Aula 01

Django

O que é?

Django é um framework web de alto nível escrito em Python que segue o padrão de design de model-template-view. Ele é utilizado para desenvolvimento de aplicações web, com o objetivo de simplificar o processo de desenvolvimento, fornecendo componentes prontos para serem utilizados, tais como gerenciamento de banco de dados, formulários, autenticação de usuários, entre outros.

A estrutura do Django é baseada em três componentes principais :

- Model: representa a camada de dados, responsável por gerenciar informações e realizar operações de negócios;

- View: responsável por apresentar os dados ao usuário;

- Template: é a camada de apresentação, responsável por definir a estrutura básica da página, como posições de elementos e estilos;

Preparando o Local de Trabalho

Ambiente Virtual

Um ambiente virtual é um ambiente isolado em seu sistema operacional que permite que você instale pacotes e gerencie suas dependências de forma separada das outras aplicações em seu computador. Isso é útil porque diferentes aplicações podem exigir diferentes versões de bibliotecas e pacotes, e o uso de um ambiente virtual permite que você mantenha as versões corretas para cada aplicação.

Podemos criar um ambiente virtual específico para a aplicação e, em seguida, instalar pacotes e gerenciar suas dependências nesse ambiente. Quando estivermos prontos para trabalhar na aplicação, iremos ativar o ambiente virtual e ter certeza de que estamos usando as versões corretas dos pacotes.

Há duas ferramentas que possibilitam trabalhar com ambientes virtuais :

- venv: é uma ferramenta embutida na distribuição do Python a partir da versão 3.3; é uma alternativa ao virtualenv e fornece as mesmas funcionalidades básicas, mas sem a necessidade de instalar uma ferramenta externa;

- virtualenv: é uma ferramenta externa que pode ser instalada através do gerenciador de pacotes **pip**; é uma das ferramentas mais antigas e amplamente utilizadas para criar ambientes virtuais em Python;

Ambos venv e virtualenv são ferramentas válidas para criar ambientes virtuais em Python, mas venv é uma opção nativa e mais simples que está incluída nas distribuições mais recentes do Python. Se estivermos usando uma versão recente do Python, é recomendável usar venv. Caso contrário, pode ser mais conveniente usar virtualenv.

Ambos irão criar uma pasta no local onde foram executados, onde ficará uma cópia da instalação do python e, depois de ativado, todos os pacotes que forem instalados com o pip. Isso é ótimo para isolar o Python do Sistema Operacional de ter muitos pacotes não usados (ou até mesmo com muitas versões diferentes).

Vamos ver como configurar cada um deles.

venv

Como descrito anteriormente, não é necessário instalar qualquer pacote para usar a funcionalidade.

Para criarmos um ambiente virtual com venv, basta executar o comando abaixo no terminal:

```
C:\Users\augusto.hertzog\um_projeto>python -m venv .venv
```

Onde:

- **python** é o interpretador instalado na máquina;
- **-m <modulo>** o módulo fornecido está localizado no caminho do módulo Python e é executado como um script, neste caso é o módulo venv;
- **.venv** é o nome da pasta que será criada para guardar uma cópia do Python e tudo o que for instalado usando **pip**; .venv é o nome que se costuma dar para essa finalidade, mas ela pode ter qualquer nome;

Mais conteúdo e explicações podem ser encontrados aqui Linha de Comando e Ambiente.

virtualenv

Como explicado acima, é o módulo que serviu de referência para o atual built-in venv. O funcionamento é praticamente igual, só tendo que instalar o pacote antes de utilizá-lo.

Para instalar, basta realizar a chamada do pip como abaixo:

```
C:\Users\augusto.hertzog\um_projeto>pip install virtualenv
```

Para criar um ambiente virtual, usamos o comando abaixo:

```
C:\Users\augusto.hertzog\um_projeto>python -m virtualenv .venv
```

Ativando o Ambiente Virtual

Há duas formas de ativar o ambiente virtual:

- executando a partir da raiz do projeto
  - Windows: .venv\Scripts\activate
  - Linux: source .venv/bin/activate
- navegando até a pasta Scripts e ativando:
  - Windows:
    - cd .venv\Scripts
    - activate
  - Linux:
    - cd .venv/bin
    - activate

Independente da forma, quando ativado o terminal exibirá o nome da pasta do ambiente virtual no começo da linha:

```
C:\Users\augusto.hertzog\um_projeto>.venv\Scripts\activate

(.venv) C:\Users\augusto.hertzog\um_projeto>
```

Isso indica que nosso ambiente virtual está ativo e operante.

Agora já podemos instalar os pacotes necessários diretamente na pasta do projeto:

```
(.venv) C:\Users\augusto.hertzog\um_projeto>pip install pylint
```

```
(.venv) C:\Users\augusto.hertzog\um_projeto>pip install autopep8
```

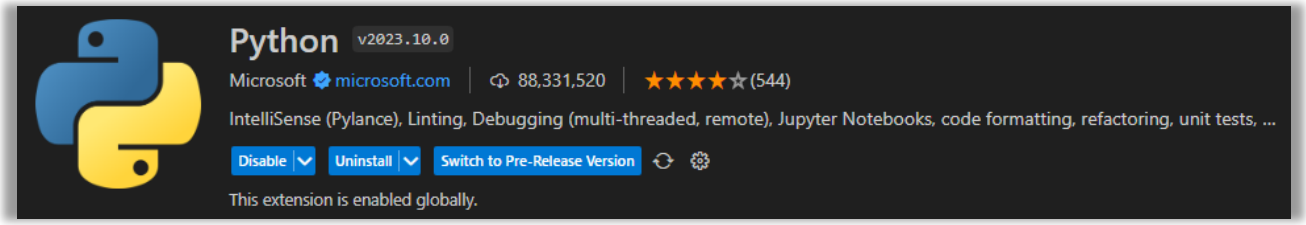
Desativando o Ambiente Virtual

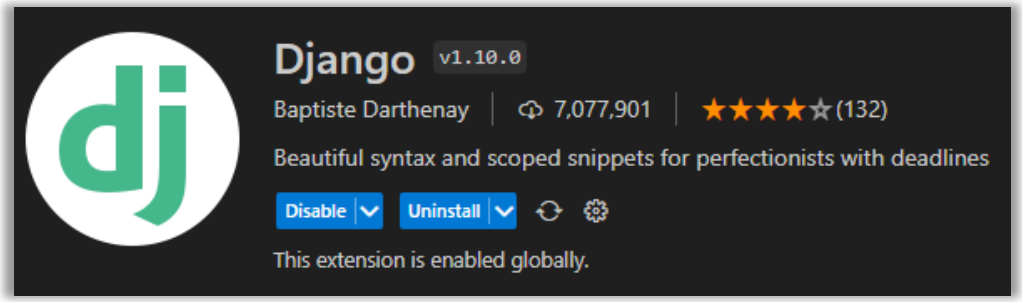
Para desativar o Ambiente Virtual, basta chamarmos a qualquer momento no terminal o comando deactivate:

```
(.venv) C:\Users\augusto.hertzog\um_projeto>deactivate
C:\Users\augusto.hertzog\um_projeto>
```

VS Code

Para o VS Code, tenha certeza de que as extensões do Python e do Django (Baptiste Darthenay) estejam instaladas:





Iniciando o Projeto

Para iniciarmos um novo projeto, primeiro criaremos uma pasta com o nome do nosso projeto, que vai ser chamada de curso.

Vamos seguir o passo a passo abaixo:

- criando a pasta do projeto:

```
C:\Users\augusto.hertzog>mkdir curso
```

  - mkdir = make directory = criar pasta

- após criada, entraremos nela com o comando cd:

```
C:\Users\augusto.hertzog>cd curso

C:\Users\augusto.hertzog\curso>
```

  - cd = change directory = mudar pasta

- dentro da pasta, criaremos nosso ambiente virtual:

```
C:\Users\augusto.hertzog\curso>python -m venv .venv
```

- depois o ativamos:

```
C:\Users\augusto.hertzog\curso>.venv\Scripts\activate

(.venv) C:\Users\augusto.hertzog\curso>
```

- agora, vamos instalar alguns pacotes para nos ajudar na formatação do código e verificação de erros:

```
(.venv) C:\Users\augusto.hertzog\curso>pip install pylint autopep8
```

    - pode ser que apareça uma notificação avisando que a versão do pip que está sendo usada está desatualizada. Para atualizar, basta seguir a instrução que vai aparecer:

```
[notice] A new release of pip available: 22.3.1 -> 23.1.2
[notice] To update, run: python.exe -m pip install --upgrade pip

(.venv) C:\Users\augusto.hertzog\curso>python -m pip install --upgrade pip
```

- agora, finalmente, iremos instalar o Framework do Django:

```
(.venv) C:\Users\augusto.hertzog\curso>pip install django
```

- uma vez instalado, podemos verificar a versão dele com o comando abaixo:

```
(.venv) C:\Users\augusto.hertzog\curso>django-admin --version
4.2.2
```

- para uma lista de comandos do Django, basta executar o comando abaixo:

```
(.venv) C:\Users\augusto.hertzog\curso>django-admin --help

Type 'django-admin help <subcommand>' for help on a specific subcommand.
```

- agora, vamos iniciar o nosso site:

```
(.venv) C:\Users\augusto.hertzog\curso>django-admin startproject curso .
```

- sobre o comando acima:
    - **django-admin**: é usado quando queremos especificar alguma funcionalidade envolvendo o Django;
    - **startproject**: ele indica que estamos iniciando um novo projeto;
    - **curso**: é o nome do nosso projeto;
    - **.**: indica o local onde a pasta acima especificada será criada, nesse caso, na pasta atual que estamos;

- finalmente, inicializaremos o nosso servidor:

```
(.venv) C:\Users\augusto.hertzog\curso>python manage.py runserver
```

- sobre o comando acima:
  - python: é a chamada do interpretador Python do nosso Ambiente Virtual;
  - manage.py: um dos vários arquivos criados quando executamos o comando startproject, veremos mais sobre ele adiante;
  - runserver: é o comando que usamos para executar um servidor local e assim poder acessar o site através do browser;

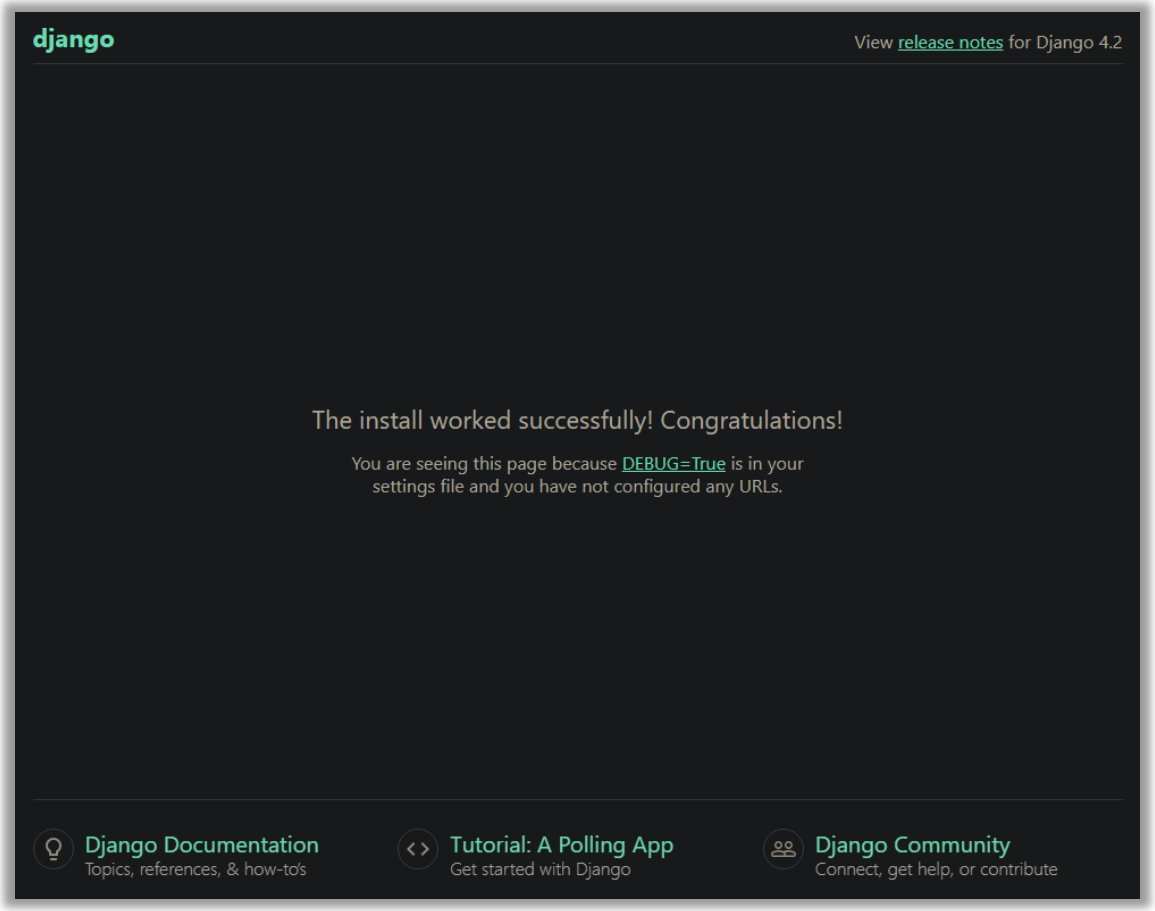
Após executando o comando acima, deve aparecer algo como isso no terminal:

```
(.venv) C:\Users\augusto.hertzog\curso>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until y
ou apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
June 13, 2023 - 16:10:07
Django version 4.2.2, using settings 'curso.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

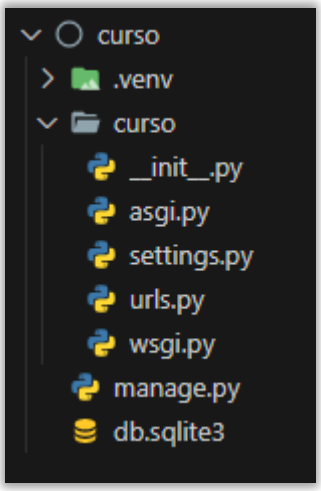
Entre no link especificado acima para ver o servidor executando seu site.



et voilà, temos nossa primeira página!

Conhecendo as Pastas e Arquivos

Após executado o comando startproject, iremos ter uma estrutura de projeto parecida com a seguinte:



Vamos passar pelos arquivos e pastas que temos até então:

- **db.sqlite3**: repare que surgiu um novo arquivo no seu projeto. Ele é um arquivo de banco de dados, que vai guardar tudo que salvarmos no Django. Não vamos nos preocupar com ele agora;

- **manage.py**: arquivo muito importante durante o desenvolvimento. Ele faz a mesma coisa que o django-admin faz. Todos os comandos podem ser executados também pelo arquivo;
    - a diferença dele para o django-admin é que ele configura uma variável de ambiente para dentro do seu projeto, carregando todas as configurações de settings.py para nosso site;

```
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'curso.settings')
```

Se tentarmos executar o servidor diretamente pelo django-admin,

```
(.venv) C:\Users\augusto.hertzog\curso>django-admin runserver
```

vai ocorrer o erro abaixo, pois ele não encontrou a variável de ambiente do projeto, a DJANGO_SETTINGS_MODULE:

```
  File "C:\Users\augusto.hertzog\curso\.venv\Lib\site-packages\django\conf\__i
nit__.py", line 102, in __getattr__
    self._setup(name)
  File "C:\Users\augusto.hertzog\curso\.venv\Lib\site-packages\django\conf\__i
nit__.py", line 82, in _setup
    raise ImproperlyConfigured(
django.core.exceptions.ImproperlyConfigured: Requested setting DEBUG, but sett
ings are not configured. You must either define the environment variable DJANG
O_SETTINGS_MODULE or call settings.configure() before accessing settings.
```

Por enquanto, a única finalidade do django-admin é a execução do startproject. Para todos os demais, usaremos o manage.py.

Pasta curso Criada

Essa pasta foi criada ao executarmos o comando startproject. Ela terá os arquivos essenciais de configuração para todo o nosso site.

- **__init__.py**: é o módulo responsável por indicar que aquela pasta é um pacote do python. Podemos usar para inicializar determinadas tarefas do pacote, mas não vamos nos preocupar com ele por hora;

- **asgi.py e wsgi.py**: são arquivos de configuração usados quando publicamos nosso site, para realizar a conexão entre o servidor web e o Django. Às vezes um será utilizado, às vezes outro. Vai depender do servidor que nosso site ficará hospedado;

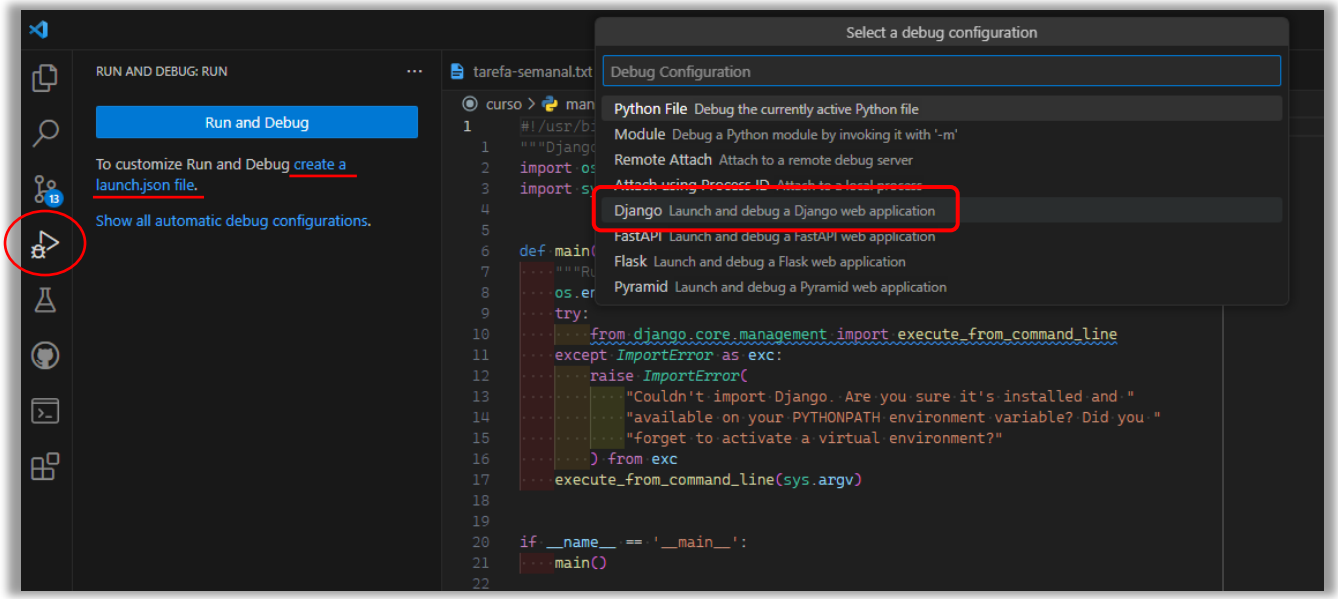
- **settings.py**: é o arquivo de configuração do nosso site Django. Todas as configurações que precisamos para o Django funcionar corretamente precisam estar dentro desse arquivo. Ele é responsável por especificar como o nosso site deve se comportar. Vamos utilizar muito esse arquivo.

- **urls.py**: é outro arquivo muito importante. É a porta de entrada da nossa aplicação. É onde vamos cadastras nossos aplicativos (páginas) do site;

Criando um Debugger

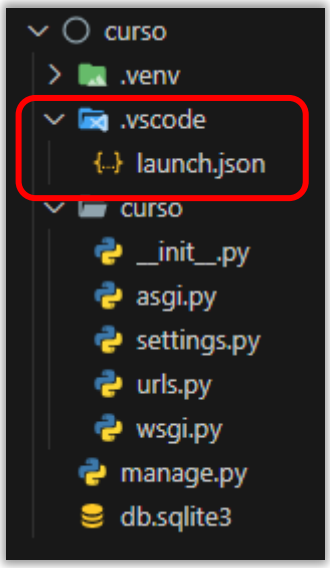
Para melhor desenvolver nosso site, temos que adicionar a pasta onde ele está ao VS Code. Para isso, vá na barra de menu e escolha Arquivo > Abrir Pasta. Navegue até a pasta onde está o seu projeto (aquela criada usando o mkdir) e selecione ela.

Agora vamos criar um debugger. Dessa forma, não será preciso reiniciar e interromper o servidor sempre que realizarmos qualquer alteração.



Para cria-lo, no VS Code, vamos na opção da barra lateral "Run and Debug" (indicado com um círculo vermelho). Depois vamos em "create a launch.json file" (sublinhado em vermelho). Quando clicado, vai ser aberto uma caixa no meio do VS Code com as opções disponíveis de debug. Como nosso site está sendo feito em Django, selecionaremos a opção Django da lista (caixa vermelha). Ele deve estar igual ao código abaixo:

Ele será criado em uma nova pasta chamada .vscode dentro do nosso projeto:



Exercício para Praticar

1. Explore os arquivos manage.py, settings.py e urls.py criados nos passos acima.
    a. Como os arquivos asgi.py e wsgi.py são usados apenas quando for publicado em um servidor web real, não se preocupe com eles até o final do curso;
2. Veja que os arquivos manage.py e settings.py possuem diversos links. Explore-os.
3. Crie novos projetos seguindo as instruções acima para que se sinta confortável com o processo.
4. Realize o passo-a-passo para o site que você quer desenvolver. Nas próximas aulas, ele será usado para dar continuidade ao projeto.