Entregable de Programación Web 2 - Proyecto Final
“AEM”

Edsel Yael Alvan Ventura
Alex Williams Olaechea Carlo
Frank Leny Ccapa Usca
ealvan@unsa.edu.pe
aolaechea@unsa.edu.pe
fccapau@unsa.edu.pe

Descripción del proyecto:

Este proyecto permitirá emular una red social con algunas de sus funciones correspondientes, así mismo este contará con un login para el ingreso respectivo del usuario,se almacenará los usuarios en una base de datos el cual permitirá que estos puedan publicar preguntas, agregar las distintas opciones

Planeación y sesiones:

Las sesiones se acordaron por unanimidad los **viernes, jueves, sábado y domingos de 6pm a 8pm.**

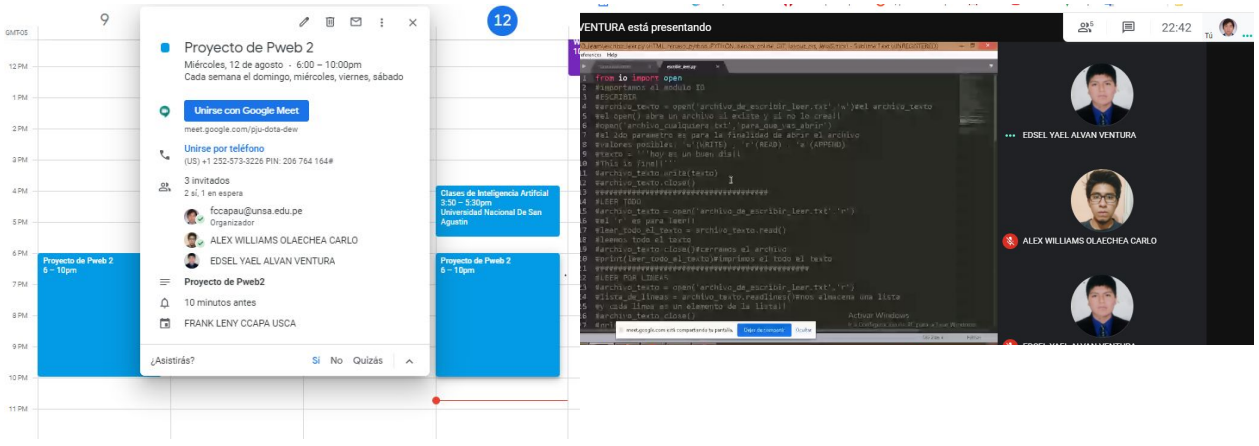
Planeación:

Para trabajar mejor, nos ayudamos de Github, una herramienta muy útil para trabajar en equipo.

Durante las sesiones:

1. Se muestran los cambios y mejoras que se hicieron al proyecto.
2. Se observa los errores que hubo para poder solucionarlo
3. Se discute las ideas que se tien para poder debatirlas.
4. Al final de cada sesión dejamos tareas asignadas a cada integrante.

Imagen 1 sesiones



Herramientas a utilizar:

- GitHub

link al repositorio

GitHub es una forja para alojar proyectos utilizando el sistema de control de versiones Git.

- DJango

Django es un framework de desarrollo web de código abierto,

Modelos:

Pregunta: Este modelo consta de los siguientes campos.

- pregunta_txt
- area
- pub_fecha

Opciones: Al igual que el modelo anterior este modelo también consta de los siguientes campos

- pregunta_txt
- opcion_txt
- votos

Views:

1. **home:** Se mostrará cuando se realice login
2. **welcome:** Mostrará la bienvenida a una persona que no sea usuario**.**
3. **registro:** Es para crear usuarios.
4. **login:** Es para el inicio de sesión de usuarios.
5. **logoutU:** Es para cerrar sesión.
6. **detalle:** Nos muestra la pregunta con sus opciones
7. **votar:** Cambia el número de votos de una opción
8. **resultado:** Nos muestra los resultados de la encuesta
9. **crear_opcion:** Es para crear una o varias nuevas opciones
10. **tablon:** Nos muestra un tablón con todas las preguntas
11. **preguntaCreateVieew:** Es para crear una pregunta.
12. **borrar:** Borra una pregunta concreta.
13. **PreguntaQueryView:** Retorna un json.
14. **tablonAjaxView:** Nos muestra un template con un tablón pedido con ajax.

Templates:

Encuestas

- borrar.html
- crear_opcion.html
- crearPregunta.html
- detalle.html
- resultado.html
- tablon.html
- tablonAjax.html

- base.html
- base2.html
- login.html
- registro.html

Static

css

- detalle.css
- encuestas.css
- login.css
- tabla.css

Ejemplo de cómo podemos trabajar con Github:

La aplicación será calificada sobre 20 y deberá incluir:

- Una app independiente

```python
from django.contrib import admin
from django.urls import path,include
from encuestas import views
urlpatterns = [
    path('admin/', admin.site.urls),
    path('',views.welcome, name='bienvenido'),
    path('encuestas/',include('encuestas.urls')),
    path('registro/',views.registro, name='registro'),
    path('login/',views.login, name = 'login'),
    path('logout/',views.logoutU, name='logout'),
]
```

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

app_name = 'encuestas'

urlpatterns = [
    path('', views.home, name='home'),
    path('<int:preguntaID>/',views.detalle, name="detalle"),
    path('tablon/',views.tablon, name = 'tablon'),
```

URLs propios, usando reverse **(2 puntos)**

```python
    except:
        raise Http404("lo sentimos no se pudo crear la bbdd")
    return HttpResponseRedirect(reverse('encuestas:detalle', args = (pregunta.id,)))
else:
```

```python
    return render(request,'encuestas/detalle.html',diccionario)
else:
    opcion_selec.votos += 1
    opcion_selec.save()
    return HttpResponseRedirect(reverse('encuestas:resultado',args=(pregunta.id,)))
```

```python
        if user is not None:
            auth_login(request,user)
            return HttpResponseRedirect(reverse('encuestas:home',))
```

```python
def registro(request):
    if request.user.is_authenticated:
        return HttpResponseRedirect(reverse('encuestas:home'))
```

Plantillas propias **(1 puntos)//En inicio//hazle tu frank, creo que sabes este :)**

Que usen widgets de manera elegante **(1 puntos)**

```
class Meta:
  model=Pregunta
  fields = ['pregunta_txt','area']
  widgets = {
          'pregunta_txt':forms.TextInput(attrs={'class':'pregunta',
                                                'placeholder':'ingrese su pregunta'}),
      }
```

```
    'password2',
    ]
    widgets = {
        'username':forms.TextInput(attrs={'class':'pregunta',
                                          'placeholder':'ingrese su usuario'}),
        'email':forms.EmailInput(attrs={'class':'special',
                                        'placeholder':'ingrese su email'}),
    }
```

Vistas de Listado, Detalle, Crear, Actualizar y Borrar **(4 puntos)**

Vista tablon:

```
def tablon(request):
    lista = Pregunta.objects.all()
    if lista:
        return render(request,'encuestas/tablon.html',{'lista':lista,})
    else:
        raise Http404("lo sentimos, aun no se han publicado preguntas")
```

Vista crear:

```
def preguntaCreateView(request):

    if request.method == "POST":
        form = PreguntaForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect("encuestas:tablon")
    else:
        form = PreguntaForm()
    context = {
       'form': form,
    }
    return render(request, 'encuestas/crearPregunta.html', context)
```

Vista actualizar://un moment

Vista borrar:

```python
def borrar(request,preguntaID):
    try:
        pregunta = Pregunta.objects.get(pk=preguntaID)
    except:
        raise Http404("lo sentimos, la pregunta no existe")
    try:
        pregunta.delete()
        pregunta.save()
    except:
        raise Http404("lo sentimos ocurrio un inesperado error,intentelo mas tarde")
    return render(request,'encuestas/borrar.html',{})
```

Formulario con restricciones de seguridad adicionales **(2 puntos)**

```python
                                        'placeholder':'ingrese su pregu
        }
def clean_pregunta_txt(self):
  ''' hara una validacion, si no hay un '?' o '¿' en la pregunta'''
  pregunta = self.cleaned_data.get("pregunta_txt")
  fin = pregunta[-1]
  ini = pregunta[0]
  if fin != '?' or ini != '¿':
     raise forms.ValidationError("una pregunta debe tener un '¿' y '?'")
  return pregunta
```

Vista de consultas que devuelven Json **(3 puntos)**

```python
class PreguntaQueryView(View):
    def get(self, request):
        return JsonResponse(list(queryset.values()), safe = False)
```

Programa cliente para hacer y consumir las consultas
Con Ajax **(2 puntos)**

```
<script>
    function loadDoc() {
        var xhttp = new XMLHttpRequest();
        xhttp.onreadystatechange = function() {
            if (this.readyState == 4 && this.status == 200) {
                var data = JSON.parse(this.responseText);
                var html = '<table>\n';
                html += '  <tr>\n';
                html += '    <th>Preguntas</th>\n';
                html += '    <th>Fechas</th>\n';
    html += '    <th>Area</th>\n';
    html += '  </tr>\n';
                for(i in data){
                    var pregunta = data[i].pregunta_txt;
        var fecha = data[i].pub_fecha;
        var area = data[i].area;
                    html += '  <tr>\n';
                    html += '    <td>' + pregunta + '</td>\n';
        html += '    <td>' + fecha + '</td>\n';
        html += '    <td>' + area + '</td>\n';
                    html += '  </tr>\n';
                }
                html += '</table>\n';
                document.getElementById("demo").innerHTML = html;
            }
        };
        xhttp.open("GET", "http://127.0.0.1:8000/encuestas/query/", true);
        xhttp.send();
    }
```


Al menos dos modelos **(2 puntos)**

Modelo Pregunta:

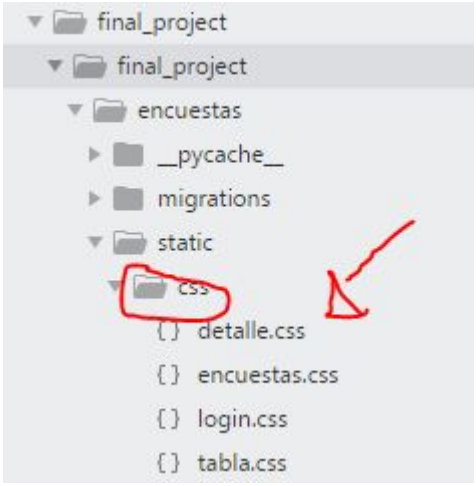
```
# Create your models here.
class Pregunta(models.Model):
    pregunta_txt = models.CharField(max_length=100)

    opciones = [('comida','Comida'),
                ('deporte','Deporte'),
                ('salud','Salud'),
                ('farandula','Farandula'),
                ('noticias','Noticias'),
                ('cine','Cine'),
                ('ciencias','Ciencias'),
                ]
    area = models.CharField(max_length=20,choices=opciones,default='salud',)
    pub_fecha = models.DateTimeField('Fecha de Publicacion', default = timezo
    def __str__(self):
        return f'{self.pregunta_txt}'
```

Modelo Opción(*Modelo con clave externa: foreign key (2 puntos)***):**

```python
class Opcion(models.Model):
    pregunta_txt = models.ForeignKey(Pregunta, on_delete = models.CASCADE)
    opcion_txt = models.CharField(max_length=100)
    votos = models.IntegerField(default=0)
    def __str__(self):
        return f'opinion: {self.opcion_txt} votos: {self.votos}'
```

CSS **(2 puntos)**



Publicó su aplicación en el web (+3 puntos)//f papu

Descargar un informe como archivos pdf (+2 puntos)
Enviar correo (+2 puntos)

URLs propios, usando reverse **(2 puntos)**	✔
Plantillas propias **(1 puntos)**	✔
Vistas de Listado, Detalle, Crear, Actualizar y Borrar **(4 puntos)**	✔
Formulario con restricciones de seguridad adicionales **(2 puntos)**	✔
Vista de consultas que devuelva Json **(3 puntos)**	✔
Con Ajax **(2 puntos)**	✔
Con AJAX, javascript **(2 puntos)**	✔
Al menos dos modelos **(2 puntos)**	✔
Modelo con clave externa: foreign key (2 puntos)	✔
Descargar un informe como archivos pdf (+2 puntos)	//ahoritta

Enviar correo (+2 puntos)	tengo que hacer

Informe de trabajo ya realizado:

- Creación reutilizo el modelo Usuario que nos brinda el django para poder realizar una especie de login // De tal manera podíamos reciclar el Formulario que tiene DJango para crear usuarios así como ese modelo que se llama “User” bueno esto en primera instancia solo para hacer el Formulario donde los datos que nos brinda el DJango son: “username” - “first_name” - “last_name” - “email”. Así en la vista para poder reciclar el CreateVIew que existe en DJango se creó una vista de clase tendra herencia del CreateView... En esta vista tendría que ser asignado los valores de model = User -- template_name = “Donde esta ubicado” -- form_class = RegistroForm (este es el formulario que se creo) y el succes_url = reverse_lazy() que nos permitirá irnos al momento de registrarnos a otra ventana
- Creación de un login… Se crea un login primero creando un formulario donde hayun user name y un password y un widget del password que lo vaya a mostrar como si fuera contraseña… Ahora pasando a la vista se define una funcion (no se reciclo el login que ya existe en el DJAngo por error (aunque se sigue intentando)) y en esta vista recogerá los datos que sean de tipo post y con la funcion authenticate (aca ingresar el username y password) y de esa manera se logea **(Esta forma de login trabajada creemos que es muy insegura)**
- Se creó los templates y vistas que forman parte de lo que es el modelo Publicaciones: Detalles (aún en un proceso) pero ya muestra las Publicaciones, estos se listan, crean, se actualizan y borran…
- Se crearon los templates de usuario, editar perfil, detalles.

Trabajo a realizar e ideas a implementar:

Modelo amigo: este contará con el atributo id_nombre_amigo este será la foreankey a utilizar con el campo id_nombre_amigo del modelo usuario creando de esta manera la relación entre tablas uno a muchos.
planeamos impleamentar la **llamada ajax** cuando un usuario quiera ver los estados de sus amigos(en proceso)
Implementación de angular para una mejor vista ante el usuario
mejorar nuestras hojas de estilo
Planeamos mandar un correo al momento de login para que avise que usted pertenecera a nuestra pagina

Usuario

id_nombreamigo

estado int 1, 2, ,4

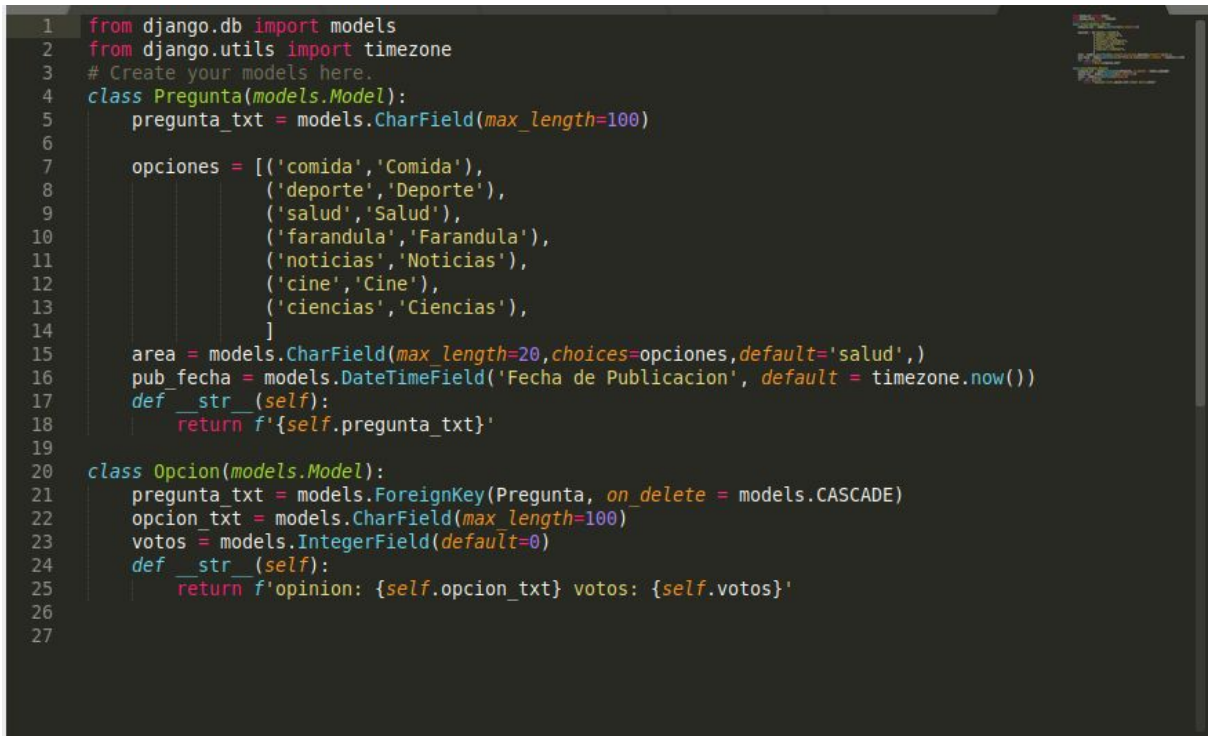
amigos
id_ nombreamigo
añadir

Publicacion
estado int 1 ,2 ,,4

Anexos:

Imagen Templates

Imagen Modelos



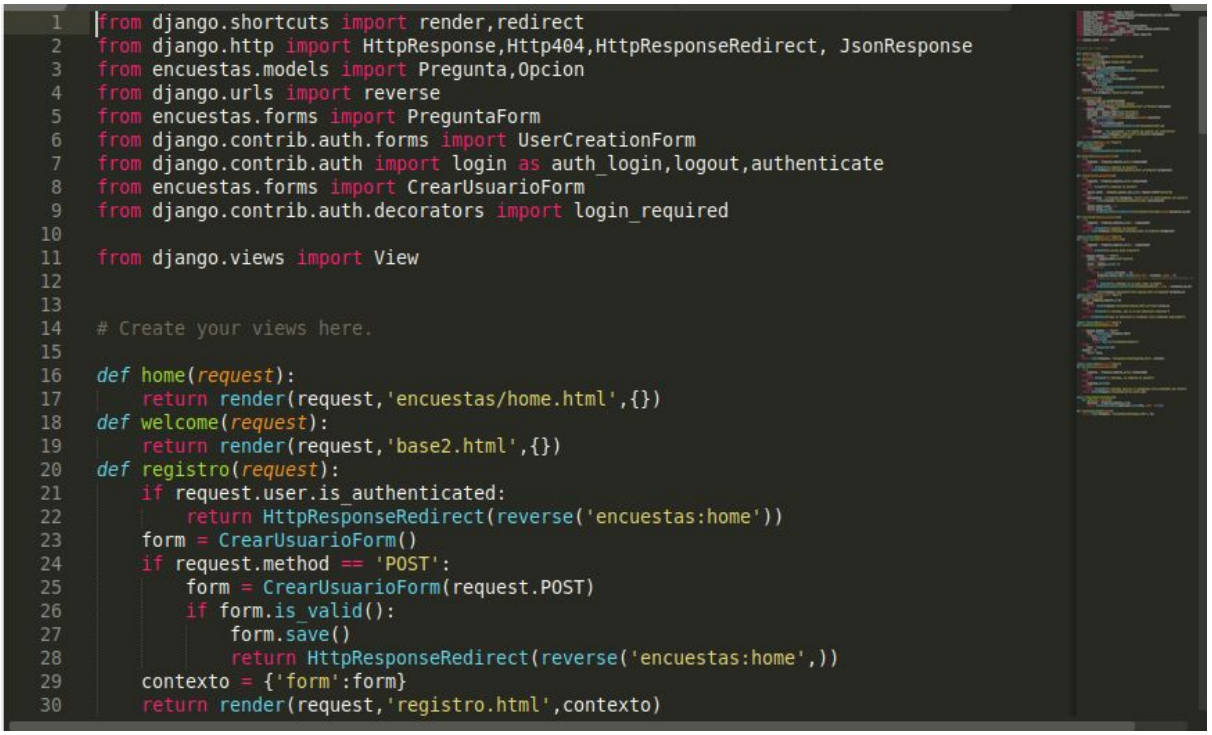
```python
from django.db import models
from django.utils import timezone
# Create your models here.
class Pregunta(models.Model):
    pregunta_txt = models.CharField(max_length=100)

    opciones = [('comida','Comida'),
                ('deporte','Deporte'),
                ('salud','Salud'),
                ('farandula','Farandula'),
                ('noticias','Noticias'),
                ('cine','Cine'),
                ('ciencias','Ciencias'),
                ]
    area = models.CharField(max_length=20,choices=opciones,default='salud',)
    pub_fecha = models.DateTimeField('Fecha de Publicacion', default = timezone.now())
    def __str__(self):
        return f'{self.pregunta_txt}'

class Opcion(models.Model):
    pregunta_txt = models.ForeignKey(Pregunta, on_delete = models.CASCADE)
    opcion_txt = models.CharField(max_length=100)
    votos = models.IntegerField(default=0)
    def __str__(self):
        return f'opinion: {self.opcion_txt} votos: {self.votos}'


```

Imagen view Parte 1



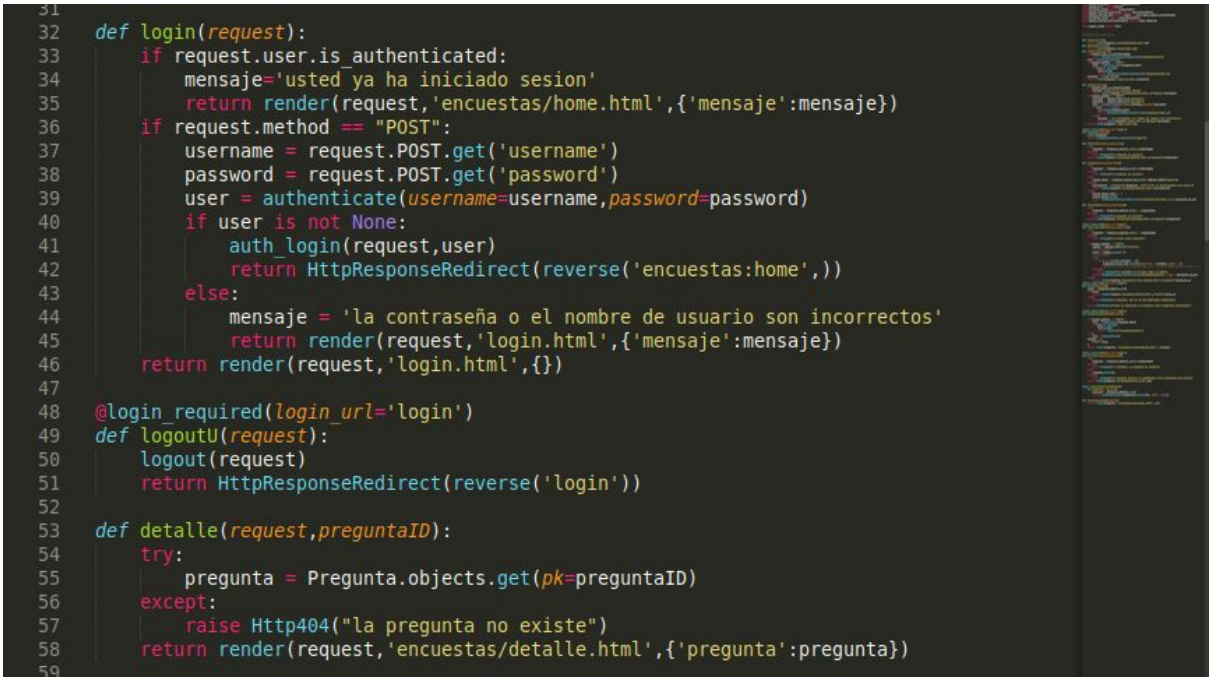
```python
from django.shortcuts import render,redirect
from django.http import HttpResponse,Http404,HttpResponseRedirect, JsonResponse
from encuestas.models import Pregunta,Opcion
from django.urls import reverse
from encuestas.forms import PreguntaForm
from django.contrib.auth.forms import UserCreationForm
from django.contrib.auth import login as auth_login,logout,authenticate
from encuestas.forms import CrearUsuarioForm
from django.contrib.auth.decorators import login_required

from django.views import View


# Create your views here.

def home(request):
    return render(request,'encuestas/home.html',{})
def welcome(request):
    return render(request,'base2.html',{})
def registro(request):
    if request.user.is_authenticated:
        return HttpResponseRedirect(reverse('encuestas:home'))
    form = CrearUsuarioForm()
    if request.method == 'POST':
        form = CrearUsuarioForm(request.POST)
        if form.is_valid():
            form.save()
            return HttpResponseRedirect(reverse('encuestas:home',))
    contexto = {'form':form}
    return render(request,'registro.html',contexto)
```

Imagen view Parte 2



```
def login(request):
    if request.user.is_authenticated:
        mensaje='usted ya ha iniciado sesion'
        return render(request,'encuestas/home.html',{'mensaje':mensaje})
    if request.method == "POST":
        username = request.POST.get('username')
        password = request.POST.get('password')
        user = authenticate(username=username,password=password)
        if user is not None:
            auth_login(request,user)
            return HttpResponseRedirect(reverse('encuestas:home',))
        else:
            mensaje = 'la contraseña o el nombre de usuario son incorrectos'
            return render(request,'login.html',{'mensaje':mensaje})
    return render(request,'login.html',{})

@login_required(login_url='login')
def logoutU(request):
    logout(request)
    return HttpResponseRedirect(reverse('login'))

def detalle(request,preguntaID):
    try:
        pregunta = Pregunta.objects.get(pk=preguntaID)
    except:
        raise Http404("la pregunta no existe")
    return render(request,'encuestas/detalle.html',{'pregunta':pregunta})
```

Imagen view Parte 3

```
def votar(request,preguntaID):
    try:
        pregunta = Pregunta.objects.get(pk=preguntaID)
    except:
        raise Http404("la pregunta no existe")
    try:
        opcion_selec = pregunta.opcion_set.get(pk=request.POST['opcion'])
    except:
        diccionario = {'pregunta':pregunta, 'error':'tu no seleccionaste una opcion'}
        return render(request,'encuestas/detalle.html',diccionario)
    else:
        opcion_selec.votos += 1
        opcion_selec.save()
        return HttpResponseRedirect(reverse('encuestas:resultado',args=(pregunta.id,)))

def resultado(request,preguntaID):
    try:
        pregunta = Pregunta.objects.get(pk = preguntaID)
    except:
        raise Http404("la pregunta no existe")
    return render(request,'encuestas/resultado.html',{'pregunta':pregunta})
```

Imagen view Parte 4

```python
@login_required(login_url='login')
def crear_opcion(request,preguntaID):
    try:
        pregunta = Pregunta.objects.get(id = preguntaID)
    except:
        raise Http404("no existe esta pregunta")

    if request.method == "POST":
        cadena = request.POST.get("lista");
        #print(cadena)
        lista = cadena.split(",")
        #print(lista)
        try:
            for x in range(len(lista) - 1):
                pregunta.opcion_set.create(opcion_txt = lista[x], votos = 0)
                #pregunta.opcion_set.create(opcion_txt = request.POST['opcion'],votos = 0)
        except:
            raise Http404("lo sentimos no se pudo crear la bbdd")
        return HttpResponseRedirect(reverse('encuestas:detalle', args = (pregunta.id,)))
    else:
        return render(request,'encuestas/crear_opcion.html',{'pregunta':pregunta,})
```