

Informe del Trabajo Final

Tema: Coder Dojo

Nota

Estudiante(s)	Escuela	Asignatura
Mariel Alisson Jara Mamani mjarama@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Programación Web 2 Semestre: I Código: 1702122

Laboratorio	Tema	Duración
Final	Coder Dojo	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 B	Del 19 Julio 2024	Al 25 Julio 2024

Trabajo Final

1 Coder Dojo

Coder Dojo es una iniciativa global sin fines de lucro que enseña programación y habilidades tecnológicas a niños y jóvenes, fomentando un ambiente divertido y colaborativo. Los participantes pueden aprender a desarrollar sitios web, aplicaciones, juegos y más, con la guía de mentores voluntarios.















































Es por eso que, aquí en Arequipa, el comité de la IEEE de la UNSA ha decidido implementar este proyecto de forma real para Arequipa y Tacna, brindando a los jóvenes de estas regiones la oportunidad de explorar el fascinante universo de la codificación.

1.1 Descripción del proyecto

Para la realización de este proyecto se ha dividido en dos partes: Backend y Frontend. Por el lado del Backend se ha utilizado Django Rest Framework. Por otro lado, en el Frontend se ha utilizado React JS. Para el diseño se hizo uso de la herramienta de diseño Figma. Se ha creado una API REST para la comunicación con el Frontend. A continuación, se muestra la estructura de la aplicación: A continuación, se ve la estructura del proyecto como tal:

- **Implementación:** En Coder Dojo existen tres tipos de roles: administrador, profesor y estudiante.
 - **Administrador:**
 - * Puede ver, editar y eliminar la lista de usuarios.
 - * Puede crear cursos y asignarles un profesor.
 - * Puede ver la lista de cursos.
 - **Profesor:**
 - * Puede ver los cursos que enseña.
 - * Puede crear y asignar tareas para cada curso.
 - * Puede ver la lista de tareas y calificarlas.
 - **Estudiante:**
 - * Puede ver los cursos en los que está inscrito.
 - * Puede ver las tareas asignadas.
 - * Puede ver sus calificaciones.

2 Commits

assign and submit task Alanj20 committed 4 hours ago	0b27539		
fixing the grade and assignment style Alanj20 committed 6 hours ago	9209b1e		
create the student app Alanj20 committed 9 hours ago	ce8a821		
add views in the teacher app Alanj20 committed 9 hours ago	27343e1		
create the teacher app Alanj20 committed 9 hours ago	1e64bdc		
create the admin app Alanj20 committed 10 hours ago	e21db8a		
Merge pull request #8 from Alanj20/designBoard Alanj20 committed 11 hours ago	8c7e7fb		
Assignments and deliveries Alanj20 committed 11 hours ago	2fe7552		
Commits on Jul 24, 2024			
mizy courses and task assign Alanj20 committed 2 days ago	3b56bd4		
assign task Alanj20 committed 2 days ago	327f304		
Commits on Jul 23, 2024			
tasks list Alanj20 committed 2 days ago	b2b5b07		
create Task Alanj20 committed 2 days ago	555c083		
register login Alanj20 committed 2 days ago	53f09e7		
edit option course Alanj20 committed 2 days ago	5e5902f		
remove course Alanj20 committed 2 days ago	97e9ba5		
update color Form Alanj20 committed 2 days ago	40b4a88		
update Color Alanj20 committed 2 days ago	63cd271		
HomeAboutUs Alanj20 committed 2 days ago	dca03c4		
home y footer Alanj20 committed 2 days ago	dbbfc91		
change menu responsive Alanj20 committed 2 days ago	a5fd1af		
update Alanj20 committed 3 days ago	c1ad7fd		
Commits on Jul 22, 2024			
Merge pull request #7 from Alanj20/studentConfig Alanj20 committed 3 days ago	73c25f7		
changes Alanj20 committed 3 days ago	22eb48e		

Lista de commits realizados en el proyecto.

3 Equipos y materiales utilizados

- Cuenta en GitHub con el correo institucional.
- Sistema Operativo Microsoft Windows 11
- Visual Studio Code
- Git
- GitHub
- Windows PowerShell
- Navegador Google Chrome, Microsoft Edge
- Django Rest Framework
- React JS
- Figma

4 Backend

Para el lado del Backend(Servidor) se ha utilizado Django Rest Framework, es un framework de alto nivel para el desarrollo de aplicaciones web en Python. Este trabajo se ha dividido en 4 aplicaciones(System, Admin, Teacher, Student) para una mejor organización del proyecto.

4.1 Conexión con el Frontend

Para la comunicación entre el Frontend y el Backend se ha utilizado Axios, una librería de JavaScript que se encarga de realizar peticiones HTTP. Se ha utilizado CORS para permitir la comunicación entre el Frontend y el Backend.

4.2 Flujo de Trabajo

El flujo de trabajo de la aplicación es el siguiente:

1. El usuario enviara una solicitud HTTP al servidor.
2. La solicitud se enrutara a urls.py y se enviara a la vista correspondiente.
3. La vista procesara la solicitud y devolvera una respuesta (Get o Post).
4. La respuesta se enviara al Frontend a través de los Serializers.

A continuación, se muestra la estructura de la aplicación:

4.3 Configuración

Este archivo contiene la configuración principal del proyecto Django. Aquí se definen los ajustes clave del proyecto, incluyendo las configuraciones para la base de datos, seguridad, aplicaciones instaladas, y más. Instalamos CORS, Django Rest Framework y JWT para la autenticación de los usuarios.

```
1  """
2  Django settings for coderdojo project.
3
4  Generated by 'django-admin startproject' using Django 5.0.6.
5
6  For more information on this file, see
7  https://docs.djangoproject.com/en/5.0/topics/settings/
8
9  For the full list of settings and their values, see
10 https://docs.djangoproject.com/en/5.0/ref/settings/
11 """
12
13 from pathlib import Path
14
15 # Build paths inside the project like this: BASE_DIR / 'subdir'.
16 BASE_DIR = Path(__file__).resolve().parent.parent
17
18
19 # Quick-start development settings - unsuitable for production
20 # See https://docs.djangoproject.com/en/5.0/howto/deployment/checklist/
21
22 # SECURITY WARNING: keep the secret key used in production secret!
23 SECRET_KEY = 'django-insecure-jsjipc83tn+!f-8!64ipz&1*f)kelikt26wj$h%uqdy682^7e5'
24
25 # SECURITY WARNING: don't run with debug turned on in production!
26 DEBUG = True
27
```

```
28 ALLOWED_HOSTS = []
29
30
31 # Application definition
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     'rest_framework',
41     'rest_framework_simplejwt',
42     'corsheaders',
43     'system',
44 ]
45
46 MIDDLEWARE = [
47     'django.middleware.security.SecurityMiddleware',
48     'django.contrib.sessions.middleware.SessionMiddleware',
49     'django.middleware.common.CommonMiddleware',
50     'django.middleware.csrf.CsrfViewMiddleware',
51     'django.contrib.auth.middleware.AuthenticationMiddleware',
52     'django.contrib.messages.middleware.MessageMiddleware',
53     'django.middleware.clickjacking.XFrameOptionsMiddleware',
54     'corsheaders.middleware.CorsMiddleware',
55 ]
56
57 # CORS
58 CORS_ORIGIN_ALLOW_ALL = True
59 CORS_ALLOW_CREDENTIALS = True
60 CORS_ALLOWED_ORIGINS = [
61     "http://localhost:5173",
62 ]
63
64 REST_FRAMEWORK = {
65     'DEFAULT_AUTHENTICATION_CLASSES': [
66         'rest_framework_simplejwt.authentication.JWTAuthentication',
67     ],
68     'DEFAULT_PERMISSION_CLASSES': [
69         'rest_framework.permissions.IsAuthenticated',
70     ],
71 }
72
73 ROOT_URLCONF = 'coderdojo.urls'
74
75 TEMPLATES = [
76     {
77         'BACKEND': 'django.template.backends.django.DjangoTemplates',
78         'DIRS': [],
79         'APP_DIRS': True,
80         'OPTIONS': {
81             'context_processors': [
82                 'django.template.context_processors.debug',
83                 'django.template.context_processors.request',
```

```
84         'django.contrib.auth.context_processors.auth',
85         'django.contrib.messages.context_processors.messages',
86     ],
87 },
88 },
89 ]
90
91 WSGI_APPLICATION = 'coderdojo.wsgi.application'
92
93
94 # Database
95 # https://docs.djangoproject.com/en/5.0/ref/settings/#databases
96
97 DATABASES = {
98     'default': {
99         'ENGINE': 'django.db.backends.sqlite3',
100         'NAME': BASE_DIR / 'db.sqlite3',
101     }
102 }
103
104
105 # Password validation
106 # https://docs.djangoproject.com/en/5.0/ref/settings/#auth-password-validators
107
108 AUTH_PASSWORD_VALIDATORS = [
109     {
110         'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
111     },
112     {
113         'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
114     },
115     {
116         'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
117     },
118     {
119         'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
120     },
121 ]
122
123
124 # Internationalization
125 # https://docs.djangoproject.com/en/5.0/topics/i18n/
126
127 AUTH_USER_MODEL = 'system.Usuario'
128
129 LANGUAGE_CODE = 'en-us'
130
131 TIME_ZONE = 'UTC'
132
133 USE_I18N = True
134
135 USE_TZ = True
136
137
138 # Static files (CSS, JavaScript, Images)
139 # https://docs.djangoproject.com/en/5.0/howto/static-files/
```



```
140
141 STATIC_URL = 'static/'
142
143 # Default primary key field type
144 # https://docs.djangoproject.com/en/5.0/ref/settings/#default-auto-field
145
146 DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

4.4 Admin

Este archivo se encarga de registrar los modelos de la aplicación en el panel de administración de Django.

```
1 from django.contrib import admin
2 from .models import Usuario, Curso, Tarea, Entrega
3
4 admin.site.register(Usuario)
5 admin.site.register(Curso)
6 admin.site.register(Tarea)
7 admin.site.register(Entrega)
```

4.5 App

Este archivo se encarga de configurar la aplicación principal del proyecto.

```
1 from django.apps import AppConfig
2
3
4 class SystemConfig(AppConfig):
5     default_auto_field = 'django.db.models.BigAutoField'
6     name = 'system'
```

4.6 Models

- **UsuarioManager:** Clase que se encarga de la creación de usuarios. Hereda de `BaseUserManager` (clase para gestionar usuarios personalizados), crea un usuario un correo electrónico y la contraseña, se creo este usuario para crear un super usuario, el cual tendra permisos administrativos en la aplicación.
- **Usuario:** Clase que se encarga de la creación de usuarios como tal. Hereda de `AbstractBaseUser`. Se crea la clase `Type` interna que se encarga de definir los tipos de usuario (`Admin`, `Teacher`, `Student`).
- **Curso:** Clase que se encarga de la creación de cursos. Requiere de un profesor, usamos `limit_choices_to` para que solo se pueda seleccionar un profesor y la relacion `many to many` con los estudiantes (un estudiante puede estar matriculado en varios cursos y un curso puede tener varios estudiantes).
- **Tarea:** Clase que se encarga de la creación de tareas. Se relaciona a un curso a traves de la `foreign key`, contiene una fecha de entrega, nombre, descripcion y un booleano para saber si la tarea ya fua asignada.
- **Entrega:** Clase que se encarga de la creación de entregas. Se relaciona a una tarea y a un estudiante a traves de la `foreign key` (solo existe una tarea pero existen varias entregas dependiendo a la cantidad de estudiantes), contiene una descripcion, un campo de enlace, una calificación.

```
1 from django.db import models
2 from django.contrib.auth.models import AbstractUser, BaseUserManager
3
4 class UsuarioManager(BaseUserManager):
5     def create_user(self, email, password):
6         if not email:
7             raise ValueError('Ingrese un correo')
8         if not password:
9             raise ValueError('Ingrese una contraseña')
10
11         norm_email = self.normalize_email(email)
12         usuario = self.model(
13             username = norm_email,
14             email = norm_email,
15         )
16         usuario.set_password(password)
17         usuario.save(using=self._db)
18         return usuario
19
20     def create_superuser(self, email, password):
21         usuario = self.create_user(email, password)
22         usuario.is_admin = True
23         usuario.is_superuser = True
24         usuario.is_staff = True
25         usuario.tipo = Usuario.Types.ADMIN
26         usuario.save(using=self._db)
27         return usuario
28
29 class Usuario(AbstractUser):
30     class Types(models.TextChoices):
31         ADMIN = 'AD', 'Administrador'
32         STUDENT = 'ST', 'Estudiante'
33         TEACHER = 'TC', 'Docente'
34
35     email = models.EmailField(unique=True) #pk
36     name = models.CharField(max_length=50)
37     tipo = models.CharField(
38         max_length=2,
39         choices=Types.choices,
40     )
41
42     objects = UsuarioManager()
43     REQUIRED_FIELDS = []
44     USERNAME_FIELD = 'email'
45
46     def __str__(self):
47         return "{Name: " + self.name+",Email: "+self.email+"Password: "+self.password+",Type: "+self.tipo+"}"
48
49 class Curso(models.Model):
50     nombre = models.CharField(max_length=50)
51     descripcion = models.TextField()
52     docente = models.ForeignKey(Usuario, limit_choices_to={'tipo': Usuario.Types.TEACHER}, on_delete=models.CASCADE)
53     estudiantes = models.ManyToManyField(Usuario, limit_choices_to={'tipo': Usuario.Types.STUDENT}, related_name='')
54
55     def __str__(self):
56         return "{Course: " + self.nombre+",Teacher: "+self.docente.name+"}"
```

```
57
58 class Tarea(models.Model):
59     nombre = models.CharField(max_length=50)
60     descripcion = models.TextField()
61     curso = models.ForeignKey(Curso, on_delete=models.CASCADE)
62     fecha_entrega = models.DateTimeField()
63     asignada = models.BooleanField(default=False)
64
65     def __str__(self):
66         return "{Task: " + self.nombre+",Course: "+self.curso.nombre+"}"
67
68 class Entrega(models.Model):
69     tarea = models.ForeignKey(Tarea, on_delete=models.CASCADE)
70     estudiante = models.ForeignKey(Usuario, on_delete=models.CASCADE)
71     enlace = models.URLField()
72     calificacion = models.FloatField(null=True, blank=True)
73
74     def __str__(self):
75         return "{Task: " + self.tarea.nombre+",Student: "+self.estudiante.username+"}"
```

4.7 Serializers

Para emplear Django REST Framework se ha creado un archivo serializers.py en la app principal system. En este archivo se crean las clases Serializers que se encargan de serializar los datos de los modelos, esto es importante porque nos va a permitir enviar objetos JSON al Frontend.

- **UsuarioSerializer:** Clase que se encarga de serializar los datos de los usuarios. Se crea un campo de solo lectura para el tipo de usuario.
- **CursoSerializer:** Clase que se encarga de serializar los datos de los cursos. Se crea un campo de solo lectura para el profesor.
- **TareaSerializer:** Clase que se encarga de serializar los datos de las tareas. Se crea un campo de solo lectura para el curso.
- **EntregaSerializer:** Clase que se encarga de serializar los datos de las entregas. Se crea un campo de solo lectura para la tarea.

```
1 from rest_framework import serializers
2 from .models import Usuario, Curso, Tarea, Entrega
3
4 class UsuarioSerializer(serializers.ModelSerializer):
5     class Meta:
6         model = Usuario
7         fields = ['id', 'name', 'email', 'password', 'tipo',]
8         extra_kwargs = {'password': {'write_only': True, 'required': False}}
9
10    def create(self, validated_data):
11        user = Usuario.objects.create_user(
12            email=validated_data['email'],
13            password=validated_data['password']
14        )
15        user.name = validated_data['name']
16        user.tipo = validated_data['tipo']
17        user.save()
18        return user
19
```

```
20 def update(self, instance, validated_data):
21     instance.name = validated_data.get('name', instance.name)
22     instance.email = validated_data.get('email', instance.email)
23     instance.tipo = validated_data.get('tipo', instance.tipo)
24     if 'password' in validated_data and validated_data['password']:
25         instance.set_password(validated_data['password'])
26     instance.save()
27     return instance
28
29 class CursoSerializer(serializers.ModelSerializer):
30     docente = serializers.PrimaryKeyRelatedField(queryset=Usuario.objects.filter(tipo=Usuario.Types.TEACHER))
31     estudiantes = UsuarioSerializer(read_only=True, many=True, required=False)
32     class Meta:
33         model = Curso
34         fields = ['id', 'nombre', 'descripcion', 'docente', 'estudiantes']
35
36     def validate(self, data):
37         nombre = data.get('nombre')
38         docente = data.get('docente')
39         if Curso.objects.filter(nombre=nombre, docente=docente).exists():
40             raise serializers.ValidationError("El curso ya existe")
41         return data
42
43 class TareaSerializer(serializers.ModelSerializer):
44     curso = serializers.PrimaryKeyRelatedField(queryset=Curso.objects.all())
45
46     class Meta:
47         model = Tarea
48         fields = ['id', 'nombre', 'descripcion', 'curso', 'fecha_entrega', 'asignada']
49
50     def to_representation(self, instance):
51         representation = super().to_representation(instance)
52         entrega = Entrega.objects.filter(tarea=instance).first()
53         representation['calificacion'] = entrega.calificacion if entrega else None
54         return representation
55
56 class EntregaSerializer(serializers.ModelSerializer):
57     tarea = TareaSerializer(read_only=True)
58     estudiante = UsuarioSerializer(read_only=True)
59
60     class Meta:
61         model = Entrega
62         fields = ['id', 'tarea', 'estudiante', 'enlace', 'calificacion']
```

4.8 Urls

Para gestionar las URLs en una aplicación Django REST Framework, configuramos rutas específicas para cada función y grupo de usuarios.

- **Autenticación de Tokens:** Se ha utilizado el método de autenticación de Token para la autenticación de los usuarios. TokenObtainPairView se encarga de generar el token de acceso y TokenRefreshView se encarga de refrescar el token.
- **Registro y Login:** Se ha creado una vista para el registro de usuarios y otra vista para el login de usuarios.
- **Rutas para cada tipo de usuario:** Se ha creado rutas para cada tipo de usuario (Admin, Teacher, Student).

```
1 from django.urls import include, path
2 from .import views
3 from admin import views as admin_views
4 from teacher import views as teacher_views
5 from student import views as student_views
6 from rest_framework_simplejwt.views import TokenObtainPairView, TokenRefreshView
7
8 urlpatterns = [
9     # Tokens de autenticación
10    # Token => Existe una cuenta
11    path("system/token/", TokenObtainPairView.as_view(), name="get_token"),
12    path("system/token/refresh/", TokenRefreshView.as_view(), name="refresh"),
13    # Usuarios
14    path("system-auth/", include("rest_framework.urls", namespace="rest_framework")),
15
16    # Registro de usuarios
17    path("system/user/register/", views.CreateUserView.as_view(), name="register"),
18
19    # Login
20    path("system/user/login/", views.LoginView.as_view(), name="login"),
21
22
23    # ADMIN
24    # usuarios
25    path("system/user/list/", admin_views.UserListView.as_view(), name="user_list"),
26    path("system/user/<int:pk>/", admin_views.UserDetailView.as_view(), name="user_detail"),
27    # cursos
28    path("system/course/create/", admin_views.CourseCreateView.as_view(), name="course_create"),
29    path("system/course/list/", admin_views.CourseListView.as_view(), name="course_list"),
30    path("system/course/<int:pk>/", admin_views.CourseDetailView.as_view(), name="course_detail"),
31    path("system/teacher/list/", admin_views.TeacherListView.as_view(), name="teacher_list"),
32
33
34    # TEACHER
35    # Teachers
36    path("system/teacher/course/<int:pk>/", teacher_views.CoursesByTeacherView.as_view(), name="teacher_course"),
37    # Tareas
38    path("system/course/<int:curso_id>/task/create/", teacher_views.TaskCreateView.as_view(), name="task_create"),
39    path("system/course/<int:curso_id>/task/list/", teacher_views.TaskListView.as_view(), name="task_list"),
40    path("system/course/<int:curso_id>/task/assign/", teacher_views.AssignTaskView.as_view(), name="task_detail"),
41    # Entregas
42    path("system/course/task/deliveries/grade/", teacher_views.GradeDeliveryView.as_view(), name="delivery_grade"),
43    path("system/course/task/deliveries/", teacher_views.DeliveryByTaskView.as_view(), name="deliveries_list"),
44
45
46    # STUDENT
47    # inscripción a cursos
48    path("system/student/enroll/<int:pkC>/<int:pkE>/", student_views.CoursesOfAStudentView.as_view(), name="enroll"),
49    # my courses
50    path("system/student/<int:pk>/courses/", student_views.StudentMyCourses.as_view(), name="student_mycourses"),
51    # tareas asignadas a un estudiante
52    path("system/student/<int:student_id>/assigned_tasks/", student_views.AssignedTasksView.as_view(), name="assigned_tasks"),
53    # tareas entregadas por un estudiante
54    path("system/student/<int:student_id>/submitted_tasks/", student_views.SubmittedTasksView.as_view(), name="submitted_tasks"),
55    # entregas
56    path("system/student/<int:user_id>/delivery/", student_views.UpdateEntregaView.as_view(), name="create_entrega")
```

57
58]
59

4.9 System

Esta app contiene las vistas para el registro y login de usuarios.

4.9.1 Views

- **CreateUserView:** Vista que se encarga de la creación o registro de usuarios. Se ha utilizado el método de autenticación de Token, el cual se encarga de generar el token de acceso.
- **LoginView:** Vista que se encarga de verificar las credenciales de los usuarios y de generar el token de acceso, esto para poder logearnos.

```
1  # Register
2  from rest_framework import generics, permissions
3  from .models import *
4  from .serializers import *
5  # Login
6  from django.contrib.auth import authenticate, login
7  from rest_framework.views import APIView
8  from rest_framework.response import Response
9  from rest_framework import status
10 from rest_framework_simplejwt.tokens import RefreshToken
11
12 class CreateUserView(generics.CreateAPIView):
13     queryset = Usuario.objects.all()
14     serializer_class = UsuarioSerializer
15     permission_classes = [permissions.AllowAny]
16
17     def get_serializer_context(self):
18         context = super().get_serializer_context()
19         context.update({"request": self.request})
20         return context
21
22 class LoginView(APIView):
23     permission_classes = [permissions.AllowAny]
24
25     def post(self, request):
26         email = request.data.get("email")
27         password = request.data.get("password")
28
29         user = authenticate(request, email=email, password=password)
30         if user is not None:
31             login(request, user)
32             refresh = RefreshToken.for_user(user)
33             return Response({
34                 "message": "Usuario autenticado",
35                 "user": UsuarioSerializer(user).data,
36                 "access_token": str(refresh.access_token),
37                 "refresh_token": str(refresh),
38             })
39         else:
40             return Response({"message": "Usuario no autenticado"}, status=status.HTTP_401_UNAUTHORIZED)
41
```

4.10 Admin

Esta app contiene las vistas para el administrador, el cual puede ver, editar y eliminar la lista de usuarios, crear cursos y asignarles un profesor y ver la lista de cursos.

4.10.1 Views

- **UserListView:** Vista que se encarga de mostrar la lista de usuarios.
- **UserDetailView:** Vista que se encarga de mostrar los detalles de un usuario, se ha utilizado el método put para actualizar o editar los datos de un usuario y el método delete para eliminar a un usuario.
- **CursoCreateView:** Vista que se encarga de crear un curso.
- **CursoListView:** Vista que se encarga de mostrar la lista de cursos.
- **CursoDetailView:** Vista que se encarga de mostrar los detalles, de igual forma se ha utilizado el método put y delete para realizar las acciones correspondientes.
- **TeacherListView:** Vista que se encarga de lista a todos los usuarios que tiene el rol TC (Teacher).

```
1  # Models
2  from system.models import *;
3  from system.serializers import *;
4  # Rest Framework
5  from rest_framework import generics, permissions
6  from rest_framework.views import APIView
7  from rest_framework.response import Response
8  from rest_framework import status
9
10 class UserListView(APIView):
11     permission_classes = [permissions.AllowAny]
12
13     def get(self, request):
14         print(request.user)
15         users = Usuario.objects.exclude(tipo='AD')
16         serializer = UsuarioSerializer(users, many=True)
17         return Response(serializer.data, status=status.HTTP_200_OK)
18
19 class UserDetailView(APIView):
20     permission_classes = [permissions.AllowAny]
21
22     def put(self, request, pk):
23         print(request.data)
24         try:
25             user = Usuario.objects.get(pk=pk)
26             print(user)
27             serializer = UsuarioSerializer(user, data=request.data, partial=True)
28             if serializer.is_valid():
29                 if 'password' in request.data and request.data['password']:
30                     user.set_password(request.data['password'])
31                     serializer.save()
32                 return Response(serializer.data, status=status.HTTP_200_OK)
33             print(serializer.errors)
34             return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
35         except Usuario.DoesNotExist:
```

```
36         return Response({"error": "Usuario no encontrado."}, status=status.HTTP_404_NOT_FOUND)
37
38     def delete(self, request, pk):
39         try:
40             user = Usuario.objects.get(pk=pk)
41             user.delete()
42             return Response(status=status.HTTP_204_NO_CONTENT)
43         except Usuario.DoesNotExist:
44             return Response({"error": "Usuario no encontrado."}, status=status.HTTP_404_NOT_FOUND)
45
46     class CourseCreateView(generics.CreateAPIView):
47         queryset = Curso.objects.all()
48         serializer_class = CursoSerializer
49         permission_classes = [permissions.AllowAny]
50
51     def get_serializer_context(self):
52         context = super().get_serializer_context()
53         context.update({"request": self.request})
54         return context
55
56
57     class CourseListView(APIView):
58         permission_classes = [permissions.AllowAny]
59
60     def get(self, request):
61         courses = Curso.objects.all()
62         serializer = CursoSerializer(courses, many=True)
63         return Response(serializer.data, status=status.HTTP_200_OK)
64
65     class CourseDetailView(APIView):
66         permission_classes = [permissions.AllowAny]
67
68     def get(self, request, pk):
69         try:
70             course = Curso.objects.get(pk=pk)
71             serializer = CursoSerializer(course)
72             return Response(serializer.data, status=status.HTTP_200_OK)
73         except Curso.DoesNotExist:
74             return Response({"error": "Curso no encontrado."}, status=status.HTTP_404_NOT_FOUND)
75
76     def put(self, request, pk):
77         print("DATA: ", request.data)
78         try:
79             course = Curso.objects.get(pk=pk)
80             serializer = CursoSerializer(course, data=request.data, partial=True)
81             if serializer.is_valid():
82                 serializer.save()
83                 return Response(serializer.data, status=status.HTTP_200_OK)
84             return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
85         except Curso.DoesNotExist:
86             return Response({"error": "Curso no encontrado."}, status=status.HTTP_404_NOT_FOUND)
87
88     def delete(self, request, pk):
89         try:
90             course = Curso.objects.get(pk=pk)
91             course.delete()
```



```
92     return Response(status=status.HTTP_204_NO_CONTENT)
93 except Curso.DoesNotExist:
94     return Response({"error": "Curso no encontrado."}, status=status.HTTP_404_NOT_FOUND)
95
96 class TeacherListView(APIView):
97     permission_classes = [permissions.AllowAny]
98
99     def get(self, request):
100         teachers = Usuario.objects.filter(tipo='TC')
101         serializer = UsuarioSerializer(teachers, many=True)
102         return Response(serializer.data, status=status.HTTP_200_OK)
103
```

4.11 Teacher

Esta app contiene las vistas para el profesor, el cual puede ver los cursos que enseña, crear y asignar tareas para cada curso y ver la lista de tareas y calificarlas.

- **CourseByTeacherView:** Vista que se encarga de mostrar los cursos que enseña un profesor.
- **TaskCreateView:** Vista que se encarga de crear una tarea.
- **TaskListView:** Vista que se encarga de mostrar la lista de tareas de un curso específico.
- **AssignTaskView:** Vista que se encarga de asignar una tarea a todos los estudiante de un curso, crea una entrega.
- **DeliveryByTaskView:** Vista que se encarga de mostrar las entregas de una tarea específica.
- **GradeDeliveryView:** Vista de modificar la calificación de una entrega.

```
1  # Models
2  from system.models import *;
3  from system.serializers import *;
4  # Rest Framework
5  from rest_framework import generics, permissions
6  from rest_framework.views import APIView
7  from rest_framework.response import Response
8  from rest_framework import status
9
10 class CoursesByTeacherView(APIView):
11     permission_classes = [permissions.AllowAny]
12
13     def get(self, request, pk):
14         courses = Curso.objects.filter(docente=pk)
15         serializer = CursoSerializer(courses, many=True)
16         return Response(serializer.data, status=status.HTTP_200_OK)
17
18 # Task
19 class TaskCreateView(generics.CreateAPIView):
20     queryset = Tarea.objects.all()
21     serializer_class = TareaSerializer
22     permission_classes = [permissions.AllowAny]
23
24     def post(self, request, *args, **kwargs):
25         curso_id = self.kwargs.get('curso_id')
26         data = request.data.copy()
27         data['curso'] = curso_id
```

```
28         serializer = self.get_serializer(data=data)
29
30         if serializer.is_valid():
31             self.perform_create(serializer)
32             headers = self.get_success_headers(serializer.data)
33             return Response(serializer.data, status=status.HTTP_201_CREATED, headers=headers)
34         return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
35
36 class TaskListView(APIView):
37     permission_classes = [permissions.AllowAny]
38
39     def get(self, request, curso_id):
40         tasks = Tarea.objects.filter(curso=curso_id)
41         serializer = TareaSerializer(tasks, many=True)
42         return Response(serializer.data, status=status.HTTP_200_OK)
43
44
45 # Assign Task
46 class AssignTaskView(APIView):
47     permission_classes = [permissions.AllowAny]
48
49     def post(self, request, *args, **kwargs):
50         task_id = request.data.get('task_id')
51         course_id = kwargs.get('curso_id')
52         try:
53             tarea = Tarea.objects.get(id=task_id)
54             curso = Curso.objects.get(id=course_id)
55             estudiantes = curso.estudiantes.all()
56             print("S", estudiantes)
57
58             for estudiante in estudiantes:
59                 Entrega.objects.get_or_create(
60                     tarea=tarea,
61                     estudiante=estudiante,
62                     defaults={'enlace': ''}
63                 )
64
65             tarea.asignada = True
66             tarea.save()
67             print(tarea.__dict__)
68             return Response({"status": "Tasks assigned successfully"}, status=status.HTTP_200_OK)
69         except (Tarea.DoesNotExist, Curso.DoesNotExist):
70             return Response({"error": "Task or course not found"}, status=status.HTTP_404_NOT_FOUND)
71
72 # list deliveries by task
73 class DeliveryByTaskView(APIView):
74     permission_classes = [permissions.AllowAny]
75
76     def post(self, request, *args, **kwargs):
77         task_id = request.data.get('task_id')
78         try:
79             task = Tarea.objects.get(id=task_id)
80             deliveries = Entrega.objects.filter(tarea=task)
81             serializer = EntregaSerializer(deliveries, many=True)
82             return Response(serializer.data, status=status.HTTP_200_OK)
83         except:
```

```
84         return Response({"error": "Deliveries not found"}, status=status.HTTP_404_NOT_FOUND)
85
86     # Update delivery grade
87     class GradeDeliveryView(APIView):
88         permission_classes = [permissions.AllowAny]
89
90         def post(self, request, *args, **kwargs):
91             entrega_id = request.data.get('delivery_id')
92             grade = request.data.get('grade')
93             try:
94                 print(entrega_id, grade)
95                 entrega = Entrega.objects.get(id=entrega_id)
96
97                 entrega.calificacion = grade
98                 entrega.save()
99                 return Response({"status": "Grade assigned successfully"}, status=status.HTTP_200_OK)
100             except Entrega.DoesNotExist:
101                 return Response({"error": "Delivery not found"}, status=status.HTTP_404_NOT_FOUND)
```

4.12 Student

Esta app contiene las vistas para el estudiante, el cual puede ver los cursos en los que está inscrito, ver las tareas asignadas y ver sus calificaciones.

- **CoursesOfAStudentView:** Vista que se encarga de inscribir un estudiante en un curso.
- **StudentMyCoursesView:** Vista que se encarga de mostrar los cursos en los que está inscrito un estudiante.
- **AssignTasksView:** Vista que se encarga de mostrar las tareas asignadas a un estudiante.
- **SubmittedTasksView:** Vista que se encarga de mostrar la tareas que ya han sido entregadas por un estudiante, así como su calificación (Si es que le calificaron o no).
- **UpdateEntregaView:** Vista que se encarga de actualizar la entrega de un estudiante para una tarea específica.

```
1  # Models
2  from system.models import *;
3  from system.serializers import *;
4  # Rest Framework
5  from rest_framework import generics, permissions
6  from rest_framework.views import APIView
7  from rest_framework.response import Response
8  from rest_framework import status
9
10 # Enroll a student in a course
11 class CoursesOfAStudentView(APIView):
12     permission_classes = [permissions.AllowAny]
13
14     def post(self, request, pkC, pkE):
15         print(pkC, pkE)
16         try:
17             course = Curso.objects.get(id=pkC)
18             print(course)
19             student = Usuario.objects.get(id=pkE, tipo='ST')
20             print(student)
```

```
21         course.estudiantes.add(student)
22         course.save()
23         return Response({"message": "Estudiante inscrito con el éxito"}, status=status.HTTP_200_OK)
24     except Curso.DoesNotExist:
25         return Response({"error": "Curso no encontrado."}, status=status.HTTP_404_NOT_FOUND)
26     except Usuario.DoesNotExist:
27         return Response({"error": "Estudiante no encontrado."}, status=status.HTTP_404_NOT_FOUND)
28
29     # Student my courses
30     class StudentMyCourses(APIView):
31         permission_classes = [permissions.AllowAny]
32
33         def get(self, request, pk):
34             try:
35                 student = Usuario.objects.get(id=pk, tipo='ST')
36                 courses = student.cursos.all()
37                 serializer = CursoSerializer(courses, many=True)
38                 return Response(serializer.data, status=status.HTTP_200_OK)
39             except Usuario.DoesNotExist:
40                 return Response({"error": "Estudiante no encontrado."}, status=status.HTTP_404_NOT_FOUND)
41
42     # Student list Tasks
43     class AssignedTasksView(generics.ListAPIView):
44         serializer_class = EntregaSerializer
45         permission_classes = [permissions.AllowAny]
46
47         def get_queryset(self):
48             student_id = self.kwargs.get('student_id')
49             try:
50                 user = Usuario.objects.get(id=student_id, tipo=Usuario.Types.STUDENT)
51                 return Entrega.objects.filter(estudiante_id=user.id, enlace='')
52             except Usuario.DoesNotExist:
53                 return Tarea.objects.none()
54
55     # Student list delivered
56     class SubmittedTasksView(generics.ListAPIView):
57         serializer_class = EntregaSerializer
58         permission_classes = [permissions.AllowAny]
59
60         def get_queryset(self):
61             student_id = self.kwargs.get('student_id')
62             try:
63                 user = Usuario.objects.get(id=student_id, tipo=Usuario.Types.STUDENT)
64                 return Entrega.objects.filter(estudiante=user).exclude(enlace='')
65             except Usuario.DoesNotExist:
66                 return Entrega.objects.none()
67
68     # Student create deliveries
69     class UpdateEntregaView(generics.CreateAPIView):
70         serializer_class = EntregaSerializer
71         permission_classes = [permissions.AllowAny]
72
73         def post(self, request, user_id):
74             try:
75                 user = Usuario.objects.get(id=user_id)
76             except Usuario.DoesNotExist:
```

```
77     return Response({"error": "Usuario no encontrado"}, status=status.HTTP_404_NOT_FOUND)
78
79     if user.tipo != Usuario.Types.STUDENT:
80         return Response({"error": "Permiso denegado"}, status=status.HTTP_403_FORBIDDEN)
81
82     tarea_id = request.data.get('tarea')
83     url = request.data.get('url')
84
85     try:
86         tarea = Tarea.objects.get(id=tarea_id)
87     except Tarea.DoesNotExist:
88         return Response({"error": "Tarea no encontrada"}, status=status.HTTP_404_NOT_FOUND)
89
90     try:
91         entrega = Entrega.objects.get(tarea=tarea, estudiante=user)
92         entrega.enlace = url
93         entrega.save()
94         return Response(EntregaSerializer(entrega).data, status=status.HTTP_200_OK)
95     except Entrega.DoesNotExist:
96         entrega = Entrega.objects.create(
97             tarea=tarea,
98             estudiante=user,
99             enlace=url
100         )
101     return Response(EntregaSerializer(entrega).data, status=status.HTTP_201_CREATED)
```

4.13 Base de Datos

5 Frontend

Para el lado del Frontend se ha utilizado React JS, es una biblioteca de JavaScript para construir interfaces de usuario. Se ha dividido en 9 componentes (Utilities, Components, Home, Admin, Teacher, Student) para una mejor organización del proyecto. Se hizo uso de los routers de React para la navegación entre las páginas. Para el estilo se ha utilizado Tailwind CSS, nos va a permitir que todo nuestra aplicacion web sea responsive y se adapte a cualquier dispositivo, asi como el modo Dark para una mejor experiencia de usuario. El cual es un framework de diseño de componentes de interfaz de usuario de código abierto que ayuda a crear diseños modernos y personalizables. Para la contruccion del entorno usamos Vite, que es un entorno de desarrollo web que permite a los desarrolladores crear aplicaciones web modernas con una configuración mínima.

5.1 Conexión con el Backend

Para la comunicación entre el Frontend y el Backend se ha utilizado Axios, una librería de JavaScript que se encarga de realizar peticiones HTTP. Se ha utilizado CORS para permitir la comunicación entre el Frontend y el Backend.

5.2 Utilities

- **UserContext:**
- **UserProvider:** Envuelve los componentes de la aplicación, permitiendo los usuarios acceder a las rutas de acuerdo al tipo de usuario.
- **PrivateRoute:** Componente de ruta que redirige a la página de inicio de sesión si el usuario no está autenticado.

5.2.1 UserContext

Proporciona un contexto global para el estado de autenticación del usuario y funciones relacionadas.

UserProvider: Proporciona un contexto global para el estado de autenticación del usuario y funciones relacionadas.

```
1 import React, {createContext, useContext, useState, useEffect} from "react";
2
3 const UserContext = createContext();
4
5 export const useUser = () => {
6   return useContext(UserContext);
7 }
8
9 export const UserProvider = ({children}) => {
10   const [user, setUser] = useState(JSON.parse(localStorage.getItem('user')));
11
12   const login = (userData, tokens) => {
13     setUser(userData);
14     localStorage.setItem('user', JSON.stringify(userData));
15     localStorage.setItem('access_token', tokens.access);
16     localStorage.setItem('refresh_token', tokens.refresh);
17   };
18
19   const logout = () => {
20     setUser(null);
21     localStorage.removeItem('user');
22     localStorage.removeItem('access_token');
23     localStorage.removeItem('refresh_token');
24   };
25
26   return (
27     <UserContext.Provider value={{ user, login, logout }}>
28       {children}
29     </UserContext.Provider>
30   );
31 }
32
33
```

5.2.2 PrivateRoute

Componente de ruta que redirige a la página de inicio de sesión si el usuario no está autenticado, de lo contrario, muestra el componente solicitado.

```
1 import React from "react";
2 import { Navigate, Outlet } from "react-router-dom";
3 import { useUser } from "./useContext";
4
5 function PrivateRoute({ allowedRoles }) {
6   const { user } = useUser();
7   console.log("PrivateRoute", user);
8
9   if (!user) {
10     console.log("Login required");
11     return <Navigate to="/" />;
12   }
13 }
14
```

```
13
14   if(allowedRoles && !allowedRoles.includes(user.tipo)){
15       console.log("User not allowed");
16       return <Navigate to="/access-denied" />;
17   }
18
19   return <Outlet />;
20 }
21
22 export default PrivateRoute;
```

5.3 AccessDenied

Componente que se muestra cuando un usuario intenta acceder a una página sin los permisos necesarios.

```
1 function AccessDenied() {
2   return (
3     <div className="flex items-center justify-center min-h-screen bg-primary-dark">
4       <div className="max-w-md w-full p-6 bg-white rounded-lg shadow-lg">
5         <h2 className="text-3xl font-bold text-center mb-4 text-black">Access Denied</h2>
6         <p className="text-center text-gray-600">
7           You do not have permission to access this page.
8         </p>
9       </div>
10    </div>
11  );
12 }
13
14 export default AccessDenied;
```

5.4 Components

5.4.1 RegisterUser

El componente RegisterUser realiza dos solicitudes principales: primero, envía una solicitud POST a `http://localhost:8000/system/user/register/` para registrar un nuevo usuario con los datos del formulario (email, name, tipo, password). Una vez registrado, realiza otra solicitud POST a `http://localhost:8000/system/user/login/` para iniciar sesión con las credenciales del usuario recién registrado (email y password). Esto permite al usuario registrarse e iniciar sesión de inmediato, obteniendo así los tokens de acceso necesarios para la autenticación.

```
1 import { useState } from 'react';
2 import axios from 'axios';
3 import { Link } from 'react-router-dom';
4 import { useNavigate } from 'react-router-dom';
5 import { useUser } from './useContext';
6
7 function RegisterUser() {
8   const {login} = useUser();
9   const [formData, setFormData] = useState({
10     email: '',
11     name: '',
12     tipo: 'ST',
13     password: ''
14   });
15   const [formDataLogin, setFormDataLogin] = useState({
16     email: '',
```

```
17     password: ''
18   });
19   const navigate = useNavigate();
20
21   const handleChange = (e) => {
22     setFormData({ ...formData, [e.target.name]: e.target.value });
23   };
24
25   const handleSubmit = async (e) => {
26     e.preventDefault();
27     try {
28       const response = await axios.post('http://localhost:8000/system/user/register/', formData);
29       console.log('Usuario registrado:', response.data);
30       formDataLogin.email = formData.email;
31       formDataLogin.password = formData.password;
32       loginUser(e);
33     } catch (error) {
34       console.error('Error al registrar usuario:', error);
35     }
36   };
37
38   const loginUser = async (e) => {
39     e.preventDefault();
40     const { email, password } = formDataLogin;
41     try {
42       const response = await axios.post('http://localhost:8000/system/user/login/', { email, password });
43       const access = response.data.access_token;
44       const refresh = response.data.refresh_token;
45       const user = response.data.user;
46       const userT = user.tipo;
47       login(user, {access: access, refresh: refresh});
48
49       if (userT === 'ST') {
50         navigate('/student', { state: user });
51       } else if (userT === 'TC') {
52         navigate('/teacher', { state: user });
53       } else if (userT === 'AD') {
54         console.log('Admin', user);
55         navigate('/admin', { state: user });
56       } else {
57         navigate('/');
58       }
59     } catch (error) {
60       console.error('Error al iniciar sesión (No existe la cuenta):', error);
61     }
62   };
63
64   return (
65     <div className="flex items-center justify-center min-h-screen bg-secondary-light dark:bg-secondary-dark">
66       <div className="max-w-md w-full p-6 bg-card-light dark:bg-card-dark rounded-lg shadow-lg">
67         <h1 className="text-3xl font-bold text-center mb-2 text-primary-light dark:text-primary-dark">Welcome Co
68         <h2 className="text-3xl font-bold text-center mb-0">Registrate</h2>
69         <p className="text-center text-muted-foreground mb-5">Crea una cuenta como estudiante o docente</p>
70         <form onSubmit={handleSubmit} className="space-y-4">
71           <input
72             type="text"
```



```

73         name="name"
74         value={formData.name}
75         onChange={handleChange}
76         placeholder="Nombre"
77         required
78         className="w-full px-3 py-2 border border-secondary-light dark:border-secondary-dark rounded-md focus:outline-none"
79     />
80     <input
81         type="email"
82         name="email"
83         value={formData.email}
84         onChange={handleChange}
85         placeholder="Correo electrónico"
86         required
87         className="w-full px-3 py-2 border border-secondary-light dark:border-secondary-dark rounded-md focus:outline-none"
88     />
89     <input
90         type="password"
91         name="password"
92         value={formData.password}
93         onChange={handleChange}
94         placeholder="Contraseña"
95         required
96         className="w-full px-3 py-2 border border-secondary-light dark:border-secondary-dark rounded-md focus:outline-none"
97     />
98     <select
99         name="tipo"
100         value={formData.tipo}
101         onChange={handleChange}
102         className="w-full px-3 py-2 border border-secondary-light dark:border-secondary-dark rounded-md focus:outline-none"
103     >
104         <option value="ST">Estudiante</option>
105         <option value="TC">Docente</option>
106     </select>
107     <button
108         type="submit"
109         className="w-full bg-primary-light dark:bg-primary-dark text-text-light dark:text-text-dark py-2 px-3"
110     >
111         Registrarse
112     </button>
113 </form>
114 <p className="text-center mt-4 text-muted-foreground">
115     ¿Ya tienes una cuenta? <Link to="/login" className="text-primary-light dark:text-primary-dark">Inicia sesión</Link>
116 </p>
117 </div>
118 </div>
119 );
120
121 }
122
123
124 export default RegisterUser;

```

5.4.2 LoginUser

El componente LoginUser realiza una solicitud POST a `http://localhost:8000/system/user/login/` para iniciar sesión con las credenciales del usuario (email y password). Una vez iniciada la sesión, el componente almacena los tokens de acceso y refresco en el almacenamiento local del navegador, lo que permite al usuario permanecer autenticado incluso después de cerrar la página.

```
1  import { useState } from 'react';
2  import { useNavigate, Link } from 'react-router-dom';
3  import axios from 'axios';
4  import { useUser } from './useContext';
5
6  function LoginUser(){
7    const navigate = useNavigate();
8    const { login } = useUser();
9    const [formData, setFormData] = useState({
10      email: '',
11      password: ''
12    });
13    const [userType, setUserType] = useState(null);
14
15    const handleChange = (e) => {
16      setFormData({ ...formData, [e.target.name]: e.target.value });
17    };
18
19    const handleSubmit = async (e) => {
20      e.preventDefault();
21      const { email, password } = formData;
22      try {
23        const response = await axios.post('http://localhost:8000/system/user/login/', { email, password });
24        const access = response.data.access_token;
25        const refresh = response.data.refresh_token;
26        const user = response.data.user;
27        const userT = user.tipo;
28        setUserType(userT);
29
30        login(user, {access: access, refresh: refresh});
31
32        if (userT === 'ST') {
33          navigate('/student', { state: user });
34        } else if (userT === 'TC') {
35          navigate('/teacher', { state: user });
36        } else if (userT === 'AD') {
37          console.log('Admin', user);
38          navigate('/admin', { state: user });
39        } else {
40          setUserType(null);
41          navigate('/');
42        }
43      } catch (error) {
44        console.error('Error al iniciar sesión (No existe la cuenta):', error);
45        setUserType(null);
46      }
47    }
48  }
49
50  return (
```

```

51 <div className="flex items-center justify-center min-h-screen bg-secondary-light dark:bg-secondary-dark">
52   <div className="max-w-md w-full p-6 bg-card-light dark:bg-card-dark rounded-lg shadow-lg">
53     <h1 className="text-3xl font-bold text-center mb-2 text-primary-light dark:text-primary-dark">Welcome Co
54     <h2 className="text-3xl font-bold text-center mb-0.5 text-primary-light dark:text-primary-dark">Iniciar
55     <p className="text-center text-muted-foreground mb-5">
56       Ingresa tus credenciales para acceder a Coder Dojo
57     </p>
58     <form onSubmit={handleSubmit} className="space-y-4">
59       <div className="space-y-2">
60         <input
61           type="email"
62           name="email"
63           value={formData.email}
64           onChange={handleChange}
65           placeholder="Correo electrónico"
66           required
67           className="w-full px-3 py-2 border border-secondary-light dark:border-secondary-dark rounded-md fo
68         />
69       </div>
70       <div className="space-y-2">
71         <input
72           type="password"
73           name="password"
74           value={formData.password}
75           onChange={handleChange}
76           placeholder="Contraseña"
77           required
78           className="w-full px-3 py-2 border border-secondary-light dark:border-secondary-dark rounded-md fo
79         />
80       </div>
81       <button
82         type="submit"
83         className="w-full bg-primary-light dark:bg-primary-dark text-text-light dark:text-text-dark py-2 px-
84       >
85         Iniciar sesión
86       </button>
87     </form>
88     <p className="text-center mt-4 text-muted-foreground">
89       ¿No tienes una cuenta? <Link to="/register" className="text-primary-light dark:text-primary-dark">Regi
90     </p>
91     <p className="text-center mt-4 text-green-500">
92       {userType === 'ST' ? 'Estudiante' : userType === 'PR' ? 'Profesor' : 'Acceso denegado(Ingresa un usuar
93     </p>
94   </div>
95 </div>
96 );
97 }
98
99 export default LoginUser;

```

5.5 Home

5.5.1 Home

El componente Home muestra la página de inicio de la aplicación, contiene subcomponentes (HomeNavigation, HomeMain, HomeRol, HomeAboutUs, HomeFooter), los cuales nos permite visualizar las

diferentes partes de la página de inicio. Tema Oscuro, por si es que el usuario prefiere un tema oscuro.

```

1  import { useEffect, useState } from "react";
2  import HomeNavigation from './HomeNavigation';
3  import HomeMain from './HomeMain';
4  import HomeFooter from './HomeFooter';
5  import HomeRol from './HomeRol';
6  import HomeAboutUs from './HomeAboutUs';
7  function Home() {
8      const [isDarkMode, setIsDarkMode] = useState(false);
9
10     useEffect(() => {
11         if (isDarkMode) {
12             document.documentElement.classList.add('dark');
13         } else {
14             document.documentElement.classList.remove('dark');
15         }
16     }, [isDarkMode]);
17
18     return (
19         <div className="w-screen
20         dark:bg-[#2c2c2c]
21         ">
22             <HomeNavigation isDarkMode={isDarkMode} setIsDarkMode={setIsDarkMode} />
23             <HomeMain />
24             <HomeRol />
25             <HomeAboutUs />
26             <HomeFooter />
27
28         </div>
29     );
30 }
31
32 export default Home;

```

subsubsectionHomeNavigation El componente HomeNavigation muestra la barra de navegación de la página de inicio, contiene un menú de navegación con enlaces a las diferentes secciones de la página, así como los botones de inicio de sesión y registro.

```

1  import { useState } from 'react';
2  import Button from './Button';
3
4  function HomeNavigation({ isDarkMode, setIsDarkMode }) {
5      const [isMenuOpen, setIsMenuOpen] = useState(false);
6      return (
7          <header className="
8              text-primary-dark dark:text-text-light py-6 px-6 shadow-sm flex justify-between items-center absolute w-full
9              <div className="flex items-center gap-4">
10                  <a href="#" className="flex items-center">
11                      <span className="text-xl md:text-2xl font-bold text-text-dark dark:text-text-light">
12                          <i className="text-3xl text-text-dark dark:text-text-light bx bx-code-alt"></i> CoderDojo
13                      </span>
14                  </a>
15              </div>
16              <nav className={`flex-col md:flex-row md:flex gap-8 items-center ${isMenuOpen ? 'flex' : 'hidden'}
17                  md:flex absolute md:relative top-full right-0 md:top-0 md:left-auto
18                  px-8 py-4 md:py-0 md:px-0

```

```

19     md:w-auto bg-menu-light dark:bg-menu-dark md:bg-transparent
20     md:dark:bg-transparent
21     animate-open-menu md:animate-none
22     z-50 md:z-auto`
23 }>
24     <a href="#" className="font-semibold
25     border-b-2 border-transparent
26     hover:border-primary-light
27     dark:hover:border-text-light
28     ">Home</a>
29     <a href="#" className="font-semibold
30     border-b-2 border-transparent
31     hover:border-primary-light
32     dark:hover:border-text-light">Nosotros</a>
33     <a href="#" className="font-semibold
34     border-b-2 border-transparent
35     hover:border-primary-light
36     dark:hover:border-text-light">Más Información</a>
37     <Button text="Register" icon="bx bx-user-plus" url='register/' />
38     <Button text="Login" icon="bx bx-log-in" url='login/' />
39 </nav>
40 <div className="flex items-center gap-4">
41     <button
42         onClick={() => setIsDarkMode(!isDarkMode)}
43     >
44         <i className={`w-8 h-8 text-2xl
45             text-primary-light dark:text-text-light
46             ${isDarkMode ? 'bx bx-sun' : 'bx bx-moon'}}`></i>
47         <span className="sr-only">Toggle dark mode</span>
48     </button>
49     <button
50         className="md:hidden"
51         onClick={() => setIsMenuOpen(!isMenuOpen)}
52     >
53         <i className="bx bx-menu text-3xl"></i>
54     </button>
55 </div>
56 </header>
57 );
58 }
59
60 export default HomeNavigation;

```

5.5.2 HomeMain

El componente HomeMain muestra la sección principal de la página de inicio, contiene un mensaje de bienvenida y una imagen de fondo, contiene dos subcomponentes (Cloud, Button). Cloud es un svg que a través de animaciones se mueve de un lado a otro de la pantalla. Button es un botón que nos redirige a la página de registro.

```

1 import Button from './Button';
2 import HomeEvent from './HomeEvent';
3 import HomeRol from './HomeRol';
4 import Cloud from './Cloud';
5
6 function HomeMain() {
7     return (

```

```

8   <main className="
9   bg-bgHome
10  dark:bg-gradient-to-r dark:from-slate-900 dark:to-slate-700
11  flex flex-col items-center justify-center h-screen"
12  >
13    <section className="container mx-auto flex flex-col lg:flex-row justify-center
14    gap-8 items-center py-36 px-2 md:px-12 lg:px-24 h-full
15    "
16    >
17      <Cloud className="w-[15%] absolute top-[35%] left-[10%] z-0 animate-move-cloud" />
18      <Cloud className="w-[15%] absolute top-[15%] left-[35%] z-0 animate-move-cloud"/>
19      <Cloud className="w-[15%] absolute top-[30%] right-[10%] z-0 animate-move-cloud"/>
20      <Cloud className="w-[15%] absolute top-[50%] right-[20%] z-0 animate-move-cloud"/>
21      <div className='
22      relative z-20
23      p-10 md:px-[7%] flex flex-col items-center justify-center z-1'
24      >
25        <h1 className="text-5xl md:text-9xl font-bold mb-4 text-text-dark dark:text-text-light flex justify-ce
26        <i className="text-4xl md:text-9xl text-text-dark dark:text-text-light bx bx-code-alt"></i>
27        CoderDojo
28      </h1>
29      <p className='text-center text-lg font-bold dark:text-text-light'>
30        Explora el mundo de la codificación y la programación con CoderDojo.
31      </p>
32      <p className="text-center text-md text-muted-foreground
33      dark:text-text-light
34      mb-8 font-semibold max-[50vw]" >
35        Unete a nuestra comunidad de jóvenes programadores y descubre la alegría de<br></br> programar de un
36      </p>
37      <div className="flex gap-4 items-center">
38        <Button text="Empezar Ahora" icon="bx bx-user-plus" url='register/' />
39        <a href="#" className="
40        text-primary-light rounded-md px-4 py-2
41        border-2 border-text-light
42        bg-text-light
43        hover:bg-transparent
44        dark:text-text-dark dark:border-text-dark">
45          Más Información
46        </a>
47      </div>
48    </div>
49  </section>
50  <div className='absolute -bottom-4 flex w-screen gap-4 md:gap-40 '>
51    <svg className='w-full h-auto' viewBox="0 0 356 152" fill="none" xmlns="http://www.w3.org/2000/svg">
52      <path d="M87.6551 1L1 16.9696V151H354L286.483 80.2776L161.02 71.7224L87.6551 1Z" fill="#0B2D5F" stroke
53    </svg>
54    <svg className='w-full h-auto' viewBox="0 0 357 152" fill="none" xmlns="http://www.w3.org/2000/svg">
55      <path d="M269.099 1L356 16.9696V151H2L69.7078 80.2776L195.527 71.7224L269.099 1Z" fill="#0B2D5F" strok
56    </svg>
57  </div>
58 </main>
59 );
60 }
61 export default HomeMain;

```

5.5.3 HomeRol

El componente HomeRol muestra la sección de roles de la página de inicio, contiene tres subcomponentes (CardRol), los cuales nos permite visualizar los roles de los usuarios (Admin, Teacher, Student).

```

1  import HomeRolCard from "../HomeRolCard"
2  function HomeSection() {
3    return (
4      <section className="
5        w-screen flex flex-col justify-center
6        gap-8 items-center py-36 px-10 md:px-12 lg:px-24 h-full
7        dark:bg-gradient-to-t from-primary-light to-primary-dark
8        "
9
10     >
11       <h2 className="text-3xl md:text-5xl font-bold text-primary-light dark:text-text-light">Descubre tu rol en
12       <i className="text-3xl md:text-5xl text-primary-light dark:text-text-light bx bx-code-alt ml-2" ></i>
13       CoderDojo</h2>
14       <p className="dark:text-text-light text-2xl font-semibold mb-8">¿Eres administrador, profesor o estudiante?
15       <div className="grid md:grid-cols-3 gap-8 text-center">
16         <HomeRolCard rol="Estudiante" icon="bx bx-user">
17           Explora proyectos de codificación, completa tareas y aprende a tu propio ritmo.
18         </HomeRolCard>
19         <HomeRolCard rol="Profesor" icon="bx bxs-school">
20           Asigna tareas de codificación, supervisa el progreso de los estudiantes y brinda orientación.
21         </HomeRolCard>
22         <HomeRolCard rol="Administrador" icon="bx bx-cog">
23           Administra usuarios, asigna roles y supervisa el progreso de los estudiantes.
24         </HomeRolCard>
25       </div>
26     </section>
27   )
28 }
29 export default HomeSection;

```

5.5.4 HomeAboutUs

El componente HomeAboutUs muestra la sección de acerca de nosotros de la página de inicio, contiene un mensaje de acerca de nosotros y una imagen de fondo.

```

1  import Image from './img/coderDojoMain.png'
2
3  function HomeAboutUs(){
4    return (
5      <section className="
6        w-screen flex flex-col justify-center
7        gap-8 items-center py-36 px-10 md:px-12 lg:px-24 h-full
8        dark:bg-[#2c2c2c]
9        "
10
11     >
12       <h2 className="text-3xl md:text-5xl font-bold text-primary-light dark:text-text-light">Quiénes somos?
13       </h2>
14       <div className="grid md:grid-cols-2 gap-8 text-center items-center justify-center md:w-[80%]">
15         <p className="text-center font-semibold dark:text-text-light">
16           En CoderDojo, nuestra misión es empoderar a los jóvenes mediante la enseñanza de la programación y habilidad.
17           Nuestro programa Programación para todos: CoderDojo está diseñado para ser accesible y divertido, permitiéndote aprender a tu propio ritmo.

```

```

18     </p>
19     <img src={Image} alt="CoderDojo" className="rounded-lg"/>
20   </div>
21 </section>
22 )
23 }
24
25 export default HomeAboutUs;

```

5.5.5 HomeFooter

El componente HomeFooter muestra el pie de página de la página de inicio, contiene un mensaje de derechos de autor y enlaces a las redes sociales. Este componente se muestra en todas las páginas de la aplicación.

```

1  import Image from './img/logo.png';
2
3  function HomeFooter() {
4    return (
5      <footer className="relative text-text-light
6      bg-primary-light md:bg-transparent
7      dark:text-text-dark py-8 px-6 md:px-12 lg:px-24 min-h-[200px]">
8        <div className="relative z-10 container max-w-7xl mx-auto
9        md:mt-[5%]
10        flex flex-col md:flex-row justify-between items-center gap-8">
11          <div className="flex items-center gap-4
12          dark:text-text-light
13          ">
14            <img src={Image} alt="Logo" height="100" width="100" />
15            <span className="text-2xl font-bold">
16              <i className="text-3xl text-text-light bx bx-code-alt"></i> Coder Dojo
17            </span>
18          </div>
19          <nav className="flex flex-col items-center
20          md:flex-row gap-4 font-bold dark:text-text-light">
21            <a href="/" className="
22            border-b-2 border-transparent
23            hover:border-text-light">Home</a>
24            <a href="#" className="
25            border-b-2 border-transparent
26            hover:border-text-light">Cursos</a>
27            <a href="#" className="
28            border-b-2 border-transparent
29            hover:border-text-light">Tareas</a>
30          </nav>
31          <div className="flex flex-col items-center md:items-start gap-4 font-bold dark:text-text-light">
32            <p>Redes Sociales</p>
33            <div className="flex gap-4">
34              <a href="#" className="text-2xl"><i className="bx bxl-linkedin-square"></i></a>
35              <a href="#" className="text-2xl"><i className="bx bxl-instagram"></i></a>
36              <a href="#" className="text-2xl"><i className="bx bxl-facebook"></i></a>
37            </div>
38            <p className="text-center">&copy; 2024 Coder Dojo. Todos los derechos reservados.</p>
39          </div>
40        </div>
41        <svg className="absolute top-0 left-0 w-full h-full z-0 md:h-auto" viewBox="0 0 876 192" fill="none" xmlns="http://www.w3.org/2000/svg">
42          <path d="M107 17.5063C65.4719 17.6862 1 9.00631 1 9.00631V191.506H875V9.00631C875 9.00631 835.344 34.956" />

```



```
43     </svg>
44   </footer>
45 );
46 }
47
48 export default HomeFooter;
```

5.6 Admin

5.6.1 Admin

El componente Admin muestra la página de inicio del administrador, contiene subcomponentes (AdminNavigation, AdminMain, AdminFooter), los cuales nos permite visualizar las diferentes partes de la página de inicio del administrador. Hacemos uso de los routers de React para subnavegar entre las páginas.

```
1  import HomeFooter from "../Home/HomeFooter";
2  import AdminMain from "../AdminMain";
3  import AdminList from "../AdminList";
4  import AdminNavigation from "../AdminNavigation";
5  import { useLocation, Outlet, Route, Routes } from 'react-router-dom';
6  import AdminCourse from "../AdminCourse";
7  import AdminCourseCard from "../AdminCourseCard";
8  import { useEffect, useState } from "react";
9
10 function Admin() {
11   const location = useLocation();
12   const user = location.state;
13   const [isDarkMode, setIsDarkMode] = useState(false);
14
15   useEffect(() => {
16     if (isDarkMode) {
17       document.documentElement.classList.add('dark');
18     } else {
19       document.documentElement.classList.remove('dark');
20     }
21   }, [isDarkMode]);
22   return (
23     <div className="flex flex-col min-h screen
24       dark:bg-primary-dark dark:text-text-light"
25     >
26       <AdminNavigation isDarkMode={isDarkMode} setIsDarkMode={setIsDarkMode}/>
27       <main className="flex-grow min-h-[60vh]
28     ">
29         <Routes>
30           <Route index element={<AdminMain />} />
31           <Route path="users" element={<AdminList />} />
32           <Route path="courses" element={<AdminCourseCard />} />
33           <Route path="courses/create" element={<AdminCourse />} />
34         </Routes>
35       </main>
36       <Outlet />
37       <HomeFooter />
38     </div>
39   );
40 }
41 export default Admin;
```

5.6.2 AdminNavigation

El componente AdminNavigation muestra la barra de navegación de la página de inicio del administrador, contiene un menú de navegación con enlaces a las diferentes secciones de la página, nuestro usuario y un botón de cierre de sesión.

```

1  import React, { useState } from "react";
2  import { useUser } from "../../components/useContext";
3  import Button from "../Home/Button";
4
5  function AdminNavigation({ isDarkMode, setIsDarkMode }) {
6    const { user, logout } = useUser();
7    const [isMenuOpen, setIsMenuOpen] = useState(false);
8
9    return (
10     <header className="
11     dark:bg-primary-light
12     text-primary-dark dark:text-text-light py-6 px-6 shadow-sm flex justify-between items-center w-full z-1000">
13       <div className="flex items-center gap-4">
14         <a href="/" className="flex items-center">
15           <span className="text-xl md:text-2xl font-bold text-text-dark dark:text-text-light">
16             <i className="text-3xl text-text-dark dark:text-text-light bx bx-code-alt"></i> CoderDojo
17           </span>
18         </a>
19       </div>
20       <nav className={`flex-col md:flex-row md:flex gap-8 items-center ${isMenuOpen ? 'flex' : 'hidden'}
21       md:flex absolute md:relative top-24 md:top-0 right-0 md:left-auto
22       px-8 py-4 md:py-0 md:px-0
23       md:w-auto bg-menu-light dark:bg-menu-dark md:bg-transparent
24       md:dark:bg-transparent
25       animate-open-menu md:animate-none
26       z-50 md:z-auto`>
27         <a href="#" className="font-semibold hover:border-b-2 border-primary-light dark:border-text-dark">Cursos
28         <a href="#" className="font-semibold hover:border-b-2 border-primary-light dark:border-text-dark">Usuarios
29         <div className="flex items-center gap-2">
30           <i className="text-gray-700 rounded-full bg-gray-300 bx bx-user p-2"></i>
31           <span>Bienvenido, Admin</span>
32         </div>
33       </nav>
34       <div className="flex items-center gap-4">
35         <Button onClick={logout}
36
37           text="Cerrar Sesión" icon="bx bx-log-out" url="/" />
38         <button
39           onClick={() => setIsDarkMode(!isDarkMode)}
40
41         >
42           <i className={`w-8 h-8 text-2xl
43           text-primary-light dark:text-text-light
44           ${isDarkMode ? 'bx bx-sun' : 'bx bx-moon'}}`></i>
45           <span className="sr-only">Toggle dark mode</span>
46         </button>
47         <button
48           className="md:hidden"
49           onClick={() => setIsMenuOpen(!isMenuOpen)}
50         >
51           <i className="bx bx-menu text-3xl"></i>

```

```
52     </button>
53   </div>
54 </header>
55 );
56 }
57
58 export default AdminNavigation;
```

5.6.3 AdminMain

El componente AdminMain muestra la sección principal de la página de inicio del administrador, donde se muestra dos secciones (Users, Courses), los cuales a través del botón de ver nos redirige a la lista de usuarios y cursos correspondiente. Hacemos dos solicitudes a `http://localhost:8000/system/user/list/` y `http://localhost:8000/system/course/list/` para obtener la lista de usuarios y cursos.

```
1  import { useEffect, useState } from "react";
2  import AdminUsersCard from "../AdminUsersCard";
3  import { Link, useNavigate } from "react-router-dom";
4  import axios from "axios";
5  import { useUser } from "../../components/useContext";
6  function AdminMain() {
7    const user = useUser().user;
8    const [users, setUsers] = useState([]);
9    const [courses, setCourses] = useState([]);
10   const navigate = useNavigate();
11
12   useEffect(() => {
13     const fetchUsers = async () => {
14       const token = localStorage.getItem('access_token');
15       if (user.tipo !== 'AD') {
16         navigate('access-denied/');
17       } else {
18         try {
19           const response = await axios.get('http://localhost:8000/system/user/list/', {
20             headers: {
21               Authorization: `Bearer ${token}`,
22             },
23           });
24           setUsers(response.data);
25         } catch (error) {
26           console.error('Error al obtener los usuarios:', error);
27           navigate('/admin');
28         }
29       }
30     };
31
32     fetchUsers();
33   }, [user, navigate]);
34
35   const handleGetUsers = () => {
36     navigate('/admin/users', { state: { users } });
37   }
38
39   const handleGetCourses = async () => {
40     try {
41       const response = await axios.get('http://localhost:8000/system/course/list/');
42       setCourses(response.data);
```

```

43     console.log("Users", users);
44     navigate('/admin/courses', { state: { courses: response.data, users } });
45   } catch (error) {
46     console.error('Error al obtener los cursos:', error);
47     navigate('/admin');
48   }
49 };
50
51 return (
52   <main className="flex-1">
53     <section className="container mx-auto py-12 px-6">
54       <div className="space-y-4">
55         <h1 className="text-3xl font-bold">Panel de Administración</h1>
56         <p className="text-gray-500">Bienvenido, Admin.</p>
57         <div className="grid grid-cols-1 md:grid-cols-2 gap-8">
58           <h2 className="text-2xl font-bold col-span-full">Administrar Usuarios</h2>
59
60           <AdminUsersCard title="Usuarios" info={'Agrega, edita y elimina usuarios de la plataforma.'}>
61             <button onClick={handleGetUsers}
62               className="inline-flex items-center gap-2
63                 bg-primary-light dark:bg-primary-dark
64                 text-white px-4 py-2 rounded-md hover:bg-[#0a2a54]"
65             >
66               Ver Usuarios
67             </button>
68           </AdminUsersCard>
69           <AdminUsersCard title="Cursos" info={'Agrega, edita y elimina cursos de la plataforma.'}>
70             <button onClick={handleGetCourses}
71               className="inline-flex items-center gap-2
72                 bg-primary-light dark:bg-primary-dark
73                 text-white px-4 py-2 rounded-md hover:bg-[#0a2a54]"
74             >
75               Ver Cursos
76             </button>
77             <Link to='courses/create/'>
78               <button
79                 className="inline-flex items-center gap-2 bg-primary-light dark:bg-primary-dark text-white px-4 py-2 rounded-md hover:bg-[#0a2a54]"
80               >
81                 Crear Curso
82               </button>
83             </Link>
84           </AdminUsersCard>
85         </div>
86       </div>
87     </section>
88   </main>
89 )
90 }
91 export default AdminMain;
92

```

5.6.4 AdminList

El componente AdminList muestra la lista de usuarios en forma de una tabla, donde cada fila contiene un icon edit o remove, el cual nos permite editar o eliminar un usuario. Tenemos dos métodos handleSubmitEdit y handleRemove, el cual nos permite editar o eliminar un usuario, haciendo uso de las solici-

tudes `http://localhost:8000/system/user/${editingUser.id}/` y `http://localhost:8000/system/user/${userId}/`.

```

1  import { Link, useLocation } from 'react-router-dom';
2  import Card from './Utilities/';
3  import { useState } from 'react';
4  import axios from 'axios';
5
6  function AdminList() {
7    const location = useLocation();
8    const [users, setUsers] = useState(location.state.users);
9    const [editingUser, setEditingUser] = useState(null);
10   const [formData, setFormData] = useState({ name: '', email: '', role: '', password: '' });
11   const [showForm, setShowForm] = useState(false);
12
13   const handleEditClick = (user) => {
14     setEditingUser(user);
15     setFormData({ name: user.name, email: user.email, role: user.tipo, password: '' }); // Resetea la contraseña
16     setShowForm(true); // Muestra el formulario al editar
17   };
18
19   const handleDelete = async (userId) => {
20     try {
21       await axios.delete(`http://localhost:8000/system/user/${userId}/`);
22       // Actualiza la lista de usuarios después de eliminar
23       location.state.users = users.filter(user => user.id !== userId);
24       setUsers(users.filter(user => user.id !== userId));
25     } catch (error) {
26       console.error('Error al eliminar el usuario:', error);
27     }
28   };
29
30   const handleSubmitEdit = async (e) => {
31     e.preventDefault();
32     try {
33       const response = await axios.put(`http://localhost:8000/system/user/${editingUser.id}/`, formData);
34       const updatedUser = response.data;
35       // Actualiza la lista de usuarios
36       const updatedUsers = users.map(user => (user.id === updatedUser.id ? updatedUser : user));
37       setUsers(users.map(user => (user.id === updatedUser.id ? updatedUser : user)));
38       location.state.users = updatedUsers;
39       setEditingUser(null);
40       setShowForm(false); // Oculta el formulario después de guardar
41     } catch (error) {
42       console.error('Error al editar el usuario:', error);
43     }
44   };
45
46   return (
47     <section className="container mx-auto py-12 px-6">
48       <div className="space-y-4">
49         <Link className="border-spacing-1 hover:border-b-black" to="/admin">
50           <i className="bx bx-arrow-back"></i> Regresar
51         </Link>
52         <h2 className="text-2xl font-bold">Lista de Usuarios</h2>
53
54         {showForm && (
55           <div className="shadow-md p-3 rounded-md dark:bg-menu-dark md:w-[60vw]">

```

```

56 <h3 className="text-lg font-semibold">Editar Usuario</h3>
57 <form onSubmit={handleSubmitEdit} className="mb-4 grid md:grid-cols-4 gap-3">
58   <label htmlFor="name">Nombre</label>
59   <input
60     className='bg-menu-light py-1 px-2 rounded-2xl'
61     type="text"
62     name="name"
63     value={formData.name}
64     onChange={(e) => setFormData({ ...formData, name: e.target.value })}
65     placeholder="Nombre"
66     required
67   />
68   <label htmlFor="email">Correo</label>
69   <input
70     className='bg-menu-light py-1 px-2 rounded-2xl'
71     type="email"
72     name="email"
73     value={formData.email}
74     onChange={(e) => setFormData({ ...formData, email: e.target.value })}
75     placeholder="Correo"
76     required
77   />
78   <label htmlFor="role">Rol</label>
79   <select
80     className='bg-menu-light py-1 px-2 rounded-2xl'
81     name="role"
82     value={formData.role}
83     onChange={(e) => setFormData({ ...formData, role: e.target.value })}
84     required
85   >
86     <option className="pr-2 px-2 rounded-2xl" value="AD">Admin</option>
87     <option value="TC">Profesor</option>
88     <option value="ST">Estudiante</option>
89   </select>
90   <label htmlFor="password">Contraseña</label>
91   <input
92     className='bg-menu-light py-1 px-2 rounded-2xl'
93     type="password"
94     name="password"
95     value={formData.password}
96     onChange={(e) => setFormData({ ...formData, password: e.target.value })}
97     placeholder="Contraseña (dejar vacío si no se cambia)"
98   />
99   <div className='flex flex-col md:flex-row gap-2'>
100     <button className="bg-primary-light rounded-lg p-2 text-text-light" type="submit">Guardar</button>
101     <button
102       type="button"
103       className="bg-primary-light rounded-lg p-2 text-text-light"
104       onClick={() => {
105         setFormData({ name: '', email: '', role: '', password: '' });
106         setEditingUser(null);
107         setShowForm(false);
108       }}
109     >
110       Cancelar
111     </button>

```

```

112     </div>
113   </form>
114 </div>
115 })
116
117 {users.length === 0 ? (
118   <p className="text-gray-500">No hay usuarios disponibles.</p>
119 ) : (
120   <Card>
121     <div className="overflow-x-auto">
122       <table className="min-w-full bg-white">
123         <thead>
124           <tr className='hover:bg-gray-50'>
125             <th className="py-3 px-4 text-left text-gray-500 font-medium">Nombre</th>
126             <th className="py-3 px-4 text-left text-gray-500 font-medium">Correo</th>
127             <th className="py-3 px-4 text-left text-gray-500 font-medium">Rol</th>
128             <th className="py-3 px-4 text-left text-gray-500 font-medium">Acciones</th>
129           </tr>
130         </thead>
131         <tbody className='text-text-dark'>
132           {users.map(user => (
133             <tr key={user.id} className="border-b border-gray-200 hover:bg-gray-50">
134               <td className="py-3 px-4">{user.name}</td>
135               <td className="py-3 px-4">{user.email}</td>
136               <td className="py-3 px-4">{user.tipo}</td>
137               <td>
138                 <div className="flex items-center gap-2">
139                   <button className="hover:text-blue-500" onClick={() => handleEditClick(user)}>
140                     <i className="bx bx-edit"></i>
141                   </button>
142                   <button className="hover:text-red-500" onClick={() => handleDelete(user.id)}>
143                     <i className="bx bx-trash"></i>
144                   </button>
145                 </div>
146               </td>
147             </tr>
148           )})}
149         </tbody>
150       </table>
151     </div>
152   </Card>
153 })
154 </div>
155 </section>
156 );
157 }
158
159 export default AdminList;

```

5.6.5 AdminCourseCard

El componente AdminCourseCard muestra la lista de cursos en forma de una tarjeta, donde cada tarjeta contiene dos botones edit y remove, los cuales nos van a permitir editar o eliminar un curso. Tenemos tres métodos searchTeacher, handleDeleteCourse y handleSubmitEdit, el primero nos permite listar a los profesores, el segundo nos permite eliminar un curso y el tercero editar, todo esto haciendo uso de las solicitudes [http://localhost:8000/system/course/\\$id/](http://localhost:8000/system/course/$id/) y [http://localhost:8000/system/course/\\$editingCourse.id/](http://localhost:8000/system/course/$editingCourse.id/)

```
1 import React from 'react';
2 import Card from './Utilities';
3 import { Link, useLocation } from 'react-router-dom';
4 import axios from 'axios';
5 import { useState } from 'react';
6
7 function AdminCourseCard() {
8   const location = useLocation();
9   console.log(location.state);
10  const users = location.state.users;
11  const [courses, setCourses] = useState(location.state.courses);
12  const [editingCourse, setEditingCourse] = useState(null);
13  const [formData, setFormData] = useState({ nombre: '', descripcion: '', docente: '' });
14  const [showForm, setShowForm] = useState(false);
15
16  const searchTeacher = (id) => {
17    const teacher = users.find(user => user.id === id);
18    return teacher ? `${teacher.name}` : 'No asignado';
19  }
20
21  const handleEditClick = (course) => {
22    setEditingCourse(course);
23    setFormData({ nombre: course.nombre, descripcion: course.descripcion, docente: course.docente });
24    setShowForm(true);
25  };
26
27  const handleDeleteCourse = async (id) => {
28    console.log(courses);
29    console.log(id);
30    try {
31      await axios.delete(`http://localhost:8000/system/course/${id}/`);
32      location.state.courses = courses.filter(course => course.id !== id);
33      setCourses(courses.filter(course => course.id !== id));
34      alert('Curso eliminado exitosamente');
35    } catch (error) {
36      console.error('Error al eliminar el curso:', error);
37    }
38  }
39
40  const handleSubmitEdit = async (e) => {
41    e.preventDefault();
42    try {
43      const response = await axios.put(`http://localhost:8000/system/course/${editingCourse.id}/`, formData);
44      const updatedCourse = response.data;
45      const updatedCourses = courses.map(course => (course.id === updatedCourse.id ? updatedCourse : course));
46      setCourses(courses.map(course => (course.id === updatedCourse.id ? updatedCourse : course)));
47      location.state.courses = updatedCourses;
48      setEditingCourse(null);
49      setShowForm(false);
50    } catch (error) {
51      console.error('Error al editar el curso:', error);
52    }
53  }
54
55  return (
56    <section className="container mx-auto py-12 px-6">
```



```

57 <div className="space-y-4">
58   <Link className='border-spacing-1 hover:border-b-black' to='/admin'>
59     <i className='bx bx-arrow-back'></i> Regresar
60   </Link>
61   <h2 className="text-2xl font-bold">Lista de Cursos</h2>
62
63   {showForm && (
64     <div className='shadow-md p-3 rounded-md dark:bg-menu-dark md:w-[70vw]'>
65       <h3 className="text-lg font-semibold">Editar Curso</h3>
66       <form onSubmit={handleSubmitEdit} className="mb-4 grid md:grid-cols-4 gap-3">
67         <label htmlFor="nombre" className="font-medium">Nombre</label>
68         <input
69           id="nombre"
70           className="bg-menu-light dark:text-menu-dark py-1 px-2 rounded-2xl"
71           type="text"
72           name="nombre"
73           value={formData.nombre}
74           onChange={(e) => setFormData({ ...formData, nombre: e.target.value })}
75           required
76         />
77
78         <label htmlFor="descripcion" className="font-medium">Descripción</label>
79         <textarea
80           id="descripcion"
81           className="bg-menu-light py-1 px-2 rounded-2xl dark:text-menu-dark"
82           name="descripcion"
83           value={formData.descripcion}
84           onChange={(e) => setFormData({ ...formData, descripcion: e.target.value })}
85           required
86         />
87
88         <label htmlFor="docente" className="font-medium">Docente</label>
89         <select
90           id="docente"
91           className="bg-menu-light dark:text-menu-dark py-1 px-2 rounded-2xl"
92           name="docente"
93           value={formData.docente}
94           onChange={(e) => setFormData({ ...formData, docente: e.target.value })}
95           required
96         >
97           <option value="" disabled>Seleccionar</option>
98           {users.filter(user => user.tipo === 'TC').map(user => (
99             <option key={user.id} value={user.id}>{user.name}</option>
100           ))}
101         </select>
102
103         <button
104           type="submit"
105           className="bg-primary-light dark:bg-primary-dark text-text-light dark:text-text-light py-2 px-4"
106           >Guardar
107         </button>
108         <button
109           onClick={() => setShowForm(false)}
110           className="bg-primary-light dark:bg-primary-dark text-text-light dark:text-text-light py-2 px-4"
111           >Cancelar
112         </button>

```

```

113     </form>
114   </div>
115   )}
116
117   {courses.length === 0 ? (
118     <p className="text-muted-foreground dark:text-muted-foreground">No hay cursos registrados</p>
119   ) : (
120     <div className="grid grid-cols-1 md:grid-cols-2 gap-8">
121       {courses.map(course => (
122         <Card key={course.id}>
123           <div className="flex flex-col justify-between">
124             <h1 className="text-xl font-bold text-primary-light dark:text-primary-dark">
125               {course.nombre}
126             </h1>
127             <p className="text-sm text-muted-foreground dark:text-muted-foreground">
128               Profesor: {searchTeacher(course.docente)}
129             </p>
130           </div>
131           <p className="font-medium">
132             {course.descripcion}
133           </p>
134           <div className="flex justify-end gap-2">
135             <button
136               onClick={() => handleEditClick(course)}
137               className="bg-primary-light dark:bg-primary-dark text-text-light dark:text-text-light py-2 p
138             <i className="bx bx-edit mr-1"></i>
139               Editar
140             </button>
141             <button
142               onClick={() => handleDeleteCourse(course.id)}
143               className="bg-primary-light dark:bg-primary-dark text-text-light dark:text-text-light py-2 p
144             <i className="bx bx-trash mr-1"></i>
145               Eliminar
146             </button>
147           </div>
148         </Card>
149       )})}
150     </div>
151   )}
152 </div>
153 </section>
154 );
155 }
156
157 export default AdminCourseCard;

```

5.6.6 AdminCourse

El componente AdminCourse muestra el formulario para la creación de un curso, contiene un formulario con los campos name, description, teacher, el cual nos permite crear un curso, haciendo uso de la solicitud `http://localhost:8000/system/course/create/` y para filtrar a los profesor `http://localhost:8000/system/teacher/list/`.

```

1 import Card from './Utilities/';
2 import { Link } from 'react-router-dom';
3 import axios from 'axios';
4 import { useEffect, useState } from 'react';
5 function AdminCourse() {

```

```

6  const [title, setTitle] = useState('');
7  const [description, setDescription] = useState('');
8  const [teacher, setTeacher] = useState('');
9  const [teachers, setTeachers] = useState([]);
10
11  useEffect(() => {
12    const getTeachers = async () => {
13      try {
14        const response = await axios.get('http://localhost:8000/system/teacher/list/');
15        setTeachers(response.data);
16      } catch (error) {
17        console.log(error);
18      }
19    }
20    getTeachers();
21  }, []);
22
23  const handleSubmit = async (e) => {
24    e.preventDefault();
25    try {
26      const response = await axios.post('http://localhost:8000/system/course/create/', {
27        nombre: title,
28        descripcion: description,
29        docente: teacher
30      });
31      if (response.status === 201) {
32        alert('Curso creado correctamente');
33      }
34    } catch (error) {
35      if (error.response.status === 400) {
36        alert(error.response.data.non_field_errors[0]);
37      } else {
38        alert('Error al crear el curso');
39      }
40    }
41  }
42  return (
43    <section className="container mx-auto py-12 px-6">
44      <Link className="border-spacing-1 hover:border-b-black" to="/admin">
45        <i className="bx bx-arrow-back"></i> Regresar
46      </Link>
47      <form onSubmit={handleSubmit}>
48        <div className="space-y-4">
49          <div className="flex items-center justify-between">
50            <h1 className="text-3xl font-bold">Agregar Curso</h1>
51          </div>
52          <p className="text-muted-foreground dark:text-text-light">
53            Llenar el formulario para agregar un nuevo curso a tu plataforma Coder Dojo.
54          </p>
55
56          <div className="shadow-md p-3 rounded-md dark:bg-menu-dark md:w-[50vw]">
57            <div className="grid gap-3 mb-4">
58              <label htmlFor="title" className="block text-sm font-medium dark:text-text-light">Titulo</label>
59              <input
60                id="title"
61                value={title}

```

```

62         onChange={(e) => setTitle(e.target.value)}
63         placeholder="Ingrese el título del curso"
64         className="bg-menu-light dark:text-menu-dark py-1 px-2 rounded-2xl"
65         required
66     />
67 </div>
68 <div className="grid gap-3 mb-4">
69     <label htmlFor="description" className='block text-sm font-medium dark:text-text-light
70     '>Descripción</label>
71     <textarea
72         id="description"
73         value={description}
74         onChange={(e) => setDescription(e.target.value)}
75         placeholder="Ingrese la descripción del curso"
76         rows={3}
77         className="bg-menu-light py-1 px-2 rounded-2xl dark:text-menu-dark"
78         required
79     />
80 </div>
81 <div className="grid gap-3 mb-4">
82     <label htmlFor="teacher" className='block text-sm font-medium dark:text-text-light'>Profesor</label>
83     <select
84         id="teacher"
85         value={teacher}
86         onChange={(e) => setTeacher(e.target.value)}
87         placeholder="Ingrese el nombre del teacher"
88         className="bg-menu-light dark:text-menu-dark py-1 px-2 rounded-2xl"
89         required
90     >
91         <option value="" disabled>Seleccione un teacher</option>
92         {teachers.map((profesor) => (
93             <option key={profesor.id} value={profesor.id}>{profesor.name}</option>
94         ))}
95     </select>
96 </div>
97 <div className='flex justify-end mt-2'>
98     <button
99         type="submit"
100         className="bg-primary-light rounded-lg p-2 text-text-light hover:bg-[#0a2a54]"
101     >
102         <i className='bx bx-save'></i> Guardar
103     </button>
104 </div>
105 </div>
106 </div>
107 </form>
108 </section>
109 );
110 }
111
112
113 export default AdminCourse;

```

5.7 Teacher

5.7.1 Teacher

El componente Teacher muestra la página de inicio del profesor, contiene subcomponentes (TeacherNavigation, TeacherMain, TeacherFooter), los cuales nos permite visualizar las diferentes partes de la página de inicio del profesor. Hacemos uso de los routers de React para subnavegar entre las páginas.

```
1
2 import {Outlet, Route, Routes } from 'react-router-dom';
3
4 import TeacherNavigation from './TeacherNavigation';
5 import TeacherMain from './TeacherMain';
6 import TeacherCourse from './TeacherCourse';
7 import HomeFooter from './Home/HomeFooter';
8 import { useLocation } from 'react-router';
9 import { useEffect, useState } from 'react';
10
11
12 function Teacher() {
13   const location = useLocation();
14   const user = location.state;
15   const [isDarkMode, setIsDarkMode] = useState(false);
16
17   useEffect(() => {
18     if (isDarkMode) {
19       document.documentElement.classList.add('dark');
20     } else {
21       document.documentElement.classList.remove('dark');
22     }
23   }, [isDarkMode]);
24
25   return (
26     <div className="flex flex-col min-h screen">
27       <TeacherNavigation isDarkMode={isDarkMode} setIsDarkMode={setIsDarkMode}/>
28       <main className="flex-grow min-h-[80vh]">
29         <Routes>
30           <Route index element={<TeacherMain/>} />
31           <Route path="course/:id" element={<TeacherCourse />} />
32         </Routes>
33       </main>
34       <Outlet />
35       <HomeFooter />
36     </div>
37   );
38 }
39 export default Teacher;
```

5.7.2 TeacherNavigation

El componente TeacherNavigation muestra la barra de navegación de la página de inicio del profesor, contiene un menú de navegación con enlaces a las diferentes secciones de la página, nuestro usuario y un botón de cierre de sesión.

```
1 import React, { useState } from "react";
2 import { useNavigate, Link } from "react-router-dom";
3 import { useUser } from "../../components/useContext";
```

```

4  import Button from "../Home/Button";
5
6  function AdminNavigation({ isDarkMode, setIsDarkMode }) {
7      const { user, logout } = useUser();
8      const [isMenuOpen, setIsMenuOpen] = useState(false);
9
10     return (
11         <header className="
12         dark:bg-primary-light
13         text-primary-dark dark:text-text-light py-6 px-6 shadow-sm flex justify-between items-center w-full z-1000"
14         <div className="flex items-center gap-4">
15             <a href="/" className="flex items-center">
16                 <span className="text-xl md:text-2xl font-bold text-text-dark dark:text-text-light">
17                     <i className="text-3xl text-text-dark dark:text-text-light bx bx-code-alt"></i> CoderDojo
18                 </span>
19             </a>
20         </div>
21         <nav className={`flex-col md:flex-row md:flex gap-8 items-center ${isMenuOpen ? 'flex' : 'hidden'}
22         md:flex absolute md:relative top-24 md:top-0 right-0 md:left-auto
23         px-8 py-4 md:py-0 md:px-0
24         md:w-auto bg-menu-light dark:bg-menu-dark md:bg-transparent
25         md:dark:bg-transparent
26         animate-open-menu md:animate-none
27         z-50 md:z-auto`
28         >
29             <a href="#" className="font-semibold
30             border-b-2 border-transparent
31             hover:border-primary-light
32             dark:hover:border-text-light
33             ">Cursos</a>
34             <a href="#" className="font-semibold
35             border-b-2 border-transparent
36             hover:border-primary-light
37             dark:hover:border-text-light
38             ">Usuarios</a>
39             <a className="flex items-center gap-2">
40                 <i className="text-gray-700 rounded-full bg-gray-300 bx bx-user p-2"></i>
41                 <span>Bienvenido, {user.name}</span>
42             </a>
43         </nav>
44         <div className="flex items-center gap-4">
45             <Button onClick={logout}
46
47             text="Cerrar Sesión" icon="bx bx-log-out" url="/" />
48             <button
49                 onClick={() => setIsDarkMode(!isDarkMode)}
50             >
51                 <i className={`w-8 h-8 text-2xl
52                 text-primary-light dark:text-text-light
53                 ${isDarkMode ? 'bx bx-sun' : 'bx bx-moon'}}`></i>
54                 <span className="sr-only">Toggle dark mode</span>
55             </button>
56             <button
57                 className="md:hidden"
58                 onClick={() => setIsMenuOpen(!isMenuOpen)}
59             >

```

```
60     <i className="bx bx-menu text-3xl"></i>
61     </button>
62   </div>
63 </header>
64 );
65 }
66
67 export default AdminNavigation;
```

5.7.3 TeacherMain

El componente TeacherMain muestra la sección principal de la página de inicio del profesor, donde se nos muestra la lista de cursos que enseña, los cuales a través del botón de ver nos redirige a ver el curso de forma más detallada. Hacemos una solicitud a `http://localhost:8000/teacher/course/${user.id}/` para obtener la lista de cursos.

```
1  import { useEffect, useState } from "react";
2  import axios from "axios";
3  import TeacherCoursesCard from "../TeacherCoursesCard";
4  import { useUser } from "../../components/useContext";
5
6  function TeacherMain() {
7    const { user } = useUser()
8    const [cursos, setCursos] = useState([])
9
10   useEffect(() => {
11     const getCursos = async () => {
12       try {
13         const response = await axios.get(`http://localhost:8000/system/teacher/course/${user.id}/`)
14         setCursos(response.data)
15       } catch (error) {
16         console.error('Error al obtener los cursos:', error)
17       }
18     }
19     getCursos()
20   }, [user.id], cursos)
21
22   return (
23     <main className="flex-1">
24       <section className="container mx-auto py-12 px-6">
25         <div className="space-y-4">
26           <h1 className="text-3xl font-bold">Panel de Grupos</h1>
27           <p className="text-gray-500">Bienvenido Profesor/a, {user.name}.</p>
28           <div className="grid grid-cols-1 md:grid-cols-2 gap-8">
29             {cursos.length === 0 ? (
30               <h2 className="text-2xl font-bold">Usted no tiene cursos</h2>
31             ) : (
32               <>
33                 <h2 className="text-2xl font-bold col-span-full">Cursos</h2>
34                 <div className="grid gap-3">
35                   {cursos.map((curso) => (
36                     <TeacherCoursesCard
37                       key={curso.id}
38                       user={user}
39                       course={curso}
40                     />
41                   ))}
```

```

42     </div>
43   </>
44   })
45 </div>
46 </div>
47 </section>
48 </main>
49 )
50 }
51 export default TeacherMain;

```

- **TeacherCourse:** El componente TeacherCourse muestra la lista de tareas, nos permite crear una asignar una tareas, hacemos uso del subcomponente task. Hacemos solicitudes [http://localhost:8000/system/course/\\$course.id/task/list/](http://localhost:8000/system/course/$course.id/task/list/) [http://localhost:8000/system/course/\\$course.id/task/assign/](http://localhost:8000/system/course/$course.id/task/assign/)

```

1  import { useLocation, Link } from "react-router-dom"
2  import { useUser } from "../../components/useContext"
3  import TeacherCreateTask from "../TeacherCreateTask";
4  import axios from "axios";
5  import { useEffect, useState } from "react";
6  import { Task } from "../Task";
7
8  function TeacherCourse() {
9    const { user } = useUser();
10   const location = useLocation();
11   const { course } = location.state;
12   const [tasks, setTasks] = useState([]);
13
14   useEffect(() => {
15     const getTasks = async () => {
16       try {
17         const response = await axios.get(`http://localhost:8000/system/course/${course.id}/task/list/`);
18         console.log(response.data);
19         setTasks(response.data); // Guarda las tareas en el estado
20       } catch (error) {
21         alert('Error al cargar las tareas');
22       }
23     };
24     getTasks();
25   }, [course.id]);
26
27   const handleAssignTask = async (taskId) => {
28     try {
29       const response = await axios.post(`http://localhost:8000/system/course/${course.id}/task/assign/`, {
30         alert('Tarea asignada correctamente');
31         console.log(response.data);
32         window.location.reload();
33       }
34     } catch (error) {
35       alert('Error al asignar la tarea');
36     }
37   }
38
39   if (!course) {
40     return <div><h1 className="text-2xl">Loading...</h1></div>;
41   }

```



```

42
43   return (
44     <main className="flex-1">
45       <section className="container mx-auto py-12 px-6">
46         <Link className='border-spacing-1 hover:border-b-black' to='/teacher'><i><i className='bx bx-arrow-
47         <div className="space-y-4">
48           <div className="grid gap-3 md:grid-cols-2">
49             <h1 className="text-3xl font-bold">Curso: {course.nombre}</h1>
50             <button className="w-[200px] bg-primary-light hover:bg-primary-dark text-white font-bold py-2 p
51           </div>
52           <p className="text-gray-500">Profesor: {user.name}</p>
53           <p className="text-gray-500">Descripción: {course.descripcion}</p>
54         </div>
55         <TeacherCreateTask cursoId={course.id} />
56       </section>
57       <section className="container mx-auto py-12 px-6">
58         <h1 className="text-3xl font-bold">Tareas No Asignadas</h1>
59         <div className="grid gap-4 mt-6">
60           {tasks.filter(task => !task.asignada).map((task) => (
61             <Task key={task.id} task={task} handleAssignTask={handleAssignTask} />
62           ))}
63         </div>
64         <h1 className="text-3xl font-bold">Tareas Asignadas</h1>
65         <div className="grid gap-4 mt-6">
66           {tasks.filter(task => task.asignada).map((task) => (
67             <Task key={task.id} task={task} handleAssignTask={handleAssignTask} />
68           ))}
69         </div>
70       </section>
71     </main>
72   )
73 }
74
75 export default TeacherCourse

```

- **Task:** El componente Task nos permite ver la entregas de esa tarea al momento de asignarla, así como calificar las entregas. Todo esto a través de las solicitudes `http://localhost:8000/system/course/task/deliveries/` y `http://localhost:8000/system/course/task/deliveries/grade/`

```

1  import axios from "axios"
2  import { useState } from "react"
3  // eslint-disable-next-line react/prop-types
4  export function Task({ task = {}, handleAssignTask }) {
5
6     const [entregas, setEntregas] = useState([]);
7     const [grade, setGrade] = useState(0);
8
9     const loadDeliveries = async (taskId) => {
10       try {
11         const response = await axios.post(`http://localhost:8000/system/course/task/deliveries/`, { task_id:
12         console.log(response.data);
13         setEntregas(response.data);
14       } catch (error) {
15         alert('Error al cargar las entregas');
16       }
17     }

```

```

18
19  const submitGrade = async (deliveryId, grade) => {
20    console.log(deliveryId, grade);
21    try {
22      const response = await axios.post(`http://localhost:8000/system/course/task/deliveries/grade/`, { del
23      console.log(response.data);
24      window.location.reload();
25    } catch (error) {
26      alert('Error al calificar la entrega');
27    }
28  }
29
30  return <div className="bg-gray-200 p-4 rounded-lg" key={task.id}>
31    <h2 className="text-2xl font-bold">{task.nombre}</h2>
32    <p className="text-gray-500">{task.descripcion}</p>
33    <p className="text-gray-500">Fecha de entrega: {task.fecha_entrega}</p>
34    {task.asignada ? (
35      <div>
36        <button
37          onClick={() => loadDeliveries(task.id)}
38          className="bg-primary-light hover:bg-primary-dark text-white font-bold py-2 px-4 rounded">Cargar
39        </button>
40      </div>
41      {entregas.length !== 0 && (
42        <div>
43          <h3>Entregas</h3>
44          <div className="grid gap-4 mt-6">
45            {entregas.length !== 0 && entregas.map((entrega) => (
46              <div className="bg-white dark:bg-gray-800 rounded-md shadow-md p-4" key={entrega.id}>
47                <h2 className="text-xl font-bold text-primary-light dark:text-primary-dark">{entrega.es
48                <label>Ver Entrega: </label>
49                <div className="bg-gray-300 rounded-lg px-2 py-1">
50                  {entrega.enlace.length !== 0 ? (
51                    <a href={entrega.enlace} target="_blank" rel="noreferrer" className="text-primary-l
52                    ">{entrega.enlace}
53                    </a>) : (<p className="text-primary-light dark:text-primary-dark">No hay enlace</p>
54                  </div>
55                <div className="flex gap-2 items-center">
56                  <label>Nota: </label>
57                  <input className="text-center
58                    font-bold text-primary-light dark:text-primary-dark
59                    border-gray-300 border-2 rounded-lg px-2 py-1 w-20 my-1
60                    "
61                    onChange={(e) => setGrade(e.target.value)}
62                    type="text" placeholder="0/20" />
63                  <button
64                    className="
65                      bg-primary-light
66                      text-text-light
67                      px-3 py-1 rounded-lg
68                      hover:bg-primary-dark
69                      flex items-center gap-2
70                    "
71                    onClick={() => submitGrade(entrega.id, grade)} >{entrega.calificacion?"Calificado":
72                  </div>
73                </div>

```

```

74         })}
75         </div>
76       </div>
77     })
78   </div>
79 </div>
80 ) : (
81   <button
82     onClick={() => handleAssignTask(task.id)}
83     className="bg-primary-light hover:bg-primary-dark text-white font-bold py-2 px-4 rounded">Asignar T
84   </button>
85   </div>
86 </div>
87 }

```

5.8 Student

5.8.1 Student

El componente Student muestra la página de inicio del estudiante, contiene subcomponentes (StudentNavigation, StudentMain, StudentFooter), los cuales nos permite visualizar las diferentes partes de la página de inicio del estudiante. Hacemos uso de los routers de React para subnavegar entre las páginas.

```

1  import React from 'react';
2  import { Outlet, Route, Routes } from 'react-router-dom';
3  import StudentNavigation from './StudentNavigation';
4  import StudentMain from './StudentMain';
5  import HomeFooter from './Home/HomeFooter';
6  import { useEffect, useState } from 'react';
7  import { useLocation } from 'react-router';
8
9  function Student(){
10    const location = useLocation();
11    const user = location.state;
12    const [isDarkMode, setIsDarkMode] = useState(false);
13
14    useEffect(() => {
15      if (isDarkMode) {
16        document.documentElement.classList.add('dark');
17      } else {
18        document.documentElement.classList.remove('dark');
19      }
20    }, [isDarkMode]);
21
22    return (
23      <div className="flex flex-col min-h screen">
24        <StudentNavigation isDarkMode={isDarkMode} setIsDarkMode={setIsDarkMode}/>
25        <main className="flex-grow min-h-[80vh]">
26          <Routes>
27            <Route index element={<StudentMain/>} />
28            <Route path="course/:id" element={<StudentNavigation />} />
29          </Routes>
30        </main>
31        <Outlet />
32        <HomeFooter />
33      </div>

```

```
34     );
35   }
36
37   export default Student;
```

5.8.2 StudentNavigation

El componente StudentNavigation muestra la barra de navegación de la página de inicio del estudiante, contiene un menú de navegación con enlaces a las diferentes secciones de la página, nuestro usuario y un botón de cierre de sesión.

```
1  import React, { useState } from 'react';
2  import { Link } from 'react-router-dom';
3  import { useUser } from "../../components/useContext";
4  import Button from '../Home/Button';
5
6  function StudentNavigation({isDarkMode, setIsDarkMode}) {
7    const { user, logout } = useUser();
8    const [isMenuOpen, setIsMenuOpen] = useState(false);
9
10   return (
11     <header className="
12       dark:bg-primary-light
13       text-primary-dark dark:text-text-light py-6 px-6 shadow-sm flex justify-between items-center w-full z-1000"
14       <div className="flex items-center gap-4">
15         <a href="/" className="flex items-center">
16           <span className="text-xl md:text-2xl font-bold text-text-dark dark:text-text-light">
17             <i className="text-3xl text-text-dark dark:text-text-light bx bx-code-alt"></i> CoderDojo
18           </span>
19         </a>
20       </div>
21       <nav className={`flex-col md:flex-row md:flex gap-8 items-center ${isMenuOpen ? 'flex' : 'hidden'}
22         md:flex absolute md:relative top-24 md:top-0 right-0 md:left-auto
23         px-8 py-4 md:py-0 md:px-0
24         md:w-auto bg-menu-light dark:bg-menu-dark md:bg-transparent
25         md:dark:bg-transparent
26         animate-open-menu md:animate-none
27         z-50 md:z-auto`
28       >
29         <a href="#" className="font-semibold hover:border-b-2 border-primary-light dark:border-text-dark">Cursos
30         <a href="#" className="font-semibold hover:border-b-2 border-primary-light dark:border-text-dark">Tareas
31       <div className="flex items-center gap-2">
32         <i className="text-gray-700 rounded-full bg-gray-300 bx bx-user p-2"></i>
33         <span>Bienvenido, {user.name}</span>
34       </div>
35     </nav>
36     <div className="flex items-center gap-4">
37       <Button onClick={logout}
38
39         text="Cerrar Sesión" icon="bx bx-log-out" url="/" />
40     <button
41       onClick={() => setIsDarkMode(!isDarkMode)}
42     >
43       <i className={`w-8 h-8 text-2xl
44         text-primary-light dark:text-text-light
45         ${isDarkMode ? 'bx bx-sun' : 'bx bx-moon'}}`></i>
46       <span className="sr-only">Toggle dark mode</span>
```

```

47     </button>
48     <button
49       className="md:hidden"
50       onClick={() => setIsMenuOpen(!isMenuOpen)}
51     >
52       <i className="bx bx-menu text-3xl"></i>
53     </button>
54   </div>
55 </header>
56 );
57 }
58 export default StudentNavigation;

```

5.8.3 StudentMain

El componente StudentMain funciona como un dashboard, donde se muestra todo los cursos (Inscritos o Inscribirse), los cursos inscritos como tal, tareas asignadas de todos los cursos y tareas enviadas de todos los cursos. Para esto hacemos uso de las solicitudes `http://localhost:8000/system/course/list/` `http://localhost:8000/system/student/enroll/${courseId}/${user.id}/` (inscribirse a un curso), `http://localhost:8000/system/st` etc.

```

1  import { useEffect, useState } from 'react';
2  import axios from 'axios';
3  import { useUser } from '../components/useContext';
4
5  function StudentMain() {
6    const { user } = useUser()
7    const [cursos, setCursos] = useState([])
8    const [misCursos, setMisCursos] = useState([])
9    const [entregasAsignadas, setEntregasAsignadas] = useState([])
10   const [url, setUrl] = useState('')
11   const [entregasEnviadas, setEntregasEnviadas] = useState([])
12
13   useEffect(() => {
14     const getCursos = async () => {
15       try {
16         const response = await axios.get(`http://localhost:8000/system/course/list/`)
17         setCursos(response.data)
18       } catch (error) {
19         console.error('Error al obtener los cursos:', error)
20       }
21     }
22     getCursos()
23   }, [])
24
25   const enrollCourse = async (courseId) => {
26     console.log('Enroll course:', courseId)
27     try {
28       const response = await axios.post(`http://localhost:8000/system/student/enroll/${courseId}/${user.id}/`)
29       console.log("Cambios " + response.data)
30       window.location.reload();
31     } catch (error) {
32       console.error('Error al unirse al curso:', error)
33     }
34   }
35
36   const statusEnroll = (courseId) => {

```

```
37     const course = misCursos.find((curso) => curso.id === courseId)
38     return course ? 'Inscrito' : 'Inscribirse'
39 }
40
41
42 useEffect(() => {
43     const getMyCourses = async () => {
44         try {
45             const response = await axios.get(`http://localhost:8000/system/student/${user.id}/courses/`)
46             setMisCursos(response.data)
47         } catch (error) {
48             console.error('Error al obtener mis cursos:', error)
49         }
50     }
51     getMyCourses()
52 }, [user.id])
53
54 useEffect(() => {
55     const getAssignedTasks = async () => {
56         try {
57             const response = await axios.get(`http://localhost:8000/system/student/${user.id}/assigned_tasks/`, { us
58             console.log("Asignadas", response.data)
59             setEntregasAsignadas(response.data)
60         } catch (error) {
61             console.error('Error al obtener las tareas asignadas:', error)
62         }
63     }
64
65     const getSubmittedTasks = async () => {
66         try {
67             const response = await axios.get(`http://localhost:8000/system/student/${user.id}/submitted_tasks/`)
68             console.log("Entregas", response.data)
69             setEntregasEnviadas(response.data)
70         } catch (error) {
71             console.error('Error al obtener las entregas:', error)
72         }
73     }
74
75     getAssignedTasks()
76     getSubmittedTasks()
77 }, [user.id])
78
79 const handleSubmitDelivery = async (taskId, url) => {
80     if (!url) {
81         alert('Por favor, ingrese un enlace de entrega')
82         return
83     }
84     try {
85         const response = await axios.post(`http://localhost:8000/system/student/${user.id}/delivery/`, { tarea: ta
86         alert('Entrega enviada correctamente')
87         console.log(response.data)
88         window.location.reload();
89     } catch (error) {
90         console.error('Error al enviar la entrega:', error)
91     }
92 }
```

```

93
94 const changeColor = (grade) =>{
95     if(grade >= 0 && grade <= 10){
96         return 'text-red-500'
97     } else if(grade > 10 && grade <= 15){
98         return 'text-yellow-500'
99     } else if(grade > 15 && grade <= 20){
100         return 'text-green-500'
101     }
102 }
103
104 return (
105     <main className="flex-1">
106         <section className="container mx-auto py-12 px-6">
107             <div className="space-y-4">
108                 <h1 className="text-3xl font-bold">Panel de Estudio</h1>
109                 <p className="text-gray-500">Bienvenido Estudiante, {user.name}</p>
110                 <div className="grid grid-cols-1 md:grid-cols-2 gap-8">
111                     <section>
112                         <h1>Todos los cursos</h1>
113                         <div className="grid gap-3">
114                             {cursos.map((curso) => (
115                                 <div className="bg-white dark:bg-gray-800 rounded-md shadow-md p-4" key={curso.id}>
116                                     <h2 className="text-xl font-bold text-primary-light dark:text-primary-dark">{curso.nombre}</h2>
117                                     <p className="text-sm text-muted-foreground dark:text-muted-foreground">Profesor: {curso.id}</p>
118                                     <p className="font-medium">{curso.descripcion}</p>
119                                     <div className="flex justify-end">
120                                         <button
121                                             onClick={() => enrollCourse(curso.id)}
122
123                                             className={` ${statusEnroll(curso.id) !== 'Inscrito' ? 'bg-primary-light dark:bg-primary-
124                                             >
125                                             <i className="bx bx-edit mr-1"></i>
126                                             {statusEnroll(curso.id)}
127                                         </button>
128                                     </div>
129                                 </div>
130                             ))}
131                         </div>
132                     </section>
133                     <section>
134                         <h1>Mis Cursos</h1>
135                         <div className="grid md:grid-cols-2 gap-3 ">
136                             {misCursos.map((curso) => (
137                                 <div className="bg-card-opt dark:bg-gray-800 rounded-md shadow-md p-4" key={curso.id}>
138                                     <h2 className="text-xl font-bold text-primary-light dark:text-primary-dark">{curso.nombre}</h2>
139                                     <p className="text-sm text-muted-foreground dark:text-muted-foreground">Profesor: {curso.doc}</p>
140                                     <p className="font-medium">{curso.descripcion}</p>
141                                 </div>
142                             ))}
143                         </div>
144                     </section>
145                     <section>
146                         <h1>Tareas Asignadas</h1>
147                         <div className="grid gap-3">
148                             {entregasAsignadas.map((entrega) => (

```

```

149     <div className="bg-white dark:bg-gray-800 rounded-md shadow-md p-4" key={entrega.id}>
150       <h2 className="text-xl font-bold text-primary-light dark:text-primary-dark">{entrega.tarea.n
151       <p className="text-sm text-muted-foreground dark:text-muted-foreground">Curso: {entrega.tare
152       <p className="font-medium">{entrega.tarea.descripcion}</p>
153       <p className="font-medium">Fecha de entrega: {new Date(entrega.tarea.fecha_entrega).toLocale
154       <input
155         type="url"
156         placeholder="Enlace de la entrega"
157         onChange={(e) => setUrl(e.target.value)}
158         className="block mt-2 p-2 border border-gray-300 rounded"
159       />
160       <button
161         onClick={() => { handleSubmitDelivery(entrega.tarea.id, url) }}
162         className='bg-primary-light dark:bg-primary-dark rounded-lg text-text-light px-4 py-2 mt-2
163       >
164         <i className="bx bx-upload mr-1"></i>
165         Enviar entrega
166       </button>
167     </div>
168   )}
169 </div>
170 </section>
171 <section>
172   <h1>Tareas Enviadas</h1>
173   <div className="grid gap-3">
174     {entregasEnviadas.map((entrega) => (
175       <div className="bg-white dark:bg-gray-800 rounded-md shadow-md p-4" key={entrega.id}>
176         <h2 className="text-xl font-bold text-primary-light dark:text-primary-dark">{entrega.tarea.n
177         <p className="text-sm text-muted-foreground dark:text-muted-foreground">Curso: {entrega.tare
178         <p className="font-medium">Fecha de entrega: {new Date(entrega.tarea.fecha_entrega).toLocale
179         <p className="font-medium">Entrega: <a className="text-primary-light"
180           href={entrega.enlace} target="_blank" rel="noreferrer">{entrega.enlace}</a></p>
181         <p className={changeColor(entrega.calificacion)}>Calificación: {entrega.calificacion}</p>
182       </div>
183     ))}
184   </div>
185 </section>
186 </div>
187 </div>
188 </section>
189 </main >
190 )
191 }
192
193 export default StudentMain;

```

5.9 App

El componente App es el componente principal de la aplicación, contiene el componente UserProvider, el cual nos permite acceder a las rutas de acuerdo al tipo de usuario. Hacemos uso de los routers de React para la navegación entre las páginas.

```

1 import { BrowserRouter as Router, Routes, Route } from 'react-router-dom'
2 //Home
3 import Home from './designUI/Home/Home'
4 //Register and Login
5 import RegisterUser from './components/RegisterUser'

```



```
6 import LoginUser from './components/LoginUser'
7 //Admin
8 import Admin from './designUI/Admin/Admin'
9 import AccessDenied from './components/AccessDenied'
10 import Teacher from './designUI/Teacher/Teacher'
11 import { UserProvider } from './components/useContext'
12 import PrivateRoute from './components/PrivateRoute'
13 import Student from './designUI/Student/Student'
14 function App() {
15   return (
16     <UserProvider>
17       <Router>
18         <Routes>
19           <Route path="/" element={<Home />} />
20           <Route path="register/" element={<RegisterUser />} />
21           <Route path="login/" element={<LoginUser />} />
22           <Route path="access-denied/" element={<AccessDenied />} />
23
24           <Route element={<PrivateRoute allowedRoles={['AD']} />>
25             <Route path="admin/*" element={<Admin />} />
26           </Route>
27
28           <Route element={<PrivateRoute allowedRoles={['TC']} />>
29             <Route path="teacher/*" element={<Teacher />} />
30           </Route>
31
32           <Route element={<PrivateRoute allowedRoles={['ST']} />>
33             <Route path="student/*" element={<Student />} />
34           </Route>
35
36         </Routes>
37       </Router>
38     </UserProvider>
39   )
40 }
41
42 export default App
```

5.10 Main

El componente Main nos permite envolver todo el componente App.

```
1 import React from 'react'
2 import ReactDOM from 'react-dom/client'
3 import App from './App.jsx'
4 import './index.css'
5
6 ReactDOM.createRoot(document.getElementById('root')).render(
7   <React.StrictMode>
8     <App />
9   </React.StrictMode>,
10 )
```

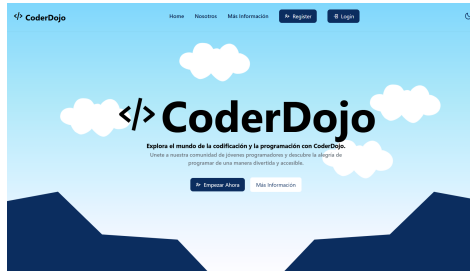
5.11 Index HTML

El archivo index.html es el archivo principal de la aplicación, contiene el punto de entrada de la aplicación, donde se renderiza el componente Main, aquí se ponen las dependencias de los iconos y

fuentes usadas.

```
1  <!doctype html>
2  <html lang="en">
3
4  <head>
5    <meta charset="UTF-8" />
6    <link rel="icon" type="image/svg+xml" href="/vite.svg" />
7    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
8    <!--Icons-->
9    <link rel="stylesheet" href="https://unpkg.com/boxicons@latest/css/boxicons.min.css">
10   <title>Vite + React</title>
11 </head>
12
13 <body>
14   <div id="root"></div>
15   <script type="module" src="/src/main.jsx"></script>
16 </body>
17
18 </html>
```

6 Pruebas



Página Principal.



Página Principal.



Página Principal en modo Responsive.

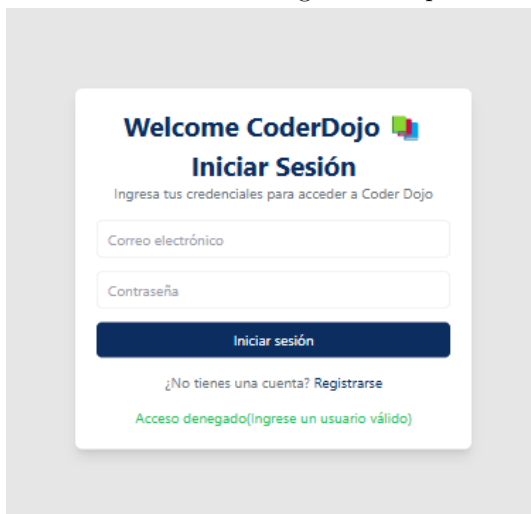


Quienes somos?

En CoderDojo, nuestra misión es empoderar a los jóvenes mediante la excelencia de la programación y habilidades tecnológicas en un entorno inclusivo y lúdico. Creemos que cada joven tiene el potencial de convertirse en un creador digital, y trabajamos para proporcionar las herramientas y el apoyo necesarios para que puedan explorar y desarrollar sus habilidades. Nuestro programa Programación para todos: CoderDojo está diseñado para ser accesible y divertido, permitiendo a los participantes sumergirse en el emocionante mundo de la codificación. Ofrecemos sesiones guiadas por apasionados mentores, que ayudan a los jóvenes a descubrir su potencial y a desarrollar su creatividad a través de proyectos prácticos y colaborativos.



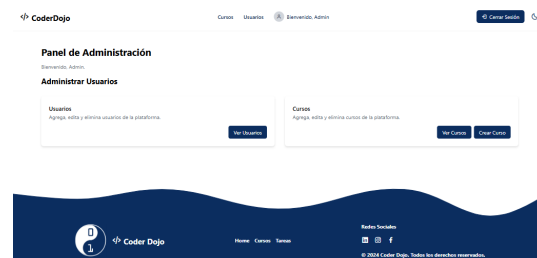
Contenido de la Pagina Principal.



Pagina de Inicio de Sesión.



Pagina de Registro.



Pagina Principal del Administrador.

[illegible]

Lista de Usuarios.

Registrar

Lista de Usuarios

Editar Usuario

Nombre

Grady

Correo

g@gmail.com

Rol

Estudiante

Contraseña

Contraseña (Mínimo 6 caracteres)

Guardar

Cancelar

Editar Usuario.

The screenshot shows the AWS IAM console 'Groups' page. On the left, the 'Groups' list includes 'Programmatic-User-1', 'Programmatic-User-2', 'Group-Programmatic', and 'Group-Programmatic-2'. The main area displays the details for 'Programmatic-User-1', showing its description, permissions (No Policy Attached), and the 'Users' tab which lists 'Programmatic-User-1'.

Lista de Cursos.

Recurso: **Lista de Cursos**

Editar Curso

Nombre: Descripción:

Código:

Programación web 2

Portador: Gestión

Reservados: Puesto 1

Programación web 1

Portador: Gestión

Reservados: 1 a la programación web 1

Editar Curso.

Regresar

Agregar Curso

Llenar el formulario para agregar un nuevo curso a tu plataforma Coder Dojo.

Título

Ingresa el título del curso

Descripción

Ingresa la descripción del curso

Profesor

Selecciona un teacher

Guardar

Crear Curso.

The screenshot shows the CoderDojo website's 'Painel de Grupos' (Groups Panel). The header features the CoderDojo logo, navigation links for 'Cursos', 'Eventos', 'Biblioteca', and 'Sobre', and a 'Cursos' button. The main content area is titled 'Painel de Grupos' and includes a sub-header 'Selecione o idioma do seu navegador'. Below this, there are two course listings. The first is 'Curso Python 2' with a description 'Um curso para quem quer aprender Python' and a 'Ver mais' button. The second is 'Curso Python 3' with a description 'Um curso para quem quer aprender Python' and a 'Ver mais' button.

Pagina Principal del Profesor.

Crear una tarea por curso específico.

Tareas No Asignadas

Tareas Asignadas

Introducción a HTML

`introduccion.html`

[Ver tarea](#)

Introducción a HTML2

`introduccion2.html`

[Ver tarea](#)

Asignar tarea a los estudiantes.

[illegible]

Lista de tareas asignadas.

CoderDojo Cursos Tareas Bienvenido, Dora [Cerrar Sesión](#)

Panel de Estudio

Bienvenido Estudiante Dora

Todos los cursos

Programación web 2

Problema 1

Requisito: Puesto 1

[Comenzar](#)

Programación web 1

Problema 1

Bienvenido a la programación web

[Comenzar](#)

Programación web 3

Problema 3

Bienvenido a la programación web 3

[Comenzar](#)

Base de Datos

Problema 3

Un curso para profundizar el estudio de datos

[Comenzar](#)

Curso Prueba 1

Problema 1

[Comenzar](#)

Curso Prueba 2

Problema 1

Ujshygrfhwethghyulshyghrfe

[Comenzar](#)

Curso Prueba 3

Problema 1

Ujshygrfhwethghyulshyghrfe

[Comenzar](#)

Tareas Asignadas

Tareas Enviadas

Dashboard del Estudiante.

Tareas Asignadas

erfdwdfgh

Curso 1

hghghghgh

Fecha de entrega: 26/7/2024, 19:00:00

[Entrar a la entrega](#)

[Enviar entrega](#)

Tareas Enviadas

Introducción a HTML

Curso 1

Fecha de entrega: 2/6/2024, 19:00:00

Entrega: 100

Calificación: 1

Introducción a HTML2

Curso 1

Fecha de entrega: 4/6/2024, 19:00:00

Entrega: <https://www.figma.com/design/8lhwPwmdBulhconRseafRtU/HomeDqjgProde-id=24-786-48hAu682wepwulsh-0>

Calificación: 20

tarea1

Curso 1

Fecha de entrega: 25/7/2024, 19:00:00

Entrega: <https://github.com/Atom20/Coder-Dojo/commit/main/>

Calificación: 10


Introducción a HTML3

Curso 1

Fecha de entrega: 25/7/2024, 19:00:00

Entrega: <https://www.figma.com/design/8lhwPwmdBulhconRseafRtU/HomeDqjgProde-id=0-186-48hAu682wepwulsh-0>

Calificación: 10

 **CoderDojo** [Home](#) [Cursos](#) [Tareas](#) [Redes Sociales](#)

© 2024 CoderDojo. Todos los derechos reservados.

Lista de tareas asignadas y enviadas.

7 URL del repositorio en GitHub

- https://github.com/Alsnj20/Coder_Dojo

8 URL del Figma

- <https://www.figma.com/design/BHv11PwnbRuHxonNeatPhi/HomeDojo?node-id=0-1&t=2rFp0cSv5894SCWR-0>

9 URL del Deploy

- <https://www.heroku.com>

10 Estructura del Trabajo Final

- El contenido que se entrega para este trabajo final es el siguiente

```
CoderDojo/
|-- backend/
|   |-- admin/
|   |-- system/ # Aqui se encuentra todo el codigo fuente
|   |-- teacher/
|   |-- student/
|   |-- manage.py
|   |-- coderdojo/
|   |-- requirements.txt
|   |-- db.sqlite3
|-- frontend/
|   |-- node_modules/
|   |-- public/
|   |-- src/ # Aqui se encuentra todo el codigo fuente
|       |-- components/
|       |-- designUI/
|       |-- App.jsx
|       |-- index.html
|       |-- main.jsx
|   |-- package.json
|   |-- vite.config.js
|   |-- tailwind.config.js
|   |-- postcss.config.js
|   |-- README.md
|   |-- .gitignore
|---Latex/
|   |-- trabajofinal.tex
|   |-- trabajofinal.pdf
|   |-- img/
|---README.md
|---.gitignore
```


11 Rúbrica

Tabla: Rúbrica para contenido del Informe y evidencias

Contenido y demostración		Puntos	Checklist	Estudiante	Profesor
1. GitHub	Repositorio se pudo clonar y se evidencia la estructura adecuada para revisar los entregables. (Se descontará puntos por error o observación)	4	×	4	
2. Commits	Hay porciones de código fuente asociado a los commits planificados con explicaciones detalladas. (El profesor puede preguntar para refrendar calificación)	4	×	4	
3. Ejecución	Se incluyen comandos para ejecuciones y pruebas del código fuente explicadas gradualmente que permitirían replicar el proyecto. (Se descontará puntos por cada omisión)	4	×	4	
4. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación)	2	×	2	
7.Ortografía	El documento no muestra errores ortográficos. (Se descontará puntos por error encontrado)	2	×	1	
8. Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente con explicaciones puntuales pero precisas, agregando diagramas generados a partir del código fuente y refleja un acabado impecable. (El profesor puede preguntar para refrendar calificación)	4	×	4	
Total		20	Completo	19	

12 Referencias

- <https://github.com/>
- <https://git-scm.com/>
- <https://www.w3schools.com/python/>