Minicurso Vim

André William Régis
Pedro Santi Binotto

UFSC

2024

Material de apoio

```
$ git clone –recursive
https://github.com/AndreKuru/Minicurso-Vim.git
```

pratical-vim-examples

Vim

Principais modos

Normal: navegar e editar

Insert: inserção de texto

Visual: editar através de seleção

Command-line: retentor de usuários de primeira viagem

Digitação ideal

Imagem não adaptada.
Disponível em: Wikimedia

Movimentação básica

j: ↓

k: ↑

l: →

h: ←

Jogo para praticar:
vim-adventures.com

Imagem adaptada.
Disponível em: Wikimedia

Quando não se precisa manter a mão no mouse

u: **u**ndo change

dd: **d**elete line

yy: **y**ank line

p: **p**ut text

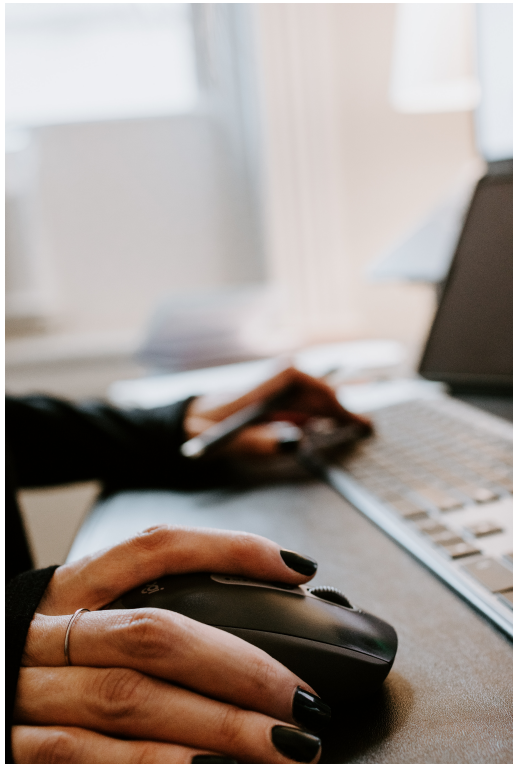


Imagem não adaptada.
Disponível em:
Unsplash

Mas se quiser pode

:set mouse=a

Experimentando cada modo

i: switch to **I**nsert mode

v: switch to **V**isual mode

:: switch to Command-Line mode

Como sair do vim?

:q : **q**uit

:q! : **q**uit without writing

:w : **w**rite

:wq : **w**rite and **q**uit

:x : write (if needed) and quit

Comando ponto

`.`: "Repeat last change [...]"

Mudanças comuns

x : delete [count] char(s)

s : **s**ubstitute [count] char(s)

[] : "[...] are optional."

[count] : "An optional number that may precede the command to multiply or iterate the command."

d{motion} : **d**elete

c{motion} : **c**hange

{} : "[...] must appear, but which can take a number of different values."

{motion} : "A command that moves the cursor."

Movimentação na linha

0: start of the line

$: end of the line

^: start non-blank of the line

g_: end non-blank of the line

Movimentação através de palavras

w: **w**ord foward

e: word **e**nd foward

b: word **b**ackward

ge: word end backward

word: "[...]
letters, digits and underscores,
or a sequence of other non-blank
characters [...]"

blank characters: "space and tab"

Movimentação através de PALAVRAS

W: **W**ORD foward

E: WORD **e**nd foward

B: WORD **b**ackward

gE: WORD end backward

WORD: "[...] sequence of non-blank characters [...]"

blank characters: "space and tab"

Buscas dentro da linha

f: **f**ind char
to the right

t: **t**ill before char
to the right

F: **f**ind char
to the left

T: **t**ill after char
to the left

;: "Repeat latest f, t, F or T [...]"

,: "Repeat latest f, t, F or T
in opposite direction [...]"

Buscas no arquivo

/: search forward for the pattern

?: search backward for the pattern

***** : search forward for the nearest word

#: search backward for the nearest word

n: "Repeat latest "/"or "?"[...]"

N: "Repeat latest "/"or "?" in opposite direction [...]"

Operadores

c: **c**hange

d: **d**elete

y: **y**ank

>: shift right

<: shift left

=: autoident

g~: swap case

gu: make lowercase

gU: make uppercase

Teclas especiais

<Enter>: Enter

<CR>: Carriage Return

<Esc>: Escape

<Up>: Up arrow key

<S-...>: Shift + ...

<S-Down>: Shift + Down arrow key

<A-...>: Alt + ...

Ctrl-A: Ctrl + a

<C-...>: Ctrl + ...

<C-x>: Ctrl + x

Aritmética simples

<C-a> : **a**dd [count] to the number at or after the cursor

<C-x> : subtract [count] to the number at or after the cursor

Nem tudo se transforma

i: **i**nsert text

a: **a**ppend text

I: **i**nsert text at the start of the line

A: **a**ppend text at the end of the line

o: **o**btain new line below

O: **o**btain new line above

Correções rápidas

<C-h>: delete back one character (<BS>)

<C-w>: delete back one **w**ord

<C-u>: delete back to the start of line

Transições

<Esc>: switch to Normal mode

<C-[>: switch to Normal mode

<C-o>: switch to Insert Normal mode

Acessar registradores

<C-r>{register}: "Insert the contents of a register."

{register}:

- **+**: the clipboard contents
- *****: the clipboard contents (X11: primary selection)
- **0**: contains the text from the most recent yank command
- **1-9**: contains the text deleted in ascending order by the most recent delete or change command
- **=**: the expression register

Registradores limpos

Registradores substituíveis:

a, **b**, **c**, **d**, **e**, **f**, **g**,
h, **i**, **j**, **k**, **l**, **m**, **n**,
o, **p**, **q**, **r**, **s**, **t**, **u**,
v, **w**, **x**, **y**, **z**

Registradores extensíveis:

A, **B**, **C**, **D**, **E**, **F**, **G**,
H, **I**, **J**, **K**, **L**, **M**, **N**,
O, **P**, **Q**, **R**, **S**, **T**, **U**,
V, **W**, **X**, **Y**, **Z**

Repetição é a chave

@: : execute the last ex command

@@ : execute the last **@**

@{register} : execute the contents of register

q : record/stop macro

Inserir caracteres por código

<C-v>191: £

<C-v>u00bf: £

<C-v>u00b0: ř

<C-v>{nondigit}: e.g. **<Home>**

<C-k>{char1}{char2}

<C-k>?I: £

<C-k>12: ¡

ga: print the ascii value

Mais um modo

R: **R**eplace mode

gR: Virtual **R**eplace mode

r: **r**eplace single character

gr: virtual **r**eplace single character

Cada modo uma orientação

v : switch character-wise **V**isual mode

V : switch line-wise **V**isual Line mode

<C-v> : switch block-wise **V**isual Block mode

<Esc> : switch to Normal mode

<C-[> : switch to Normal mode

gv : re-select last **v**isual selection

o : go to **o**ther end of selected text

Seleção entre delimitadores

a) or **ab**: **a** pair of **(**parentheses**)**

a]: **a** pair of **[**brackets**]**

a} or **aB**: **a** pair of {braces}

a>: **a** pair of <>

a': **a** pair of 'single quotes'

a‘: **a** pair of ‘backticks‘

a": **a** pair of **""**double quotes**"**

at: **a** pair of **t**ags

Seleção entre delimitadores

i) or **ib**: **i**nside of **(**parentheses**)**

i} or **iB**: **i**nside of {braces}

i': **i**nside of 'single quotes'

i": **i**nside of "double quotes"

i]: **i**nside of **[**brackets**]**

i>: **i**nside of <>

**i`**: **i**nside of `backticks`

it: **i**nside of **t**ags

Seleção de objeto

aw: **a**round of **w**ord

iw: **i**nside of **w**ord

aW: **a**round of **W**ORD

iW: **i**nside of **W**ORD

as: **a**round of **s**entence

is: **i**nside of **s**entence

ap: **a**round of **p**aragraph

ip: **i**nside of **p**aragraph

Falando do ex

:print : print current line

:3print : print line 3

:2,4print : print lines 2 to 4

:'<,'>print : print lines selected from visual mode

Comandos básicos

:[range]print

:[range]join

:[range]copy {address}

:[range]move {address}

[range] : **{address}[,{address}]**

{address} :

- **number** : absolute line number
- **.** : current line
- **$** : last line in the file
- **%** : current file
- **/{pattern}[/]** : next line [...]
- **?{pattern}[?]** : previous line [...]

Comandos básicos

:[range]delete [x]

:[range]yank [x]

:[range]put [x]

[x]: "[into register x]"

a, **b**, **c**, **d**, **e**, **f**, **g**, **h**,
i, **j**, **k**, **l**, **m**, **n**, **o**, **p**,
q, **r**, **s**, **t**, **u**, **v**, **w**, **x**,
y, **z**

Os mais populares

:[range]normal {address}

:[range]substitute/{pattern}/{string}/[flags]

:[range]global/{pattern}/[cmd]

[cmd]: command from command line

Comandos externos

:shell : start shell

:!{cmd} : execute **{cmd}** in the shell

Comandos externos integrados com o arquivo atual

:read !{cmd}: execute **{cmd}** in the shell
and put the output as text below cursor

:[range]write !{cmd}: execute **{cmd}** in the shell
with **[range]** lines as input

:[range] !{filter}: execute **{cmd}** in the shell
and put the output as text below cursor
with **[range]** lines as input

Definição de *patterns*

Patterns = Regex + Magic

> *Some characters in the pattern, such as letters, are taken literally [...] Other characters have a special meaning without a backslash. If a character is taken literally or not depends on the 'magic' option and the items in the pattern mentioned next[Moo].*

Definição de *patterns*

REGEX

Uma **expressão regular** pode ser definida como uma linguagem ou sequência de caracteres que é aceita por um autômato finito;

Expressões Regulares são usualmente adotadas como uma solução para identificar ocorrências de padrões em processamento de texto.

Uma breve história

O conceito foi formalizado em 1951 pelo matemárico americano Stephen Cole Kleene [**?**, kleene1956]

Expressões regulares foram popularizadas da década de 1980 em diante, tornando-se o foco principal de muitas ferramentas essenciais de sistemas UNIX, como `grep`, `sed`, e `AWK`.

Alguns exemplos de expressões regulares

```
/[A-Z]\w*/
```

```
/\.[a-z]\+\(config\|rc\)\?\.\(json\|ya\?ml\)/
```

Operadores

^ Início de linha

$ Final de linha

* Zero ou mais vezes

+ Uma ou mais vezes

|: Operador "OU"

?: Opcional

[]: Agrupamento de classes

(): Agrupamento de expressões

Classes de caracteres

`\w`: Word character

`\d`: Digit

`\s`: Whitespace

Rolar texto

<C-E>: scroll window
(**E**xtra lines)

<C-Y>: scroll window
(**Y**rineu)

<C-D>: scroll window
Donwards

<C-U>: scroll window
Upwards

<C-F>: scroll window
Forwards

<C-B>: scroll window
Backwards

Saltando pelo texto

[count]G: go to line number

%: jump to matching parenthesis

m{letter}: set **{mark}** **{letter}** at cursor position

`{mark}: jump to a **{mark}**

Navegando pelos saltos

<C-]>: "Jump to the definition of the keyword under the cursor."

<C-o>: go to **o**lder cursor position in jump list.

<C-i>: go to ~~**i**rineu~~ newer cursor position in jump list.

Autocomplete

<C-n>: find **n**ext match

<C-p>: find **p**revious match

Buffers, windows, tabs

Summary:

A buffer is the in-memory text of a file.
A window is a viewport on a buffer.
A tab page is a collection of windows[Moo].

Buffers no Vim

A window is a viewport onto a buffer. You can use multiple windows on one buffer, or several windows on different buffers.

A buffer is a file loaded into memory for editing. The original file remains unchanged until you write the buffer to the file[Moo].

Buffers no Vim

A buffer can be in one of three states:

active: The buffer is displayed in a window. If there is a file for this buffer, it has been read into the buffer. The buffer may have been modified since then and thus be different from the file.

hidden: The buffer is not displayed. If there is a file for this buffer, it has been read into the buffer. Otherwise it's the same as an active buffer, you just can't see it.

inactive: The buffer is not displayed and does not contain anything. Options for the buffer are remembered if the file was once loaded. It can contain marks from the **shada** file. But the buffer doesn't contain text[Moo].

Trabalhando com *buffers*

Buffer states			
state	displayed in window	loaded	**:buffers** shows
active	yes	yes	'a'
hidden	no	yes	'h'
inactive	no	no	' '

Trabalhando com *buffers*

:e : **e**dit file

:bn : **b**uffer **n**ext

:bp : **b**uffer **p**revious

:bd : **b**uffer **d**elete

:b[n] : ex.: *buffer**1*** *buffer**2***

Windows: *viewports* bara *buffers*

:split : Divide horizontalmente a tela com uma nova janela;

:split ./outro/arquivo.txt

:vsplit : Divide verticalmente a tela com uma nova janela;

:vsplit ./outro/arquivo.txt

:close[n] : Fecha uma janela (janela ativa ou **[n]**);

:close , **:close 1**

Windows: *viewports* bara *buffers*

<C-W>s: **:split**

<C-W>v: **:vsplit**

Note: All **CTRL-W** commands can also be executed with **:wincmd**, for those places where a Normal mode command can't be used or is inconvenient[Moo].

Windows: *viewports* bara *buffers*

<C-W>[hjkl]: Move to window above/below/right of/left of current one.

<C-W>[HJKL]: Move window above/below/to the right/to the left.

Note: All **CTRL-W** commands can also be executed with **:wincmd**, for those places where a Normal mode command can't be used or is inconvenient[Moo].

Tab-pages

A tab page holds one or more windows. You can easily switch between tab pages, so that you have several collections of windows to work on different things[Moo].

Tab-pages

:tabe : **tab e**dit

:tabnew : **tab new**

:tabc : **tab c**lose

:tabn : **tab n**ext

:tabp : **tab p**revious

:tabs : **tabs** (plural de tab)

O sucessor moderno

Neovim is exactly what it claims to be. It fixes every issue I have with Vim.

—Geoff Greer

Full-screen Neovim looks cool as hell!

—DHH

A nice looking website, that's one thing Neovim did right.

—Bram Moolenaar

Imagem não adaptada.
Disponível em: Neovim

O sucessor moderno

API versionada, documentada;

Completamente configurável em Lua;

Ecossistema extensivo de plugins desenvolvidos pela comunidade;

Analisador sintático *TreeSitter* permite highlighting e navegação de código a nivel de objetos sintáticos;

Cliente de LSP embutido (mesmo protocolo do *VSCode*!) habilita funcionalidades de inteligência de código (refatoração automática, *code actions*, navegação por referência, etc...);

Language Server Protocol

O que é LSP?

> *"Adding features like auto complete, go to definition, or documentation on hover for a programming language takes significant effort. Traditionally this work had to be repeated for each development tool, as each tool provides different APIs for implementing the same feature.* ***A Language Server is meant to provide the language-specific smarts and communicate with development tools over a protocol that enables inter-process communication****[mic]."*

Language Server Protocol

Protocolo

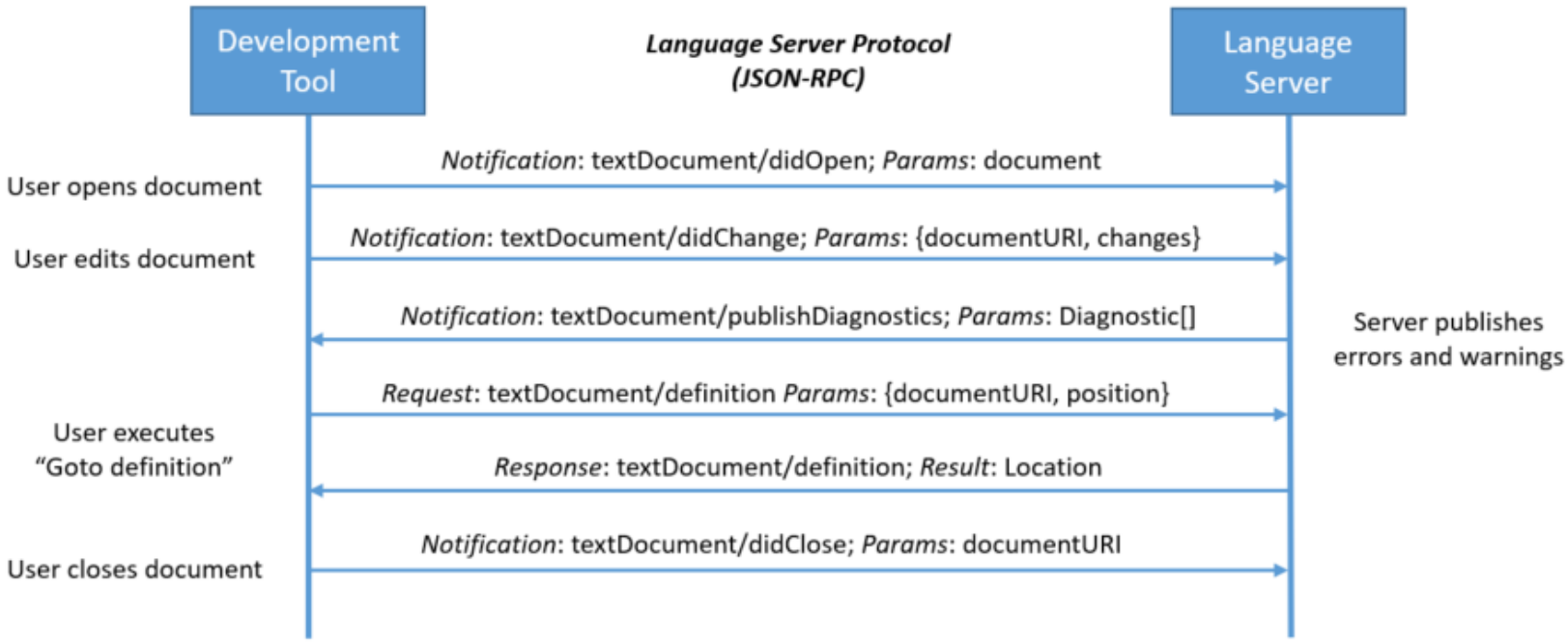


Imagem não adaptada.
Disponível em: Microsoft - Language Server Protocol

Language Server Protocol - Instalação e Configuração

Suporte nativo da API

```
lspconfig.rust_analyzer.setup {
settings = {
  ['pyright'] = {
    python = {
      analysis = {
        diagnosticMode = "workspace",
        typeCheckingMode = "off",
      },
    },
  },
  ['rust-analyzer'] = {},
  ['gopls'] = {},
  ['typescript-language-server'] = {},
  ...
},
```

Além de oferecer *bindinds* nativas para a comunicação com LSPs, a configuração dos clientes pode ser feita de forma simples e direta através do pacote `nvim-lspconfig`

Language Server Protocol - Instalação e Configuração

`nvim-lspconfig` + `Mason.nvim` = Configuração rápida, fácil, e portátil

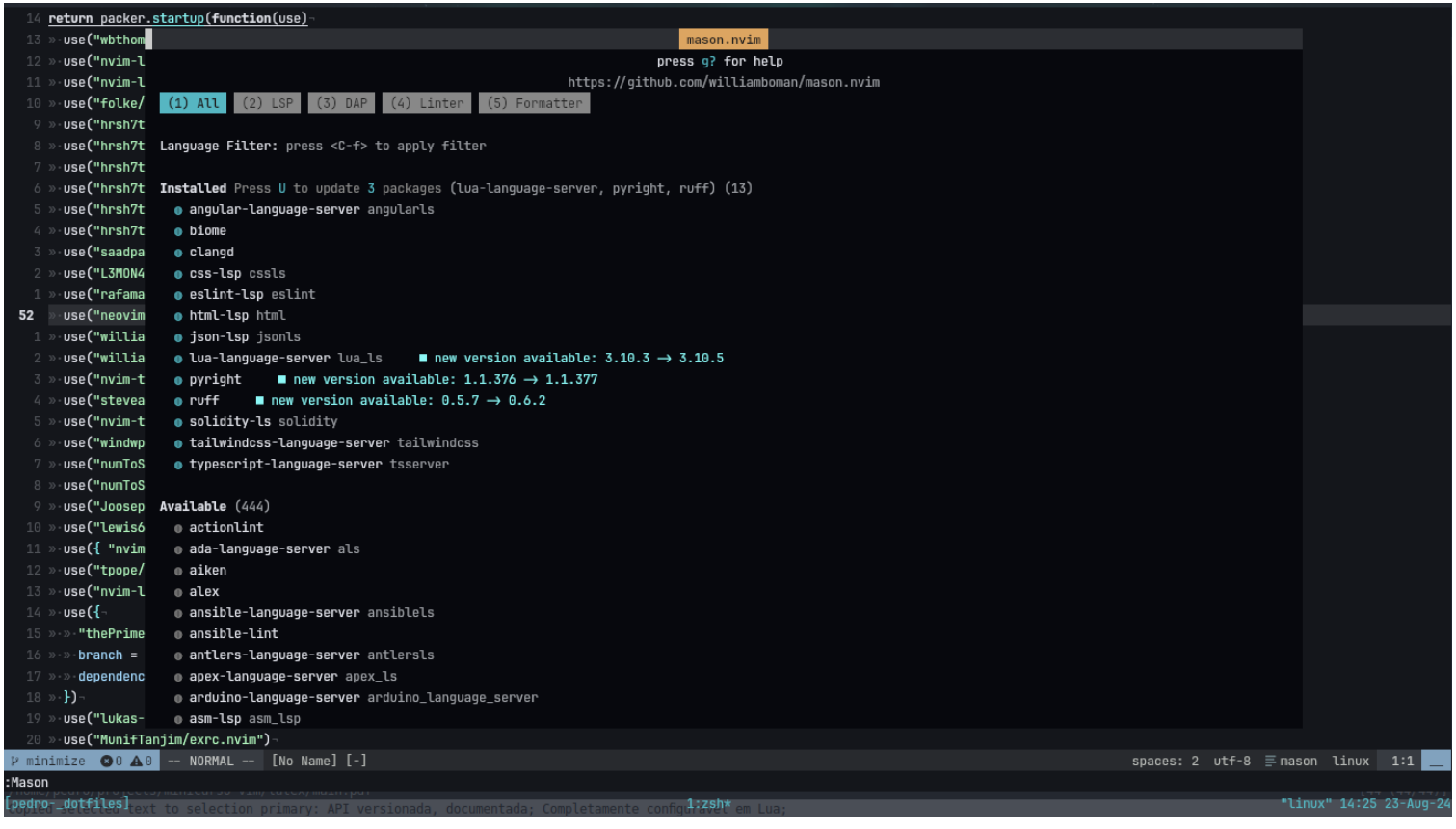
mason.nvim

Imagem não adaptada.
Disponível em: Mason.nvim

Portable package manager for Neovim that runs everywhere Neovim runs. Easily install and manage LSP servers, DAP servers, linters, and formatters.[mas]

Language Server Protocol - Instalação e Configuração

Um package-manager para servidores LSP



Instalação de LSPs via *Mason.nvim*

TreeSitter

O motor sintático

"Tree-sitter is a parser generator tool and an incremental parsing library. It can build a concrete syntax tree for a source file and efficiently update the syntax tree as the source file is edited."

TreeSitter

O motor sintático

General enough to parse any programming language;

Fast enough to parse on every keystroke in a text editor;

Robust enough to provide useful results even in the presence of syntax errors;

Dependency-free so that the runtime library (which is written in pure C) can be embedded in any application;

TreeSitter

O motor sintático

> *"The goal of nvim-treesitter is both to provide a simple and easy way to use the interface for tree-sitter in Neovim and to provide some basic functionality such as highlighting based on it."*

TreeSitter

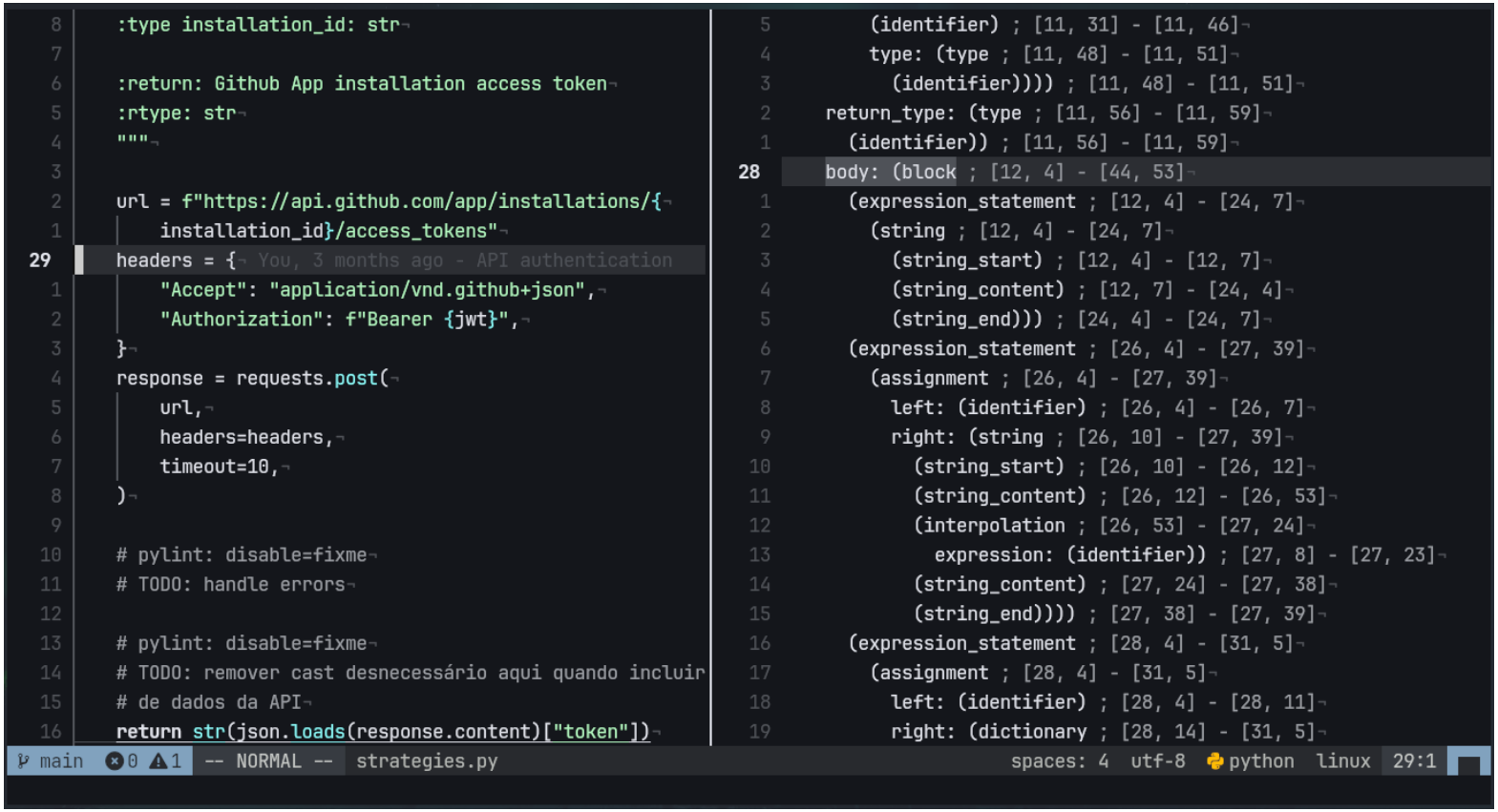
O motor sintático

```
int tslua_add_language(lua_State *L)
{
  if (lua_gettop(L) < 2 || !lua_isstring(L, 1) || !lua_isstring(L, 2)) {
    return luaL_error(L, "string expected");
  }

  const char *path = lua_tostring(L, 1);
  const char *lang_name = lua_tostring(L, 2);

  if (pmap_has(cstr_t)(langs, lang_name)) {
    return 0;
  }

#define BUFSIZE 128
  char symbol_buf[BUFSIZE];
  snprintf(symbol_buf, BUFSIZE, "tree_sitter_%s", lang_name);
#undef BUFSIZE

  uv_lib_t lib;
  if (uv_dlopen(path, &lib)) {
    snprintf((char *)IObuff, IOSIZE, "Failed to load parser: uv_dlopen: %s",
             uv_dlerror(&lib));
    uv_dlclose(&lib);
    lua_pushstring(L, (char *)IObuff);
    return lua_error(L);
  }

  TSLanguage *(*lang_parser)(void);
  if (uv_dlsym(&lib, symbol_buf, (void **)&lang_parser)) {
    snprintf((char *)IObuff, IOSIZE, "Failed to load parser: uv_dlsym: %s",
             uv_dlerror(&lib));
    uv_dlclose(&lib);
    lua_pushstring(L, (char *)IObuff);
    return lua_error(L);
  }

  TSLanguage *lang = lang_parser();
  if (lang == NULL) {
    return luaL_error(L, "Failed to load parser: internal error");
  }

  pmap_put(cstr_t)(langs, xstrdup(lang_name), lang);

  lua_pushboolean(L, true);
  return 1;
}
```

Highlighting de sintaxe através da árvore construída

TreeSitter

O motor sintático



A ferramenta permite inspecionar a estrutura da árvore sintática em tempo real

Plugins

Plugin managers

`pathogen`

`vundle`

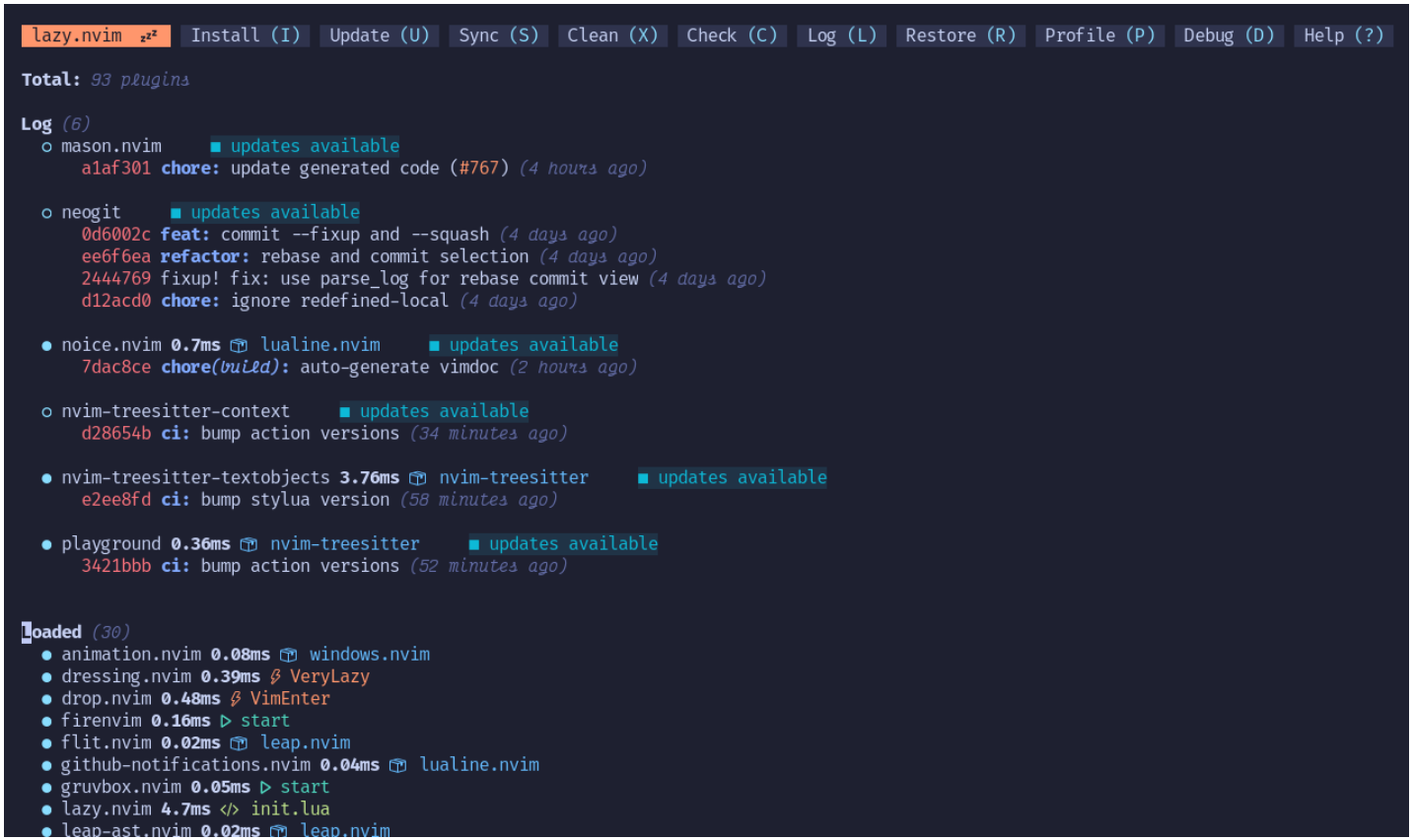
`vim-plug`

`packer.nvim`

`lazy.nvim`

Plugins

Plugin managers



Instalação interativa de plugins com `lazy.nvim`

Lua

Brasil mentioned!?!?!?

designed and developed at

PUC RIO

Imagem não adaptada.
Disponível em: www.lua.org

Lua

API

The "Vim API"inherited from Vim: Ex-commands and builtin-functions as well as user-functions in Vimscript. These are accessed through `vim.cmd()` and `vim.fn` respectively.

The "Nvim API"written in C for use in remote plugins and GUIs; see api. These functions are accessed through `vim.api`.

Recomendações

vim-adventures.com

```
$ vimtutor
```

:help

Referência


S. C. Kleene, *Representation of events in nerve nets and finite automata*, pp. 3–42, Princeton University Press, Princeton, 1956.


Mason.nvim - portable package manager for neovim that runs everywhere neovim runs, `https://github.com/williamboman/mason.nvim`, Accessed: 2024-09-1.


Microsoft - language server protocol, `https://microsoft.github.io/language-server-protocol/`, Accessed: 2024-09-1.


B Moolenaar, *Vim reference manual*, Accessed: 2024-09-1.


Drew Neil, *Practical vim: Edit text at the speed of thought*, Practical Vim (2015), 1–356.