

Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Análisis y Diseño de Sistemas 1
Curso de Vacaciones, Junio 2021
Ing. José Manuel Ruíz Juárez
Aux. César Sazo



SCRUMS Grupo#14

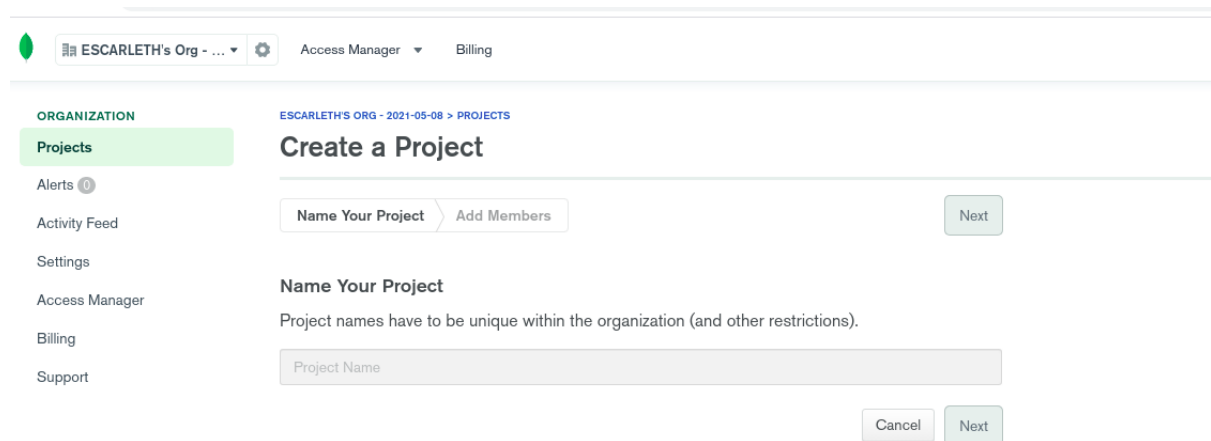
Tecnologías Utilizadas

- MongoDB
- Angular
- Node.js

Mongo: Debido a que el modelo es únicamente una tabla se decidió utilizar una base de datos no relacional en este caso usamos Atlas Mongo que es un servicio de mongo en la Nube que nos permite a todos tener acceso al mismo modelo y trabajar de una manera más eficiente y rápida ya que solo se necesita configurar una vez y todo el grupo la puede utilizar desde cualquier lugar.

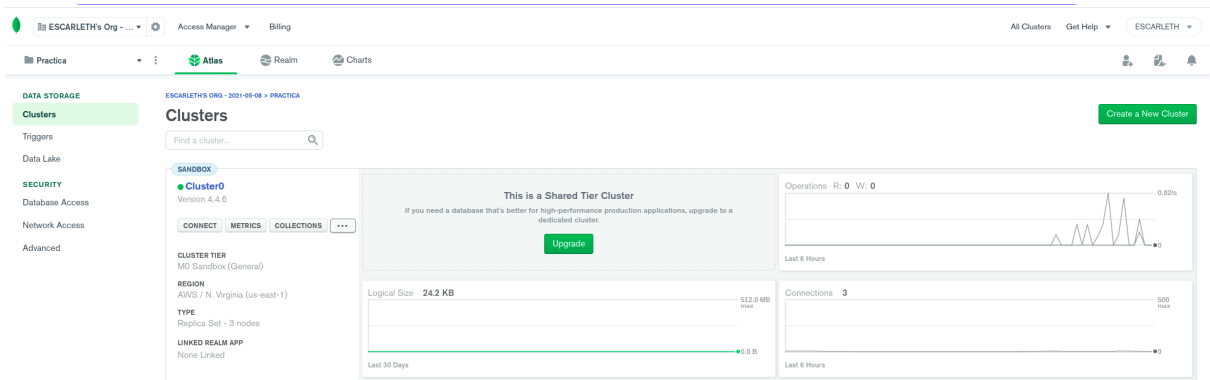
Pasos para configuración:

- Ingresar a <https://cloud.mongodb.com/>
- crear una cuenta ya sea gratuita o de pago
- Crear un nuevo proyecto



The screenshot shows the MongoDB Atlas web interface for creating a new project. At the top, there's a navigation bar with the MongoDB logo, a dropdown menu for 'ESCARLETH's Org', and links for 'Access Manager' and 'Billing'. On the left, a sidebar lists navigation options: 'ORGANIZATION', 'Projects' (highlighted), 'Alerts', 'Activity Feed', 'Settings', 'Access Manager', 'Billing', and 'Support'. The main content area is titled 'Create a Project' and includes a breadcrumb 'ESCARLETH'S ORG - 2021-05-08 > PROJECTS'. Below the title, there are two tabs: 'Name Your Project' (active) and 'Add Members'. A 'Next' button is located to the right of these tabs. Under the 'Name Your Project' tab, there's a heading 'Name Your Project' followed by the instruction 'Project names have to be unique within the organization (and other restrictions)'. Below this is a text input field labeled 'Project Name'. At the bottom right, there are 'Cancel' and 'Next' buttons.

- Crear un nuevo Cluster



- Crear Usuario para acceso a la base de datos

ESCARLETH'S ORG - 2021-05-08 > PRACTICA

Database Access

Database Users

Custom Roles

+ ADD NEW DATABASE USER

User Name	Authentication Method	MongoDB Roles	Resources	Actions
 andrea	SCRAM	readWriteAnyDatabase@admin	All Resources	<div><div>EDIT</div><div>DELETE</div></div>

- Crear el Acceso a la red para este proyecto se permite cualquier conexión

ESCARLETH'S ORG - 2021-05-08 > PRACTICA

Network Access

IP Access List

Peering

Private Endpoint

+ ADD IP ADDRESS

You will only be able to connect to your cluster from the following list of IP Addresses:

IP Address	Comment	Status	Actions
0.0.0.0/0 (includes your current IP address)		<div><div></div>Active</div>	<div><div>EDIT</div><div>DELETE</div></div>

- Se creó la Base de Datos y el Modelo Estudiante

ESCARLETH'S ORG - 2021-05-08 > PRACTICA > CLUSTERS

Cluster0 VERSION 4.4.6 REGION AWS N. Virginia (us-east-1)

Overview Real Time Metrics **Collections** Search Profiler Performance Advisor Online Archive Command Line Tools

DATABASES: 1 COLLECTIONS: 1 VISUALIZE YOUR DATA REFRESH

Practica1

Estudiante

Practica1.Estudiante

COLLECTION SIZE: 831B TOTAL DOCUMENTS: 4 INDEXES TOTAL SIZE: 36KB

Find Indexes Schema Anti-Patterns Aggregation Search Indexes

FILTER {"filter":"example"} Find Reset

QUERY RESULTS 1-4 OF 4

```

{
  "_id": "ObjectId(\"60be7c797c2629ee6d6b4423\")",
  "DPI": "2301603280101",
  "CARNE": "201503378",
  "NOMBRES": "Prueba",
  "APELLIDO": "Prueba",
  "NACIONALIDAD": "guatemalteco",
  "SEXO": "MASCULINO",
  "IDENTIDADRACIAL": "ladino",
  "FACULTAD": "INGENIERIA",
  "CARRERA": "SISTEMAS"
}

{
  "_id": "ObjectId(\"60bee250055094719c01a88\")",
  "DPI": "540546",
  "CARNE": "5505",
  "NOMBRES": "fgfdgd",
  "APELLIDO": "fgfdgd",
  "NACIONALIDAD": "gfdg",
  "SEXO": "gfdg",
  "FACULTAD": "INGENIERIA",
  "CARRERA": "SISTEMAS"
}

```

- Luego se Generó la URL para la conexión

×

Connect to Cluster0

✓ Setup connection security

✓ Choose a connection method

Connect

1 Select your driver and version

DRIVER

VERSION

Node.js

3.6 or later

2 Add your connection string into your application code

☐ Include full driver code example

mongodb+srv://andrea:<password>@cluster0.lg3fq.mongodb.net/myFirstDatabase?retryWrites=true&w=majority

Replace **<password>** with the password for the **andrea** user. Replace **myFirstDatabase** with the name of the database that connections will use by default. Ensure any option params are [URL encoded](#).

Having trouble connecting? [View our troubleshooting documentation](#)

Go Back

Close

NodeJS: Node es una tecnología bastante potente que se adapta bien para trabajar con Mongo y Angular además es una herramienta bastante sencilla de utilizar.

Pasos para configuración:

- npm init para crear nuestro package.json
- creamos el archivo server.js
- Instalamos los módulos necesarios
- npm install express --save
- npm install cors --save
- npm install mongoose --save
- creamos nuestra conexión con Mongo

```
require('dotenv').config();
const express = require('express');
const cors = require('cors');
const mongoose = require('mongoose');
const Estudiantes = require('./models');
let port = 3000;
const app = express();

app.use(cors());
app.use(express.json());

const mongodbURL = `mongodb+srv://andrea:1364@cluster0.lg3fq.mongodb.net/Practical?retryWrites=true&w=m`;
mongoose.connect(mongodbURL, { useNewUrlParser: true, useUnifiedTopology: true })
  .then(db => console.log('DB is connected to', db.connection.host))
  .catch(err => {
    console.log("ERROR AL CONECTAR LA DB!!!!");
    console.error(err);
  });
```

- Creamos Nuestro modelo

```
1  const mongoose = require('mongoose');
2
3  const SchemaEstudiante = new mongoose.Schema({
4    _id: mongoose.Schema.Types.ObjectId,
5    DPI: String,
6    CARNE: Number,
7    NOMBRES: String,
8    APELLIDO: String,
9    NACIONALIDAD: String,
10   SEXO: String,
11   IDENTIDADRACIAL: String,
12   FACULTAD: String,
13   CARRERA: String
14  });
15
16
17  module.exports = new mongoose.model('Estudiante', SchemaEstudiante, 'Estudiante');
```

- Se agregan los GET,POST,PUT dependiendo de lo que se quiera realizar

```
1 app.get('/estudiante', async(req, res) => {
2   const response = await Estudiantes.find({});
3   res.json({ listaUsuarios: response });
4 });
5
6 app.post('/estudiante', async(req, res) => {
7   let data = req.body;
8
9   const response = await Estudiantes.find({ DPI: data.DPI })
10
11   if (response.length == 0) {
12     data._id = new mongoose.Types.ObjectId();
13     let nuevoEstudiante = new Estudiantes(data);
14     nuevoEstudiante.save()
15       .then(result => {
16         res.json({
17           status: true,
18           msg: "Estudiante guardado con exito",
19           _id: result._id
20         });
21       })
22       .catch(err => {
23         console.log(err)
24         res.json({
25           status: false,
26           msg: "Error el estudiante no se guardo"
27         })
28       })
29   } else {
30     res.json({
31       status: false,
32       msg: "El estudiante ya existe"
33     });
34   }
35 });
```

- Se agrega donde escuchara el Puerto

```
1 app.listen(port, () => {
2   console.log(`Server listening at http://localhost:\${port}`);
3 });
```

- Y por último se corre el servidor con el comando NPM start

Angular: Es un framework que ofrece muchas funcionalidades fáciles de implementar, es bastante rápido de desarrollar y para un proyecto corto se adapta bastante bien.

Pasos para configuración:

- Iniciar proyecto ng new Practica
- Una vez iniciado procedemos instalar Bootstrap y JQuery npm install bootstrap, npm install JQuery
- Luego procedemos a crear nuestro primer componente que será nuestra barra en el header con el comando ng generate component top-bar
- Aquí creamos nuestra barra de navegación

```
1 <section class="ftco-section">
2
3   <div class="container">
4     <nav class="navbar navbar-expand-lg ftco_navbar ftco-navbar-light" id="ftco-navbar">
5       <div class="container">
6         <a class="navbar-brand" routerLink="/Bienvenida">Grupo 14 AYD1</a>
7         <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#ftco-
8         <span class="fa fa-bars"></span> Menu
9       </button>
10      <div class="collapse navbar-collapse" id="ftco-nav">
11        <ul class="navbar-nav ml-auto">
12          <li class="nav-item"><a class="nav-link" routerLink="/Bienvenida" routerLinkAct
13          <li class="nav-item">
14            <a class="nav-link" routerLink="DatosEstudiante">Datos</a>
15          </li>
16          <li class="nav-item"><a routerLink="/AgregarEstudiante" class="nav-link">Agre
17        </ul>
18      </div>
19    </div>
20  </nav>
21
22 </div>
23
24 </section>
```

- Una Vez hecho esto agregamos la etiqueta a nuestro app.component

```
<> app.component.html x
src > app > <> app.component.html > app-top-bar
1 <app-top-bar>
2 </app-top-bar>
```

- Luego debemos de configurar nuestro Router este lo encontramos en app-routing.module.ts este archivo se crea automáticamente cuando se crea el proyecto, aquí se agregan las rutas que tendrá nuestra página web.

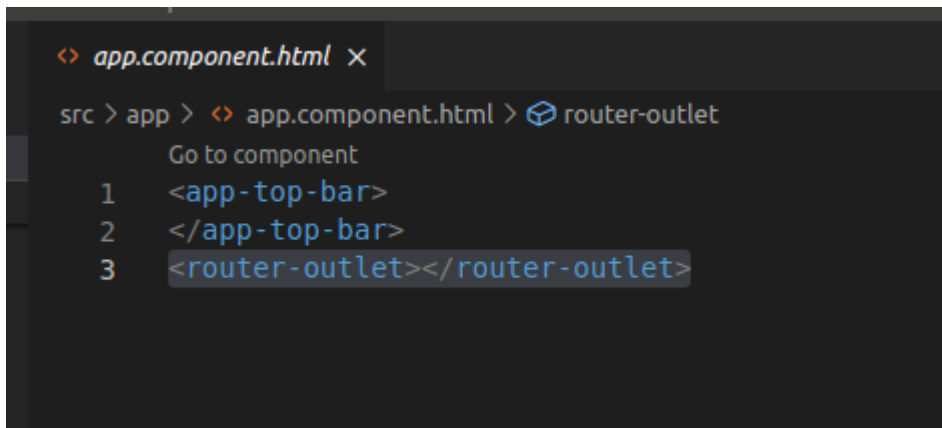
```

import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { DatosComponent } from '../datos/datos.component';
import { CommonModule } from '@angular/common';
import { BrowserModule } from '@angular/platform-browser';
import { BienvenidaComponent } from '../bienvenida/bienvenida.component';
import { AgregarestudianteComponent } from '../agregarestudiante/agregarestudiante.component';

const routes: Routes = [
  { path: "DatosEstudiante", component: DatosComponent},
  { path: "Bienvenida", component: BienvenidaComponent},
  { path: "AgregarEstudiante", component: AgregarestudianteComponent},
];
@NgModule({
  imports: [
    RouterModule.forRoot(routes)
  ],
  exports: [RouterModule]
})
export class AppRoutingModule { }

```

- Se debe de agregar la etiqueta <router-outlet></router-outlet> para que funcione el ruteo



```

app.component.html
src > app > app.component.html > router-outlet
Go to component
1 <app-top-bar>
2 </app-top-bar>
3 <router-outlet></router-outlet>

```

- Para conectar con nuestro servidor debemos de crear un servicio con el comando ng generate service estudiante
- debemos de importar httpcliente y declararlo en el constructor para luego implementar nuestro método GET, POST


```

TS estudiantes.service.ts X
src > app > TS estudiantes.service.ts > ...
1  import { HttpClient, HttpResponse, HttpHeaders, HttpClientModule } from '@angular/common/http';
2  import { Injectable } from '@angular/core';
3
4  @Injectable({
5    providedIn: 'root'
6  })
7  export class EstudiantesService {
8
9    constructor(private httpClient: HttpClient) { }
10
11    addEstudiante(estudiante:any){
12      return new Promise<void>((resolve, reject) => {
13        this.httpClient.post('http://localhost:3000' + '/estudiante', estudiante).subscribe((resp: any) => {
14
15          resolve();
16          console.log(resp);
17          if(resp["status"] == true){ alert("Estudiante agregado con exito");}
18          else{
19            alert("El Estudiante ya existe DPI repetido");
20          }
21        });
22      });
23    }
24
25  }
26

```

- Luego en nuestro componente importamos el servicio y lo declaramos en el constructor

```

TS agregarestudiante.component.ts X
src > app > agregarestudiante > TS agregarestudiante.component.ts > ...
1  import { Component, OnInit } from '@angular/core';
2  import {EstudiantesService} from "../estudiantes.service";
3  import { Estudiante } from '../Modelos/Estudiante';
4  @Component({
5    selector: 'app-agregarestudiante',
6    templateUrl: './agregarestudiante.component.html',
7    styleUrls: ['./agregarestudiante.component.css']
8  })
9  export class AgregarestudianteComponent implements OnInit {
10
11    constructor(private servicioesdutiante:EstudiantesService) { }
12
13    ngOnInit(): void {
14    }
15
16    async crearEstudiante(dpi: string,carne:any,nombres:string,apellido:string,nacionalidad:string,s
17    const estu =new  Estudiante(dpi,carne,nombres,apellido,nacionalidad,sexo,identidad,facultad,carr
18    await this.servicioesdutiante.addEstudiante(estu);
19  }
20
21  }
22

```

- Por último se llama al método para conectar con nuestro servidor

```
agregarestudiante.component.html x
src > app > agregarestudiante > agregarestudiante.component.html > div.container
29
30 label>
31 ad" class="form-control" id="txtidendidad" placeholder="IDENTIDAD RACIAL" required #txtidendidad>
32
33
34 </label>
35 id" class="form-control" id="txtfacultad" placeholder="FACULTAD" required #txtfacultad>
36
37
38 /label>
39 " class="form-control" id="txtcarrera" placeholder="CARRERA" required #txtcarrera>
40
41
42
43 -primary btn-block" (click)="crearEstudiante(txtdpi.value,txtcarne.value,txtnombres.value,txtapellido.value,txtnacionalidad.value,tx
44
45
46
47
48
49
```

- Dependiendo de lo que se quiera realizar se implementa el método en el servicio para conectar con nuestro servidor y luego se manda a llamar desde algún componente que hayamos creado.