Nombre: **Eduardo Schiller Ruiz**		**Matrícula:** **AL02921126**
Nombre del curso: Diseño de aplicaciones web	**Nombre del profesor:** Francisco Javier Olivera Guerrero	
Módulo: Módulo 2	**Actividad:** Evidencia 2 / Avance proyecto con React	
Fecha: 29 de Marzo de 2025		

Requerimientos para el funcionamiento de la aplicacion:

django 5.1.7

Python 3.13.2

asgiref 3.8.1

psycop 2.9.10

djangorestframework 3.15.2

drf-yasg 1.21.10

django cors headers 4.7.0

reactc 19.0.0

Vite 6.2.3

react-hook-form 7.55.0

react-hot-toast 2.5.2

react-router-dom 7.4.1

axios 1.8.4

NPM 11.2.0

Tailwind 4.0.17

Para ejecutar servidor BackEnd
\proyecto_inv_v35\proyecto_inv\proyecto_inv\backend> python manage.py runserver

Para ejecutar servidor FrontEnd
\proyecto_inv_v35\proyecto_inv\proyecto_inv\frontend\my_react_app> npm run dev

Se creo una base datos con las tablas necesarias para la aplicacion

Usuario: Almacena información de cuentas de usuario incluyendo credenciales de inicio de sesión y estado.

Linea: Representa líneas de investigación o áreas de enfoque, con información descriptiva.

Unidad: Representa unidades organizativas o departamentos, incluyendo su ubicación física.

Area: Subdivisiones dentro de unidades, estableciendo la jerarquía organizacional.

TipoEstudiante: Categoriza a los estudiantes por su tipo (ej. licenciatura, posgrado).

Carrera: Almacena programas académicos o trayectorias profesionales que los estudiantes pueden cursar.

Especialidad: Registra áreas de especialización dentro de disciplinas académicas.

NivelEdu: Niveles educativos vinculados a especialidades, representando cualificaciones académicas.

NivelSNII: Almacena niveles del Sistema Nacional de Investigadores (SNI) para investigadores en México.

Articulo: Artículos o publicaciones científicas con detalles de publicación y metadatos.

Evento: Eventos académicos como conferencias o talleres, con fechas y categorización.

TipoEvento: Categorías para diferentes tipos de eventos académicos.

Herramienta: Herramientas, metodologías o recursos utilizados en proyectos de investigación.

Proyecto: Proyectos de investigación con sus cronogramas e información de estado.

Investigador: Investigadores con su información personal, credenciales y relaciones con áreas, niveles educativos y clasificación SNI.

Estudiante: Estudiantes asociados con investigadores, carreras y tipos de estudiante.

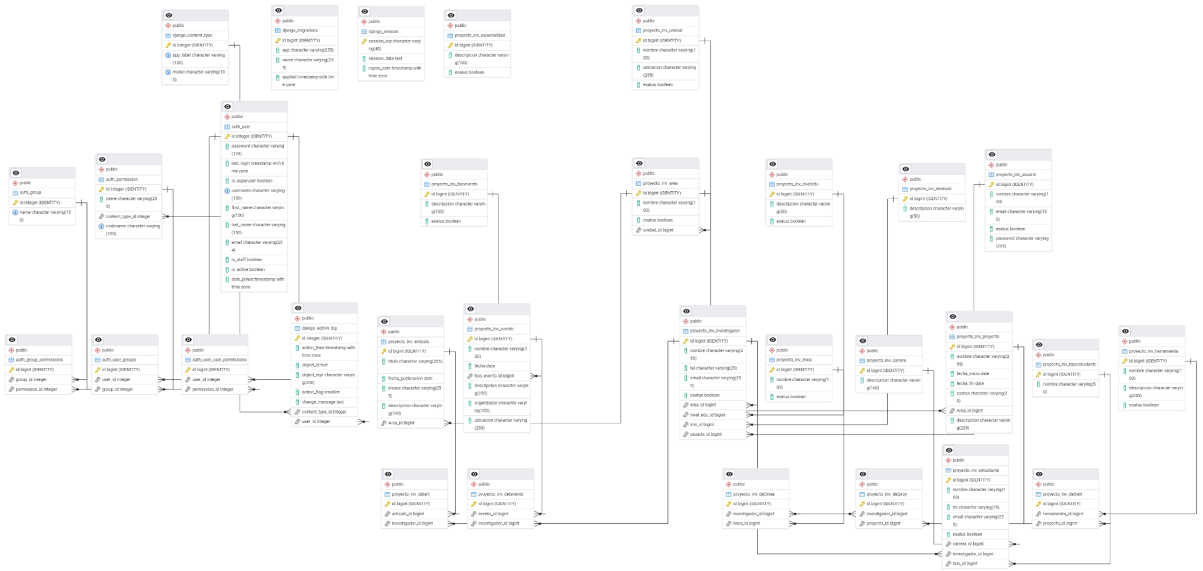
DetLinea: Tabla de unión que conecta investigadores con sus líneas de investigación.

DetArt: Tabla de unión que asocia investigadores con sus artículos publicados.

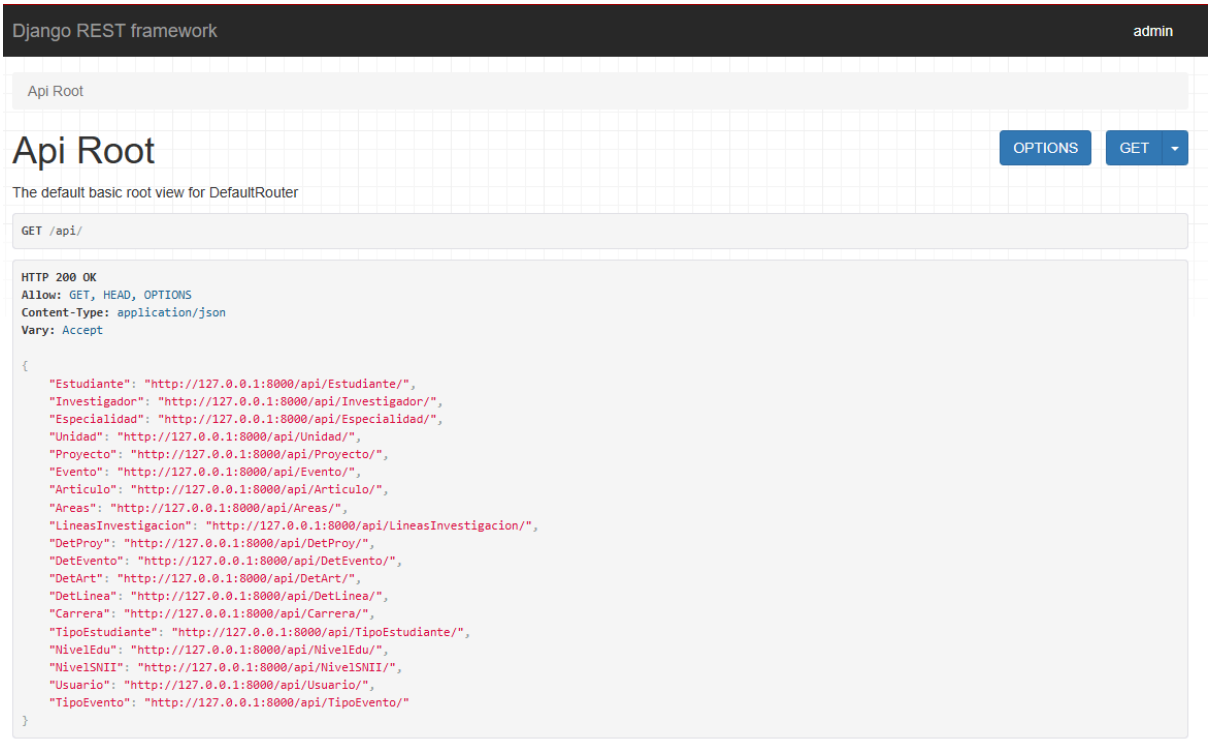
DetEvento: Tabla de unión que vincula investigadores con eventos en los que participan.

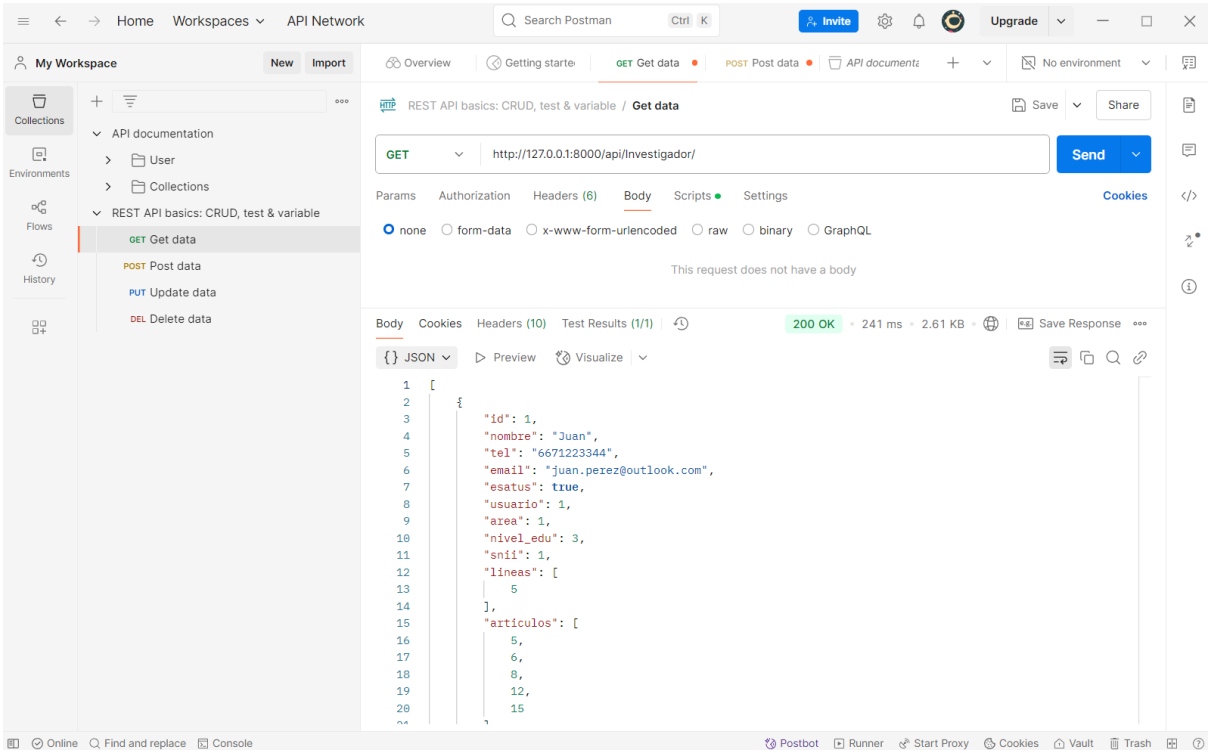
DetProy: Tabla de unión que conecta investigadores con sus proyectos.

DetHerr: Tabla de unión que asocia proyectos con las herramientas que utilizan.



Se creó una api con *serializers* para cada *modelo* en el archivo serielizers.py





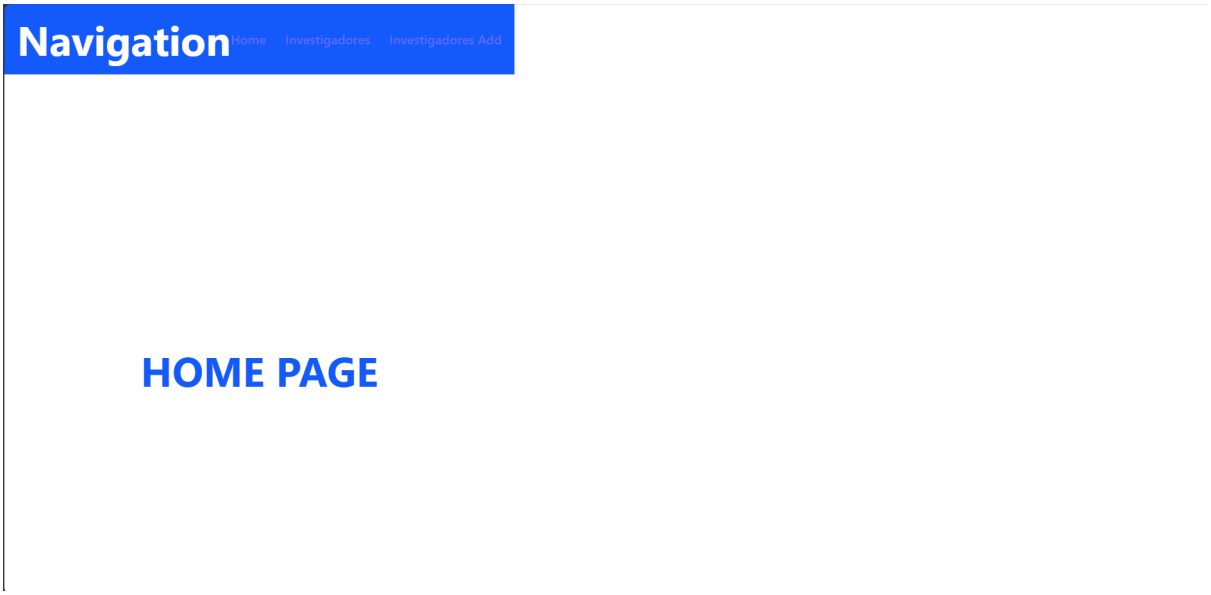
Se crearon los “Endpoints” para acceder a los datos de la API con react

```python
#ENDPOINTS FAK

def get_usuarios(request):

    usuarios = Usuario.objects.all().values( 'nombre', 'email')

    return JsonResponse(list(usuarios), safe=False)



def get_areas(request):

    areas = Area.objects.all().values('nombre')

    return JsonResponse(list(areas), safe=False)



def get_niveledu(request):

    niveledu = NivelEdu.objects.all().values('descripcion')

    return JsonResponse(list(niveledu), safe=False)



def get_nivelsnii(request):

    nivelsnii = NivelSNII.objects.all().values('descripcion')

    return JsonResponse(list(nivelsnii), safe=False)



@api_view(['POST'])

def create_investigador(request):

    serializer = InvestigadorSerializer(data=request.data)

    if serializer.is_valid():

        serializer.save()

        return Response(serializer.data, status=201)
```

```
    return Response(serializer.errors, status=400)


@api_view(['DELETE'])
def delete_investigador(request, id):
    try:
        investigador = Investigador.objects.get(id=id)
        investigador.delete()
        return Response({"message": "Investigador eliminado
correctamente"}, status=200)
    except Investigador.DoesNotExist:
        return Response({"error": "Investigador no encontrado"},
status=404)
    except Exception as e:
        return Response({"error": str(e)}, status=500)


@api_view(['GET', 'PUT'])
def investigador_detail(request, id):
    try:
        investigador = Investigador.objects.get(id=id)
    except Investigador.DoesNotExist:
        return Response({"error": "Investigador no encontrado"},
status=404)


    if request.method == 'GET':
        serializer = InvestigadorSerializer(investigador)
        return Response(serializer.data)
```

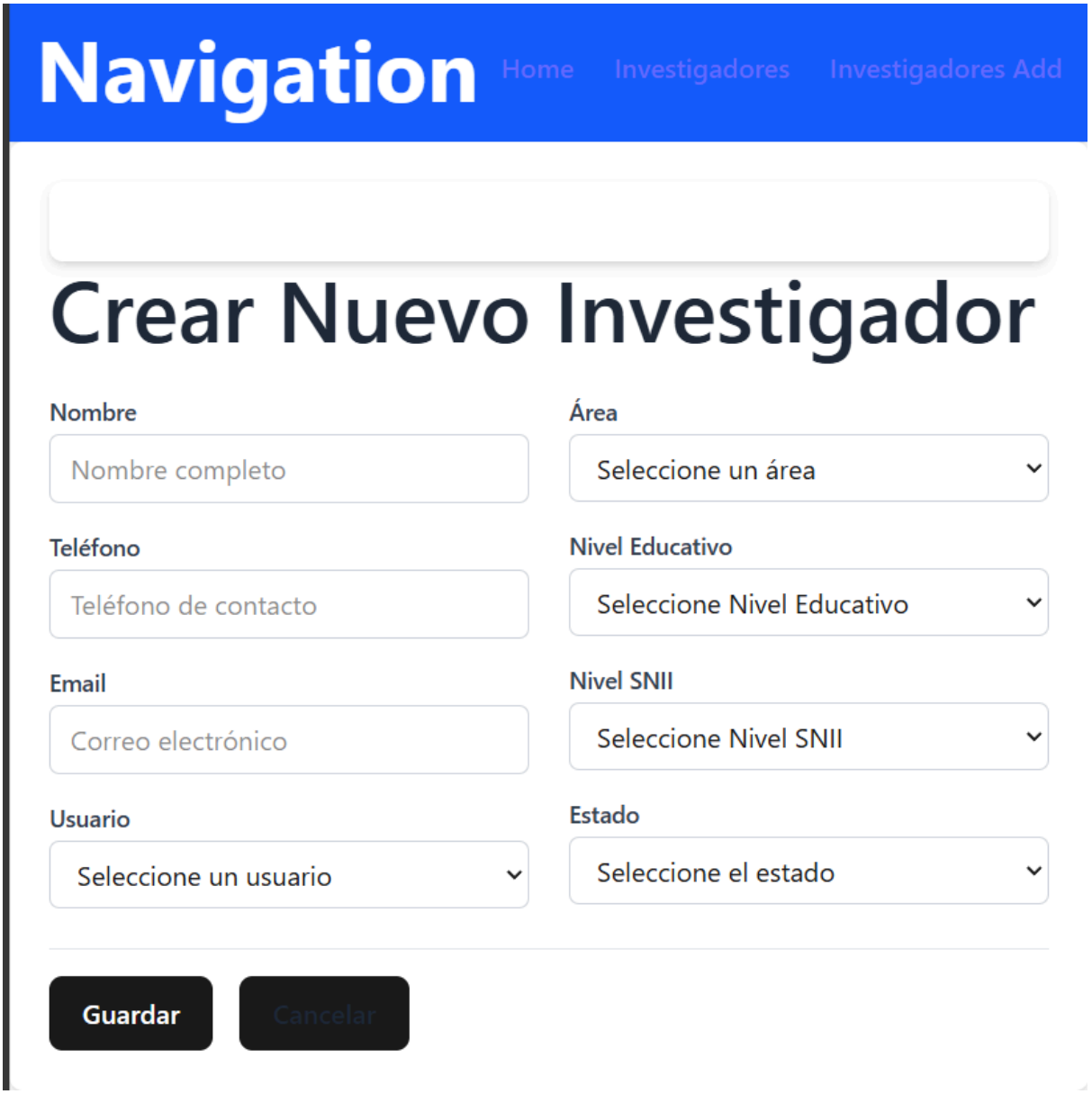
```
    if request.method == 'PUT':
        serializer = InvestigadorSerializer(investigador,
data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data)
        return Response(serializer.errors, status=400)
```

Página de home con menú de navegación a la pagina de investigadores



Página de investigadores con el id del área del investigador

Pagina para crear un investigador nuevo



Pagina para eliminar un investigador ya existente

Navigation Home Investigadores Investigadores Add

Editar Investigador

Nombre

Nombre completo

Área

Seleccione un área

Teléfono

Teléfono de contacto

Nivel Educativo

Seleccione Nivel Educativo

Email

Correo electrónico

Nivel SNII

Seleccione Nivel SNII

Usuario

Seleccione un usuario

Estado

Seleccione el estado

Actualizar Cancelar Eliminar

Información solicitada para tramite de nivel SN

RFC

CORREO

TELÉFONO

ÁREA

CAMPO

DISCIPLINA

SUBDISCIPLINA

DOMICILIO

INSTITUCION ACTUAL

NIVEL AL QUE ASPIRA

PERFIL PARA EVALUACION

COMROBANTE DE GRADO

COMPROBANTE DE NACIONALIDAD