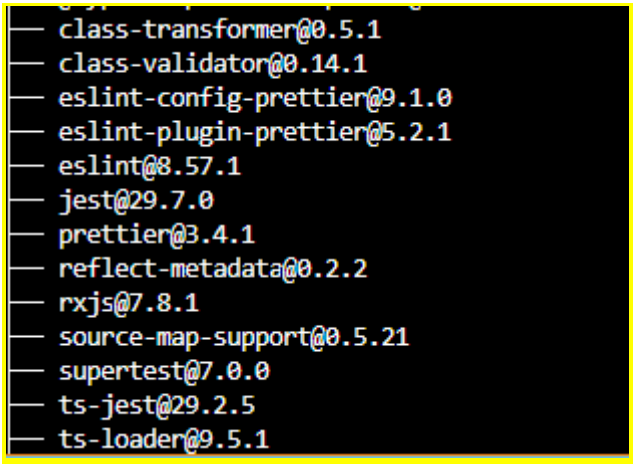
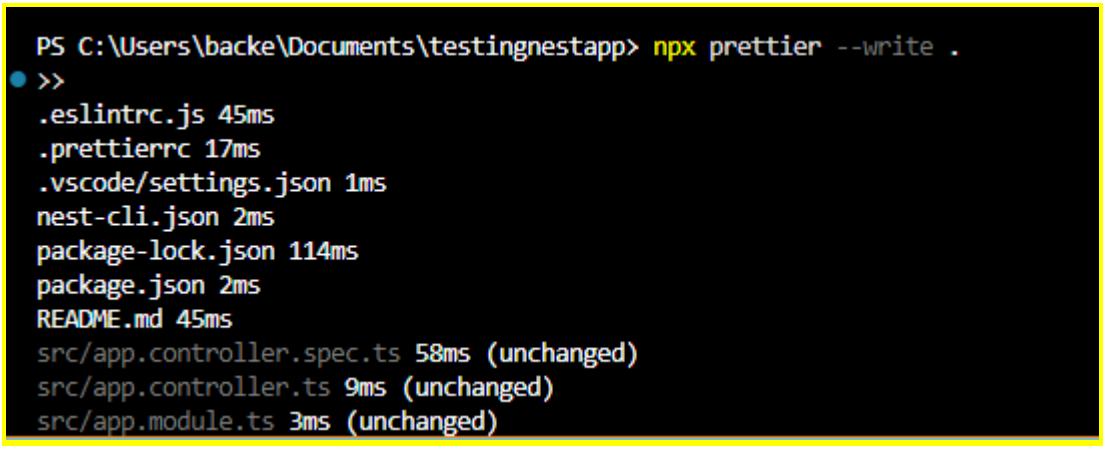
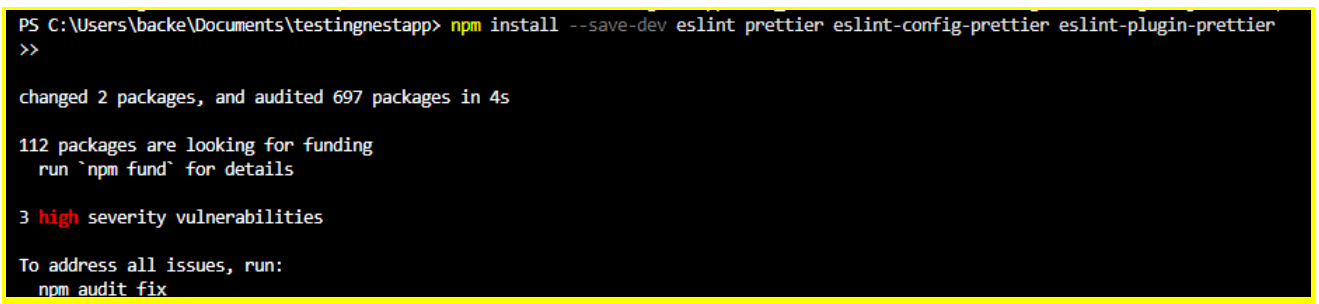
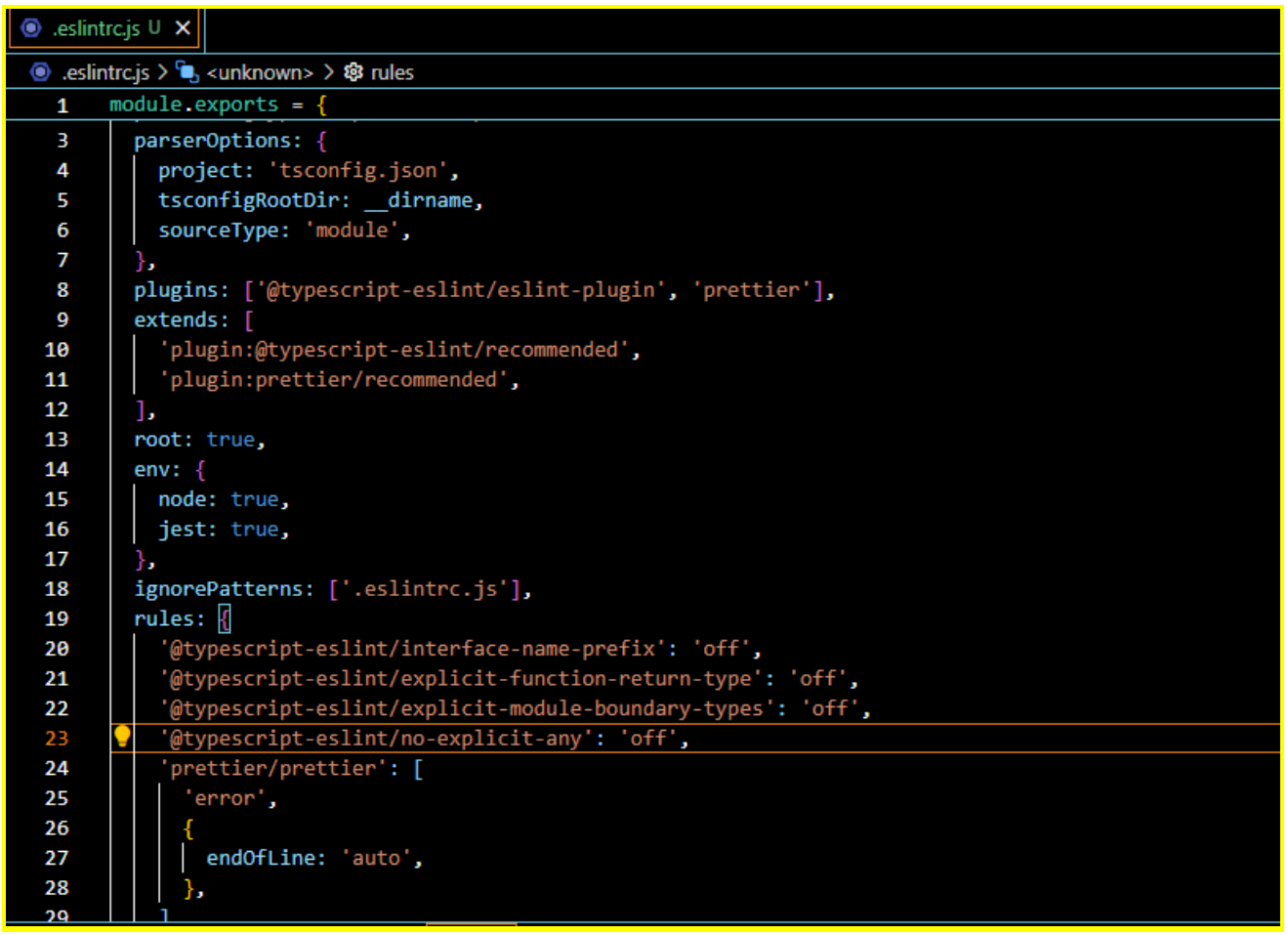
PUNTO 7: Clean code

Para la implementación del clean code estamos utilizando las herramientas de ***Prettier*** y ***ESlinter***, pero el uso de estas herramientas no resulta siendo más allá de anecdótico esto porque el framework que hemos usado (***NestJS***) no presenta necesidad de generar una configuración ya que este paquete viene listo para ser usado solamente con la invocación de un comando si no se tiene instalada la extensión de dichos linters, por tanto generalmente el código al menos en forma de todos los integrantes del equipo va a ser similar por las restricciones que generan estos linters (por ejemplo ESlinter no permite ejecutar si no esta todo según la configuración dada por defecto). A pesar de ello, se adjunta ejemplo de la sencilla implementación a continuación:

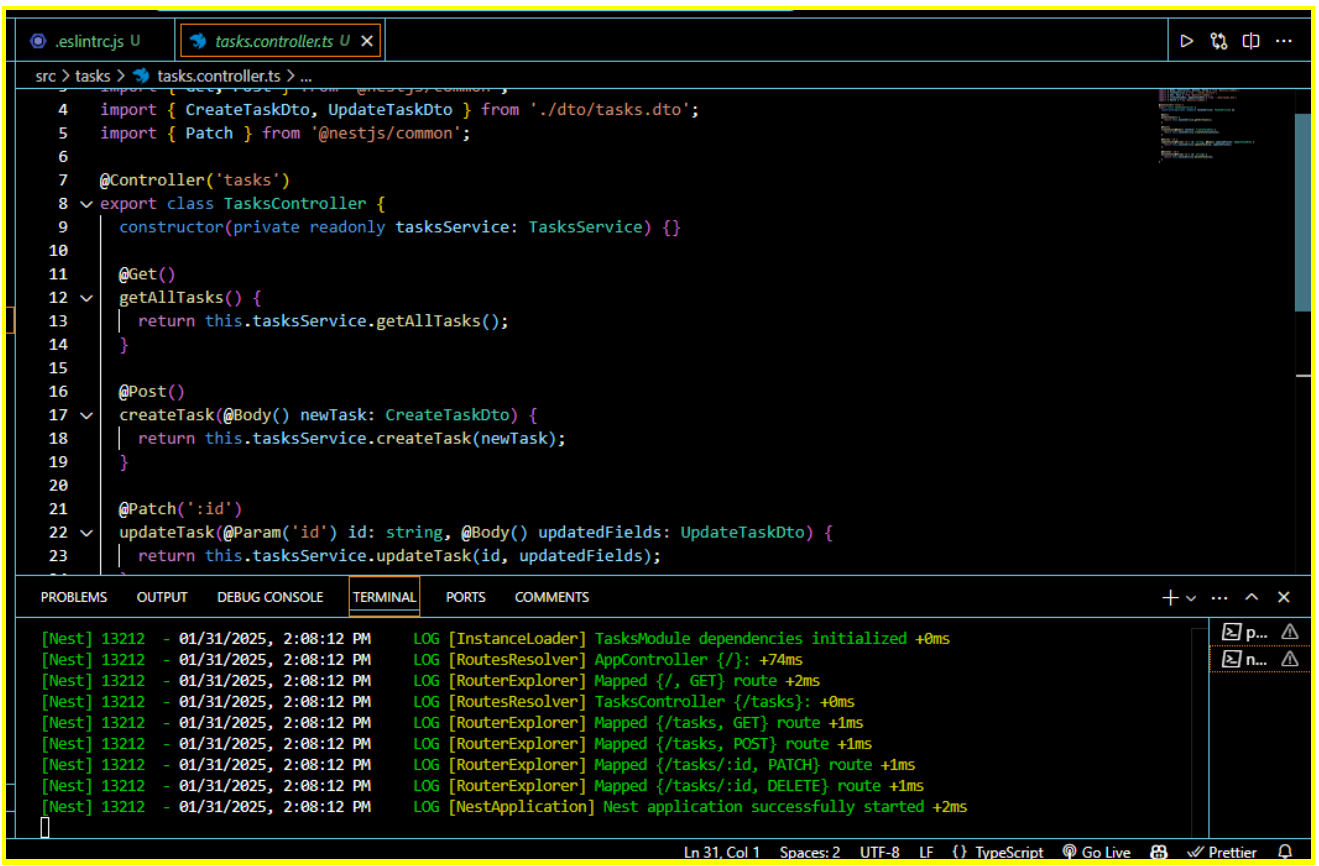








```
.eslintrc.js U
.eslintrc.js > <unknown> > rules
1   module.exports = {
3     parserOptions: {
4       project: 'tsconfig.json',
5       tsconfigRootDir: __dirname,
6       sourceType: 'module',
7     },
8     plugins: ['@typescript-eslint/eslint-plugin', 'prettier'],
9     extends: [
10      'plugin:@typescript-eslint/recommended',
11      'plugin:prettier/recommended',
12    ],
13    root: true,
14    env: {
15      node: true,
16      jest: true,
17    },
18    ignorePatterns: ['.eslintrc.js'],
19    rules: {
20      '@typescript-eslint/interface-name-prefix': 'off',
21      '@typescript-eslint/explicit-function-return-type': 'off',
22      '@typescript-eslint/explicit-module-boundary-types': 'off',
23      '@typescript-eslint/no-explicit-any': 'off',
24      'prettier/prettier': [
25        'error',
26        {
27          endOfLine: 'auto',
28        },
29      ]
```



```
.eslintrc.js U   tasks.controller.ts U
src > tasks > tasks.controller.ts > ...
4   import { CreateTaskDto, UpdateTaskDto } from './dto/tasks.dto';
5   import { Patch } from '@nestjs/common';
6
7   @Controller('tasks')
8   export class TasksController {
9     constructor(private readonly tasksService: TasksService) {}
10
11    @Get()
12    getAllTasks() {
13      return this.tasksService.getAllTasks();
14    }
15
16    @Post()
17    createTask(@Body() newTask: CreateTaskDto) {
18      return this.tasksService.createTask(newTask);
19    }
20
21    @Patch(':id')
22    updateTask(@Param('id') id: string, @Body() updatedFields: UpdateTaskDto) {
23      return this.tasksService.updateTask(id, updatedFields);

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  COMMENTS

[Nest] 13212  - 01/31/2025, 2:08:12 PM     LOG [InstanceLoader] TasksModule dependencies initialized +0ms
[Nest] 13212  - 01/31/2025, 2:08:12 PM     LOG [RoutesResolver] AppController {/}: +74ms
[Nest] 13212  - 01/31/2025, 2:08:12 PM     LOG [RouterExplorer] Mapped {/, GET} route +2ms
[Nest] 13212  - 01/31/2025, 2:08:12 PM     LOG [RoutesResolver] TasksController {/tasks}: +0ms
[Nest] 13212  - 01/31/2025, 2:08:12 PM     LOG [RouterExplorer] Mapped {/tasks, GET} route +1ms
[Nest] 13212  - 01/31/2025, 2:08:12 PM     LOG [RouterExplorer] Mapped {/tasks, POST} route +1ms
[Nest] 13212  - 01/31/2025, 2:08:12 PM     LOG [RouterExplorer] Mapped {/tasks/:id, PATCH} route +1ms
[Nest] 13212  - 01/31/2025, 2:08:12 PM     LOG [RouterExplorer] Mapped {/tasks/:id, DELETE} route +1ms
[Nest] 13212  - 01/31/2025, 2:08:12 PM     LOG [NestApplication] Nest application successfully started +2ms

Ln 31, Col 1  Spaces: 2  UTF-8  LF  {} TypeScript  Go Live  Prettier
```

```
{
  "extends": ["plugin:prettier/recommended"],
  "rules": {
    "prettier/prettier": ["error", { "endOfLine": "auto" }]
  }
}
```

Con esta configuración, Prettier formatea automáticamente el código al guardar los archivos, asegurando consistencia entre los integrantes del equipo.

Quejas

Las quejas que encontramos ante esta aplicación son:

- Aunque si existe un estándar detrás de la generación de NestJS, las necesidades avanzadas de clean code no pueden ser medidas por parte de los mismos, por ejemplo los nombres de las funciones, la separación de responsabilidades y un largo etcétera.
- La configuración que provee NestJS es bastante básica, y al aplicar una plantilla recomendada predeterminada no conocemos más en el proyecto los principios de formato que usa el linter.
- En general el código de mis compañeros fue fácil de entender porque sigue la estructura del framework, pero hubo partes donde me costó un poco seguir la lógica, principalmente por la falta de experiencia.También vi que en algunos archivos había lógica mezclada, como controladores que manejaban validaciones en lugar de delegarlas a los servicios. La herramienta de análisis ayuda con cosas básicas como formato y errores, pero hay detalles que dependen más del criterio de cada uno. Entendí que era lo que estaban haciendo mis compañeros con sus modificaciones pero si me tocara modificarlo a mí solo, me costaría y tardaría un rato en poder hacerlo bien.