

flexbox-boilerplate: All files - gitk

File Edit View Help

master remotes/origin/master Acrescenta um `README.md` com instruções para clonagem.

- Corrige bug do código original.
- Seta `trim\_trailing\_whitespace` e `insert\_final\_newline` para `true`.
- Criação do `.editorconfig`.
- Initial commit.

SHA1 ID: 8226d1229c95ba5d1dbac8276053bc1b04924bfa

Find commit containing: Exact All fields

Search

Diff Old version New version Lines of context: 3 Ignore space change Line diff

Author: Rui Leite <ruipimentelleite@gmail.com> 2020-02-20 21:26:44  
Committer: Rui Leite <ruipimentelleite@gmail.com> 2020-02-20 21:26:44  
Parent: [1d0b87f0d5081a136974e2dd8da0826bbde90b45](#) (Seta `trim\_trailing\_whitespace` e `insert\_final\_newline` para `true`.)  
Child: [8fb038b87198141010b6c638311059b1205c23e3](#) (Acrescenta um `README.md` com instruções para clonagem.)  
Branches: master, remotes/origin/master  
Follows:  
Precedes:

Corrige bug do código original.

index.html

```
index 036f1ce..0c718c6 100644
@@ -6,16 +6,17 @@
<title>Flexbox: Holy Grail</title>
<link rel="stylesheet" href="style.css">
</head>
<body>
<body class="vltic1 flex">
<!--
ORIGINAL SOURCE: "The Holy Grail" by Scott McKee
URL: https://codepen.io/thedigitalman/pen/raHCj
MODIFIED BY: Rui Pimentel Leite
-->
```

Patch Tree

Comments

index.html

styles.css

Rui Leite <ruipimentelleite@gmail.com> 2020-02-26 17:17:22  
Rui Leite <ruipimentelleite@gmail.com> 2020-02-20 21:26:44  
Rui Leite <ruipimentelleite@gmail.com> 2020-02-19 23:05:05  
Rui Leite <ruipimentelleite@gmail.com> 2020-02-19 23:03:00  
Rui Leite <ruipimentelleite@gmail.com> 2020-02-19 23:01:30

# WEB11 (Git) — Aula 1

CURSO DE ESPECIALIZAÇÃO EM DESENVOLVIMENTO  
WEB COM *FRAMEWORKS* MODERNOS

# APRESENTAÇÃO

- Professor:
  - Rui Pimentel Leite
  - Analista de TI na UTFPR
  - Mestre em Engenharia Biomédica com ênfase em Processamento Digital de Sinais pela UTFPR
  - Engenheiro de Computação pela UTFPR

# APRESENTAÇÃO

- Cronograma:
  - Aulas no período da manhã e da tarde
  - Encontros:
    - [09/04/2022] Aulas 1 e 2
    - [23/04/2022] Aulas 3 e 4  
(+ especificação do trabalho)
    - [07/05/2022] Entrega do trabalho e Aula 5

# APRESENTAÇÃO

- Metodologia:
  - Tarefas em sala de aula:
    - [T1] Tarefa 1 (máx.: 30)
    - [T2] Tarefa 2 (máx.: 30)
    - [T3] Tarefa 3 (máx.: 40)
  - [TF] Trabalho Final (máx.: 100)
  - Média:  $(T1 + T2 + T3 + TF) / 2$  (máx.: 100)

# APRESENTAÇÃO

- Git:
  - Ferramenta mais utilizada para versionamento de código-fonte
  - Base do método de trabalho no mercado
  - Parece simples, mas não é
  - Ganhos perceptíveis de produtividade

# APRESENTAÇÃO

- Conteúdo:
  - Introdução ao Git e seus principais conceitos
  - Produzindo código versionado
  - Fluxos de trabalho individual e colaborativo
  - Fluxo de trabalho profissional
  - Produtividade e técnicas avançadas

# INTRODUÇÃO E PRINCIPAIS CONCEITOS

- Relatório
- Relatório (correção)
- Relatório João final
- Relatório v2
- Relatório v2 (corrigido)
- Relatório v3
- Relatório v. final



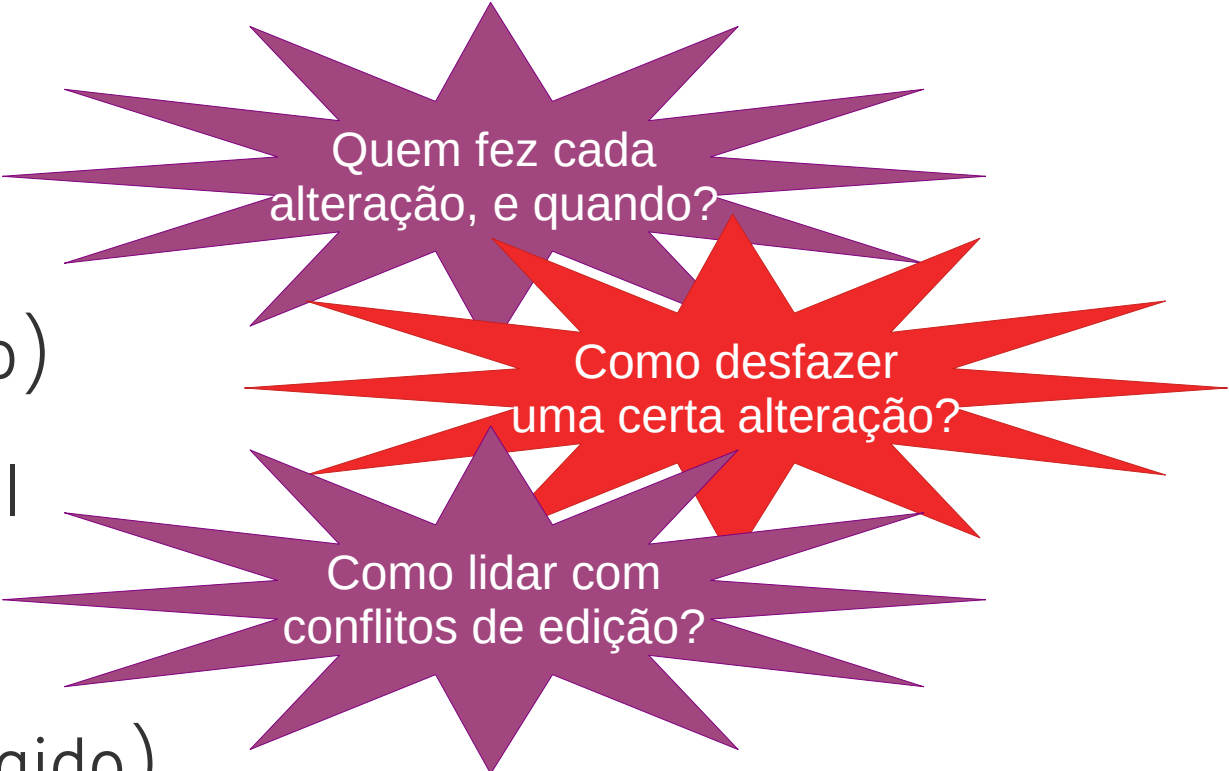
Qual é a versão mais atualizada?

Como compartilhar os arquivos?

O que foi alterado a cada versão?

# INTRODUÇÃO E PRINCIPAIS CONCEITOS

- Relatório
- Relatório (correção)
- Relatório João final
- Relatório v2
- Relatório v2 (corrigido)
- Relatório v3
- Relatório v. final



Quem fez cada alteração, e quando?

Como desfazer uma certa alteração?

Como lidar com conflitos de edição?



# INTRODUÇÃO E PRINCIPAIS CONCEITOS

- Problemas ao lidar com múltiplos arquivos:
  - Arquivos que não podem ser renomeados
  - Apenas alguns são alterados a cada versão
  - Duplicar a pasta do projeto é custoso
- Dificuldades ao lidar com outros colaboradores:
  - Compactar a pasta para envio
  - Conflitos de edição frequentes

# INTRODUÇÃO E PRINCIPAIS CONCEITOS

- Dificuldades no desenvolvimento de software:
  - Qual versão está em produção?
  - Qual versão o homologador está testando?
  - Qual o código mais atualizado?
  - Qual versão introduziu um certo *bug*?



# INTRODUÇÃO E PRINCIPAIS CONCEITOS

- Dificuldades no desenvolvimento de software:
  - Qual a produtividade de um desenvolvedor?
  - Múltiplos projetos em simultâneo
  - Revisão de código
  - Segurança da base de código

# INTRODUÇÃO E PRINCIPAIS CONCEITOS

- *Version Control Systems (VCSs)*:
  - Softwares especializados em controle de versões de documentos
  - **Embutidos** em editores online, processadores de texto, planilhas eletrônicas e CMSs (*Content Management Systems*) como o da Wikipedia
  - Ou disponíveis como softwares **stand-alone**

# INTRODUÇÃO E PRINCIPAIS CONCEITOS

- Versão, revisão e identificador de revisão:
  - Cada estado memorizado do conjunto de documentos constitui uma versão
  - Versões são mais comumente chamadas de revisões
  - É muito comum atribuir à revisão um número, um nome ou alguma outra espécie de identificador



# INTRODUÇÃO E PRINCIPAIS CONCEITOS

- Cada revisão possui:
  - Identificador
  - Autor
  - Data e hora
  - Alterações em relação à revisão anterior
  - Anotações



# INTRODUÇÃO E PRINCIPAIS CONCEITOS

- Revisões podem ser:
  - Visualizadas
  - Comparadas entre si
  - Desfeitas
  - Restauradas
  - Mescladas

# INTRODUÇÃO E PRINCIPAIS CONCEITOS

- Repositório:
  - Conjunto de revisões
  - Geralmente representa um projeto, com um ou mais documentos
    - No caso do Git, o projeto é um diretório
  - Registra o estado inicial do(s) arquivo(s) e todas as modificações subsequentes



# INTRODUÇÃO E PRINCIPAIS CONCEITOS

- O que pode ser feito com um repositório:
  - Recuperar uma revisão (*check out*)
  - Salvar alterações na cópia local (*working copy*)
  - Descartar as alterações locais (fazendo *check out* novamente)
  - Submeter as alterações locais como uma nova revisão (*check in* ou *commit*)



# INTRODUÇÃO E PRINCIPAIS CONCEITOS

- Vantagens de se usar um VCS:
  - Resolve todos os problemas já comentados
  - Elimina a insegurança do desenvolvedor ao alterar o código-fonte
  - Padroniza o método de trabalho da equipe, eliminando perdas por desorganização



# INTRODUÇÃO E PRINCIPAIS CONCEITOS

- Dificuldade de usar um VCS:
  - Quase sempre exige treinamento e ferramentas específicas

# INTRODUÇÃO E PRINCIPAIS CONCEITOS

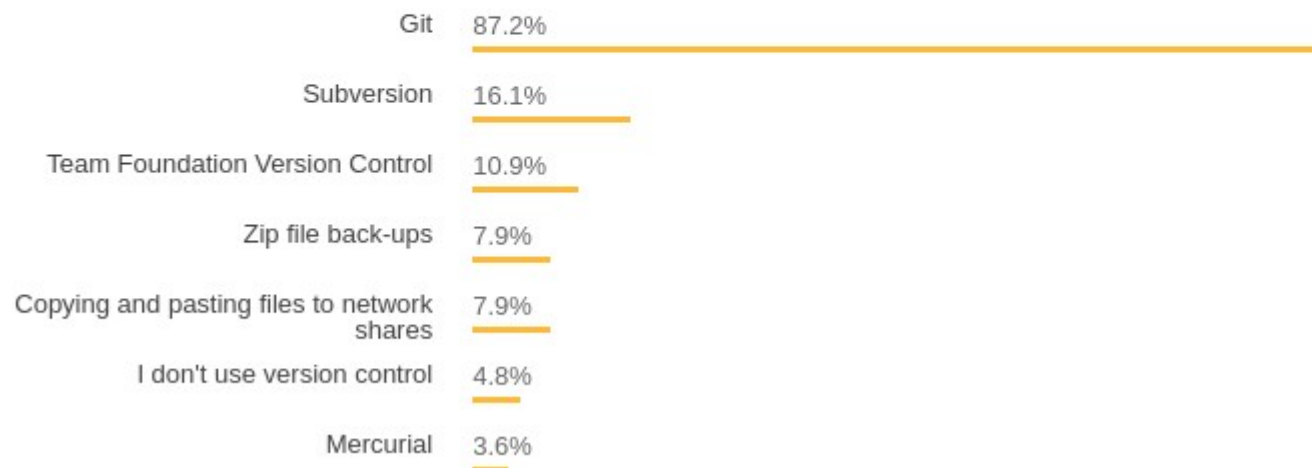
- VCSs:
  - Git: mais utilizado, permite DevOps
  - Subversion (SVN): mais simples, mais performático
    - É possível migrar para o Git
  - Team Foundation Version Control (TFVC):  
centralizado, associado à Azure
  - Mercurial: em desuso

# INTRODUÇÃO E PRINCIPAIS CONCEITOS

## Version Control

All Respondents

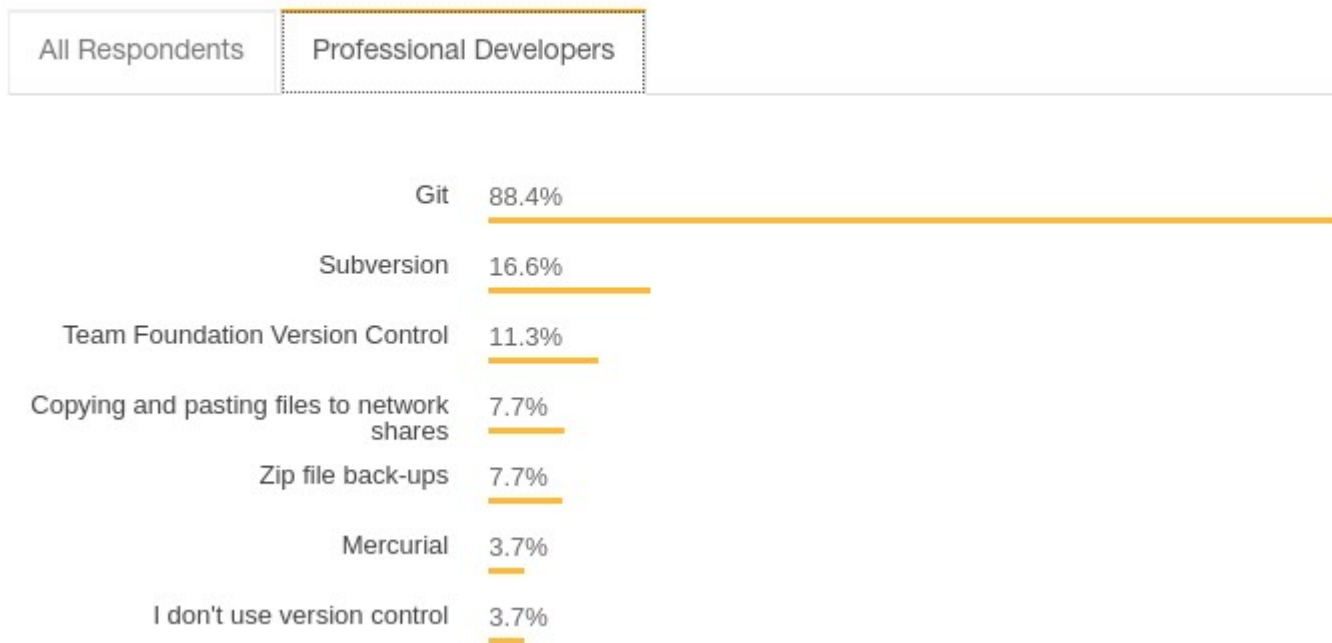
Professional Developers



[https://insights.stackoverflow.com/survey/2018/#work-\\_-version-control](https://insights.stackoverflow.com/survey/2018/#work-_-version-control)

# INTRODUÇÃO E PRINCIPAIS CONCEITOS

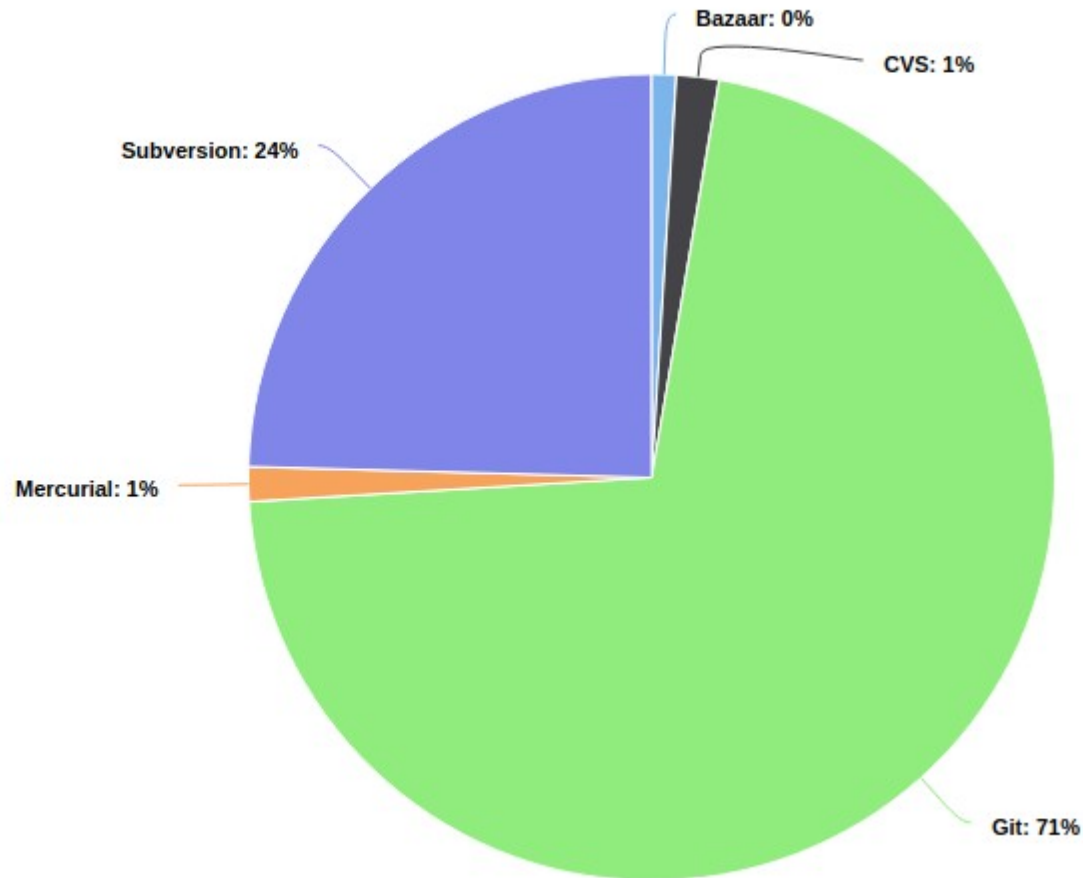
## Version Control



[https://insights.stackoverflow.com/survey/2018/#work-\\_-version-control](https://insights.stackoverflow.com/survey/2018/#work-_-version-control)

# INTRODUÇÃO E PRINCIPAIS CONCEITOS

Compare Repositories



<https://www.openhub.net/repositories/compare>

# INTRODUÇÃO E PRINCIPAIS CONCEITOS

- Git:
  - Baseado em um modelo de desenvolvimento de código-fonte distribuído e não-linear
  - Extremamente popular entre programadores
  - Pode ser usado para versionar qualquer conjunto de arquivos, sendo especialmente efetivo para arquivos de texto plano



# INTRODUÇÃO E PRINCIPAIS CONCEITOS

- Binários:
  - Executáveis binários
  - Vídeos / maioria das imagens
  - ZIP / DOC
- Texto plano (*plain text*):
  - TXT
  - HTML / CSS / JS

# INTRODUÇÃO E PRINCIPAIS CONCEITOS

- Git:
  - Livre e de código aberto
  - Motivado pelo fim da licença livre do BitKeeper, o que também deu origem ao Mercurial
  - Criado pelo próprio Linus Torvalds em 2005 para apoiar o desenvolvimento do *kernel* do Linux
  - Utilizado em inúmeros projetos *open source*

# INTRODUÇÃO E PRINCIPAIS CONCEITOS

- GitHub:
  - Gerenciador de repositórios Git baseado na Web
  - *“Git server as a service”*
  - Oferece as funcionalidades de gerenciamento de código-fonte do Git, mais recursos próprios



# INTRODUÇÃO E PRINCIPAIS CONCEITOS

- GitHub:
  - Hospeda diversos projetos, inclusive o [kernel do Linux](#) e o [próprio Git](#)
  - Oferece planos gratuitos e pagos

# INTRODUÇÃO E PRINCIPAIS CONCEITOS

- Preparação da máquina virtual do curso:
  - Copiar
  - Adicionar ao VirtualBox
  - Restaurar *snapshot* inicial

# INTRODUÇÃO E PRINCIPAIS CONCEITOS

- Instalação do Git:
  - <https://git-scm.com/downloads>
  - `sudo add-apt-repository ppa:git-core/ppa`
  - `sudo apt update`
  - `sudo apt install git`

# INTRODUÇÃO E PRINCIPAIS CONCEITOS

- Autocompletar no Linux (`bash-completion`):
  - `sudo apt install git-core bash-completion`
  - Para surtir efeito, é preciso reiniciar o Terminal

# INTRODUÇÃO E PRINCIPAIS CONCEITOS

- Instalação do Git no Windows:
  - *Checkout Windows-style, commit Unix-style*  
`git config --global core.autocrlf true`
  - *Checkout as-is, commit Unix-style*  
`git config --global core.autocrlf input`
  - *Checkout as-is, commit as-is*  
`git config --global core.autocrlf false`



# INTRODUÇÃO E PRINCIPAIS CONCEITOS

- Configuração inicial das variáveis:
  - `git config --global user.name "Nome do desenvolvedor"`
  - `git config --global user.email "desenvolvedor@empresa"`

# INTRODUÇÃO E PRINCIPAIS CONCEITOS

- Instalação do `git-gui`:
  - Utilitário gráfico instalado separadamente
  - `sudo apt install git-gui gitk`



# INTRODUÇÃO E PRINCIPAIS CONCEITOS

- Explorando o primeiro repositório:
  - Acessar  
<https://github.com/ruipimentel/flexbox-boilerplate>

# INTRODUÇÃO E PRINCIPAIS CONCEITOS

- Explorando o primeiro repositório:
  - Clonar o repositório a partir do GitHub:
    - *“Clone or download”*
    - `ls -l`
    - `git clone https://github.com/ruip... .git`
  - Abrir o arquivo HTML no navegador:
    - `firefox flexbox-boilerplate/index.html`

# INTRODUÇÃO E PRINCIPAIS CONCEITOS

- [alternativa] Clonar o repositório para um diretório específico:
  - `git clone https://github.com/r....git fb`
  - Abrir o arquivo HTML no navegador:
    - `firefox fb/index.html`

# INTRODUÇÃO E PRINCIPAIS CONCEITOS

- Neste momento, é muito comum ter que executar comandos de instalação de dependências, compilação do *software* etc.
- Normalmente, as instruções se encontram na página do repositório no GitHub/GitLab/BitBucket, na página Web do autor, ou em algum arquivo dentro do próprio repositório (ex.: **README.md**)

# INTRODUÇÃO E PRINCIPAIS CONCEITOS

- Explorando o primeiro repositório:
  - Visualizar arquivos do projeto:
    - `cd flexbox-boilerplate/`
    - `ls -la`
  - Abrir o arquivo HTML no navegador:
    - `firefox fb/index.html`

# INTRODUÇÃO E PRINCIPAIS CONCEITOS

- Visualizando o histórico do repositório:
  - `git log`
    - Navegação igual à do `less` do Linux
    - Tecla `q` para sair



# INTRODUÇÃO E PRINCIPAIS CONCEITOS

- Informações que compõem um *commit*:
  - *Hash* (SHA-1)
  - Autor (nome e e-mail)
  - Data
  - Mensagem de *commit*

# INTRODUÇÃO E PRINCIPAIS CONCEITOS

- Visualizando o histórico do repositório:
  - `git show {{hash}}`
    - Exibe detalhes e *diff* da versão especificada
  - `git log -p`
    - Exibe o histórico já com o *diff*
  - `git whatchanged`
    - Exibe um resumo das alterações introduzidas em cada versão

# INTRODUÇÃO E PRINCIPAIS CONCEITOS

- Informações que compõem um *commit*:
  - *Hash* (SHA-1)
  - Autor (nome e e-mail)
  - Data
  - Mensagem de *commit*
  - Diferenças (*diff/patch*) nos arquivos, inclusive modo de acesso (ex.: 644 = -rw-r--r--)

# INTRODUÇÃO E PRINCIPAIS CONCEITOS

- Recuperando uma versão para o sistema de arquivos (HD):
  - `git checkout {{hash}}`

# INTRODUÇÃO E PRINCIPAIS CONCEITOS

- Recapitulando:
  - `git clone` traz uma cópia de um repositório
  - `git log` exhibe um histórico de *commits*
  - Cada *commit* introduz diferenças em relação à versão anterior
  - `git checkout` aplica uma versão

# INTRODUÇÃO E PRINCIPAIS CONCEITOS

- Um repositório Git consiste de uma “árvore” de *commits* (mais especificamente, um grafo direcionado acíclico)
- `git log --graph --oneline --all`

```
rui@gotham:~/Documents/Projetos/CEFWM/temp/flexbox-boilerplate$ git log --graph --oneline --all
* 8fb038b (HEAD -> master, origin/master, origin/HEAD) Acrescenta um `README.md` com instruções para clonagem.
* 8226d12 Corrige bug do código original.
* 1d0b87f Seta `trim_trailing_whitespace` e `insert_final_newline` para `true`.
* 7d9f31b Criação do `.editorconfig`.
* 27a8dd8 Initial commit.
rui@gotham:~/Documents/Projetos/CEFWM/temp/flexbox-boilerplate$
```

# INTRODUÇÃO E PRINCIPAIS CONCEITOS

- O comando `gitk` abre uma interface gráfica com a mesma finalidade do `git log`
  - Frequentemente, é útil chamar essa aplicação com `gitk &`, que permite continuar utilizando o Terminal para outros comandos

# INTRODUÇÃO E PRINCIPAIS CONCEITOS

The screenshot shows the GitKraken application window titled 'flexbox-boilerplate: All files - gitk'. The interface includes a menu bar (File, Edit, View, Help), a commit log on the left, a commit details pane in the center, and a diff view on the right.

**Commit Log:**

- **master** — **remotes/origin/master** Acrescenta um `README.md` com instruções para clonagem. Rui Leite <ruipimentelleite@gmail.com> 2020-02-26 17:17:22
- Corrige bug do código original. Rui Leite <ruipimentelleite@gmail.com> 2020-02-20 21:26:44
- Seta `trim\_trailing\_whitespace` e `insert\_final\_newline` para `true`. Rui Leite <ruipimentelleite@gmail.com> 2020-02-19 23:05:05
- Criação do `.editorconfig`. Rui Leite <ruipimentelleite@gmail.com> 2020-02-19 23:03:00
- Initial commit. Rui Leite <ruipimentelleite@gmail.com> 2020-02-19 23:01:30

**Commit Details:**

**SHA1 ID:** 8226d1229c95ba5d1dbac8276053bc1b04924bfa

**Find:** commit containing: [Search]

**Diff View:**

Author: Rui Leite <ruipimentelleite@gmail.com> 2020-02-20 21:26:44  
Committer: Rui Leite <ruipimentelleite@gmail.com> 2020-02-20 21:26:44  
Parent: 1d0b87f0d5081a136974e2dd8da0826bbde90b45 (Seta `trim\_trailing\_whitespace` e `insert\_final\_newline` para `true`.)  
Child: 8fb038b87198141010b6c638311059b1205c23e3 (Acrescenta um `README.md` com instruções para clonagem.)  
Branches: master, remotes/origin/master  
Follows:  
Precedes:

Corrige bug do código original.

**Diff:**

```
----- index.html -----
index 036f1ce..0c718c6 100644
@@ -6,16 +6,17 @@
     <title>Flexbox: Holy Grail</title>
     <link rel="stylesheet" href="styles.css">
   </head>
-  <body>
+  <body class="vertical-flexbox">
     <!--
       ORIGINAL SOURCE: "The Holy Grail" by Scott McKee
       URL: https://codepen.io/thedigitalman/pen/RAHCj
+     MODIFIED BY: Rui Pimentel Leite
     -->
```

**Tree View:**

- index.html
- styles.css



# INTRODUÇÃO E PRINCIPAIS CONCEITOS

- Tarefas:
  - Visualizar o código do repositório no *commit* inicial
  - Determinar em qual versão o *bug* da barra de rolagem foi solucionado, e qual a data desta versão
  - Determinar em qual versão o arquivo **README.md** foi criado, e qual a data dessa versão

# INTRODUÇÃO E PRINCIPAIS CONCEITOS

- Já somos capazes de:
  - Obter um repositório pronto da Web
  - Executar diferentes versões do software
  - Observar as diferenças entre uma versão e outra
- Mas como criar um repositório?
- Como nasce um *commit*?