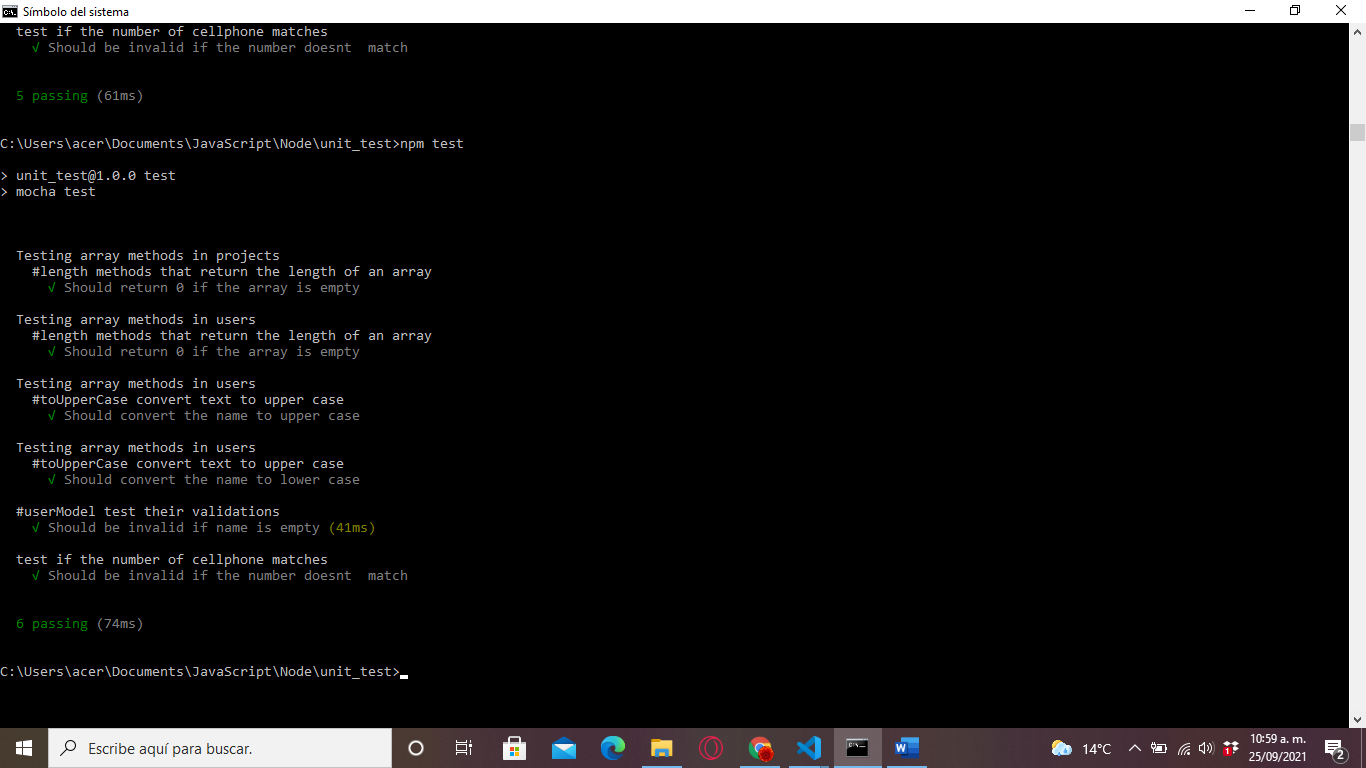
Se validó el nombre del usuario



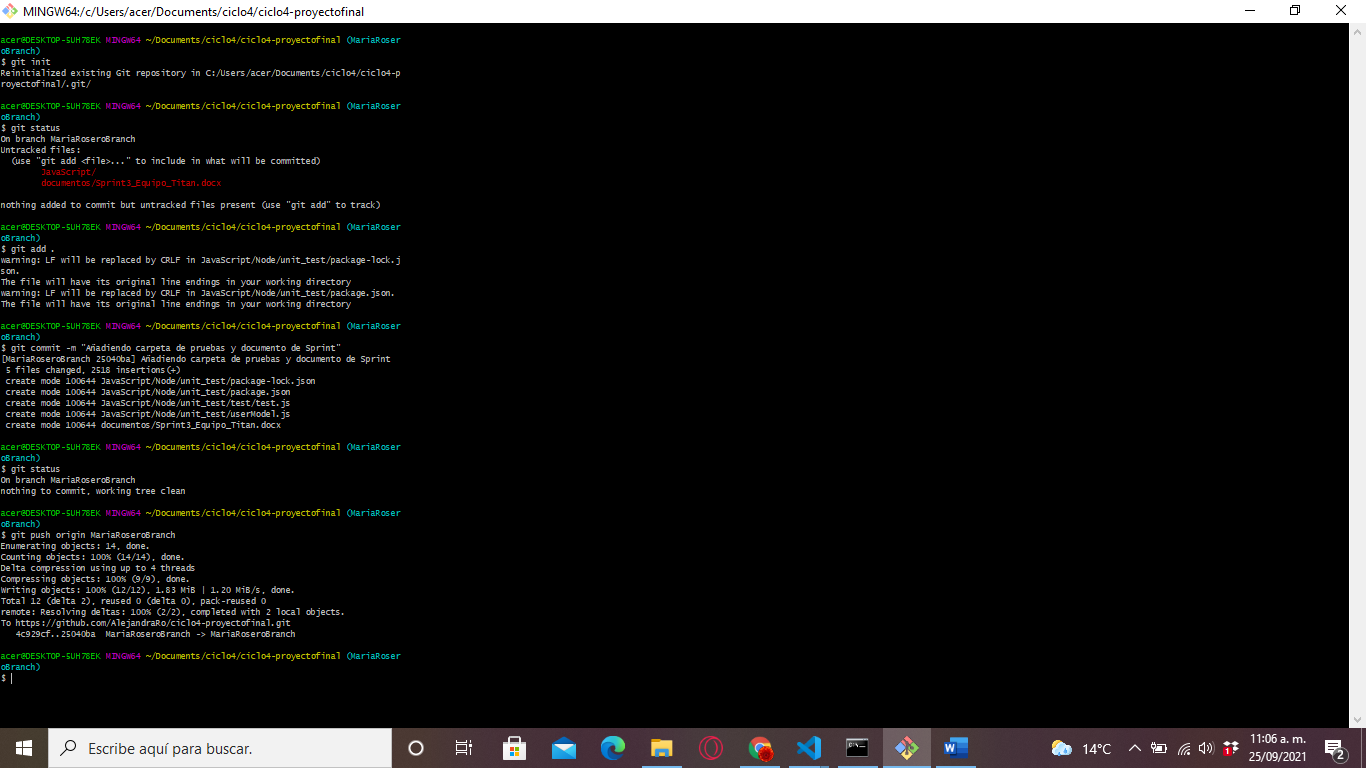
Y que el número de celular corresponda al almacenado en la base de datos



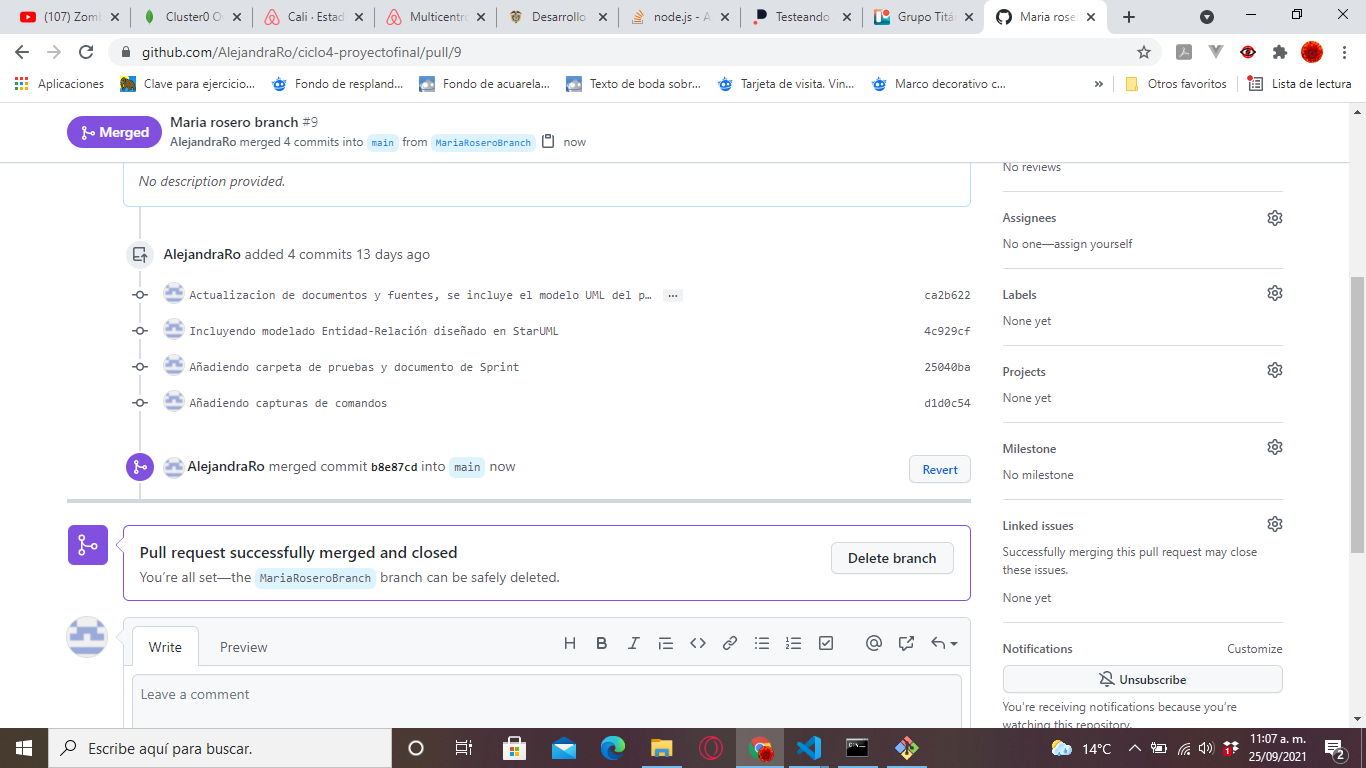
Al correr el test se obtuvo el siguiente resultado



Subiendo cambios a github



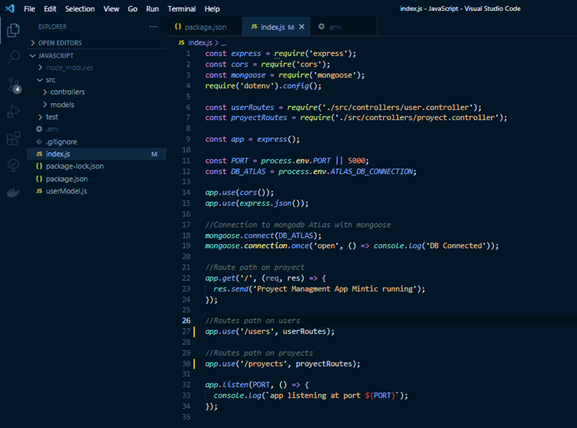
Haciendo pull request



**CREACIÓN DEL PROYECTO BACKEND**

Dentro del package.json instalamos expressjs como framework de node.js que nos permite gestionar el enrutamiento, manejar peticiones con diferentes verbos HTTP y conexión con la base de datos (mongodb Atlas).

1. Cargamos el paquete de express y definimos una aplicación de express mediante el objeto app que se usará para definir rutas y demás características de express.
2. Declaramos el puerto donde la aplicación estará escuchando las peticiones del servidor.
3. Definimos la ruta raíz del proyecto usando el método app.get(), y las rutas generales para usuarios y proyectos e importamos los controlladores que poseen las rutas para realizar diferentes solicitudes HTTP.
4. Express usa el método express.json() para reconocer la solicitud como objeto json.



2

3

4

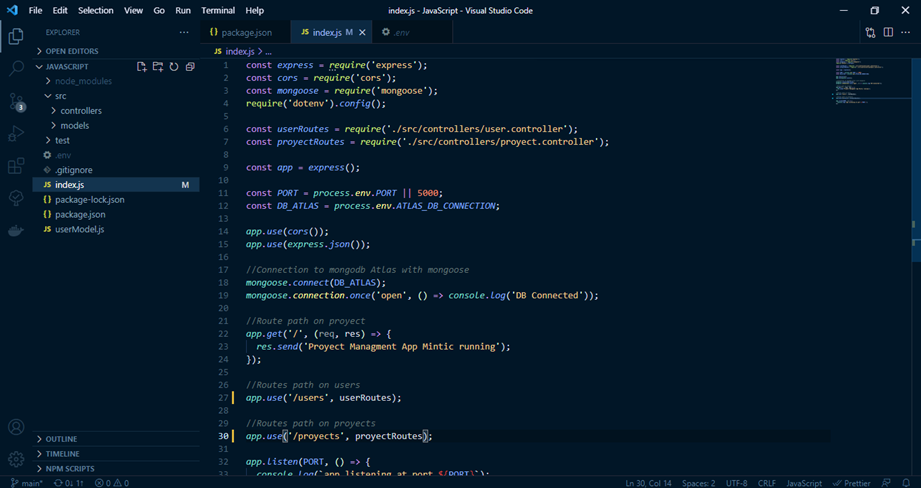
2

1

3

1

1. Instalamos dotenv para manejar variables de entorno, específicamente para guardar el enlace de conexión del cluster de mongodb Atlas y el puerto.
2. Instalamos cors (Cross-Origin Resource Sharing) para permite o restringe las solicitudes a la app desde otro dominio
3. Instalamos mongoose para realizar la conexión con la base de datos, para definir el esquema de las colecciones y escribir operaciones CRUD.



5

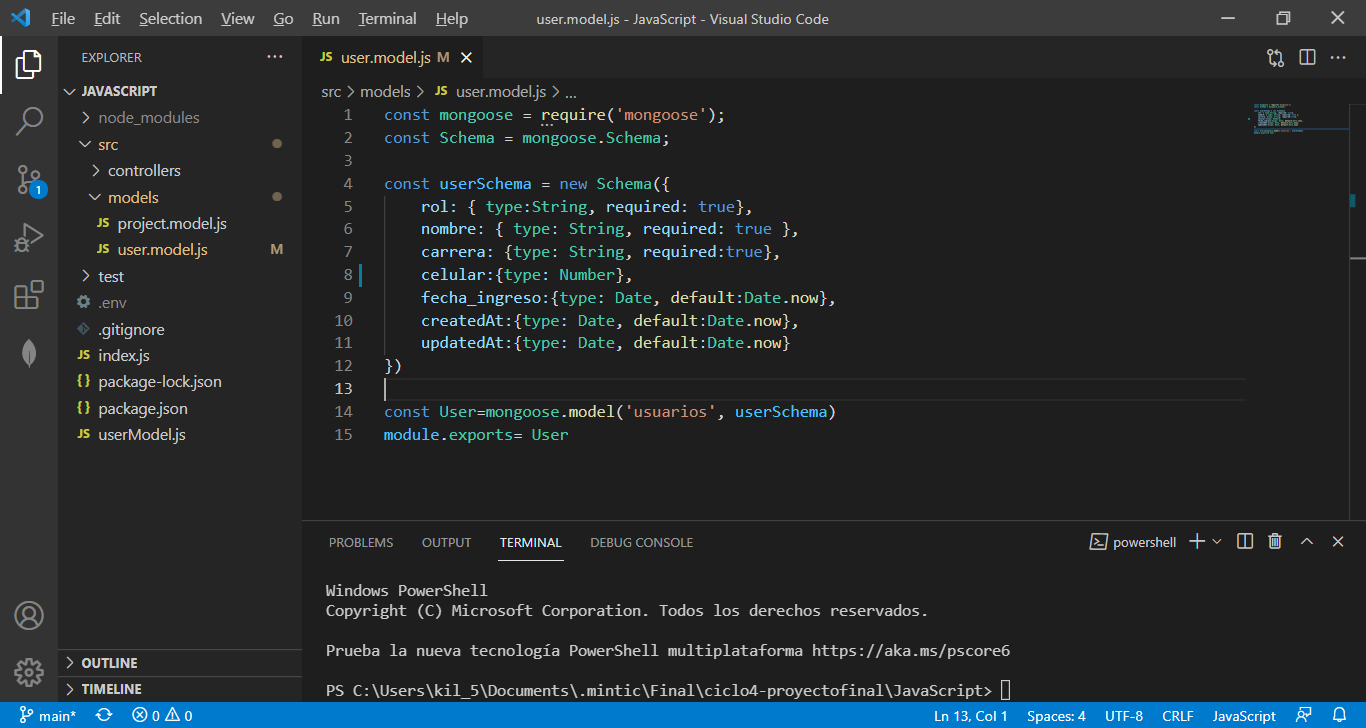
6

7

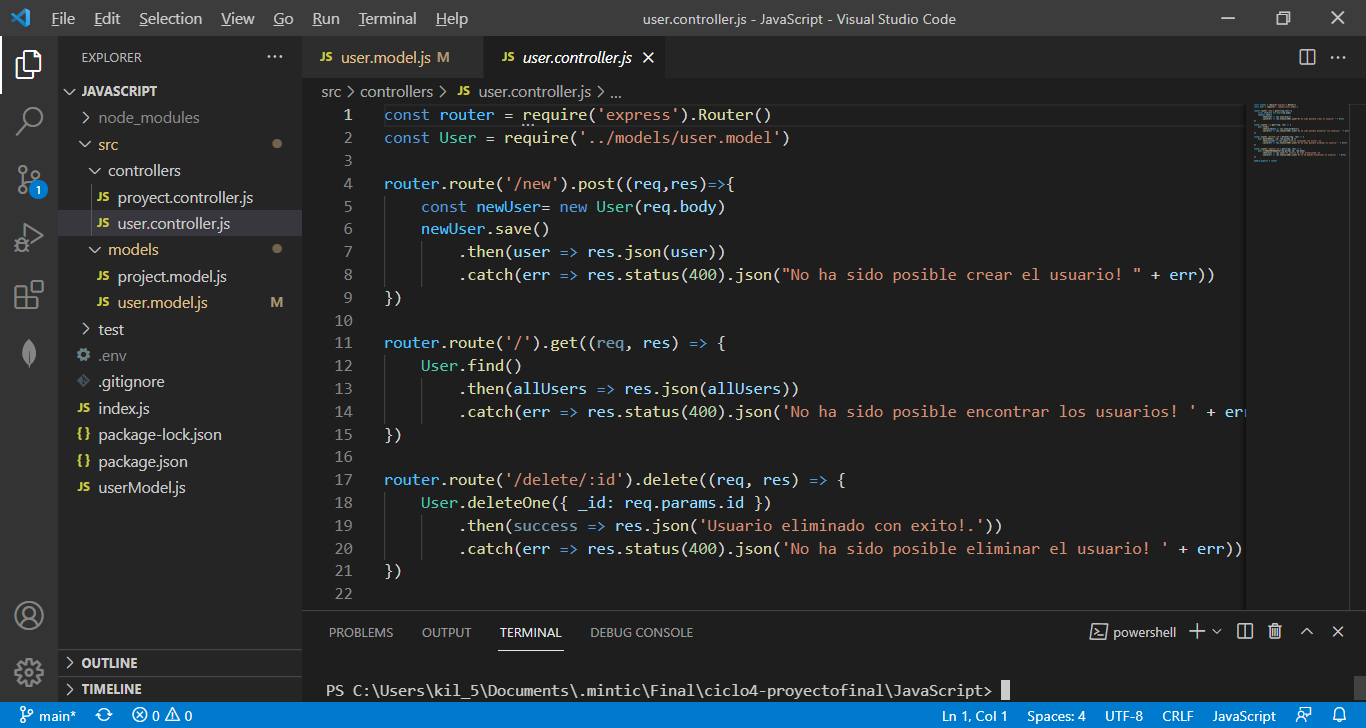


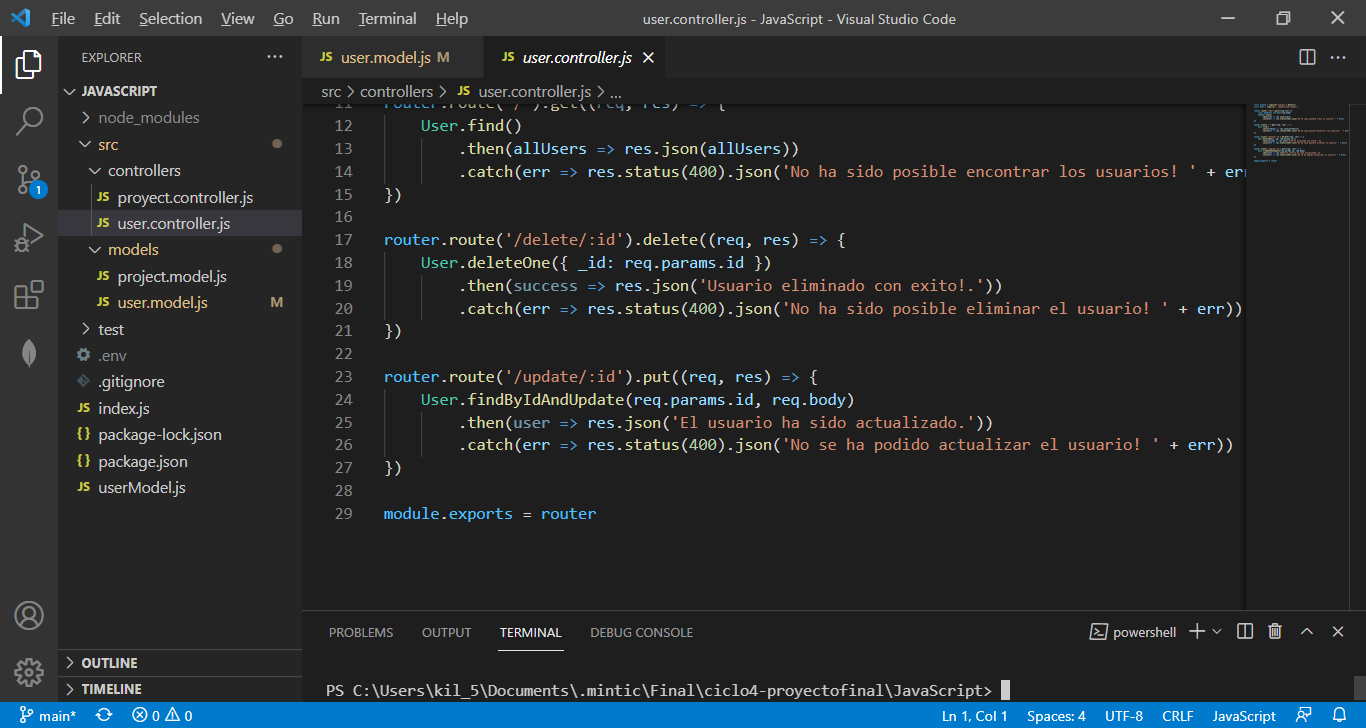
5

1. Una vez obtenida la conexión, se procede a realizar el modelo de usuarios de la colección existente en la base de datos para hacer el CRUD de este

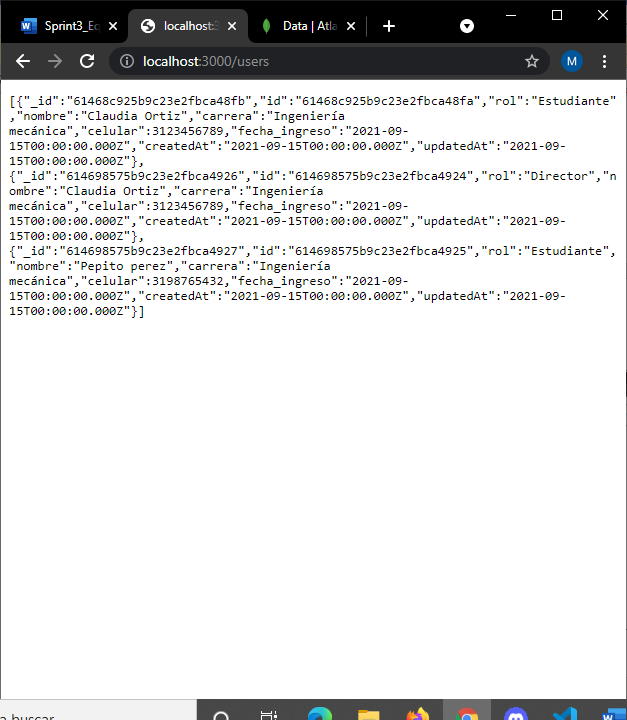


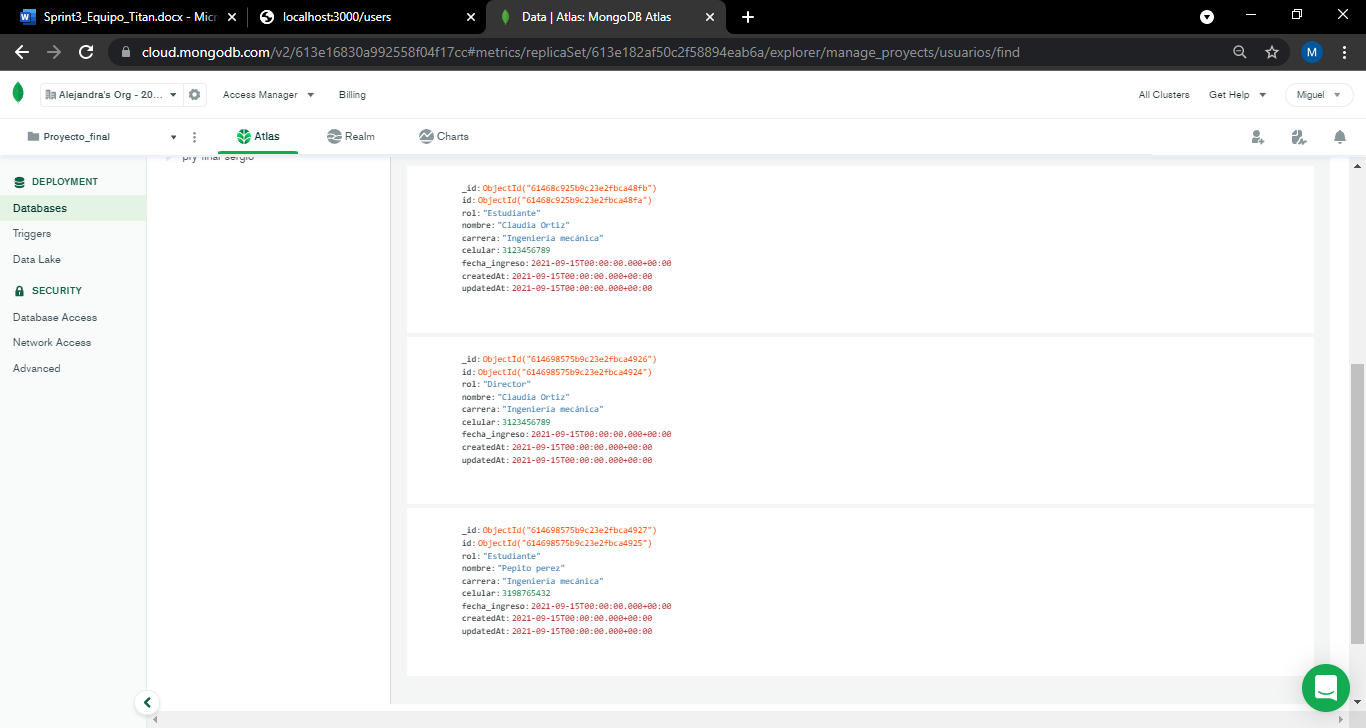
1. Utilizando un controlador, tenemos el CRUD básico para los usuarios





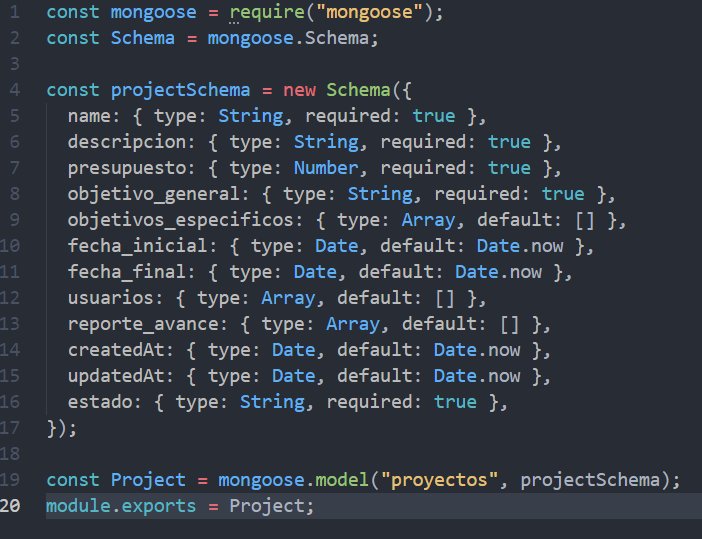
1. Probando que la conexión sea efectiva y traiga los datos de la base de datos usando la ruta designada a obtener usuarios





Como podemos observar, los datos traídos son los mismos datos existentes en la base de datos

1. Posteriormente se realizar el modelo de proyectos de la colección existente en la base de datos para hacer el CRUD de este



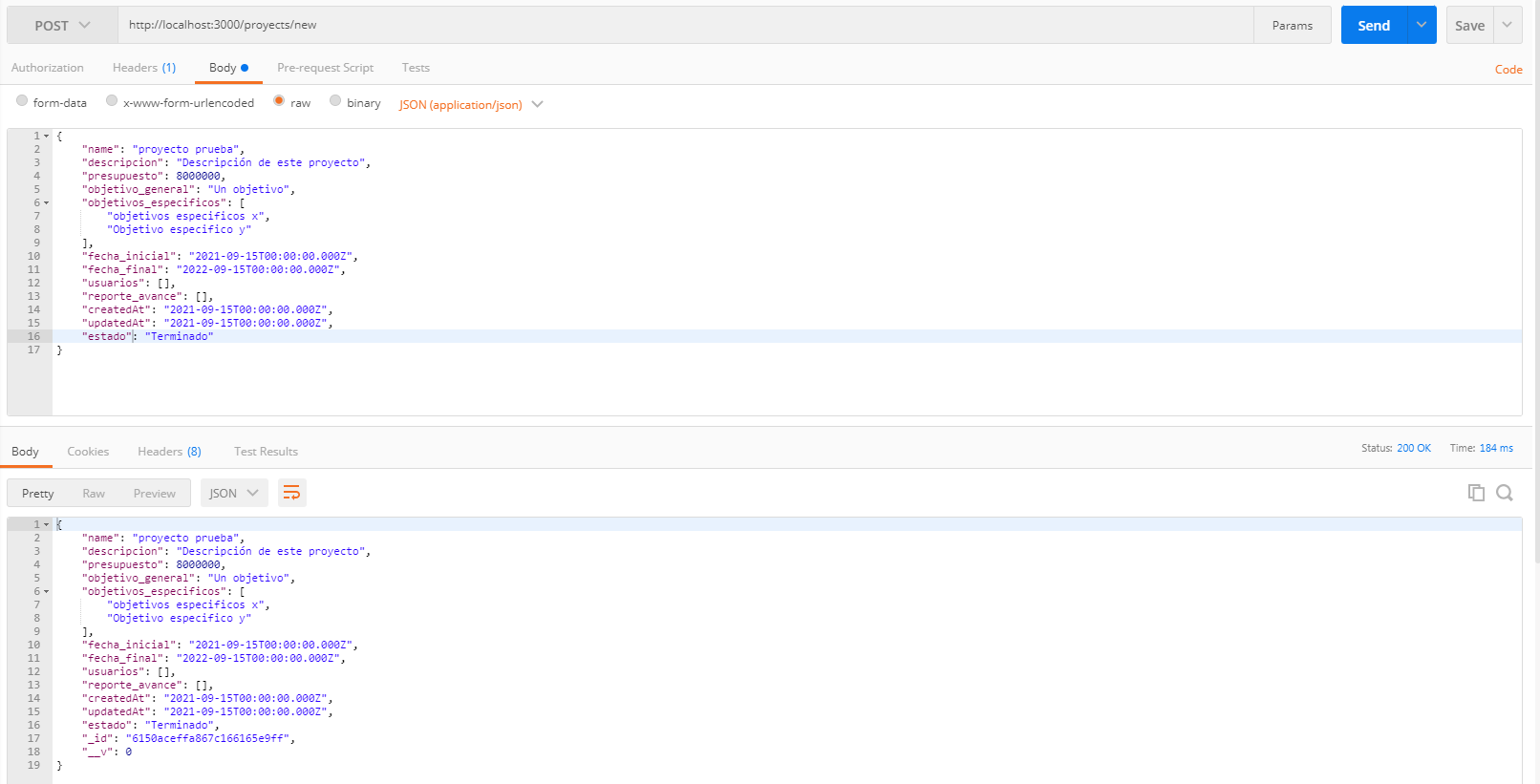
1. Creación del CRUD de proyectos



1. Probando la petición GET del CRUD de proyectos

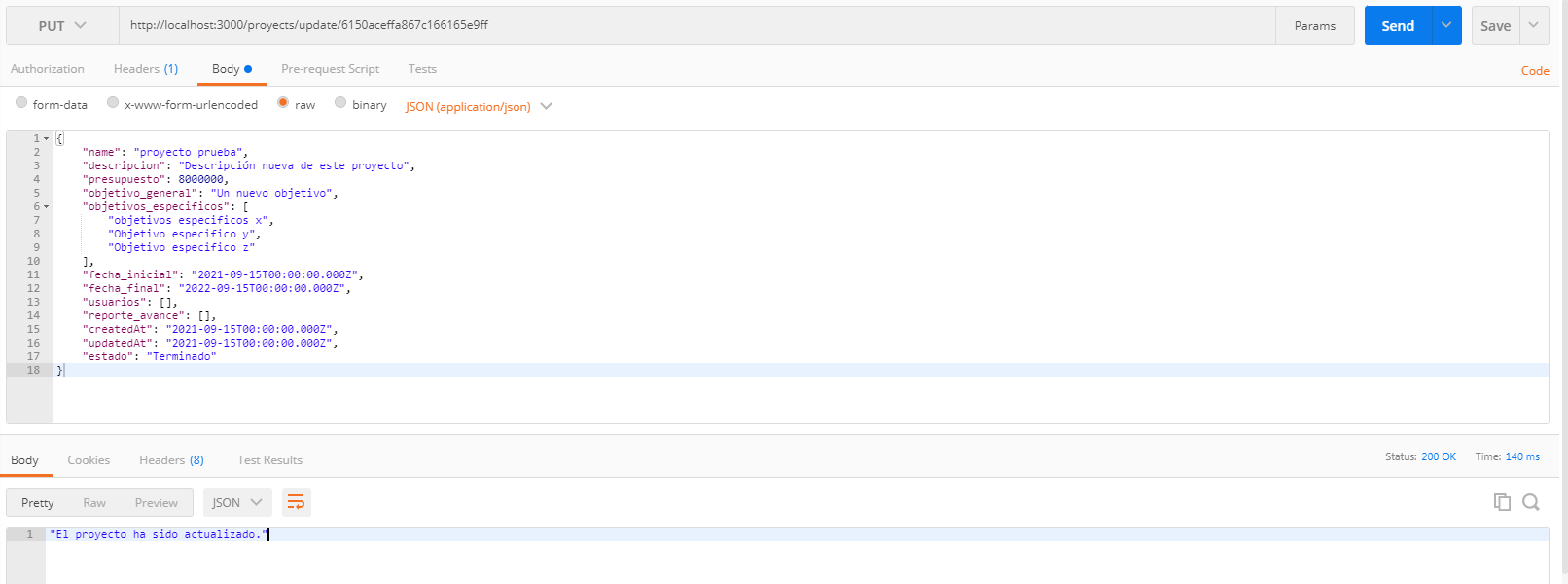


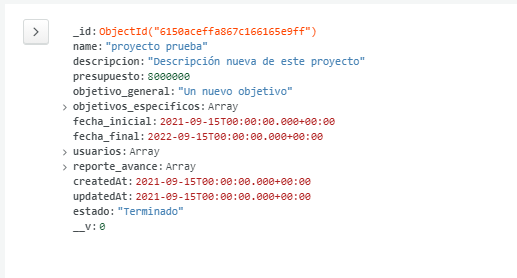
1. Probando la petición POST del CRUD de proyectos



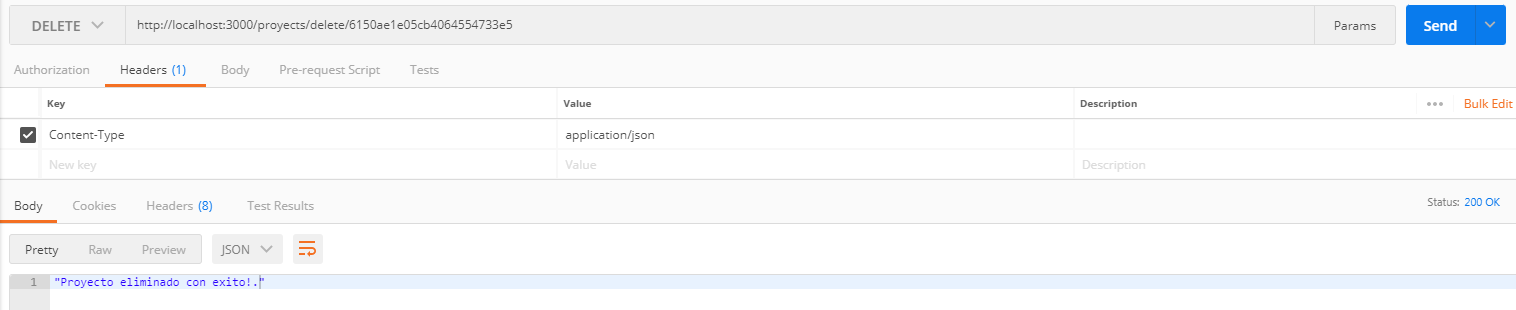


1. Probando la petición PUT del CRUD de proyectos





1. Probando la petición DELETE del CRUD de proyectos



**HU\_014 y HU\_015 Contrucción de pipelines en Azure Devops**

