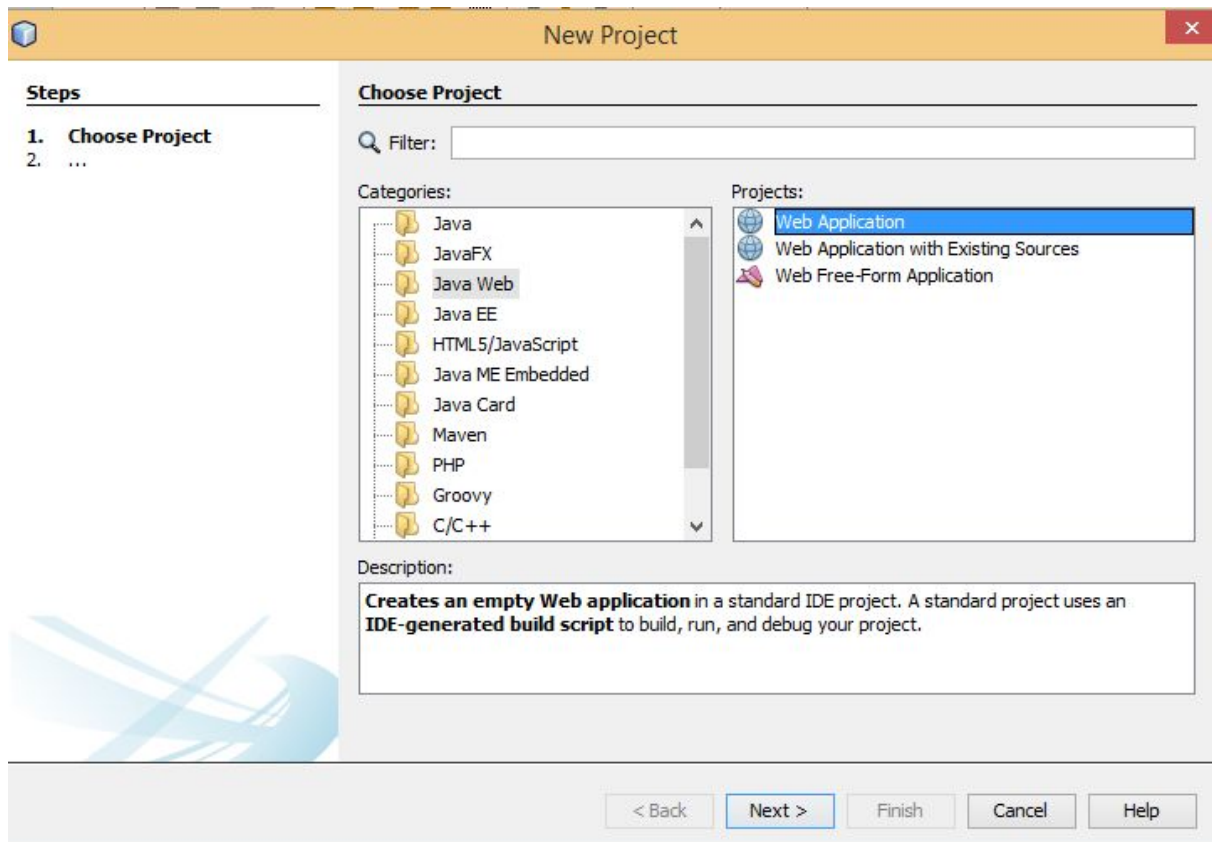


TRABAJO API REST PROGRAMACIÓN WEB

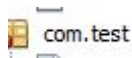
Abad L. Freddy L.

freddy.abadl@ucuenca.edu.ec

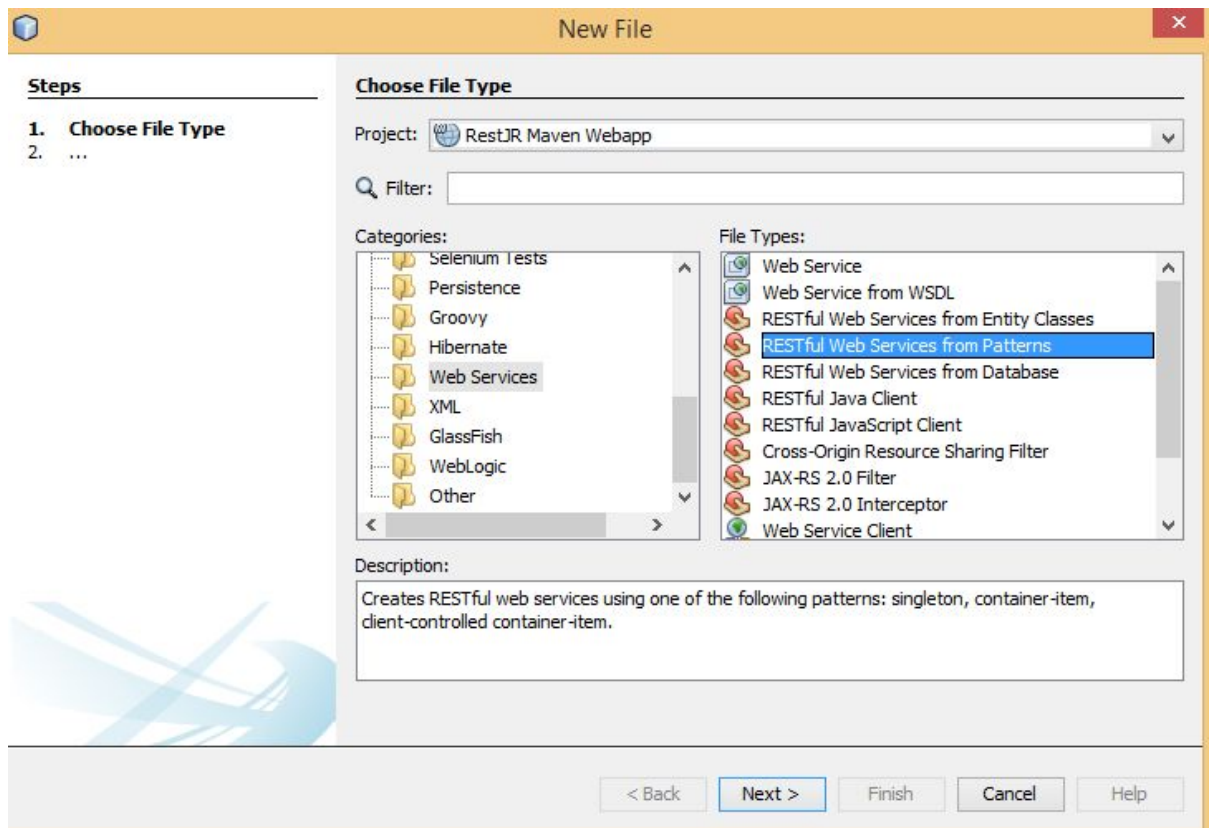
I. Método RestApi WebService Crear una Aplicación Web



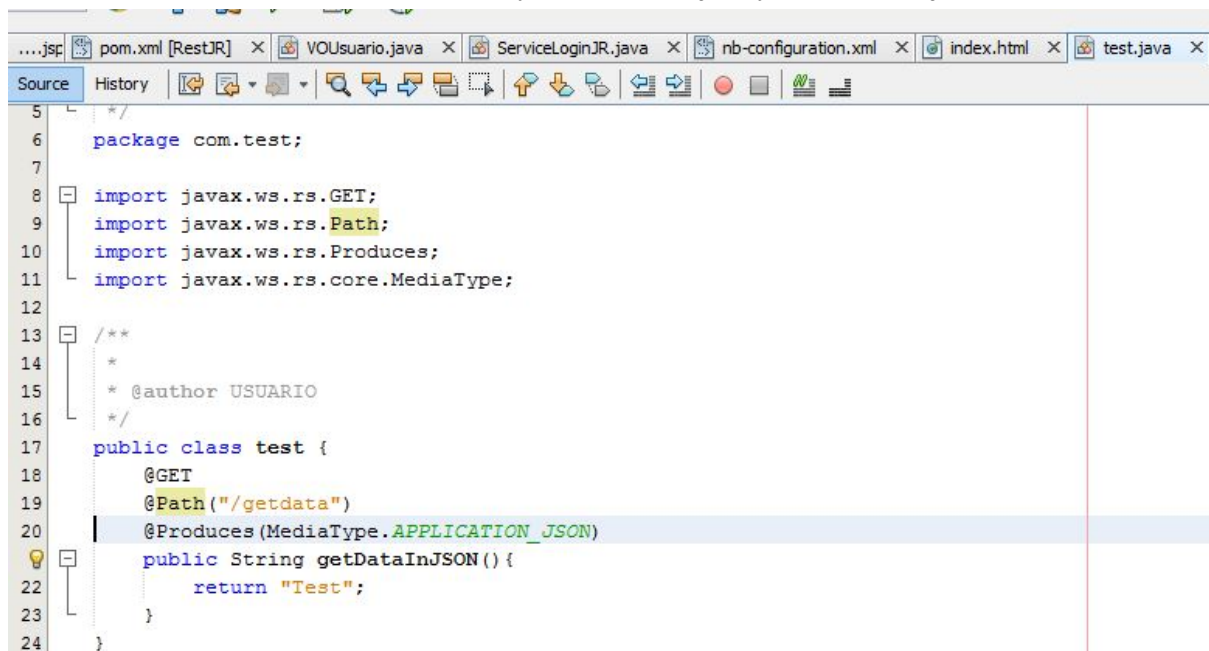
Crear un paquete con nombre com.test y en este el archivo test.java



Agregar WebServicefromPatern en el paquete



Eliminamos Archivo MyPathResource.java y creamos test.java

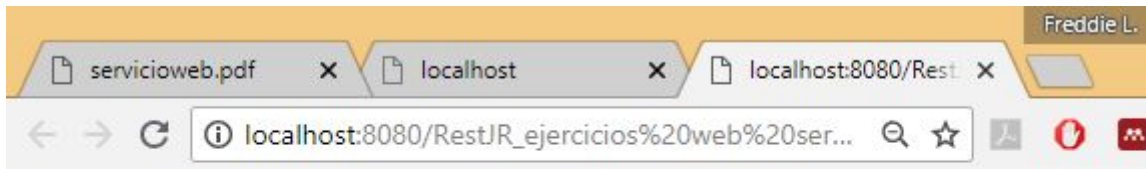


Agregamos en resource de el archivo ApplicationConfig.java

```
public Set<Class<?>> getClasses() {
    Set<Class<?>> resources = new java.util.HashSet<>();
    addRestResourceClasses(resources);
    return resources;
}
```

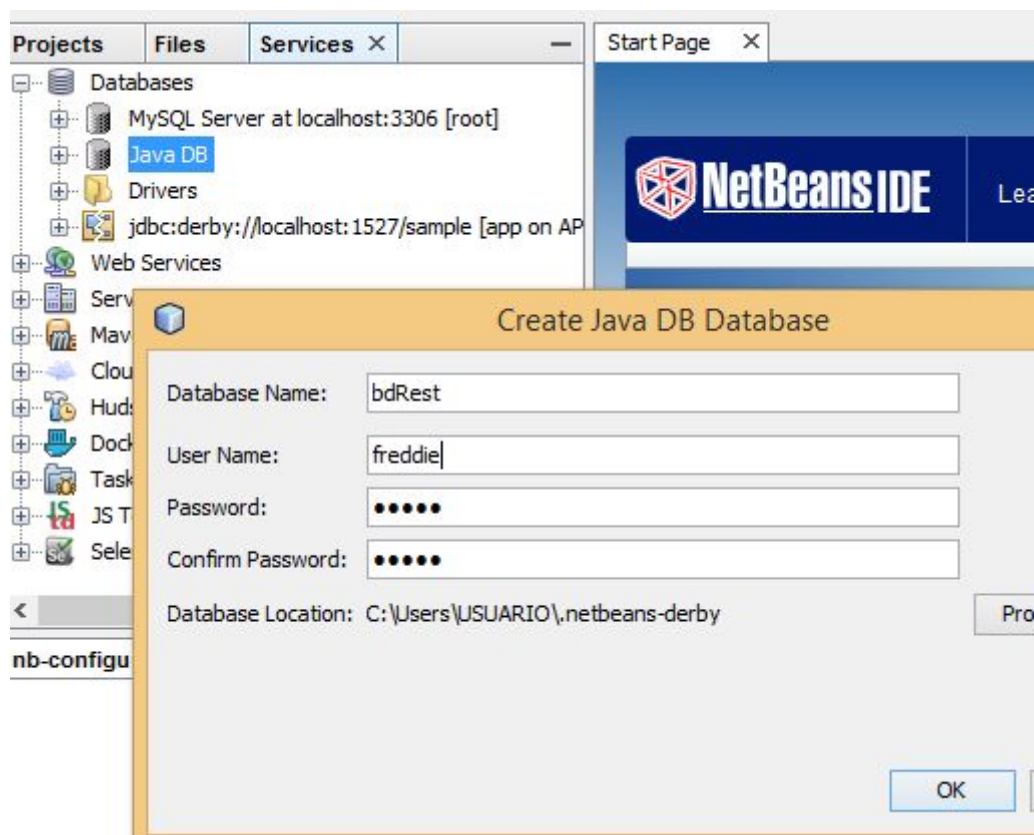
```
private void addRestResourceClasses(Set<Class<?>> resources) {
    resources.add(com.test.test.class);
}
```

Corremos Proyecto

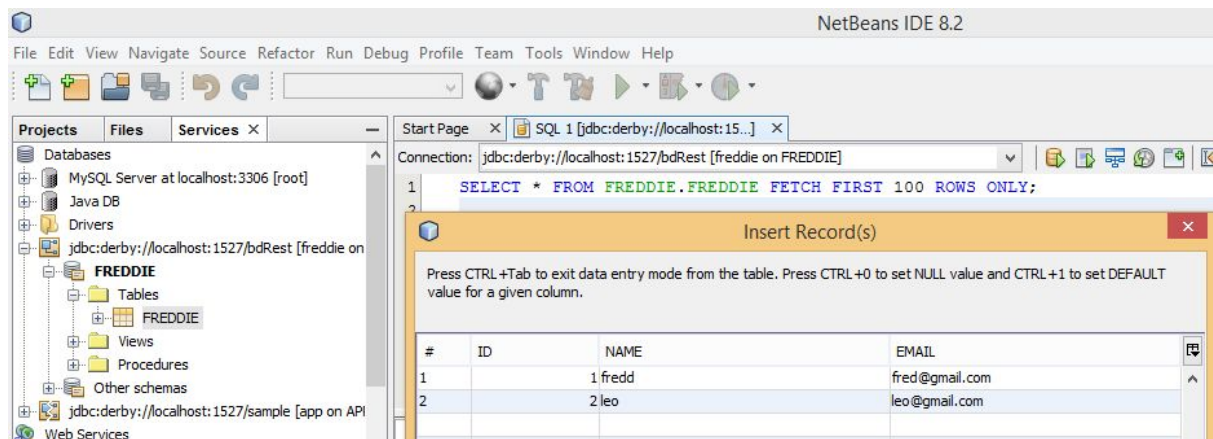


RestApi WebService - GET - Freddy Abad

Creamos una Base de Datos



Agregamos una tabla y datos



Copiar paths necesarios de jdbc>Properties

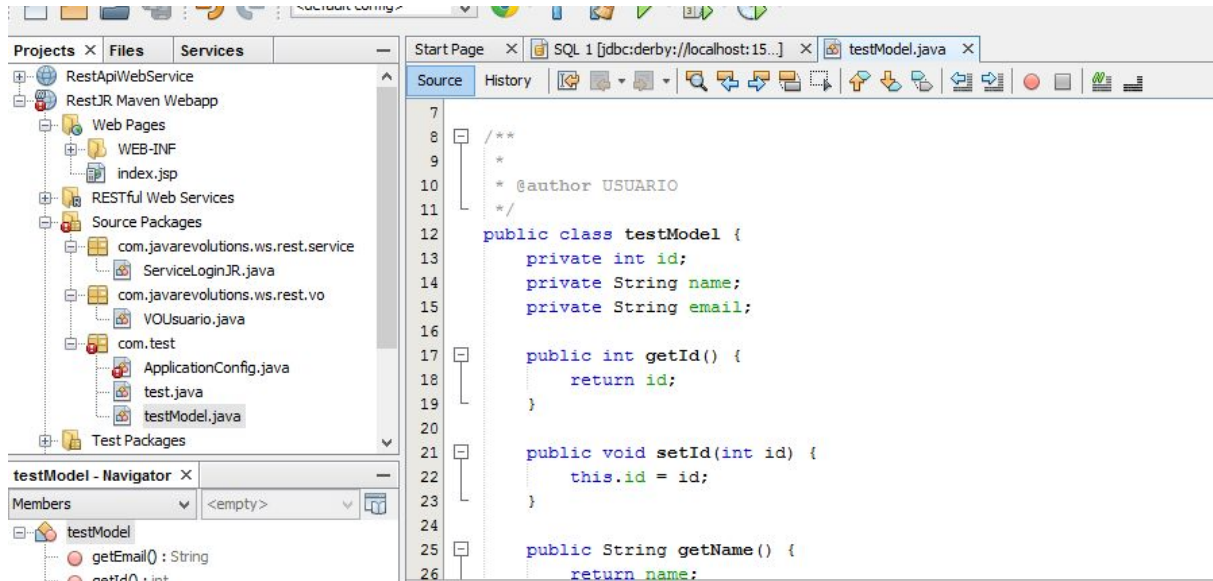
Database URL

jdbc:derby://localhost:1527/bdRest

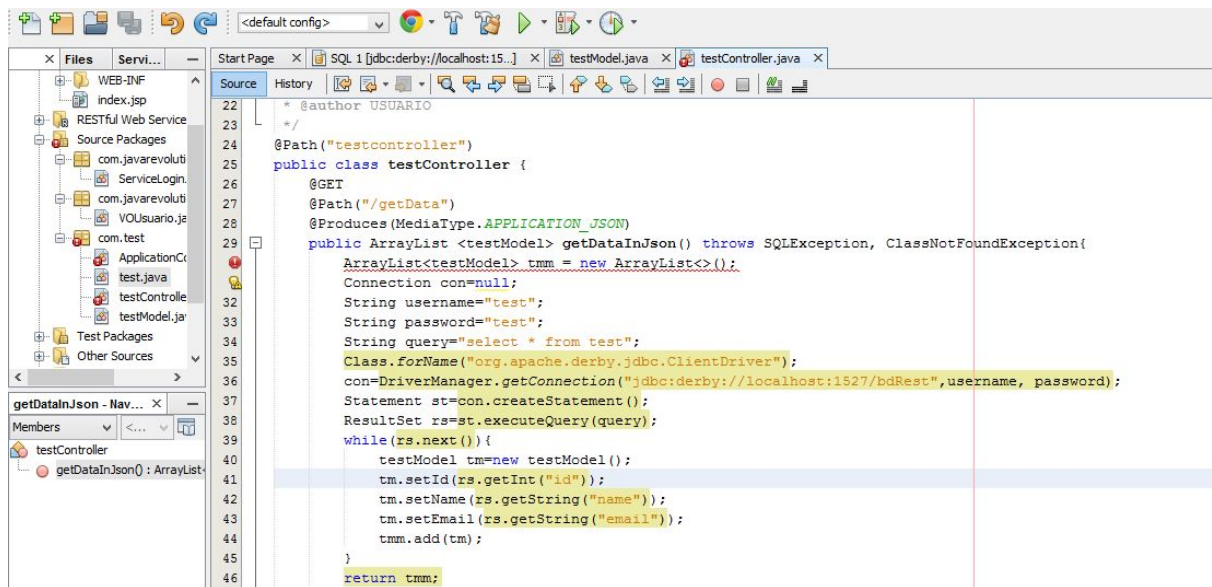
Driver Class

org.apache.derby.jdbc.ClientDriver

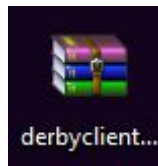
Creación de testModel



Crear clase classController



Agregar librería derbyclient.jar



Comprobación

```

[{"email":"abc@gmail.com","id":1,"name":"abc\n"}, {"email":"abc2@gmail.com","id":2,"name":"abc2"}]

```

LISTO.

II. MÉTODO.

API REST EN NODEJS

El presente artículo presenta los pasos a seguir para construir una API REST en NodeJS.

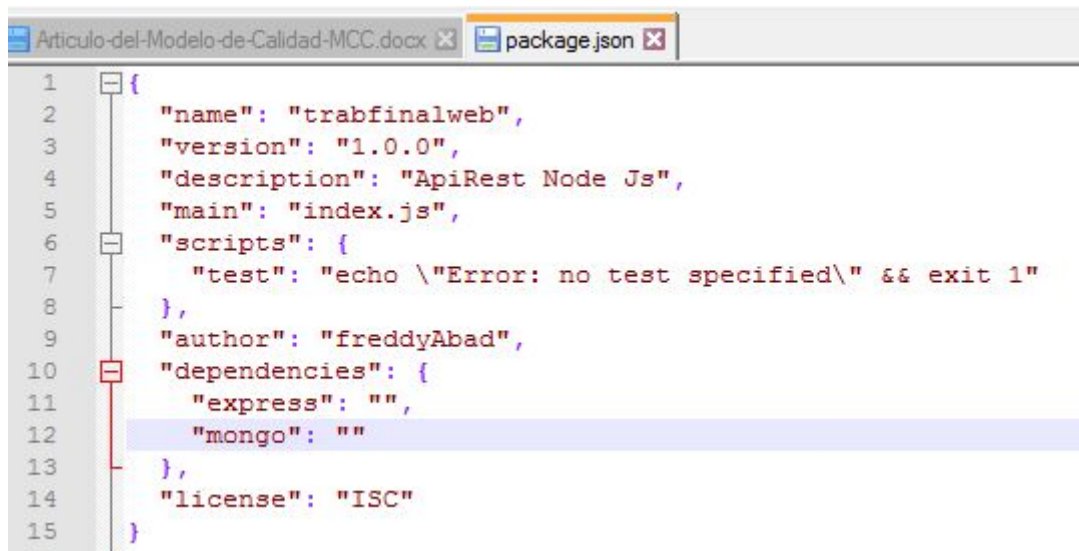
1.- Crear un directorio para el proyecto

```
C:\Users\USUARIO\Desktop>mkdir trabFinalWeb  
C:\Users\USUARIO\Desktop>cd trabFinalWeb  
C:\Users\USUARIO\Desktop\trabFinalWeb>dir  
Volume in drive C is T110653400C  
Volume Serial Number is 8034-9AEC
```

2.- Configurar archivo package.json

```
C:\Users\USUARIO\Desktop\trabFinalWeb>npm init  
This utility will walk you through creating a package.json file.  
It only covers the most common items, and tries to guess sensible defaults.  
  
See 'npm help json' for definitive documentation on these fields  
and exactly what they do.  
  
Use 'npm install <pkg>' afterwards to install a package and  
save it as a dependency in the package.json file.  
  
Press ^C at any time to quit.  
package name: (trabfinalweb)  
version: (1.0.0)  
description: ApiRest Node Js  
git repository:  
keywords:  
author: freddyAbad  
license: (ISC)  
About to write to C:\Users\USUARIO\Desktop\trabFinalWeb\package.json:  
<  
  "name": "trabfinalweb",  
  "version": "1.0.0",  
  "description": "ApiRest Node Js",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \"Error: no test specified\" && exit 1"  
  },  
  "author": "freddyAbad",  
  "license": "ISC"  
>
```

3.- Agregamos dependencias en package json



The screenshot shows a code editor with two tabs: 'Articulo-del-Modelo-de-Calidad-MCC.docx' and 'package.json'. The 'package.json' file is open, showing the following JSON structure:

```
1 {  
2   "name": "trabfinalweb",  
3   "version": "1.0.0",  
4   "description": "ApiRest Node Js",  
5   "main": "index.js",  
6   "scripts": {  
7     "test": "echo \"Error: no test specified\" && exit 1"  
8   },  
9   "author": "freddyAbad",  
10  "dependencies": {  
11    "express": "",  
12    "mongo": ""  
13  },  
14  "license": "ISC"  
15 }
```

4.- Construir módulos de Node

```
C:\Users\USUARIO\Desktop\trabFinalWeb>npm install
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN trabfinalweb@1.0.0 No repository field.
added 58 packages in 23.136s
```

5.- Crear y modificar server.js

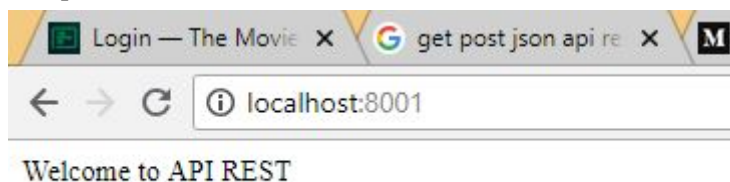
(Crear un servidor web capaz de servir peticiones web, sobre el cual montare la API REST)

```
1 var express = require('express')
2 var http = require('http')
3 var app = express()
4
5 app.get('/', (req, res) => {
6   res.status(200).send("Welcome to API REST")
7 })
8
9 http.createServer(app).listen(8001, () => {
10   console.log('Server started at http://localhost:8001');
11 });
12
```

Correr el servidor

```
C:\Users\USUARIO\Desktop\trabFinalWeb>node server.js
Server started at http://localhost:8001
```

Comprobación del servidor



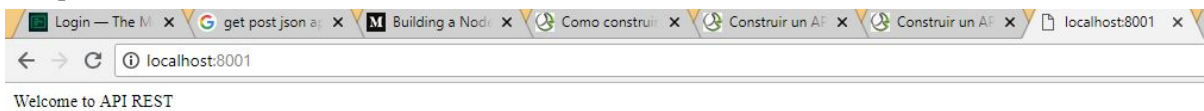
Modificación de server.js, para definir dos Routers, los cuales atienden peticiones GET, pero actúan por separado.

```
server.js — C:\Users\USUARIO\Desktop\trabFinalWeb —
File Edit View Selection Find Packages Help

Project
  ▾ trabFinalWeb
    ▸ node_modules
    package-lock.json
    package.json
    server.js

server.js
1  var express = require('express')
2  var http = require('http')
3  var app = express()
4
5  var users = ['oscar', 'juan', 'marcos']
6
7  app.get('/users', (req, res) => {
8    res.send(users)
9  })
10
11 app.get('/', (req, res) => {
12   res.status(200).send("Welcome to API REST")
13 })
14
15 http.createServer(app).listen(8001, () => {
16   console.log('Server started at http://localhost:8001');
17 });
```

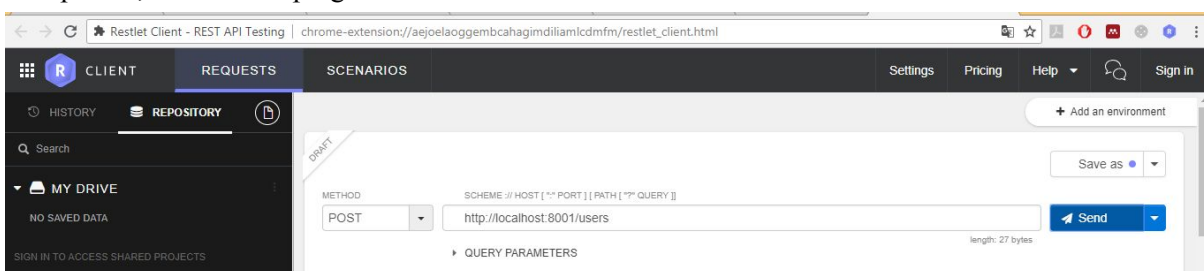
Comprobación con localhost:8001 & localhost:8001/users



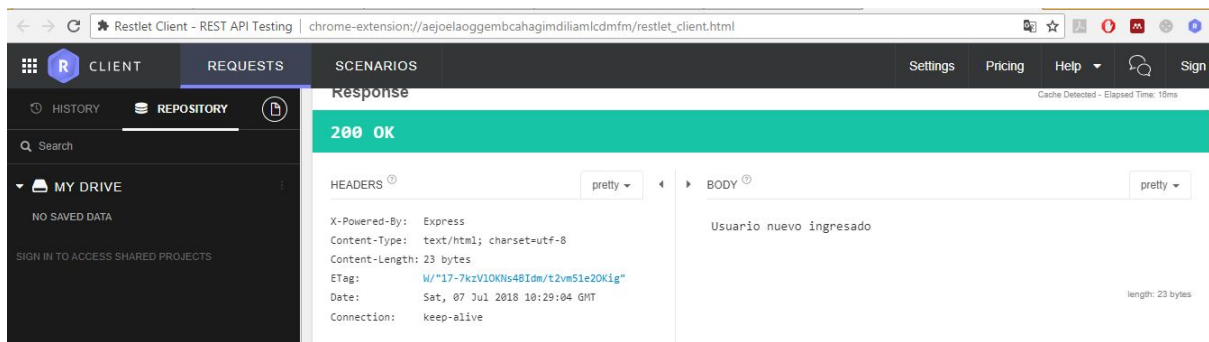
Añadir funcionalidades , donde el nuevo Router atiende peticiones POST en el path /users, y tiene como funcionalidad agregar un nuevo usuario a la lista actual de usuario y retorna un mensaje al cliente.

```
19  app.post('/users', (req, res) => {
20    users.push('User ' + users.length)
21    res.send("New user add")
22  })
23
```

Para probar, se utiliza el plugin de Reslet Client:



Comprobación



Modificación de server.js para interacción entre paths

