Acesso ao banco de dados a partir de uma aplicação

1) Baixe o arquivo **dvdrental.tar** do classroom, ele é um backup de um banco de dados do Postgresql

2) Inicie o container do Postgresql criado anteriormente, aqui vou me referir a ele como postgres-tads. As informações de como criar o container estão no material da **Aula 2 – 05/09/2023**!
`docker start  postgres-tads`

3) Copie o arquivo dvdrental.tar para o container, esse comando deve ser executo a partir da pasta onde o arquivo dvdrental.tar está armazenado
`docker cp dvdrental.tar postgres-tads:/dvdrental.tar`

4) Acesse o container via linha de comando
`docker exec -it postgres-tads bash`

5) Mude para o usuário postgres
`su - postgres`

6) Acesse o postgres através do cliente modo texto
`psql`

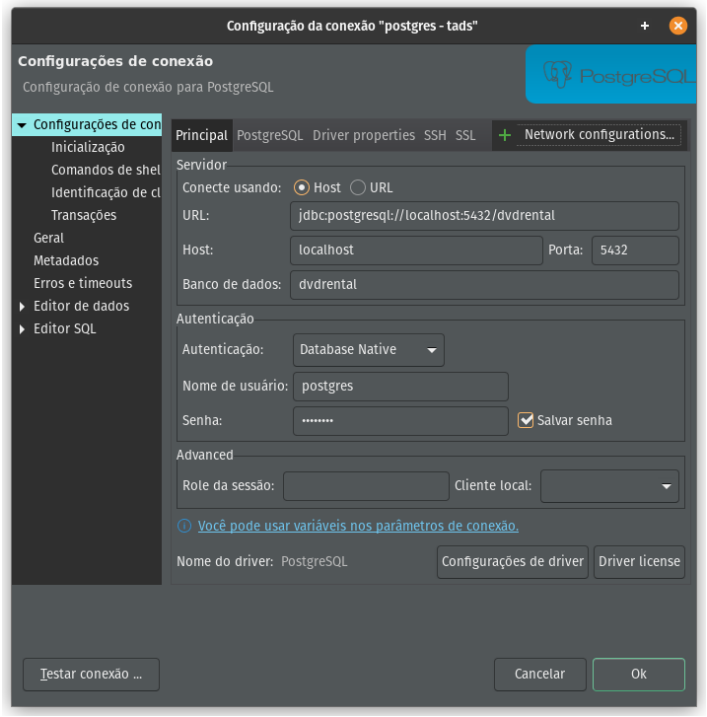
7) Crie o banco de dados **dvdrental**
`CREATE DATABASE dvdrental;`

8) Saia do cliente digitando \q e ENTER

9) Restaure o backup do banco
`pg_restore -U postgres -d dvdrental /dvdrental.tar`

Encerre a sessão do container usando o comando `exit` duas vezes. A primeira para sair do usuário postgres (passo 5) e a segunda para sair da sessão do container (passo 4)

10) Crie uma conexão com o banco de dados através do DBeaver e verifique se as tabelas foram criadas e populadas! Informe corretamente o nome da banco de dados na tela de conexão!



11) Instale o python versão 3 em seu computador! No Linux ele já vem instalado, mas no MAC e no Windows pode ser necessário fazer a instalação dessa versão. Verifique se o python está corretamente instalado com o comando abaixo e siga as orientações da página oficial se for necessário: https://wiki.python.org/moin/BeginnersGuide/Download

```
python3 --version
```

12) Crie um diretório para sua aplicação. Aqui não vou informar o caminho atual, devido as diferenças de como os SOs lidam com isso!

```
mkdir dvdrental
```

ou

```
md dvdrental
```

13) Entre no diretório criado

```
cd dvdrental
```

14) Crie um ambiente virtual para o python3 com o comando abaixo, se não funcionar deve ser necessário instalar esse módulo (virtualenv) em seu computador. Esse link https://rasa.com/docs/rasa/installation/environment-set-up/ pode trazer alguma orientação

```
python3 -m venv .venv
```

15) Ative o ambiente virtual

```
source .venv/bin/activate
```

ou

```
.venv\Scripts\activate.bat
```

16) Instale o Django

```
pip install django
```

17) Instale o binário do driver para conexão com o postgres, qualquer problema com essa instalação pode solicitar instalção de outros programas no computador para resolver. Faça uma busca para o seu problema específico.

```
pip install psycopg2-binary
```

18) Crie o projeto Django com o comando abaixo. Não esqueça do ponto no final do comando!

```
django-admin startproject dvdrental .
```

19) Crie um aplicação para o projeto com o comando abaixo

```
django-admin startapp main
```

20) Abra o projeto no vscode e faça as seguintes modificações no arquivo **dvdrental/settings.py**

a) Adicione a aplicação main à lista de INSTALLED_APPS

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'main'
]
```

b) Comente a configuração do SQLite3

```
# DATABASES = {
#     'default': {
#         'ENGINE': 'django.db.backends.sqlite3',
#         'NAME': BASE_DIR / 'db.sqlite3',
#     }
# }
```

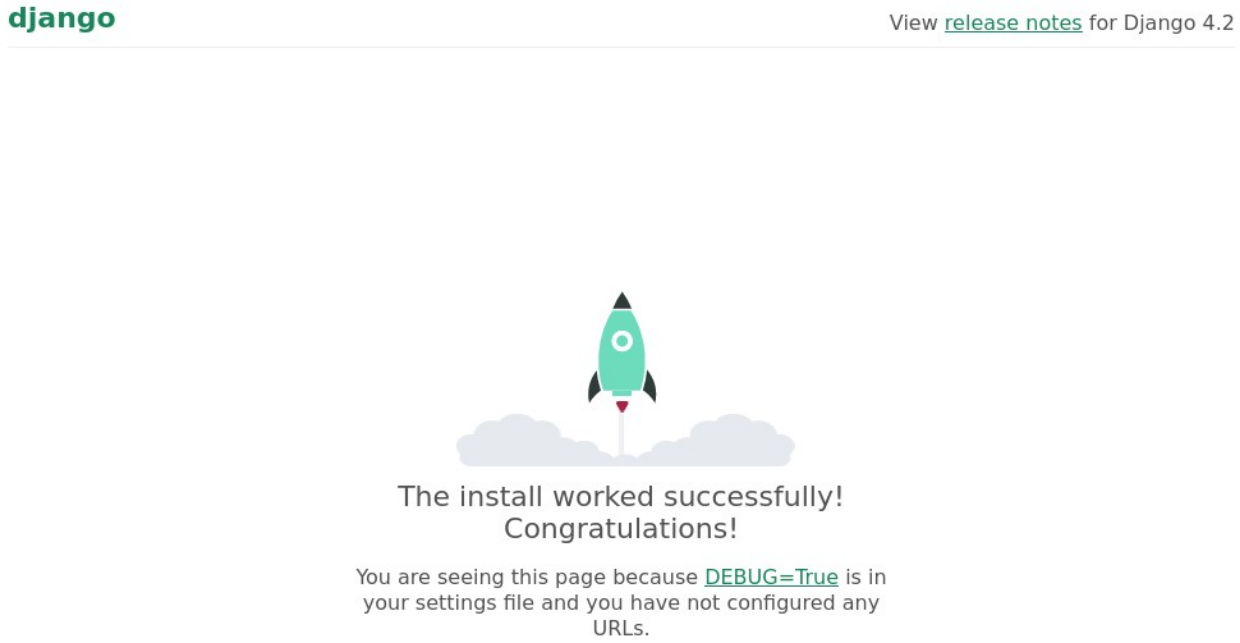
c) Insira logo abaixo ao texto anterior a configuração de acesso ao Postgres, configure os parâmetros abaixo de acordo com o dados de acesso de seu banco de dados

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'dvdrental',
        'USER': 'postgres',
        'PASSWORD': 'postgres',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}
```

20) Teste o projeto com o comando abaixo e abra o navegador no endereço **http://localhost:3131/**

```
python manage.py runserver 0.0.0.0:3131
```

Deve ser exibido a página abaixo:



21) Interrompa com um **CTRL+C** após o teste, caso algum erro seja apresentado pode ser devido ao ambiente de execução que não ficou corretamente configurado!

22) Execute o comando abaixo para criar o arquivo models.py a partir do banco de dados existente e confira o arquivo main/models.py no vscode

```
python manage.py inspectdb > main/models.py
```

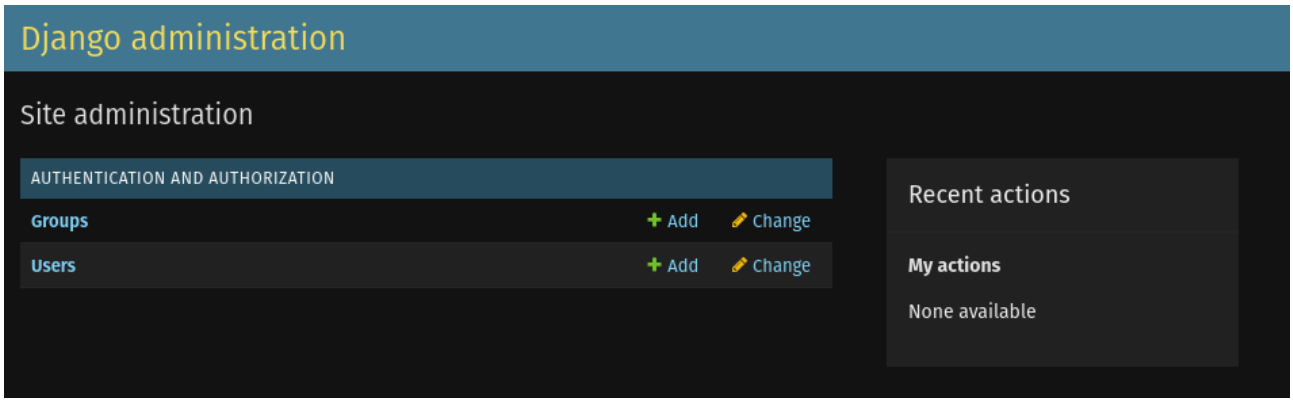
23) Faça a migração das modificações necessárias para o Django no banco de dados dvdrental com o comando abaixo:
`python manage.py migrate`

24) Cria uma conta de superusuário para o sistema com o comando abaixo. O e-mail é opcional!
`python manage.py createsuperuser`

```
$ python manage.py createsuperuser
Username (leave blank to use 'henrique'): admin
Email address:
Password:
Password (again):
The password is too similar to the username.
This password is too short. It must contain at least 8 characters.
This password is too common.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
```

25) Inicie novamente o projeto como mostrado no passo 20 e acesse o endereço
http://localhost:3131/admin/

26) Faça o login com o usuário criado no passo 24, a página abaixo deve ser exibida:

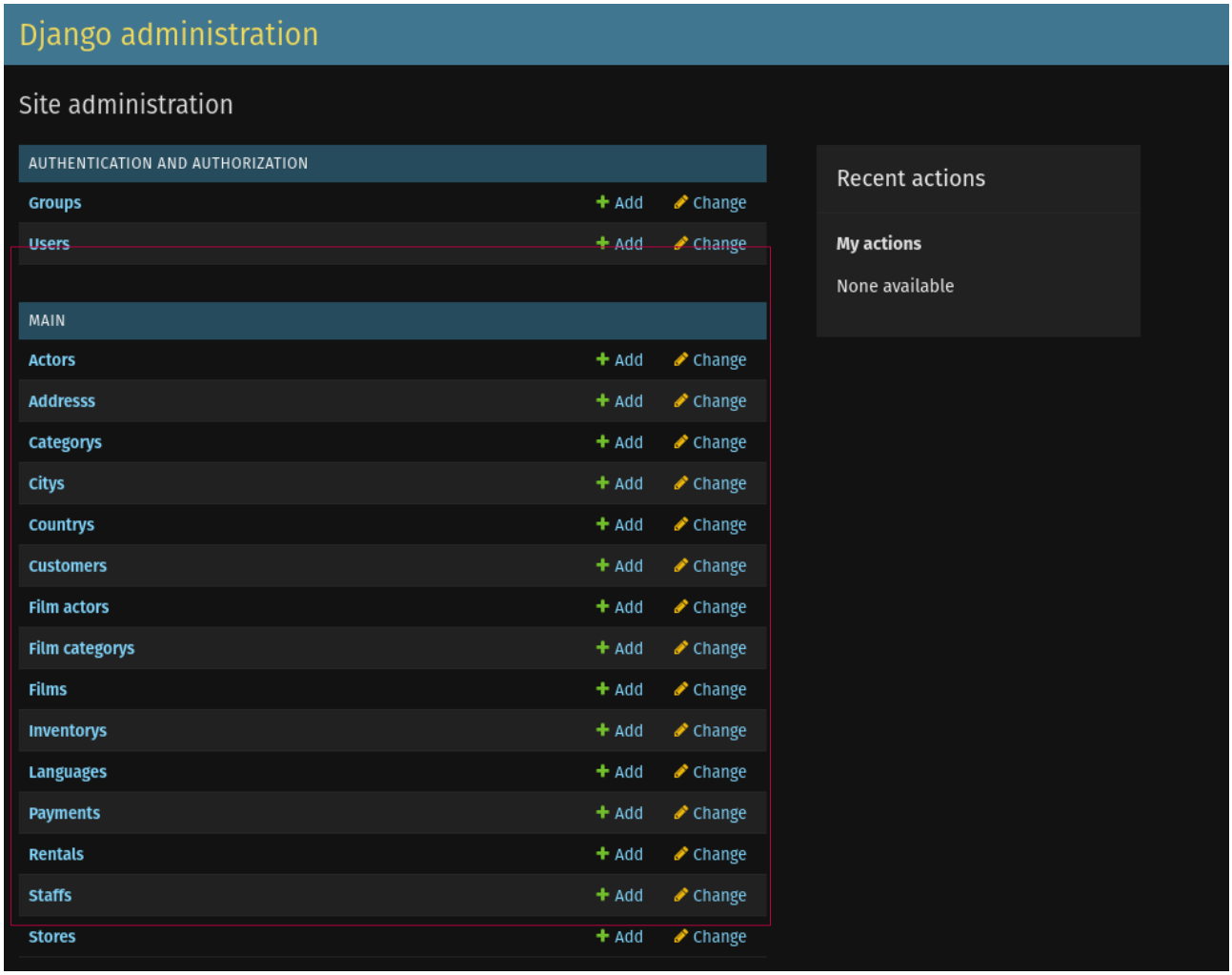


27) Interrompa com um CTRL+C e modifique o arquivo **main/admin.py** apara o conteúdo abaixo:

```
from django.contrib import admin
from main.models import (
    Actor,
    Address,
    Category,
    City,
    Country,
    Customer,
    Film,
    FilmActor,
    FilmCategory,
    Inventory,
    Language,
    Payment,
    Rental,
    Staff,
    Store
)
```

admin.site.register(Actor)
admin.site.register(Address)
admin.site.register(Category)
admin.site.register(City)
admin.site.register(Country)
admin.site.register(Customer)
admin.site.register(Film)
admin.site.register(FilmActor)
admin.site.register(FilmCategory)
admin.site.register(Inventory)
admin.site.register(Language)
admin.site.register(Payment)
admin.site.register(Rental)
admin.site.register(Staff)
admin.site.register(Store)

28) Inicie novamente o projeto como mostrado no passo 20 e acesse o endereço **http://localhost:3131/admin/** e explore as novas opções adicionadas, destacadas em vermelho, com o comando do passo 27



29) Interrompa com um **CTRL+C** e modifique o arquivo **main/views.py** para o conteúdo abaixo:

```
from django.http import HttpResponse
from django.core import serializers
from main.models import *
```

```
def filmes_view(request):
    qs = Film.objects.all()
    qs_json = serializers.serialize('json', qs)
    return HttpResponse(qs_json, content_type='application/json')

def atores_view(request):
    qs = Actor.objects.all()
    qs_json = serializers.serialize('json', qs)
    return HttpResponse(qs_json, content_type='application/json')

def aluguel_view(request):
    qs = Rental.objects.filter(inventory__film__language__name = 'English')
    print(qs.query)
    qs_json = serializers.serialize('json', qs)
    return HttpResponse(qs_json, content_type='application/json')
```

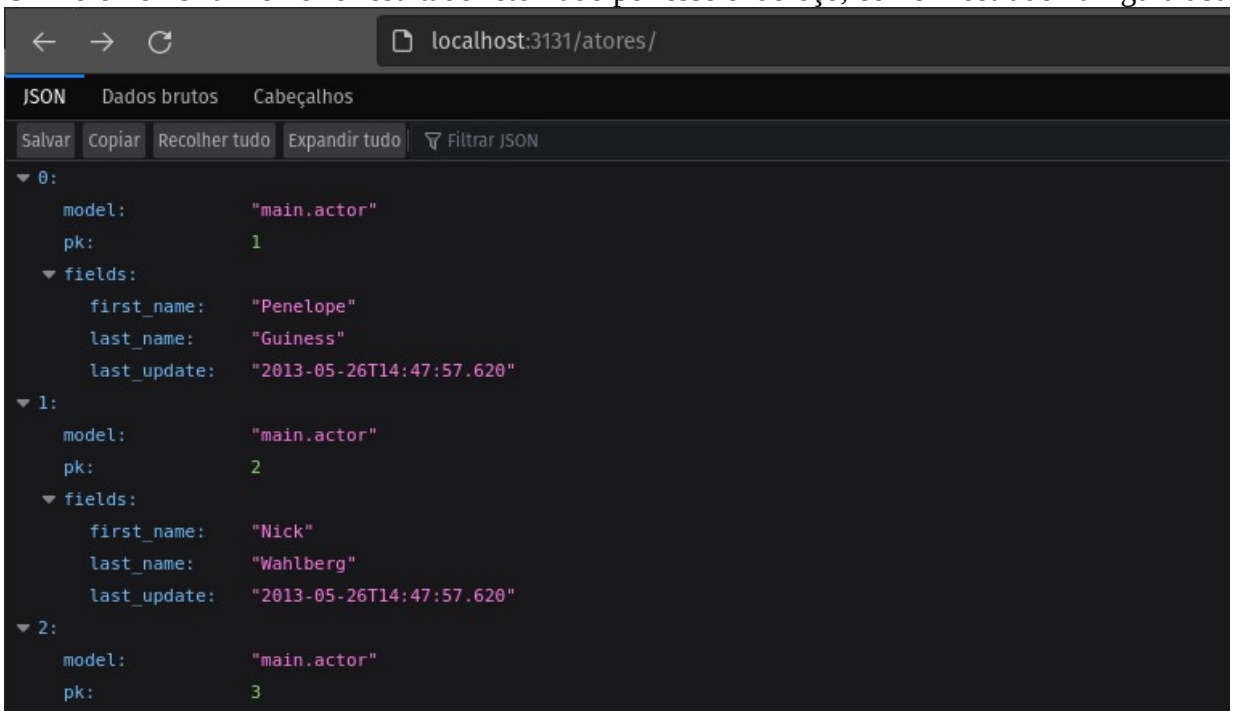
29) Modifique o arquivo **dvdrental/urls.py** para o conteúdo abaixo:

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

urlpatterns = [
    path('admin/', admin.site.urls),
    path("filmes/", views.filmes_view),
    path("atores/", views.atores_view),
    path("alugueis/", views.aluguel_view),

]
```

30) Inicie novamente o projeto como mostrado no passo 20 e acesse os endereços:

- http://localhost:3131/atores/
- http://localhost:3131/filmes/
- http://localhost:3131/alugueis/

O Firefox exibirá melhor o resultado retornado por esse endereço, como mostrado na figura abaixo:



31) Modifique o arquivo **dvdrental/settings.py** e adicione as linhas abaixo ao final do arquivo. Essa modificação exibirá os comandos SQL que são enviados para o BD no terminal onde o sistema foi iniciado como mostrado no passo 20

```
LOGGING = {
    'version': 1,
    'filters': {
        'require_debug_true': {
            '()': 'django.utils.log.RequireDebugTrue',
        }
    },
    'handlers': {
        'console': {
            'level': 'DEBUG',
            'filters': ['require_debug_true'],
            'class': 'logging.StreamHandler',
        }
    },
    'loggers': {
        'django.db.backends': {
            'level': 'DEBUG',
            'handlers': ['console'],
        }
    }
}
```

Exemplo da saída no terminal:

```
henrique at pop-os in ~/dev/IFPE/dvdrental (.venv)
$ python manage.py runserver 0.0.0.0:3131
Performing system checks...

System check identified no issues (0 silenced).
(0.002)
            SELECT
                c.relname,
                CASE
                    WHEN c.relispartition THEN 'p'
                    WHEN c.relkind IN ('m', 'v') THEN 'v'
                    ELSE 't'
                END,
                obj_description(c.oid, 'pg_class')
            FROM pg_catalog.pg_class c
            LEFT JOIN pg_catalog.pg_namespace n ON n.oid = c.relnamespace
            WHERE c.relkind IN ('f', 'm', 'p', 'r', 'v')
                AND n.nspname NOT IN ('pg_catalog', 'pg_toast')
                AND pg_catalog.pg_table_is_visible(c.oid)
        ; args=None; alias=default
(0.000) SELECT "django_migrations"."id", "django_migrations"."app", "django_migrations"."name", "django_migra
tions"."applied" FROM "django_migrations"; args=(); alias=default
September 27, 2023 - 03:51:28
Django version 4.2.5, using settings 'dvdrental.settings'
Starting development server at http://0.0.0.0:3131/
Quit the server with CONTROL-C.

(0.001) SELECT "actor"."actor_id", "actor"."first_name", "actor"."last_name", "actor"."last_update" FROM "act
or"; args=(); alias=default
```

32) Copie os comandos para o DBeaver e veja o plano de execução de cada um.

Obs:

- Ignore a parte final do comando (args e alias)
- Remova as aspas extras adicionada pelo DBeaver quando o comando é colado

33) Outra forma de ver os comandos executados no postgres é fazendo uma alteração em seu arquivo de configuração. Essa página https://tableplus.com/blog/2018/10/how-to-show-queries-log-in-postgresql.html explica como realizar esse processo, mas no passo “Then restart the server” basta reiniciar o container com o comando abaixo:

```
docker restart postgres-tads
```