



Міністерство освіти і науки України  
Національний технічний університет України  
“Київський політехнічний інститут імені Ігоря Сікорського”  
Факультет інформатики та обчислювальної техніки  
Кафедра інформаційних систем та технологій

## Лабораторна робота №2

### **Налаштування проекту та інструментів розробки**

Виконали:  
Палінчак І.В.  
Хохол М.В.

Перевірив:  
Нечай Д.О.

Київ 2024

Завдання:

1. Створити пакети
2. Вибрати стиль коду
3. Налаштувати форматтер
4. Налаштувати лінер або інший статичний аналізатор
5. Налаштувати Git-hook на коміт та пуш. Перевіряти форматування, лінер, тести(для початку завжди успішна команда), збірку\компіляцію проекту (фронт\TS\...).

## Хід роботи

**Пакетний менеджер** — це інструмент, який дозволяє керувати залежностями та встановлювати пакети в проєкті. У даному випадку було обрано **npm** (Node Package Manager), який є стандартним менеджером пакетів для JavaScript і дозволяє автоматично завантажувати й оновлювати пакети для проєкту, а також керувати скриптами для виконання різних завдань. Використано **npm** для налаштування залежностей у фронтенд-частині проєкту на **React**.

Завдяки налаштуванням лінера, форматтера та Git-хуків проєкт отримав більш стабільну та якісну структуру коду.

Налаштовано пакети для Front-End та Back-end частини застосунку

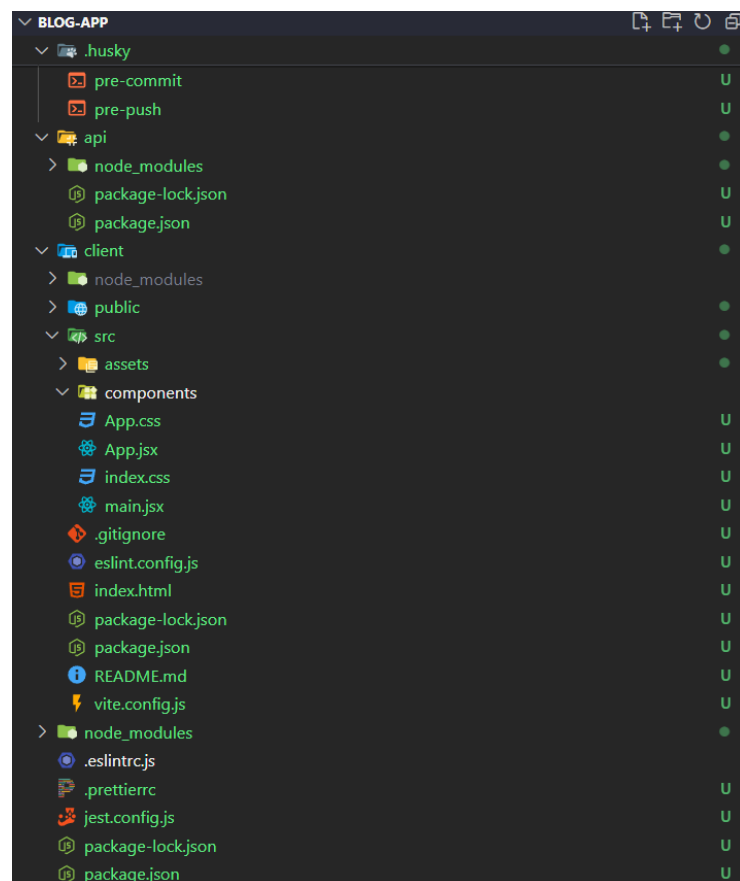


Рисунок 1 – структура частини проєкту після створення

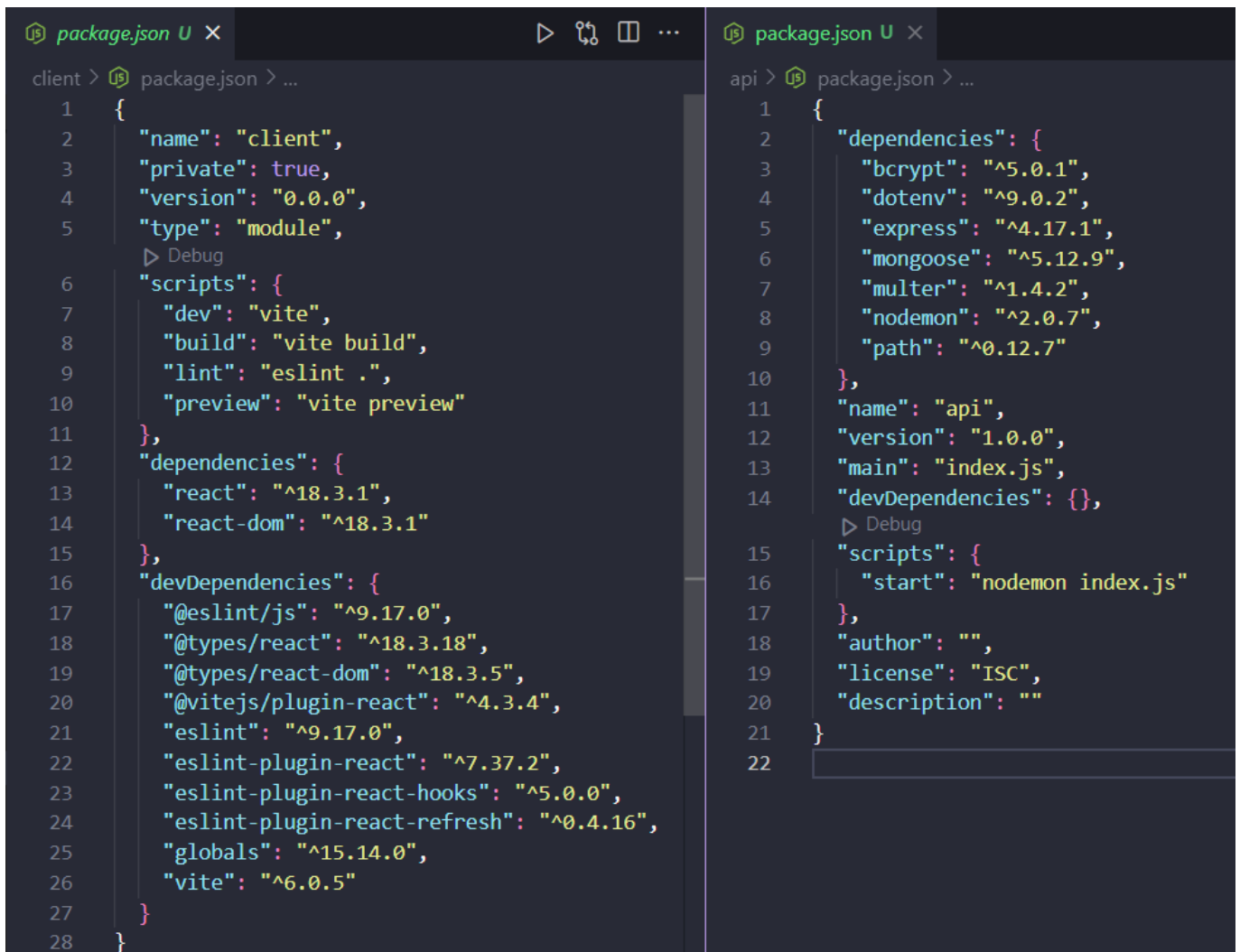


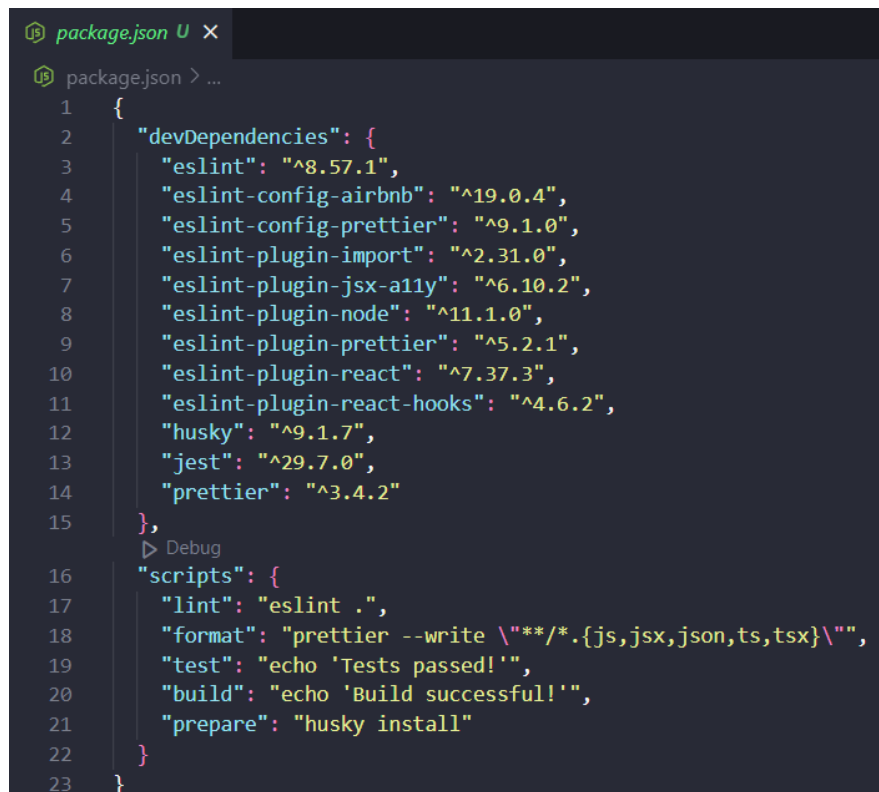
Рисунок 2 – Налаштування пакетів що відповідають за frontend та backend частини застосунка

У файлі .prettierrc були налаштовані головні параметри:



Рисунок 3 – Налаштування prettier

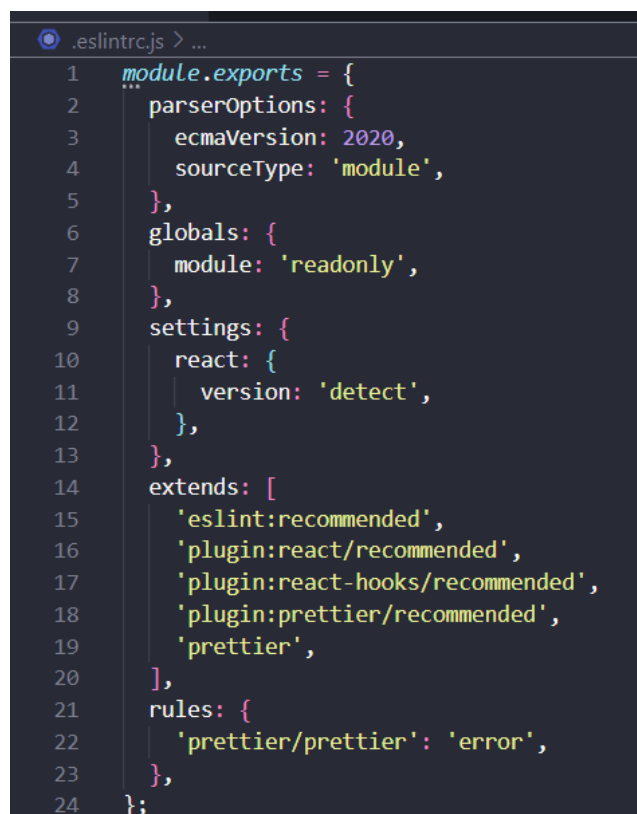
В пакеті `package.json` в корні проєкту також додано скрипт для автоматичного форматування всіх файлів проєкту за допомогою команди:



```
1 {
2   "devDependencies": {
3     "eslint": "^8.57.1",
4     "eslint-config-airbnb": "^19.0.4",
5     "eslint-config-prettier": "^9.1.0",
6     "eslint-plugin-import": "^2.31.0",
7     "eslint-plugin-jsx-a11y": "^6.10.2",
8     "eslint-plugin-node": "^11.1.0",
9     "eslint-plugin-prettier": "^5.2.1",
10    "eslint-plugin-react": "^7.37.3",
11    "eslint-plugin-react-hooks": "^4.6.2",
12    "husky": "^9.1.7",
13    "jest": "^29.7.0",
14    "prettier": "^3.4.2"
15  },
16  "scripts": {
17    "lint": "eslint .",
18    "format": "prettier --write \"**/*.{js,jsx,json,ts,tsx}\"",
19    "test": "echo 'Tests passed!'",
20    "build": "echo 'Build successful!'",
21    "prepare": "husky install"
22  }
23 }
```

Рисунок 4 – Налаштування файлу залежностей в корні проєкту

Основний файл для конфігурації лінера у проєкті



```
1 module.exports = {
2   parserOptions: {
3     ecmaVersion: 2020,
4     sourceType: 'module',
5   },
6   globals: {
7     module: 'readonly',
8   },
9   settings: {
10    react: {
11      version: 'detect',
12    },
13  },
14  extends: [
15    'eslint:recommended',
16    'plugin:react/recommended',
17    'plugin:react-hooks/recommended',
18    'plugin:prettier/recommended',
19    'prettier',
20  ],
21  rules: {
22    'prettier/prettier': 'error',
23  },
24 };
```

Рисунок 5 – Налаштування файлу `.eslintrc.js` в корні проєкту

## Вибір стилю коду:

Для забезпечення єдиного стилю коду в проєкті було обрано гайдлайн від Airbnb для JavaScript та React. Це дозволяє підтримувати високий рівень консистентності та зручності у читанні коду.

## Налаштування форматтеру:

Для автоматичного форматування коду було обрано Prettier. Це інструмент, що забезпечує однорідне форматування коду в проєкті, зокрема стосується таких параметрів, як використання одинарних лапок, наявність ком в кінці елементів та відсутність крапки з комою.

**Git хуки** — це скрипти, які автоматично виконуються на різних етапах роботи з репозиторієм Git, наприклад, перед комітом або пушем. Вони дозволяють автоматизувати процеси, такі як перевірка стилю коду, запуск тестів або форматування коду перед виконанням Git-операцій. Git хуки налаштовуються в каталозі `.git/hooks` або через інструменти, як-от **Husky**, для полегшення управління ними в проєктах.

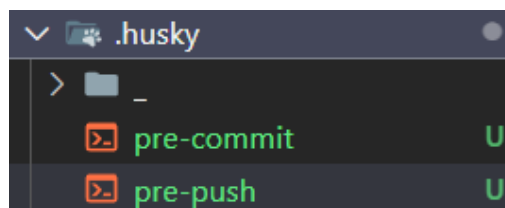
## Налаштування Git-хуків

Для автоматизації перевірок перед комітами та пушами було використано **Husky**. Це дозволяє виконувати певні перевірки, такі як лінтинг, форматування, тести та збірка проєкту, перед виконанням комітів і пушів.

Для налаштування хук команд:

- **pre-commit** хук перевіряє лінтинг і форматування коду.
- **pre-push** хук перевіряє статус тестів і збірки проєкту.

Для цього в `package.json` були додані такі скрипти:



```
PS C:\Users\PC\Desktop\blog\blog-app> git commit -m "testing the husky hook"
> lint
> eslint .

Warning: React version was set to "detect" in eslint-plugin-react settings, but the "react" package is not installed. Assuming latest React version for linting.

> format
> prettier --write "**/*.js,jsx,json,ts,tsx"
```

```
PS C:\Users\PC\Desktop\blog\blog-app> git push

> test
> echo 'Tests passed!'

'Tests passed!'

> build
> echo 'Build successful!'

'Build successful!'
Enumerating objects: 11637, done.
Counting objects: 100% (11637/11637), done.
Delta compression using up to 16 threads
Compressing objects: 100% (8610/8610), done.
Writing objects: 100% (11636/11636), 15.44 MiB | 4.31 MiB/s, done.
Total 11636 (delta 2556), reused 11632 (delta 2554), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2556/2556), done.
To https://github.com/maximkhokhol/blog-app.git
75599d7..27cdc19  main -> main
```

Рисунок 6.1, 6.2, 6.3 – Розташування хуків та скрипт який відбувається перед комітом, якщо перевірка стилізації не пройде – коміт не буде створений, перевірка перед пушем на віддалений репозиторій

#### Висновок:

Виконано налаштування проекту: створено необхідні пакети, обрано стиль коду, впроваджено форматтер Prettier і лінтер ESLint з підтримкою React, а також налаштовано Git-хуки Husky для автоматичної перевірки форматування, лінтингу, тестів і компіляції перед комітом і пушем. Проект готовий до командної розробки з дотриманням стандартів коду та автоматизованими перевітками.