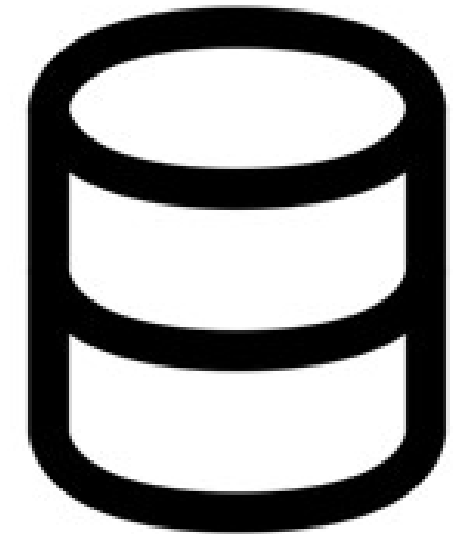




Exercício

- Primeiro vamos rever as duplas da atividade anterior.
- Verifique a conexão com o postgres.
- Crie um novo
 - **projeto Django** (Nome de sua preferencia)
 - **Aplicação** (Nome de sua preferencia)
 - Postgres **database** (Nome de sua preferencia)
- Instale
 - Pip install psycopg2-binary
 - Pip install djangoestframework
 - Pip install django-filter
 - Pip freeze > requirements.txt
- Atualize **settings.py** (INSTALLED APPS) / (DATABASES)



database

Exercício

- Crie os modelos feitos por vocês da atividade em dupla.
 - Faça Herança de um modelo-base (ModelBase)
 - Não esqueça do META
 - Crie os campos de cada um dos modelos.
 - Fique atento aos PKs e FKs
 - Adicione a função built-in “__str__” pra deixar mais bonito.
- No terminal local, realize as **migrações**.
 - Python manage.py showmigrations
 - Python manage.py makemigrations
 - Python manage.py migrate



models

Exercício

- Na Aplicação,
 - Crie arquivo novo chamado “**serializers.py**”
 - Crie as validações que façam sentido pro projeto de vocês.
 - Use Check de min_value/max_value
 - Use as validações de PK e FK
 - Filtre pra mostrar somente os campos (Fields) que você quer.
- No arquivo “**views.py**”
 - Crie os viewsets que configuram os endpoints.
 - Cada viewset deve conter queryset e serializers. Pois ele une-os.



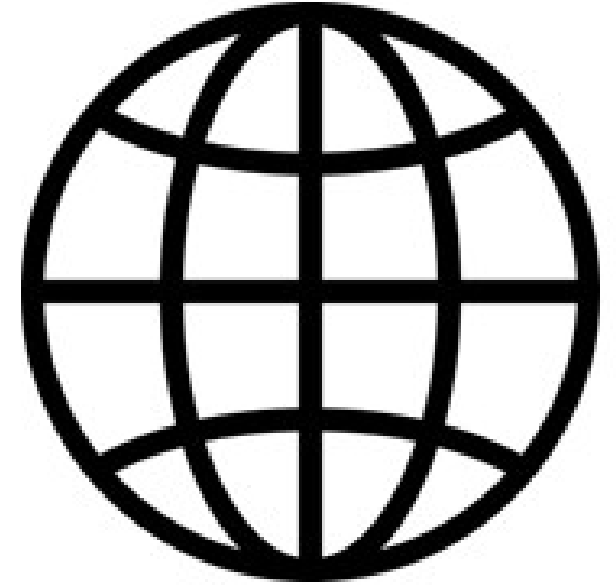
Serializers
JSON=Django



viewsets

Exercício

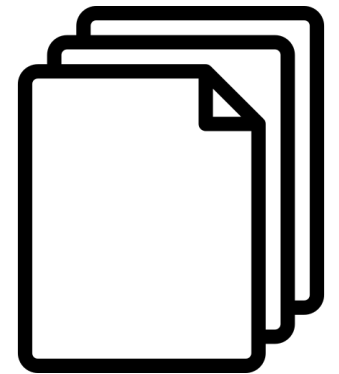
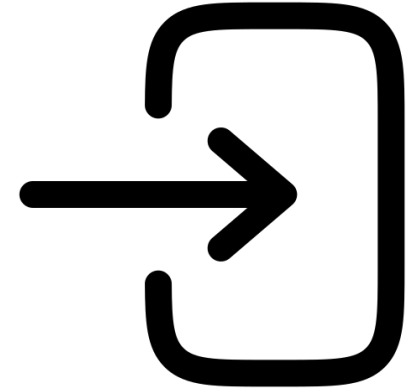
- Na Aplicação
 - Crie arquivo novo “**urls.py**”
 - Importem path, include, DefaultRouter
 - Importem os viewsets e criem um router
 - Registrem as rotas pra cada viewset
 - Salvem o router
- Vá para o projeto Django > arquivo “**urls.py**”
 - Importe o router com as Urls que você fez no passo anterior.
 - atualize incluindo as urls da aplicação de vocês.



endpoint
urls

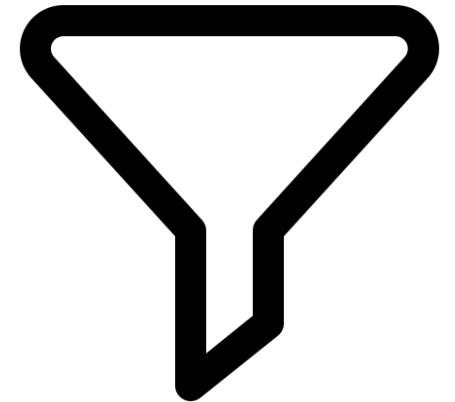
Exercício

- Faça permissões
 - No arquivo **views.py** atualize as viewsets
 - Importe “permissions” (rest_framework)
 - Adicione “permission_classes = [permissions.IsAuthenticated]” em todos os viewsets.
- Faça Paginação
 - No arquivo **settings.py** crie uma nova variável.
 - REST_FRAMEWORK
 - 'DEFAULT_PAGINATION_CLASS':
'rest_framework.pagination.LimitOffsetPagination',



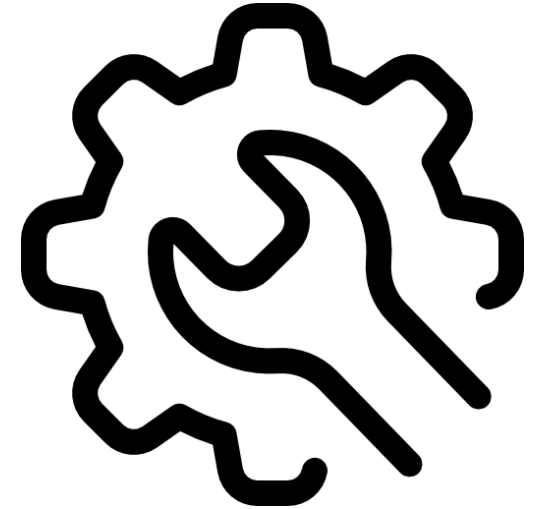
Exercício

- No projeto Django, **settings.py**
 - Adicione a linha
 - 'DEFAULT_FILTER_BACKENDS': 'django_filters.rest_framework.DjangoFilterBackend'
 - Dentro da variável REST_FRAMEWORK
- Na Aplicação, Faça Filtros
 - Crie um novo arquivo **filters.py** na aplicação
 - Escreva os filtros que fazem sentido para seu projeto
 - Use lookup expressions. (icontains, exact, gte, lte,)
- Importe esses filtros para o arquivo **view.py**
 - Importe também DjangoFilterBackend
 - Adicione duas linhas em cada viewset:
 - Filter_backends = [DjangoFilterBackend]
 - Filterset_class = (o seu filtro aqui)



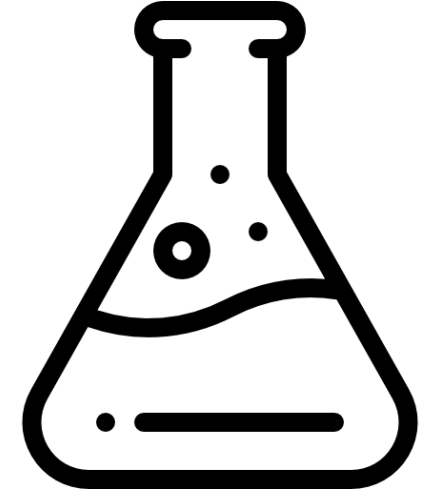
Exercício

- Crie um novo super usuário
 - Python manage.py createsuperuser
- Configure seu aplicativo para rodar
 - Veja se é necessário ver as configurações do interpretador (python)
 - Python manage.py runserver
 - Ou abra a edição de rodar o projeto, atualize para 'localhost', '8000', e
 - PYTHONUNBUFFERED=1;DJANGO_SETTINGS_MODULE=**seuProjDjango**.settings
- Rode ou aperte 'Play' para rodar



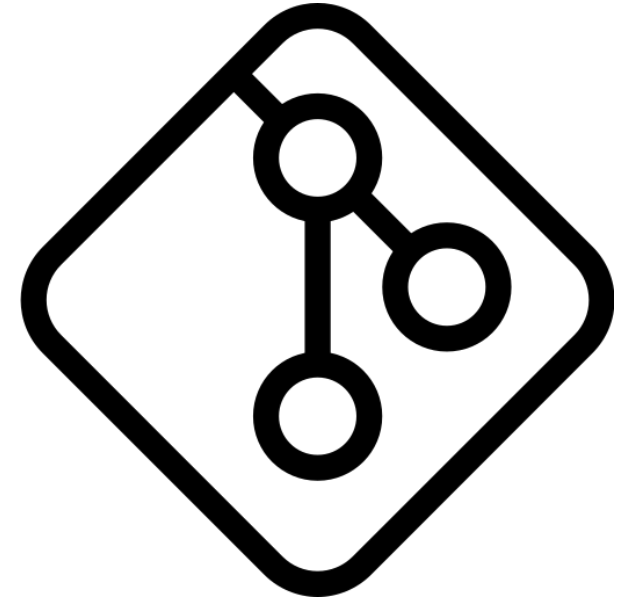
Exercício

- Vamos Testar
- Faça Login
- Faça (pelo menos) 3 POSTS de:
 - Cada uma das Entidades que você criou
- Escolha 1 das Entidade e Faça o GET com limit e offset
 - Faça novamente o GET, mas dessa vez, diretamente pela URL
- Escolha outra Entidade e Faça um GET filtrando por Id
- Escolha outra Entidade e Faça um GET filtrando por 2 campos da sua escolha
 - Faça novamente o GET, mas dessa vez, diretamente pela URL



Exercício

- Faça o link do projeto Django com sua conta GitHub.
- Faça o GIT “ver” seus arquivos
 - Prepare suas alterações para commitar
 - Realize o commit com uma mensagem.
- Faça o **push** do seu projeto para uma Branch “main” remota.
- Veja seu novo repositório no site GitHub



Obrigado!



    [@fpftech.educacional](https://www.instagram.com/fpftech.educacional)