Programação em Python (Back-end)

Instrutor: Pablo Araujo

(21) 97172-1697

Aula 5

Revisão – Migração e Model

Migrations: Arquivos gerados pelo Django que descrevem alterações de esquema (criar/alterar tabelas). Comandos essenciais:
python manage.py makemigrations → gera os arquivos de migração.
python manage.py migrate → aplica as migrações no banco.

Entidade (model): classe Python que representa uma tabela no banco; campos da classe → colunas da tabela.

Revisão – Campos Model

• CharField(max_length=...): texto curto (obrigatório passar max_length).
• TextField(): texto longo (sem max_length).
• IntegerField(): número inteiro.
• DecimalField(max_digits=..., decimal_places=...): valores monetários/precisos.
• BooleanField(default=...): verdadeiro/falso.
• DateField() / DateTimeField: data / data e hora.
• EmailField() → campos com validação específica.

Revisão – Atividade

Criar um Cliente com vários campos para fixar tipos e parâmetros.

```
class Cliente(models.Model):
    nome = ?
    cpf = ?
    rg = ?
    email = ?
    telefone = ?
    endereco = ?
    data_nascimento = ?
    sexo = ?
    ativo = ?
    criado_em = ?
    atualizado_em = ?
```

Conhecimento

Admin: Na ultima aula criamos valores na tabela através da interface de admin, porém a interface é para administradores/developers gerirem dados

No dia a dia: aplicações e usuários interagem via interfaces (web/mobile) que fazem requisições HTTP para a aplicação (views / APIs).

Fluxo típico: front-end > requisição HTTP > view Django (validação/negócio) > ORM > banco > response.

Métodos HTTP

Métodos HTTP (resumo prático)
• **GET** → recuperar/consultar recursos (seguro, idempotente).
• **POST** → criar recursos (não idempotente).
• **PUT** → substituir recurso inteiro (idempotente).
• **PATCH** → atualizar parcialmente (não necessariamente idempotente).
• **DELETE** → remover recurso (idempotente).

Observação: ao digitar uma URL no navegador ou clicar link, o método enviado é **GET**.

Conhecimento – Criando URLs

1. Criar a view (‘app’/views.py):

```
from django.http import JsonResponse, HttpResponseNotAllowed
from .models import Cliente

def listar_clientes(request):
    if request.method != "GET":
        return HttpResponseNotAllowed(["GET"])
    clientes = Cliente.objects.all().order_by("nome").values("id","nome","email","ativo")
    return JsonResponse(list(clientes), safe=False)
```

2. Criar clientes/urls.py (se ainda não existir):

```
from django.urls import path
from . import views

urlpatterns = [
    path('clientes/', views.listar_clientes, name='listar_clientes'),
]
```

3. Incluir as urls do app no projeto:

```
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('clientes.urls’)),
]
```

Conhecimento – Testar no navegador

1. Rodar Servidor
python manage.py runserver

2. Acessar no navegador
http://127.0.0.1:8000/clientes/

3. Visualização:
você verá o JSON com todos os clientes ordenados por nome.

Conhecimento – Atividade

1. Faça as etapas já conhecidas:
criar um ambiente virtual, instalar o Django, criar um projeto “MeuProjeto”, criar um app “Produtos”, adicionar no settings e criar superuser

2. Criar a model Produto:
Deverá ter os campos obrigatórios: nome, custo, preço e categoria.

3. Gerar migrações e criar pelo menos 5 registros no admin.

4. Criar a view para listar todos os usuários:

5. Configurar url do app e do projeto(produtos/**)**

6. Testar no navegador

Dúvidas ?

Obrigado!

Instrutor: Pablo Araujo