Documentos: Permite alojar documentos gerais do projeto, não relacionados com uma versão de desenvolvimento ou considerados estáticos. Uma notícia para a comunicação social é um exemplo de um documento para esta área.

Ficheiros: Permite alojar ficheiros do projeto em que se espera que existam versões diferentes destes ficheiros. Um exemplo será a publicação de versões diferentes de um programa que esteja a ser desenvolvido.

Repositórios: Permite alojar conteúdos de trabalho do projeto geridos por um sistema de controlo de versões

Resumindo

Docs e Files são para mais próprios para upload.

O módulo de Repositórios merece uma atenção particular porque permite associar ao projeto um repositório de ficheiros gerido por um sistema de *controlo de versões*. Estes sistemas são muito úteis, especialmente em projetos de software, porque mantêm um registo histórico da evolução do projeto.

O CodeUA suporta dois sistemas alternativos de controlo de versões:

* Apache Subversion (SVN);
* Git.

À extração de uma versão do repositório para uma árvore de trabalho dá-se o nome de check out. Já ao processo inverso, o de criar uma nova versão no repositório a partir de uma árvore de trabalho, dá-se o nome de check in ou commit.

A partir de uma mesma versão no repositório podem criar-se linhas de evolução diferentes, às quais se dá o nome de branches (ramos). Dessa forma, a evolução das versões do software faz-se através de sucessivos commits no mesmo ramo e de ramificações a partir de algumas versões commited.

Working tree (árvore de trabalho) - A árvore de trabalho é um diretório no sistema de ficheiros ao qual se encontra associado um repositório (tipicamente existe nesta diretoria um subdiretório chamado .git). A árvore de trabalho inclui todos os ficheiros e subdiretórios nela existentes. No fundo é onde o programador desenvolve o seu código e tem os seus ficheiros. Cada programador pode ter uma ou mais árvores de trabalho associadas a um mesmo repositório.

Repository (repositório) - Um repositório é uma coleção de commits, sendo que cada um destes é um registo do estado em que se encontrava uma dada árvore de trabalho numa data passada. De uma forma simplista pode-se considerar que um commit é uma alteração ao código ou versões. Os commits de um repositório podem ainda ser identificados como ramos, caso sejam o início de um ramo, ou possuir etiquetas (tags) de forma a serem facilmente identificados por um nome.

Check out (extração) - Quando se faz uma extração de uma versão (ou de um commit) de um repositório cria-se uma árvore de trabalho com todos os ficheiros e diretorias pertencentes a essa versão. O processo de extração regista também na árvore de trabalho o identificador do ramo ou commit do qual a árvore de trabalho atual descende. Esse identificador é genericamente designado por HEAD.

Commit (ou check in) - Um commit é uma cópia de uma árvore de trabalho realizada num dado momento. Para além disso, um commit é igualmente uma evolução de algo que existia anteriormente, que é o commit a partir do qual a árvore de trabalho foi criada (indicada por HEAD). O commit anterior torna-se pai do commit atual. É esta relação entre commits que dá por sua vez origem à noção de histórico de revisões (revision history).

Branch (ramo) - Um ramo é uma sequência de commits sucessivos que formam uma linha de desenvolvimento independente. No Git os ramos podem ser referenciados por um nome que funciona como sinónimo do último commit nesse ramo. A linha principal de desenvolvimento na maioria dos repositórios é feita num ramo chamado master. Embora seja um nome definido por omissão, não é de qualquer forma especial.

A função do repositório resume-se a manter cópias congeladas dos conteúdos de um diretório tiradas em diversos momentos ao longo do tempo. A estas cópias chamam-se snapshots (instantâneos).

Internamente, o Git partilha uma estrutura em tudo similar, embora tenha uma ou outra diferença. Em primeiro, representa os conteúdos de um ficheiro em **blobs**, que também são muito semelhantes a um diretório. O nome de um blob, também chamado o seu hash id, é um número calculado pelo algoritmo Secure Hashing Algorithm (versão 1) (SHA-1) aplicado sobre o tamanho e conteúdos do próprio blob. O hash id parece um número arbitrário, mas tem duas propriedades interessantes:

* Primeiro, certifica que o conteúdo do blob não foi alterado;
* Segundo, garante que blobs de conteúdo igual têm o mesmo nome, independentemente de onde apareçam.