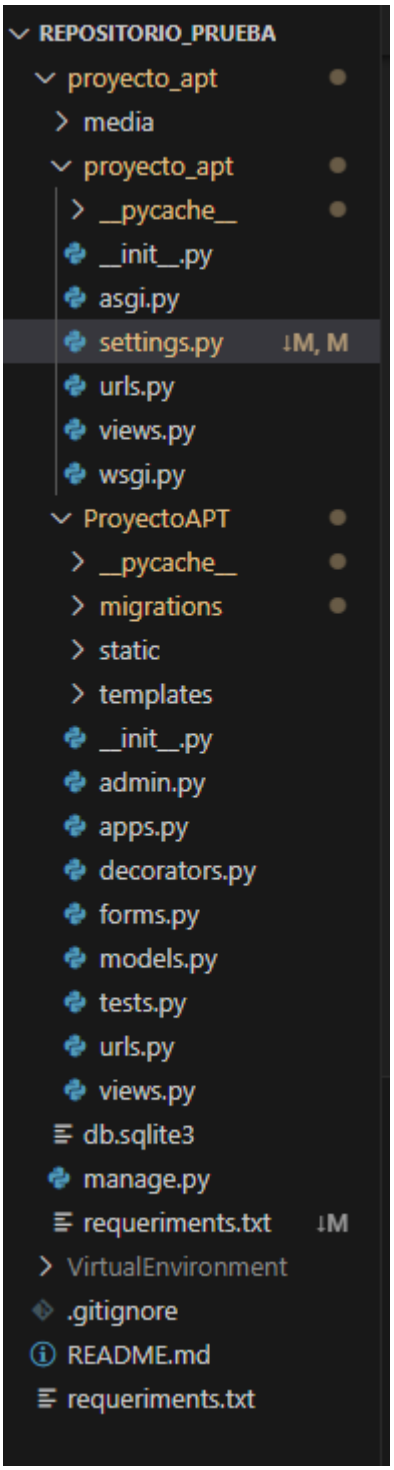
BackEnd Evidencia

Repositorio



Models.py

Tipo Usuario(superuser)

models.py

proyecto_apt > ProyectoAPT > models.py > ...

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

# Create your models here.

#TipoUsuario
class TipoUsuario(models.Model):
        id= models.BigAutoField(primary_key=True)
        nombre_tipo_usuario = models.CharField(max_length=100)
        descripcion = models.CharField(max_length=100)
        def __str__(self):
            return str(self.nombre_tipo_usuario)
        # DICT
        #2.Estudiante
        #1.Paciente

class CustomUserManager(BaseUserManager):
    def create_user(self, email, password=None, **extra_fields):
        if not email:
            raise ValueError('El usuario debe tener un email')
        email = self.normalize_email(email)
        extra_fields.setdefault('username', email.split('@')[0])  # Generar un username a partir del email
        user = self.model(email=email, **extra_fields)
        user.set_password(password)
        user.save(using=self._db)
        return user

    def create_superuser(self, email, password=None, **extra_fields):
        extra_fields.setdefault('is_staff', True)
        extra_fields.setdefault('is_superuser', True)
        extra_fields.setdefault('username', email.split('@')[0])

        if extra_fields.get('is_staff') is not True:
            raise ValueError('El superusuario debe tener is_staff=True.')
        if extra_fields.get('is_superuser') is not True:
            raise ValueError('El superusuario debe tener is_superuser=True.')

        return self.create_user(email, password, **extra_fields)
```

Tipo de tratamientos y horarios

```python
#MODELO DE USUARIO;
class customuser(AbstractUser):
    id = models.BigAutoField(primary_key=True)
    email = models.EmailField(unique=True)
    rut = models.CharField(max_length=100, unique=True)
    id_tipo_user = models.ForeignKey('TipoUsuario', on_delete=models.SET_NULL, null=True)
    descripcion = models.TextField(null=True)
    imageBlob = models.ImageField(upload_to='imagenes_usuario/', blank=True, null=True)

    USERNAME_FIELD = 'email'  # Usar email para el inicio de sesión
    REQUIRED_FIELDS = []

    objects = CustomUserManager()

    def save(self, *args, **kwargs):
        if not self.username:  # Generar un username basado en el email
            self.username = self.email.split('@')[0]
        super().save(*args, **kwargs)

    def __str__(self):
        return self.email


class tipoTratamiento(models.Model):
    id = models.BigAutoField(primary_key=True)
    nombreTratamiento= models.CharField(max_length=50)
    descripcion = models.TextField()

    def __str__(self):
        return str(self.id)

class horarios(models.Model):
    id = models.BigAutoField(primary_key=True)
    tipoTratamiento = models.ForeignKey(tipoTratamiento, on_delete = models.SET_NULL, null=True, default=None)
    inicio = models.TimeField()
    fecha_seleccionada = models.DateField()
    estudiante = models.ForeignKey(customuser, on_delete=models.SET_NULL, null=True, default=None)
    def __str__(self):
        return str(self.id)
```

Forms.py

Formulario del Registro e Inicio sesión

proyecto_apt > ProyectoAPT > forms.py > ...

```python
## USER CREATION FORM
from django.contrib.auth.forms import UserCreationForm, AuthenticationForm, SetPasswordForm, PasswordResetForm #importa autenticacion de forms
from .models import * #importa modelos
from django import forms #importa formularios
from typing import Any #Te entrega todo (preguntarle a chatgpt)
from django.utils import timezone
from datetime import datetime, timedelta, time

# Set default values to use:
inicioB = ["",time(9,0),time(9,30), time(10,0), time(10,30), time(11,0), time(11,30), time(12,0), time(12,30), time(13,0), time(13,30), time(14,0),

from django import forms
from django.contrib.auth.forms import UserCreationForm
from .models import customuser

class CustomUserCreationForm(UserCreationForm):
    class Meta:
        model = customuser
        fields = ['first_name', 'last_name', 'email', 'rut', 'id_tipo_user', 'password1', 'password2']

    def __init__(self, *args, **kwargs):
        super(CustomUserCreationForm, self).__init__(*args, **kwargs)
        self.fields['email'].widget.attrs.update({'placeholder': 'Email'})
        self.fields['first_name'].widget.attrs.update({'placeholder': 'Nombre'})
        self.fields['last_name'].widget.attrs.update({'placeholder': 'Apellido'})
        self.fields['rut'].widget.attrs.update({'placeholder': 'RUT'})
        self.fields['password1'].widget.attrs.update({'placeholder': 'Contraseña'})
        self.fields['password2'].widget.attrs.update({'placeholder': 'Confirmar Contraseña'})

class UserLoginForm(AuthenticationForm):
    username = forms.EmailField(label="Email", widget=forms.EmailInput(attrs={'class': 'form-control'}))

    def __init__(self, *args, **kwargs):
        super(UserLoginForm, self).__init__(*args, **kwargs)

    def clean_username(self):
        email = self.cleaned_data.get('username')
        if not customuser.objects.filter(email=email).exists():
            raise forms.ValidationError("No existe un usuario con este email.")
        return email

```

Calendario e Horarios

```python
class horariosForm(forms.ModelForm):
    class Meta:
        model = horarios
        fields =['tipoTratamiento',
                 'inicio',
                 'fecha_seleccionada',
                 'estudiante']

    def __init__(self, *args: Any, **kwargs):
        super(horariosForm, self).__init__(*args, **kwargs)
        #Add tipo de tratamiento
        self.fields['tipoTratamiento'] = forms.ModelChoiceField(
            queryset=tipoTratamiento.objects.all(),
            empty_label=None,
            widget=forms.Select(attrs={'class':'form-control','id':'id_tipoTratamiento',  'hidden':True})
        )
        self.fields['tipoTratamiento'].label = "Tipo de tratamiento"

        self.fields['fecha_seleccionada'] = forms.DateField(
            label="Seleccione su fecha!",
            required=True,
            widget=forms.TextInput(attrs={'class': 'form-control', 'id': 'id_fecha_seleccionada', 'type': 'date', }),

        )
        self.fields['fecha_seleccionada'].widget.attrs.update({'class': 'form-control', 'type':'date'})
        idTipoEstudiante = TipoUsuario.objects.filter(nombre_tipo_usuario='Estudiante').first()
        self.fields['estudiante'] = forms.ModelChoiceField(
            queryset=customuser.objects.filter(id_tipo_user=idTipoEstudiante), #Modificable
            empty_label=None,
            widget=forms.Select(attrs={'class':'form-control', 'hidden':True})
        )
        self.fields['estudiante'].label = "Estudiante"
        # Personalizar el label para mostrar el 'first_name'
        self.fields['estudiante'].label_from_instance = lambda obj: f"{obj.first_name} {obj.last_name}"

        self.fields['inicio'] = forms.ChoiceField(  #
            label="Hora de inicio:",
            choices=[(inicioB[i], str(inicioB[i])) for i in range(1, len(inicioB))],
            widget=forms.Select(attrs={'class': 'form-control', 'id': 'id_HorIni'}),
            required=False
        )
```

Views.py

proyecto_apt > ProyectoAPT > views.py > ...

```python
from django.shortcuts import render, redirect
from django.contrib.auth.decorators import login_required
from django.contrib.auth import login, logout, authenticate
from .decorators import user_not_authenticated
from django.contrib import messages
from .forms import CustomUserCreationForm, UserLoginForm # USERS LOGIN FORMS
from .forms import horariosForm # HORARIOS CHECK
from .models import *
from django.http import JsonResponse
from datetime import time, timedelta, datetime


def index(request):
    return render(request, 'APT/index.html')


def register(request):
    if request.method == "POST":
        form = CustomUserCreationForm(request.POST)
        if form.is_valid():
            user = form.save()
            login(request, user)
            messages.success(request, "Tu cuenta ha sido creada con éxito.")

            tipo_usuario = TipoUsuario.objects.get(id=user.id_tipo_user_id)


             # Redirigimos según el tipo de usuario
            if tipo_usuario.nombre_tipo_usuario == 'Estudiante':
                return redirect('infoestudiante')  # Redirige a la vista de infoestudiante

            elif tipo_usuario.nombre_tipo_usuario == 'Paciente':
                return redirect('index')  # Redirige a la página principal o cualquier otra página
        else:
            for error in list(form.errors.values()):
                messages.error(request, error)

    else:
        form = CustomUserCreationForm()

    return render(request, "autorizacion/registro.html", {"form": form})

```

proyecto_apt > ProyectoAPT > views.py > ...

```python
@login_required
def custom_logout(request):
    logout(request)
    messages.info(request, "Logged out successfully!")
    return redirect('/')


@user_not_authenticated
def loginUser(request):
    if request.method == "POST":
        form = UserLoginForm(request=request, data=request.POST)
        if form.is_valid():
            user = authenticate(
                email=form.cleaned_data["username"],  # El campo 'username' se utiliza como email
                password=form.cleaned_data["password"],
            )
            if user is not None:
                login(request, user)
                messages.success(request, f"Bienvenido <b>{user.email}</b>! Has iniciado sesión")

                # Redirige según el tipo de usuario
                if user.id_tipo_user and user.id_tipo_user.nombre_tipo_usuario == "Estudiante":
                    return redirect('infoestudiante')  # Redirige a la vista de estudiante
                elif user.id_tipo_user and user.id_tipo_user.nombre_tipo_usuario == "Paciente":
                    return redirect('index')  # Redirige a la vista de paciente
                else:
                    return redirect('/')  # Redirige a una vista por defecto si no se encuentra el tipo de usuario

            else:
                messages.error(request, "Credenciales inválidas.")
        else:
            for key, error in list(form.errors.items()):
                messages.error(request, error)

    form = UserLoginForm()
    return render(
        request=request,
        template_name="autorizacion/login.html",
        context={"form": form}
    )
```

```python
@login_required
def tratamientosForm(request, estudianteID):
    form = horariosForm(request.POST or None)

    context = {'form':form,'estudianteID':estudianteID}
    if request.method=='POST':
        form = horariosForm(request.POST)

        if form.is_valid():

            form.save()
        else:
            print(form.errors)
    return render(request, 'APT/horariosEstudianteTratamiento.html', context)

@login_required
def servicios(request):
    return render(request, 'APT/servicios.html')

@login_required
def calendar_est(request):
    return render(request, 'estudiante/calendario_est.html')

@login_required
def infoestudiante(request):
    return render(request, 'estudiante/infopersonal.html', {'user': request.user})

@login_required
def notifiaciones_est(request):
    return render(request, 'estudiante/notificaciones_estudiante.html')

@login_required
def pacientes_est(request):
    return render(request, 'estudiante/pacientes_estudiante.html')

@login_required
def publicacion_est(request):
    return render(request, 'estudiante/publicacion_estudiante.html')
```

Urls.py

proyecto_apt > ProyectoAPT > urls.py > ...

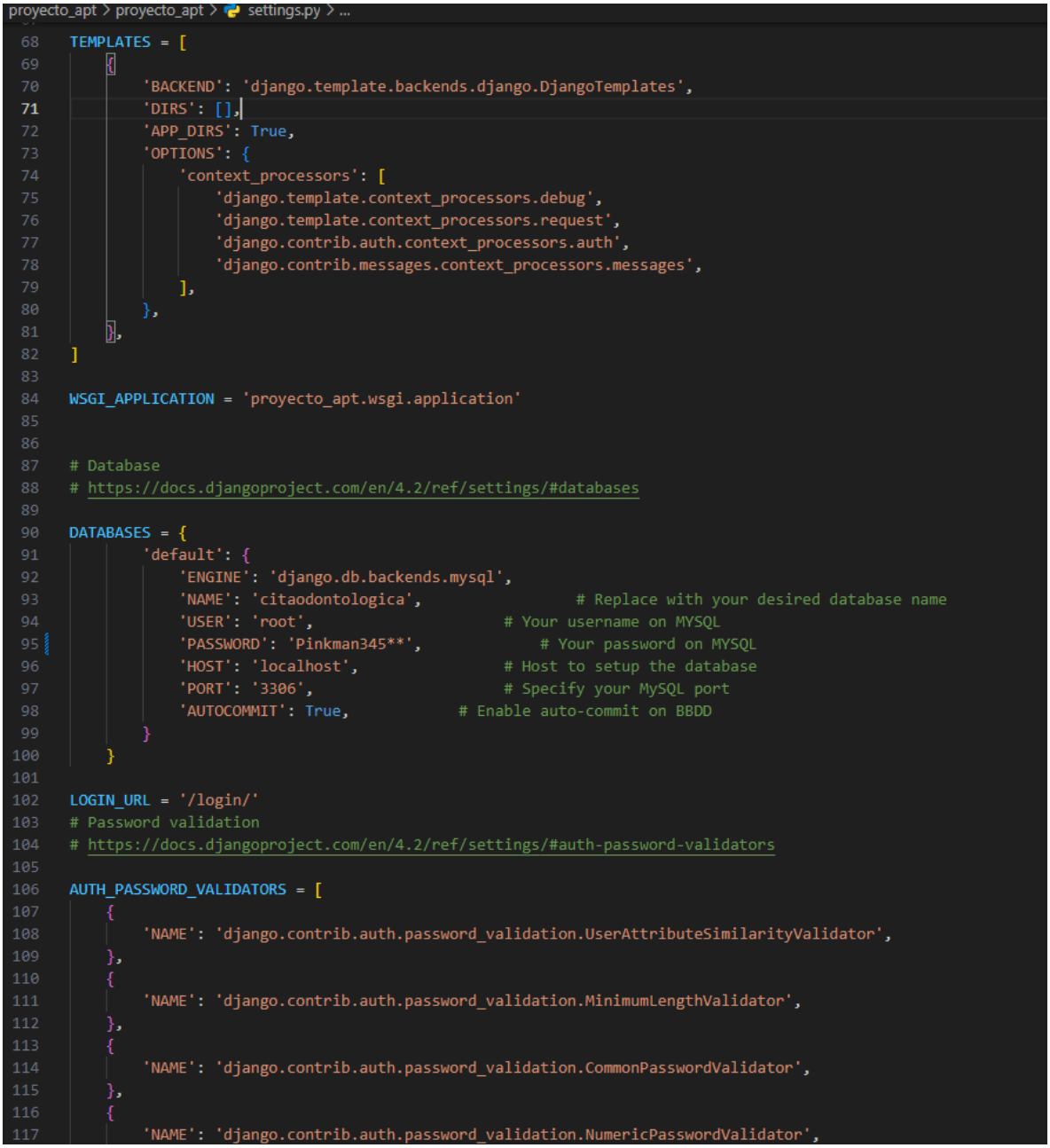
```python
from django.contrib import admin
from django.urls import path, include # Setup include to allow other app's urls to
from . import views # COMO QUE NO EXISTE? XD

from django.conf import settings
from django.conf.urls.static import static


urlpatterns = [
    path('', views.index, name='index'),
    # Horarios
    path('Horarios', views.registroHoras, name='horarios'),
    path('Horarios/<int:estudianteID>', views.tratamientosForm, name='tratamientosEstudiante'),
    # There we will enable the horario LIST
    path('obtener-horarios-disponibles/', views.obtener_horarios_disponibles, name='obtener_horarios_disponibles'),



    # AUTH
    path('login/', views.loginUser, name='login'),
    path('registro/', views.register, name='registro'),
    path('logout/', views.custom_logout, name='logout'),

    #Servicios
    path('servicios/', views.servicios, name='servicios'),

    # Añadir a subPage de Estudiante....
    path('estudiante/infopersonal/', views.infoestudiante, name="infoestudiante"),
    path('estudiante/notificaciones_estudiante/', views.notifiaciones_est, name="notificaciones"),
    path('estudiante/pacientes_estudiante/', views.pacientes_est, name="pacientes_est"),
    path('estudiante/publicacion_estudiante/', views.publicacion_est, name="publicacion_est"),
    path('estudiante/calendario_est/',  views.calendar_est, name="calendario"),



]
# Solo en modo de desarrollo
if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Admin.py

proyecto_apt > ProyectoAPT > admin.py > ...

```python
from django.contrib import admin
from .models import *

# Register your models here.



class usuariosAdmin(admin.ModelAdmin):
    # full_name= ('first_name'.join('last_name'))
    list_display = ['id', 'rut', 'getTipoUsuario', 'username', 'getfullname', 'getEmail', 'isSuperAdmin', 'descripcion', 'imageBlob']
    def getfullname(self, obj):
        return f"{obj.first_name} {obj.last_name}"
    def getTipoUsuario(self, obj):
        return f"{obj.id_tipo_user}"
    def getEmail(self, obj):
        return f"{obj.email}"
    def isSuperAdmin(self, obj):
        return f"{obj.is_superuser}"
    getEmail.short_description = 'E-mail'
    getTipoUsuario.short_description = 'Tipo de usuario'
    getfullname.short_description = 'Nombre del usuario'
    isSuperAdmin.short_description = 'Super admin?'

admin.site.register(customuser,usuariosAdmin)

class TipoUsuarioAdmin(admin.ModelAdmin): #Esto entrega 2 tipos de usuario
    fields = ["id", "nombre_tipo_usuario", "descripcion"]

admin.site.register(TipoUsuario,TipoUsuarioAdmin)

class TipoUsuarioTratamiento(admin.ModelAdmin):
    fields= ['nombreTratamiento', 'descripcion']
admin.site.register(tipoTratamiento,TipoUsuarioTratamiento)

class HorariosAdmin(admin.ModelAdmin):
    fields= ['tipoTratamiento',
             'inicio',
             'fecha_seleccionada',
             'estudiante']
admin.site.register(horarios,HorariosAdmin)
```

Settings.py


proyecto_apt > proyecto_apt > settings.py > ...
TEMPLATES = [
{
'BACKEND': 'django.template.backends.django.DjangoTemplates',
'DIRS': [],
'APP_DIRS': True,
'OPTIONS': {
'context_processors': [
'django.template.context_processors.debug',
'django.template.context_processors.request',
'django.contrib.auth.context_processors.auth',
'django.contrib.messages.context_processors.messages',
],
},
},
]
WSGI_APPLICATION = 'proyecto_apt.wsgi.application'
# Database
# https://docs.djangoproject.com/en/4.2/ref/settings/#databases
DATABASES = {
'default': {
'ENGINE': 'django.db.backends.mysql',
'NAME': 'citaodontologica', # Replace with your desired database name
'USER': 'root', # Your username on MYSQL
'PASSWORD': 'Pinkman345**', # Your password on MYSQL
'HOST': 'localhost', # Host to setup the database
'PORT': '3306', # Specify your MySQL port
'AUTOCOMMIT': True, # Enable auto-commit on BBDD
}
}
LOGIN_URL = '/login/'
# Password validation
# https://docs.djangoproject.com/en/4.2/ref/settings/#auth-password-validators
AUTH_PASSWORD_VALIDATORS = [
{
'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
},
{
'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
},
{
'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
},
{
'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',