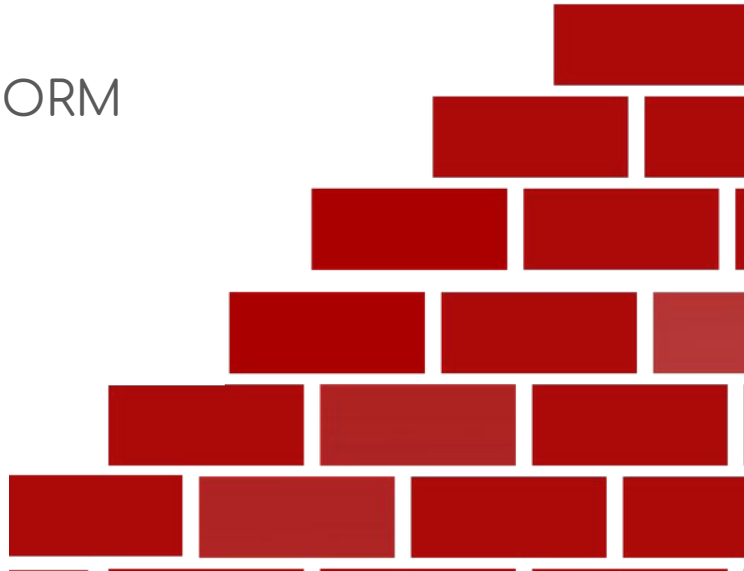
Tijolinho por tijolinho: construindo APIs com Django

Oi!

- Co-organizadora do PyLadies Fortaleza <3
- Desenvolvedora na Conceptu
    - Backend no ClubeWatt
- Concluinte em Engenharia de Computação - IFCE
- Técnica em Informática - IFRN

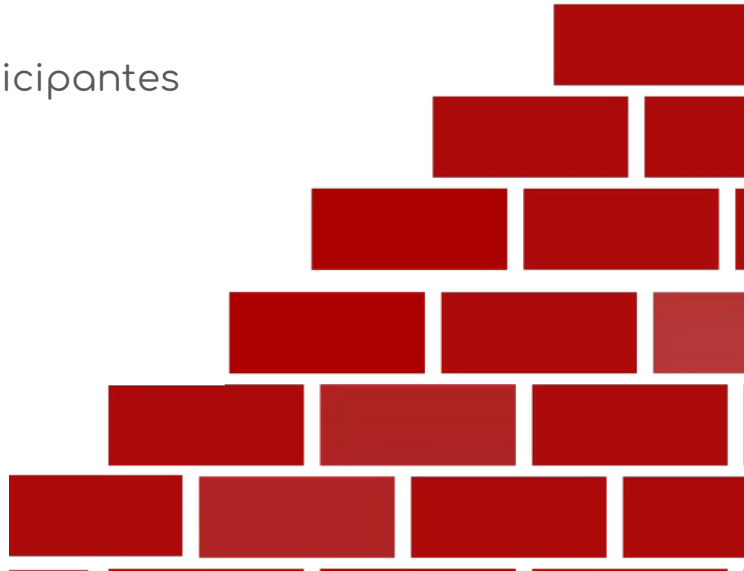
Django Rest Framework - DRF

- Framework Django para construções de APIs Web
- Desenvolvido em 2011
- Serialização suporta dados ORM e não ORM
- Arquitetura MVT



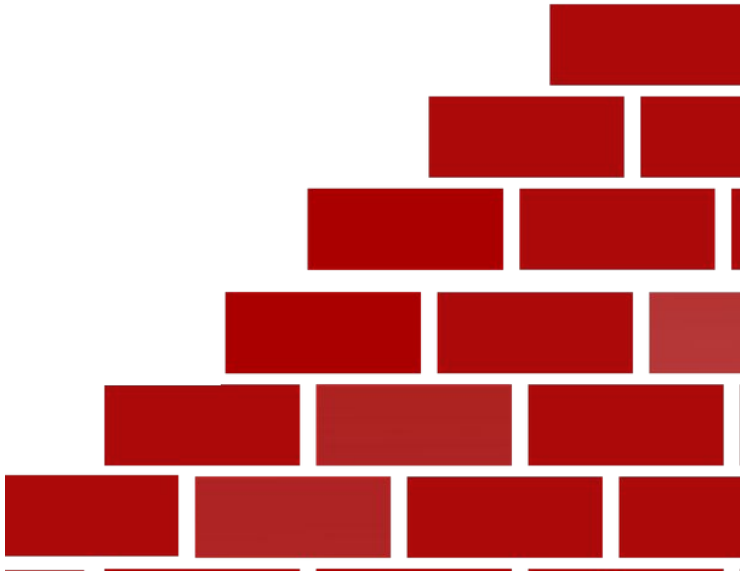
O que iremos fazer...

- Desenvolver uma API REST para gerenciar os participantes da PyBR e poderá:
    - Cadastrar novos participantes
    - Listar dados gerais e individuais dos participantes
    - Atualizar dados
    - Apagar dados
    - Restringir uso com autenticação



Iniciando um projeto

```
$ django-admin startproject nome_do_projeto
$ python manage.py startapp nome_do_app
$ python manage.py migrate
$ python manage.py createsuperuser
```



event > models.py > ...

```python
from django.db import models


class Participant(models.Model):
    first_name = models.CharField(max_length=50)
    last_name = models.CharField(max_length=50)
    email = models.CharField(max_length=50)
    city = models.CharField(max_length=50)

    def __str__(self):
        return f"{self.first_name} {self.last.name}"

```

$ python manage.py makemigrations

$ python manage.py migrate

event > serializers.py > ...

```python
from rest_framework import serializers

from event.models import Participant


class ParticipantSerializer(serializers.ModelSerializer):
    class Meta:
        model = Participant
        fields = '__all__'

```

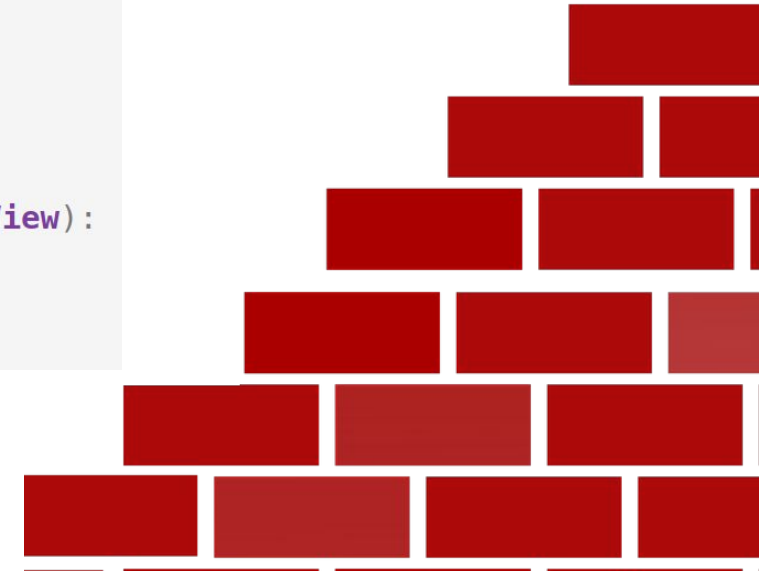
Também é possível criar serializers usando serializers.Serializer, definindo cada field que será utilizado

event > views.py > ...

```python
from django.shortcuts import render

from event.models import Participant
from event.serializers import ParticipantSerializer
from rest_framework.generics import (
    ListCreateAPIView, RetrieveUpdateDestroyAPIView
)


class ParticipantList(ListCreateAPIView):
    queryset = Participant.objects.all()
    serializer_class = ParticipantSerializer


class ParticipantDetail(RetrieveUpdateDestroyAPIView):
    queryset = Participant.objects.all()
    serializer_class = ParticipantSerializer

```

Outras opções:

- APIView
- GenericAPIView + Mixins



Classy Django REST Framework (http://www.cdrf.co/)

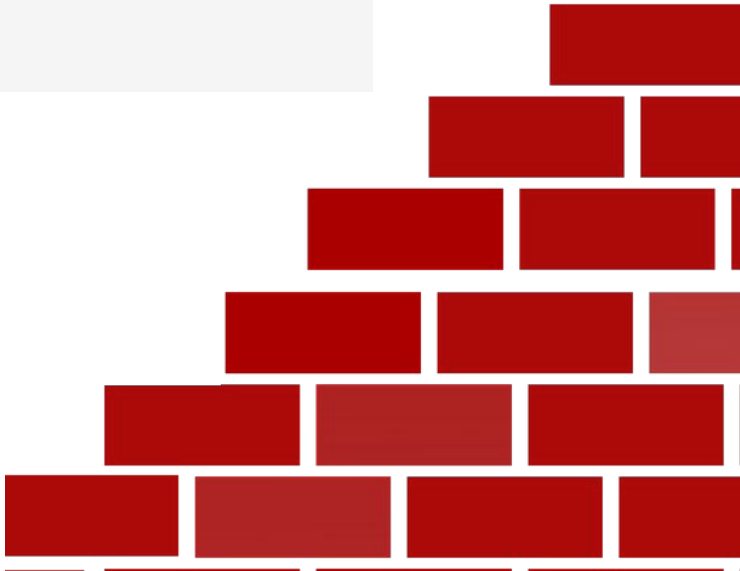
event > urls.py > ...

```python
from django.urls import path
from rest_framework.urlpatterns import format_suffix_patterns

from event.views import ParticipantDetail, ParticipantList


urlpatterns = [
    path('participants/', ParticipantList.as_view()),
    path('participants/<int:pk>/', ParticipantDetail.as_view()),
]

urlpatterns = format_suffix_patterns(urlpatterns)

```

Definindo a permissão

pybr15 > settings.py > ...

```
125  REST_FRAMEWORK = {
126      'DEFAULT_PERMISSION_CLASSES': [
127          'rest_framework.permissions.IsAuthenticated',
128      ]
129  }
```

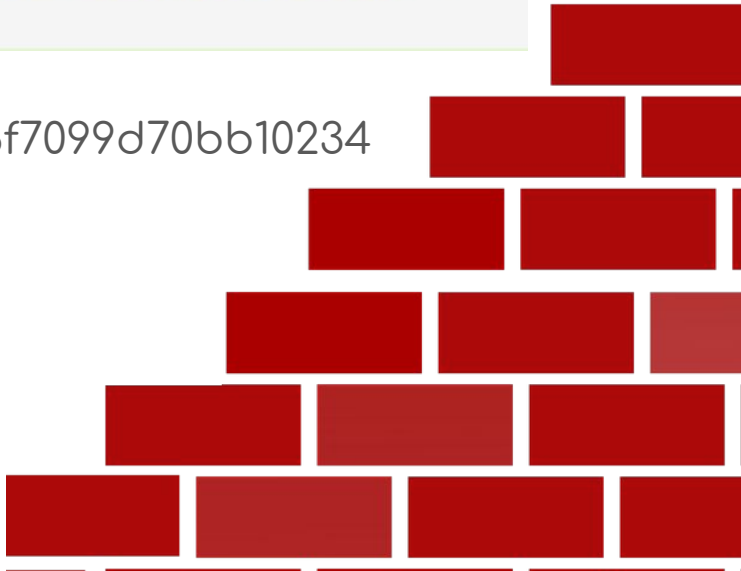


Definindo a autenticação

pybr15 > settings.py > ...

```
127  REST_FRAMEWORK = {
128      'DEFAULT_AUTHENTICATION_CLASSES': [
129          'rest_framework.authentication.TokenAuthentication',
130      ],
```

Authorization: Token b853436ab42ff013658f1fe56f7099d70bb10234



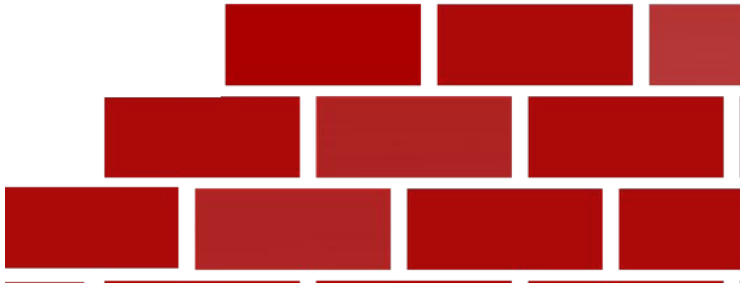
Definindo a autenticação

pybr15 > settings.py > ...

```
33  INSTALLED_APPS = [
34      'django.contrib.admin',
35      'django.contrib.auth',
36      'django.contrib.contenttypes',
37      'django.contrib.sessions',
38      'django.contrib.messages',
39      'django.contrib.staticfiles',
40
41      'rest_framework',
42      'rest_framework.authtoken',
43
44      'event.apps.EventConfig',
45  ]
```

pybr15 > urls.py > ...

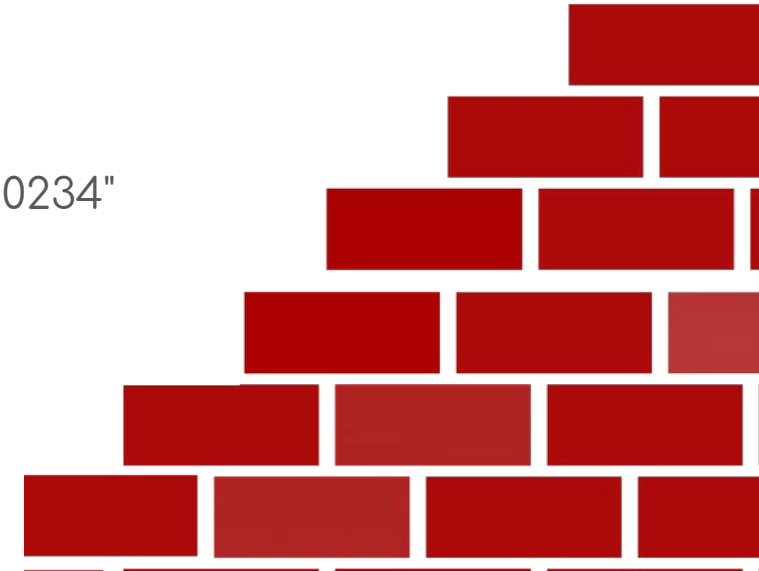
```
1   from django.contrib import admin
2   from django.urls import path, include
3   from rest_framework.authtoken import views
4
5
6   urlpatterns = [
7       path('admin/', admin.site.urls),
8       path('api-token-auth/', views.obtain_auth_token),
9       path('', include('event.urls')),
10  ]
11
```



Gerando token

```
$ curl -X POST http://localhost:8000/api-token-auth/ -H 'Content-Type:
application/json'  -d '{"username": "username", "password": "pass"}'
```

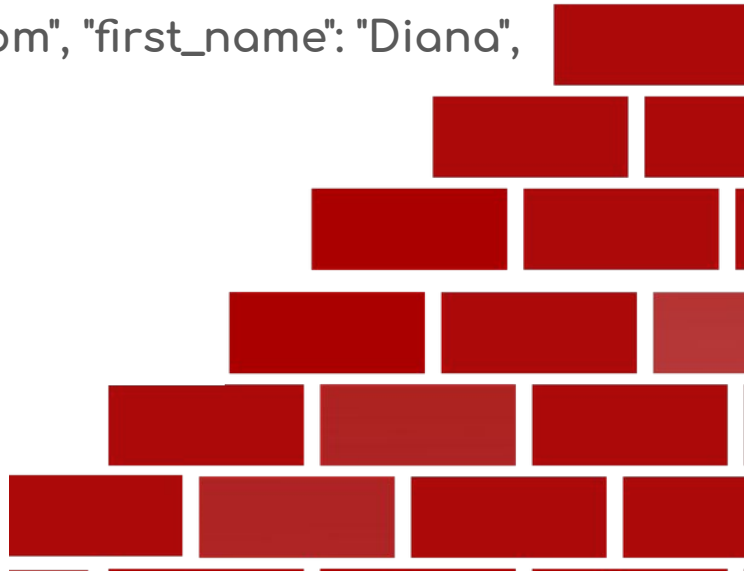
```
{
    "token": "b853436ab42ff013658f1fe56f7099d70bb10234"
}
```



Consumindo a API

```
$ curl -X POST http://localhost:8000/participants/ -H 'Authorization: Token
b853436ab42ff013658f1fe56f7099d70bb10234' -H 'Content-Type: application/json'
-d '{"city": "Themyscira", "email": "wonder@woman.com", "first_name": "Diana",
"last_name": "Prince"}'
```

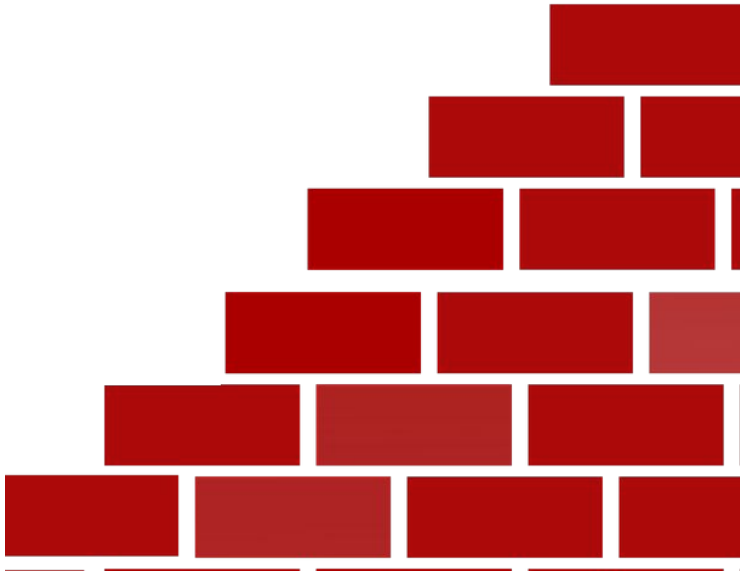
Inserindo novo participante



Consumindo a API

$ curl -X GET http://localhost:8000/participants/1/ -H 'Authorization: Token b853436ab42ff013658f1fe56f7099d70bb10234'

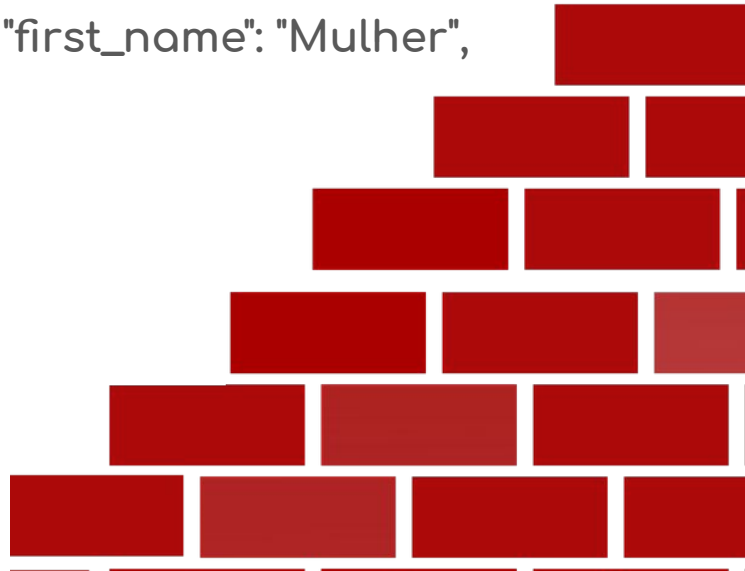
Listando todos os participantes



Consumindo a API

```
curl -X PUT http://localhost:8000/participants/6/ -H 'Authorization: Token
b853436ab42ff013658f1fe56f7099d70bb10234' -H 'Content-Type: application/json'
-d '{"city": "Gotham", "email": "wonder@woman.com", "first_name": "Mulher",
"last_name": "Maravilha"}'
```

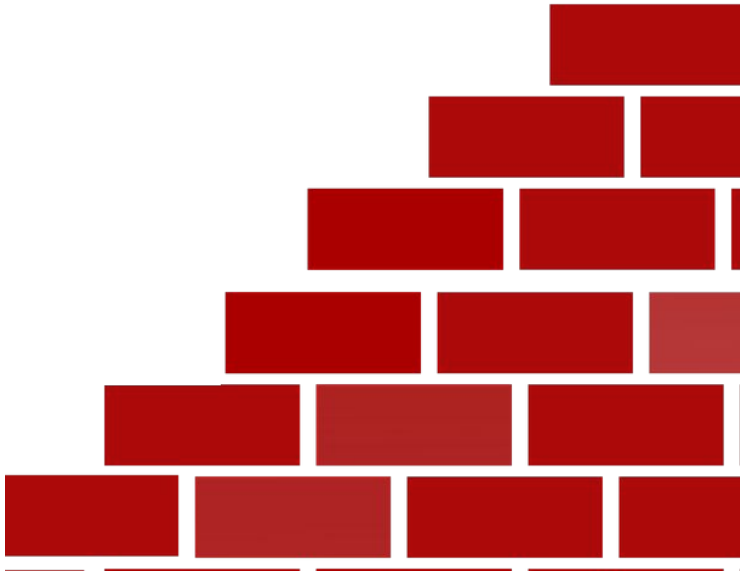
Atualizando dados do participante



Consumindo a API

curl -X DELETE http://localhost:8000/participants/6/ -H 'Authorization: Token b853436ab42ff013658f1fe56f7099d70bb10234'

Excluindo participante



Dicas importantes

- Use o dicionário REST_FRAMEWORK no settings.py para inserir todas as configurações gerais do DRF
- Escolha a forma de desenvolver as views de acordo com sua necessidade
- Você também pode acessar a API via browser
- Organize bem o seu código!

OBRIGADA!

github.com/ryllari

ryllari

linkedin.com/in/ryllari

fb.com/pyladiesfortaleza
@pyladiesfort
@pyladiesfortaleza