



**UNIVERSIDAD AUTONOMA DE CHIAPAS
FACULTAD DE CONTADURIA Y ADMINISTRACION
CAMPUS I TUXTLA GUTIERREZ.**

**LICENCIATURA EN INGENIERÍA EN DESARROLLO Y
TECNOLOGÍAS DE SOFTWARE.**

ASIGNATURA:

TALLER DE DESARROLLO 4.

ACTIVIDAD:

EXAMEN

ALUMNO:

GARCIA MONTEJO JESUS ALBERTO.

MATRICULA: A210324

CATEDRATICO:

DR. GUTIERREZ ALFARO LUIS.

SEMESTRE: 6°

GRUPO: N

LUGAR: TUXTLA GTZ.

FECHA: 21/02/2024

CREACION DE UN PROYECTO EN NESTJS.

Paso 1. Crea una carpeta para tu proyecto. Abre PowerShell y crea una carpeta en tu disco local (por ejemplo, disco C):

```
md Examen
```

```
PS C:\> md Examen

Directorio: C:\

Mode                LastWriteTime         Length Name
----                -
d-----          20/02/2024   23:27             Examen

PS C:\> cd Examen
PS C:\Examen>
```

Una vez creada y estemos ubicados dentro de la ruta de nuestra carpeta creada pasamos al siguiente paso.

Paso 2. Ejecutaremos el siguiente comando el cual nos creara nuestro proyecto con la estructura base de NestJS.

```
PS C:\> cd Examen
PS C:\Examen> nest new api-examen
[?] We will scaffold your app in a few seconds..

Which package manager would you like to use? (Use arrow keys)
> npm
> yarn
> pnpm
```

Selecciona la opción de npm.

```
PS C:\Examen> nest new api-examen
[?] We will scaffold your app in a few seconds..

Which package manager would you like to use? npm
CREATE api-examen/.eslintrc.js (688 bytes)
CREATE api-examen/.prettierrc (54 bytes)
CREATE api-examen/nest-cli.json (179 bytes)
CREATE api-examen/package.json (2019 bytes)
CREATE api-examen/README.md (3413 bytes)
CREATE api-examen/tsconfig.build.json (101 bytes)
CREATE api-examen/tsconfig.json (567 bytes)
CREATE api-examen/src/app.controller.ts (286 bytes)
CREATE api-examen/src/app.module.ts (259 bytes)
CREATE api-examen/src/app.service.ts (150 bytes)
CREATE api-examen/src/main.ts (216 bytes)
CREATE api-examen/src/app.controller.spec.ts (639 bytes)
CREATE api-examen/test/jest-e2e.json (192 bytes)
CREATE api-examen/test/app.e2e-spec.ts (654 bytes)

Installation in progress... [?]

Successfully created project api-examen
Get started with the following commands:

$ cd api-examen
$ npm run start

Thanks for installing Nest [?]
Please consider donating to our open collective
to help us maintain this package.

[?] Donate: https://opencollective.com/nest

PS C:\Examen>
```

Nota: Si no tienes instalado Nest CLI, puedes instalarlo con el siguiente comando.

```
npm i -g @nestjs/cli
```

```
PS C:\Examen> npm i -g @nestjs/cli
```

Paso 3. Con “cd nombreproyecto” entramos a la carpeta de nuestro proyecto, y ejecutamos el siguiente comando.

Esto es para Instalar el módulo HBS (Handlebars) en tu proyecto.

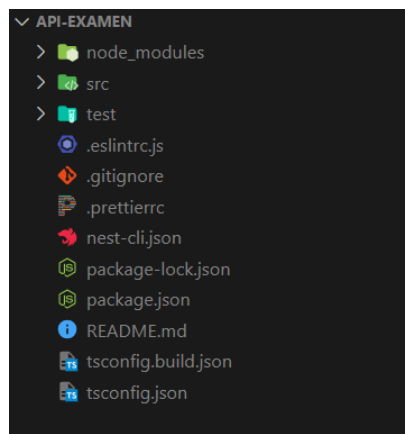
```
npm install --save hbs
```

```
PS C:\Examen> cd api-examen  
PS C:\Examen\api-examen> npm install --save hbs
```

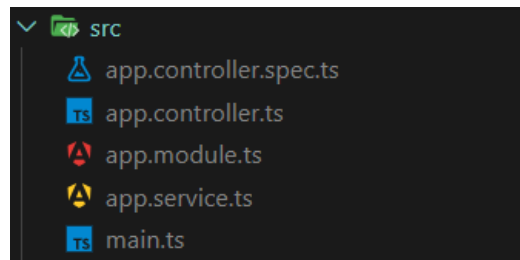
Una vez termine la instalación ejecutamos “code .” de la siguiente manera, para abrir nuestro proyecto en Visual Studio Code.

```
PS C:\Examen> cd api-examen  
PS C:\Examen\api-examen> npm install --save hbs  
  
added 7 packages, and audited 730 packages in 3s  
  
115 packages are looking for funding  
  run `npm fund` for details  
  
found 0 vulnerabilities  
PS C:\Examen\api-examen> code .
```

Esta sería la estructura base que se nos generó.



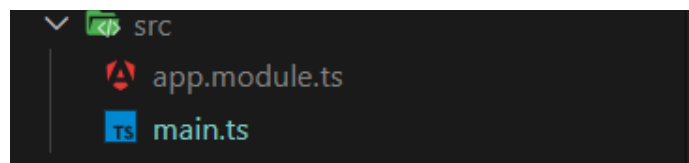
Paso 4. Dentro de la carpeta “src” encontraremos un controlador, un módulo, un servicio y un archivo llamado main.



Paso 5. Eliminar los siguientes archivos de la carpeta "src":.

- app.controller.spec.ts
- app.controller.ts
- app.services.ts

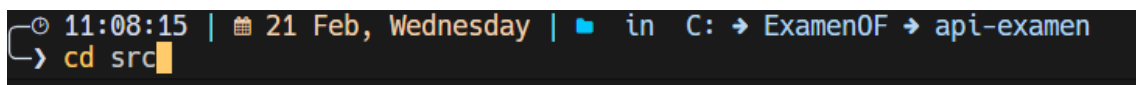
Quedando de la siguiente manera.



Paso 6. Abrir la terminal integrada de Visual Studio Code y navegar a la carpeta "src".

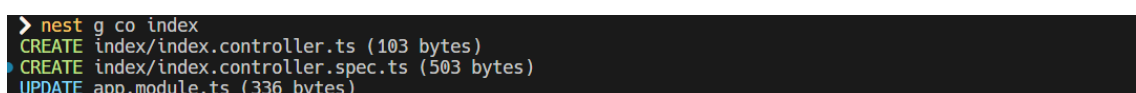
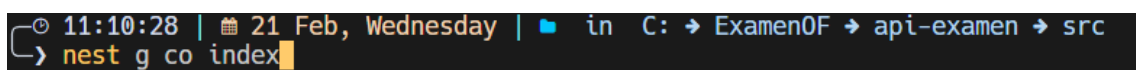
```
cd src
```

De la siguiente manera.



Una vez estando dentro de la ruta de la carpeta “src” ejecutamos el siguiente comando para crear un controlador, el cual llamare “index”, ejecutando el siguiente comando.

```
nest g co index
```



Paso 7. Regresar a la ruta principal ejecutando el comando `cd ...`

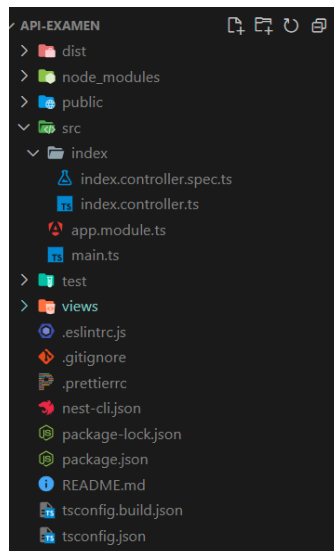
```
11:11:01 | 21 Feb, Wednesday | in C: → Examen0F → api-examen  
>
```

Paso 8. Crearemos una carpeta llamada “views” y una llamada “public” (fuera de la carpeta `src`), “views” contendrá nuestro archivo HTML, o en este caso `hbs`, y “public”, los estilos `css` y las imágenes.

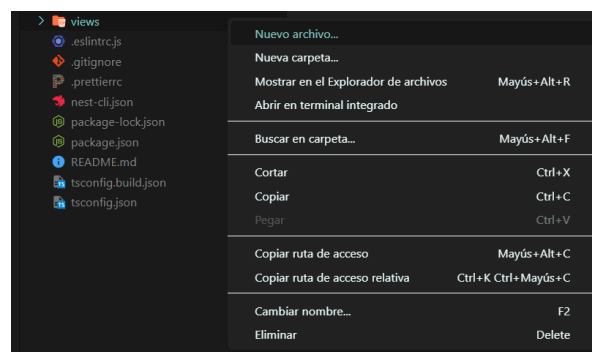
```
11:08:15 | 21 Feb, Wednesday | in C: → Examen0F → api-examen  
> md views
```

```
10:48:26 | 21 Feb, Wednesday | in C: → Examen0F → api-examen  
> md public
```

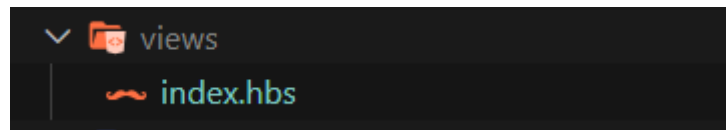
Como podemos observar se agregaron las carpetas que creamos.



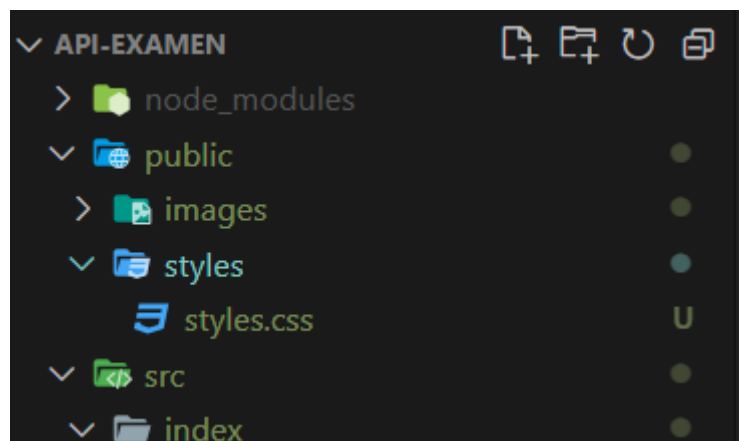
Paso 9. Dentro de la carpeta “views” crearemos un archivo `hbs`, o HTML, dando clic derecho sobre la carpeta y dando nuevamente clic en “Nuevo archivo”.



Damos clic en “Nuevo archivo” y nombramos a nuestro archivo con su extensión, para esta práctica “index.hbs”.



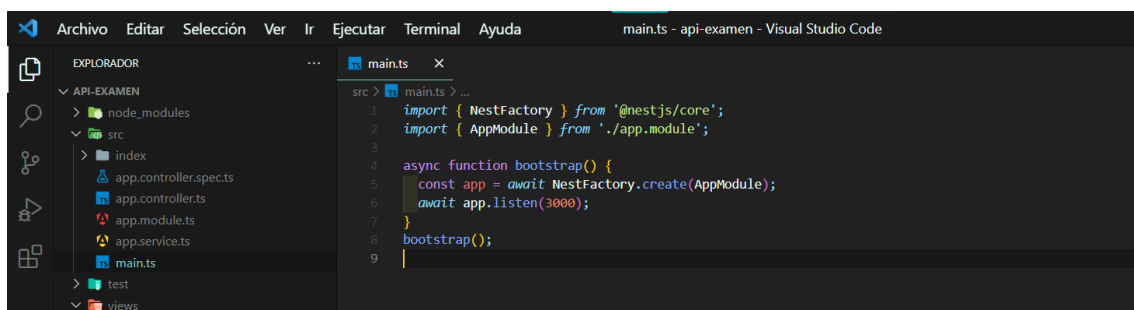
Repetimos pero ahora dentro de la carpeta “public” crearemos una carpeta llamada “styles”, y una llamada “images” y dentro de “styles” creamos un archivo llamado “styles.css”, quedando de la siguiente manera.



Nota: Dentro de “images” se podrá alojar imágenes que necesitemos para nuestro diseño.

Paso 10. Recordemos que dentro de la carpeta “src” esta nuestro archivo principal, es decir el “main.ts” lo modificamos de acuerdo a nuestras necesidades, para esta práctica se hará de la siguiente manera.

Por defecto tendremos de la siguiente manera.



Y lo modificaremos, quedando de la siguiente manera.

```
main.ts U X
src > main.ts > ...
1 import { NestFactory } from '@nestjs/core';
2 import { AppModule } from './app.module';
3 import { join } from 'path';
4 import { NestExpressApplication } from '@nestjs/platform-express';
5 //import * as express from 'express';
6
7 async function bootstrap() {
8   const app = await NestFactory.create<NestExpressApplication>(AppModule);
9
10  app.useStaticAssets(join(__dirname, '..', 'public'));
11  app.setBaseViewsDir(join(__dirname, '..', 'views'));
12  app.setViewEngine('hbs');
13
14  await app.listen(3000);
15 }
16 bootstrap();
```

Paso 11. Ahora trabajaremos con nuestro modulo, abrimos el archivo llamado app.module.

```
main.ts app.module.ts X
src > app.module.ts > ...
1 import { Module } from '@nestjs/common';
2 import { AppController } from './app.controller'; No se encuentra el módulo "./app.controller" ni sus declaraciones de tipos corr
3 import { AppService } from './app.service'; No se encuentra el módulo "./app.service" ni sus declaraciones de tipos correspondien
4 import { IndexController } from './index/index.controller';
5
6 @Module({
7   imports: [],
8   controllers: [AppController, IndexController],
9   providers: [AppService],
10 })
11 export class AppModule {}
12
```

Como anteriormente eliminamos el controlador y el servicio que se generaron por defecto, nos marca un error. Entonces procedemos a eliminar las líneas correspondientes al controlador y servicio eliminados anteriormente.

```
main.ts app.module.ts X
src > app.module.ts > ...
1 import { Module } from '@nestjs/common';
2 import { IndexController } from './index/index.controller';
3
4 @Module({
5   imports: [],
6   controllers: [IndexController],
7   providers: [],
8 })
9 export class AppModule {}
10
```

Paso 12. Configurar el controlador creado anteriormente según tus necesidades.

Por defecto estará así.

```
main.ts app.module.ts index.controller.ts X
src > index > index.controller.ts > ...
1 import { Controller } from '@nestjs/common';
2
3 @Controller('index')
4 export class IndexController {}
5
```

Lo dejaremos de la siguiente manera, para esta práctica.

```
main.ts app.module.ts index.controller.ts
src > index > index.controller.ts > ...
1 import { Get, Controller, Render } from '@nestjs/common';
2
3 @Controller('index')
4 export class IndexController {
5   @Get()
6   @Render('index')
7   root() {
8     return {
9       nombre: 'Jesus Alberto Garcia Montejo',
10      semestre: '6° N'
11     };
12   }
13 }
14
15
```

Paso 13. Diseñamos el archivo "index.hbs", y el de "styles.css" según nuestras necesidades.

```
main.ts app.module.ts index.controller.ts index.hbs
views > index.hbs > ...
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Saludo</title>
8   <link rel="stylesheet" href="../styles/styles.css" <!-- Ruta al archivo CSS -->
9
10 <script>
11   document.addEventListener("DOMContentLoaded", function() {
12     const nombre = document.querySelector(".nombre"); // Selecciona el elemento con la clase "nombre"
13     const semestre = document.querySelector(".semestre"); // Selecciona el elemento con la clase "semestre"
14
15     // Agrega un evento para cuando el cursor entra al elemento con la clase "nombre"
16     nombre.addEventListener("mouseenter", function() {
17       // Agrega una clase para aplicar la animación
18       nombre.classList.add("animacion");
19     });
20
21     // Agrega un evento para cuando el cursor sale del elemento con la clase "nombre"
22     nombre.addEventListener("mouseleave", function() {
23       // Remueve la clase de la animación al salir del elemento
24       nombre.classList.remove("animacion");
25     });
26
```

Paso 14. Compilar y construir el proyecto utilizando el siguiente comando

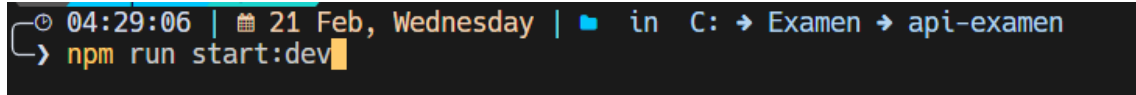
npm run build

```
04:29:06 | 21 Feb, Wednesday | in C: → Examen → api-examen
> npm run build
```

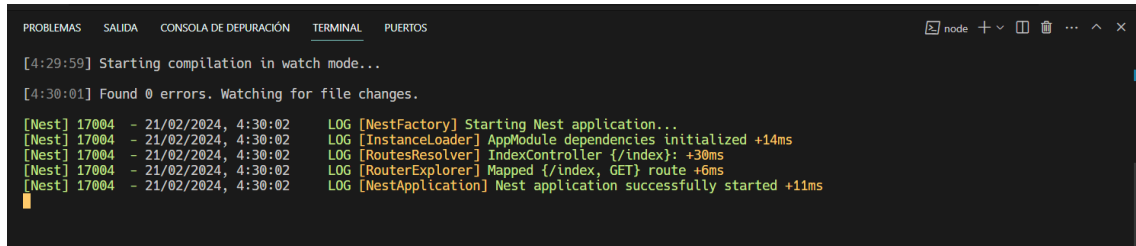
```
> npm run build
● > api-examen@0.0.1 build
> nest build
```


Paso 15. Una vez compilado, ejecutar el proyecto utilizando el siguiente comando

`npm run start:dev`



```
04:29:06 | 21 Feb, Wednesday | in C: → Examen → api-examen  
> npm run start:dev
```

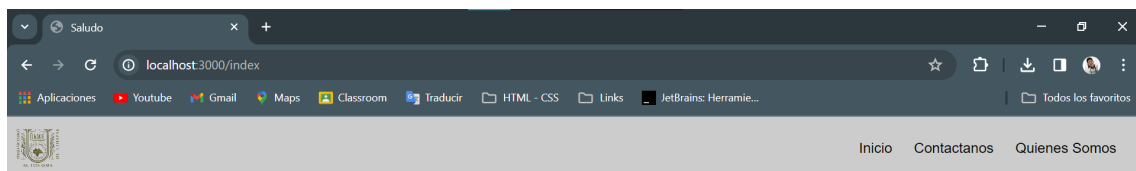


```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS  
[4:29:59] Starting compilation in watch mode...  
[4:30:01] Found 0 errors. Watching for file changes.  
[Nest] 17004 - 21/02/2024, 4:30:02 LOG [NestFactory] Starting Nest application...  
[Nest] 17004 - 21/02/2024, 4:30:02 LOG [InstanceLoader] AppModule dependencies initialized +14ms  
[Nest] 17004 - 21/02/2024, 4:30:02 LOG [RoutesResolver] IndexController {/index}: +30ms  
[Nest] 17004 - 21/02/2024, 4:30:02 LOG [RouterExplorer] Mapped {/index, GET} route +6ms  
[Nest] 17004 - 21/02/2024, 4:30:02 LOG [NestApplication] Nest application successfully started +11ms
```

Como ya nos encargamos de dar direccionamiento es decir configurar nuestro controlador, main, y modulo para que estos se puedan comunicar e indicar cuál será la ruta por la cual se ejecutara, entonces al ejecutar el programa este realiza el mapeo y nos indica la ruta por la cual se estará ejecutando.

Nota: El proyecto estará disponible en la dirección <http://localhost:3000/index>.

Asi se nos visualizara.



Jesus Alberto Garcia Montejo

6° N