Um init.vim pra você se inspirar 🔥

fonte

Link para uma gist com o arquivo completo 🗂️

Se você quiser copiar todo o arquivo para usar no seu vim, pode acessar por aqui.

Plug um dos gerenciadores de plugins mais famosos 🔌

Se você configura o NeoVim com vimscript, ele pode ser uma boa escolha.

```
call plug#begin()
" Visual """""
Plug 'sainnhe/sonokai'
Plug 'lukas-reineke/indent-blankline.nvim'
" AirLine """""
Plug 'vim-airline/vim-airline'
Plug 'vim-airline/vim-airline-themes'
Plug 'edkolev/tmuxline.vim'
" NerdTree """""
Plug 'preservim/nerdtree'
Plug 'Xuyuanp/nerdtree-git-plugin'
" Patterns and produtivity
Plug 'dense-analysis/ale'
Plug 'mg979/vim-visual-multi'
Plug 'preservim/nerdcommenter'
Plug 'mattn/emmet-vim'
Plug 'akinsho/toggleterm.nvim'
Plug 'nvim-lua/plenary.nvim'
Plug 'nvim-telescope/telescope.nvim'
Plug 'thaerkh/vim-workspace'
" Linters, Formaters and Language Servers
Plug 'prettier/vim-prettier', { 'do': 'yarn install --frozen-
lockfile --production' }
Plug 'ianks/vim-tsx'
" Coc """""""""""
Plug 'neoclide/coc.nvim' , { 'branch' : 'release' }
Plug 'sheerun/vim-polyglot'
call plug#end()
```

Minhas configurações ⚙️

```
" Global Sets """"""""
filetype plugin on
set clipboard+=unnamedplus
syntax on             " Enable syntax highlight
set nu
set tabstop=2         " Show existing tab with 4 spaces width
set softtabstop=2     " Show existing tab with 4 spaces width
set shiftwidth=2     " When indenting with '>', use 4 spaces
width
set expandtab         " On pressing tab, insert 4 spaces
set smarttab          " insert tabs on the start of a line
according to shiftwidth
set smartindent       " Automatically inserts one extra level of
indentation in some cases
set hidden            " Hides the current buffer when a new file
is openned
set incsearch         " Incremental search
set ignorecase        " Ingore case in search
set smartcase         " Consider case if there is a upper case
character
set scrolloff=8       " Minimum number of lines to keep above and
below the cursor
set colorcolumn=160   " Draws a line at the given line to keep
aware of the line size
set signcolumn=yes    " Add a column on the left. Useful for
linting
set cmdheight=2       " Give more space for displaying messages
set updatetime=100    " Time in miliseconds to consider the
changes
```

```
set encoding=UTF-8   " The encoding should be utf-8 to activate
the font icons
set nobackup         " No backup files
set nowritebackup    " No backup files
set splitright       " Create the vertical splits to the right
set splitbelow       " Create the horizontal splits below
set autoread         " Update vim after file update from outside
set mouse=a          " Enable mouse support
filetype on          " Detect and set the filetype option and
trigger the FileType Event
filetype plugin on   " Load the plugin file for the file type,
if any
filetype indent on   " Load the indent file for the file type,
if any
```

Meus remaps 🧭

```
" Remaps """"""""""
" Map Leader """""
let mapleader="\<space>"

" Source config """""""""
nmap <leader>! :source ~/dotfiles/.config/nvim/init.vim<cr>
nmap <leader>@ :vsplit ~/dotfiles/.config/nvim/init.vim<cr>

" Shortcuts for split navigation
map <C-h> <C-w>h
map <C-j> <C-w>j
map <C-k> <C-w>k
map <C-l> <C-w>l

" Adding an empty line below, above and below with insert
nmap op o<Esc>k
nmap oi O<Esc>j
nmap oo A<CR>

nmap mm :bn<CR>
nmap nn :bp<CR>

" Delete a buffer
nmap çç :bd<CR>
```

```
" Create splits
nmap qq :vsplit<CR>

" Close splits and others
nmap tt :qa<CR>

" Add a comma in the line ends
nmap <leader>, :%s/$/,<CR>G$xggVGyy:noh<CR>

" Replace
nmap rp :%s/

" Replace across files
nmap <Leader>rp :windo %s/

" Clear find
nmap <Leader>cf :noh<cr>

" Panel Resizing """"""""""
nnoremap <silent><Leader>+ :exe "resize " . (winheight(0) * 3/2)<CR>
nnoremap <silent><Leader>- :exe "resize " . (winheight(0) * 2/3)<CR>
nnoremap <silent><Leader>> :exe "vertical resize " . (winwidth(0) * 3/2)<CR>
nnoremap <silent><Leader>< :exe "vertical resize " . (winwidth(0) * 2/3)<CR>
```

Configurações do Emmet 😁

Esse plugin é para prover funções parecidas com o emmet do VS Code.

```
" Emmet """"""""""""""""""
let g:user_emmet_leader_key=','
let g:user_emmet_install_global = 0
autocmd FileType html,css,vue,ts,ts,js,jsx EmmetInstall
```

Toggleterm 👨🏻‍💻

Este plugin permite que você tenha um tipo de terminal integrado, parecido com o VS Code.

```
" Toggle term """""""""""""
let g:toggleterm_terminal_mapping = '<C-x>'
" or manually...
autocmd TermEnter term://*toggleterm#*
      \ tnoremap <silent><C-x> <Cmd>exe v:count1 . "ToggleTerm"
<CR>
nnoremap <silent><C-x> <Cmd>exe v:count1 . "ToggleTerm"<CR>
inoremap <silent><C-x> <Esc><Cmd>exe v:count1 . "ToggleTerm"<CR>
```

Prettier 😎

Algumas configurações do plugin official do prettier

```
" Prettier """"""""""""""
let g:prettier#quickfix_enabled = 1
nmap fd <Plug>(Prettier)
" Workaround to solve parse error
let g:prettier#config#single_quote = 'true'
let g:prettier#config#trailing_comma = 'es5'
let g:prettier#config#semi = 'false'
let g:prettier#config#parser = 'babylon'
```

Tema 📸

```
" Themes """"""""""""""""
if exists('+termguicolors')
  let &t_8f = "\<Esc>[38;2;%lu;%lu;%lum"
  let &t_8b = "\<Esc>[48;2;%lu;%lu;%lum"
  set termguicolors
endif
let g:sonokai_style = 'andromeda'
let g:sonokai_enable_italic = 1
let g:sonokai_disable_italic_comment = 0
colorscheme sonokai
```

Airline ✈️

Ele coloca uma linha bem bonita em baixo do editor, com várias informações úteis.

```
" AirLine """""""""""""""""
let g:airline#extensions#tabline#left_sep = ' '
let g:airline#extensions#tabline#left_alt_sep = '|'
let g:airline_theme = 'sonokai'
let g:airline#extensions#tabline#enabled = 1
let g:airline_powerline_fonts = 1
let g:airline#extensions#tabline#formatter = 'unique_tail'
```

Integração do Tmux com airline ❤️

```
" Tmux AirLine """""""""""
let g:tmuxline_preset = {
      \'c'    : '#H',
      \'win'  : '#I #W',
      \'cwin' : '#I #W',
      \'x'    : '%a',
      \'y'    : '%R'}
"
let g:tmuxline_theme = {
    \   'c'    : [ 244, 236 ],
    \   'x'    : [ 244, 236 ],
    \   'y'    : [ 253, 239 ],
    \   'win'  : [ 103, 236 ],
    \   'cwin' : [ 236, 103 ],
    \   'bg'   : [ 244, 236 ],
    \ }
```

Explorador de arquivos 📂

```
" NerdTree """"""""""""""""""
nmap <C-a> :NERDTreeToggle<CR>
" Open the existing NERDTree on each new tab.
autocmd BufWinEnter * if getcmdwintype() == '' | silent
NERDTreeMirror | endif
```

Gerenciar autosave e sessões □

```
" vim-workspace """""""""""""
let g:workspace_autosave_always = 1
" let g:workspace_autocreate = 1
" let g:workspace_session_directory = $HOME . '/.vim/sessions/'
```

Lint engine 🧹

```
" ALE """"""""""""""""""""""
let g:ale_linters = {
\ 'javascript': ['prettier'],
\ 'css': ['prettier'],
\ 'yaml': ['prettier'],
\ 'json': ['prettier'],
\}

let g:ale_fixers = {
\   '*': ['trim_whitespace'],
\}
let g:ale_fix_on_save = 0
```

Buscador de arquivos e palavras (tipo Ctrl-P e Ctrl-F) 🔎

```
" Telescope """"""""""""""
nnoremap ff <cmd>Telescope find_files find_command=rg,--ignore,-
-hidden,--files<cr>
nnoremap fg <cmd>Telescope live_grep<cr>
nnoremap fb <cmd>Telescope buffers<cr>
```

Pluguin para requisições HTTP 🍊

```
"Rest Client """"""""""""''
noremap <Leader>\ :CocCommand rest-client.request <cr>
```

Comentador de código 🙈

```
" Nerd Commenter
let g:NERDSpaceDelims = 1
```

Auto complete e language server 🍹

```
" Coc Of Completion """""""
let g:coc_global_extensions = [
\ 'coc-pairs',
\ 'coc-json',
\ 'coc-tsserver',
\ 'coc-restclient',
\]

set shortmess+=c
inoremap <silent><expr> <TAB>
      \ pumvisible() ? "\<C-n>" :
      \ <SID>check_back_space() ? "\<TAB>" :
      \ coc#refresh()
inoremap <expr><S-TAB> pumvisible() ? "\<C-p>" : "\<C-h>"

function! s:check_back_space() abort
  let col = col('.') - 1
  return !col || getline('.')[col - 1]  =~# '\s'
endfunction
if has('nvim')
  inoremap <silent><expr> <c-space> coc#refresh()
else
  inoremap <silent><expr> <c-@> coc#refresh()
endif

inoremap <silent><expr> <cr> pumvisible() ?
coc#_select_confirm()
                              \: "\<C-g>u\<CR>\<c-
r>=coc#on_enter()\<CR>"
"
" Formatting selected code.
xmap <leader>fd <Plug>(coc-format-selected)
nmap <leader>fd <Plug>(coc-format-selected)

" Navigate on diagnostics
nmap <silent> [d <Plug>(coc-diagnostic-prev)
nmap <silent> ]d <Plug>(coc-diagnostic-next)

" GoTo code navigation.
```

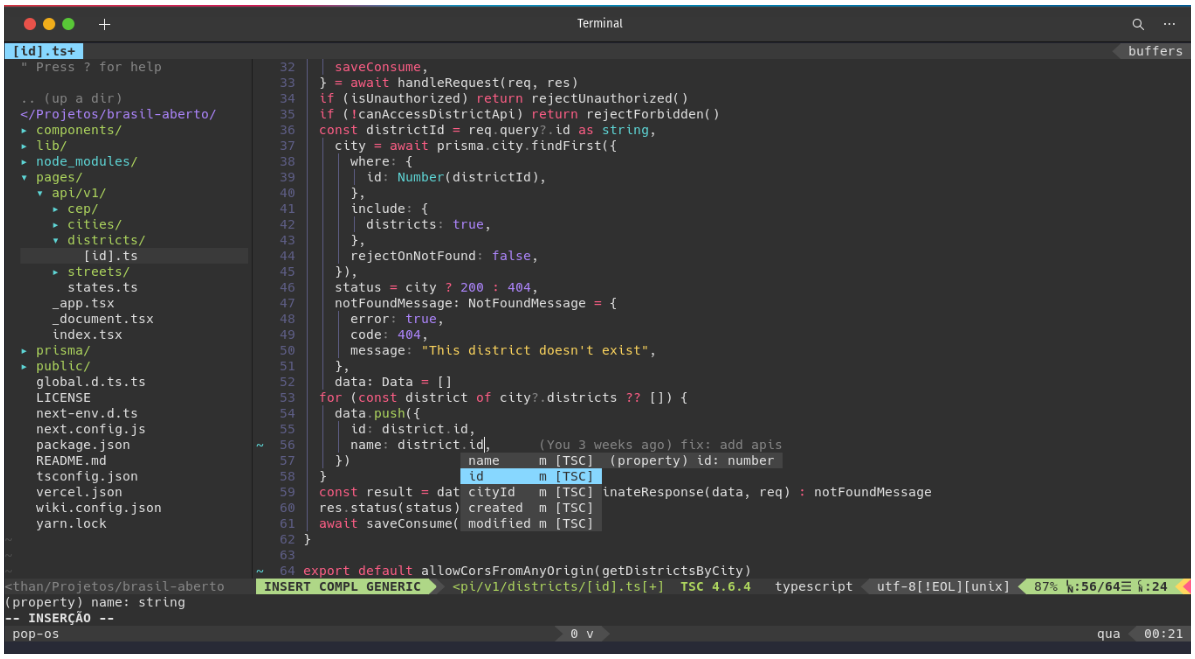
```
nmap <silent> gd <Plug>(coc-definition)
nmap <silent> gy <Plug>(coc-type-definition)
nmap <silent> gi <Plug>(coc-implementation)
nmap <silent> gr <Plug>(coc-references)

" Use K to show documentation in preview window.
nnoremap <silent> K :call <SID>show_documentation()<CR>

" Symbol renaming.
nmap <leader>rn <Plug>(coc-rename)

" Apply AutoFix to problem on the current line.
nmap <leader>qf  <Plug>(coc-fix-current)

" Run the Code Lens action on the current line.
nmap <leader>cl  <Plug>(coc-codelens-action)

function! s:show_documentation()
  if (index(['vim','help'], &filetype) >= 0)
    execute 'h '.expand('<cword>')
  elseif (coc#rpc#ready())
    call CocActionAsync('doHover')
  else
    execute '!' . &keywordprg . " " . expand('<cword>')
  endif
endfunction

autocmd CursorHold * silent call CocActionAsync('highlight')

augroup mygroup
  autocmd!
  " Setup formatexpr specified filetype(s).
  autocmd FileType typescript,json setl
formatexpr=CocAction('formatSelected')
  " Update signature help on jump placeholder.
  autocmd User CocJumpPlaceholder call
CocActionAsync('showSignatureHelp')
augroup end
" Remap <C-f> and <C-b> for scroll float windows/popups.
if has('nvim-0.4.0') || has('patch-8.2.0750')
```

```
  nnoremap <silent><nowait><expr> <C-f> coc#float#has_scroll() ?
coc#float#scroll(1) : "\<C-f>"
  nnoremap <silent><nowait><expr> <C-b> coc#float#has_scroll() ?
coc#float#scroll(0) : "\<C-b>"
  inoremap <silent><nowait><expr> <C-f> coc#float#has_scroll() ?
"\<c-r>=coc#float#scroll(1)\<cr>" : "\<Right>"
  inoremap <silent><nowait><expr> <C-b> coc#float#has_scroll() ?
"\<c-r>=coc#float#scroll(0)\<cr>" : "\<Left>"
  vnoremap <silent><nowait><expr> <C-f> coc#float#has_scroll() ?
coc#float#scroll(1) : "\<C-f>"
  vnoremap <silent><nowait><expr> <C-b> coc#float#has_scroll() ?
coc#float#scroll(0) : "\<C-b>"
endif

" Add `:Format` command to format current buffer.
command! -nargs=0 Format :call CocActionAsync('format')

" Add `:Fold` command to fold current buffer.
command! -nargs=? Fold :call     CocAction('fold', <f-args>)

" Add `:OR` command for organize imports of the current buffer.
command! -nargs=0 OR   :call     CocActionAsync('runCommand',
'editor.action.organizeImport')

" Add (Neo)Vim's native statusline support.
" NOTE: Please see `:h coc-status` for integrations with
external plugins that
" provide custom statusline: lightline.vim, vim-airline.
set statusline^=%{coc#status()}%
{get(b:,'coc_current_function','')}

" Mappings for CoCList
" Show all diagnostics.
nnoremap <silent><nowait> <space>a  :<C-u>CocList
diagnostics<cr>
" Manage extensions.
nnoremap <silent><nowait> <space>e  :<C-u>CocList extensions<cr>
" Show commands.
nnoremap <silent><nowait> <space>c  :<C-u>CocList commands<cr>
```

Integração com Git 💾

```
" Coc-git """""""""""""
" navigate chunks of current buffer
nmap ]c <Plug>(coc-git-prevchunk)
nmap [c <Plug>(coc-git-nextchunk)
" Adicionar alterações ao staging
nmap <leader>ss :CocCommand git.chunkStage<cr>
" Desfazer alterações
nmap <leader>uu :CocCommand git.chunkUndo<cr>
" Manter alterações remotas
nmap <leader>ki :CocCommand git.keepIncoming<cr>
" Manter alterações locais
nmap <leader>kc :CocCommand git.keepCurrent<cr>
" Mantes ambas as alterações
nmap <leader>kb :CocCommand git.keepBoth<cr>
nmap gs <Plug>(coc-git-chunkinfo)
```

Resultado



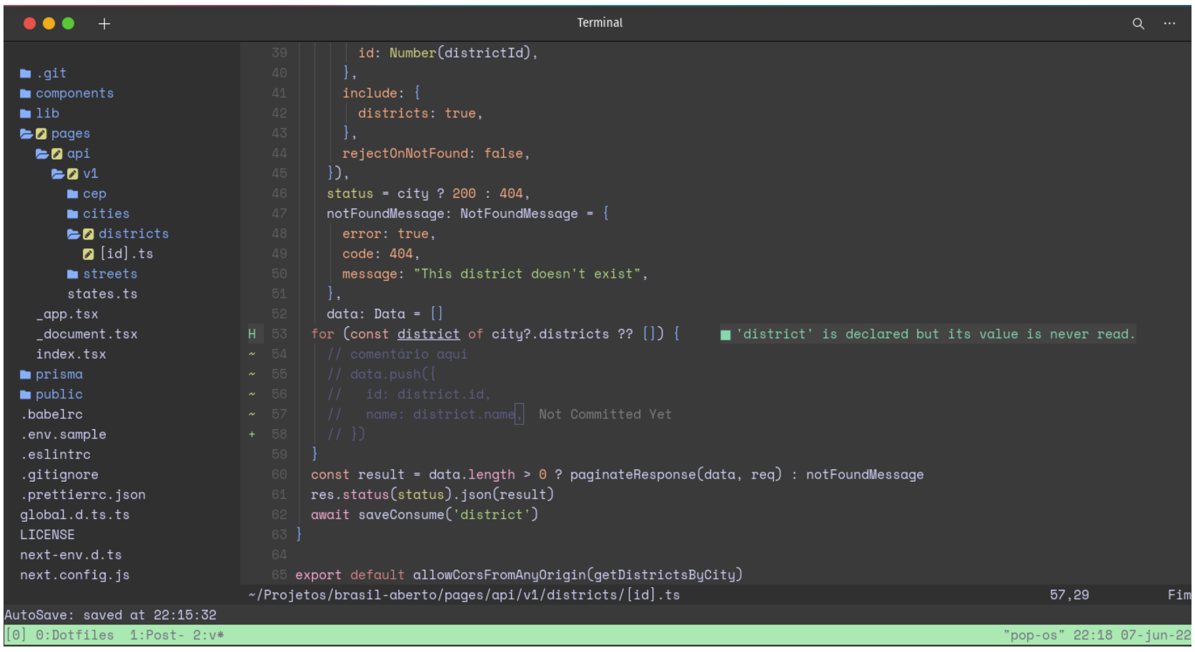
```
" Press ? for help

.. (up a dir)
</Projetos/brasil-aberto/
▸ components/
▸ lib/
▸ node_modules/
▾ pages/
  ▾ api/v1/
    ▸ cep/
    ▸ cities/
    ▾ districts/
        [id].ts
    ▸ streets/
      states.ts
    _app.tsx
    _document.tsx
    index.tsx
▸ prisma/
▸ public/
  global.d.ts.ts
  LICENSE
  next-env.d.ts
  next.config.js
  package.json
  README.md
  tsconfig.json
  vercel.json
  wiki.config.json
  yarn.lock
```

```
    saveConsume,
  } = await handleRequest(req, res)
  if (isUnauthorized) return rejectUnauthorized()
  if (!canAccessDistrictApi) return rejectForbidden()
  const districtId = req.query?.id as string,
    city = await prisma.city.findFirst({
      where: {
        id: Number(districtId),
      },
      include: {
        districts: true,
      },
      rejectOnNotFound: false,
    }),
    status = city ? 200 : 404,
    notFoundMessage: NotFoundMessage = {
      error: true,
      code: 404,
      message: "This district doesn't exist",
    },
    data: Data = []
  for (const district of city?.districts ?? []) {
    data.push({
      id: district.id,
      name: district.id,       (You 3 weeks ago) fix: add apis
    })                  name     m [TSC]  (property) id: number
  }                     id       m [TSC]
  const result = dat cityId   m [TSC] inateResponse(data, req) : notFoundMessage
  res.status(status) created  m [TSC]
  await saveConsume( modified m [TSC]
}

export default allowCorsFromAnyOrigin(getDistrictsByCity)
```

```
<than/Projetos/brasil-aberto  INSERT COMPL GENERIC  <pi/v1/districts/[id].ts[+]  TSC 4.6.4  typescript  utf-8[!EOL][unix]  87% ␤:56/64☰ ␤:24
(property) name: string
-- INSERÇÃO --
 pop-os                                   0 v                                   qua  00:21
```

Se você gostou do conteúdo, confira meu [outro post sobre neovim]() em que eu mostro minhas novas configurações utilizando a linguagem lua! 🌔

> Esse é o visual da nova configuraçao



```
.git
components
lib
pages
  api
    v1
      cep
      cities
      districts
        [id].ts
      streets
      states.ts
  _app.tsx
  _document.tsx
  index.tsx
prisma
public
.babelrc
.env.sample
.eslintrc
.gitignore
.prettierrc.json
global.d.ts.ts
LICENSE
next-env.d.ts
next.config.js
```

```
        id: Number(districtId),
      },
      include: {
        districts: true,
      },
      rejectOnNotFound: false,
    }),
    status = city ? 200 : 404,
    notFoundMessage: NotFoundMessage = {
      error: true,
      code: 404,
      message: "This district doesn't exist",
    },
    data: Data = []
H   for (const district of city?.districts ?? []) {      'district' is declared but its value is never read.
~     // comentário aqui
~     // data.push({
~     //   id: district.id,
~     //   name: district.name,  Not Committed Yet
+     // })
  }
  const result = data.length > 0 ? paginateResponse(data, req) : notFoundMessage
  res.status(status).json(result)
  await saveConsume('district')
}

export default allowCorsFromAnyOrigin(getDistrictsByCity)
~/Projetos/brasil-aberto/pages/api/v1/districts/[id].ts                    57,29          Fim
AutoSave: saved at 22:15:32
[0] 0:Dotfiles  1:Post- 2:v*                                    "pop-os" 22:18 07-jun-22
```