**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: core.**

**Archivos:**

**Views.py:**

```
from django.shortcuts import render


def index(request):

    return render(request, 'core/index.html')


def about(request):

    return render(request, 'core/about.html')
```

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

**Archivos:**

**urls.py:**

```
from django.urls import path


from . import views


app_name = 'dashboard'


urlpatterns = [
    path('', views.dashboard, name='index'),
]
```

**Views.py:**

```
from django.contrib.auth.decorators import login_required
from django.shortcuts import render

from lead.models import Lead
from client.models import Client
from team.models import Team


@login_required
def dashboard(request):
    team = request.user.userprofile.get_active_team()

    leads = Lead.objects.filter(team=team, converted_to_client=False).order_by('-created_at')[0:5]
    clients = Client.objects.filter(team=team).order_by('-created_at')[0:5]

    return render(request, 'dashboard/dashboard.html', {
        'leads': leads,
        'clients': clients,
    })
```

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

**Archivos:**

**models.py:**

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

from team.models import Team

class Client(models.Model):
    team = models.ForeignKey(Team, related_name='clients', on_delete=models.CASCADE)
    name = models.CharField(max_length=255)
    email = models.EmailField()
    description = models.TextField(blank=True, null=True)
    created_by = models.ForeignKey(User, related_name='clients', 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 Comment(models.Model):
    team = models.ForeignKey(Team, related_name='client_comments', on_delete=models.CASCADE)
    client = models.ForeignKey(Client, related_name='comments', on_delete=models.CASCADE)
    content = models.TextField(blank=True, null=True)
    created_by = models.ForeignKey(User, related_name='client_comments', on_delete=models.CASCADE)
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):

        return self.created_by.username
```

```
class ClientFile(models.Model):
    team = models.ForeignKey(Team, related_name='client_files', on_delete=models.CASCADE)
    client = models.ForeignKey(Client, related_name='files', on_delete=models.CASCADE)
    file = models.FileField(upload_to='clientfiles/')
    created_by = models.ForeignKey(User, related_name='client_files', 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.decorators import login_required
from django.http import HttpResponse
from django.shortcuts import render, redirect, get_object_or_404

from .forms import AddClientForm, AddCommentForm, AddFileForm
from .models import Client

from team.models import Team

@login_required
def clients_export(request):
    clients = Client.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(['Client', 'Description', 'Created at', 'Created by'])

    # Escribir datos de clientes
    for client in clients:
        writer.writerow([client.name, client.description, client.created_at, client.created_by])

    return response

@login_required
def clients_list(request):
    team = request.user.userprofile.active_team
    clients = team.clients.all()

    return render(request, "client/clients_list.html", {"clients": clients})

@login_required
def clients_add_file(request, pk):
    if request.method == 'POST':
        form = AddFileForm(request.POST, request.FILES)

        if form.is_valid():
            file = form.save(commit=False)
            file.team = request.user.userprofile.active_team
            file.client_id = pk
            file.created_by = request.user
            file.save()

            return redirect('clients:detail', pk=pk)
    return redirect('clients:detail', pk=pk)
```

```
@login_required
def clients_detail(request, pk):
    team = request.user.userprofile.active_team
    client = get_object_or_404(Client, created_by=request.user, pk=pk)

    if request.method == 'POST':
        form = AddCommentForm(request.POST)

        if form.is_valid():
            comment = form.save(commit=False)
            comment.team = request.user.userprofile.get_active_team()
            comment.created_by = request.user
            comment.client = client
            comment.save()

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

    else:
        form = AddCommentForm()

    return render(request, "client/clients_detail.html", {
        "client": client,
        "form": form,
        "fileform" : AddFileForm(),
    })

@login_required
def clients_add(request):
    team = request.user.userprofile.get_active_team()

    if request.method == "POST":
        form = AddClientForm(request.POST)
```

```
        if form.is_valid():
            client = form.save(commit=False)
            client.created_by = request.user
            client.team = team
            client.save()

            messages.success(request, "El cliente fue creado.")

            return redirect("clients:list")
    else:
        form = AddClientForm()

    return render(request, "client/clients_add.html", {"form": form, "team": team})


@login_required
def clients_delete(request, pk):
    team = request.user.userprofile.active_team
    client = get_object_or_404(Client, created_by=request.user, pk=pk)
    client.delete()

    messages.success(request, "El cliente fue eliminado")

    return redirect("clients:list")


@login_required
def clients_edit(request, pk):
    team = request.user.userprofile.active_team
    client = get_object_or_404(Client, created_by=request.user, pk=pk)

    if request.method == "POST":
```

```
        form = AddClientForm(request.POST, instance=client)

        if form.is_valid():
            form.save()

            messages.success(request, "Los cambios fueron guardados.")

            return redirect("clients:list")
    else:
        form = AddClientForm(instance=client)

    return render(request, "client/clients_edit.html", {"form": form})
```

**urls.py:**

```
from django.urls import path
from . import views

app_name = 'clients'
urlpatterns = [
    path("", views.clients_list, name="list"),
    path("<int:pk>/", views.clients_detail, name="detail"),
    path("<int:pk>/delete/", views.clients_delete, name="delete"),
    path("<int:pk>/edit/", views.clients_edit, name="edit"),
    path("<int:pk>/add-comment/", views.clients_detail, name="add_comment"),
    path("<int:pk>/add-file/", views.clients_add_file, name='add_file'),
    path("add/", views.clients_add, name="add"),
    path('export/', views.clients_export, name='export'),
]
```

**admin.py:**

```
from django.contrib import admin
from.models import Client, Comment
admin.site.register(Client)
admin.site.register(Comment)
```

**forms.py:**

```
from django import forms
from .models import Client, Comment, ClientFile


class AddClientForm(forms.ModelForm):
    name = forms.CharField(
        widget=forms.TextInput(attrs={"class": "mb-2 w-full py-4 px-6 rounded-xl bg-gray-100"})
    )
    email = forms.CharField(
        widget=forms.TextInput(attrs={"class": "mb-2 w-full py-4 px-6 rounded-xl bg-gray-100"})
    )
    description = forms.CharField(
        widget=forms.Textarea(attrs={"rows":"5", "class": "mb-2 w-full bg-gray-100 rounded-xl"})
    )

    class Meta:
        model = Client
        fields = ('name', 'email', 'description',)


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 = ClientFile
        fields = ('file',)
```

**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'),
]
```

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

**Archivos:**

**Admin.py:**

```
from django.contrib import admin
from .models import Team, Plan

admin.site.register(Team)
admin.site.register(Plan)
```

**context_processors.py:**

```
from .models import Team

def active_team(request):
    if request.user.is_authenticated:
        active_team = request.user.userprofile.get_active_team()
        if not active_team:
            active_team = Team.objects.filter(created_by=request.user)[0]
```

```
    else:
        active_team = None
    return {'active_team': active_team}
```

**models.py:**

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


class Plan(models.Model):
    name = models.CharField(max_length=50)
    price = models.IntegerField()
    description = models.TextField(blank=True, null=True)
    max_leads = models.IntegerField()
    max_clients = models.IntegerField()

    def __str__(self):
        return self.name


class Team(models.Model):
    plan = models.ForeignKey(Plan, related_name='teams', blank=True, null=True, on_delete=models.CASCADE)
    name = models.CharField(max_length=100)
    members = models.ManyToManyField(User, related_name='teams')
    created_by = models.ForeignKey(User, related_name='created_teams', on_delete=models.CASCADE)
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.name

    def get_plan(self):
        if self.plan:
            return self.plan
        else:
            if Plan.objects.count() > 0:
                self.plan = Plan.objects.all().first()
```

```
            self.save()
        else:
            plan = Plan.objects.create(name='Free', price=0, max_leads=3, max_clients=3)
            self.plan = plan
            self.save()
        return self.plan
```

**forms.py:**

```
from django import forms
from .models import Team


class TeamForm(forms.ModelForm):
    class Meta:
        model = Team
        fields = ("name",)
        labels = {
            'name': 'Nombre del equipo',
        }
```

**views.py:**

```
from django.contrib import messages
from django.contrib.auth.decorators import login_required
from django.shortcuts import render, get_object_or_404, redirect
from .forms import TeamForm
from .models import Team


@login_required
def teams_list(request):
    teams = Team.objects.filter(members__in=[request.user])
    return render(request, 'team/teams_list.html', {'teams': teams})


@login_required
def teams_activate(request,pk):
    team = Team.objects.filter(members__in=[request.user]).get(pk=pk)
    userprofile = request.user.userprofile
```

```
    userprofile.active_team = team
    userprofile.save()

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

@login_required
def detail(request, pk):
    team = get_object_or_404(Team, members__in=[request.user], pk=pk)

    return render(request, 'team/detail.html', {'team':team})

@login_required
def edit_team(request, pk):
    team = get_object_or_404(Team, created_by=request.user, pk=pk)

    if request.method == 'POST':
        form = TeamForm(request.POST, instance=team) #Mensaje activado

        if form.is_valid():
            form.save()

            messages.success(request, 'Se ha guardado correctamente')

            return redirect('userprofile:myaccount')
    else:
        form = TeamForm(instance=team)

    return render(request, 'team/edit_team.html', {
        'team': team,
        'form': form
    })
```

**urls.py:**

```
from django.urls import path
from . import views

app_name = 'team'
urlpatterns = [
    path('', views.teams_list, name='list'),
    path('<int:pk>/', views.detail, name='detail'),
    path('<int:pk>/edit/', views.edit_team, name='edit'),
    path('<int:pk>/activate/', views.teams_activate, name='activate'),
]
```

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

**Archivos:**

**Admin.py:**

```
from django.contrib import admin
from .models import Userprofile
admin.site.register(Userprofile)
```

**Forms.py:**

```
from django import forms
from django.contrib.auth.forms import UserCreationForm, AuthenticationForm
from django.contrib.auth.models import User

INPUT_CLASS = 'w-full my-4 py-4 px-6 rounded-xl bg-gray-100'
username="Nombre de usuario"
class LoginForm(AuthenticationForm):

    username = forms.CharField(label="Nombre de Usuario", widget=forms.TextInput(attrs={

        'placeholder': 'Ingrese su usuario',
        'class': INPUT_CLASS
    }))

    password = forms.CharField(label="Contraseña", widget=forms.PasswordInput(attrs={
```

```
        'placeholder': 'Ingrese su contraseña',
        'class': INPUT_CLASS
    }))


class SignupForm(UserCreationForm):
    class Meta:
        model = User
        fields = ('username', 'email', 'password1', 'password2')

    username = forms.CharField(label="Nombre de usuario",widget=forms.TextInput(attrs={
        'placeholder': 'Escriba su nuevo usuario',
        'class': INPUT_CLASS
    }))
    email = forms.CharField(label="Correo electronico",widget=forms.EmailInput(attrs={
        'placeholder': 'Ingrese su correo',
        'class': INPUT_CLASS
    }))
    password1 = forms.CharField(label="Constraseña", widget=forms.PasswordInput(attrs={
        'placeholder': 'Escriba su contraseña',
        'class': INPUT_CLASS
    }))
    password2 = forms.CharField(label="Repetir contraseña", widget=forms.PasswordInput(attrs={
        'placeholder': 'Repita la contraseña',
        'class': INPUT_CLASS
    }))
```

**Urls.py:**

```
from django.urls import path
from . import views
app_name = 'userprofile'

urlpatterns = [
    path('myaccount/', views.myaccount, name='myaccount'),
```

```
    path('sign-up/', views.signup, name='signup'),
    path('log-in/', views.myaccount, name='user_login')
]
```

**models.py:**

```
from django.contrib.auth.models import User
from django.db import models
from django.urls import reverse
from team.models import Team


class Userprofile(models.Model):
    user = models.OneToOneField(User, related_name='userprofile', on_delete=models.CASCADE)
    active_team = models.ForeignKey(Team, related_name='userprofiles', blank=True, null=True,
on_delete=models.CASCADE)


    def get_active_team(self):
        if self.active_team:
            return self.active_team
        else:
            return Team.objects.filter(members__in=[self.user.id]).first()


    def get_absolute_url(self):
        return reverse('login')
```

**Views.py:**

```
from django.contrib.auth.decorators import login_required
from django.shortcuts import render, redirect
from .forms import SignupForm
from .models import Userprofile


from team.models import Team


def signup(request):
    if request.method == 'POST':
        form = SignupForm(request.POST)
```

```
        if form.is_valid():
            user = form.save()

            team = Team.objects.create(name='The team name', created_by=user)
            team.members.add(user)
            team.save()

            Userprofile.objects.create(user=user, active_team=team)

            return redirect('/log-in/')
    else:
        form = SignupForm()

    return render(request, 'userprofile/signup.html', {
        'form': form
    })

@login_required
def myaccount(request):
    return render(request, 'userprofile/myaccount.html')
```