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

Contents

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.

Diagramas UML

1ª versão

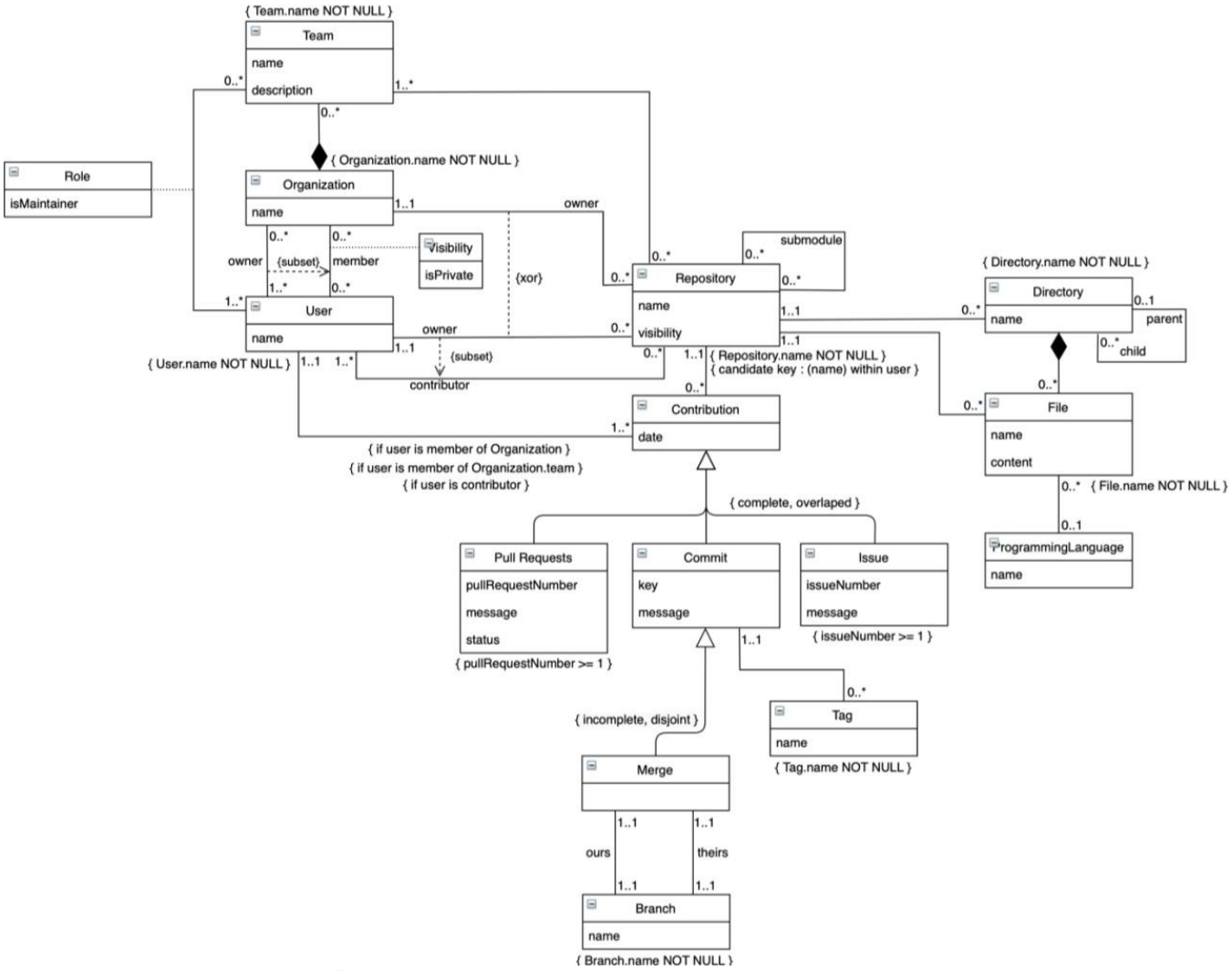


Figura 1 - Diagrama UML – 1ª versão.

2ª versão – Diagrama revisto após comentários

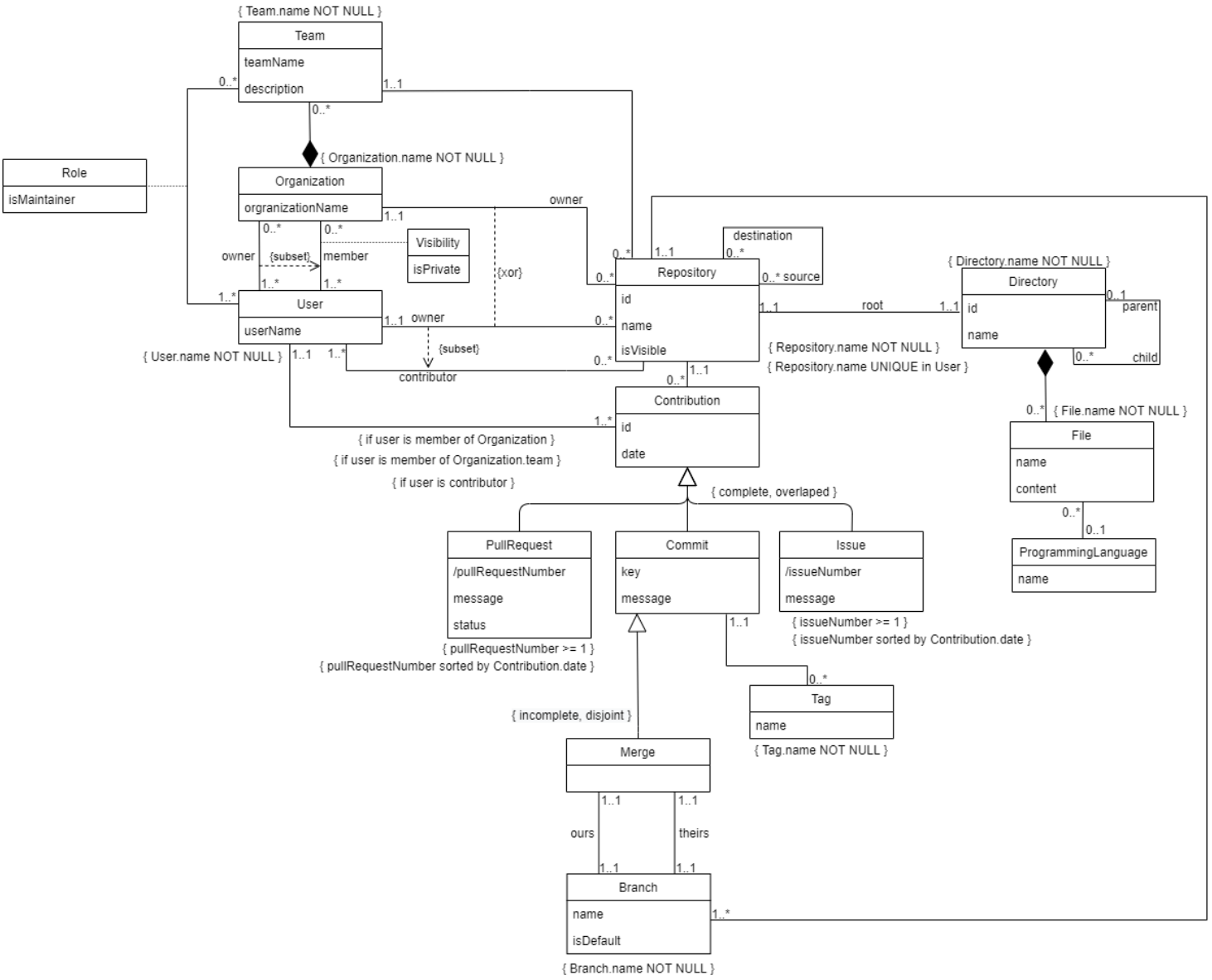


Figura 2 - Diagrama UML – 2ª versão.

[clicar na imagem para aceder ao link]

Esquema Relacional

Relações

User(<u>userName</u>)

Organization(<u>organizationName</u>)

Team(<u>teamName</u>, <u>organization</u>->Organization)

Repository(<u>ID</u>, name, rootDirectory->Directory, isVisible)
ID → name, rootDirectory, isVisible

Branch(<u>name</u>, <u>repository</u>->Repository, isDefault)
name, repository → isDefault

Contribution(<u>ID</u>, user->User, repository->Repository, date)
ID → user, repository, date

Tag(<u>name</u>, <u>commit</u>->Commit)

Directory(<u>ID</u>, name)
ID → name

File(<u>name</u>, <u>directory</u>->Directory, content, programmingLanguage->ProgrammingLanguage)
name, directory → content, programmingLanguage

ProgrammingLanguage(<u>name</u>)

Ligações

OwnerRepository(<u>user</u>->User, <u>repository</u>->Repository)

ContributorRepository(<u>user</u> ->User, <u>repository</u>->Repository)

TeamRepository(<u>teamName</u>->Team, <u>teamOrganization</u>->Team, <u>repository</u>->Repository)

OrganizationRepository(<u>organization</u>->Organization, <u>repository</u>->Repository)

OrganizationUserOwner(<u>user</u>->User, <u>organization</u>->Organization)

OrganizationUserMember(<u>user</u>->User, <u>organization</u>->Organization, isPrivate)

Associações

TeamRole(user->User, teamName->Team, teamOrganization->Team, isMaintainer)
user, teamName, teamOrganization → isMaintainer

Auto associações

Submodule(source->Repository, destination->Repository)

FolderRelationship(child->Directory, parent-> Directory)

Generalizações

PullRequest(ID->Contribution.ID, pullRequestNumber, status, message)
ID → pullRequestNumber, status, message

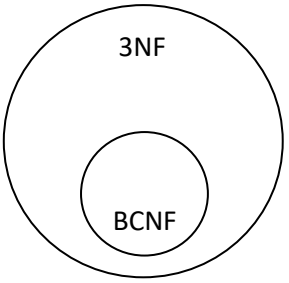
Commit(ID->Contribution.ID, commitHash, message)
ID → commitHash, message

Issue(ID->Contribution.ID, issueNumber, message)
ID → issueNumber, message

Merge(ID->Commit.ID, oursName->Branch, oursRepository->Branch, theirsName->Branch, theirsRepository->Branch)
ID → oursName, oursRepository, theirsName, theirsRepository

Análise dependências funcionais e formas normais

A *Boyce-Codd Normal Form, BCNF*, é uma forma normal – um método utilizado com o propósito de estruturar uma base de dados – muito utilizada na normalização de base de dados. Foi desenvolvida em 1974, por Raymond F.Boyce e Edgar F. Codd, com o intuito de ultrapassar limitações da 3ª forma normal, como a redundância.



Um esquema relacional encontra-se na BCNF se para qualquer dependência X → Y:

- Y ϵ X
- X é uma super chave

Partindo do princípio que, se uma relação estiver na BCNF, estará também na 3NF, efetuar-se-á apenas a prova para a BCNF, mostrando que todas as as dependências funcionais mencionadas no capítulo anterior se encontram nesta forma normal.

User
{ userName }$^{+}$ = { userName }

Organization
{ organizationName }$^{+}$ = { organizationName }

Team
{ teamName, organization }$^{+}$ = { teamName, organization }

Repository
{ ID }$^{+}$ = { ID, name, rootDirectory, isVisible }

Branch
{ name, repository }$^{+}$ = { name, repository, isDefault }

Contribution
{ ID }$^{+}$ = { ID, user, repository, date }

Tag
{ name, commit }$^{+}$ = { name, commit }

Directory
{ ID }$^{+}$ = { ID, name }

File
{ name, directory }$^{+}$ = { name, directory, content, programmingLanguage }

ProgrammingLanguage
{ name }$^{+}$ = { name }

OwnerRepository
{ user, repository }$^{+}$ = { user, repository }

ContributorRepository
{ user, repository }$^{+}$ = { user, repository }

TeamRepository
{ teamName, teamOrganization, repository }$^{+}$ = { teamName, teamOrganization, repository }

OrganizationRepository
{ organization, repository }$^{+}$ = { organization, repository }

OrganizationUserOwner
{ user, organization }$^{+}$ = { user, organization }

OrganizationUserMember
{ user, organization }$^{+}$ = { user, organization, isPrivate }

TeamRole
{ user, teamName, teamOrganization }$^{+}$ = { user, teamName, teamOrganization, isMaintainer }

Submodule
{ source, destination }$^{+}$ = { source, destination }

FolderRelationship
{ child, parent }$^{+}$ = { child, parent }

PullRequest
{ ID }$^{+}$ = { ID, pullRequestNumber, status, message }

Commit
{ ID }$^{+}$ = { ID, commitHash, message }

Issue
{ ID }$^{+}$ = { ID, issueNumber, message }

Merge
{ ID }$^{+}$ = { ID, ours, theirs }

Restrições

User

- → *userName* é chave primária (***PRIMARY KEY***)
- → *userName* não pode corresponder a uma *string* vazia (***NOT NULL***)

Organization

- → *organizationName* é chave primária (***PRIMARY KEY***)
- → *organizationName* não pode corresponder a uma *string* vazia (***NOT NULL***)

Team

- → *teamName* e *organization* são chave primária composta (***PRIMARY KEY***)
- → *organization* é chave estrangeira (***FOREIGN KEY***)
- → *teamName* não pode corresponder a uma *string* vazia (***NOT NULL***)

Repository

- → *ID* é chave primária (***PRIMARY KEY***)
- → *ID* tem que ter um valor superior ou igual a 1 (***CHECK***)
- → *rootDirectory* é chave estrangeira (***FOREIGN KEY***)
- → *rootDirectory* é único, *i.e.* dois repositórios não podem ter o mesmo *rootDirectory* (***UNIQUE***)
- → *name* não pode corresponder a uma *string* vazia (***NOT NULL***)
- → *isVisible* só pode ter o valor 0 ou 1 (***CHECK***)

Branch

- → *name* e *repository* são chave primária composta (***PRIMARY KEY***)
- → *repository* é chave estrangeira (***FOREIGN KEY***)
- → *name* não pode corresponder a uma *string* vazia (***NOT NULL***)
- → *name* tem o valor pré-definido "*main*" (***DEFAULT***)
- → *isDefault* só pode ter o valor 0 ou 1 (***CHECK***)

Contribution

- → *ID* é chave primária (***PRIMARY KEY***)
- → *ID* tem que ter um valor superior ou igual a 1 (***CHECK***)
- → *user* e *repository* são chaves estrangeiras (***FOREIGN KEY***)
- → só membros de uma organização podem fazer uma contribuição para um repositório da mesma[1]
- → só membros de uma equipa podem fazer uma contribuição para um repositório da mesma[1]
- → só contribuidores de um repositório "particular" podem fazer uma contribuição para o mesmo[1]

[1] Restrições a serem implementadas na terceira entrega do projeto, dado requererem a implementação de *triggers*.

Tag

→ *name* e *commit* são chave primária composta (***PRIMARY KEY***)
→ *commit* é chave estrangeira (***FOREIGN KEY***)
→ *name* não pode corresponder a uma string vazia (***NOT NULL***)
→ *name* é única no âmbito de um repositório (***UNIQUE***) [1]

Directory

→ *ID* é chave primária (***PRIMARY KEY***)
→ *ID* tem que ter um valor superior ou igual a 1 (***CHECK***)
→ *name* não pode corresponder a uma *string* vazia (***NOT NULL***)
→ *name* não pode ter *length* superior a 255 caracteres (***CHECK***)

File

→ *name* e *directory* são chave primária composta (***PRIMARY KEY***)
→ *directory* e *programmingLanguage* são chaves estrangeiras (***FOREIGN KEY***)
→ *name* não pode corresponder a uma *string* vazia (***NOT NULL***)
→ *name* não pode ter *length* superior a 255 caracteres (***CHECK***)
→ *name* não pode conter nenhum dos caracteres especiais que se seguem: / | \ * ? : " < > (***CHECK***) [1]

Programming Language

→ *name* é chave primária (***PRIMARY KEY***)
→ *name* não pode corresponder a uma *string* vazia (***NOT NULL***)

OwnerRepository

→ *user* e *repository* são chave primária composta (***PRIMARY KEY***)
→ *user* e *repository* são chaves estrangeiras (***FOREIGN KEY***)
→ no âmbito de um utilizador, apenas é permitido um repositório com um determinado nome (***UNIQUE***) [1]

ContributorRepository

→ *user* e *repository* são chave primária composta (***PRIMARY KEY***)
→ *user* e *repository* são chaves estrangeiras (***FOREIGN KEY***)
→ no âmbito de um utilizador, apenas é permitido um repositório com um determinado nome (***UNIQUE***) [1]

TeamRepository

→ *teamName*, *teamOrganization* e *repository* são chave primária composta (***PRIMARY KEY***)
→ *teamName*, *teamOrganization* e *repository* são chaves estrangeiras (***FOREIGN KEY***)
→ no âmbito de uma equipa, apenas é permitido um repositório com um determinado nome (***UNIQUE***) [1]

OrganizationRepository

→ *organization* e *repository* são chave primária composta (***PRIMARY KEY***)
→ *organization* e *repository* são chaves estrangeiras (***FOREIGN KEY***)
→ no contexto de uma organização, apenas é permitido um repositório com um determinado nome (***UNIQUE***) [1]

OrganizationUserOwner

→ *user* e *organization* são chave primária composta (***PRIMARY KEY***)
→ *user* e *organization* são chaves estrangeiras (***FOREIGN KEY***)

OrganizationUserMember

→ *user* e *organization* são chave primária composta (***PRIMARY KEY***)
→ *user* e *organization* são chaves estrangeiras (***FOREIGN KEY***)
→ *isPrivate* só pode ter o valor 0 ou 1 (***CHECK***)

TeamRole

→ *user*, *teamName* e *teamOrganization* são chave primária composta (***PRIMARY KEY***)
→ *user*, *teamName* e *teamOrganization* são chaves estrangeiras (***FOREIGN KEY***)
→ *isMaintainer* só pode ter o valor 0 ou 1 (***CHECK***)

Submodule

→ *source* e *destination* são chave primária composta (***PRIMARY KEY***)
→ *source* e *destination* são chaves estrangeiras (***FOREIGN KEY***)

FolderRelashionship

→ *child* e *parent* são chave primária composta (***PRIMARY KEY***)
→ *child* e *parent* são chaves estrangeiras (***FOREIGN KEY***)

PullRequest

→ *ID* é chave primária (***PRIMARY KEY***)
→ *ID* é chave estrangeira (***FOREIGN KEY***)
→ *status* só pode ter o valor *OPEN* (1) ou *CLOSED* (0) (***CHECK***)
→ o valor pré-definido de *status* é *OPEN* (1) (***DEFAULT***)

Commit

→ *ID* é chave primária (***PRIMARY KEY***)
→ *ID* é chave estrangeira (***FOREIGN KEY***)
→ *commitHash* tem que ter *length* igual a 40 caracteres (***CHECK***)

Issue

→ *ID* é chave primária (***PRIMARY KEY***)
→ *ID* é chave estrangeira (***FOREIGN KEY***)

Merge

- → *ID* é chave primária (***PRIMARY KEY***)
- → *ID* é chave estrangeira (***FOREIGN KEY***)
- → *oursName*, *oursRepository*, *theirsName*, *theirsRepository* são chaves estrangeiras (***FOREIGN KEY***)

Considerações

Finalmente, um aspeto a salientar sobre as Chaves Estrangeiras é o de que usamos os mecanismos providenciados pelo *SQLite* para o reforço da integridade referencial. Apesar de usarmos a mesma restrição em quase todas as Chaves Estrangeiras - "*ON UPDATE CASCADE ON DELETE CASCADE*" - considerámos que esta seria a melhor opção e a que mais se adequa a todas as relações, ou seja, propagar qualquer alteração pelas tabelas referenciadas.

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).

SQLite Organization. *SQLite Documentation*. 2021. https://sqlite.org/docs.html (acedido em 03 de abril de 2021).