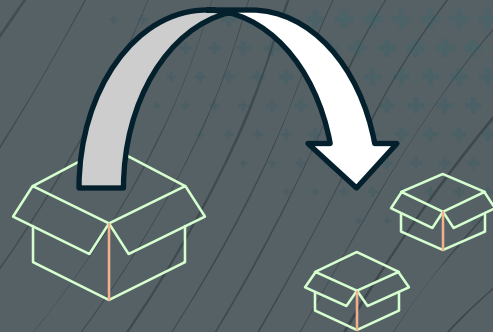


# Convertendo projeto Django Monolítico para Django Rest

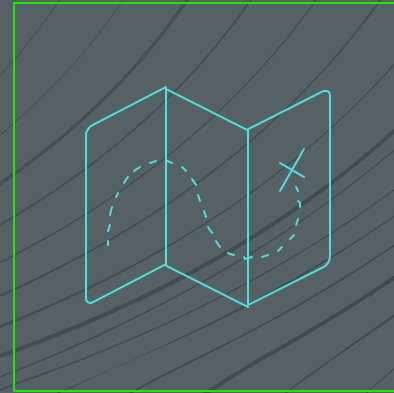
Integrantes: Andrew  
Gabriel

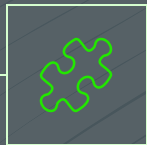


Professor: Ricardo Terra

## → O que será apresentado:

- Contextualização
- Execução / Realização
- Limitações
- Dificuldades







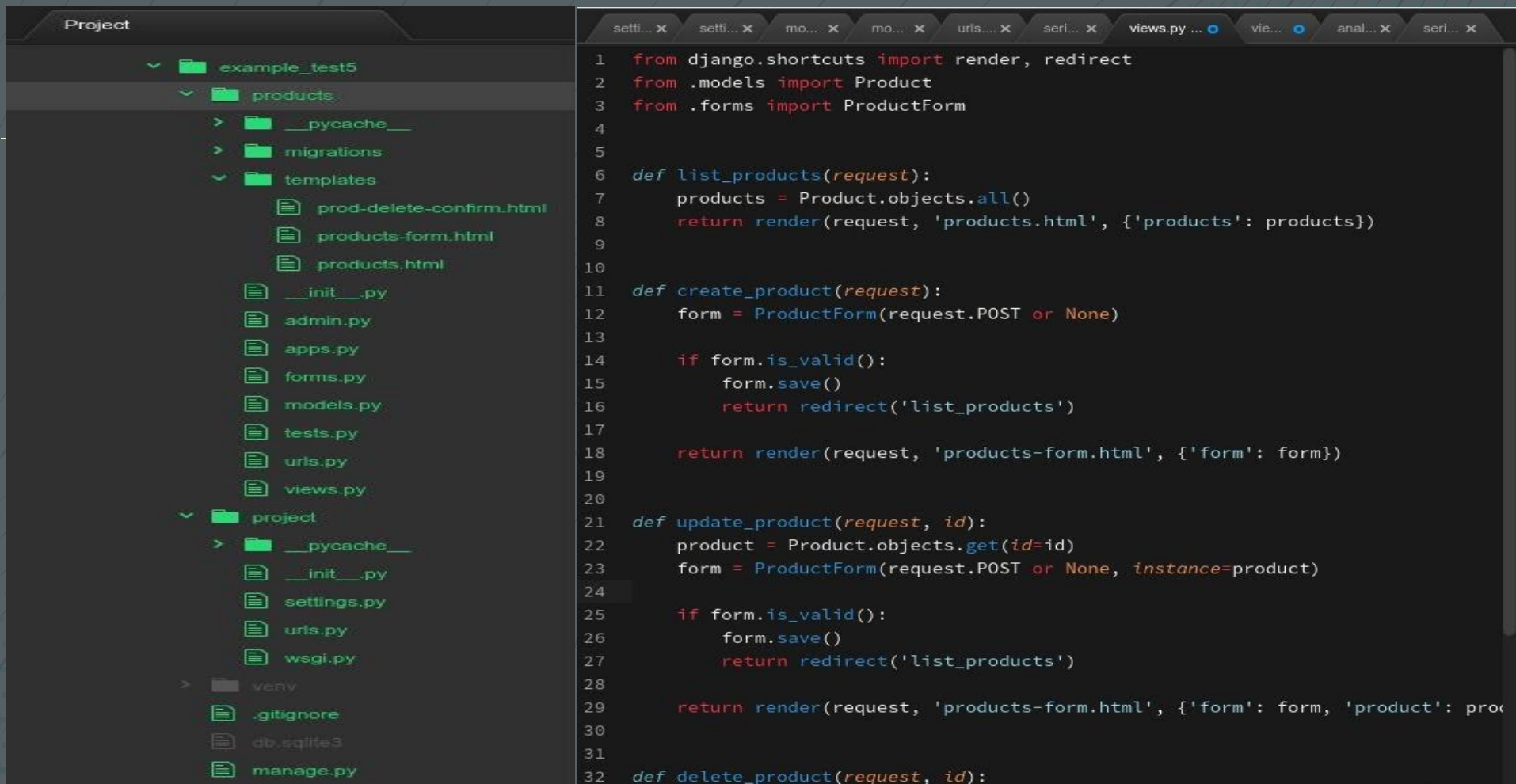
# 1. Contextualização

Passando de um sistema Monolítico em Django para Restful

# Explicações Gerais

- Python (AST, Astor) 
- Rest e Restful
- Django Framework (monolítico) 
- O que são apps no Django
- Django Rest 

# ▣ Django Framework (monolítico)



The image shows a Django project structure on the left and a code editor on the right. The project structure includes a 'Project' folder with subfolders like 'example\_test5', 'products', 'templates', 'project', and 'venv'. The 'products' folder contains files like 'prod-delete-confirm.html', 'products-form.html', 'products.html', 'views.py', 'urls.py', 'models.py', 'forms.py', 'apps.py', 'admin.py', 'tests.py', and 'init.py'. The 'project' folder contains 'init.py', 'settings.py', 'urls.py', 'wsgi.py', and 'manage.py'. The code editor on the right shows the 'views.py' file with the following code:

```
1 from django.shortcuts import render, redirect
2 from .models import Product
3 from .forms import ProductForm
4
5
6 def list_products(request):
7     products = Product.objects.all()
8     return render(request, 'products.html', {'products': products})
9
10
11 def create_product(request):
12     form = ProductForm(request.POST or None)
13
14     if form.is_valid():
15         form.save()
16         return redirect('list_products')
17
18     return render(request, 'products-form.html', {'form': form})
19
20
21 def update_product(request, id):
22     product = Product.objects.get(id=id)
23     form = ProductForm(request.POST or None, instance=product)
24
25     if form.is_valid():
26         form.save()
27         return redirect('list_products')
28
29     return render(request, 'products-form.html', {'form': form, 'product': product})
30
31
32 def delete_product(request, id):
```

# ■ Django Restful

Project

- example\_test5\_restful
  - products
    - \_\_pycache\_\_
    - migrations
      - \_\_init\_\_.py
      - admin.py
      - apps.py
      - forms.py
      - models.py
      - serializers.py
      - tests.py
      - urls.py
      - views.py
  - project
    - \_\_pycache\_\_
    - \_\_init\_\_.py
    - settings.py
    - urls.py
    - wsgi.py
  - venv
  - .gitignore
  - db.sqlite3
  - manage.py

settin... x

settin... x

mod... x

mod... x

urls.py x

serial... x

view... o

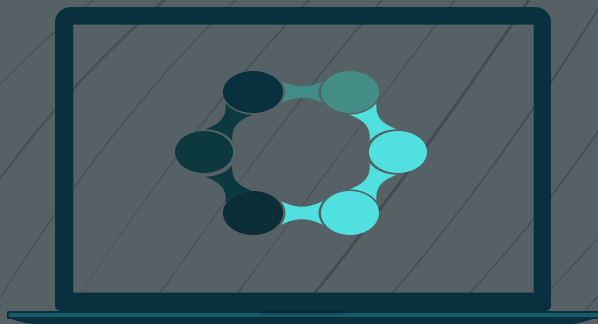
views.py — ex... o

anali... x

```
1 from django.shortcuts import render, redirect
2 from .models import Product
3
4 from rest_framework.response import Response
5 from .serializers import ProductSerializer
6 from rest_framework.decorators import api_view
7
8
9 @api_view(['GET'])
10 def list_products(request):
11     products = Product.objects.all()
12     serializer = ProductSerializer(products, many=True)
13
14     return Response(serializer.data)
15
16 @api_view(['POST'])
17 def create_product(request):
18     serializer = ProductSerializer(data=request.data)
19
20     if serializer.is_valid():
21         serializer.save()
22
23     return Response(serializer.data)
24
25 @api_view(['POST'])
26 def update_product(request, id):
27     product = Product.objects.get(id=id)
28     serializer = ProductSerializer(data=request.data, instance=product)
29
30     if serializer.is_valid():
31         serializer.save()
32
```



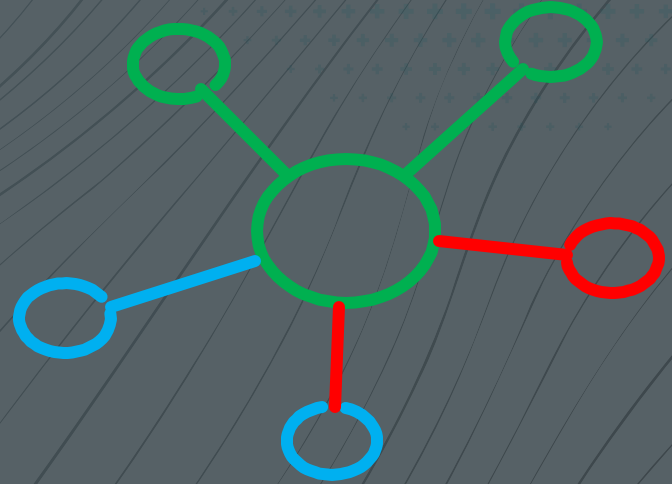
## 2. Execução / Realização







# 3. Limitações







## 4. Dificuldades





Obrigado !

