

Universidad Autónoma de San Luis Potosí



Facultad de Ingeniería

Área de Ciencias de la Computación

Aplicaciones Web Escalables

Llamas De la Torre María Jocelyn

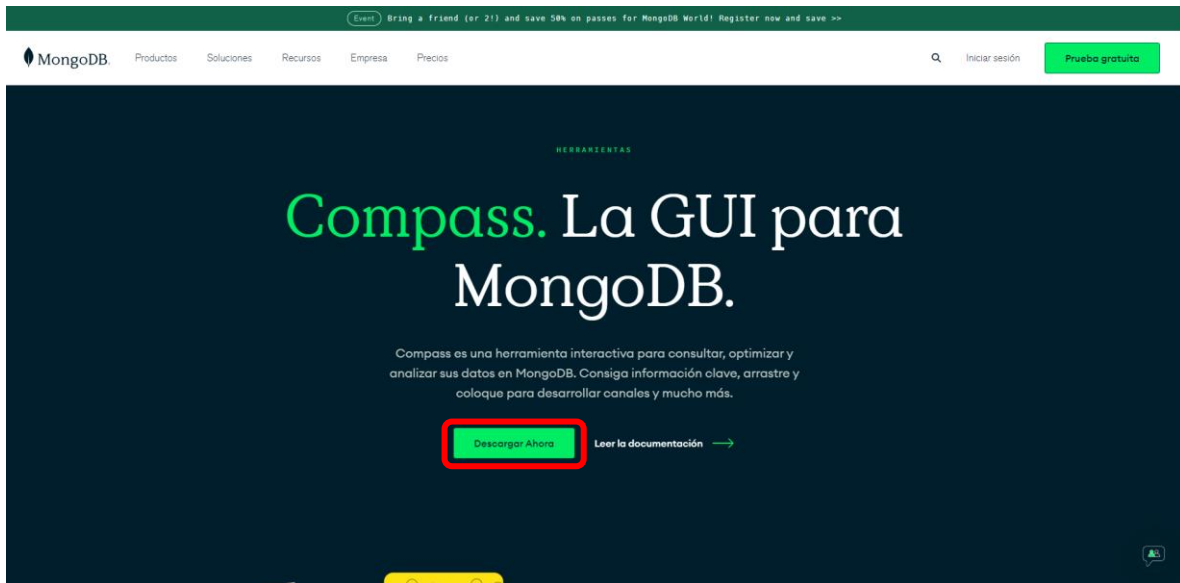
Moctezuma Estrada Jorge Eduardo

Manual de Programador

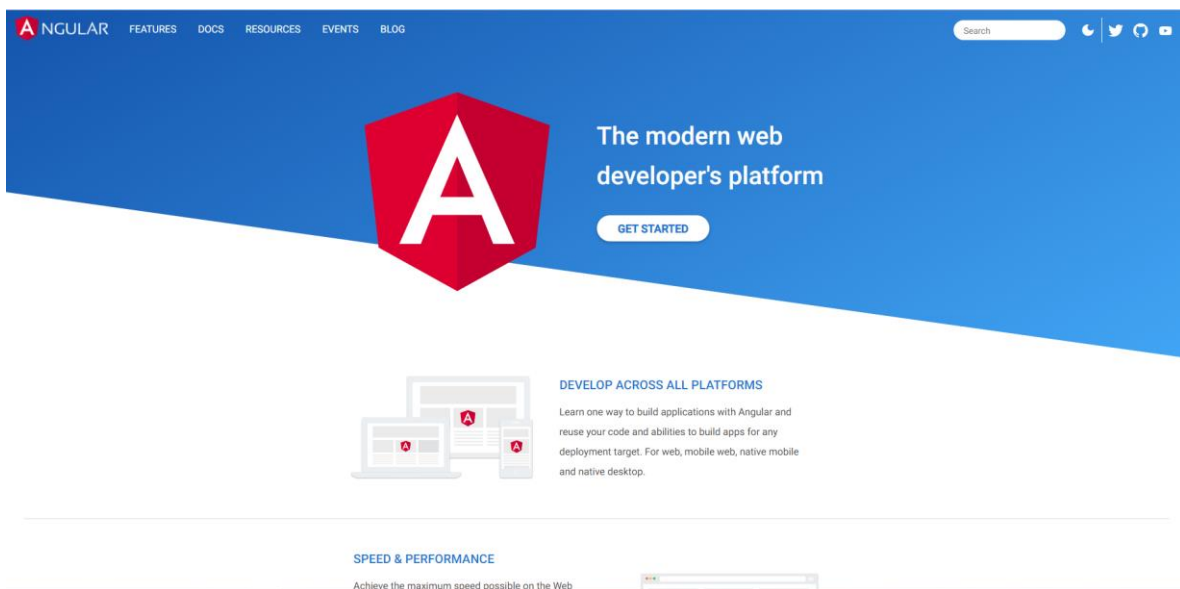


Descargas

Mongo Compass: <https://www.mongodb.com/es/products/compass>



Angular: <https://angular.io/>



Inicios de sesión

MongoDB: <https://www.mongodb.com/es>

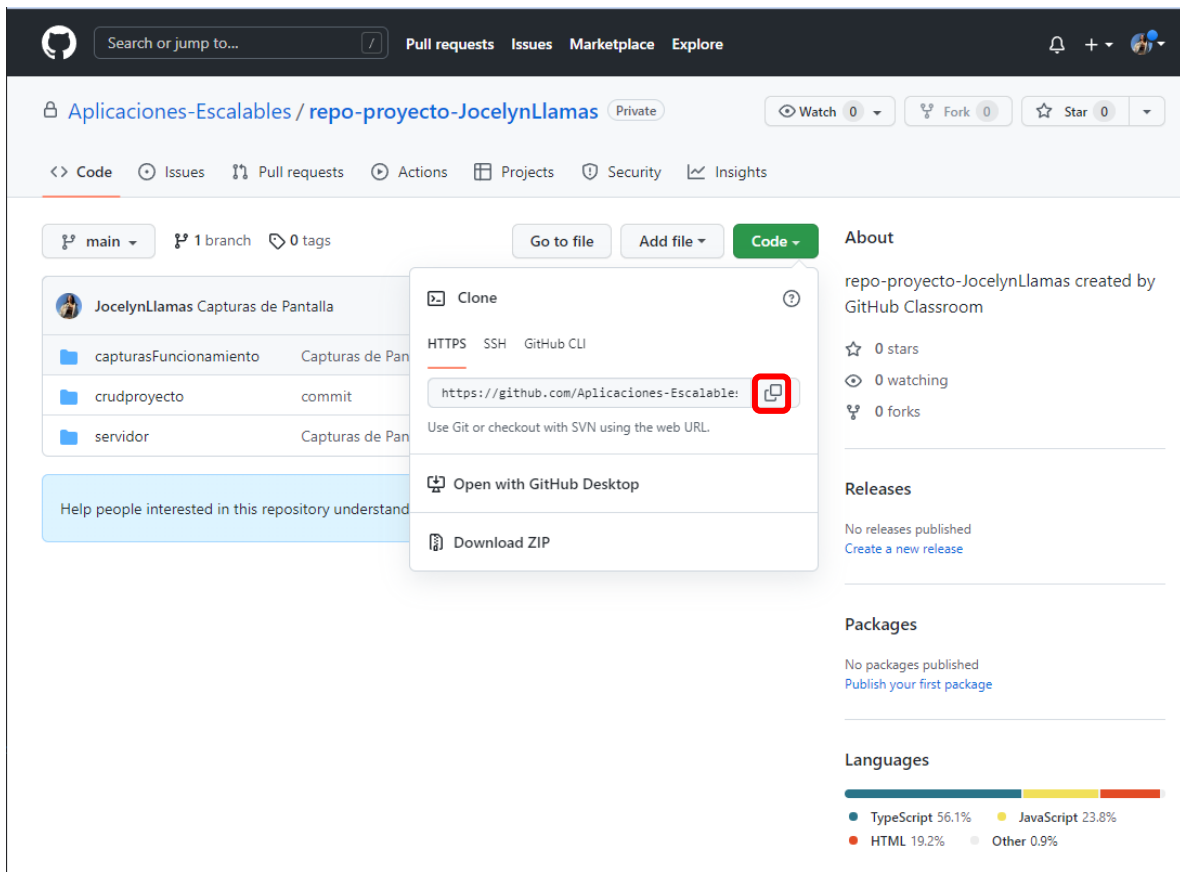


Montaje del Sistema

Una vez instalado MongoCompass e iniciado sesión en MongoDB, procedemos a clonar el repositorio en el que se encuentra el proyecto:

<https://github.com/Aplicaciones-Escalables/repo-proyecto-JocelynLlamas.git>

Copiamos el link del código y nos dirigimos a una terminal, ya sea CMD, PowerShell o GitBash, para clonar el repositorio a nuestra carpeta.



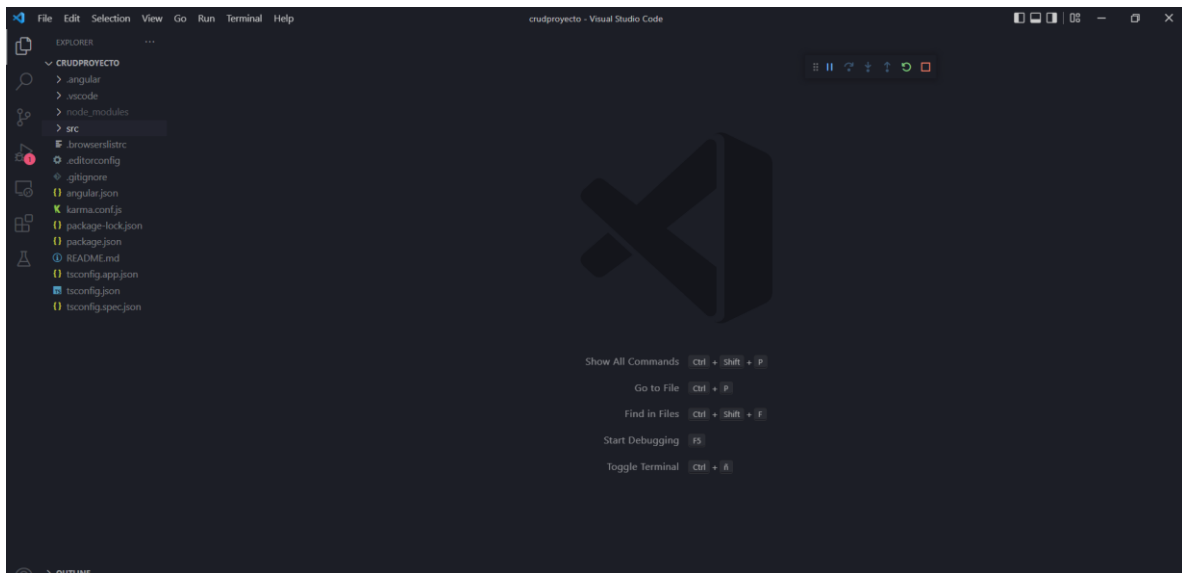
```
MINGW64/c/Users/Jocelyn Llamas/PROYECTO

Jocelyn Llamas@DESKTOP-6MPAC08 MINGW64 ~
$ cd PROYECTO/

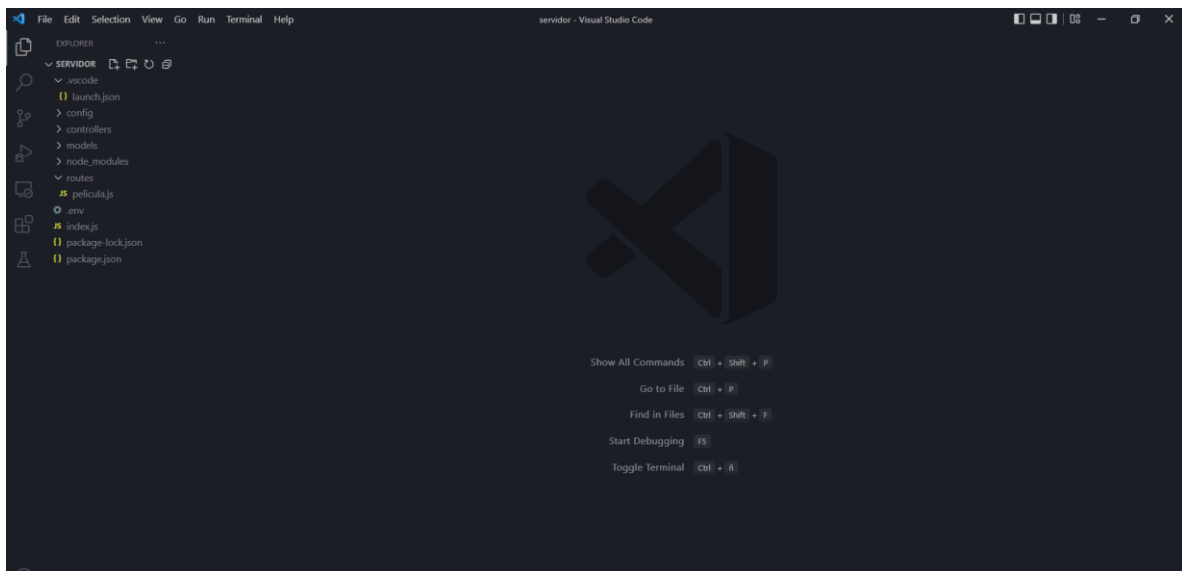
Jocelyn Llamas@DESKTOP-6MPAC08 MINGW64 ~/PROYECTO
$ git clone https://github.com/Aplicaciones-Escalables/repo-proyecto-JocelynLlamas.git
Cloning into 'repo-proyecto-JocelynLlamas'...
remote: Enumerating objects: 2577, done.
remote: Counting objects: 100% (2577/2577), done.
remote: Compressing objects: 100% (2213/2213), done.
remote: Total 2577 (delta 266), reused 2576 (delta 265), pack-reused 0
Receiving objects: 100% (2577/2577), 4.19 MiB | 4.39 MiB/s, done.
Resolving deltas: 100% (266/266), done.
Updating files: 100% (2327/2327), done.

Jocelyn Llamas@DESKTOP-6MPAC08 MINGW64 ~/PROYECTO
$ |
```

Una vez clonado el repositorio, nos dirigimos a Visual Studio Code y abrimos la carpeta de "crudproyecto".

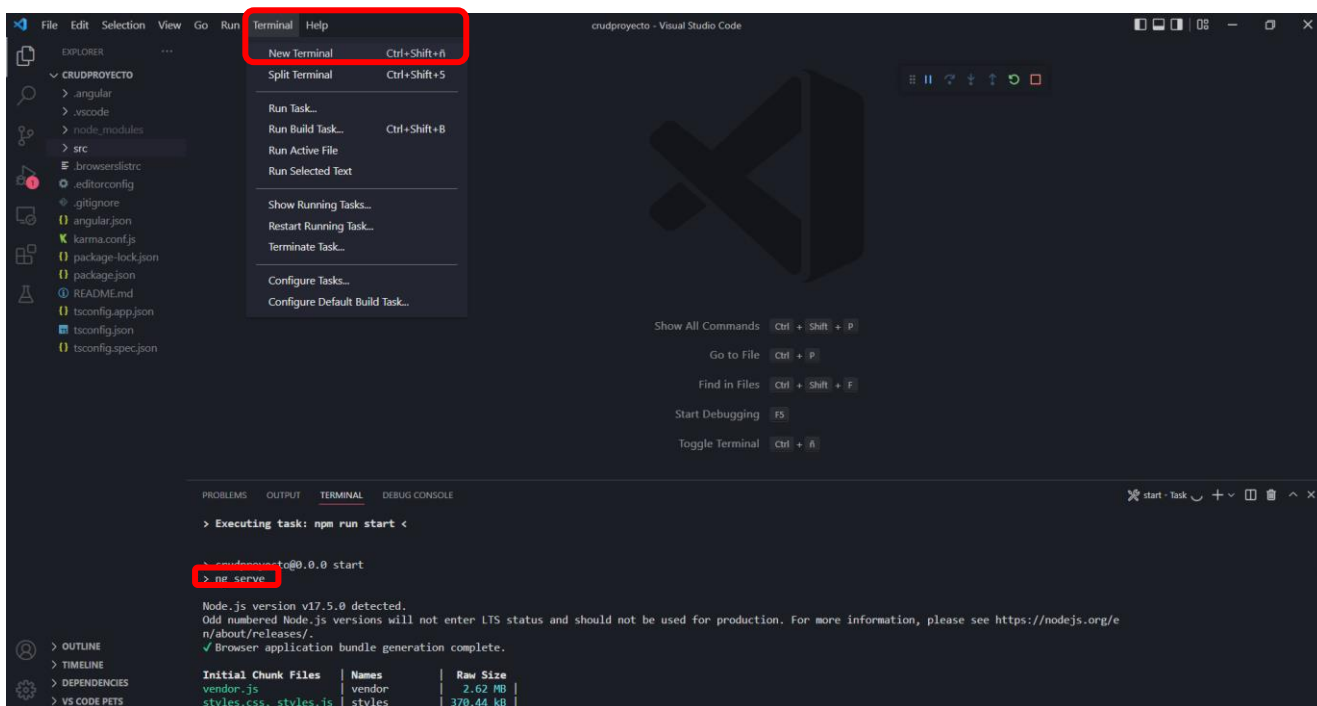


A continuación, abrimos otra ventana de Visual Studio Code y abrimos la carpeta "servidor".



Ejecución

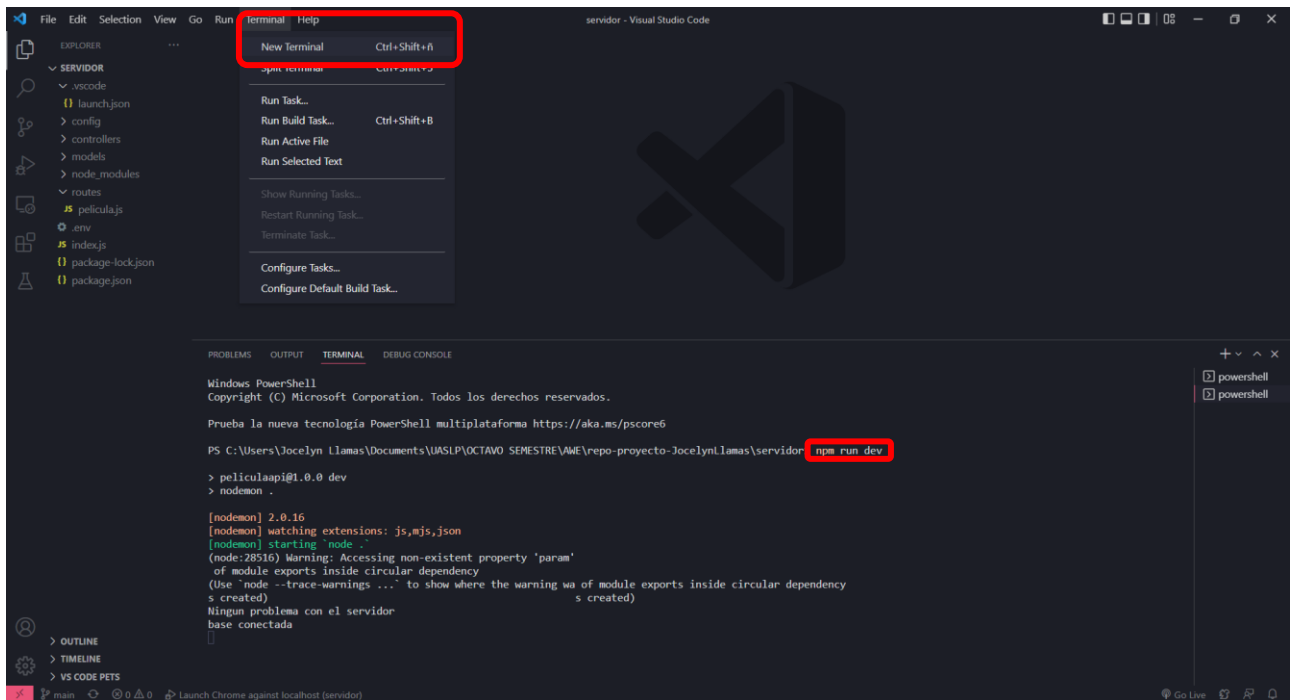
Para visualizar el Proyecto abriremos una nueva terminal en la carpeta de "crudproyecto" y a continuación pondremos el comando `ng serve` o simplemente daremos clic a la tecla "F5".



Nos aparecerá una nueva ventana en el navegador con la interfaz del CRUD.

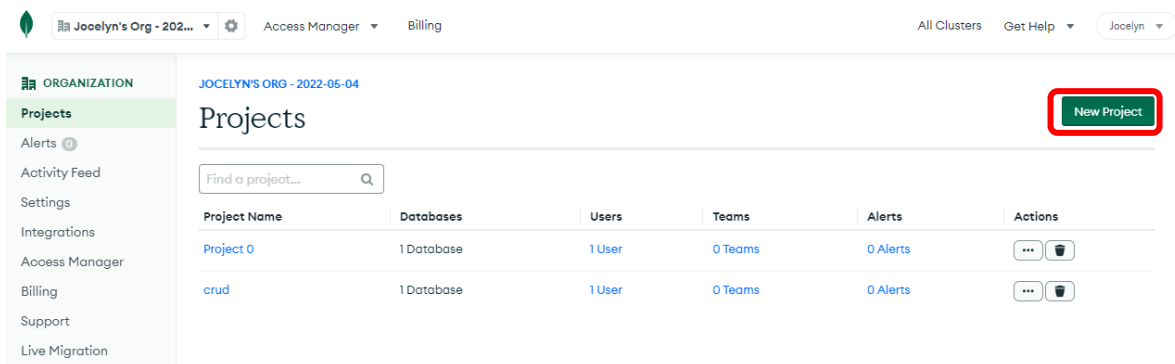


A continuación, nos dirigimos a la ventana donde abrimos la carpeta "servidor", creamos una nueva terminal y a continuación pondremos el comando `npm run dev` o simplemente daremos clic a la tecla "F5".



Desarrollador

Para poder visualizar la base de datos inicia sesión en MongoDB y crea un nuevo proyecto.



A continuación, haz clic en Database y crea un Cluster de tipo Shared.

The screenshot shows the MongoDB Atlas console interface. The top navigation bar includes the user profile 'Jocelyn's Org - 202...', 'Access Manager', 'Billing', 'All Clusters', 'Get Help', and the user name 'Jocelyn'. The left sidebar contains navigation links: 'crud', 'Atlas', 'Realm', and 'Charts'. The main content area is titled 'Database Deployments' and shows a 'Cluster0' deployment. A red box highlights the 'Database' link in the left sidebar. Another red box highlights the '+ Create' button in the top right corner of the 'Database Deployments' section. Below the deployment details, there are charts for 'Connections', 'In/Out B/s', and 'Data Size'. A table at the bottom lists deployment details: VERSION (5.0.8), REGION (AWS / N. Virginia (us-east-1)), CLUSTER TIER (M0 Sandbox (General)), TYPE (Replica Set - 3 nodes), BACKUPS (Inactive), LINKED REALM APP (None Linked), and ATLAS SEARCH (Create Index). Below this, there is a section titled 'Create a Shared Cluster' with three tabs: 'PREVIEW', 'Serverless', and 'Shared'. The 'Shared' tab is selected and highlighted with a red box. Below the tabs, there is a message about the sandbox environment. The 'Cloud Provider & Region' section shows 'AWS, N. Virginia (us-east-1)' selected. Below this, there are buttons for 'aws', 'Google Cloud', and 'Azure'. A table of regions is shown, with 'N. Virginia (us-east-1)' selected and highlighted with a red box. At the bottom, there is a 'FREE' badge and a 'Create Cluster' button.

Database Deployments

Cluster0

Connections 14.0

In 45.0 B/s

Out 615.8 B/s

Data Size 4.0 KB

Enhance Your Experience

Upgrade

VERSION	REGION	CLUSTER TIER	TYPE	BACKUPS	LINKED REALM APP	ATLAS SEARCH
5.0.8	AWS / N. Virginia (us-east-1)	M0 Sandbox (General)	Replica Set - 3 nodes	Inactive	None Linked	Create Index

CLUSTERS > CREATE A SHARED CLUSTER

Create a Shared Cluster

PREVIEW Serverless Dedicated Shared

For learning and exploring MongoDB in a sandbox environment. Basic configuration controls.

No credit card required to start. Upgrade to dedicated clusters for full functionality. Explore with sample datasets. Limit of one free cluster per project.

Cloud Provider & Region

AWS, N. Virginia (us-east-1)

aws Google Cloud Azure

★ Recommended region ⓘ ⚠ Dedicated tier region ⓘ

NORTH AMERICA	EUROPE	AUSTRALIA
N. Virginia (us-east-1) ★	Ireland (eu-west-1) ★	Sydney (ap-southeast-2) ★
Oregon (us-west-2) ★	Paris (eu-west-3) ★	ASIA
Ohio (us-east-2) ★ ⚠	Stockholm (eu-north-1) ★	Seoul (ap-northeast-2)

FREE

Free forever! Your M0 cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.

Cancel Create Cluster

Una vez creado el Cluster, haz clic en Database Access y crea un usuario y contraseña.

JOCELYN'S ORG - 2022-05-04 > CRUD

Database Access

Database Users Custom Roles

+ ADD NEW DATABASE USER

User Name	Authentication Method	MongoDB Roles	Resources	Actions
jocelyn	SCRAM	readWriteAnyDatabase@admin	All Resources	EDIT DELETE

Haz clic en Network Access e ingresa la dirección IP 0.0.0.0/0.

JOCELYN'S ORG - 2022-05-04 > CRUD

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)	desarrollo	Active	EDIT DELETE

Ahora regresa a la sección de Database y haz clic en Connect.

JOCELYN'S ORG - 2022-05-04 > CRUD

Database Deployments

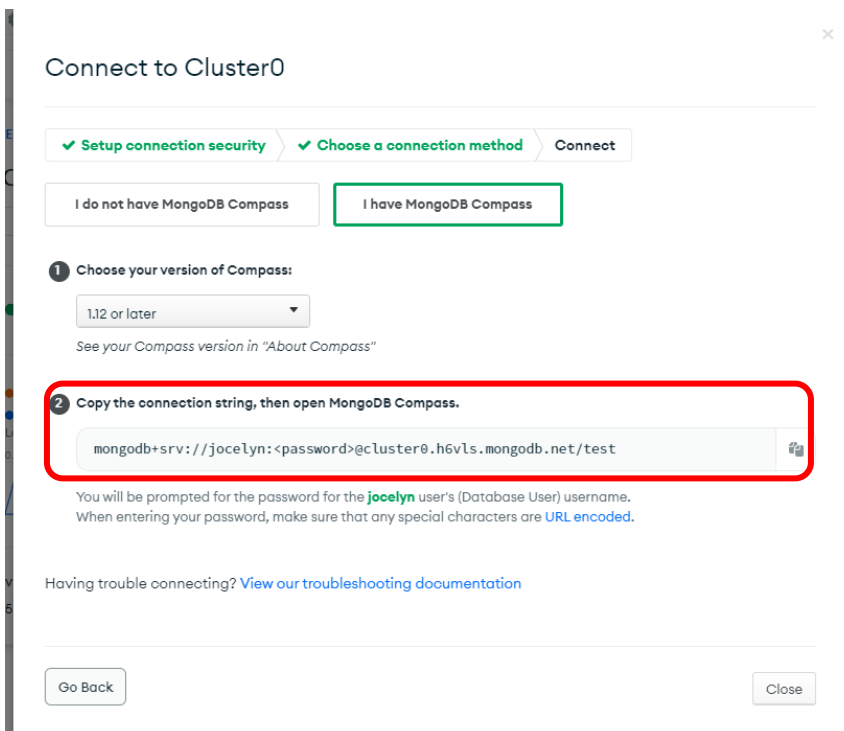
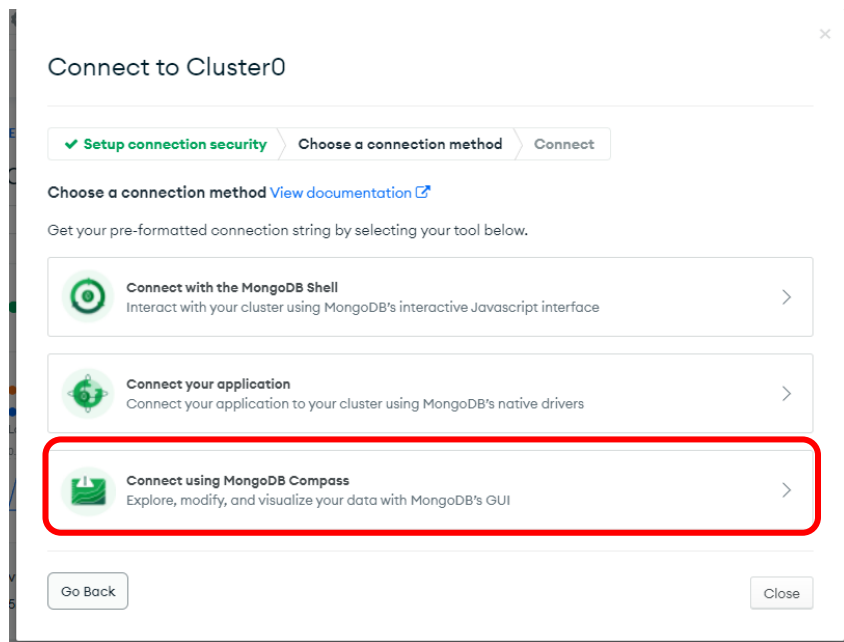
Cluster0 Connect View Monitoring Browse Collections

FREE SHA

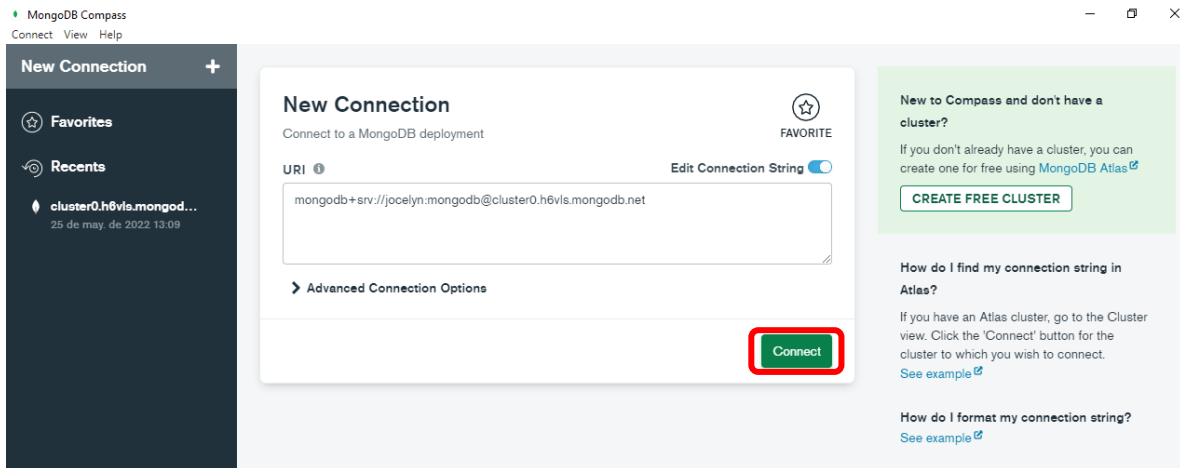
Connections 14.0 In 46.0 B/s Out 622.7 B/s Data Size 36.0 KB

VERSION	REGION	CLUSTER TIER	TYPE	BACKUPS	LINKED REALM APP	ATLAS SEARCH
5.0.8	AWS / N. Virginia (us-east-1)	M0 Sandbox (General)	Replica Set - 3 nodes	Inactive	None Linked	Create Index

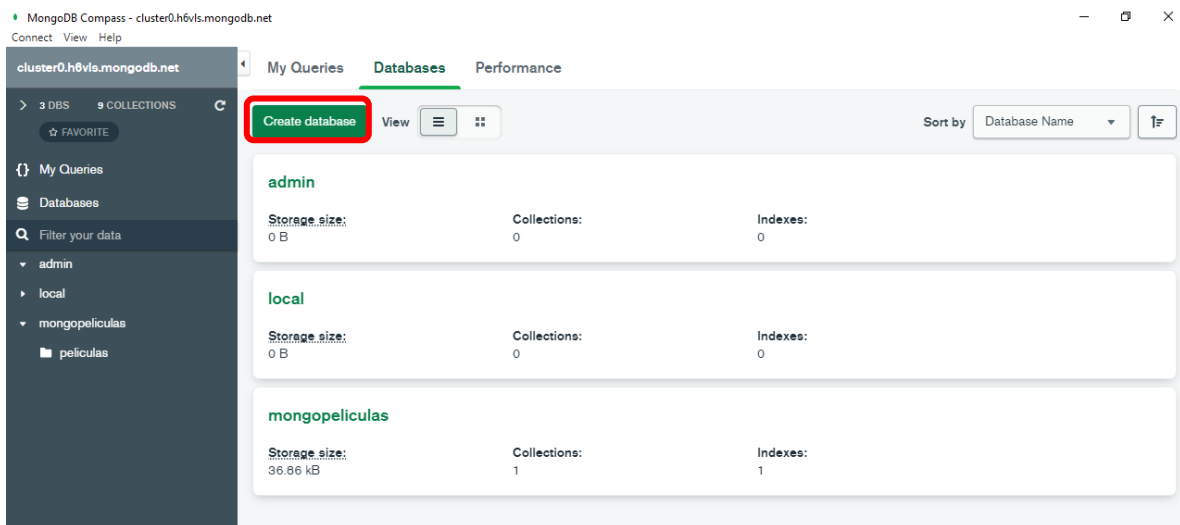
Selecciona la opción de Connect using MongoDB Compass y copia la dirección de conexión.



Abre MongoCompass y pega la dirección en el campo de URI, quita la parte de test de la url y en vez de <password> ingresa la contraseña que pusiste en Database Access de MongoDB, después haz clic en Connect.



Crea una base de datos con el nombre mongopeliculas y con una colección llamada películas.



×

Create Database

Database Name

mongopeliculas

Collection Name

películas

☐ Capped Collection

Fixed-size collections that support high-throughput operations that insert and retrieve documents based on insertion order. [Learn More](#)

☐ Use Custom Collation

Collation allows users to specify language-specific rules for string comparison, such as rules for lettercase and accent marks. [Learn More](#)

☐ Time-Series

Time-series collections efficiently store sequences of measurements over a period of time. [Learn More](#)

Cancel

Create Database

Una vez hecho esto podras observar los cambios que se hagan en la base de datos desde la colección películas.

Cada vez que hagas un cambio para visualizarlo haz clic en Refresh.

MongoDB Compass - cluster0.h6vls.mongodb.net/mongopeliculas.peliculas

Connect View Help

cluster0.h6vls.mongodb.net

> 3 DBS 9 COLLECTIONS

☆ FAVORITE

My Queries

Databases

Filter your data

admin

local

mongopeliculas

- películas

Documents

mongopeliculas.pe...

+

mongopeliculas.peliculas

0 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Explain Plan Indexes Validation

FILTER { field: 'value' }

OPTIONS FIND RESET ↺ ⋮

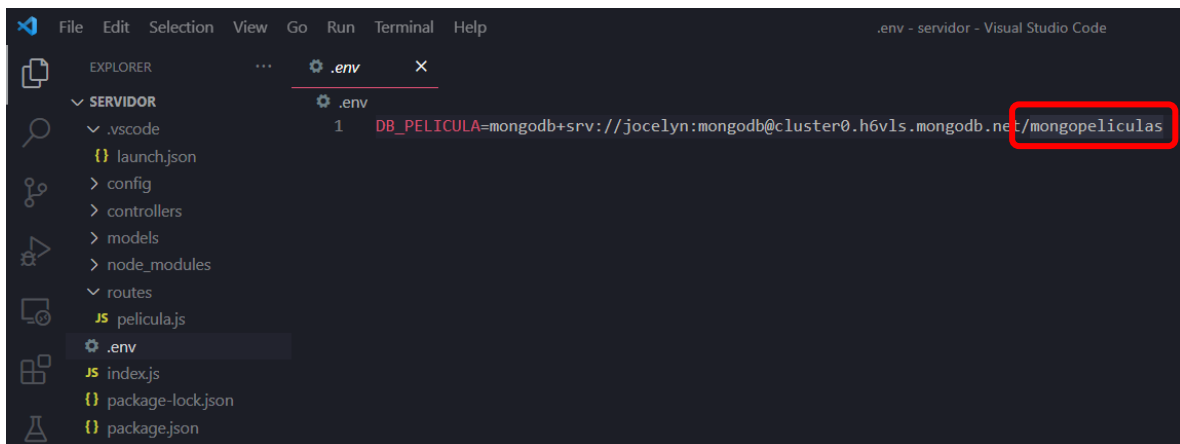
ADD DATA + VIEW [] []

Displaying documents 1 - 1 of 1 < > REFRESH

```
{
  "_id": ObjectId('628e74cc75b002ce4f95d8a9'),
  "nombre": "La llegada",
  "genero": "Acción",
  "director": "Denis Villeneuve",
  "anio": 2020,
  "fechaInicio": "2022-05-25T18:23:32.075+00:00",
  "_v": 0
}
```

Código

En la carpeta `servidor` se encuentra un archivo llamado `.env` el cual tiene la url del servidor de mongo, al final de este link se le agrega el nombre de la base de datos creada en Mongo Compass.

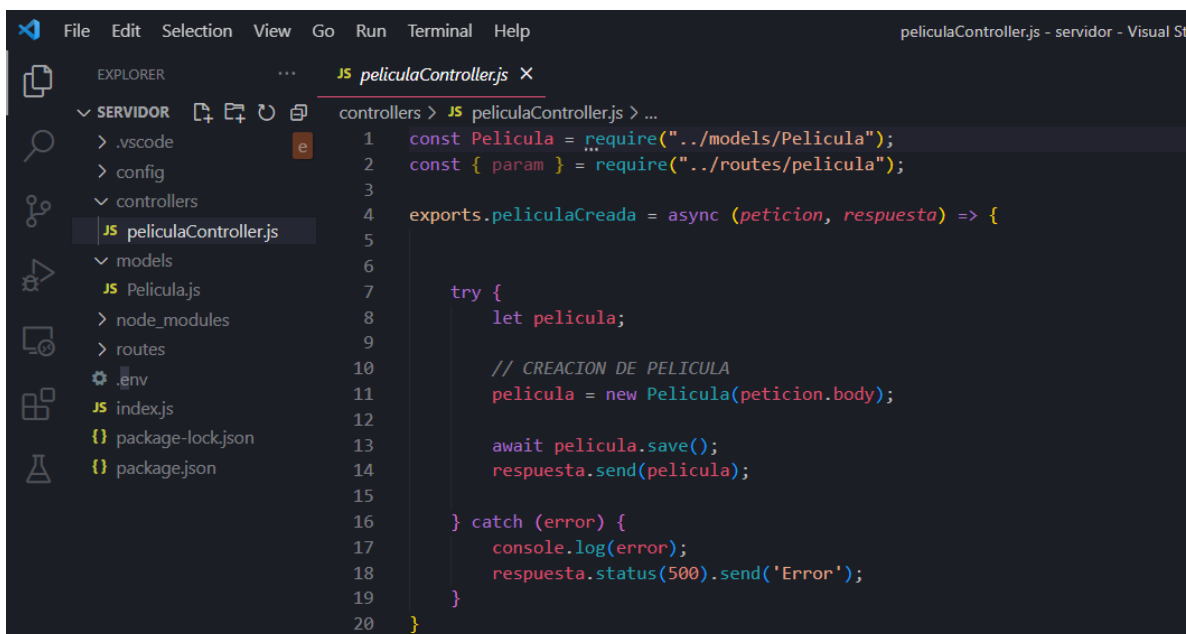


```
File Edit Selection View Go Run Terminal Help
.env - servidor - Visual Studio Code

EXPLORER
  SERVERIDOR
    .vscode
      launch.json
    config
    controllers
    models
    node_modules
    routes
      pelicula.js
    .env
    index.js
    package-lock.json
    package.json

.env
1 DB_PELICULA=mongodb+srv://jocelyn:mongodb@cluster0.h6vls.mongodb.net/mongopeliculas
```

En la carpeta `controllers` se encuentra el controlador que contiene todas las funciones necesarias para hacer las acciones de agregar, editar y eliminar una película de la base de datos.

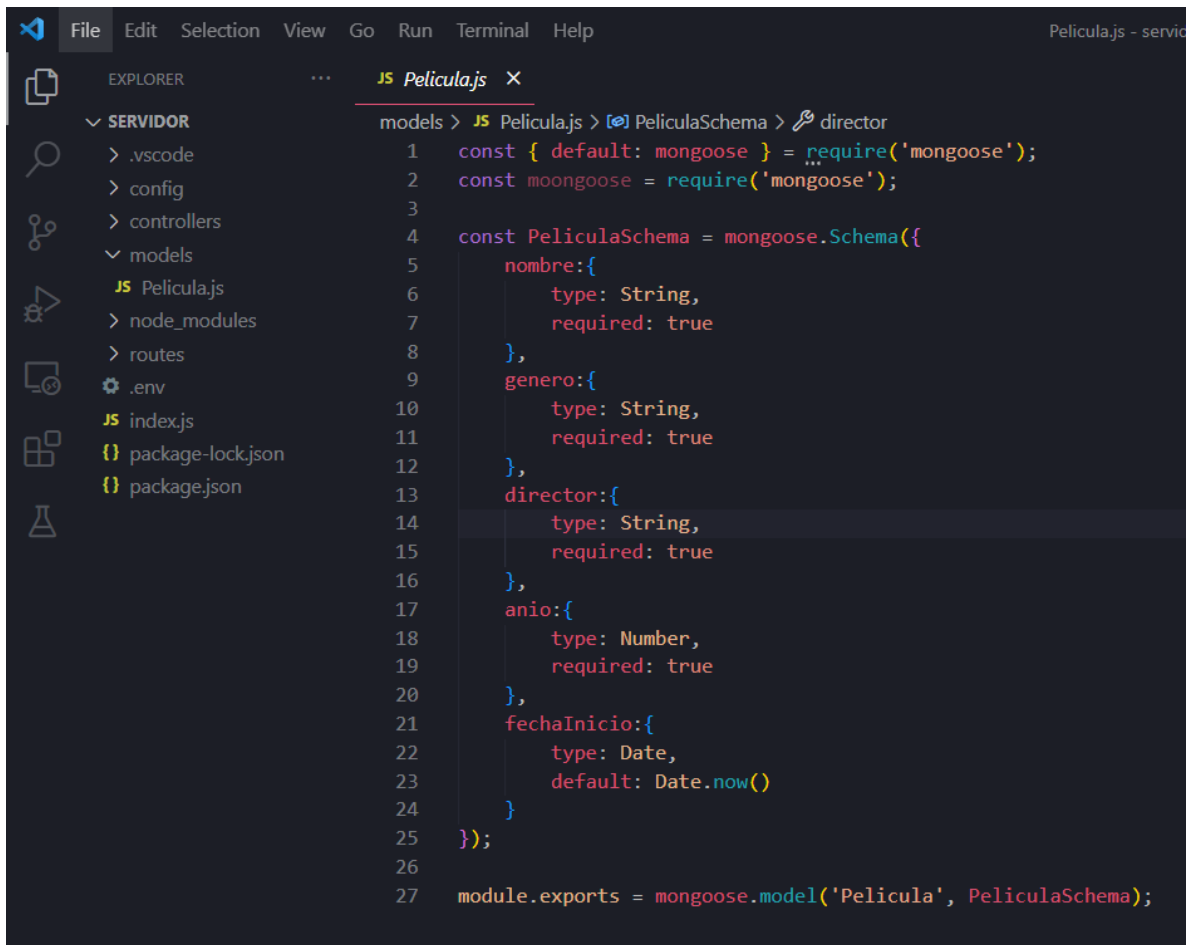


```
File Edit Selection View Go Run Terminal Help
peliculaController.js - servidor - Visual Studio Code

EXPLORER
  SERVERIDOR
    .vscode
    config
    controllers
      peliculaController.js
    models
      Pelicula.js
    node_modules
    routes
    .env
    index.js
    package-lock.json
    package.json

controllers > JS peliculaController.js > ...
1 const Pelicula = require("../models/Pelicula");
2 const { param } = require("../routes/pelicula");
3
4 exports.peliculaCreada = async (peticion, respuesta) => {
5
6   try {
7     let pelicula;
8
9     // CREACION DE PELICULA
10    pelicula = new Pelicula(peticion.body);
11
12    await pelicula.save();
13    respuesta.send(pelicula);
14
15  } catch (error) {
16    console.log(error);
17    respuesta.status(500).send('Error');
18  }
19 }
20 }
```

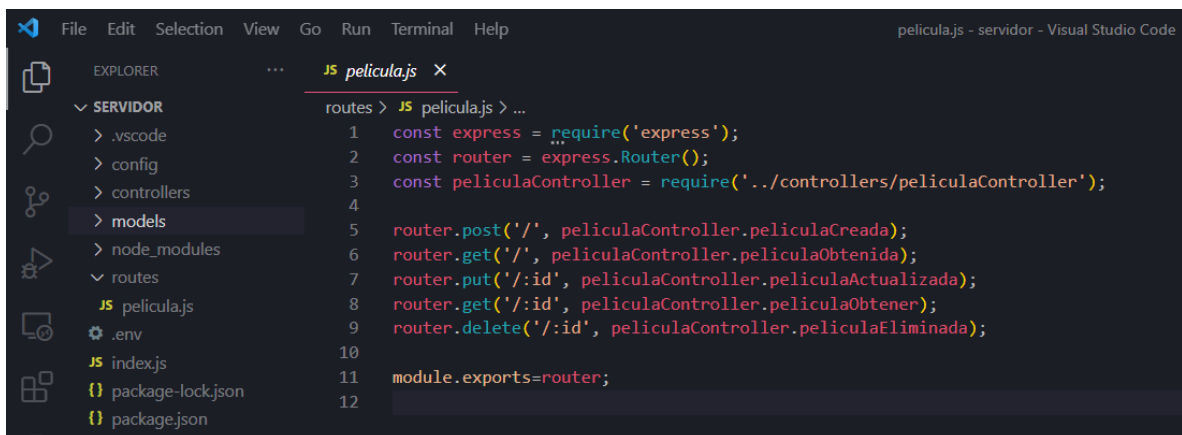
En la carpeta `models` se encuentra el modelo que contiene la declaración de los datos que se necesitan para llenar el crud.



The screenshot shows the Visual Studio Code interface with the Explorer panel on the left displaying the project structure. The 'models' directory is expanded, showing 'Película.js' as the active file. The main editor displays the content of 'Película.js', which defines a Mongoose schema for a movie. The code includes imports for 'mongoose', 'mongoose.Schema', and 'mongoose.model'. The schema 'PelículaSchema' is defined with fields: 'nombre' (String, required), 'genero' (String, required), 'director' (String, required), 'anio' (Number, required), and 'fechaInicio' (Date, default: Date.now()). The schema is then used to create a model 'Película'.

```
1 const { default: mongoose } = require('mongoose');
2 const mongoose = require('mongoose');
3
4 const PeliculaSchema = mongoose.Schema({
5   nombre:{
6     type: String,
7     required: true
8   },
9   genero:{
10    type: String,
11    required: true
12  },
13  director:{
14    type: String,
15    required: true
16  },
17  anio:{
18    type: Number,
19    required: true
20  },
21  fechaInicio:{
22    type: Date,
23    default: Date.now()
24  }
25 });
26
27 module.exports = mongoose.model('Película', PeliculaSchema);
```

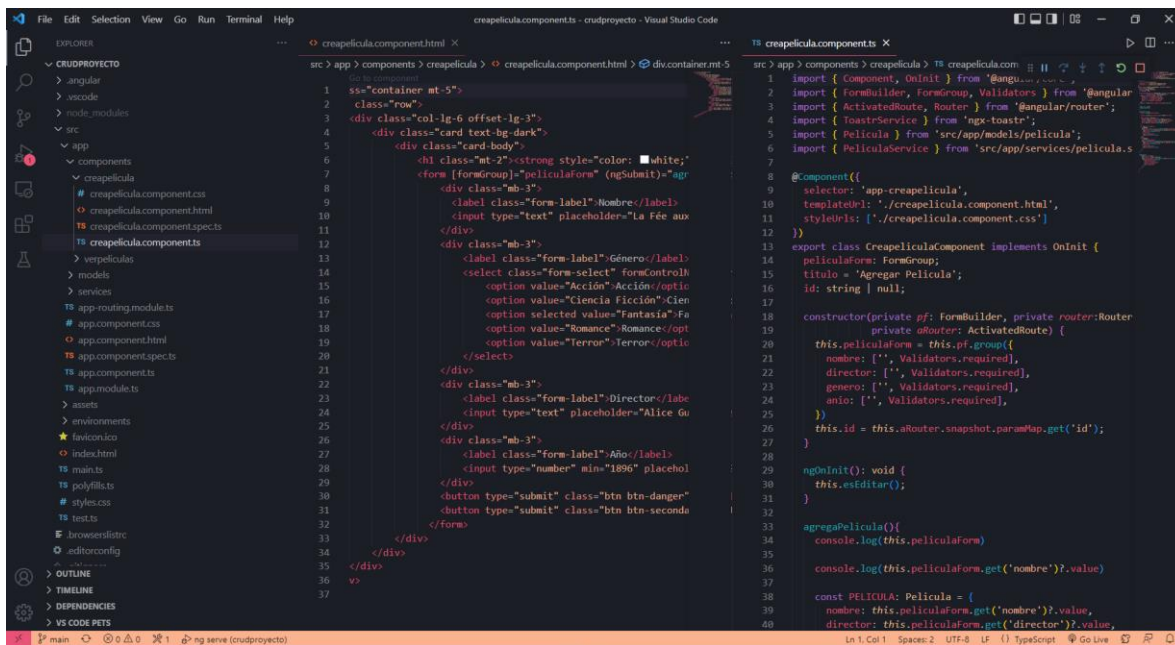
En la carpeta `routes` se encuentra el archivo que contiene las rutas que se usaran para cada caso de edición, agregar o eliminar un elemento recuperando su id.



The screenshot shows the Visual Studio Code interface with the Explorer panel on the left displaying the project structure. The 'routes' directory is expanded, showing 'pelicula.js' as the active file. The main editor displays the content of 'pelicula.js', which defines Express.js routes for movie operations. The code imports 'express', 'express.Router', and 'peliculaController'. It defines routes for creating, getting, updating, deleting, and obtaining movies. The routes are then exported as the 'router'.

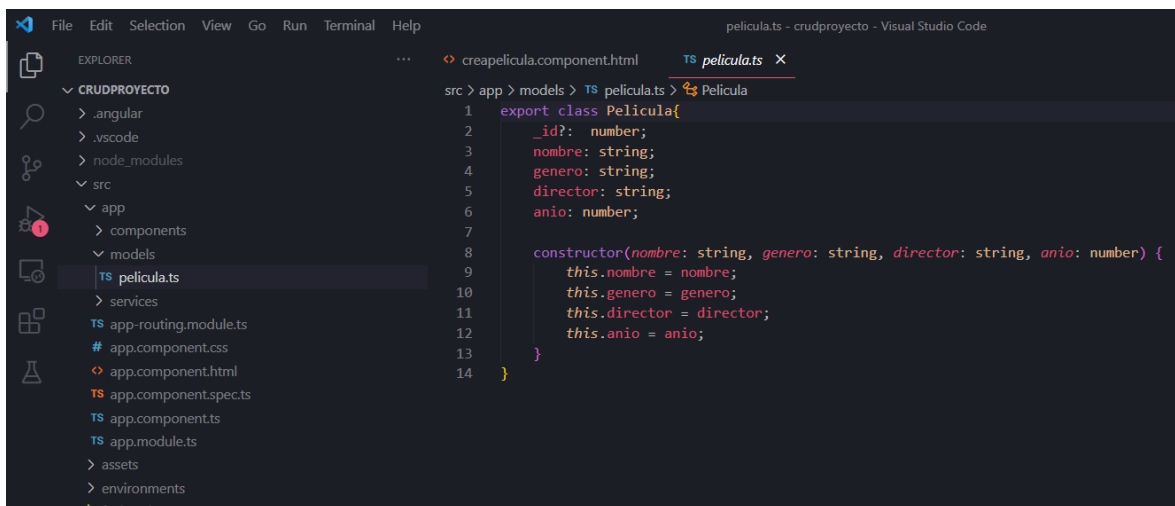
```
1 const express = require('express');
2 const router = express.Router();
3 const peliculaController = require('../controllers/peliculaController');
4
5 router.post('/', peliculaController.peliculaCreada);
6 router.get('/', peliculaController.peliculaObtenida);
7 router.put('/:id', peliculaController.peliculaActualizada);
8 router.get('/:id', peliculaController.peliculaObtener);
9 router.delete('/:id', peliculaController.peliculaEliminada);
10
11 module.exports=router;
12
```

En la carpeta crudproyecto en la ruta /src/app/components se encuentran dos carpetas creapelicula y verpeliculas, las cuales tienen los archivos necesarios tanto para editar el front end como para recuperar la información del back end.



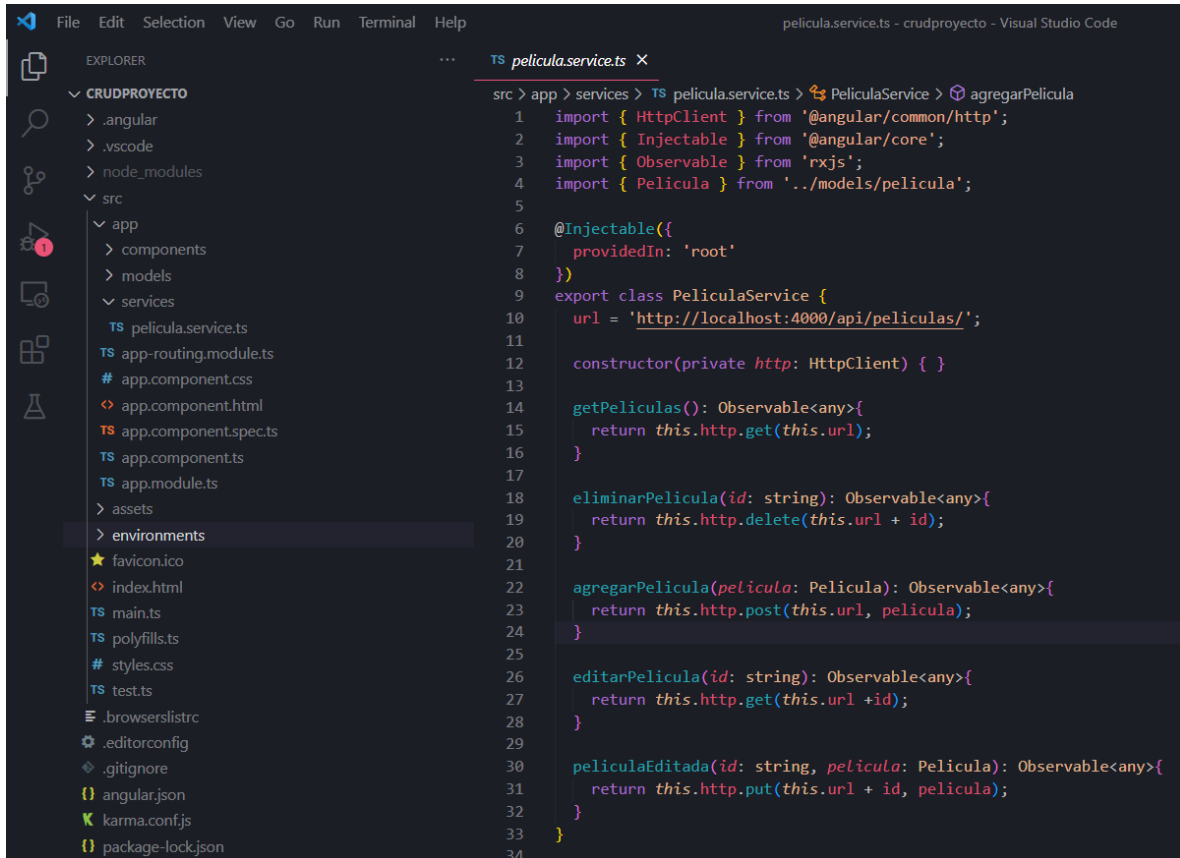
The screenshot shows the Visual Studio Code editor with two files open: `creapelicula.component.html` and `creapelicula.component.ts`. The left sidebar shows the Explorer view with the project structure. The `creapelicula.component.html` file contains HTML code for a form with three input fields for 'Nombre', 'Género', and 'Director', each with a placeholder and a label. The `creapelicula.component.ts` file contains TypeScript code for the `CreapeliculaComponent` class, which implements `OnInit` and `ngOnInit` methods, and has a `agregaPelicula` method that logs the form data.

En la ruta /src/app/models se encuentra un archivo en el que se declaran los datos que se mostrarán en el crud y que se podrán modificar.



The screenshot shows the Visual Studio Code editor with the `pelicula.ts` file open. The left sidebar shows the Explorer view with the project structure. The `pelicula.ts` file contains TypeScript code for the `Pelicula` class, which has properties `id`, `nombre`, `genero`, `director`, and `anio`, and a `constructor` method that initializes these properties.

Por último, en la ruta `/src/app/services` se encuentra el archivo del servicio que contiene las funciones de las acciones que se pueden realizar en el crud.



```
src > app > services > TS pelicula.service.ts > PelículaService > agregarPelícula
1  import { HttpClient } from '@angular/common/http';
2  import { Injectable } from '@angular/core';
3  import { Observable } from 'rxjs';
4  import { Película } from '../models/pelicula';
5
6  @Injectable({
7    providedIn: 'root'
8  })
9  export class PelículaService {
10    url = 'http://localhost:4000/api/peliculas/';
11
12    constructor(private http: HttpClient) { }
13
14    getPelículas(): Observable<any>{
15      return this.http.get(this.url);
16    }
17
18    eliminarPelícula(id: string): Observable<any>{
19      return this.http.delete(this.url + id);
20    }
21
22    agregarPelícula(película: Película): Observable<any>{
23      return this.http.post(this.url, película);
24    }
25
26    editarPelícula(id: string): Observable<any>{
27      return this.http.get(this.url + id);
28    }
29
30    películaEditada(id: string, película: Película): Observable<any>{
31      return this.http.put(this.url + id, película);
32    }
33  }
34
```