



# Configurações do ambiente

## Introdução

Fala Dev! Seja muito bem vindo à **Next Level Week**, Mission: **ReactJS**!

Para começar a preparar você para essa semana incrível de muito conteúdo e aprendizado, vamos começar configurando nosso ambiente de desenvolvimento, com algumas ferramentas fundamentais para chegarmos no fim desse evento com nosso projeto finalizado.

## Preparando o ambiente

Sem mais delongas, vamos ao conteúdo principal desse guia: configuração do seu ambiente para o NLW. Teremos três etapas principais na seção "**Instalação**":

- Node + NPM;
- Yarn;
- Visual Studio Code e configurações.


Se você já participou de alguma das nossas outras NLWs, já deve ter tudo isso instalado, mas talvez desatualizado. Por isso, preparamos uma seção "**Atualização**" para você atualizar suas dependências caso precise, mas sugerimos dar uma olhada pelo menos no guia Visual Studio Code na seção "**Instalação**" pois o Diego deixou algumas configs especialmente para você 💜


Preparados? Então vamos lá!

## Guias



[Instalação das ferramentas](#)

 [Atualização \(versões diferentes\)](#)

 [Tive problemas, e agora?](#)

## Opcional - Configurações adicionais do VS Code

Se você já configurou todo o ambiente seguido os passos anteriores e quer deixar o Visual Studio Code com as mesmas configurações usadas pelo Diego nessa trilha, aqui vão algumas dicas:

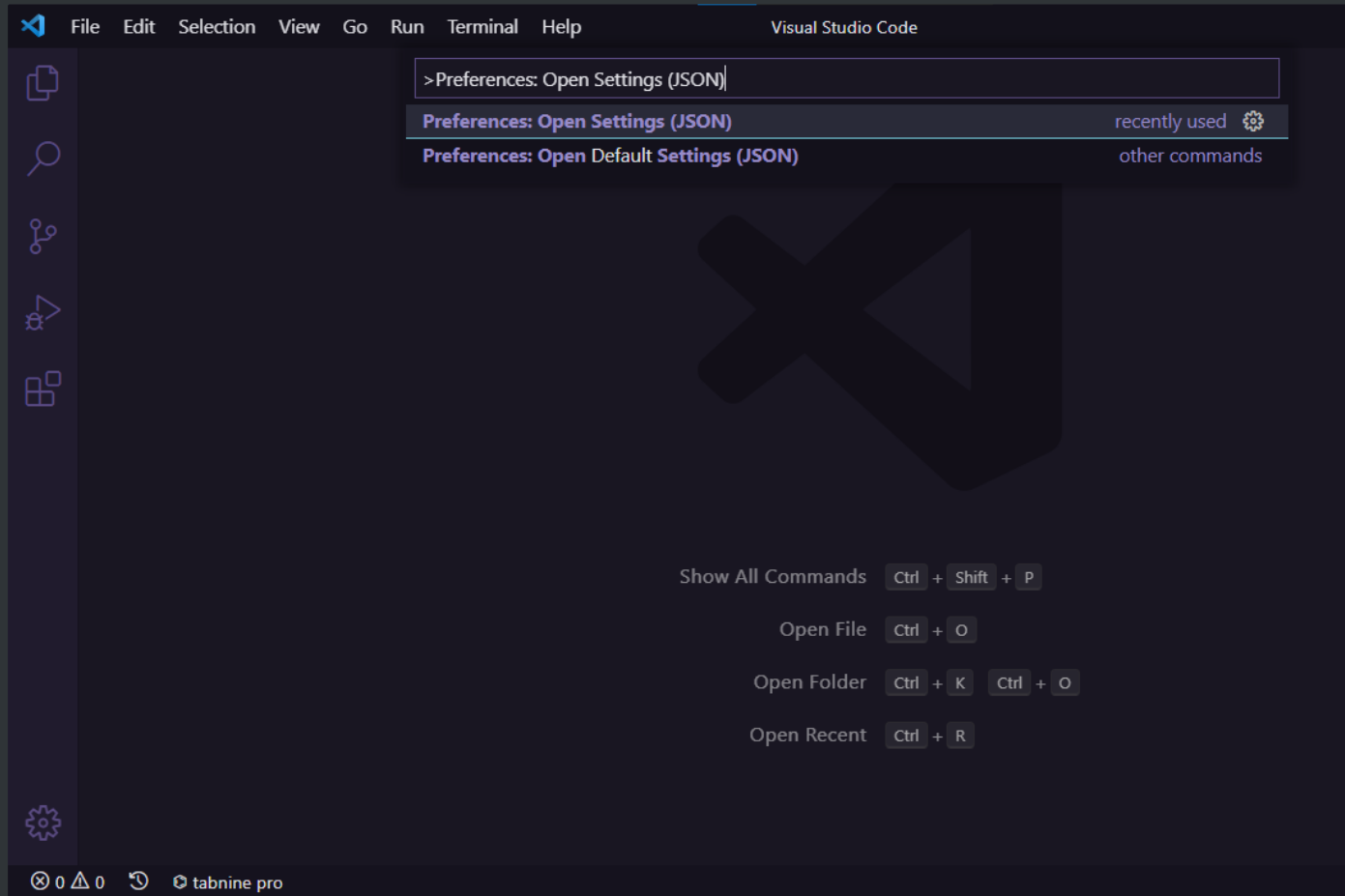
### Extensões

Você pode instalar as seguintes extensões a partir do menu de extensões do próprio VS Code:

- [Code Spell Checker](#): Essa extensão faz a correção ortográfica no nosso código, funcionando melhor com camelcase (por padrão, corrige apenas o inglês). Essa extensão é bastante útil mas é totalmente opcional;
- [Portuguese - Code Spell Checker](#): É um dicionário de português para que a extensão **Code Spell Checker** consiga fazer também a correção ortográfica em Português;
- [Color Highlight](#): Essa extensão reconhece cores CSS escritas em qualquer lugar do nosso código. Por padrão reconhece apenas cores em hexadecimal mas você pode configurar para reconhecer cores no formato de palavras como `"red"` ou `"yellow"`. É uma extensão bastante útil, já que reconhece as cores diretamente no código;
- [CSS Modules](#): Essa é uma extensão que fornece o autocomplete para CSS Modules. Recomendamos o uso dela já que vai te ajudar com o autocomplete;
- [Tabnine](#): O Tabnine é uma extensão que usa de inteligência artificial para identificar o contexto do código e fornecer o autocomplete. Suporta diversas linguagens incluindo JavaScript e TypeScript. Existe uma versão paga mas também é possível usar a versão gratuita da extensão. Seu uso é totalmente opcional;
- [vscode-styled-components](#): Essa extensão fornece o syntax highlighting e intelliSense para a biblioteca [styled-components](#). Se você for utilizar essa biblioteca, o uso da extensão é bastante recomendado.

# Configurações do VS Code

As seguintes configurações podem ser acessadas no VS Code apertando **Ctrl + Shift + P** (ou **cmd + Shift + P**, digitando **Preferences: Open Settings (JSON)** e entrando na opção encontrada:



No arquivo JSON que abriu, adicione as seguintes configurações (certifique-se de adicionar dentro das chaves **{ }**):

```
"terminal.integrated.fontSize": 14, "editor.tabSize": 2,
"editor.fontSize": 16, "editor.lineHeight": 26,
"editor.semanticHighlighting.enabled": false, "editor.rulers": [80, 120],
"editor.codeActionsOnSave": { "source.fixAll.eslint": true },
"files.exclude": { "**/.git": true, "**/.svn": true, "**/.hg": true,
"**/CVS": true, "**/.DS_Store": true, // "**/node_modules": true },
"files.associations": { ".sequelizerc": "javascript", ".stylelintrc":
"json", ".prettierrc": "json", "*.tsx": "typescriptreact" },
"typescript.tsserver.log": "verbose", "material-icon-
theme.activeIconPack": "nest", "material-icon-
theme.folders.associations": { "infra": "app", "entities": "class",
"domain": "class", "schemas": "class", "typeorm": "database",
"repositories": "mappings", "http": "container", "migrations": "tools",
"modules": "components", "implementations": "core", "dtos": "typescript",
"fakes": "mock", "websockets": "pipe", "protos": "pipe", "grpc": "pipe",
"providers": "include", "subscribers": "messages", "useCases":
"controller", "kafka": "scripts", "mappers": "meta", "_shared": "shared",
"eslint-config": "tools", "kube": "kubernetes" }, "material-icon-
theme.files.associations": { "ormconfig.json": "database",
"tsconfig.json": "tune", "*.proto": "3d", "*.webpack.js": "webpack" },
"window.menuBarVisibility": "toggle", "tabnine.experimentalAutoImports":
true, "cSpell.enableFiletypes": [ "!asciidoc", "!c", "!cpp", "!csharp",
"!go", "!handlebars", "!haskell", "!jade", "!java", "!latex", "!php",
"!pug", "!python", "!restructuredtext", "!rust", "!scala", "!scss" ],
"cSpell.language": "en,pt", "editor.suggestSelection": "first",
"cSpell.userWords": [ "chakra", "middlewares", "prefetch", "rocketseat"
], "workbench.productIconTheme": "fluent-icons",
"terminal.integrated.showExitAlert": false,
"splitHTMLAttributes.closingBracketOnNewLine": true, "window.zoomLevel":
1
```