

Desenvolvimento Web

WEBINAR 03 – Unidade 02

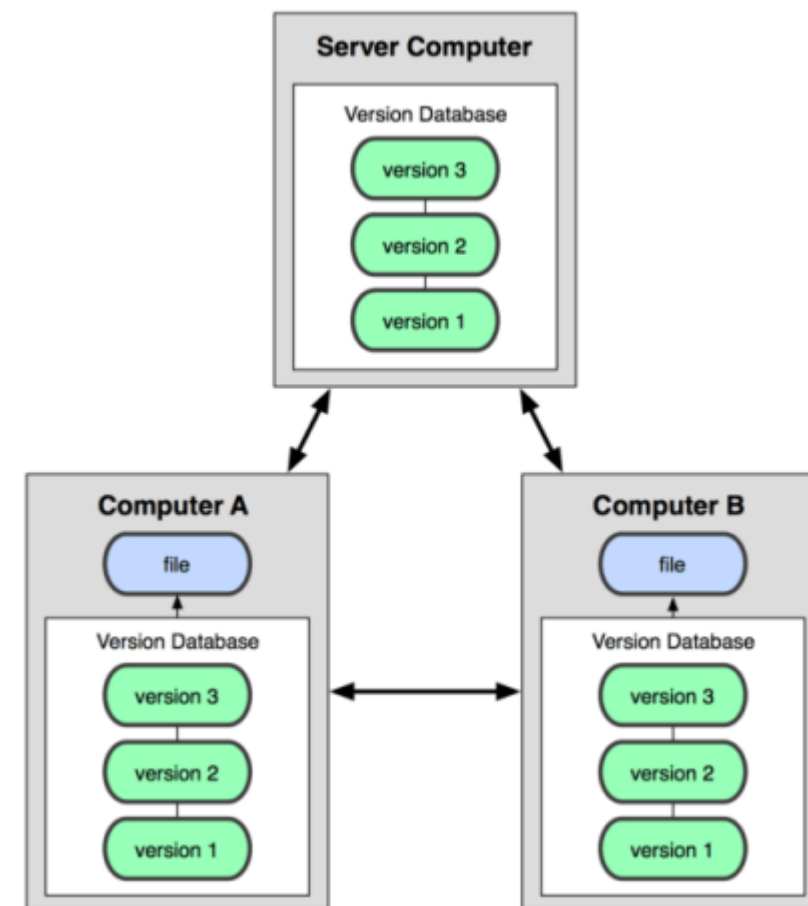


UNIVERSIDADE
CANDIDO
MENDES

EAD ■

■ Sistema de Gerenciamento de Versões

- Existem dois tipos de VCS (Version Control System):
 - DVCS (*Distributed Version Control System*):
Sistemas de Controle de Versões Distribuído
 - Mais conhecidos: Git e Mercurial;
 - Os clientes não só fazem cópias das últimas versões dos arquivos, mas também mantêm cópias completas do repositório;
 - Cada check-out é um backup dos dados do servidor;



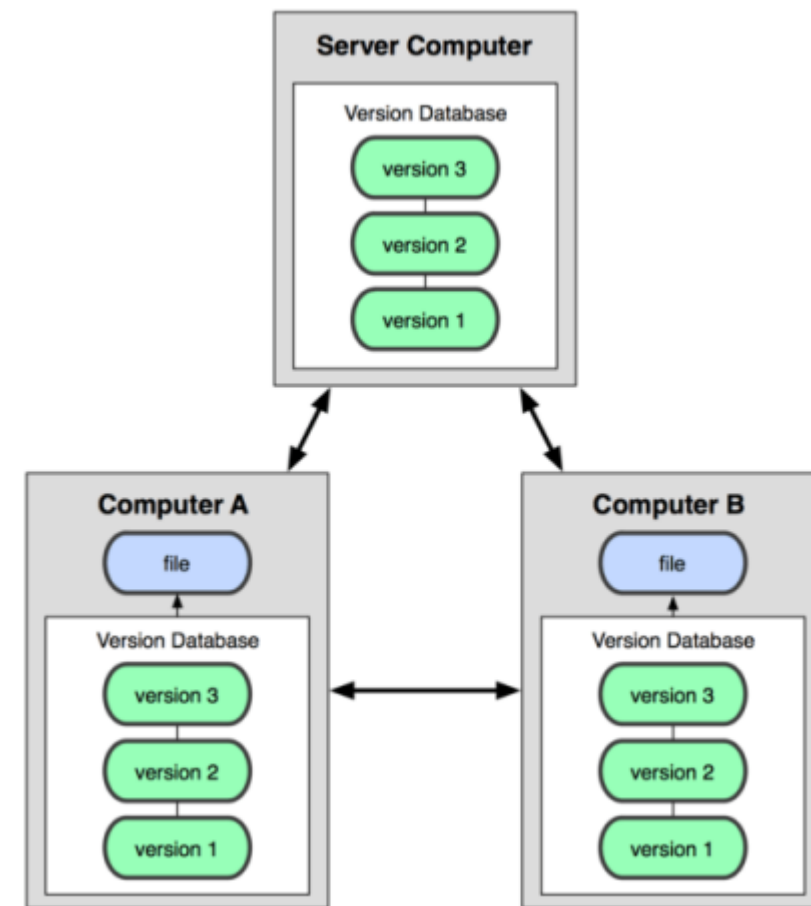
■ Sistema de Gerenciamento de Versões

■ Vantagens do CVCS:

- Se um servidor falha, qualquer um dos repositórios dos clientes pode ser copiado de volta para o servidor e restaurar o projeto;
- Se o servidor ficar fora do ar um repositório de um cliente pode enviar os arquivos para outros computadores;

■ Desvantagens do CVCS:

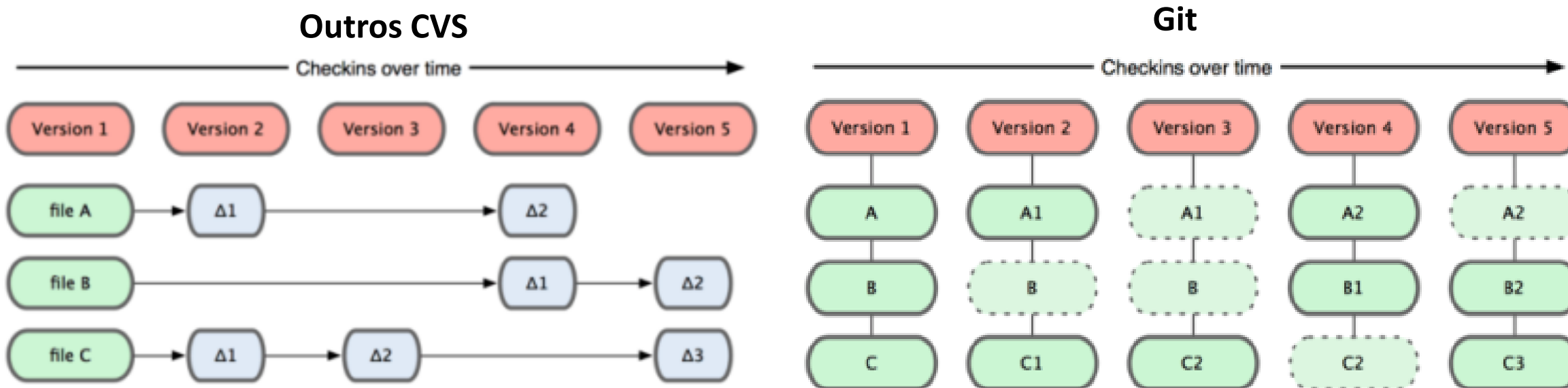
- A administração se torna mais complexa, pois o ADM nem sempre poderá ter o controle do todo;



■ Sistema de Gerenciamento de Versões

■ Controle de Versão com Git:

- O Git funciona diferentemente de outros CVS
- Não armazena os deltas de diferenças das versões, mas sim snapshots;



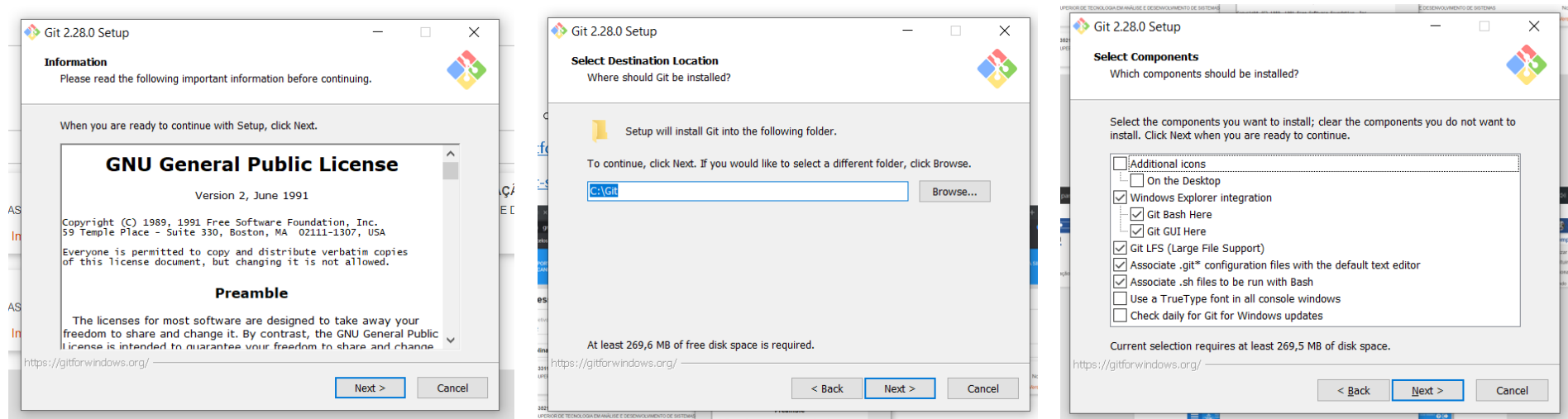
■ Sistema de Gerenciamento de Versões

- A cada commit é armazenado um snapshot de todos os arquivos, naquele momento;
- Se não houver modificações é armazenada somente um link para o arquivo idêntico anterior;
- Antes que o arquivo seja armazenado no Git passa ser referenciado por um checksum;
- Impossível alterar algum arquivo sem que o Git tenha conhecimento;

■ Introdução ao Git

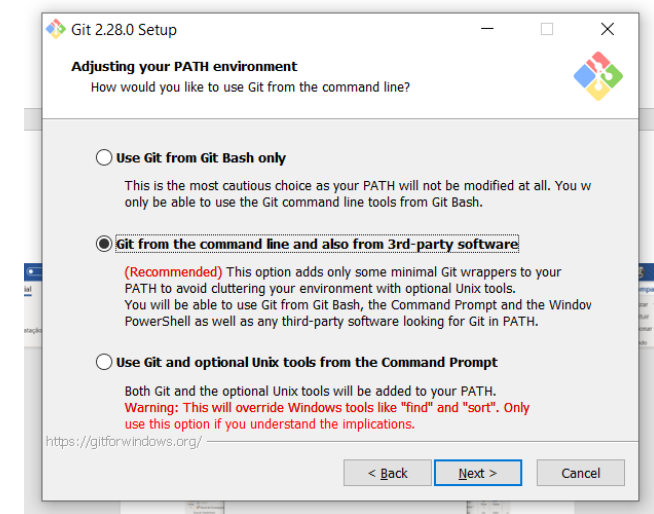
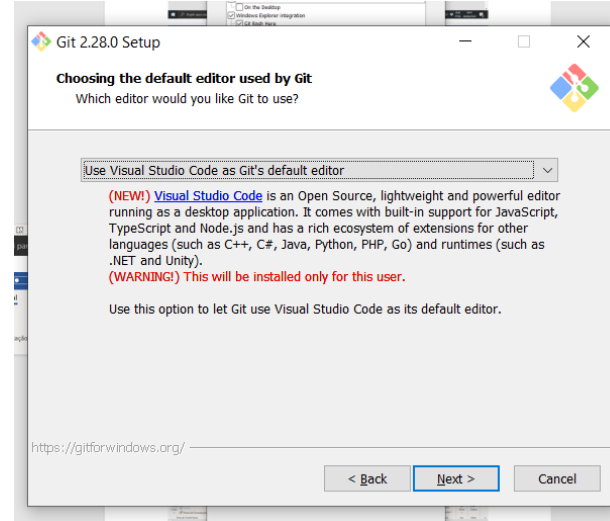
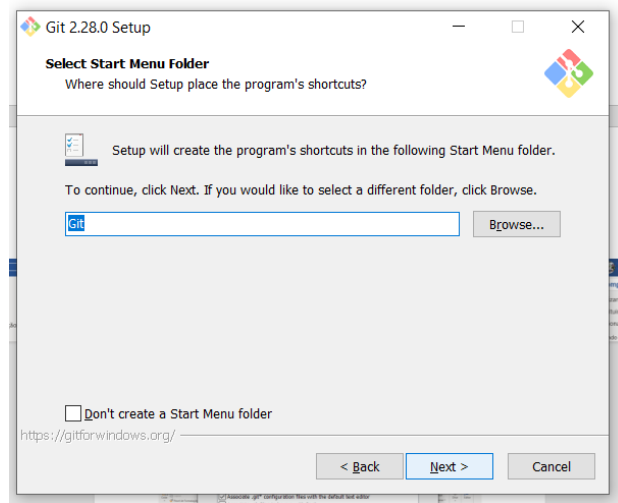
■ Instale o git

- <https://gitforwindows.org/>
- <https://git-scm.com/downloads>



- # Introdução ao Git
- ## Instale o git

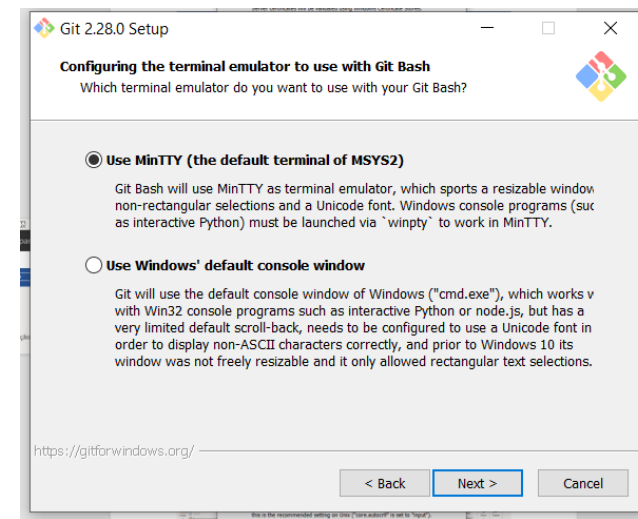
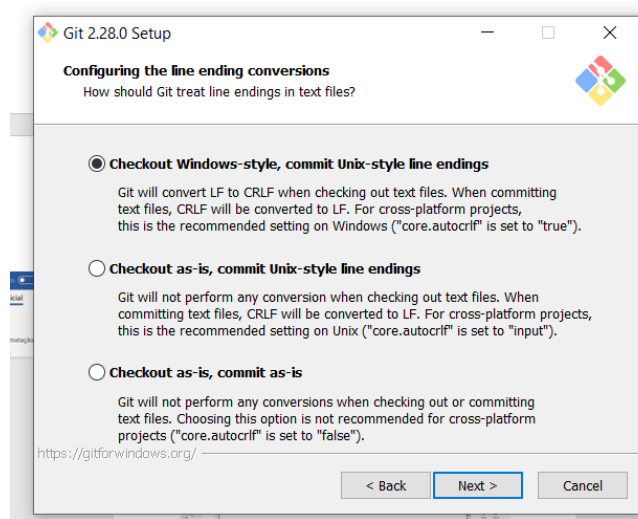
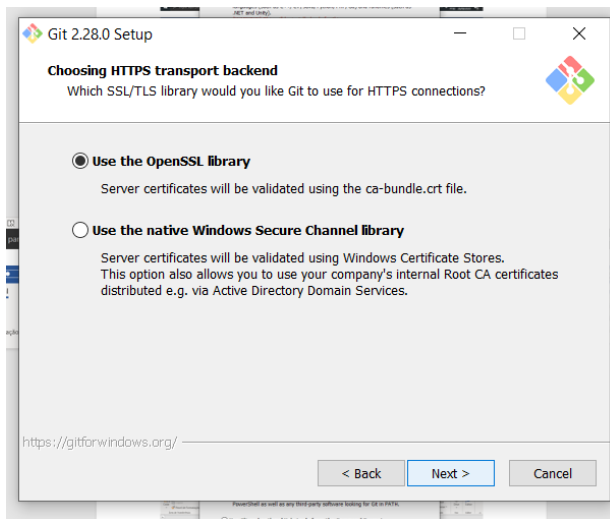
 - <https://gitforwindows.org/>
 - <https://git-scm.com/downloads>



■ Introdução ao Git

■ Instale o git

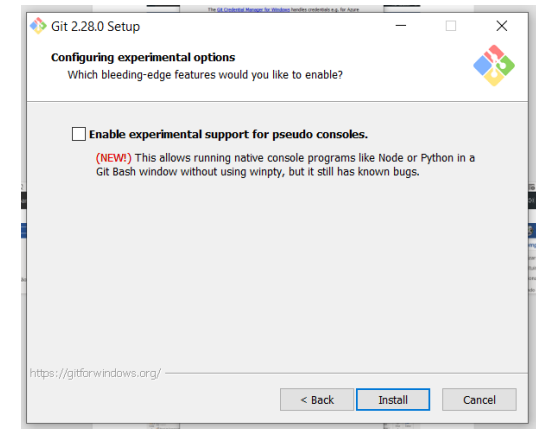
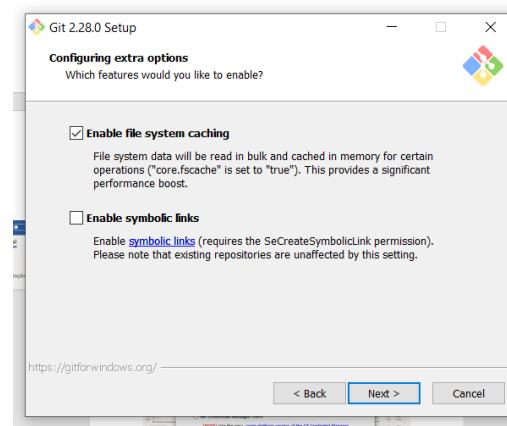
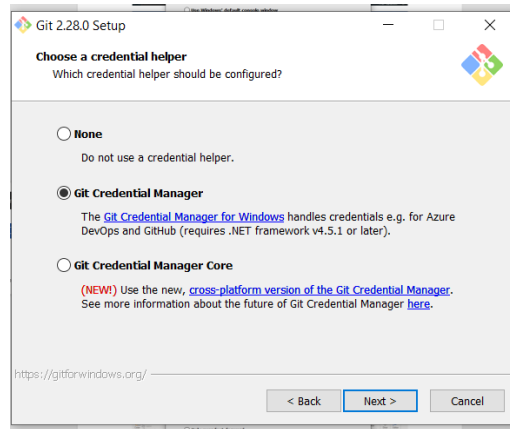
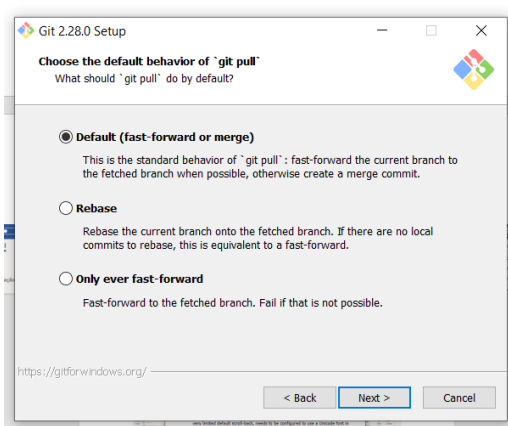
- <https://gitforwindows.org/>
- <https://git-scm.com/downloads>



■ Introdução ao Git

■ Instale o git

- <https://gitforwindows.org/>
- <https://git-scm.com/downloads>



■ Introdução ao Git

- Verificar se o Git está instalado em seu computador e verifica a versão do mesmo:
 - `git -version`
- Pedindo ajuda no git:
 - `git help`
 - `git help [comando]`
- Definir o nome de usuário e endereço de e-mail
 - `git config [--system | --global | --local] user.name <seu nome>`
 - `git config [--system | --global | --local] user.email <seu e-mail>`
- Configurando o editor de texto padrão do Git
 - Para resolução do conflitos
 - `git config [--system | --global | --local] core.editor "code --wait" (para o VSCode)`
- Configurando as informações no terminal
 - `git config --global color.ui <true ou false> (colorido ou monocromático)`

■ Introdução ao Git

- Estrutura de comandos no Git:

- `git [comando] [--flags] [argumentos]`

- `git status --short` (ou utilizar flags curtas: `-s`)

- `git add README.md`

- Iniciar um repositório em um diretório existente:

- `git init`

- Iniciar um repositório em um novo diretório:

- `git init <diretório>`

- Exemplo: `git init projectA`

- Irá criar um diretório chamado `projectA` e então iniciar o gerenciador de versão para este projeto.

- Iniciar um diretório a partir de um repositório remoto (do github) :

- Clonando um repositório existente

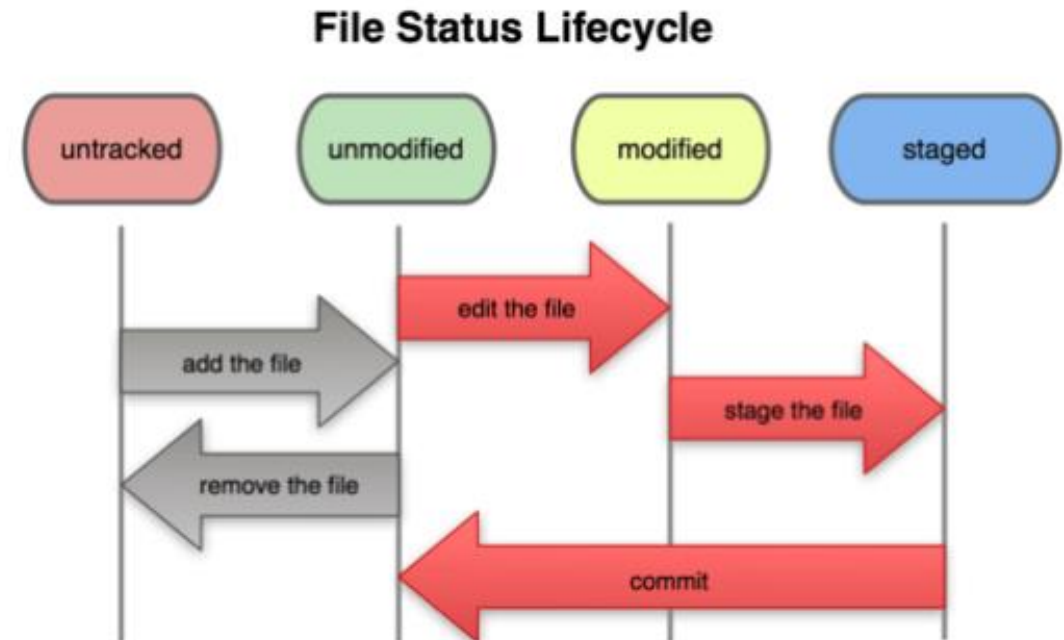
- `git clone git://github.com/<nome_usuario>/<repositorio>.git`

- Clonando um repositório existente dentro de uma pasta:

- `git clone git://github.com/<nome_usuario>/<repositorio>.git <diretório>`

■ Introdução ao Git

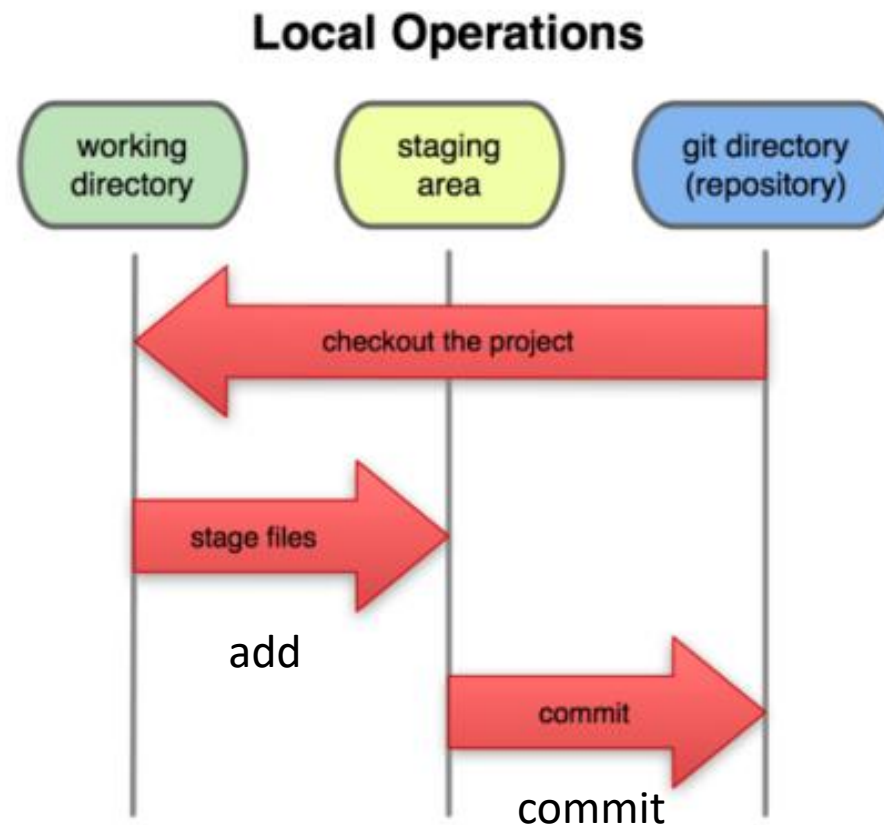
- Ciclo de vida do arquivo:
 - Usuário cria um arquivo chamado README.txt:
 - O arquivo inicialmente fica em um estado de "Não rastreado";
 - O arquivo é adicionado para área de arquivos elencados (staging área):
 - O arquivo passa para o estado de "Pronto para se tornar uma versão" (staged);
 - O usuário então faz uma modificação no arquivo README.txt:
 - O arquivo passa para o estado de "modificado"
 - O usuário novamente adiciona o arquivo na staging área:
 - O arquivo volta a ficar no estado de "Pronto para se tornar uma versão" (staged);
 - O usuário então realiza o commit do arquivo:
 - O arquivo README.txt passa a fazer parte de uma versão



■ Introdução ao Git

■ Os estados do Git:

- Modificado (*modified*):
 - Quando os arquivos sofreram mudanças, mas ainda não foram consolidados na base de dados;
- Preparado (*staged*):
 - Quando os arquivos modificados são marcados, em sua versão atual, para que façam parte do snapshot do próximo commit;
- Consolidado (*committed / commitado*):
 - Quando os arquivos estão seguramente armazenados em sua base de dados local;



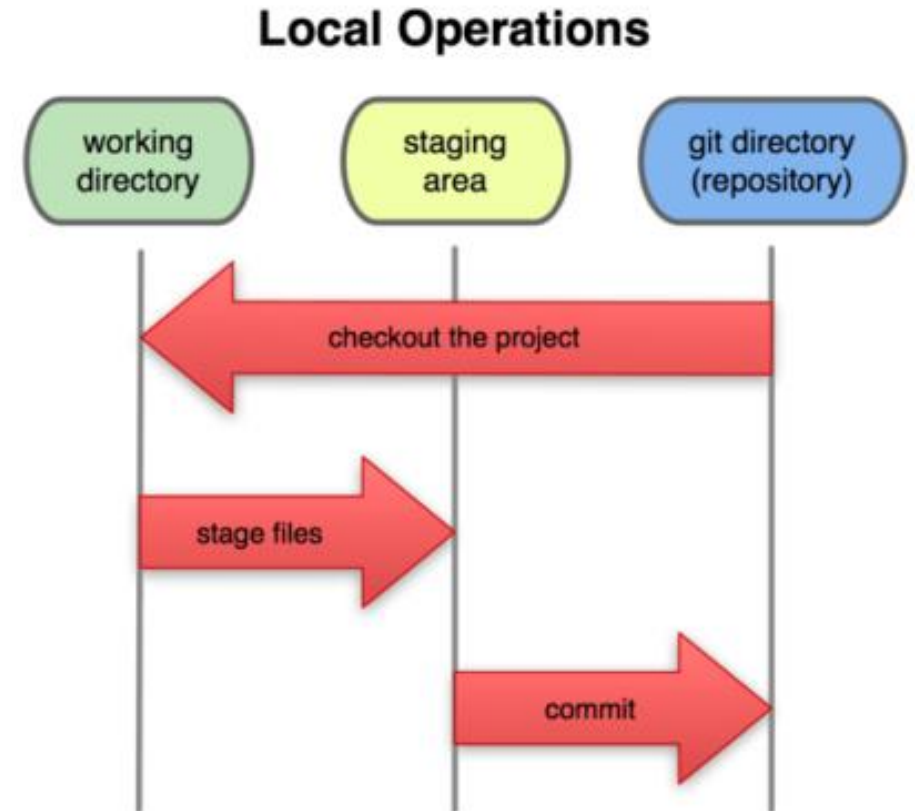
■ Introdução ao Git

- As localizações do Git:
 - Na estação de trabalho:
 - Diretório de Projetos, composto por:
 - Diretório de Trabalho;
 - Área de preparação;
 - Diretório Git (repositório local);
 - Num servidor on-premises ou nuvem:
 - Repositório remoto;
 - Armazena a versão “oficial” do projeto;



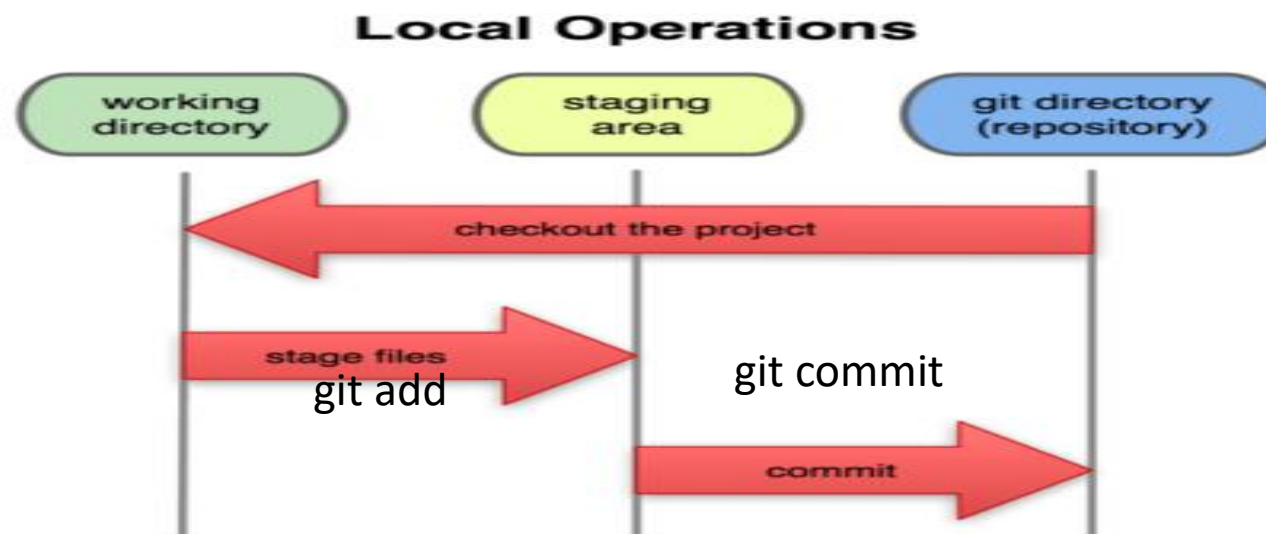
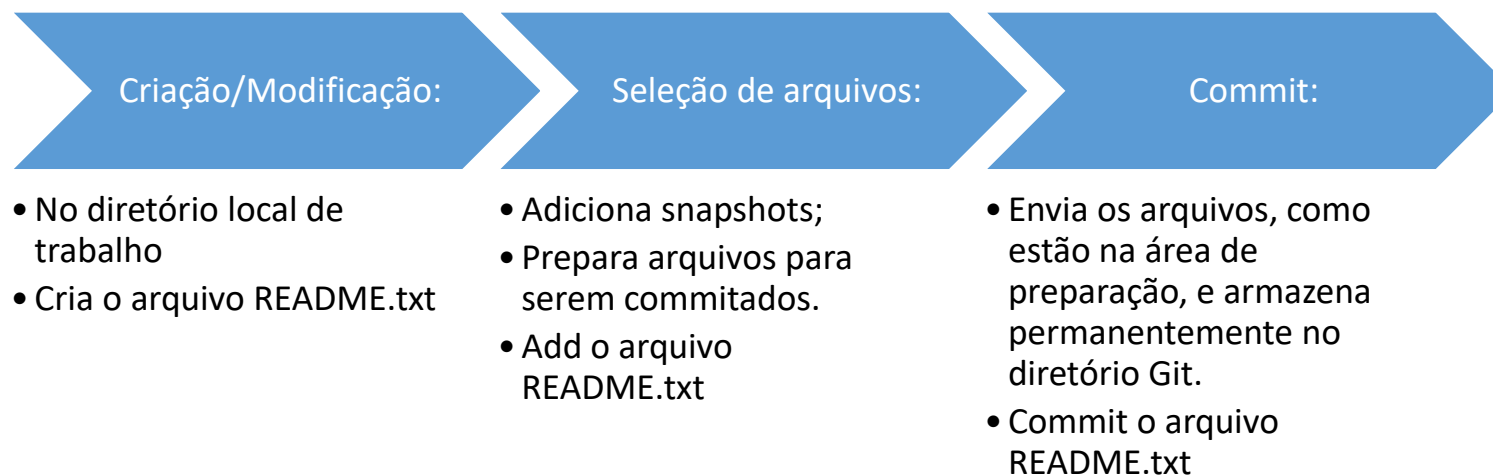
■ Introdução ao Git

- As três seções do Git:
 - Diretório de Trabalho: arquivos são obtidos a partir da base de dados no diretório do Git e colocados em disco para utilização ou modificação;
 - Área de preparação: arquivo contido diretório Git que armazena as informações sobre os arquivos que serão enviados no próximo commit;
 - Diretório Git (repositório): armazena os metadados e o banco de objetos de seu projeto;



■ Introdução ao Git

■ Workflow do Git:



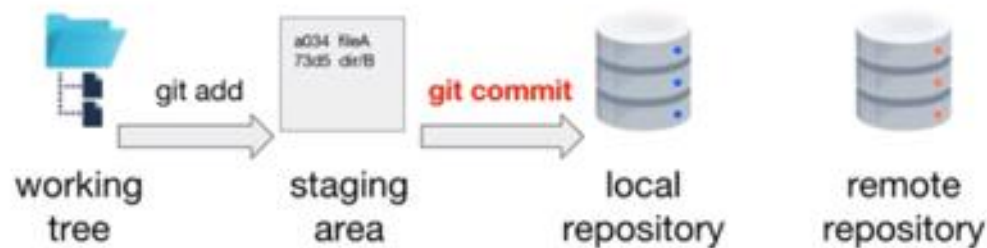
■ Introdução ao Git

- Verificar o status dos arquivos:
 - `git status` (adicione a flag `-s` para short, é uma boa visualização quando temos vários arquivos)
- Iniciar o monitoramento dos arquivos:
 - Adiciona o arquivo na staging area
 - `git add <arquivo>` (add um arquivo específico)
 - `git add *.py` (add todos os arquivos .py)
 - `git add .` (add todos os arquivos)
 - `Git add --all` (add todos os arquivos)
 - `git add <diretório>/*.txt` (ex.: `git add dirA/*.txt` - adiciona todos os arquivos txt dentro do diretório dirA)



■ Introdução ao Git

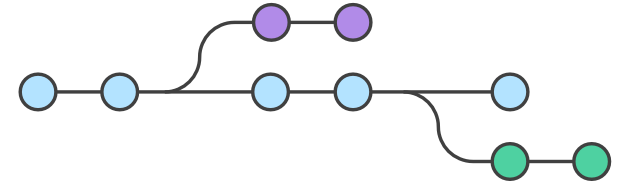
- Verificar mudanças selecionadas ou não selecionadas:
 - Verificar exatamente o que foi alterado e não está na sua staging area
 - `git diff`
 - Ao adicionar na staging area - `git add` - o `git diff` não retornará nada
- Verificar mudanças nos arquivos que estão na staging area
 - `git diff --staged`
- Commitar mudanças para o repositório:
 - `git commit` (irá abrir o seu editor padrão para inserir a mensagem, após salvar o arquivo estará commitado)
 - `git commit -m "mensagem"`
 - `git commit -a -m "mensagem"` (irá fazer o commit de todos os arquivos que estão sendo "trackeados", mesmo que não tenham sido adicionados na staging area)



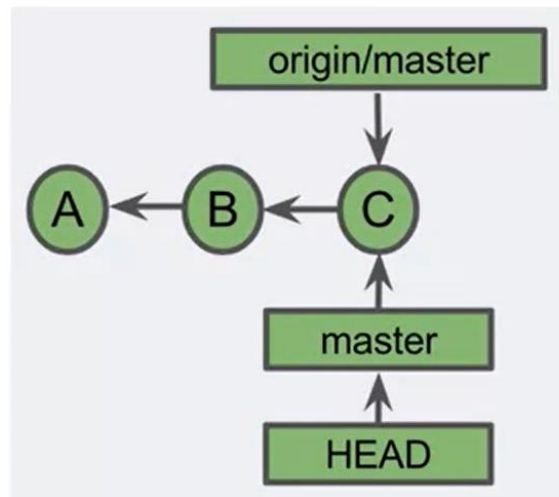
■ Sistema de Gerenciamento de Versões

■ Colaboração entre desenvolvedores:

- Tracking branch: é um branch local que representa um branch remoto
<remoto>/<branch>

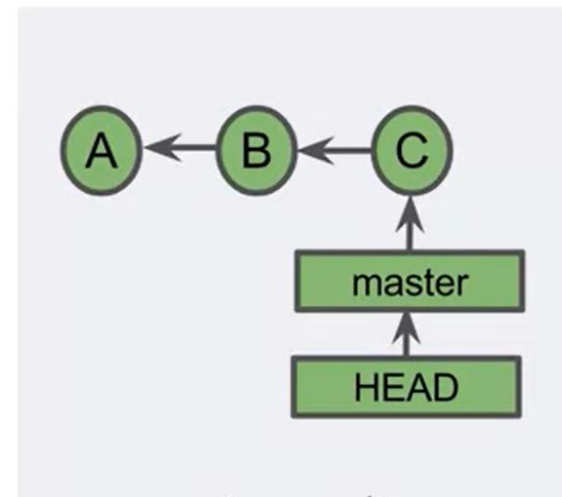


Repositório local



←
clone

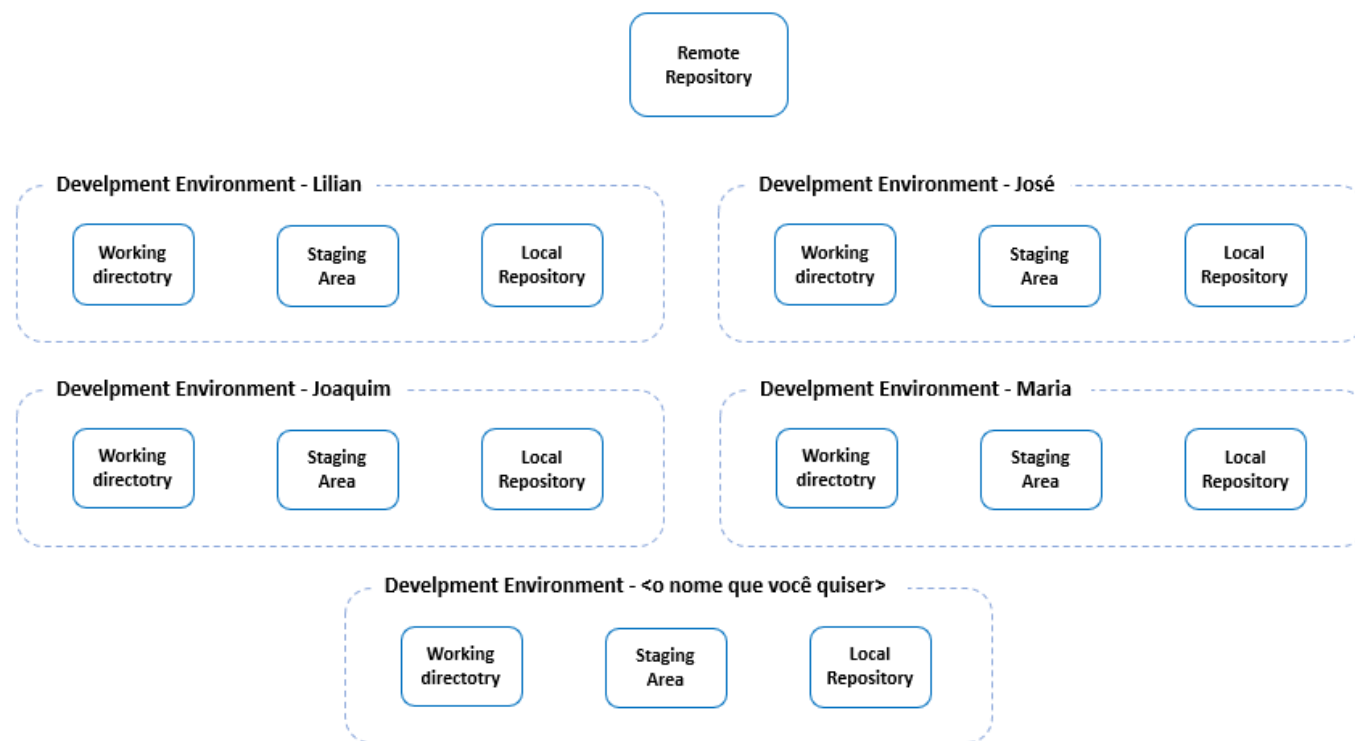
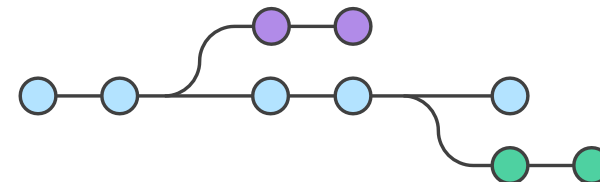
Repositório remoto (origin)



■ Sistema de Gerenciamento de Versões

■ Colaboração entre desenvolvedores:

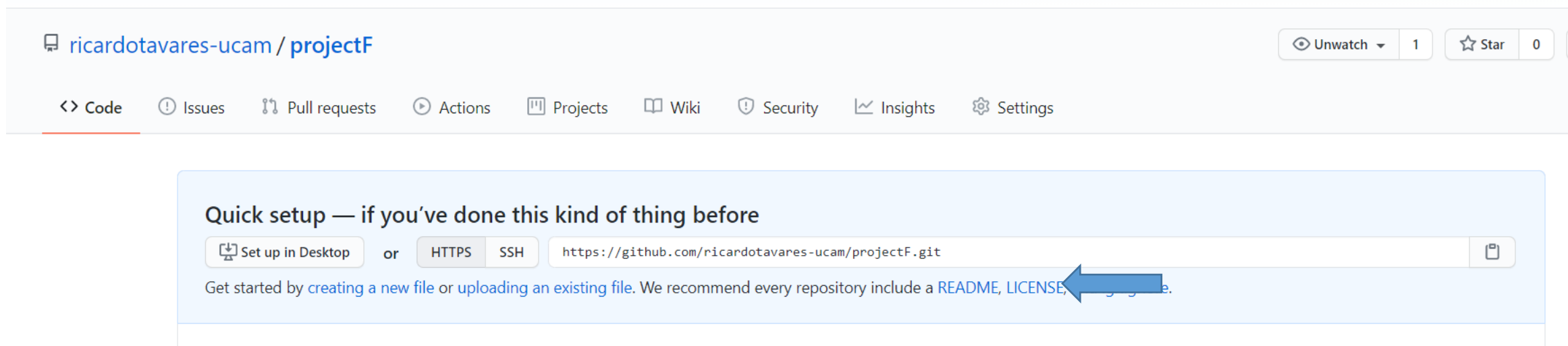
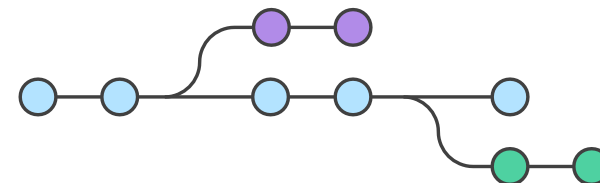
- Tracking branch: é um branch local que representa um branch remoto <remoto>/<branch>



■ Sistema de Gerenciamento de Versões

■ Colaboração entre desenvolvedores:

- Adicionando nosso trabalho local para um repositório remoto:
- Linkando a um repositório remoto no github

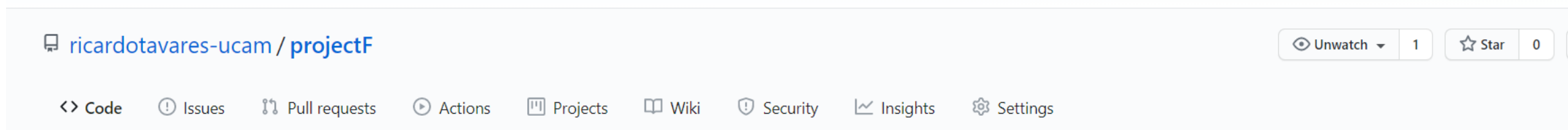
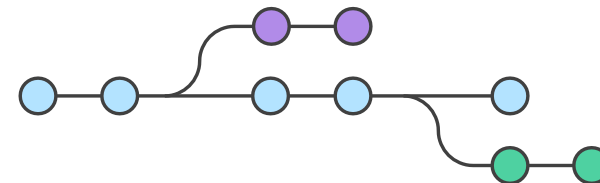


- `git remote add desenvolvimento https://github.com/ricardotavares-ucam/projectF.git`
- `git remote add production https://github.com/ricardotavares-ucam/production.git`
- `git branch desenvolvimento`
- `git push -u desenvolvimento desenvolvimento`
- `git push -u production master`


■ Sistema de Gerenciamento de Versões

■ Colaboração entre desenvolvedores:

- Adicionando nosso trabalho local para um repositório remoto:
- Linkando repositórios remotos ao repositório local



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

 Set up in Desktop

or

HTTPS

SSH

<https://github.com/ricardotavares-ucam/projectF.git>



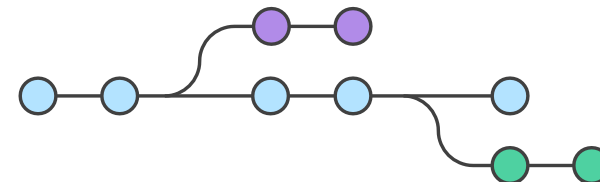
Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [CONTRIBUTING](#) file.

- `git remote add desenvolvimento https://github.com/ricardotavares-ucam/projectF.git`
- `git remote add production https://github.com/ricardotavares-ucam/production.git`
- `git branch desenvolvimento`
- `git push -u desenvolvimento desenvolvimento`
- `git push -u production master`

SEMPRE EXECUTE UM GIT FETCH OU GIT PULL ANTES DE EXECUTAR GIT PUSH

■ Sistema de Gerenciamento de Versões

- Repositórios remotos somente são atualizados localmente com comandos de rede:
 - git clone;
 - git pull;
 - git fetch;
 - git push;
- Utilize o git status para saber se seu repositório local está a frente do repositório remoto (desde a última vez que utilizou um comando de rede):
 - git status;

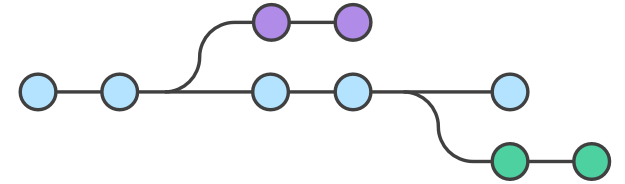


```
$ git commit -m "added feature 2"
[master 63f4add] added feature 2
 1 file changed, 1 insertion(+)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

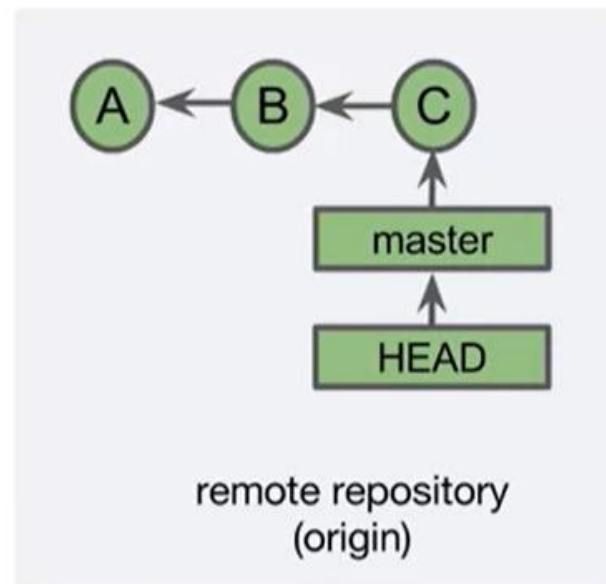
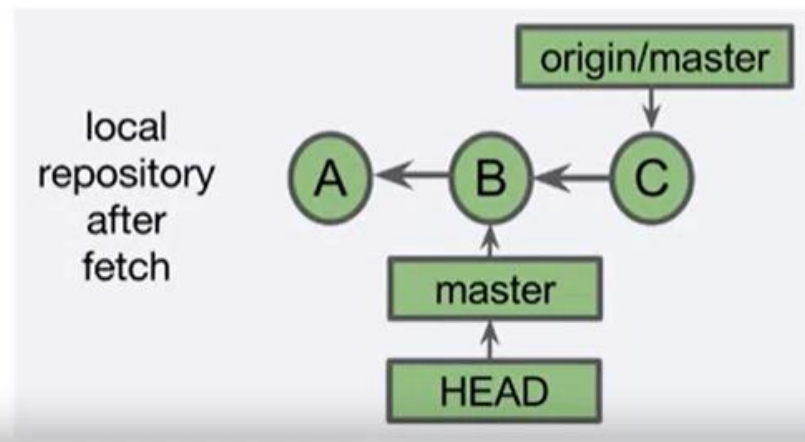
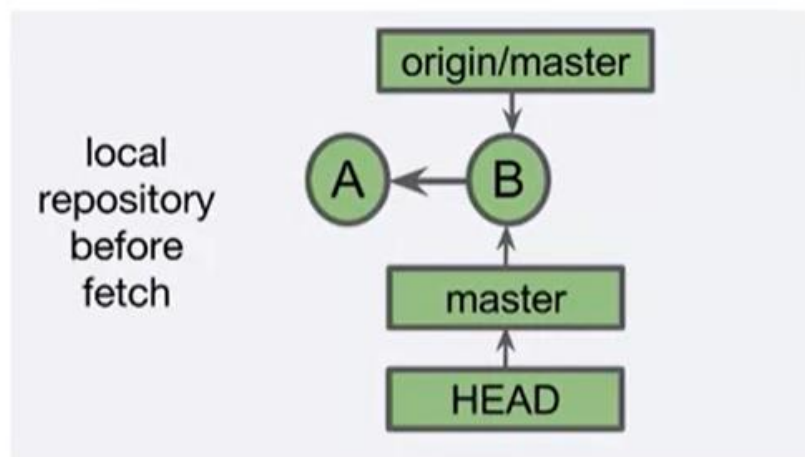
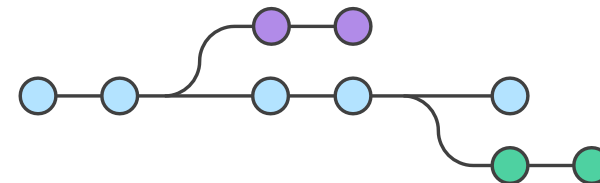
■ Sistema de Gerenciamento de Versões

- Git push: irá enviar as atualizações do seu repositório local para o repositório remoto;
- Git fetch: adiciona novos objetos e referências provenientes do repositório remoto (mas não executa os merges commits);
- Git pull: adiciona os novos objetos e referências provenientes do repositório remoto e executa os merges, quando necessários;



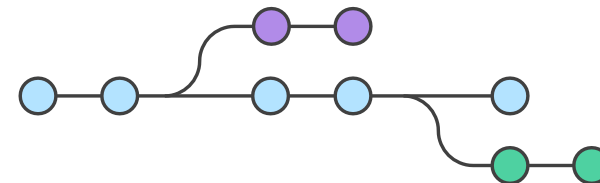
■ Sistema de Gerenciamento de Versões

- Git fetch: adiciona novos objetos e referências provenientes do repositório remoto (mas não executa os merges commits);



■ Sistema de Gerenciamento de Versões

- Após o git fetch:
- Utilize o git status para saber se seu repositório local está a frente do repositório remoto (desde a última vez que utilizou um comando de rede):
 - git status;

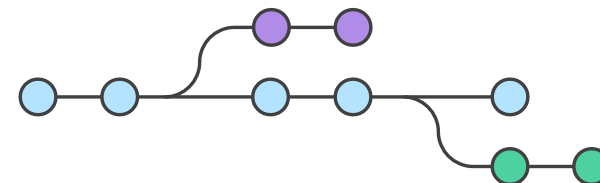


```
$ git fetch
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From https://bitbucket.org/me/projecta
 667fd0d..53d1b4d  master    -> origin/master
$ git status
On branch master
Your branch is behind 'origin/master' by 1 commit, and can be fast-forwarded.
  (use "git pull" to update your local branch)

nothing to commit, working tree clean
```

■ Sistema de Gerenciamento de Versões

- Git pull: adiciona novos objetos e referências provenientes do repositório remoto e combina com o git merge FETCH_HEAD
- É realizada a integração no repositório local no branch ativo;

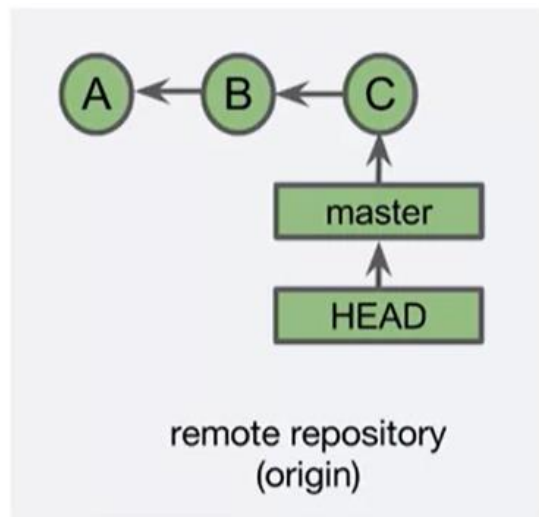
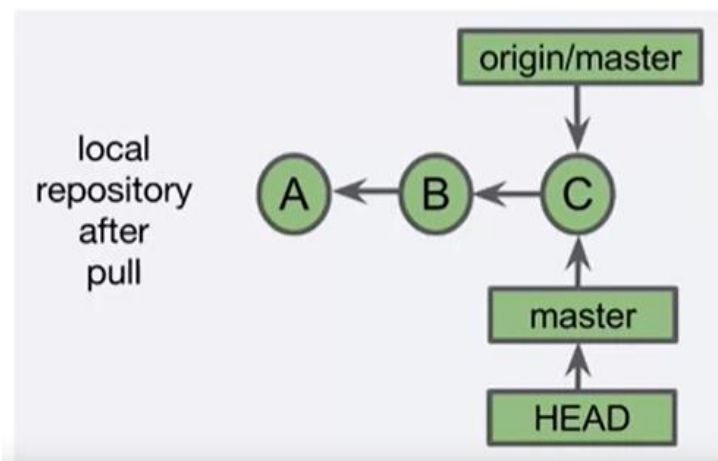
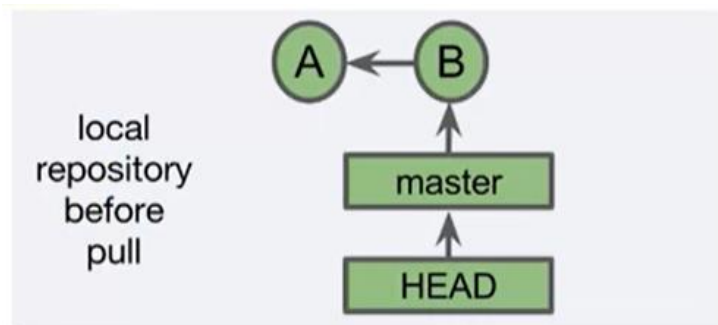
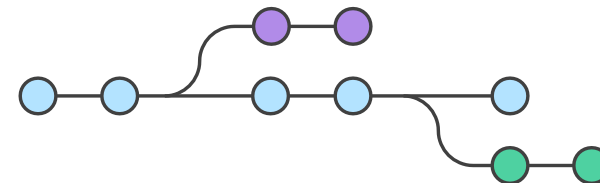


```
$ git pull
Updating a65b42d..482b095
Fast-forward
 fileA.txt | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

- git pull --ff (padrão): executa um fast-forward merge, senão não for possível, irá executar um merge commit;
- git pull --no-ff : sempre executa um merge commit;
- git pull --ff-only: somente fast-forward merge, irá cancelar quando for necessário fazer um merge commit;

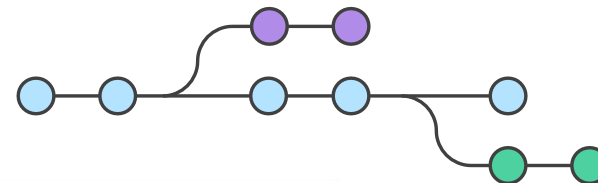
■ Sistema de Gerenciamento de Versões

- git pull (fast-forward)



■ Sistema de Gerenciamento de Versões

- git pull (fast-forward)



```
$ git status
On branch master
Your branch is behind 'origin/master' by 1 commit, and can be
fast-forwarded.
(use "git pull" to update your local branch)
```

```
nothing to commit, working tree clean
```

```
$ git pull
```

```
Updating 667fd0d..53d1b4d
```

```
Fast-forward
```

```
fileA.txt | 1 +
```

```
1 file changed, 1 insertion(+)
```

```
$ git log --oneline
```

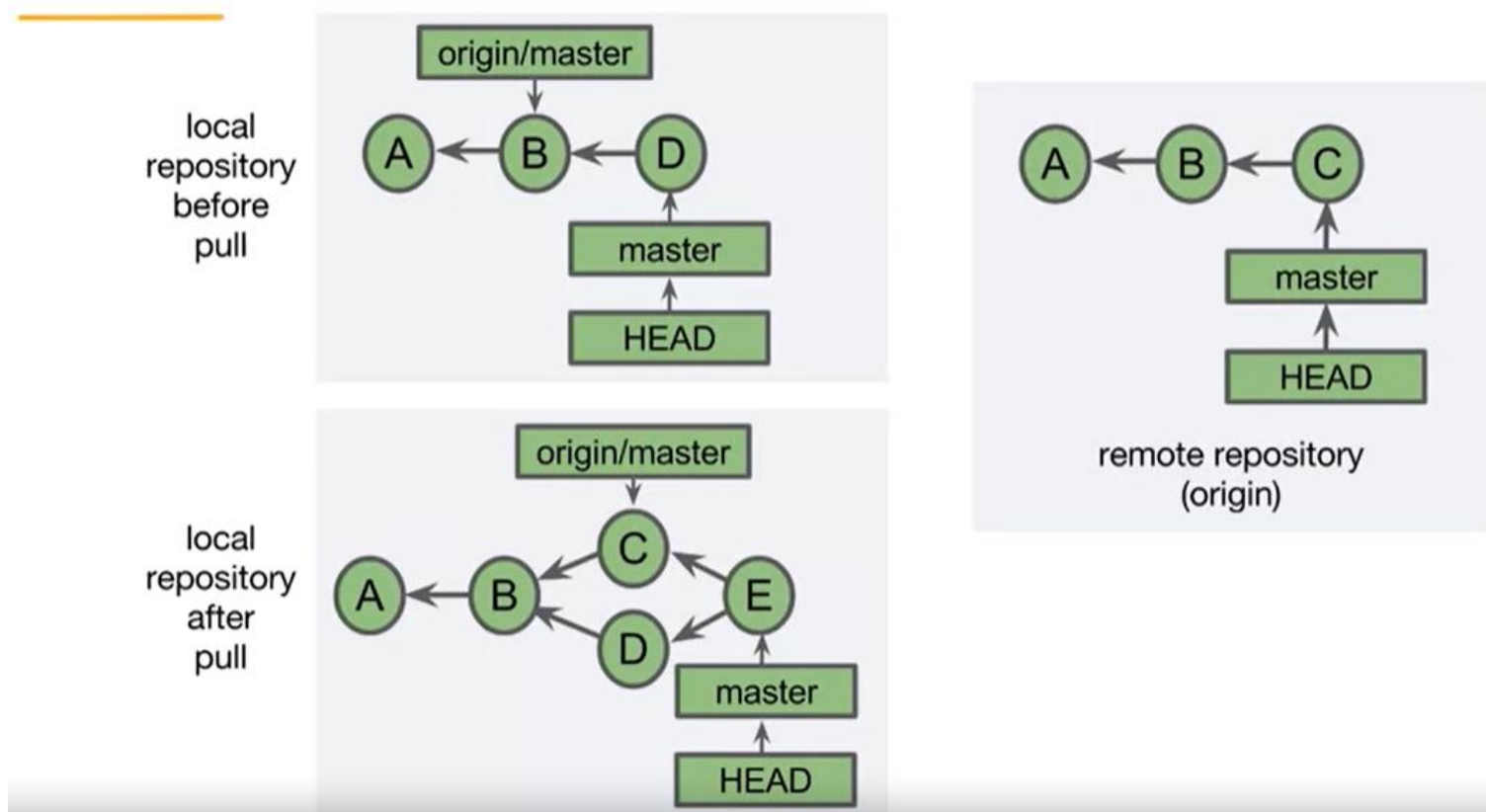
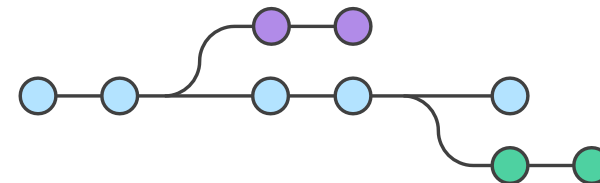
```
53d1b4d (HEAD -> master, origin/master) added feature 3
```

```
667fd0d added feature 2
```

```
5d5e128 initial commit
```

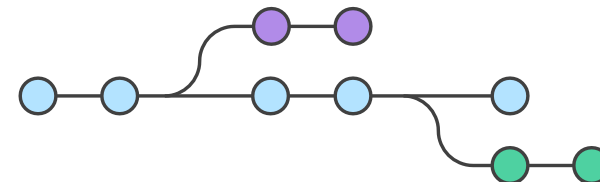
■ Sistema de Gerenciamento de Versões

- git pull (merge commit)



■ Sistema de Gerenciamento de Versões

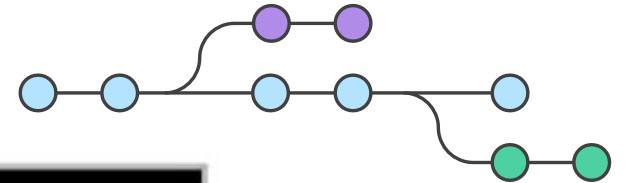
- git pull (merge commit)



```
$ echo "feature4" >> fileA.txt
$ git pull
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From https://bitbucket.org/me/projecta
   53d1b4d..63f4add  master    -> origin/master
Updating 53d1b4d..63f4add
error: Your local changes to the following files would be overwritten by merge:
       fileA.txt
Please commit your changes or stash them before you merge.
Aborting
```

■ Sistema de Gerenciamento de Versões

- git pull (merge commit)



```
$ touch fileC.txt # new file
$ git add fileC.txt
$ git commit -m "added fileC.txt"
[master 6209be3] added fileC.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 fileC.txt
$ git pull
remote: Counting objects: 3, done.
(snip)
From https://bitbucket.org/me/projecta
   c2ccd19..6d5539b  master      -> origin/master
Merge made by the 'recursive' strategy.
   fileA.txt | 1 +
   1 file changed, 1 insertion(+)
$ git log --oneline --graph -4
*    c090487 (HEAD -> master) Merge branch 'master' of https://bitbucket.org/me/projecta
|\
| * 6d5539b (origin/master) added feature 5
* | 6209be3 added fileC.txt
|/
* c2ccd19 added fileB.txt
```


■ DESENVOLVIMENTO WEB COM PYTHON

- Arquitetura MVC (Model-View-Controller):
 - O modelo MVC é um modelo arquitetural de três camadas separadas, a fim de fornecer um nível de abstração relacionados a cada uma das camadas, dividido em uma camada modelo, uma camada de visão e uma camada de controle;
 - Organizar a arquitetura de um sistema complexo em partes menores, ou camadas, em que cada camada possui certa independência uma das outras;
 - Uma arquitetura bem projetada deve atender aos requisitos funcionais e não funcionais de um sistema de informação e ser flexível para atender a requisitos voláteis;

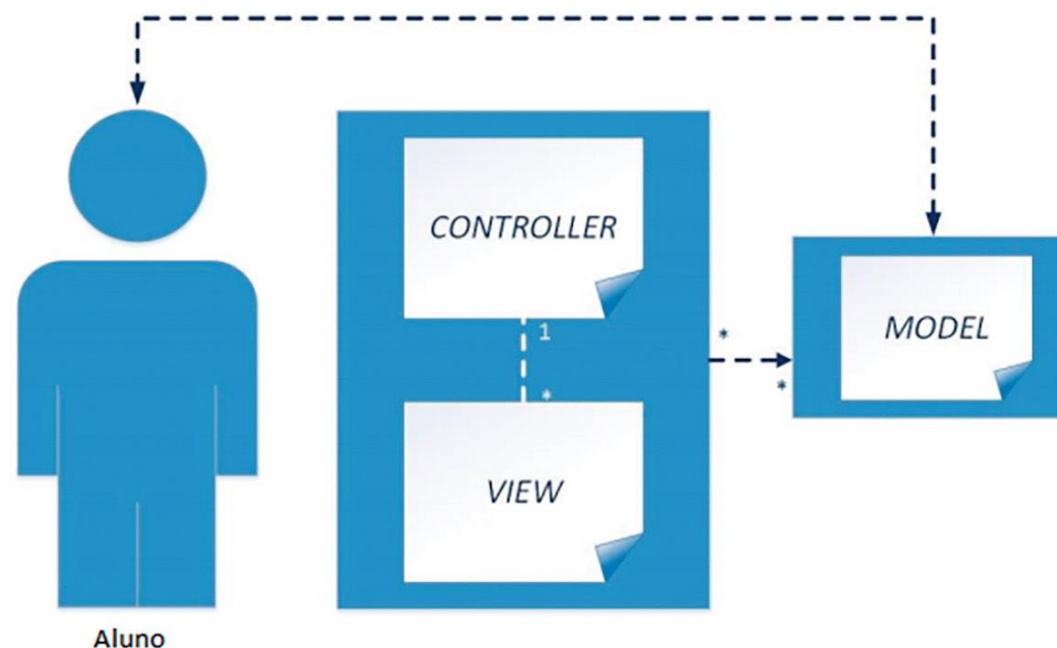
■ DESENVOLVIMENTO WEB COM PYTHON

- Arquitetura MVC (Model-View-Controller):
 - Nas camadas de abstração do MVC, as camadas mais altas interagem com as camadas de abstração mais baixas;
 - Mudanças em uma camada mais baixa, que não afetem sua interface, não implicarão em mudanças nas camadas mais superiores;
 - Do mesmo modo, as mudanças em uma camada mais alta, que não impliquem na criação/modificação de serviços em uma camada mais baixa, não afetarão as camadas mais inferiores;
 - Em resumo, há interdependência entre as camadas, mas cada uma delas possui graus de independência uma das outras;

■ DESENVOLVIMENTO WEB COM PYTHON

■ Arquitetura MVC (Model-View-Controller):

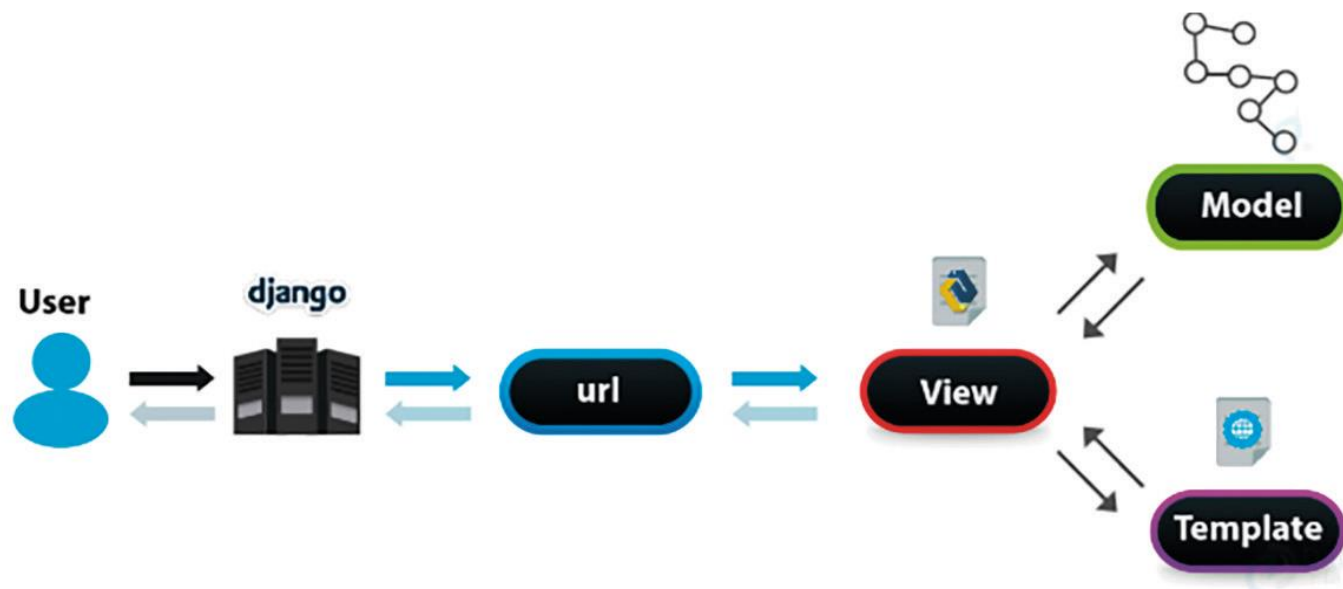
- No componente Model, temos o objeto da aplicação (modelo de negócio) ou lógica de como funciona o sistema (registrando os alunos e lançando as notas);
- No componente View, temos a interface de visualização do usuário, ou seja, a parte que o usuário (aluno, professor, pai do aluno etc.) utiliza para interagir com o sistema de informação;
- No componente Controller, temos o responsável por trabalhar as entradas de dados da View e as reações (de acordo com as entradas) do modelo de negócio (Model) do sistema;



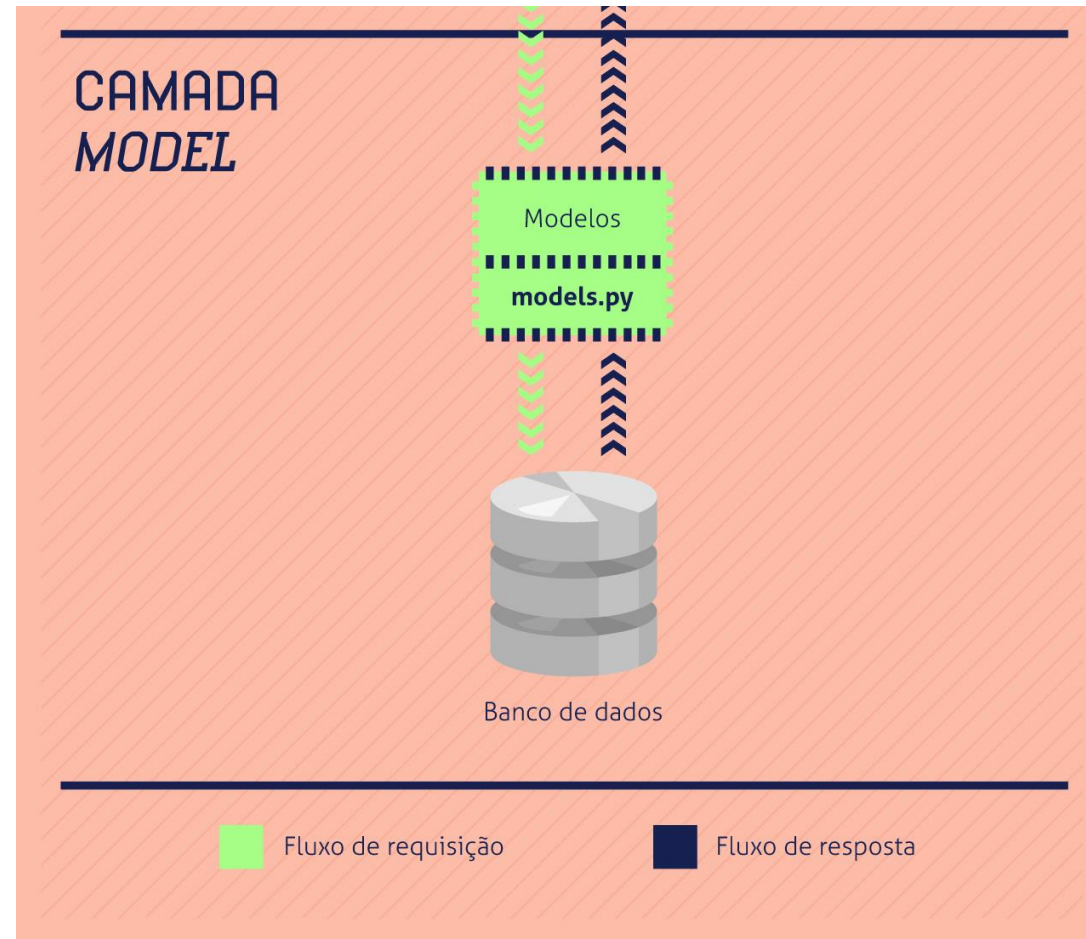
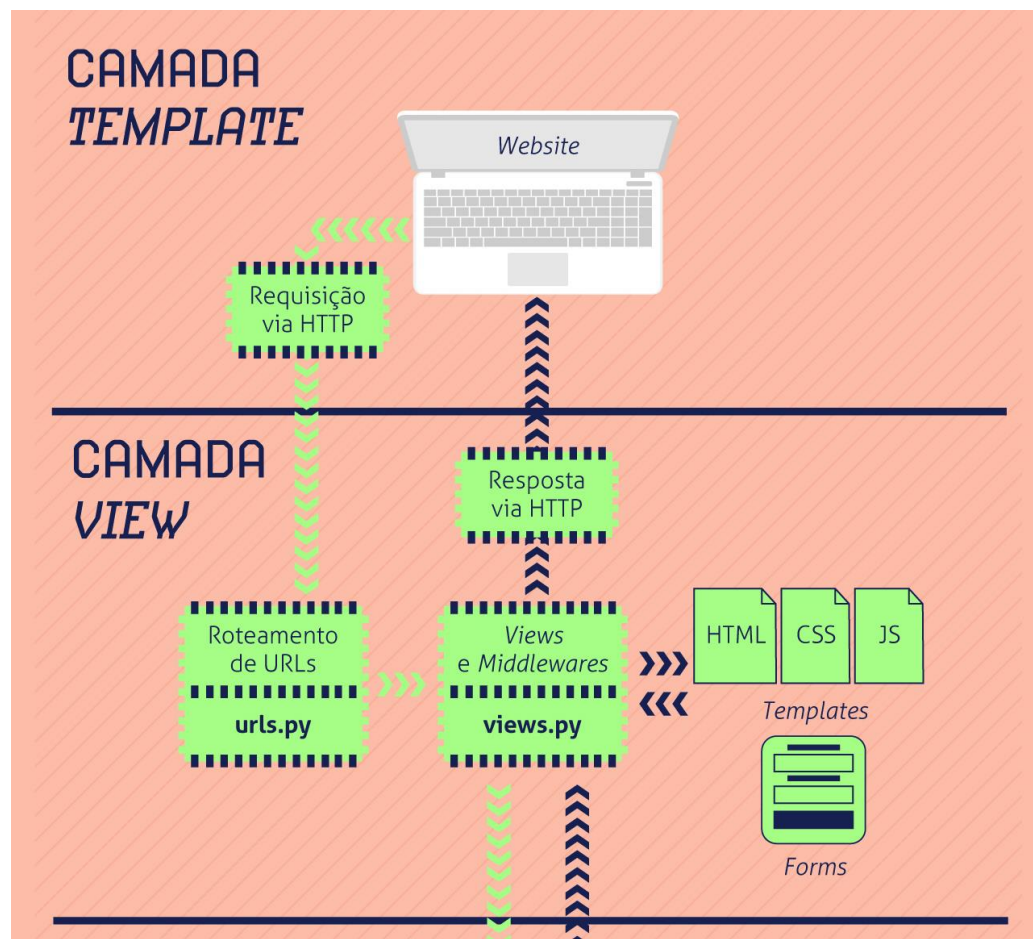
■ DESENVOLVIMENTO WEB COM PYTHON

■ Django e sua arquitetura MVT:

- Segue um modelo de arquitetura model-template-view (MTV);
- Model: responsável pelo modelo de negócios da aplicação, onde ocorre toda sua interação com o banco de dados utilizado; Realiza as regras de negócio da aplicação;
- Template: onde ocorre a visualização intuitiva, que é a interface do usuário com o site desenvolvido em Django; Responsável pela interface com o usuário, realizando requisições e recebendo respostas via HTTP;
- View: responsável pelo controle das requisições. Realiza o controle das requisições feitas e orquestra as telas de visualização com o modelo de negócio;



■ DESENVOLVIMENTO WEB COM PYTHON



■ DESENVOLVIMENTO WEB COM PYTHON

■ Instalação do Django (utilizando conda):

- `conda create --name <nome_do_ambiente> python=3.7` (em frente)
- `conda install -c anaconda django` (ou `pip install django`)
- `django-admin --version` (verificar se o Django está devidamente instalado)
- Para checar outras funcionalidades: `django-admin help <subcommand>`

■ Criação de um novo projeto no Django:

- `django-admin.py startproject <nome_do_projeto>`
- Passamos para o comando `startproject`, que criará um novo projeto com toda a estrutura de diretórios para podermos começar a desenvolver;

■ Verificando se o projeto foi criado e está tudo correto:

- Acesse o diretório do projeto `<nome_do_projeto>`;
- `python manage.py runserver`
- acesse `http://localhost:8000` ou `http://127.0.0.1:8000/`

■ DESENVOLVIMENTO WEB COM PYTHON

<http://127.0.0.1:8000/> ou <http://localhost:8000/>

django

[View release notes for Django 3.1](#)



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.



Django Documentation
Topics, references, & how-to's



Tutorial: A Polling App
Get started with Django



Django Community
Connect, get help, or contribute

■ DESENVOLVIMENTO WEB COM PYTHON

■ Estrutura de pastas no Django:

- <nome_do_projeto>/settings.py: Arquivo que contém configurações do projeto, como configurações do banco de dados, aplicativos instalados, configuração de arquivos estáticos, dentre outros;
- <nome_do_projeto>/urls.py: Arquivo de configuração de rotas (ou URLConf). É nele que é configurado quem irá responde a URL passada;
- <nome_do_projeto>/wsgi.py: Aqui configuramos a interface entre o servidor de aplicação e nossa aplicação Django.
- manage.py: Arquivo gerado automaticamente pelo Django que expõe comandos importantes para manutenção da nossa aplicação.

■ DESENVOLVIMENTO WEB COM PYTHON

■ Criando um novo app no projeto:

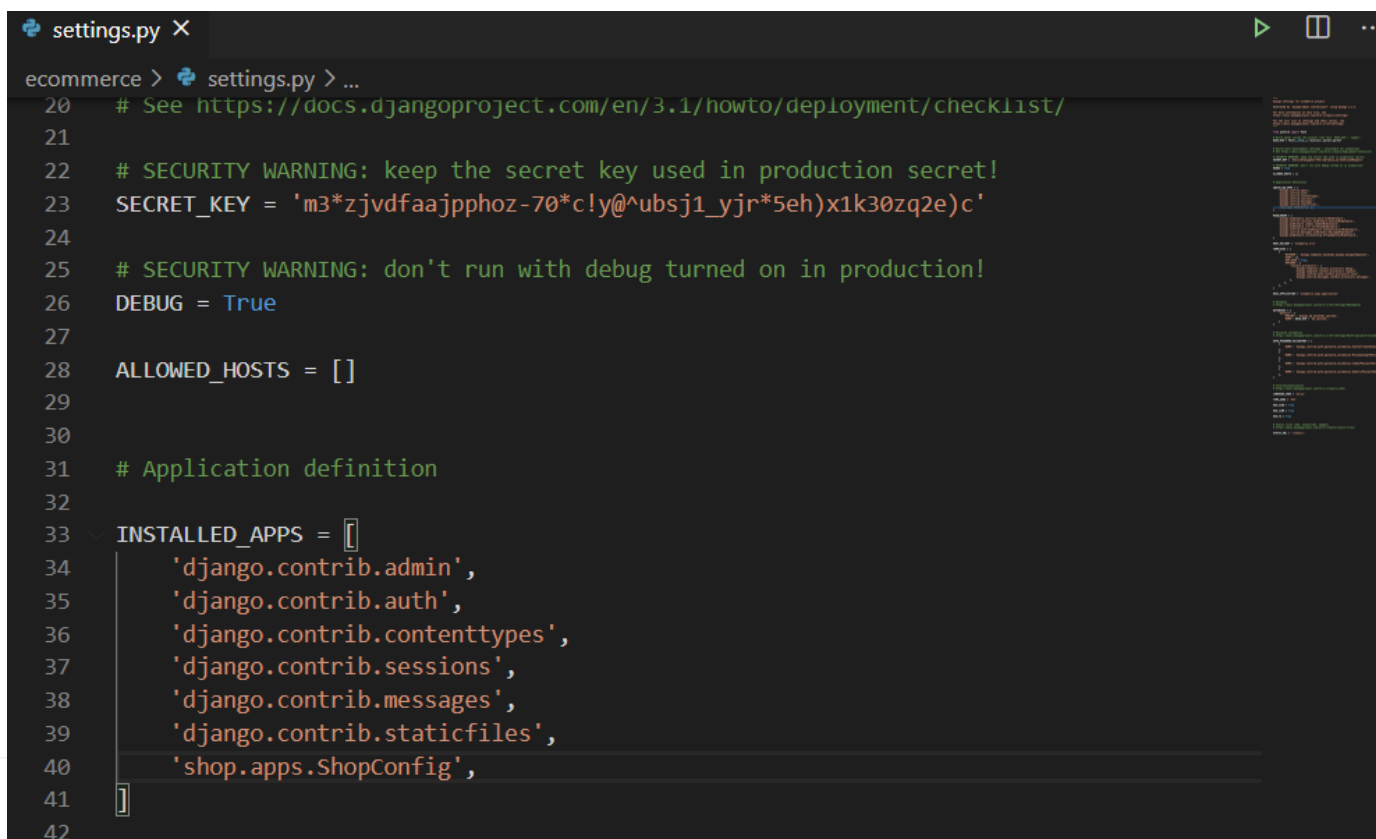
- Um app é uma aplicação web responsável pela realização de tarefas determinadas pelo programador;
- Um projeto é uma coleção de arquivos de configuração e aplicações web que resultam num sistema web;
 - Exemplo: um sistema de vendas online pode possuir um app vendas e um app carrinho_de_compras.
- Importante é que um app pode estar presente em vários projetos;
 - DRY -> Don't repeat yourself
- Comando: `django-admin.py startapp <nome_do_app>`
- Namespace:
 - É o que define de onde a rota pertence (a qual app aquela rota foi criada)
 - São essenciais para ter um código fácil de ler e de manter
 - Crie subpastas com o nome do app dentro de: templates e static

■ DESENVOLVIMENTO WEB COM PYTHON

■ Adicionando um novo app no projeto:

■ Precisamos adicionar o app ao projeto original:

- Em settings.py (na pasta do projeto) -> INSTALLED_APPS -> adicione:
<nome_do_app>.apps.<nome_do_app>Config (ou somente o nome do app – ex: 'shop',) Não esqueça da virgula no final



```
settings.py X
ecommerce > settings.py > ...
20 # See https://docs.djangoproject.com/en/3.1/howto/deployment/checklist/
21
22 # SECURITY WARNING: keep the secret key used in production secret!
23 SECRET_KEY = 'm3*zjvdfaaajpphoz-70*c!y@^ubsj1_yjr*5eh)x1k30zq2e)c'
24
25 # SECURITY WARNING: don't run with debug turned on in production!
26 DEBUG = True
27
28 ALLOWED_HOSTS = []
29
30
31 # Application definition
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     'shop.apps.ShopConfig',
41 ]
42
```

■ DESENVOLVIMENTO WEB COM PYTHON

■ Hello Word:

- Antes de fazer o migration e iniciar a camada models do django, vamos fazer um Hello World para entender como funcionam as urls e views do Django.
- 1) crie função para gerenciar as views e chamada de templates;
- 2) crie um arquivo urls.py (dentro do app) para gerenciar as urls do app;
- 3) inclua o arquivo urls do app no arquivo urls do projeto, dessa forma linkamos o app ao projeto;

```
manage.py  views.py  X
shop > views.py > index
1  from django.shortcuts import render
2
3  # inclua a seguinte linha
4  from django.http import HttpResponse
5
6  # Create your views here.
7
8  # A partir desse ponto vamos criar funções responsáveis por coordenar
9  # a apresentação de views e templates do sistema web
10
11 def index(response):
12     return HttpResponse('<h1>Olá turma de desenvolvimento web</h1>')
```

■ DESENVOLVIMENTO WEB COM PYTHON

■ Hello Word:

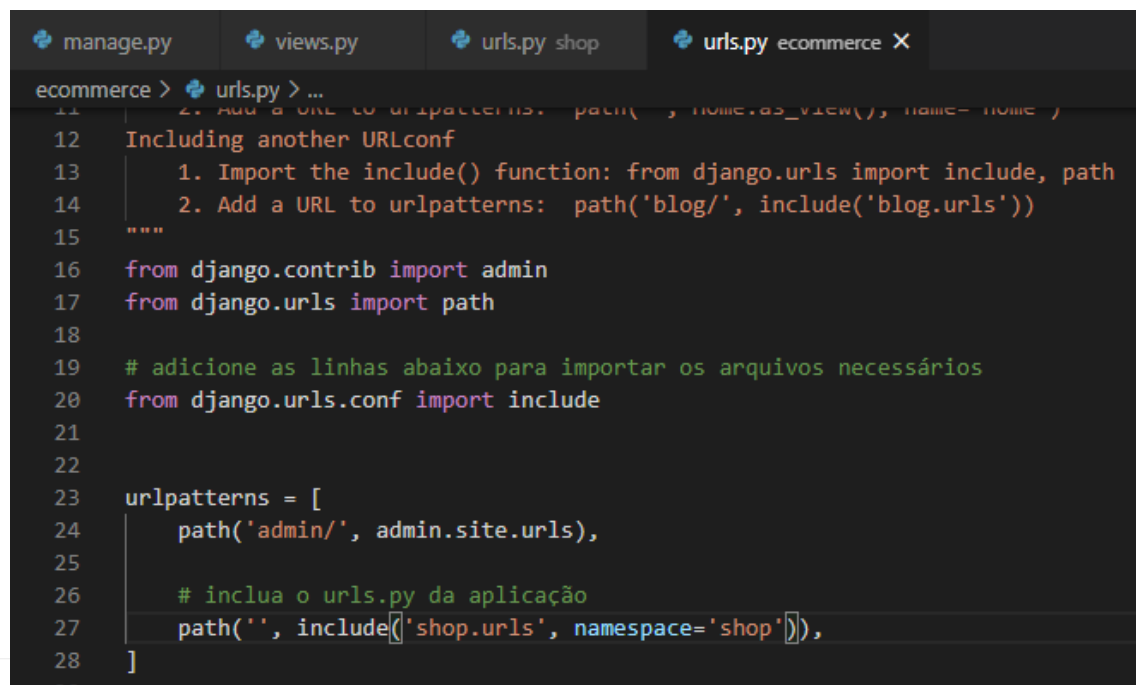
- Antes de fazer o migration e iniciar a camada models do django, vamos fazer um Hello World para entender como funcionam as urls e views do Django.
- 1) crie função para gerenciar as views e chamada de templates;
- 2) crie um arquivo urls.py (dentro do app) para gerenciar as urls do app;
- 3) inclua o arquivo urls do app no arquivo urls do projeto, dessa forma linkamos o app ao projeto;

```
manage.py  views.py  urls.py shop X  urls.py ecommerce
shop > urls.py >...
1  #insira a linhas abaixo para realizar as importações necessárias
2  from django.urls import path
3  from . import views
4
5  # defina o nome do app como namespace
6  # logo, sempre que um namespace for passado no gerenciador de url
7  # saberá de qual aplicativo está tratando
8  # e qual arquivo urls deverá buscar o roteamento de urls
9  app_name = 'shop'
10
11 # urlpatterns irá conter a lista de roteamentos de URLs
12 urlpatterns = [
13     path('', views.index, name='index'),
14     path('page2/', views.page2, name='page2'),
15     path('<int:id>', views.getId, name='getId'),
16     path('page2/<int:id>', views.page2, name='getId_page2'],
17 ]
18
```

■ DESENVOLVIMENTO WEB COM PYTHON

■ Hello Word:

- Antes de fazer o migration e iniciar a camada models do django, vamos fazer um Hello World para entender como funcionam as urls e views do Django.
- 1) crie função para gerenciar as views e chamada de templates;
- 2) crie um arquivo urls.py (dentro do app) para gerenciar as urls do app;
- 3) inclua o arquivo urls do app no arquivo urls do projeto, dessa forma linkamos o app ao projeto;



```
ecommerce > urls.py > ...
11 2. Add a URL to urlpatterns: path('...', home.as_view(), name=home)
12 Including another URLconf
13 1. Import the include() function: from django.urls import include, path
14 2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
15 """
16 from django.contrib import admin
17 from django.urls import path
18
19 # adicione as linhas abaixo para importar os arquivos necessários
20 from django.urls.conf import include
21
22
23 urlpatterns = [
24     path('admin/', admin.site.urls),
25
26     # inclua o urls.py da aplicação
27     path('', include('shop.urls', namespace='shop')),
28 ]
29
```



UNIVERSIDADE
CANDIDO
MENDES

EAD ■

