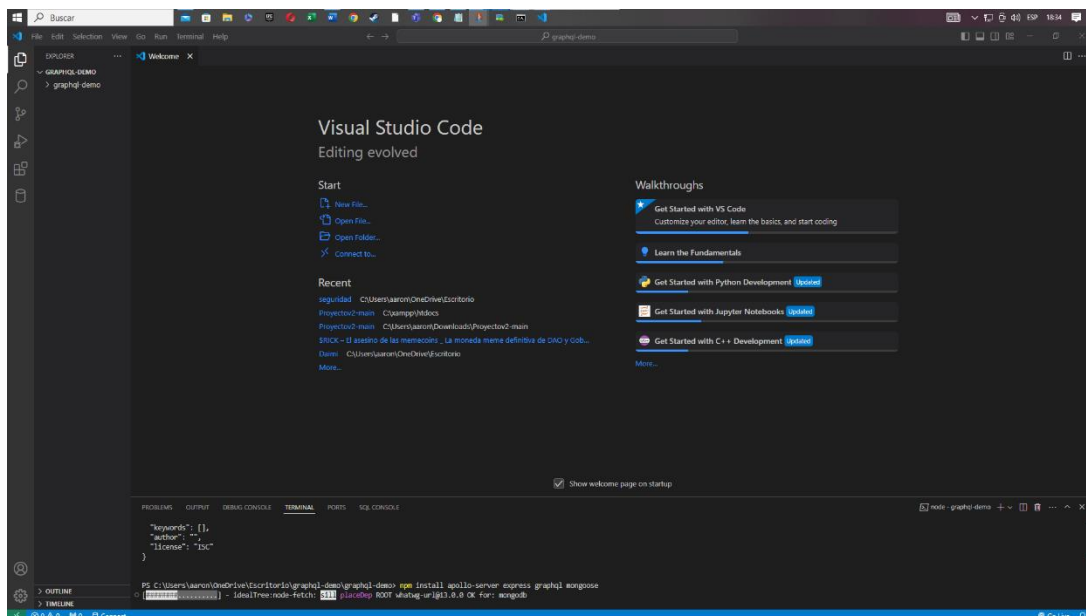
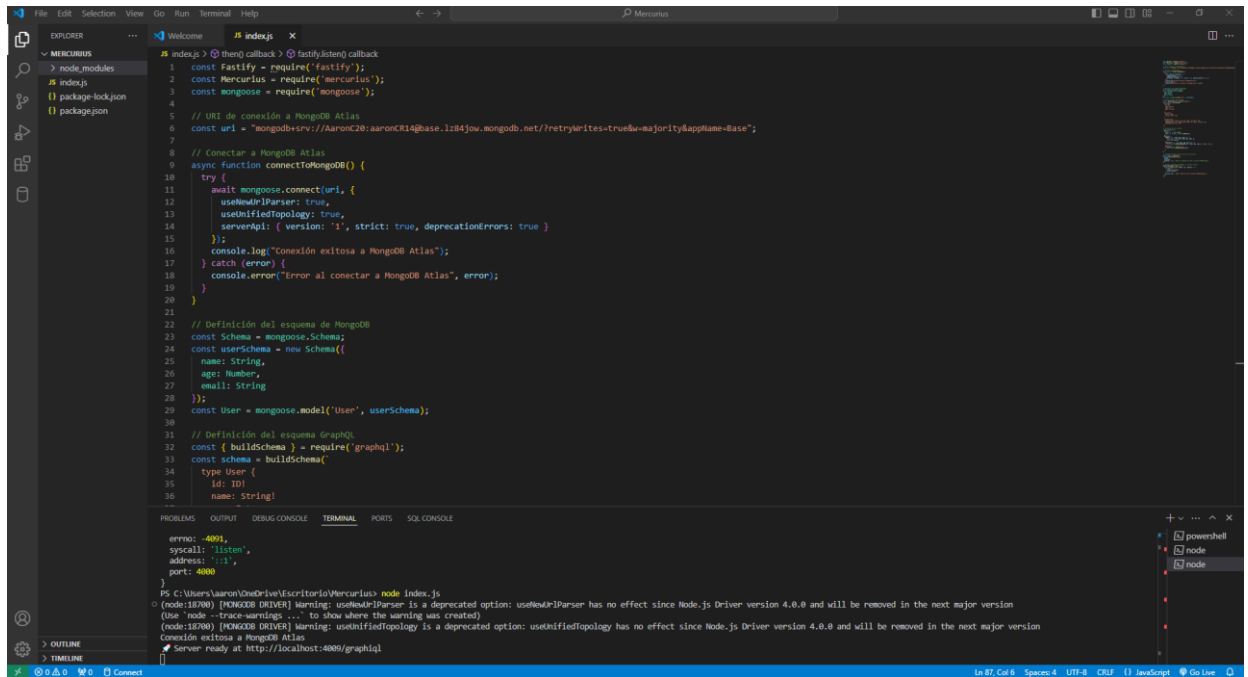


Terminamos de configurarlo y le damos a crear nuevo usuario



Una vez creado el usuario y base de datos Abrimos Visual Studio Code y seleccionamos la carpeta en la cual vamos a realizarlo

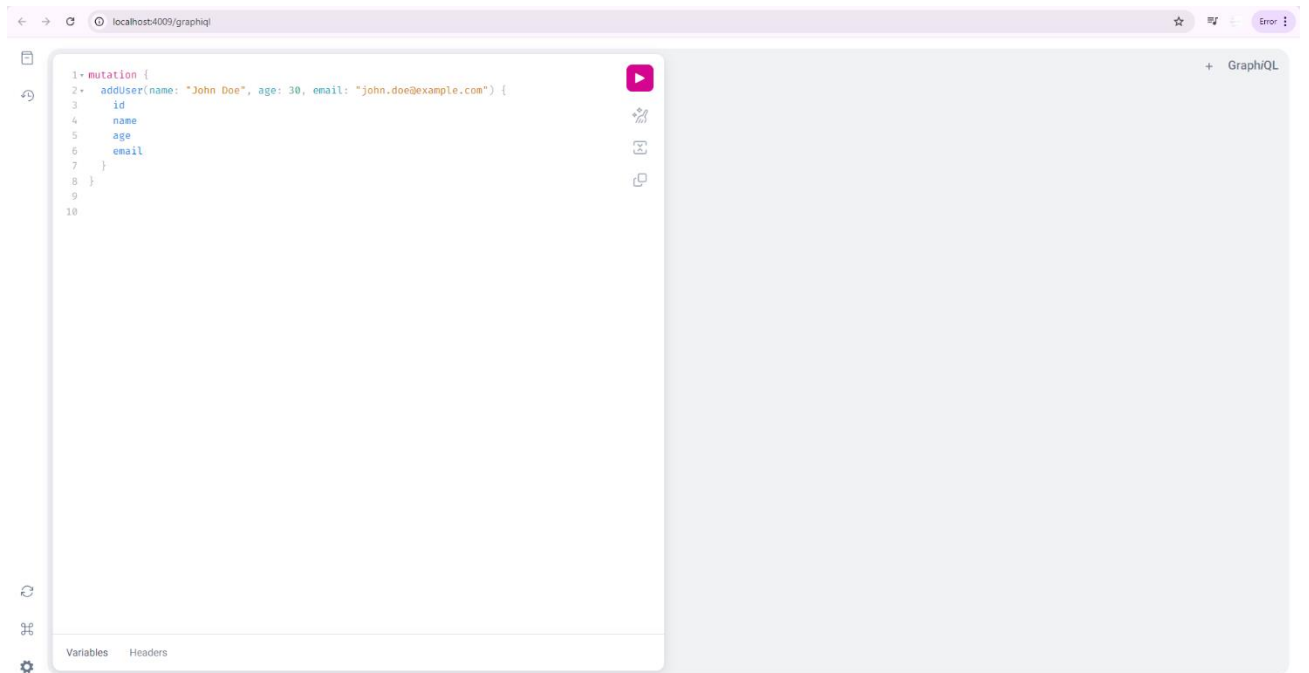


En mi caso utilicé Mercurius, entonces procedí a instalarlo en la terminal con le siguiente comando: `npm install fastify mercurius mongoose`

Creamos la carpeta index.js la cual lleva el código en el cual se realiza el proceso necesario.

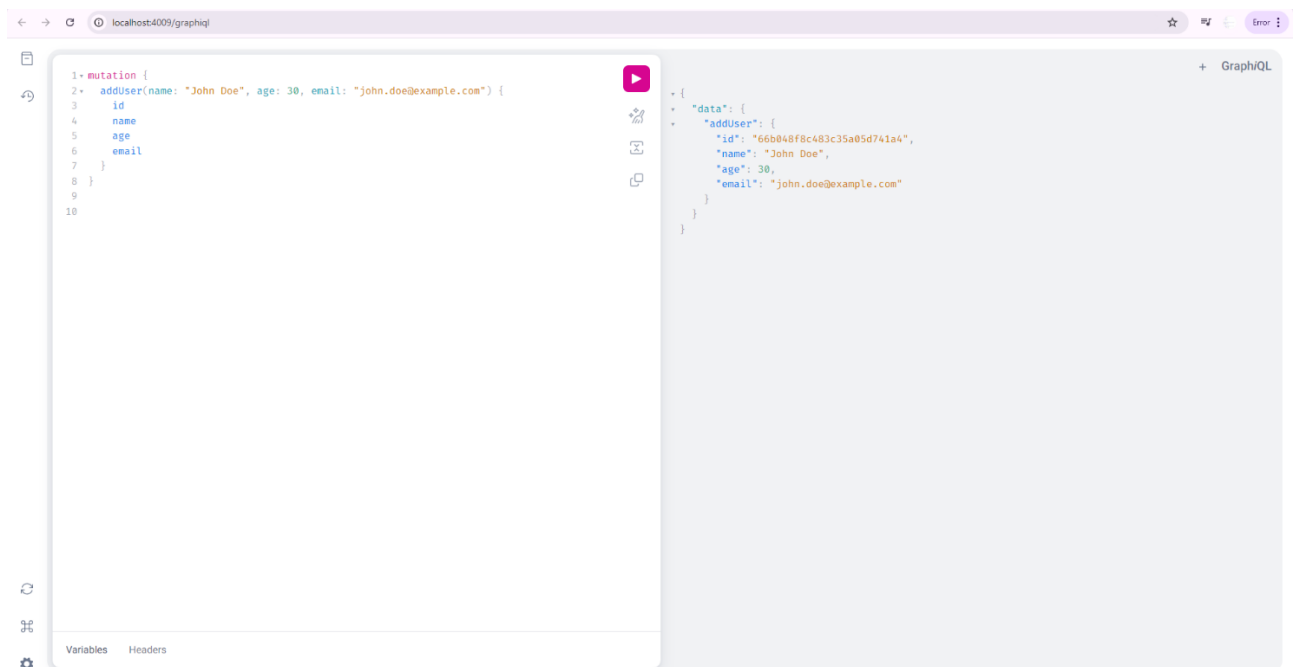
```
PS C:\Users\aaaron\OneDrive\Escritorio\Mercurius> node index.js
(node:18780) [MONGODB DRIVER] Warning: useNewUrlParser is a deprecated option: useNewUrlParser has no effect since Node.js Driver version 4.0.0 and will be removed in the next major version
(Use 'node --trace-warnings ...' to show where the warning was created)
(node:18780) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated option: useUnifiedTopology has no effect since Node.js Driver version 4.0.0 and will be removed in the next major version
Conexión exitosa a MongoDB Atlas
Server ready at http://localhost:4000/graphql
```

Una vez ejecutado el comando `node index.js` el cual ejecuta el `index.js` que creamos anteriormente. Nos brindará la url en la cual procederemos a realizar lo siguiente

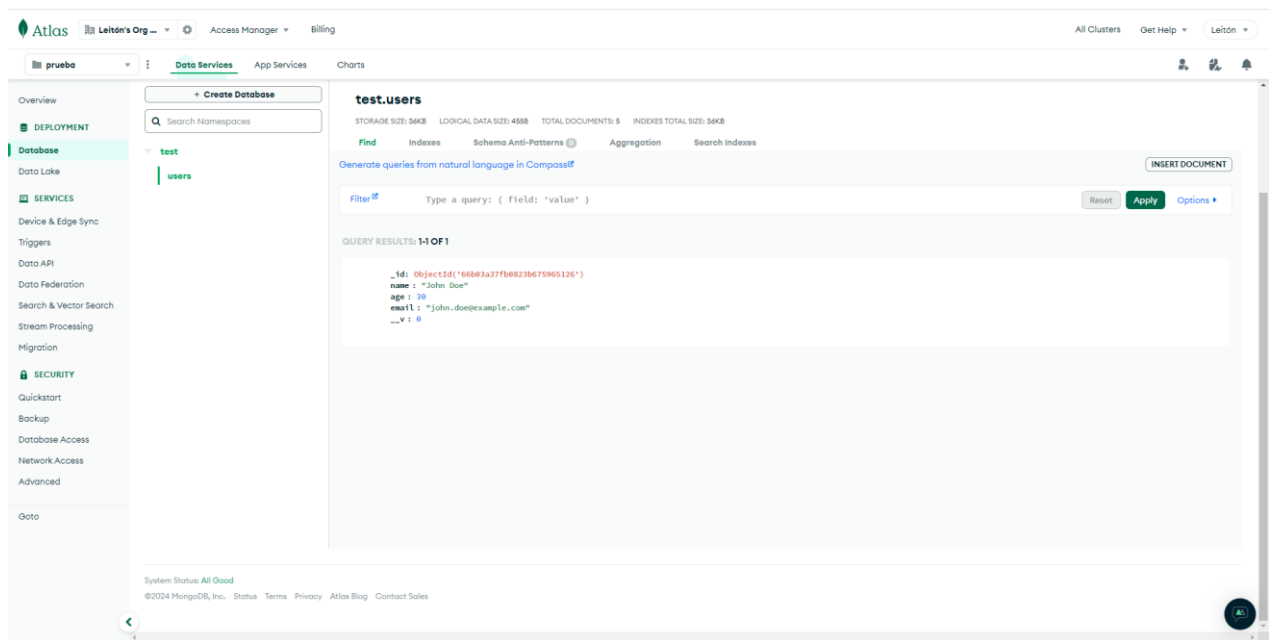


Se utiliza Mercurius como el servidor GraphQL y GraphiQL como la interfaz gráfica de usuario para interactuar con el servidor GraphQL

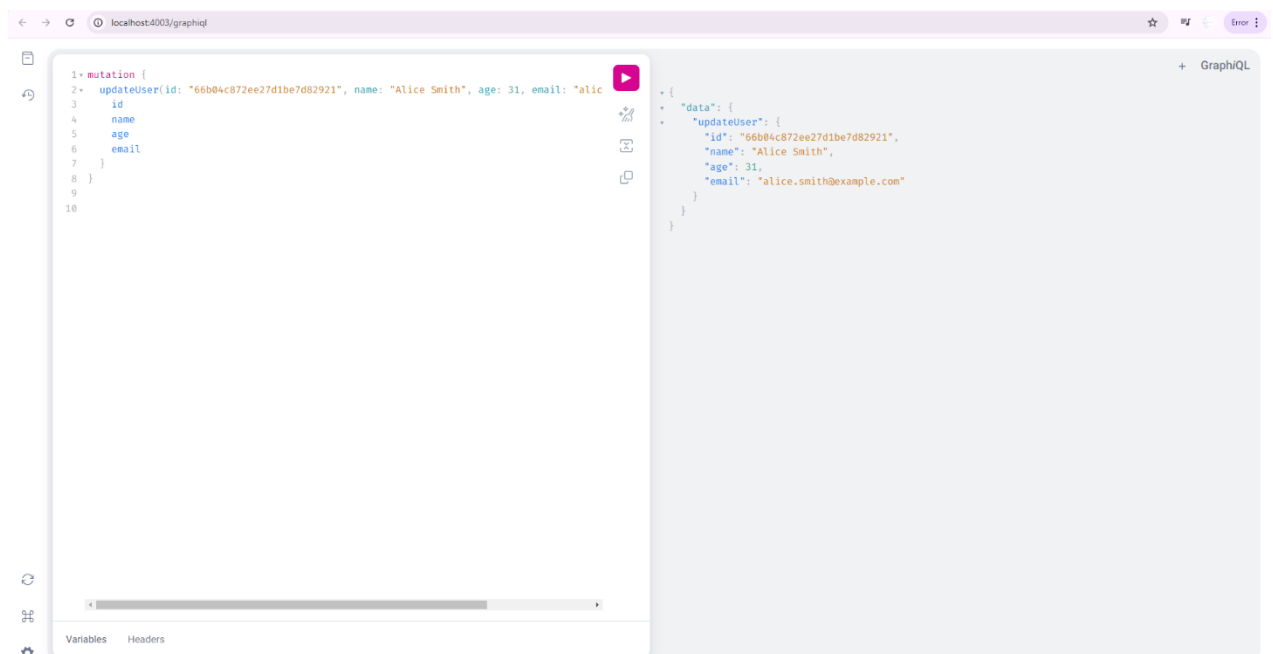
Mercurius es un plugin para Fastify que permite crear un servidor GraphQL de manera eficiente



Una vez ejecutada la mutación, iremos a verificar en nuestra base de datos de mongoDB Atlas que se haya hecho la inserción correctamente.



Podremos observar que los datos que ingresamos en la interfaz fueron añadidos correctamente en la base de datos.



En este caso realicé un update

test.users

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 279B TOTAL DOCUMENTS: 3 INDEXES TOTAL SIZE: 36KB

[Find](#) [Indexes](#) [Schema Anti-Patterns](#) [Aggregation](#) [Search Indexes](#)

[Generate queries from natural language in Compass](#) INSERT DOCUMENT

[Filter](#) Type a query: { field: 'value' } Reset Apply Options

QUERY RESULTS: 1-1 OF 1

```
{
  "_id": ObjectId('66b04c872ee27d1be7d82921'),
  "name": "Alice Smith",
  "age": 31,
  "email": "alice.smith@example.com",
  "__v": 0
}
```

El cual fue aplicado también en la base de datos.

Estas serían las consultas que podríamos realizar:

(Consultar Todos los Usuarios)

```
query {
  users {
    id
    name
    age
    email
  }
}
```

(Consultar un Usuario Específico o campos específicos)

```
query {  
  user(id: "ID_DEL_USUARIO") {  
    id  
    name  
    age  
    email  
  }  
}
```

(Consultar el Total de Usuarios)

```
query {  
  totalUsers  
}
```

(Eliminar un usuario)

```
mutation {  
  deleteUser(id: "ID_DEL_USUARIO") {  
    id  
    name  
    age  
    email  
  }  
}
```

(contar el número de usuarios por edad)

```
query {  
  usersByAge {  
    age  
    count  
  }  
}
```

(Agregar un nuevo usuario)

```
mutation {  
  addUser(name: "Alice", age: 30, email: "alice@gmail.com") {  
    id  
    name  
    age  
    email  
  }  
}
```

(Modificar un usuario)

```
mutation {  
  updateUser(id: "ID_DEL_USUARIO", name: "Bob Marley", age: 26, email:  
"bob.marley@gmail.com") {  
    id  
    name  
    age  
    email  
  }  
}
```