El futuro digital es de todos

MinTIC

Ciclo 4a:

Desarrollo de aplicaciones web

UNIVERSIDAD NACIONAL DE COLOMBIA

MisiónTIC2022

VERSIÓN 1.0

Unidad de educación continua y permanente
Facultad de Ingeniería

Unidad Camilo Torres
Calle 44 # 45-67
Bloque B5 piso 1

(57) + 316 5000
uec_fibog@unal.edu.co

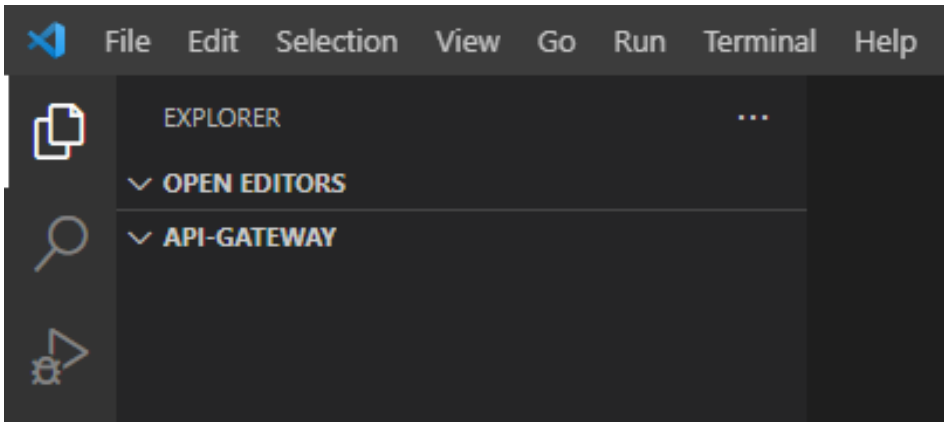
Actividad Práctica

(API Gateway - Parte 1)

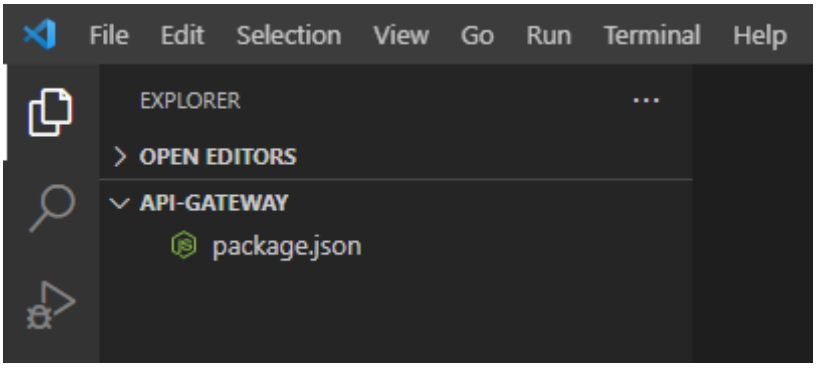
Para implementar el API Gateway, se utilizará *JavaScript*, junto con el entorno *NodeJS* y el servidor de GraphQL: *Apollo Server*. Como se vio anteriormente, NodeJS es un entorno de ejecución para el desarrollo de servidores. En el ciclo 3, este se utilizó para la creación del servidor Front-End, sin embargo, este es útil para cualquier tipo de servidor, razón por la cual, se utilizará en el desarrollo del API Gateway.

Estructura base del API Gateway

Para iniciar con el desarrollo del componente, primero se debe crear la carpeta del proyecto, y se debe abrir en Visual Studio Code. En este caso, esta carpeta se llamará *Api-gateway*:



Existen múltiples formas de iniciar un proyecto de NodeJS, sin embargo, el método más rápido es creando dentro de la carpeta del proyecto un archivo con el nombre *package.json*. Al hacerlo, el contenido de la carpeta debe ser el siguiente:

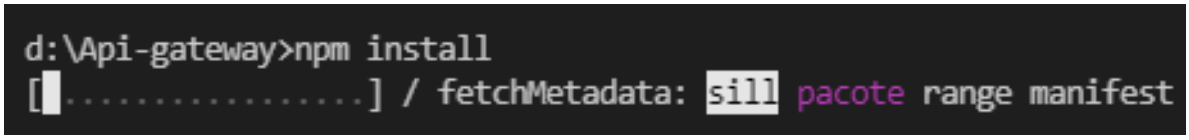


Todo proyecto de NodeJS debe tener este archivo, pues en él se encuentran los atributos del proyecto que utilizará NodeJS para administrar y configurar el servidor. Algunos atributos definidos dentro de este archivo son el nombre, la versión, y las dependencias del proyecto:

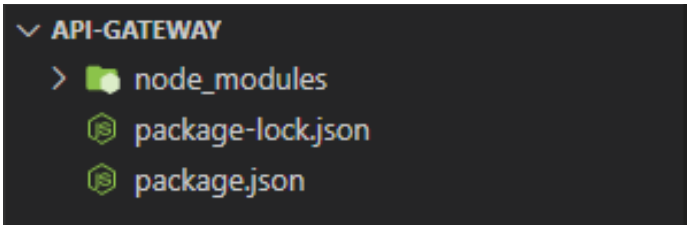
```
{
    "name": "apigateway",
    "version": "1.0.0",
    "description": "",
    "main": "src/index",
    "scripts": {},
    "author": "",
    "license": "ISC",
    "dependencies": {
        "apollo-datasource-rest": "^3.1.1",
        "apollo-server": "^3.1.2",
        "graphql": "^15.5.1",
        "lodash": "^4.17.21",
        "node-fetch": "^2.6.1"
    }
}
```

Como se puede notar, se utiliza un JSON con sus respectivas llaves y valores para configurar el proyecto. En este caso, los dos atributos definidos más importantes son el atributo “main”, y el atributo “dependencies”, pues se encargan de definir dónde se encuentra el archivo inicial, y cuáles son las dependencias del proyecto, respectivamente.

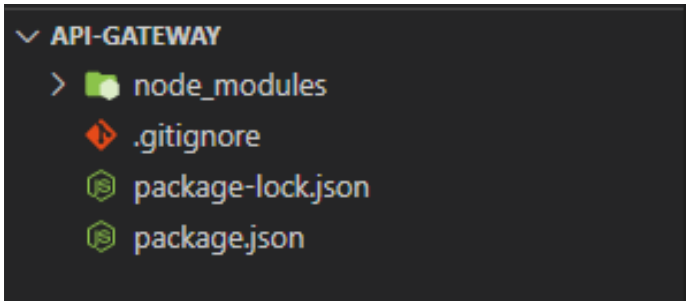
Una vez se configura el servidor, es necesario instalar las dependencias que se utilizan en el proyecto. Para ello, se debe abrir una terminal dentro de la carpeta del proyecto, y se debe utilizar el comando *npm install*:



Una vez finaliza su ejecución, este comando crea la carpeta *node_modules*, y el archivo *package-lock.json*. Tanto la carpeta como el archivo no deben ser modificados por el desarrollador, pues son aquello que utiliza NodeJS para la ejecución del servidor. En este punto, el contenido del proyecto debe ser el siguiente:



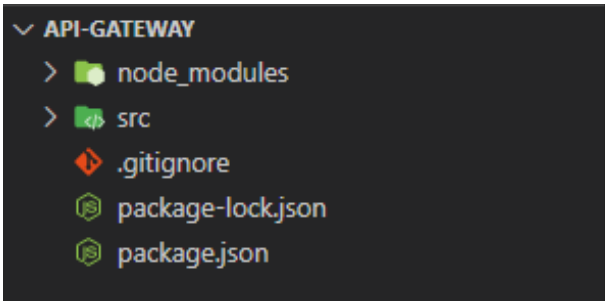
Ahora, para llevar el control de versiones del proyecto se debe crear el archivo *.gitignore*:



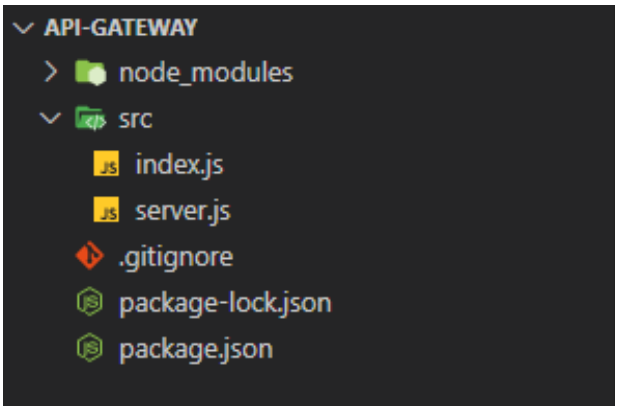
El código de dicho archivo debe ser similar al siguiente:

```
node_modules
dist

# local env files
.env.local
.env.*.local

# Log files
npm-debug.log*
yarn-debug.log*
yarn-error.log*

# Editor directories and files
.vscode
```

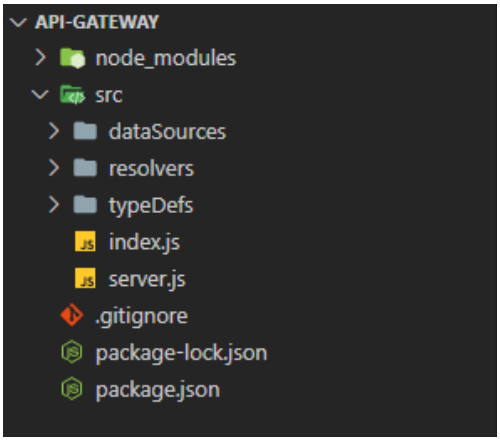
Luego de realizar las configuraciones pertinentes, se debe crear la estructura del proyecto. Para ello, lo primero que se debe hacer es crear la carpeta *src*, donde se ubicarán todos los archivos que conforman el API Gateway. Al hacerlo, la estructura del proyecto será la siguiente:



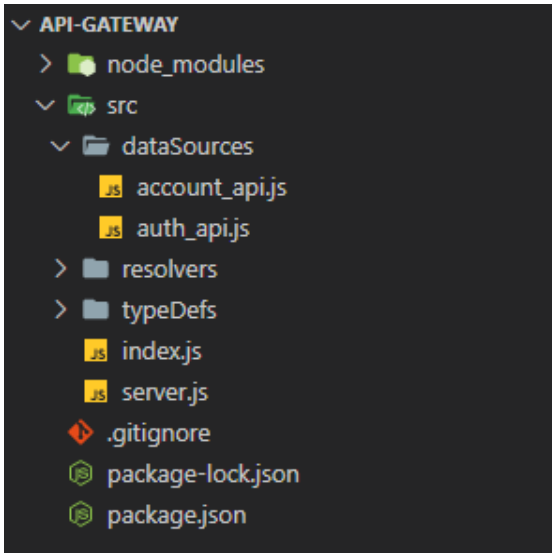
Dentro de esta carpeta se deben crear los archivos *index.js*, y *server.js*. Como se verá en próximas sesiones, estos se utilizarán para crear el servidor, y para definir algunos atributos de configuración del API Gateway respectivamente. Al crear ambos archivos, la estructura del proyecto será la siguiente:



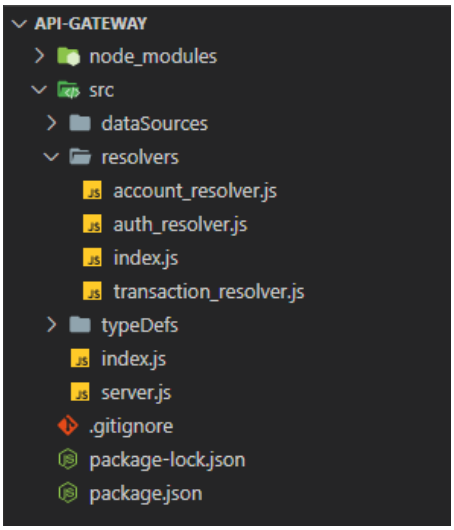
Ahora, dentro de la carpeta *src/* se deben crear las carpetas *dataSources*, *resolvers*, y *typeDefs*, donde se definirán cada uno de los componentes de la estructura de un servidor GraphQL. Al hacerlo, la estructura del proyecto será la siguiente:



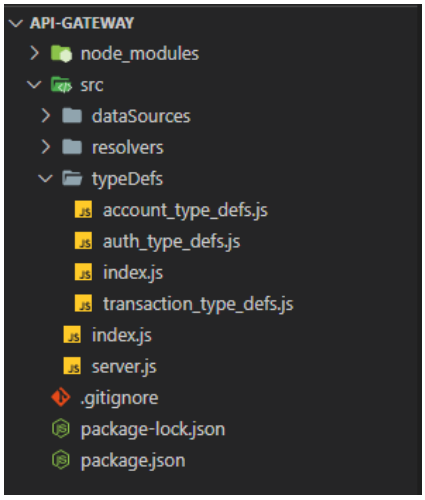
Dentro de la carpeta *dataSources/*, se ubicarán los archivos de conexión del API Gateway con cada microservicio. Por ello, en esta carpeta se deben crear los archivos *auth_api.js* y *account_api.js*. Al hacerlo, la estructura del proyecto será la siguiente:



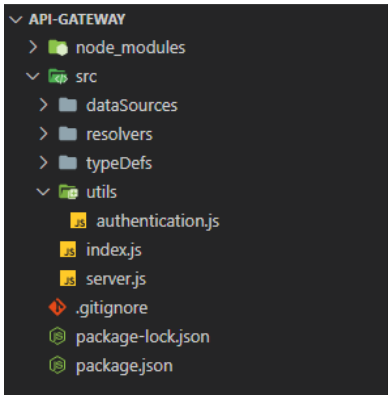
Adicionalmente, dentro de la carpeta *resolvers* se ubicarán los archivos que contendrán la definición de cada resolver del API Gateway. Para ello, es necesario crear los archivos *account_resolver.js*, *auth_resolver.js* y *transaction_resolver.js*, acompañados de un archivo index.js, que se encargará de integrar todos los resolvers en un único archivo al cual accederá el servidor GraphQL. Al crear todos estos archivos la estructura del proyecto será la siguiente:



Por otro lado, dentro de la carpeta *typeDefs* se ubicarán los archivos que contendrán la definición de cada uno de los tipos de datos que se utilizarán para realizar las solicitudes al servidor GraphQL. Para ello, se deben crear los archivos *account_type_defs.js*, *auth_type_defs.js* y *transaction_type_defs.js*, acompañados de un archivo index.js, que se encargará de integrar todos los *typeDefs* en un único archivo al cual accederá el servidor GraphQL. Al crear todos estos archivos la estructura del proyecto será la siguiente:



Finalmente, en la carpeta *src/* se debe crear la carpeta *utils*, donde se ubicarán archivos de configuración adicionales, como en este caso, el archivo *authentication.js*, donde se definirá el contexto que permitirá al servidor GraphQL verificar si una petición tiene un access token válido o no. Al crear ambos, la carpeta utils, y el archivo *authentication.js* dentro de esta carpeta, la estructura del proyecto debe ser la siguiente:



Nota: de ser necesario, en el material de la clase se encuentra un archivo llamado *C4a.AP.07. api_gateway.rar*, con el desarrollo del componente realizado en esta guía.