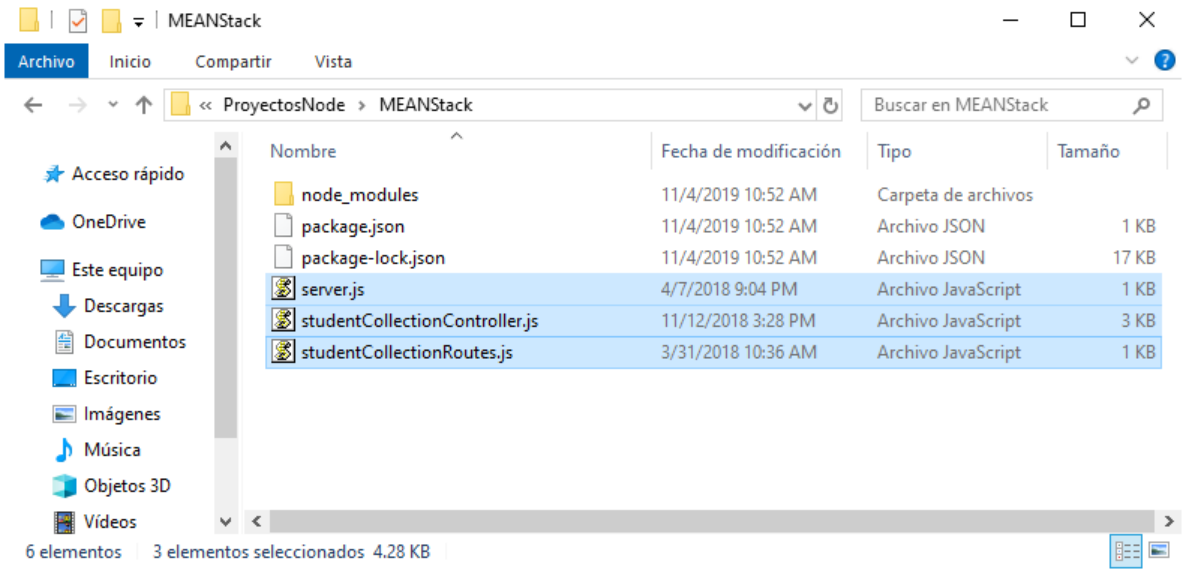
Práctica Grupal: Cliente REST con Angular

1. Iniciar el manejador de base de datos de MongoDB, y un Shell de MongoDB.
2. Crear la base de datos llamada “alumnos_final” y cargar el script proporcionado con esta práctica.
3. Cargar el script proporcionado en clase con el comando:
4. Validar la carga del script con Compass o con comandos.
5. Crear una carpeta para el proyecto de los servicios REST de NodeJS, e inicializarlo con npm init.
6. Instalar express, body-parser, y mongodb en el proyecto de NodeJS.
7. Colocar los archivos proporcionados en la clase dentro de la carpeta del proyecto:



8. Revisamos el archivo “server.js” ya que contiene elementos nuevos:

```
var express = require('express'),
  app = express(),
  port = process.env.PORT || 8585,
  bodyParser = require('body-parser');

app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());
```

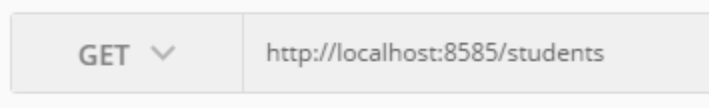
```
//Por configuracion default de seguridad, NodeJS no permite invocaciones
//de aplicaciones externas. Tenemos que permitirlo ya que invocaremos los
//servicios REST desde Angular que estará corriendo en el puerto 4200.
app.use(function (req, res, next) {

    res.setHeader('Access-Control-Allow-Origin', 'http://localhost:4200');
    res.setHeader('Access-Control-Allow-Methods', 'GET, POST');
    res.setHeader('Access-Control-Allow-Headers', 'X-Requested-With,content-type');
    res.setHeader('Access-Control-Allow-Credentials', true);

    next();

});

var routes = require('./studentCollectionRoutes');
routes(app);

app.listen(port);

console.log('Servidor escuchando en puerto: ' + port);
```

9. Ejecutar el servidor.

10. Utilice Postman para probar sus servicios:

    a. GET Students:

        i. Elija el método GET y utilice la URL http://localhost:8585/students:

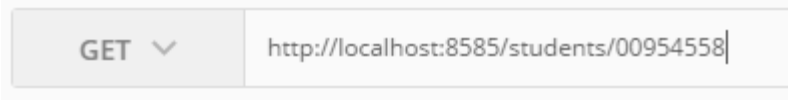


        ii. Haga clic en SEND. Postman debe mostrar todos los registros de la colección "alumno". Valide la información mostrada utilizando Compass:


```
[{"_id":"5be9ea9923455af8dc169f43","nombre":"Rick Sanchez","matricula":"00954558","foto":"http://pm1.narvii.com/6545
/8f765c17c81c456b0640a1513fa2d5862ac59c42_00.jpg"},{"_id":"5be9ea9923455af8dc169f44","nombre":"Morty Smith","matricula":"00954557"
,"foto":"https://images.8tracks.com/cover/i/012/765/788/Eyepatchevilmorty-8791.jpg"},{"_id":"5be9ea9923455af8dc169f45","nombre"
:"Jerry Smith","matricula":"00954556","foto":"https://pbs.twimg.com/profile_images/738078769920020481/xpW4r-Tr_400x400.jpg"}]
```


    b. GET student por matrícula:

        i. Elija el método GET y utilice la URL http://localhost:8585/students/00954558:



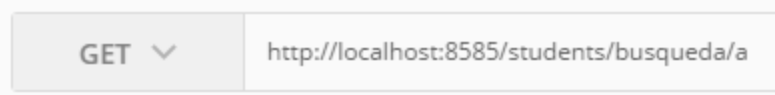
ii. Haga clic en SEND. Postman debe mostrar el registro que corresponda a esa matrícula de "alumno". Valide la información mostrada utilizando Compass:


```
{"_id":"5be9ea9923455af8dc169f43","campus":"CEM","nombre":"Rick Sanchez","matricula":"00954558","materias":[{"_id":"TC3052","nombre"
    :"Laboratorio de Desarrollo de Aplicaciones Web"},{"_id":"TC2026","nombre":"Desarrollo de Aplicaciones Web"},{"_id":"TC2025"
    ,"nombre":"Programacion Avanzada"}],"foto":"http://pm1.narvii.com/6545/8f765c17c81c456b0640a1513fa2d5862ac59c42_00.jpg"}
```


c. GET student por palabra clave:

i. En Postman, elija el método GET y utilice la URL

http://localhost:8585/students/busqueda/a



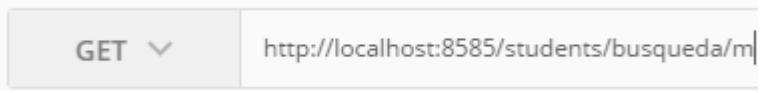

```
[{"_id":"5be9ea9923455af8dc169f43","campus":"CEM","nombre":"Rick Sanchez","matricula":"00954558","materias":[{"_id":"TC3052","nombre"
    :"Laboratorio de Desarrollo de Aplicaciones Web"},{"_id":"TC2026","nombre":"Desarrollo de Aplicaciones Web"},{"_id":"TC2025"
    ,"nombre":"Programacion Avanzada"}],"foto":"http://pm1.narvii.com/6545/8f765c17c81c456b0640a1513fa2d5862ac59c42_00.jpg"}]
```


En el log de NodeJS debe mostrarse:

```
Resultados Obtenidos: 1
```

ii. En Postman, elija el método GET y utilice la URL

http://localhost:8585/students/busqueda/m



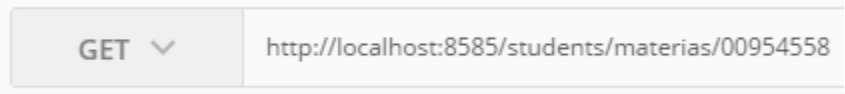

```
[{"_id":"5be9ea9923455af8dc169f44","campus":"CEM","nombre":"Morty Smith","matricula":"00954557","materias":[{"_id":"TC3052","nombre"
    :"Laboratorio de Desarrollo de Aplicaciones Web"},{"_id":"TC2025","nombre":"Programacion Avanzada"}],"foto":"https://images.8tracks
    .com/cover/i/012/765/788/Eyepatchevilmorty-8791.jpg"},{"_id":"5be9ea9923455af8dc169f45","campus":"CEM","nombre":"Jerry Smith"
    ,"matricula":"00954556","materias":[{"_id":"TC2025","nombre":"Programacion Avanzada"}],"foto":"https://pbs.twimg.com/profile_images
    /738078769920020481/xpW4r-Tr_400x400.jpg"}]
```


En el log de NodeJS debe mostrarse:

```
Resultados Obtenidos: 2
```

d. GET materias de un estudiante en específico (únicamente los nombres de las materias):

i. En Postman, elija el método GET y utilice la URL
http://localhost:8585/students/materias/00954558




```
{"materias":[{"nombre":"Laboratorio de Desarrollo de Aplicaciones Web"},{"nombre":"Desarrollo de Aplicaciones Web"},{"nombre"
    :"Programacion Avanzada"}]}
```

11. Crear una carpeta para el proyecto de Angular.

12. Abrir una ventana de comandos y ubicarse dentro de la carpeta creada en el paso anterior, y crear un proyecto de Angular.

13. Abrir el proyecto en Atom.

14. Integrar bootstrap al proyecto.

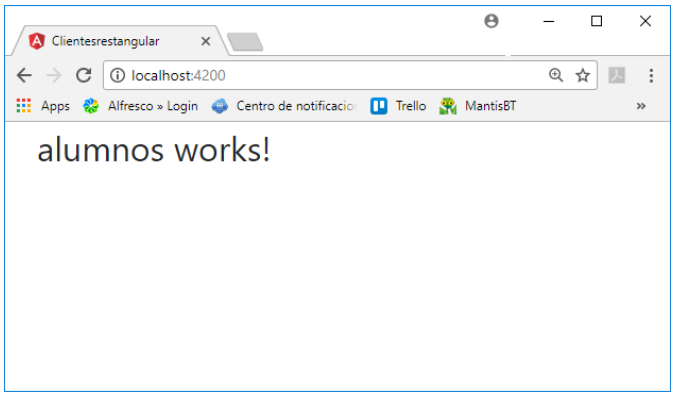
15. Validar que el proyecto se ejecute y muestre la página default.

16. Crear un componente llamado “componentes/alumnos”.

17. Cambiamos el código del archivo “app.component.html” por el siguiente:

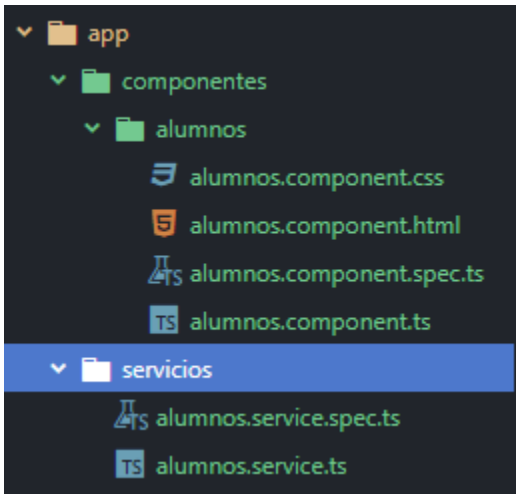
```
<div class="container">
  <app-alumnos></app-alumnos>
</div>
```

18. Validamos el cambio. La aplicación debe mostrar lo siguiente:



19. Crear un servicio con el comando “ng g s servicios/alumnos”:

```
PS C:\JMA\Personal\ITESM\SemestreAgostoDiciembre2018\ProyectosAngular\ClienteRESTAngular\clienterestangular> ng g s servicios/alumnos
CREATE src/app/servicios/alumnos.service.spec.ts (380 bytes)
CREATE src/app/servicios/alumnos.service.ts (136 bytes)
```



20. Declarar el servicio en el archivo “app.module.ts”:

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';
import { AlumnosComponent } from './componentes/alumnos/alumnos.component';

import { AlumnosService } from './servicios/alumnos.service';
import { HttpClientModule } from '@angular/common/http';

@NgModule({
  declarations: [
    AppComponent,
    AlumnosComponent
  ],
  imports: [
    BrowserModule,
    HttpClientModule
  ],
  providers: [
    AlumnosService
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

21. Cambiamos el código del archivo “alumnos.service.ts” por el siguiente:

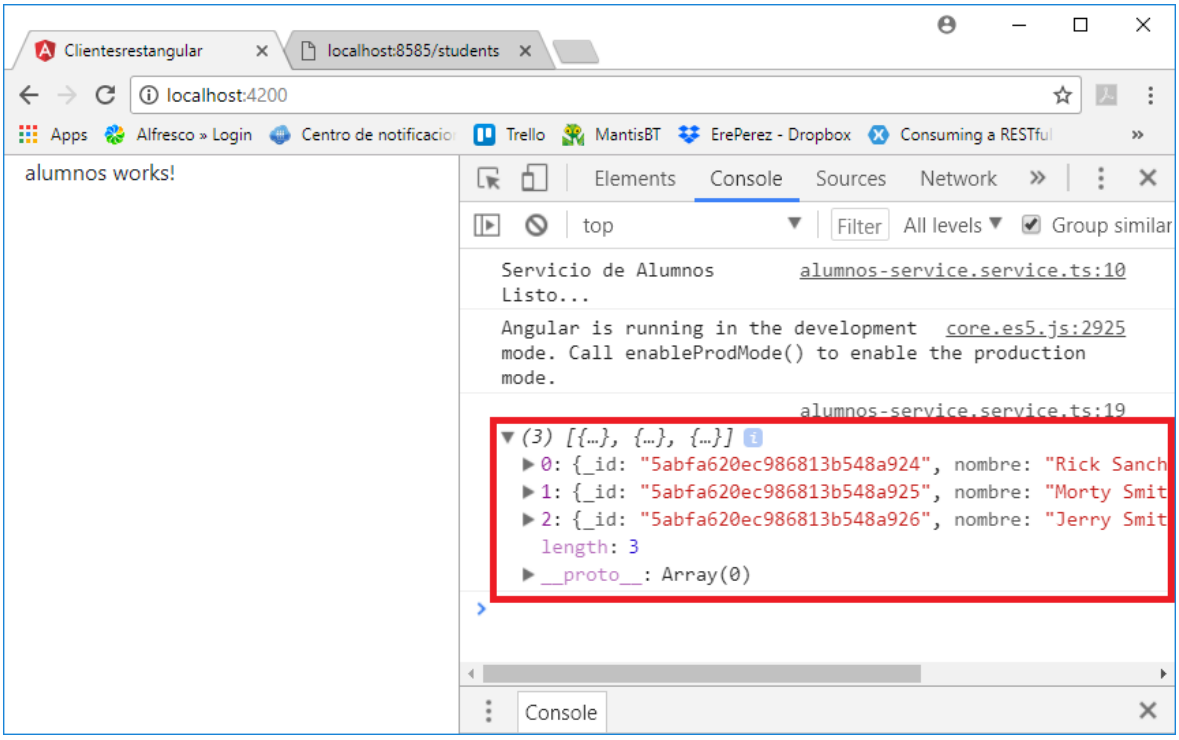
```
import { Injectable } from '@angular/core';

import { HttpClient } from "@angular/common/http";

//No es un @Component si no un @Injectable
@Injectable()
export class AlumnosService {

 constructor( public httpClient:HttpClient ) {
  console.log('Servicio de Alumnos Listo...');
 }

//Regresa un Observable
 getAlumnos(){
  let servicioRest = 'http://localhost:8585/students';
  return this.httpClient.get(servicioRest);
 }

}
```

22. Cambiamos el código del archivo “alumnos.component.ts” por el siguiente:

```
import { Component, OnInit } from '@angular/core';

import { AlumnosService } from "../../servicios/alumnos.service";

@Component({
 selector: 'app-alumnos',
 templateUrl: './alumnos.component.html',
 styleUrls: ['./alumnos.component.css']
})
export class AlumnosComponent implements OnInit {

//Vamos a utilizer el pipe async que solo puede recibir un Observable o un Promise
 alumnosAsincrono:any;

 constructor( public alumnosService : AlumnosService ) {

  this.alumnosAsincrono = new Promise( (resolve, reject) => {
   this.alumnosService.getAlumnos().subscribe(
    alumnos => {
     console.log(alumnos);
     resolve(alumnos);
    }
   )
  })
```

```
  }

  ngOnInit() {
  }

}
```

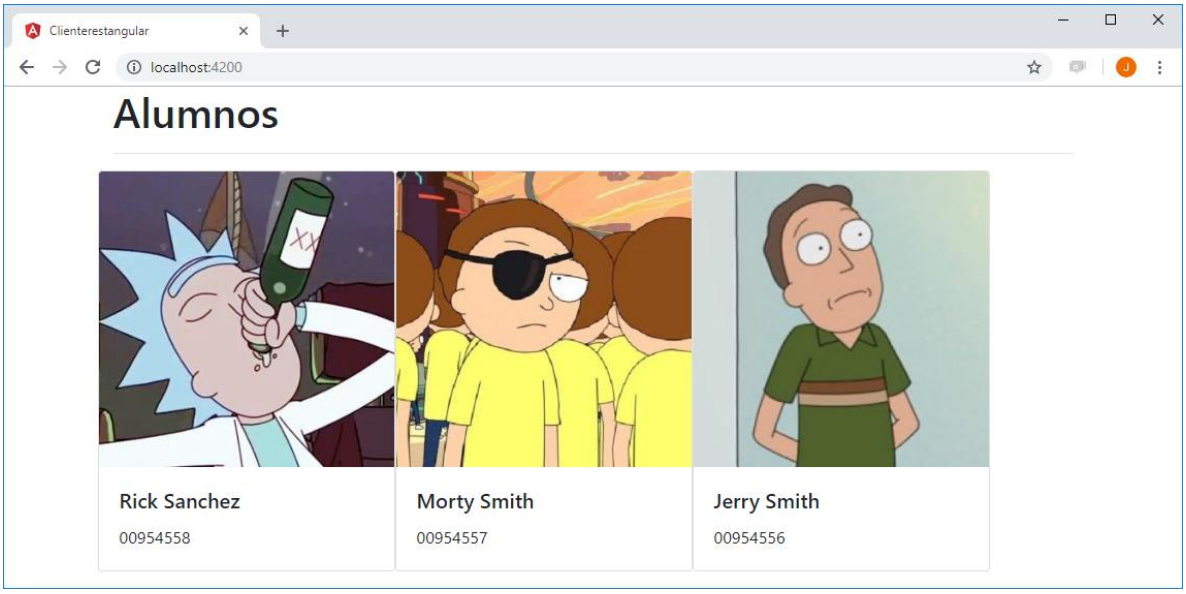
23. Validamos la invocación del servicio en la consola del navegador:



24. Cambiamos el código del archivo “alumnos.component.html” por el siguiente:

```
<h1>Alumnos</h1>
<hr>
<div class="row">
 <div class="card" style="width: 18rem;" *ngFor="let alumno of alumnosAsincrono | async; let i
= index">
   <img class="card-img-top" [src]="alumno.foto" [alt]="alumno.nombre">
   <div class="card-body">
    <h5 class="card-title">{{alumno.nombre}}</h5>
    <p class="card-text">{{alumno.matricula}}</p>
   </div>
 </div>
</div>
```

25. Validamos la funcionalidad actual:



26. Implementamos el buscador de alumnos:

    a. Agregamos el siguiente código al archivo “alumnos.component.html”:

```
<h1>Alumnos</h1>
<hr>
<form class="form-inline my-2 my-lg-0">
  <input class="form-control mr-sm-2" type="text" placeholder="Palabras de Búsqueda" #palabrasBusqueda
  <button class="btn btn-outline-success my-2 my-sm-0" type="button" (click)="buscarAlumnos(palabrasBu
</form>
<hr>
<div class="row">
  <div class="card" style="width: 18rem;" *ngFor="let alumno of alumnosAsincrono | async; let i = inde
    <img class="card-img-top" [src]="''+alumno.foto" [alt]="alumno.nombre">
    <div class="card-body">
      <h5 class="card-title">{{alumno.nombre}}</h5>
      <p class="card-text">{{alumno.matricula}}</p>
    </div>
  </div>
</div>
```

```
<form class="form-inline my-2 my-lg-0">
 <input class="form-control mr-sm-2" type="text" placeholder="Palabras de Búsqueda" #palabrasBusqueda>
 <button class="btn btn-outline-success my-2 my-sm-0" type="button" (click)="buscarAlumnos(palabrasBusqueda.value)">Buscar Alumnos</button>
</form>
<hr>
```

b. Implementamos el cliente del servicio REST en el archivo “alumnos.service.ts”:

```
getAlumnosPalabraClave(palabraClave:string){
  //Utilizamos template literals
  let servicioRest = `http://localhost:8585/students/busqueda/${palabraClave}`;
  return this.httpClient.get(servicioRest);
 }
```

c. Implementamos la función “buscarAlumnos(palabraClave)” en el archivo “alumnos.component.ts”:

```
buscarAlumnos(palabraClave:string){
  if(palabraClave.length > 0){
   this.alumnosAsincrono = new Promise( (resolve, reject) => {
    this.alumnosService.getAlumnosPalabraClave(palabraClave).subscribe(
     alumnos => {
      console.log(alumnos);
      resolve(alumnos)
     }
    )
   })
  }
 }
```

d. Validamos la funcionalidad: