DOCUMENTACIÓN PROYECTO SOCIOS MALLORCA:

REPOSITORIO:

https://github.com/Bsilvaor/Proyecto_Examen/

Hemos creado un proyecto mediante Django al cual he llamado proyecto_mallorca que crea una clase Socio con los campos (DNI (PK), Número de socio y contraseña)

NOTA: he optado por poner el DNI como primary key ya que me parece más lógico que utilizar el id que se crea por defecto ya que el DNI también será único.

Como se nos pide en el examen, procedo a introducir las capturas de mi **views.py** y mis **urls.py** tanto del **proyecto_mallorca** como de la app **sociosmallorca:**

views.py:

```python
from django.shortcuts import render
from django.http import JsonResponse
from django.views.decorators.csrf import csrf_exempt
from .models import Socio
import json

# Create your views here.

@csrf_exempt
def introducir_socio(request):
    if request.method == 'POST':
        try:
            body = json.loads(request.body.decode('utf-8'))
            dni = body.get('dni')
            numerosocio = body.get('numerosocio')
            contraseña = body.get('contraseña')

            Socio.objects.create(dni=dni, numerosocio=numerosocio, contraseña=contraseña)
            return JsonResponse({'mensaje': 'Socio creado correctamente'})
        except Exception as e:
            return JsonResponse({'error': str(e)}, status=500)


def obtener_socio(request):
    if request.method == 'GET':
        socios = Socio.objects.all()
        lista_socios = [{'dni': Socio.dni, 'numerosocio': Socio.numerosocio} for Socio in socios]
        return JsonResponse({'Socios': lista_socios})
    else:
        return JsonResponse({'error': 'Método no permitido'}, status=405)

@csrf_exempt
def borrar_socio(request):
        try:
            body = json.loads(request.body.decode('utf-8'))
            dni = body.get('dni')
            socios = Socio.objects.get(dni=dni)
            socios.delete()
            return JsonResponse({'mensaje': 'Alumno eliminado correctamente'})
        except Socio.DoesNotExist:
            return JsonResponse({'error': 'Alumno no encontrado'}, status=404)
```

NOTAS:

He decidido que ya que el **obtener_socio** se hace por GET, en la variable **lista_socios** sólo mostrar el DNI y el número de socio:

En **introducir_socio** utilizo la función **TRY** junto con **EXCEPT** para crear un socio y en caso de que no se cree correctamente actúe el **except Exception as e**, que captura o coge cualquier tipo de excepción que pueda ocurrir durante la ejecución del **TRY**

En **obtener_socio** en cambio **utilizo** un **ELSE** ya que es una vista por GET de los socios ya creados, en los cuales como ya he dicho anteriormente solo se visualizan el DNI y el número de socio por motivos de seguridad obviando la contraseña.

En **borrar_socio** el try-except **Socio.DoesNotExist:** Al igual que en la primera función, este bloque try intenta hacer algo, en este caso, intenta eliminar un socio de la base de datos. Si ocurre un error mientras se intenta hacer esto, el bloque except se activa. En este caso, se está capturando un tipo específico de error que Django proporciona cuando intentamos

eliminar un socio que no existe en la base de datos. Por lo tanto, si el socio que intentamos eliminar no se encuentra en la base de datos, se enviará un mensaje de error diciendo que el socio no se encontró.

urls.py (sociosmallorca app):

```python
from django.urls import path
from . import views

urlpatterns = [
    path('introducir_socio/', views.introducir_socio, name='introducir_socio'),
    path('obtener_socio/', views.obtener_socio, name='obtener_socio'),
    path('borrar_socio/', views.borrar_socio, name='borrar_socio'),
    path('editar_contraseñasocio/', views.editar_contraseñasocio, name='editar_contraseñasocio'),
]
```

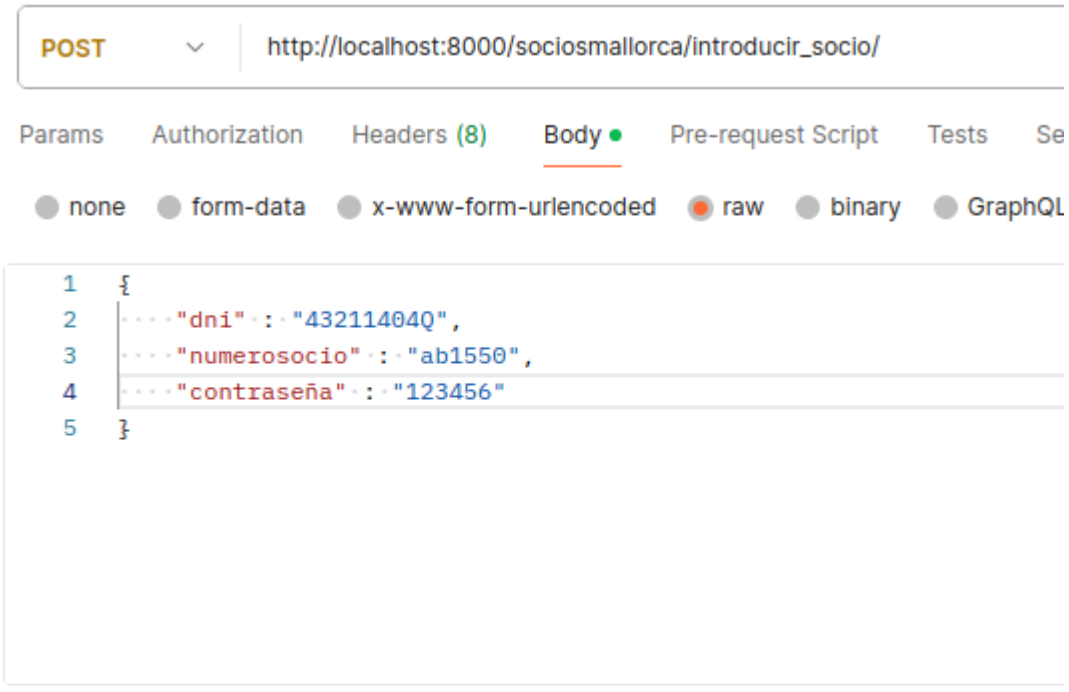
urls.py(proyecto_mallorca):

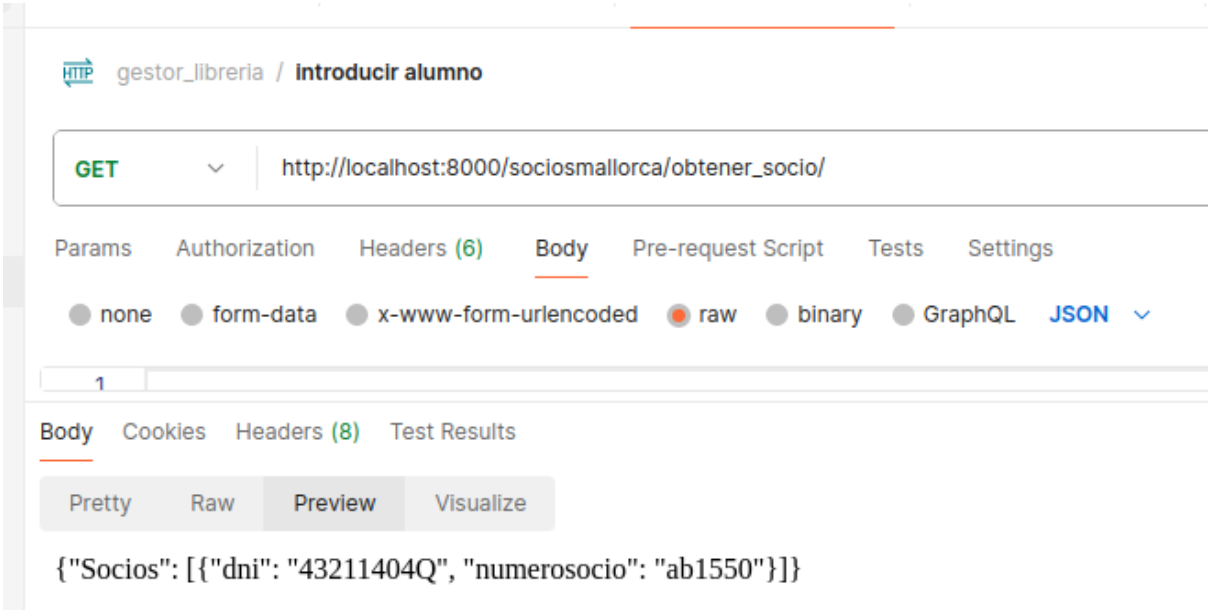
```python
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('sociosmallorca/', include('sociosmallorca.urls')),
]
```

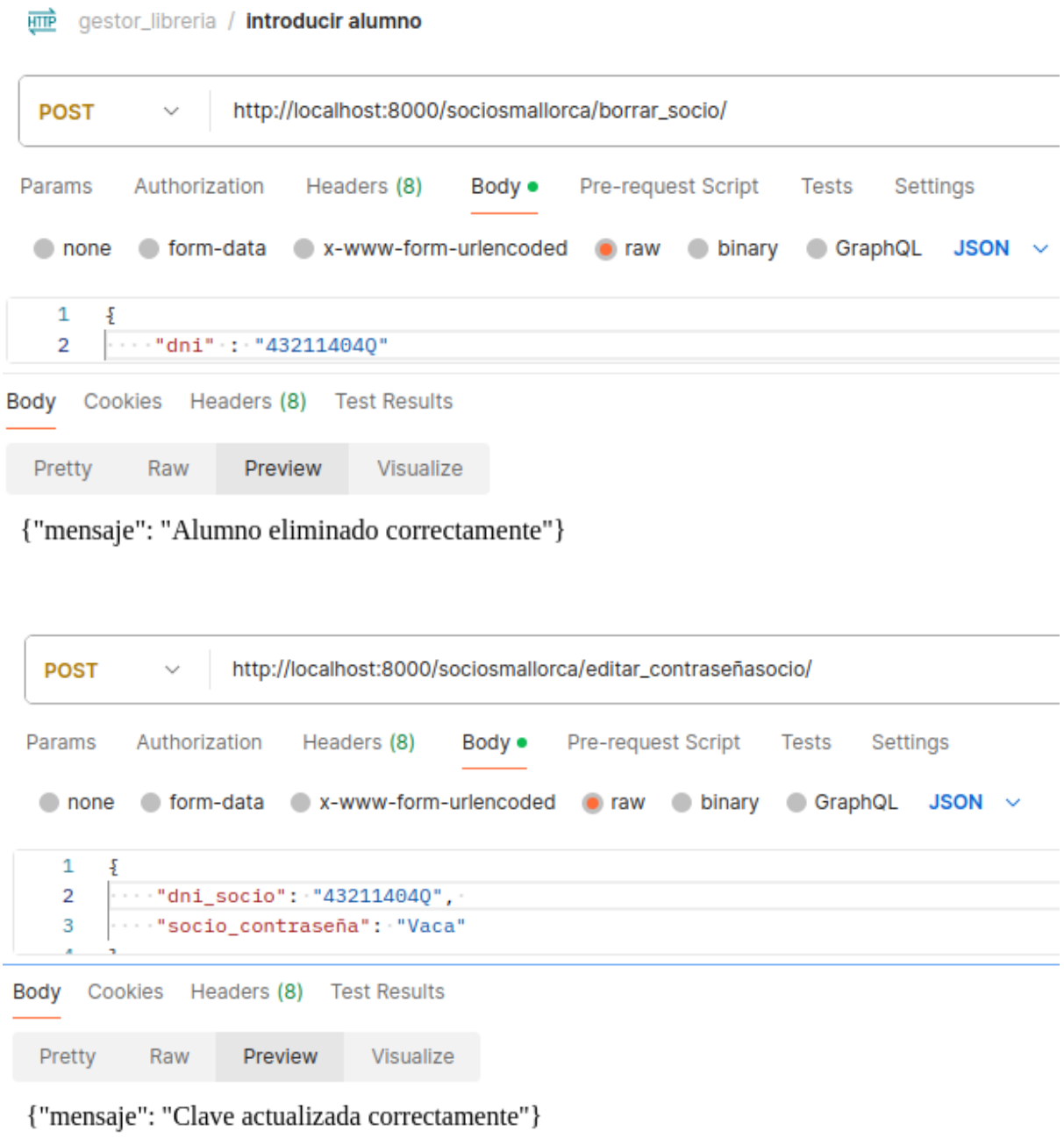
Pregunta extra:

Que tablas crearías en un tablero de trello para llevar a cabo este proyecto? Se te ocurre alguna tarjeta? ¿Cómo dividir el trabajo?
Se tiene que entregar en PDF y explicando un poco lo que habéis hecho.

IMÁGENES DE COMPROBACIÓN DE LA APP EN POSTMAN:


POST
http://localhost:8000/sociosmallorca/introducir_socio/
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Se
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
{
"dni": "43211404Q",
"numerosocio": "ab1550",
"contraseña": "123456"
}
Body
Cookies
Headers (8)
Test Results
Pretty
Raw
Preview
Visualize
{"mensaje": "Socio creado correctamente"}


gestor_libreria / Introducir alumno
GET
http://localhost:8000/sociosmallorca/obtener_socio/
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Body
Cookies
Headers (8)
Test Results
Pretty
Raw
Preview
Visualize
{"Socios": [{"dni": "43211404Q", "numerosocio": "ab1550"}]}



PREGUNTA EXTRA:

RESPUESTA:

Para organizar un proyecto de desarrollo de software en un tablero de Trello lo haría de la siguiente manera (según la documentación que he podido ver):

Lista de Backlog o Lista de Tareas Pendientes:
Esta lista contendrá todas las tareas pendientes de realizar durante el proyecto.
Aquí se pueden incluir tarjetas para funcionalidades específicas, correcciones de errores, tareas de infraestructura, etc.

Ejemplo de tarjeta: "Implementar vista para cambiar contraseña de socio"

Lista de En Progreso:
Esta lista contendrá todas las tarjetas que actualmente están siendo trabajadas por los miembros del equipo.
Ejemplo de tarjeta: "Desarrollando función para cambiar contraseña en el backend"

Lista de Revisión o Lista de Pendiente de Revisión:
Aquí se colocan las tarjetas que están listas para ser revisadas por otros miembros del equipo o por el líder del proyecto.
Ejemplo de tarjeta: "Función de cambio de contraseña lista para revisión"

Lista de Terminadas o Lista de Hecho:
Esta lista contiene las tarjetas que han sido completadas y están listas para su implementación o despliegue.
Ejemplo de tarjeta: "Implementación de la función de cambio de contraseña completa"

Lista de Implementadas o Lista de Desplegadas:
En esta lista se colocan las tarjetas que ya han sido implementadas en el sistema y están listas para ser probadas por los usuarios finales.
Ejemplo de tarjeta: "Función de cambio de contraseña desplegada en producción"

Para dividir el trabajo de manera eficiente, se podrían asignar diferentes tarjetas a los miembros del equipo según sus habilidades y disponibilidad.
Por ejemplo, un desarrollador podría encargarse de implementar la funcionalidad en el backend, mientras que otro podría trabajar en la interfaz de usuario. Además, puedes asignar etiquetas a las tarjetas para indicar la prioridad, el tipo de tarea (como frontend, backend, diseño, etc.), y cualquier otra información relevante.