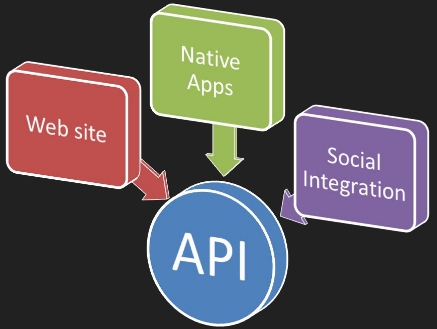
django REST framework

django



django REST framework

Conteúdo

- O Projeto (Setup)
- CBV: API View
  a. Métodos (get, post, put, delete)
  b. Response, Status Code
  c. Roteamento (URLS)
- Serializer
  a. data vs Instance
  b. ModelSerializer
  c. Serializers Fields
  d. Validação Geral e Atributos
  e. Métodos: create, update, validade
  f. Relacionamentos (related)
- CBV: GenericViews
  a. list, create, retrieve, update, destroy
- Permissão
- Autenticação
- Paginação
- Caching
- Throttling
- Filtering
- Pagination

Instalação

```
# Terminal
$ pip install django
$ pip install djangorestframework


# settings.py
INSTALLED_APPS = [
    ...
    'rest_framework',
  ]
```

Artefatos Principais

Views

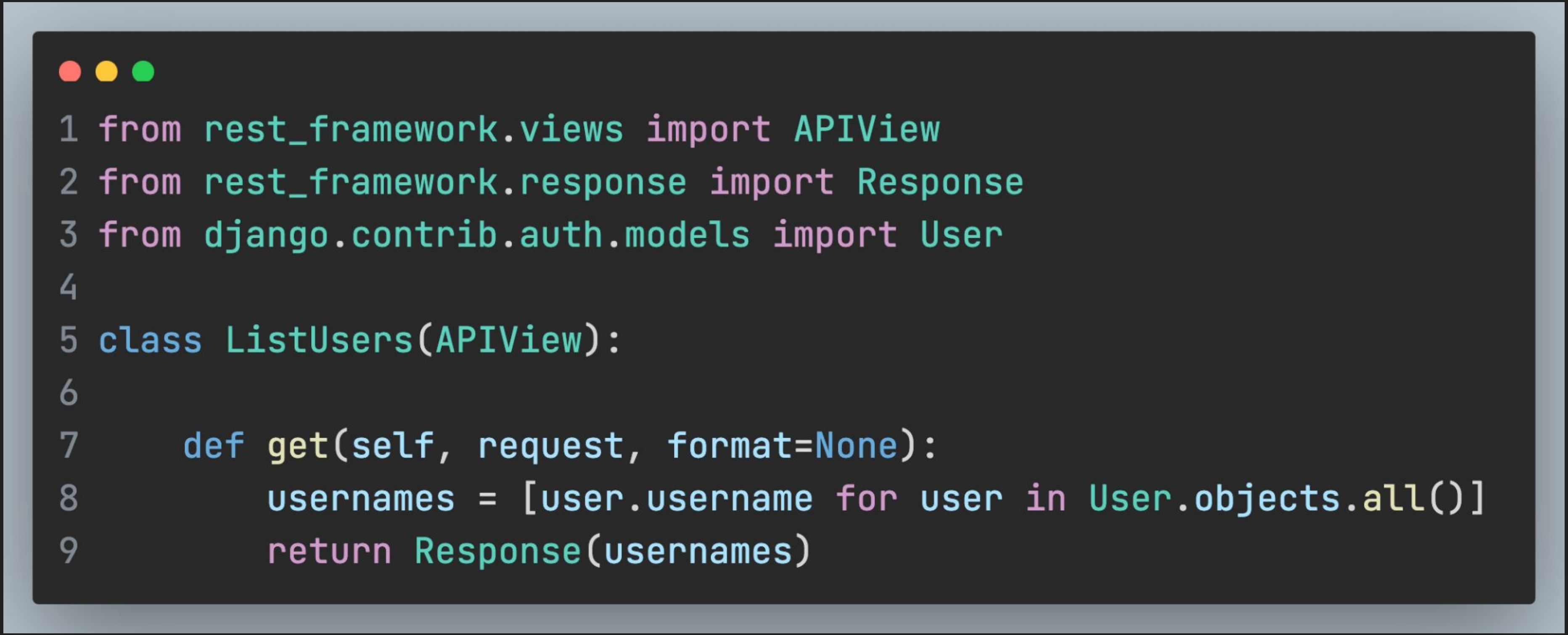
Request & Response

Serializers

JSON ↔ Objetos, Validação

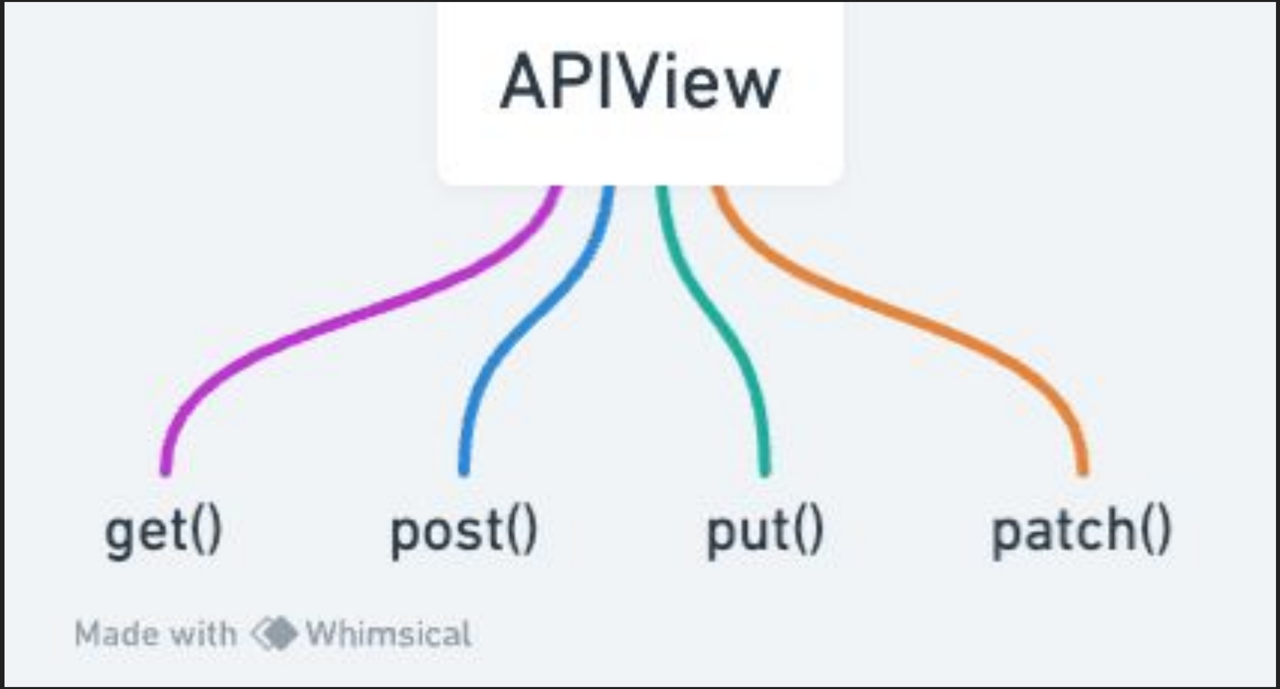
django REST framework

APIView *- Classe Based*



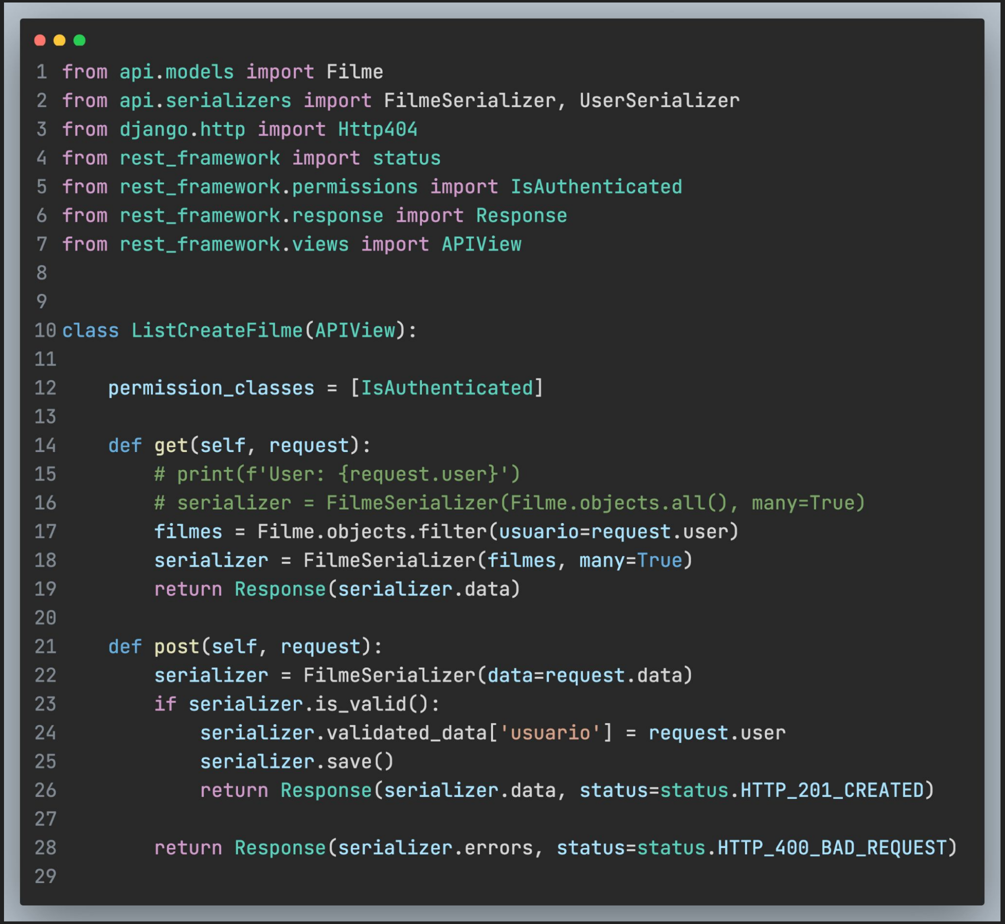
```python
from rest_framework.views import APIView
from rest_framework.response import Response
from django.contrib.auth.models import User

class ListUsers(APIView):

    def get(self, request, format=None):
        usernames = [user.username for user in User.objects.all()]
        return Response(usernames)
```

APIView: Classe Methods



Exemplos APIView:

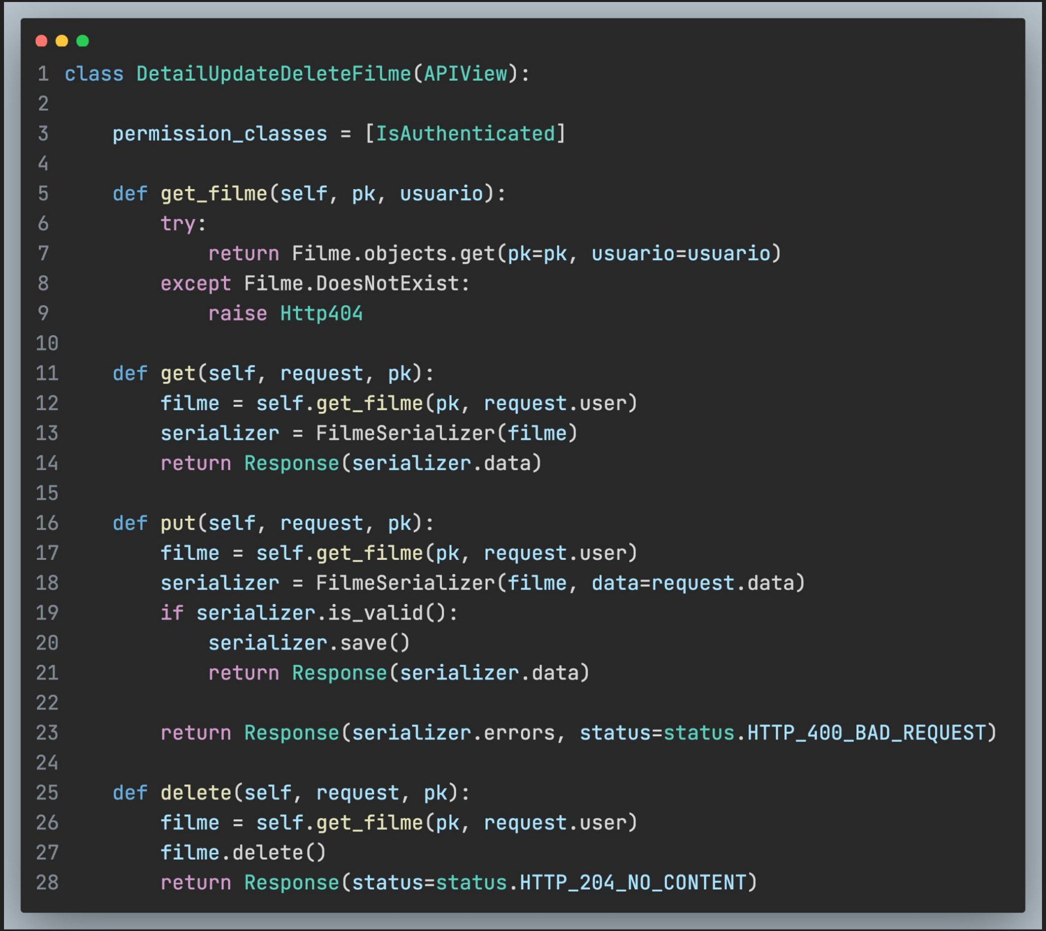
- get(all)
- post



```python
from api.models import Filme
from api.serializers import FilmeSerializer, UserSerializer
from django.http import Http404
from rest_framework import status
from rest_framework.permissions import IsAuthenticated
from rest_framework.response import Response
from rest_framework.views import APIView


class ListCreateFilme(APIView):

    permission_classes = [IsAuthenticated]

    def get(self, request):
        # print(f'User: {request.user}')
        # serializer = FilmeSerializer(Filme.objects.all(), many=True)
        filmes = Filme.objects.filter(usuario=request.user)
        serializer = FilmeSerializer(filmes, many=True)
        return Response(serializer.data)

    def post(self, request):
        serializer = FilmeSerializer(data=request.data)
        if serializer.is_valid():
            serializer.validated_data['usuario'] = request.user
            serializer.save()
            return Response(serializer.data, status=status.HTTP_201_CREATED)

        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

```

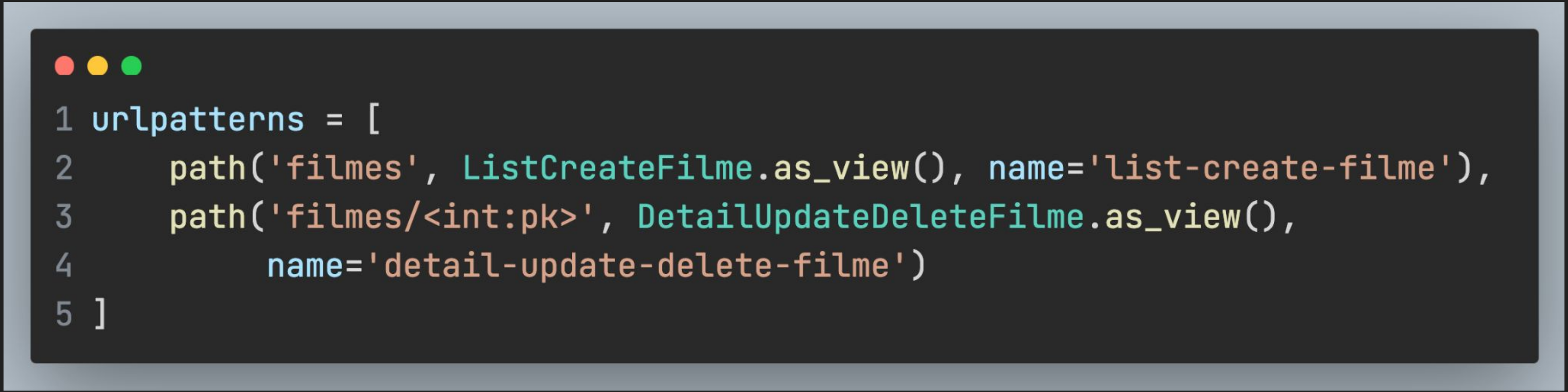
Exemplos APIView:

- get(id)
- put
- delete



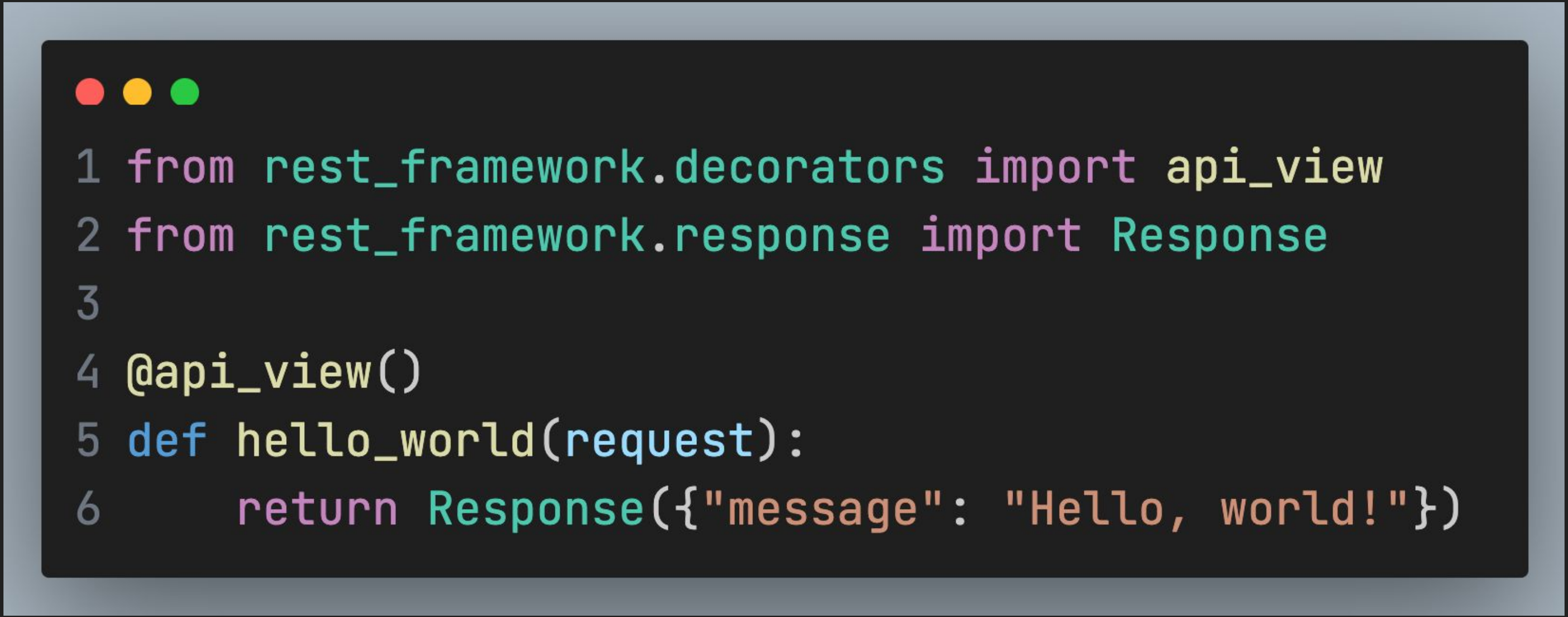
```python
class DetailUpdateDeleteFilme(APIView):

    permission_classes = [IsAuthenticated]

    def get_filme(self, pk, usuario):
        try:
            return Filme.objects.get(pk=pk, usuario=usuario)
        except Filme.DoesNotExist:
            raise Http404

    def get(self, request, pk):
        filme = self.get_filme(pk, request.user)
        serializer = FilmeSerializer(filme)
        return Response(serializer.data)

    def put(self, request, pk):
        filme = self.get_filme(pk, request.user)
        serializer = FilmeSerializer(filme, data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data)

        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

    def delete(self, request, pk):
        filme = self.get_filme(pk, request.user)
        filme.delete()
        return Response(status=status.HTTP_204_NO_CONTENT)
```

Mapeando Rotas



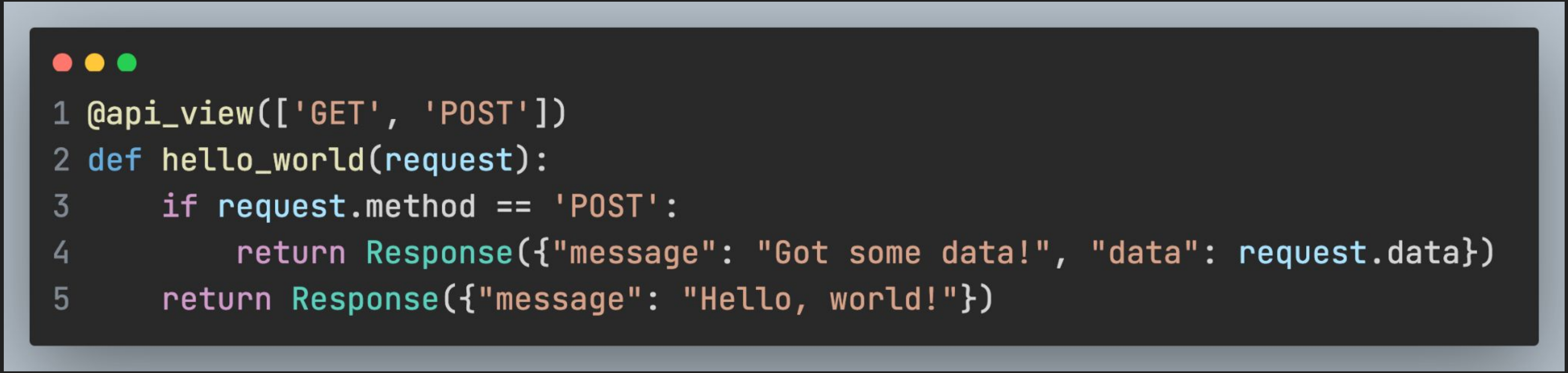
```python
urlpatterns = [
    path('filmes', ListCreateFilme.as_view(), name='list-create-filme'),
    path('filmes/<int:pk>', DetailUpdateDeleteFilme.as_view(),
         name='detail-update-delete-filme')
]
```

APIView *- Function Based*



```python
from rest_framework.decorators import api_view
from rest_framework.response import Response

@api_view()
def hello_world(request):
    return Response({"message": "Hello, world!"})
```

APIView *- Function Based*



```python
@api_view(['GET', 'POST'])
def hello_world(request):
    if request.method == 'POST':
        return Response({"message": "Got some data!", "data": request.data})
    return Response({"message": "Hello, world!"})
```

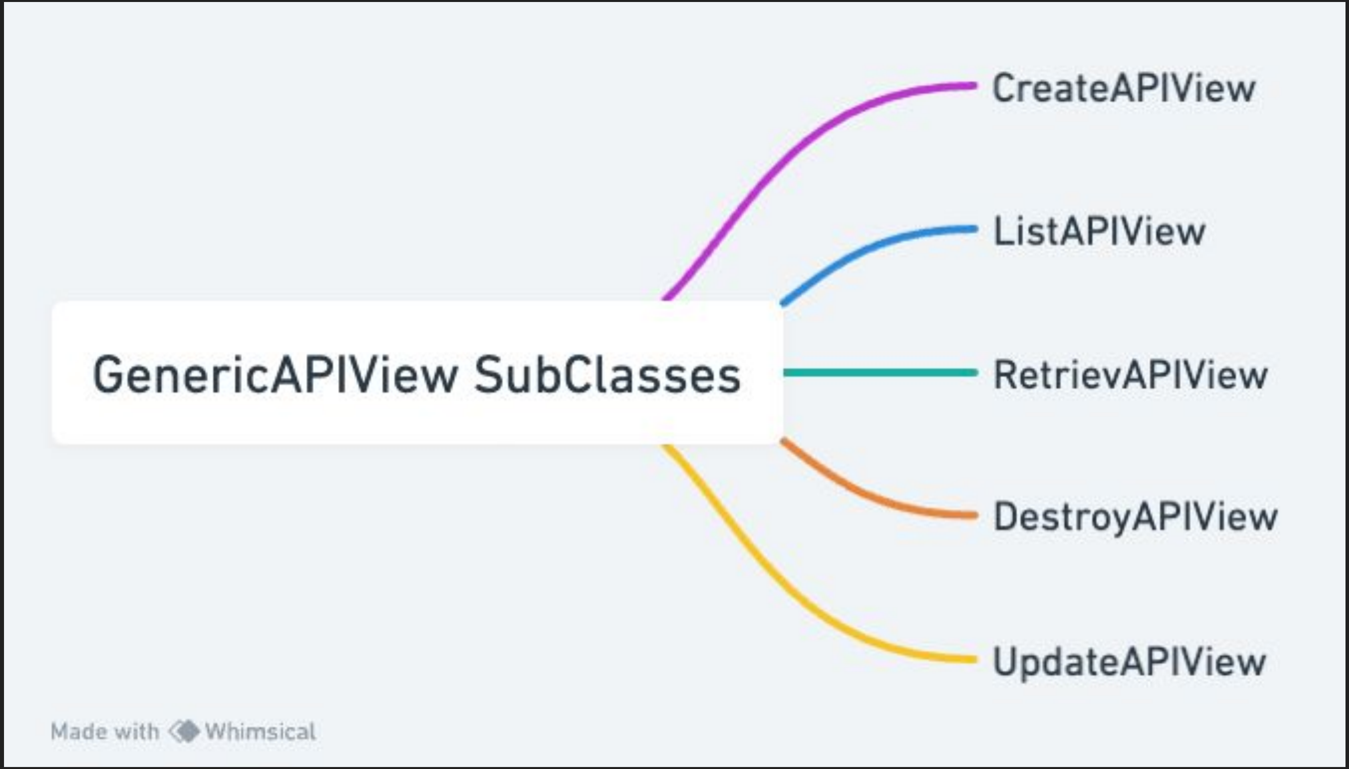
Generic Views

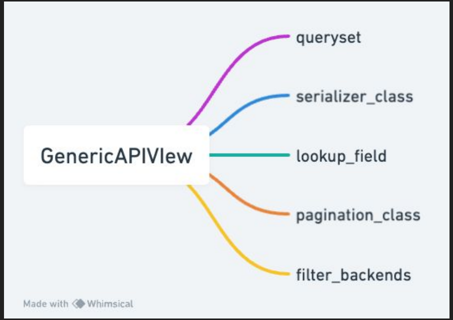
"

Django's generic views... were developed as a shortcut for common usage patterns... They take certain common idioms and patterns found in view development and abstract them so that you can quickly write common views of data without having to repeat yourself.

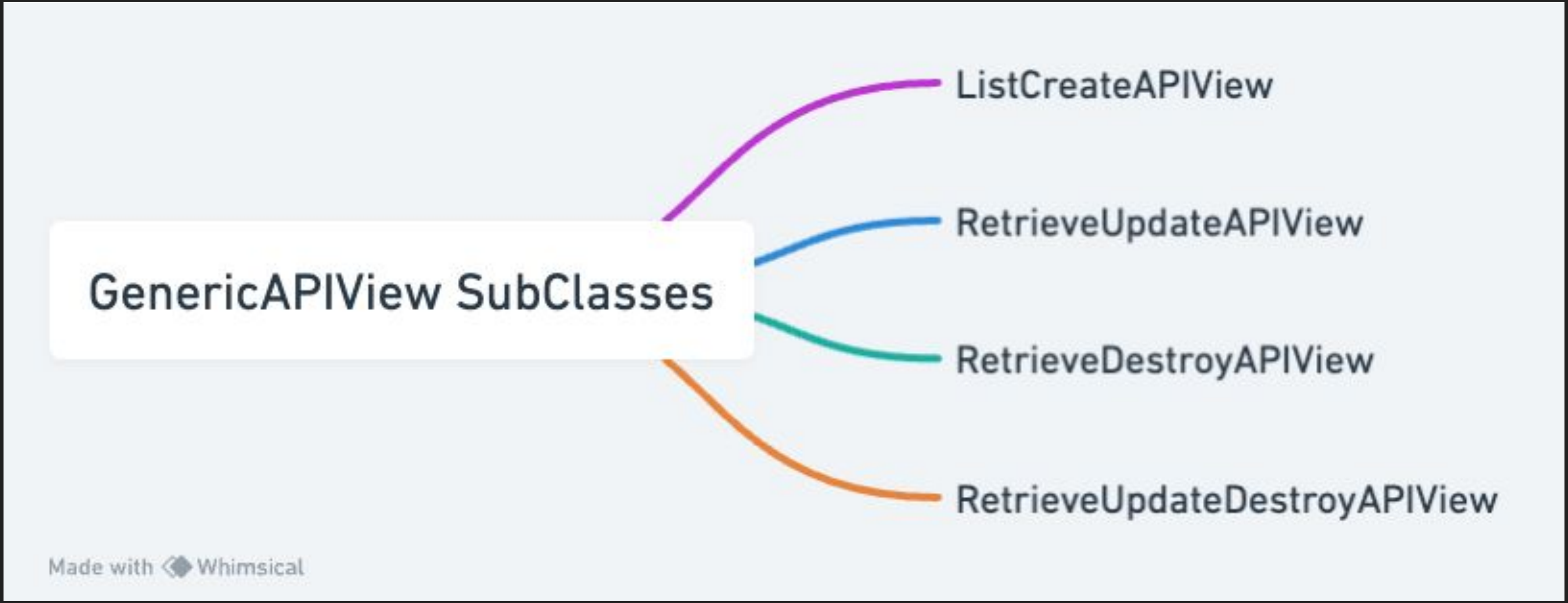
Django docs.

DRF: Classe Base - GenericAPIView

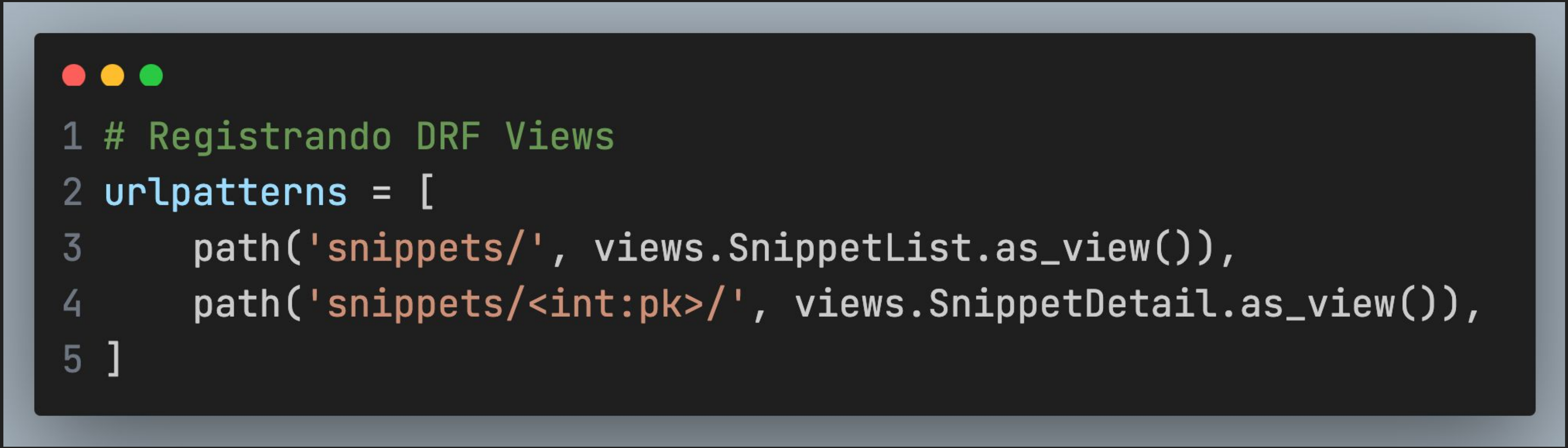




DRF: Classe Base - GenericAPIView

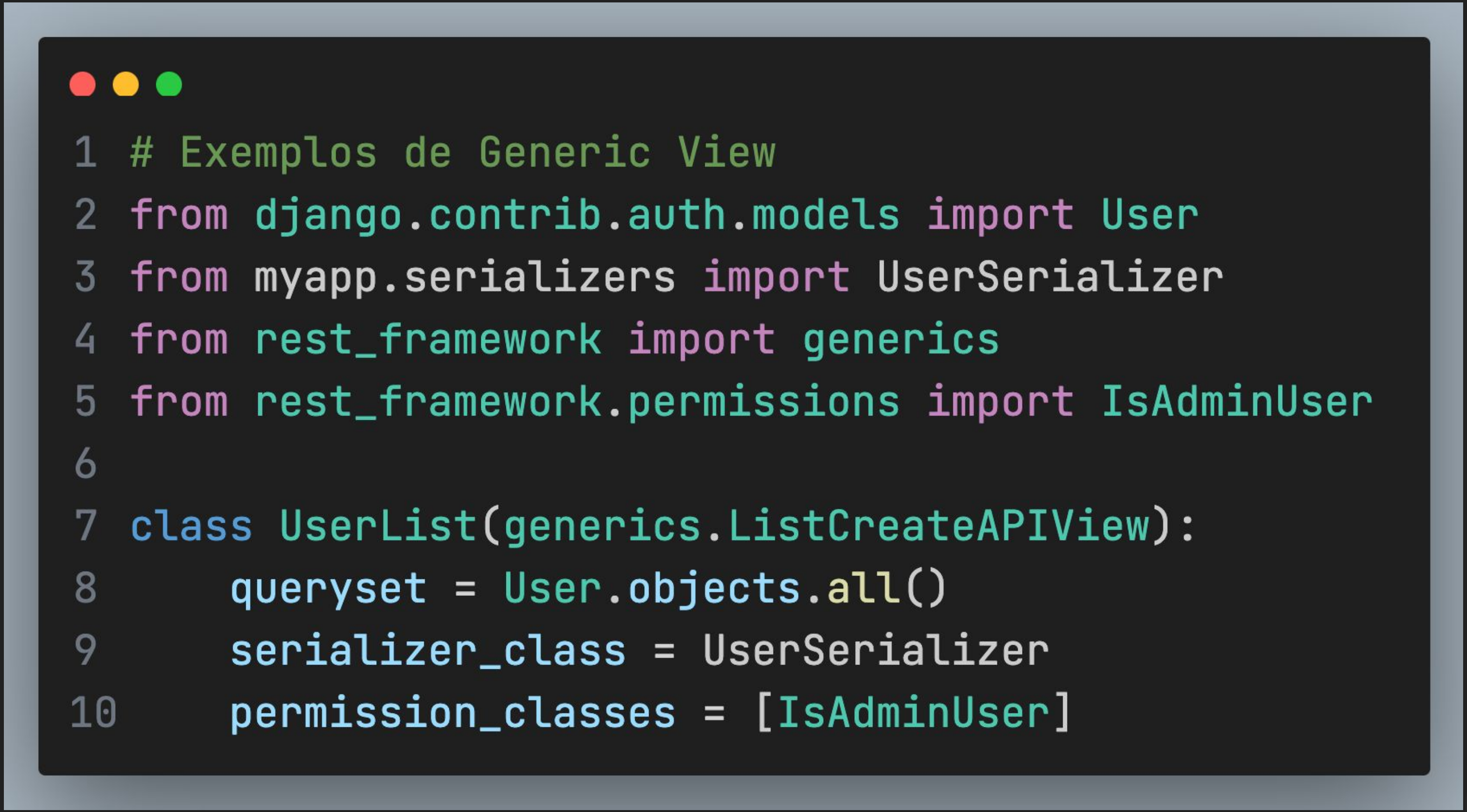


DRF: Registrando Rotas para Views



```python
# Registrando DRF Views
urlpatterns = [
    path('snippets/', views.SnippetList.as_view()),
    path('snippets/<int:pk>/', views.SnippetDetail.as_view()),
]
```

DRF: Exemplos de Views

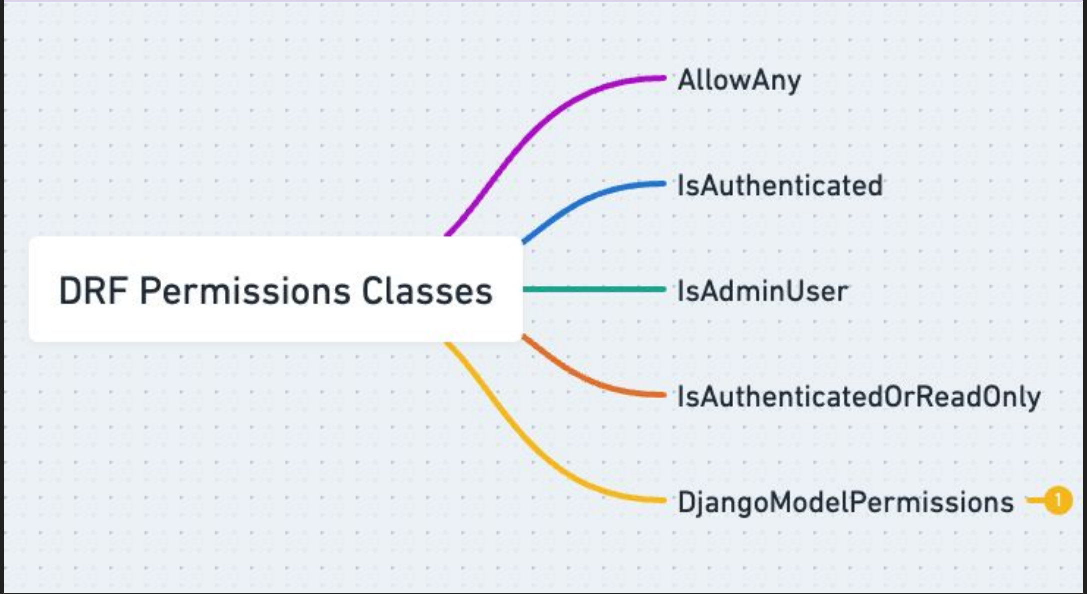


```python
# Exemplos de Generic View
from django.contrib.auth.models import User
from myapp.serializers import UserSerializer
from rest_framework import generics
from rest_framework.permissions import IsAdminUser

class UserList(generics.ListCreateAPIView):
    queryset = User.objects.all()
    serializer_class = UserSerializer
    permission_classes = [IsAdminUser]
```

DRF - Controle de Acesso (Permissões)

- Integrado ao Sistema de Auth
- Baseado em Classes de Permissão
- Controlam o Acesso em Alto Nível
- Filtro/Middleware que é executado antes de completar a request

```python
from rest_framework import permissions
```

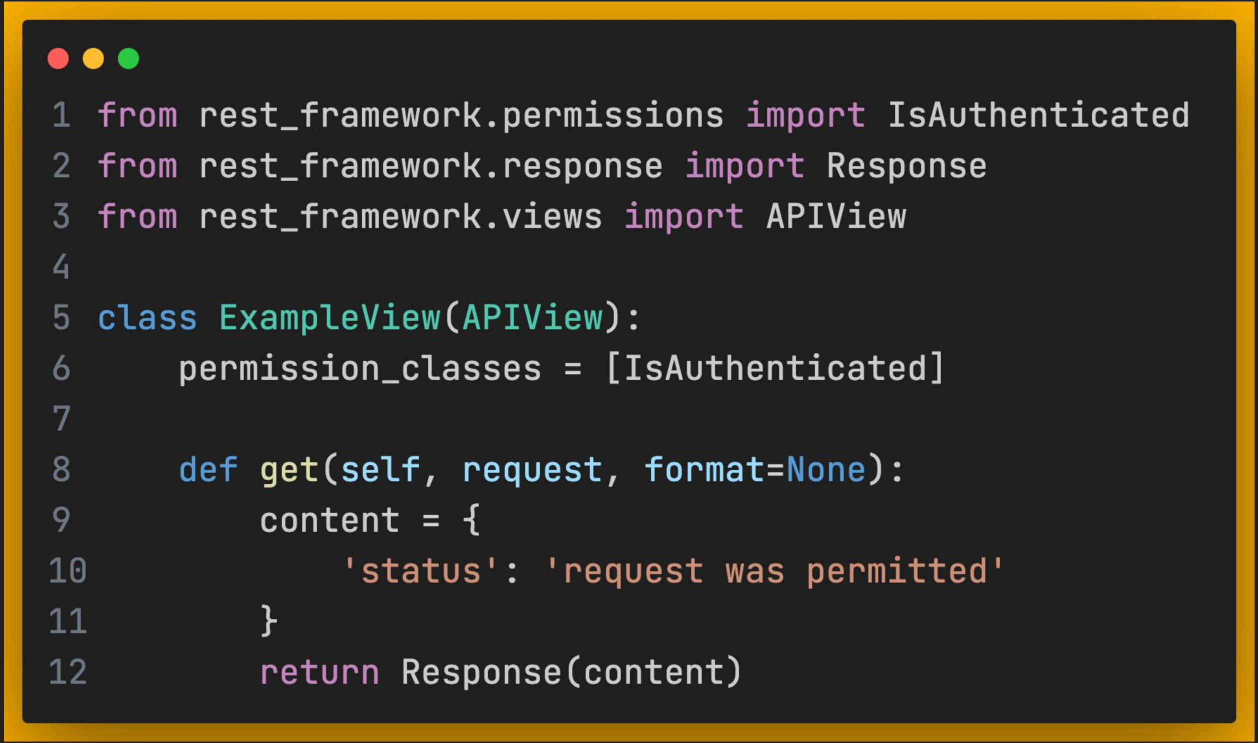


Before running the main body of the view each permission in the list is checked. If any permission check fails, an `exceptions.PermissionDenied`or `exceptions.NotAuthenticated`exception will be raised, and the main body of the view will not run.

DRF Docs

django REST framework

DRF - Controle de Acesso (Permissões)

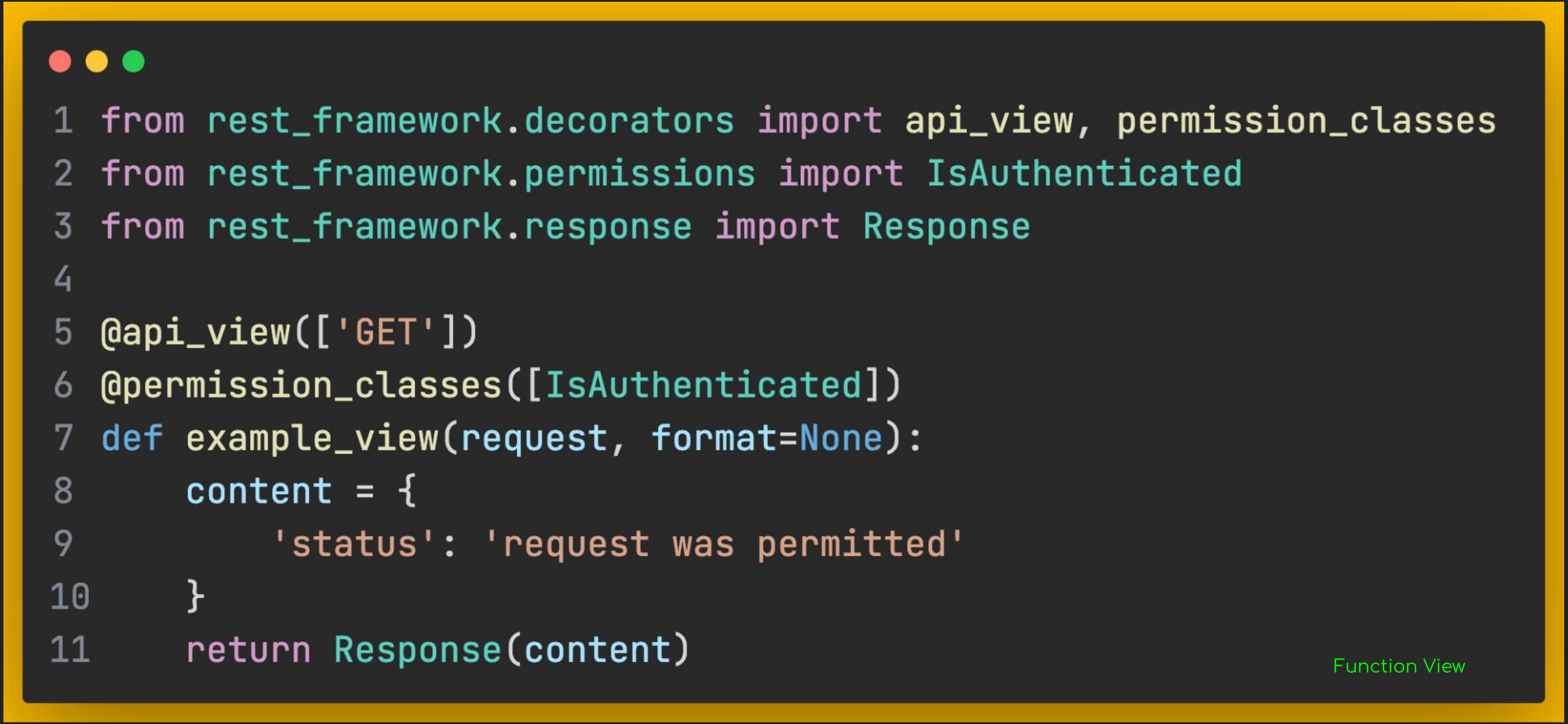


```python
from rest_framework.permissions import IsAuthenticated
from rest_framework.response import Response
from rest_framework.views import APIView

class ExampleView(APIView):
    permission_classes = [IsAuthenticated]

    def get(self, request, format=None):
        content = {
            'status': 'request was permitted'
        }
        return Response(content)
```

Before running the main body of the view each permission in the list is checked. If any permission check fails, an `exceptions.PermissionDenied` or `exceptions.NotAuthenticated` exception will be raised, and the main body of the view will not run.

DRF Docs

DRF - Controle de Acesso (Permissões)



```python
from rest_framework.decorators import api_view, permission_classes
from rest_framework.permissions import IsAuthenticated
from rest_framework.response import Response

@api_view(['GET'])
@permission_classes([IsAuthenticated])
def example_view(request, format=None):
    content = {
        'status': 'request was permitted'
    }
    return Response(content)
```

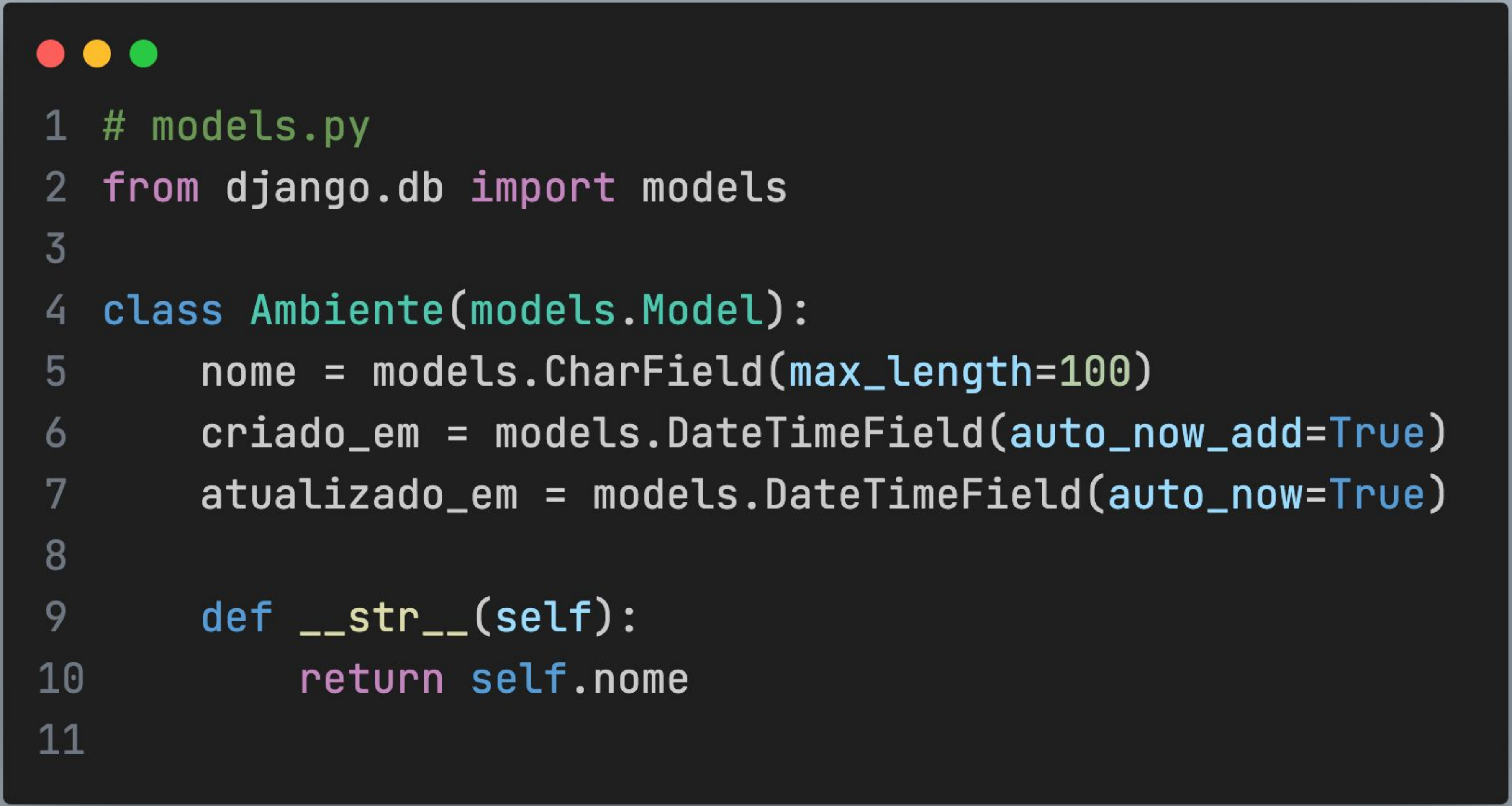
Viewsets

DRF Viewset Methods

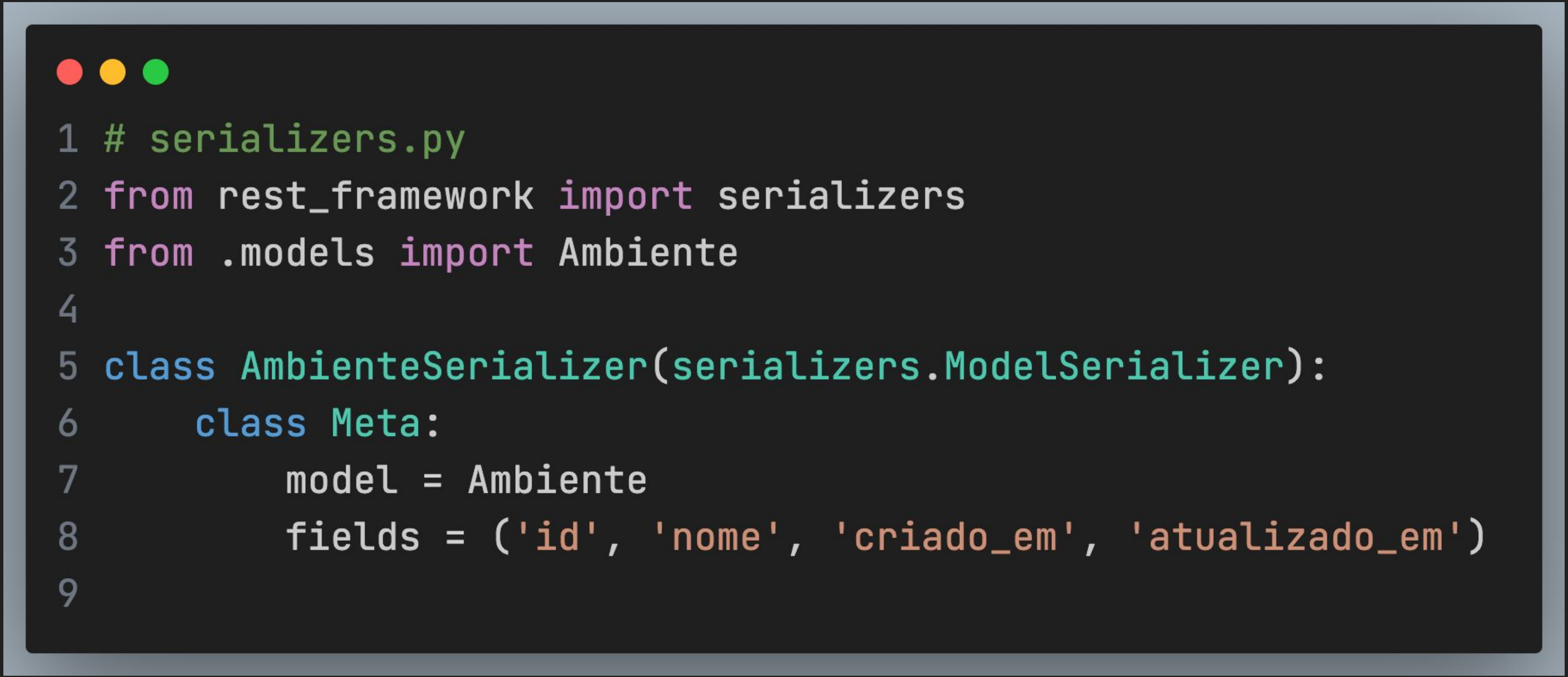
- **list()**: Used to handle HTTP GET requests and return a list of objects.
- **create()**: Handles HTTP POST requests to create a new object.
- **retrieve()**: Handles HTTP GET requests for individual objects by their primary key.
- **update()**: Handles HTTP PUT requests to update an existing object by its primary key.
- **partial_update()**: Handles HTTP PATCH requests to partially update an existing object by its primary key.
- **destroy()**: Handles HTTP DELETE requests to delete an existing object by its primary key.

Exemplo CRUD API REST completo para Ambiente via ViewSet

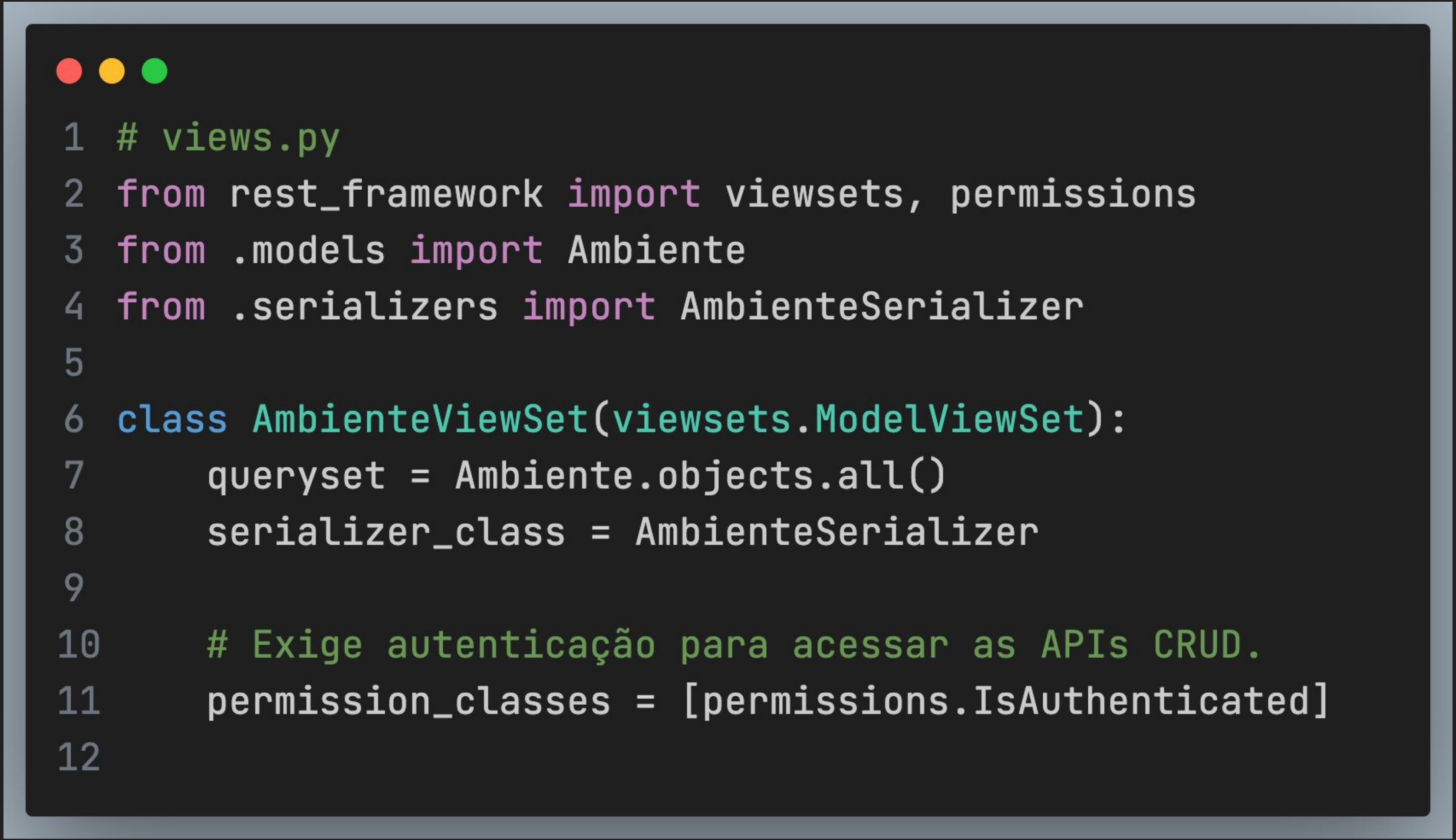
Me feat.



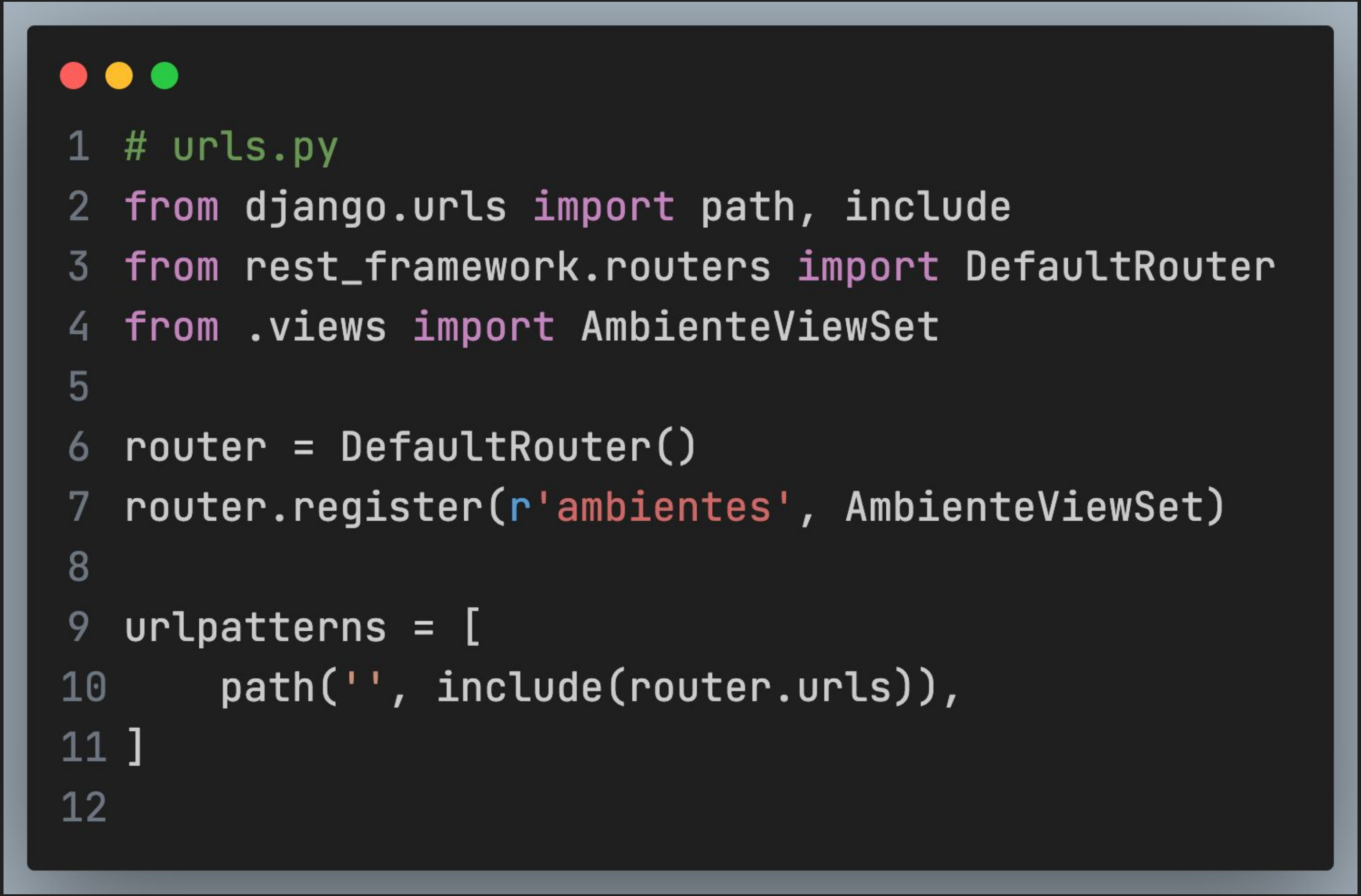
```python
# models.py
from django.db import models

class Ambiente(models.Model):
    nome = models.CharField(max_length=100)
    criado_em = models.DateTimeField(auto_now_add=True)
    atualizado_em = models.DateTimeField(auto_now=True)

    def __str__(self):
        return self.nome

```



```python
# serializers.py
from rest_framework import serializers
from .models import Ambiente

class AmbienteSerializer(serializers.ModelSerializer):
    class Meta:
        model = Ambiente
        fields = ('id', 'nome', 'criado_em', 'atualizado_em')

```

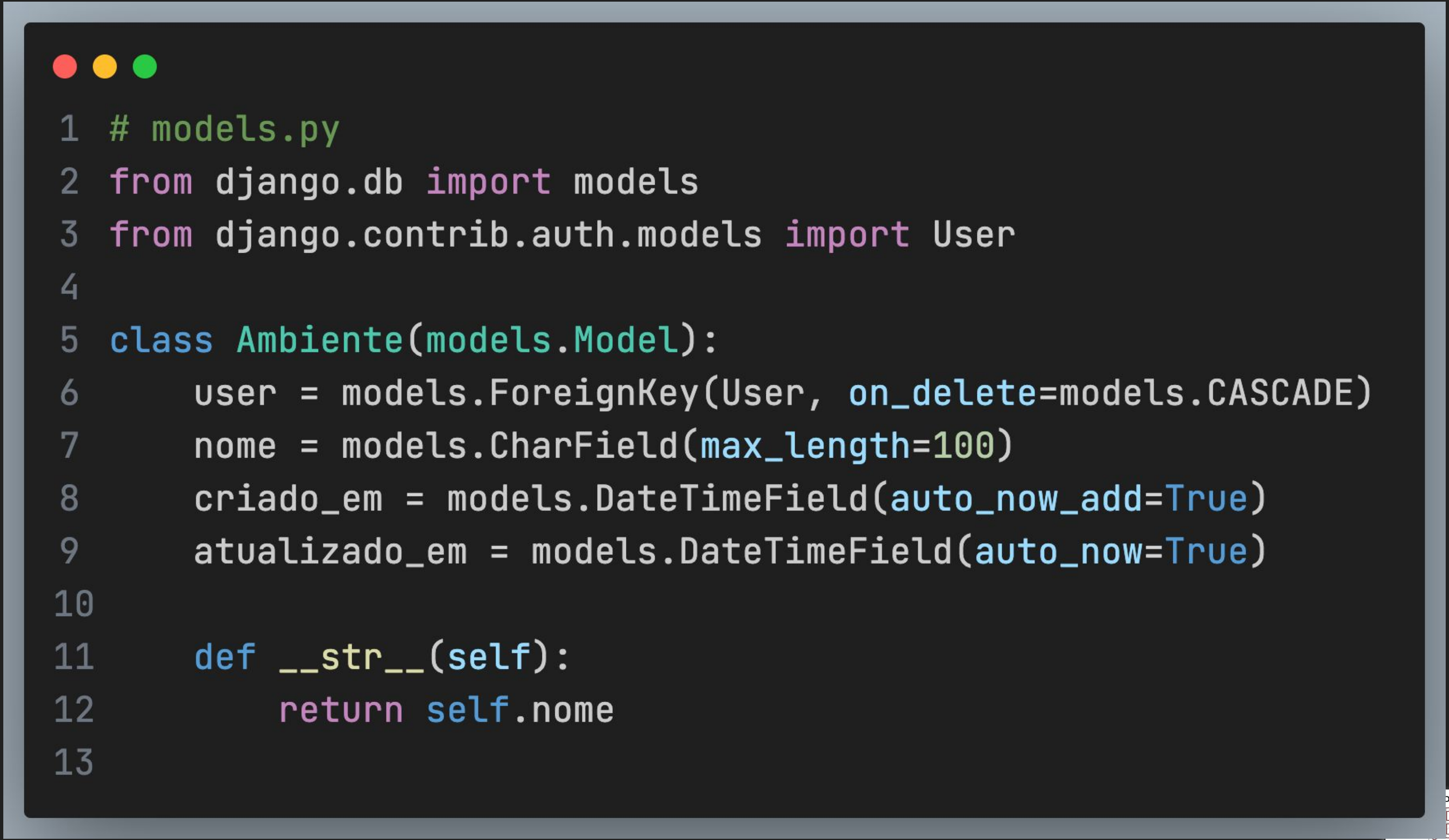


```python
# views.py
from rest_framework import viewsets, permissions
from .models import Ambiente
from .serializers import AmbienteSerializer

class AmbienteViewSet(viewsets.ModelViewSet):
    queryset = Ambiente.objects.all()
    serializer_class = AmbienteSerializer

    # Exige autenticação para acessar as APIs CRUD.
    permission_classes = [permissions.IsAuthenticated]

```

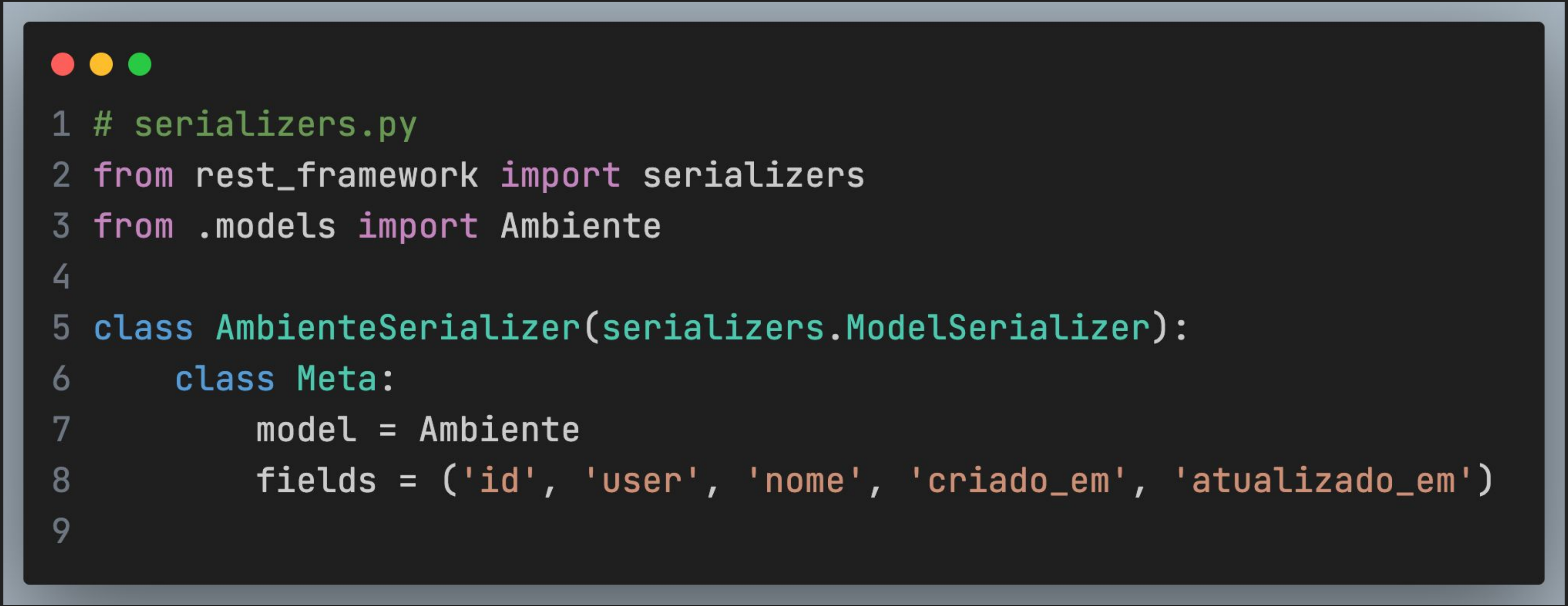


```python
# urls.py
from django.urls import path, include
from rest_framework.routers import DefaultRouter
from .views import AmbienteViewSet

router = DefaultRouter()
router.register(r'ambientes', AmbienteViewSet)

urlpatterns = [
    path('', include(router.urls)),
]

```

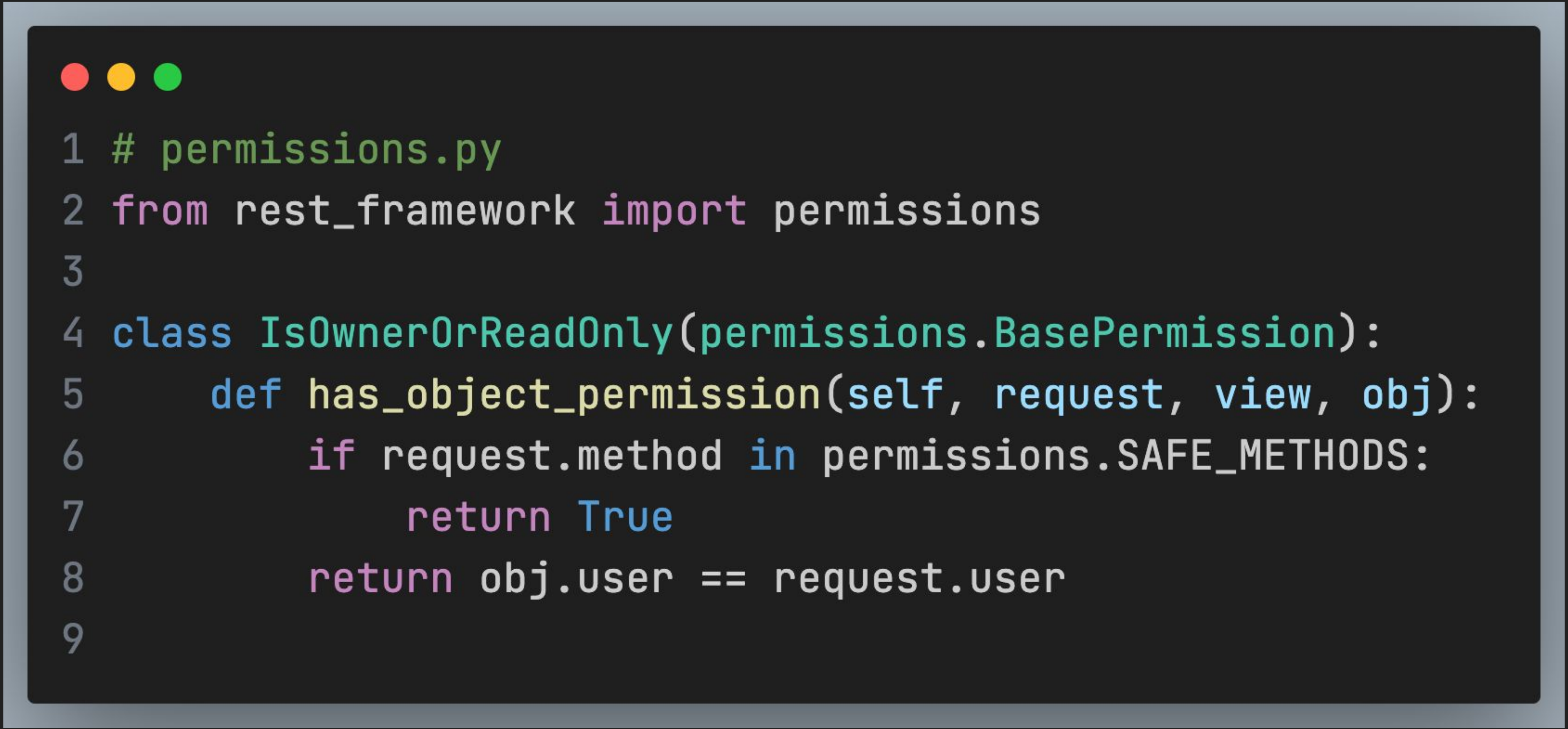
Alterar para incluir Restrições de Permissão…



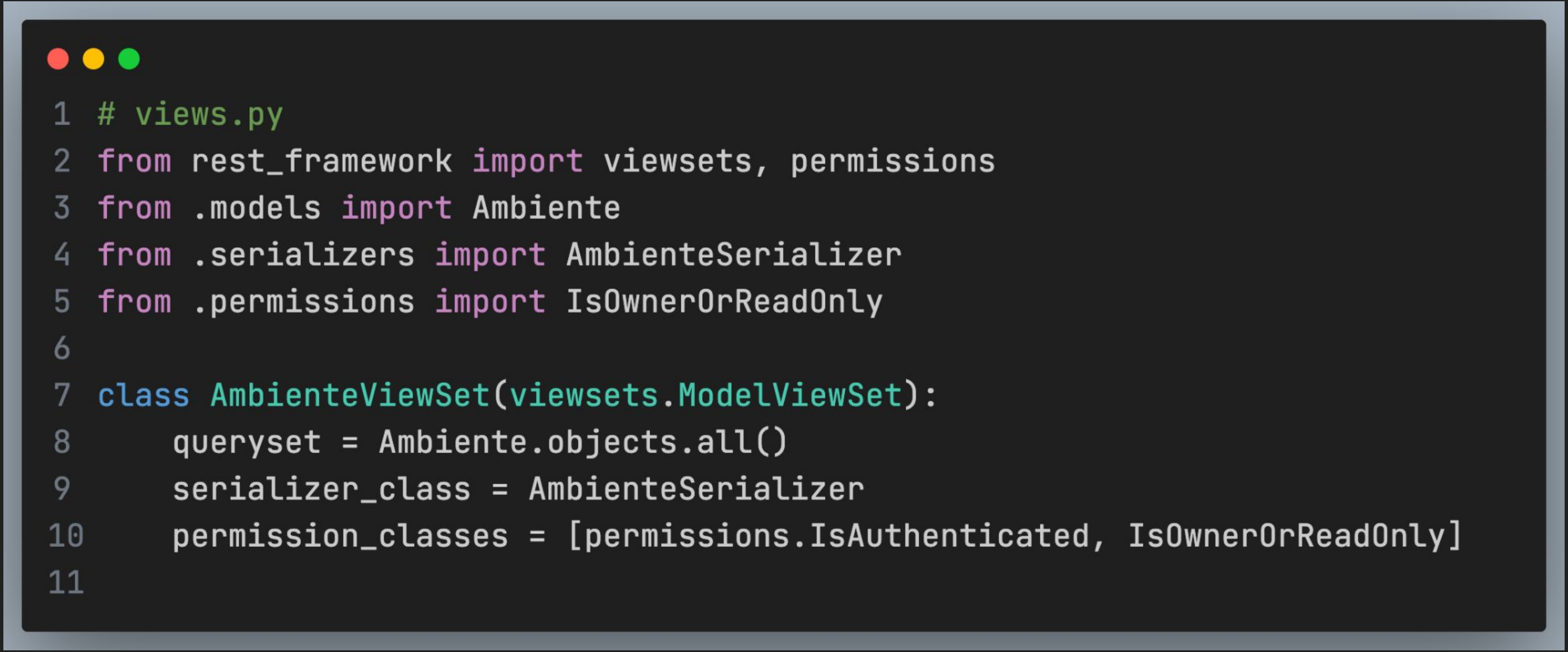
```python
# models.py
from django.db import models
from django.contrib.auth.models import User

class Ambiente(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    nome = models.CharField(max_length=100)
    criado_em = models.DateTimeField(auto_now_add=True)
    atualizado_em = models.DateTimeField(auto_now=True)

    def __str__(self):
        return self.nome

```



```python
# serializers.py
from rest_framework import serializers
from .models import Ambiente

class AmbienteSerializer(serializers.ModelSerializer):
    class Meta:
        model = Ambiente
        fields = ('id', 'user', 'nome', 'criado_em', 'atualizado_em')

```

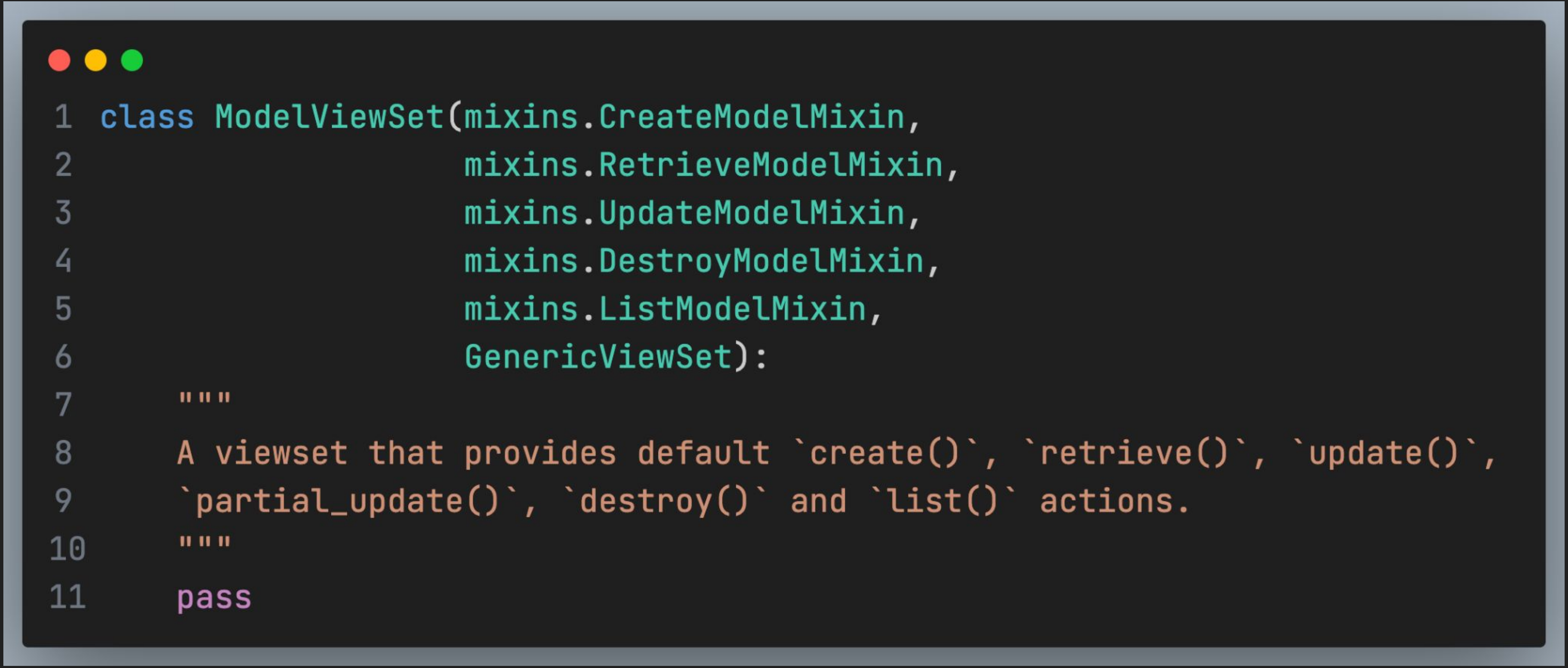


```python
# permissions.py
from rest_framework import permissions

class IsOwnerOrReadOnly(permissions.BasePermission):
    def has_object_permission(self, request, view, obj):
        if request.method in permissions.SAFE_METHODS:
            return True
        return obj.user == request.user

```

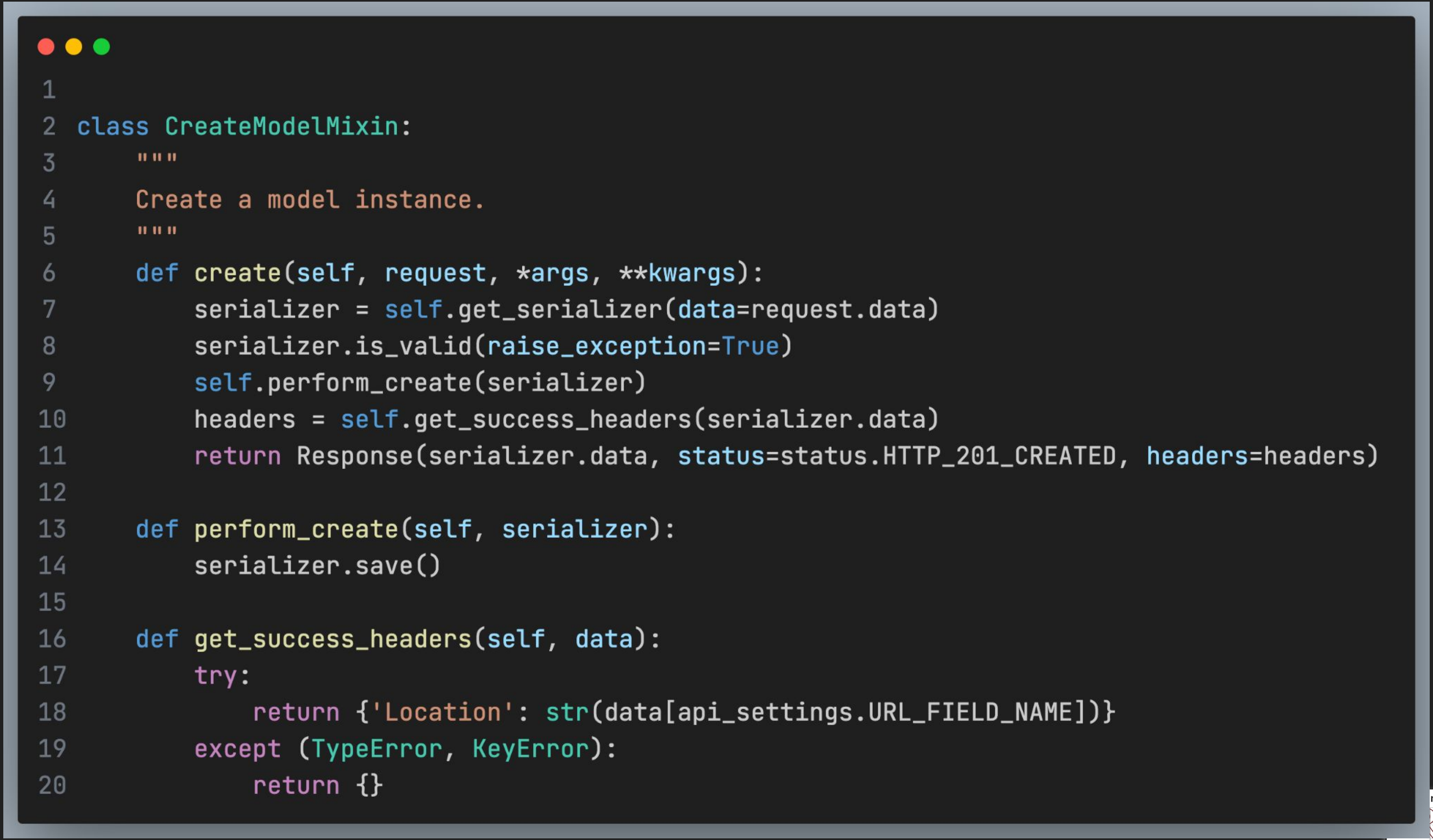


```python
# views.py
from rest_framework import viewsets, permissions
from .models import Ambiente
from .serializers import AmbienteSerializer
from .permissions import IsOwnerOrReadOnly

class AmbienteViewSet(viewsets.ModelViewSet):
    queryset = Ambiente.objects.all()
    serializer_class = AmbienteSerializer
    permission_classes = [permissions.IsAuthenticated, IsOwnerOrReadOnly]

```

Como "ModelViewSet" Funciona?



```python
class ModelViewSet(mixins.CreateModelMixin,
                   mixins.RetrieveModelMixin,
                   mixins.UpdateModelMixin,
                   mixins.DestroyModelMixin,
                   mixins.ListModelMixin,
                   GenericViewSet):
    """
    A viewset that provides default `create()`, `retrieve()`, `update()`,
    `partial_update()`, `destroy()` and `list()` actions.
    """
    pass
```



```python

class CreateModelMixin:
    """
    Create a model instance.
    """
    def create(self, request, *args, **kwargs):
        serializer = self.get_serializer(data=request.data)
        serializer.is_valid(raise_exception=True)
        self.perform_create(serializer)
        headers = self.get_success_headers(serializer.data)
        return Response(serializer.data, status=status.HTTP_201_CREATED, headers=headers)

    def perform_create(self, serializer):
        serializer.save()

    def get_success_headers(self, data):
        try:
            return {'Location': str(data[api_settings.URL_FIELD_NAME])}
        except (TypeError, KeyError):
            return {}
```