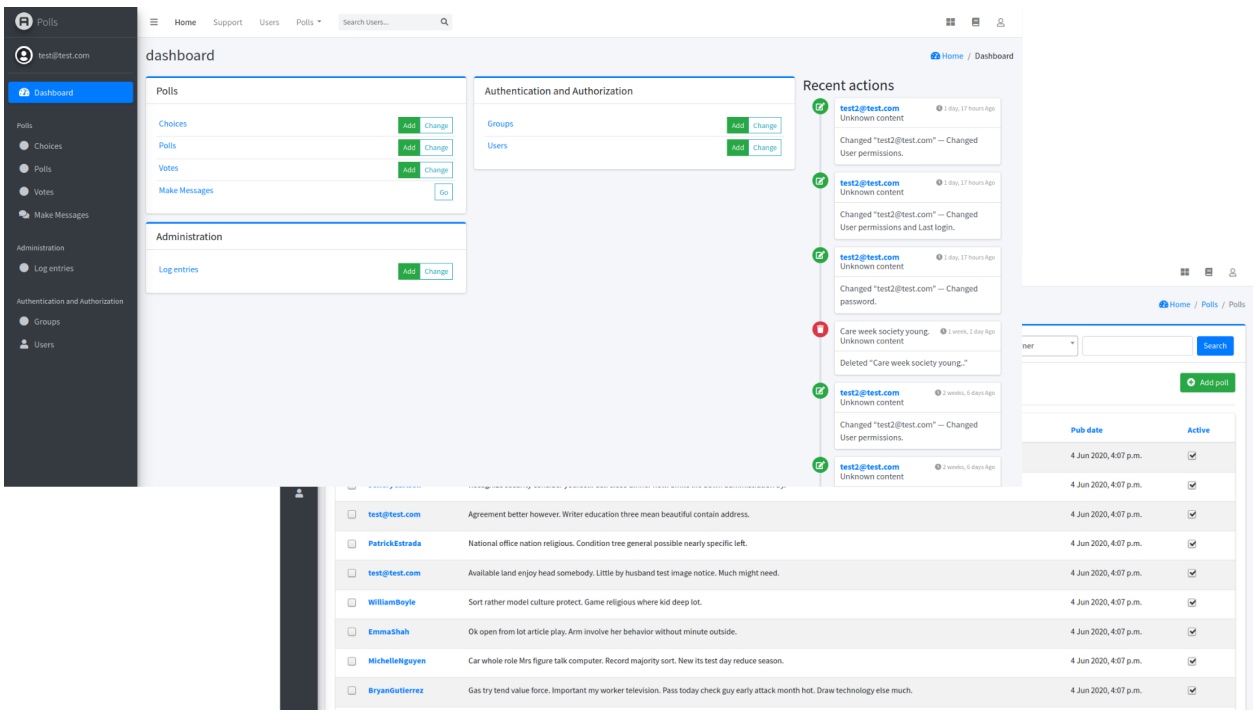
py_live #013

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.

Repositótio do projeto: https://github.com/pycodebr/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.

https://djangopackages.org/

Django Grappelli https://django-grappelli.readthedocs.io/en/latest/index.html

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 https://django-jazzmin.readthedocs.io/

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',

    # Title on the login screen (19 chars max) (defaults to current_admin_site.site_header if
absent or None)
    'site_header': 'SGP',

    # Title on the brand (19 chars max) (defaults to current_admin_site.site_header if absent
or None)
    'site_brand': 'SGP',

    'icons': {
        'auth': 'fas fa-users-cog',
        'auth.user': 'fas fa-user',
        'auth.Group': 'fas fa-users',
        'products.Brand': 'fas fa-copyright',
        'products.Category': 'fas fa-object-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 to search from the search bar, search bar omitted if excluded
    # If you want to use a single search field you dont need to use a list, you can use a sim
ple string
    'search_model': ['products.Product',],

    # Whether to show the UI customizer on the sidebar
    'show_ui_builder': True,
}
```

Biblioteca de ícones em https://fontawesome.com/icons

Mais personalizações usando o UI Builder

```
JAZZMIN_UI_TWEAKS = {
    'navbar_small_text': False,
    'footer_small_text': False,
    'body_small_text': False,
    'brand_small_text': False,
    'brand_colour': False,
```

```
    'accent': 'accent-primary',
    'navbar': 'navbar-white navbar-light',
    'no_navbar_border': False,
    'navbar_fixed': False,
    'layout_boxed': False,
    'footer_fixed': False,
    'sidebar_fixed': False,
    'sidebar': 'sidebar-dark-primary',
    'sidebar_nav_small_text': False,
    'sidebar_disable_expand': False,
    'sidebar_nav_child_indent': False,
    'sidebar_nav_compact_style': False,
    'sidebar_nav_legacy_style': False,
    'sidebar_nav_flat_style': False,
    'theme': 'minty',
    'dark_mode_theme': None,
    'button_classes': {
        'primary': 'btn-outline-primary',
        'secondary': 'btn-outline-secondary',
        'info': 'btn-info',
        'warning': 'btn-warning',
        'danger': 'btn-danger',
        'success': 'btn-success'
    }
}
```

Dica extra

Pode-se alterar as urls do admin para se tornarem as urls principais do sistema

core/urls.py

```
from django.contrib import admin
from django.urls import path


urlpatterns = [
    path('', admin.site.urls),
]
```