EXERCISES QUE TRABAJAREMOS EN EL CUE

- **EXERCISE 1: CREAR UNA APLICACIÓN APP QUE CONTENGA UN CRUD LLAMADO CRUDAPP**
- **EXERCISE 2: CONFIGURACIÓN DE LAS URLS DEL PROYECTO Y LA APLICACIÓN**
- **EXERCISE 3: CREACIÓN DE LAS VISTAS DE LA APLICACIÓN**
- **EXERCISE 4: CREANDO EL MODELO**
- **EXERCISE 5: REGISTRAR EL MODELO EN EL SITIO ADMINISTRATIVO DE DJANGO**
- **EXERCISE 6: CREAR LAS MIGRACIONES DE LA APLICACIÓN AL PROYECTO**
- **EXERCISE 7: CREACIÓN DE LAS VISTAS PARA INSERTAR**
- **EXERCISE 8: CREACIÓN DE LAS VISTAS PARA MOSTRAR**
- **EXERCISE 9: CREACIÓN DE LA VISTA DE ACTUALIZACIÓN**
- **EXERCISE 10: CREACIÓN DE LA VISTA DE ELIMINACIÓN**

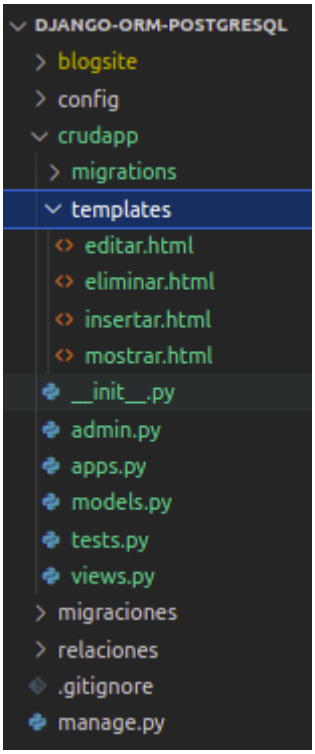
EXERCISE 1: CREAR UNA APLICACIÓN APP QUE CONTENGA UN CRUD LLAMADO CRUDAPP

A continuación, construiremos una aplicación **crudapp** como parte del proyecto. También ejecutaremos tareas CRUD más adelante. Y, para crear una aplicación, ejecute el siguiente comando en la terminal:

```
$ python manage.py startapp crudapp
```

Se crea la carpeta Plantillas dentro del directorio de la aplicación crudapp, y crearemos archivos HTML dentro de dicha carpeta.

- `mostrar.html:` para mostrar los datos obtenidos.
- `insertar.html:` para agregar un nuevo empleado.
- `editar.html:` para actualizar el empleado existente.
- `eliminar.html:` para eliminar una entidad de empleado.



En el archivo `settings.py` debemos incluir la aplicación crudapp:

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    # App Local
    'blogsite.apps.BlogsiteConfig',
    'relaciones.apps.RelacionesConfig',
    'migraciones.apps.MigracionesConfig',
    'crudapp.apps.CrudappConfig',
]
```

Previamente ya hemos configurado el directorio `Templates`, y la base de datos.

EXERCISE 2: CONFIGURACIÓN DE LAS URLS DEL PROYECTO Y LA APLICACIÓN

Las URLs del proyecto. Agregue el siguiente código en el archivo `urls.py` del proyecto.

CONFIG/URLS.PY

```python
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('post/', include('blogsite.urls'))
    path('crudapp/', include('crudapp.urls')),
]
```

Las URLs de la aplicación: en primer lugar, debemos crear un archivo con el nombre `urls.py` en el directorio de la aplicación. Ahora, definimos la ruta de las diferentes vistas creadas en el archivo views.py de la aplicación.

CRUDAPP/URLS.PY

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

urlpatterns = [
    path('', views.insertar_emp, name='insertar-emp'),
    path('mostrar/', views.mostrar_emp, name='mostrar-emp'),
    path('editar/<int:pk>', views.editar_emp, name='editar-emp'),
    path('eliminar/<int:pk>', views.eliminar_emp, name='eliminar-emp'),
]
```

Aquí creamos las siguientes vistas:

- **`insertar_emp:`** para insertar el detalle del empleado.
- **`mostrar_emp:`** para mostrar los detalles de los empleados.
- **`editar_emp:`** para actualizar los detalles existentes de un empleado.
- **`eliminar_emp:`** para eliminar un empleado existente.

EXERCISE 3: CREACIÓN DE LAS VISTAS DE LA APLICACIÓN

En esta sección, veremos la función de vistas que tenemos que crear en el archivo `views.py`.

CRUDAPP/VIEWS.PY

```python
from django.shortcuts import render, redirect
from .forms import EmpleadoForm
from .models import Empleado

# Crear Empleados

def insertar_emp(request):
    pass

# obtener los Empleados

def mostrar_emp(request):
    pass

# Editar Empleados
def editar_emp(request):
    pass

# Eliminar Empleados

def eliminar_emp(request):
    pass
```

EXERCISE 4: CREANDO EL MODELO

Para crear el modelo se debe agregar el siguiente código dentro del archivo `models.py` de la aplicación crudapp.

CRUDAPP/MODELS.PY

```python
from django.db import models

class Empleado(models.Model):
    emp_id = models.CharField(max_length=3)
    emp_nombre = models.CharField(max_length=200)
    emp_correo = models.EmailField()
    emp_designacion = models.CharField(max_length=150)

    class Meta:
        db_table="Employee"
```

EXERCISE 5: REGISTRAR EL MODELO EN EL SITIO ADMINISTRATIVO DE DJANGO

Agregue el siguiente código en el archivo `admin.py`.

CRUDAPP/ADMIN.PY

```
1 from django.contrib import admin
2 from .models import Empleado
3
4 admin.site.register(Empleado)
```

Para registrar el modelo de empleado en el sitio de administración, usamos la función `admin.site.register()` con el nombre del modelo.

EXERCISE 6: CREAR LAS MIGRACIONES DE LA APLICACIÓN AL PROYECTO

Si los modelos han sido modificados, este comando prepara un archivo `makemigrations` para nuestro nuevo modelo, o crea un nuevo archivo de migraciones para cualquier cambio. Las modificaciones de la base de datos no se crean ni se ven afectadas por él.

```
 1 $ python manage.py makemigrations
 2 Migrations for 'crudapp':
 3   crudapp/migrations/0001_initial.py
 4     - Create model Empleado
 5
 6 $ python manage.py migrate
 7 Operations to perform:
 8   Apply all migrations: admin, auth, blogsite, contenttypes, crudapp,
 9 migraciones, relaciones, sessions
10 Running migrations:
11   Applying crudapp.0001_initial... OK
```

EXERCISE 7: CREACIÓN DE LAS VISTAS PARA INSERTAR

Editamos el archivo de vistas.

CRUDAPP/VIEWS.PY

```
1 # Crear Empleados
2
3 def insertar_emp(request):
4     if request.method == "POST":
5         emp_id = request.POST['emp_id']
6         emp_nombre = request.POST['emp_nombre']
```

```
        emp_correo = request.POST['emp_correo']
        emp_designacion = request.POST['emp_designacion']
        empleado = Empleado(emp_id=emp_id, emp_nombre=emp_nombre,
emp_correo=emp_correo,
                            emp_designacion= emp_designacion)
        empleado.save()
         return redirect('mostrar/')
    else:
        return render(request, 'insertar.html')
```

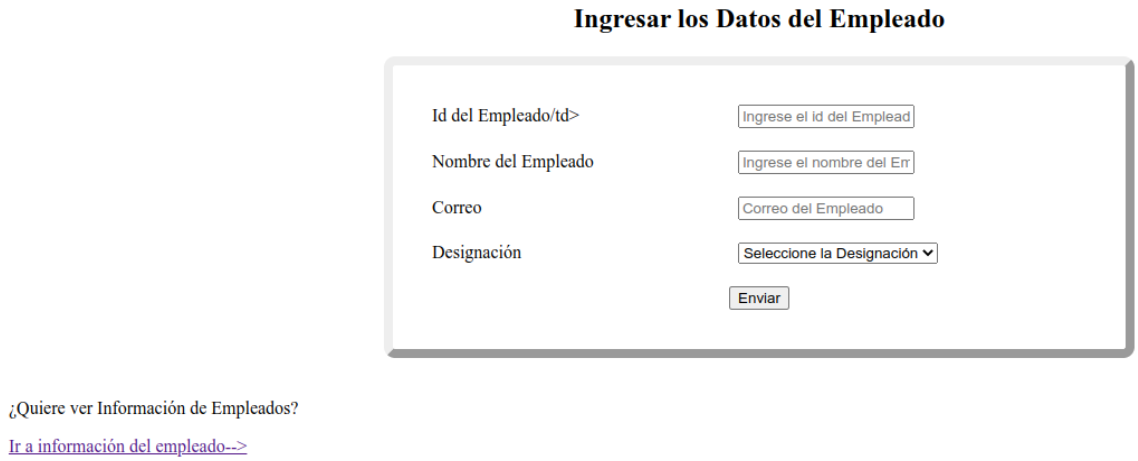
Editamos la plantilla HTML para insertar datos de empleados.

CRUDAPP/TEMPLATES/INSERTAR.HTML

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Empleados</title>
</head>
<style>
    table {
        border: 8px outset;
        border-radius: 10px;
        border-spacing: 10px;
        padding: 20px;
        margin-left: auto;
        margin-right: auto;
    }

    th,
    td {
        padding: 5px;
    }
</style>

<body>
    <h2 style="text-align:center"> Ingresar los Datos del Empleado</h2>
    <form method="POST">
        {% csrf_token %}
        <table style="width:50%" align="center">
            <tr>
                <td>Id del Empleado/td>
                <td><input type="text" placeholder="Ingrese el id del
Empleado" name="emp_id"</td>
            </tr>
```

```
            <tr>
                <td>Nombre del Empleado</td>
                <td><input type="text" placeholder="Ingrese el nombre
del Empleado" name="emp_nombre"</td>
            </tr>
            <tr>
                <td>Correo</td>
                <td><input type="email" placeholder="Correo del
Empleado" name="emp_correo" </td>
            </tr>
            <tr>
                <td>Designación</td>
                <td><select name="emp_designacion">
                        <option selected disabled=true>Seleccione la
Designación</option>
                        <option> Project Manager </option>
                        <option> Programador </option>
                        <option> Soporte Técnico </option>
                        <option> Desarrollador Web </option>
                        </option>
                    </select></td>
            </tr>
            <tr>
                <td colspan="2" align="center"><input type="submit"
class="btn btn-success"> </td>
            </tr>
        </table>

        <br><br>
        <div class="alert alert-danger" role="alert">
            ¿Quiere ver Información de Empleados?
        </div>

        <p>
            <a href="{% url 'mostrar-emp' %}">Ir a información del
empleado--></a>
        </p>
    </form>
</body>

</html>

```

Consulta el ingreso en: http://localhost:8000/crudapp/



EXERCISE 8: CREACIÓN DE LAS VISTAS PARA MOSTRAR

En esta sección aprenderemos a crear una función de visualización que muestre la información de todos los empleados agregados por el usuario.

Agrega el siguiente código en el archivo `views.py`.

CRUDAPP/VIEWS.PY

```
# obtener los Empleados

def mostrar_emp(request):
    empleados = Empleado.objects.all()
    return render(request,'mostrar.html',{'empleados':empleados} )
```

Editamos el archivo de la plantilla que muestra los empleados:

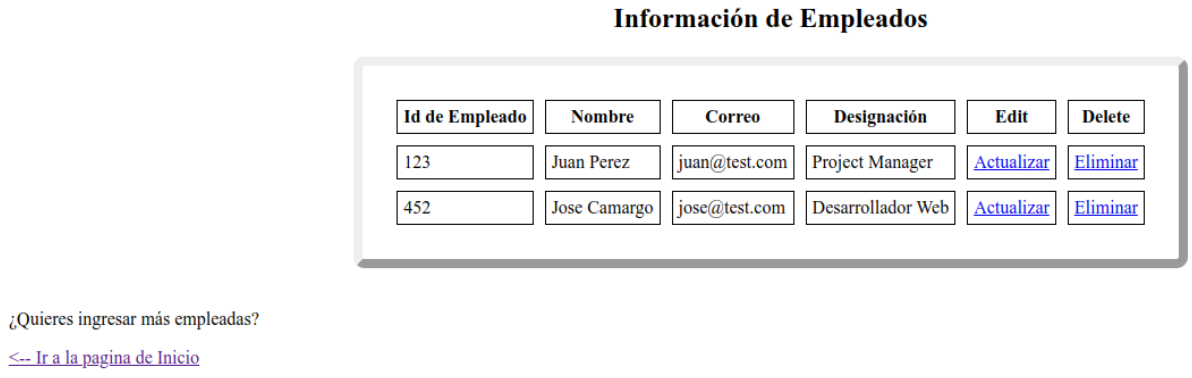
CRUDAPP/TEMPLATES/MOSTRAR.HTML

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Buscar Empleados</title>
</head>
<style>
    table {
        border: 8px outset;
```

```
13          border-radius: 10px;
14          border-spacing: 10px;
15          padding: 20px;
16          margin-left: auto;
17          margin-right: auto;
18      }
19
20      th,
21      td {
22          padding: 5px;
23          border: 1px solid;
24      }
25  </style>
26
27  <body>
28      <h2 style="text-align:center"> Información de Empleados </h2>
29      <table align="center" style="margin: 0px auto;">
30          <thead>
31              <tr>
32                  <th>Id de Empleado</th>
33                  <th>Nombre</th>
34                  <th>Correo</th>
35                  <th>Designación</th>
36                  <th>Edit</th>
37                  <th>Delete</th>
38              </tr>
39          </thead>
40          <tbody>
41              {% for empleado in empleados %}
42              <tr>
43                  <td>{{empleado.emp_id}}</td>
44                  <td>{{empleado.emp_nombre}}</td>
45                  <td>{{empleado.emp_correo}}</td>
46                  <td>{{empleado.emp_designacion}}</td>
47                  <td>
48                      <a href="/editar/{{empleado.pk}}">Actualizar</a>
49                  </td>
50                  <td>
51                      <a href="/eliminar/{{empleado.pk}}">Eliminar</a>
52                  </td>
53              </tr>
54              {% endfor %}
55          </tbody>
56      </table>
57
58      <br><br>
59      <div class="alert alert-danger" role="alert">
60          ¿Quieres ingresar más empleadas?
61      </div>
62
63      <p>
```

```
64          <a href="{% url 'insertar-emp' %}"><-- Ir a la pagina de
65 Ixnicio</a>
66      </p>
67 </body>
68
69 </html>
70
```

Consultando en: http://localhost:8000/crudapp/mostrar/



Información de Empleados

Id de Empleado	Nombre	Correo	Designación	Edit	Delete
123	Juan Perez	juan@test.com	Project Manager	Actualizar	Eliminar
452	Jose Camargo	jose@test.com	Desarrollador Web	Actualizar	Eliminar

¿Quieres ingresar más empleadas?

<-- Ir a la pagina de Inicio

EXERCISE 9: CREACIÓN DE LA VISTA DE ACTUALIZACIÓN

En esta sección aprenderemos a crear una función de visualización que actualice la información del empleado existente.

CRUDAPP/VIEWS.PY

```python
# Editar Empleados
def editar_emp(request,pk):
    empleado = Empleado.objects.get(id=pk)
    # if request.method == 'POST':
    #     return redirect('/crudapp/mostrar')

    context = {
        'empleado': empleado,
    }

    return render(request=request, template_name='editar.html',
context=context)

def actualizarempleado(request, id):
    emp_id = request.POST['emp_id']
    emp_nombre = request.POST['emp_nombre']
    emp_correo = request.POST['emp_correo']
    emp_designacion = request.POST['emp_designacion']
    empleado = Empleado.objects.get(id=id)
    empleado.emp_id = emp_id
    empleado.emp_nombre = emp_nombre
    empleado.emp_correo = emp_correo
    empleado.emp_designacion = emp_designacion
    empleado.save()
    return redirect('/crudapp/mostrar')
```

Agregamos una nueva URL:

CRUDAPP/URLS.PY

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

urlpatterns = [
    path('', views.insertar_emp, name='insertar-emp'),
    path('mostrar/', views.mostrar_emp, name='mostrar-emp'),
    path('editar/<int:pk>', views.editar_emp, name='editar-emp'),
    path('editar/actualizarempleado/<int:id>', views.actualizarempleado,
name='actualizarempleado'),
    path('eliminar/<int:pk>/', views.eliminar_emp, name='eliminar-emp'),
]
```

Creamos la plantilla de editar:

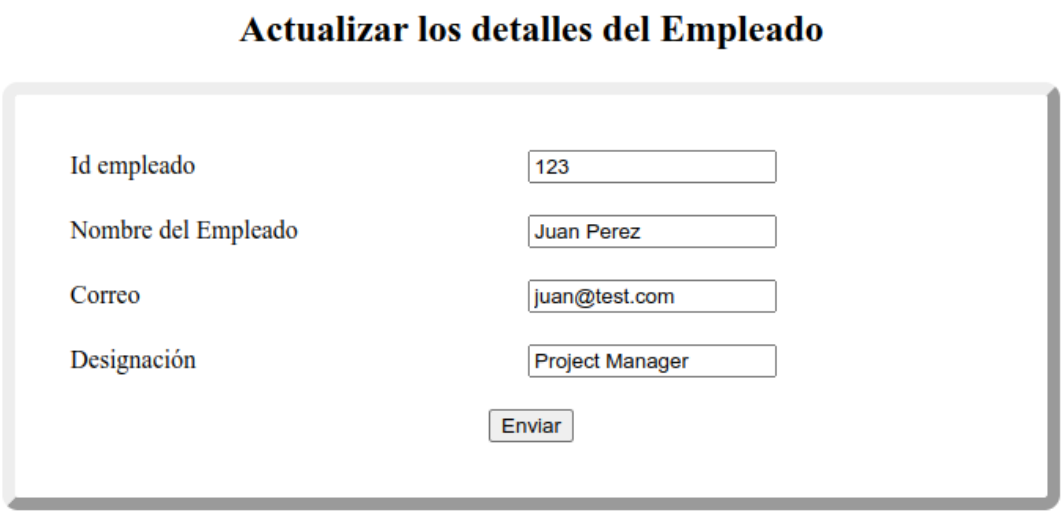
CRUDAPP/TEMPLATES/EDITAR.HTML

```html
<!DOCTYPE html>
<html lang="en">
```

```
2
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-
7  scale=1.0">
8      <title>Actualizar Empleados</title>
9  </head>
10 <style>
11     table {
12         border: 8px outset;
13         border-radius: 10px;
14         border-spacing: 10px;
15         padding: 20px;
16         margin-left: auto;
17         margin-right: auto;
18     }
19
20     th,
21     td {
22         padding: 5px;
23     }
24 </style>
25
26 <body>
27     <h2 style="text-align:center"> Actualizar los detalles del
28 Empleado</h2>
29     <form action="actualizarempleado/{{ empleado.id }}"method="POST">
30         {% csrf_token %}
31         <table style="width:50%" align="center">
32             <tr>
33                 <td>Id empleado</td>
34                 <td><input type="text" value="{{ empleado.emp_id }}"
35 name="emp_id"</td>
36             </tr>
37             <tr>
38                 <td>Nombre del Empleado</td>
39                 <td><input type=" text" value="{{ empleado.emp_nombre
40 }}" name="emp_nombre" </td>
41             </tr>
42             <tr>
43                 <td>Correo</td>
44                 <td><input type="email" value="{{ empleado.emp_correo
45 }}" name="emp_correo" </td>
46             </tr>
47             <tr>
48                 <td>Designación</td>
49                 <td><input type="text" value="{{
50 empleado.emp_designacion}}" name="emp_designacion" </td>
51             </tr>
52             <tr>
```

```
                <td colspan="2" align="center"><input type="submit"
class="btn btn-success"> </td>
            </tr>
        </table>
    </form>
</body>

</html>

```

Al ingresar o presionar en Actualizar, se muestra la siguiente imagen:



EXERCISE 10: CREACIÓN DE LA VISTA DE ELIMINACIÓN

Crear una función de visualización que elimine la entrada especificada de un registro.

CRUDAPP/VIEWS.PY

```
def eliminar_emp(request, pk):
    empleado = Empleado.objects.get(id=pk)

    if request.method == 'POST':
        empleado.delete()
        return redirect('/crudapp/mostrar')

    context = {
        'empleado': empleado,
    }

    return render(request, 'eliminar.html', context)
```

Creamos el template para la eliminación de registro de empleado:

CRUDAPP/TEMPLATES/ELIMINAR.HTML

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Eliminar un Empleado</title>
</head>

<body>
    <form action="" method="POST">
        {% csrf_token %}

        <br><br>
        <div class="alert alert-danger" role="alert">
            Estas seguro que deseas eliminar al "{{
empleado.emp_nombre}}"?
        </div>

        <p>
            <a href="{% url 'mostrar-emp' %}"><-- Retornar</a>
        </p>

        <p>
            <input class="btn btn-danger" type="submit" value="Confirm">
        </p>
    </form>
</body>

</html>
```

Al seleccionar **Eliminar**, obtenemos: