# Comandos projeto django

Python3 manage.py pip3

(ls) = O comando "ls" é uma abreviação para "list" e é usado no terminal do Linux para exibir uma lista dos arquivos e diretórios presentes no diretório atual. Quando você executa o comando "ls" sem nenhum argumento, ele exibe apenas os nomes dos arquivos e diretórios. Para exibir mais informações sobre esses arquivos e diretórios, você pode usar argumentos opcionais com o comando "ls", como "ls -l" que exibe mais detalhes sobre os arquivos e diretórios, incluindo permissões, proprietários e datas de modificação.

(**ls -l ) =** O comando "ls -l" no terminal do Linux lista o conteúdo do diretório atual em formato de lista longa, mostrando detalhes como permissões de arquivos, proprietário, grupo, tamanho, data de modificação e nome do arquivo ou diretório. É uma maneira mais detalhada de visualizar os arquivos e diretórios presentes no diretório atual.

( **mkdir) =**O comando "mkdir" no terminal Linux é usado para criar um diretório (pasta) no sistema de arquivos. Por exemplo, se você quiser criar um diretório chamado "docs", pode digitar "mkdir docs" no terminal e o diretório será criado no local em que você estiver atualmente.

(**touch) =** O "touch" é um comando utilizado em sistemas Linux para criar um arquivo vazio ou atualizar o horário de modificação de um arquivo existente. Ele é usado principalmente para fins de organização, como criar arquivos temporários ou atualizar datas de arquivos, sem a necessidade de abrir diretamente e editar os arquivos. O comando mais básico é simplesmente "touch nome\_do\_arquivo" para criar um novo arquivo vazio ou atualizar a data do arquivo existente.

(**apt) =** O comando "apt" é uma ferramenta utilizada no sistema operacional Linux para gerenciar pacotes de software. Ele permite instalar, atualizar e remover programas facilmente a partir do terminal.  
  
Algumas das funções possíveis com o comando "apt" são:  
  
- Instalar um programa: `sudo apt-get install nome\_do\_programa`  
- Atualizar os pacotes já instalados: `sudo apt-get update` seguido de `sudo apt-get upgrade`  
- Remover um programa: `sudo apt-get remove nome\_do\_programa`  
  
Existem muitas outras opções e recursos disponíveis no comando "apt". Você pode conferir a documentação completa executando o comando `man apt` no terminal.

(**manage.py) =** O `manage.py` é um arquivo utilizado no framework web Python chamado Django. Ele é responsável por gerenciar a aplicação Django, executando comandos para realizar tarefas como criar o banco de dados, rodar o servidor de desenvolvimento, criar um novo aplicativo dentro da estrutura do projeto, entre outros.  
  
No terminal do Linux, podemos usar o `manage.py` executando comandos através do comando `python`, seguido do caminho relativo para o arquivo `manage.py` e o comando desejado. Por exemplo:  
  
```  
python manage.py runserver  
```  
  
Este comando irá iniciar o servidor de desenvolvimento do Django para a aplicação.  
  
Existem diversos outros comandos disponíveis no `manage.py`, que podem ser listados digitando `python manage.py --help` no terminal.

(**pip) =** O pip é um gerenciador de pacotes para Python que permite baixar, instalar e gerenciar pacotes e bibliotecas Python de terceiros. Com o pip, você pode facilmente instalar recursos e bibliotecas adicionais em seu ambiente Python para trabalhar em seus projetos. Ao executar o comando "pip" no terminal do Linux, você poderá acessar a ferramenta e executar várias operações, como instalar ou atualizar um pacote, ver os pacotes instalados em seu sistema e configurar as opções do pip.

(**runserver) =** O comando "runserver" é geralmente utilizado para executar um servidor local em um projeto web. No Linux, esse comando é frequentemente usado para executar projetos feitos em Python usando frameworks como Django ou Flask. Quando executado no terminal, o "runserver" inicia o servidor local e permite que você teste e visualize seu projeto web no navegador em seu computador. Geralmente, o comando é executado especificando o nome do arquivo principal do projeto, como "python manage.py runserver" para projetos Django.

(**migrate) =** O comando "migrate" no terminal Linux é usado para migrar um banco de dados para uma nova versão. Ele é comumente usado com o framework de migração de banco de dados chamado "Liquibase". O "migrate" é usado para aplicar ou reverter migrações em um banco de dados configurado para o Liquibase. O comando ajuda na manutenção e evolução de um banco de dados em projetos de software.

(**makemigrations) =** O comando "makemigrations" é utilizado no terminal Linux para criar um arquivo de migração no Django, que é um framework de desenvolvimento web em Python. Esse comando é usado principalmente quando se faz alterações no modelo de dados de um projeto Django. Através deste comando, o Django irá criar um arquivo de migração no qual estão contidas as instruções necessárias para alterar a estrutura do banco de dados, como criar, modificar ou excluir tabelas, colunas ou chaves. Depois disso, é necessário rodar o comando "migrate" para efetivar as alterações no banco de dados.

(**sudo) =** O comando "sudo" no terminal linux serve para executar um comando com privilégios de superusuário (root), o que permite ao usuário realizar tarefas que exigem permissões especiais, como instalar ou atualizar programas, alterar configurações de rede ou de sistema, entre outras ações. É importante utilizar o comando com cuidado, pois ele pode afetar o funcionamento do sistema operacional se usado incorretamente.

(**AdminLTE2) =** AdminLTE2 não é um comando ou programa padrão de terminal do Linux. Na verdade, é um template de painel de administração de interface do usuário gratuito e de código aberto para aplicações baseadas em Bootstrap. Isso pode ser baixado e utilizado para criar uma interface de usuário para uma aplicação web, mas não tem nada a ver com o terminal Linux ou qualquer outro terminal.

(**cd) =** O comando "cd" é utilizado no terminal Linux para mudar o diretório atual de trabalho. Ele significa "Chamar Diretório" (Change Directory) e é usado para navegar entre pastas. Por exemplo, se você estiver no diretório /home/user e quiser ir para o diretório "Documentos", você deve executar o comando "cd Documentos". Se você quiser voltar para o diretório anterior, pode executar o comando "cd ..".

(**Journalclt**) =O comando "journalctl" é usado no Linux para exibir o registro do sistema, que inclui informações sobre eventos, mensagens de erro, falhas e outras atividades do sistema operacional. Ele exibe o registro de eventos do sistema, em ordem cronológica inversa, permitindo que os usuários visualizem informações relevantes em um formato fácil de ler. O comando também pode ser usado para filtrar as mensagens por tipo de mensagem, nível de prioridade, intervalo de tempo e outros critérios. Ele é uma ferramenta útil para a solução de problemas em sistemas Linux.

(**clear) =** O comando "clear" no terminal do Linux limpa a tela, removendo todo o histórico de comandos e saídas anteriores, deixando a tela em branco e pronta para um novo conjunto de comandos e mensagens. Isso pode ser útil quando a tela do terminal fica muito confusa e desorganizada com muitas informações anteriores. Para executar o comando "clear", basta digitar "clear" na linha de comando e pressionar Enter.

(**get) =** O comando "get" no terminal Linux é utilizado para transferir um arquivo de um servidor remoto para um computador local, utilizando o protocolo FTP (File Transfer Protocol). É comum utilizar esse comando em conjunto com outros comandos FTP, como "open", "login", "cd" e "ls", para navegar e transferir arquivos de um servidor remoto. Por exemplo, para fazer o download de um arquivo chamado "arquivo.txt" de um servidor FTP remoto, você pode usar o seguinte comando no terminal:  
  
```  
ftp> get arquivo.txt  
```  
  
Esse comando vai baixar o arquivo "arquivo.txt" do servidor remoto para o diretório atual do terminal local.

(**upgrade) =** Upgrade no terminal do Linux é um comando usado para atualizar os pacotes de software do sistema operacional para a versão mais recente disponível. Ele atualiza tanto os pacotes instalados quanto suas dependências. O comando para atualizar os pacotes varia dependendo da distribuição Linux sendo utilizada, mas geralmente pode ser feito com "sudo apt-get upgrade" para distribuições baseadas em Debian e "sudo dnf upgrade" para distribuições baseadas em Fedora, por exemplo. É importante executar regularmente o comando de atualização para manter o sistema operacional seguro e funcionando corretamente.

**(update) =** O comando "apt-get update" (ou simplesmente "update") no terminal do Linux é usado para atualizar a lista de pacotes disponíveis para download nos repositórios configurados no sistema. Isso garante que o sistema possa instalar as últimas atualizações, correções de bugs e patches de segurança disponíveis para os pacotes instalados. O comando não atualiza os pacotes instalados no sistema, apenas a lista de pacotes disponíveis para download. Para atualizar os pacotes instalados, é necessário usar o comando "apt-get upgrade" ou "apt upgrade".

(**purge) =** O comando "purge" no terminal Linux é geralmente usado para desinstalar completamente um pacote e seus arquivos de configuração associados do sistema. No context do Django, "purge" não é um comando padrão ou oficial do framework, portanto, pode ser algo específico de um projeto ou biblioteca em particular. Sem mais informações, não posso fornecer uma resposta mais precisa.

(**remove) =** No terminal Linux, o comando "remove" (ou "apt-get remove") é utilizado para remover um pacote instalado do sistema. No contexto do Django, isso significa que se você tiver instalado o Django no seu sistema usando o "apt-get" (ou outro gerenciador de pacotes), você pode desinstalá-lo usando esse comando. Por exemplo, se você quiser remover a versão atual do Django do seu sistema, você pode executar o comando "sudo apt-get remove python3-django".

(**autoremove) =** O comando "autoremove" no terminal Linux não tem relação direta com o framework Django em si. Ele faz parte do Advanced Packaging Tool (APT) e sua função é remover automaticamente os pacotes que foram instalados como dependências e que já não são mais necessários para o sistema funcionar corretamente.

O uso desse comando pode liberar espaço em disco e ajudar a evitar conflitos entre pacotes instalados. No entanto, é importante ter cuidado ao utilizá-lo, pois ele pode remover pacotes que ainda são necessários para outros softwares funcionarem corretamente. Por isso, recomenda-se fazer uma análise cuidadosa antes de utilizar o "autoremove".

**(import) =** O comando "import" é usado na linguagem Python para incluir módulos ou pacotes em um programa. No contexto do Django, que é um framework web em Python, o comando "import" é usado para importar os componentes necessários em um projeto Django, como modelos, visualizações, formulários, etc. O comando "import" é utilizado dentro do interpretador do Python ou em um arquivo .py para importar um pacote, módulo ou uma classe. É importante lembrar que o Django utiliza a estrutura Model-View-Template (MTV) e, portantoNo terminal Linux, o comando "django-admin startproject" é usado para criar um novo projeto Django. Quando você adiciona "-c" após este comando, significa que você está criando um projeto usando um esqueleto (template) que já existe em vez de começar com um projeto vazio. Especificamente, o "-c" significa "template" e é usado para especificar o nome de um esqueleto de projeto que será usado para criar o novo projeto. Por exemplo, se você quiser criar um novo projeto Django usando o esqueleto "mysite\_template", o comando seria "django-admin startproject mysite -c mysite\_template"., é necessário importar os respectivos módulos para cada uma dessas camadas do projeto.

(**startproject) =** No terminal Linux, o comando "django-admin startproject" é usado para criar um novo projeto Django. Quando você adiciona "-c" após este comando, significa que você está criando um projeto usando um esqueleto (template) que já existe em vez de começar com um projeto vazio. Especificamente, o "-c" significa "template" e é usado para especificar o nome de um esqueleto de

projeto que será usado para criar o novo projeto. Por exemplo, se você quiser criar um novo projeto Django usando o esqueleto "mysite\_template", o comando seria "django-admin startproject mysite -c mysite\_template".

(**-m) =** O parâmetro "-m" no terminal do Django é usado para especificar um módulo Python que será executado. Geralmente é usado para iniciar um servidor de desenvolvimento do Django, por exemplo:  
  
```  
python -m django runserver  
```  
  
Esse comando inicia o servidor de desenvolvimento do Django definido no módulo "django" usando o pacote "runserver".

(**urllib3) =**O `urllib3` é uma biblioteca HTTP utilizada em Python para lidar com conexões HTTP. No contexto do Django, o `urllib3` pode ser usado para fazer requisições HTTP para outras APIs ou serviços externos a partir de sua aplicação. Ele é comumente utilizado em conjunto com outras bibliotecas, como o `requests`, para facilitar a execução de chamadas HTTP.  
  
Ao utilizar o `urllib3` no terminal Linux, você pode fazer chamadas diretamente na linha de comando usando o programa `curl`. Por exemplo, um comando simples para fazer uma chamada HTTP GET a um endpoint usando o `curl` e o `urllib3` seria:  
  
```  
curl --http1.1 http://www.exemplo.com  
```  
  
Esse comando usaria o `urllib3` para fazer a conexão HTTP à URL especificada e retornar a resposta na saída do terminal.

(**nano) =** O "nano" é um editor de texto que pode ser usado no terminal do sistema operacional Linux para criar e editar arquivos de texto. No contexto do Django, ele pode ser usado para editar e atualizar o código-fonte dos arquivos Python usados para criar e personalizar aplicativos web com o framework Django. O uso do "nano" pode simplificar a edição de arquivos de configuração do Django, como o arquivo settings.py, que contém as configurações globais do aplicativo.

(**args) =** O `args` no terminal Linux Django é usado para passar argumentos adicionais para o comando que está sendo executado pelo Django. Especificamente, ele é usado para passar argumentos para um comando personalizado que você pode ter definido em seu projeto Django usando o `Command` framework do Django. No comando personalizado, você pode especificar os argumentos que deseja receber usando o método `add\_arguments()` e, em seguida, acessá-los usando `args`. Por exemplo, se você tiver um comando personalizado chamado `import\_data`, que recebe um argumento `file`, você pode executá-lo da seguinte maneira:  
  
```  
python manage.py import\_data --file=data.csv  
```  
  
Neste caso, o argumento `--file` é passado para o comando personalizado `import\_data` usando `args`. O valor do argumento é `data.csv`. Em seu comando personalizado, você pode acessar o valor usando `args['file']`.

**(-r) =** No terminal do Linux, o parâmetro "-r" (ou "--noreload") é usado no framework Django para evitar que o servidor web recarregue automaticamente após a detecção de um arquivo atualizado. Isso pode ser útil em situações em que o código atualizado ainda não foi testado ou em situações em que a atualização automática pode afetar o desempenho do servidor. Ao especificar esse parâmetro, a opção de recarregamento automático é desabilitada. Por exemplo, para iniciar o servidor web do Django sem recarregamento automático, você pode usar o seguinte comando: "python manage.py runserver -r".

(**-p) =** A opção "-p" é utilizada no comando "runserver" do Django, que é utilizado para executar um servidor local de desenvolvimento. O "-p" é seguido do número da porta em que o servidor deve ser iniciado. Por exemplo:  
  
```  
python manage.py runserver -p 8000  
```  
  
Este comando inicia o servidor na porta 8000. Se a opção "-p" não for especificada, o servidor será iniciado na porta padrão 8000. É importante lembrar que essa opção pode ser substituída pela opção de abreviação "-port" ou pela opção textual "--port".

(**merge) =** O comando `merge` no terminal Linux geralmente é usado para mesclar duas ou mais branches (ramificações) do código fonte no Django. O que acontece é que o comando `merge` combina as alterações em duas branches, resultando em uma nova branch combinada que contém ambas as alterações. Isso é útil quando várias pessoas estão trabalhando no mesmo projeto e em diferentes branches. Para realizar um merge, é preciso estar na branch de destino e utilizar o comando `git merge <nome\_da\_branch>`. É importante estar atento e realizar testes antes de fazer o merge para garantir que o código esteja funcionando corretament

(**--no-header) =**A opção `--no-header` é usada com o comando `django-admin inspectdb` no terminal do Linux. Essa opção informa ao Django para não incluir o cabeçalho padrão em seu output, o que pode ser útil quando você está gerenciando o output desse comando. Por exemplo, se você estiver usando o output em um script ou importando-o em um banco de dados externo. Quando essa opção é usada, apenas as informações do modelo são retornadas, sem o cabeçalho informativo.

(**showmigrations) =** O comando "showmigrations" é usado no terminal do Django para mostrar todas as migrações disponíveis em um projeto. Ele exibe o status atual de cada migração, indicando se ela já foi aplicada ao banco de dados ou não. Isso é útil para verificar o estado atual das migrações e garantir que todas estejam devidamente aplicadas, evitando erros de compatibilidade ou integridade do banco de dados. O comando pode ser executado digitando "python manage.py showmigrations" no terminal do Django.

(**createsuperuser) =** O comando "createsuperuser" no terminal do Django é usado para criar um superusuário com privilégios administrativos para o sistema de gerenciamento de conteúdo Django.  
  
Quando você digita esse comando, é solicitado que você insira um nome de usuário, endereço de e-mail e senha. Depois de inserir essas informações, o superusuário é criado e você pode usá-lo para acessar o painel de administração do Django e gerenciar usuários, grupos e outras configurações do site.

(**mv)=** No contexto do Django, "mv" é um comando do Terminal Linux que é usado para renomear ou mover arquivos e diretórios de um local para outro. Por exemplo, se você quiser renomear um arquivo chamado "meuarquivo.txt" para "novoarquivo.txt", você pode usar o comando:  
  
```  
mv meuarquivo.txt novoarquivo.txt  
```  
  
Da mesma forma, se você quiser mover um arquivo chamado "meuarquivo.txt" para um diretório chamado "meudiretorio", você pode usar o comando:  
  
```  
mv meuarquivo.txt meudiretorio/  
```  
  
Note que o "/" no final do nome do diretório diz ao Linux para mover o arquivo para dentro desse diretório. Portanto, é importante garantir que o nome do diretório esteja correto e que o diretório exista antes de usar o comando `mv`.

(**sudo su ) =** O comando "sudo su" no terminal Linux permite que o usuário acesse a conta de superusuário no sistema operacional. Ao executar esse comando, você estará mudando para a conta de superusuário, que tem permissões de acesso elevadas e pode realizar tarefas que exigem privilégios de administrador.  
  
No contexto do Django, o uso do "sudo su" no terminal pode ser útil para executar certas tarefas que requerem permissões de acesso exclusivas do superusuário, como instalar pacotes do sistema ou configurar as permissões de arquivos e pastas necessárias para executar o aplicativo. No entanto, certifique-se de ter cuidado ao usar a conta de superusuário, pois isso pode apresentar riscos de segurança e causar danos ao seu sistema se você executar comandos incorretos por engano.

(**source) =** O comando "source" no terminal Linux é usado para executar os comandos de um arquivo de script no terminal atual. Em um ambiente do Django, o uso mais comum do comando "source" é executar o arquivo "activate" dentro do ambiente virtual do Django. Isso permite que você ative o ambiente virtual, que contém as dependências necessárias para executar o seu aplicativo Django. Por exemplo, para ativar o ambiente virtual "myenv" no diretório "/home/user/myproject", você pode executar o seguinte comando no terminal:  
  
source /home/user/myproject/myenv/bin/activate  
  
Depois de executar este comando, você verá que o prompt do terminal mudará para indicar que o ambiente virtual está ativo, como por exemplo "(myenv) user@hostname:~/myproject$". Agora você pode executar comandos relacionados ao Django no terminal.

(**su) =** O comando "su" no terminal Linux é usado para trocar de usuário e assumir a identidade de outro usuário no sistema. Já o "django" é um framework para desenvolvimento web em Python. Portanto, a combinação desses dois comandos geralmente não é necessária no contexto do desenvolvimento com Django, a menos que você precise fazer a troca de usuário para acessar pastas ou arquivos restritos do sistema enquanto trabalha em seu projeto Django.

(**snap) =** O comando "snap" é uma ferramenta de gerenciamento de pacotes em distribuições Linux baseadas em Ubuntu. No contexto do desenvolvimento web com Django, o comando "snap" pode ser usado para instalar o Python e outras dependências necessárias para o desenvolvimento e implantação do projeto. Além disso, ele pode ser usado para instalar outras ferramentas úteis para o desenvolvimento web, como o Git, Node.js, etc. Para mais informações sobre como usar o comando "snap", você pode consultar a documentação oficial do Ubuntu: <https://snapcraft.io/docs>.

O comando "snap" é uma ferramenta de gerenciamento de pacotes em distribuições Linux baseadas em Ubuntu. No contexto do desenvolvimento web com Django, o comando "snap" pode ser usado para instalar o Python e outras dependências necessárias para o desenvolvimento e implantação do projeto. Além disso, ele pode ser usado para instalar outras ferramentas úteis para o desenvolvimento web, como o Git, Node.js, etc. Para mais informações sobre como usar o comando "snap", você pode consultar a documentação oficial do Ubuntu: https://snapcraft.io/docs. O comando "snap" é uma ferramenta de gerenciamento de pacotes em distribuições Linux baseadas em Ubuntu. No contexto do desenvolvimento web com Django, o comando "snap" pode ser usado para instalar o Python e outras dependências necessárias para o desenvolvimento e implantação do projeto. Além disso, ele pode ser usado para instalar outras ferramentas úteis para o desenvolvimento web, como o Git, Node.js, etc. Para mais informações sobre como usar o comando "snap", você pode consultar a documentação oficial do Ubuntu: https://snapcraft.io/docs.