Entregable de Programación Web 2 - Proyecto Final
“Encuestas”

Edsel Yael Alvan Ventura
Alex Williams Olaechea Carlo
Frank Leny Ccapa Usca
Escuela Profesional de Ingeniería de Sistemas
Universidad Nacional de San Agustín, Arequipa, Perú
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.**

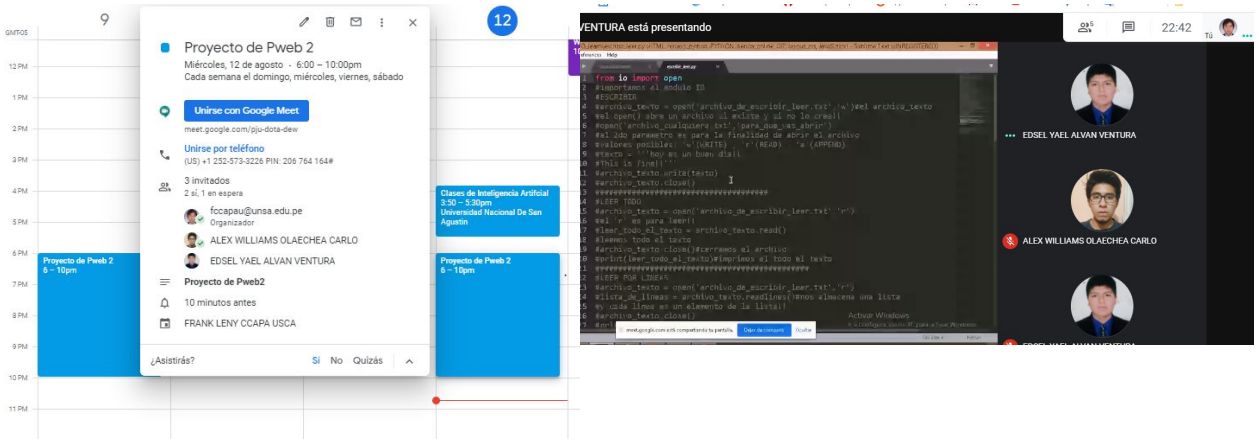
Planeación:

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



Repartición del proyecto:

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 devuelve un Json con los objetos Pregunta, la cual después la mostramos en el template ‘tablonAjax’

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
Independencia de Urls.

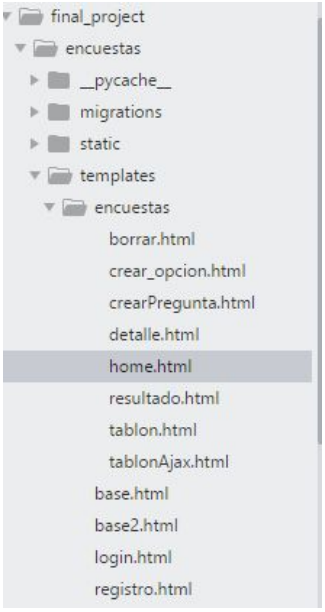
```python
from django.contrib import admin
from django.urls import path,include
from encuestas import views
urlpatterns = [
    path('admin/', admin.site.urls),
    path('',views.welcome, name='bienvenido'),
    path('encuestas/',include('encuestas.urls')),
    path('registro/',views.registro, name='registro'),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

Independencia de Templates.



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

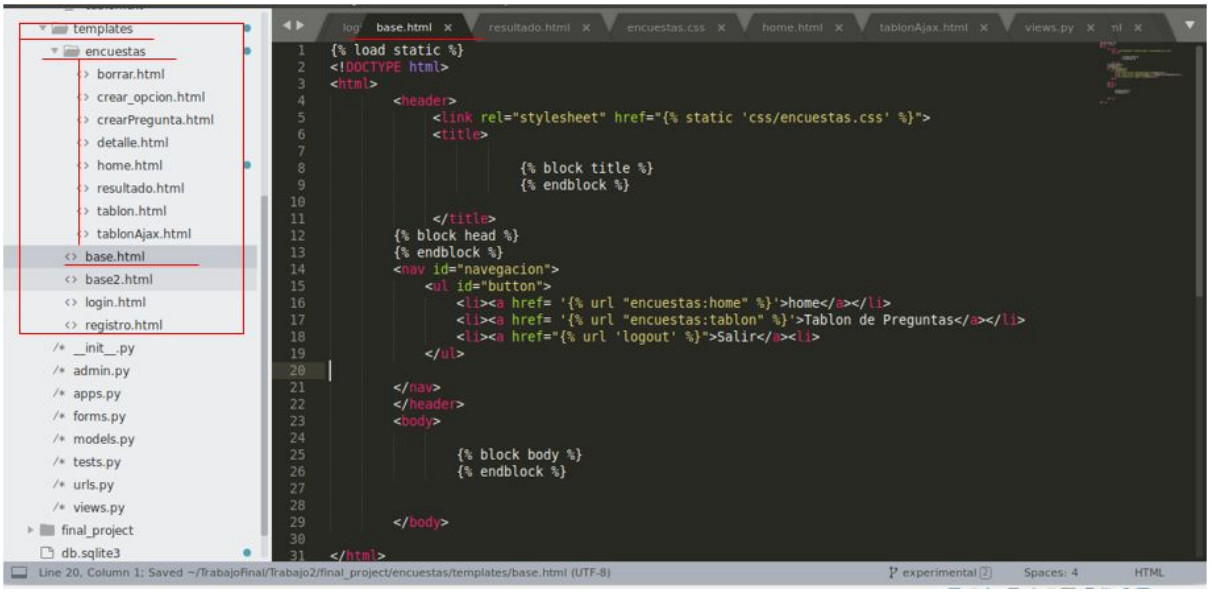
```python
    except:
        raise Http404("lo sentimos no se pudo crear la bbdd")
    return HttpResponseRedirect(reverse('encuestas:detalle', args = (pregunta.id,)))
else:
```

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

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

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

Plantillas propias **(1 puntos)**



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:

```python
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:

```python
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:

```python
class PreguntaUpdateView(UpdateView):
    model = Pregunta
    fields = {
            'pregunta_txt',
            'area',
            'pub_fecha',
            }
    template_name_suffix = '_form_update'
```

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)**

```
                                        '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)**

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

class OpcionQueryView(View):
    def get(self, request):
        queryset = Opcion.objects.all()
        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:

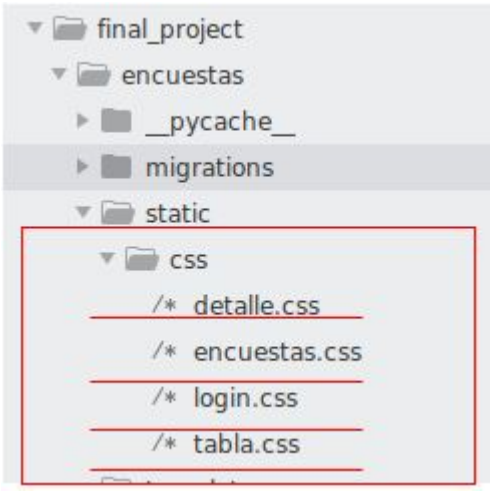
```
# Create your models here.
class Pregunta(models.Model):
    pregunta_txt = models.CharField(max_length=100)

    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)

Descargar un informe como archivos pdf (+2 puntos)

```html
        <button type="button" onclick="loadDoc()">Obtener tablon</button>
    </div>
    <a href="{% static 'data/projectoFinal.pdf' %}" download="projetoFinal.pdf">
    Descargue el informe del projecto!! </a>
```

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 devuelven 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)	✔
Enviar correo (+2 puntos)	