Apontamentos de Laboratório de Aplicações 2018-2019

David Sousa-Rodrigues[1]

[1] <david.s.rodrigues@ipleiria.pt>
<prof.david.esad@gmail.com>

Conteúdo

Apontamentos de Laboratório de Aplicações

Estes apontamentos seguem aproximadamente os tópicos do programa da cadeira de Laboratórios de Aplicações, não sendo exaustivos na cobertura de todo o programa. Esta primeira versão encontra-se bastante incompleta e pretende apenas dar um pequeno enquadramento ao aluno. Os ficheiros fonte destes apontamentos encontram-se online no repositório https://github.com/sixhat/LabApps sendo que serão continuamente melhorados. O repositório terá assim sempre a versão mais actual deste documento.

Introdução

O Laboratório de Aplicações visa dotar o aluno de competências básicas nas ferramentas de gestão, produção e desenvolvimento colaborativo para aplicações de media digital. Exemplos destas aplicações incluem websites, jornais online, aplicações de telemóvel, e ou tablet, aplicações desktop, lojas de comércio eletrónico, etc.

No final do curso o aluno será capaz de trabalhar com as ferramentas utilizadas pelos profissionais da indústria e compreender o seu papel nas diversas etapas de desenvolvimento de um produto digital.

Como ler este documento

Este documento não é um manual exaustivo de toda a matéria que o aluno deve estudar e não deve ser entendido como a única fonte de informação sobre os elementos do programa da cadeira. Serve principalmente para ajudar o aluno a entrar nos tópicos que serão

lecionados. Muitas vezes as instruções iniciais que permitem ao aluno explorar um tópico são as mais difíceis de memorizar. Esta sebenta pretende ser uma ajuda nessa fase inicial de aprendizagem de uma matéria nova, quando tudo parece estranho e mais difícil.

A formatação deste documento é simples e compartimentada em capítulos que poderão ser consultados independentemente dos restantes. Por isso cada capítulo inclui no final um conjunto de referências sobre o tópico em questão. Para além disso o aluno vai encontrar blocos de código (html, css e javascript) ou comandos de terminal (assinalados com um `$` no princípio da linha). Estes blocos estão assinalados com uma fonte `mono-espaçada` e são assinalados com uma linha na margem esquerda. Por exemplo, o seguinte comando de terminal `echo` vai imprimir a linha `isto é um comando no terminal`.

```
$ echo 'isto é um comando no terminal'
isto é um comando no terminal
```

Pode enviar as alterações para <prof.david.esad@gmail.com> ou através de um Pull Request no Github em https://github.com/sixhat/LabApps

Esta é uma versão preliminar do documento pelo que é natural que possa conter pequenas gralhas e incorreções, apesar de todo o esforço para as evitar. Se as encontrar, por favor comunique-mas para melhorar as próximas versões deste documento.

Ciclo de Vida de um Projeto

Tipicamente um projeto dentro de uma empresa passa por diversas etapas.

- Concepção
- Design
- Realização
- Serviço

Plataformas de Trabalho Colaborativo

- Modelos de gestão de projeto
  - Waterfall
  - Scrum
  - Agile
  - XP
  - Kanban
- Modelação dos processos
  - BPMN
  - UML2
  - Fluxograma
- Ferramentas de trabalho colaborativo
  - Trello
  - Slack

– Github.

Sistemas de Controlo do Versões

Se pequenos projectos podem podem ser executados por um designer ou programador individual, a maioria dos projetos requerem a participação de diversos membros da equipa. Ainda por cima os membros da equipa podem não se encontrar fisicamente no mesmo local. Por esta razão o desenvolvimento de um projecto requer a utilização de mecanismo de coordenação e controlo das diversas versões que vão sendo desenvolvidas pelos diversos utilizadores.

Estes sistemas de coordenação e controlo são conhecidos em inglês pela sigla VCS (de *Version Control Systems*). Existem diversos sistemas em uso na atualidade mas talvez o mais popular seja o git.

O Git é um sistema de controlo de versões distribuído usado principalmente no desenvolvimento de software, mas pode ser utilizado para fazer o registo histórico de qualquer tipo de arquivo. O Git teve o seu desenvolvimento inicial por Linus Torvalds para ser utilizado no desenvolvimento do kernel do sistema operativo linux, mas sendo uma ferramenta *open-source*, tornou-se rapidamente popular como ferramenta de controlo de versões de muitos projetos.

Funcionamento do Git

O Git funciona através do armazenamento de um "snapshot" do projeto a cada versão criada — a cada **commit**, na linguagem do Git. Neste "snapshot" todos os ficheiros que foram modificados são adicionados ao sistema de versionamento e todos os que não foram modificados são mantidos e apenas uma referência a estes é atualizada no *snapshot*. Desta forma o `git` funciona como um sistema de ficheiros sempre atualizados e que permite compreender o histórico de alterações através da sequência de *snapshots* que foram tiradas ao longo do tempo.

O Git p ode ser instalado a partir do website https://git-scm.com/ . No momento da escrita desta sebenta a versão mais actual é a **2.21.0**. O Git é uma ferramenta de baixo nível, que permite executar inúmeras tarefas de criação, clonagem e coordenação dos nossos repositórios. Este baixo nível obriga a utilizar a ferramenta a partir da linha de comandos.

Para facilitar a utilização do **git**, várias interfaces gráficas foram criadas, sendo que a mais popular é talvez a GitHub Desktop (https://desktop.github.com/). Esta interface gráfica foi desenvolvida principalmente para gerir os repositórios do site **github.com** mas é uma ferramenta excelente para quem quer começar a utilizar o Git sem ter que utilizar a linha de comandos.

Nesta cadeira vamos utilizar o GitHub Desktop, mas para uma utilização mais avançada recomenda-se a utilização da linha de comandos.

Fluxo de trabalho típico com o git

Imaginando que vai começar a trabalhar num projecto existente vamos descrever a sequência de passos normalmente utilizadas para colaborar com a equipa utilizando o Git. No exemplo seguinte vamos utilizar o repositório https://github.com/sixhat/utils.dave.p5.js como exemplo.

O primeiro passo consiste em fazer uma cópia do repositório do projeto central da equipa. Para tal utilizamos o comando **clone**:

```
$ git clone https://github.com/sixhat/utils.dave.p5.js
```

Este comando vai descarregar uma cópia completa do repositório para que possa trabalhar localmente. Vai criar uma pasta **utils.dave.p5.js** com todos os ficheiros necessários. O passo seguinte é editar e fazer as alterações que são necessárias.

Imaginando que criaram um ficheiro inexistente no repositório chamado **readme.txt**. Uma forma de saber se o repositório sofreu alterações é utilizar o comando **status.**

```
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

    readme.txt

nothing added to commit but untracked files present (use "git add" to track)
```

Como se pode ver o ficheiro `readme.txt` está numa secção de ficheiros **untracked**. Isto significa que o ficheiro não está a ser arquivado e versionado. O primeiro passo é adicioná-lo para que passe a ser versionado.

```
$ git add readme.txt
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
```

```
    new file:   readme.txt
```

Agora o sistema vai registar as alterações ao ficheiro `readme.txt`. O próximo passo é fazer um **commit** das alterações para se criar um *snapshot* local.

Sempre que se procedem a alterações deve-se fazer **commit** das alterações. O passo de **commit** serve para registar as alterações num **log** (um livro de registos).

```
$ git commit -m "criei ficheiro com instruções para o user"
[master (root-commit) 58afec0] criei ficheiro com instruções para o user
 1 file changed, 5 insertions(+)
 create mode 100644 readme.txt
```

Note que o processo vai dar informação sobre o n.º de alterações efectuadas pelo Git no repositório. Neste momento o nosso ficheiro `readme.txt` está registado e será incluído nos *snapshots* subsequentes. Uma forma de ver o histórico de *snapshots* é utilizando o comando **log**.

```
$ git log
commit 58afec0a65c8badc4b6c7d6575cf4314ad400f83 (HEAD -> master)
Author: David.Rodrigues <david@sixhat.net>
Date:   Sat Dec 15 20:51:50 2018 +0000

    criei ficheiro com instruções para o user
```

Agora que as suas contribuições para o projecto foram registadas no repositório local é altura de as enviar para o repositório central que ainda desconhece as alterações que efectuou localmente — uma das vantagens do git é a possibilidade de trabalhar offline num projecto e só quando se voltar a estar online aí então enviar as nossas contribuições para o repositório central da equipa.

```
$ git push origin master
```

Esta operação vai enviar as alterações que efectuou para o repositório central.

O fluxo de trabalho mostrado acima é um processo muito simples que poderá funcionar para projetos pequenos com poucos contribuidores. Quando os projetos aumentam de tamanho ou o n.º de contribuidores é maior o Git inclui funcionalidades extra que permitem uma colaboração mais eficiente, nomeadamente através da criação de *branches*.

Referências

- Documentação oficial do Git - https://git-scm.com/doc

- Pro Git - everything you need to know about git - Scott Chacon e Ben Straub, Apress. (gratuíto online em https://git-scm.com/book/en/v2).