ESLint

Introdução

Uma outra ferramenta que nos auxilia no momento de padronizarmos o nosso projeto, e talvez seja a mais importante, é o **Eslint**. Com ele conseguimos automatizar os **padrões de códigos** do nosso projeto, e podemos utiliza-lo para projetos em **NodeJS, ReactJS e React Native**.

Por exemplo, no **Javascript** o uso do **ponto e vírgula** ao final de uma linha é **facultativo**, ou seja, diferente de algumas linguagens, a falta dele não interfere para que o código seja compilado. Outra utilização que também é opcional é o uso de **aspas duplas** ou **aspas simples**.

Já quando estamos criando um objeto, o uso da **vírgula** no último item do objeto também é opcional, como podemos ver no exemplo abaixo.

```
const aluno = { nome: "Mariana", idade: 20, }; const aluno = { nome: "Daniel", idade: 21 };
```

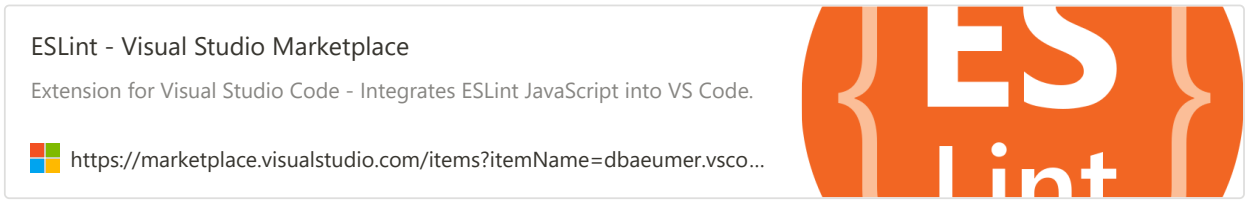
JavaScript

No primeiro objeto utilizamos **vírgula** após o valor dentro do atributo **idade**, já no segundo não utilizamos, o que **não** interfere na execução do código.

O **Eslint** integra, não somente para o **VSCode,** mas também com qualquer outro tipo de editor, o que mais uma vez ajuda na padronização do código, caso um outro desenvolvedor esteja desenvolvendo no mesmo projeto, mas não queira usar o **VSCode**.

Instalação

Antes de iniciar de fato a configuração do **Eslint** em nosso projeto, precisamos instalar a **extensão** do **Eslint** no **VSCode.** É ela quem irá nos auxiliar para que nossas configurações sejam entendidas dentro do nosso código.



Uma outra configuração que é geral e precisamos fazer para o **VSCode** formatar o código sempre que salvarmos algum arquivo é adicionar uma opção chamada `codeActionsOnSave` nas configurações, assim como mostrado abaixo:

```
"editor.codeActionsOnSave": { "source.fixAll.eslint": true }
```

JSON

Node

Pra começar, vamos instalar o **Eslint** como uma dependência de desenvolvimento dentro do nosso projeto **NodeJS**.

```
yarn add eslint@6.8.0 -D
```

Bash

💡 É importante que a versão instalada seja a 6.8.0 ou inferior, pois as versões 7.* estão com algumas incompatibilidades e gerando alguns erros inconvenientes

Após a instalação, precisamos inicializar o **eslint** pra conseguirmos inserir as configurações dentro do projeto.

Faremos isso inserindo o seguinte código no terminal:

```
yarn eslint --init
```

Bash

Ao inserir a linha acima, serão feitas algumas perguntas para configuração do projeto, conforme iremos ver à seguir:

1 - **How would you like do use Eslint?** (Qual a forma que queremos utilizar o **Eslint**)

- **To check syntax only** ⇒ Checar somente a sintaxe
- **To check syntax and find problems** ⇒ Checar a sintaxe e encontrar problemas
- **To check syntax, find problems and enforce code style** ⇒ Checar a sintaxe, encontrar problemas e forçar um padrão de código

Nós iremos escolher a última opção `To check syntax, find problems and enforce code style`.

2 - What type of modules does your project use? (Qual tipo de módulo seu projeto usa?)

- **JavaScript modules (import/export)**
- **CommonsJS (require/exports)**

Como em nosso projeto estamos utilizando o **Typescript,** vamos selecionar a **primeira** opção `Javascript modules (import/export)`

3 - Which framework does your project use? (Qual framework seu projeto está utilizando?)

- **React**
- **Vue.JS**
- **None of these**

Como estamos configurando o nosso **backend** vamos escolher a opção `None of these`

4 - Does your project use TypeScript? (Seu projeto está utilizando Typescript?)

- **No**
- **Yes**

Vamos selecionar a opção `Yes`.

5 - Where does your code run? (Onde seu código está rodando?)

- **Browser**
- **Node**

Vamos selecionar a opção **Node**, para isso, utilizamos a tecla `Espaço` para desmarcar o **Browser** e selecionarmos a opção `Node`

6 - How would you like to define a style for your project? (Qual guia de estilo queremos utilizar?)

- **Use a popular style guide** ⇒ Padrões de projetos já criados anteriormente por outra empresa
- **Answer questions about your style** ⇒ Criar seu próprio padrão de projeto

Vamos selecionar a primeira opção `Use a popular style guide`

7 - Which style guide do you want to follow? (Qual guia de estilo você deseja seguir?)

- **Airbnb:** https://github.com/airbnb/javascript
- **Standard:** https://github.com/standard/standard

- **Google:** https://github.com/google/eslint-config-google

Nós iremos utilizar a primeira opção `Airbnb`. Com ela, nós vamos definir que nosso projeto utilizará **ponto e vírgula** ao final de cada linha, utilizará **aspas simples** e algumas outras configurações. Para saber todas as possíveis configurações, acessar a documentação da guia desejada.
Lembrando que, não há um padrão correto, nós iremos utilizar o **Airbnb**, porém você pode utilizar qualquer guia, desde que seu time todo também esteja utilizando.

8 - What format do you want your config file to be in? (Qual formato de configuração do Eslint que você deseja salvar?)

- **Javascript**
- **YAML**
- **JSON**

Vamos selecionar a opção `JSON`.

```
Checking peerDependencies of eslint-config-airbnb-base@latest @typescript-
eslint/eslint-plugin@latest eslint-config-airbnb-base@latest eslint@^5.16.0 || ^6.8.0
|| ^7.2.0 eslint-plugin-import@^2.21.2 @typescript-eslint/parser@latest
```

Bash

Depois que respondemos as perguntas, o **ESLint** irá informar quais as dependências serão necessárias a instalação, de acordo com a sua respostas dadas.

10 - Would you like to install them now with npm? (Você deseja instalar as dependências agora utilizando npm?)

Por fim, será perguntado se você deseja instalar utilizando o **NPM.** Caso estivéssemos utilizando o **NPM** a resposta seria `Yes`, mas como estamos utilizando o **Yarn** vamos responder `No`.

E como respondemos `No`, vamos ter que adicionar manualmente as dependências. Para isso, basta copiar os pacotes listados acima da pergunta e adicionar com `yarn add` apenas removendo o `eslint@^5.16.0 || ^6.8.0` pois já temos o **ESLint** instalado, o comando final vai ficar:

```
yarn add @typescript-eslint/eslint-plugin@latest eslint-config-airbnb-base@latest
eslint-plugin-import@^2.21.2 @typescript-eslint/parser@latest -D
```

Bash

Com as dependências instaladas vamos criar na raiz do projeto um arquivo `.eslintignore` com o conteúdo abaixo para ignorar o Linting em alguns arquivos:

```
/*.js node_modules dist
```

Plain Text

Agora vamos começar a configuração do arquivo que foi gerado na inicialização do **ESLint**, o `.eslintrc.json` , a primeira coisa a ser feita é adicionar dentro de `"extends"` a linha:

```
"plugin:@typescript-eslint/recommended"
```

JSON

Por fim, para que o **NodeJS** consiga entender arquivos **Typescript** nas importações pois por padrão vai ser apresentado um erro dizendo que as importações de arquivos **Typescript** não foram resolvidas, para resolver isso basta instalar uma dependência que habilite essa funcionalidade:

```
yarn add eslint-import-resolver-typescript -D
```

Bash

Agora para que essa configuração funcione corretamente vamos adicionar logo abaixo das `"rules"` no `.eslintrc.json` o seguinte:

```
"settings": { "import/resolver": { "typescript": {} } }
```

JSON

E dentro das `"rules"` adicione o trecho abaixo para que as importações de arquivos `.tsx` não precisem da extensão do arquivo:

```
"import/extensions": [ "error", "ignorePackages", { "ts": "never" } ]
```

JSON

Para finalizar e aplicar todas as mudanças vamos fechar o VS Code e reabrir na **pasta raiz** do projeto, pois senão o **ESLint** não vai reconhecer as dependências instaladas e aplicar as regras de Linting.

Feito isso, para verificar se está realmente funcionando basta reabrir qualquer arquivo do projeto e tentar errar algo no código para que ele mostre o erro e formate automaticamente quando o arquivo for salvo.

O arquivo `.eslintrc.json` finalizado com todas as mudanças tem que ficar assim:

```
{ "env": { "es2020": true, "node": true }, "extends": [ "airbnb-base",
"plugin:@typescript-eslint/recommended" ], "parser": "@typescript-eslint/parser",
"parserOptions": { "ecmaVersion": 2018, "sourceType": "module" }, "plugins": [
"@typescript-eslint" ], "rules": { "import/extensions": [ "error", "ignorePackages", {
"ts": "never" } ] }, "settings": { "import/resolver": { "typescript": {} } } }
```

JSON

Pronto, nosso **ESLint** já está funcionando e estamos quase finalizando as configurações para padronizar nosso código.

Antes de darmos prosseguimento, para entendermos os próximos passos do **Prettier**, dentro da pasta `src` crie um pasta `routes` e crie um arquivo `index.ts` com o seguinte conteúdo dentro dele

```
// src/routes/index.ts import { Router } from 'express'; const routes = Router();
export default routes;
```

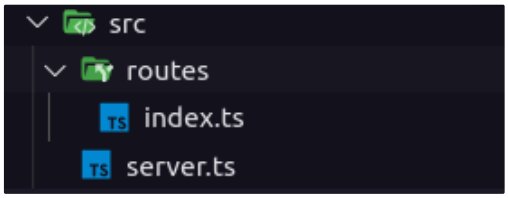
JavaScript

Após isso, no seu arquivo `server.ts` importar o arquivo de rotas:

```
// src/server.ts import routes from './routes
```

JavaScript

Você terá a seguinte estrutura de pastas.



Por fim, o que falta fazer é instalar e configurar o Prettier

ReactJS

tro to the H

Como estamos criando o projeto utilizando o `create-react-app` temos que remover uma configuração do `package.json` para evitar conflitos com as configurações que vamos fazer abaixo, para isso remova o trecho abaixo do `package.json`:

```
"eslintConfig": { "extends": "react-app" },
```

JSON

Pra começar, vamos instalar o **Eslint** como uma dependência de desenvolvimento dentro do nosso projeto **ReactJS**.

```
yarn add eslint@6.8.0 -D
```

Bash

💡 É importante que a versão instalada seja a 6.8.0 ou inferior, pois as versões 7.* estão com algumas incompatibilidades e gerando alguns erros inconvenientes

Após a instalação, precisamos inicializar o **ESLint** pra conseguirmos inserir as configurações dentro do projeto.

Faremos isso inserindo o seguinte código no terminal:

```
yarn eslint --init
```

Bash

Ao inserir a linha acima, serão feitas algumas perguntas para configuração do projeto, conforme iremos ver à seguir:

1 - **How would you like do use Eslint?** (Qual a forma que queremos utilizar o **Eslint**)

- **To check syntax only** ⇒ Checar somente a sintaxe
- **To check syntax and find problems** ⇒ Checar a sintaxe e encontrar problemas
- **To check syntax, find problems and enforce code style** ⇒ Checar a sintaxe, encontrar problemas e forçar um padrão de código

Nós iremos escolher a última opção `To check syntax, find problems and enforce code style`.

2 - **What type of modules does your project use?** (Qual tipo de módulo seu projeto usa?)

- **JavaScript modules (import/export)**
- **CommonsJS (require/exports)**

Como em nosso projeto estamos utilizando o **Typescript,** vamos selecionar a **primeira** opção `Javascript modules (import/export)`

3 - **Which framework does your project use?** (Qual framework seu projeto está utilizando?)

- **React**
- **Vue.JS**
- **None of these**

Como estamos configurando o nosso **frontend** vamos escolher a opção `React`

4 - **Does your project use TypeScript?** (Seu projeto está utilizando Typescript?)

- **No**

- **Yes**

Vamos selecionar a opção `Yes`.

5 - Where does your code run? (Onde seu código está rodando?)

- **Browser**
- **Node**

Vamos deixar a opção que vem por padrão selecionada, que é **Browser** e apertamos o enter.

6 - How would you like to define a style for your project? (Qual guia de estilo queremos utilizar?)

- **Use a popular style guide** ⇒ Padrões de projetos já criados anteriormente por outra empresa
- **Answer questions about your style** ⇒ Criar seu próprio padrão de projeto

Vamos selecionar a primeira opção `Use a popular style guide`

7 - Which style guide do you want to follow? (Qual guia de estilo você deseja seguir?)

- **Airbnb:** https://github.com/airbnb/javascript
- **Standard:** https://github.com/standard/standard
- **Google:** https://github.com/google/eslint-config-google

Nós iremos utilizar a primeira opção `Airbnb`. Com ela, nós vamos definir que nosso projeto utilizará **ponto e vírgula** ao final de cada linha, utilizará **aspas simples** e algumas outras configurações. Para saber todas as possíveis configurações, acessar a documentação da guia desejada.
Lembrando que, não há um padrão correto, nós iremos utilizar o **Airbnb**, porém você pode utilizar qualquer guia, desde que seu time todo também esteja utilizando.

8 - What format do you want your config file to be in? (Qual formato de configuração do Eslint que você deseja salvar?)

- **Javascript**
- **YAML**
- **JSON**

Vamos selecionar a opção `JSON`

Depois que respondemos as perguntas, o **ESLint** irá informar quais as dependências necessárias de acordo com a sua configuração e pedir para instalá-las automaticamente. Caso estivéssemos utilizando o **NPM** a resposta seria `Yes`, mas como estamos utilizando o **Yarn** vamos responder `No`.

```
Checking peerDependencies of eslint-config-airbnb@latest The config that you've selected requires the following dependencies: eslint-plugin-react@^7.19.0 @typescript-eslint/eslint-plugin@latest eslint-config-airbnb@latest eslint@^5.16.0 || ^6.8.0 eslint-plugin-import@^2.20.1 eslint-plugin-jsx-a11y@^6.2.3 eslint-plugin-react-hooks@^2.5.0 || ^1.7.0 @typescript-eslint/parser@latest ? Would you like to install them now with npm? No
```

Bash

E como respondemos `No` vamos ter que adicionar manualmente as dependências, para isso basta copiar os pacotes listados acima da pergunta e adicionar com `yarn add` apenas removendo o `eslint@^5.16.0 || ^6.8.0` pois já temos o **ESLint** instalado, e remover a versão `1.7.0` do trecho `eslint-plugin-react-hooks@^2.5.0 || ^1.7.0` o comando final vai ficar:

```
yarn add eslint-plugin-react@^7.19.0 @typescript-eslint/eslint-plugin@latest eslint-config-airbnb@latest eslint-plugin-import@^2.20.1 eslint-plugin-jsx-a11y@^6.2.3 eslint-plugin-react-hooks@^2.5.0 @typescript-eslint/parser@latest -D
```

Bash

Com as dependências instaladas vamos criar na raiz do projeto um arquivo `.eslintignore` com o conteúdo abaixo para ignorar o Linting em alguns arquivos:

```
**/*.js node_modules build /src/react-app-env.d.ts
```

Plain Text

Agora vamos começar a configuração do arquivo que foi gerado na inicialização do **ESLint**, o `.eslintrc.json` , a primeira coisa a ser feita é adicionar dentro de `"extends"` a linha:

```
"plugin:@typescript-eslint/recommended"
```

JSON

Nos `"plugins"` vamos adicionar apenas o dos Hooks, como abaixo:

```
"react-hooks"
```

JSON

E para finalizarmos a configuração nesse arquivo vamos adicionar algumas linhas nas `"rules"` , como abaixo:

```
"react-hooks/rules-of-hooks": "error", "react-hooks/exhaustive-deps": "warn", "react/jsx-filename-extension": [1, { "extensions": [".tsx"] }], "import/prefer-default-export": "off",
```

JSON

Por fim, para que o **ReactJS** consiga entender arquivos **Typescript** nas importações pois por padrão vai ser apresentado um erro dizendo que as importações de arquivos **Typescript** não foram resolvidas, para resolver isso basta instalar uma dependência que habilite essa funcionalidade:

```
yarn add eslint-import-resolver-typescript -D
```

Bash

Agora para que essa configuração funcione corretamente vamos adicionar logo abaixo das `"rules"` no `.eslintrc.json` o seguinte:

```
"settings": { "import/resolver": { "typescript": {} } }
```

JSON

E dentro das `"rules"` adicione o trecho abaixo para que as importações de arquivos `.tsx` não precisem da extensão do arquivo:

```
"import/extensions": [ "error", "ignorePackages", { "ts": "never", "tsx": "never" } ]
```

JSON

Para finalizar e aplicar todas as mudanças vamos fechar o VS Code e reabrir na **pasta raiz** do projeto, pois senão o **ESLint** não vai reconhecer as dependências instaladas e aplicar as regras de Linting.

Feito isso, para verificar se está realmente funcionando basta reabrir qualquer arquivo do projeto e tentar errar algo no código para que ele mostre o erro e formate automaticamente quando o arquivo for salvo.

O arquivo `.eslintrc.json` finalizado com todas as mudanças tem que ficar assim:

```
{ "env": { "browser": true, "es6": true }, "extends": [ "plugin:react/recommended", "airbnb", "plugin:@typescript-eslint/recommended" ], "globals": { "Atomics": "readonly", "SharedArrayBuffer": "readonly" }, "parser": "@typescript-eslint/parser", "parserOptions": { "ecmaFeatures": { "jsx": true }, "ecmaVersion": 2018, "sourceType": "module" }, "plugins": [ "react", "react-hooks", "@typescript-eslint" ], "rules": { "react-hooks/rules-of-hooks": "error", "react-hooks/exhaustive-deps": "warn", "react/jsx-filename-extension": [1, { "extensions": [".tsx"] }], "import/prefer-default-export": "off", "import/extensions": [ "error", "ignorePackages", { "ts": "never", "tsx": "never" } ] }, "settings": { "import/resolver": { "typescript": {} } } }
```

JSON

Pronto, nosso **ESLint** já está funcionando e estamos quase finalizando as configurações para padronizar nosso código. Por fim, o que falta fazer é instalar e configurar o **Prettier**

React Native

Pra começar, vamos instalar o **Eslint** como uma dependência de desenvolvimento dentro do nosso projeto **ReactJS**.

```bash
yarn add eslint@6.8.0 -D
```

Bash

> 💡 É importante que a versão instalada seja a 6.8.0 ou inferior, pois as versões 7.* estão com algumas incompatibilidades e gerando alguns erros inconvenientes

Após a instalação, precisamos inicializar o **ESLint** pra conseguirmos inserir as configurações dentro do projeto.

Faremos isso inserindo o seguinte código no terminal:

```bash
yarn eslint --init
```

Bash

Ao inserir a linha acima, serão feitas algumas perguntas para configuração do projeto, conforme iremos ver à seguir:

1 - **How would you like do use Eslint?** (Qual a forma que queremos utilizar o **Eslint**)

- **To check syntax only** ⇒ Checar somente a sintaxe
- **To check syntax and find problems** ⇒ Checar a sintaxe e encontrar problemas
- **To check syntax, find problems and enforce code style** ⇒ Checar a sintaxe, encontrar problemas e forçar um padrão de código

Nós iremos escolher a última opção `To check syntax, find problems and enforce code style`.

2 - **What type of modules does your project use?** (Qual tipo de módulo seu projeto usa?)

- **JavaScript modules (import/export)**
- **CommonsJS (require/exports)**

Como em nosso projeto estamos utilizando o **Typescript,** vamos selecionar a **primeira** opção `Javascript modules (import/export)`

3 - **Which framework does your project use?** (Qual framework seu projeto está utilizando?)

- **React**
- **Vue.JS**
- **None of these**

Como estamos configurando o nosso **mobile** vamos escolher a opção `React`

4 - Does your project use TypeScript? (Seu projeto está utilizando Typescript?)

- **No**
- **Yes**

Vamos selecionar a opção `Yes`.

5 - Where does your code run? (Onde seu código está rodando?)

- **Browser**
- **Node**

Vamos desmarcar as duas opções, utilizamos a tecla `Espaço` para desmarcar as opções.

6 - How would you like to define a style for your project? (Qual guia de estilo queremos utilizar?)

- **Use a popular style guide ⇒** Padrões de projetos já criados anteriormente por outra empresa
- **Answer questions about your style ⇒** Criar seu próprio padrão de projeto

Vamos selecionar a primeira opção `Use a popular style guide`

7 - Which style guide do you want to follow? (Qual guia de estilo você deseja seguir?)

- **Airbnb:** https://github.com/airbnb/javascript
- **Standard:** https://github.com/standard/standard
- **Google:** https://github.com/google/eslint-config-google

Nós iremos utilizar a primeira opção `Airbnb`. Com ela, nós vamos definir que nosso projeto utilizará **ponto e vírgula** ao final de cada linha, utilizará **aspas simples** e algumas outras configurações. Para saber todas as possíveis configurações, acessar a documentação da guia desejada.
Lembrando que, não há um padrão correto, nós iremos utilizar o **Airbnb**, porém você pode utilizar qualquer guia, desde que seu time todo também esteja utilizando.

8 - What format do you want your config file to be in? (Qual formato de configuração do Eslint que você deseja salvar?)

- **Javascript**
- **YAML**
- **JSON**

Vamos selecionar a opção `JSON`

Depois que respondemos as perguntas, o **ESLint** irá informar quais as dependências necessárias de acordo com a sua configuração e pedir para instalá-las automaticamente. Caso estivéssemos utilizando o **NPM** a resposta seria `Yes`, mas como estamos utilizando o **Yarn** vamos responder `No`.

```
Checking peerDependencies of eslint-config-airbnb@latest The config that you've selected requires the following dependencies: eslint-plugin-react@^7.19.0 @typescript-eslint/eslint-plugin@latest eslint-config-airbnb@latest eslint@^5.16.0 || ^6.8.0 eslint-plugin-import@^2.20.1 eslint-plugin-jsx-a11y@^6.2.3 eslint-plugin-react-hooks@^2.5.0 || ^1.7.0 @typescript-eslint/parser@latest ? Would you like to install them now with npm? No
```

Bash

E como respondemos `No` vamos ter que adicionar manualmente as dependências, para isso basta copiar os pacotes listados acima da pergunta e adicionar com `yarn add` apenas removendo o `eslint@^5.16.0 || ^6.8.0` pois já temos o **ESLint** instalado, e remover a versão `1.7.0` do trecho `eslint-plugin-react-hooks@^2.5.0 || ^1.7.0` o comando final vai ficar:

```
yarn add eslint-plugin-react@^7.19.0 @typescript-eslint/eslint-plugin@latest eslint-config-airbnb@latest eslint-plugin-import@^2.20.1 eslint-plugin-jsx-a11y@^6.2.3 eslint-plugin-react-hooks@^2.5.0 @typescript-eslint/parser@latest -D
```

Bash

Com as dependências instaladas vamos criar na raiz do projeto um arquivo `.eslintignore` com o conteúdo abaixo para ignorar o Linting em alguns arquivos:

```
**/*.js node_modules build
```

Plain Text

Agora vamos começar a configuração do arquivo que foi gerado na inicialização do **ESLint**, o `.eslintrc.json`, a primeira coisa a ser feita é adicionar dentro de `"extends"` a linha:

```
"plugin:@typescript-eslint/recommended"
```

JSON

Por estar no mobile, utilizando **React Native** vamos adicionar uma configuração no `"globals"`, como mostrado abaixo:

```
"__DEV__": "readonly"
```

JSON

Nos `"plugins"` vamos adicionar apenas o dos Hooks, como abaixo:

```
"react-hooks"
```

JSON

E para finalizarmos a configuração nesse arquivo vamos adicionar algumas linhas nas `"rules"` , como abaixo:

```
"react-hooks/rules-of-hooks": "error", "react-hooks/exhaustive-deps": "warn",
"react/jsx-filename-extension": [1, { "extensions": [".tsx"] }], "import/prefer-
default-export": "off",
```

JSON

Por fim, para que o **ReactJS** consiga entender arquivos **Typescript** nas importações pois por padrão vai ser apresentado um erro dizendo que as importações de arquivos **Typescript** não foram resolvidas, para resolver isso basta instalar uma dependência que habilite essa funcionalidade:

```
yarn add eslint-import-resolver-typescript -D
```

Bash

Agora para que essa configuração funcione corretamente vamos adicionar logo abaixo das `"rules"` no `.eslintrc.json` o seguinte:

```
"settings": { "import/resolver": { "typescript": {} } }
```

JSON

E dentro das `"rules"` adicione o trecho abaixo para que as importações de arquivos `.tsx` não precisem da extensão do arquivo:

```
"import/extensions": [ "error", "ignorePackages", { "ts": "never", "tsx": "never" } ]
```

JSON

Para finalizar e aplicar todas as mudanças vamos fechar o VS Code e reabrir na **pasta raiz** do projeto, pois senão o **ESLint** não vai reconhecer as dependências instaladas e aplicar as regras de Linting.

Feito isso, para verificar se está realmente funcionando basta reabrir qualquer arquivo do projeto e tentar errar algo no código para que ele mostre o erro e formate automaticamente quando o arquivo for salvo.

O arquivo `.eslintrc.json` finalizado com todas as mudanças tem que ficar assim:

```
{ "env": { "es6": true }, "extends": [ "plugin:react/recommended", "airbnb",
"plugin:@typescript-eslint/recommended" ], "globals": { "Atomics": "readonly",
"SharedArrayBuffer": "readonly", "__DEV__": "readonly" }, "parser": "@typescript-
eslint/parser", "parserOptions": { "ecmaFeatures": { "jsx": true }, "ecmaVersion":
2018, "sourceType": "module" }, "plugins": [ "react", "react-hooks", "@typescript-
eslint" ], "rules": { "react-hooks/rules-of-hooks": "error", "react-hooks/exhaustive-
deps": "warn", "react/jsx-filename-extension": [1, { "extensions": [".tsx"] }],
"import/prefer-default-export": "off", "import/extensions": [ "error",
"ignorePackages", { "ts": "never", "tsx": "never" } ] }, "settings": {
"import/resolver": { "typescript": {} } } }
```

JSON

Pronto, nosso **ESLint** já está funcionando e estamos quase finalizando as configurações para padronizar nosso código. Por fim, o que falta fazer é instalar e configurar o **Prettier**

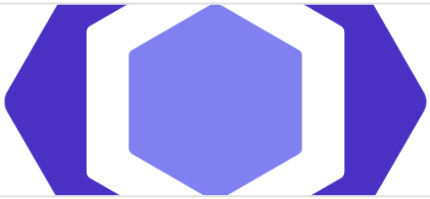
Documentação

Para mais informações sobre as configurações do ESLint, você pode consultar a documentação oficial

List of available rules

Rules in ESLint are grouped by category to help you understand their purpose. No rules are enabled by default. The "extends":

https://eslint.org/docs/rules/



Para projetos utilizando **React** ou **React Native**, podemos consultar a documentação do **eslint-plugin-react**