U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Modelação UML

Projeto de Base de Dados

20/21

Beatriz Aguiar up201906230
Margarida Vieira up201907907
Miguel Rodrigues up201906042

Índice

Descrição

O que é o GitHub?

GitHub é uma plataforma de hospedagem de código-fonte e arquivos que usa um sistema de controlo de versões, designadamente o *Git*. É comummente usado, sobretudo, por desenvolvedores de software para hospedar os seus projetos *open-source*.

GitHub no Modelo Conceptual

Utilizadores, Organizações e Equipas

Para usufruírem da plataforma, todos os **utilizadores** se devem registar com a criação de um respetivo nome de utilizador. Estes, por sua vez, podem fazer parte de **organizações** e, no âmbito destas, de **equipas**. Logicamente, a existência das duas últimas, implica que a elas esteja associado, pelo menos, um utilizador. No caso das organizações, este utilizador necessário é o *owner*.

Repositórios

Para cumprir com o propósito da hospedagem de informação, surgem os **repositórios**, que têm um nome identificador e podem, quanto à visibilidade, ser privados ou públicos. Um repositório pode estar associado quer a um utilizador enquanto entidade independente, quer a uma organização ou a uma equipa desta, tornando-se essencial conhecer quais as características desta associação.

Quando um utilizador procede à criação de um novo repositório, torna-se automaticamente o seu *owner* e cabe a este gerir quais os utilizadores aos quais pretende atribuir permissão para alterar o conteúdo do mesmo. Assim que o faz, estes tornam-se *contributors* do repositório em causa. No âmbito de uma organização, existem *owners* e *members*, sendo os primeiros um *subset* dos segundos, diferindo apenas em questões administrativas. Quanto às equipas, os membros podem ter associados a si o *role maintainer*, que lhes concede permissões extra (e.g. adicionar membros à equipa).

Contribuições - *Commits*, *Issues* e *Pull Requests*

A interação de um utilizador, desde que este tenha permissão para, com um repositório, faz-se por meio de uma **contribuição**. Uma contribuição ocorre numa data e pode ser do tipo *pull request*, *commit* ou *issue*.

O ***commit*** pode ser encarado como uma contribuição mais elementar, onde simplesmente um pedaço de código é adicionado ou removido do repositório.

Um ***issue***, como o próprio nome indica, é um problema com o estado atual do repositório, poderá ser uma falha no código ou mesmo uma sugestão para melhorar algum aspeto menos conseguido. Para cada instância de *issue* existe um identificador único atribuído de acordo com a ordem de instanciação.

Por último, a terceira forma de contribuição são os ***pull requests***. Estes também possuem, à semelhança dos *issues*, um identificador único de acordo com a sua ordem de instanciação, o seu

estado atual (e.g. aberto, fechado, etc.) e ainda estão associados a um *merge commit*, o que significa que, por causa desta associação, a generalização de contribuição será completa e sobreposta.

Commits – Tags, *Merge Commits* e *Branches*

Associada ao *commit* temos a classe ***tag*** que, tal como o nome indica, representa uma etiqueta para um determinado *commit*. Deste modo é mais fácil ao utilizador voltar a uma determinada versão do seu código que necessita de ser revista ou alterada, uma vez que a *tag* permite identificar o *commit* de uma forma mais humana.

Ainda relativamente aos *commits*, certas instanciações serão especiais, daí surgir uma generalização para os ***merge commits***. Como este tipo de *commits* ocorre de forma esporádica, a generalização em causa é incompleta e exclusiva. A partir da instanciação deste tipo de *commits* interessa saber quais os **branches** envolvidos nesse *merge.* Portanto, dadas estas circunstâncias optamos por criar uma nova relação - Branch - e as respetivas associações - *ours* e *theirs* – que, à semelhança da sintaxe do próprio *Git*, representam os branches envolvidos.

Diretórios, ficheiros e linguagens de programação

Cada repositório, e partindo do princípio que não foi criado com o propósito de continuar vazio, é constituído por **diretórios**, comummente designados por pastas, e **ficheiros**. Cada diretório tem um nome e pode conter mais subdiretórios e/ou ficheiros. De cada ficheiro, para além do seu nome, interessa saber o seu conteúdo e, no caso de não se tratar de um simples ficheiro de texto, a **linguagem de programação** que lhe está associada. Esta última caracteriza-se pelo seu nome. Por último, um repositório pode ainda conter *submodules* que, visualmente, se assemelham a subdiretórios, mas que são, na verdade, ligações/referências para outros repositórios e que, nesse sentido, serão tratados, no contexto desta base de dados, como meros subrepositórios.

Restrições em detalhe[1]

Tal como é visível no diagrama abaixo, impõem-se algumas restrições ao modelo conceptual supra descrito.

Relativamente ao nome dos repositórios, este tem que ser único no contexto de cada utilizador, ou seja, o mesmo utilizador não pode ser *owner* de dois repositórios com o mesmo nome e, obviamente, este não pode corresponder a uma *string* vazia, o que se verifica, aliás, para qualquer variável *name* que surja no diagrama.

No que diz respeito à associação entre utilizador e contribuição, esta só pode ocorrer se se verificar que o utilizador é contribuidor do repositório no qual a contribuição está a incidir, ou membro da (equipa da) organização à qual o repositório pertence.

Relativamente ao identificador único, quer de um *pull request*, quer de um *issue*, este é obrigatoriamente um número inteiro maior que zero.

[1] Algumas das restrições impostas e até certos aspetos da implementação, poderão, no futuro, ser alvo de alterações.

Diagrama UML

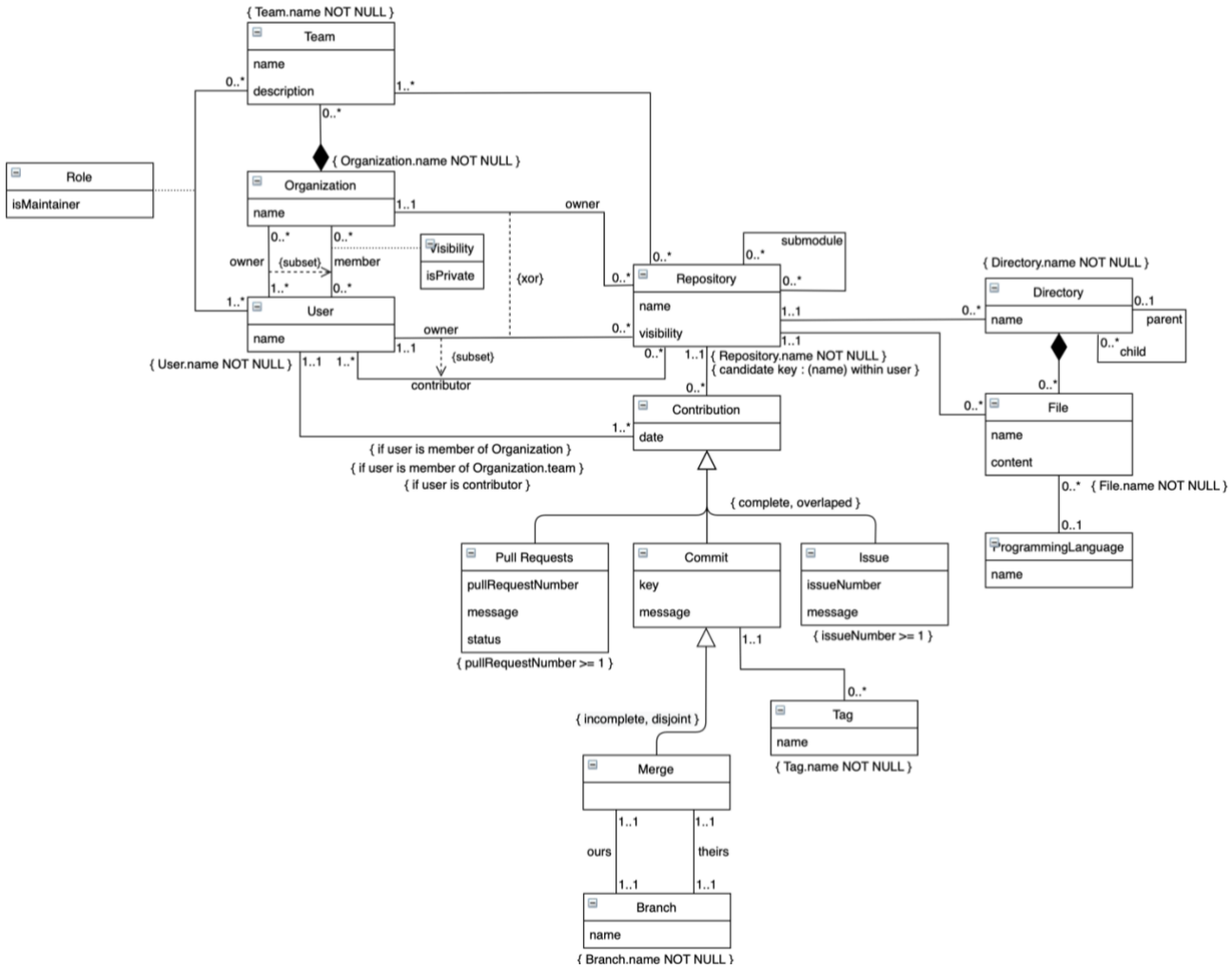

Figura 1 - Gráfico UML

[clicar na imagem para aceder ao link]

Referências

Git and Software Freedom Conservancy. *Git.* 2021. https://git-scm.com/ (acedido em 7 de Março de 2021).
GitHub, Inc. *GitHub Documentation.* 2021. https://docs.github.com/en (acedido em 7 de Março de 2021).