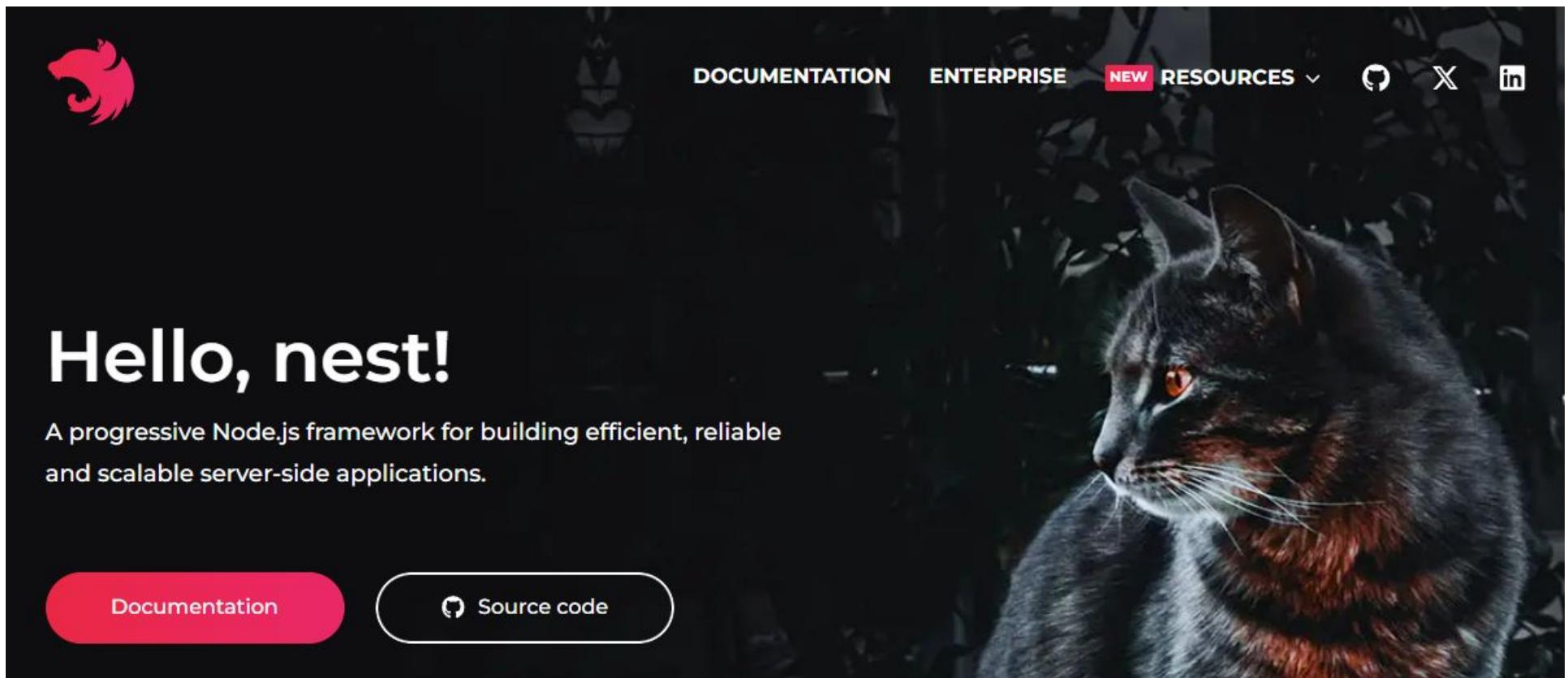


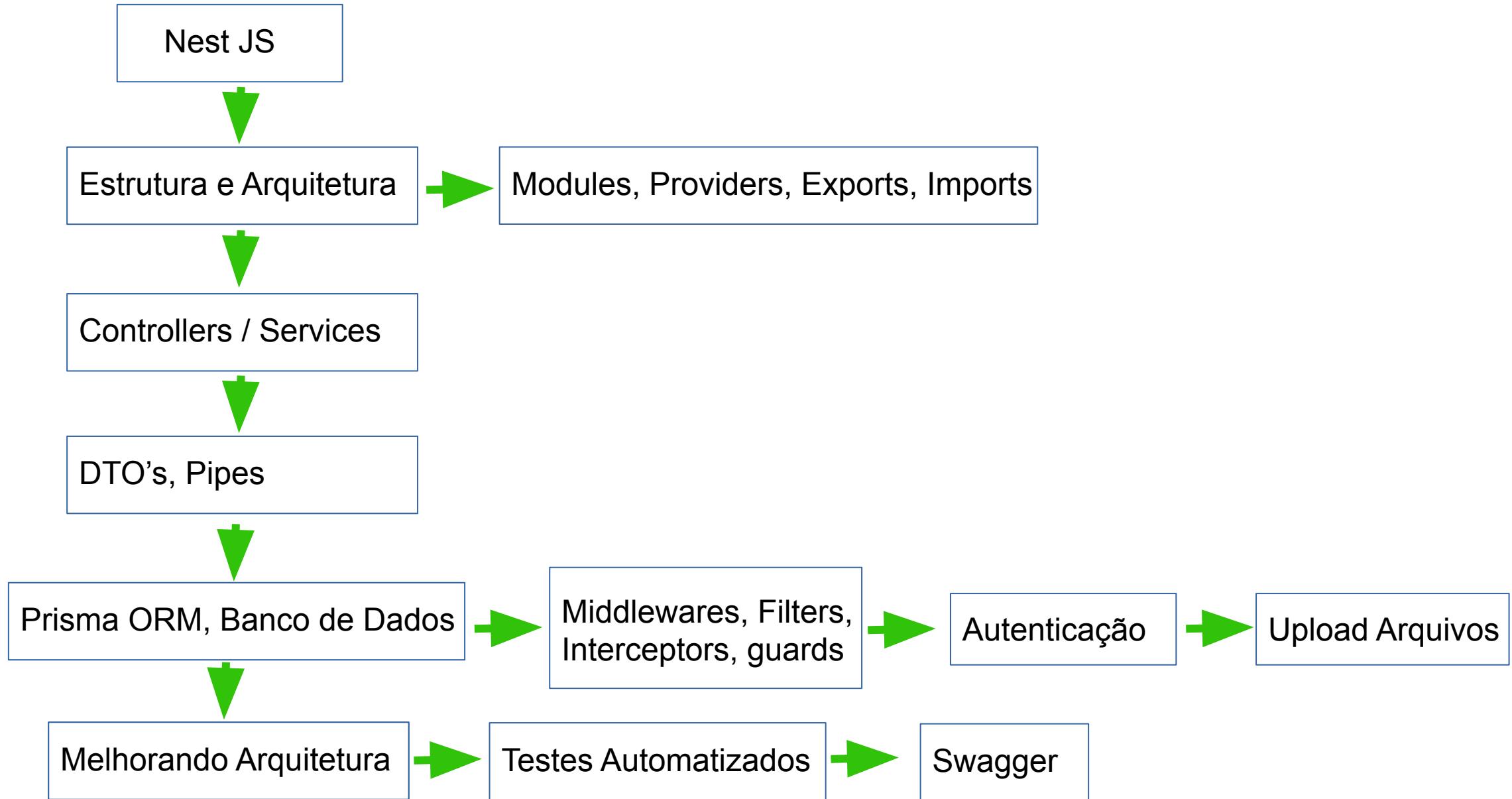
Nest.js

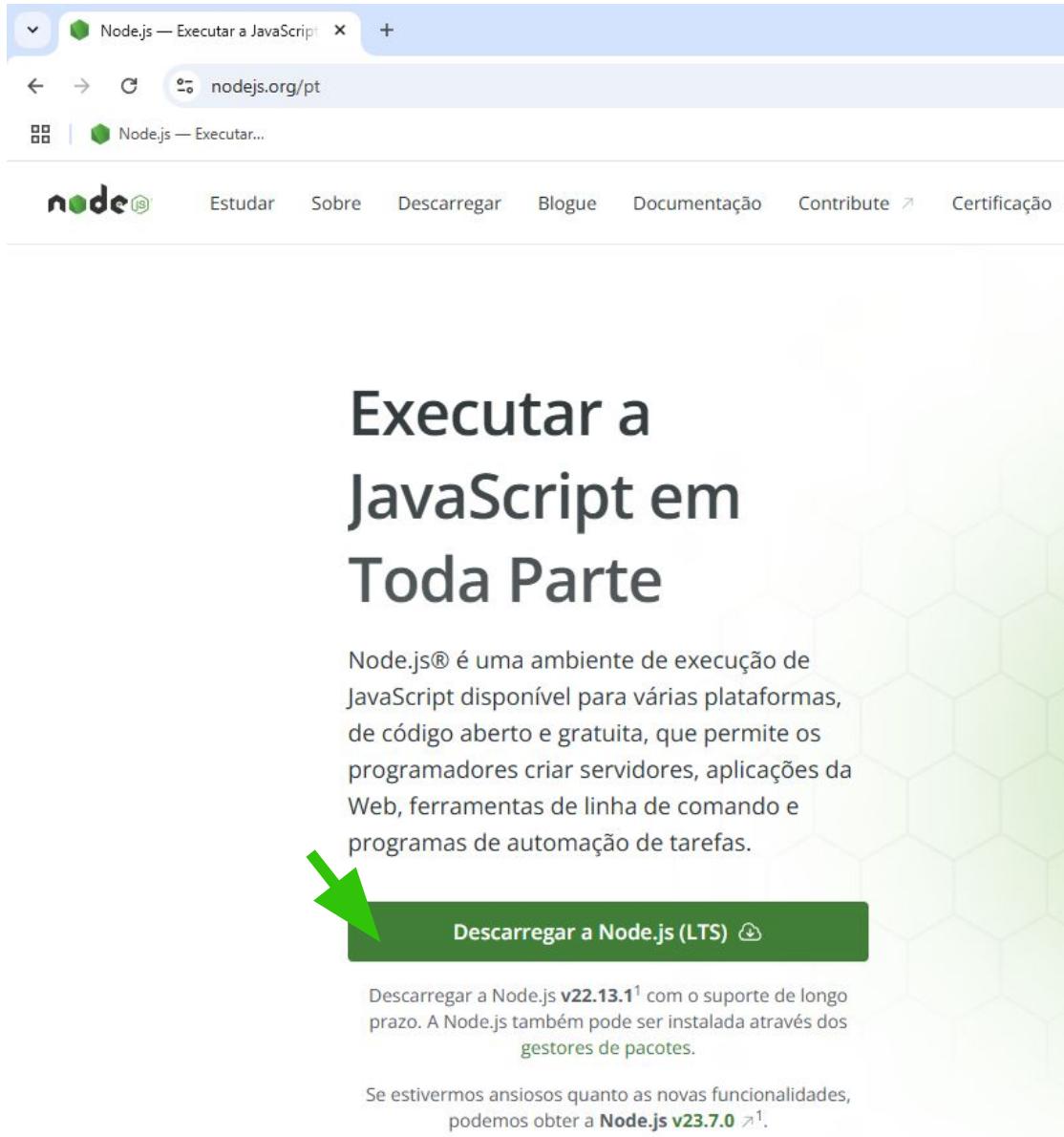


Prof. Dr. Robyson Aggio
www.aggioirati.com.br



Introdução





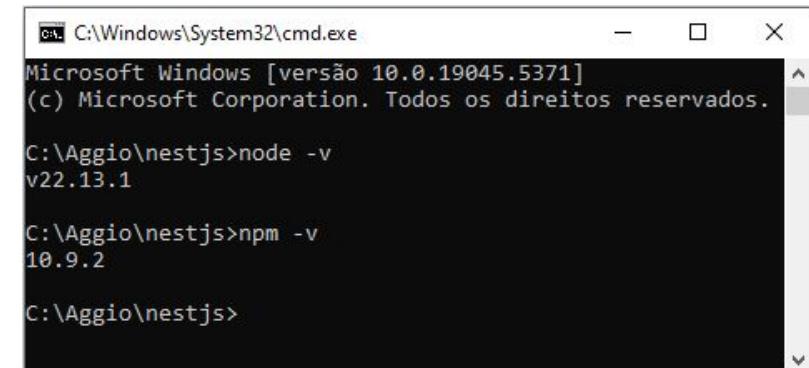
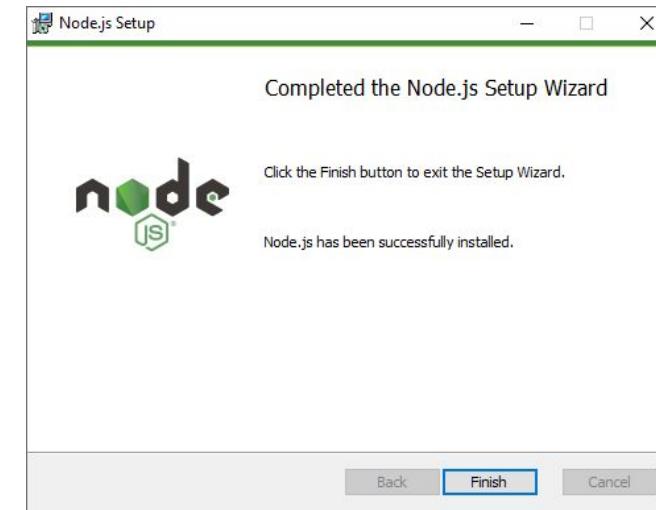
Executar a JavaScript em Toda Parte

Node.js® é uma ambiente de execução de JavaScript disponível para várias plataformas, de código aberto e gratuita, que permite os programadores criar servidores, aplicações da Web, ferramentas de linha de comando e programas de automação de tarefas.

Descarregar a Node.js (LTS)

Descarregar a Node.js **v22.13.1¹** com o suporte de longo prazo. A Node.js também pode ser instalada através dos gestores de pacotes.

Se estivermos ansiosos quanto as novas funcionalidades, podemos obter a **Node.js v23.7.0¹**.



```
C:\Windows\System32\cmd.exe
Microsoft Windows [versão 10.0.19045.5371]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Aggio\nestjs>node -v
v22.13.1

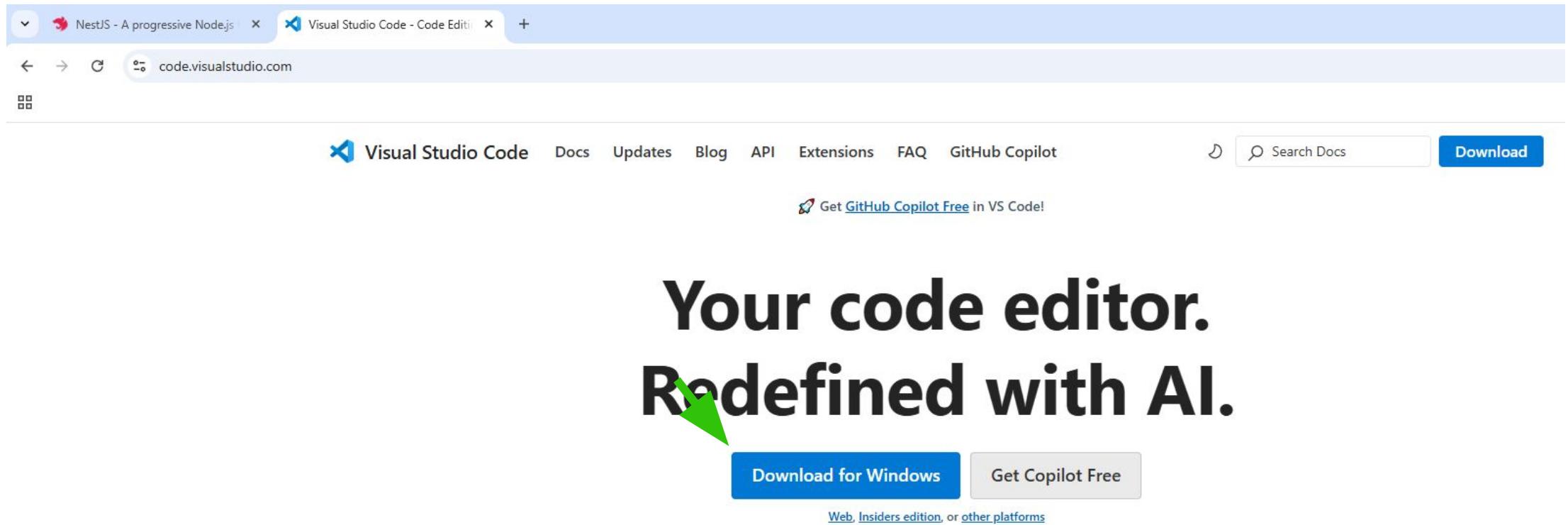
C:\Aggio\nestjs>npm -v
10.9.2

C:\Aggio\nestjs>
```

npm = Node Package Manager



Visual Studio Code



The screenshot shows the official website for Visual Studio Code at code.visualstudio.com. The page features a large headline: "Your code editor. Redefined with AI." A green arrow points from the word "Redefined" towards the "Download for Windows" button. Below the headline are two buttons: "Download for Windows" (blue) and "Get Copilot Free" (grey). At the bottom, there's a link: "Web, Insiders edition, or other platforms". The top navigation bar includes links for Visual Studio Code, Docs, Updates, Blog, API, Extensions, FAQ, GitHub Copilot, a search bar, and a "Download" button.

NestJS - A progressive Node.js X Visual Studio Code - Code Editor X +

code.visualstudio.com

Visual Studio Code Docs Updates Blog API Extensions FAQ GitHub Copilot

Search Docs Download

Get GitHub Copilot Free in VS Code!

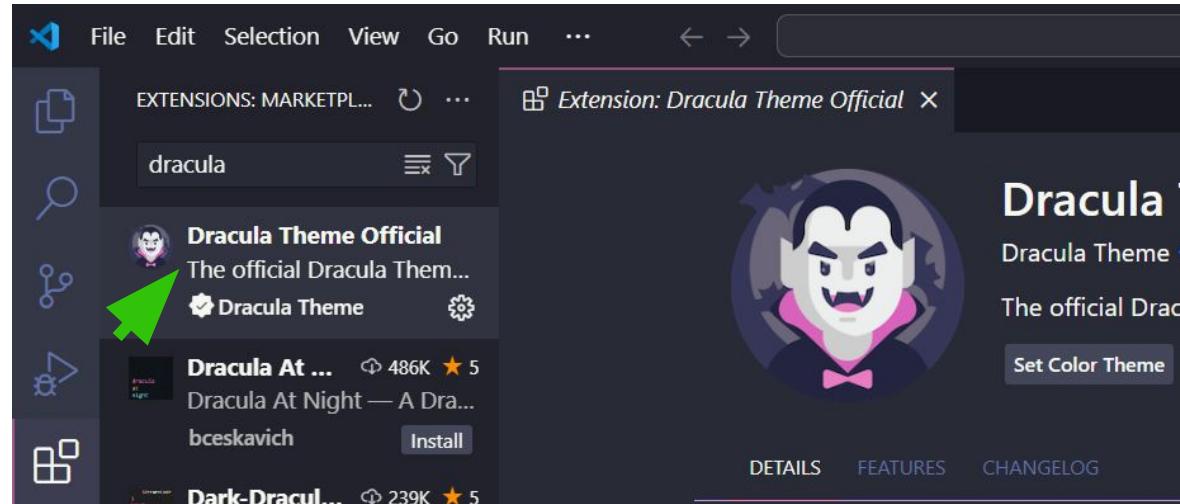
Your code editor. Redefined with AI.

[Download for Windows](#) [Get Copilot Free](#)

Web, Insiders edition, or other platforms



Visual Studio Code



File Edit Selection View Go Run ...

EXTENSIONS: MARKETPL... ⏪ ...

dracula ⏪ ...

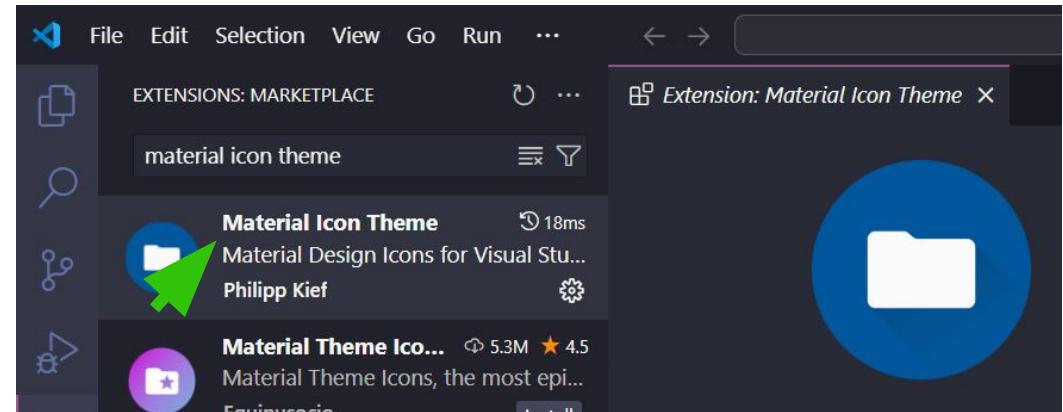
Dracula Theme Official
The official Dracula Theme...
Dracula Theme

Dracula At ... ⏪ 486K ★ 5
Dracula At Night — A Dra...
bceskavich **Install**

Dark-Dracul... ⏪ 239K ★ 5

DETAILS FEATURES CHANGELOG

A green arrow points to the "Dracula Theme Official" extension entry.



File Edit Selection View Go Run ...

EXTENSIONS: MARKETPLACE ⏪ ...

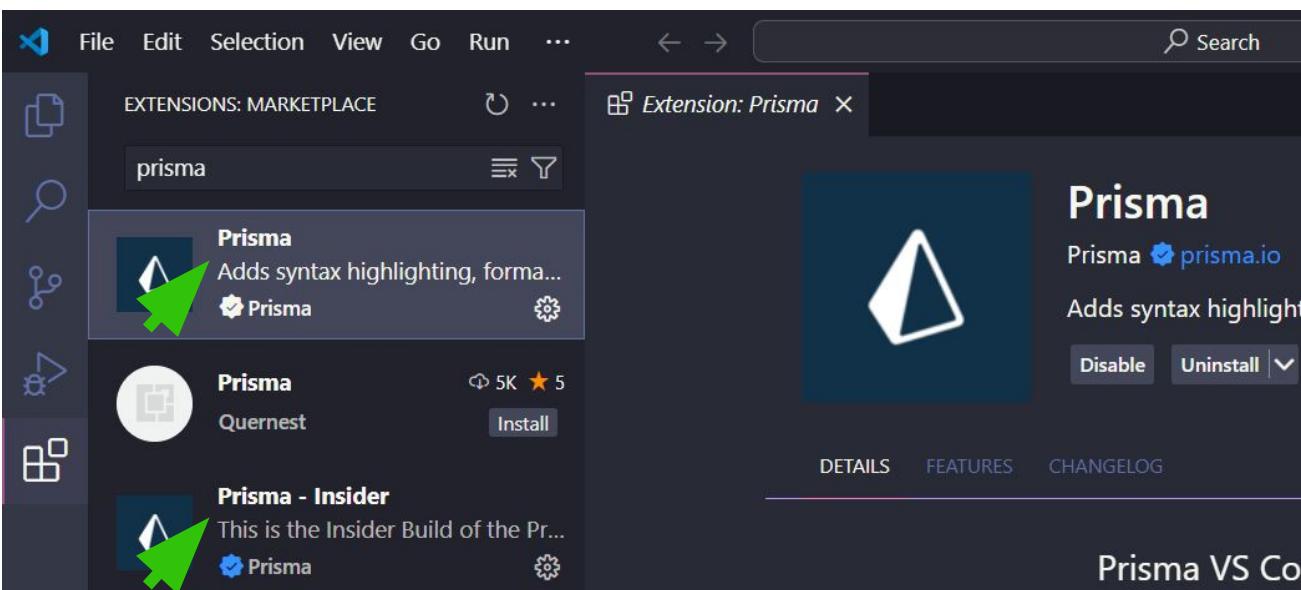
material icon theme ⏪ ...

Material Icon Theme ⏪ 18ms
Material Design Icons for Visual Studio Code...
Philipp Kief

Material Theme Ico... ⏪ 5.3M ★ 4.5
Material Theme Icons, the most epic icons ever!...
Equipuración **Install**

Extension: Material Icon Theme X

A green arrow points to the "Material Icon Theme" extension entry.



File Edit Selection View Go Run ...

EXTENSIONS: MARKETPLACE ⏪ ...

prisma ⏪ ...

Prisma
Adds syntax highlighting, form...
Prisma

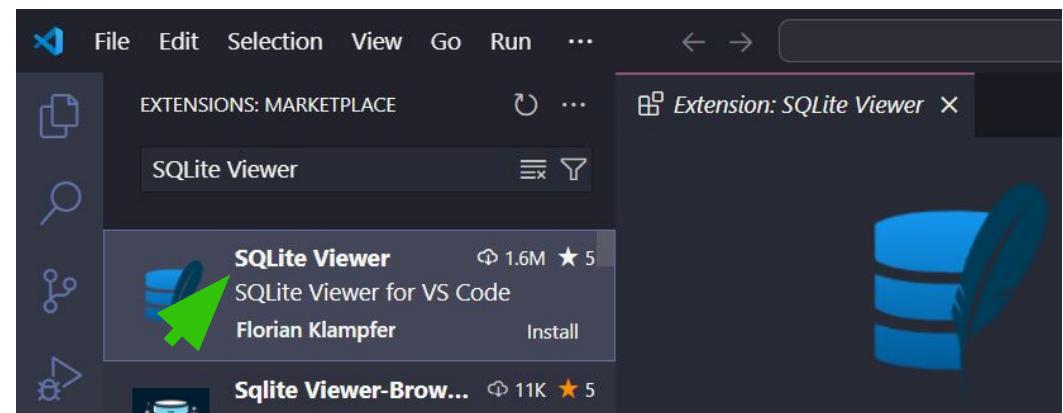
Prisma ⏪ 5K ★ 5
Quernest **Install**

Prisma - Insider
This is the Insider Build of the Prisma extension...
Prisma

DETAILS FEATURES CHANGELOG

Prisma VS Code

Three green arrows point to the "Prisma" extension entry.



File Edit Selection View Go Run ...

EXTENSIONS: MARKETPLACE ⏪ ...

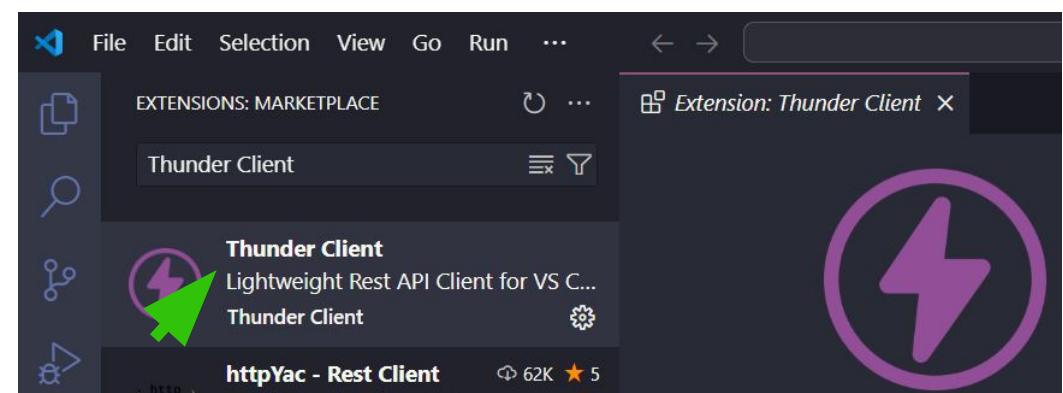
SQLite Viewer ⏪ ...

SQLite Viewer ⏪ 1.6M ★ 5
SQLite Viewer for VS Code
Florian Klampfer **Install**

Sqlite Viewer-Brow... ⏪ 11K ★ 5
Browse and manage your sqlite data...
Sqlite Viewer

Extension: SQLite Viewer X

A green arrow points to the "SQLite Viewer" extension entry.



File Edit Selection View Go Run ...

EXTENSIONS: MARKETPLACE ⏪ ...

Thunder Client ⏪ ...

Thunder Client
Lightweight Rest API Client for VS Code...
Thunder Client

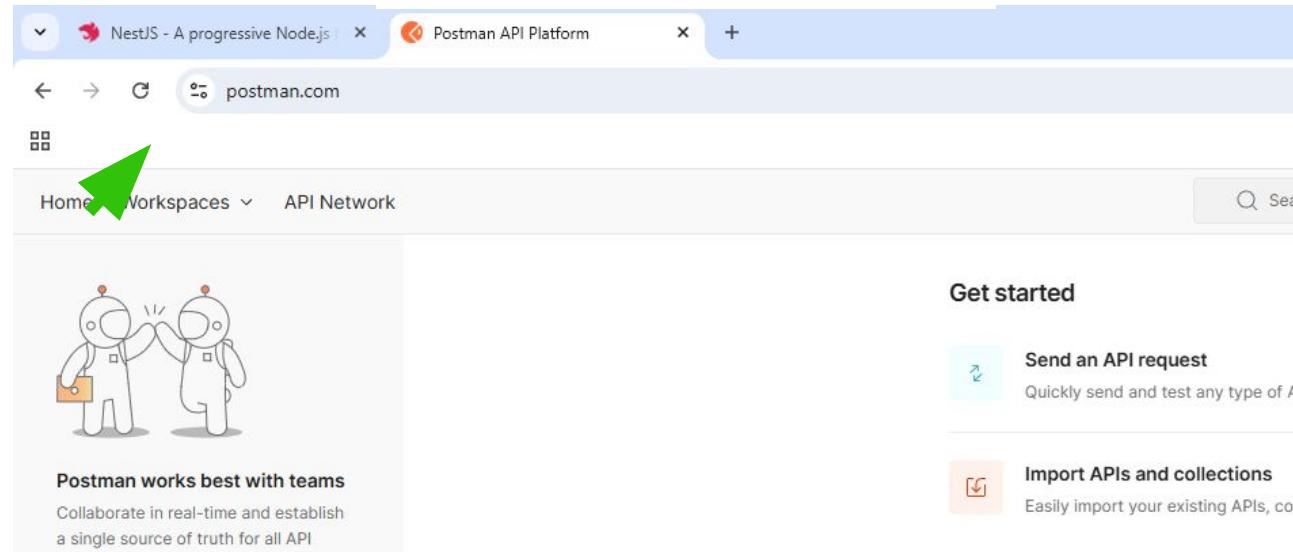
httpYac - Rest Client ⏪ 62K ★ 5
A REST client for VS Code...
httpYac

Extension: Thunder Client X

A green arrow points to the "Thunder Client" extension entry.



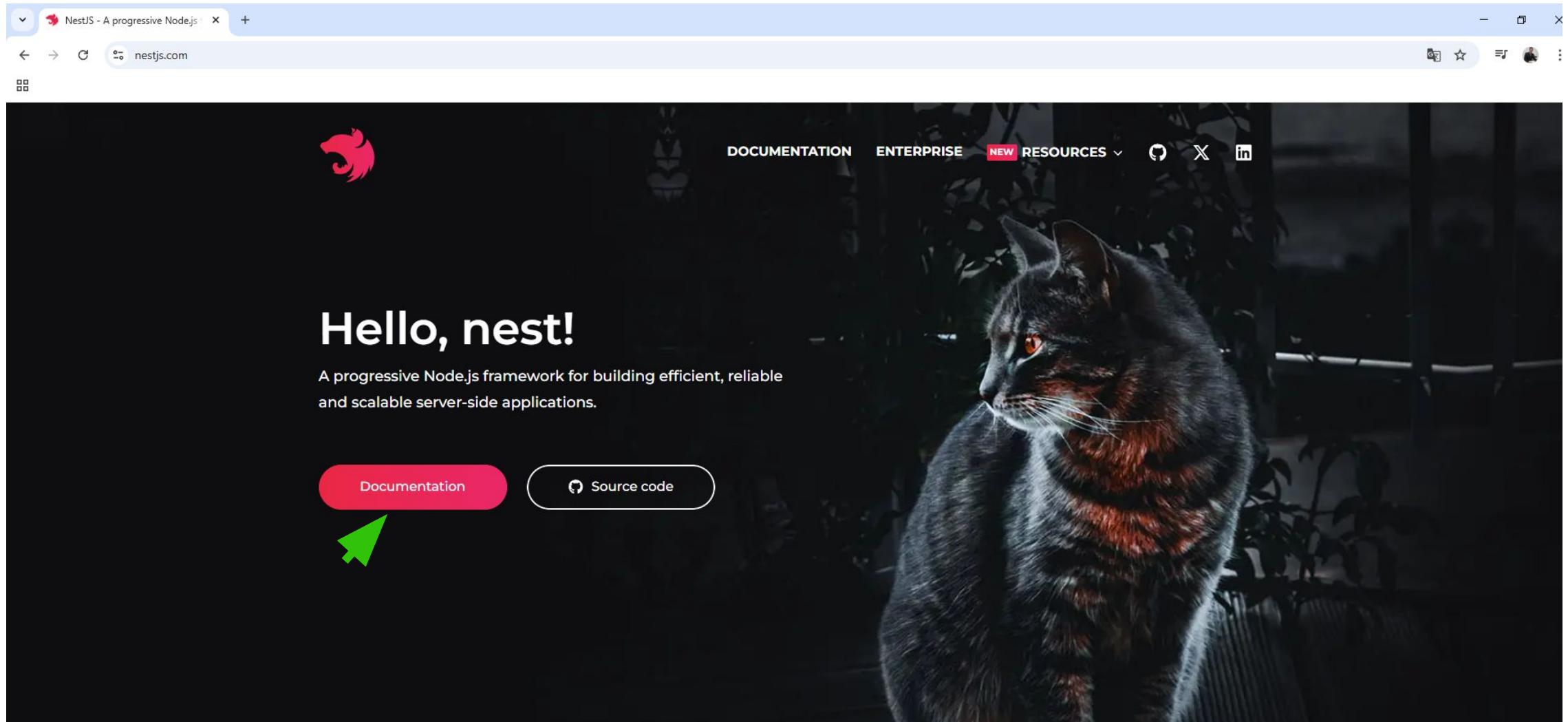
POSTMAN



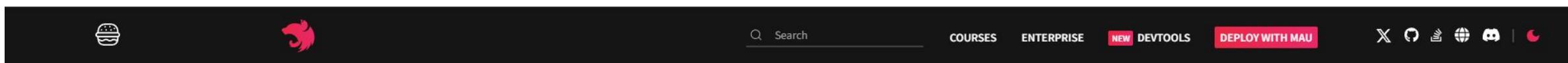
The screenshot shows the Postman API Platform homepage. At the top, there is a navigation bar with tabs for "Home", "Workspaces", and "API Network". A green arrow points to the "Home" tab. Below the navigation bar, there is a section titled "Postman works best with teams" featuring two cartoon astronauts shaking hands. To the right of this section is a "Get started" area with three main sections: "Send an API request", "Import APIs and collections", and "Create a collection". The "Send an API request" section includes a sub-section for "Quickly send and test any type of API".



Nest JS



A screenshot of a web browser displaying the Nest.js homepage. The URL 'nestjs.com' is visible in the address bar. The page features a large, dark background image of a black cat with glowing red eyes. In the top left corner, the Nest.js logo is displayed. The top navigation bar includes links for 'DOCUMENTATION', 'ENTERPRISE', and 'NEW RESOURCES'. A green cursor arrow points towards the 'Documentation' button, which is highlighted with a pink rounded rectangle.



INTRODUCTION

OVERVIEW

- First steps
- Controllers
- Providers
- Modules
- Middleware
- Exception filters
- Pipes
- Guards
- Interceptors
- Custom decorators

FUNDAMENTALS

TECHNIQUES

SECURITY

GRAPHQL

WEBSOCKETS

MICROSERVICES

DEPLOYMENT

STANDALONE APPS

CLI

OPENAPI

RECIPES

FAQ

DEVTOOLS

MIGRATION GUIDE

OFFICIAL COURSES

DISCOVER

Introduction

Nest (NestJS) is a framework for building efficient, scalable [Node.js](#) server-side applications. It uses progressive JavaScript, is built with and fully supports [TypeScript](#) (yet still enables developers to code in pure JavaScript) and combines elements of OOP (Object Oriented Programming), FP (Functional Programming), and FRP (Functional Reactive Programming).

Under the hood, Nest makes use of robust HTTP Server frameworks like [Express](#) (the default)

Nest provides a level of abstraction above these common Node.js frameworks (Express/Firebase) and the myriad of third-party modules which are available for the underlying platform.

Philosophy

In recent years, thanks to Node.js, JavaScript has become the “lingua franca” of the web front-end and [Vue](#), which improve developer productivity and enable the creation of fast, testable, and maintainable applications. While there are many frameworks and libraries that exist for Node (and server-side JavaScript), none of them effectively solve the main problem.

Nest provides an out-of-the-box application architecture which allows developers and teams to build large-scale, maintainable, and testable server-side applications. This architecture is heavily inspired by Angular.

Installation

To get started, you can either scaffold the project with the [Nest CLI](#), or [clone a starter project](#).

To scaffold the project with the Nest CLI, run the following commands. This will create a new directory for your project, install the required dependencies, and generate several initial modules, creating a conventional base structure for your project. Creating a new project with the [Nest CLI](#) is recommended for first time users. We'll continue with this approach in [First Steps](#).

```
$ npm i -g @nestjs/cli  
$ nest new project-name
```

```
C:\Windows\System32\cmd.exe  
Microsoft Windows [versão 10.0.19045.5371]  
(c) Microsoft Corporation. Todos os direitos reservados.  
  
C:\Aggio\nestjs>node -v  
v22.13.1  
  
C:\Aggio\nestjs>npm -v  
10.9.2  
  
C:\Aggio\nestjs>npm i -g @nestjs/cli
```



Design and Development tips in your inbox. Every weekday.



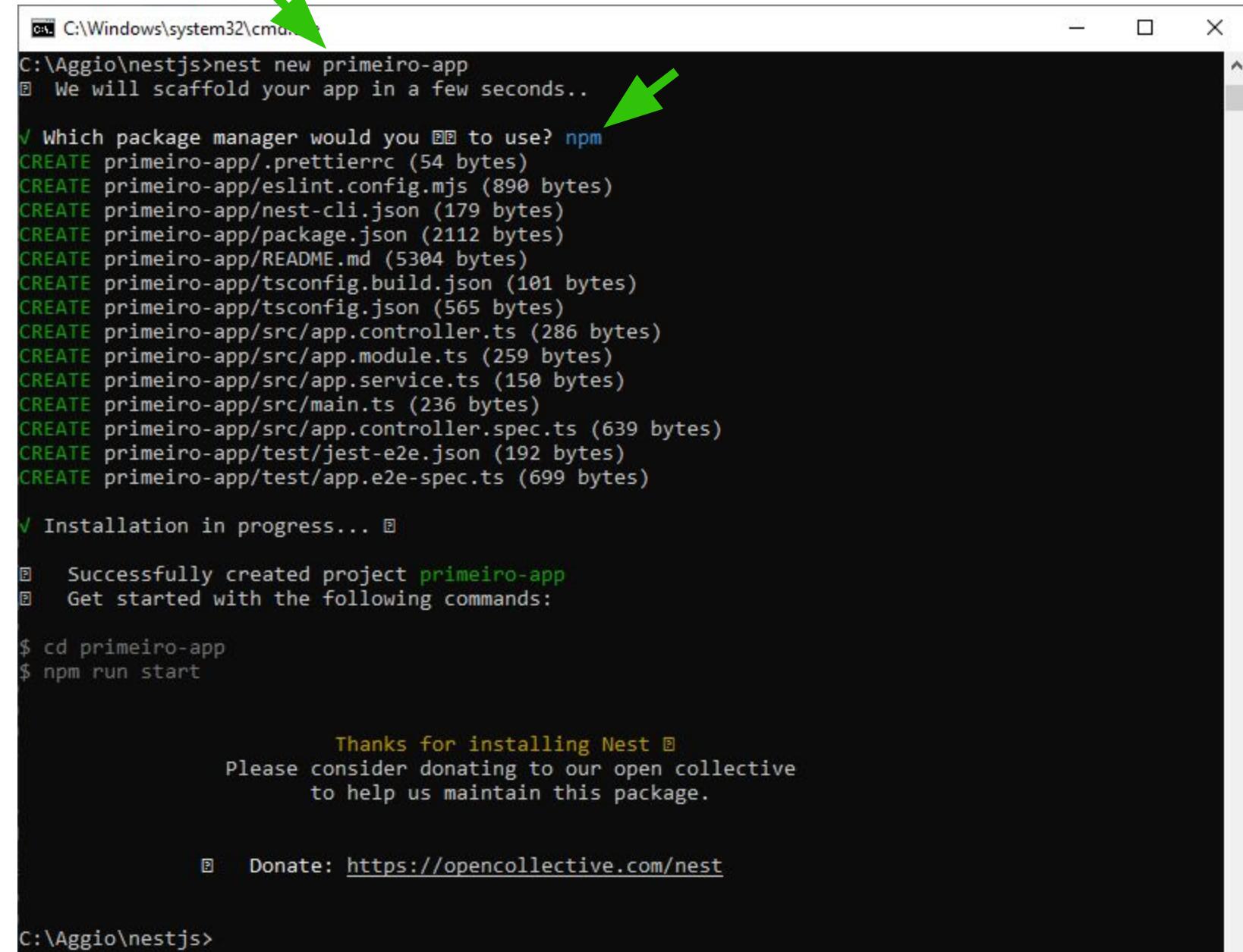
```
C:\Windows\system32\cmd.exe
C:\Aggio\nestjs>nest --help
Usage: nest <command> [options]

Options:
  -v, --version                                Output the current version.
  -h, --help                                     Output usage information.

Commands:
  new[n [options] [name]]                        Generate Nest application.
  build [options] [apps...]                      Build Nest application.
  start [options] [app]                           Run Nest application.
  info|i                                         Display Nest project details.
  add [options] <library>                         Adds support for an external library to your
                                                project.
  generate|g [options] <schematic> [name] [path] Generate a Nest element.
  Schematics available on @nestjs/schematics collection:

  name          alias          description
  application   application  Generate a new application workspace
  class          cl            Generate a new class
  configuration config        Generate a CLI configuration file
  controller    co            Generate a controller declaration
  decorator     d             Generate a custom decorator
  filter         f             Generate a filter declaration
  gateway       ga            Generate a gateway declaration
  guard         gu            Generate a guard declaration
  interceptor   itc           Generate an interceptor declaration
  interface     itf           Generate an interface
  library       lib           Generate a new library within a monorepo
  middleware   mi            Generate a middleware declaration
  module        mo            Generate a module declaration
  pipe          pi            Generate a pipe declaration
  provider      pr            Generate a provider declaration
  resolver      r             Generate a GraphQL resolver declaration
  resource      res           Generate a new CRUD resource
  service       s             Generate a service declaration
  sub-app       app           Generate a new application within a monorepo

C:\Aggio\nestjs>
```



```
C:\Windows\system32\cmd C:\Aggio\nestjs>nest new primeiro-app
  We will scaffold your app in a few seconds.. ↴

  ✓ Which package manager would you like to use? npm
CREATE primeiro-app/.prettierrc (54 bytes)
CREATE primeiro-app/eslint.config.mjs (890 bytes)
CREATE primeiro-app/nest-cli.json (179 bytes)
CREATE primeiro-app/package.json (2112 bytes)
CREATE primeiro-app/README.md (5304 bytes)
CREATE primeiro-app/tsconfig.build.json (101 bytes)
CREATE primeiro-app/tsconfig.json (565 bytes)
CREATE primeiro-app/src/app.controller.ts (286 bytes)
CREATE primeiro-app/src/app.module.ts (259 bytes)
CREATE primeiro-app/src/app.service.ts (150 bytes)
CREATE primeiro-app/src/main.ts (236 bytes)
CREATE primeiro-app/src/app.controller.spec.ts (639 bytes)
CREATE primeiro-app/test/jest-e2e.json (192 bytes)
CREATE primeiro-app/test/app.e2e-spec.ts (699 bytes)

  ✓ Installation in progress... ↴

  ⓘ Successfully created project primeiro-app
  ⓘ Get started with the following commands:

$ cd primeiro-app
$ 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

C:\Aggio\nestjs>
```

```
c:\ C:\Windows\system32\cmd.exe
✓ Installation in progress...
  Successfully created project primeiro-app
  Get started with the following commands:

$ cd primeiro-app
$ npm run start

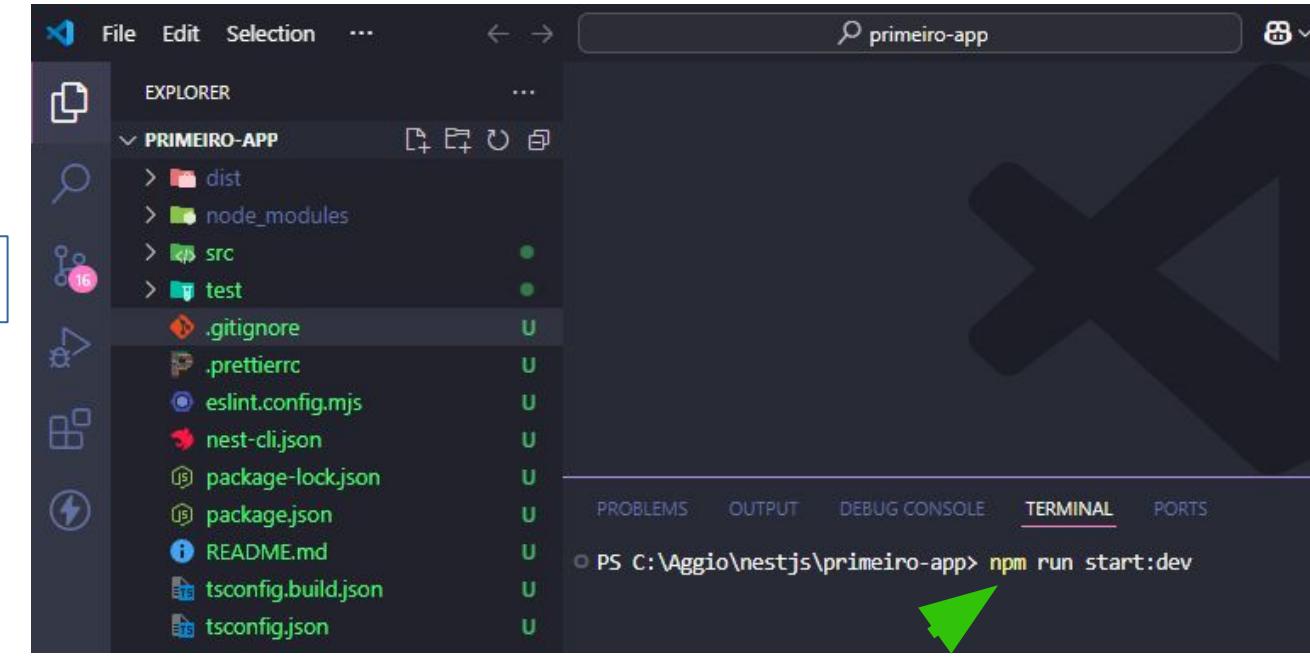
      Thanks for installing Nest
      Please consider donating to our open collective
      to help us maintain this package.

      ⚡  Donate: https://opencollective.com/nest
```

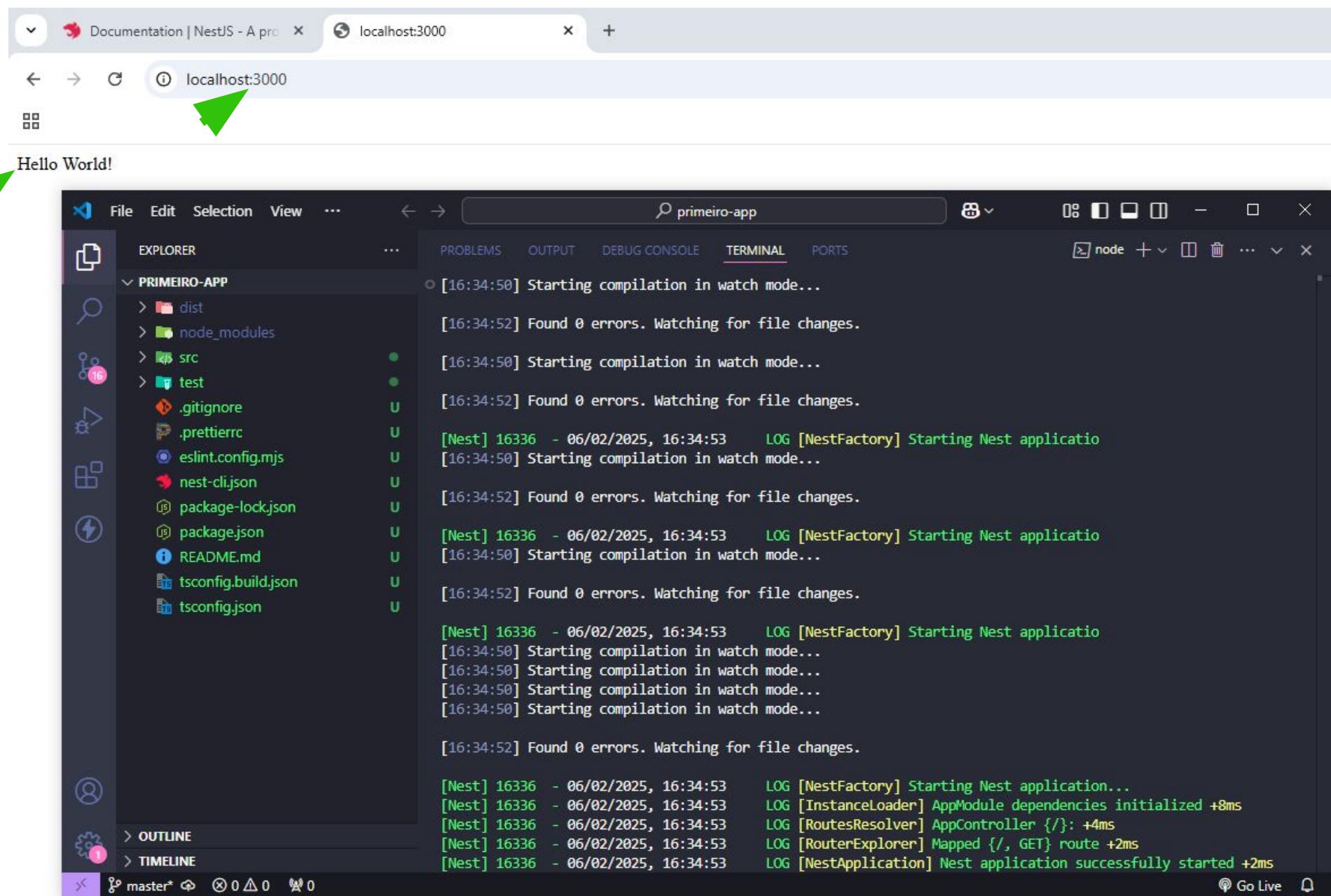
Terminal no VSCode:
ctrl + j

```
C:\ Prompt de Comando
C:\Aggio\nestjs>cd primeiro-app
C:\Aggio\nestjs\primeiro-app>code .
C:\Aggio\nestjs\primeiro-app>npm run start:dev
```

OU



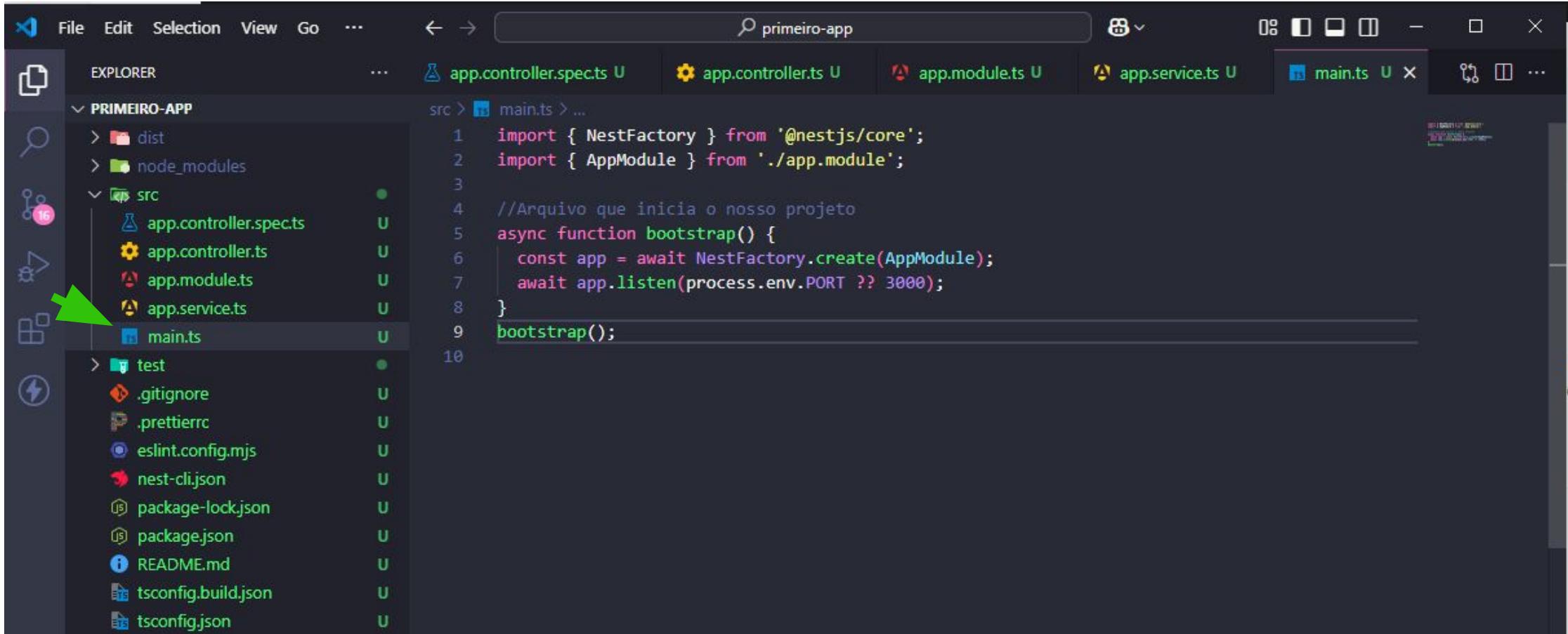
The screenshot shows the Visual Studio Code (VSCode) interface. The Explorer sidebar on the left displays the project structure for 'PRIMEIRO-APP' with files like dist, node_modules, src, test, .gitignore, .prettierrc, eslint.config.mjs, nest-cli.json, package-lock.json, package.json, README.md, tsconfig.build.json, and tsconfig.json. The status bar at the bottom shows the command PS C:\Aggio\nestjs\primeiro-app> npm run start:dev.



The image shows a dual-screen setup. The top screen displays a web browser with two tabs: 'Documentation | NestJS - A pro' and 'localhost:3000'. The 'localhost:3000' tab is active, showing the text 'Hello World!'. The bottom screen shows a Microsoft Visual Studio Code (VS Code) interface. The Explorer sidebar on the left lists files and folders for a project named 'PRIMEIRO-APP', including 'dist', 'node_modules', 'src', 'test', '.gitignore', '.prettierrc', 'eslint.config.mjs', 'nest-cli.json', 'package-lock.json', 'package.json', 'README.md', 'tsconfig.build.json', and 'tsconfig.json'. The Terminal tab is selected, displaying a log of NestJS compilation and startup messages. The log includes multiple entries of '[16:34:50] Starting compilation in watch mode...', '[16:34:52] Found 0 errors. Watching for file changes.', and '[Nest] 16336 - 06/02/2025, 16:34:53 LOG [NestFactory] Starting Nest application...'. The status bar at the bottom of VS Code indicates the file is 'master*' and shows icons for Go Live and other development tools.

```
[16:34:50] Starting compilation in watch mode...
[16:34:52] Found 0 errors. Watching for file changes.
[16:34:50] Starting compilation in watch mode...
[16:34:52] Found 0 errors. Watching for file changes.
[Nest] 16336 - 06/02/2025, 16:34:53 LOG [NestFactory] Starting Nest application...
[16:34:50] Starting compilation in watch mode...
[16:34:52] Found 0 errors. Watching for file changes.
[Nest] 16336 - 06/02/2025, 16:34:53 LOG [NestFactory] Starting Nest application...
[16:34:50] Starting compilation in watch mode...
[16:34:52] Found 0 errors. Watching for file changes.
[Nest] 16336 - 06/02/2025, 16:34:53 LOG [NestFactory] Starting Nest application...
[Nest] 16336 - 06/02/2025, 16:34:53 LOG [InstanceLoader] AppModule dependencies initialized +8ms
[Nest] 16336 - 06/02/2025, 16:34:53 LOG [RoutesResolver] AppController {}/: +4ms
[Nest] 16336 - 06/02/2025, 16:34:53 LOG [RouterExplorer] Mapped {/, GET} route +2ms
[Nest] 16336 - 06/02/2025, 16:34:53 LOG [NestApplication] Nest application successfully started +2ms
```

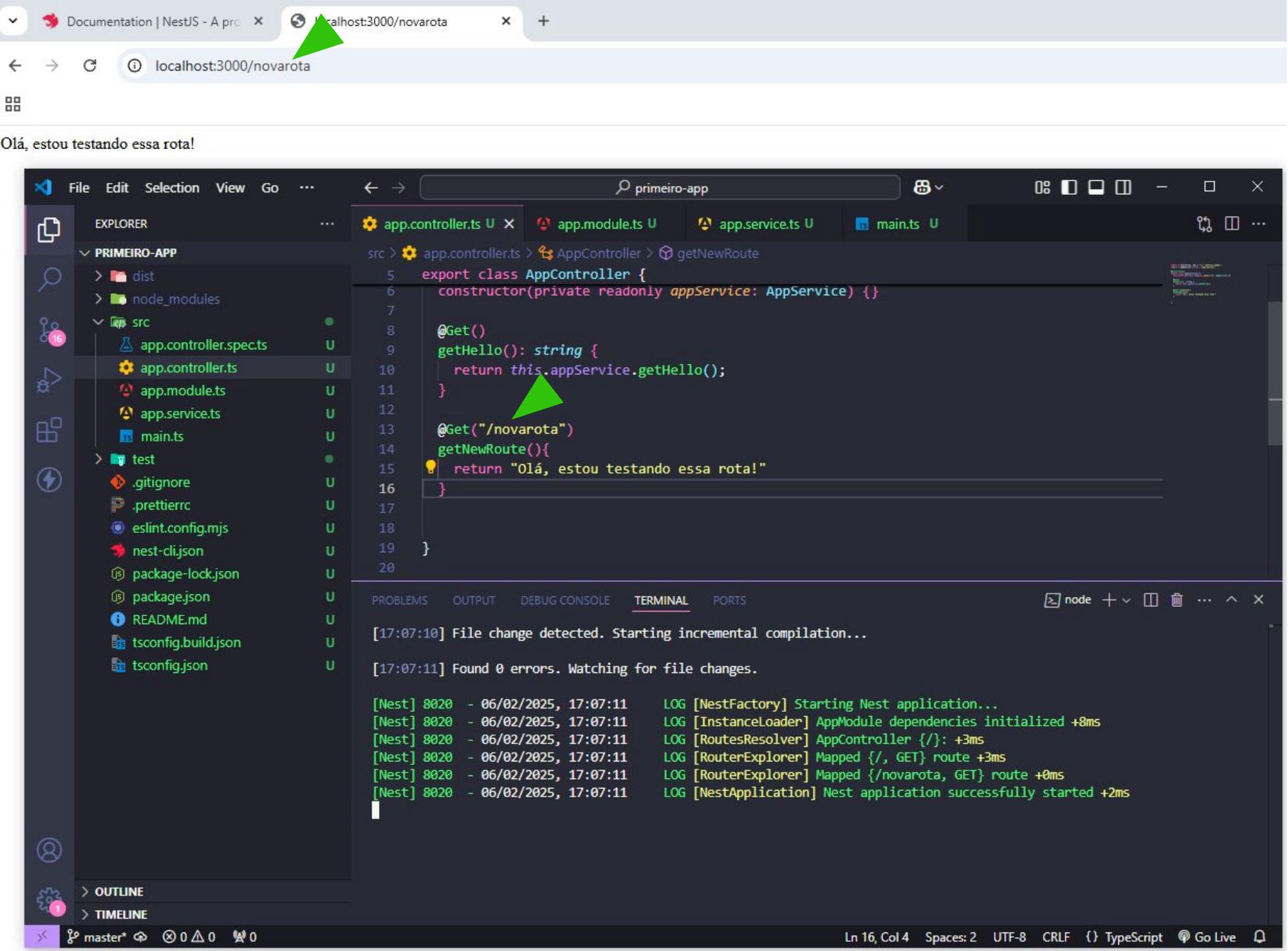
Estrutura do Projeto



The screenshot shows the VS Code interface with the following details:

- File Explorer (Left):** Shows the project structure under "PRIMEIRO-APP". The "src" folder is expanded, displaying files: app.controller.spec.ts, app.controller.ts, app.module.ts, app.service.ts, and main.ts. A green arrow points to the "src" folder icon.
- Search Bar (Top):** Searches for "primeiro-app".
- Code Editor (Center):** Displays the content of main.ts.

```
1 import { NestFactory } from '@nestjs/core';
2 import { AppModule } from './app.module';
3
4 //Arquivo que inicia o nosso projeto
5 async function bootstrap() {
6     const app = await NestFactory.create(AppModule);
7     await app.listen(process.env.PORT ?? 3000);
8 }
9 bootstrap();
10
```
- Taskbar (Top Right):** Shows tabs for app.controller.spec.ts, app.controller.ts, app.module.ts, app.service.ts, and main.ts. The main.ts tab is active.



The screenshot shows a browser window at `localhost:3000/novarota` displaying the message "Olá, estou testando essa rota!" (Hello, I'm testing this route!). A green arrow points from this message to the corresponding line in the `app.controller.ts` file in the VS Code editor.

The VS Code editor displays the `app.controller.ts` file with the following code:

```
export class AppController {
  constructor(private readonly appService: AppService) {}

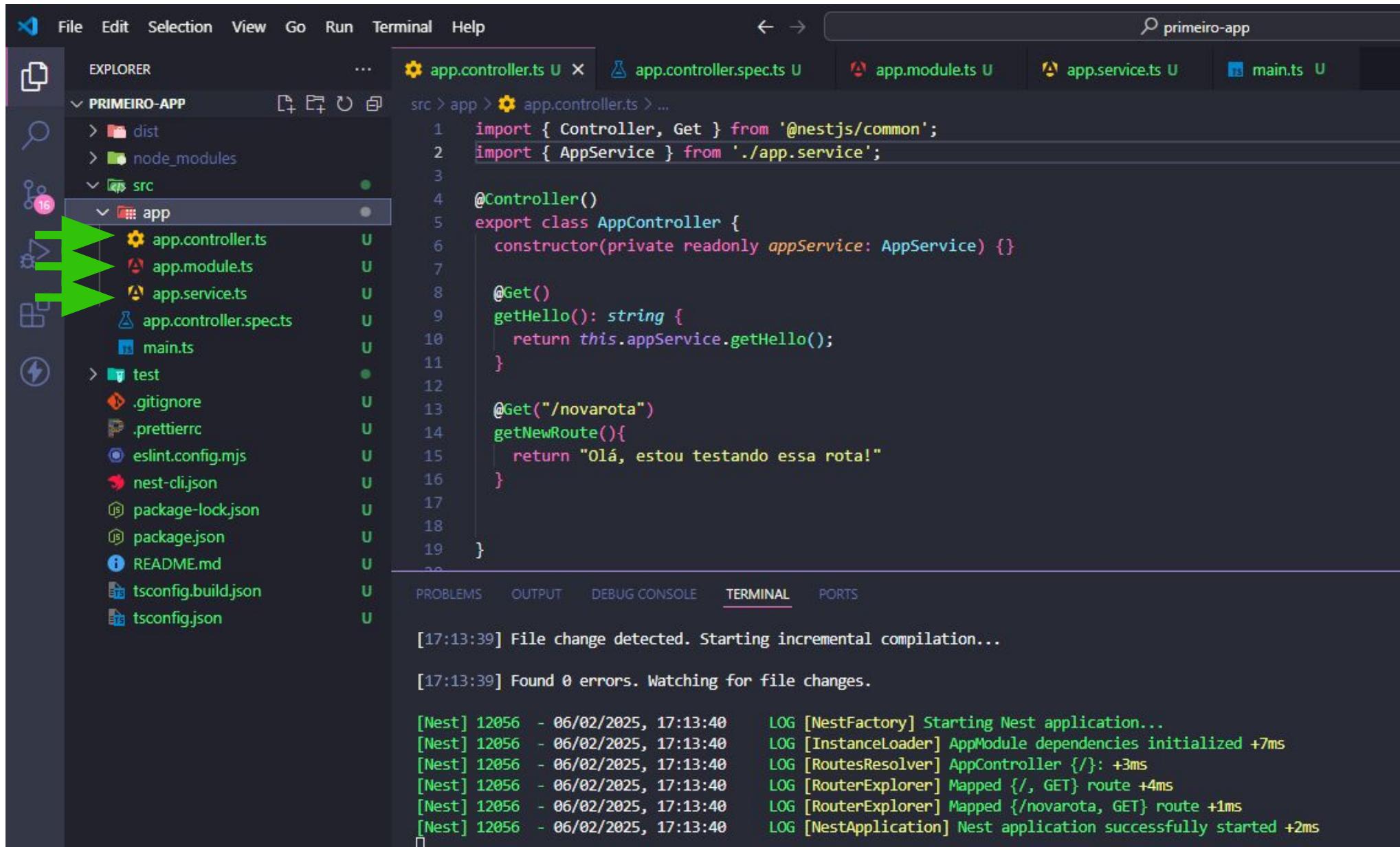
  @Get()
  getHello(): string {
    return this.appService.getHello();
  }

  @Get("/novarota")
  getNewRoute(){
    return "Olá, estou testando essa rota!"
  }
}
```

The terminal output shows the application starting and logs for the new route:

```
[17:07:10] File change detected. Starting incremental compilation...
[17:07:11] Found 0 errors. Watching for file changes.

[Nest] 8020 - 06/02/2025, 17:07:11   LOG [NestFactory] Starting Nest application...
[Nest] 8020 - 06/02/2025, 17:07:11   LOG [InstanceLoader] AppModule dependencies initialized +8ms
[Nest] 8020 - 06/02/2025, 17:07:11   LOG [RoutesResolver] AppController {}/: +3ms
[Nest] 8020 - 06/02/2025, 17:07:11   LOG [RouterExplorer] Mapped {/, GET} route +3ms
[Nest] 8020 - 06/02/2025, 17:07:11   LOG [RouterExplorer] Mapped {/novarota, GET} route +0ms
[Nest] 8020 - 06/02/2025, 17:07:11   LOG [NestApplication] Nest application successfully started +2ms
```



The screenshot shows the VS Code interface with the following details:

- File Explorer (Left):** Shows the project structure under "PRIMEIRO-APP". The "src" folder contains "app" which has "app.controller.ts", "app.module.ts", "app.service.ts", "app.controller.spec.ts", and "main.ts". There are also "test", ".gitignore", ".prettierrc", "eslint.config.mjs", "nest-cli.json", "package-lock.json", "package.json", "README.md", "tsconfig.build.json", and "tsconfig.json". A green arrow highlights the "src" folder.
- Editor (Top Right):** The file "app.controller.ts" is open. The code is as follows:

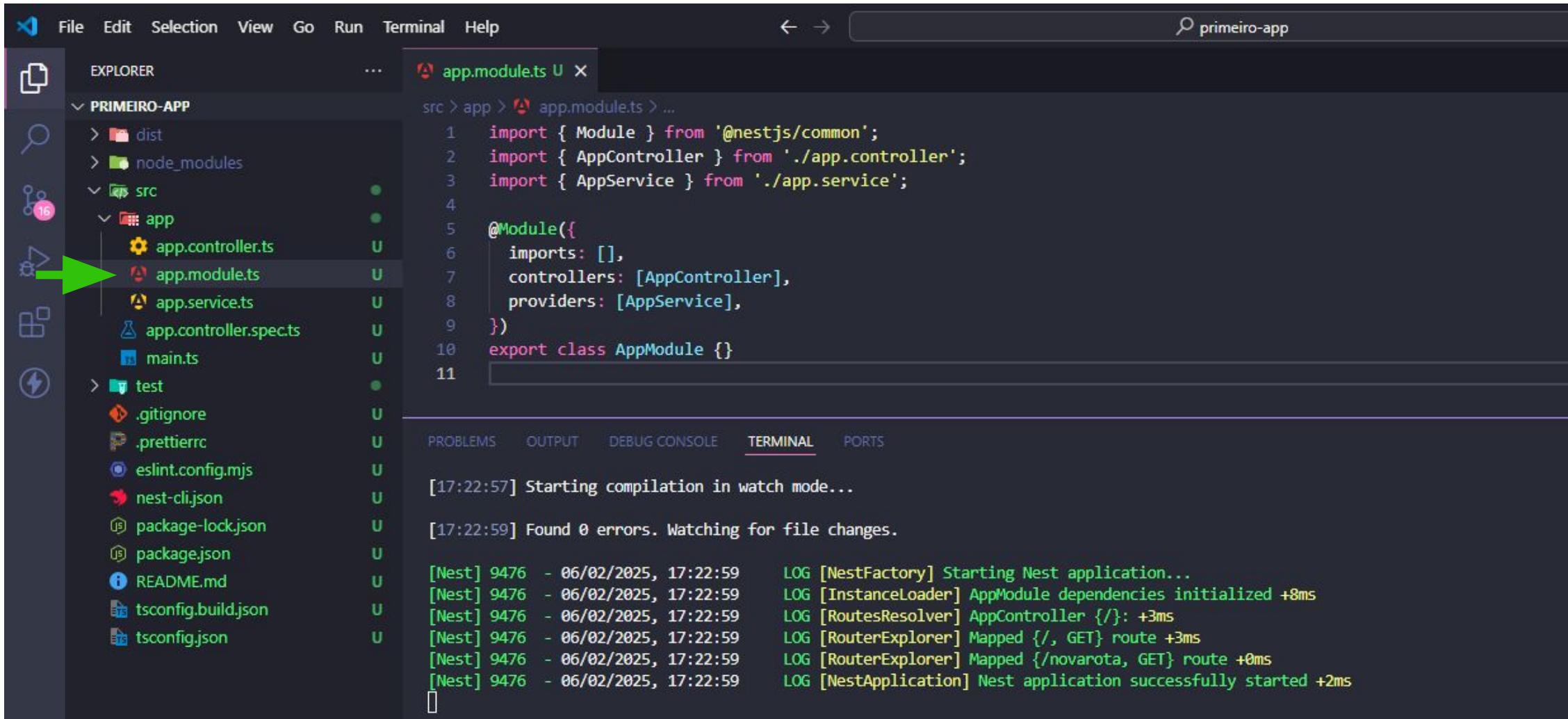
```
1 import { Controller, Get } from '@nestjs/common';
2 import { AppService } from './app.service';
3
4 @Controller()
5 export class AppController {
6   constructor(private readonly appService: AppService) {}
7
8   @Get()
9   getHello(): string {
10     return this.appService.getHello();
11   }
12
13   @Get("/novarota")
14   getNewRoute(){
15     return "Olá, estou testando essa rota!"
16   }
17
18 }
19 }
```

- Terminal (Bottom):** Shows the output of the application's log and compilation process.

```
[17:13:39] File change detected. Starting incremental compilation...
[17:13:39] Found 0 errors. Watching for file changes.

[Nest] 12056 - 06/02/2025, 17:13:40 LOG [NestFactory] Starting Nest application...
[Nest] 12056 - 06/02/2025, 17:13:40 LOG [InstanceLoader] AppModule dependencies initialized +7ms
[Nest] 12056 - 06/02/2025, 17:13:40 LOG [RoutesResolver] AppController {}/: +3ms
[Nest] 12056 - 06/02/2025, 17:13:40 LOG [RouterExplorer] Mapped {/, GET} route +4ms
[Nest] 12056 - 06/02/2025, 17:13:40 LOG [RouterExplorer] Mapped{/novarota, GET} route +1ms
[Nest] 12056 - 06/02/2025, 17:13:40 LOG [NestApplication] Nest application successfully started +2ms
```

Módulo Principal do Projeto



The screenshot shows a VS Code interface with the following details:

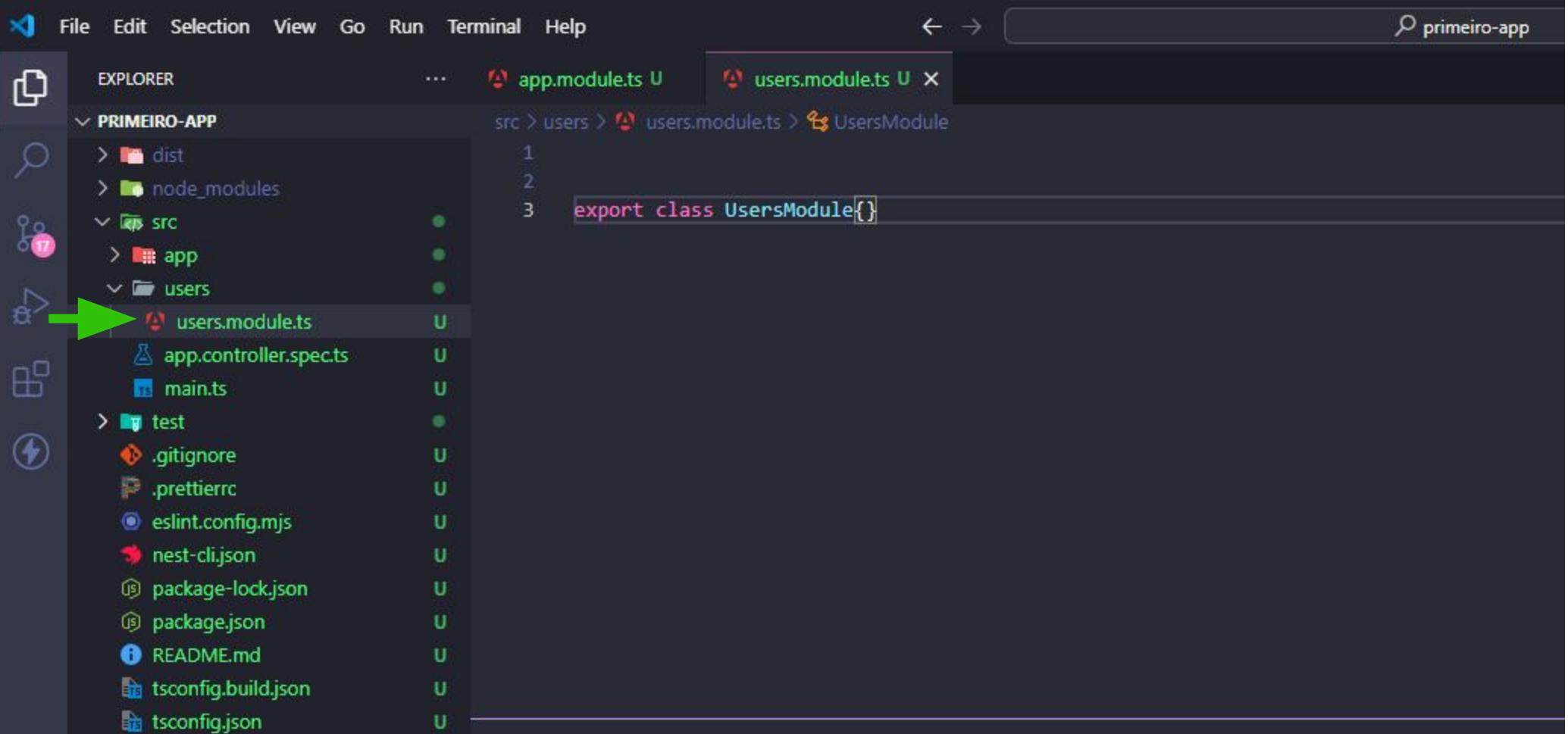
- File Explorer:** Shows the project structure under "PRIMEIRO-APP". The "app" folder contains "app.controller.ts", "app.module.ts", "app.service.ts", "app.controller.spec.ts", and "main.ts". A green arrow points to the "app.module.ts" file.
- Editor:** The "app.module.ts" file is open, displaying NestJS module configuration. It imports common, AppController, and AppService, defines a module with controllers and providers, and exports the AppModule class.
- Terminal:** The terminal shows the application starting in watch mode, finding 0 errors, and logging Nest application startup information.

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

```
[17:22:57] Starting compilation in watch mode...
[17:22:59] Found 0 errors. Watching for file changes.

[Nest] 9476 - 06/02/2025, 17:22:59    LOG [NestFactory] Starting Nest application...
[Nest] 9476 - 06/02/2025, 17:22:59    LOG [InstanceLoader] AppModule dependencies initialized +8ms
[Nest] 9476 - 06/02/2025, 17:22:59    LOG [RoutesResolver] AppController {/}: +3ms
[Nest] 9476 - 06/02/2025, 17:22:59    LOG [RouterExplorer] Mapped {/, GET} route +3ms
[Nest] 9476 - 06/02/2025, 17:22:59    LOG [RouterExplorer] Mapped {/novarota, GET} route +0ms
[Nest] 9476 - 06/02/2025, 17:22:59    LOG [NestApplication] Nest application successfully started +2ms
```

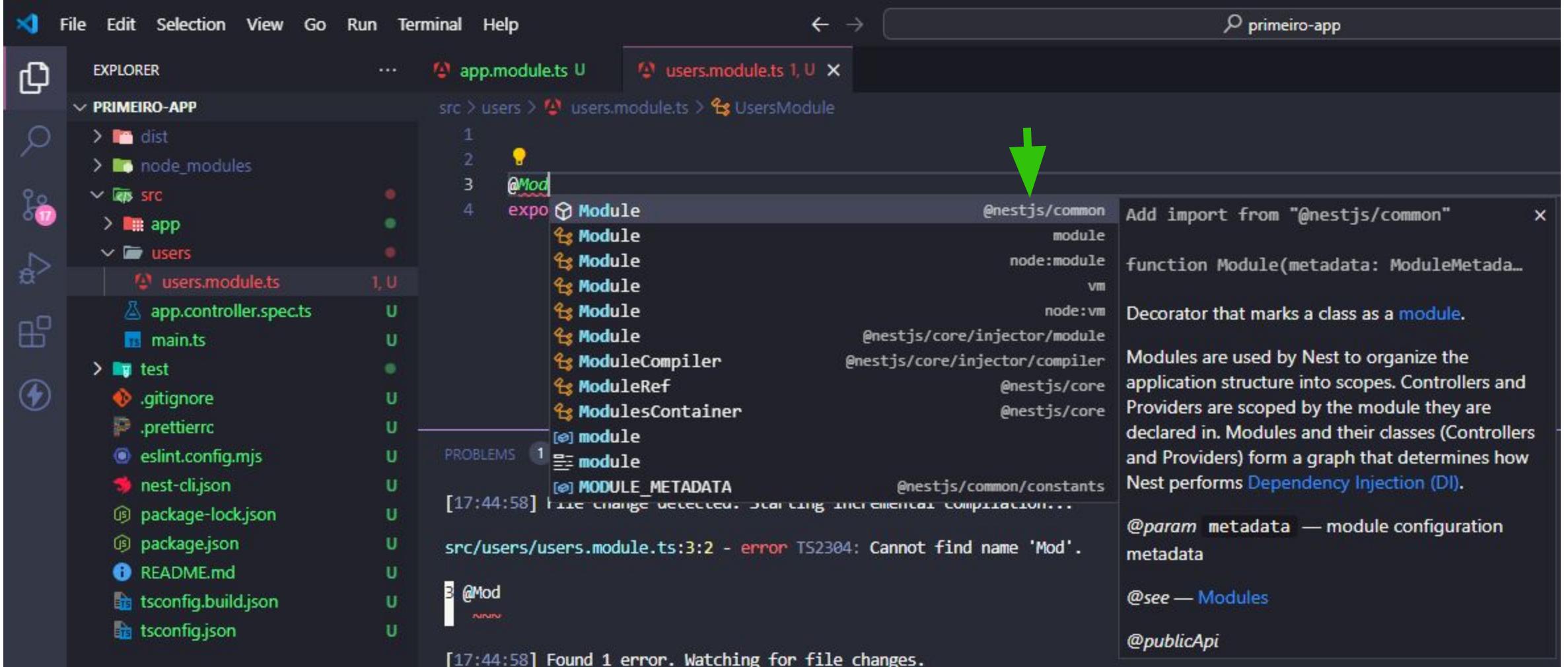
Criando um Módulo



The screenshot shows the Visual Studio Code interface with the title bar "primeiro-app". The Explorer sidebar on the left displays the project structure under "PRIMEIRO-APP": "dist", "node_modules", "src" (which contains "app" and "users" folders). The "users" folder is expanded, showing files like "app.controller.spec.ts", "main.ts", and "test" folder, along with configuration files ".gitignore", ".prettierrc", "eslint.config.mjs", "nest-cli.json", and package files "package-lock.json", "package.json", "README.md", "tsconfig.build.json", and "tsconfig.json". A green arrow points to the "users.module.ts" file in the "users" folder. The "app.module.ts" and "users.module.ts" tabs are open in the main editor area. The code in "users.module.ts" is:

```
1
2
3 export class UsersModule{}
```

Criando um Módulo



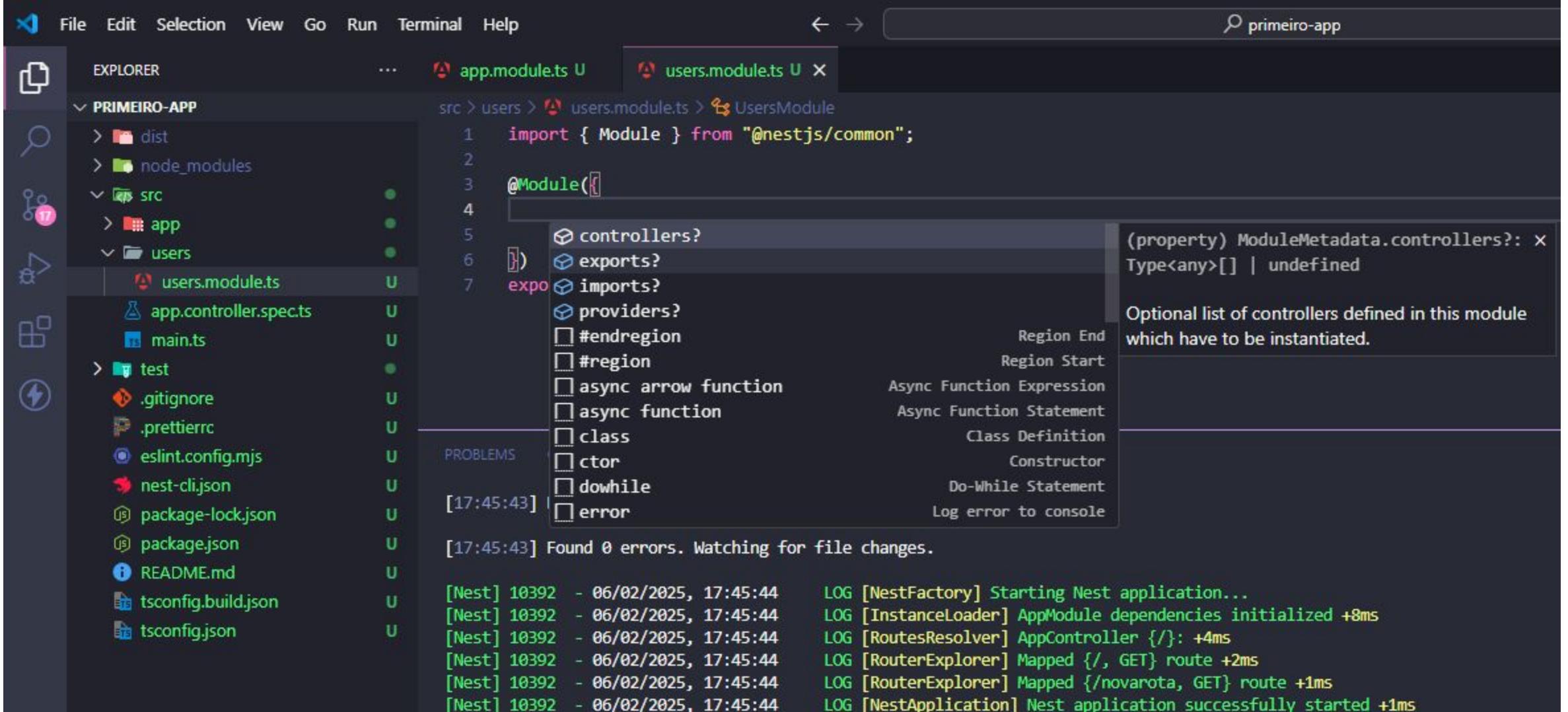
The screenshot shows a VS Code interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Search Bar:**primeiro-app
- Explorer:** PRIMEIRO-APP folder structure:
 - dist
 - node_modules
 - src
 - app
 - users
 - users.module.ts (selected)
 - app.controller.spec.ts
 - main.ts
 - test
 - .gitignore
 - .prettierrc
 - eslint.config.mjs
 - nest-cli.json
 - package-lock.json
 - package.json
 - README.md
 - tsconfig.build.json
 - tsconfig.json
- Editor:** users.module.ts (1, U)
Content:

```
1
2
3 @Mod
4 expo
```

A green arrow points from the word "Mod" in the code to the tooltip.- IntelliSense:** A tooltip for the `@Mod` decorator is displayed:
 - Add import from "@nestjs/common"
 - function Module(metadata: ModuleMetadata): void
 - Decorator that marks a class as a [Module](#).
 - Modules are used by Nest to organize the application structure into scopes. Controllers and Providers are scoped by the [module](#) they are declared in. Modules and their classes (Controllers and Providers) form a [graph](#) that determines how Nest performs [Dependency Injection \(DI\)](#).
 - `@param metadata — module configuration metadata`
 - `@see — Modules`
 - `@publicApi`
- Problems:** 1 [17:44:58] File change detected. Starting incremental compilation...
- Bottom Status Bar:** [17:44:58] Found 1 error. Watching for file changes.

Criando um Módulo



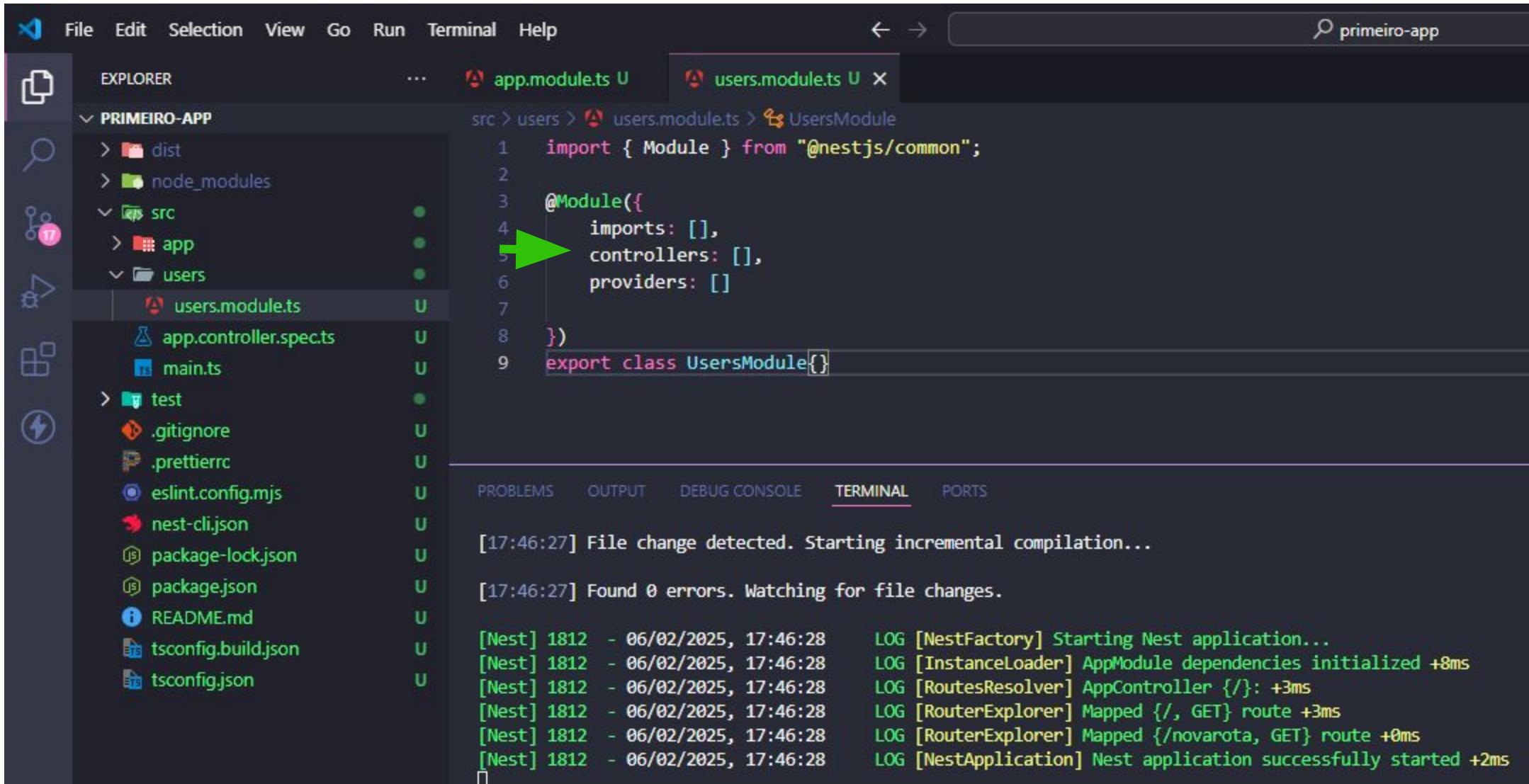
The screenshot shows a VS Code interface with the following details:

- File Explorer:** Shows the project structure under "PRIMEIRO-APP".
- Editor:** The file "users.module.ts" is open. The code defines a module named "UsersModule". A tooltip is displayed over the "controllers?" field, providing documentation for the optional list of controllers.
- Terminal:** Shows the application starting up with logs from NestFactory, InstanceLoader, RoutesResolver, RouterExplorer, and NestApplication.
- Bottom Status Bar:** Shows the message "Found 0 errors. Watching for file changes."

```
import { Module } from '@nestjs/common';
@Module({
  controllers?: [],
  exports?: [],
  imports?: [
    providers?
  ],
  #endregion
  #region
  async arrow function
  async function
  class
  ctor
  dowhile
  error
})
export class UsersModule {}
```

```
[Nest] 10392 - 06/02/2025, 17:45:44     LOG [NestFactory] Starting Nest application...
[Nest] 10392 - 06/02/2025, 17:45:44     LOG [InstanceLoader] AppModule dependencies initialized +8ms
[Nest] 10392 - 06/02/2025, 17:45:44     LOG [RoutesResolver] AppController {}/: +4ms
[Nest] 10392 - 06/02/2025, 17:45:44     LOG [RouterExplorer] Mapped {/, GET} route +2ms
[Nest] 10392 - 06/02/2025, 17:45:44     LOG [RouterExplorer] Mapped {/novarota, GET} route +1ms
[Nest] 10392 - 06/02/2025, 17:45:44     LOG [NestApplication] Nest application successfully started +1ms
```

Criando um Módulo



The screenshot shows a VS Code interface with the following details:

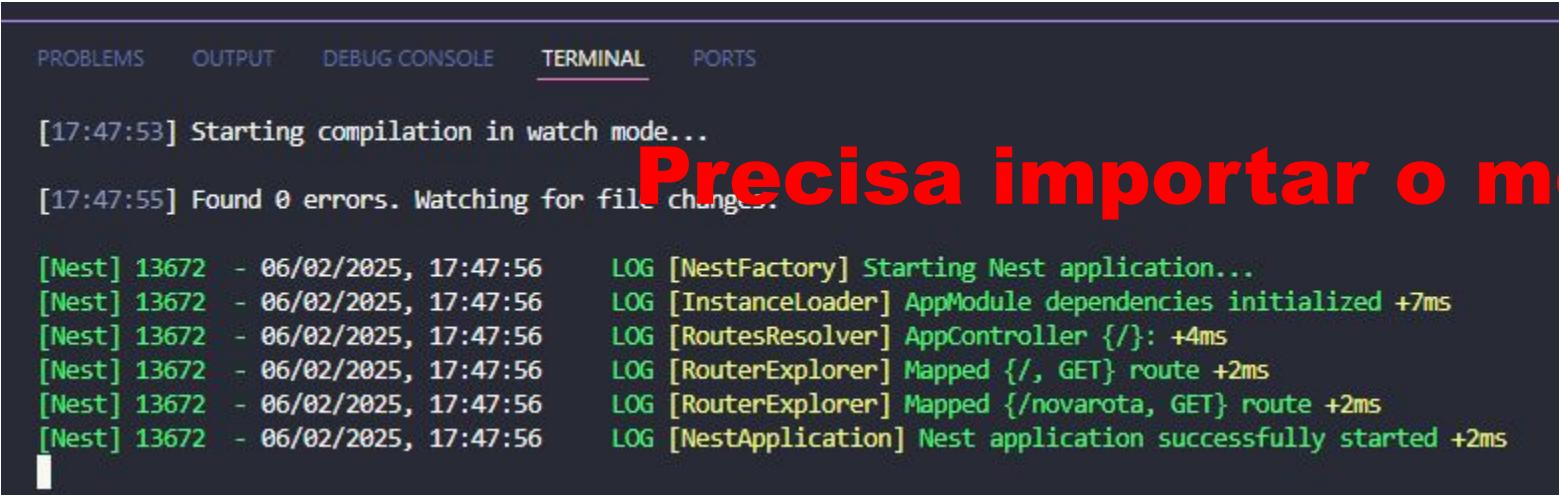
- File Explorer (Left):** Shows the project structure under "PRIMEIRO-APP". The "src" folder contains "app" and "users" folders. "users" contains "users.module.ts", "app.controller.spec.ts", and "main.ts".
- Code Editor (Center):** Displays "users.module.ts" with the following code:

```
import { Module } from '@nestjs/common';
@Module({
  imports: [],
  controllers: [],
  providers: []
})
export class UsersModule{}
```

A green arrow points to the "@Module" decorator.
- Terminal (Bottom):** Shows the output of the application's startup logs:

```
[17:46:27] File change detected. Starting incremental compilation...
[17:46:27] Found 0 errors. Watching for file changes.
[Nest] 1812 - 06/02/2025, 17:46:28    LOG [NestFactory] Starting Nest application...
[Nest] 1812 - 06/02/2025, 17:46:28    LOG [InstanceLoader] AppModule dependencies initialized +8ms
[Nest] 1812 - 06/02/2025, 17:46:28    LOG [RoutesResolver] AppController {/}: +3ms
[Nest] 1812 - 06/02/2025, 17:46:28    LOG [RouterExplorer] Mapped {/, GET} route +3ms
[Nest] 1812 - 06/02/2025, 17:46:28    LOG [RouterExplorer] Mapped {/novarota, GET} route +0ms
[Nest] 1812 - 06/02/2025, 17:46:28    LOG [NestApplication] Nest application successfully started +2ms
```

Criando um Módulo

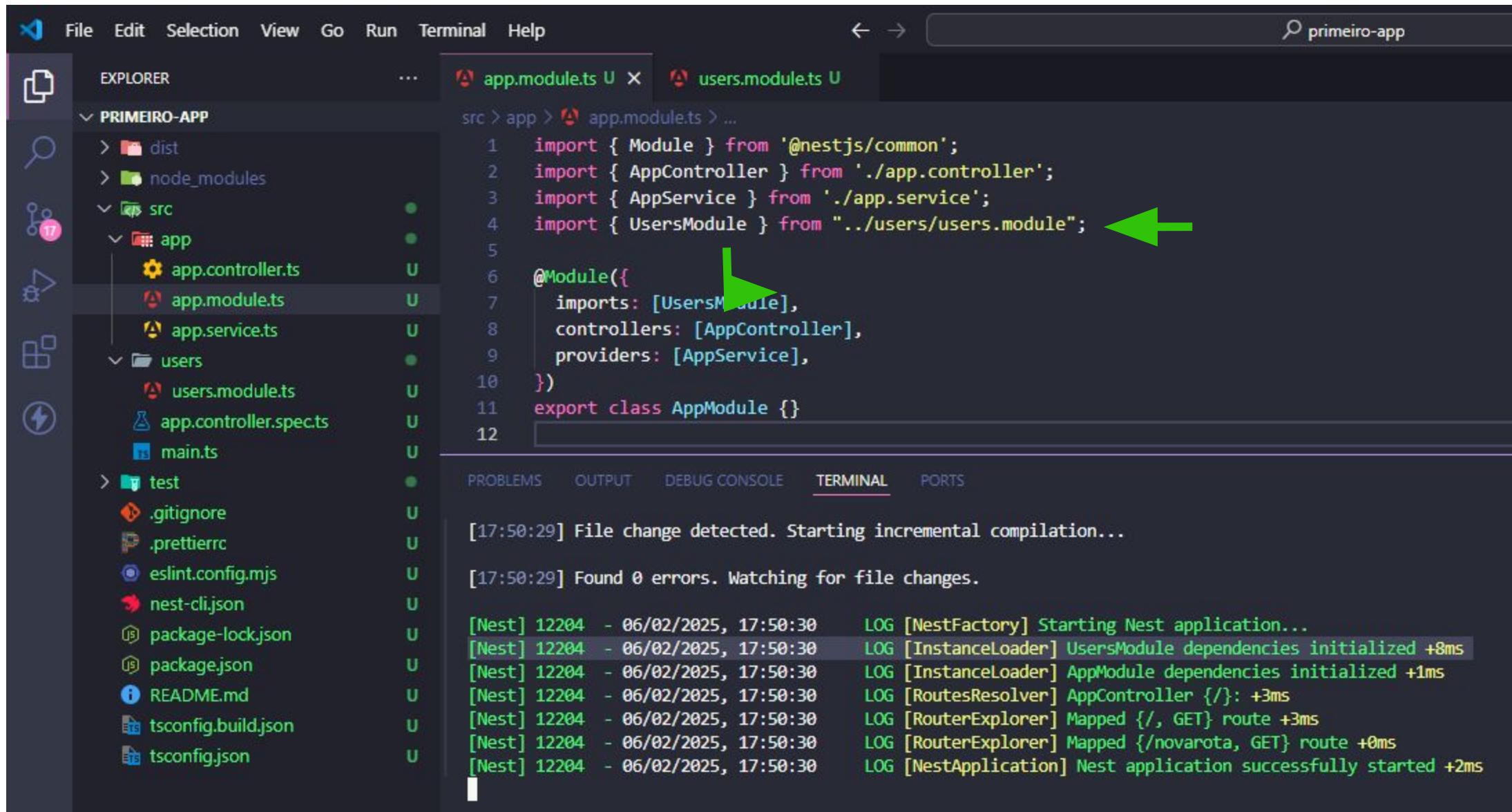


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
[17:47:53] Starting compilation in watch mode...
[17:47:55] Found 0 errors. Watching for file changes...
[Nest] 13672 - 06/02/2025, 17:47:56    LOG [NestFactory] Starting Nest application...
[Nest] 13672 - 06/02/2025, 17:47:56    LOG [InstanceLoader] AppModule dependencies initialized +7ms
[Nest] 13672 - 06/02/2025, 17:47:56    LOG [RoutesResolver] AppController {/}: +4ms
[Nest] 13672 - 06/02/2025, 17:47:56    LOG [RouterExplorer] Mapped {/, GET} route +2ms
[Nest] 13672 - 06/02/2025, 17:47:56    LOG [RouterExplorer] Mapped {/novarota, GET} route +2ms
[Nest] 13672 - 06/02/2025, 17:47:56    LOG [NestApplication] Nest application successfully started +2ms
```

Precisa importar o módulo users

Criando um Módulo



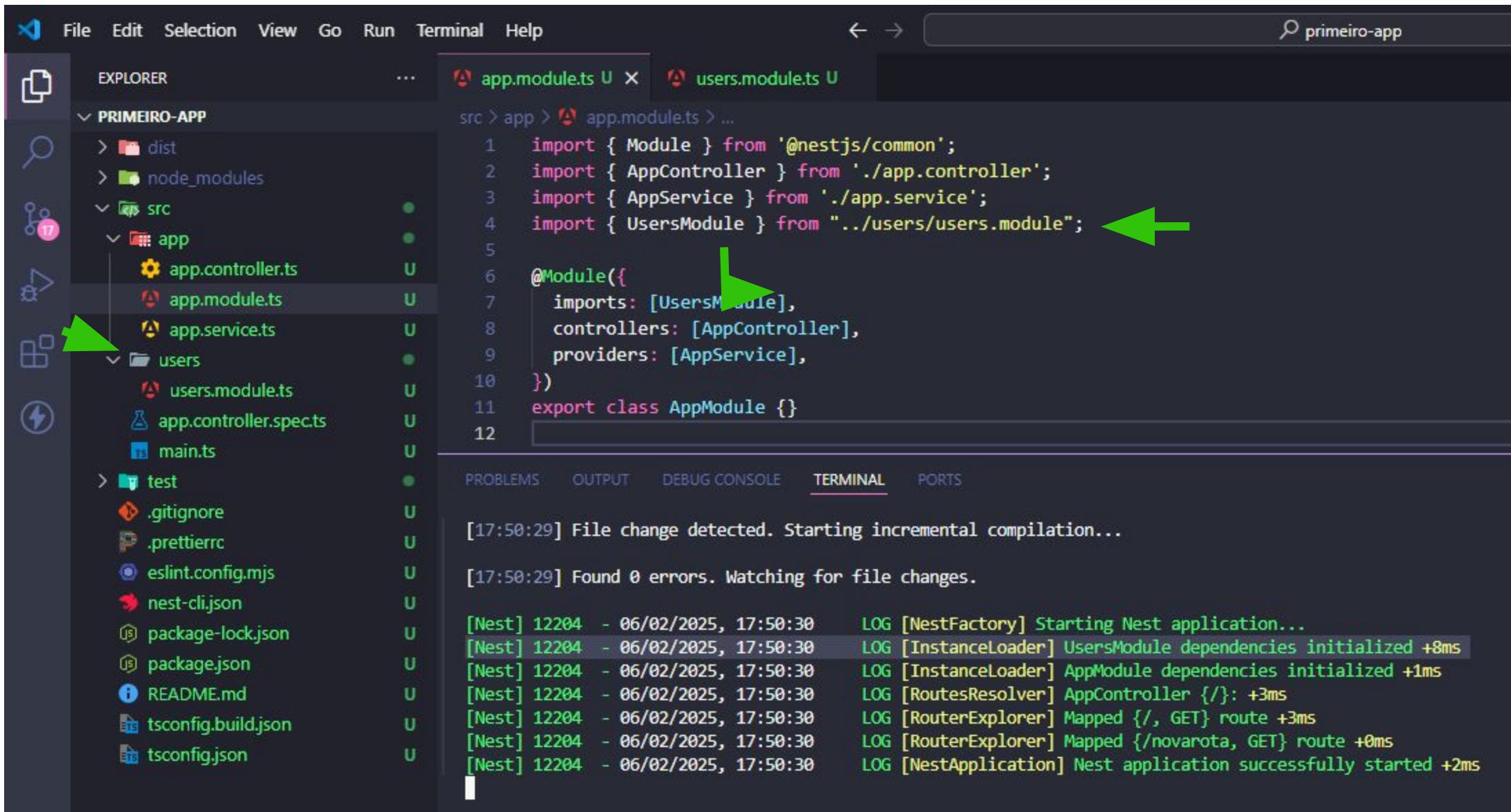
The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows the project structure under "PRIMEIRO-APP". The "src/app/app.module.ts" file is selected.
- Editor:** Displays the code for `app.module.ts`. A green arrow points to the line `import { UsersModule } from "../users/users.module";`.
- Terminal:** Shows the output of the application's startup logs, indicating successful module loading and application start.

```
src > app > app.module.ts > ...
1  import { Module } from '@nestjs/common';
2  import { AppController } from './app.controller';
3  import { AppService } from './app.service';
4  import { UsersModule } from "../users/users.module";
5
6  @Module({
7    imports: [UsersModule],
8    controllers: [AppController],
9    providers: [AppService],
10   })
11 export class AppModule {}
```

[17:50:29] File change detected. Starting incremental compilation...
[17:50:29] Found 0 errors. Watching for file changes.
[Nest] 12204 - 06/02/2025, 17:50:30 LOG [NestFactory] Starting Nest application...
[Nest] 12204 - 06/02/2025, 17:50:30 LOG [InstanceLoader] UsersModule dependencies initialized +8ms
[Nest] 12204 - 06/02/2025, 17:50:30 LOG [InstanceLoader] AppModule dependencies initialized +1ms
[Nest] 12204 - 06/02/2025, 17:50:30 LOG [RoutesResolver] AppController {/}: +3ms
[Nest] 12204 - 06/02/2025, 17:50:30 LOG [RouterExplorer] Mapped {/, GET} route +3ms
[Nest] 12204 - 06/02/2025, 17:50:30 LOG [RouterExplorer] Mapped {/novarota, GET} route +0ms
[Nest] 12204 - 06/02/2025, 17:50:30 LOG [NestApplication] Nest application successfully started +2ms

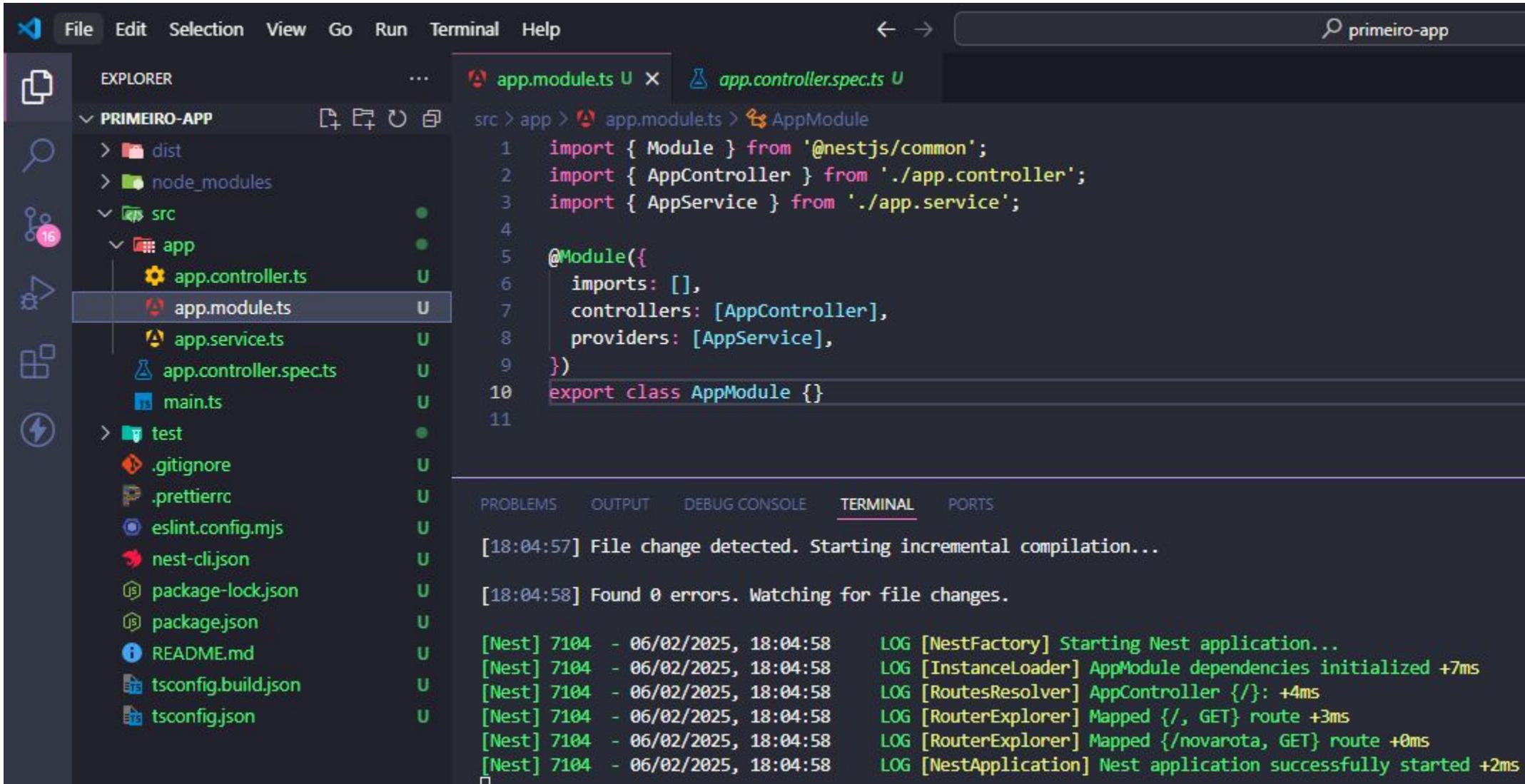
Excluir as setas verdes



The screenshot shows the VS Code interface with the following details:

- File Explorer (EXPLORER):** Shows the project structure under "PRIMEIRO-APP". The "src" folder is expanded, showing "app", "users", and other files.
- Code Editor:** Displays two files: "app.module.ts" and "users.module.ts".
 - "app.module.ts": Contains code for the AppModule, including imports for Module, AppController, AppService, and UsersModule, and a module definition with providers for UsersController and AppService.
 - "users.module.ts": Contains code for the UsersModule, including imports for Module and UsersModule, and a module definition with providers for AppController and AppService.
- Terminal:** Shows the output of the application's startup process, including logs from NestFactory, InstanceLoader, RoutesResolver, RouterExplorer, and NestApplication.

Criando um Módulo



The screenshot shows a VS Code interface with the following details:

- File Explorer:** Shows the project structure under "PRIMEIRO-APP". The "src/app" folder contains "app.controller.ts", "app.module.ts", "app.service.ts", and "main.ts".
- Editor:** The "app.module.ts" file is open, displaying the following code:

```
src > app > app.module.ts > AppModule
1 import { Module } from '@nestjs/common';
2 import { AppController } from './app.controller';
3 import { AppService } from './app.service';

@Module({
  imports: [],
  controllers: [AppController],
  providers: [AppService],
})
export class AppModule {}
```

- Terminal:** Shows the output of the application's startup process:

```
[18:04:57] File change detected. Starting incremental compilation...
[18:04:58] Found 0 errors. Watching for file changes.
[Nest] 7104 - 06/02/2025, 18:04:58    LOG [NestFactory] Starting Nest application...
[Nest] 7104 - 06/02/2025, 18:04:58    LOG [InstanceLoader] AppModule dependencies initialized +7ms
[Nest] 7104 - 06/02/2025, 18:04:58    LOG [RoutesResolver] AppController {/}: +4ms
[Nest] 7104 - 06/02/2025, 18:04:58    LOG [RouterExplorer] Mapped {/, GET} route +3ms
[Nest] 7104 - 06/02/2025, 18:04:58    LOG [RouterExplorer] Mapped {/novarota, GET} route +0ms
[Nest] 7104 - 06/02/2025, 18:04:58    LOG [NestApplication] Nest application successfully started +2ms
```

Criando um Módulo com CLI

File Edit Selection View Go Run Terminal Help

EXPLORER

PRIMEIRO-APP

- > dist
- > node_modules
- > src
 - app
 - app.controller.ts
 - app.module.ts
 - app.service.ts
 - app.controller.spec.ts
 - main.ts
 - test
 - .gitignore
 - .prettierrc
 - eslint.config.mjs
 - nest-cli.json
 - package-lock.json
 - package.json
 - README.md
 - tsconfig.build.json
 - tsconfig.json

src > app > app.module.ts > AppModule

```
1 import { Module } from '@nestjs/common';
2 import { AppController } from './app.controller';
3 import { AppService } from './app.service';
4
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Aggio\nestjs\primeiro-app> nest --help

Usage: nest <command> [options]

Options:

- v, --version Output the current version.
- h, --help Output usage information.

Commands:

- new[n [options]] [name] Generate Nest application.
- build [options] [apps...] Build Nest application.
- start [options] [app] Run Nest application.
- info|i Display Nest project details.
- add [options] <library> Adds support for an external library to your project.
- generate|g [options] <schematic> [name] [path] Generate a Nest element.

Schematics available on [@nestjs/schematics](#) collection:

name	alias	description
application	application	Generate a new application workspace
class	cl	Generate a new class
configuration	config	Generate a CLI configuration file
controller	co	Generate a controller declaration
decorator	d	Generate a custom decorator
filter	f	Generate a filter declaration
gateway	ga	Generate a gateway declaration
guard	gu	Generate a guard declaration
interceptor	itc	Generate an interceptor declaration
interface	itf	Generate an interface
library	lib	Generate a new library within a monorepo
middleware	mi	Generate a middleware declaration
module	mo	Generate a module declaration
pipe	pi	Generate a pipe declaration
provider	pr	Generate a provider declaration
resolver	r	Generate a GraphQL resolver declaration
resource	res	Generate a new CRUD resource
service	s	Generate a service declaration
sub-app	app	Generate a new application within a monorepo

PS C:\Aggio\nestjs\primeiro-app> nest g module tasks

Criando um Módulo com CLI

The image shows a screenshot of the Visual Studio Code (VS Code) interface. The title bar includes the application name "primeiro-app". The left sidebar is the "EXPLORER" view, showing the project structure under "PRIMEIRO-APP". A green arrow points from the "tasks.module.ts" file in the "src/tasks" folder. The main editor area displays the following code:

```
1 import { Module } from '@nestjs/common';
2
3 @Module({})
4 export class TasksModule {}
```

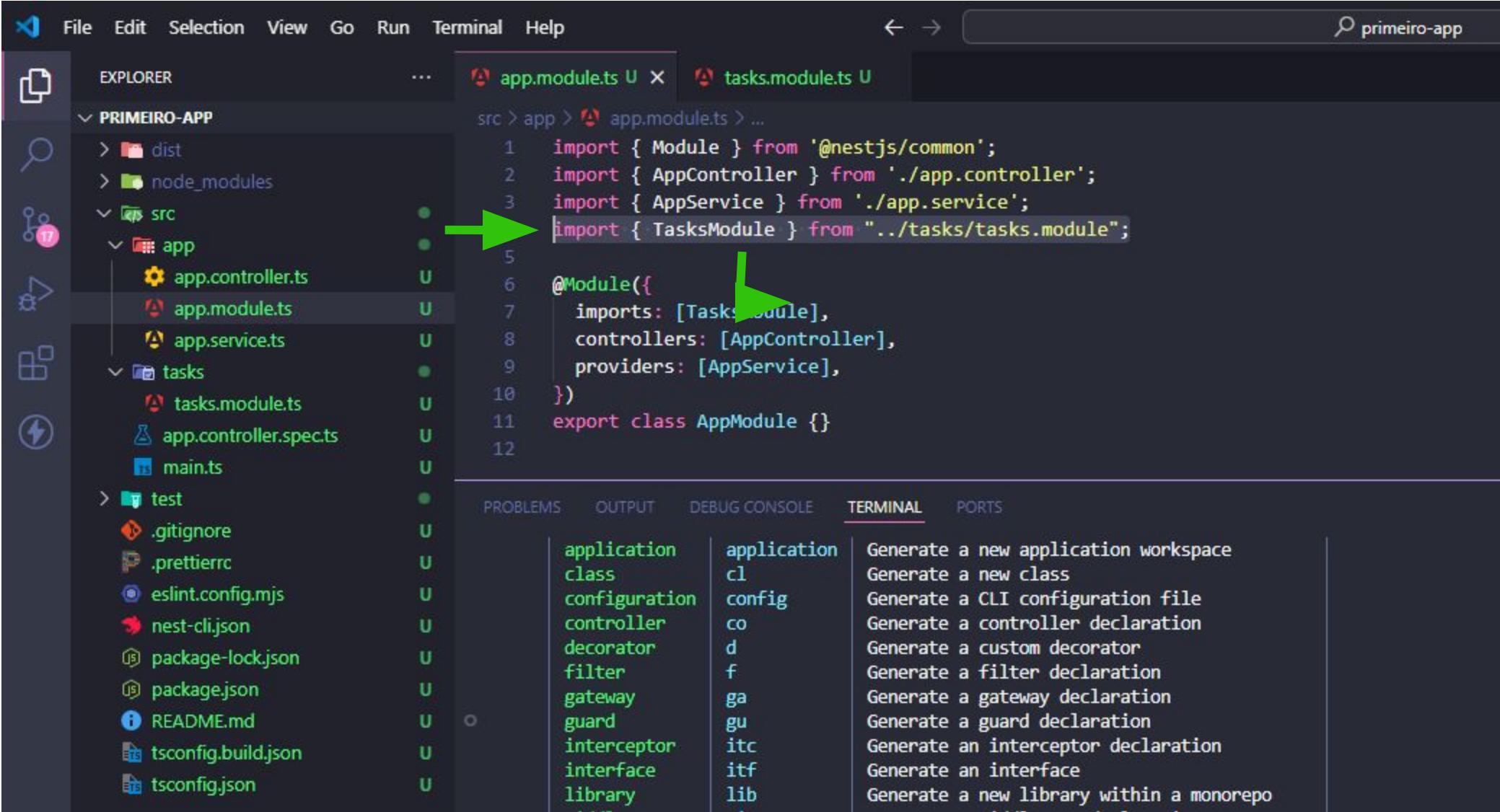
The "TERMINAL" tab is active at the bottom, showing the command "nest g module tasks" being run. The terminal output is:

```
PS C:\Aggio\nestjs\primeiro-app> nest g module tasks
CREATE src/tasks/tasks.module.ts (86 bytes)
```

A large callout box covers the right side of the screen, containing the Nest CLI help documentation for generating modules. It lists various commands like "new", "build", and "start", along with their descriptions. Below this, a table provides a mapping of common generator names to their aliases and descriptions.

name	alias	description
application	application	Generate a new application workspace
class	cl	Generate a new class
configuration	config	Generate a CLI configuration file
controller	co	Generate a controller declaration
decorator	d	Generate a custom decorator
filter	f	Generate a filter declaration
gateway	ga	Generate a gateway declaration
guard	gu	Generate a guard declaration
interceptor	itc	Generate an interceptor declaration
interface	itf	Generate an interface
library	lib	Generate a new library within a monorepo
middleware	mi	Generate a middleware declaration
module	mo	Generate a module declaration
pipe	pi	Generate a pipe declaration
provider	pr	Generate a provider declaration
resolver	r	Generate a GraphQL resolver declaration
resource	res	Generate a new CRUD resource
service	s	Generate a service declaration
sub-app	app	Generate a new application within a monorepo

PS C:\Aggio\nestjs\primeiro-app>



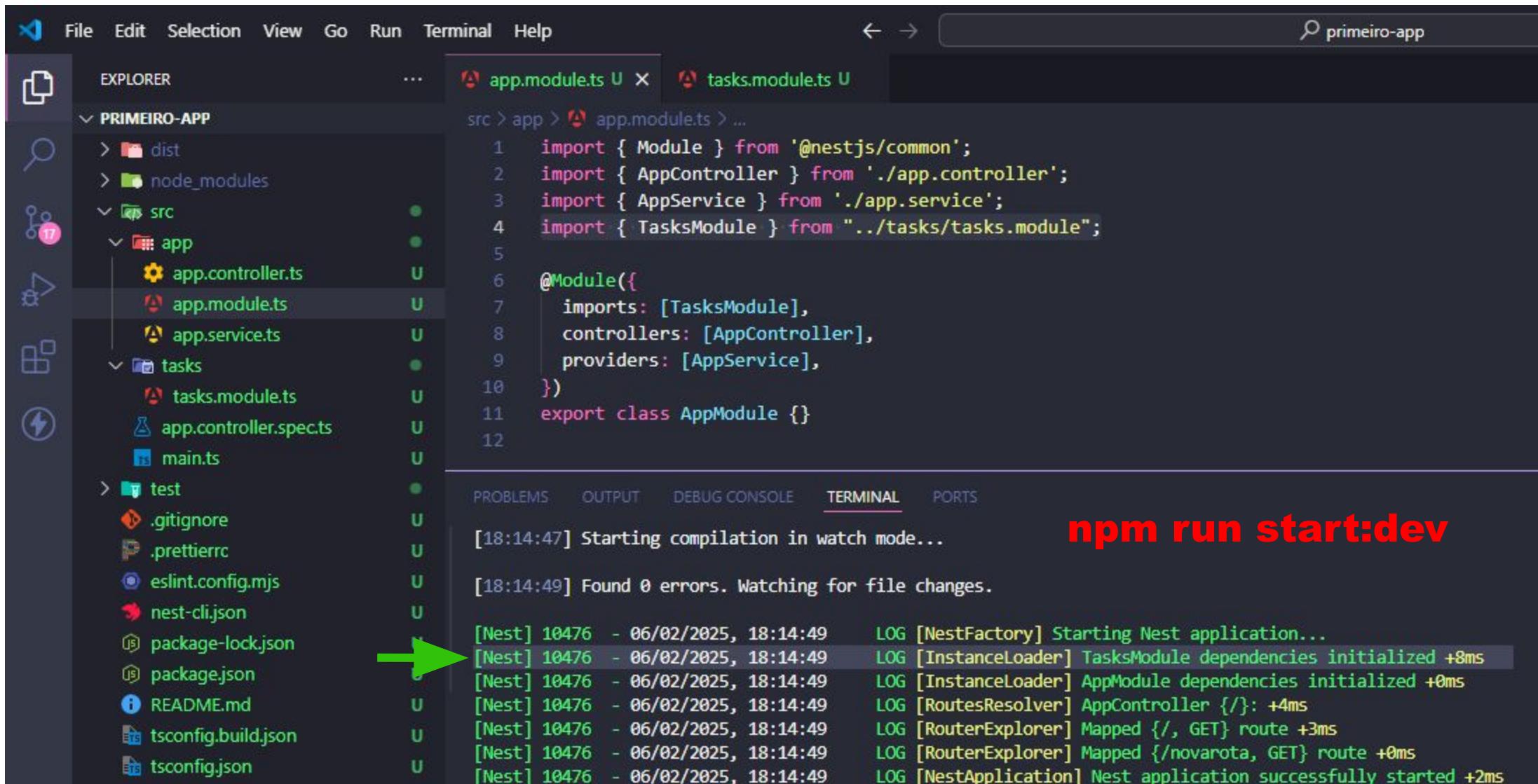
The screenshot shows the VS Code interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Search Bar:** primeiro-app
- Explorer:** Shows the project structure under PRIMEIRO-APP:
 - dist
 - node_modules
 - src
 - app
 - app.controller.ts
 - app.module.ts
 - app.service.ts
 - tasks
 - tasks.module.ts
 - app.controller.spec.ts
 - main.ts
 - test
 - .gitignore
 - .prettierrc
 - eslint.config.mjs
 - nest-cli.json
 - package-lock.json
 - package.json
 - README.md
 - tsconfig.build.json
 - tsconfig.json
- Code Editor:** The app.module.ts file is open, showing the following code:

```
1 import { Module } from '@nestjs/common';
2 import { AppController } from './app.controller';
3 import { AppService } from './app.service';
4 import { TasksModule } from '../tasks/tasks.module';
5
6 @Module({
7   imports: [TasksModule],
8   controllers: [AppController],
9   providers: [AppService],
10 })
11 export class AppModule {}
```

A green arrow points to the line "import { TasksModule } from '../tasks/tasks.module';". Another green arrow points to the line "export class AppModule {}".
- Terminal:** A dropdown menu is open with the following options:

application	application	Generate a new application workspace
class	cl	Generate a new class
configuration	config	Generate a CLI configuration file
controller	co	Generate a controller declaration
decorator	d	Generate a custom decorator
filter	f	Generate a filter declaration
gateway	ga	Generate a gateway declaration
guard	gu	Generate a guard declaration
interceptor	itc	Generate an interceptor declaration
interface	itf	Generate an interface
library	lib	Generate a new library within a monorepo



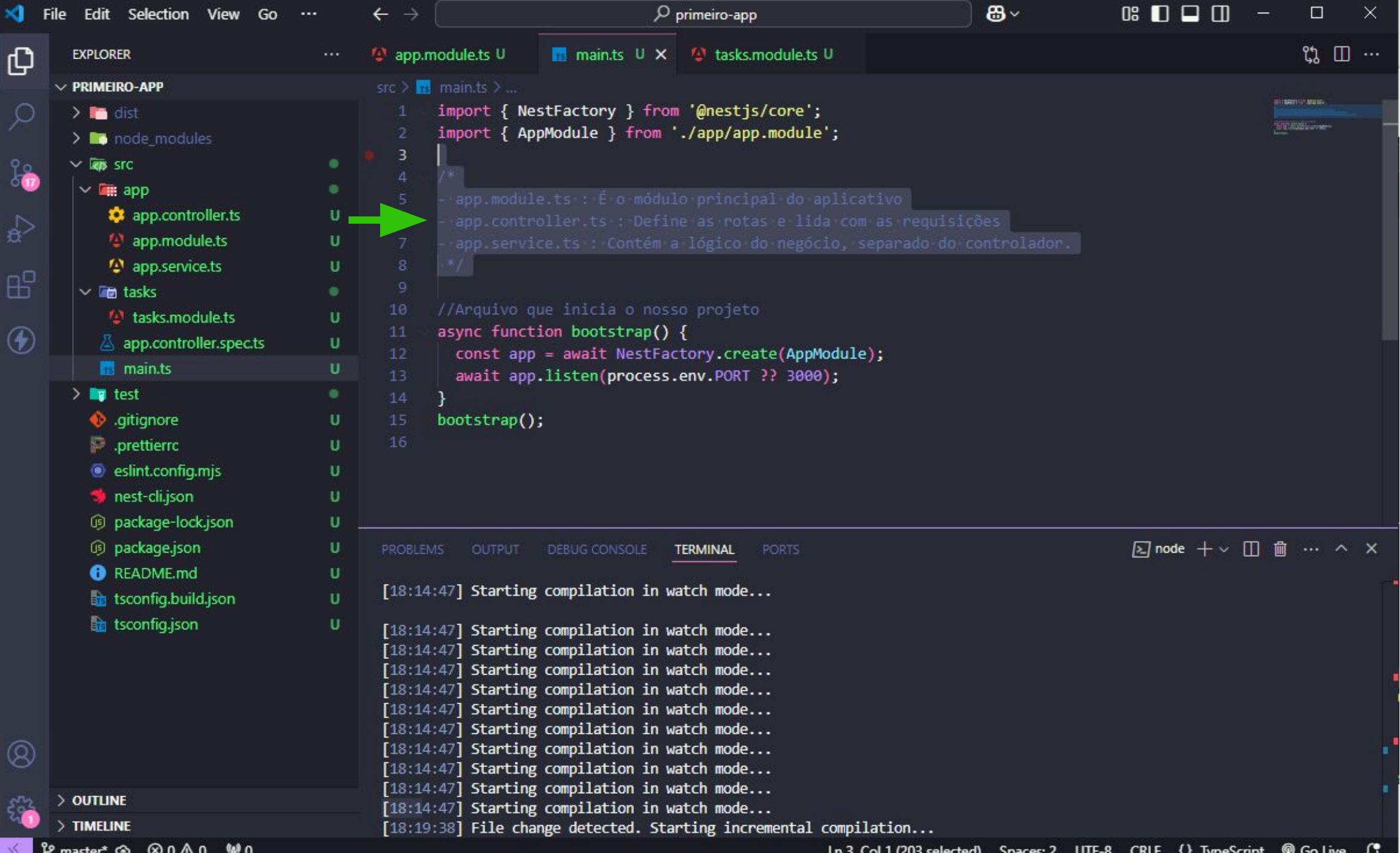
The screenshot shows a VS Code interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Search Bar:** primeiro-app
- Explorer:** Shows the project structure under 'PRIMEIRO-APP'. The 'src' folder contains 'app' (with app.controller.ts, app.module.ts, app.service.ts), 'tasks' (with tasks.module.ts, app.controller.spec.ts, main.ts), and 'test' (with .gitignore, .prettierrc, eslint.config.mjs, nest-cli.json, package-lock.json, package.json, README.md, tsconfig.build.json, tsconfig.json). There are 17 untracked files indicated by a red circle with the number 17.
- Code Editor:** Displays the content of 'app.module.ts'. The code imports 'Module' from '@nestjs/common', 'AppController' from './app.controller', 'AppService' from './app.service', and 'TasksModule' from '../tasks/tasks.module'. It defines an @Module block with imports: [TasksModule], controllers: [AppController], providers: [AppService], and a single export class AppModule {}.
- Terminal:** Shows the command 'npm run start:dev' being run, followed by the application's startup logs:

```
[18:14:47] Starting compilation in watch mode...
[18:14:49] Found 0 errors. Watching for file changes.

[Nest] 10476 - 06/02/2025, 18:14:49    LOG [NestFactory] Starting Nest application...
[Nest] 10476 - 06/02/2025, 18:14:49    LOG [InstanceLoader] TasksModule dependencies initialized +8ms
[Nest] 10476 - 06/02/2025, 18:14:49    LOG [InstanceLoader] AppModule dependencies initialized +0ms
[Nest] 10476 - 06/02/2025, 18:14:49    LOG [RoutesResolver] AppController {/}: +4ms
[Nest] 10476 - 06/02/2025, 18:14:49    LOG [RouterExplorer] Mapped {/, GET} route +3ms
[Nest] 10476 - 06/02/2025, 18:14:49    LOG [RouterExplorer] Mapped {/novarota, GET} route +0ms
[Nest] 10476 - 06/02/2025, 18:14:49    LOG [NestApplication] Nest application successfully started +2ms
```

npm run start:dev



The screenshot shows the Visual Studio Code interface with the following details:

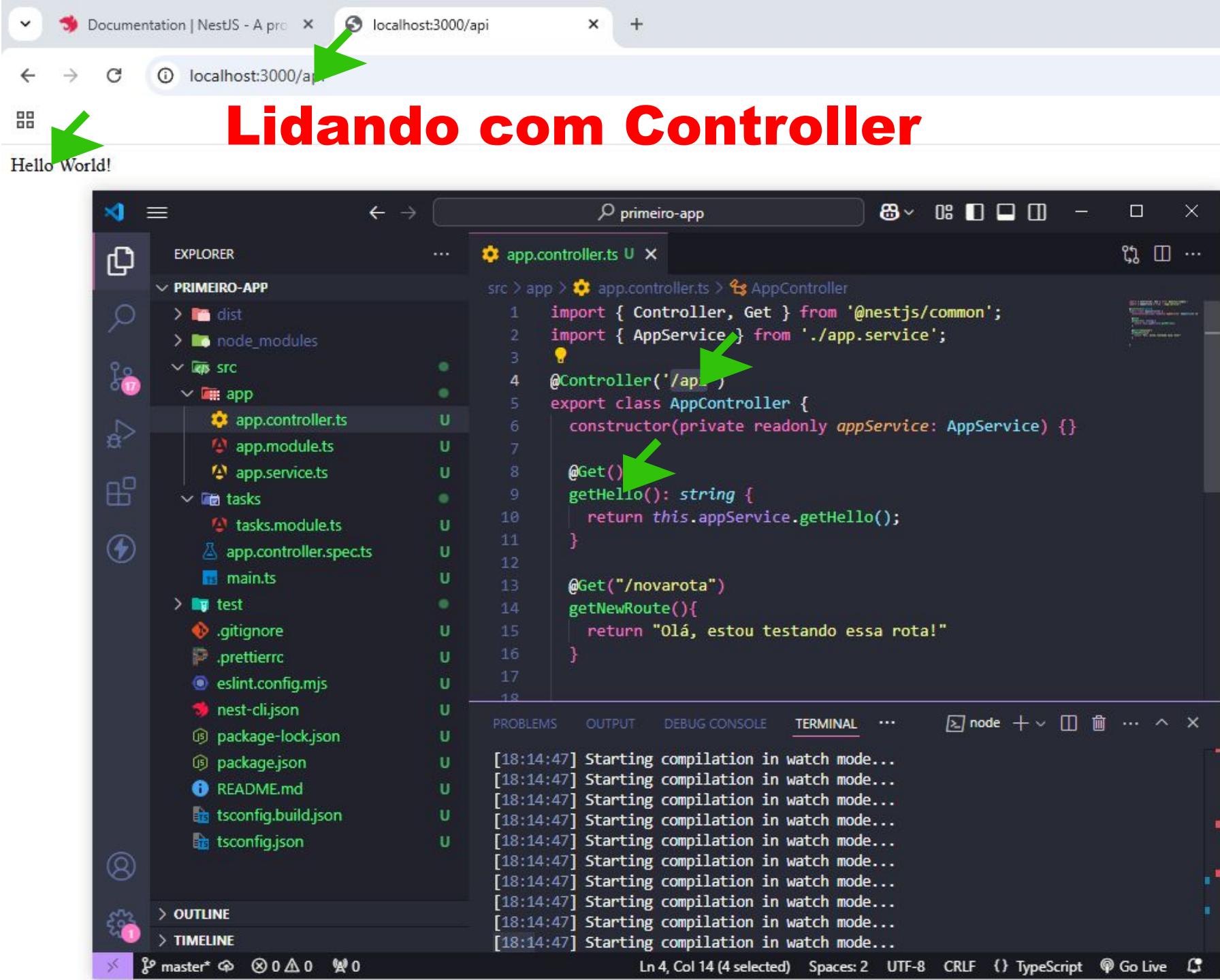
- File Menu:** File, Edit, Selection, View, Go, ...
- Search Bar:**primeiro-app
- Editor:** The main.ts file is open, showing NestJS application bootstrap code.
- Explorer Sidebar:** Shows the project structure under PRIMEIRO-APP, including src, app, tasks, test, and various configuration files.
- Terminal:** Displays log messages from the terminal window, indicating compilation in watch mode and starting incremental compilation after a file change.

```
import { NestFactory } from '@nestjs/core';
import { AppModule } from './app/app.module';

/*
- app.module.ts : É o módulo principal do aplicativo
- app.controller.ts : Define as rotas e lida com as requisições
- app.service.ts : Contém a lógica do negócio, separado do controlador.
*/
//Arquivo que inicia o nosso projeto
async function bootstrap() {
  const app = await NestFactory.create(AppModule);
  await app.listen(process.env.PORT ?? 3000);
}
```

[18:14:47] Starting compilation in watch mode...
[18:19:38] File change detected. Starting incremental compilation...

Ln 3, Col 1 (203 selected) Spaces: 2 UTF-8 CRLF {} TypeScript ⚡ Go Live



The screenshot shows a browser window at the top with the URL `localhost:3000/api` and a large red title "Lidando com Controller" below it. A green arrow points from the browser's address bar to the title. Another green arrow points from the browser's back button to the VS Code interface.

The VS Code interface displays the file `app.controller.ts` with the following code:

```
import { Controller, Get } from '@nestjs/common';
import { AppService } from './app.service';

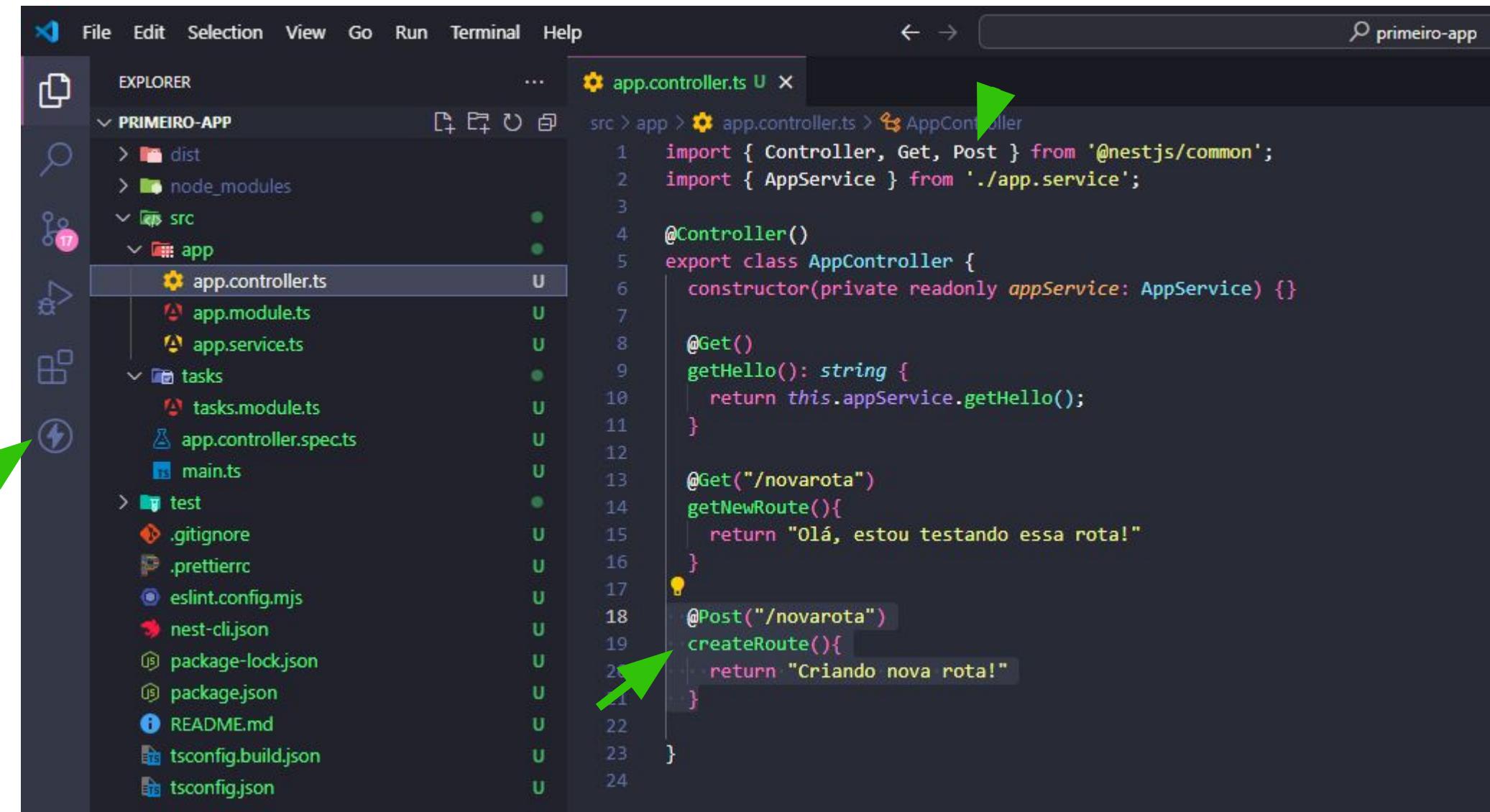
@Controller('/api')
export class AppController {
  constructor(private readonly appService: AppService) {}

  @Get()
  getHello(): string {
    return this.appService.getHello();
  }

  @Get("/novarota")
  getNewRoute(){
    return "Olá, estou testando essa rota!"
  }
}
```

The code includes imports for `@nestjs/common` and `AppService`, and defines a controller with two methods: `getHello()` and `getNewRoute()`.

Lidando com Controller

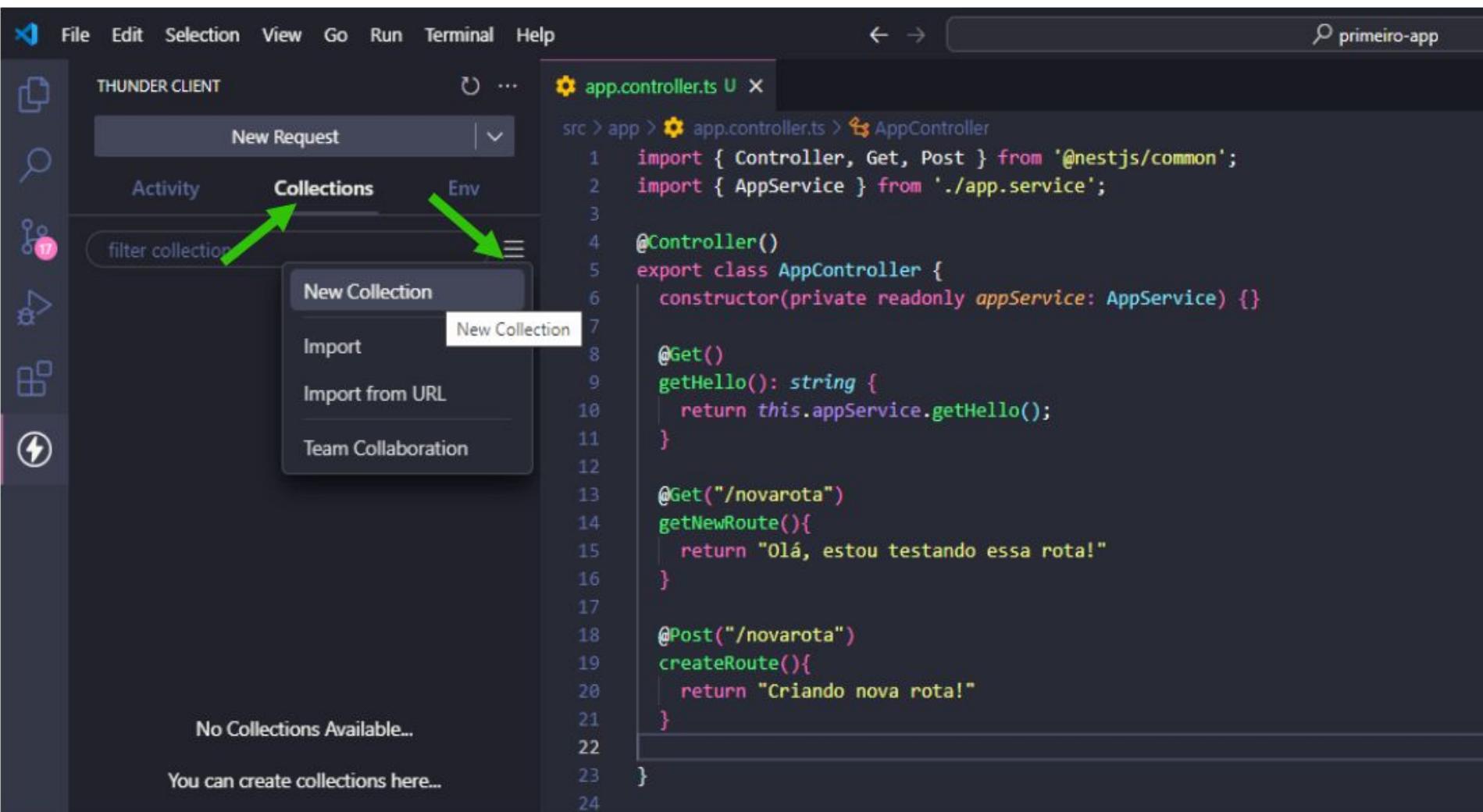


The screenshot shows the VS Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Search Bar:**primeiro-app
- Explorer Sidebar:** Shows the project structure under PRIMEIRO-APP, including dist, node_modules, src (with app, tasks, test), and various configuration files like .gitignore, .prettierrc, eslint.config.mjs, nest-cli.json, package-lock.json, package.json, README.md, tsconfig.build.json, and tsconfig.json. A green arrow points to the 'File' icon in the sidebar.
- Code Editor:** The current file is app.controller.ts, which contains NestJS controller logic. The code includes imports for Controller, Get, Post from '@nestjs/common' and AppService from './app.service'. It defines an AppController class with a constructor taking an appService parameter. It has two methods: a simple @Get() method returning 'Olá, estou testando essa rota!' and a @Post('/novarota') method returning 'Criando nova rota!'. A green arrow points to the code editor area.

```
src > app > app.controller.ts > AppController
1 import { Controller, Get, Post } from '@nestjs/common';
2 import { AppService } from './app.service';
3
4 @Controller()
5 export class AppController {
6   constructor(private readonly appService: AppService) {}
7
8   @Get()
9   getHello(): string {
10     return this.appService.getHello();
11   }
12
13   @Get("/novarota")
14   getNewRoute(){
15     return "Olá, estou testando essa rota!";
16   }
17
18   @Post("/novarota")
19   createRoute(){
20     return "Criando nova rota!";
21   }
22
23 }
24 }
```

Lidando com Controller

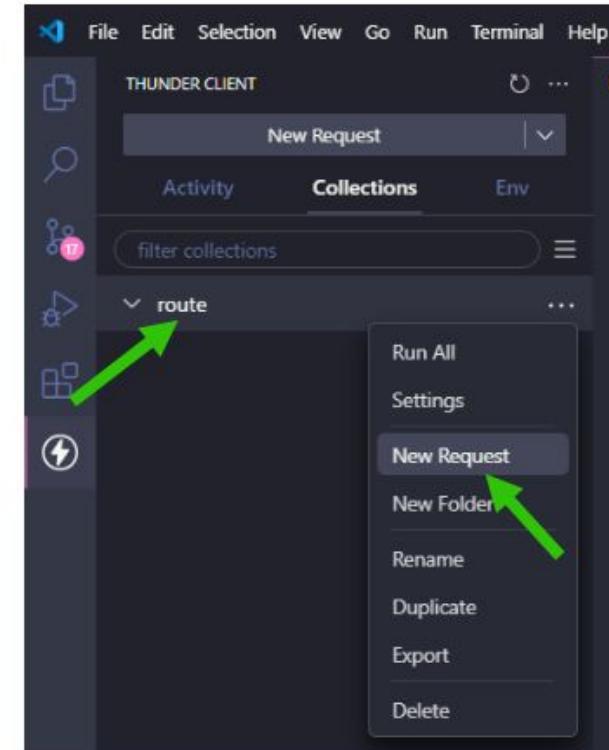


The screenshot shows the Thunder Client interface with the file `app.controller.ts` open. The code defines an `AppController` with methods for `@Get()` and `@Post("/novarota")`. A context menu is open over the word `route`, with the "New Collection" option highlighted. Two green arrows point from the "New Collection" button to the "Collections" tab in the top navigation bar and to the context menu item.

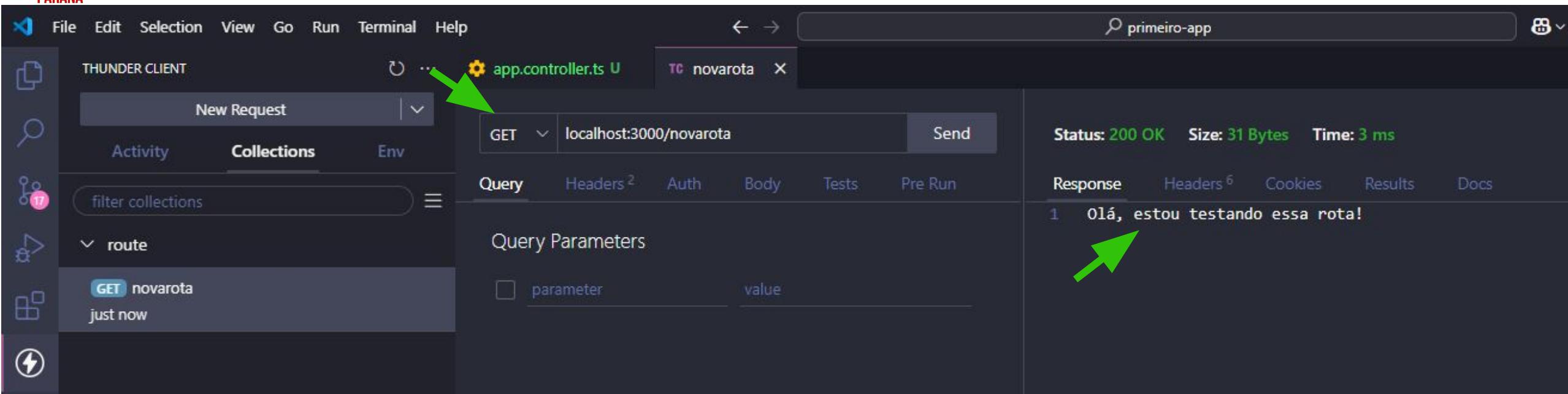
```
src > app > app.controller.ts > AppController
1 import { Controller, Get, Post } from '@nestjs/common';
2 import { AppService } from './app.service';
3
4 @Controller()
5 export class AppController {
6   constructor(private readonly appService: AppService) {}
7
8   @Get()
9   getHello(): string {
10     return this.appService.getHello();
11   }
12
13   @Get("/novarota")
14   getNewRoute(){
15     return "Olá, estou testando essa rota!"
16   }
17
18   @Post("/novarota")
19   createRoute(){
20     return "Criando nova rota!"
21   }
22 }
```

No Collections Available...

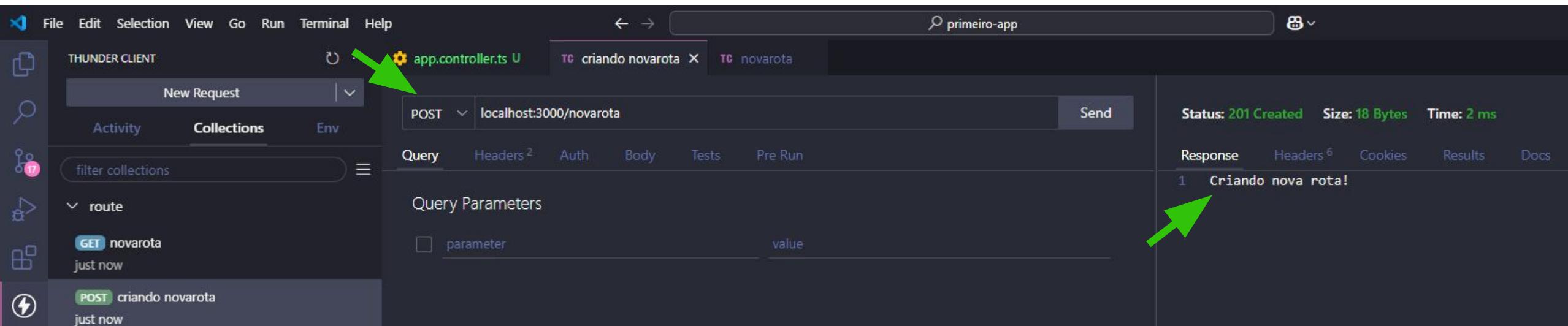
You can create collections here...



Lidando com Controller

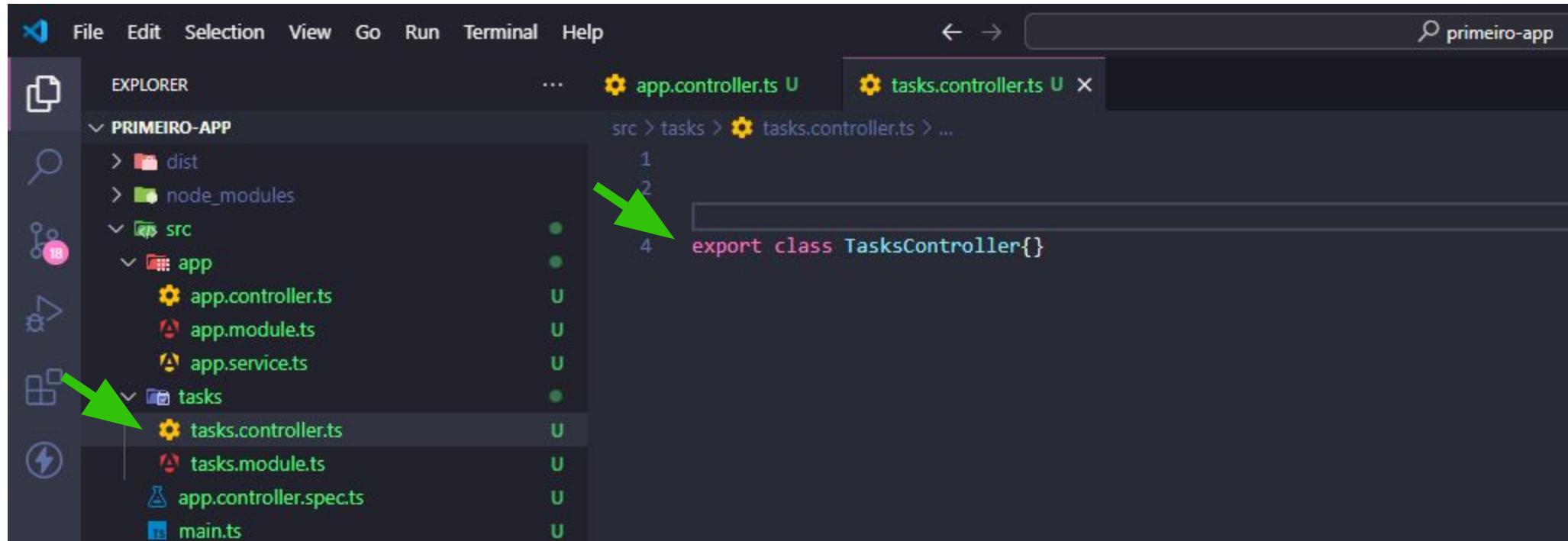


The screenshot shows the Thunder Client interface. In the top bar, the collection is set to "app.controller.ts" and the route is "novarota". A green arrow points to the "app.controller.ts" label. The main area displays a "New Request" configuration for a "GET" method to "localhost:3000/novarota". The response panel on the right shows a status of "200 OK", size of "31 Bytes", and time of "3 ms". The response body contains the message "Olá, estou testando essa rota!".



The screenshot shows the Thunder Client interface. In the top bar, the collection is set to "app.controller.ts" and the route is "novarota". A green arrow points to the "app.controller.ts" label. The main area displays a "New Request" configuration for a "POST" method to "localhost:3000/novarota". The response panel on the right shows a status of "201 Created", size of "18 Bytes", and time of "2 ms". The response body contains the message "Criando nova rota!".

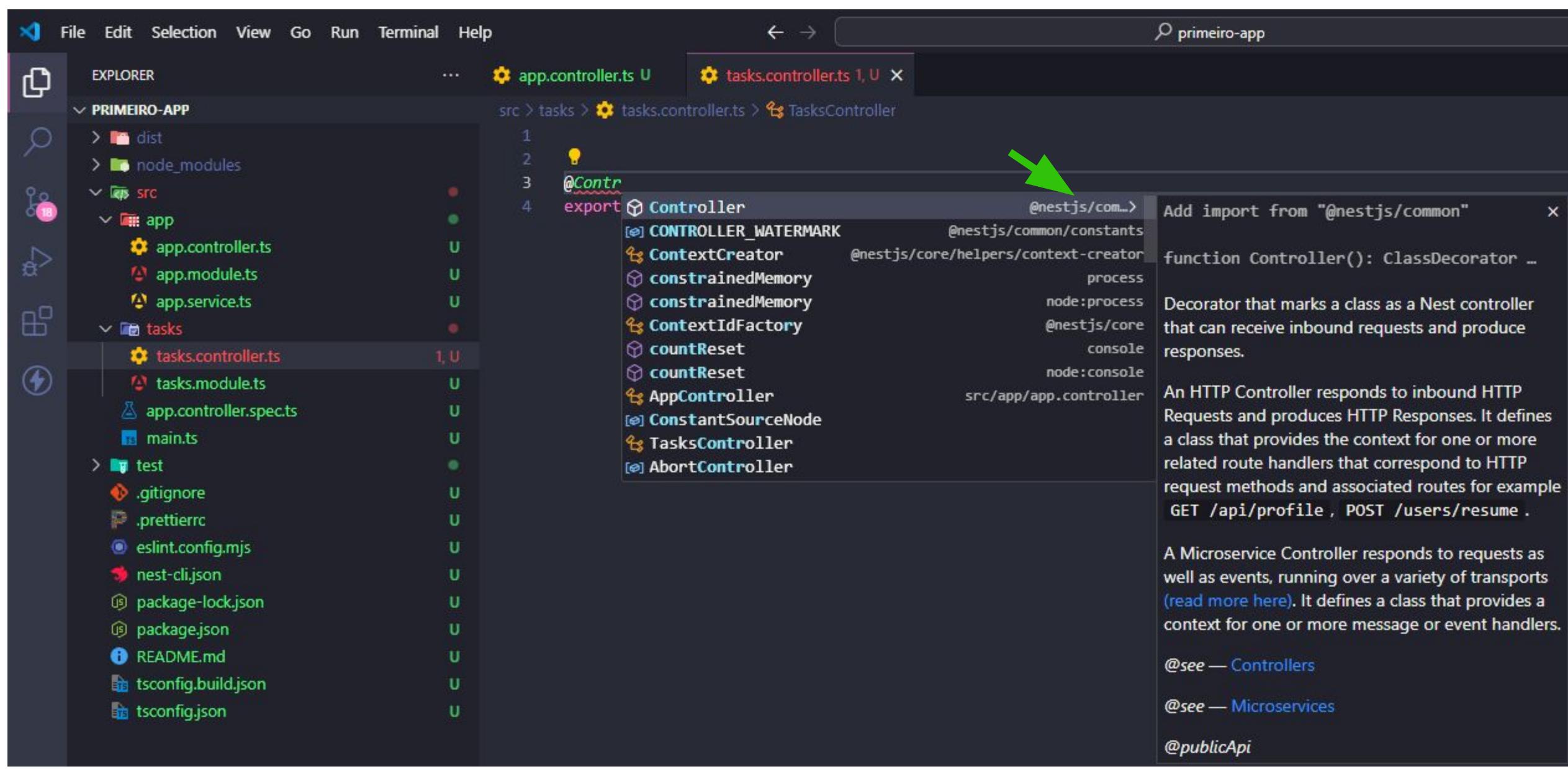
Gerando Controller



The screenshot shows a dark-themed interface of Visual Studio Code. In the top left, the menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The top right features a search bar with the text "primeiro-app". The left sidebar, titled "EXPLORER", displays a project structure under "PRIMEIRO-APP": "dist", "node_modules", "src" (which contains "app" and "tasks" folders). The "tasks" folder is expanded, showing files: "tasks.controller.ts", "tasks.module.ts", "app.controller.spec.ts", and "main.ts". A green arrow points from the "tasks" folder icon towards the "tasks.controller.ts" file in the main editor area. The editor tab shows the code for "tasks.controller.ts":

```
1
2
3
4 export class TasksController{}
```

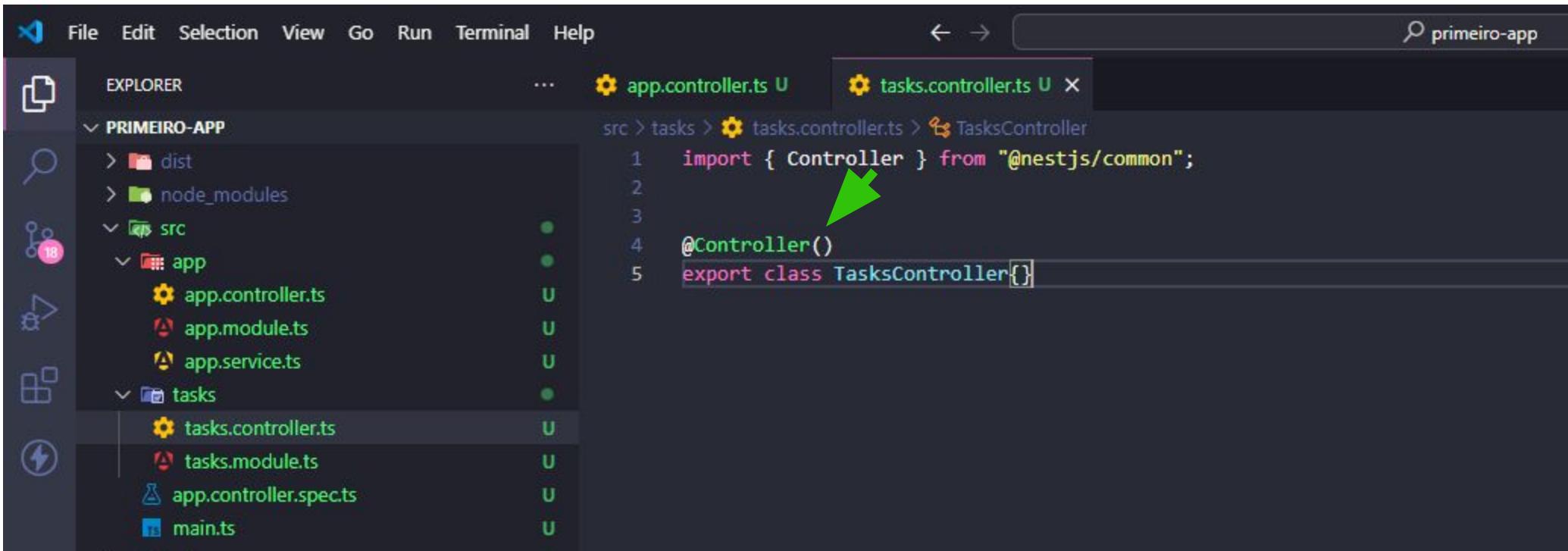
Gerando Controller



The screenshot shows a VS Code interface with the following details:

- File Explorer:** Shows the project structure under "PRIMEIRO-APP".
- Editor:** Displays two files: "app.controller.ts" and "tasks.controller.ts".
- Code Completion:** A tooltip is open over the word "@Controller" in "tasks.controller.ts".
 - Definition:** Points to the "Controller" class in "@nestjs/common".
 - Description:** States: "Decorator that marks a class as a Nest controller that can receive inbound requests and produce responses."
 - Details:** Lists related classes: "CONTROLLER_WATERMARK", "ContextCreator", "constrainedMemory", "constrainedMemory", "ContextIdFactory", "countReset", "countReset", "AppController", "ConstantSourceNode", "TasksController", and "AbortController".
 - Usage Examples:** Shows examples of HTTP and Microservice controllers.
 - See Also:** References "Controllers" and "Microservices".
 - Annotations:** Includes "@publicApi".

Gerando Controller



The screenshot shows a VS Code interface with the title bar "primeiro-app". The left sidebar displays the project structure under "EXPLORER":

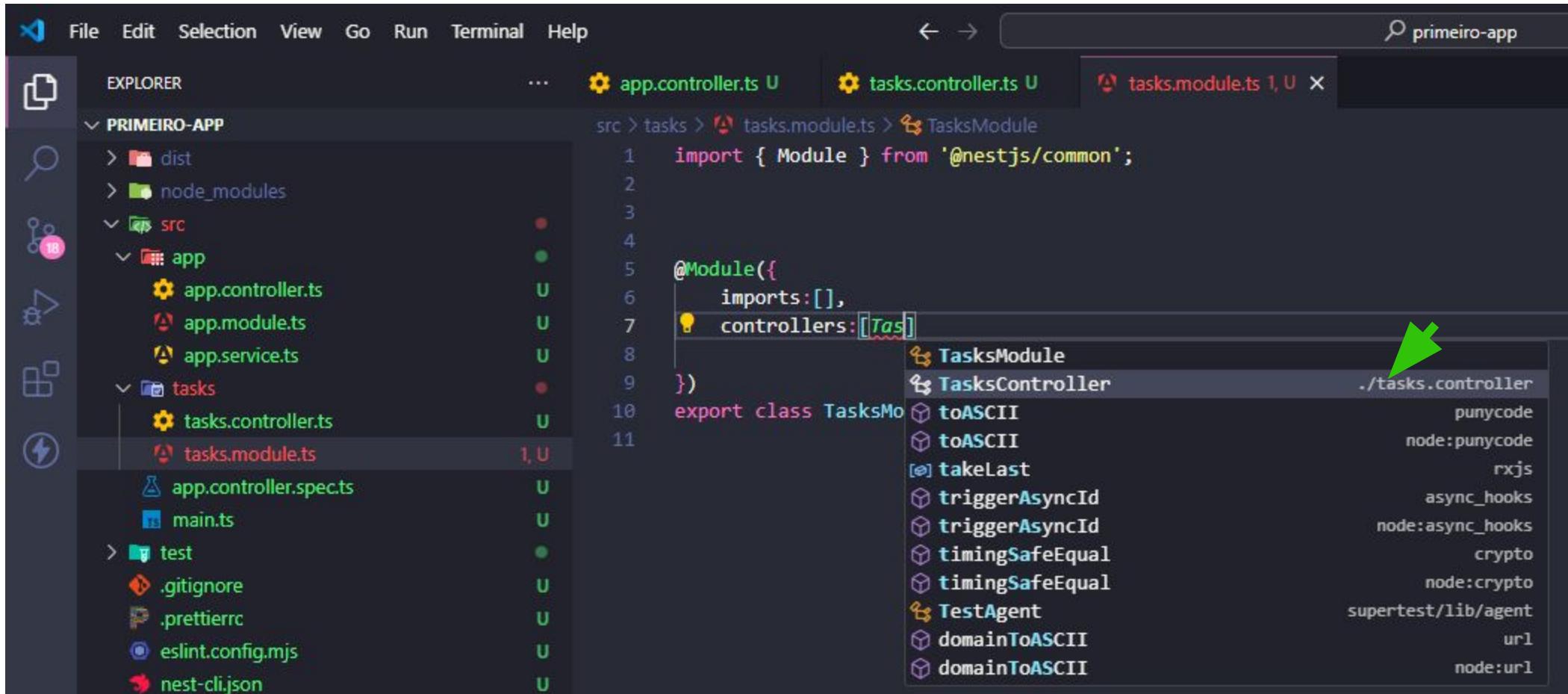
- PRIMEIRO-APP
 - > dist
 - > node_modules
 - > src
 - app
 - app.controller.ts
 - app.module.ts
 - app.service.ts
 - tasks
 - tasks.controller.ts
 - tasks.module.ts
 - app.controller.spec.ts
 - main.ts

The "tasks.controller.ts" file is open in the editor, showing the following code:

```
import { Controller } from '@nestjs/common';
@Controller()
export class TasksController{}
```

A green arrow points to the "@Controller()" decorator on line 4.

Gerando Controller

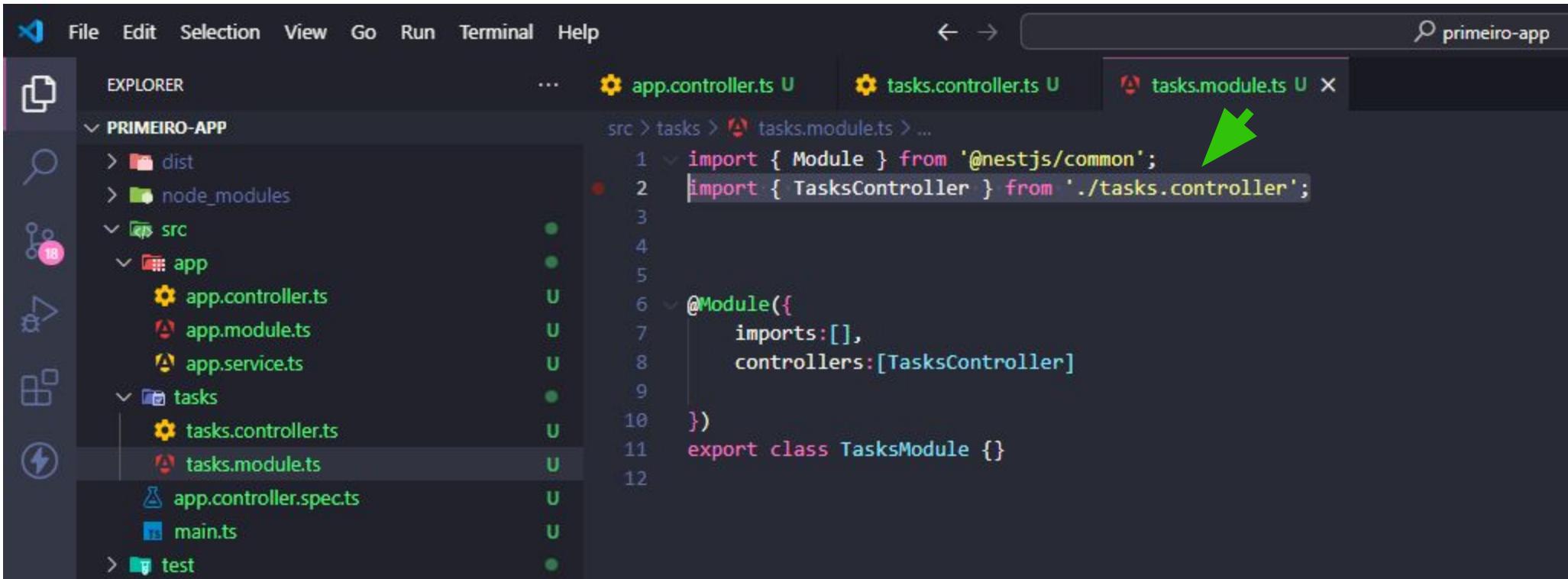


The screenshot shows a VS Code interface with the following details:

- File Explorer:** Shows the project structure under "PRIMERO-APP".
 - src:** Contains **app** (with `app.controller.ts`, `app.module.ts`, `app.service.ts`) and **tasks** (with `tasks.controller.ts`, `tasks.module.ts`, `app.controller.spec.ts`, `main.ts`).
 - test:** Contains `.gitignore`, `.prettierrc`, `eslint.config.mjs`, and `nest-cli.json`.
- Editor:** Displays `tasks.module.ts` with the following code:

```
1 import { Module } from '@nestjs/common';
2
3
4 @Module({
5   imports: [],
6   controllers: [TasksController]
7 })
8 export class TasksModule {
9 }
```
- Code Completion:** A dropdown menu is open at the bottom right of the editor, listing suggestions for the `controllers` array. The suggestion `./tasks.controller` is highlighted with a green arrow.
- Terminal:** Shows the command `primeiro-app`.

Gerando Controller



The screenshot shows the VS Code interface with the title bar "primeiro-app". The left sidebar (EXPLORER) displays the project structure:

- PRIMEIRO-APP
 - > dist
 - > node_modules
 - src
 - app
 - app.controller.ts
 - app.module.ts
 - app.service.ts
 - tasks
 - tasks.controller.ts
 - tasks.module.ts
 - app.controller.spec.ts
 - main.ts
 - > test

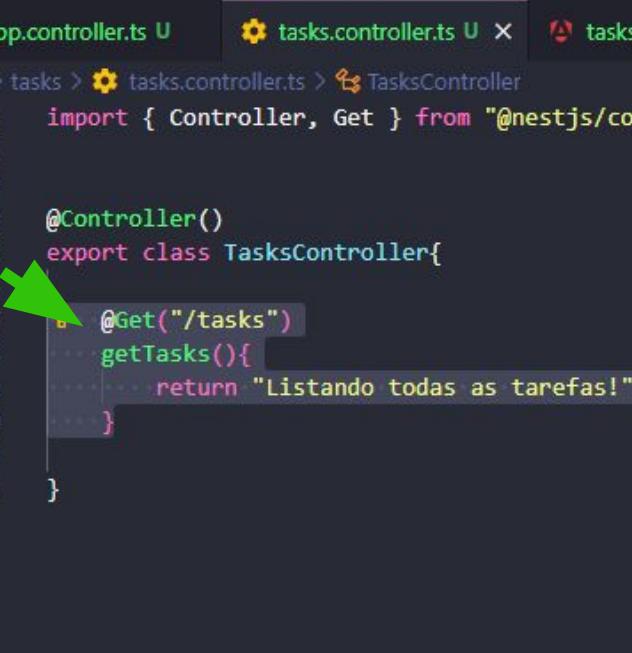
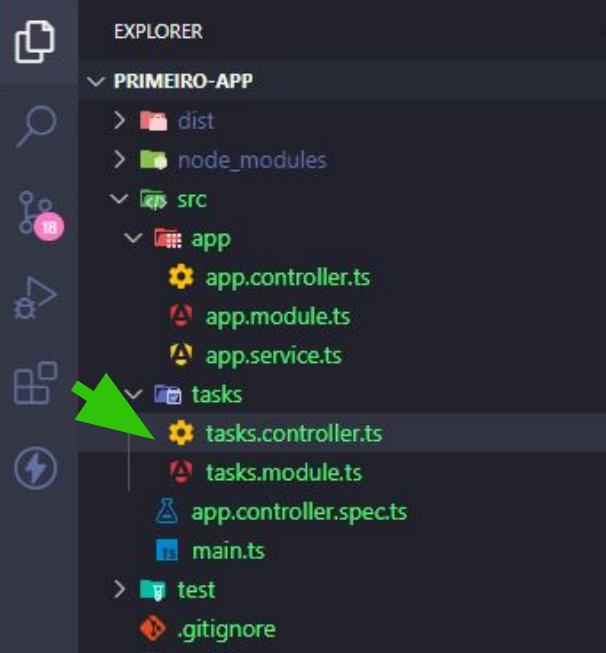
The main editor area shows the content of tasks.module.ts:

```
1 import { Module } from '@nestjs/common';
2 import { TasksController } from './tasks.controller';
3
4
5
6 @Module({
7   imports: [],
8   controllers: [TasksController]
9 })
10
11 export class TasksModule {}
```

A green arrow points to the line "import { TasksController } from './tasks.controller';".

Gerando Controller

Deletar



```
src > tasks > tasks.controller.ts > TasksController
1 import { Controller, Get } from '@nestjs/common';
2
3
4 @Controller()
5 export class TasksController{
6   @Get("/tasks")
7   getTasks(){
8     return "Listando todas as tarefas!"
9   }
10 }
11
12 }
```



New Request

Activity Collections Env

filter collections

route

GET novarota just now

POST criando novarota 15 mins ago

Query Parameters

parameter value

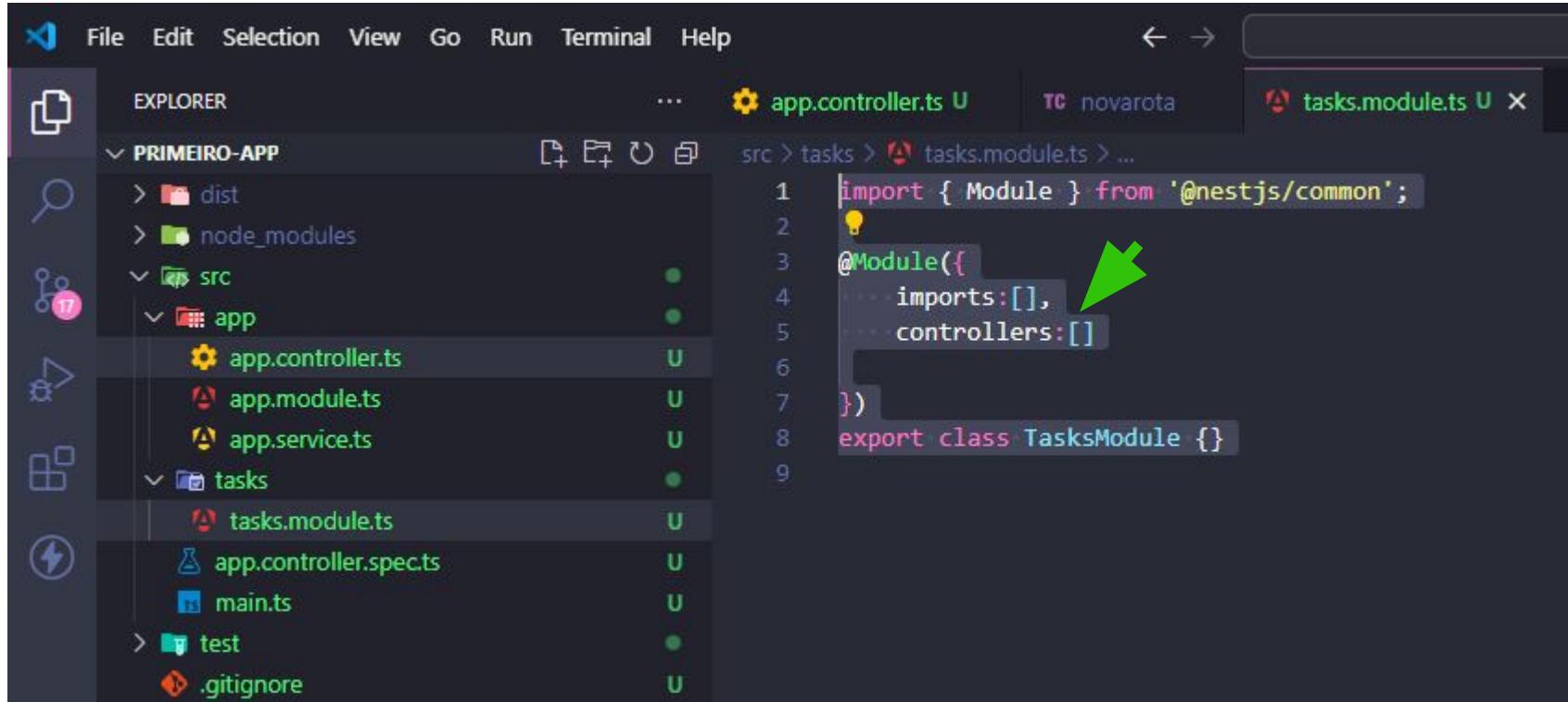
Send

Status: 200 OK Size: 26 Bytes Time: 3 ms

Response Headers Cookies Results Docs

1 Listando todas as tarefas!

Gerando Controller com CLI



The screenshot shows a VS Code interface with the following details:

- File Explorer:** Shows the project structure under "PRIMEIRO-APP".
 - dist
 - node_modules
 - src
 - app
 - app.controller.ts
 - app.module.ts
 - app.service.ts
 - tasks
 - tasks.module.ts
 - app.controller.spec.ts
 - main.ts
 - test
 - .gitignore
- Terminal:** Shows the command used to generate the controller.

```
nest generate controller tasks
```
- Code Editor:** Displays the generated `tasks.module.ts` file.

```
import { Module } from '@nestjs/common';
import { TasksModule } from './tasks.module';

@Module({
  imports: [],
  controllers: []
})
export class TasksModule {}
```

A green arrow points to the `controllers: []` line.

Gerando Controller com CLI

The screenshot shows a VS Code interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Explorer:** Shows the project structure for "PRIMEIRO-APP". The "src" folder contains "app", "tasks", and "test" subfolders. "app" contains "app.controller.ts", "app.module.ts", and "app.service.ts". "tasks" contains "tasks.module.ts", "app.controller.spec.ts", and "main.ts".
- Terminal:** The terminal shows the command "nest --help" being run, displaying usage information and a list of commands.
- Schematics Table:** A table listing various CLI commands and their descriptions, with green arrows pointing to the "name" column.

```
1 import { Module } from '@nestjs/common';
2
3 @Module({
4   imports:[],
5   controllers:[]
6 })
7 export class TasksModule {}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Aggio\nestjs\primeiro-app> nest --help
Usage: nest <command> [options]

Options:
  -v, --version                                Output the current version.
  -h, --help                                     Output usage information.

Commands:
  new[n [options] [name]]                        Generate Nest application.
  build [options] [apps...]                      Build Nest application.
  start [options] [app]                           Run Nest application.
  info|i                                         Display Nest project details.
  add [options] <library>                         Adds support for an external library to your project.
  generate|g [options] <schematic> [name] [path]  Generate a Nest element.
  schematics available on @nestjs/schematics collection:
```

name	alias	description
application	application	Generate a new application workspace
class	cl	Generate a new class
configuration	config	Generate a CLI configuration file
controller	co	Generate a controller declaration
decorator	d	Generate a custom decorator
filter	f	Generate a filter declaration
gateway	ga	Generate a gateway declaration
guard	gu	Generate a guard declaration
interceptor	itc	Generate an interceptor declaration
interface	itf	Generate an interface
library	lib	Generate a new library within a monorepo
middleware	mi	Generate a middleware declaration
module	mo	Generate a module declaration
pipe	pi	Generate a pipe declaration
provider	pr	Generate a provider declaration
resolver	r	Generate a GraphQL resolver declaration
resource	res	Generate a new CRUD resource
service	s	Generate a service declaration
sub-app	app	Generate a new application within a monorepo

PS C:\Aggio\nestjs\primeiro-app> nest g controller tasks

Gerando Controller com CLI

The screenshot shows the VS Code interface with the following details:

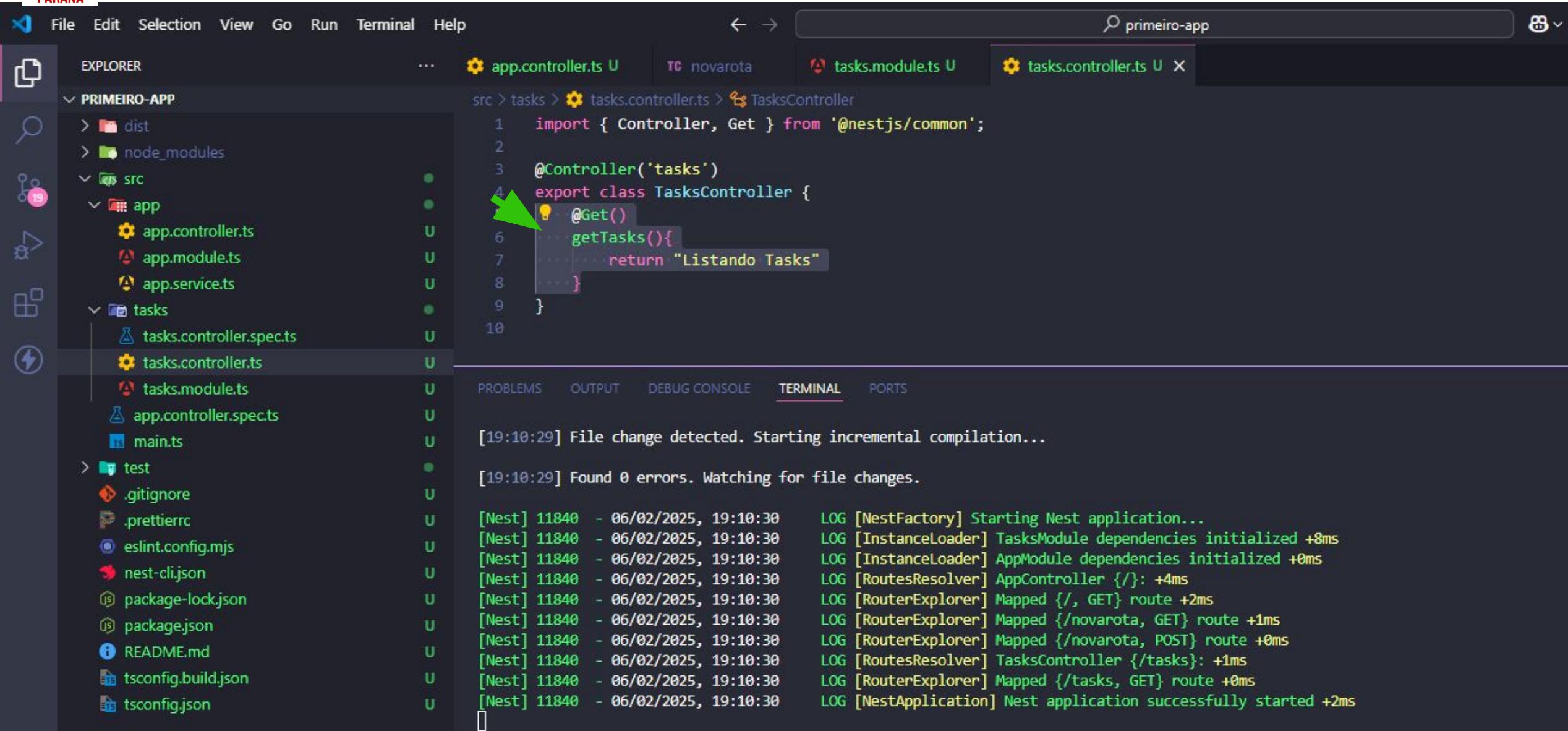
- File Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Search Bar:** primeiro-app
- Explorer:** Shows the project structure under PRIMEIRO-APP:
 - dist
 - node_modules
 - src
 - app
 - app.controller.ts
 - app.module.ts
 - app.service.ts
 - tasks
 - tasks.controller.spec.ts
 - tasks.controller.ts
 - tasks.module.ts
 - test
 - .gitignore
 - .prettierrc
 - .eslint.config.mjs
 - nest-cli.json
 - package-lock.json
 - package.json
 - README.md
 - tsconfig.build.json
 - tsconfig.json
- Code Editor:** Displays the tasks.module.ts file content:

```
1 import { Module } from '@nestjs/common';
2 import { TasksController } from './tasks.controller';
3 
4 @Module({
5   imports:[],
6   controllers:[TasksController]
7 })
8 export class TasksModule {}
```
- Terminal:** Shows the command used to generate the controller and its output:

```
PS C:\Aggio\nestjs\primeiro-app> nest g controller tasks
CREATE src/tasks/tasks.controller.ts (103 bytes)
CREATE src/tasks/tasks.controller.spec.ts (503 bytes)
UPDATE src/tasks/tasks.module.ts (196 bytes)
```
- Bottom Status Bar:** Shows the path PS C:\Aggio\nestjs\primeiro-app

A green arrow points to the 'tasks' folder in the Explorer sidebar, and another green arrow points to the terminal output at the bottom.

Gerando Controller com CLI



The screenshot shows a VS Code interface with the following details:

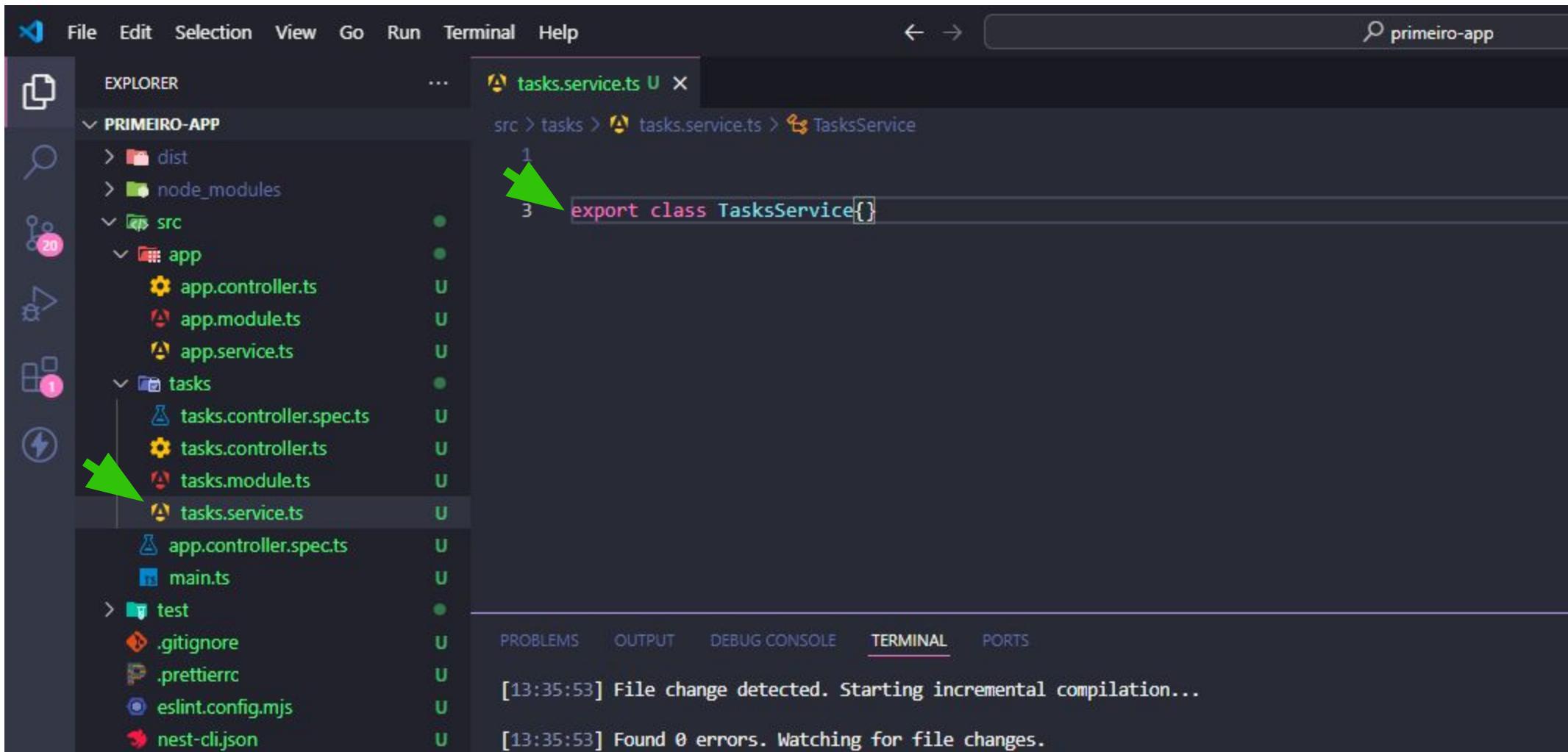
- File Explorer (Left):** Shows the project structure under "PRIMEIRO-APP". The "src" folder contains "app" and "tasks" subfolders. "app" has files: app.controller.ts, app.module.ts, and app.service.ts. "tasks" has files: tasks.controller.spec.ts, tasks.controller.ts, tasks.module.ts, app.controller.spec.ts, main.ts, and test subfolder with .gitignore, .prettierrc, eslint.config.mjs, nest-cli.json, package-lock.json, package.json, README.md, tsconfig.build.json, and tsconfig.json.
- Editor (Top Right):** The file "tasks.controller.ts" is open. A green arrow points to the first line of code:

```
1 import { Controller, Get } from '@nestjs/common';
```
- Terminal (Bottom):** Shows the output of the application startup and route mapping.

```
[19:10:29] File change detected. Starting incremental compilation...
[19:10:29] Found 0 errors. Watching for file changes.

[Nest] 11840 - 06/02/2025, 19:10:30    LOG [NestFactory] Starting Nest application...
[Nest] 11840 - 06/02/2025, 19:10:30    LOG [InstanceLoader] TasksModule dependencies initialized +8ms
[Nest] 11840 - 06/02/2025, 19:10:30    LOG [InstanceLoader] AppModule dependencies initialized +0ms
[Nest] 11840 - 06/02/2025, 19:10:30    LOG [RoutesResolver] AppController {/}: +4ms
[Nest] 11840 - 06/02/2025, 19:10:30    LOG [RouterExplorer] Mapped {/, GET} route +2ms
[Nest] 11840 - 06/02/2025, 19:10:30    LOG [RouterExplorer] Mapped {/novarota, GET} route +1ms
[Nest] 11840 - 06/02/2025, 19:10:30    LOG [RouterExplorer] Mapped {/novarota, POST} route +0ms
[Nest] 11840 - 06/02/2025, 19:10:30    LOG [RoutesResolver] TasksController {/tasks}: +1ms
[Nest] 11840 - 06/02/2025, 19:10:30    LOG [RouterExplorer] Mapped {/tasks, GET} route +0ms
[Nest] 11840 - 06/02/2025, 19:10:30    LOG [NestApplication] Nest application successfully started +2ms
```

Gerando Service / Provider



The screenshot shows the VS Code interface with the following details:

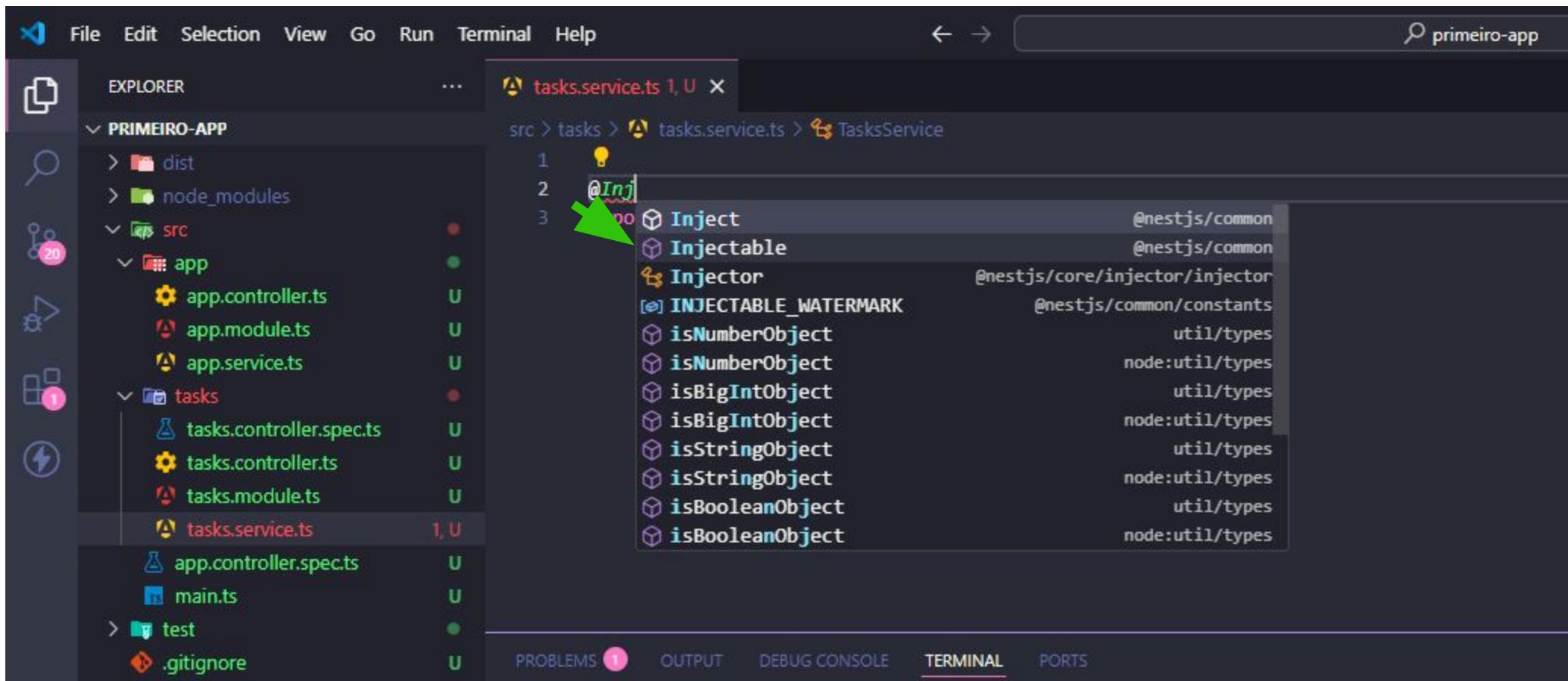
- File Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Search Bar:** primeiro-app
- Explorer Sidebar:** Shows the project structure under 'PRIMEIRO-APP'. The 'tasks' folder is expanded, showing files: tasks.controller.spec.ts, tasks.controller.ts, tasks.module.ts, and tasks.service.ts. A green arrow points to the 'tasks.service.ts' file.
- Code Editor:** The file 'tasks.service.ts' is open. The code is as follows:

```
1  export class TasksService{}  
3
```

A green arrow points to the 'export class TasksService{}' line.
- Terminal:** Shows log messages:

```
[13:35:53] File change detected. Starting incremental compilation...  
[13:35:53] Found 0 errors. Watching for file changes.
```
- Bottom Navigation:** PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (underlined), PORTS.

Gerando Service / Provider



The screenshot shows a VS Code interface with the title bar "primeiro-app". The Explorer sidebar on the left shows a project structure for "PRIMEIRO-APP" with files like "app.controller.ts", "app.module.ts", "app.service.ts", "tasks.controller.ts", "tasks.module.ts", and "tasks.service.ts". The "tasks.service.ts" file is open in the editor, showing the following code:

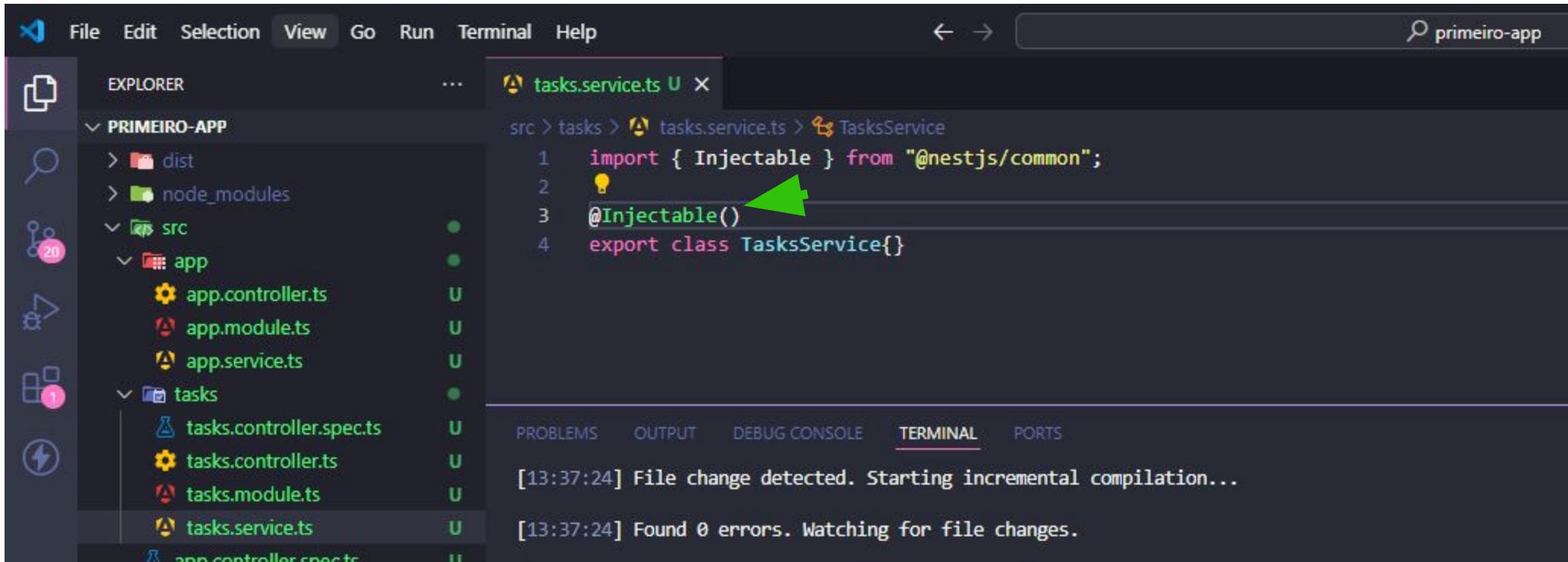
```
1 1 @Inj
2 2 @Inject
3 3 @Injectable
```

A green arrow points from the word "Inject" in the second line to the code completion dropdown menu. The menu lists several NestJS-related types and decorators, all starting with "@":

- @Injectable
- @Injector
- @INJECTABLE_WATERMARK
- @isNumberObject
- @isNumberObject
- @isBigIntObject
- @isBigIntObject
- @isStringObject
- @isStringObject
- @isBooleanObject
- @isBooleanObject

The "Injectable" entry is highlighted in the dropdown. The status bar at the bottom shows tabs for PROBLEMS (1), OUTPUT, DEBUG CONSOLE, TERMINAL (underlined), and PORTS.

Gerando Service / Provider



File Edit Selection View Go Run Terminal Help

primeiro-app

EXPLORER

PRIMEIRO-APP

- > dist
- > node_modules
- src
 - app
 - app.controller.ts
 - app.module.ts
 - app.service.ts
 - tasks
 - tasks.controller.spec.ts
 - tasks.controller.ts
 - tasks.module.ts
 - tasks.service.ts
 - app_controller.spec.ts

tasks.service.ts

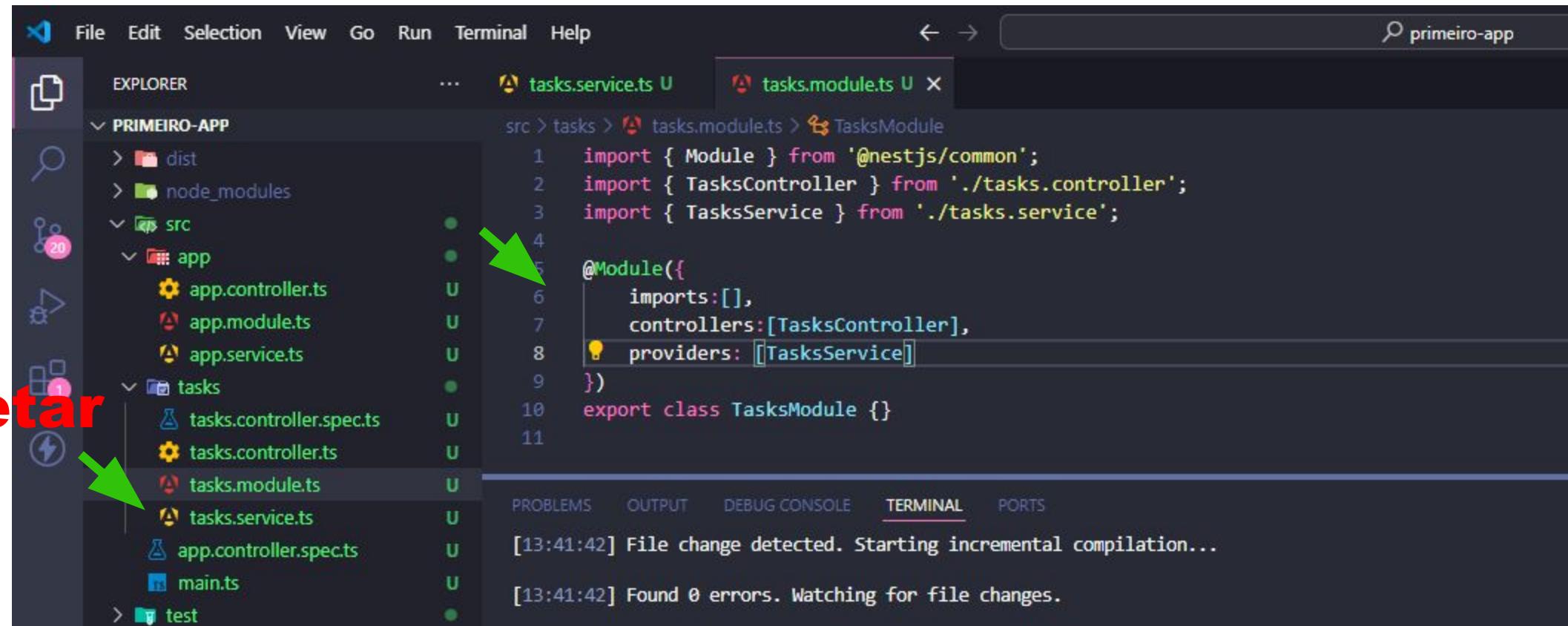
```
src > tasks > tasks.service.ts > TasksService
1 import { Injectable } from '@nestjs/common';
2
3 @Injectable()
4 export class TasksService{}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

[13:37:24] File change detected. Starting incremental compilation...

[13:37:24] Found 0 errors. Watching for file changes.

Gerando Service / Provider



File Edit Selection View Go Run Terminal Help

primeiro-app

EXPLORER

PRIMEIRO-APP

src > tasks > tasks.module.ts > TasksModule

```
1 import { Module } from '@nestjs/common';
2 import { TasksController } from './tasks.controller';
3 import { TasksService } from './tasks.service';

4 @Module({
5   imports:[],
6   controllers:[TasksController],
7   providers: [TasksService]
8 })
9 export class TasksModule {}
```

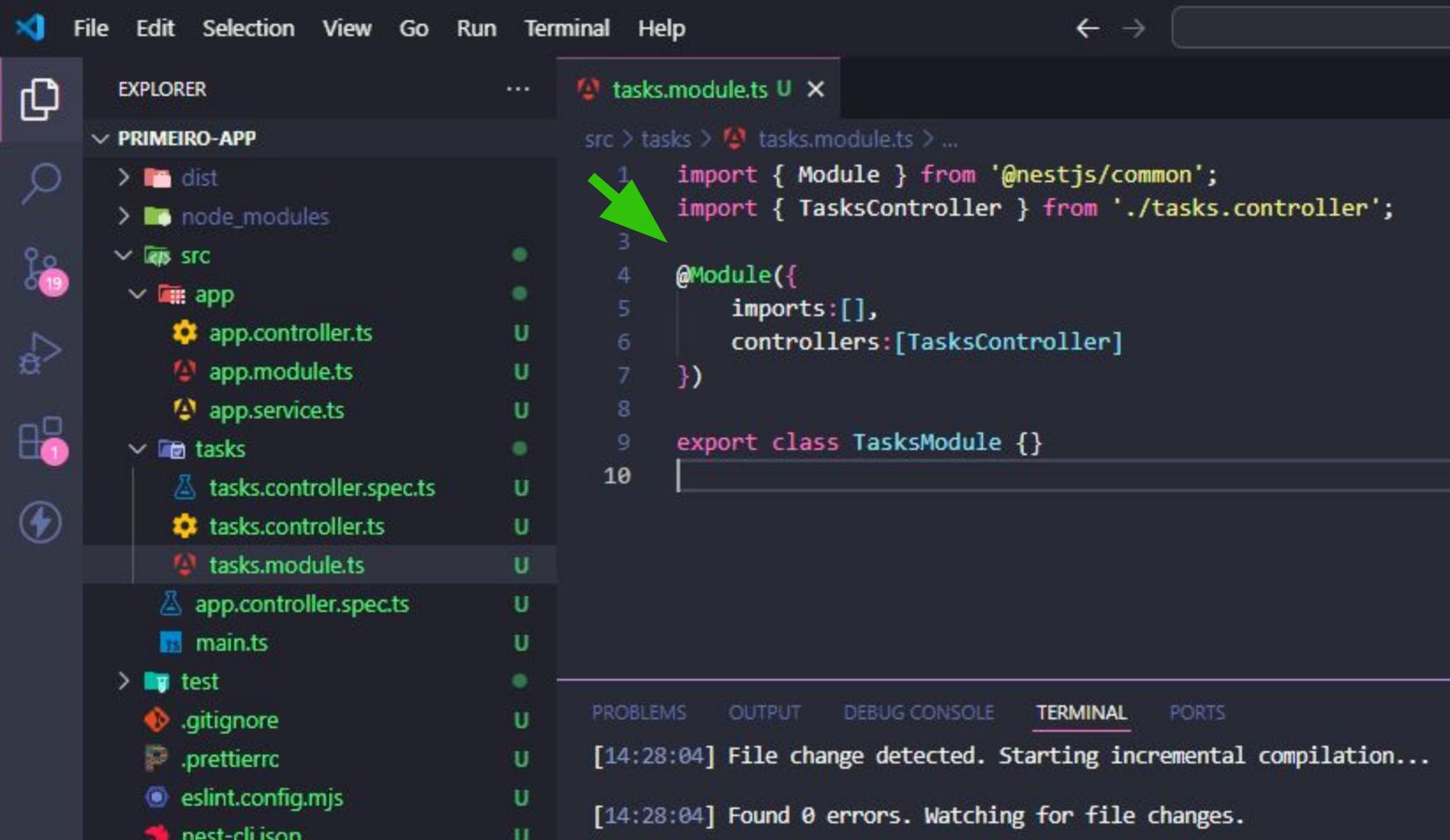
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

[13:41:42] File change detected. Starting incremental compilation...

[13:41:42] Found 0 errors. Watching for file changes.

deletar

Gerando Service / Provider



The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows the project structure under "PRIMEIRO-APP". The "tasks" folder contains "tasks.controller.spec.ts", "tasks.controller.ts", and "tasks.module.ts".
- Editor:** The "tasks.module.ts" file is open. The code is as follows:

```
1 import { Module } from '@nestjs/common';
  import { TasksController } from './tasks.controller';

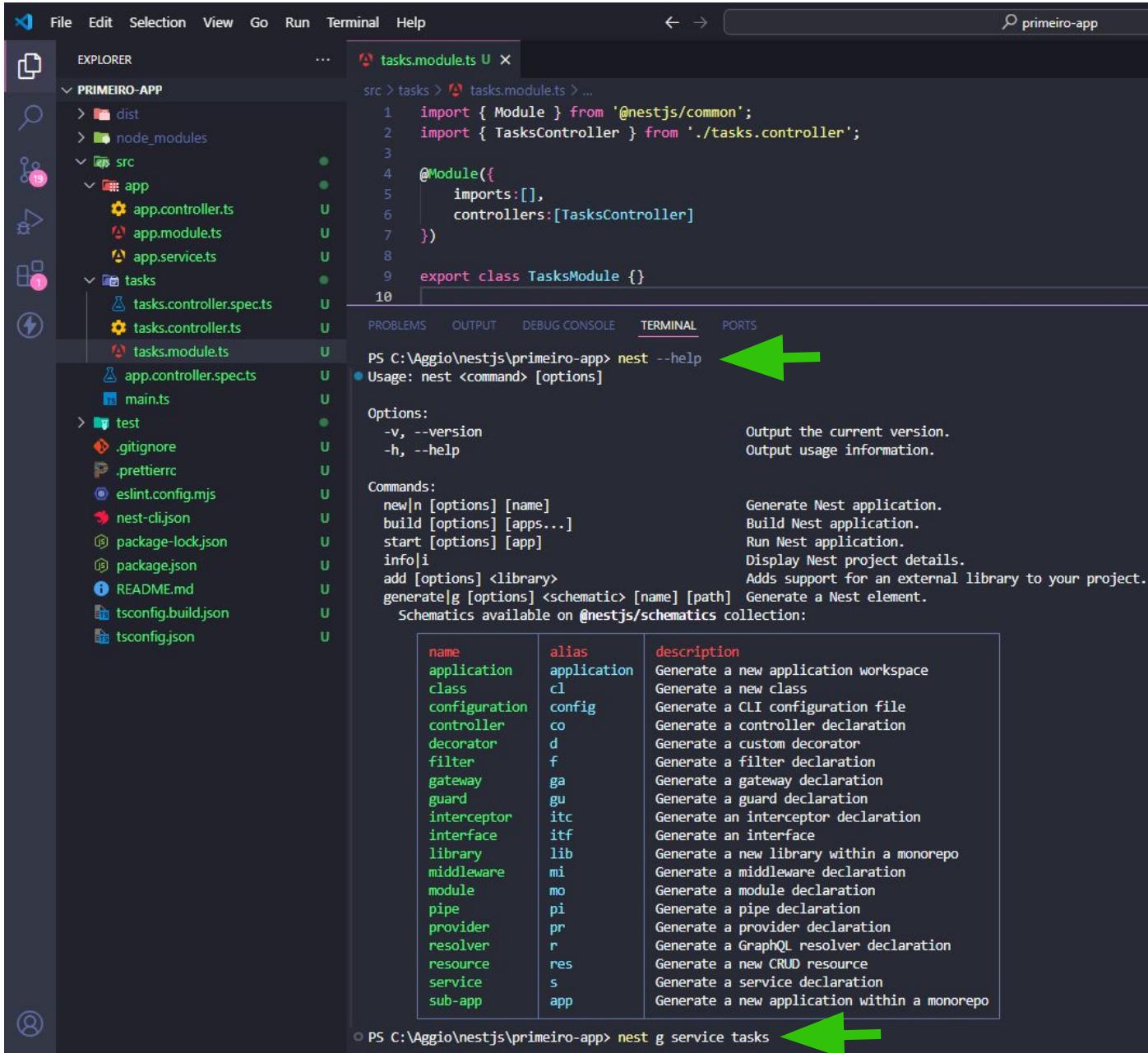
3 @Module({
4   imports: [],
5   controllers: [TasksController]
6 })
7

9 export class TasksModule {}
```

A green arrow points to the `@Module({})` decorator.
- Terminal:** Shows logs of the build process:

```
[14:28:04] File change detected. Starting incremental compilation...
[14:28:04] Found 0 errors. Watching for file changes.
```

Gerando Service / Provider



The screenshot shows the VS Code interface with the following details:

- File Structure (Explorer):** The project structure is displayed under "PRIMEIRO-APP". It includes a "dist" folder, "node_modules", and a "src" folder containing "app" (with "app.controller.ts", "app.module.ts", "app.service.ts"), "tasks" (with "tasks.controller.spec.ts", "tasks.controller.ts", "tasks.module.ts"), and other files like "main.ts", "test", ".gitignore", and configuration files.
- Terminal Output:** The terminal shows the command "nest --help" being run, followed by the Nest CLI usage information and a table of available commands.
- Table of Commands:** A table listing various Nest CLI commands with their descriptions and aliases.

```
PS C:\Aggio\nestjs\primeiro-app> nest --help
Usage: nest <command> [options]

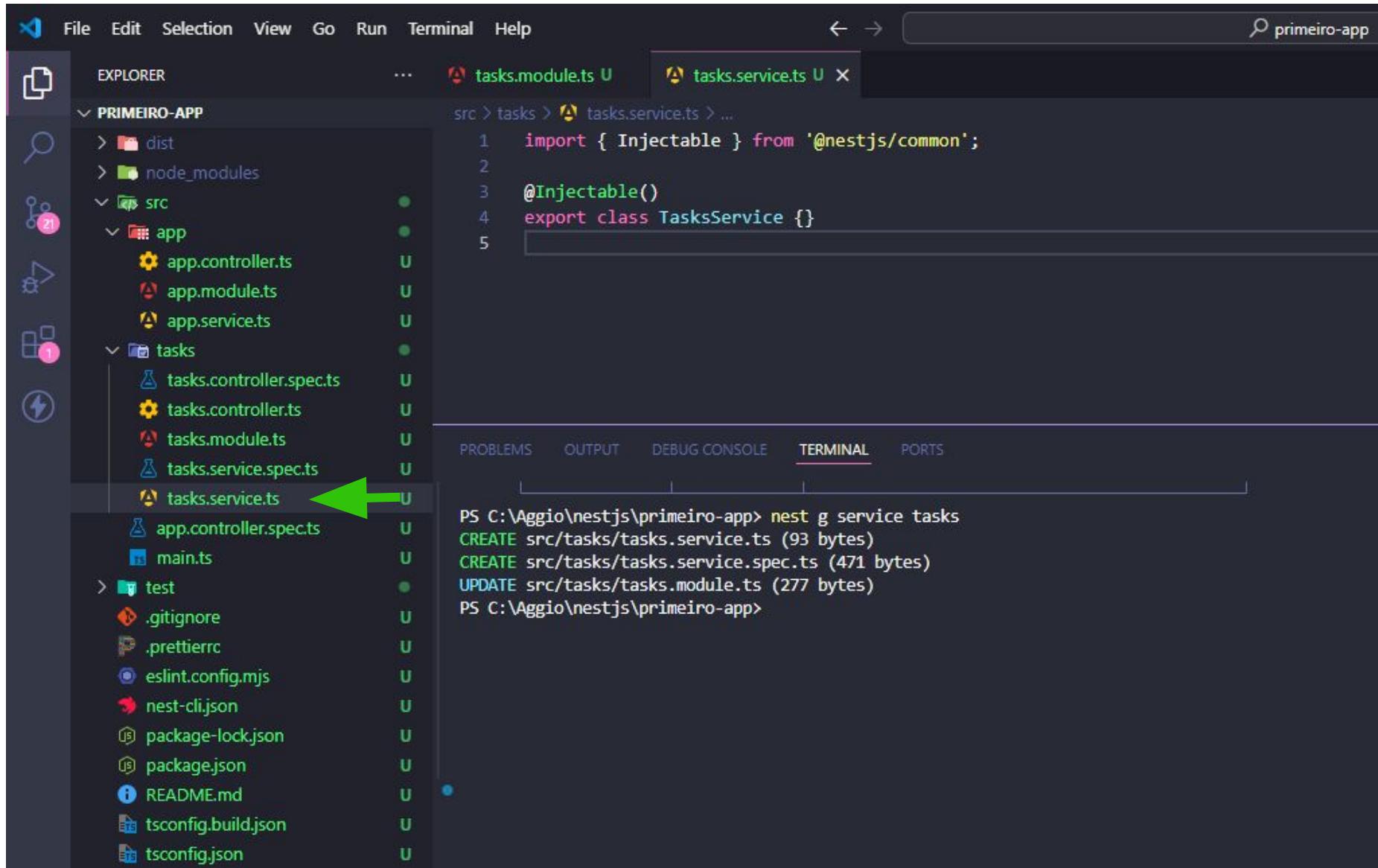
Options:
  -v, --version          Output the current version.
  -h, --help              Output usage information.

Commands:
  new[n [options]] [name]  Generate Nest application.
  build [options] [apps...] Build Nest application.
  start [options] [app]     Run Nest application.
  info|i                Display Nest project details.
  add [options] <library>   Adds support for an external library to your project.
  generate|g [options] <schematic> [name] [path] Generate a Nest element.

  Schematics available on @nestjs/schematics collection:
```

name	alias	description
application	application	Generate a new application workspace
class	c1	Generate a new class
configuration	config	Generate a CLI configuration file
controller	co	Generate a controller declaration
decorator	d	Generate a custom decorator
filter	f	Generate a filter declaration
gateway	ga	Generate a gateway declaration
guard	gu	Generate a guard declaration
interceptor	itc	Generate an interceptor declaration
interface	itf	Generate an interface
library	lib	Generate a new library within a monorepo
middleware	mi	Generate a middleware declaration
module	mo	Generate a module declaration
pipe	pi	Generate a pipe declaration
provider	pr	Generate a provider declaration
resolver	r	Generate a GraphQL resolver declaration
resource	res	Generate a new CRUD resource
service	s	Generate a service declaration
sub-app	app	Generate a new application within a monorepo

Gerando Service / Provider



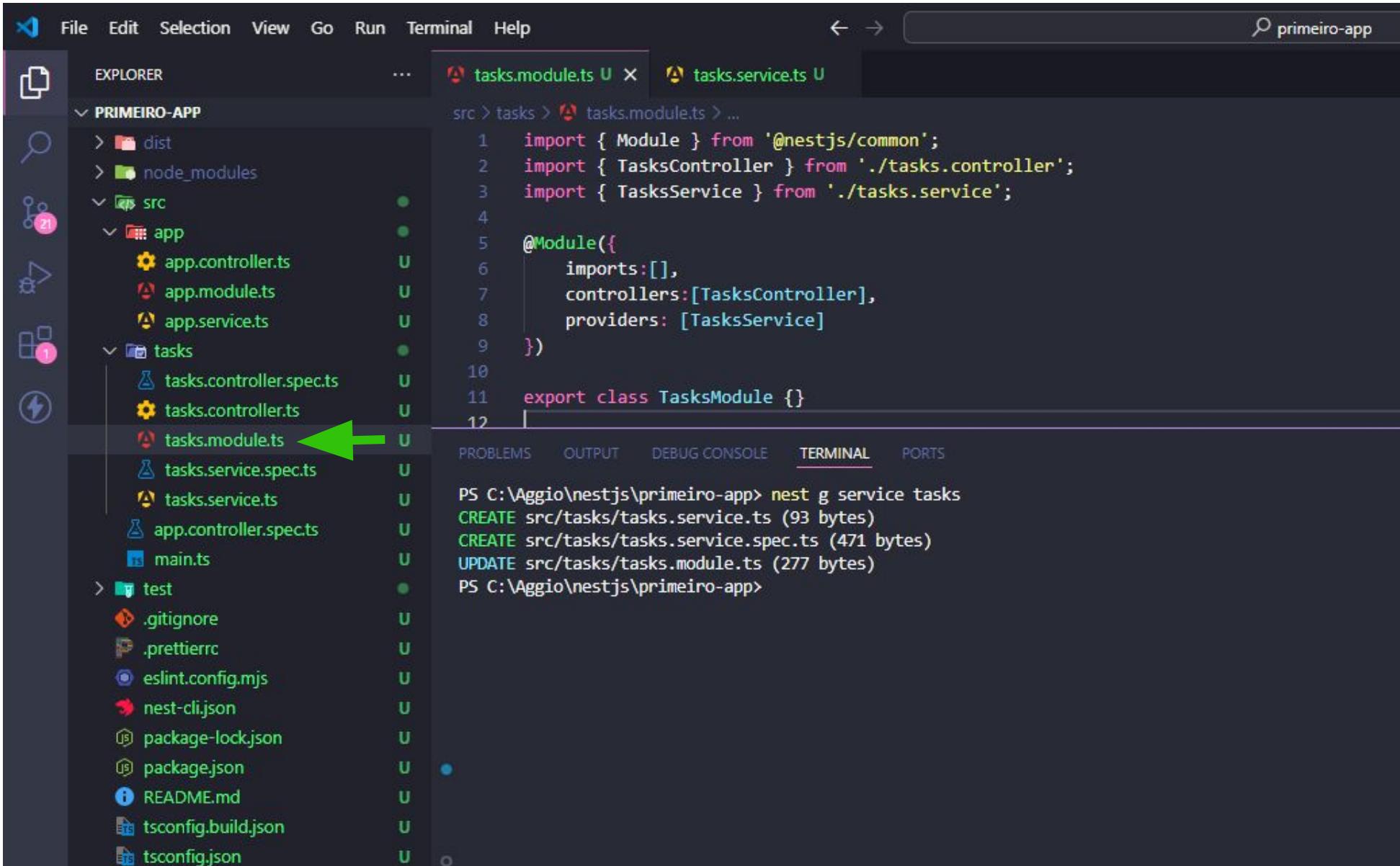
The screenshot shows a VS Code interface with the following details:

- File Explorer (Left):** Shows the project structure under "PRIMEIRO-APP". The "tasks" folder is expanded, showing "tasks.module.ts", "tasks.service.ts", and "tasks.service.spec.ts". A green arrow points to "tasks.service.ts".
- Editor (Top Right):** Displays the content of "tasks.service.ts". It contains the following code:

```
import { Injectable } from '@nestjs/common';
@Injectable()
export class TasksService {}
```
- Terminal (Bottom Right):** Shows the command-line output of running the Nest CLI to generate the service:

```
PS C:\Aggio\nestjs\primeiro-app> nest g service tasks
CREATE src/tasks/tasks.service.ts (93 bytes)
CREATE src/tasks/tasks.service.spec.ts (471 bytes)
UPDATE src/tasks/tasks.module.ts (277 bytes)
PS C:\Aggio\nestjs\primeiro-app>
```

Gerando Service / Provider



The screenshot shows a VS Code interface with the title bar "primeiro-app". The Explorer sidebar on the left displays the project structure of "PRIMEIRO-APP" with files like "app.controller.ts", "app.module.ts", "app.service.ts", and "tasks". A green arrow points to the "tasks.module.ts" file in the Explorer. The code editor shows the content of "tasks.module.ts":

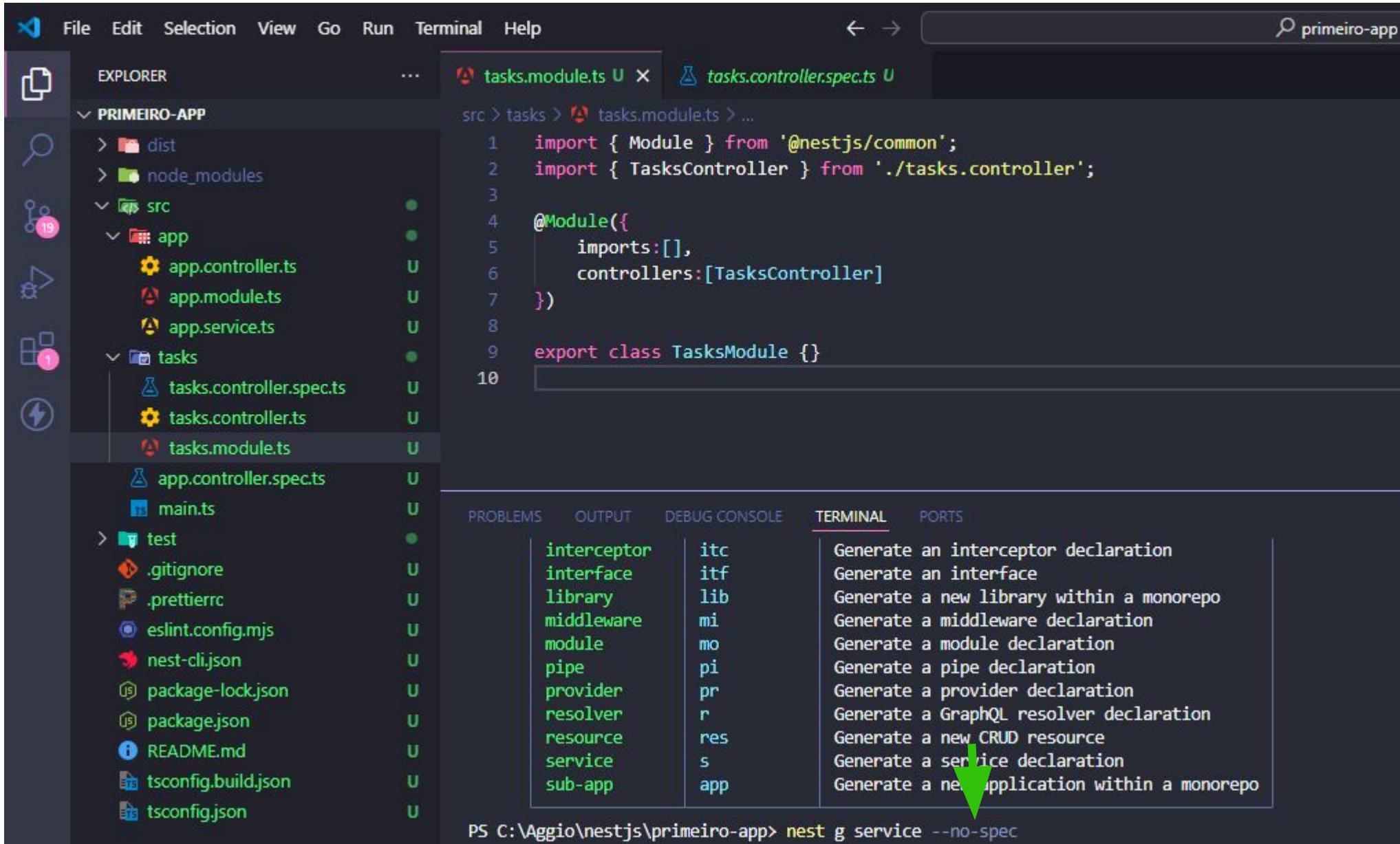
```
1 import { Module } from '@nestjs/common';
2 import { TasksController } from './tasks.controller';
3 import { TasksService } from './tasks.service';

4 @Module({
5   imports: [],
6   controllers: [TasksController],
7   providers: [TasksService]
8 })
9
10 export class TasksModule {}
```

The terminal at the bottom shows the command "nest g service tasks" being run, which generates the service files:

```
PS C:\Aggio\nestjs\primeiro-app> nest g service tasks
CREATE src/tasks/tasks.service.ts (93 bytes)
CREATE src/tasks/tasks.service.spec.ts (471 bytes)
UPDATE src/tasks/tasks.module.ts (277 bytes)
PS C:\Aggio\nestjs\primeiro-app>
```

Gerando Service / Provider



The screenshot shows a VS Code interface with the following details:

- File Explorer:** Shows the project structure under "PRIMEIRO-APP".
- Editor:** Displays the file `tasks.module.ts` with the following code:

```
1 import { Module } from '@nestjs/common';
2 import { TasksController } from './tasks.controller';
3
4 @Module({
5   imports: [],
6   controllers: [TasksController]
7 })
8
9 export class TasksModule {}
```
- Terminal:** Shows the command `nest g service --no-spec` being run.
- Bottom Right:** A context menu is open, listing options for generating declarations like interceptor, interface, library, middleware, module, pipe, provider, resolver, resource, service, and sub-app.

Gerando Service / Provider

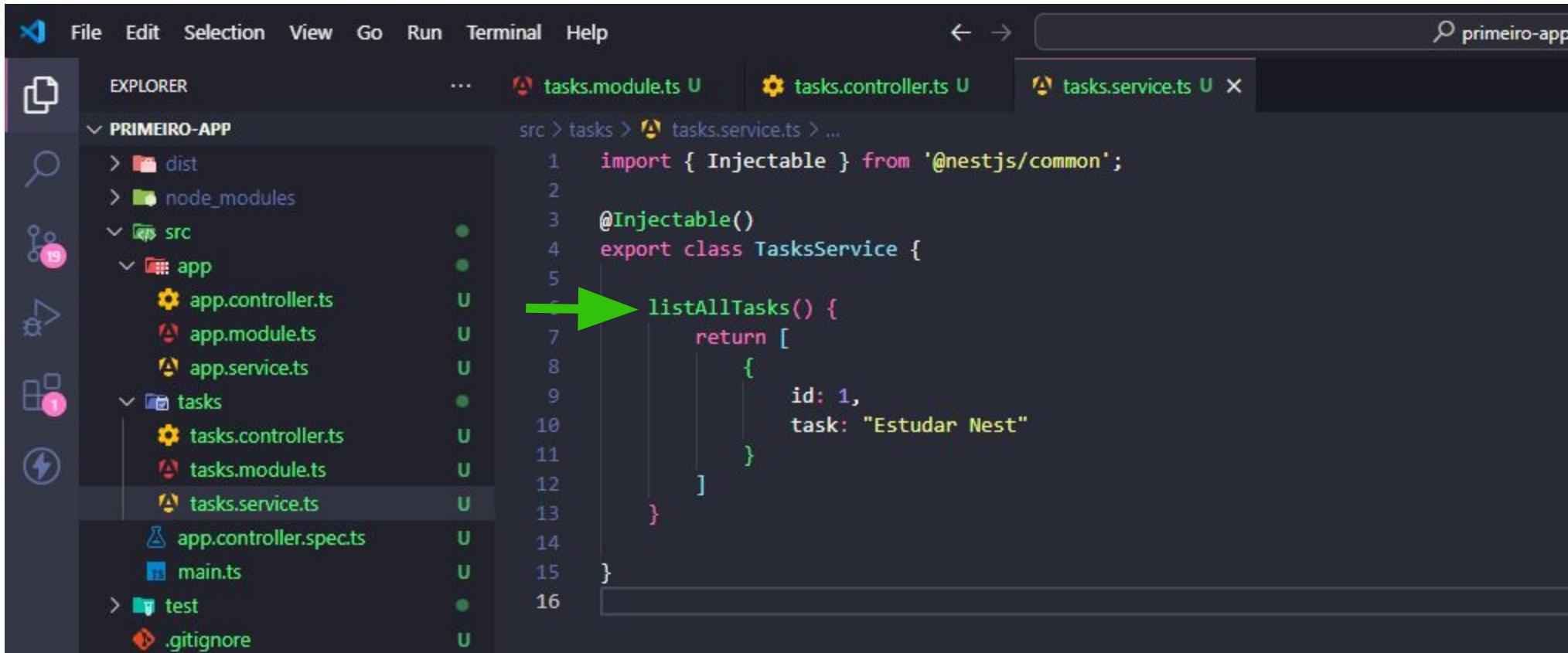
The screenshot shows a VS Code interface with the following details:

- File Structure (EXPLORER):** The project structure is displayed under "PRIMEIRO-APP". It includes a "src" folder containing "app", "tasks", and "test" subfolders. "app" contains "app.controller.ts", "app.module.ts", and "app.service.ts". "tasks" contains "tasks.controller.spec.ts", "tasks.controller.ts", and "tasks.module.ts". "test" contains ".gitignore", ".prettierrc", "eslint.config.mjs", "nest-cli.json", "package-lock.json", "package.json", "README.md", "tsconfig.build.json", and "tsconfig.json".
- Code Editor (tasks.module.ts):** The code defines a TasksModule with imports for TasksController and TasksService, and controllers for TasksController.
- Terminal:** The terminal shows the command "nest g service tasks --no-spec" being run, which creates a new service file "src/tasks/tasks.service.ts" (93 bytes) and updates the module file "src/tasks/tasks.module.ts" (277 bytes).
- IntelliSense Suggestion Box:** A suggestion box is open over the "nest g" command, listing various generator options like "module", "pipe", "provider", etc., each with a description and a green arrow pointing to the "service" option.

Sem gerar o arquivo de test

Generator	Description
module	Generate a module declaration
pipe	Generate a pipe declaration
provider	Generate a provider declaration
pipe	Generate a pipe declaration
provider	Generate a provider declaration
provider	Generate a provider declaration
resolver	Generate a resolver declaration
resource	Generate a new CRUD resource
service	Generate a service declaration
sub-app	Generate a new application within a monorepo

Gerando Service / Provider



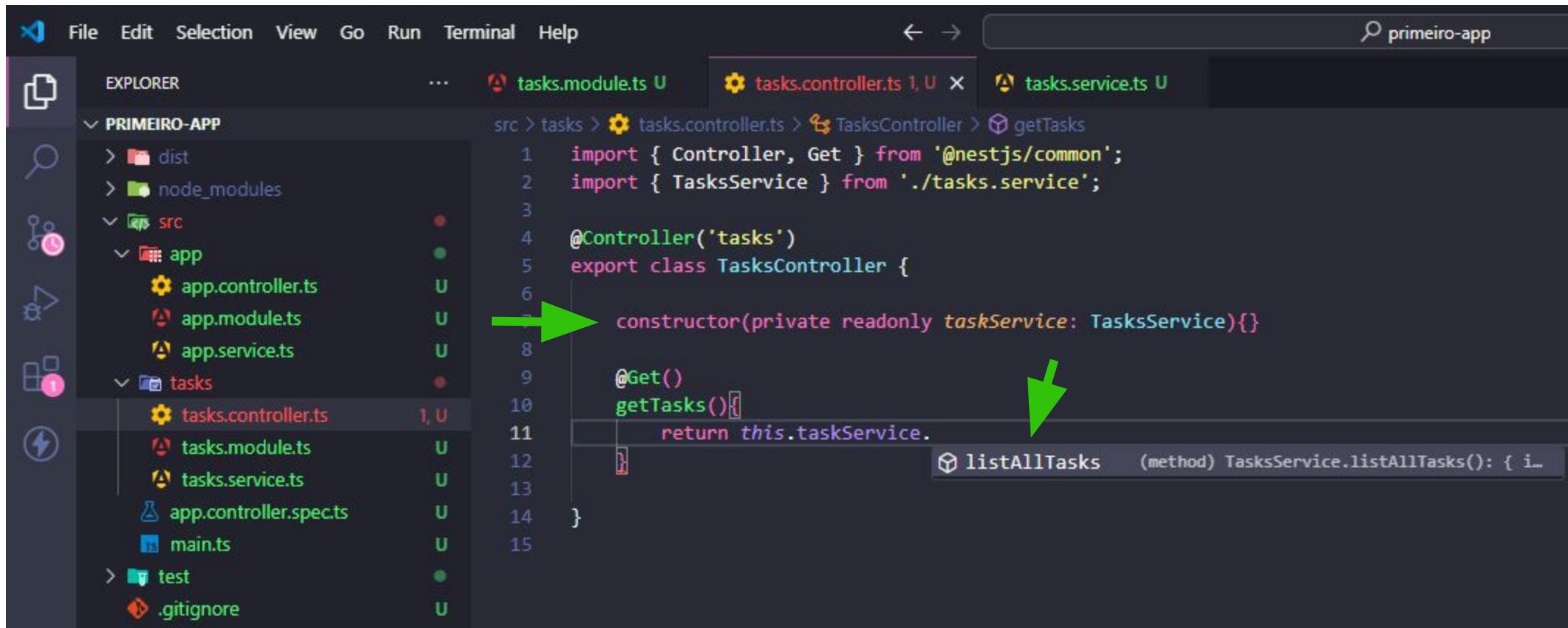
The screenshot shows the VS Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Search Bar:**primeiro-app
- Explorer:** Shows the project structure under PRIMEIRO-APP:
 - dist
 - node_modules
 - src
 - app
 - app.controller.ts
 - app.module.ts
 - app.service.ts
 - tasks
 - tasks.controller.ts
 - tasks.module.ts
 - tasks.service.ts
 - app.controller.spec.ts
 - main.ts
 - test
 - .gitignore
- Editor:** tasks.service.ts file open, showing NestJS code.

```
1 import { Injectable } from '@nestjs/common';
2
3 @Injectable()
4 export class TasksService {
5
6   listAllTasks() {
7     return [
8       {
9         id: 1,
10        task: "Estudar Nest"
11      }
12    ]
13  }
14
15 }
16 
```

A large green arrow points to the `listAllTasks()` method.

Gerando Service / Provider



File Edit Selection View Go Run Terminal Help ← → 🔍 primeiro-app

EXPLORER

PRIMEIRO-APP

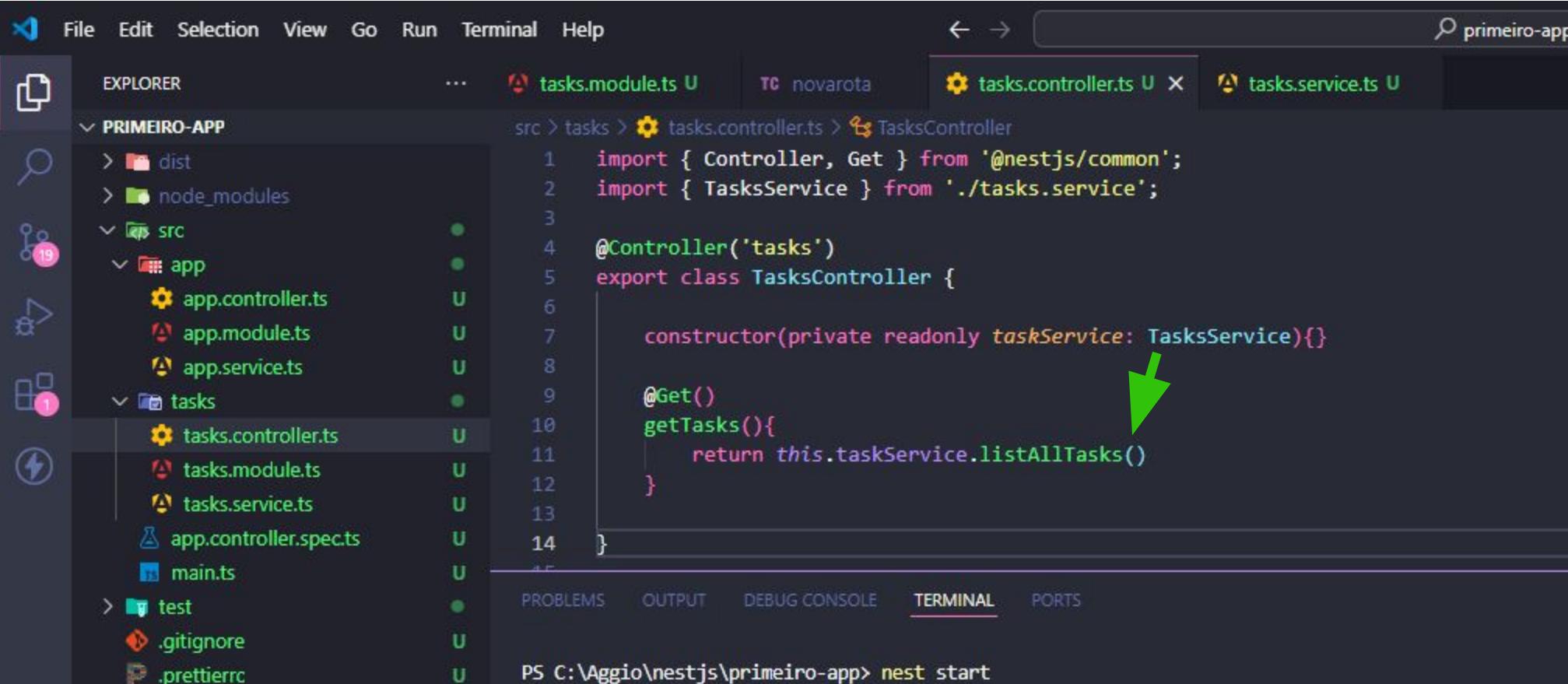
- > dist
- > node_modules
- src
 - app
 - app.controller.ts
 - app.module.ts
 - app.service.ts
 - tasks
 - tasks.controller.ts 1, U
 - tasks.module.ts
 - tasks.service.ts
 - app.controller.spec.ts
 - main.ts
- test
- .gitignore

tasks.module.ts U tasks.controller.ts 1, U tasks.service.ts U

```
src > tasks > tasks.controller.ts > TasksController > getTasks
1 import { Controller, Get } from '@nestjs/common';
2 import { TasksService } from './tasks.service';
3
4 @Controller('tasks')
5 export class TasksController {
6   constructor(private readonly taskService: TasksService){}
7
8   @Get()
9   getTasks(){
10     return this.taskService.
11   }
12 }
13
14 }
15 }
```

listAllTasks (method) TasksService.listAllTasks(): { i...

Gerando Service / Provider

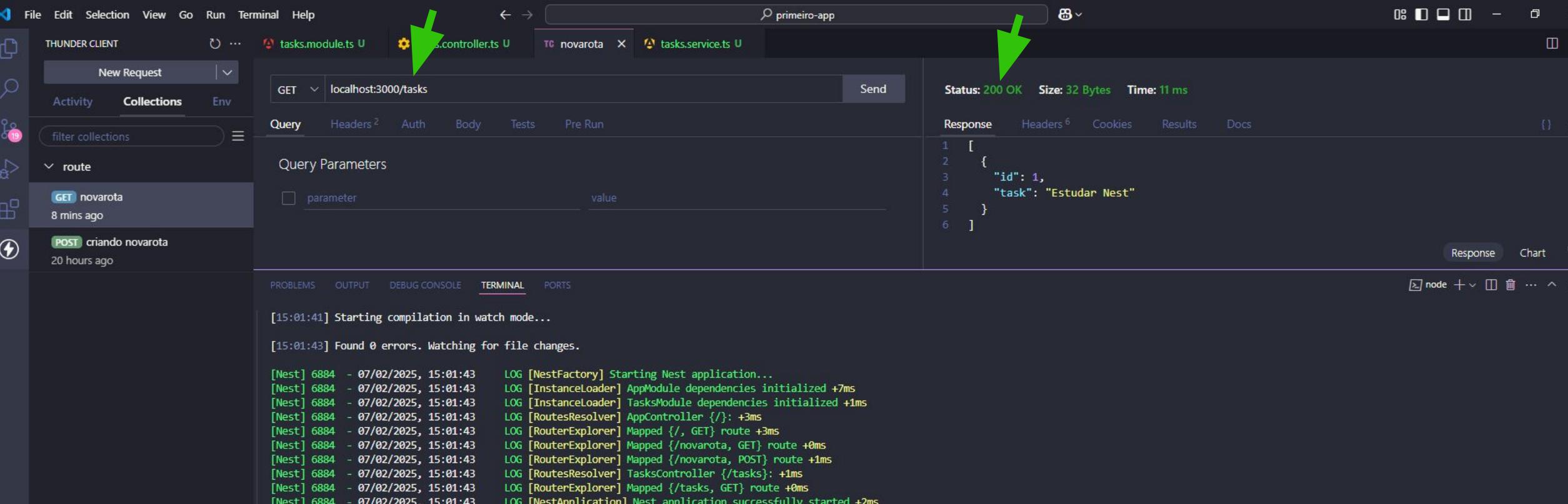


The screenshot shows the VS Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Search Bar:**primeiro-app
- Explorer:** PRIMEIRO-APP folder structure:
 - dist
 - node_modules
 - src
 - app
 - app.controller.ts
 - app.module.ts
 - app.service.ts
 - tasks
 - tasks.controller.ts
 - tasks.module.ts
 - tasks.service.ts
 - app.controller.spec.ts
 - main.ts
 - test
 - .gitignore
 - .prettierrc
- Editor:** tasks.controller.ts (highlighted)

```
1 import { Controller, Get } from '@nestjs/common';
2 import { TasksService } from './tasks.service';
3
4 @Controller('tasks')
5 export class TasksController {
6
7   constructor(private readonly taskService: TasksService){}
8
9   @Get()
10  getTasks(){
11    return this.taskService.listAllTasks()
12  }
13}
14}
```
- Terminal:** PS C:\Aggio\nestjs\primeiro-app> nest start
- Bottom Navigation:** PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (underlined), PORTS

Gerando Service / Provider



The screenshot shows the Thunder Client interface for testing a Nest.js application named "primeiro-app".

Request:

- Method: GET
- URL: localhost:3000/tasks
- Headers: None
- Body: None
- Query Parameters: None

Response:

- Status: 200 OK
- Size: 32 Bytes
- Time: 11 ms
- Content:

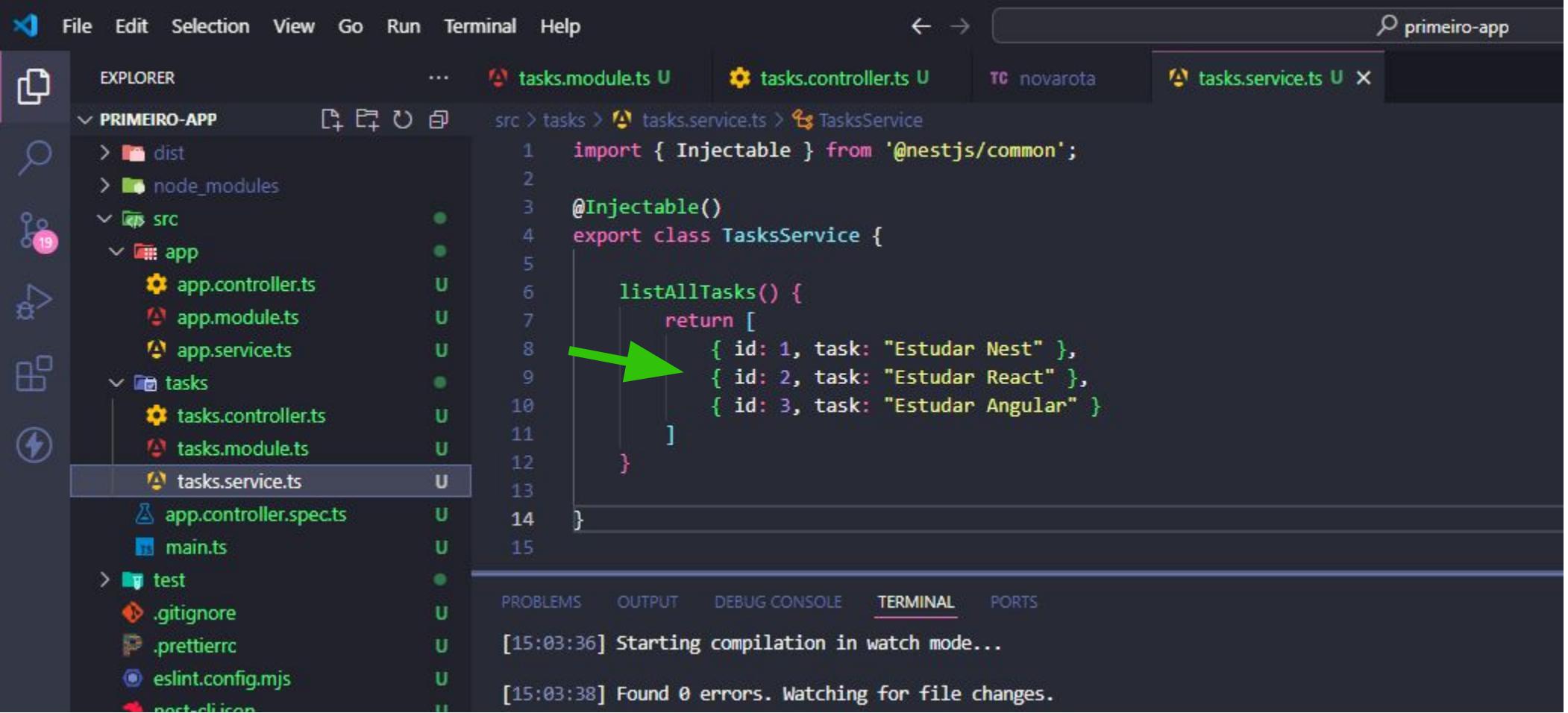
```
[{"id": 1, "task": "Estudar Nest"}]
```

Terminal Output:

```
[15:01:41] Starting compilation in watch mode...
[15:01:43] Found 0 errors. Watching for file changes.

[Nest] 6884 - 07/02/2025, 15:01:43    LOG [NestFactory] Starting Nest application...
[Nest] 6884 - 07/02/2025, 15:01:43    LOG [InstanceLoader] AppModule dependencies initialized +7ms
[Nest] 6884 - 07/02/2025, 15:01:43    LOG [InstanceLoader] TasksModule dependencies initialized +1ms
[Nest] 6884 - 07/02/2025, 15:01:43    LOG [RoutesResolver] AppController {/}: +3ms
[Nest] 6884 - 07/02/2025, 15:01:43    LOG [RouterExplorer] Mapped {/, GET} route +3ms
[Nest] 6884 - 07/02/2025, 15:01:43    LOG [RouterExplorer] Mapped {/novarota, GET} route +0ms
[Nest] 6884 - 07/02/2025, 15:01:43    LOG [RouterExplorer] Mapped {/novarota, POST} route +1ms
[Nest] 6884 - 07/02/2025, 15:01:43    LOG [RoutesResolver] TasksController {/tasks}: +1ms
[Nest] 6884 - 07/02/2025, 15:01:43    LOG [RouterExplorer] Mapped {/tasks, GET} route +0ms
[Nest] 6884 - 07/02/2025, 15:01:43    LOG [NestApplication] Nest application successfully started +2ms
```

Gerando Service / Provider



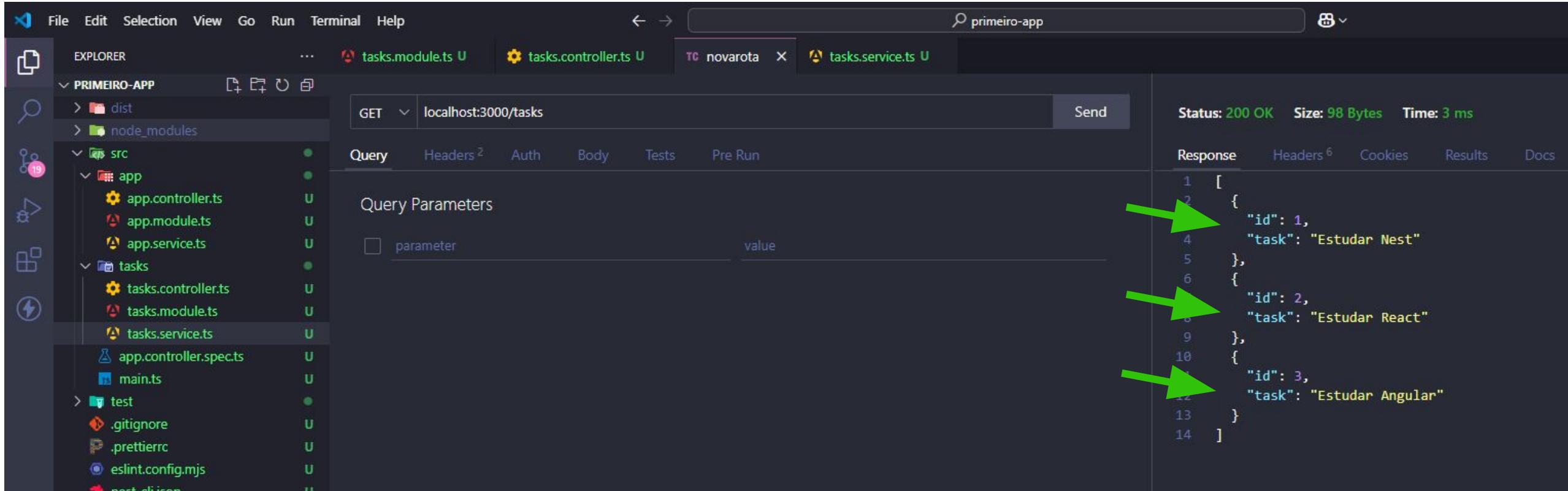
The screenshot shows the VS Code interface with the title bar "primeiro-app". The Explorer sidebar on the left shows the project structure under "PRIMEIRO-APP", including files like "app.controller.ts", "app.module.ts", "app.service.ts", "tasks.controller.ts", "tasks.module.ts", and "tasks.service.ts". The "tasks.service.ts" file is currently selected and open in the editor. The code in the editor is:

```
import { Injectable } from '@nestjs/common';
@Injectable()
export class TasksService {
  listAllTasks() {
    return [
      { id: 1, task: "Estudar Nest" },
      { id: 2, task: "Estudar React" },
      { id: 3, task: "Estudar Angular" }
    ]
  }
}
```

A green arrow points to the line containing the array of tasks. The terminal at the bottom shows the output of the application starting in watch mode.

```
[15:03:36] Starting compilation in watch mode...
[15:03:38] Found 0 errors. Watching for file changes.
```

Gerando Service / Provider



File Edit Selection View Go Run Terminal Help ← → 🔍 primeiro-app

EXPLORER ... tasks.module.ts U tasks.controller.ts U novarota x tasks.service.ts U

PRIMEIRO-APP GET localhost:3000/tasks Send

Status: 200 OK Size: 98 Bytes Time: 3 ms

Response Headers Cookies Results Docs

```
1 [  
2 {  
3 "id": 1,  
4 "task": "Estudar Nest"  
5 },  
6 {  
7 "id": 2,  
8 "task": "Estudar React"  
9 },  
10 {  
11 "id": 3,  
12 "task": "Estudar Angular"  
13 }  
14 ]
```

Query Headers² Auth Body Tests Pre Run

Query Parameters

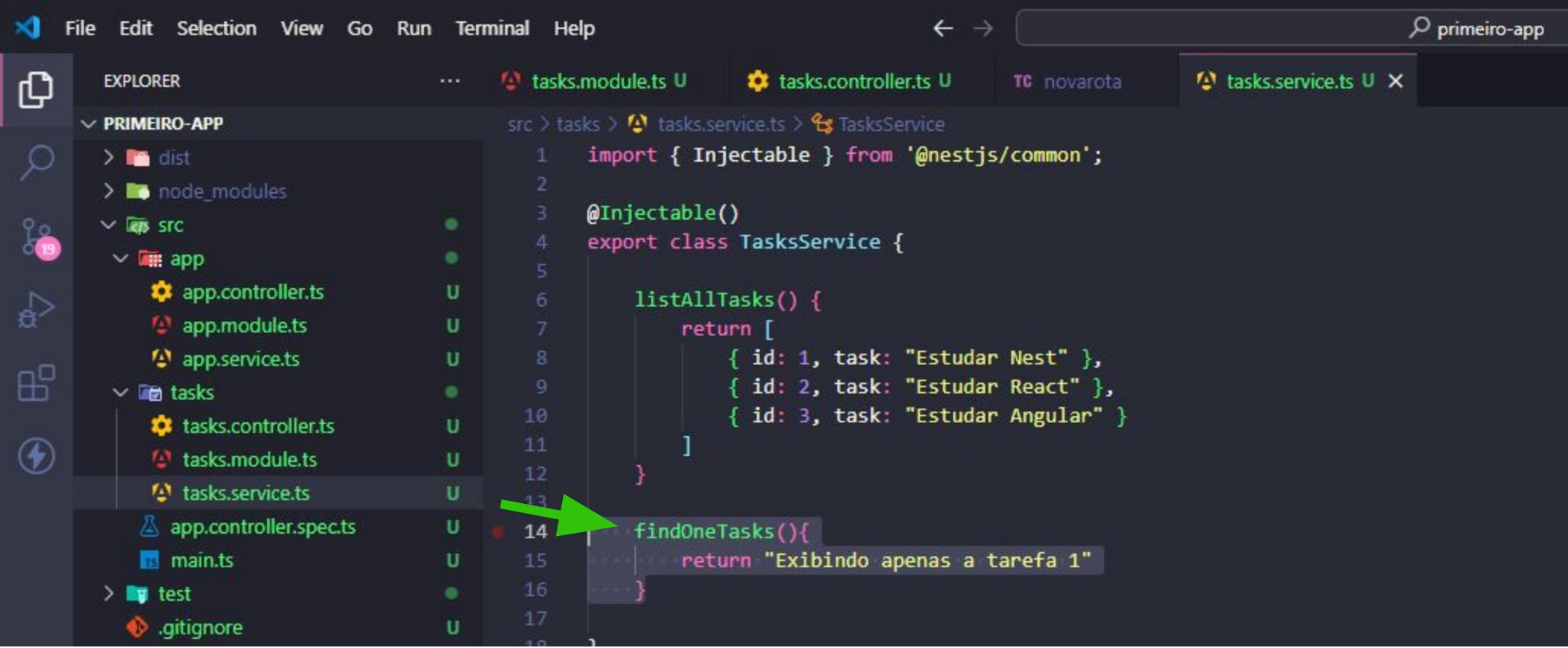
parameter value

dist node_modules

src app
app.controller.ts app.module.ts app.service.ts
tasks tasks.controller.ts tasks.module.ts tasks.service.ts
app.controller.spec.ts main.ts

test .gitignore .prettierrc eslint.config.mjs postcss.config.js

Gerando Service / Provider

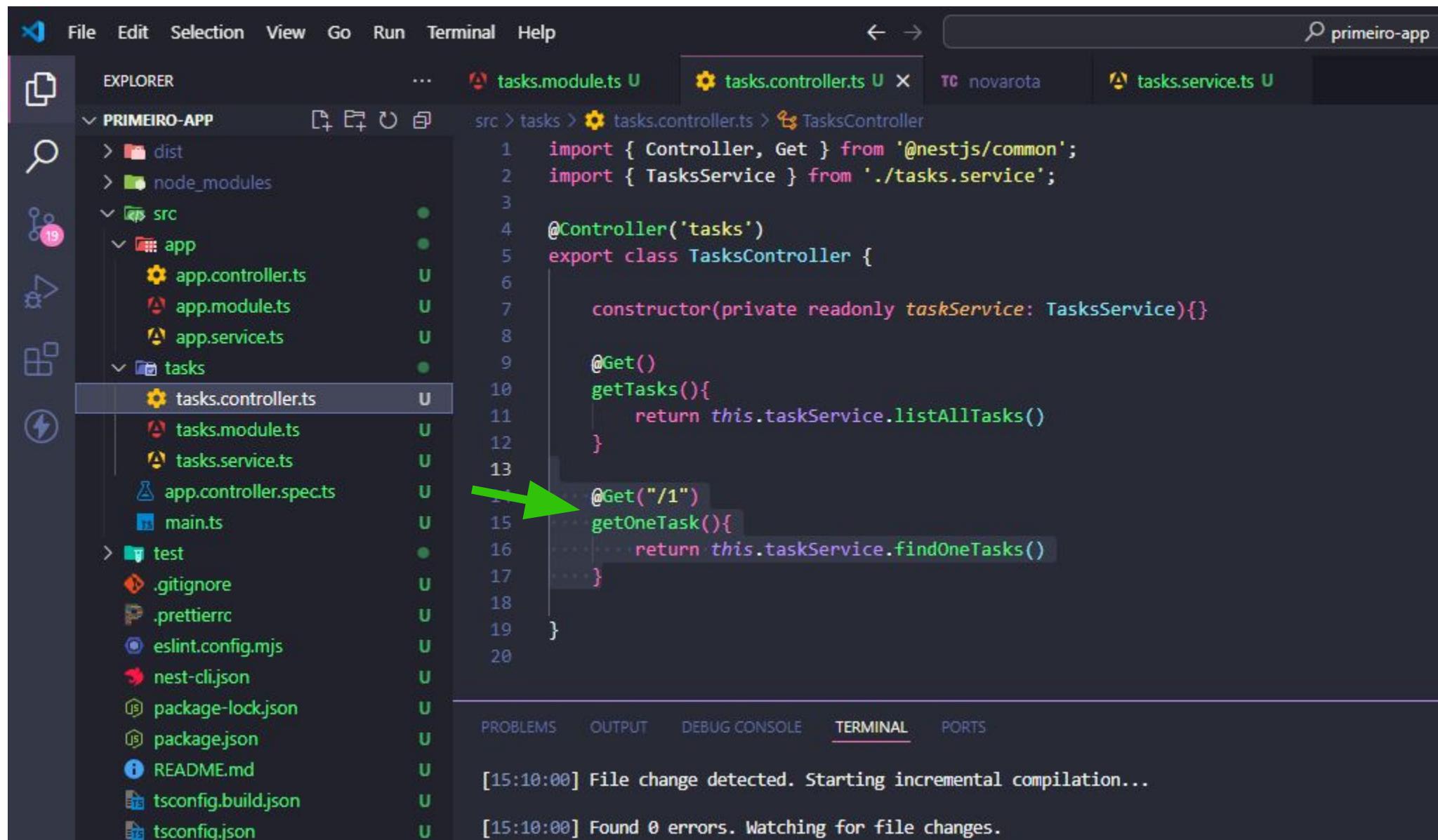


The screenshot shows the VS Code interface with the file `tasks.service.ts` open. The Explorer sidebar on the left shows the project structure of `PRIMEIRO-APP`, including files like `app.controller.ts`, `app.module.ts`, `app.service.ts`, and `tasks` folder containing `tasks.controller.ts`, `tasks.module.ts`, and `tasks.service.ts`. The `tasks.service.ts` file contains the following code:

```
1 import { Injectable } from '@nestjs/common';
2
3 @Injectable()
4 export class TasksService {
5
6   listAllTasks() {
7     return [
8       { id: 1, task: "Estudar Nest" },
9       { id: 2, task: "Estudar React" },
10      { id: 3, task: "Estudar Angular" }
11    ]
12  }
13
14  findOneTasks(){
15    return "Exibindo apenas a tarefa 1"
16  }
17
18}
```

A green arrow points to the `findOneTasks()` method at line 14, which returns the string "Exibindo apenas a tarefa 1".

Gerando Service / Provider



The screenshot shows the VS Code interface with the following details:

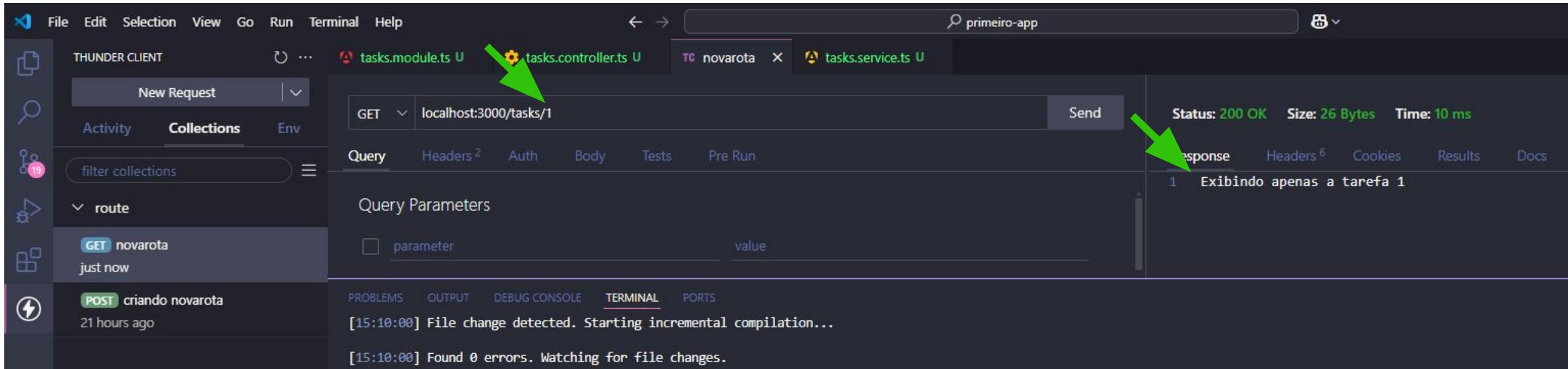
- File Explorer:** Shows the project structure under "PRIMEIRO-APP". The "tasks" folder contains "tasks.controller.ts", which is currently selected.
- Editor:** Displays the code for "tasks.controller.ts".

```
1 import { Controller, Get } from '@nestjs/common';
2 import { TasksService } from './tasks.service';
3
4 @Controller('tasks')
5 export class TasksController {
6   constructor(private readonly taskService: TasksService){}
7
8   @Get()
9   getTasks(){
10     return this.taskService.listAllTasks()
11   }
12
13   @Get("/1")
14   getOneTask(){
15     return this.taskService.findOneTask()
16   }
17 }
18
19 }
```

A green arrow points to the `@Get("/1")` annotation.
- Terminal:** Shows log messages:

```
[15:10:00] File change detected. Starting incremental compilation...
[15:10:00] Found 0 errors. Watching for file changes.
```

Gerando Service / Provider



Atividade

- 1) Crie as camadas: Guests, Users e Teachers**
- 2) Implemente sem utilizar a CLI: Module, Controller e Service**
- 3) Implemente o verbo Get, para listar tudo e por id**
- 4) Teste as rotas no Postman ou Thunder Client**