

Usando o GitHub a partir do Eclipse**Configuração**

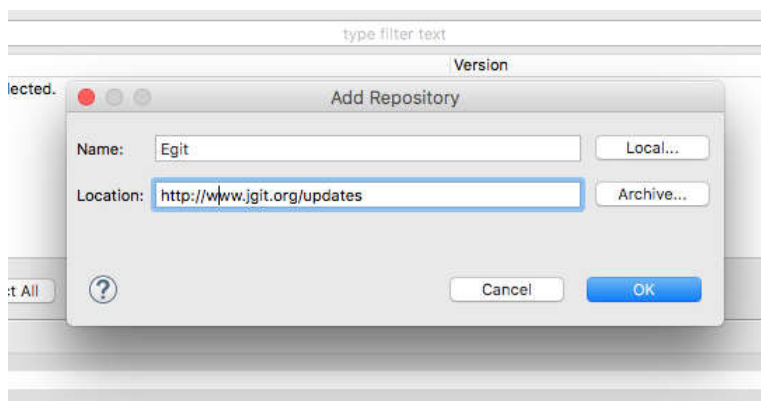
Será usado um plugin do Eclipse chamado EGit. Para verificar se já está instalado, vá em Help, InstallationDetails e digite EGit no filtro. Se achar é porque já está instalado.

Se precisar instalá-lo, vá em Help >Install New Software. Clique em Add e preencha os campos conforme a seguir:

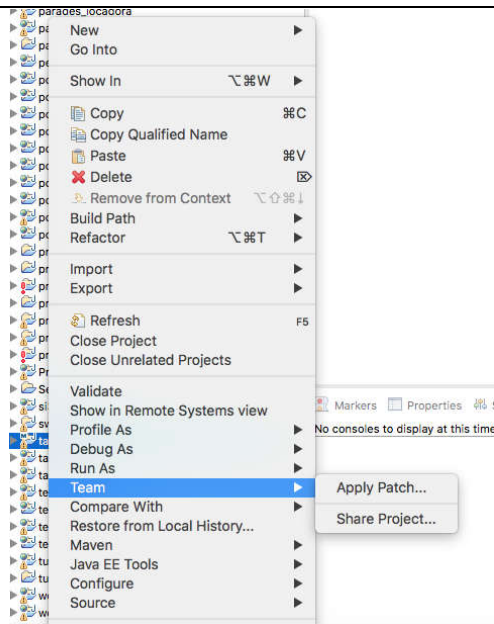
Name: EGit

Location: <http://www.jgit.org/updates>

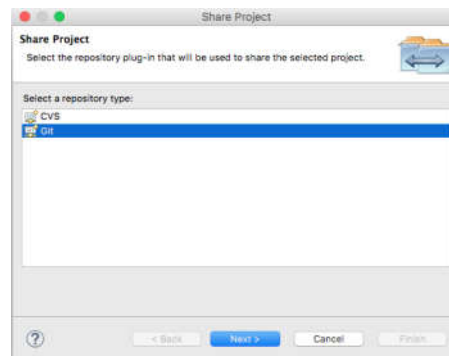
E siga o procedimento de instalação.

**Enviando um projeto local para o GitHub**

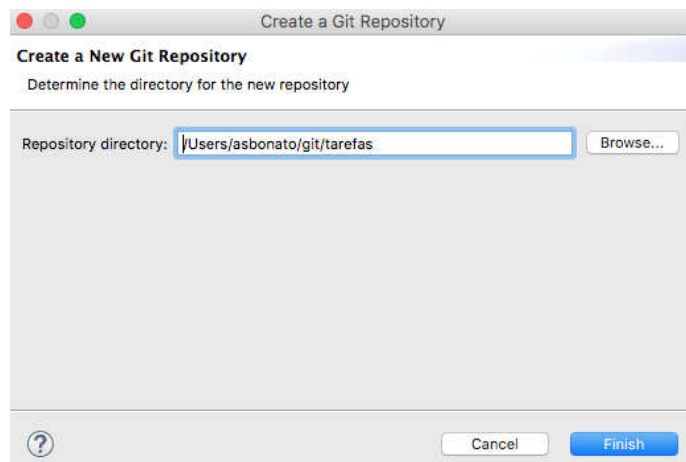
Clique com o botão direito no nome do projeto e escolha Team >Share Project...



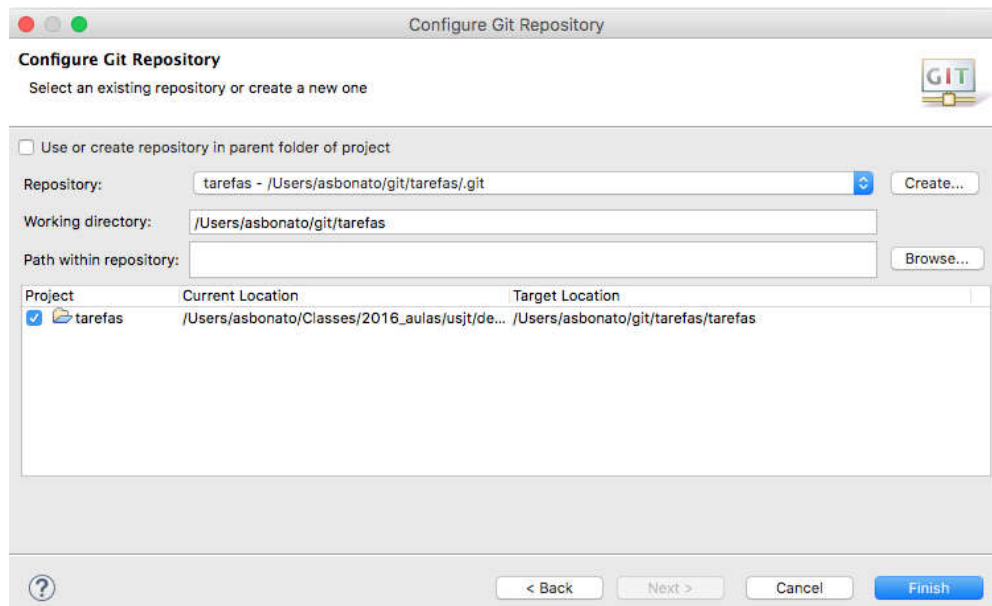
Escolha Git e clique Next.



Informe o local do repositório ou clique em Create para criar um. Não escolha seu pendrive como local. Crie uma pasta para abrigar o seu projeto. Clique em Finish.



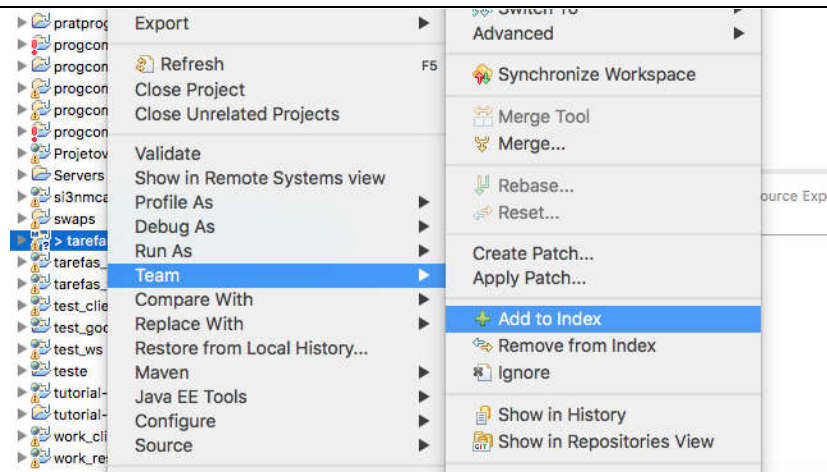
Verifique o projeto está selecionado e clique em Finish.



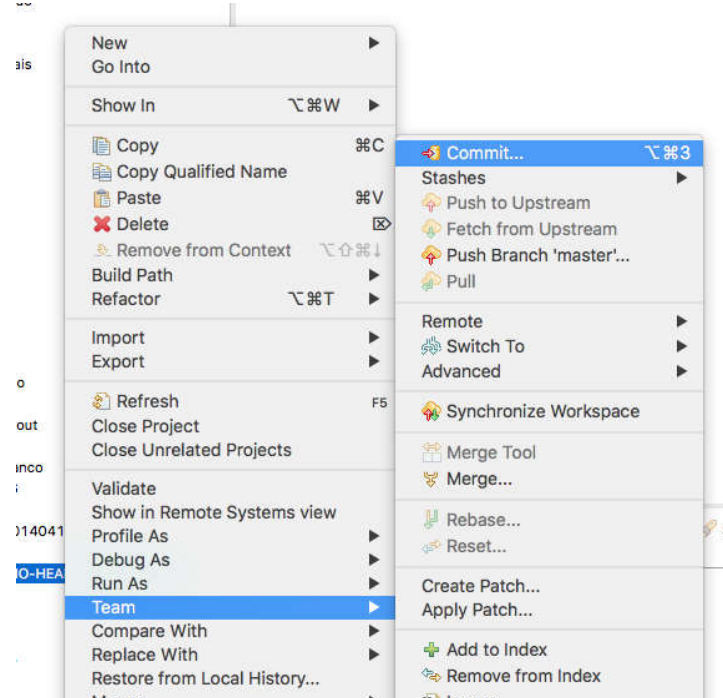
Seu projeto deve ficar assim:



NO-HEAD quer dizer que ainda não há nada no repositório do Git. Antes de adicionar, crie a área de "trabalho" do seu projeto, a INDEX. Clique com o botão direito sobre o nome do projeto > Team > Add to Index.

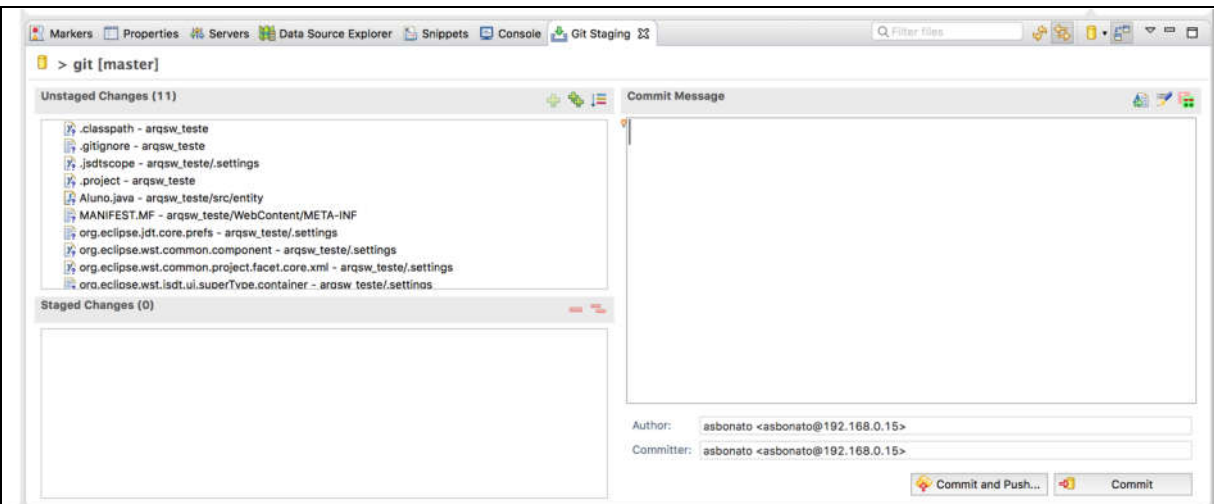


Depois, faça o primeiro Commit. Commit é salvar uma versão no repositório. Botão direito sobre o nome do projeto > Team >Commit.

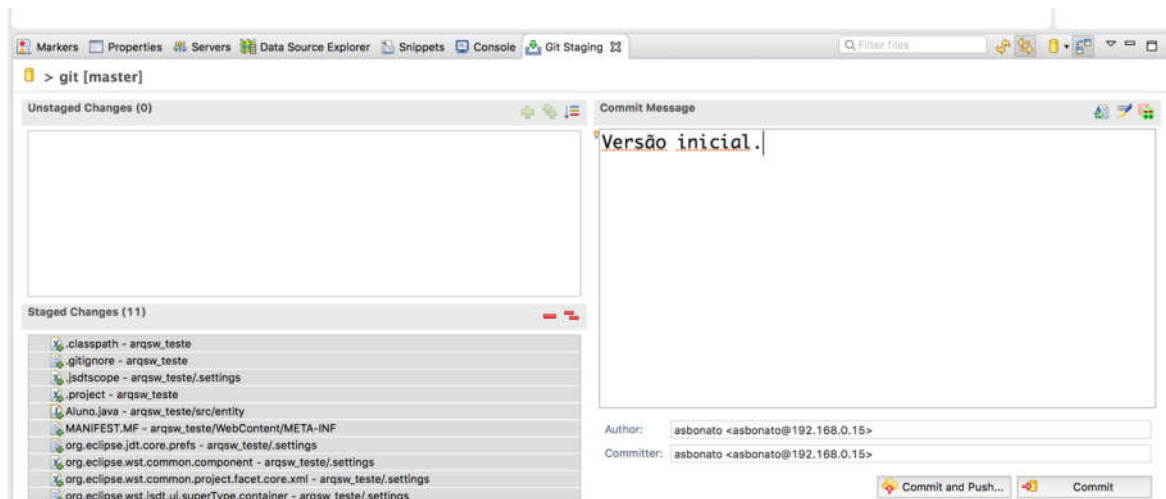


Selecione os arquivos que estão na janela UnstagedChanges (clique no primeiro, segure o Shift e clique no último) e arraste-os para a janela StagedChanges; ou clique no ++ verdinho se estiver disponível. Seus arquivos alterados, enquanto estiverem na área Unstaged, serão ignorados pelo Git. Quando forem movidos para a área Staged, serão comitados no próximo commit.

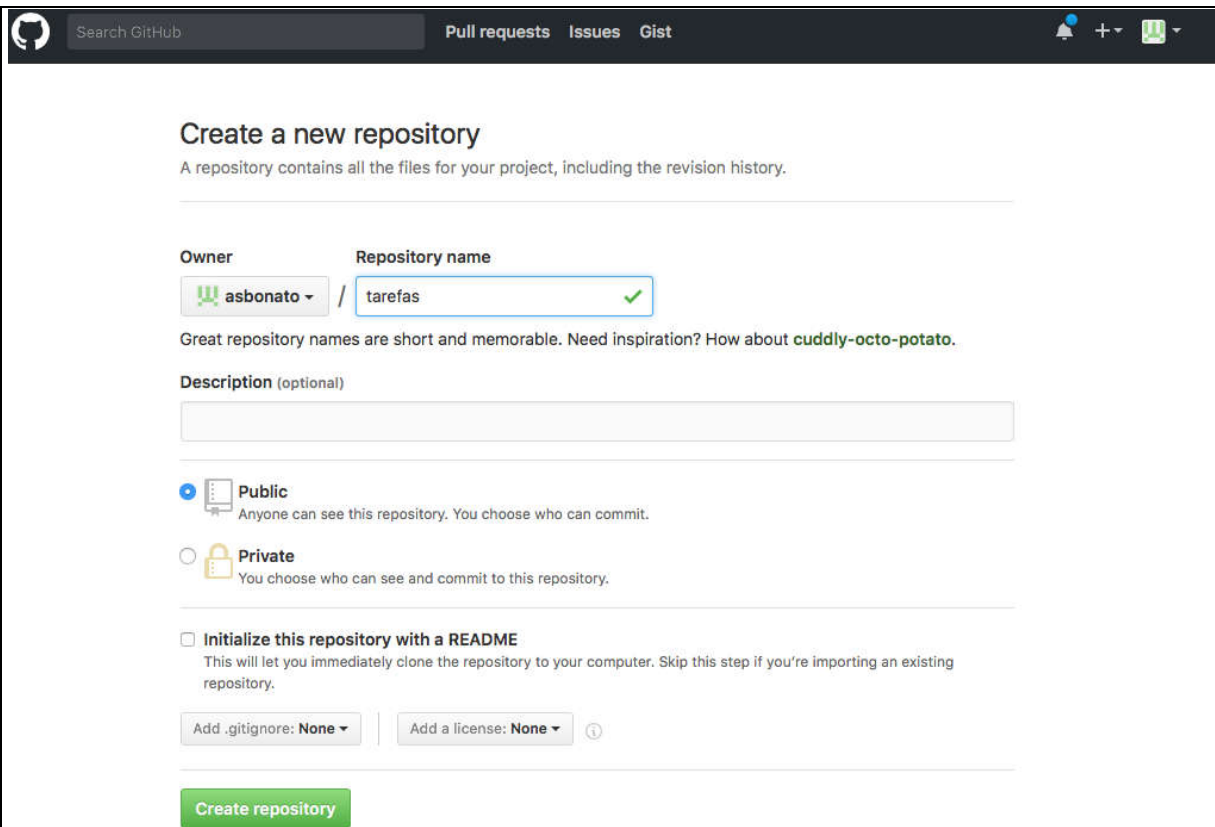
Escreva a CommitMessage, algo que informe a qual versão se refere o arquivo, ou qual a mudança que foi feita.



Clique depois em CommitandPush. Se você clicar em Commit, salvará no repositório local somente. Se clicar em CommitandPush, salvará no local e exportará para o GitHub, que é o que queremos. Se você não clicou em CommitandPush, mas em Commit, não tem problema. Clique com o botão direito sobre o nome do projeto > Team > Remote >Push que tem o mesmo efeito.



Preencha os dados da sua conta no GitHub. Se não tiver, vá em www.github.com e crie uma. Depois, crie um novo repositório. A tela seguinte é a de criação do repositório tarefas na minha conta do GitHub. Clique em Createrepository.



Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: asbonato / Repository name: tarefas

Great repository names are short and memorable. Need inspiration? How about [cuddly-octo-potato](#).

Description (optional)

☒ **Public**
Anyone can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** | Add a license: **None**

Create repository

Ainda no GitHub, clique no Copyto clipboard do Quick setup.



Quick setup — if you've done this kind of thing before

☒ Set up in Desktop or ☐ HTTPS ☐ SSH <https://github.com/asbonato/tarefas.git>

We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

Copy to clipboard

Para preencher os dados da conta, cole a url copiada no campo URI. O restante será preenchido sozinho (exceto usuário e senha). Não clique em Store in SecureStore para não salvar sua senha no computador do laboratório. Clique em Next.

Push Branch master

Destination Git Repository
Enter the location of the destination repository.

Remote name:

Location

URI:

Host:

Repository path:

Connection

Protocol:

Port:

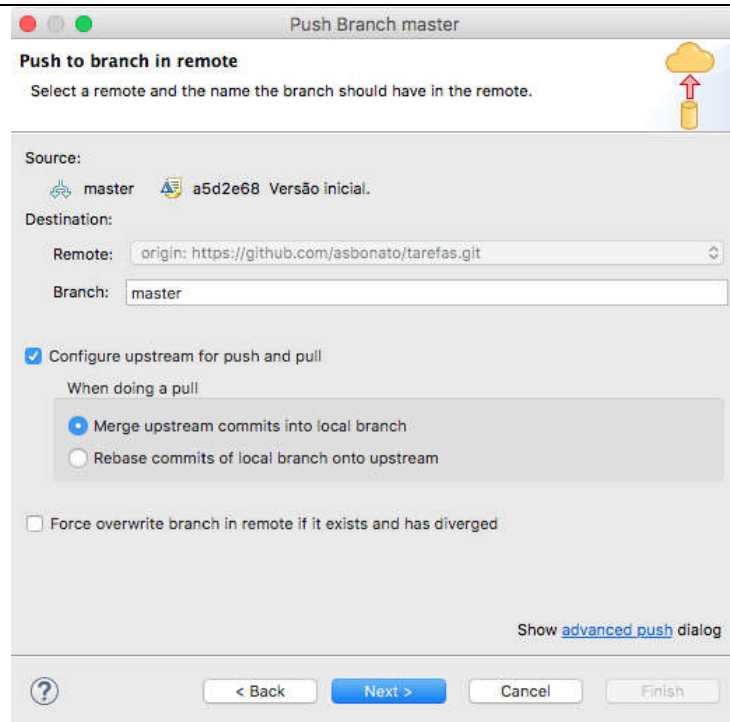
Authentication

User:

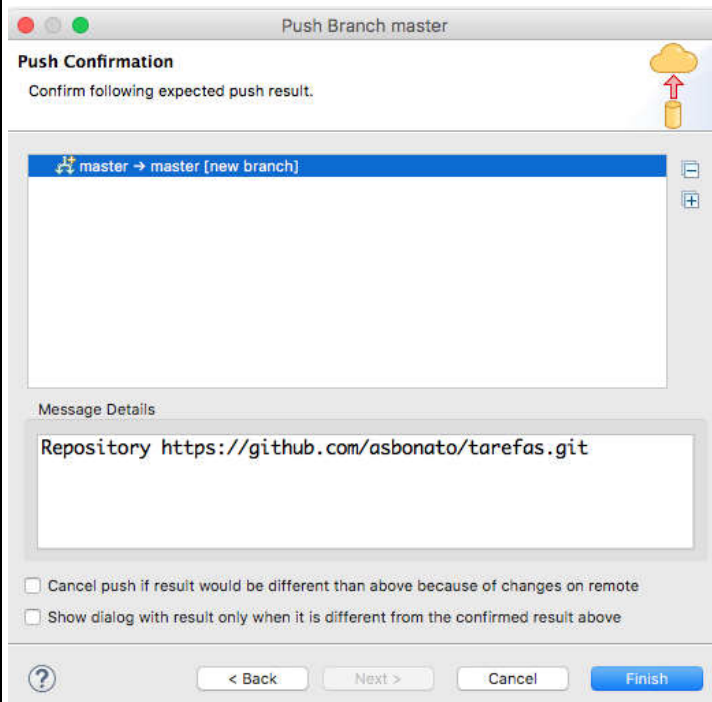
Password:

☒ Store in Secure Store

Clique novamente em Next



Clique em Finish.



Verifique seu repositório no GitHub.

asbonato / tarefas

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Pulse Graphs Settings

Branch: master tarefas / tarefas /

Create new file Upload files Find file History

asbonato Versão inicial. Latest commit a5d2e68 15 minutes ago

..		
.settings	Versão inicial.	15 minutes ago
Users/asbonato/Classes/2016_aulas/usjt/dev_web/workspace/tarefas/target/m...	Versão inicial.	15 minutes ago
WebContent	Versão inicial.	15 minutes ago
src/br/usjt/tarefas	Versão inicial.	15 minutes ago
.classpath	Versão inicial.	15 minutes ago
.gitignore	Versão inicial.	15 minutes ago
.project	Versão inicial.	15 minutes ago
pom.xml	Versão inicial.	15 minutes ago

Alterando o código

As alterações no código também precisam ser "comitadas", neologismo que quer dizer que precisam ser salvas no repositório.

Para fazer isso o processo é semelhante. Clique com o botão direito sobre o nome do projeto, vá em Team >Commit. Preencha a mensagem de commit. Veja que as classes alteradas aparecem na lista. Clique em CommitandPush.

Commit Changes

Commit Changes to Git Repository

Commit message

Adicionei comentários.

Author: asbonato <asbonato@gmail.com>

Committer: asbonato <asbonato@gmail.com>

Files (1/1)

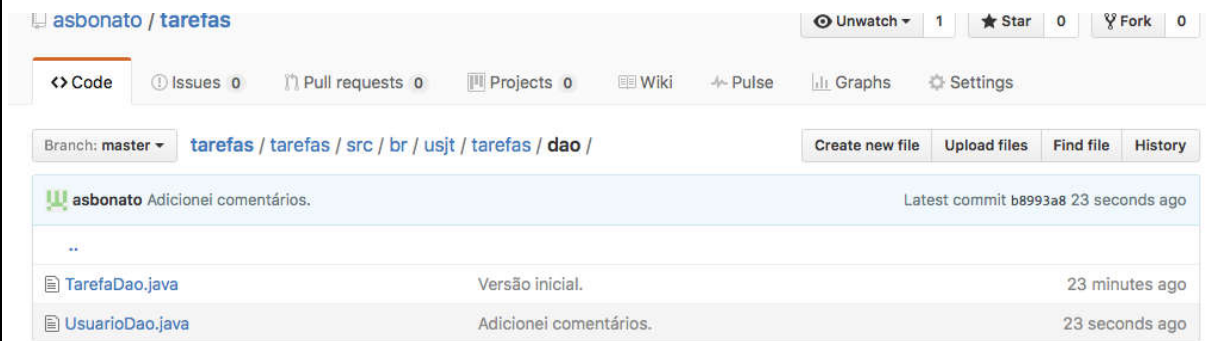
type filter text

Status	Path
<input checked="" type="checkbox"/>	tarefas/src/br/usjt/tarefas/dao/UsuarioDao.java

Open [Git Staging](#) view

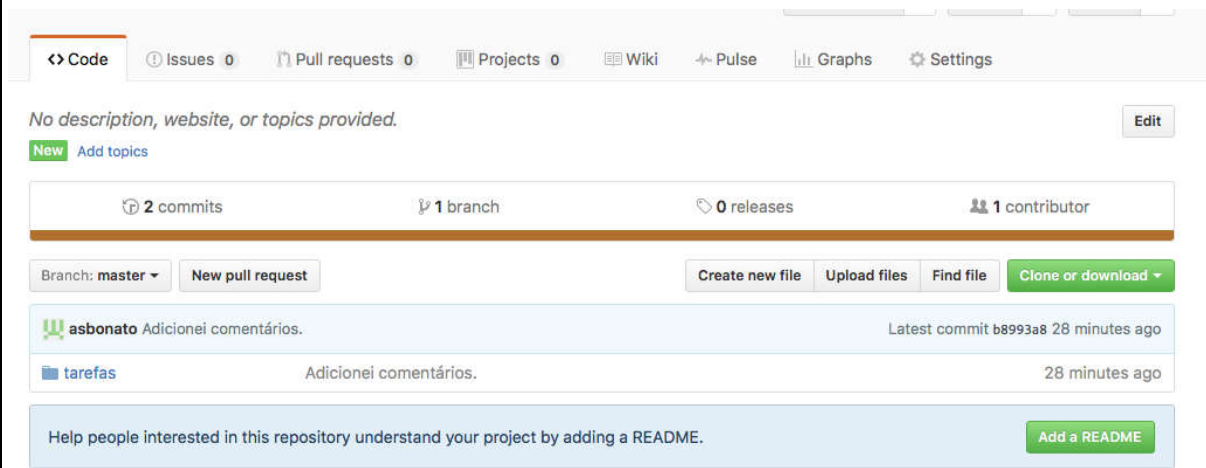
Commit and Push Cancel Commit

O código aparece alterado no repositório do GitHub.

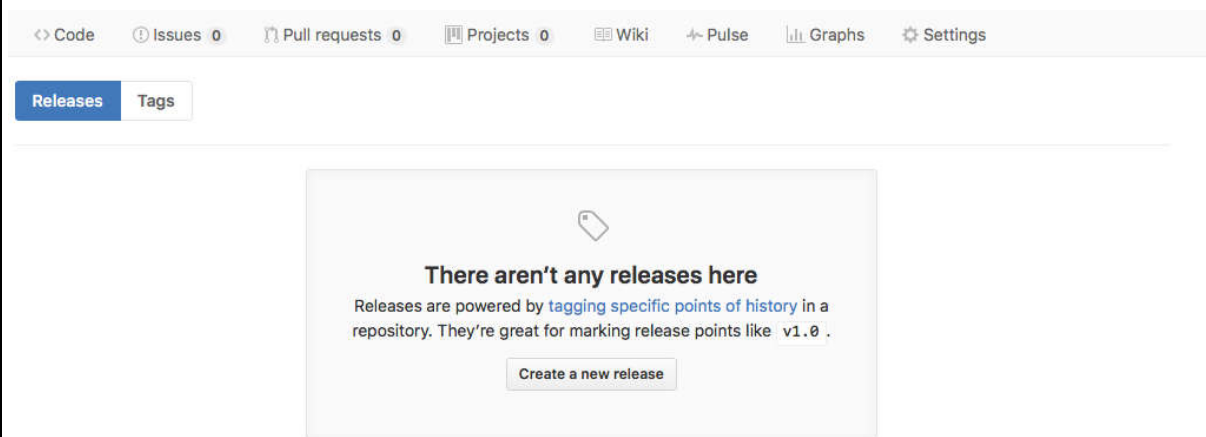


Entregando versões para seu professor (gerando releases no GitHub)

Para entregar uma versão, gere um release e mande um e-mail com o link do seu release para ele. Não trabalhe com vários repositórios. Para gerar um release clique em Code e depois em Releases (no caso 0 releases) do seu repositório.



Clique em Create a New Release



Preencha as informações do release. No nome, use o nome da aula a que se refere a entrega. Nos comentários, escreva o que foi feito. Clique em Publish release.

Releases

Tags

1 @ Target: master ▼

Excellent! This tag will be created from the target when you publish this release.

Aula03 - DAO, TO, Service e Factory

Write

Preview

Markdown supported

Refatoração do Model usando os patterns DAO, Factory, Service e TO (Javabeau).

Attach files by dragging & dropping or selecting them.

Attach binaries by dropping them here or selecting them.

☐ This is a pre-release
We'll point out that this release is identified as non-production ready.

Publish release

Save draft

Pronto, seu release fica assim.

Releases

Tags


Edit release

Delete

Latest release

1
b8993a8

Aula03 - DAO, TO, Service e Factory

 asbonato released this 34 minutes ago

Refatoração do Model usando os patterns DAO, Factory, Service e TO (Javabeau).

Downloads

Source code (zip)

Source code (tar.gz)

Mande o link para o professor de lab por e-mail ou combine outro formato de entrega com ele.

<https://github.com/asbonato/tarefas/releases/tag/1>

Clonando seu projeto para outros Eclipse

- 1.1. Acesse o GitHub;
- 1.2. Localize à direita a seção HTTPS clone URL e clique no botão copy to clipboard.

Clone seu projeto pelo Eclipse:

- 2.1. Abra seu Eclipse;
- 2.2. Acesse File > Import...;
- 2.3. Escolha a opção Projects from Git;
- 2.4. Escolha a opção Clone URI;

Os campos referente a localização devem ser preenchidos automaticamente. Caso contrário, retorne ao item 1.2.

- 2.5. Clique em Next;
- 2.6. Em Local destination você pode alterar o local onde o projeto será clonado, definindo por exemplo seu workspace;
- 2.7. Em Select a wizard to use importing projects escolha a opção Import as general project;
- 2.8. Finish.

Mais sobre o Git

Dudler, Roger; **Git - Guia Prático**; disponível online em http://rogerdudler.github.io/git-guide/index.pt_BR.html. Acessado em 28/02/2017.