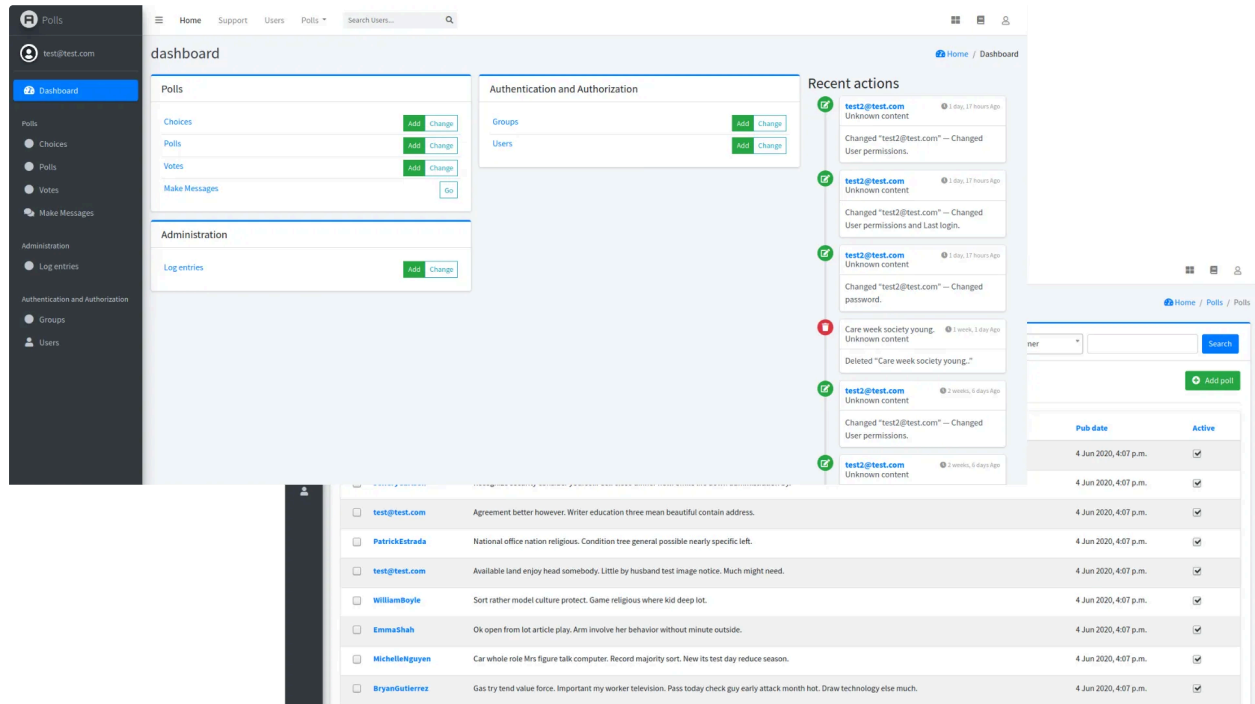


py_live #013

Sistemas com Django Admin



<https://whimsical.com/django-admin-Xcxu1UctinYneVvQcWNU3W>

O que é o Django Admin?

Django Admin é um Painel Administrativo, uma interface pré-construída que permite aos administradores do sistema gerenciar e visualizar facilmente os dados da aplicação, sem que você precise escrever código adicional para criar painéis de controle personalizados.

Por que Usar o Django Admin?

- Produtividade
- Interface amigável e pronta para uso
- Customização completa

- Segurança e estrutura de autenticação e permissões

Quando utilizar Django Admin?

- Administração interna
- Prototipagem rápida
- Sistemas de suporte
- Sistemas de backoffice e gestão em geral
- Gestão de conteúdos de sites e blogs

Vamos praticar

Criaremos um projeto Django para cadastro de produtos utilizando apenas os recursos do Django Admin, e também personalizar a interface com algumas bibliotecas de UI.

Repositório do projeto:  [sgp](#)

Criar venv e instalar Django 5.0 (5.1 com problema de compatibilidade com o jazzmin)

```
python -m venv venv venv/Scripts/activate pip install django==5.0
```

Criar e inicializar projeto Django

```
django-admin startproject core . python manage.py migrate python manage.py createsuperuser
```

Rodar o sistema e acessar o admin

```
python manage.py runserver
```

Criar app de produtos

```
python manage.py startapp products
```

core/settings.py

```
INSTALLED_APPS = [ 'products', ] LANGUAGE_CODE = 'pt-br' TIME_ZONE =  
'America/Sao_Paulo'
```

Modelar o sistema

products/models.py

```

from django.db import models
class Brand(models.Model):
    name = models.CharField(max_length=100, verbose_name='Nome')
    is_active = models.BooleanField(default=True, verbose_name='Ativo')
    description = models.TextField(null=True, blank=True, verbose_name='Descrição')
    created_at = models.DateTimeField(auto_now_add=True, verbose_name='Criado em')
    updated_at = models.DateTimeField(auto_now=True, verbose_name='Atualizado em')
    class Meta:
        ordering = ['name']
        verbose_name = 'Marca'
    def __str__(self):
        return self.name
class Category(models.Model):
    name = models.CharField(max_length=100, verbose_name='Nome')
    is_active = models.BooleanField(default=True, verbose_name='Ativo')
    description = models.TextField(null=True, blank=True, verbose_name='Descrição')
    created_at = models.DateTimeField(auto_now_add=True, verbose_name='Criado em')
    updated_at = models.DateTimeField(auto_now=True, verbose_name='Atualizado em')
    class Meta:
        ordering = ['name']
        verbose_name = 'Categoria'
    def __str__(self):
        return self.name
class Product(models.Model):
    title = models.CharField(max_length=100, verbose_name='Título')
    brand = models.ForeignKey(Brand, on_delete=models.PROTECT, related_name='products', verbose_name='Marca')
    category = models.ForeignKey(Category, on_delete=models.PROTECT, related_name='products', verbose_name='Categoria')
    price = models.DecimalField(max_digits=10, decimal_places=2, verbose_name='Preço')
    is_active = models.BooleanField(default=True, verbose_name='Ativo')
    description = models.TextField(null=True, blank=True, verbose_name='Descrição')
    created_at = models.DateTimeField(auto_now_add=True, verbose_name='Criado em')
    updated_at = models.DateTimeField(auto_now=True, verbose_name='Atualizado em')
    class Meta:
        ordering = ['title']
        verbose_name = 'Produto'
    def __str__(self):
        return self.title

```

Configurar o admin do sistema

products/admin.py

```

from django.contrib import admin from .models import Brand, Category, Product
@admin.register(Brand) class BrandAdmin(admin.ModelAdmin):
    list_display = ('name', 'is_active', 'description', 'created_at', 'updated_at')
    search_fields = ('name',) list_filter = ('is_active',)
@admin.register(Category) class CategoryAdmin(admin.ModelAdmin):
    list_display = ('name', 'is_active', 'description', 'created_at', 'updated_at')
    search_fields = ('name',) list_filter = ('is_active',)
@admin.register(Product) class ProductAdmin(admin.ModelAdmin):
    list_display = ('title', 'brand', 'category', 'price', 'is_active', 'created_at', 'updated_at')
    search_fields = ('title', 'brand__name', 'category__name')
    list_filter = ('is_active', 'brand', 'category')

```

Acessar admin, criar usuário e brincar com as permissões e grupos

Criar exportação para csv no admin

products/admin.py


```

import csv from django.http import HttpResponse from django.contrib
import admin from .models import Brand, Category, Product
@admin.register(Brand) class BrandAdmin(admin.ModelAdmin):
    list_display = ('name', 'is_active', 'description', 'created_at', 'updated_at')
    search_fields = ('name',) list_filter = ('is_active',)
@admin.register(Category) class CategoryAdmin(admin.ModelAdmin):
    list_display = ('name', 'is_active', 'description', 'created_at', 'updated_at')
    search_fields = ('name',) list_filter = ('is_active',)
@admin.register(Product) class ProductAdmin(admin.ModelAdmin):
    list_display = ('title', 'brand', 'category', 'price', 'is_active', 'created_at', 'updated_at')
    search_fields = ('title', 'brand__name', 'category__name')
    list_filter = ('is_active', 'brand', 'category')
    def export_to_csv(self, request, queryset):
        response = HttpResponse(content_type='text/csv')
        response['Content-Disposition'] = 'attachment; filename="products.csv"'
        writer = csv.writer(response)
        writer.writerow(['título', 'marca', 'categoria', 'preço', 'ativo', 'descrição', 'criado em', 'atualizado em'])
        for product in queryset:
            writer.writerow([product.title, product.brand.name, product.category.name, product.price, product.is_active, product.description, product.created_at, product.updated_at])
        return response
    export_to_csv.short_description = 'Exportar para CSV'
    actions = [export_to_csv]

```

Vamos deixar a coisa mais interessante...

Vamos instalar e explorar algumas bibliotecas de UI para o admin.

 Django Packages Django Packages : Reusable apps, sites and tools directory f...

Django Grappelli Documentation — Django Grappelli 4.0.1 documentation

Instalação

```
pip install django-grappelli
```

core/settings.py

```
INSTALLED_APPS = [ 'grappelli', 'django.contrib.admin', ... ] GRAPPELLI_ADMIN_TITLE = 'SGP'
```

core/urls.py

```
from django.contrib import admin from django.urls import path, include urlpatterns = [ path('grappelli/', include('grappelli.urls')), path('admin/', admin.site.urls), ]
```

Django Jazzmin Jazzmin

Instalação


```
pip install django-jazzmin
```

core/settings.py

```
INSTALLED_APPS = [ 'jazzmin', 'django.contrib.admin', ... ]
```

Algumas personalizações

```
JAZZMIN_SETTINGS = { # title of the window (Will default to current_
admin_site.site_title if absent or None) 'site_title': 'SGP', # Titl
e on the login screen (19 chars max) (defaults to current_admin_sit
e.site_header if absent or None) 'site_header': 'SGP', # Title on th
e brand (19 chars max) (defaults to current_admin_site.site_header i
f absent or None) 'site_brand': 'SGP', 'icons': { 'auth': 'fas fa-us
ers-cog', 'auth.user': 'fas fa-user', 'auth.Group': 'fas fa-users',
'products.Brand': 'fas fa-copyright', 'products.Category': 'fas fa-o
bject-group', 'products.Product': 'fas fa-box', }, # Welcome text on
the login screen 'welcome_sign': 'Bem-vindo(a) ao SGP', # Copyright
on the footer 'copyright': 'PycodeBR LTDA', # List of model admins t
o search from the search bar, search bar omitted if excluded # If yo
u want to use a single search field you dont need to use a list, you
can use a simple string 'search_model': ['products.Product'], # Whe
ther to show the UI customizer on the sidebar 'show_ui_builder': Tru
e, }
```

Biblioteca de ícones em  fontawesome Find Icons with the Perfect Look & Feel | Font Awesome

Mais personalizações usando o UI Builder

1A77MTM HIT TUBE/KO f *hooker small tooth* *Feline* *foster small tooth*