Tutorial de configurações do GetoriiHost

**CONFIGURANDO O AMBIENTE**

Downloads e instalações necessárias:

* Anaconda Python
* VSCode
* PostgreSQL 16+
* DataBase Client(Pluggin de gerenciamento de BD - extensão)
* Git

Preparando o ambiente (terminal)

* conda env list
* conda create -n gestoriihost python=3.11.9
* conda activate gestoriihost
* pip install Django==4.1
* pip install psycopg2
* pip install psycopg2-binary

Preparando o ambiente (IDE)

* Abrir um terminal no VSCode
* *conda activate gestoriihost*
* CTRL+SHIFT+P: Selecionar o interpretador python correspondente ao ambiente virtual chamado gestoriihost

**PREPARANDO A APLICAÇÃO Django**

* Navegar via terminal (cd) até a pasta onde você irá criar o projeto Django, use:
* django-admin startproject gestoriihost
* Abra a pasta criada pelo Django no VSCode
* Renomeie a pasta *gestoriihost/gestoriihost* para *gestoriihost/config*
* Acesse o arquivo manage.py na raiz e altere a linha onde está *gestoriihost.settings* para *config.settings*.
* Acesse o arquivo wsgi.py em config e altere *gestoriihost.settings*. para *config.settings*.
* Crie no app config um arquivo chamado middleware.py e adicione o seguinte código:

| #classe para desabilitar cache e evitar problemas com dados armazenados pelo navegador  class NoCacheMiddleware:  def \_\_init\_\_(self, get\_response):  self.get\_response = get\_response  def \_\_call\_\_(self, request):  response = self.get\_response(request)  response['Cache-Control'] = 'no-cache, no-store, must-revalidate'  response['Pragma'] = 'no-cache'  response['Expires'] = '0'  return response |
| --- |

* Crie na raiz da aplicação uma pasta chamada resources.
* Dentro de resources crie uma pasta chamada static.
* Dentro de static crie 3 pastas: css, img e js.
* Acesse o arquivo settings.py dentro de config e:
  + Altere *WSGI\_APPLICATION = "gestoriihost.wsgi.application"* para *config.wsgi.application*
  + Importe os seguintes pacotes no cabeçalho:

import os

from django.contrib.messages import constants as messages

* Em MIDDLEWARE adicione:

'config.middleware.NoCacheMiddleware',

* Altere *ROOT\_URLCONF = "gestoriihost.urls"* para *config.urls.*
* Adicione a seção abaixo após ROOT\_URLCONF:

#integra com o sistema de auth padrao do Django

#AUTH\_USER\_MODEL = 'cadastros.Usuario'

AUTHENTICATION\_BACKENDS = [

#'cadastros.backends.EmailBackend', #backend de autenticacao personalizado

'django.contrib.auth.backends.ModelBackend', #backend de autenticacao padrao

]

* Em TEMPLATES altere a linha DIRS para:

"DIRS": [os.path.join(BASE\_DIR, "resources")], #define o caminho da pasta que tera os recursos usados pelos templates

* Após a sessão TEMPLATES adicione essa nova seção:

#URL de redirecionamento apos login

LOGIN\_REDIRECT\_URL = 'core:main'

LOGOUT\_REDIRECT\_URL = 'autenticacao:login'

LOGIN\_URL = 'autenticacao:login'

* Altere DATABASES para:

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.postgresql',

'NAME': 'db\_pypel',

'USER': 'postgres',

'PASSWORD': 'root',

'HOST': 'localhost',

'PORT': '5432',

}

}

* Adicione após DATABASES:

# Settings for messages

MESSAGE\_TAGS = {

messages.DEBUG: 'debug',

messages.INFO: 'info',

messages.SUCCESS: 'success',

messages.WARNING: 'warning',

messages.ERROR: 'danger',

}

* Altere LANGUAGE\_CODE e TIME\_ZONE para:

LANGUAGE\_CODE = "pt-BR"

TIME\_ZONE = "America/Sao\_Paulo"

* Altere a seção Static files para:

STATICFILES\_FINDERS = [

'django.contrib.staticfiles.finders.FileSystemFinder',

'django.contrib.staticfiles.finders.AppDirectoriesFinder',

]

STATIC\_URL = "/static/"

STATICFILES\_DIRS = [

os.path.join(BASE\_DIR, "resources/static/"),

]

* Crie as duas variáveis globais no final do arquivo com:

#Configurações de variáveis globais

NUMBER\_GRID\_PAGES = 20

NUMBER\_GRID\_MODAL = 20

**CRIANDO OS APPS Django**

Vamos criar os apps responsáveis por autenticar, guardar os códigos reutilizáveis e o que armazenara os cadastros do sistema. Para isso execute:

* python manage.py startapp core
* python manage.py startapp autenticacao
* python manage.py startapp cadastros
* Em settings.py dentro do app config adicione os apps dentro de INSTALLED\_APPS para ficar da seguinte forma:

INSTALLED\_APPS = [

"django.contrib.admin",

"django.contrib.auth",

"django.contrib.contenttypes",

"django.contrib.sessions",

"django.contrib.messages",

"django.contrib.staticfiles",

"autenticacao",

"core",

"cadastros",

]

**CRIANDO O MODELO DE USUÁRIO DO SISTEMA**

* Insira o seguinte o código dentro de models.py no app cadastros

from django.db import models

from django.contrib.auth.models import AbstractBaseUser, BaseUserManager

from django.db.models.deletion import ProtectedError

🐒olhar com o ciniro como a gente pode criar nossas tabelas com base nos nosso projeto de hotelaria, a gente já tem um der ai meio que no pente, dá pra expressar bastante ideia pra ele, acho que ele da um help pa noisclass Departamento(models.Model):

nome = models.CharField(max\_length=500)

sigla = models.CharField(max\_length=30)

def delete(self, \*args, \*\*kwargs):

if self.usuario\_set.exists():

raise ProtectedError(

"Não é possível excluir este departamento, pois ele possui usuarios vinculados.",

self

)

super().delete(\*args, \*\*kwargs)

def \_\_str\_\_(self):

return self.nome

class Perfil(models.Model):

nome = models.CharField(max\_length=100, unique=True)

def delete(self, \*args, \*\*kwargs):

if self.usuario\_set.exists():

raise ProtectedError(

"Não é possível excluir este perfil, pois ele possui usuarios vinculados.",

self

)

super().delete(\*args, \*\*kwargs)

def \_\_str\_\_(self):

return self.nome

class UsuarioManager(BaseUserManager):

def create\_user(self, email, nome, password=None):

if not email:

raise ValueError('O usuário deve ter um endereço de e-mail')

user = self.model(email=self.normalize\_email(email), nome=nome)

user.set\_password(password)

user.save(using=self.\_db)

return user

def create\_superuser(self, email, nome, password=None):

user = self.create\_user(email, nome, password)

user.is\_admin = True

user.save(using=self.\_db)

return user

class Usuario(AbstractBaseUser):

nome = models.CharField(max\_length=300)

email = models.EmailField(unique=True)

is\_active = models.BooleanField(default=True)

is\_admin = models.BooleanField(default=False)

departamento = models.ForeignKey(Departamento, on\_delete=models.PROTECT)

perfis = models.ManyToManyField(Perfil)

objects = UsuarioManager()

USERNAME\_FIELD = 'email'

REQUIRED\_FIELDS = ['nome']

def \_\_str\_\_(self):

return self.email

def has\_perm(self, perm, obj=None):

return True

def has\_module\_perms(self, app\_label):

return True

@property

def is\_staff(self):

return self.is\_admin

def tem\_perfil(self, perfil\_nome):

return self.perfis.filter(nome=perfil\_nome).exists()

**CRIANDO O BANCO DE DADOS**

* Verifique se o psql está acessível via terminal como uma variável de ambiente. Isso depende de como está configurado o seu sistema operacional. Se não for o caso, identifique como fazer isso e adicione a variável PATH ao ambiente.
* Acesse o psql via terminal com:

sudo -u postgres psql

* Digite a senha do seu usuário sudo
* Acesse o PostgreSQL
* Digite a senha root do usuário postgres
* Crie o banco:

CREATE DATABASE db\_pypel;

* Conceda privilégios no banco:

GRANT ALL PRIVILEGES ON DATABASE bd\_gestoriihost TO postgres;

* Cheque a criação do banco:

\l

* Saia do psql

\q

**ACESSE O BANCO VIA EXTENSÃO VS CODE**

* Entre no DB Client e crie uma conexão com o banco de dados preenchendo os campos com os dados condizentes com a string de conexão com o banco de dados.
* Visualize o banco.

**GERANDO O BANCO DE DADOS COM BASE NO MODELO**

* No terminal digite:
  + python manage.py makemigrations
  + python manage.py migrate
* Abra o DB Client e cheque se as tabelas foram criadas corretamente.
* Descomente em *settings.py* no diretório config os modelos de autenticação de usuário:

#integra com o sistema de auth padrao do Django

AUTH\_USER\_MODEL = 'cadastros.Usuario'

AUTHENTICATION\_BACKENDS = [

'cadastros.backends.EmailBackend', #backend de autenticacao personalizado

'django.contrib.auth.backends.ModelBackend', #backend de autenticacao padrao

]

**TESTANDO A APLICAÇÃO**

* python manage.py runserver
* Se tiver feito tudo certo verá o foguetinho do Django!