Atom
O Editor Open Source do GitHub

Ricardo Henrique Brunetto[1]

[1]Departamento de Informática – Universidade Estadual de Maringá (UEM)
Maringá – PR – Brasil

ra94182@uem.br

Resumo. *O presente trabalho visa a apresentar de forma instrutiva as funcionalidades do editor de texto Atom e tratar a respeito de suas vantagens, desvantagens, formas de instalação e recursos que o diferenciam dos demais editores. Este trabalho segue referência direta da documentação oficial do Atom [GitHub], embora algumas seções não sejam mencionadas a fim de manter concisão.*

Contents

1. Introdução

O Atom é um editor de texto de código aberto desenvolvido pelo GitHub e liberado como beta em junho de 2015 sob a Licença MIT [MIT], projetado para combinar flexibilidade e extensibilidade. De forma geral, o Atom é um editor focado em desenvolvimento de código, oferecendo recursos e ferramentas para solucionar dos mais ínfimos aos mais significativos inconvenientes do ofício. Além disso, por ser um software livre e aberto e fornecer suporte à extensões criadas pelos própriros usuários, o Atom passou a ser conhecido como ”Editor de Texto Hackeável do Século 21”.

O Atom foi desenvolvido através do Electron e de outras tecnologias web (HTML, Javascript e CSS). O ponto-chave do Atom como editor de texto é um compromisso com a hackeabilidade e usabilidade, o que significa que os principais avanços e diferenciais do software se concentram em proporcionar ao usuário uma forma de personalizar por completo sua experiência. Dessa forma, o usuário consegue criar *plug-ins* e extensões conforme sua necessidade e propósito.

Dessa forma, o Atom é composto por seu núcleo, ferramentas e componentes que são adotados como oficiais e vêm instalados (bem como o próprio editor), e pelos *add-ons* desenvolvidos pelos próprios usuários e disponíveis em repositórios na web (em especial no GitHub). A respeito do núcleo do Atom, alguns aspectos são interessantes e passíveis de abordagem.

Duas décadas de desenvolvimento Web permitiram que a mesma evoluísse para uma incrível e poderosa plataforma. Contudo, codificar é uma tarefa especial que requer ferramentas dedicadas. Por isso, o Atom não foi escrito como uma aplicação web tradicional, mas sim como uma variante específica do Chromium, dedicada à escrita de texto.

Abrindo um rápido parênteses, o Chromium é um conjunto de projetos que incluem um navegador web de código aberto e livre que a Google usa como base para o desenvolvimento do Google Chrome, e um sistema operacional onde a Google também se baseia para o Chrome OS. Mais informações podem ser encontradas em [Chromium].

Outro grande benefício é garantir que tudo está rodando na mais nova versão do Chromium. Isso significa que não há preocupações em relação à compatibilidade ou versionamentos. Dessa forma, os mais recentes recursos e frameworks desenvolvidos na Web para uso em aplicações podem ser incluídos no Atom sem que haja problemas de compatibilidade com projetos já em desenvolvimento.

Desenvolver um editor de texto baseado em tecnologias Web é um acerto no sentido em que se tem grande capacidade de crescimento, visto que, embora as tecnolgias nativas variem entre si, as tecnologias Web permanecem por serem multi-plataforma e irrestritas quanto às possiblidades de desenvolvimento.

De acordo com os próprios desenvolvedores, o Atom exerce um papel complementar à função do GitHub de proporcionar software melhor através do trabalho em equipe, o que implica que o editor é, na verdade, um investimento à longo prazo, sempre respaldado pelo suporte do próprio GitHub. Além disso, conforme outros editores como Emacs e Vim demonstraram com o passar dos anos, o desenvolvimento de um software estável, de grande comunidade e eficiente, necessita ter código aberto.

2. Instalação

Disponível nos sitemas operacionais Mac, Windows e Linux, instalar o Atom é realmente simples, em todos eles. O primeiro passo é fazer o *download* da versão compatível com o sistema operacional em https://atom.io.

A seguir serão explorados detalhes particulares da instação do Atom em cada um dos sistemas operacionais em que está disponível.

2.1. Instalação no Windows

Dentre as opções de *download* mencionadas acima, há uma versão com o *Windows Installer*. Após realizar a instalação seguindo as instruções do instalador, será instalado o Atom,bem como seus atalhos na área de trabalho e menu de inicialização, e serão adicionados os comandos `atom` e `apm` na variável do sistema `PATH`, para que também se possa manipular o programa por linha de comando através do terminal (no Windows, `cmd`).

Além disso, as opções `Abrir com Atom` e as associações em `Abrir com...` serão criadas automaticamente. Contudo, isso pode ser controlado através de um painel no próprio Atom. Basta acessar o menu `File > Settings` e escolher `System` no painel lateral. Em ordem, de acordo com a Figura 2.1, as opções: Registra o Atom no menu de associação de arquivos (`Abrir com...`); Mostra a opção `Abrir com Atom` no menu de contexto de arquivos; Mostra a opção `Abrir com Atom` no menu superior do Explorer.

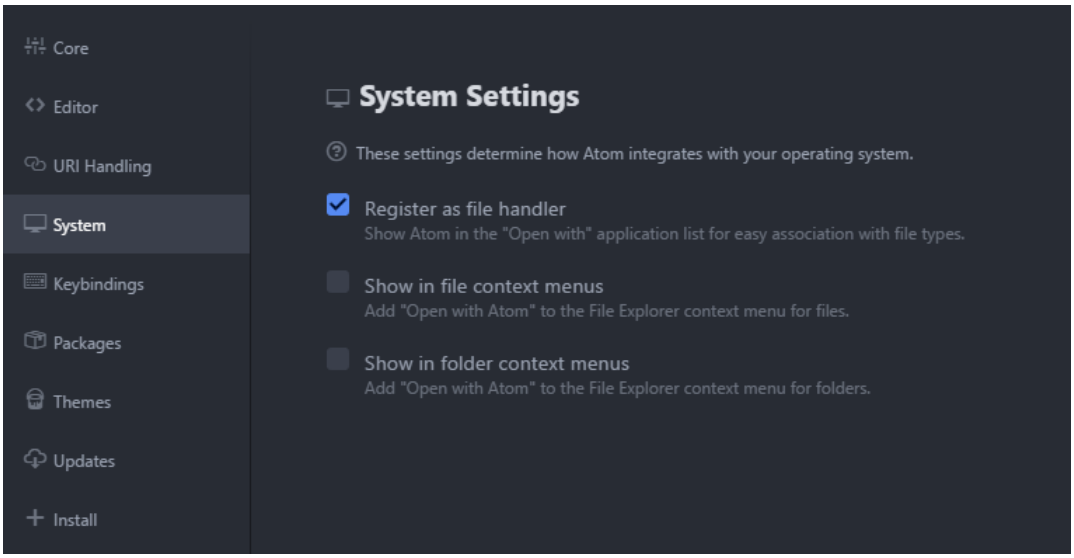

Figure 1. Menu de controle das opções do Sistema no Windows.

2.1.1. Modo Portátil

O Atom possui um Modo Portátil (*Portable Mode*) disponível para ser utilizado.

As configurações e estados do Atom são armazenados em *%userprofile%\.atom* (no Windows, *%userprofile%* por padãro referencia *C:\users\NOME_DE_USUARIO*).

Assim, o usuário pode executar o Atom com modo portátil onde o programa e suas configurações estão armazenados juntos, tal como em um dispositivo de armazenamento removível (*ex: pendrives*).

Para configurar o modo portátil, basta fazer o download da versão correspondente ao sistema operacional em https://github.com/atom/atom/releases/latest e extrair no dispositivo de armazenamento removível. Após isso, crie uma pasta intitulada *.atom* no mesmo diretório que contém a pasta extraída. Por exemplo, se a extração foi realizada em *E:*, então deve-se ter:

```
E:\atom-1.14\atom.exe
E:\.atom
```

Algumas observações em relação ao Modo Portátil são válidas:

1. O diretório *.atom* deve ser gravável (*writable*);
2. Não é necessário criar a pasta *.atom*. Pode-se copiar uma já existente;
3. O Atom pode armazenar configurações de usuário do Electron na pasta *electronUserData*;
4. Tambem é possível criar uma variável de ambiente `ATOM_HOME` para referenciar o diretório *.atom* ou um script (*.sh, .cmd*) para inicializá-lo;
5. A instação em Modo Portátil **não** atualizará automaticamente.

2.1.2. Configurações de Firewall e Proxy

Em caso de problemas com o Firewall ou erros de SSL ao instalar os pacotes, pode-se desativar a opção de rigorosidade do SSL através do prompt de comando do (no Windows: `[Win + R] > cmd > [ENTER]`) com a seguinte linha de comando:

```
apm config set strict-ssl false
```

Caso o problema seja em relação ao Proxy, pode-se configurar um proxy HTTP(S) através da seguinte linha de comando:

```
apm config set https-proxy SEU_ENDERECO_PROXY
```

onde `SEU_ENDERECO_PROXY` é o endereço de proxy a ser utilizado.

2.2. Mac

De forma semelhante à instalação no Windows (vide Seção 2.1), a instação do Atom segue o padrão do sistema operacional. Nesse caso, o padrão zip para as instalações no Mac. O *download* pode ser feito alternativamente em https://github.com/atom/atom/releases/latest através do arquivo *atom-mac.zip*. Após baixar o arquivo, basta extraí-lo e arrastar o `Atom` para a pasta *Aplicações* (ou *Applications*).

Ao abrir a aplicação pela primeira vez, os comandos `atom` e `apm` serão instalados. Em alguns casos, o Atom pode não conseguir realizar esta etapa, visto que pode ser necessária a senha do adminstrador. Para verificar se houove sucesso, basta abrir o terminal de comando do Mac e inserir a linha de comando `which atom`. Em caso de sucesso, será retornado o local de instalação (em geral, `/usr/local/bin/atom`) e, caso contrário, nada será retornado.

Uma nova tentativa de instalação dos comandos pode ser feita na Paleta de Comandos (*Command Palette*) do Atom (explanada na Seção 3.1.1) através do seguinte comando: `Window: Install Shell Commands`.

2.2.1. Modo Portátil

As configurações de instalação e as observações válidas são exatamente como mencionado na Seção 2.1.1. As diferenças, contudo, estão na forma como os arquivos são organizados. No Mac, a hierarquia das pastas para a configuração do modo portátil (*.atom*) seria:

```
/MyUSB/Atom.app
/MyUSB/.atom
```

Além disso, as configurações e estados do Atom **não** são armazenados em *%userprofile%\.atom* pois o Mac não fornece uma variável *%userprofile%*, sendo tais configurações armazenadas na pasta *home* do usuário.

2.2.2. Configurações de Firewall e Proxy

Em caso de problemas com Firewall ou Proxy, basta seguir as instruções da Seção 2.1.2, pois os comandos (e suas sintaxes) são idênticos nos sistemas operacionais disponíveis.

2.3. Linux

No Linux, pode-se utilizar o Gerenciador de Pacotes (*Package Manager*) para configurar um dos repositórios oficiais do Atom, o que também permtirá atualizar o Atom quando possível.

A seguir, serão explanados detalhes para as principais distribuições do Linux onde o Atom está presente. Vale salientar que, para distribuições baseadas nas apresentadas, pode-se seguir as mesmas instruções. **Todas** as instruções a seguir são realizadas em linha de comando do **Terminal** do Linux. Não é utilizado o recurso gráfico dos sistemas operacionais para tal. Além disso, todos os comandos que incluirem `sudo` podem requerer senha de administrador.

2.3.1. Debian e Ubuntu (deb/apt)

Inicialmente, adicionam-se os repositórios oficiais para o Gerenciador de Pacotes das distribuições através das seguintes linhas de comando:

```
$ curl -L https://packagecloud.io/AtomEditor/atom/gpgkey | sudo apt-key add -
$ sudo sh -c 'echo "deb [arch=amd64] \
        https://packagecloud.io/AtomEditor/atom/any/ any main" > \
        /etc/apt/sources.list.d/atom.list'
$ sudo apt-get update
```

Após atualizar os repositórios, instala-se o Atom através do comando `apt-get`:

```
$ sudo apt-get install atom
```

Ou, para instalar o Atom versão Beta:

```
$ sudo apt-get install atom-beta
```

Alternativamente, pode-se instalar o Atom através do pacote *.deb*:

```
$ sudo dpkg -i atom-amd64.deb
$ sudo apt-get -f install
```

onde a segunda linha serve para instalar as dependências do Atom, caso estejam faltando.

2.3.2. Red Hat e CentOS ou Fedora

Aqui ocorrem dois casos diferentes, uma vez que as distribuições Red Hat e CentOS (e suas derivadas) fazem uso do comando `yum` para instalações, em face ao comando `dnf` usado pelo Fedora e suas derivações.

Partindo do mesmo princípio, adicionam-se inicialmente os repositórios oficiais do Atom no Gerenciador de Pacotes:

```
$ sudo rpm --import https://packagecloud.io/AtomEditor/atom/gpgkey
$ sudo sh -c 'echo -e "[Atom]\nname=Atom \
  Editor\nbaseurl=https://packagecloud.io/AtomEditor/atom/el/7/ \
  \$basearch\nenabled=1\ngpgcheck=0\nrepo_gpgcheck=1 \
  \ngpgkey=https://packagecloud.io/AtomEditor/atom/gpgkey" > \
  /etc/yum.repos.d/atom.repo'
```

Neste momento, instala-se o Atom através da linha de comando:

- Red Hat / CentOS

```
$ sudo yum install atom
$ sudo yum install atom-beta
```

- Fedora

```
$ sudo dnf install atom
$ sudo dnf install atom-beta
```

2.3.3. Modo Portátil

As configurações de instalação e as observações válidas são exatamente como mencionado na Seção 2.1.1. As diferenças, contudo, estão na forma como os arquivos são organizados. No Linux, a hierarquia das pastas para a configuração do modo portátil (*.atom*) seria:

```
/media/myusb/atom-1.14/atom
/media/myusb/.atom
```

Além disso, as configurações e estados do Atom **não** são armazenados em *%userprofile%\.atom* pois as distribuições do Linux não fornecem uma variável *%userprofile%*, sendo tais configurações armazenadas na pasta *home* do usuário.

2.3.4. Configurações de Firewall e Proxy

Em caso de problemas com Firewall ou Proxy, basta seguir as instruções da Seção 2.1.2, pois os comandos (e suas sintaxes) são idênticos nos sistemas operacionais disponíveis.

2.4. Construindo do Código-Fonte

Além de baixar o instalador, também é possível compilar diretamente o código-fonte nos sistemas operacionais Mac, Windows, Linux e FreeBSD.

Os detalhes de como fazer isso estão disponíveis na Seção 4.1.

3. Uso do Atom

Aqui serão explorados alguns detalhes a respeito da utilização do Atom. A Seção iniciar-se-á com um manual básico de utilização e, posteriormente, explorará as funções do editor que podem ser utilizadas pelo usuário. Algumas destas funções foram comentadas na Seção 1.

3.1. Básico

Aqui são exploradas as configurações e recursos básicos do Atom.

3.1.1. Paleta de Comandos

A Paleta de Comandos pode ser acessada através do menu superior em `View > Toggle Command Palette` ou pelo atralho de teclas padrão: `Ctrl + Shift + P` no Windows e no Linux; ou `Cmd + Shift + P`, no Mac.

Em resumo, a Paleta de Comandos é um menu guiado por buscas. Nele, pode-se encontrar quaisquer tarefas que são possíveis de serem realizadas no Atom. Serve, principalmente, para evitar a busca exaustiva em todos os menus para encontrar determinada funcionalidade, conforme ilustra a Figura 3.1.1.

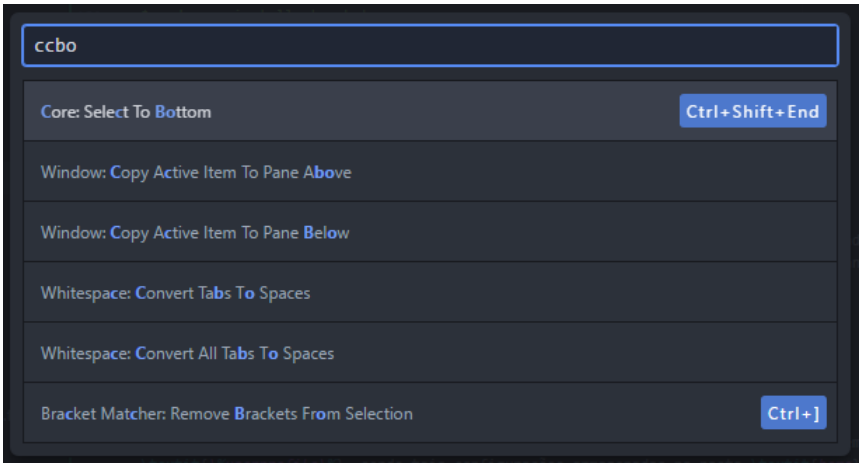

Figure 2. Paleta de Comandos do Atom no Windows.

A busca na Paleta de Comandos é extremamente eficiente, uma vez que faz uso de uma expressão regular baseada unicamente na ordem dos caracteres inseridos. Além de encontrar os comandos, o menu da paleta também exibe os atalhos de tecla associados a eles. Mais da Paleta de Comandos será explorada durante este artigo.

3.1.2. Configurações e Preferências

No Atom, as Configurações e Preferências podem ser modificadas no Painel (costumeiramente referenciado na literatura como *View*) de Configurações. As opções variam entre a troca de tema, quebras de linha, tamanho da tabulação, velocidade do *scroll* e gerenciamento de pacotes (*Packages*, a ser tratado na Seção 3.2).

Cabe salientar que essas informações são mantidas na pasta *.atom*.

O acesso ao Painel de Configurações do Atom pode se dar de diferentes maneiras:

- No menu superior, `File > Settings`
- Na Paleta de Comandos, `settings-view:open`
- Pelo atalho de teclas padrão `Ctrl + ,`

No Painel de Configurações, existem outros paineis anexados, conforme se nota no menu lateral. As opções permitem visualizar categorias específicas das configurações, o que facilita ao usuário encontrar a alteração que deseja realizar.

Neste menu também é possível encontrar a opção para alterar os temas do Atom (vide Seção 3.2). Por padrão, o Atom possui quatro diferentes temas de Interface Gráfica, claras e escuras, bem como oito temas diferentes para sintaxe. Os temas de Interface Gráfica controlam elementos como as abas e menus, enquanto os temas de Sintaxe modificam o destaque de sintaxe (*syntax highlighting*). Além disso, há a possibilidade do usuário baixar um tema pronto do repositório do Atom em https://atom.io, customizar um tema já existente (mais informações em http://flight-manual.atom.io/using-atom/sections/basic-customization/) ou criar seu próprio tema (mais informações em http://flight-manual.atom.io/hacking-atom/sections/creating-a-theme/).

Outro recurso interessante que pode ser encontrado neste Painel é a quebra de linha automática. Isso é particularmente útil ao fazer do Atom um editor de documentos (principalmente em LaTeX). O recurso está disponível neste mesmo Painel de Configurações, permitindo inclusive substituir espaços por tabulações e vice-versa.

3.1.3. Pastas de Projeto

O Atom possui uma estrutura chamada de Árvore de Visualização (*Tree View*), por padrão anexada ao menu lateral esquerdo, onde o usuário pode visualizar toda a hierarquia de diretórios e arquivos que está manipulando, o que permite navegar com facilidade entre os mesmos para que sejam editados com um único clique. Pode-se alternar a exibição da Árvore com `Ctrl + \` e focar através de `Alt + \`. Quando a Árvore possui foco, pode-se pressionar `A` para adicionar um arquivo, `M` para movê-lo e `Delete` para deletá-lo.

Para adicionar uma Pasta de Projeto (*Project Folder*) e, desse modo, constar na Árvore de Visualização, pode-se utilizar o menu superior em `File > Add Project Folder` ou usar o atalho padrão `Ctrl + Shift + A` e navegar até a pasta desejada. Alternativamente, por linha de comando (em todos os Sistemas Operacionais), pode-se utilizar:

```
$ atom <lista-de-diretorios>
```

onde `<lista-de-diretorios>` é a sequência de diretórios a serem concebidos como Pasta de Projeto no Atom, separados por espaço.

O Atom também suporta facilidades em relação à busca de arquivos. Dada uma pasta de projeto que esteja selecionada, o usuário pode pressionar `Ctrl + T` ou `Ctrl + P` e buscar pelo nome do arquivo (funcionalidade chamada de *Fuzzy Finder*) desejado, em uma sintaxe de busca semelhante à explanada na Seção 3.1.1, com a busca pela Paleta de Comandos. Além disso, existem parâmetros para filtrar a busca de acordo com o tipo de arquivo e iniciam com "." após a *string* de busca. A lista destes parâmetros pode ser contrada aqui: http://flight-manual.atom.io/getting-started/sections/atom-basics/.

3.2. Packages

O Atom por si próprio é constituído de um núcleo base composto por um determinado número de pacotes (*packages*) e uma funcionalidade capaz de adicionar outros. O *Fuzzy Finder*, a tela de boas vindas, os temas, a Árvore de Visualização, o Painel de Configurações, são todos pacotes independentes que são mantidos por APIs e servem ao Atom.

Isso implica que existem muitas outras funcionalidades passíveis de serem desenvolvidas por quaisquer usuário e disponibilizadas nos repositórios do GitHub. Para instalar novos pacotes, basta acessar o Painel de Configurações (vide Seção 3.1.2), clicar em *Install* no menu lateral esquerdo, e verificar as opções disponíveis.

Os pacotes listados são disponibilizados em https://atom.io/packages, que é o repositório oficial de pacotes do Atom.

Todos os pacotes instalados podem ser desabilitados ou desinstalados de acordo com a necessidade do usuário. Ao acessar as configurações de um pactoe, pode-se visualizar as opções relacionadas, desde atualização, configurações particulares do pacote, versionamento e código-fonte.

Pode-se também instalar temas a partir do Painel de Configurações, conforme já abordado na Seção 3.1.2. Pode-se, então, pesquisar pelo nome dos temas correspondentes para que então se tenha uma lista de todos os temas disponíveis no repositório oficial.

3.2.1. Linha de Comando

Pode-se instalar pacotes através da linha de comando, utilizando o comando *atm*, instalado junto com o Atom, conforme descrito na Seção 2.

Para realizar a instalação de determinado pacote, basta utilizar a seguinte sintaxe de comando:

```
$ apm install <package_name>
```

para instalar a versão mais recente do pacote ou

```
$ apm install <package_name>@<package_version>
```

para instalar uma versão específica.

Pode-se, inclusive, utilizar o `atm` para encontrar pacotes e, posteriormente, instalá-los, através do comando:

```
$ apm search <termo>
```

Ou, ainda, utilizar

```
$ apm view <package_name>
```

para obter mais detalhes sobre um determinado pacote.

3.3. Navegação no Atom

Além de fazer uso do convencional clique de mouse e setas para se locomover dentro dos arquivos, o Atom fornece uma forma prática através de atalhos do teclado, a fim de manter o usuário exclusivamente no teclado e tornar o trabalho mais rápido.

O Atom suporta todos os atalhos padrão das teclas do Windows, Linux e Mac para locomoção em arquivos com *scroll* (ex: `Ctrl + End` para ir para o fim da página). Esses atalhos universais dentro do sistema operacional permanecem funcionais no Atom.

Pode-se mover para uma específica linha ou coluna do arquivo através do atalho `Ctrl + G` e inserindo as posições de acordo com a sintaxe `linha` ou `linha:coluna`.

O Atom também permite que se navegue pelo arquivo através de símbolos (funções ou definições), de forma intuitíva através do Painel de Símbolos (*Symbols View*), acessível através do atalho `Ctrl + R` (Windows e Linux) ou `Cmd + R` (Mac). Este painel exibe uma lista de todos os símbolos no arquivo atual. A partir deste painel pode-se acessar um filtro de similaridade através de `Ctrl + T` (Windows e Linux) ou `Cmd + T`.

Pode-se também fazer uso de *tags* para localização no arquivo, mas isso requer que se gere as tags através do pacote `ctags`, cujas informações podem ser encontradas em https://ctags.io/.

Além disso, o Atom possui nativamente um pacote denominado `bookmarks`, que permite que o usuário faça uma marcação (denominada *bookmark*) em determinada linha e avance para lá de forma rápida e prática. Em ordem:

- `Alt + Ctrl + F2` adiciona um *bookmark* (ou remove, caso já exista) na linha atual;
  `Cmd + F2`, no Mac.
- `F2` avança para o próximo *bookmark*;
- `Shift + F2` permite navegar ciclicamente entre os *bookmarks*;
- `Ctrl + F2` exibe uma lista com os *bookmarks* atuais, permitindo que usuário se dirija a qualquer um.

3.4. Edições no Atom

Existem alguns recursos fundamentais que fazem do Atom um excelente utilitário para manipular quaisquer tipos de arquivos de texto. Dentre estes recursos, está um conjunto de combinações de tecla que permitem fazer alterações em grande escala de forma rápida e prática.

3.4.1. Básico

Algumas manipulações são básicas e não são recursos exclusivos do Atom como editor de texto e executam operações essenciais:

- `Ctrl + J` (ou `Cmd + J` no Mac): insere a linha abaixo no final da linha atual;
- `Ctrl + Up/Down` (ou `Cmd + Up/Down` no Mac): move a linha atual para cima/baixo;
- `Ctrl + Shift + D` (ou `Cmd + Shift + D` no Mac): duplica a linha atual;
- `Ctrl + U` (ou `Cmd + U` no Mac): transforma a palavra atual apenas em letras maiúsculas *Upper case*;
- `Ctrl + L` (ou `Cmd + L` no Mac): transforma a palavra atual apenas em letras minúsculas *Lower case*;
- `Ctrl + T` (apenas no Mac): transpõe caracteres, trocando os dois caracteres de cada lado do cursor;
- `Ctrl + Shift + K`: exclui a linha atual;
- `Ctrl + Backspace` (ou `Alt + Backspace` no Mac): deleta até o início da palavra atual;
- `Ctrl + Delete` (ou `Alt + Delete` no Mac): deleta até o fim da palavra atual;

3.4.2. Múltilplos Cursores

Um recurso interessante no Atom que, apesar de estar presente em outros editores de texto não é tão bem aproveitado, é a possibilidade de utilizar múltilplos cursores de texto para editar o arquivo. A praticidade oferecida por este recurso torna o Atom flexível para editar quaisquer arquivos de texto.

- `Ctrl + Clique` (ou `Cmd + Clique` no Mac): Adiciona um novo cursor na posição do clique;
- `Ctrl + Alt + Up/Down` ou (`Ctrl + Shift + Up/Down` no Mac): Adiciona um novo cursor na linha acima/abaixo do cursor atual;
- `Ctrl + D` (ou `Cmd + D` no Mac): Seleciona a próxima ocorrência seleção atual no documento, conforme ilustra a Figura 3.4.2;
- `Alt + F3` (ou `Cmd + Ctrl + G` no Mac): Seleciona todas as ocorrências da seleção atual no documento;

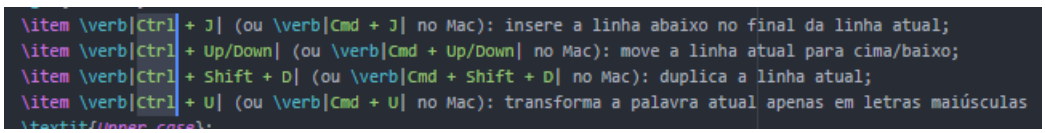

Figure 3. Recurso de múltilplos cursores pala selecionar a palavra "Ctrl".

Esta funcionalidade pode ser extremamente útil para tarefas repetitivas, como renomear variáveis ou alterar o formato de algum texto. É possível também usar a seleção do mouse com a tecla `Ctrl` (ou `Cmd` no Mac) para selecionar outras regiões simultaneamente.

[1]Entende-se por limitadores as maneiras tradicionais de delimitar regiões: `[]`, `()` e `{}`.

3.4.3. Regiões

Atom lida com regiões de maneira inteligente. A aparência dos limitadores[1] é, de fato, dada pelo destque de sintaxe a que está submetido. Contudo, o Atom autocompleta os limitadores, `""`, `''` e <> (caso a sintaxe suporte). Além disso, o Atom oferece alguns atalhos para navegar entre as regiões:

- `Ctrl + M` Avança para o limitador que faz par com o atual;
- `Alt + Ctrl + ,` (`Cmd + Ctrl + M` no Mac) Seleciona todo o texto entre os limitadores que o cursor se encontra;
- `Alt + Ctrl + .` (`Alt + Cmd + .` no Mac) fecha a atual tag XML/HTML;

Essa funcionalidade é implementada pelo pacote de limitador chamado `bracket-matcher`, cuja documentação é encontrada em https://github.com/atom/bracket-matcher.

4. Hackeando o Atom

Aqui existem inúmeros tópicos a respeito de orientar o usuário de forma instrutiva a desevolver seus próprios pacotes, funcionalidades e customizações diversas do Atom. Contudo, estas informações não serão abordadas aqui visto que o escopo do artigo contempla apenas a parte introdutória e de essencial entendimento. Contudo, grande parte das informações aqui omitidas podem ser encontradas em https://flight-manual.atom.io/hacking-atom.

Este trecho pode requerer determinado conhecimento a respeito de tecnologias específicas.

4.1. Hackeando o Núcleo do Atom

Se um usuário por acaso encontrar algum *bug* no Atom ou apenas queira experimentar uma determinada funcionalidade particular do núcleo ou do sistema, é possível executar o Atom em Modo Desenvolvedor (com mais explanações na Seção 4.1.1) com acesso a uma cópia local do código-fonte do Atom.

Basta executar um *fork* (instruções em https://help.github.com/articles/fork-a-repo/) no repositório oficial do Atom para ter acesso local ao código-fonte. Feito isso, basta clonar o repositório que foi ramificado para o repositório local, o que pode ser feito através da linha de comando:

```
$ git clone git@github.com:nomeusuario/atom.git
```

onde `nomeusuario` é o Nome de Usuário do usuário.

Dentro do diretório clonado, pode-se instlar as dependências necessárias através da seguinte linha de comando:

```
$ script/bootstrap
```

válida para todos os Sistemas Operacionais.

4.1.1. Modo Desenvolvedor

Dado um repositório local com o código-fonte do Atom e executado o *script* de *bootstrap* para instalar as dependências, pode-se executar o Atom em Modo Desenvolvedor (dora-

vante chamado Dev Mode). Antes de tudo, deve-se criar uma variável de ambiente do Sistema Operacional denominada `ATOM_DEV_RESOURCE_PATH` que apontará para o diretório clonado. Feito isso, basta, em qualquer sistema operacional, inserir a seguinte linha de comando:

```
$ atom --dev pasta
```

onde `pasta` é um parâmetro opcional para abrir uma Pasta de Projeto.

Existem duas vantagens principais em utilizar o Atom no Modo Desenvolvedor:

1. Se a variável `ATOM_DEV_RESOURCE_PATH` for corretamente criada, o Atom estará sendo executado a partir do código-fonte local e não será mais necessário executar `script/bootstrap` a cada vez que o código do Atom for alterado, bastando reiniciá-lo.
2. Pacotes já existentes na pasta `.atom` são carregados em vez dos que estão em outros locais (ainda que com mesmo nome), o que significa que o usuário pode desenvolver versões dos pacotes localmente e, facilmente, retornar a versões estáveis executando o Atom fora do Dev Mode.

4.1.2. Executando Testes Localmente

Para executar os testes no Núcleo do Atom a partir do terminal, deve-se primeiramente verificar se a variável `ATOM_DEV_RESOURCE_PATH` explanada na Seção 4.1.1 está criada. Feito isso, roda-se a seguinte linha de código na pasta do código-fonte local:

```
$ atom --test spec
```

As instruções a seguir servem para a plataforma Linux. Para Mac e Windows, consulte https://flight-manual.atom.io/hacking-atom/sections/hacking-on-atom-core/.

Antes de sequer iniciar, alguns requisitos devem ser atendidos:

- Deve-se possuir um Sistema Operacional com aquitetura 64 ou 32 bits;
- C++11 *toolchain*;
- Git;
- Node.js 6.0 ou superior;
- Verificar se o *node-gyp* está utilizando *python2* (comando de verificação (pode requerer autorização de administrador): `npm config set python /usr/bin/python2 -g`)
- Bibliotecas do *Libsecret*, disponíveis em https://wiki.gnome.org/Projects/Libsecret.

A seguir serão abordados procedimentos preparatórios em algumas distribuições do Linux e, posteriormente, no parágrafo **Instruções**, serão prestadas orientações gerais para todas as distribuições.

Ubuntu/Debian Requer instação dOS cabeçalhos utilizados pelo GNOME:

```
$ sudo apt-get install build-essential git libsecret-1-dev fakeroot rpm \
  libx11-dev libxkbfile-dev
```

Após isso, basta executar `script/build`.

Em caso de erro, pode ser necessário um compulador alternativo ao C++11:

```
$ sudo add-apt-repository ppa:ubuntu-toolchain-r/test
$ sudo apt-get update
$ sudo apt-get install gcc-5 g++-5
$ sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-5 80 \
  --slave /usr/bin/g++ g++ /usr/bin/g++-5
$ sudo update-alternatives --config gcc # choose gcc-5 from the list
```

Fedora / CentOS / RHEL Basta executar

```
$ sudo yum install -y make gcc gcc-c++ glibc-devel git-core libsecret-devel \
  rpmdevtools
```

Arch Primeiramente executa-se `export PYTHON=/usr/bin/python2` e, então, executa-se:

```
$ sudo pacman -S --needed gconf base-devel git nodejs npm libsecret python2 \
  libx11 libxkbfile
```

Slackware Basta executar

```
$ sbopkg -k -i node -i atom
```

openSUSE Basta executar

```
$ sudo zypper install nodejs nodejs-devel make gcc gcc-c++ glibc-devel git-core \
  libsecret-devel rpmdevtools libX11-devel libxkbfile-devel
```

Instruções Feito isso, basta executar:

```
$ script/build
```

Pode-se utilizar os seguintes parâmetros:

- `--compress-artifacts` : compacta a aplicação em `out/atom-{arch}.tar.gz`
- `--create-debian-package` : cria um pacote *.deb* em `out/atom-{arch}.deb`
- `--create-rpm-package` : cria um pacote *.rpm* em `out/atom-{arch}.rpm`
- `--install[=dir]` : instala a aplicação em `${dir}`, onde por padrão `${dir} = /usr/local`

References

Chromium. Chromium projects. `https://www.chromium.org/`. Acessado em: 27/12/2017.

GitHub. Atom flight manual. `http://flight-manual.atom.io`. Acessado em: 23/12/2017.

MIT, I. d. T. d. M. Licença mit. `https://pt.wikipedia.org/wiki/Licen%C3%A7a_MIT`. Acessado em: 03/01/2017.