



Tópicos Especiais



Prof. MSc. Rafael Staiger Bressan  
rafael.bressan@unicesumar.edu.br

# Git-SCM

## Source Control Management

- Sistema de controle de versões distribuídos



**git**

Linux Torvals (criador do Linux) - 2005



# O que é controle de versão?

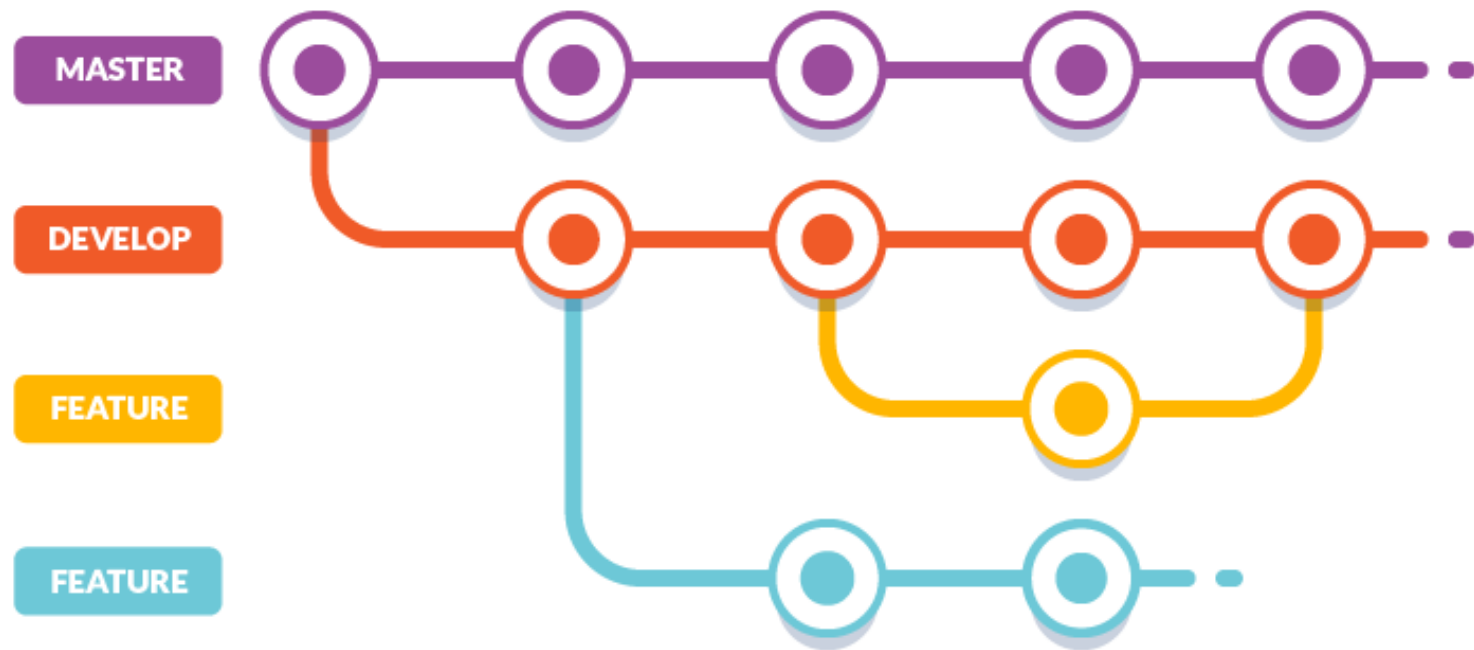


# O que é controle de versão?

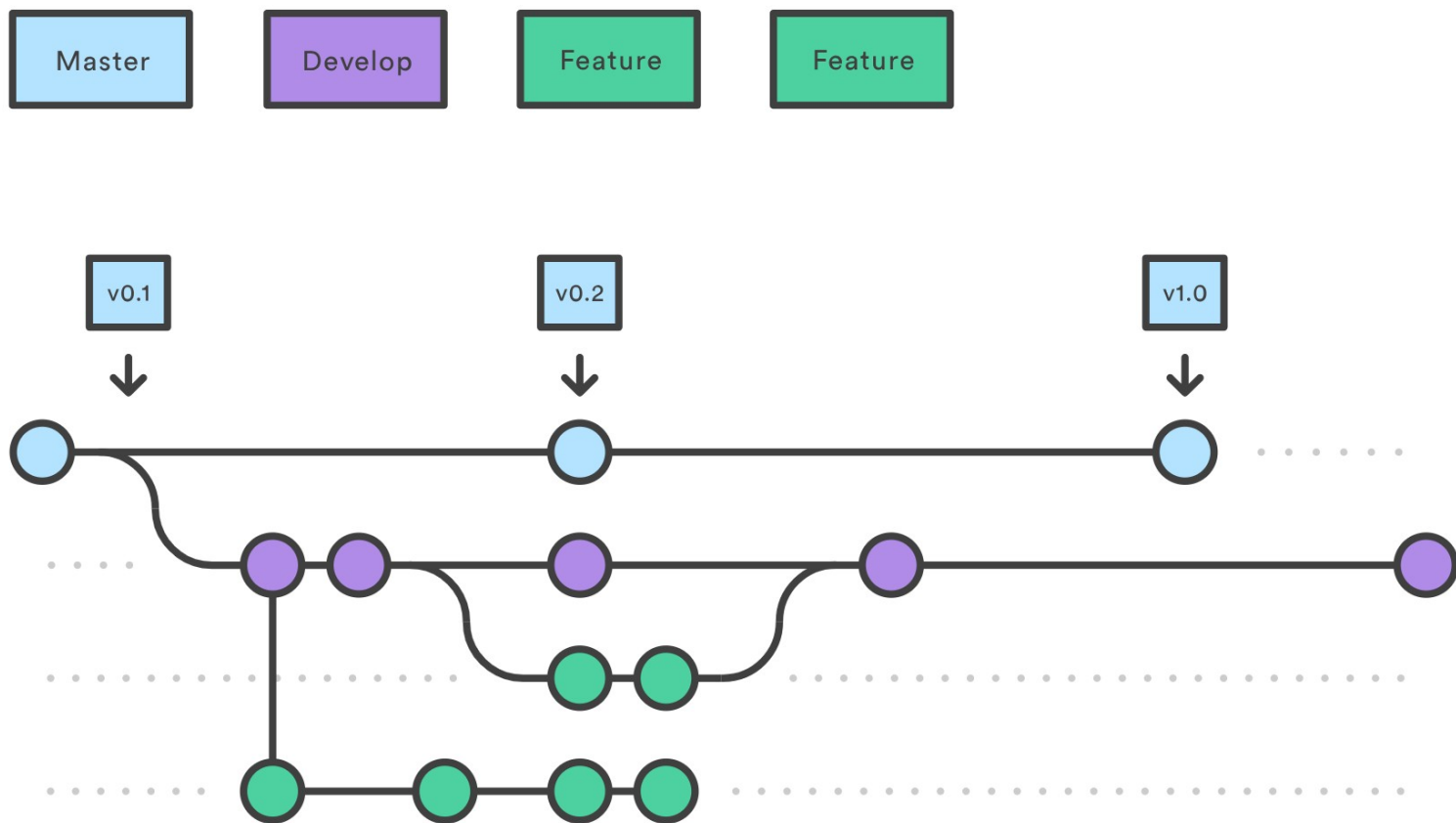
- Sistema que registra as mudanças feitas em um arquivo ou um conjunto de arquivos ao longo do tempo de forma que você possa recuperar versões específicas;
- Documentação - GIT
  - <https://git-scm.com/book/pt-br/v2>



# O que é controle de versão?



# O que é controle de versão?





# Git Flow

- Maneira dinâmica e ativa de organização de branches dentro do Git.
- Site Oficial
  - <https://danielkummer.github.io/git-flow-cheatsheet/index.html>





# Branches (Ramos)

- Master
- Develop
- Features
- Release
- Hotfixes



# Branches (Ramos)

- Master
  - Só deve existir uma branch Master no projeto;
  - Cópia fiel do sistema em produção;
  - Um espelho de todo o sistema que está operando em produção.



# Branches (Ramos)

- Develop
  - Só deve existir uma branch Develop no projeto;
  - Base de trabalho e desenvolvimento do projeto;
  - Sempre está sincronizada com a Master;
  - Todas as novas funcionalidades (Features) e toda a evolução do projeto é feita com base na branch Develop.



# Branches (Ramos)

- Features

- Conjunto de branches que compõe toda a evolução do projeto;
- Com base na Develop, toda nova funcionalidade a ser implementada no sistema é feita pelo Fluxo (Flow) Features;
- Podem existir N branches para este Fluxo (Flow).
- Em um mesmo Fluxo Features várias branches de novas funcionalidades podem coexistir simultaneamente. Todas serão “mergeadas” uma a uma para a Develop.



# Branches (Ramos)

- Release
  - Só deve existir apenas uma branch Release;
  - E a branch que controla a versão do sistema;
  - Quando há uma quantidade X de merges das Features com a Develop, um novo Release pode ser criado, criando uma nova versão para o sistema.
  - Cada arquiteto ou responsável pelo projeto decide quando criar uma nova Release com base no que já foi produzido.
  - Geralmente é nesta branch que acontecem todos os tipos de testes. Quando um Release é aprovado, ele é “mergeado” com a Master e com a Develop, para manter a base de desenvolvimento atualizada.



# Branches (Ramos)

- Hotfixes
  - Esta branch é utilizada para gerar correções que ocorrem em produção.
  - Geralmente um erro no sistema de produção não pode esperar todo o fluxo de Develop, Feature, Release pra depois voltar para a Master.
  - Erros no sistema de produção requerem imediatismo e é pra isso que a Hotfixes existe;
  - Quando um erro é encontrado em produção, uma Hotfix é criada com base na Master, corrigida e “mergeada” de volta para a Master e juntamente para a Develop.



# Instalando o Git

<https://git-scm.com/book/pt-br/v2/Come%C3%A7ando-Instalando-o-Git>



# Prática – 1

## Preparo do Ambiente

- Instalando o GIT
  - Ubuntu
    - `sudo apt-get install git`
  - Demais distribuições
    - <https://git-scm.com/book/pt-br/v2/Come%C3%A7ando-Instalando-o-Git>
- Configurações iniciais.
  - `git config --global user.name "NAME"`
  - `git config --global user.email "nome@email.com"`
  -

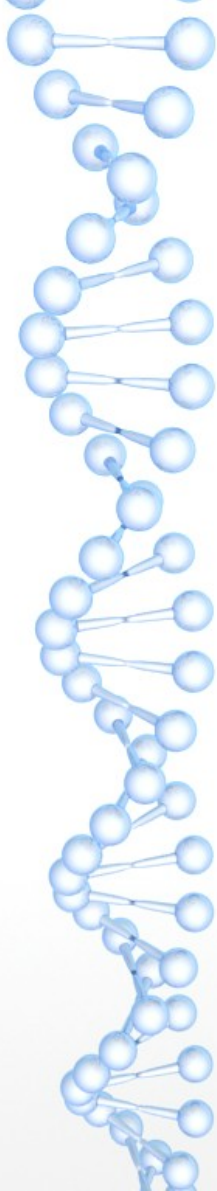




## Prática – 2

# Criando um repositório online.

- Crie um conta no github
  - <https://github.com/>
  - Abra o Github, faça login com a sua conta e clique em New repository
  - Em seguida, basta colocar o nome e descrição do repositório que você está criando e clicar em Create repository



## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner

Repository name \*

/ the-messenger

Great repository names are short and memorable. Need inspiration? How about **upgraded-funicular?**

Description (optional)

Messaging system between friends



**Public**

Anyone can see this repository. You choose who can commit.



**Private**

You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

☒ **Initialize this repository with a README**

This will let you immediately clone the repository to your computer.

Add .gitignore: **None** ▼

Add a license: **GNU General Public License v3.0** ▼



Create repository

&lt;&gt; Code

! Issues 0

Pull requests 0

▶ Actions

Projects 0

Wiki

Security

Insights

Settings

## Messaging system between friends

Edit

[Manage topics](#)

1 commit

1 branch

0 packages

0 releases

1 contributor

Branch: master ▾

New pull request

Create new file

Upload files

Find file

Clone or download ▾

Initial commit

Latest commit 1338fc8 now

LICENSE

Initial commit

now

README.md

Initial commit

now

README.md



# the-messenger

Messaging system between friends



# Prática – 3

## Comandos Básicos

- Logue-se no shell
  - Escolha o local desejado...
  - Recomendo criar um diretório (git\_workspace) para trabalhar com os projetos.
- Comandos para criar o diretório e acessá-lo
  - `mkdir NomeDoProjeto`
  - `cd NomeDoProjeto`



# Prática – 3

## Comandos Básicos

- Acesse o seu projeto criado no github e clique em (Clone or download), copie o link (.git)
  - `https://github.com/___NOME___/the-messenger.git`
- Com essas informações, podemos “clonar” o repositório para nossa máquina.



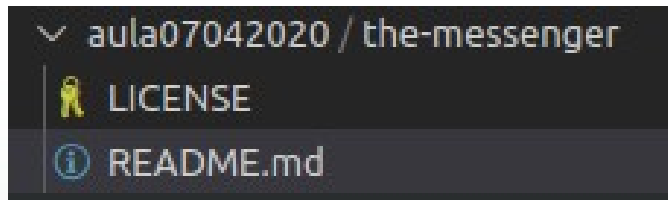
# Prática – 3

## Clonando repositório existente

- Acesse o diretório (git\_workspace) e clone o diretório com comando (git clone).

git clone [https://github.com/\\_\\_\\_NOME\\_\\_\\_/the-messenger.git](https://github.com/___NOME___/the-messenger.git)

- Após clone, acesse o diretório, verifique os arquivos





# Prática – 3

## Realizando alterações

- Estrutura do projeto
  - LICENSE
  - README.md
  - .git (Diretório oculto)
    - Diretório e Arquivos do GIT



# Prática – 3

## Realizando alterações

- Altere o arquivo (README.md)

```
1 | # TM - The Messenger
2 | Messaging system between friends
3 |
4 | # Introduction
5 | - Brief introduction to the system.
6 |
7 | # Requirements
8 | - Tests performed on Ubuntu 18.04.3 LTS with Python (3.6.8)
9 | ## libraries
10 | - Django==3.0.3
11 | - python-decouple==3.3
```

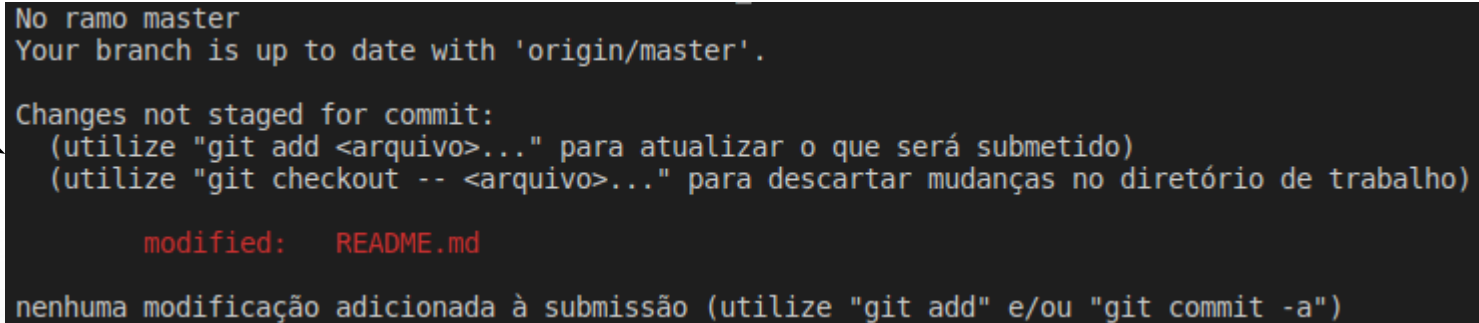




# Prática – 4

## Comandos Básicos

- Verificar em qual branch estamos trabalhando
  - git branch
- Verificar quais arquivos foram alterados
  - git status



```
No ramo master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (utilize "git add <arquivo>..." para atualizar o que será submetido)
  (utilize "git checkout -- <arquivo>..." para descartar mudanças no diretório de trabalho)

    modified:   README.md

nenhuma modificação adicionada à submissão (utilize "git add" e/ou "git commit -a")
```



# Prática – 4

## Comandos Básicos

- Adiciona os arquivos que serão enviados
  - `git add NOME DO ARQUIVO`
- Adiciona todos os arquivos
  - `git add *`
- Verificar quais arquivos foram alterados
  - `git status`

```
No ramo master
Your branch is up to date with 'origin/master'.

Mudanças a serem submetidas:
  (use "git reset HEAD <file>..." to unstage)

    modified:   README.md
```



# Prática – 4

## Comandos Básicos

- Realiza o commit descrevendo as alterações realizadas.
  - git commit

<tipo>: <O que foi feito?>

(Espaço em branco)

<Por que foi feito?>

(Espaço em branco)

<Detalhes técnicos>:

```
docs: Inserção da descrição do sistema.  
  
Atualização das informações sobre a aplicação.  
  
* Versão do SO e dependências.  
# Please enter the commit message for your changes. Lines starting  
# with '#' will be ignored, and an empty message aborts the commit.  
#  
# No ramo master  
# Your branch is up to date with 'origin/master'.  
#  
# Mudanças a serem submetidas:  
#   modified:   README.md  
#
```



# Prática – 4

## Comandos Básicos

- Persistir as alterações no repositório remoto.
  - `git push origin master`

Branch local



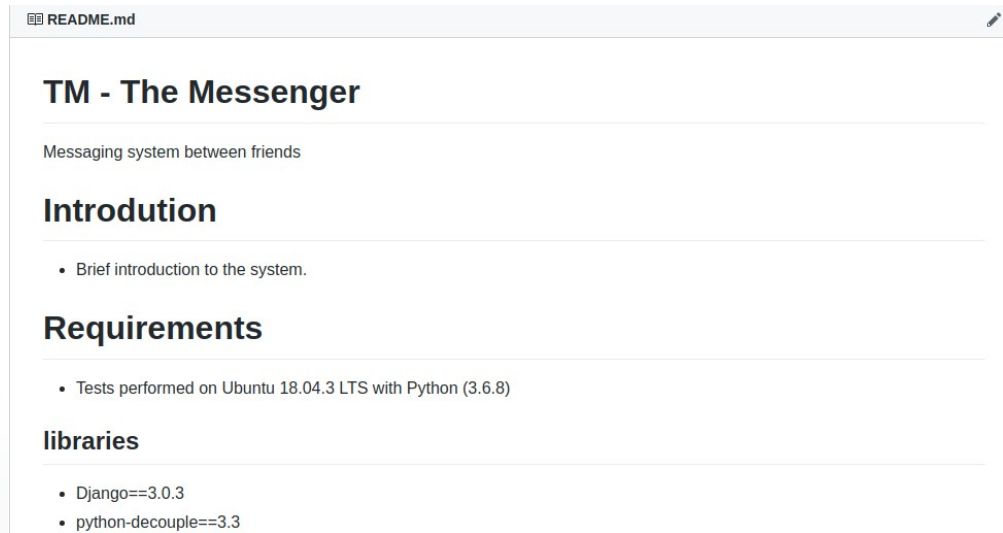
Branch remota



# Prática – 4

## Comandos Básicos

- Acesse o repositório.
- Verifique as alterações.





# Links Comandos

- <https://git-scm.com/book/pt-br/v2>
- [https://rogerdudler.github.io/git-guide/index.pt\\_BR.html](https://rogerdudler.github.io/git-guide/index.pt_BR.html)
- <https://www.hostinger.com.br/tutoriais/comandos-basico-s-de-git/>



# Criação do Projeto TM – The Messenger

- Preparo do ambiente de desenvolvimento
  - Definições - Tecnologias
    - Python (Django)
    - HTML5 / CSS / JS
      - Bootstrap
    - Sqlite3
    - .....



# Criação do Projeto TM – The Messenger

- Preparo do ambiente de desenvolvimento

- Master #Verifica as branches

```
git branch
```

- Develop

```
* master
```

```
#Cria a branch "Develop"
```

```
git checkout -b develop
```

```
Switched to a new branch 'develop'
```

```
#Verifica as branches
```

```
git branch
```

```
* develop  
master
```



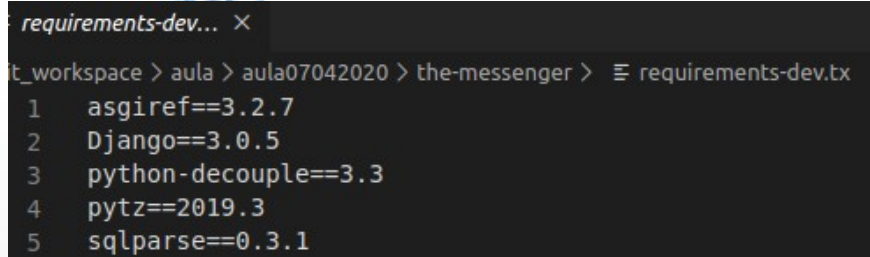


# Criação do Projeto TM – The Messenger

- Trocando entre branch - `git checkout nameBranch`.  
Deixar na branch "develop..."
  - `git checkout master`
  - `git checkout develop`

# Criação do Projeto TM – The Messenger

- Prepare o projeto base.
  - Instale e prepare o (virtualenv)
  - Ative a virtualização
  - Instale
    - Django



```
requirements-dev... x
t_workspace > aula > aula07042020 > the-messenger > requirements-dev.txt
1 asgiref==3.2.7
2 Django==3.0.5
3 python-decouple==3.3
4 pytz==2019.3
5 sqlparse==0.3.1
```

LINUX

```
$ virtualenv -p python3 .venv
$ . .venv/bin/activate
(.venv) ... $ pip3 install django
(.venv) ... $ pip3 install python-decouple
(.venv) ... $ pip3 freeze > requirements-dev.txt
(.venv) ... $ django-admin startproject project .
```

# Criação do Projeto TM – The Messenger (Linux)

- `$ virtualenv -p python3 .venv`
- `$ . .venv/bin/activate`
- `(.venv) ... $ pip3 install django`
- `(.venv) ... $ pip3 install python-decouple`
- `(.venv) ... $ pip3 freeze > requirements-dev.txt`
- `(.venv) ... $ django-admin startproject project .`
- **\*\*\*\* Atualize / altere as configurações iniciais**
  - `(settings.py)`
- `(.venv) ....$ python3 manage.py makemigrations`
- `(.venv) ....$ python3 manage.py migrate`
- `(.venv) ....$ python3 manage.py createsuperuser`
- `(.venv) ....$ python3 manage.py runserver`



The install worked successfully! Congratulations!

You are seeing this page because `DEBUG=True` is in your settings file and you have not configured any URLs.



# Criação do Projeto TM – The Messenger

- Atualiza repositório com a nova branch (develop)
- Cria um arquivo (.gitignore) para o GIT ignorar os arquivos listados.

```
git_workspace > aula > aula07042020 > the-messenger > .gitignore
1  # Items ignored by git
2  .idea
3  .venv
4  *.pyc
```

# Criação do Projeto TM – The Messenger

- Atualiza repositório com a nova branch (develop)
  - #Verificar se esta na branch develop
    - git branch
  - # Adiciona os arquivos
    - git add \*
  - # Adiciona a descrição do commit
    - git commit
  - Envia a nova branch
    - git push origin develop

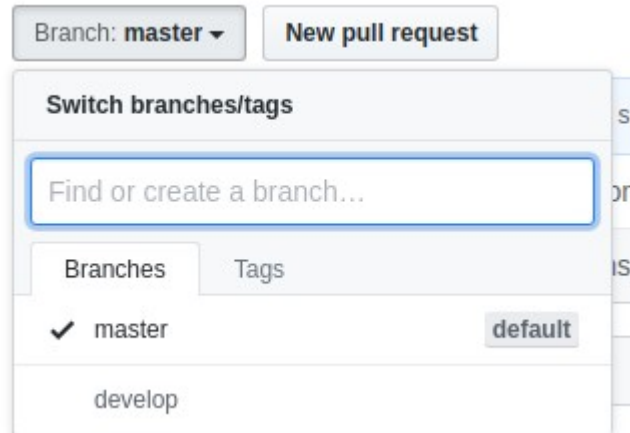
```
feat: Preparo do ambiente

Preparo do ambiente Django.

* Configuração base do Django
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# No ramo develop
# Mudanças a serem submetidas:
#   new file:   db.sqlite3
#   new file:   manage.py
#   new file:   project/_init_.py
#   new file:   project/asgi.py
#   new file:   project/settings.py
#   new file:   project/urls.py
#   new file:   project/wsgi.py
#   new file:   requirements-dev.tx
#
# Arquivos não monitorados:
#   .gitignore
#
```

# Criação do Projeto TM – The Messenger

- Verifique as alterações no github.





# Prática

- Apague o diretório (the-messenger) do seu computador.
  - O projeto está salvo no github.
- Realize os procedimentos de clone do projeto no github.
- Acesse o diretório “the-messenger”
- Mude a branch para “develop” (**git checkout develop**) e realize os procedimentos de instalação.
  - \$ virtualenv -p python3 .venv
  - \$ . .venv/bin/activate
  - (.venv) ... \$ pip3 install -r requirements-dev.txt
  - (.venv) ....\$ python3 manage.py makemigrations
  - (.venv) ....\$ python3 manage.py migrate
  - (.venv) ....\$ python3 manage.py runserver

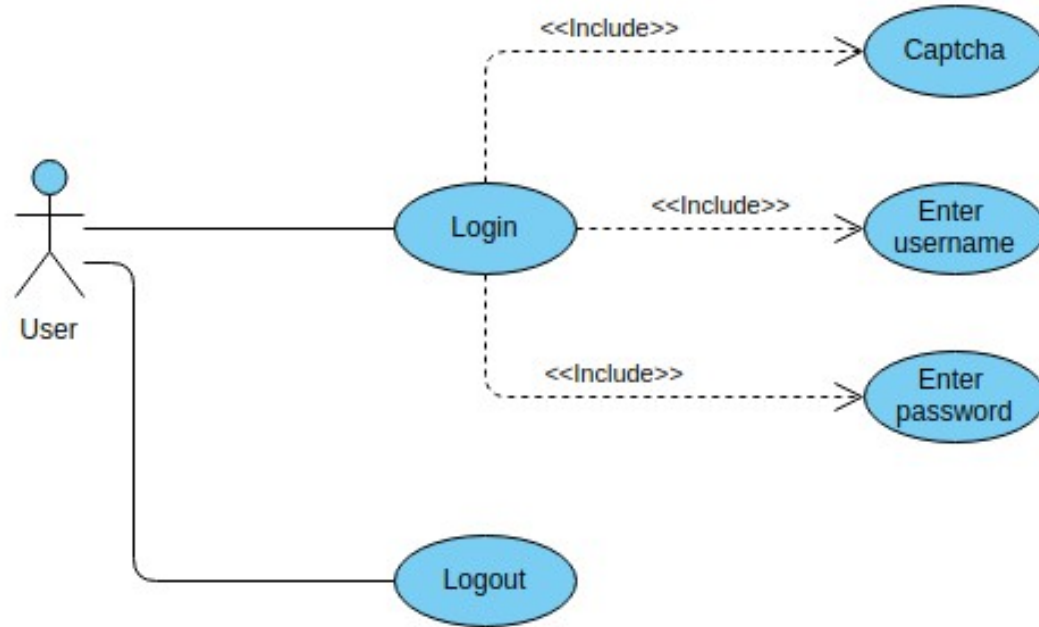


# TM – The Messenger

- Elaboração da proposta.
  - Elabore o escopo do projeto. Descreve a ideia principal e as funcionalidades globais do sistema.
- Features - (Casos de uso)
  - 01 - Sistema de Login
    - Sistema de login
      - feature/UC01-login-system



# Criação do Projeto TM – The Messenger





# Dicas

- Django
  - Contém os registros necessários para login.
    - Verifique o exemplo:
      - <https://www.youtube.com/watch?v=rfeebcQCC34>
      - <https://www.youtube.com/watch?v=BA1tGaFmbv0>
  - Gere o arquivo (models.py) para verificar todos os recursos disponíveis no banco.
    - `python manage.py inspectdb > models.py`
      - <https://docs.djangoproject.com/pt-br/3.0/howto/legacy-databases/>