Criação de aplicações front-end utilizando React

por Eder Sampaio - Desenvolvedor Front-End

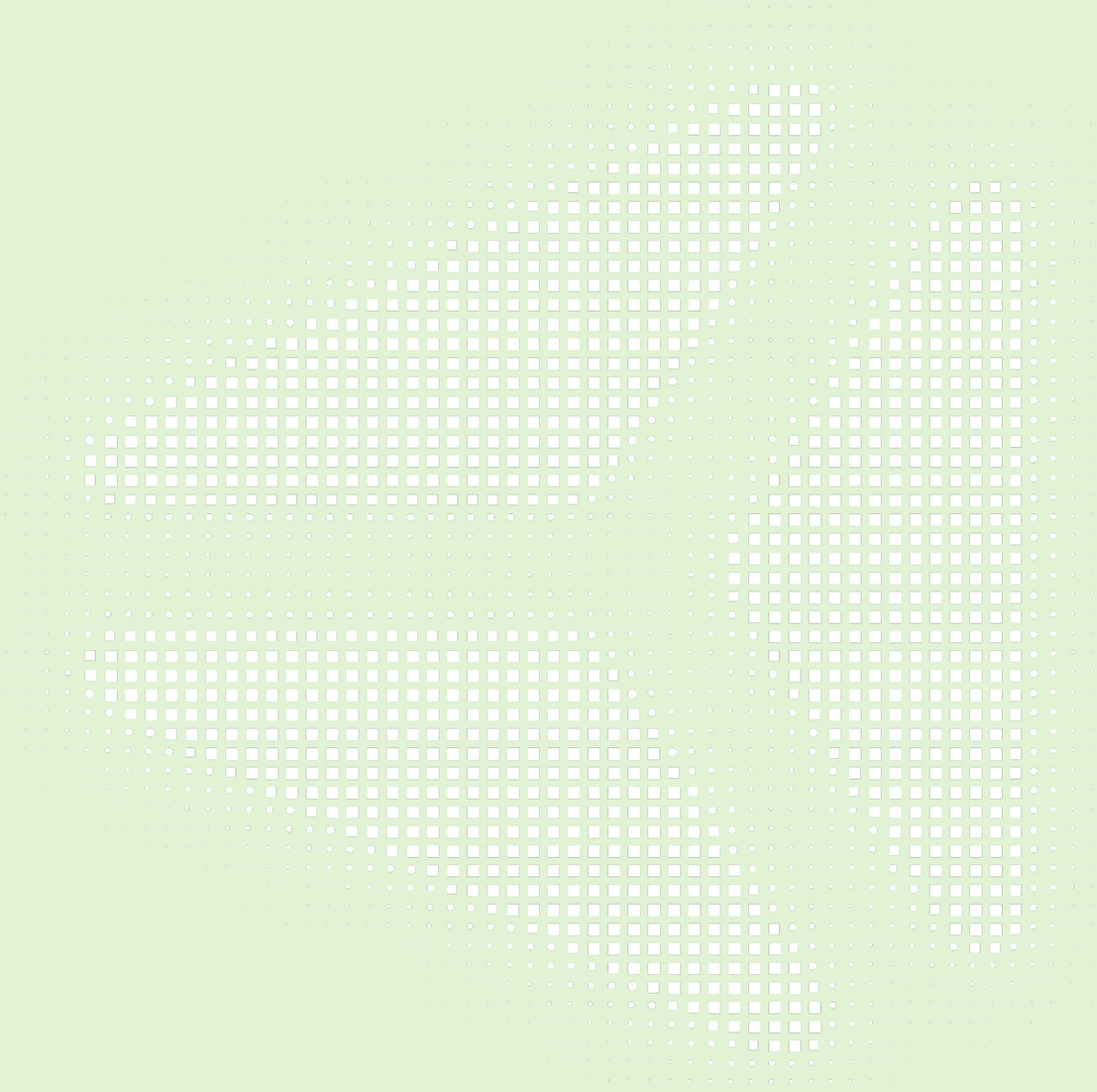
03/11/2022

Área da Trybe: Tecnologia

O que vamos aprender?

- Vite, uma alternativa ao Create React App;
- Iniciando um projeto com Vite;
- Adicionando prop-types;
- Configurando o EditorConfig;
- Configurando o ESLint;
- Configurando o Prettier;
- Configurando o Jest;
- Configurando o React Testing Library;
- Estrutura de diretórios do projeto;
- Automatizando o processo de criação de componentes e hooks (bônus);
- Iniciando um projeto utilizando o template criado.

Vite uma alternativa ao Create React App (CRA)



Vantagens do Vite

- Início instantâneo do servidor;
- Hot Module Replacement (HMR) que permanece rápido, independentemente do tamanho do aplicativo;
- Suporte para TypeScript, JSX, CSS e etc;
- Fácil configuração;
- etc...

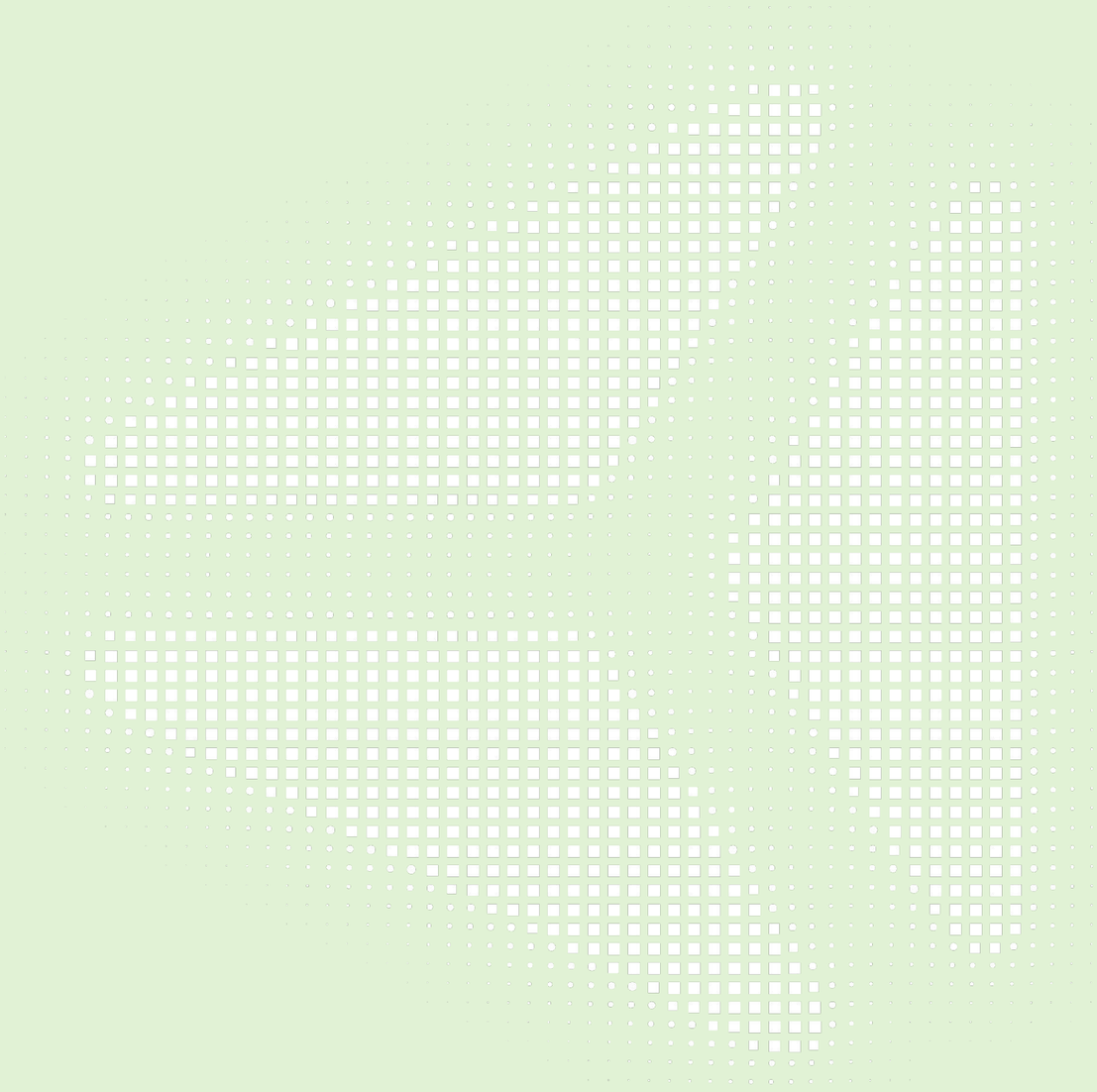
Iniciando um projeto com Vite

Instalando dependências

```
$ npm create vite@latest
```

Commit https://github.com/ederssouza/react-javascript-boilerplate/commit/bcf446fdc6fcdaccdea2f5f7fadcca4d34bcb4c8

Adicionando prop-types



Instalando dependências

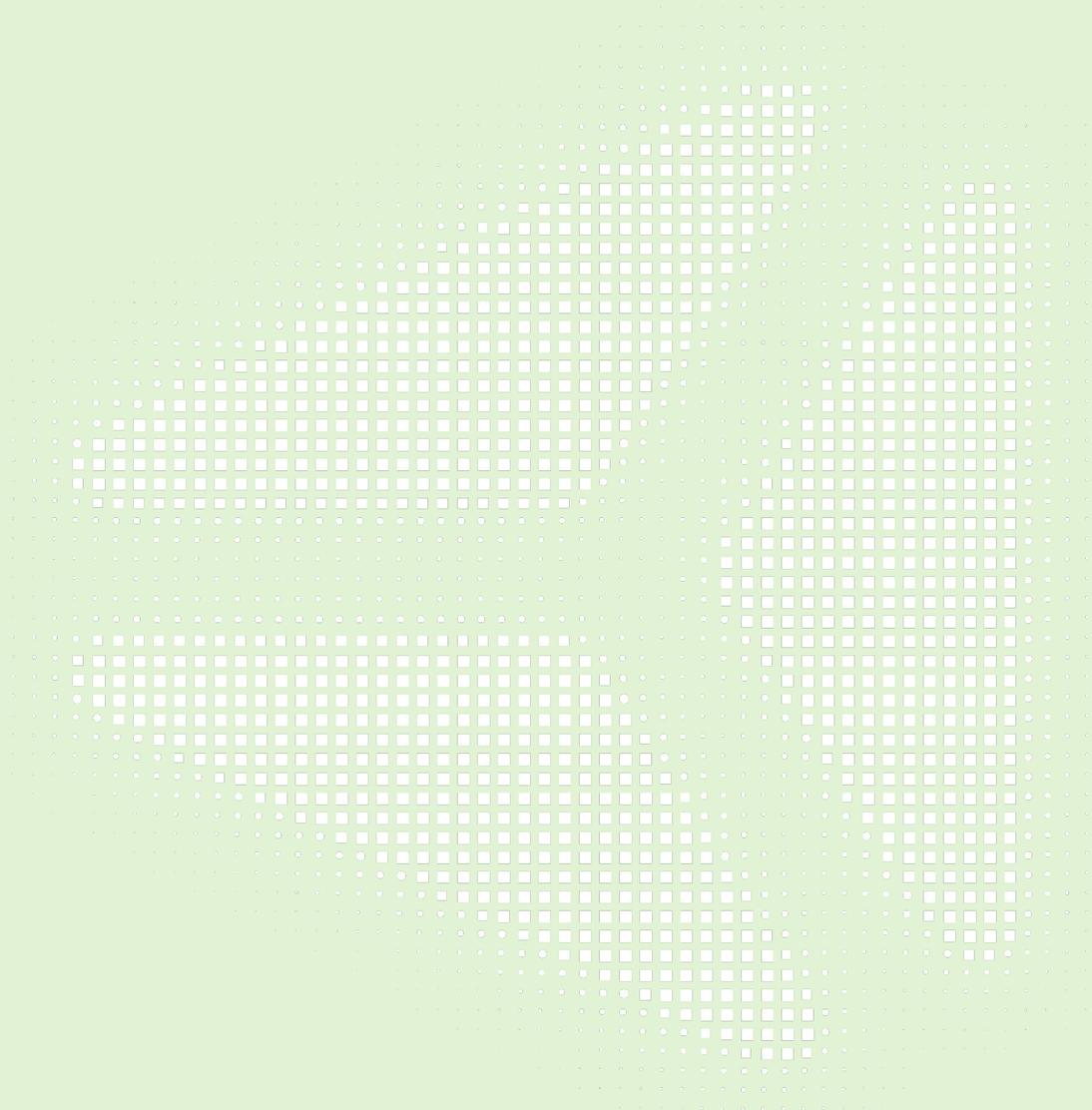
Na medida em que sua aplicação cresce, você pode capturar muitos bugs com checagem de tipos.

Para executar a checagem de tipos nas props de um componente, você pode utilizar a propriedade especial **propTypes** com auxílio da biblioteca **prop-types**.

```
$ npm install prop-types --save
```

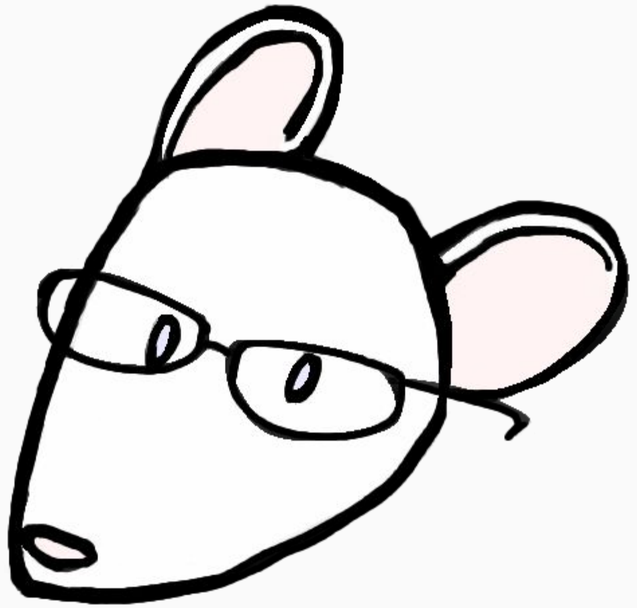
Commit: https://github.com/ederssouza/react-javascript-boilerplate/commit/4c18146afb1d2c68b00555ddc40f3ec91c204e08

Configurando o EditorConfig



Para que serve?

- O EditorConfig ajuda a manter estilos de codificação consistentes no mesmo projeto em vários editores e IDEs;
- Consiste em um formato de arquivo para definir estilos de codificação e uma coleção de plugins de editor de texto que permitem aos editores ler o formato do arquivo e aderir aos estilos definidos.



Configuração

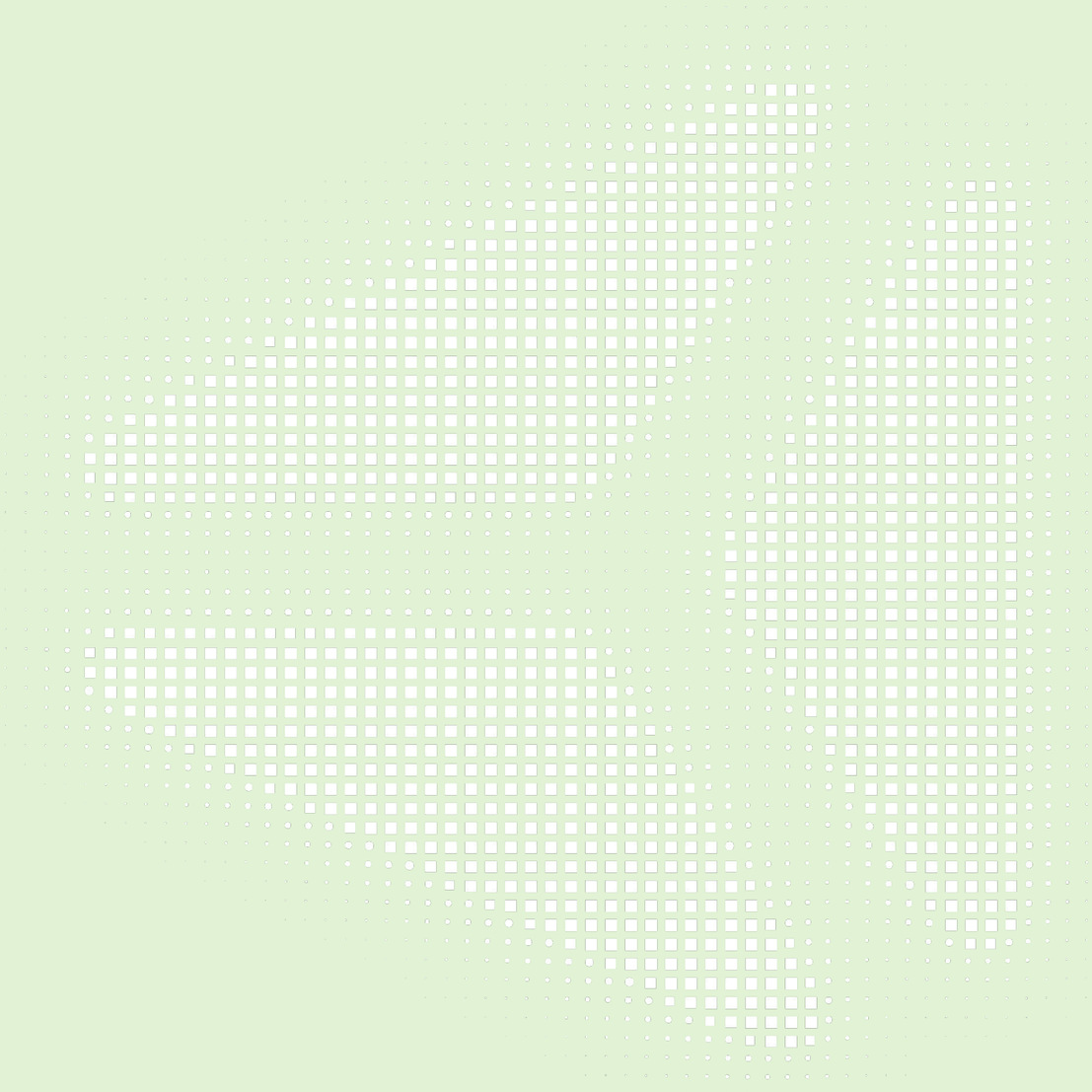
- `[*]`: as regras serão aplicadas para todos os arquivos;
- **`indent_style`**: estilo de indentação (`tab` ou `space`)
- **`indent_size`**: tamanho do recuo
- **`end_of_line`**: controla como as quebras de linha são representadas;
- **`charset`**: codificação de caracteres dos arquivos;
- **`trim_trailing_whitespace`**: indica se o espaço em branco é removido do final das linhas
- **`insert_final_newline`**: indica se o arquivo deve terminar com uma nova linha.

```
root = true

[*]
indent_style = space
indent_size = 2
end_of_line = lf
charset = utf-8
trim_trailing_whitespace = true
insert_final_newline = true

```

Dica: Instale a extensão EditorConfig no seu VS Code.

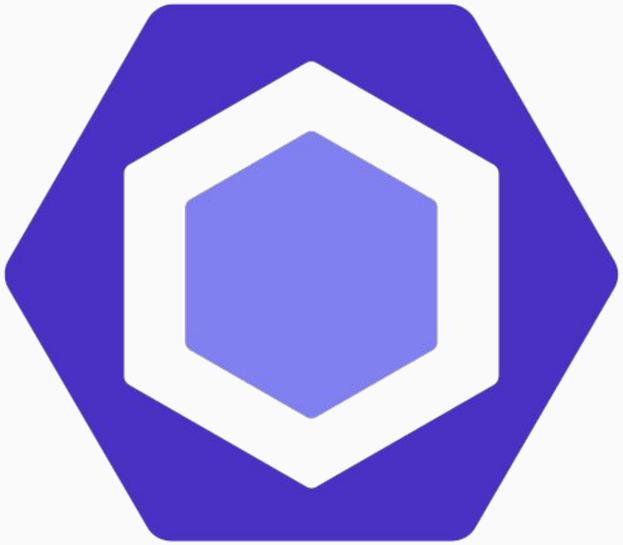
Commit: https://github.com/ederssouza/react-javascript-boilerplate/commit/ea5a58de78b63d3d51d5ac59e55b1a3f2d35026b

Configurando o ESLint



Para que serve?

- É uma ferramenta para identificar e relatar padrões encontrados em ECMAScript;
- Ele é integrado à maioria dos editores de texto e você pode executar o ESLint como parte de seu pipeline de integração contínua.



Configurando o ESlint via CLI

Execute o comando e escolha as opções conforme o bloco abaixo ou de acordo com suas preferências:

```
$ npx eslint --init

✔ How would you like to use ESLint? To check syntax and find problems
✔ What type of modules does your project use? JavaScript modules (import/export)
✔ Which framework does your project use? React
✔ Does your project use TypeScript? No
✔ Where does your code run? Browser
✔ What format do you want your config file to be in? JSON
✔ Would you like to install them now? Yes
✔ Which package manager do you want to use? npm
```

Commit https://github.com/ederssouza/react-javascript-boilerplate/commit/b319abb4a44ea8adfaae4c68709a510bf263a9bd

Instalando dependências e ajustando regras

```
$ npm install eslint-plugin-react-hooks --save-dev
```

```
// .eslintrc.json
{
  // ...
  "settings": {
    "react": {
      "version": "detect"
    }
  },
  // ...
  "plugins": [
    // ...
    "react-hooks"
  ],
  "rules": {
    "react-hooks/rules-of-hooks": "error",
    "react-hooks/exhaustive-deps": "warn",
    "react/react-in-jsx-scope": "off"
  }
}
```

- Este plugin ESLint força as Regras dos hooks:
  - **Use Hooks apenas no nível superior:** Não use Hooks dentro de loops, regras condicionais ou funções aninhadas (funções dentro de funções);
  - **Use Hooks apenas dentro de funções do React:** Não use Hooks dentro de funções JavaScript comuns.

Dica: Instale a extensão ESLint no seu VS Code.

Commit https://github.com/ederssouza/react-javascript-boilerplate/commit/414539906580bd0b0a5c8659e6736ce8a716c48c

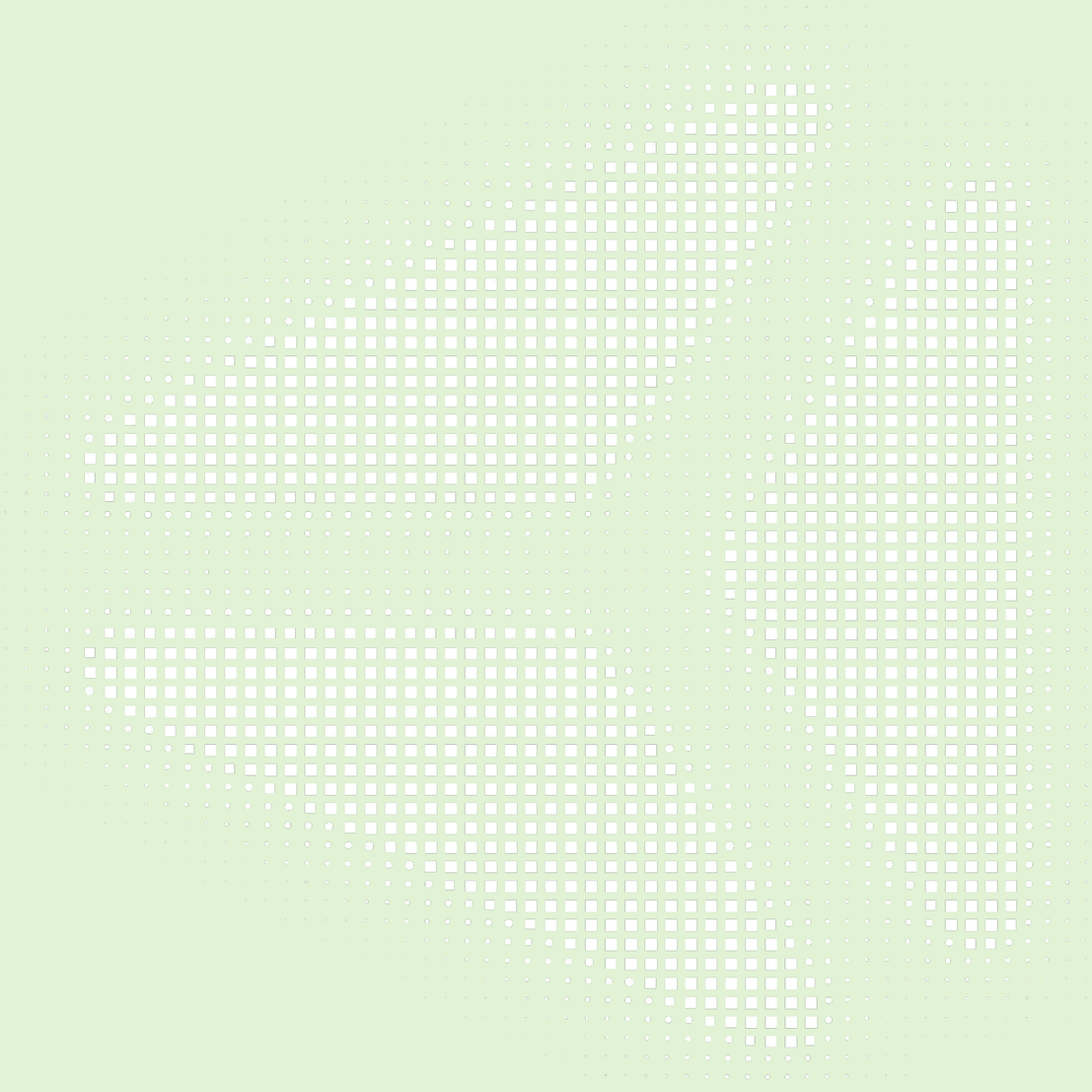
Automatizando o processo com scripts

```
{
  "scripts": {
    "lint": "eslint 'src/**/*.{js,jsx}'",
    "lint:fix": "npm run lint -- --fix"
  }
}
```

- **`npm run lint:`** procura erros em arquivos **`.js`** e **`.jsx`** que estão no diretório **`src`**;
- **`npm run lint:fix:`** procura e corrige erros em arquivos **`.js`** e **`.jsx`** que estão no diretório **`src`**.

Commit https://github.com/ederssouza/react-javascript-boilerplate/commit/806e2954979dddab55a21505ceacf89fe7c95b22

Configurando o Prettier



Para que serve?

- Prettier é uma ferramenta que auxilia na formatação de código e tem suporte a diversos tipos de arquivos como JavaScript, JSX, Angular, Vue, TypeScript, HTML, CSS, SCSS e JSON;
- Mantém a formatação do código coerente e padronizada em todos os tipos de arquivos do projeto (útil em projetos que envolvem muitas pessoas);
- Você salva o arquivo e o código é formatado;
- Não há necessidade de discutir estilo na revisão de código (processo de code review);
- Economiza tempo e energia.

Dica: Instale a extensão Prettier no seu VS Code.

Instalando dependências

```
$ npm install --exact prettier --save-dev
```

```
$ npm install eslint-plugin-prettier eslint-config-prettier --save-dev
```

Configuração

- Criar arquivos:
  - **`.prettierrc`**
  - **`.vscode/settings.json`**
- Atualizar arquivo:
  - **`.eslintrc.json`**

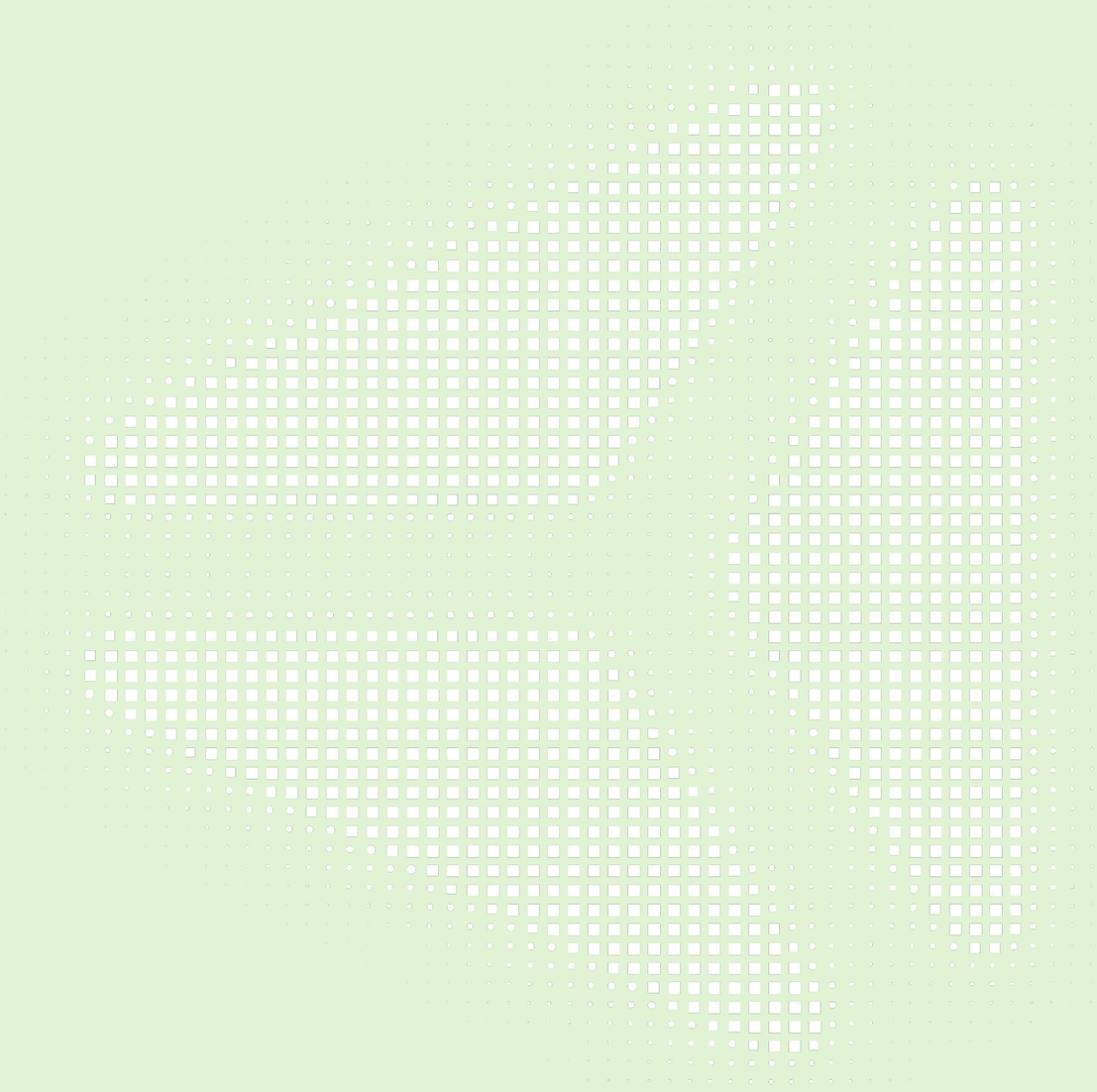
```
// .prettierrc
{
  "trailingComma": "none",
  "semi": false,
  "singleQuote": true
}

```

```
// .eslintrc.json
{
  // ...
  "extends": [
    // ...
    "plugin:prettier/recommended"
  ],
  // ...
}

```

```
// .vscode/settings.json
{
  "editor.formatOnSave": false,
  "editor.codeActionsOnSave": {
    "source.fixAll.eslint": true
  }
}
```

Commit https://github.com/ederssouza/react-javascript-boilerplate/commit/7f00aab3d42161904c86b446bbd7182e3e409a83

Configurando o Jest



Para que serve?

- Jest é um poderoso Framework de Testes em JavaScript com um foco na simplicidade;
- Funciona com projetos usando: Babel, TypeScript, Node, React, Angular, Vue e etc;
- Pode executar testes em paralelo de forma confiável;
- Cria relatórios de cobertura de código.

Instalando dependências

```
$ npm install jest @babel/preset-env @babel/preset-react jest-environment-jsdom --save-dev
```

Configuração

- Atualizar arquivos:
  - **`.eslintrc.json`**
  - **`.gitignore`**
- Criar arquivos:
  - **`babel.config.cjs`**
  - **`jest.config.cjs`**

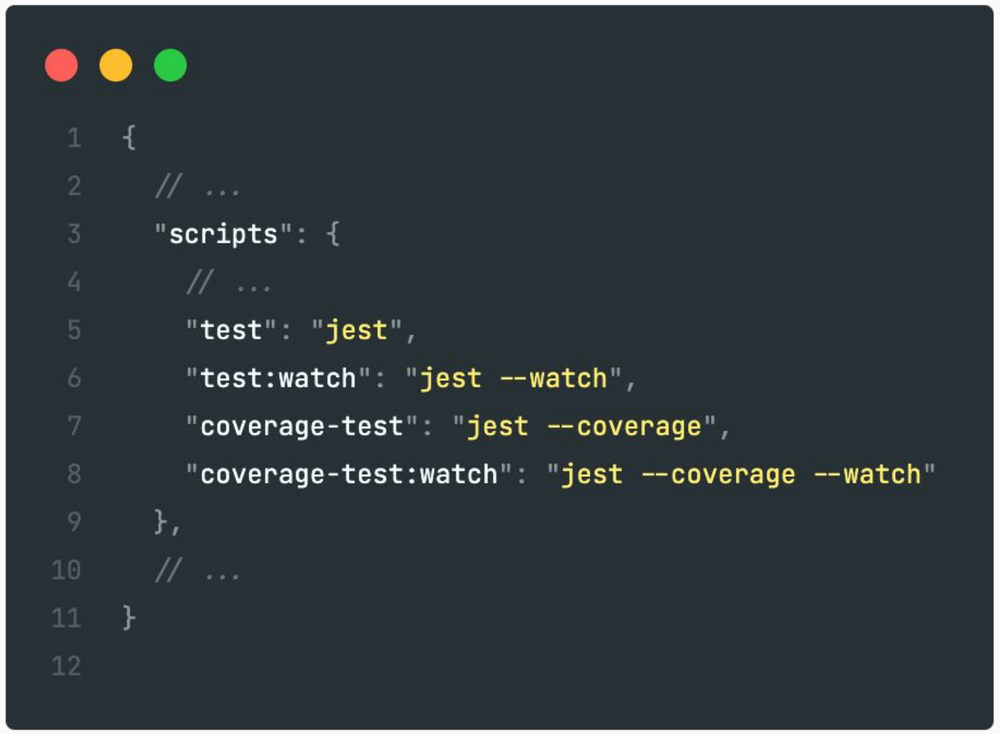
```
// .eslintrc.json
{
  "env": {
    // ...
    "node": true,
    "jest": true
  },
  // ...
}
```

```
# .gitignore

# jest
coverage
```

```
// babel.config.cjs
module.exports = {
  presets: [
    ['@babel/preset-env', { modules: 'auto' }],
    [
      '@babel/preset-react',
      {
        runtime: 'automatic'
      }
    ]
  ]
}
```

```
// jest.config.cjs
module.exports = {
  testEnvironment: 'jsdom',
  testPathIgnorePatterns: ['/dist/', '/node_modules/', '/public/'],
  collectCoverageFrom: ['src/**/*.js(x)?', '!src/**/index.js', '!src/main.jsx'],
  modulePaths: ['<rootDir>/src/'],
  transform: {
    '^.+\\.jsx?$': 'babel-jest'
  }
}
```

Automatizando o processo com scripts

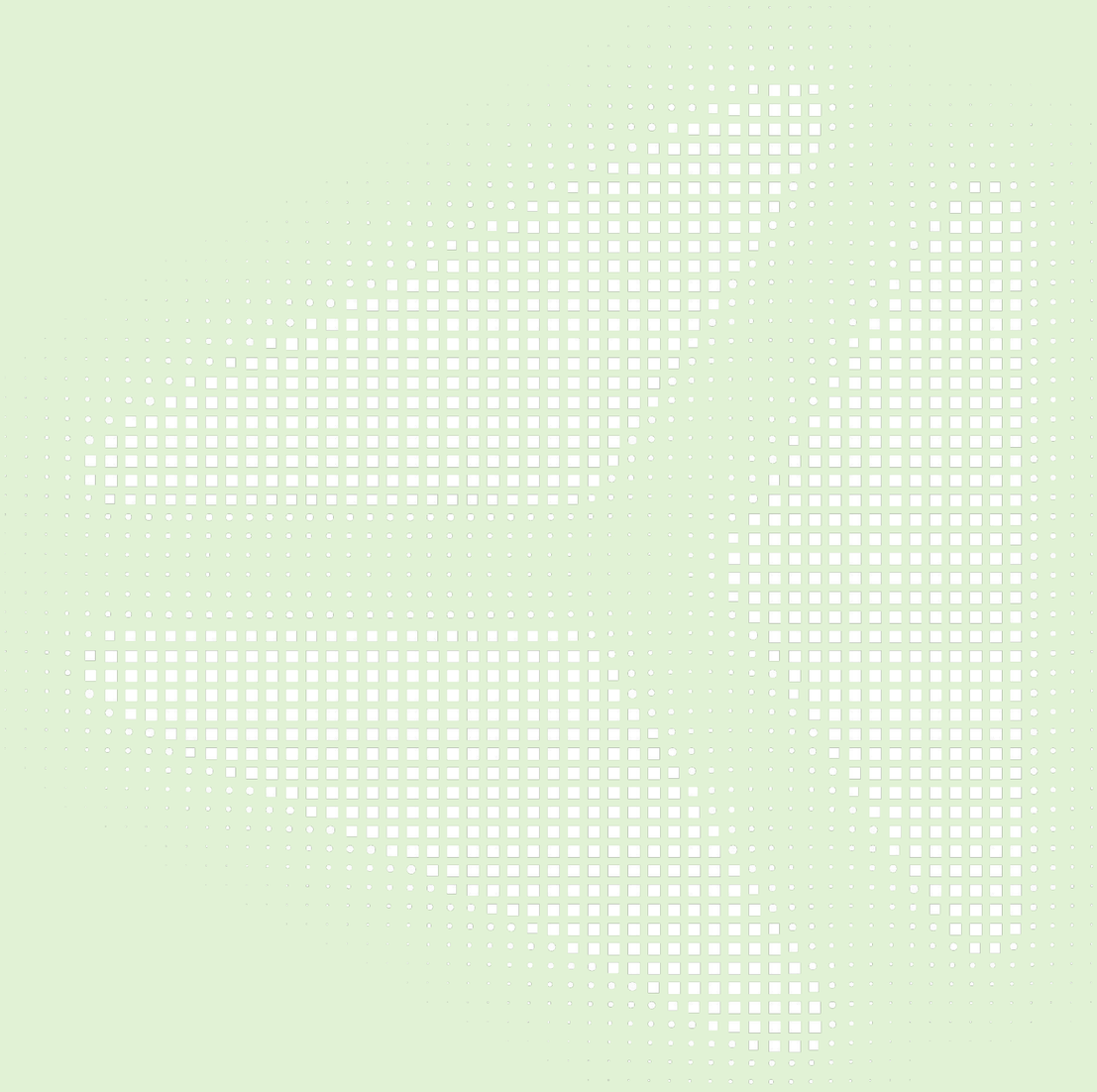


```
{
  // ...
  "scripts": {
    // ...
    "test": "jest",
    "test:watch": "jest --watch",
    "coverage-test": "jest --coverage",
    "coverage-test:watch": "jest --coverage --watch"
  },
  // ...
}
```

- **`npm run test`**: executa os testes unitários e de integração;
- **`npm run test:watch`**: executa os testes unitários e de integração em modo escuta;
- **`npm run coverage-test`**: executa os testes unitários e de integração e cria um relatório de cobertura de código;
- **`npm run coverage-test:watch`**: executa os testes unitários e de integração e cria um relatório de cobertura de código em modo escuta.

Commit https://github.com/ederssouza/react-javascript-boilerplate/commit/bac16002e35c55f37a50009e51996b7e3d9fb7b5

Configurando o React Testing Library



Para que serve?

- React Testing Library se baseia DOM Testing Library adicionando APIs para trabalhar com componentes React;
- Provê ferramentas que permitem testar componentes React de forma fácil e segura.

Instalando dependências

```
$ npm install @testing-library/react @testing-library/jest-dom --save-dev
```

Commit https://github.com/ederssouza/react-javascript-boilerplate/commit/65288716f8382181a90b401fb12e076aa570cd22

Configuração

- Atualizar arquivo:
  - **`jest.config.cjs`**
- Criar arquivos:
  - **`.jest/setup.js`**
  - **`jsconfig.json`**

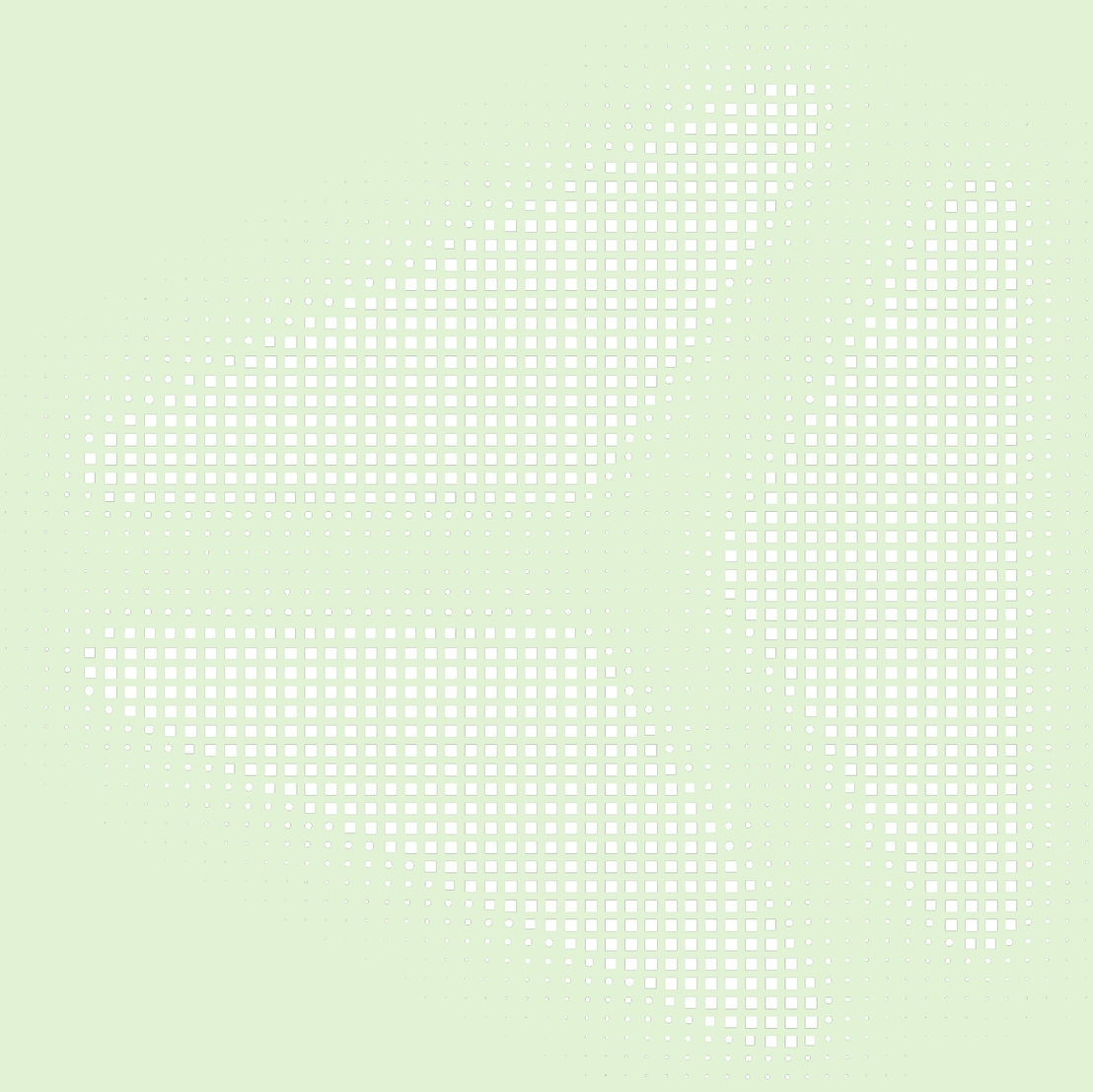
```cjs
// jest.config.cjs
module.exports = {
  // ...
  setupFilesAfterEnv: ['<rootDir>/.jest/setup.js'],
  // ...
}
```

```js
// .jest/setup.js
import "@testing-library/jest-dom";
```

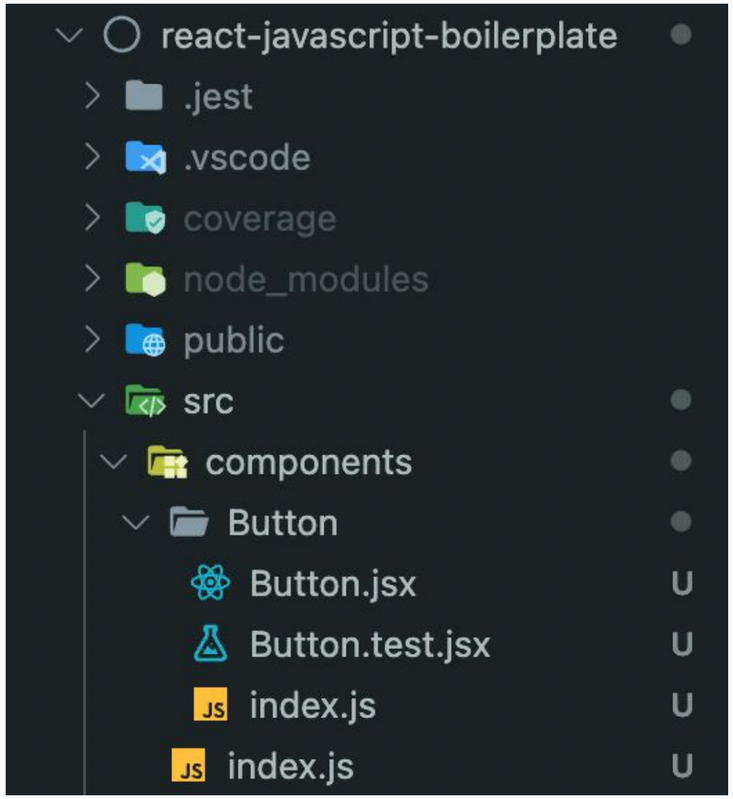
```json
// jsconfig.json
{
  "compilerOptions": {
    "jsx": "react-jsx",
    "types": ["node", "jest", "@testing-library/jest-dom"],
  }
}
```

Commit https://github.com/ederssouza/react-javascript-boilerplate/commit/bcf446fdc6fcdaccdea2f5f7fadcca4d34bcb4c8

Estrutura de diretórios do projeto

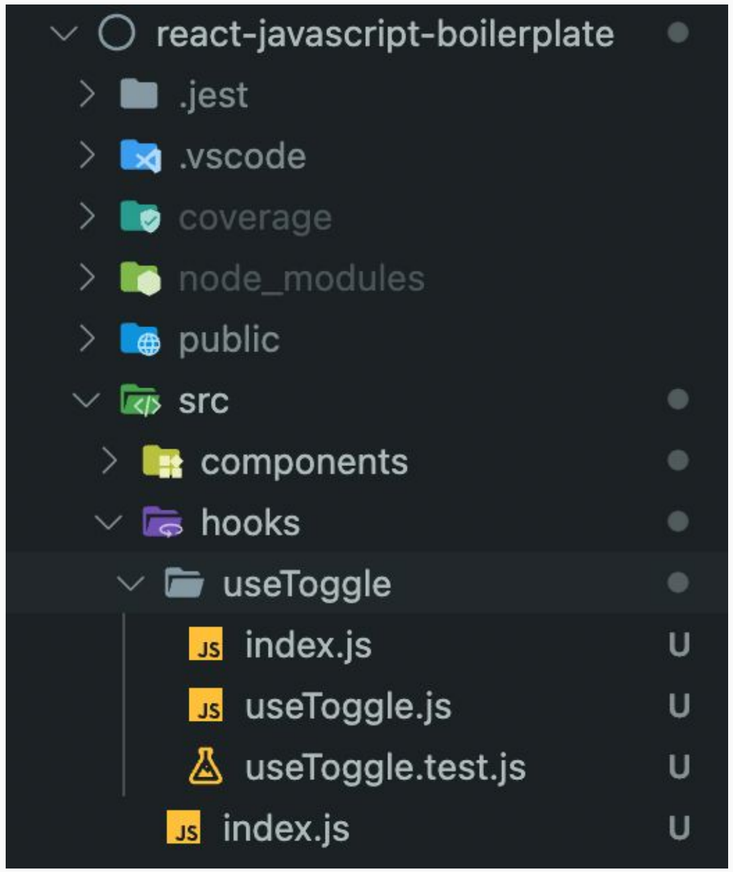


Componentes



- **`src/components/Button/Button.jsx:`** componente;
- **`src/components/Button/Button.test.jsx:`** arquivo com testes unitários / integração;
- **`src/components/Button/index.js:`** arquivo que exporta o componente;
- **`src/components/index.js:`** arquivo que exporta todos os componentes da aplicação.

Hooks



- **`src/hooks/useToggle/index.js`:** arquivo que exporta o hook;
- **`src/hooks/useToggle/useToggle.jsx`:** hook;
- **`src/hooks/useToggle/useToggle.test.jsx`:** arquivo com testes unitários / integração;
- **`src/hooks/index.js`:** arquivo que exporta todos os hooks da aplicação.

Automatizando o processo de criação de componentes e hooks (bônus)

Plop js

O Plop é uma pequena ferramenta que economiza seu tempo e ajuda sua equipe a criar novos arquivos com consistência.

Script para criar estrutura de um componente:

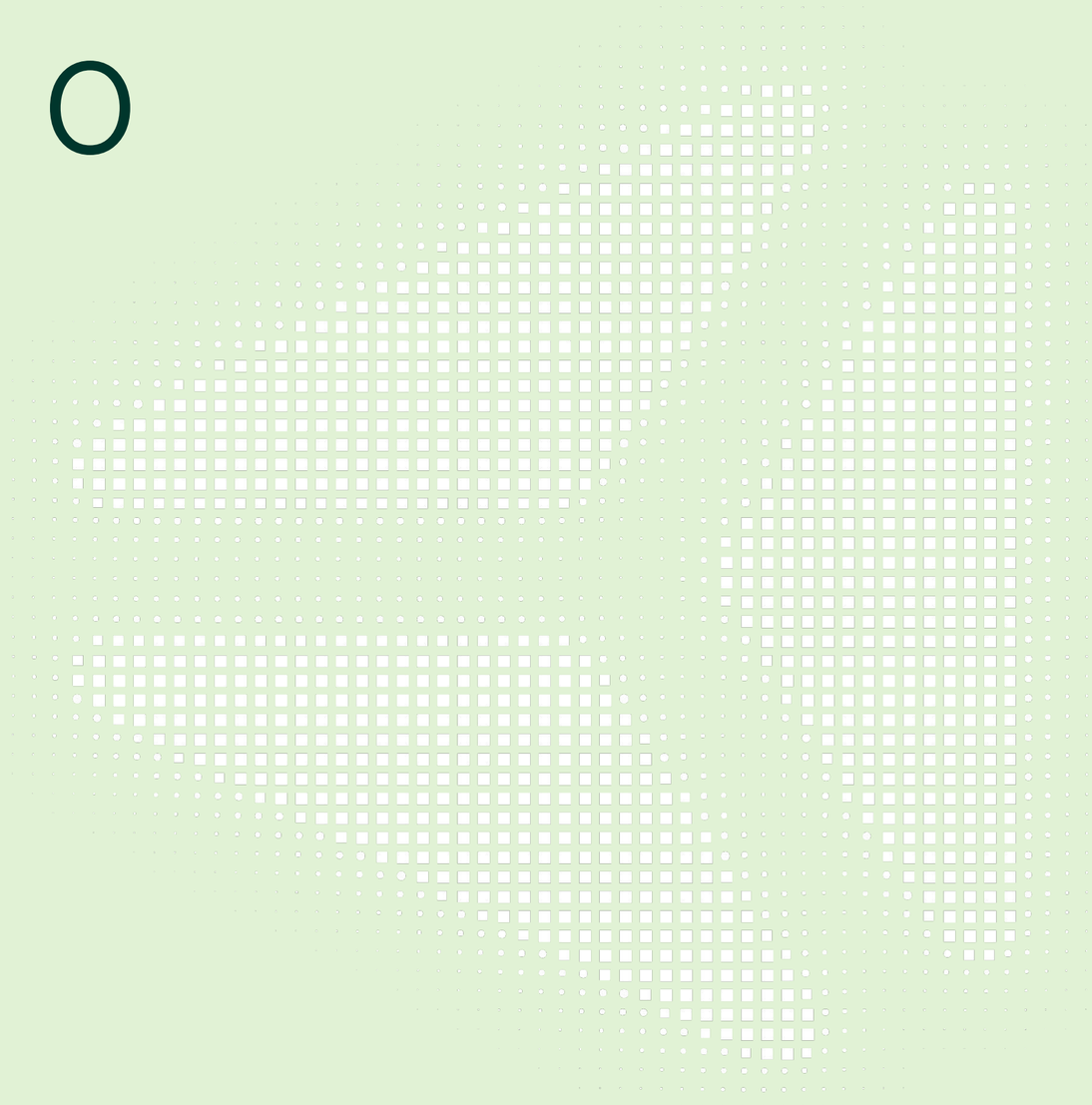
```
$ npm run generate:component
```

Script para criar estrutura de um hook:

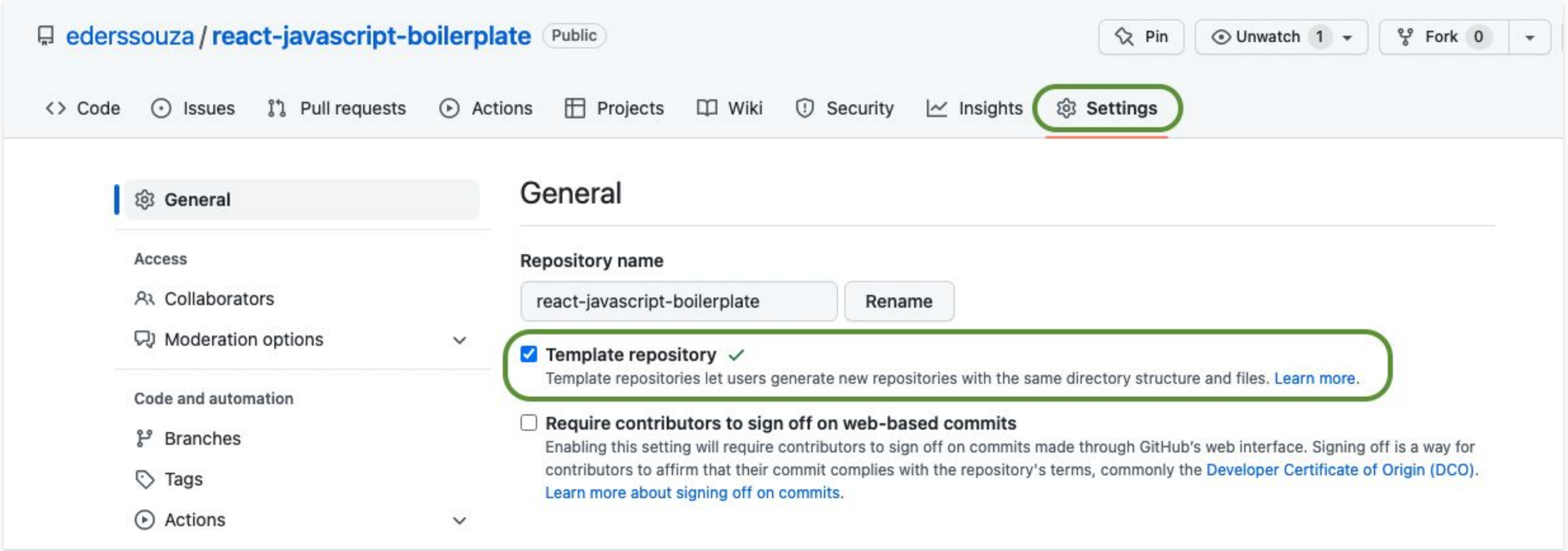
```
$ npm run generate:hook
```

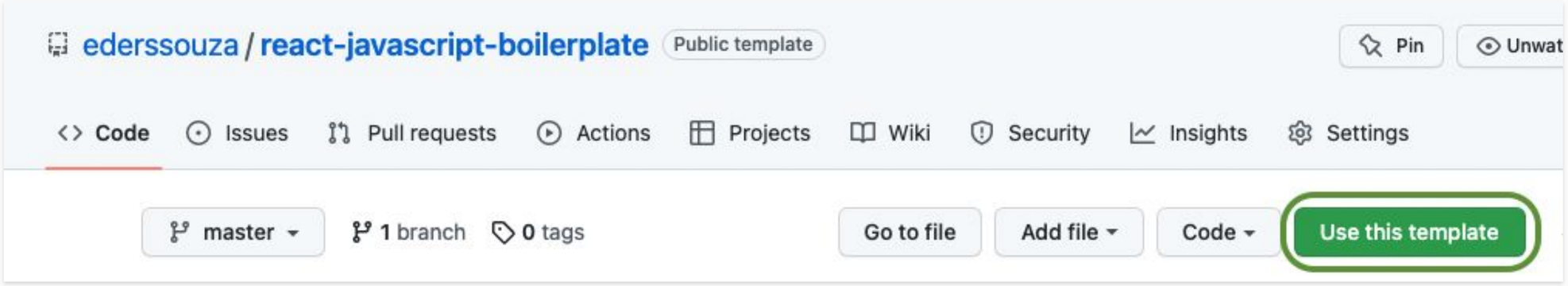
Commit https://github.com/ederssouza/react-javascript-boilerplate/commit/aaf54c33f6ceca02163359fa93fe5bafad2bfa56

Iniciando um projeto utilizando o template criado



Habilitando configuração de template

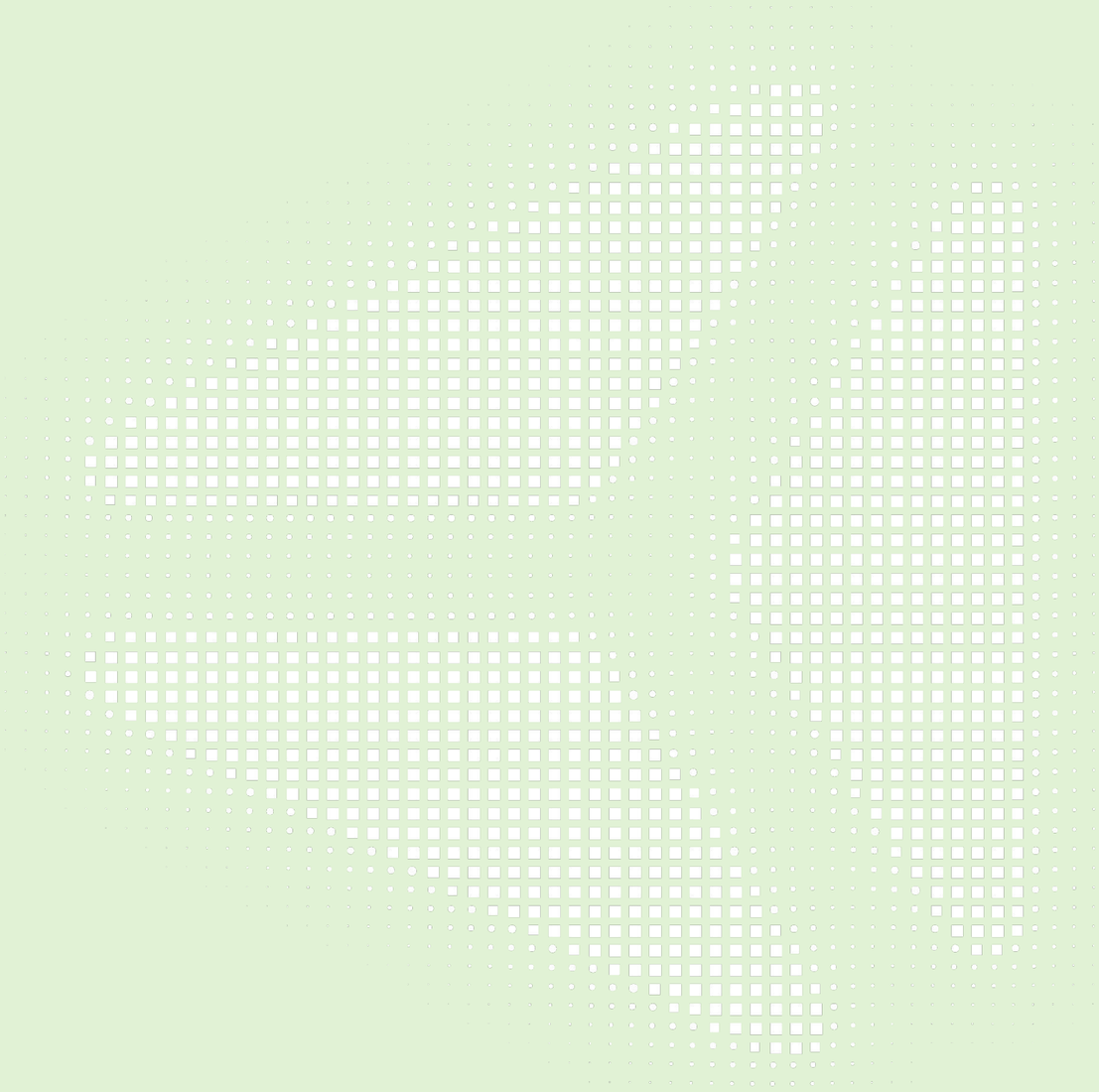




Referências

- Iniciando um projeto com Vite
- Instalando a biblioteca prop-types
- Documentação EditorConfig
- EditorConfig - Extensão para VS Code
- Regras dos Hooks
- Documentação ESLint
- ESLint - Extensão para VS Code
- Documentação Prettier
- Prettier - Extensão para VS Code
- Documentação Jest
- Documentação React Testing Library
- Live Server - Extensão para VS Code
- Documentação Plop.js
- Link do repositório do projeto com JavaScript
- Link do repositório do projeto com TypeScript

Perguntas?



trybe

www.betrybe.com