



Programa educativo

INGENIERÍA EN SISTEMAS
COMPUTACIONALES

Grupo

8“A”

Nombre de la materia

ARQUITECTURA DE SERVICIOS

Nombre de los alumnos

MIGUEL ADRIAN JIMENEZ BARAJAS
JUAN DANIEL GALVAN MOCTEZUMA
EDGAR BARAJAS RODRIGUEZ
DIEGO VALDEZ MARTINEZ

Número y Nombre del trabajo

1.1 REPORTE MÉTODO

Unidad # 3

DESARROLLO DE SERVICIOS REST

Nombre del Profesor

ROBERTO SUAREZ ZINZUN

Fecha

13 de mayo de 2025

Tabla de contenido

INTRODUCCIÓN	3
REPOSITORIO.....	3
CUENTAS Y REPORTES.....	10
RUTAS.....	12
MODELO	12
SERVICIOS	13
KYC	30
MODELO	30
RUTAS.....	30
SERVICIOS	31
CONCLUSIONES	41

INTRODUCCIÓN

Durante el desarrollo del proyecto, enfrentamos diversos ajustes necesarios para adaptar los requerimientos de manera más fluida y coherente. A pesar de que no se logró el 100% del desarrollo final, se consiguió avanzar significativamente, cumpliendo con más del 50% de los objetivos planteados. El trabajo colaborativo permitió tomar decisiones que mejoraron la estructura del proyecto y facilitaron su ejecución.

REPOSITORIO

<https://github.com/AdrianJimenezAduaeasy/Brazziono666>

USUARIOS Y AUTENTICACION

Api.py

```
● ● ●

1 from rest_framework import viewsets, permissions
2 from django.contrib.auth.models import User
3 from .serializers import UserSerializer
4
5 class RegisterViewSet(viewsets.ModelViewSet):
6     queryset = User.objects.all()
7     serializer_class = UserSerializer
8     permission_classes = [permissions.AllowAny]
9 
```

Serealizers.py

```
● ● ●

1 from rest_framework import serializers
2 from django.contrib.auth.models import User
3
4 class UserSerializer(serializers.ModelSerializer):
5     class Meta:
6         model = User
7         fields = ['id', 'username', 'email', 'first_name', 'last_name', 'password']
8         extra_kwargs = {
9             'password': {'write_only': True}
10        }
11
12    def create(self, validated_data):
13        user = User.objects.create_user(
14            username=validated_data['username'],
15            email=validated_data.get('email'),
16            password=validated_data['password'],
17            first_name=validated_data.get('first_name', ''),
18            last_name=validated_data.get('last_name', '')
19        )
20        return user
21 
```

Urls.py

```
● ● ●
1  from rest_framework import routers
2  from .api import RegisterViewSet
3  from rest_framework.authtoken.views import obtain_auth_token
4  from django.urls import path
5
6  router = routers.DefaultRouter()
7  router.register(r'api/usuarios', RegisterViewSet, basename='usuarios')
8
9  urlpatterns = router.urls + [
10      path('api/login/', obtain_auth_token, name='api_token_auth'),
11  ]
12
```

Pruebas: Login (POST)

HTTP **Usuarios / Login** Save Share

POST <http://127.0.0.1:8000/users/api/login/> Send

Params Authorization Headers (9) **Body** Scripts Settings Cookies Beautify

none form-data x-www-form-urlencoded raw binary GraphQL **JSON**

```
1 {
2     "username": "Edgar123",
3     "password": "123"
4 }
```

Body Cookies Headers (9) Test Results ⏱ 200 OK • 299 ms • 348 B • ⏵ Save Response ⚙️

[{}] JSON ▶ Preview ⏷ Visualization

token 2bd2ab50af868b2b44d6f744f0ce9129f3745cb2

Registrar Usuario (POST)

HTTP Usuarios / RegistrarUsuario

POST http://127.0.0.1:8000/users/api/usuarios/

Send

Params Authorization Headers (9) Body Scripts Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```

1 {
2   "username": "usuario2",
3   "email": "user1@example.com",
4   "password": "12345678",
5   "first_name": "Juan",
6   "last_name": "Pérez"
7 }
```

Body Cookies Headers (10) Test Results

201 Created 348 ms 425 B Save Response

{ } JSON Preview Visualization

```

1 {
2   "id": 3,
3   "username": "usuario2",
4   "email": "user1@example.com",
5   "first_name": "Juan",
6   "last_name": "Pérez"
7 }
```

Actualizar Usuario (PATCH)

HTTP Usuarios / EditarUsuario

PATCH http://127.0.0.1:8000/users/api/usuarios/1

Send

Params Authorization Headers (9) Body Scripts Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```

1 {
2   "id": 1,
3   "username": "Edgar123",
4   "email": "sistemas@aduaeeasy.com",
5   "first_name": "Edgar",
6   "last_name": "Barajas"
7 }
```

Body Cookies Headers (10) Test Results

200 OK 13 ms 441 B Save Response

{ } JSON Preview Visualization

```

1 {
2   "id": 1,
3   "username": "Edgar123",
4   "email": "sistemas@aduaeeasy.com",
5   "first_name": "Edgar",
6   "last_name": "Barajas"
7 }
```

Eliminar Usuario (DELETE)

HTTP Usuarios / EliminarUsuario

DELETE http://127.0.0.1:8000/users/api/usuarios

Send

Params Authorization Headers (7) Body Scripts Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	...

Body Cookies Headers (10) Test Results

200 OK 5 ms 429 B Save Response

{ } JSON Preview Visualization

```

1 [
2   {
3     "id": 1,
4     "username": "Edgar123",
5     "email": "sistemas@aduaeeasy.com",
6     "first_name": "Edgar",
7     "last_name": "Barajas"
8   }
9 ]

```

Ver Usuarios (GET)

HTTP Usuarios / Verusuarios

GET http://127.0.0.1:8000/users/api/usuarios/

Send

Params Authorization Headers (7) Body Scripts Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	...

Body Cookies Headers (10) Test Results

200 OK 27 ms 529 B Save Response

{ } JSON Preview Visualization

```

1 [
2   {
3     "id": 1,
4     "username": "Edgar123",
5     "email": "sistemas@aduaeeasy.com",
6     "first_name": "Edgar",
7     "last_name": "Barajas"
8   },
9   {
10    "id": 3,
11    "username": "usuario2",
12    "email": "user1@example.com",
13    "first_name": "Juan",
14    "last_name": "Pérez"
15  }
16 ]

```

PAGOS

Api.py

```
● ● ●
1 # views.py
2
3 from rest_framework import viewsets, permissions
4 from .models import Pago
5 from .serializers import PagoSerializer
6
7 class PagoViewSet(viewsets.ModelViewSet):
8     serializer_class = PagoSerializer
9     permission_classes = [permissions.IsAuthenticated]
10
11    def get_queryset(self):
12        return Pago.objects.filter(usuario=self.request.user).order_by('-fecha')
13
14    def perform_create(self, serializer):
15        serializer.save(usuario=self.request.user)
16
```

Serealizers.py

```
● ● ●
1 # serializers.py
2
3 from rest_framework import serializers
4 from .models import Pago
5
6 class PagoSerializer(serializers.ModelSerializer):
7     class Meta:
8         model = Pago
9         fields = ['id', 'tipo', 'monto', 'concepto', 'fecha']
10        read_only_fields = ['fecha']
11
12
13
```

Urls.py

```

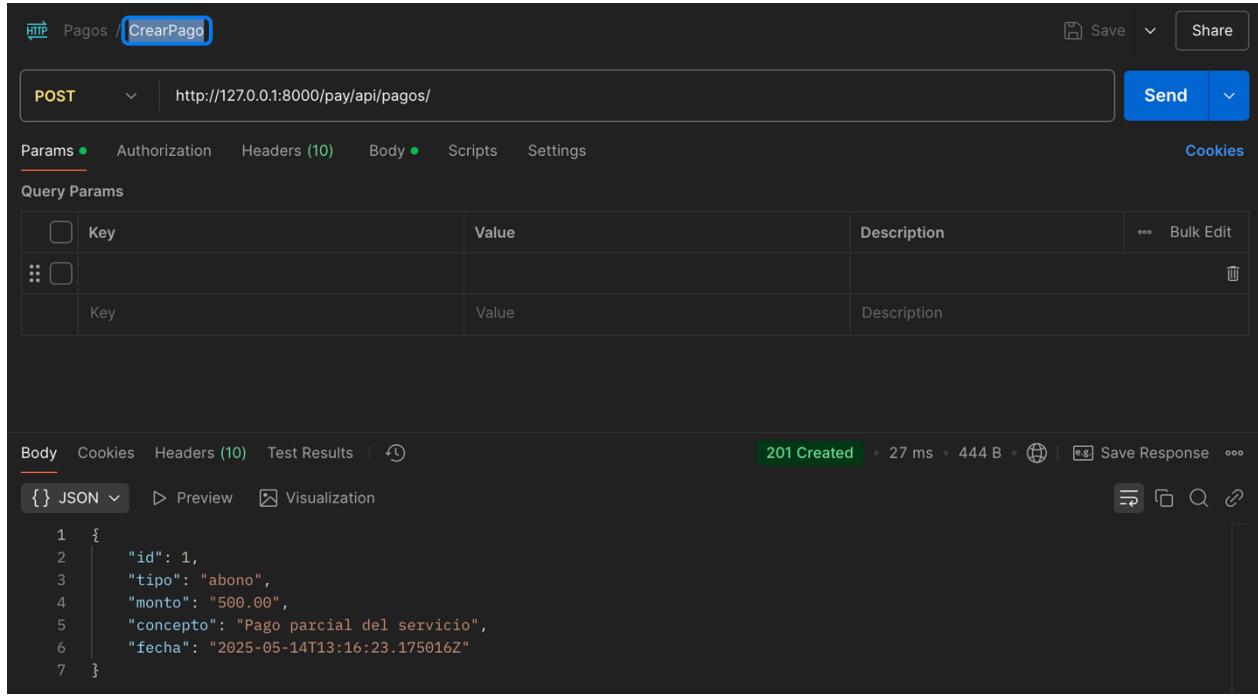
● ● ●

1 # urls.py
2 from rest_framework import routers
3 from rest_framework.routers import DefaultRouter
4 from .api import PagoViewSet
5
6 router = routers.DefaultRouter()
7 router.register(r'api/pagos', PagoViewSet, basename='pagos')
8
9 urlpatterns = router.urls
10

```

Pruebas:

Agregar Pago (POST)



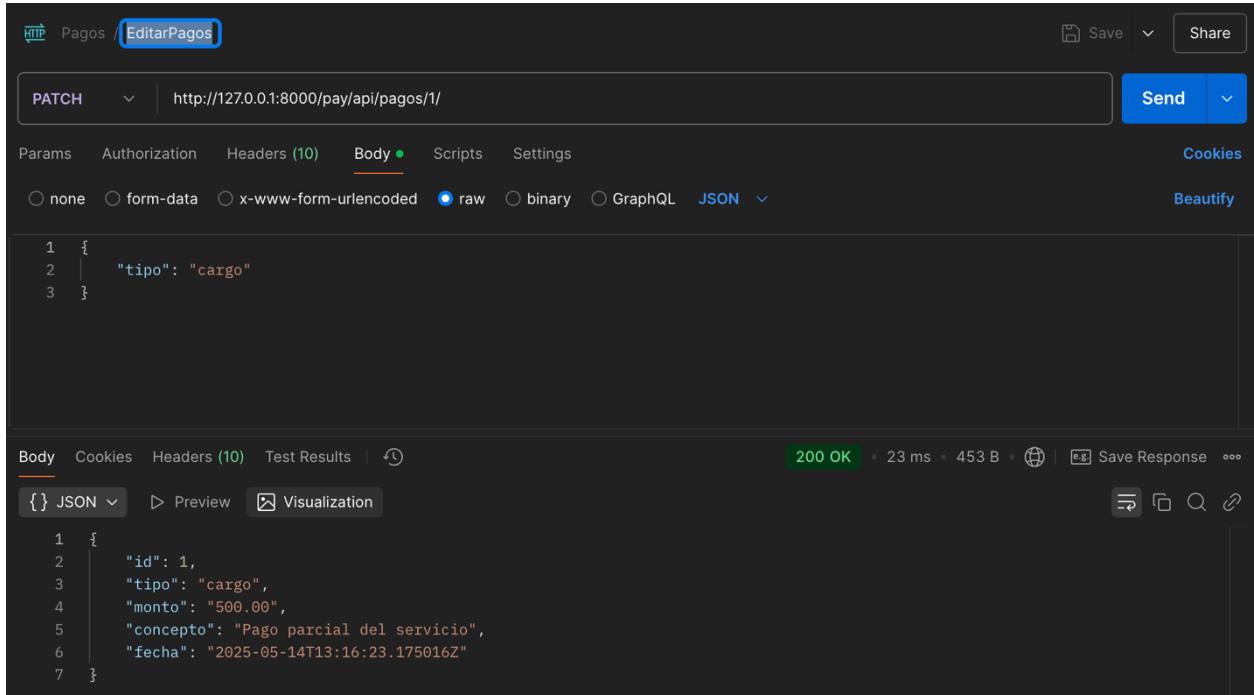
The screenshot shows the Postman application interface. At the top, there's a header bar with 'HTTP' set to 'Pages', a 'Clear Page' button, and 'Save' and 'Share' options. Below the header, a 'POST' dropdown is set to 'http://127.0.0.1:8000/pay/api/pagos/'. The 'Params' tab is selected, showing a table with one row: 'Key' (empty) and 'Value' (empty). Other tabs like 'Authorization', 'Headers (10)', 'Body', 'Scripts', and 'Settings' are visible. In the bottom right corner, the status bar shows '201 Created' with a timestamp of '27 ms', a file size of '444 B', and a globe icon. Below the status bar, there are buttons for 'JSON', 'Preview', and 'Visualization'. The JSON preview area displays the following JSON response:

```

1 {
2     "id": 1,
3     "tipo": "abono",
4     "monto": "500.00",
5     "concepto": "Pago parcial del servicio",
6     "fecha": "2025-05-14T13:16:23.175016Z"
7 }

```

Editar Pago (PATCH)



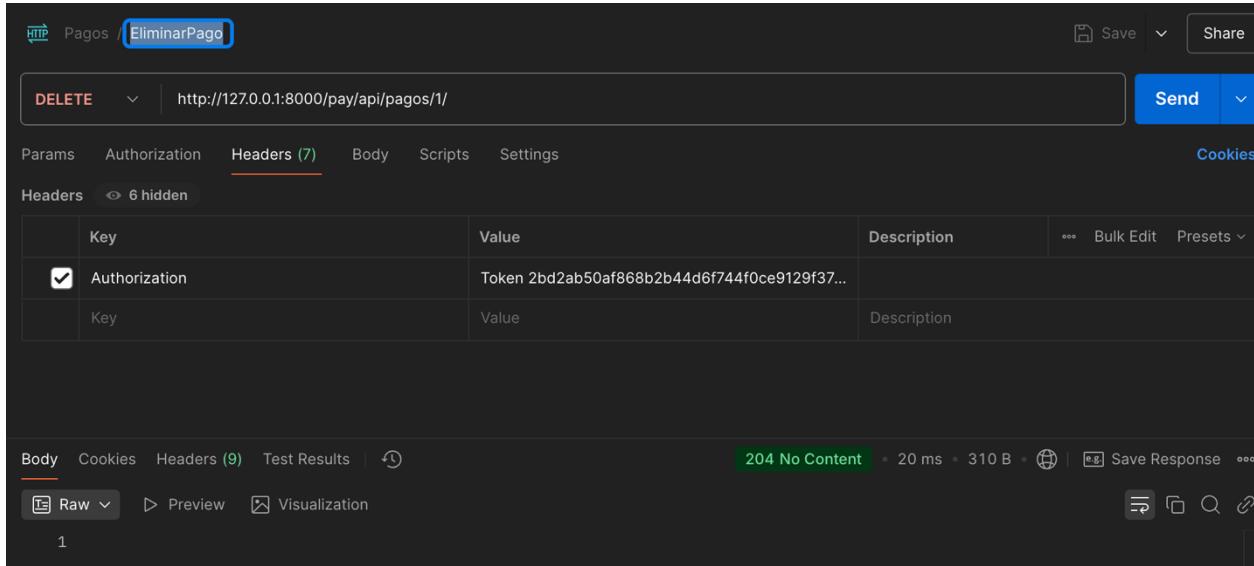
PATCH <http://127.0.0.1:8000/pay/api/pagos/1>

Params Authorization Headers (10) **Body** Scripts Settings Cookies Beautify

```
1 {  
2   "tipo": "cargo"  
3 }
```

200 OK • 23 ms • 453 B | Save Response ⚙️

Eliminar Pago (DELETE)



DELETE <http://127.0.0.1:8000/pay/api/pagos/1>

Params Authorization Headers (7) Body Scripts Settings Cookies

Headers 6 hidden

Key	Value	Description	Bulk Edit	Presets
<input checked="" type="checkbox"/> Authorization	Token 2bd2ab50af868b2b44d6f744f0ce9129f37...			
Key	Value	Description		

204 No Content • 20 ms • 310 B | Save Response ⚙️

Ver Pago (GET)

The screenshot shows the Postman application interface. At the top, there are tabs for 'HTTP' (selected), 'Pagos' (disabled), and 'VerPagos' (selected). On the right, there are 'Save' and 'Share' buttons. Below the tabs, a 'Send' button is visible. The main area shows a 'GET' request with the URL `http://127.0.0.1:8000/pay/api/pagos/`. The 'Headers' tab is selected, showing one header: 'Authorization' with the value 'Token 2bd2ab50af868b2b44d6f744f0ce9129f37...'. Other tabs include 'Params', 'Authorization', 'Body', 'Scripts', and 'Settings'. On the right, there are 'Cookies' and '6 hidden' buttons. Below the headers table, there are sections for 'Body', 'Cookies', 'Headers (10)', 'Test Results', and a preview section showing a JSON response with one item: [{}]. The status bar at the bottom indicates a '200 OK' response with 11 ms latency and 322 B size.

CUENTAS Y REPORTES

Desarrollador por: Miguel Adrian Jimenez Barajas

RUTAS

```

● ○ ●
1 from django.contrib import admin
2 from django.urls import path
3 from . import services
4 urlpatterns = [
5     path('admin/', admin.site.urls),
6     path('cuentas/crear', services.crear_cuenta, name='crear_cuenta'),
7     path('cuentas/actualizar/<int:cuenta_id>', services.actualizar_cuenta, name='actualizar_cuenta'),
8     path('cuentas/consultar', services.obtener_cuentas, name='listar_cuentas'),
9     path('cuentas/eliminar/<int:cuenta_id>', services.eliminar_cuenta, name='eliminar_cuenta'),
10    path('cuentas/consultar/<int:cuenta_id>', services.obtener_cuenta, name='consultar_cuenta'),
11    path('cuentas/recargar/<int:cuenta_id>', services.recargar_cuenta, name='recargar_cuenta'),
12    path('cuentas/reportes/crear', services.crear_reporte_cuenta, name='crear_reporte'),
13    path('cuentas/reportes/consultar', services.consultar_reportes, name='consultar_reportes'),
14    path('cuentas/reportes/consultar/<int:cuenta_id>', services.obtener_reportes_cuenta, name='consultar_reportes'),
15 ]

```

MODELO

```

● ○ ●
1 class Cuenta(models.Model):
2     id = models.AutoField(primary_key=True)
3     saldo = models.DecimalField(max_digits=10, decimal_places=2)
4     estado = models.CharField(
5         max_length=20,
6         choices=[
7             ('ACTIVA', 'Activa'),
8             ('INACTIVA', 'Inactiva'),
9         ],
10        default='ACTIVA',
11    )
12     usuario = models.ForeignKey(User, on_delete=models.CASCADE)
13
14     def __str__(self):
15         return f"{self.nombre} - {self.saldo}"

```

SERVICIOS

Crear Cuenta

```

1  @csrf_exempt
2  @api_view(['POST'])
3  def crear_cuenta(request):
4      if not request.data:
5          return Response({
6              'status': status.HTTP_400_BAD_REQUEST,
7              'message': 'Los datos de la cuenta no pueden estar vacíos',
8              'data': None
9          }, status=status.HTTP_400_BAD_REQUEST)
10
11     if Cuenta.objects.filter(usuario=request.data.get('usuario')).exists():
12         return Response({
13             'status': status.HTTP_400_BAD_REQUEST,
14             'message': 'Ya existe una cuenta para este usuario',
15             'data': None
16         }, status=status.HTTP_400_BAD_REQUEST)
17
18     serializer = CuentaSerializer(data=request.data)
19     if serializer.is_valid():
20         cuenta = serializer.save()
21         return Response({
22             'status': status.HTTP_201_CREATED,
23             'message': 'Cuenta creada exitosamente',
24             'data': CuentaSerializer(cuenta).data
25         }, status=status.HTTP_201_CREATED)
26     else:
27         return Response({
28             'status': status.HTTP_400_BAD_REQUEST,
29             'message': 'Error en los datos de la cuenta',
30             'data': serializer.errors
31         }, status=status.HTTP_400_BAD_REQUEST)

```

Restricciones

- No se permite Body vacío
- No permite mas de una cuenta por usuario
- Validar tipos de datos

Pruebas

POST http://127.0.0.1:8080/cuentas/crear

Params Authorization Headers (9) **Body** Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL

```

1  {
2      "saldo": 10000,
3      "estado": "ACTIVA",
4      "usuario": 1
5  }

```

Body Cookies Headers (10) Test Results |

{ } JSON Preview Visualize |

```

1  {
2      "status": 400,
3      "message": "Ya existe una cuenta para este usuario",
4      "data": null
5  }

```

HTTP brazzino666 / cuentas / crear cuenta

POST http://127.0.0.1:8080/cuentas/crear

Params Authorization Headers (9) **Body** Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL **JSON**

```

1  {
2      "saldo": 10000,
3      "estado": "ACTIVA",
4      "usuario": 1
5  }

```

Body Cookies Headers (10) Test Results |

{ } JSON Preview Visualize |

```

1  {
2      "status": 201,
3      "message": "Cuenta creada exitosamente",
4      "data": {
5          "id": 2,
6          "saldo": "10000.00",
7          "estado": "ACTIVA",
8          "usuario": 1
9      }
10 }

```

Consultar Cuenta

```
● ● ●

1  @csrf_exempt
2  @api_view(['GET'])
3  def obtener_cuenta(request, cuenta_id):
4      if not cuenta_id:
5          return Response({
6              'status': status.HTTP_400_BAD_REQUEST,
7              'message': 'El ID de usuario no puede ser nulo',
8              'data': None
9          }, status=status.HTTP_400_BAD_REQUEST)
10
11     try:
12         cuenta = Cuenta.objects.get(usuario= cuenta_id)
13         return Response({
14             'status': status.HTTP_200_OK,
15             'message': 'Cuenta obtenida exitosamente',
16             'data': CuentaSerializer(cuenta).data
17         })
18     except Cuenta.DoesNotExist:
19         return Response({
20             'status': status.HTTP_404_NOT_FOUND,
21             'message': 'Cuenta no encontrada',
22             'data': None
23         }, status=status.HTTP_404_NOT_FOUND)
24
```

Restricciones

- Id de usuario no puede ser nulo

Pruebas

GET <http://127.0.0.1:8080/cuentas/consultar/1>

Params Authorization Headers (9) Body ● Scripts Settings

Body Cookies Headers (10) Test Results | ⌂

{ } JSON ▾ ▷ Preview ⚡ Visualize | ▾

```
1  {
2    "status": 200,
3    "message": "Cuenta obtenida exitosamente",
4    "data": {
5      "id": 2,
6      "saldo": "10400.00",
7      "estado": "ACTIVA",
8      "usuario": 1
9    }
10 }
```

GET <http://127.0.0.1:8080/cuentas/consultar/2>

Params Authorization Headers (9) Body ● Scripts

Body Cookies Headers (10) Test Results | ⌂

{ } JSON ▾ ▷ Preview ⚡ Visualize | ▾

```
1  {
2    "status": 404,
3    "message": "Cuenta no encontrada",
4    "data": null
5 }
```

Consultar Cuentas

```
● ● ●

1  @csrf_exempt
2  @api_view(['GET'])
3  def obtener_cuentas(request): # Cambiado a recibir request
4      cuentas = Cuenta.objects.all()
5      if cuentas.exists():
6          return Response({
7              'status': status.HTTP_200_OK,
8              'message': 'Cuentas obtenidas exitosamente',
9              'data': CuentaSerializer(cuentas, many=True).data
10         })
11     else:
12         return Response({
13             'status': status.HTTP_404_NOT_FOUND,
14             'message': 'No se encontraron cuentas',
15             'data': None
16         }, status=status.HTTP_404_NOT_FOUND)
17
```

Restricciones

- Ninguna

Pruebas

GET <http://127.0.0.1:8080/cuentas/consultar>

Params Authorization Headers (9) Body ● Scripts Settings

Body Cookies Headers (10) Test Results | ⚙️

{ } JSON ▾ ▷ Preview ⚙️ Visualize | ▾

```
1  {
2      "status": 200,
3      "message": "Cuentas obtenidas exitosamente",
4      "data": [
5          {
6              "id": 1,
7              "saldo": "1000.00",
8              "estado": "INACTIVA",
9              "usuario": 1
10         }
11     ]
12 }
```

Actualizar Cuenta

```

1  @csrf_exempt
2  @api_view(['PUT'])
3  def actualizar_cuenta(request, cuenta_id): # Cambiado el orden de los parámetros
4      if not cuenta_id or not request.data:
5          return Response({
6              'status': status.HTTP_400_BAD_REQUEST,
7              'message': 'Datos incompletos para la actualización',
8              'data': None
9          }, status=status.HTTP_400_BAD_REQUEST)
10
11     try:
12         cuenta = Cuenta.objects.get(id=cuenta_id)
13         serializer = CuentaSerializer(cuenta, data=request.data, partial=True)
14
15         if serializer.is_valid():
16             cuenta_actualizada = serializer.save()
17             return Response({
18                 'status': status.HTTP_200_OK,
19                 'message': 'Cuenta actualizada exitosamente',
20                 'data': CuentaSerializer(cuenta_actualizada).data
21             })
22         else:
23             return Response({
24                 'status': status.HTTP_400_BAD_REQUEST,
25                 'message': 'Error en los datos de actualización',
26                 'data': serializer.errors
27             }, status=status.HTTP_400_BAD_REQUEST)
28
29     except Cuenta.DoesNotExist:
30         return Response({
31             'status': status.HTTP_404_NOT_FOUND,
32             'message': 'Cuenta no encontrada',
33             'data': None
34         }, status=status.HTTP_404_NOT_FOUND)
35

```

Restricciones

- Id de la cuenta obligatorio
- Validacion de campos

Pruebas

PUT http://127.0.0.1:8080/cuentas/actualizar/1

Params Authorization Headers (9) **Body** Scripts Settings

none form-data x-www-form-urlencoded raw binary

```
1 {  
2   "saldo": 1000,  
3   "estado": "INACTIVA",  
4   "usuario": 1  
5 }
```

Body Cookies Headers (10) Test Results

{ } JSON Preview

```
1 {  
2   "status": 200,  
3   "message": "Cuenta actualizada exitosamente",  
4   "data": {  
5     "id": 1,  
6     "saldo": "1000.00",  
7     "estado": "INACTIVA",  
8     "usuario": 1  
9   }  
10 }
```

Eliminar Cuenta

```
● ● ●

1 @csrf_exempt
2 @api_view(['DELETE'])
3 def eliminar_cuenta(request, cuenta_id): # Agregado request como primer parámetro
4     if not cuenta_id:
5         return Response({
6             'status': status.HTTP_400_BAD_REQUEST,
7             'message': 'El ID de cuenta no puede ser nulo',
8             'data': None
9         }, status=status.HTTP_400_BAD_REQUEST)
10
11    try:
12        cuenta = Cuenta.objects.get(id=cuenta_id)
13        cuenta.delete()
14        return Response({
15            'status': status.HTTP_200_OK,
16            'message': 'Cuenta eliminada exitosamente',
17            'data': None
18        })
19    except Cuenta.DoesNotExist:
20        return Response({
21            'status': status.HTTP_404_NOT_FOUND,
22            'message': 'Cuenta no encontrada',
23            'data': None
24        }, status=status.HTTP_404_NOT_FOUND)
```

Restricciones

- Id de cuenta obligatorio
- La cuenta debe existir

Pruebas

DELETE ▾ | http://127.0.0.1:8080/cuentas/eliminar/2

Params Authorization Headers (9) Body • Scripts Settings

Query Params

	Key
	Key

Body Cookies Headers (10) Test Results | ⏱

{ } JSON ▾ ▷ Preview ⚡ Visualize | ▾

```
1  {
2      "status": 200,
3      "message": "Cuenta eliminada exitosamente",
4      "data": null
5 }
```

DELETE ▾ | http://127.0.0.1:8080/cuentas/eliminar/3

Params Authorization Headers (9) Body • Scripts Settings

Query Params

	Key
	Key

Body Cookies Headers (10) Test Results | ⏱

{ } JSON ▾ ▷ Preview ⚡ Visualize | ▾

```
1  {
2      "status": 404,
3      "message": "Cuenta no encontrada",
4      "data": null
5 }
```

Recargar Cuenta

```

● ● ●

1  @csrf_exempt
2  @api_view(['PUT'])
3  def recargar_cuenta(request, cuenta_id):
4      if not cuenta_id or not request.data:
5          return Response({
6              'status': status.HTTP_400_BAD_REQUEST,
7              'message': 'Datos incompletos para la recarga',
8              'data': None
9          }, status=status.HTTP_400_BAD_REQUEST)
10
11     try:
12         cuenta = Cuenta.objects.get(id=cuenta_id)
13         cantidad = request.data.get('cantidad')
14
15         if cantidad <= 0:
16             return Response({
17                 'status': status.HTTP_400_BAD_REQUEST,
18                 'message': 'La cantidad a recargar debe ser mayor que cero',
19                 'data': None
20             }, status=status.HTTP_400_BAD_REQUEST)
21
22         cuenta.saldo += cantidad
23         cuenta.save()
24
25         return Response({
26             'status': status.HTTP_200_OK,
27             'message': 'Cuenta recargada exitosamente',
28             'data': CuentaSerializer(cuenta).data
29         })
30     except Cuenta.DoesNotExist:
31         return Response({
32             'status': status.HTTP_404_NOT_FOUND,
33             'message': 'Cuenta no encontrada',
34             'data': None
35         }, status=status.HTTP_404_NOT_FOUND)

```

Restricciones

- Id de la cuenta obligatorio
- La cuenta debe existir
- La cantidad debe ser mayor a 0

Pruebas

PUT ▾ | <http://127.0.0.1:8080/cuentas/recargar/3>

Params Authorization Headers (9) **Body** ● Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL **JSON** ▾

```

1  {
2    "cantidad": 100
3  }

```

Body Cookies Headers (10) Test Results | ⏱

{ } JSON ▾ ▷ Preview ⚡ Visualize | ▾

```

1  {
2    "status": 200,
3    "message": "Cuenta recargada exitosamente",
4    "data": {
5      "id": 3,
6      "saldo": "10100.00",
7      "estado": "ACTIVA",
8      "usuario": 1
9    }
10 }

```

PUT ▾ | <http://127.0.0.1:8080/cuentas/recargar/4>

Params Authorization Headers (9) **Body** ● Scripts Se

none form-data x-www-form-urlencoded raw

```

1  {
2    "cantidad": 100
3  }

```

Body Cookies Headers (10) Test Results | ⏱

{ } JSON ▾ ▷ Preview ⚡ Visualize | ▾

```

1  {
2    "status": 404,
3    "message": "Cuenta no encontrada",
4    "data": null
5  }

```

Crear Reporte Cuenta

```
1 @csrf_exempt
2 @api_view(['POST'])
3 def crear_reporte_cuenta(request):
4     if not request.data:
5         return Response({
6             'status': status.HTTP_400_BAD_REQUEST,
7             'message': 'Los datos del reporte no pueden estar vacíos',
8             'data': None
9         }, status=status.HTTP_400_BAD_REQUEST)
10
11    serializer = ReporteCuentaSerializer(data=request.data)
12    if serializer.is_valid():
13        reporte = serializer.save()
14        return Response({
15            'status': status.HTTP_201_CREATED,
16            'message': 'Reporte creado exitosamente',
17            'data': ReporteCuentaSerializer(reporte).data
18        }, status=status.HTTP_201_CREATED)
19    else:
20        return Response({
21            'status': status.HTTP_400_BAD_REQUEST,
22            'message': 'Error en los datos del reporte',
23            'data': serializer.errors
24        }, status=status.HTTP_400_BAD_REQUEST)
```

Restricciones

- Validacion de datos

Pruebas

POST <http://127.0.0.1:8080/cuentas/reportes/crear>

Params Authorization Headers (9) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary Graph

```
1 {  
2   "cuenta": 3,  
3   "motivo": "intento hacer trampa en la ruleta",  
4   "severidad": "ALTA"  
5 }
```

Body Cookies Headers (10) Test Results | ⏱

{ } JSON ▾ ▷ Preview ⚡ Visualize | ▾

```
1 {  
2   "status": 201,  
3   "message": "Reporte creado exitosamente",  
4   "data": [  
5     {"id": 3,  
6       "fecha_reporte": "2025-05-14T06:01:42.411776Z",  
7       "motivo": "intento hacer trampa en la ruleta",  
8       "severidad": "ALTA",  
9       "cuenta": 3  
10    ]  
11 }
```

Consultar Reportes de Cuenta

```
● ● ●

1 @csrf_exempt
2 @api_view(['GET'])
3 def obtener_reportes_cuenta(request, cuenta_id):
4     if not cuenta_id:
5         return Response({
6             'status': status.HTTP_400_BAD_REQUEST,
7             'message': 'El ID de cuenta no puede ser nulo',
8             'data': None
9         }, status=status.HTTP_400_BAD_REQUEST)
10
11    reportes = ReporteCuenta.objects.filter(cuenta=cuenta_id)
12    if reportes.exists():
13        return Response({
14            'status': status.HTTP_200_OK,
15            'message': 'Reportes obtenidos exitosamente',
16            'data': ReporteCuentaSerializer(reportes, many=True).data
17        })
18    else:
19        return Response({
20            'status': status.HTTP_404_NOT_FOUND,
21            'message': 'No se encontraron reportes para esta cuenta',
22            'data': None
23        }, status=status.HTTP_404_NOT_FOUND)
```

Restricciones

- Id de la cuenta obligatorio

Pruebas

GET http://127.0.0.1:8080/cuentas/reportes/consultar/2

Params Authorization Headers (7) **Body** Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
  {
    "status": 200,
    "message": "Reportes obtenidos exitosamente",
    "data": [
      {
        "id": 1,
        "fecha_reporte": "2025-05-14T04:09:01.890552Z",
        "motivo": "intento hacer trampa",
        "severidad": "MEDIA",
        "cuenta": 2
      },
      {
        "id": 2,
        "fecha_reporte": "2025-05-14T04:09:22.979384Z",
        "motivo": "intento hacer trampa en la ruleta",
        "severidad": "ALTA",
        "cuenta": 2
      }
    ]
  }

```

GET http://127.0.0.1:8080/cuentas/reportes/consultar/4

Params Authorization Headers (7) **Body** Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL

```

1
2
3
4
5
  {
    "status": 404,
    "message": "No se encontraron reportes para esta cuenta",
    "data": null
  }

```

Consultar Reportes

```

● ● ●

1 @csrf_exempt
2 @api_view(['GET'])
3 def consultar_reportes(request):
4     reportes = ReporteCuenta.objects.all()
5     if reportes.exists():
6         return Response({
7             'status': status.HTTP_200_OK,
8             'message': 'Reportes obtenidos exitosamente',
9             'data': ReporteCuentaSerializer(reportes, many=True).data
10        })
11    else:
12        return Response({
13            'status': status.HTTP_404_NOT_FOUND,
14            'message': 'No se encontraron reportes',
15            'data': None
16        }, status=status.HTTP_404_NOT_FOUND)

```

Restricciones

- Ninguna

Pruebas

GET <http://127.0.0.1:8080/cuentas/reportes/consultar>

Params	Authorization	Headers (7)	Body	Scripts	Settings
Body	Cookies	Headers (10)	Test Results		
{} JSON ▾ ▷ Preview Visualize ▾					
<pre> 1 { 2 "status": 200, 3 "message": "Reportes obtenidos exitosamente", 4 "data": [5 { 6 "id": 3, 7 "fecha_reporte": "2025-05-14T06:01:42.411776Z", 8 "motivo": "intento hacer trampa en la ruleta", 9 "severidad": "ALTA", 10 "cuenta": 3 11 } 12] 13 }</pre>					

KYC

Desarrollador por: Miguel Adrian Jimenez Barajas

MODELO

```
● ● ●

1 from django.db import models
2 from django.contrib.auth.models import User
3
4
5 class VerificacionKYC(models.Model):
6     id = models.AutoField(primary_key=True)
7     usuario = models.ForeignKey(User, on_delete=models.CASCADE)
8     documento_identidad = models.CharField(max_length=255)
9     documento_identidad_direccion = models.CharField(max_length=255)
10    foto = models.CharField(max_length=255)
11    estado = models.CharField(
12        max_length=20,
13        choices=[
14            ('PENDIENTE', 'Pendiente'),
15            ('APROBADO', 'Aprobado'),
16            ('RECHAZADO', 'Rechazado'),
17        ],
18        default='PENDIENTE',
19    )
20
```

RUTAS

```
● ● ●

1 from django.contrib import admin
2 from django.urls import path
3 from . import services
4
5 urlpatterns = [
6     path('admin/', admin.site.urls),
7     path('kyc/crear', services.crear_kyc, name='crear_kyc'),
8     path('kyc/actualizar/<int:kyc_id>', services.actualizar_kyc, name='actualizar_kyc'),
9     path('kyc/consultar', services.listar_kyc, name='listar_kycs'),
10    path('kyc/consultar/<int:kyc_id>', services.obtener_kyc, name='consultar_kyc'),
11]
```

SERVICIOS

Crear VerificacionKYC

```

1  @csrf_exempt
2  @api_view(['POST'])
3  def crear_kyc(request):
4      if not request.data:
5          return Response({
6              'status': status.HTTP_400_BAD_REQUEST,
7              'message': 'Los datos de KYC no pueden estar vacíos',
8              'data': None
9          }, status=status.HTTP_400_BAD_REQUEST)
10
11     if VerificacionKYC.objects.filter(usuario=request.data.get('usuario')).exists():
12         return Response({
13             'status': status.HTTP_400_BAD_REQUEST,
14             'message': 'Ya existe un KYC para este usuario',
15             'data': None
16         }, status=status.HTTP_400_BAD_REQUEST)
17     serializer = KycSerializer(data=request.data)
18     if serializer.is_valid():
19         kyc = serializer.save()
20         return Response({
21             'status': status.HTTP_201_CREATED,
22             'message': 'KYC creado exitosamente',
23             'data': KycSerializer(kyc).data
24         }, status=status.HTTP_201_CREATED)
25     else:
26         return Response({
27             'status': status.HTTP_400_BAD_REQUEST,
28             'message': 'Error en los datos de KYC',
29             'data': serializer.errors
30         }, status=status.HTTP_400_BAD_REQUEST)

```

Restricciones

- No se permiten campos vacíos
- Un usuario no puede tener mas de una verificación

Pruebas

POST ▾ http://127.0.0.1:8080/kyc/crear

Params Authorization Headers (9) **Body** ● Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL **JSON** ▾

```

1 {
2   "usuario": 2,
3   "documento_identidad": "https://storage.googleapis.com/testagricolato/ine.jpg",
4   "documento_identidad_direccion": "https://storage.googleapis.com/testagricolato/comprobanteDomicilio.jpeg",
5   "foto": "https://storage.googleapis.com/testagricolato/foto.jpeg"
6 }
```

Body Cookies Headers (10) Test Results | ⚡

{ } JSON ▾ ▶ Preview ⚡ Visualize | ▾

```

1 {
2   "status": 201,
3   "message": "KYC creado exitosamente",
4   "data": {
5     "id": 3,
6     "documento_identidad": "https://storage.googleapis.com/testagricolato/ine.jpg",
7     "documento_identidad_direccion": "https://storage.googleapis.com/testagricolato/comprobanteDomicilio.jpeg",
8     "foto": "https://storage.googleapis.com/testagricolato/foto.jpeg",
9     "estado": "PENDIENTE",
10    "usuario": 2
11  }
12 }
```

POST ▾ http://127.0.0.1:8080/kyc/crear

Params Authorization Headers (9) **Body** ● Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL **JSON** ▾

```

1 {
2   "usuario": 1,
3   "documento_identidad": "https://storage.googleapis.com/testagricolato/ine.jpg",
4   "documento_identidad_direccion": "https://storage.googleapis.com/testagricolato/comprobanteDomicilio.jpeg",
5   "foto": "https://storage.googleapis.com/testagricolato/foto.jpeg"
6 }
```

Body Cookies Headers (10) Test Results | ⚡

{ } JSON ▾ ▶ Preview ⚡ Visualize | ▾

```

1 {
2   "status": 400,
3   "message": "Ya existe un KYC para este usuario",
4   "data": null
5 }
```

Actualizar VerificacionKYC

```

1  @csrf_exempt
2  @api_view(['PUT'])
3  def actualizar_kyc(request, kyc_id):
4      if not kyc_id:
5          return Response({
6              'status': status.HTTP_400_BAD_REQUEST,
7              'message': 'El ID de KYC no puede ser nulo',
8              'data': None
9          }, status=status.HTTP_400_BAD_REQUEST)
10
11     try:
12         kyc = VerificacionKYC.objects.get(id=kyc_id)
13     except VerificacionKYC.DoesNotExist:
14         return Response({
15             'status': status.HTTP_404_NOT_FOUND,
16             'message': 'KYC no encontrado',
17             'data': None
18         }, status=status.HTTP_404_NOT_FOUND)
19
20     serializer = KycSerializer(kyc, data=request.data)
21     if serializer.is_valid():
22         kyc = serializer.save()
23         return Response({
24             'status': status.HTTP_200_OK,
25             'message': 'KYC actualizado exitosamente',
26             'data': KycSerializer(kyc).data
27         }, status=status.HTTP_200_OK)
28     else:
29         return Response({
30             'status': status.HTTP_400_BAD_REQUEST,
31             'message': 'Error en los datos de KYC',
32             'data': serializer.errors
33         }, status=status.HTTP_400_BAD_REQUEST)

```

Restricciones

- El id de la verificacion no puede ser nulo
- Validacion de datos

Pruebas

PUT <http://127.0.0.1:8080/kyc/actualizar/1>

Params Authorization Headers (9) **Body** Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL **JSON**

```

1  {
2    "usuario": 1,
3    "documento_identidad": "https://storage.googleapis.com/testagricolato/ine.jpg",
4    "documento_identidad_direccion": "https://storage.googleapis.com/testagricolato/comprobanteDomicilio.jpeg",
5    "foto": "https://storage.googleapis.com/testagricolato/foto.jpeg",
6    "estado": "RECHAZADO"
7  }

```

Body Cookies Headers (10) Test Results | ⏪

{ } JSON ▾ ▶ Preview ⏪ Visualize | ⏪

```

1  {
2    "status": 200,
3    "message": "KYC actualizado exitosamente",
4    "data": {
5      "id": 1,
6      "documento_identidad": "https://storage.googleapis.com/testagricolato/ine.jpg",
7      "documento_identidad_direccion": "https://storage.googleapis.com/testagricolato/comprobanteDomicilio.jpeg",
8      "foto": "https://storage.googleapis.com/testagricolato/foto.jpeg",
9      "estado": "RECHAZADO",
10     "usuario": 1
11   }
12 }

```

PUT <http://127.0.0.1:8080/kyc/actualizar/10>

Params Authorization Headers (9) **Body** Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL **JSON**

```

1  {
2    "usuario": 1,
3    "documento_identidad": "https://storage.googleapis.com/testagricolato/ine.jpg",
4    "documento_identidad_direccion": "https://storage.googleapis.com/testagricolato/comprobanteDomicilio.jpeg",
5    "foto": "https://storage.googleapis.com/testagricolato/foto.jpeg",
6    "estado": "RECHAZADO"
7  }

```

Body Cookies Headers (10) Test Results | ⏪

{ } JSON ▾ ▶ Preview ⏪ Visualize | ⏪

```

1  {
2    "status": 404,
3    "message": "KYC no encontrado",
4    "data": null
5  }

```

Consulta Individual VerificacionKYC

```
 1  @csrf_exempt
 2  @api_view(['GET'])
 3  def obtener_kyc(request, kyc_id):
 4      if not kyc_id:
 5          return Response({
 6              'status': status.HTTP_400_BAD_REQUEST,
 7              'message': 'El ID de KYC no puede ser nulo',
 8              'data': None
 9          }, status=status.HTTP_400_BAD_REQUEST)
10
11     try:
12         kyc = VerificacionKYC.objects.get(id=kyc_id)
13         return Response({
14             'status': status.HTTP_200_OK,
15             'message': 'KYC obtenido exitosamente',
16             'data': KycSerializer(kyc).data
17         })
18     except VerificacionKYC.DoesNotExist:
19         return Response({
20             'status': status.HTTP_404_NOT_FOUND,
21             'message': 'KYC no encontrado',
22             'data': None
23         }, status=status.HTTP_404_NOT_FOUND)
```

Restricciones

- El id no puede ser nulo

Pruebas

GET <http://127.0.0.1:8080/kyc/consultar/1>

Params Authorization Headers (9) Body Scripts Settings

Body Cookies Headers (10) Test Results | ⏱

{ } JSON ▾ Preview ⚡ Visualize | ▾

```

1  {
2      "status": 200,
3      "message": "KYC obtenido exitosamente",
4      "data": {
5          "id": 1,
6          "documento_identidad": "https://storage.googleapis.com/testagricolato/ine.jpg",
7          "documento_identidad_direccion": "https://storage.googleapis.com/testagricolato/comprobanteDomicilio.jpeg",
8          "foto": "https://storage.googleapis.com/testagricolato/foto.jpeg",
9          "estado": "RECHAZADO",
10         "usuario": 1
11     }
12 }
```

Consulta VerificacionKYC

```

● ● ●

1 @csrf_exempt
2 @api_view(['GET'])
3 def listar_kyc(request):
4     kyc = VerificacionKYC.objects.all()
5     serializer = KycSerializer(kyc, many=True)
6     return Response({
7         'status': status.HTTP_200_OK,
8         'message': 'Lista de KYC obtenida exitosamente',
9         'data': serializer.data
10    }, status=status.HTTP_200_OK)
```

Restricciones

- Ninguna

Pruebas

GET <http://127.0.0.1:8080/kyc/consultar>

Params Authorization Headers (9) Body ● Scripts Settings

Body Cookies Headers (10) Test Results | ⏪

{ } JSON ▾ Preview ⏪ Visualize ▾

```

1  {
2      "status": 200,
3      "message": "Lista de KYC obtenida exitosamente",
4      "data": [
5          {
6              "id": 1,
7              "documento_identidad": "https://storage.googleapis.com/testagricolato/ine.jpg",
8              "documento_identidad_direccion": "https://storage.googleapis.com/testagricolato/comprobanteDomicilio.jpeg",
9              "foto": "https://storage.googleapis.com/testagricolato/foto.jpeg",
10             "estado": "RECHAZADO",
11             "usuario": 1
12         },
13         {
14             "id": 3,
15             "documento_identidad": "https://storage.googleapis.com/testagricolato/ine.jpg",
16             "documento_identidad_direccion": "https://storage.googleapis.com/testagricolato/comprobanteDomicilio.jpeg",
17             "foto": "https://storage.googleapis.com/testagricolato/foto.jpeg",
18             "estado": "PENDIENTE",
19             "usuario": 2
20         }
21     ]
22 }
```

Apuestas

Modelos

```

● ○ ●
1  from django.db import models
2
3  class Apuesta(models.Model):
4      id_usuario = models.IntegerField(db_column='id_usuario')
5      id_juego = models.IntegerField(db_column='id_juego')
6      monto_apuesta = models.DecimalField(max_digits=10, decimal_places=2)
7      fecha_apuesta = models.DateTimeField(auto_now_add=True)
8      estado = models.CharField(max_length=50)
9
10     class Meta:
11         db_table = 'apuesta'
12
13     def __str__(self):
14         return f"Apuesta #{self.id} - Usuario {self.id_usuario}"

```

Rutas

```
● ● ●

1 from django.urls import path
2 from . import views
3
4 urlpatterns = [
5     path('', views.home, name='apuestas_home'),
6     path('realizar/', views.realizar_apuesta, name='realizar_apuesta'),
7     path('listar/', views.listar_apuestas, name='listar_apuestas'),
8 ]
```

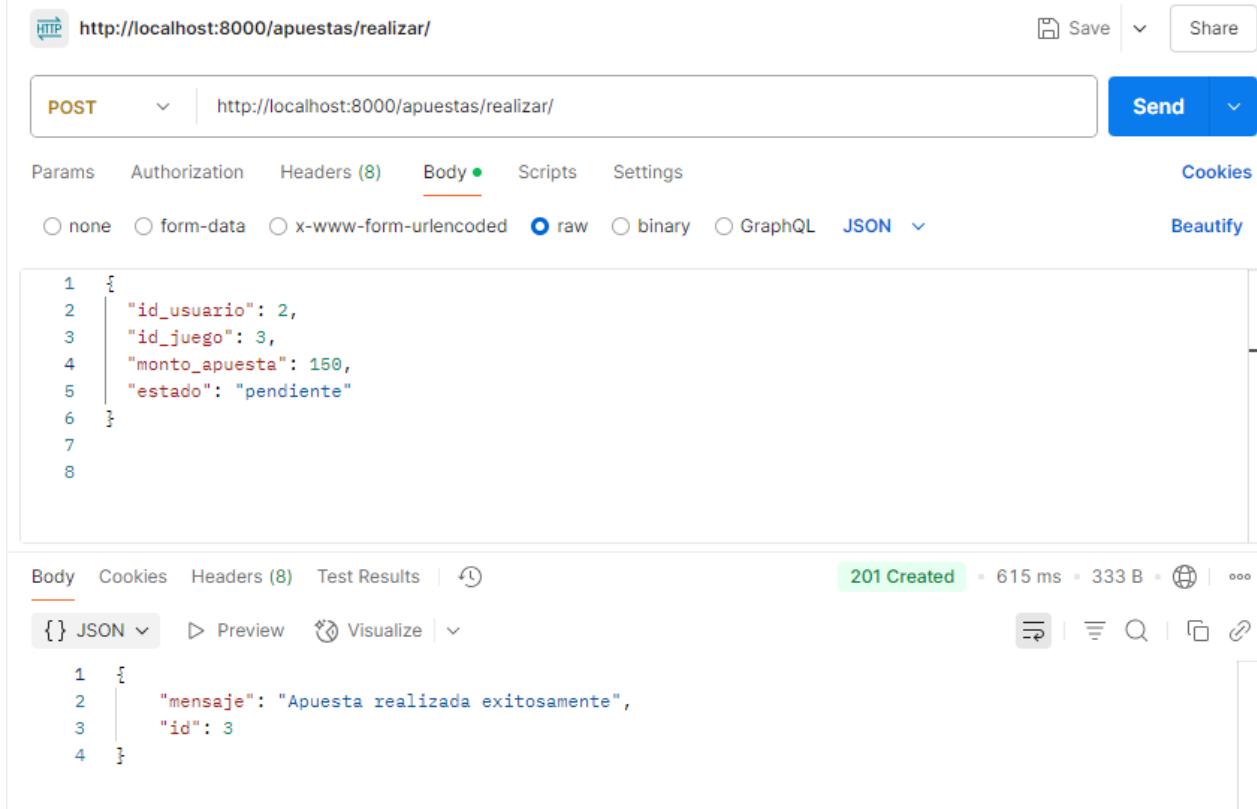
Servicios

Realizar apuestas

```
● ● ●

1 @csrf_exempt
2 def realizar_apuesta(request):
3     if request.method == 'POST':
4         data = json.loads(request.body)
5         form = ApuestaForm(data)
6         if form.is_valid():
7             apuesta = form.save()
8             return JsonResponse({'mensaje': 'Apuesta realizada exitosamente', 'id': apuesta.id}, status=201)
9         else:
10             return JsonResponse({'errores': form.errors}, status=400)
11     else:
12         return JsonResponse({'mensaje': 'Usa POST para enviar una apuesta'}, status=200)
13
14 from django.http import JsonResponse
15 from .models import Apuesta
```

Purebas



HTTP <http://localhost:8000/apuestas/realizar/>

POST <http://localhost:8000/apuestas/realizar/>

Body (8) **Headers** **Params** **Authorization** **Scripts** **Settings** **Cookies**

none form-data x-www-form-urlencoded raw binary GraphQL **JSON** Beautify

```

1  {
2    "id_usuario": 2,
3    "id_juego": 3,
4    "monto_apuesta": 150,
5    "estado": "pendiente"
6  }
7
8

```

Body Cookies Headers (8) Test Results 

201 Created • 615 ms • 333 B •  

{ } JSON   Preview  Visualize 

1 {
2 "mensaje": "Apuesta realizada exitosamente",
3 "id": 3
4 }

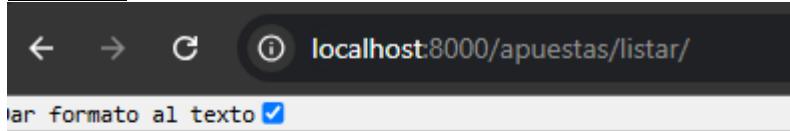
Listar apuestas

```
● ● ●  
1  def listar_apuestas(request):  
2      apuestas = Apuesta.objects.all().order_by('-fecha_apuesta')  
3  
4      lista = [  
5          {  
6              'id': a.id,  
7              'id_usuario': a.id_usuario,  
8              'id_juego': a.id_juego,  
9              'monto_apuesta': float(a.monto_apuesta),  
10             'fecha_apuesta': a.fecha_apuesta.isoformat(),  
11             'estado': a.estado,  
12         }  
13         for a in apuestas  
14     ]
```

Restricciones

- Ninguna

Pruebas



```
{
  "id": 3,
  "id_usuario": 2,
  "id_juego": 3,
  "monto_apuesta": 150,
  "fecha_apuesta": "2025-05-14T06:53:04.089279+00:00",
  "estado": "pendiente"
},
{
  "id": 2,
  "id_usuario": 2,
  "id_juego": 2,
  "monto_apuesta": 1550,
  "fecha_apuesta": "2025-05-14T05:16:16.002384+00:00",
  "estado": "pendiente"
},
{
  "id": 1,
  "id_usuario": 1,
  "id_juego": 2,
  "monto_apuesta": 1500,
  "fecha_apuesta": "2025-05-14T04:59:16.239981+00:00",
  "estado": "pendiente"
}
```

Result Grid						
	id	id_usuario	id_juego	monto_apuesta	fecha_apuesta	estado
▶	1	1	2	1500.00	2025-05-14 04:59:16.239981	pendiente
	2	2	2	1550.00	2025-05-14 05:16:16.002384	pendiente
	3	2	3	150.00	2025-05-14 06:53:04.089279	pendiente
*	NULL	NULL	NULL	NULL	NULL	NULL

CONCLUSIONES

Jimenez Barajas Miguel Adrian:

La verdad me parece que con el presente cumplimos mas del 50% pero fue un gran avance, hicimos algunos cambios para que el desarrollo fuera mas comodo pero se cumplio con los requerimientos de la app.

Juan Daniel Galvan Moctezuma

Mediante llevabamos a cabo el desarrollo tuvimos que realizar ciertos cambios para que los requerimientos se acoplaran de mejor forma entre si, con lo desarrollado creo que logramos mas del %50 asi que creo que fue un buen avance.

Edgar Barajas Rodríguez

A lo largo del proceso de desarrollo realizamos varios ajustes importantes que facilitaron el cumplimiento de los requerimientos. Aunque el proyecto aún no está terminado con mas del porcentaje requerido, logramos avanzar considerablemente y dejar una base sólida sobre la cual seguir construyendo. Considero que el trabajo en equipo fue clave para lograr más del 50% del progreso actual.