

ORM Y FORMULARIOS

---

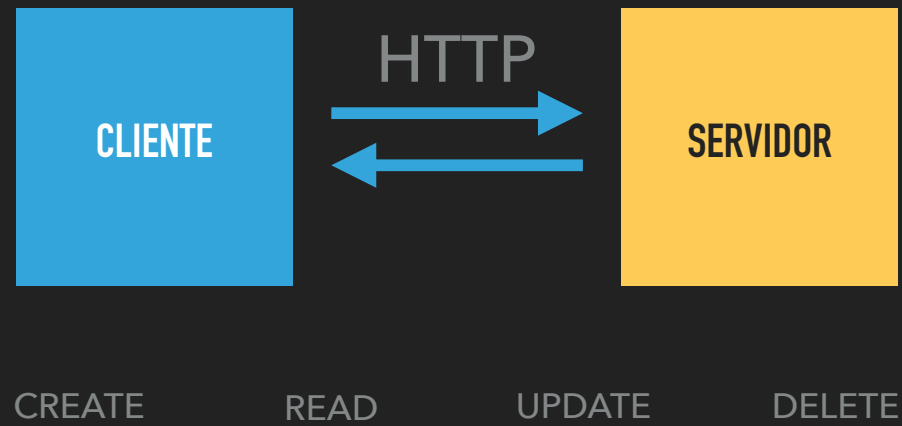
**DJANGO**

DJANGO REST FRAMEWORK

---

**API REST**

# API REST



## PRINCIPIOS REST: INTERFAZ UNIFORME Y DE FÁCIL ACCESO

- ▶ HTTP METHODS (VERBOS)

POST

GET

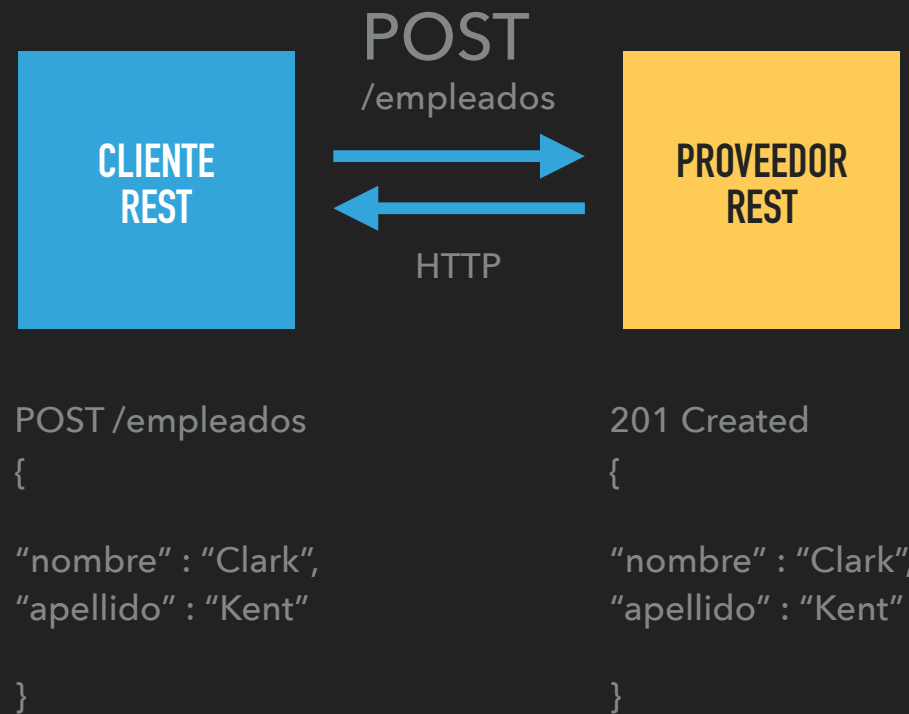
PUT

DELETE

- ▶ URI (SUSTANTIVOS)

url://empleados/

# CREATE



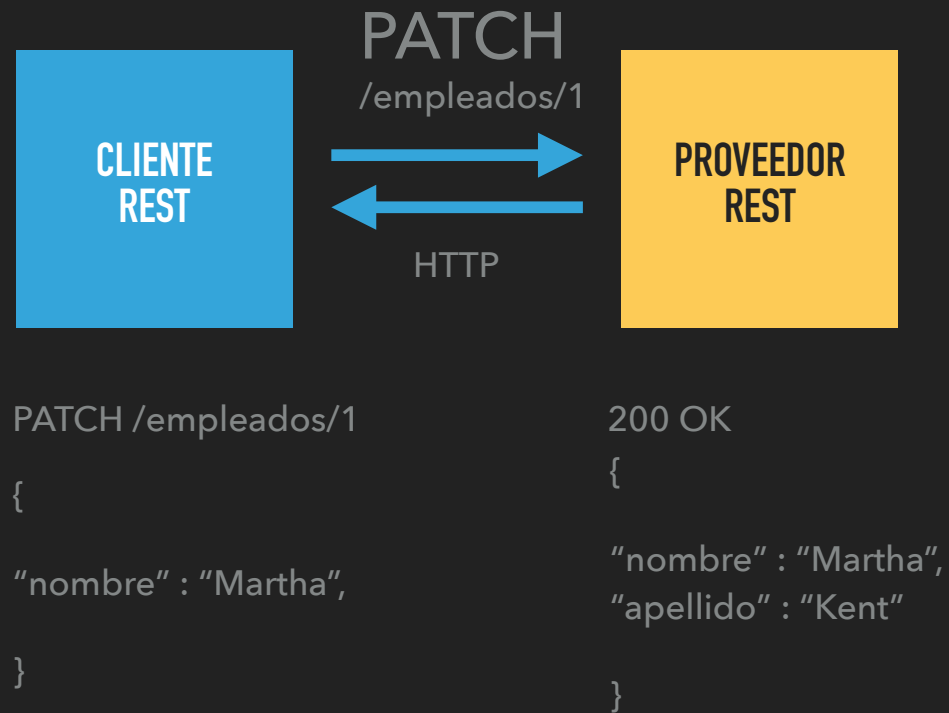
# READ



## UPDATE

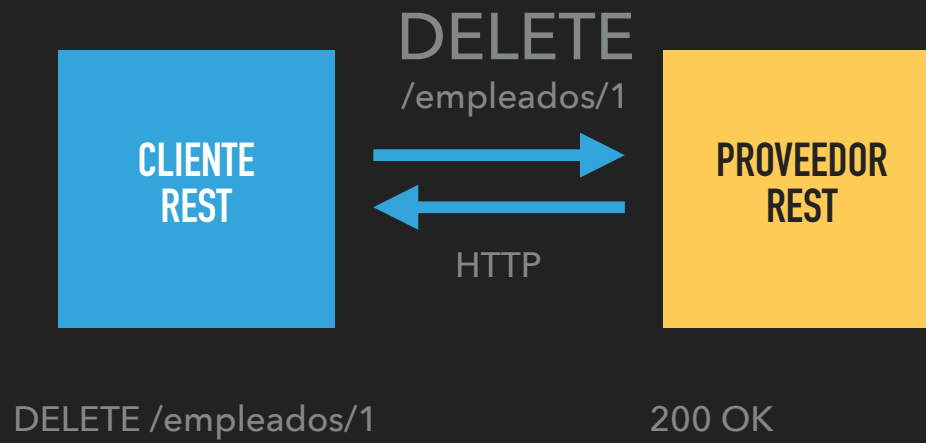


# PATCH





# DELETE



## PRINCIPIOS REST: MULTIPLES FORMATOS

- ▶ TEXT/XML

```
<empleado>
```

```
  <nombre>Clark</nombre>
```

```
</empleado>
```

- ▶ TEXT/PLAIN

```
nombre = Clark
```

- ▶ APPLICATION/JSON

```
empleado:{nombre: Clark}
```

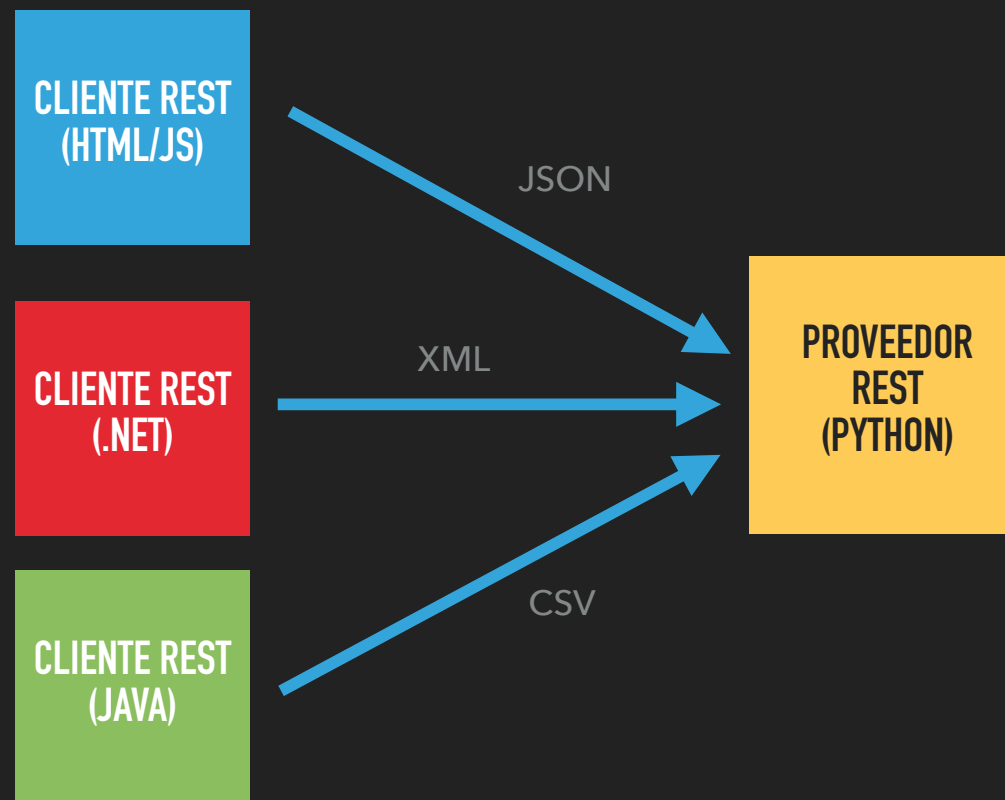
## VENTAJAS DE USAR REST

- ▶ CREATE
- ▶ READ
- ▶ UPDATE
- ▶ DELETE
- ▶ POST
- ▶ GET
- ▶ PUT
- ▶ DELETE

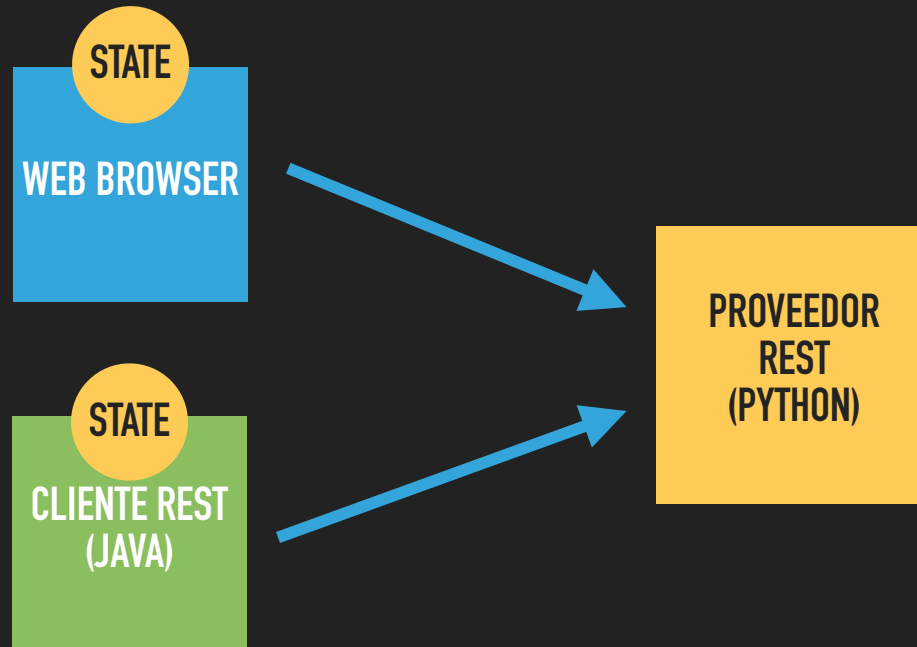
## FACIL ACCESO

- ▶ <http://www.hospital.com/pacientes>
- ▶ <http://www.hospital.com/pacientes/{id}>
- ▶ <http://www.hospital.com/prescripciones>
- ▶ <http://www.hospital.com/prescripciones/{id}>

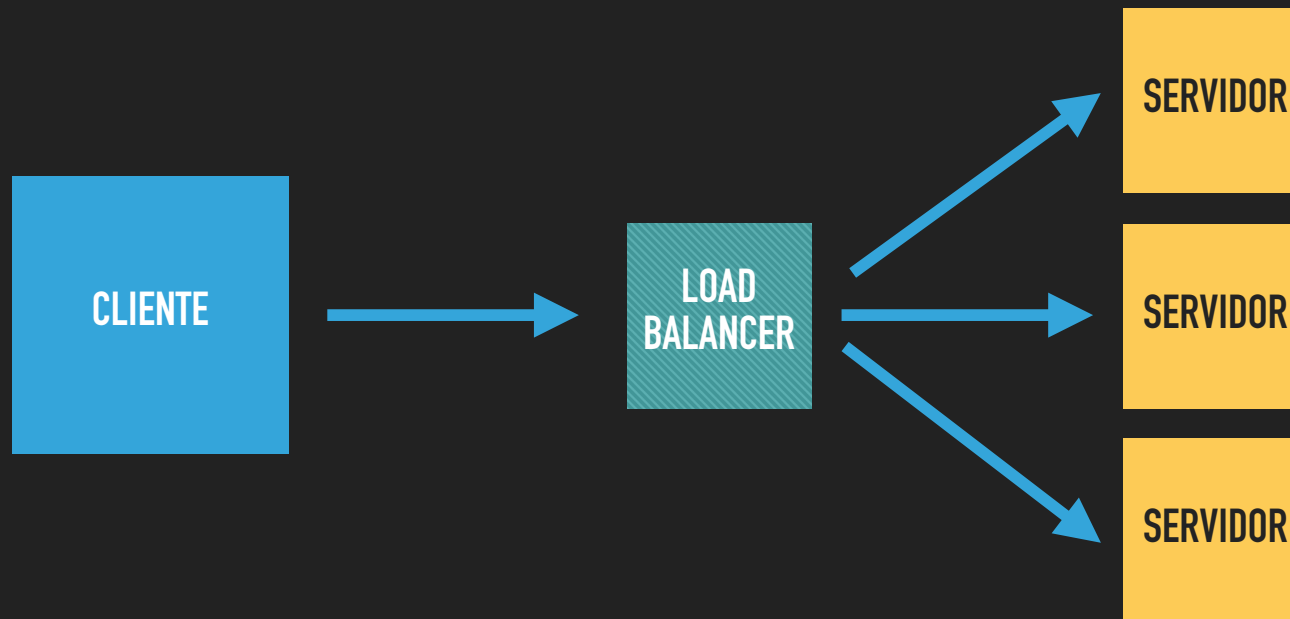
## INTEROPERABILIDAD



# STATELESS



# ESCALABILIDAD



DJANGO REST FRAMEWORK

---

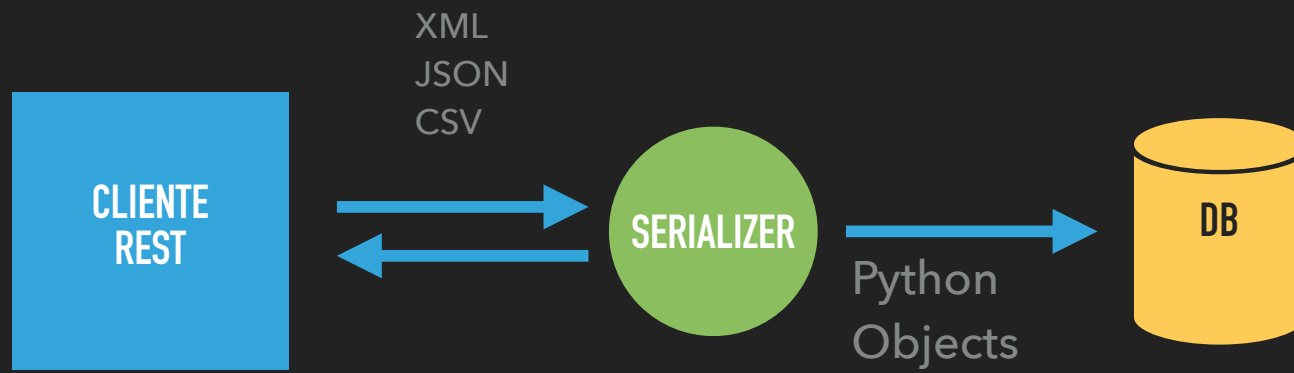
**DJANGO REST FRAMEWORK**



## TIPOS DE VISTAS

- ▶ Class Based Views (rapidas, out of the box)
- ▶ Function based Views (personalizables)

## SERIALIZERS

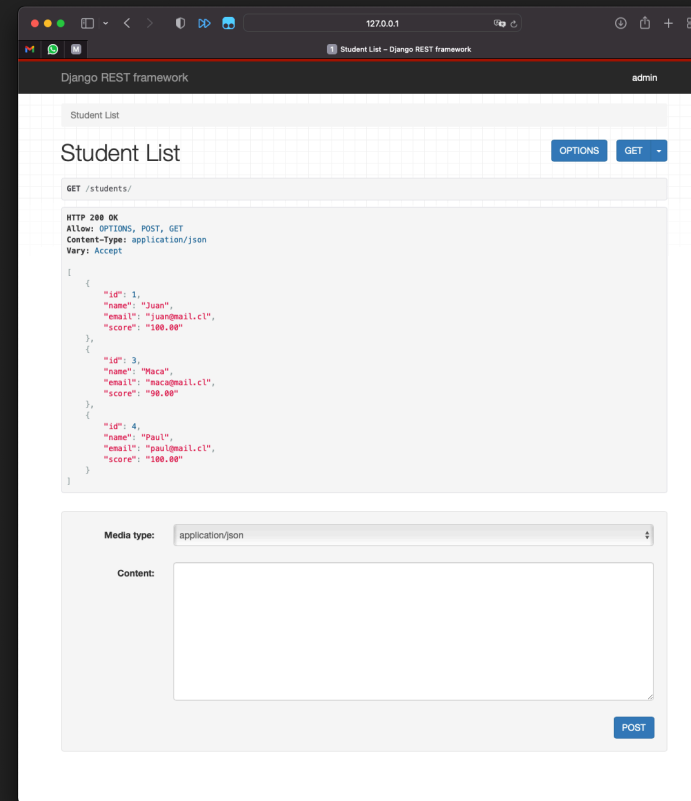


## ORM – OBJECT RELATIONAL MAPPING

- ▶ Mesmo suporte que oferece django para ORM
- ▶ `Modelo.objects.all()`
- ▶ `Modelo.objects.get(id)`
- ▶ `Modelo.objects.filter()`
- ▶ etc

# WEB BROWSABLE API

- Provee una interfaz web para manejar la API



## SEGURIDAD

- ▶ Authentication
- ▶ Authorization
- ▶ OAuth

## DOCUMENTACION EXTENSA



<https://www.django-rest-framework.org>

DJANGO REST FRAMEWORK

---

**CREANDO UNA API BASICA**

## INSTALACIÓN

```
pip install djangorestframework
```



# CREANDO UNA VISTA SENCILLA

Luego de crear todo el proyecto, app, y configurar Settings.

```
from django.shortcuts import render
from django.http import JsonResponse

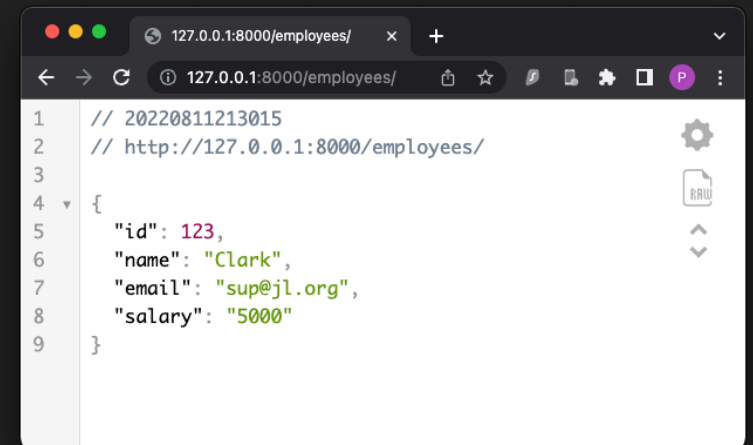
# Create your views here.
def employeeView(request):
    emp = {
        'id': 123,
        'name': 'Clark',
        'email': 'sup@jl.org',
        'salary': '5000'
    }
    return JsonResponse(emp)
```

views.py

```
from restApp import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('employees/', views.employeeView)
]
```

urls.py



http://127.0.0.1:8000/employees/

# CREANDO UN MODELO Y CONECTANDOLO A LA BASE DE DATOS

```
# Create your models here.
class Employee(models.Model):
    id = models.IntegerField(primary_key=True)
    name = models.CharField(max_length=50)
    email = models.CharField(max_length=50)
    salary = models.DecimalField(max_digits=10, decimal_places=2)

    def __str__(self):
        return str(self.id) + " " + self.name + "($ " + str(self.salary) + ")"
```

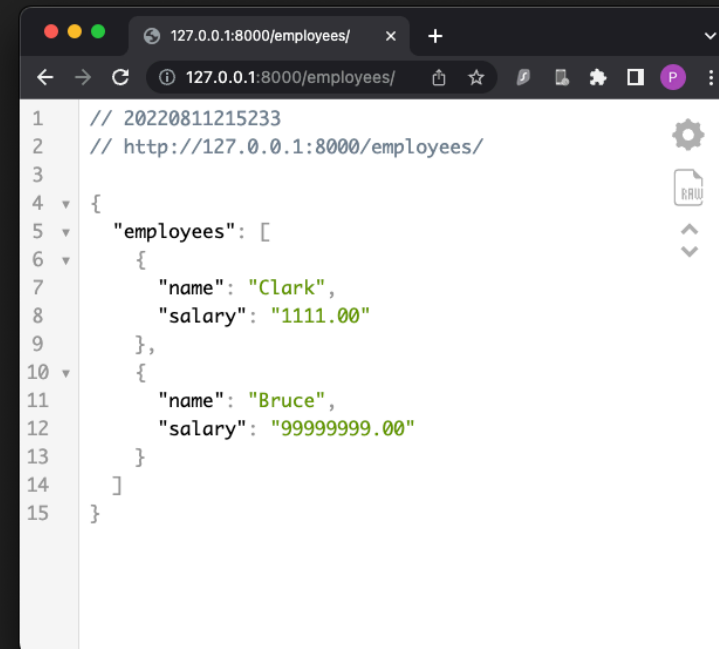
models.py

```
# Create your views here.
def employeeView(request):

    empleados = Employee.objects.all()
    data = {'employees' : list(empleados.values('name', 'salary'))}

    return JsonResponse(data)
```

views.py



The screenshot shows a web browser window with the address bar displaying `127.0.0.1:8000/employees/`. The page content shows a JSON response from the API. The response is a dictionary with a key `"employees"` pointing to a list of two employee objects. Each object has `"name"` and `"salary"` keys. The first employee is Clark with a salary of 1111.00, and the second is Bruce with a salary of 9999999.00. The browser's developer tools are open, showing the raw JSON response.

```
1 // 20220811215233
2 // http://127.0.0.1:8000/employees/
3
4 {
5   "employees": [
6     {
7       "name": "Clark",
8       "salary": "1111.00"
9     },
10    {
11      "name": "Bruce",
12      "salary": "9999999.00"
13    }
14  ]
15 }
```

NO OLVIDAR!  
PYTHON3 MANAGE.PY MAKEMIGRATIONS  
PYTHON3 MANAGE.PY MIGRATE

DJANGO REST FRAMEWORK

---

# SERIALIZERS

## STATUS CODES

- ▶ `status.HTTP_201_CREATED`
- ▶ `status.HTTP_400_BAD_REQUEST`
- ▶ `status.HTTP_204_NO_CONTENT`

## FRAMEWORK WRAPPERS

- ▶ `@api_view` (Funciones)
- ▶ `APIView` (Clases)
- ▶ `Request`
- ▶ `Response`
- ▶ `405 Method not allowed`
- ▶ `request.data`
- ▶ `ParseError`

## FUNCTION BASED VIEWS

- ▶ `@api_view(['GET', 'POST'])`  
`def student_list (request):`
- ▶ `@api_view(['GET', 'PUT', 'DELETE'])`  
`def student_detail (request, id):`

DJANGO REST FRAMEWORK

---

# FUNCTION BASED VIEWS

# CONFIGURACION INICIAL

- ▶ Creamos el proyecto
- ▶ Creamos la aplicación
- ▶ Creamos la BD
- ▶ Configuramos settings.py

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

settings.py



## DJANGO REST FRAMEWORK

---

# CREAMOS EL MODELO Y SERIALIZERS

```
# Create your models here.
class Student(models.Model):
    id = models.IntegerField(primary_key=True)
    name = models.CharField(max_length=50)
    email = models.CharField(max_length=50)
    score = models.DecimalField(max_digits=5, decimal_places=2)

    def __str__(self):
        return str(self.id) + " " + self.name + "(SCORE: " + str(self.score) + ")"
```

models.py

```
from rest_framework import serializers
from serialApp.models import Student

class StudentSerializer(serializers.ModelSerializer):
    class Meta:
        model = Student
        fields = '__all__'
```

serializers.py (hay que crear el archivo)

## DJANGO REST FRAMEWORK

# CREAMOS LAS VISTAS

```
from django.shortcuts import render
from .serializers import StudentSerializer
from .models import Student
from rest_framework.response import Response
from rest_framework import status
from rest_framework.decorators import api_view

# Create your views here.
@api_view(['GET', 'POST'])
def student_list(request):
    if request.method == 'GET':
        students = Student.objects.all()
        serializer = StudentSerializer(students, many=True)
        return Response(serializer.data)

    if request.method == 'POST':
        serializer = StudentSerializer(data = request.data)
        if serializer.is_valid() :
            serializer.save()
            return Response(serializer.data, status=status.HTTP_201_CREATED)
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
```

views.py

```
@api_view(['GET', 'PUT', 'DELETE'])
def student_detail(request, pk):
    try:
        student = Student.objects.get(pk=pk)
    except Student.DoesNotExist:
        return Response(status=status.HTTP_404_NOT_FOUND)

    if request.method == 'GET':
        serializer = StudentSerializer(student)
        return Response(serializer.data)

    if request.method == 'PUT':
        serializer = StudentSerializer(student, data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data)
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

    if request.method == 'DELETE':
        student.delete()
        return Response(status=status.HTTP_204_NO_CONTENT)
```

views.py

# CONFIGURAMOS LAS RUTAS Y EL ADMIN

```
from serialApp import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('students/', views.student_list),
    path('students/<int:pk>', views.student_detail),
]
```

urls.py

```
# Register your models here.
from serialApp.models import Student

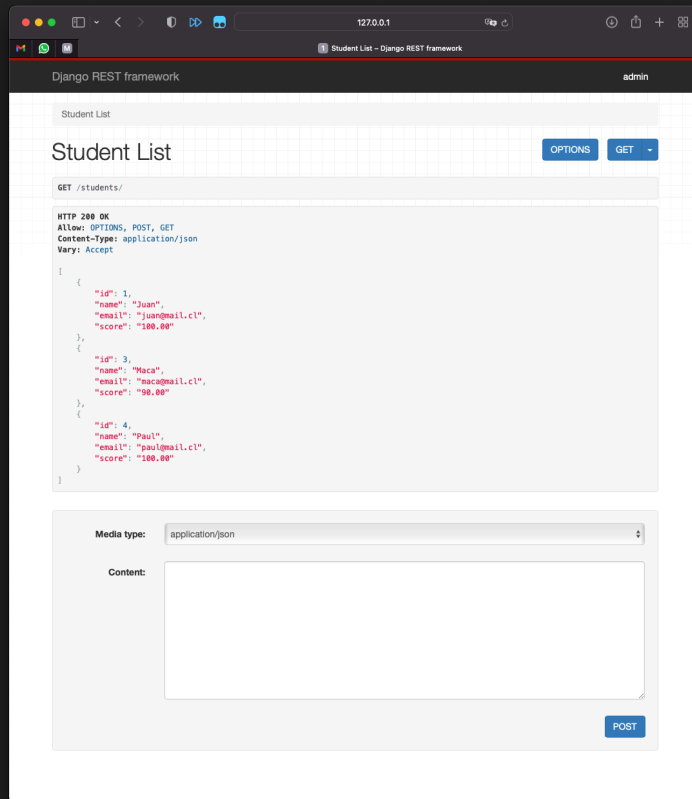
class StudentAdmin(admin.ModelAdmin):
    list_display = ['name', 'email', 'score']

# Register your models here.
admin.site.register(Student, StudentAdmin)
```

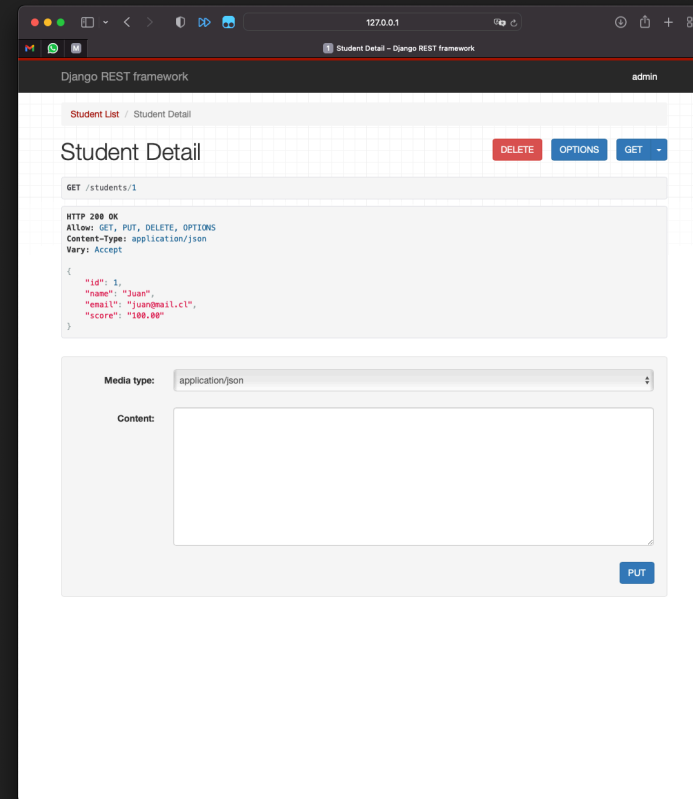
admin.py

# DJANGO REST FRAMEWORK

## PROBAMOS LAS API



<http://127.0.0.1/students>



<http://127.0.0.1/students/1>

DJANGO REST FRAMEWORK

---

# CLASS BASED VIEWS

# CONFIGURACION INICIAL

- ▶ Creamos el proyecto
- ▶ Creamos la aplicación
- ▶ Creamos la BD
- ▶ Configuramos settings.py

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

settings.py

## DJANGO REST FRAMEWORK

---

# CREAMOS EL MODELO Y SERIALIZERS

```
# Create your models here.
class Student(models.Model):
    id = models.IntegerField(primary_key=True)
    name = models.CharField(max_length=50)
    email = models.CharField(max_length=50)
    score = models.DecimalField(max_digits=5, decimal_places=2)

    def __str__(self):
        return str(self.id) + " " + self.name + "(SCORE: " + str(self.score) + ")"
```

models.py

```
from rest_framework import serializers
from cbvApp.models import Student

class StudentSerializer(serializers.ModelSerializer):
    class Meta:
        model = Student
        fields = '__all__'
```

serializers.py (hay que crear el archivo)

## DJANGO REST FRAMEWORK

# CREAMOS LAS VISTAS

```
from django.shortcuts import render
from .serializers import StudentSerializer
from .models import Student
from rest_framework.response import Response
from rest_framework import status
from rest_framework.views import APIView
from django.http import Http404

class StudentList(APIView):

    def get(self, request):
        students = Student.objects.all()
        serializer = StudentSerializer(students, many=True)
        return Response(serializer.data)

    def post(self, request):
        serializer = StudentSerializer(data = request.data)
        if serializer.is_valid() :
            serializer.save()
            return Response(serializer.data, status=status.HTTP_201_CREATED)
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
```

views.py

```
class StudentDetail(APIView):
    def get_object(self, pk):
        try:
            return Student.objects.get(pk=pk)
        except Student.DoesNotExist:
            return Http404

    def get(self, request, pk):
        student = self.get_object(pk)
        serializer = StudentSerializer(student)
        return Response(serializer.data)

    def put(self, request, pk):
        student = self.get_object(pk)
        serializer = StudentSerializer(student, 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):
        student = self.get_object(pk)
        student.delete()
        return Response(status=status.HTTP_204_NO_CONTENT)
```

views.py



# CONFIGURAMOS LAS RUTAS Y EL ADMIN

```
from cbvApp import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('students/', views.StudentList.as_view()),
    path('students/<int:pk>', views.StudentDetail.as_view()),
]
```

urls.py

```
# Register your models here.
from cbvApp.models import Student

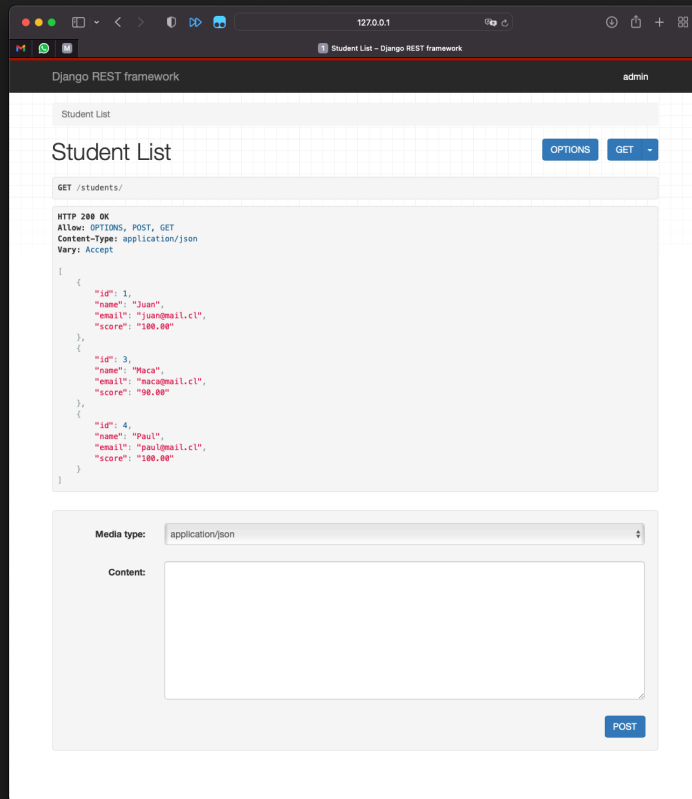
class StudentAdmin(admin.ModelAdmin):
    list_display = ['name', 'email', 'score']

# Register your models here.
admin.site.register(Student, StudentAdmin)
```

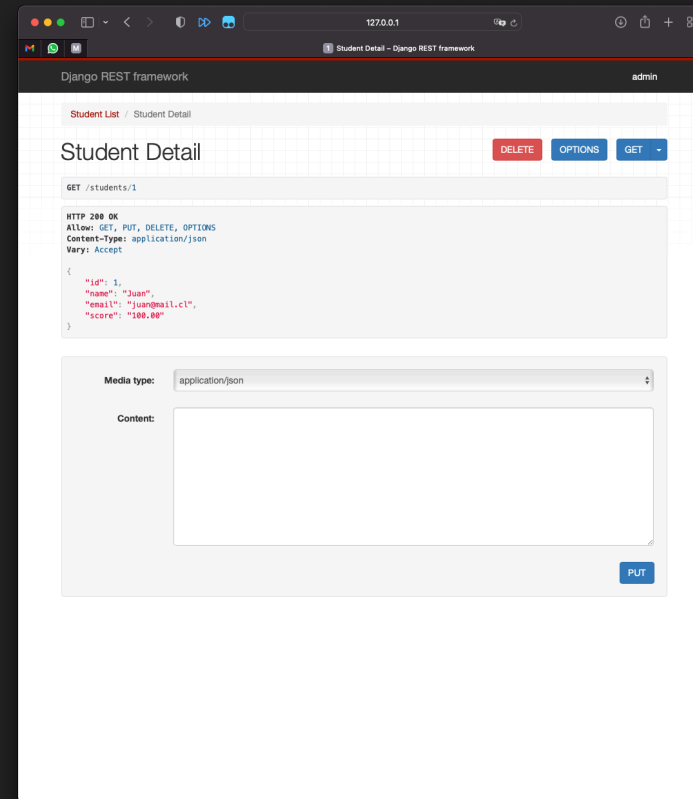
admin.py

# DJANGO REST FRAMEWORK

## PROBAMOS LAS API



<http://127.0.0.1/students>



<http://127.0.0.1/students/1>

DJANGO REST FRAMEWORK

---

**MIXINS**

## MIXINS

### ACTION METHODS

- ▶ list()
- ▶ create()
- ▶ retrieve()
- ▶ update()
- ▶ destroy()

### MIXIN CLASSES

- ▶ ListModelMixin
- ▶ CreateModelMixin
- ▶ RetrieveModelMixin
- ▶ UpdateModelMixin
- ▶ DestroyModelMixin

### HANDLER METHOD

- ▶ get()
- ▶ post()
- ▶ get()
- ▶ put()
- ▶ delete()

# SOLO MODIFICAREMOS LAS VISTAS

```
from rest_framework import generics, mixins

class StudentList(mixins.ListModelMixin, mixins.CreateModelMixin, generics.GenericAPIView):
    queryset = Student.objects.all()
    serializer_class = StudentSerializer

    def get(self, request):
        return self.list(request)

    def post(self, request):
        return self.create(request)

class StudentDetail(mixins.RetrieveModelMixin, mixins.UpdateModelMixin, mixins.DestroyModelMixin, generics.GenericAPIView):
    queryset = Student.objects.all()
    serializer_class = StudentSerializer

    def get(self, request, pk):
        return self.retrieve(request, pk)

    def put(self, request, pk):
        return self.update(request, pk)

    def delete(self, request, pk):
        return self.destroy(request, pk)
```

views.py

DJANGO REST FRAMEWORK

---

# GENERIC

# GENERIC

- ▶ CreateAPIView
- ▶ ListAPIView
- ▶ RetrieveAPIView
- ▶ DestroyAPIView
- ▶ UpdateAPIView
- ▶ **ListCreateAPIView** ← Todas las operaciones sin ID
- ▶ RetrieveUpdateAPIView
- ▶ RetrieveDestroyAPIView
- ▶ **RetrieveUpdateDestroyAPIView** ← Todas las operaciones con ID

# GENERIC – MODIFICANDO LAS VISTAS

```
from rest_framework import generics

class StudentList(generics.ListCreateAPIView):
    queryset = Student.objects.all()
    serializer_class = StudentSerializer

class StudentDetail(generics.RetrieveUpdateDestroyAPIView):
    queryset = Student.objects.all()
    serializer_class = StudentSerializer
```

views.py



DJANGO REST FRAMEWORK

---

# VIEW SETS

## DJANGO REST FRAMEWORK

---

# VIEW SETS

```
from rest_framework import viewsets

class StudentViewSets(viewsets.ModelViewSet):
    queryset = Student.objects.all()
    serializer_class = StudentSerializer
```

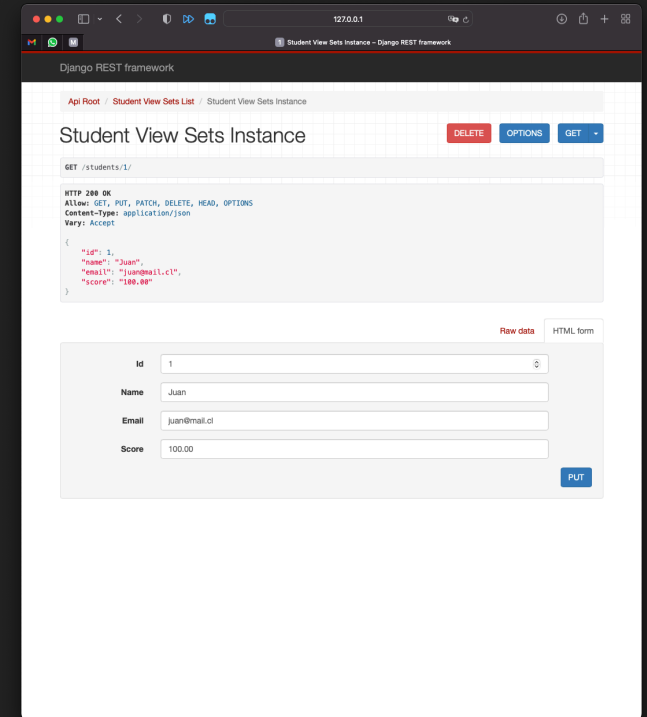
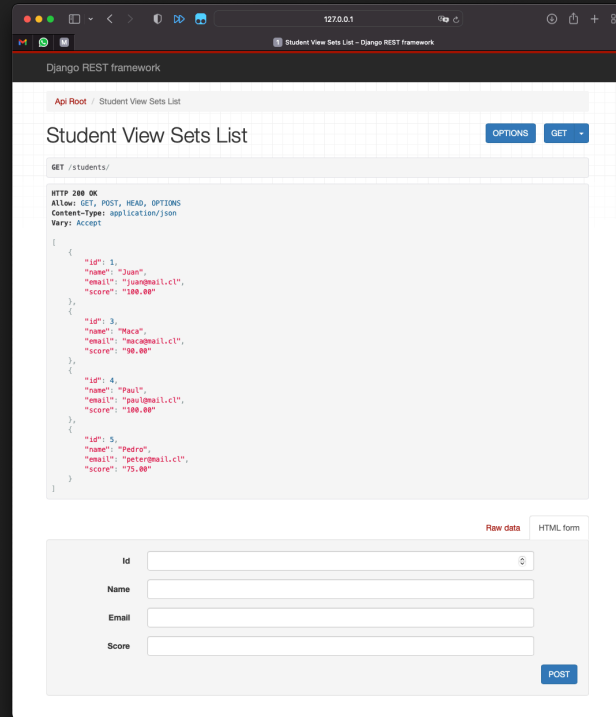
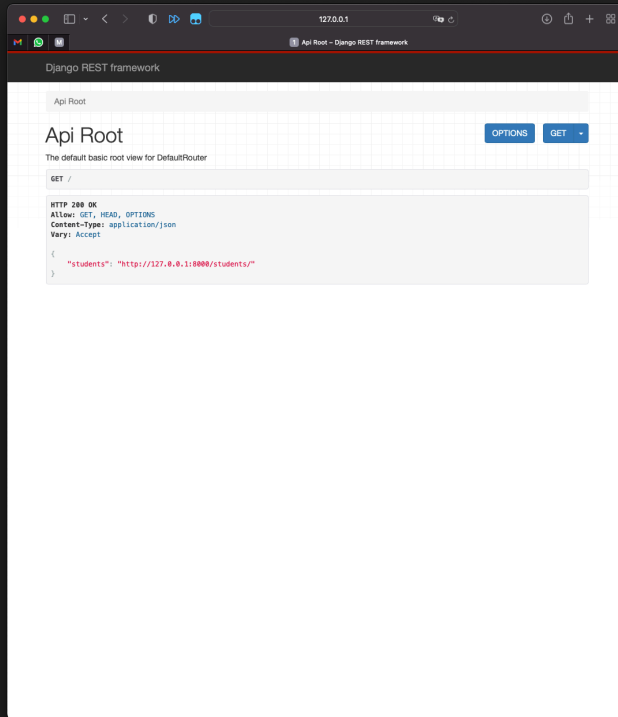
views.py

```
from django.urls import path, include
from rest_framework.routers import DefaultRouter
router = DefaultRouter()
router.register('students', views.StudentViewSets)
urlpatterns = [
    path('', include(router.urls)),
]
```

urls.py

# DJANGO REST FRAMEWORK

## VIEW SETS



DJANGO REST FRAMEWORK

---

**AUTH TOKEN**

## CONTINUARA....

<https://www.django-rest-framework.org/api-guide/authentication/>

<https://simpleisbetterthancomplex.com/tutorial/2018/11/22/how-to-implement-token-authentication-using-django-rest-framework.html>

**FIN :)**