# REST API Primeros pasos

// iniciar proyecto node.js

*npm init -y* // Se crea el package.json

// instalartypescript local si no se instaló global

*npm i typescript --save --dev* //Local

*npm i typescript -g* //Global

// Iniciar proyecto typescript

https://medium.com/@bhagyamangale/tsc-init-4665ec9d7b09

*tsc --init* //Si global. Se crea el tsconfig.json

*npx tsc --init* //Si local. Se crea el tsconfig.json

*git init* // Para crear el repositorio local

// Cambiamos la configuración del *tsconfig.json*

*"target": "es6",*

*"outDir": "./build",*

// Instalación de express, mongoose y morgan

<https://dev.to/mtee/getting-started-with-morgan-3d1m>

// Morgan is a middleware function for logging information

// about the http request/response in a server application.

// Un middleware es un bloque de código que se ejecuta entre

// la petición que hace el usuario (request) hasta que la petición llega al servidor.

*npm i express mongoose morgan*

// nodemon is a tool that helps develop node.js based

// applications by automatically restarting the node

// application when file changes in the directory are detected.

// Instalamos los tipos de datos y módulos de *desarrollo*

*npm install @types/node @types/mongoose @types/express @types/morgan nodemon typescript -D*

// Configuramos el*.gitignore*con:

**// Atención no ignorar build si lo vamos a subir a heroku**

*node\_modules*

// Creamos la carpeta src con *server.ts* //Archivo typescript

Con el contenido que presentamos¨:

import express from 'express'

import morgan from 'morgan'

class Server {

    private app: express.Application

    constructor(){

        this.app = express()

        this.config()

    }

    config(){

        this.app.set('port', process.env.PORT || 3000)

        this.app.use(morgan('dev'))  // Para que muestre las url invocadas

    }

    routes(){

    }

    start(){

        this.app.listen(this.app.get('port'),

        () => {

            console.log(`Server on port: ${this.app.get('port')}`)

        })

    }

}

const server = new Server()

server.start()

// Cambiamos el *package.json* con:

  "scripts": {

    "ts": "tsc -w",

    "dev": "nodemon ./build/server.js",

    "start": "node ./build/server.js"

  },

// Para compilar

*npm run ts* // que como tiene el -w incorporado se compilará si cambiamos el código

// Para ejecutar en desarrollo

*npm run dev* // que como tiene el nodemon se reiniciará el servidor si cambiamos

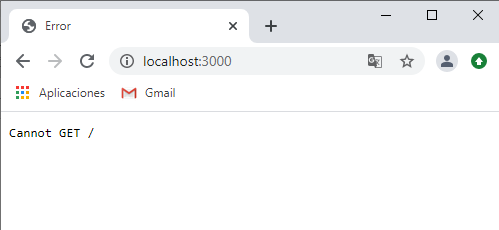
// además como tenemos

// Para ejecutar en producción

*npmstart*

// Ya podemos invocar con *localhost:3000*

Aunque como no tenemos rutas la salida en que no puede responder a la ruta /



*PS C:\Users\Adolfo3\Documents\ACurso2021\ASGBD\ProyectosTS\restapitriangulo000****>npm run dev***

*> restapitriangulo000@1.0.0 dev C:\Users\Adolfo3\Documents\ACurso2021\ASGBD\ProyectosTS\restapitriangulo000*

*>nodemon ./build/server.js*

*[nodemon] 2.0.6*

*[nodemon] to restart at any time, enter `rs`*

*[nodemon] watching path(s): \*.\**

*[nodemon] watching extensions: js,mjs,json*

*[nodemon] starting `node ./build/server.js`*

***Server on port: 3000***

***[nodemon] restarting due to changes...***

*[nodemon] starting `node ./build/server.js`*

*Server on port: 3000*

***GET / 404 1.984 ms – 139***

Git:

git init

git add .

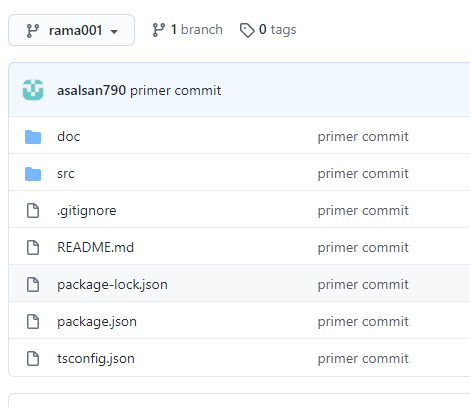
git commit -m "primer commit"

git branch -M rama001   // Escogemos el nombre de la rama

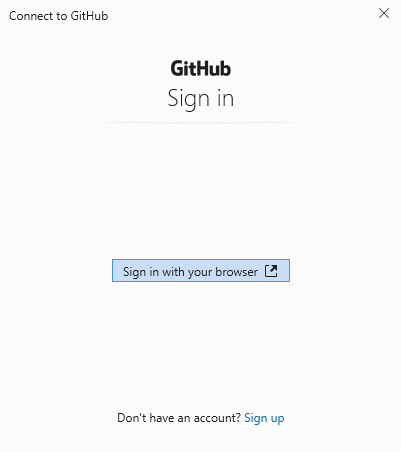
git remote add origin https://github.com/asalsan790/restapitriangulo000.git

git push -u origin rama001  // hacemo push de nuestra rama.

Usamos un nombre de rama, *rama001*, distinto del *main* que viene en la documentación



La última versión de git pide identificarse con el navegador:



En Windows 10 se puede elegir el navegador por defecto para que se abra el que deseemos:

1. Haz clic en el menú Inicio. ...
2. Haz clic en Configuración .
3. Abre las aplicaciones predeterminadas: ...
4. Haz clic en tu **navegador** actual (normalmente es **Microsoft** Edge) en la sección "Explorador web", situada en la parte inferior.

Creando nuevas ramas

//Para ver a donde apunta cada rama

git log --oneline --decorate

// Crear una nueva rama

git branch rama002

// Cambiar de rama

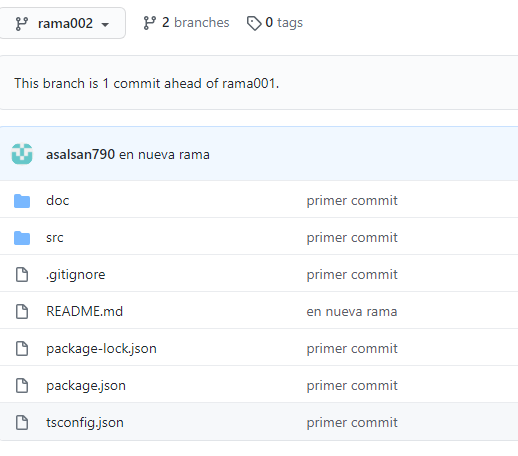
git checkout rama002

git add .

git commit -m "en nueva rama"

git push -u origin rama002

Resultado:



Para subir a heroku:

Después de crear a app en heroku desplegarla desde el GitHub donde la tenemos subida:

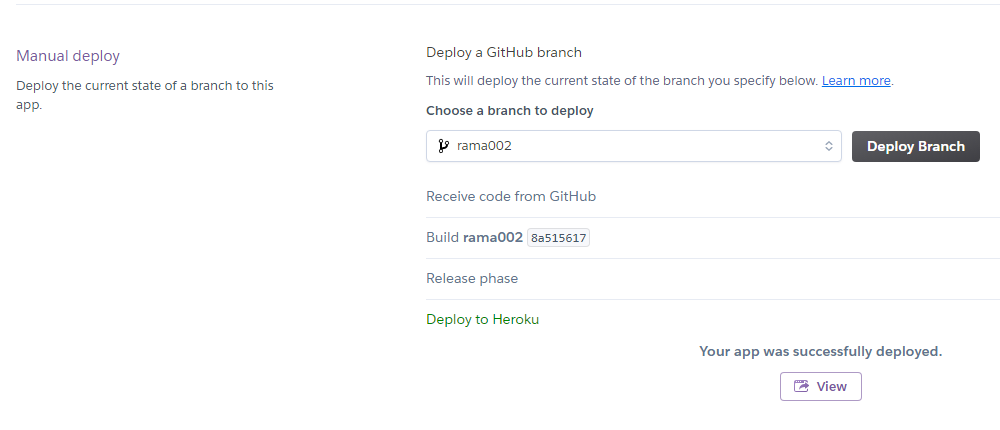
Tendrá que estar compilada de ts a js si estamos ejecutando en desarrollo

tsc –w

siempre lo estará

Luego:

* Que esté compilada
* Subida a GitHub
* Desplegada en heroku



Previamente hay que conectarse:

