



Universidad
Nacional
de Loja

[Desarrollo de Software basado en modelos]

Trabajo Final

Software de administración de empleados VIVENTO

Tutor:

RENE ROLANDO ELIZALDE SOLANO

Elaborado:

Janneth Guamán

Lenin Quizhpe

18 de noviembre de 2022

Problemática:

La gestión de empleados en una empresa mediana o grande es de vital importancia para una sólida gestión del departamento de recursos humanos, quienes son los encargados del seguimiento y bienestar de la fuerza laboral. Una empresa que dependa del recurso humano debe manejar un sistema que le permita identificar con facilidad las características propias del empleado, basados en su área de trabajo, experiencia laboral, enfermedades, discapacidad, riesgos, etc. Al conocer estas características y ser llevadas bajo un sistema de administración de empleados, permitirá mantener un ambiente laboral excelente, así mismo, la empresa tendrá conocimiento del acontecer del personal y podrá actuar con precisión ante cualquier dificultad que pueda presentarse, permitirá tener una mejor organización en las tareas asignadas al empleado, de acuerdo a su experiencia y conocimiento en el área, mejorando la productividad de la empresa y tomando el recurso humano de manera más eficiente.

Descripción de las Tecnologías:

Tecnología	Descripción
Python	Lenguaje de Programación orientado al desarrollo web, permite que los equipos trabajen en colaboración sin barreras significativas de lenguaje y experimentación.
Django	Framework web de alto nivel que permite el desarrollo rápido de sitios web seguros y mantenibles, gratuito y de código abierto
HTML 5	Estándar de referencia del software que conecta con la elaboración de páginas web, define una estructura básica y un código.
XAMPP	Paquete de software libre, que consiste principalmente en el sistema de gestión de bases de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script PHP y Perl.

Descripción de la solución:

El software a desarrollar permitirá crear, eliminar o editar un registro de datos personales de los empleados de la inmobiliaria "VIVENDO", además de ello, se registrará el cargo que tiene y el departamento al cual pertenece, mediante una lista de despliegue y opción a escoger. De la misma manera el usuario podrá crear, editar o eliminar los diferentes cargos y departamentos que puedan ser creados, según las necesidades de la empresa

Avance 1

Previo al desarrollo del código, es preferible determinar los diagramas UML, que nos orienten a identificar el cumplimiento de los requerimientos es por ello que a continuación se presentan los diagramas, identificando sus características y desarrollo:

Figura 1: Diagrama de clase

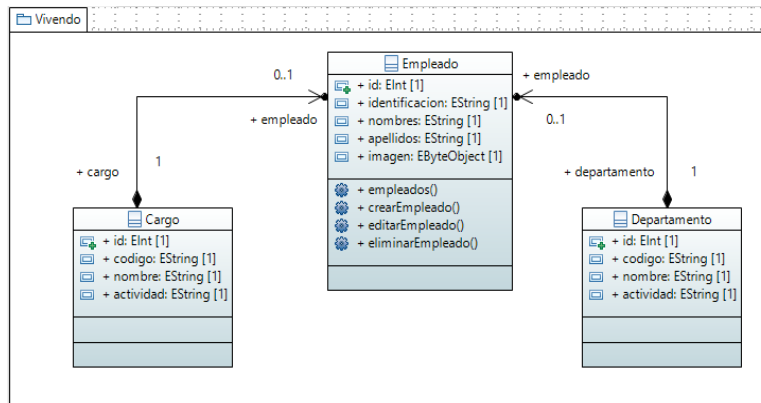


Figura 2: Diagrama de secuencias

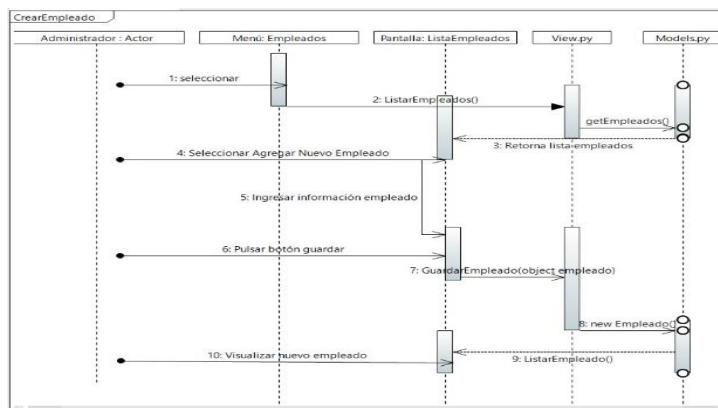
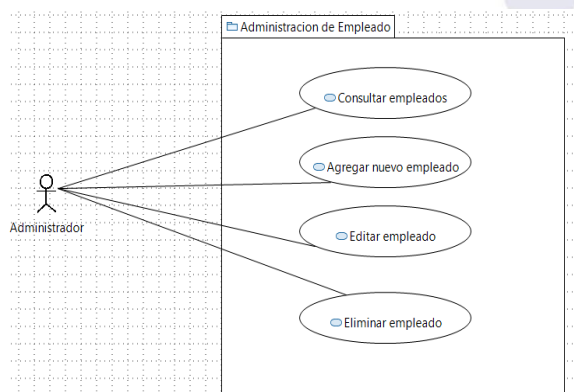


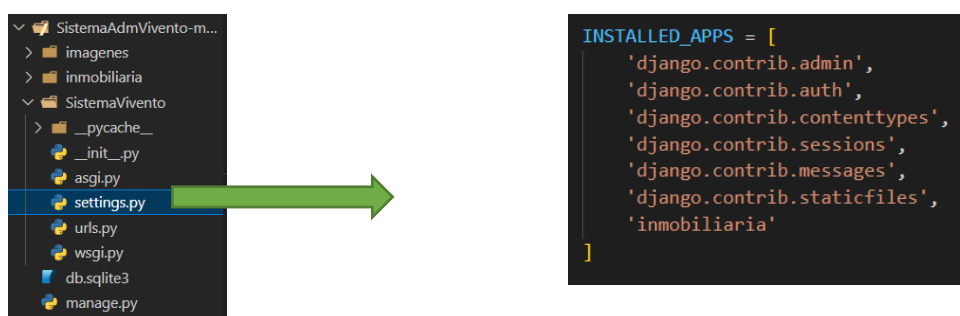
Figura 3: Diagrama de casos de uso



En función a los diagramas UML, se desarrollará el sistema web mediante el uso del framework django y su programación en Python que se detallará a continuación.

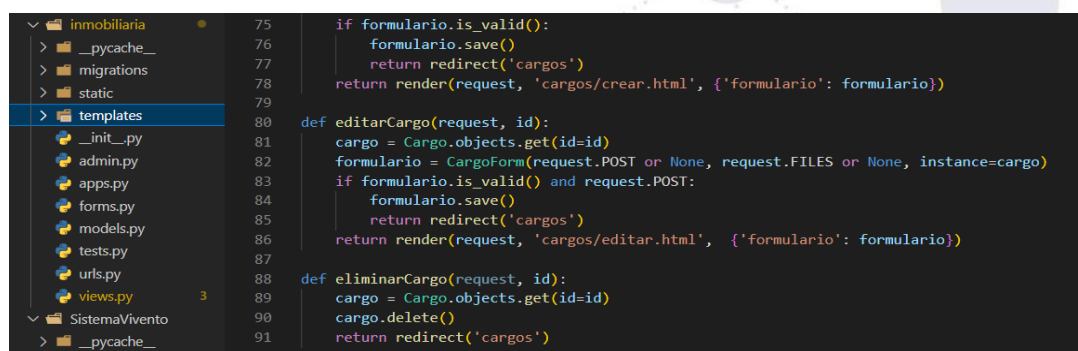
El sistema de gestión de empleados, consta de 3 módulos, “Empleado” (Realiza el registro, edición y eliminación de empleados que se vinculen a la empresa), a este empleado se le asigna un “Cargo”, y su pertenencia a algún “Departamento” de la empresa según sus necesidades. El backend a sido construido de la siguiente manera. Creación de una aplicación de nombre inmobiliaria a partide del settings.py. como se observa en la Figura 4.

Figura 4: Creación de aplicación “Inmobiliaria”



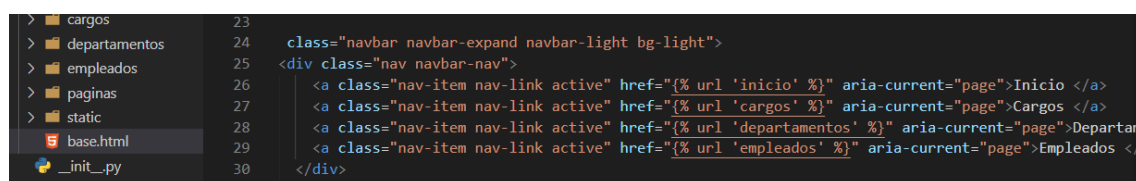
En la aplicación de inmobiliaria se crea una carpeta “templates”, en la cual django busca las páginas (cargos, departamentos, empleados) que van a interactuar con el cliente mediante código html, y su acceso se lo realiza mediante el archivo view.py, las vistas se crean mediante funciones que generan la solicitud a cada una de las páginas de interfaz de usuario “Cargos”, “Empleados” y “Departamentos”, y sus respectivas acciones “crear”, “editar”, como se observa en la Figura 5.

Figura 5: Creación de clases para las vistas de interfaz de usuario.



Creación de plantilla base.html, la cual es una estructura de contenido que se repetirá en todas las vistas, y la creación de botones de menú de acceso a las diferentes páginas, como se observa en la Figura 6.

Figura 6: Creación de contenido y menú de acceso.



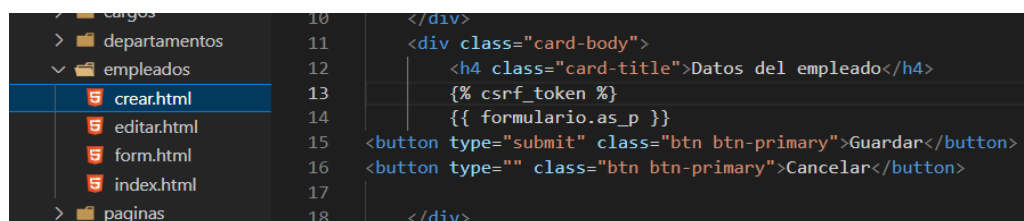
```

23
24
25
26
27
28
29
30
class="navbar navbar-expand navbar-light bg-light">
<div class="nav navbar-nav">
  <a class="nav-item nav-link active" href="{% url 'inicio' %}" aria-current="page">Inicio </a>
  <a class="nav-item nav-link active" href="{% url 'cargos' %}" aria-current="page">Cargos </a>
  <a class="nav-item nav-link active" href="{% url 'departamentos' %}" aria-current="page">Departamentos </a>
  <a class="nav-item nav-link active" href="{% url 'empleados' %}" aria-current="page">Empleados </a>
</div>

```

Creación de las vistas de gestión de datos del CRUD, el cual permitirán crear un formulario para el ingreso de los datos del empleado, editar o eliminar, según se requiera como se observa en la Figura 7a, en la Figura 7b la creación de el archivo form.html, el mismo que captura la información ingresada por el formulario.

Figura 7: vistas de gestión y formularios

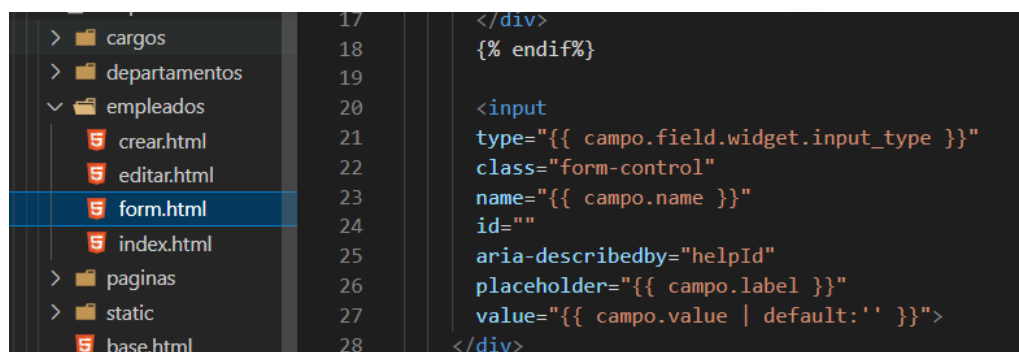


```

10
11
12
13
14
15
16
17
18
</div>
<div class="card-body">
  <h4 class="card-title">Datos del empleado</h4>
  {{ csrf_token }}
  {{ formulario.as_p }}
  <button type="submit" class="btn btn-primary">Guardar</button>
  <button type="" class="btn btn-primary">Cancelar</button>
</div>

```

a) Se crean las páginas de gestión de datos



```

17
18
19
20
21
22
23
24
25
26
27
28
</div>
{% endif%}
<input
  type="{{ campo.field.widget.input_type }}"
  class="form-control"
  name="{{ campo.name }}"
  id=""
  aria-describedby="helpId"
  placeholder="{{ campo.label }}"
  value="{{ campo.value | default:'' }}">
</div>

```

b) Se crea el formulario que guardará la información ingresada.

El archivo index.html permitirá especificar la información contenida en cada página, para nuestro caso el listado de empleados registrados se empleará una tabla detallada como se observa en el Figura 8.

Figura 8: Presentación de información en la página empleados.

```

7 <div class="card">
8   <div class="card-header">
9     <a name="" id="" class="btn btn-primary" href="{% url 'crearEmpleado' %}" role="button">Agregar
10   </div>
11   <div class="card-body">
12     <h4 class="card-title">Empleado</h4>
13
14     <div class="table-responsive">
15       <table class="table table-primary">
16         <thead>
17           <tr>
18             <th scope="col">Identificación</th>
19             <th scope="col">Apellidos</th>
20             <th scope="col">Nombres</th>
21             <th scope="col">Cargo</th>

```

Avance 2

Una vez creadas la parte lógica de software, se requiere realizar la interfaz de la navegación entre las diferentes vistas que tendrá la página web y presentación de información de empleados y su almacenamiento en el base de datos.

Para el acceso a las pantallas se requiere mediante el archivo views.py en el cual se encuentra las funciones que ligán con los archivos html, en la Figura 9, se presenta el llamado de las pantallas para mostrar, crear o editar el registro de empleados.

Figura 9: Acceso a las páginas de listar, crear, o editar información del empleado.

```

18 def empleados(request):
19     empleados = Empleado.objects.all()
20     return render(request, 'empleados/index.html', {'empleados': empleados})
21
22 def crearEmpleado(request):
23     formulario = EmpleadoForm(request.POST or None, request.FILES or None)
24     if formulario.is_valid():
25         formulario.save()
26         return redirect('empleados')
27     return render(request, 'empleados/crear.html', {'formulario': formulario})
28
29 def editarEmpleado(request, id):
30     empleado = Empleado.objects.get(id=id)
31     formulario = EmpleadoForm(request.POST or None, request.FILES or None, instance=empleado)
32     if formulario.is_valid() and request.POST:
33         formulario.save()
34         return redirect('empleados')
35

```

El archivo url.py permite establecer el acceso por url y navegación a las páginas de administración de empleados. Figura 10.

Figura 10: acceso a páginas por url.


```

9      urlpatterns = [
10          path('cargos', views.cargos, name='cargos'),
11          path('cargos/crearCargo', views.crearCargo, name='crearCargo'),
12          path('cargos/editarCargo/', views.editarCargo, name='editarCargo'),
13          path('cargos/editarCargo/<int:id>', views.editarCargo, name='editarCargo'),
14          path('eliminarCargo/<int:id>', views.eliminarCargo, name='eliminarCargo'),
15
16          path('departamentos', views.departamentos, name='departamentos'),
17          path('departamentos/crearDepartamento', views.crearDepartamento, name='crearDepartamento'),
18          path('departamentos/editarDepartamento/', views.editarDepartamento, name='editarDepartamento'),
19          path('departamentos/editarDepartamento/<int:id>', views.editarDepartamento, name='editarDepartamento'),
20          path('eliminarDepartamento/<int:id>', views.eliminarDepartamento, name='eliminarDepartamento'),

```

Para la conexión a la base de datos accedemos a la raíz principal del proyecto y en settings.py y realizar la configuración que se requiera, en el presente caso se uso mysql y XAMPP, la Figura 11a, se observa los parámetros para la base de datos y en la Figura 11b la librería pymysql, requerida para la base de datos, que se ubica en el archivo _init_.html.

Figura 11: Implementación de la base de datos.

```

78      DATABASES = {
79          'default': {
80              'ENGINE': 'django.db.backends.mysql',
81              'NAME': 'inmobiliaria',
82              'USER': 'root',
83              'PASSWORD': '',
84              'HOST': 'localhost',
85              'PORT': '3306'
86          }
87      }
88

```

a) Configuración de la base de datos

```

import pymysql
pymysql.install_as_MySQLdb()

```

b) Importar libreria pymysql

Creación de modelos para ingreso de información, se realiza mediante el archivo models.py, en él se crean las clases para la estructura de datos a ser ingresados por el usuario, inclusive el ingreso de una imagen para identificar al empleado, el código se muestra en la Figura 12.

Figura 12: Ingreso de información del empleado.

```

32      class Empleado(models.Model):
33          id = models.AutoField(primary_key=True)
34          identificacion = models.CharField(max_length=10, verbose_name='Identificación')
35          nombres = models.CharField(max_length=250, verbose_name='Nombres')
36          apellidos = models.CharField(max_length=250, verbose_name='Apellidos')
37          cargo = models.CharField(max_length=250, verbose_name='Cargo')
38          departamento = models.ForeignKey(Departamento, verbose_name='Departamento', null=False, blank=False)
39          imagen = models.ImageField(upload_to='imagenes/', verbose_name='Imagen', null=True)
40
41
42      def __str__(self):
43          fila = "Nombres: " + self.nombres + " " + "Apellidos:" + self.apellidos
44          return fila
45

```

En el siguiente apartado se registran las vistas que pueden ser ingresadas, de manera directa ubicando la vista al url de navegación, sin la necesidad de ingresar al admin de Django,

```
from django.contrib import admin
from .models import *
# Register your models here.
admin.site.register(Cargo)
admin.site.register(Departamento)
admin.site.register(Empleado)
```

A continuación, se presentan la interfaz de usuario en el que permitirá ingresar a nuevos empleados, editar su información, o eliminarlos, así mismo para las pantallas de cargo y departamentos con sus respectivas opciones para editar o eliminar los registros. El frontend quedaría estructurada de la siguiente manera.

Figura 13: Se presentan las tres vistas creadas, Pantalla inicio, empleados, cargos, departamentos.

UNIVERSIDAD NACIONAL DE LOJA

MAESTRIA EN INGENIERIA EN SOFTWARE







Sitio Administrativo de Vivienda y
Financiamiento VIVENTO
Administración de Empleados.

INTEGRANTES:

- Janneth Guamán
- Lenin Quizhpe

a) Interfaz de inicio

[Agregar nuevo empleado](#)

Identificación	Apellidos	Nombres	Cargo	Imagen	Acciones
0604704395	Oscar Estalin	Vizuete Tactaquiz	Ing. Sistemas		Editar Borrar
0605823656	Jose Fernando	Flores Guamán	Ing. Civil		Editar Borrar
0604068593	Lenin Israel	Quizhpe Narvaiz	Ing. Electrico		Editar Borrar
1700253697	Janneth Patricia	Guamán Siguenza	Secretaria		Editar Borrar
1896320254	Anita Elizabeth	Ayala Vizuete	Contadora		Editar Borrar
0502369852	Gissela Elizabeth	Mendez Guerrero	Ing. Sistemas		Editar Borrar

b) Interfaz de presentación de lista de empleados

[Agregar nuevo cargo](#)

Codigo	Nombre	Actividad	Acciones
001	Jefe Financiero	Área Financiera	Editar Borrar
002	Jefe de TIC's	Coordinador de infraestructura de tecnologías	Editar Borrar

c) Interfaz de lista de cargos

Agregar nuevo departamento

Departamentos			
Código	Nombre	Actividad	Acciones
001	Gerencia	Trámites administrativa	Editar Borrar
002	Atención al Cliente	Atender a los clientes / usuarios Vivo	Editar Borrar
003	Administrativo	Administración de información de vivo	Editar Borrar
004	TIC	Área tecnología de vivo	Editar Borrar

d) Interfaz de lista de departamentos.

Cada una de las vistas de empleado, cargo, y departamento cuentan con la opción para agregar un nuevo registro como se presenta en el Figura 14.

Crear un empleado

Datos del empleado

Identificación:

Nombres:

Apellidos:

Cargo:

Departamento:

Imagen:

Examinar... No se ha seleccionado ningún archivo.

[Guardar Información](#) [Cancelar](#)

a) Interfaz para creación de un nuevo empleado

Crear cargo

Datos del cargo

Código:

Nombre:

Actividad:

[Guardar Información](#) [Cancelar](#)

b) Interfaz para creación de un nuevo cargo

Crear un departamento

Datos del departamento

Código:

Nombre:

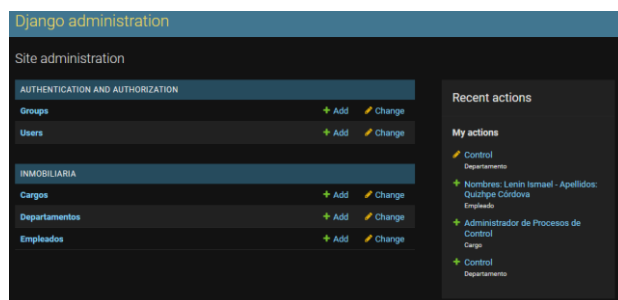
Actividad:

[Guardar Información](#) [Cancelar](#)

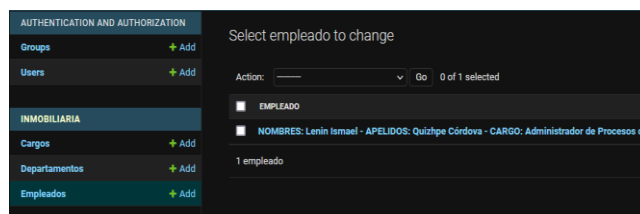
c) Interfaz para creación de un nuevo departamento

Las vistas desarrolladas fueron creadas sin tomar en cuenta el administrados de Django, aunque si la programación e interacción entre los diferentes archivos llevarían a la creación de todas las interfaces desarrolladas anteriormente, desde el administrador de Django, como se presenta en la Figura 15.

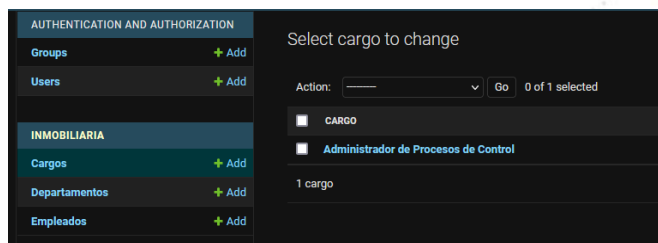
Figura 15: Interfaz de la página web vista desde el administrador de django



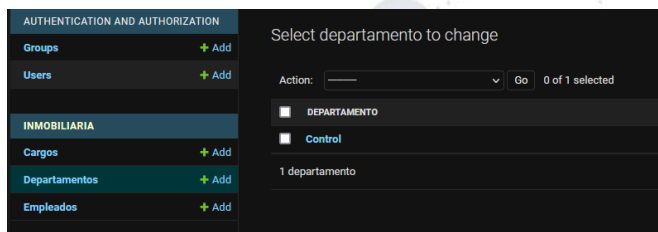
a) Interfaz de inicio



b) Interfaz de presentación de lista de empleados



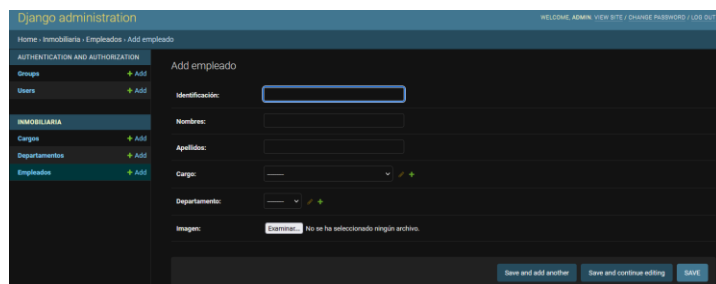
c) Interfaz de lista de cargos



d) Interfaz de lista de departamentos.

De la misma manera se presenta la interfaz para agregar nuevos empleados, cargos y departamentos, como se observa en la Figura 16.

Figura 16: Interfaz para registrar nuevo empleado, visto desde el administrador de Django.



The screenshot shows the Django administration interface for the 'Inmobiliaria' app. The left sidebar contains a menu with 'AUTHENTICATION AND AUTHORIZATION' (Groups, Users) and 'INMOBILIARIA' (Cargos, Departamentos, Empleados). The main content area is titled 'Add empleado' and contains a form with the following fields: 'Identificación' (text input), 'Nombres' (text input), 'Apellidos' (text input), 'Cargo' (dropdown menu), 'Departamento' (dropdown menu), and 'Imagen' (file upload field with a 'Seleccionar' button). At the bottom right, there are three buttons: 'Save and add another', 'Save and continue editing', and 'Save'.

Conclusiones

- La programación de vistas sin el uso de las aplicaciones de django, harían que la programación de la aplicación web, sería más complicada ya que se debería crear los botones de acceso para la eliminar, editar y vincularlos con cada vista, mientras que desde el administrador de django, su aplicación es más rápida.
- Establecer correctamente los modelos de las clases en el archivo models.py para evitar errores de relaciones en la generación de las tablas en la base de datos.
- Al manejar formularios personalizados en Django, verificar que los elementos widgets asignados para cada atributo sea el correcto para el ingreso de la información.