ORM Y FORMULARIOS

DJANGO

DJANGO APPLICATIONS

DJANGO ORM

DJANGO ORM

- **OBTENER TODOS:**
  Modelo.objects.all()
- **OBTENER UNO:**
  Modelo.objects.get(id=1)
- **OBTENER QUERY:**
  qs = Modelo.objects.all()
  print (qs.query)

DJANGO ORM – FILTRAR DATOS

- gt / gte
- lt / lte
- contains
- icontains
- startswith
- endswith

- **FILTRAR MAYOR / MENOR:**
  Modelo.objects.filter(salary_ _gt=5000)
  Modelo.objects.filter(salary_ _lte=5000)
- **CONTIENE:**
  Modelo.objects.get(nombre_ _contains='Juan')
  Modelo.objects.get(nombre_ _icontains='Juan')
- **EN LISTADO:**
  Modelo.objects.filter(id_ _in=[1,2,3])

EL PREFIJO I SIRVE PARA IGNORAR MAYUSCULAS Y MINÚSCULAS

DJANGO ORM – MULTIPLES FILTROS

- **UTILIZAR MAS DE UN PARAMETRO PARA EL FILTRO**

  Modelo.objects.filter(nombre_ _startswith='Juan') | Modelo.objects.filter(nombre_ _startswith='Pedro')
  Modelo.objects.filter(nombre_ _startswith='Jose' , apellido_ _startswith='Perez')

- **UTILIZAR Q SINTAX**

  from django.db.models import Q
  Modelo.objects.filter(Q(nombre_ _startswith='Juan') | Q(nombre_ _startswith='Pedro'))
  Modelo.objects.filter(Q(nombre_ _startswith='Juan') & Q(nombre_ _startswith='Pedro'))

- **EXCLUIR ELEMENTOS**

  Modelo.objects.exclude(salary_ _gt=5000)

| = OR
& = AND

DJANGO ORM – SELECCIONAR COLUMNAS

- **COMO SELECCIONAR SOLO ALGUNAS COLUMNAS**

```
Modelo.objects.all()
Modelo.objects.all().values_list('nombre', 'email')
Modelo.objects.all().values('nombre', 'email')
Modelo.objects.all().only('nombre', 'email') //INCLUYE TAMBIEN EL ID
```

DJANGO ORM – FUNCIONES DE AGREGACION

FUNCIONES DISPONIBLES PARA AGREGACION

```
from django.db.models import Avg, Sum, Max, Min, Count

avg = Modelo.objects.all().aggregate(Avg('salary'))       //PROMEDIO
sum = Modelo.objects.all().aggregate(Sum('salary'))       //SUMA
count = Modelo.objects.all().aggregate(Count('salary'))       //CONTAR
max = Modelo.objects.all().aggregate(Max('salary'))       //MAXIMO
min = Modelo.objects.all().aggregate(Min('salary'))       //MINIMO
```

DJANGO ORM – AGREGAR ELEMENTOS

- **INSERT**

  producto = Producto(nombre="Pepsi", precio="1500")
  producto.save()

  Producto.objects.create(nombre="Pepsi", precio="1500")

DJANGO ORM – AGREGAR ELEMENTOS EN LOTES

- **INSERT**

```
Producto.objects.bulk_create([
Producto(nombre="Pepsi", precio="1500"),
Producto(nombre="Mal Paso", precio="8500"),
Producto(nombre="Hielo", precio="2500")
])
```

DJANGO ORM – ELIMINAR ELEMENTOS

- **DELETE**

  emp = Empleado.objects.get(id=2)
  emp.delete()

- **MULTIPLES ELIMINACIONES**

  qs = Empleado.objects.filter(sueldo_ _gt=5000)
  qs.delete()

  Empleados.objects.all().delete(). //TODOS LOS EMPLEADOS

DJANGO ORM – ACTUALIZAR ELEMENTOS

- **UPDATE**

```
emp = Empleado.objects.get(id=2)
emp.nombre = “Otro Nombre"
emp.save()
```

DJANGO ORM – ORDENAR LOS RESULTADOS

- **ORDER_BY**
  emp = Empleado.objects.all() //ORDENADOS POR DEFECTO (ID)
  emp = Empleado.objects.all().order_by('salary') //ASCENDENTE
  emp = Empleado.objects.all().order_by('-salary') //DESCENDENTE

- **TOP RECORDS**
  emp = Empleado.objects.all() [0:3] //TRES PRIMEROS RESULTADOS

- **IGNORAR CASE MAYSUCULAS / MINUSCULAS**
  from django.db.models.functions import Lower
  emps = Empleado.objects.all().order_by(Lower('nombre'))

https://docs.djangoproject.com/en/4.0/ref/models/database-functions/

DJANGO APPLICATIONS

DJANGO FORMS

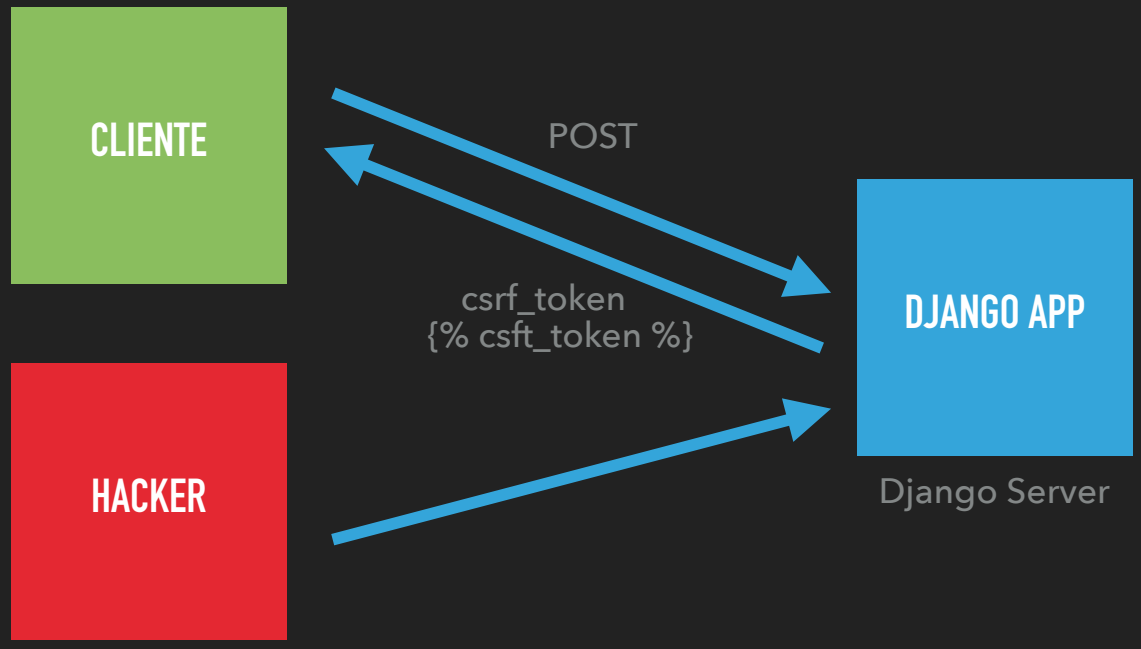
DJANGO FORMS

- Creamos un archivo nuevo forms.py dentro de la app

```python
from django import forms

class UserRegistrationForm(forms.Form):
    nombre = forms.CharField()
    fono = forms.CharField()
    email = forms.CharField()
```

https://docs.djangoproject.com/es/4.1/topics/forms/

CSRF – CROSS SITE REQUEST FORGERY



https://docs.djangoproject.com/es/4.1/topics/forms/

DJANGO FORMS

- Usar el formulario en los views

```python
from django.shortcuts import render
from . import forms

# Create your views here.
def userRegistrationView(request):
    form = forms.UserRegistrationForm()
    data = {'form' : form}
    return render(request, 'userRegistration.html', data)
```

views.py

```python
from formApp.views import userRegistrationView

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', userRegistrationView)
]
```

urls.py

```html
<body class="container mt-5">
    <div class="alert alert-info display-1 text-center">USER REGISTRATION</div>

    <form method="POST">
        {{ form.as_p }}
        {% csrf_token %}
        <input type="submit" class="btn btn-info" value="ENVIAR">
    </form>

</body>
```

userRegistration.html

https://docs.djangoproject.com/es/4.1/topics/forms/

DJANGO FORMS

- Formas para renderizar el formulario en pantalla

Form rendering options

There are other output options though for the **<label>/<input>** pairs:

- **{{ form.as_div }}** will render them wrapped in ** ** tags.
- **{{ form.as_table }}** will render them as table cells wrapped in **<tr>** tags.
- **{{ form.as_p }}** will render them wrapped in **<p>** tags.
- **{{ form.as_ul }}** will render them wrapped in **<li>** tags.

Note that you'll have to provide the surrounding **<table>** or **<ul>** elements yourself.

https://docs.djangoproject.com/es/4.1/topics/forms/

DJANGO FORMS

- Procesar el formulario en la vista

```python
# Create your views here.
def userRegistrationView(request):
    form = forms.UserRegistrationForm()

    if request.method == 'POST':
        form = forms.UserRegistrationForm(request.POST)
        if form.is_valid():
            print("Form es Valido")
            print("Nombre: ", form.cleaned_data['nombre'])
            print("Email: ", form.cleaned_data['email'])
            print("Fono: ", form.cleaned_data['fono'])


    data = {'form' : form}
    return render(request, 'userRegistration.html', data)
```

views.py

https://docs.djangoproject.com/es/4.1/topics/forms/

TIPOS DE INPUTS

- AGREGANDO DIFERENTES TIPOS DE INPUTS CON LOS WIDGETS

```python
class UserRegistrationForm(forms.Form):

    ESTADOS = [('activo', 'ACTIVO'), ('inactivo', 'INACTIVO')]

    nombre = forms.CharField()
    fono = forms.CharField()
    email = forms.CharField()
    password = forms.CharField(widget=forms.PasswordInput())
    estado = forms.CharField(widget=forms.Select(choices=ESTADOS))

    nombre.widget.attrs['class'] = 'form-control'
    fono.widget.attrs['class'] = 'form-control'
    email.widget.attrs['class'] = 'form-control'
    password.widget.attrs['class'] = 'form-control'
    estado.widget.attrs['class'] = 'form-control'
```

PERMITE AGREGAS CLASES CSS A LOS INPUTS

https://docs.djangoproject.com/es/4.1/ref/forms/widgets/

VALIDATORS

- Validaciones por defecto

```
nombre = forms.CharField()
fono = forms.CharField(required=False)
email = forms.CharField()
password = forms.CharField(widget=forms.PasswordInput())
estado = forms.CharField(widget=forms.Select(choices=ESTADOS))
codigo = forms.IntegerField()
```

forms.py

```
from django import forms
from django.core import validators

class UserRegistrationForm(forms.Form):

    ESTADOS = [('activo', 'ACTIVO'), ('inactivo', 'INACTIVO')]

    nombre = forms.CharField(validators=[
        validators.MinLengthValidator(5),
        validators.MaxLengthValidator(20)
    ])
```

forms.py

- Validaciones personalizadas

```
def clean_nombre(self):
    inputNombre = self.cleaned_data['nombre']
    if len(inputNombre) > 20 :
        raise forms.ValidationError("El largo máximo del nombre son 20 caracteres.")
    return inputNombre

def clean_email(self):
    inputEmail = self.cleaned_data['email']
    if inputEmail.find('@') == -1 :
        raise forms.ValidationError("El correo debe contener @")
    return inputEmail
```

forms.py

```
def clean(self):
    user_clean_data = super().clean()

    inputNombre = user_clean_data['nombre']
    if len(inputNombre) > 20 :
        raise forms.ValidationError("El largo máximo del nombre son 20 caracteres..")

    inputEmail = user_clean_data['email']
    if inputEmail.find('@') == -1 :
        raise forms.ValidationError("El correo debe contener @")
```

https://docs.djangoproject.com/en/4.0/ref/forms/validation/

DJANGO APPLICATIONS

MODEL FORMS

MODELS FORMS

- Crear Proyecto y Aplicación
- Crear Modelo
- Crear Formulario

CREAR MODELO Y FORMULARIO

```python
# Create your models here.
class Proyecto(models.Model):
    fechaInicio = models.DateField()
    fechaTermino = models.DateField()
    nombre = models.CharField(max_length=50)
    responsable = models.CharField(max_length=50)
    prioridad = models.IntegerField()
```

models.py

```python
from django import forms
from modelFormApp.models import Proyecto
class FormProyecto(forms.ModelForm):
    class Meta:
        model = Proyecto
        fields = '__all__'
```

forms.py

CREAR VISTA Y URLS

```python
from django.shortcuts import render
from modelFormApp.forms import FormProyecto
from modelFormApp.models import Proyecto

# Create your views here.
def index(request):
    return render(request, 'index.html')

def listadoProyectos(request):
    proyectos = Proyecto.objects.all()
    data = {'proyectos': proyectos}
    return render(request, 'proyectos.html', data)

def agregarProyecto(request):
    form = FormProyecto()
    if request.method == 'POST' :
        form = FormProyecto(request.POST)
        if form.is_valid() :
            form.save()
        return index(request)
    data = {'form' : form}
    return render(request, 'agregarProyecto.html', data)
```

views.py

```python
from modelFormApp import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.index),
    path('proyectos/', views.listadoProyectos),
    path('agregarProyecto/', views.agregarProyecto),
]
```

urls.py

CREAR TEMPLATE

```
<body class="container mt-5">
    <div class="alert alert-info display-1 text-center">SISTEMA ADMINISTRADOR</div>

    <a href="/proyectos" class="btn btn-info">PROYECTOS</a>
</body>
```

Index.html

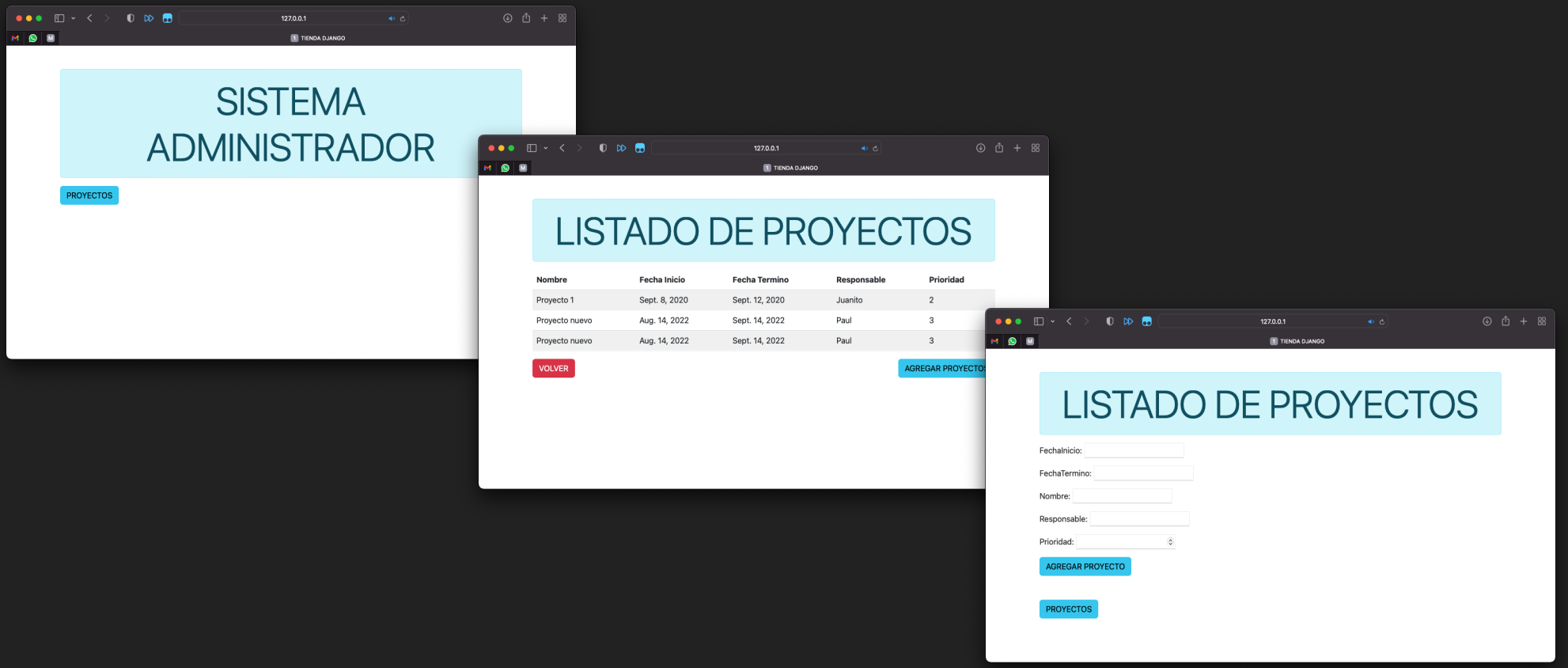
```
<body class="container mt-5">
    <div class="alert alert-info display-1 text-center">LISTADO DE PROYECTOS</div>

    <form method = 'post'>
        {{ form.as_p }}
        {% csrf_token %}
        <input type="submit" value="AGREGAR PROYECTO" class="btn btn-info">
    </form>


    <a href="../proyectos" class="btn btn-info mt-5">PROYECTOS</a>
</body>
```

agregarProyecto.html

```
<body class="container mt-5">
    <div class="alert alert-info display-1 text-center">LISTADO DE PROYECTOS</div>

    {% if proyectos %}

    <table class="table table-striped table-inverse table-responsive">
        <thead class="thead-inverse">
            <tr>
                <th>Nombre</th>
                <th>Fecha Inicio</th>
                <th>Fecha Termino</th>
                <th>Responsable</th>
                <th>Prioridad</th>
            </tr>
            </thead>
            <tbody>
                {% for proyecto in proyectos %}
                <tr>
                    <td>{{ proyecto.nombre }}</td>
                    <td>{{ proyecto.fechaInicio }}</td>
                    <td>{{ proyecto.fechaTermino }}</td>
                    <td>{{ proyecto.responsable }}</td>
                    <td>{{ proyecto.prioridad }}</td>
                </tr>
                {% endfor %}
            </tbody>
    </table>

    {% else %}

    <div class="alert alert-primary" role="alert">
        <strong>No se encuentran proyectos en el sistema</strong>
    </div>


    {% endif %}


    <a href="../" class="btn btn-danger">VOLVER</a>
    <a href="../agregarProyecto" class="btn btn-info float-end">AGREGAR PROYECTOS</a>
</body>
</html>
```

proyectos.html

RESULTADO


127.0.0.1
TIENDA DJANGO
SISTEMA ADMINISTRADOR
PROYECTOS
127.0.0.1
TIENDA DJANGO
LISTADO DE PROYECTOS
Nombre
Fecha Inicio
Fecha Termino
Responsable
Prioridad
Proyecto 1
Sept. 8, 2020
Sept. 12, 2020
Juanito
2
Proyecto nuevo
Aug. 14, 2022
Sept. 14, 2022
Paul
3
Proyecto nuevo
Aug. 14, 2022
Sept. 14, 2022
Paul
3
VOLVER
AGREGAR PROYECTOS
127.0.0.1
TIENDA DJANGO
LISTADO DE PROYECTOS
FechaInicio:
FechaTermino:
Nombre:
Responsable:
Prioridad:
AGREGAR PROYECTO
PROYECTOS

ACTUALIZAR Y ELIMINAR UN PROYECTO

```
<table class="table table-striped table-inverse table-responsive">
    <thead class="thead-inverse">
        <tr>
            <th>Nombre</th>
            <th>Fecha Inicio</th>
            <th>Fecha Termino</th>
            <th>Responsable</th>
            <th>Prioridad</th>
            <th>-</th>
        </tr>
        </thead>
        <tbody>
            {% for proyecto in proyectos %}
            <tr>
                <td>{{ proyecto.nombre }}</td>
                <td>{{ proyecto.fechaInicio }}</td>
                <td>{{ proyecto.fechaTermino }}</td>
                <td>{{ proyecto.responsable }}</td>
                <td>{{ proyecto.prioridad }}</td>
                <td><a href="/actualizarProyecto/{{ proyecto.id }}" class="btn btn-success btn-sm"><i class="bi bi-pencil"></i></td>
                <td><a href="/eliminarProyecto/{{ proyecto.id }}" class="btn btn-danger btn-sm"><i class="bi bi-trash"></i></td>
            </tr>
            {% endfor %}
        </tbody>
</table>
```

proyectos.html

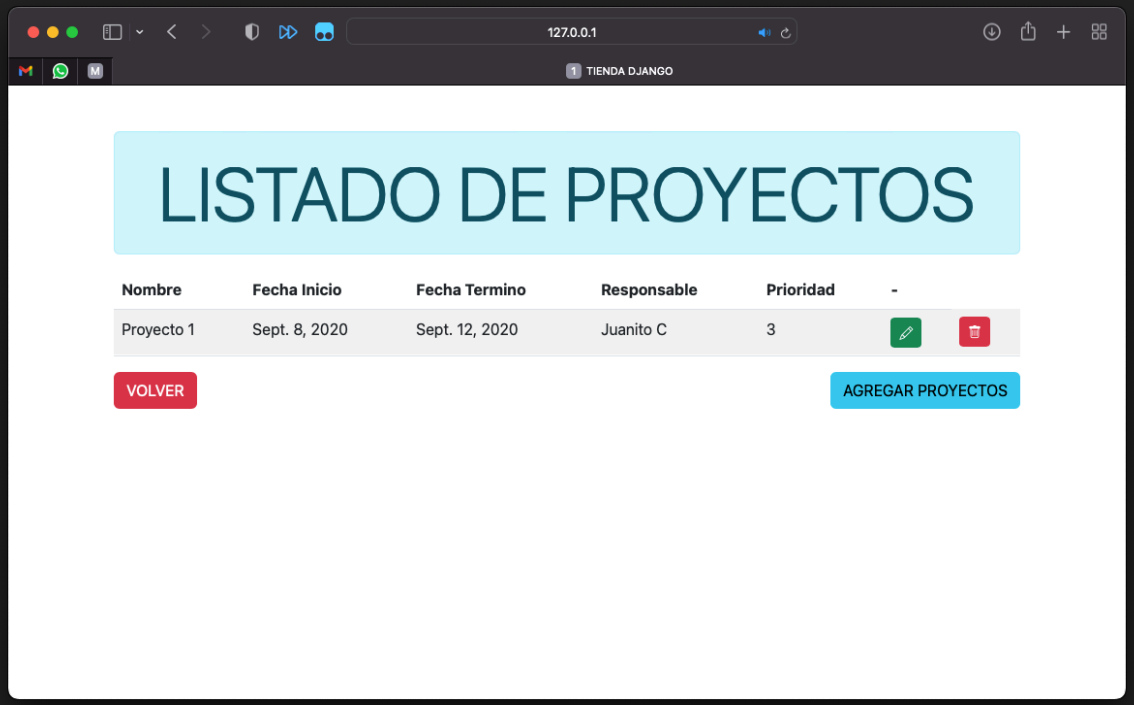
ACTUALIZAR Y ELIMINAR UN PROYECTO

```python
def eliminarProyecto(request, id):
    proyecto = Proyecto.objects.get(id = id)
    proyecto.delete()
    return redirect('/proyectos')

def actualizarProyecto(request, id):
    proyecto = Proyecto.objects.get(id = id)
    form = FormProyecto(instance=proyecto)
    if request.method == 'POST' :
        form = FormProyecto(request.POST, instance=proyecto)
        if form.is_valid() :
            form.save()
        return index(request)
    data = {'form' : form}
    return render(request, 'agregarProyecto.html', data)
```

views.py

```python
from modelFormApp import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.index),
    path('proyectos/', views.listadoProyectos),
    path('agregarProyecto/', views.agregarProyecto),
    path('eliminarProyecto/<int:id>', views.eliminarProyecto),
    path('actualizarProyecto/<int:id>', views.actualizarProyecto),
]
```

urls.py



proyectos.html