Informe Código Limpio

Presentado por:

Cristian Medina - crmedinab@unal.edu.co

Cristian Montañez - cmontanez@unal.edu.co

Justin Rodriguez - jusrodriguez@unal.edu.co

Sergio Ruiz – seruizh@unal.edu.co

Profesor:

Oscar Eduardo Alvarez Rodriguez

31/01/2024

Universidad Nacional de Colombia

Facultad de Ingeniería

Ingeniería de Software

Departamento de Ingeniería de Sistema

Código Limpio

En el desarrollo de software, mantener un código limpio y bien estructurado es fundamental para garantizar la mantenibilidad, escalabilidad y una colaboración eficiente dentro del equipo. En este informe, detallaremos la implementación de linters y de formateadores en nuestro proyecto, el cual consta de dos componentes principales: backend y frontend. Explicaremos el proceso paso a paso para configurar y aplicar herramientas de análisis estático en cada parte del sistema, asegurando el cumplimiento de buenas prácticas de codificación, la detección temprana de errores y la mejora en la calidad del código.

Backend

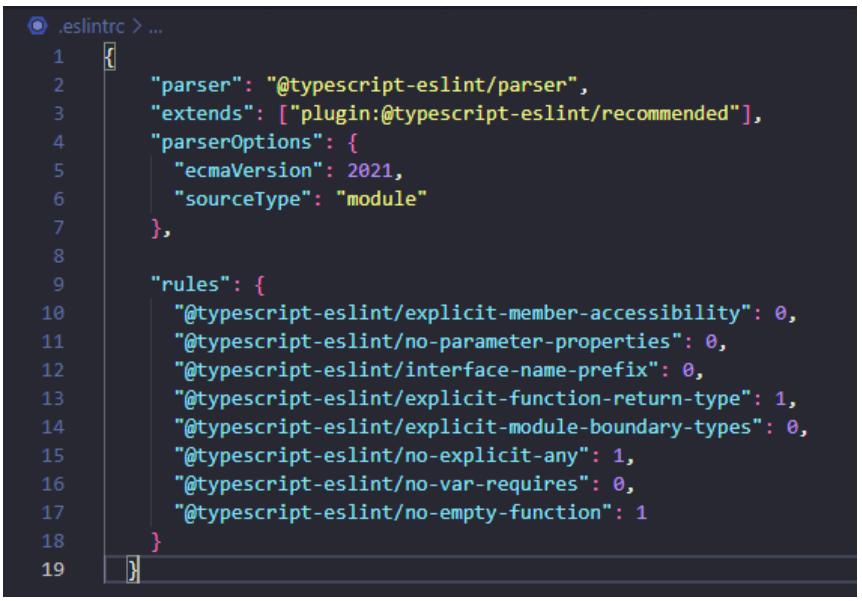
Para el desarrollo del backend, utilizaremos los linters ESLint, ya que es el más común y utilizado en proyectos que emplean JavaScript y TypeScript. En nuestro caso, trabajamos con Express y TypeScript para construir nuestro backend.

Instalación ESlint y paquetes para manejo de typescript

```
npm install eslint @typescript-eslint/eslint-plugin @typescript-eslint/parser -D
```

```
added 113 packages, and audited 260 packages in 9s

54 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Creación archivo .eslintrc en la raíz del proyecto con la siguiente configuración



```
.eslintrc > ...
{
    "parser": "@typescript-eslint/parser",
    "extends": ["plugin:@typescript-eslint/recommended"],
    "parserOptions": {
      "ecmaVersion": 2021,
      "sourceType": "module"
    },

    "rules": {
      "@typescript-eslint/explicit-member-accessibility": 0,
      "@typescript-eslint/no-parameter-properties": 0,
      "@typescript-eslint/interface-name-prefix": 0,
      "@typescript-eslint/explicit-function-return-type": 1,
      "@typescript-eslint/explicit-module-boundary-types": 0,
      "@typescript-eslint/no-explicit-any": 1,
      "@typescript-eslint/no-var-requires": 0,
      "@typescript-eslint/no-empty-function": 1
    }
}
```

Añadir los scripts lint y format al package.json para la ejecución y formateo del código

```
Debug
"scripts": {
  "dev": "nodemon src/server-with-mysql.ts",
  "start": "node build/index.js",
  "tsc": "tsc",
  "test": "echo \"Error: no test specified\" && exit 1",
  "lint": "eslint src/**/*.ts",
  "format": "eslint src/**/*.ts --fix"
},
```

Instalamos prettier para el formateo automático del código

```
npm install prettier eslint-config-prettier eslint-plugin-prettier -D
```

```
added 8 packages, and audited 268 packages in 4s

58 packages are looking for funding
  run `npm fund` for details
```

Creación del archivo .prettierrc en la raíz del proyecto con la siguiente configuración

.prettierrc > ...

```
{
    "semi": true,
    "trailingComma": "all",
    "singleQuote": true,
    "printWidth": 120,
    "tabWidth": 2
}
```

Añadimos la configuración del prettier al archivo .eslintrc

.eslintrc > ...

```
{
  "parser": "@typescript-eslint/parser",
  "extends": ["prettier", "plugin:@typescript-eslint/recommended", "plugin:prettier/recommended"],
  "parserOptions": {
    "ecmaVersion": 2021,
    "sourceType": "module"
  },

  "rules": {
    "@typescript-eslint/explicit-member-accessibility": 0,
    "@typescript-eslint/no-parameter-properties": 0,
    "@typescript-eslint/interface-name-prefix": 0,
    "@typescript-eslint/explicit-function-return-type": 1,
    "@typescript-eslint/explicit-module-boundary-types": 0,
    "@typescript-eslint/no-explicit-any": 1,
    "@typescript-eslint/no-var-requires": 0,
    "@typescript-eslint/no-empty-function": 1
  }
}
```

Ejecutamos npm run lint y se nos muestra el siguiente resultado

```
npm run lint
```

```
  3:47  warning  Delete `␍`                prettier/prettier
  4:1   warning  Delete `␍`                prettier/prettier
  5:55  warning  Replace `␍` with `;`      prettier/prettier
  6:60  warning  Delete `␍`                prettier/prettier

✖54 problems (1 error, 53 warnings)
  1 error and 53 warnings potentially fixable with the `--fix` option.
```

Ejecutamos npm run format para realizar automáticamente todas las correcciones

```
npm run format
```

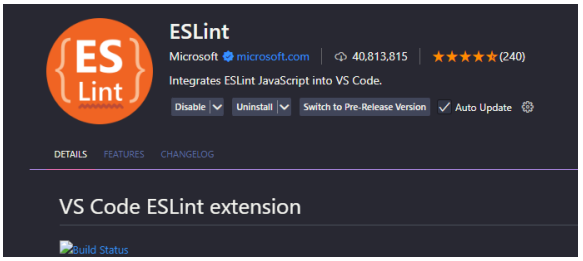
```
> cosas-perdidas-un-backend@1.0.0 format
> eslint src/**/*.ts --fix
```

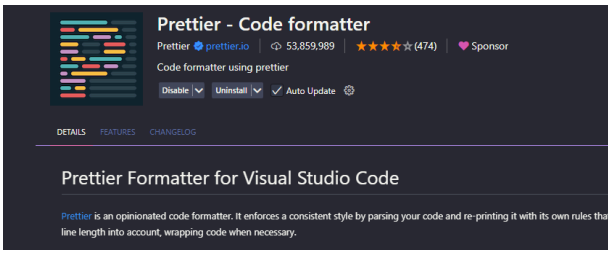
Una vez realizadas todas las correcciones ejecutamos nuevamente npm run lint en busca de errores que quizás no puedan ser solucionados automáticamente y vemos que da como resultado que no hay errores o warnings.

```
npm run lint
```

```
> cosas-perdidas-un-backend@1.0.0 lint
> eslint src/**/*.ts

PS H:\Hub\Uni\IngeSoft1\ProyectoRepo\Software_Engineering\Proyecto\backend>
```

Adicionalmente instalamos las extensiones de vscode Eslint y Prettier code formater para ver los errores en el código.





Ejemplo de código con errores (hacer zoom)

```
You, 16 hours ago | 1 author (You)
import { Router } from 'express'    Insert `;`
import { ObjectController } from '../controllers/ObjectController';    Already included file name 'h:/Hub/Uni/IngeSoft1/ProyectoRepo/Software_Engineering/Proyecto/backend/src/contro
import ObjectModel from '../models/ObjectModel';
import { authenticate } from '../middlewares/authMiddleware';

        Delete `·····↵`
export const createObjectRouter = (objectModel: ObjectModel): Router => {    'objectModel' is defined but never used.
  const objectRouter = Router();
  const objectController = new ObjectController(ObjectModel);
          Delete `·····`
  // Obtener todos los objetos
  objectRouter.get('/:user_id/objects', authenticatd, objectController.getAllObjects);    Cannot find name 'authenticatd'. Did you mean 'authenticate'?


   Delete `↵`
  // Buscar objetos por categoría, ubicación, rango de fechas y palabras clave
  objectRouter.get("/:user_id/objects/filters", authenticate, objectController.searchObjects);    Replace `"/:user_id/objects/filters"` with `'/:user_id/objects/filters'`

  return objectRouter;
}    Insert `;`
```

Ejemplo de código corregido

```
You, 16 hours ago | 1 author (You)
import { Router } from 'express';
import { ObjectController } from '../controllers/objectController';
import ObjectModel from '../models/ObjectModel';
import { authenticate } from '../middlewares/authMiddleware';

export const createObjectRouter = (objectModel: ObjectModel): Router => {
  const objectRouter = Router();
  const objectController = new ObjectController(objectModel);

  // Obtener todos los objetos
  objectRouter.get('/:user_id/objects', authenticate, objectController.getAllObjects);

  // Buscar objetos por categoría, ubicación, rango de fechas y palabras clave
  objectRouter.get('/:user_id/objects/filters', authenticate, objectController.searchObjects);

  return objectRouter;
};
```

Frontend

El frontend se está desarrollando en React y JavaScript, por lo tanto hemos decidido utilizar ESLint como linter y Prettier como formateador. Esto nos permite mantener un código limpio y uniforme, asegurando que todos los miembros del equipo sigan las mismas reglas de codificación y estilos definidos en el proyecto.

Instalación ESlint

```
PS D:\Documentos\1. Universidad\6. Sexto Semestre\Ingesoft\Software_Engineering\Proyecto\frontend> npm i -D eslint

added 1 package, changed 2 packages, and audited 243 packages in 16s

110 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

```
PS D:\Documentos\1. Universidad\6. Sexto Semestre\Ingesoft\Software_Engineering\Proyecto\frontend> npx eslint --init
You can also run this command directly using 'npm init @eslint/config@latest'.

> frontend@0.0.0 npx
> create-config

@eslint/create-config: v1.4.0

√ How would you like to use ESLint? · syntax
√ What type of modules does your project use? · esm
√ Which framework does your project use? · react
√ Does your project use TypeScript? · javascript
√ Where does your code run? · browser
The config that you've selected requires the following dependencies:

eslint, globals, eslint-plugin-react
√ Would you like to install them now? · No / Yes
√ Which package manager do you want to use? · npm
Installing...

up to date, audited 243 packages in 1s

110 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Archivo eslint.config.js

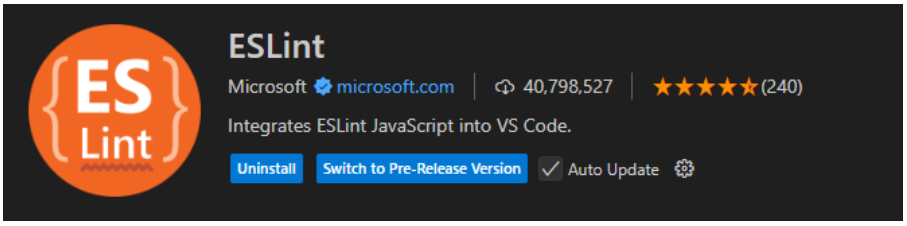
```
You, 53 seconds ago | 1 author (You)
1  import globals from "globals";      You, 29 seconds ago • Uncommitted changes
2  import pluginReact from "eslint-plugin-react";
3
4
5  /** @type {import('eslint').Linter.Config[]} */
6  export default [
7    {files: ["**/*.{js,mjs,cjs,jsx}"]},
8    {languageOptions: { globals: globals.browser }},
9    pluginReact.configs.flat.recommended,
10 ];
```

Como se puede ver, la configuración fue modificada a partir de la nueva inicialización de ESlint, esto ya que al crear un proyecto react con vite se crea un archivo eslint por defecto.

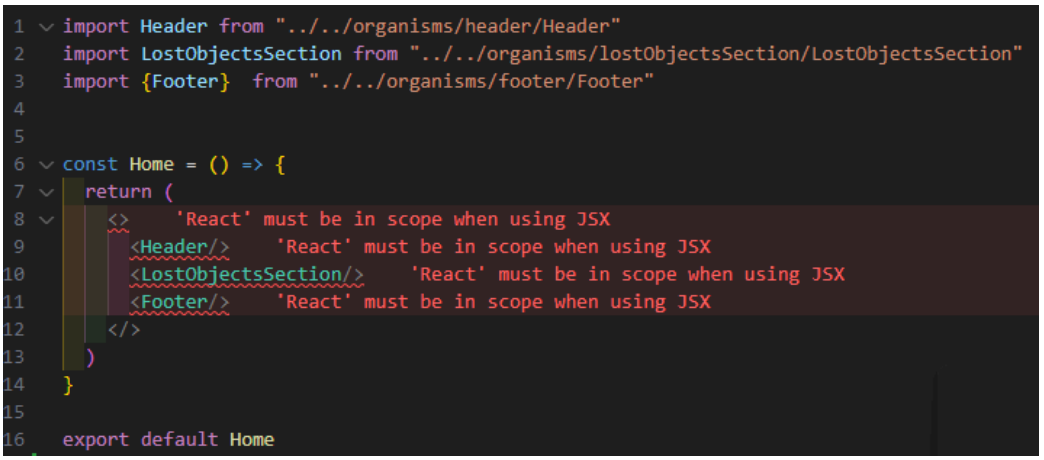
Detección De errores por consola

```
PS D:\Documentos\1. Universidad\6. Sexto Semestre\Ingesoft\Software_Engineering\Proyecto\frontend> npx eslint src/main.jsx
Warning: React version not specified in eslint-plugin-react settings. See https://github.com/jsx-eslint/eslint-plugin-react#configuration .

D:\Documentos\1. Universidad\6. Sexto Semestre\Ingesoft\Software_Engineering\Proyecto\frontend\src\main.jsx
  6:5  error  'React' must be in scope when using JSX  react/react-in-jsx-scope

✖ 1 problem (1 error, 0 warnings)
```

Uso de extensión para ver errores en el código

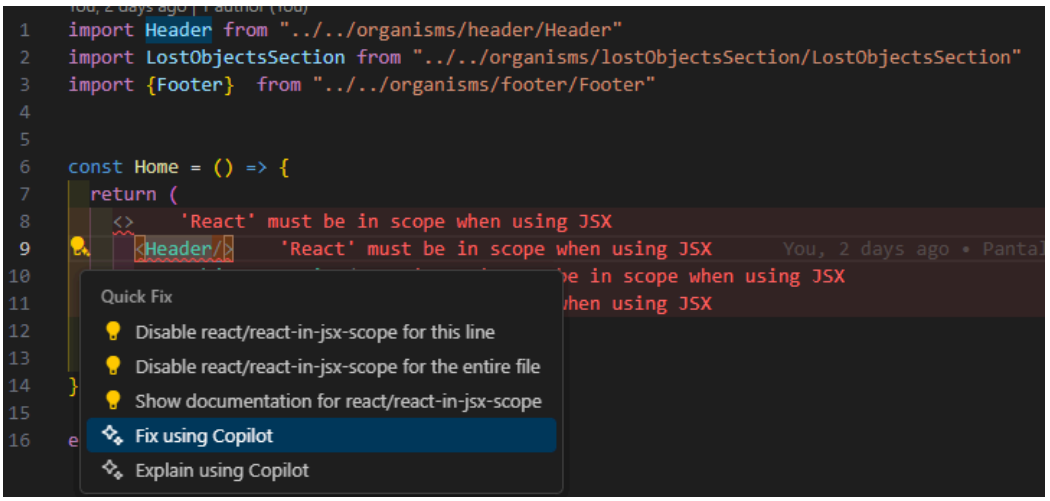


A Partir de la instalación se pueden ver los errores en el código



```
import Header from "../../organisms/header/Header"
import LostObjectsSection from "../../organisms/lostObjectsSection/LostObjectsSection"
import {Footer}  from "../../organisms/footer/Footer"


const Home = () => {
  return (
    <>    'React' must be in scope when using JSX
      <Header/>    'React' must be in scope when using JSX
      <LostObjectsSection/>    'React' must be in scope when using JSX
      <Footer/>    'React' must be in scope when using JSX
    </>
  )
}

export default Home
```

Haciendo uso de copilot y eslint arreglamos los errores mostrados



```
import Header from "../../organisms/header/Header"
import LostObjectsSection from "../../organisms/lostObjectsSection/LostObjectsSection"
import {Footer}  from "../../organisms/footer/Footer"


const Home = () => {
  return (
    <>    'React' must be in scope when using JSX
      <Header/>    'React' must be in scope when using JSX      You, 2 days ago • Panta
                                                  e in scope when using JSX
                                                  hen using JSX
Quick Fix
Disable react/react-in-jsx-scope for this line
Disable react/react-in-jsx-scope for the entire file
Show documentation for react/react-in-jsx-scope
Fix using Copilot
Explain using Copilot
```

```jsx
import React from 'react';
import Header from "../../organisms/header/Header"
import LostObjectsSection from "../../organisms/lostObjectsSection/LostObjectsSection"
import {Footer}  from "../../organisms/footer/Footer"


const Home = () => {
  return (
    <>
      <Header/>
      <LostObjectsSection/>
      <Footer/>
    </>
  )
}
      You, 2 days ago • Pantalla home creada, con su respectivo respons…
export default Home
```

Como se puede ver, por el momento ESLint está dando un error porque no se está importando React. Sin embargo, con Vite no es necesario, ya que utiliza la nueva transformación introducida en React 17. Por lo tanto, no necesitamos importar React en todos los archivos, y tampoco es correcto hacerlo. Por esta razón, decidimos configurar esta propiedad

```
    rules: {
      ...js.configs.recommended.rules,
      ...react.configs.recommended.rules,
      ...react.configs['jsx-runtime'].rules,
      ...reactHooks.configs.recommended.rules,
      'react/jsx-no-target-blank': 'off',
      'react-refresh/only-export-components': [
        'warn',
        { allowConstantExport: true },
      ],
      'reac/react-in-jsx-scope': 'off',
    },
  },
]
```

Instalación de prettier

```
PS D:\Documentos\1. Universidad\6. Sexto Semestre\Ingesoft\Software_Engineering\Proyecto\frontend> npm i -D prettier

added 1 package, and audited 244 packages in 3s

111 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Uso de Prettier

```
PS D:\Documentos\1. Universidad\6. Sexto Semestre\Ingesoft\Software_Engineering\Proyecto\frontend> npm i -D prettier
PS D:\Documentos\1. Universidad\6. Sexto Semestre\Ingesoft\Software_Engineering\Proyecto\frontend> npm i -D prettier

added 1 package, and audited 244 packages in 3s

111 packages are looking for funding
  run `npm fund` for details
  run `npm fund` for details

found 0 vulnerabilities
PS D:\Documentos\1. Universidad\6. Sexto Semestre\Ingesoft\Software_Engineering\Proyecto\frontend> npx prettier src/main.jsx
import { createRoot } from "react-dom/client";
import "./main.scss";
import App from "./App.jsx";

createRoot(document.getElementById("root")).render(<App />);
PS D:\Documentos\1. Universidad\6. Sexto Semestre\Ingesoft\Software_Engineering\Proyecto\frontend> npx prettier src/main.jsx --write
src/main.jsx 45ms
```

Configuración

.prettierrc > ...

```
{
  "tabWidth": 2,
  "semi": true,
  "singleQuote": true,
  "jsxSingleQuote": true,
  "trailingComma": "all"
}
```

Antes

```
import Header from "../../organisms/header/Header"
import BasicInformation from "../../organisms/basicInformation/BasicInformation"
import {Footer}  from "../../organisms/footer/Footer"


const LandPage = () => {
  return (
    <>
      <Header/>
      <BasicInformation/>
      <Footer/>
    </>
  )
}

export default LandPage
```

Después

```
import Header from '../../organisms/header/Header';
import BasicInformation from '../../organisms/basicInformation/BasicInformat
import { Footer } from '../../organisms/footer/Footer';

const LandPage = () => {
  return (
    <>
      <Header />
      <BasicInformation />
      <Footer />
    </>
  );
};

export default LandPage;
```

Cristian Javier Medina Barrios
Gracias a la implementación de los linters, la definición de las arquitecturas y estructuras de los proyectos por separado (frontend y backend) y la repartición organizada de tareas, se me ha facilitado la lectura del código de mis compañeros.
A pesar de que mi campo ha estado más enfocado en el backend y mi conocimiento acerca del frontend es un poco limitado, específicamente en React, siento que podría realizar modificaciones simples. En caso de que se trate de algo más avanzado, con el tiempo y la disposición suficiente, lograría implementarlo, en gran parte, gracias a los estándares definidos por el equipo, los cuales han sido de gran ayuda en mi aprendizaje, y también a que anteriormente tuve cierta experiencia con un framework de frontend.

Cristian Daniel Montanez Pineda
Como desarrollador frontend, he podido apreciar que el código de mis compañeros resulta bastante legible y coherente, gracias a la estructura y los estándares que definimos desde el inicio del proyecto. Aunque aún estoy consolidando algunos conocimientos, la utilización de metodologías como BEM y el enfoque de diseño anatómico facilita identificar en qué está trabajando cada integrante del equipo y hacen que las modificaciones sean mucho más sencillas de implementar. En lo que respecta al backend, reconozco que mi conocimiento es limitado y apenas entiendo algunos de sus aspectos, sin embargo, me esfuerzo por aprender y comprender su estructura para poder colaborar de manera efectiva.

Justin Rodriguez Sanchez
En este proyecto mi rol ha sido como desarrollador front, estamos utilizando react, tecnología que es nueva para mi y este proyecto a sido la oportunidad para aprender y fortalecer mi habilidades en este campo, por este motivo leer el código no se me facilita tanto, sin embargo gracias a la estructura que definimos desde el inicio del proyecto ha sido más fácil comprender y entender el código que realizan mis compañeros de equipo, todos intentamos utilizar las mejores prácticas, realizando retroalimentaciones y resolviendo preguntas que tengamos entre todos. El hecho de que todos tengamos clara la estructura y tipo de arquitectura (atómica) hace que las modificaciones también pueden ser más sencillas de hacer. Este proyecto ha sido un reto para mi, pese a que aun me faltan bastantes cosas que aprender, sé que los estándares que designamos desde un inicio se están cumpliendo. Por otro lado en cuanto al back es una tecnología que si desconozco en su mayoría, por lo que hacer cualquier modificación sería bastante compleja de hacer para mi, sin embargo el hecho de que se tenga la misma claridad de estructura definida desde el inicio así como en el front, también va a hacer que cualquier modificación sea más fácil de implementar en el momento que mis conocimientos sobre desarrollo back crezcan.

Sergio Ruiz Hurtado
Desde el inicio del proyecto, hicimos una distribución de responsabilidades y roles. Entonces en cada entrega, cada uno de nosotros tiene definido sus tareas, lo que nos permite saber en que está trabajando cada uno.
En cuanto al código, por el momento mi rol es trabajar en el desarrollo del frontend junto con otros dos compañeros, al usar la arquitectura atómica y la reutilización de componentes, uso y

leo el codigo que ellos, el cual está bien componetizado y es fácil de modificar, donde todos seguimos las reglas de la metodología BEM y la arquitectura atómica.

Sin embargo, aún no he revisado el código de la parte de backend, ya que al seguir la metodología Git Flow, la rama de backend está separada del frontend. Por otro lado, las pull requests del frontend sí son revisadas.