**Folder Proyecto, se llama: la.**

**Archivo setting.py, tiene los siguientes códigos:**

from pathlib import Path

import pymysql

pymysql.install\_as\_MySQLdb()

# Build paths inside the project like this: BASE\_DIR / 'subdir'.

BASE\_DIR = Path(\_\_file\_\_).resolve().parent.parent

# Quick-start development settings - unsuitable for production

# See https://docs.djangoproject.com/en/4.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!

SECRET\_KEY = 'django-insecure-c6\*k\*ur7w2q&g2&csw06ovh1mq9)e9h1@=n3@c=!27-^r3gpvk'

# SECURITY WARNING: don't run with debug turned on in production!

DEBUG = True

ALLOWED\_HOSTS = []

# Application definition

INSTALLED\_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

    'client',

    'core',

    'lead',

    'team',

    'dashboard',

    'userprofile',

]

MIDDLEWARE = [

    'django.middleware.security.SecurityMiddleware',

    'django.contrib.sessions.middleware.SessionMiddleware',

    'django.middleware.common.CommonMiddleware',

    'django.middleware.csrf.CsrfViewMiddleware',

    'django.contrib.auth.middleware.AuthenticationMiddleware',

    'django.contrib.messages.middleware.MessageMiddleware',

    'django.middleware.clickjacking.XFrameOptionsMiddleware',

]

ROOT\_URLCONF = 'la.urls'

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',

                'team.context\_processors.active\_team',

            ],

        },

    },

]

WSGI\_APPLICATION = 'la.wsgi.application'

# Database

# https://docs.djangoproject.com/en/4.2/ref/settings/#databases

DATABASES = {

    'default': {

        'ENGINE': 'django.db.backends.mysql',

        'NAME': 'crm\_la',

        'USER': 'crm\_la',

        'PASSWORD': 'admin',

        'OPTIONS': {

            'sql\_mode': 'STRICT\_TRANS\_TABLES',

        },

    }

}

# 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',

    },

]

# Internationalization

# https://docs.djangoproject.com/en/4.2/topics/i18n/

LANGUAGE\_CODE = 'en-us'

TIME\_ZONE = 'UTC'

USE\_I18N = True

USE\_TZ = True

STATIC\_URL = 'static/'

STATICFILES\_DIRS = [

    BASE\_DIR / "static",

]

MEDIA\_ROOT = BASE\_DIR / 'media'

MEDIA\_URL = 'media/'

LOGIN\_REDIRECT\_URL = 'index'

DEFAULT\_AUTO\_FIELD = 'django.db.models.BigAutoField'

**Archivo urls.py, tiene los siguientes códigos:**

from django.conf import settings

from django.conf.urls.static import static

from django.contrib import admin

from django.contrib.auth import views

from django.contrib.auth import views as auth\_views

from django.urls import path, include

from core.views import index, about

from userprofile.forms import LoginForm

urlpatterns = [

    path("", index, name="index"),

    path("dashboard/leads/", include('lead.urls')),

    path("dashboard/clients/", include('client.urls')),

    path("dashboard/teams/", include('team.urls')),

    path('dashboard/userprofile/', include('userprofile.urls')),

    path("dashboard/", include('dashboard.urls')),

    path("about/", about, name="about"),

    path("log-in/", views.LoginView.as\_view(template\_name='userprofile/login.html', authentication\_form=LoginForm), name='login'),

    path("log-out/", views.LogoutView.as\_view(next\_page='index'), name='logout'),

    path("admin/", admin.site.urls),

] + static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

**Mi Folder de aplicación, se llama: lead.**

**Archivos:**

**admin.py**

from django.contrib import admin

from .models import Lead, Comment, LeadFile

admin.site.register(Lead)

admin.site.register(Comment)

admin.site.register(LeadFile)

**forms.py:**

from django import forms

from .models import Lead, Comment, LeadFile

WIDGET\_ATTRS = {"class": "w-full py-4 px-6 rounded-xl bg-gray-100 mb-2"}

class AddLeadForm(forms.ModelForm):

    name = forms.CharField(

        label="Nombre",

        widget=forms.TextInput(attrs=WIDGET\_ATTRS),

    )

    email = forms.EmailField(

        label="Correo Electrónico",

        widget=forms.TextInput(attrs=WIDGET\_ATTRS),

    )

    description = forms.CharField(

        label="Descripción",

        widget=forms.Textarea(attrs={"rows": "5", "class": "mb-2 w-full bg-gray-100 rounded-xl"}),

    )

    priority = forms.ChoiceField(

        label="Prioridad",

        choices=Lead.CHOICES\_PRIORITY,

        widget=forms.Select(attrs=WIDGET\_ATTRS),

    )

    status = forms.ChoiceField(

        label="Estado",

        choices=Lead.CHOICES\_STATUS,

        widget=forms.Select(attrs=WIDGET\_ATTRS),

    )

    class Meta:

        model = Lead

        fields = ("name", "email", "description", "priority", "status")

class AddCommentForm(forms.ModelForm):

    content = forms.CharField(widget=forms.Textarea(attrs={"rows": "5", "class": "w-full bg-gray-100 rounded-xl"}))

    class Meta:

        model = Comment

        fields = ['content']

class AddFileForm(forms.ModelForm):

    class Meta:

        model = LeadFile

        fields = ('file',)

**models.py:**

from django.contrib.auth.models import User

from django.db import models

from team.models import Team

class Lead(models.Model):

    LOW= 'low'

    MEDIUM = 'medium'

    HIGH = 'high'

    CHOICES\_PRIORITY = (

        (LOW, 'Bajo'),

        (MEDIUM, 'Medio'),

        (HIGH, 'Alto'),

    )

    NEW = 'Nuevo'

    CONTEACTED = 'Contactado'

    WON = 'Ganador'

    LOST = 'Perdedor'

    CHOICES\_STATUS = (

        (NEW, 'Nuevo'),

        (CONTEACTED, 'Contactado'),

        (WON, 'Ganador'),

        (LOST, 'Perdedor'),

    )

    team = models.ForeignKey(Team, related\_name='leads', on\_delete=models.CASCADE)

    name = models.CharField(max\_length=50)

    email = models.EmailField()

    description = models.TextField(max\_length=300, blank=True, null=True)

    priority = models.CharField(max\_length=10, choices=CHOICES\_PRIORITY, default=MEDIUM)

    status = models.CharField(max\_length=10, choices=CHOICES\_STATUS, default=NEW)

    converted\_to\_client = models.BooleanField(default=False)

    created\_by = models.ForeignKey(User, related\_name='lead', on\_delete=models.CASCADE)

    created\_at = models.DateTimeField(auto\_now\_add=True)

    modified\_at = models.DateTimeField(auto\_now=True)

    class Meta:

        ordering = ('name',)

    def \_\_str\_\_(self):

        return self.name

class LeadFile(models.Model):

    team = models.ForeignKey(Team, related\_name='lead\_files', on\_delete=models.CASCADE)

    lead = models.ForeignKey(Lead, related\_name='files', on\_delete=models.CASCADE)

    file = models.FileField(upload\_to='leadfiles/')

    created\_by = models.ForeignKey(User, related\_name='lead\_files', on\_delete=models.CASCADE)

    created\_at = models.DateTimeField(auto\_now\_add=True)

    def \_\_str\_\_(self):

        return self.created\_by.username

class Comment(models.Model):

    team = models.ForeignKey(Team, related\_name='lead\_comments', on\_delete=models.CASCADE)

    lead = models.ForeignKey(Lead, related\_name='comments', on\_delete=models.CASCADE)

    content = models.TextField(blank=True, null=True)

    created\_by = models.ForeignKey(User, related\_name='lead\_comments', on\_delete=models.CASCADE)

    created\_at = models.DateTimeField(auto\_now\_add=True)

    def \_\_str\_\_(self):

        return self.created\_by.username

**Views.py:**

import csv

from django.contrib import messages

from django.contrib.auth.mixins import LoginRequiredMixin

from django.http import HttpResponse

from django.shortcuts import redirect, get\_object\_or\_404

from django.urls import reverse\_lazy

from django.views.generic import (ListView, DetailView, DeleteView, UpdateView, CreateView, View,)

from .forms import AddCommentForm, AddFileForm, AddLeadForm

from .models import Lead

from client.models import Client, Comment as ClientComment

from team.models import Team

def leads\_export(request):

    leads = Lead.objects.filter(created\_by=request.user)

    response = HttpResponse(

        content\_type='text/csv',

        headers={'Content-Disposition': 'attachment; filename="clients.csv"'},

    )

    # Cambiar el delimitador a punto y coma

    writer = csv.writer(response, delimiter=';')

    # Escribir la fila de encabezado

    writer.writerow(['Lead', 'Description', 'Created at', 'Created by'])

    # Escribir datos de clientes

    for lead in leads:

        writer.writerow([lead.name, lead.description, lead.created\_at, lead.created\_by])

    return response

class LeadListView(LoginRequiredMixin, ListView):

    model = Lead

    def get\_queryset(self):

        queryset = super(LeadListView, self).get\_queryset()

        return queryset.filter(created\_by=self.request.user, converted\_to\_client=False)

class LeadDetailView(LoginRequiredMixin, DetailView):

    model = Lead

    def get\_context\_data(self, \*\*kwargs):

        context = super().get\_context\_data(\*\*kwargs)

        context["form"] = AddCommentForm()

        context['fileform'] = AddFileForm

        return context

    def get\_queryset(self):

        queryset = super(LeadDetailView, self).get\_queryset()

        team = self.request.user.userprofile.active\_team

        return queryset.filter(team=team, pk=self.kwargs.get('pk'))

class LeadDeleteView(LoginRequiredMixin, DeleteView):

    model = Lead

    success\_url = reverse\_lazy("leads:list")

    def get\_queryset(self):

        queryset = super(LeadDeleteView, self).get\_queryset()

        team = self.request.user.userprofile.active\_team

        messages.success(self.request, 'El prospecto ha sido eliminado exitosamente.')  # Mensaje que se mostrará

        return queryset.filter(team=team, pk=self.kwargs.get('pk'))

    def get(self, request, \*args, \*\*kwargs):

        return self.post(request, \*args, \*\*kwargs)

class LeadUpdateView(LoginRequiredMixin, UpdateView):

    model = Lead

    form\_class = AddLeadForm

    success\_url = reverse\_lazy("leads:list")

    def get\_context\_data(self, \*\*kwargs):

        context = super().get\_context\_data(\*\*kwargs)

        context["title"] = "Edit lead"

        return context

    def form\_valid(self, form):

        messages.success(self.request, 'El prospecto ha sido actualizado exitosamente.')

        return super().form\_valid(form)

    def get\_queryset(self):

        queryset = super(LeadUpdateView, self).get\_queryset()

        team = self.request.user.userprofile.active\_team

        return queryset.filter(team=team, pk=self.kwargs.get('pk'))

class LeadCreateView(LoginRequiredMixin, CreateView):

    model = Lead

    form\_class = AddLeadForm  # Utiliza tu formulario personalizado aquí

    success\_url = reverse\_lazy("leads:list")

    def get\_context\_data(self, \*\*kwargs):

        context = super().get\_context\_data(\*\*kwargs)

        team = self.request.user.userprofile.get\_active\_team()

        context["team"] = team

        context["title"] = "Agregar Cliente Potencial"

        return context

    def form\_valid(self, form):

        form.instance.created\_by = self.request.user

        form.instance.team = self.request.user.userprofile.get\_active\_team()

        messages.success(self.request, "Cliente potencial creado exitosamente!")

        return super().form\_valid(form)

class AddFileView(LoginRequiredMixin, View):

    def post(self, request, \*args, \*\*kwargs):

        pk = kwargs.get('pk')

        form = AddFileForm(request.POST, request.FILES)

        if form.is\_valid():

            file = form.save(commit=False)

            file.team = self.request.user.userprofile.get\_active\_team()

            file.lead\_id = pk

            file.created\_by = request.user

            file.save()

        return redirect('leads:detail', pk=pk)

class AddCommentView(LoginRequiredMixin, View):

    def post(self, request, \*args, \*\*kwargs):

        pk = self.kwargs.get('pk')

        form = AddCommentForm(request.POST)

        if form.is\_valid():

            comment = form.save(commit=False)

            comment.team = self.request.user.userprofile.get\_active\_team()

            comment.created\_by = request.user

            comment.lead\_id = pk

            comment.save()

        return redirect('leads:detail', pk=pk)

class ConverToClientView(LoginRequiredMixin, View):

    def get(self, request, \*args, \*\*kwargs):

        pk = self.kwargs.get("pk")

        team = self.request.user.userprofile.active\_team

        lead = get\_object\_or\_404(Lead, created\_by=request.user, pk=pk)

        team = self.request.user.userprofile.get\_active\_team()

        client = Client.objects.create(

            name=lead.name,

            email=lead.email,

            description=lead.description,

            created\_by=request.user,

            team=team,

        )

        lead.converted\_to\_client = True

        lead.save()

        #convertir comentarios de clientes potenciales para convertir comentarios de clientes

        comments = lead.comments.all()

        for comment in comments:

            ClientComment.objects.create(

                client=client,

                content=comment.content,

                created\_by=comment.created\_by,

                team=team

            )

        messages.success(request, "El cliente potencial ha sido convertido en cliente.")

        return redirect("leads:list")

**urls.py:**

from django.urls import path

from . import views

app\_name = 'leads'

urlpatterns = [

    path('', views.LeadListView.as\_view(), name= 'list'),

    path('<int:pk>/', views.LeadDetailView.as\_view(), name='detail'),

    path('<int:pk>/delete/', views.LeadDeleteView.as\_view(), name='delete'),

    path('<int:pk>/edit/', views.LeadUpdateView.as\_view(), name='edit'),

    path('<int:pk>/convert/', views.ConverToClientView.as\_view(), name='convert'),

    path('<int:pk>/add-comment/', views.AddCommentView.as\_view(), name='add\_comment'),

    path('<int:pk>/add-file/', views.AddFileView.as\_view(), name='add\_file'),

    path('add/', views.LeadCreateView.as\_view(), name='add'),

    path('export/', views.leads\_export, name='export'),

]