CoderDojo website

Programación Web 2 - Proyecto Final

July 25, 2024

Abstract

Este proyecto está destinado a ser usado por los estudiantes del evento CoderDojo, para que puedan encontrar los recursos de aprendizaje y hacer el envío de sus tareas.

Integrantes:

- name: Arias Quispe Jhonatan David
- name: Chambilla Perca Ricardo Mauricio
- name: Carbajal Gonzales Diego Alejandro

URL a los repositorios:

- Frontend: https://github.com/JhonatanDczel/coder-dojo-front
- Backend: https://github.com/rikich3/coderDojoBack

Coder Dojo y la IEEE CS Unsa

Aplicación para el evento de Coder Dojo que está organizando la ***IEEE Computational Society rama Perú*** por parte de la ***UNSA***.

Requerimientos

Las especificaciones que recibimos de los coordinadores fueron:

1. **Tipos de Usuario**:
    - **Profesor**: Puede crear salones virtuales y asignar tareas a los estudiantes.
    - **Estudiante**: Puede unirse a los salones y acceder a las tareas asignadas.
    - **Administrador**: Gestiona la plataforma.
2. **Salones Virtuales**:
    - Los profesores pueden crear salones y asignar tareas a los estudiantes dentro de esos salones.
    - Los salones contienen a estudiantes y un profesor.
    - Los profesores pueden publicar material de estudio y otros recursos que los estudiantes pueden ver.
3. **Usuarios Objetivo**:

- Estudiantes de secundaria.
- La plataforma debe incluir elementos de gamificación para aumentar la atención y el compromiso.
- Utiliza un framework moderno desarrollado por Octolasys Group para mejorar la experiencia del usuario.

Planificación

Una vez tecibidas las especificaciones, procedimos a crear los mockups en una herramienta de diseño: `excalidraw`.

REQUERIMIENTOS:

Estudiante

Cursos

- Vista de cursos generales
- Vista de un curso, con todos los post/tareas/recursos de este

Tareas

- Vista de tareas pendientes y deadlines
- Vista en detalle de la tarea
- Entrega de la tarea solo un link

Profesor

Cursos

- Vista de cursos generales
- Vista de un curso, con todos los post/tareas/recursos de este y posibilidad de publicar algo

Tareas

- Vista de tareas por curso
- Vista en detalle de la tarea Quienes entregaron y quienes no
- Creacion de tareas (a modo de post)

Interfaz

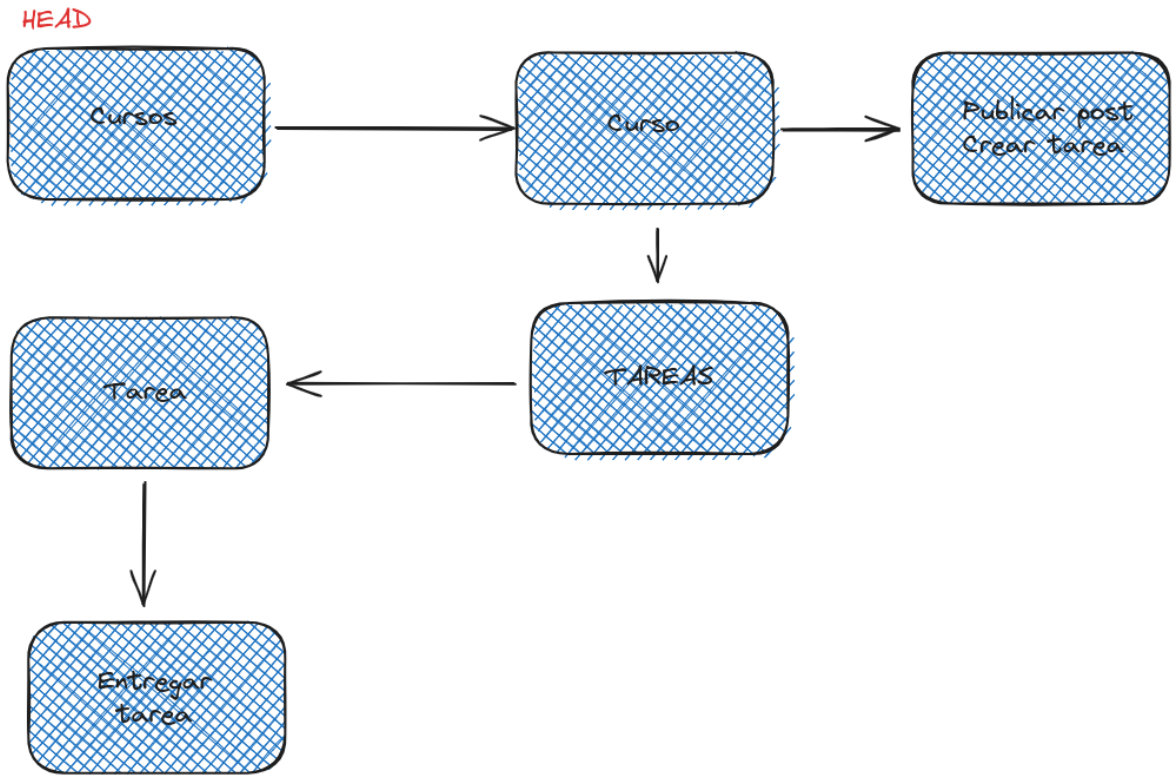
Sidebar

- Menu vertical para ir a cursos, tareas, foro
- Menu de informacion, para dar una guia al estudiante

Foro

- Vista general de los post
- Redactar un post
- Comentar / reaccionar a un post

NAVEGACIÓN DE PÁGINAS



Desarrollo

Una vez entendida la dinámica de la interacción entre las páginas, se procede con el desarrollo, que está dividido en dos partes: `frontend` y `backend`.

Stack de desarrollo

Las tecnologías que se eligieron para el proyecto, fueron:

- **Django**: Para el lado del backend.
- **Django REST Framework**: Usado para hacer las API que consumira el frontend.
- **React**: Para crear las vistas y aprovechar el aspecto reactivo de react.
- **Tailwind**: Para manejar los estilos de una mejor manera a css puro.

Frontend

Para el desarrollo del lado del frontend, se dividió en las siguientes tasks:

- **Login** - que debería tener la autenticación de los usuarios, así como el minigame `DojoType`.
- **Dashboard** - que debería tener el acceso a las vistas como los cursos, vista por curso, y las tareas que se tienen.

Login

El login se hizo en `react` con un diseño basado en componentes, se enfatizó el concepto de `gamificación` de la interfaz para hacerla más amigable para nuestro público objetivo (estudiantes de secundaria).

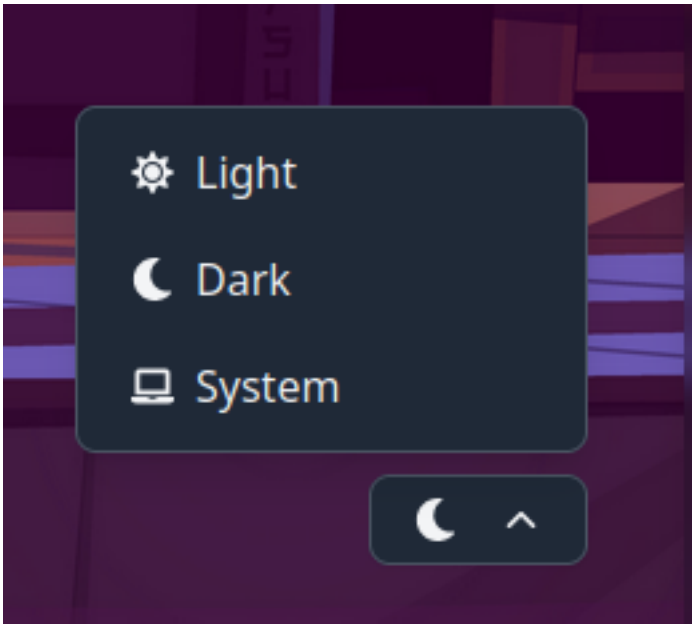
Dark - Light - System modes

Con ese objetivo en mente, se optó por una interfaz sencilla, clara y concisa, que destaque los elementos importantes y que tenga un toque fresco, esta es una vista de la interfaz:

Modo dark

Modo light

Para poder hace rel cambio entre modos dark, light o system se usó el siguiente componente:



que es un componente de react:

```
// src/components/common/ThemeSwitcher.js
import { useState, useEffect } from "react";
import { FaSun, FaMoon, FaLaptop } from "react-icons/fa";
```

```
import { IoIosArrowDown } from "react-icons/io";

const ThemeSwitcher = () => {
  const [theme, setTheme] = useState("system");
  const [isOpen, setIsOpen] = useState(false);

  useEffect(() => {
    const root = window.document.documentElement;
    if (theme === "dark") {
      root.classList.add("dark");
      localStorage.setItem("theme", "dark");
    } else if (theme === "light") {
      root.classList.remove("dark");
      localStorage.setItem("theme", "light");
    } else {
      root.classList.remove("dark");
      if (window.matchMedia("(prefers-color-scheme: dark)").matches) {
        root.classList.add("dark");
      }
      localStorage.removeItem("theme");
    }
  }, [theme]);

  const handleThemeChange = (newTheme) => {
    setTheme(newTheme);
    setIsOpen(false); // Close the dropdown after selection
  };

  return (
    <div className="relative inline-block text-left">
      <div>
        <button
          type="button"
          className="inline-flex items-center px-4 py-2 border rounded-lg bg-g
          onClick={() => setIsOpen(!isOpen)}
        >
          <span className="mr-2">
            {theme === "dark" ? (
              <FaMoon />
            ) : theme === "light" ? (
              <FaSun />
            ) : (
              <FaLaptop />
            )}
          </span>
          <IoIosArrowDown
            className={`ml-2 transition-transform ${
              isOpen ? "rotate-180" : ""
```

```
            }`}
          />
        </button>
      </div>
      {isOpen && (
        <div className="absolute bottom-full right-0 z-10 mb-2 w-48 bg-white
          <div className="p-1">
            <button
              type="button"
              className="flex items-center px-4 py-2 w-full text-gray-900 dark
              onClick={() => handleThemeChange("light")}
            >
              <FaSun className="mr-2" /> Light
            </button>
            <button
              type="button"
              className="flex items-center px-4 py-2 w-full text-gray-900 dark
              onClick={() => handleThemeChange("dark")}
            >
              <FaMoon className="mr-2" /> Dark
            </button>
            <button
              type="button"
              className="flex items-center px-4 py-2 w-full text-gray-900 dark
              onClick={() => handleThemeChange("system")}
            >
              <FaLaptop className="mr-2" /> System
            </button>
          </div>
        </div>
      )}
    </div>
  );
};

export default ThemeSwitcher;
```

Composición de la interfaz

La interfaz esta conmpuesta siguiendo reglas de código limpio y buenas prácticas de programación, a continuación tenemos el codigo del componente `HomePage` que se encarga del renderizado de la página principal:

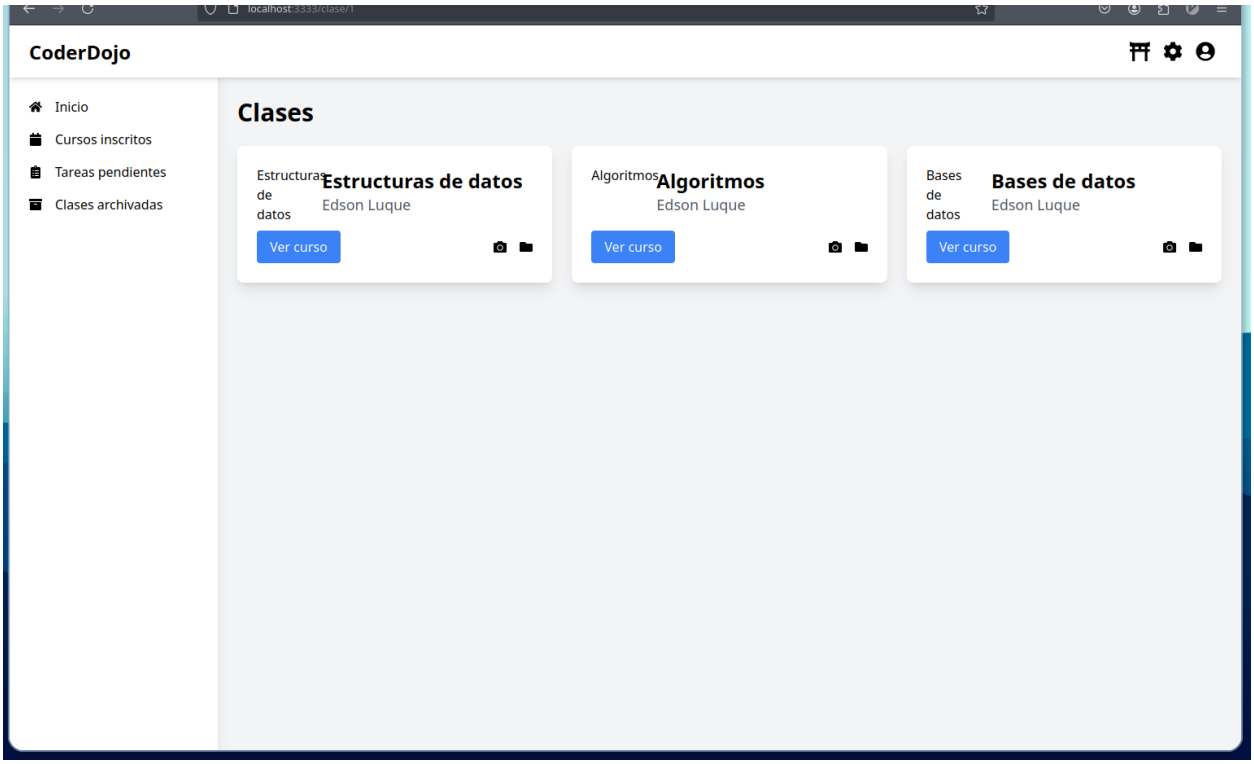
```
//src/pages/HomePage.jsx
        <div className="relative flex flex-col items-center justify-center mi
          <div className="absolute inset-0 flex items-center justify-center">
            <div className="relative bg-white bg-opacity-70 backdrop-blur-md
              <h1 className="text-4xl font-extrabold text-gray-900 mb-4">
```

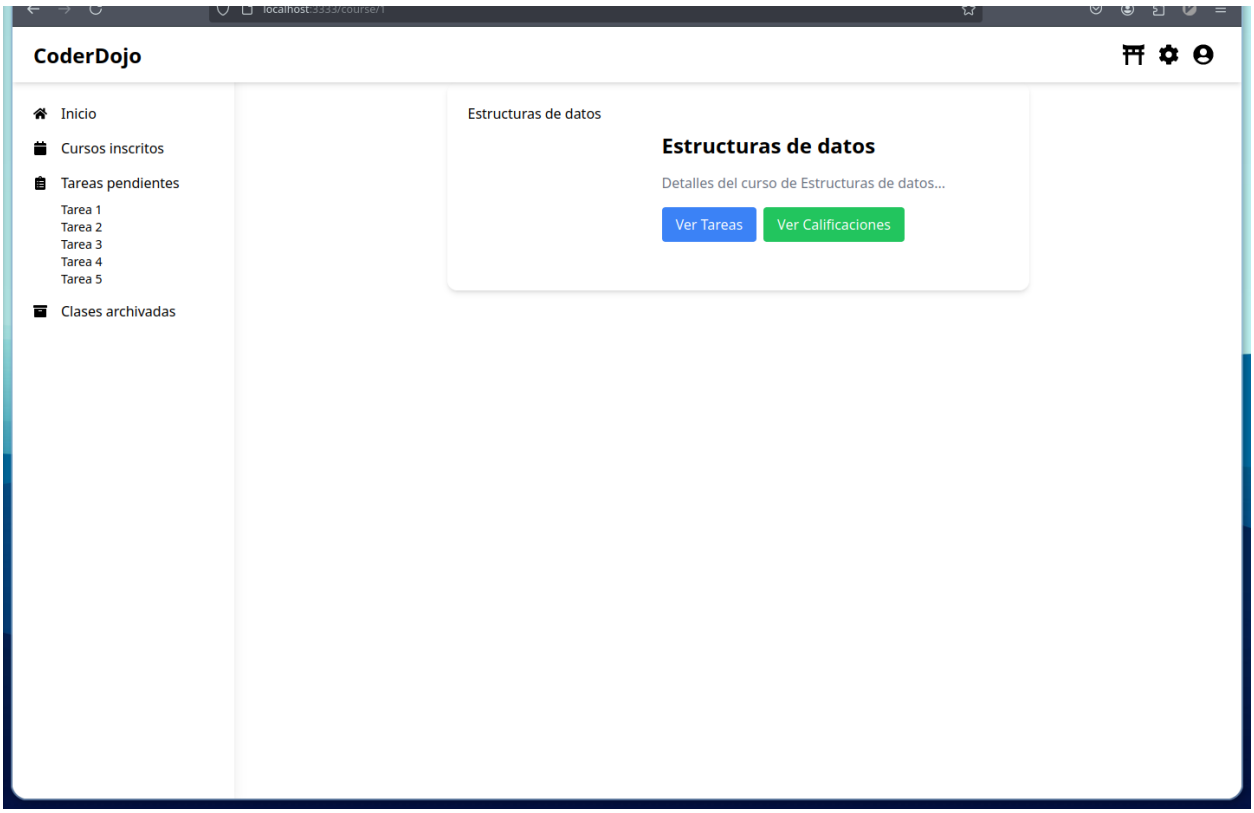
```
            ¡Bienvenido a CoderDojo!
          </h1>
          <p className="text-lg text-gray-700">
            Estamos emocionados de tenerte con nosotros. Prepárate para
            aprender y crecer en el mundo de la programación. ¡Vamos a hac
            grandes cosas juntos!
          </p>
        </div>
      </div>
      <div className="absolute top-5 left-5 p-4">
        <Logo path={IEEELogo} size="h-28"/>
      </div>
      <a href="https://coderdojo.com/en/" target="_blank" className="inset
        <div className="absolute top-5 right-5 p-4">
          <Logo path={coderDojoLogo} size={"h-[4.5rem]"}/>
        </div>
      </a>
      <div className="absolute bottom-5 left-5 p-4">
        <DojoTypeButton onClick={handleDojoTypeButtonClick} />
      </div>
      <div className="absolute bottom-5 right-5 p-4">
        <ThemeSwitcher />
      </div>
    </div>
  </div>

  <LoginForm />
```

Vistas de los cursos:

Las vistas de cursos y de cada curso individual una vez que el usuario se autentifica son las siguientes:





Composicion:

La composición para estas vistas fué la siguiente:

```
return (
  <div className="flex">
    <Sidebar />
    <div className="ml-64 w-full">
      <Navbar data={data} />
      <div className="pt-16">
        <StudentRoutes />
      </div>
      <CoursesList />
    </div>
  </div>
);
```

Backend

Modelos

Una vez entendidas las relaciones que deben tener los modelos, se procede a programarlos y migrarlos.

Modelo para los Usuarios:

```
class AppUserManager(BaseUserManager):
    def create_user(self, email, password=None):
        if not email:
            raise ValueError('An email is required.')
        if not password:
            raise ValueError('A password is required.')
        email = self.normalize_email(email)
        user = self.model(email=email)
        user.set_password(password)
        user.save()
        return user

    def create_superuser(self, email, password=None, **extra_fields):
        if not email:
            raise ValueError('An email is required.')
        if not password:
            raise ValueError('A password is required.')
        user = self.create_user(email, password, **extra_fields)
        user.is_superuser = True
        user.is_staff = True
        user.save()
        return user

class AppUser(AbstractBaseUser, PermissionsMixin):
    user_id = models.AutoField(primary_key=True)
    email = models.EmailField(max_length=50, unique=True)
    username = models.CharField(max_length=50)
    is_staff = models.BooleanField(default=False)
```

```
    is_student = models.BooleanField(default=False)
    is_teacher = models.BooleanField(default=False)

    USERNAME_FIELD = 'email'
    REQUIRED_FIELDS = ['username']
    objects = AppUserManager()

    def __str__(self):
        return self.username
```

Modelo para los Salones:

```
from django.db import models
from django.contrib.auth.models import AbstractUser

class User(AbstractUser):
    is_estudiante = models.BooleanField(default=False)
    is_docente = models.BooleanField(default=False)

class Estudiante(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    name = models.CharField(max_length=100)

class Docente(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    name = models.CharField(max_length=100)

class Clase(models.Model):
    name = models.CharField(max_length=100)
    estudiantes = models.ManyToManyField(Estudiante, related_name='clases')
    docente = models.ForeignKey(Docente, on_delete=models.CASCADE, related_nar

class Publicacion(models.Model):
    content = models.TextField()
    clase = models.ForeignKey(Clase, on_delete=models.CASCADE, related_name='p

class Asignacion(models.Model):
    title = models.CharField(max_length=100)
    description = models.TextField()
    due_date = models.DateTimeField()
    clase = models.ForeignKey(Clase, on_delete=models.CASCADE, related_name='a

class Entrega(models.Model):
    asignacion = models.ForeignKey(Asignacion, on_delete=models.CASCADE, relat
    estudiante = models.ForeignKey(Estudiante, on_delete=models.CASCADE, relat
    file = models.FileField(upload_to='entregas/')
    submitted_at = models.DateTimeField(auto_now_add=True)
```

Vistas y Serializadores

Vistas:

```
from rest_framework import generics, permissions
from rest_framework.views import APIView
from rest_framework.response import Response
from rest_framework.permissions import IsAuthenticated
from django.shortcuts import get_object_or_404
from .models import Clase, Publicacion, Asignacion, Entrega, Estudiante, Doce
from .serializers import ClaseSerializer, PublicacionSerializer, AsignacionSe

class LoginView(APIView):
    permission_classes = (permissions.AllowAny,)

    def post(self, request):
        username = request.data.get('username')
        password = request.data.get('password')
        user = authenticate(request, username=username, password=password)
        if user is not None:
            login(request, user)
            return HttpResponseRedirect('/clases')
        return Response({'error': 'Invalid Credentials'}, status=status.HTTP_

class LogoutView(APIView):
    def post(self, request):
        logout(request)
        return HttpResponseRedirect('/')

class ClaseListView(generics.ListCreateAPIView):
    serializer_class = ClaseSerializer
    permission_classes = [IsAuthenticated]

    def get_queryset(self):
        user = self.request.user
        if user.is_estudiante:
            estudiante = get_object_or_404(Estudiante, user=user)
            return estudiante.clases.all()
        elif user.is_docente:
            docente = get_object_or_404(Docente, user=user)
            return docente.clases.all()
        return Clase.objects.none()

    def perform_create(self, serializer):
        user = self.request.user
        if user.is_docente:
            docente = get_object_or_404(Docente, user=user)
            serializer.save(docente=docente)
```

```
class PublicacionListView(generics.ListCreateAPIView):
    serializer_class = PublicacionSerializer
    permission_classes = [IsAuthenticated]

    def get_queryset(self):
        user = self.request.user
        if user.is_estudiante:
            estudiante = get_object_or_404(Estudiante, user=user)
            return Publicacion.objects.filter(clase__in=estudiante.clases.all
        elif user.is_docente:
            docente = get_object_or_404(Docente, user=user)
            return Publicacion.objects.filter(clase__docente=docente)
        return Publicacion.objects.none()

class AsignacionListView(generics.ListCreateAPIView):
    serializer_class = AsignacionSerializer
    permission_classes = [IsAuthenticated]

    def get_queryset(self):
        user = self.request.user
        if user.is_estudiante:
            estudiante = get_object_or_404(Estudiante, user=user)
            return Asignacion.objects.filter(clase__in=estudiante.clases.all(
        elif user.is_docente:
            docente = get_object_or_404(Docente, user=user)
            return Asignacion.objects.filter(clase__docente=docente)
        return Asignacion.objects.none()

class ClaseDetailView(generics.RetrieveUpdateDestroyAPIView):
    queryset = Clase.objects.all()
    serializer_class = ClaseSerializer
```

Serializadores:

```
from rest_framework import serializers
from .models import Clase, Publicacion, Asignacion, Estudiante, Docente

class PublicacionSerializer(serializers.ModelSerializer):
    class Meta:
        model = Publicacion
        fields = '__all__'

class AsignacionSerializer(serializers.ModelSerializer):
    class Meta:
        model = Asignacion
        fields = '__all__'

class ClaseSerializer(serializers.ModelSerializer):
```

```
    publicaciones = PublicacionSerializer(many=True, required=False)
    asignaciones = AsignacionSerializer(many=True, required=False)
    estudiantes = serializers.ListField(child=serializers.IntegerField(), requ

    class Meta:
        model = Clase
        fields = ['id', 'nombre', 'docente', 'publicaciones', 'asignaciones',

    def create(self, validated_data):
        publicaciones_data = validated_data.pop('publicaciones', [])
        asignaciones_data = validated_data.pop('asignaciones', [])
        estudiantes_data = validated_data.pop('estudiantes', [])

        clase = Clase.objects.create(**validated_data)

        for publicacion_data in publicaciones_data:
            Publicacion.objects.create(clase=clase, **publicacion_data)

        for asignacion_data in asignaciones_data:
            Asignacion.objects.create(clase=clase, **asignacion_data)

        for estudiante_id in estudiantes_data:
            estudiante = Estudiante.objects.get(id=estudiante_id)
            estudiante.clases.add(clase)

        return clase
```

URLs:

```
from django.urls import path
from .views import LoginView, LogoutView, ClaseListView, ClaseDetailView, Pub

urlpatterns = [
    path('login/', LoginView.as_view(), name='login'),
    path('logout/', LogoutView.as_view(), name='logout'),
    path('clases/', ClaseListView.as_view(), name='clase-list'),
    path('clases/<int:pk>/', ClaseDetailView.as_view(), name='clase-detail'),
    path('publicaciones/', PublicacionListView.as_view(), name='publicacion-li
    path('asignaciones/', AsignacionListView.as_view(), name='asignacion-list
]
```

Permisos:

```
from rest_framework import permissions

class IsDocente(permissions.BasePermission):
    def has_permission(self, request, view):
        return request.user and request.user.is_authenticated and request.use
```