DESARROLLO WEB FULLSTACK

con Python y JavaScript

polotic misiones

Clase 11: Anexo 1

CRUD con Django

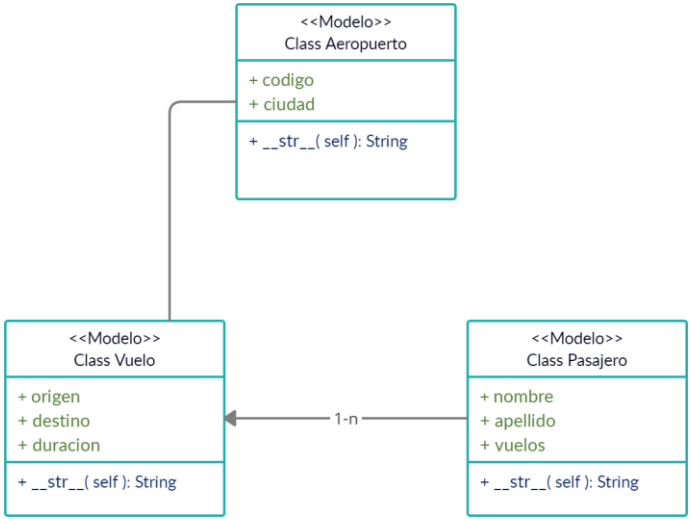
CRUD refiere a las operaciones que podemos hacer sobre los datos.

- **CREATE**: Crear los registros que se van a guardar en la base de datos.
- **READ/RETRIEVE**: Leer los registros que ya están almacenados.
- **UPDATE**: Actualizar los registros realizando una edición total o parcial de los datos.
- **DELETE/DESTROY**: Eliminar los datos guardados.

Toda aplicación debe disponer de una manera de realizar estas operaciones. El administrador de Django (Django Admin) lo hace (como ya lo hemos visto) sin que tengamos que programar. Sin embargo, puede ocurrir que querramos hacerlo personalizado.

RELACIONES

A manera de ejemplo trabajaremos sobre una aplicación ya realizada (la de las Aerolíneas). Las relaciones se definen de la siguiente manera:



FORMULARIOS DESDE MODELOS

Necesitamos especificar un formulario basado en el Modelo que queramos implementar el CRUD, ya que lo expondremos en nuestros templates y luego procesaremos en nuestras vistas. En este caso el formulario quedaría así:

```
class FormVueloCustom(forms.ModelForm):
    #campos del modelo
    class Meta:
        model = Vuelo
        fields = ('origen', 'destino', 'duracion')
```

Como en este caso de ejemplo es solo un formulario podemos ponerlo al final de **`views.py`** pero también es bueno agregarlo en un nuevo archivo denominado **`forms.py`** y luego importarlo dentro de **`views.py`**

CREATE - URL

Nos enfocaremos aquí en Vuelos.

Creamos la redireccion URL que llevará al ALTA de nuestro Vuelo:

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

app_name = "AEROLINEA"
urlpatterns = [
    path('', views.index, name="index"),
    path('<int:vuelo_id>', views.vuelo, name="vuelo"),
    path('vuelo_alta', views.vuelo_alta, name="vuelo_alta"),
    path('<int:vuelo_id>/vuelo_modificar', views.vuelo_modificar, name="vuelo_modificar"),
    path('<int:vuelo_id>/vuelo_eliminar', views.vuelo_eliminar, name="vuelo_eliminar"),
    path('<int:vuelo_id>/reserva', views.reserva, name="reserva")
]
```

CREATE - VIEW

Creamos la vista que se encargará de recibir los datos del Formulario por medio del POST y guardarlos a la base con form.save().

```
def vuelo_alta(request):
    if request.method == "POST":
        form = FormVueloCustom(request.POST)
        if form.is_valid():
            form.save()
    else:
        form = FormVueloCustom()
        return render(request, "vuelos/vuelo_alta.html", {
            "formset": form
        })
```

CREATE - TEMPLATE

Creamos el template que mostrará el formulario de alta del Vuelo.

```
{% extends "vuelos/layout.html" %}
{% block body %}
<h3>Alta de Vuelo</h3>
<form action="{% url 'AEROLINEA:vuelo_alta' %}" method="post">
    {% csrf_token %}
    {{ form }}
    <input type="submit" value="GUARDAR">
</form>
{% endblock %}
```

READ - URL

Creamos la redireccion URL que llevará a la Vista de nuestro Vuelo:

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

app_name = "AEROLINEA"
urlpatterns = [
    path('', views.index, name="index"),
    path('<int:vuelo_id>', views.vuelo, name="vuelo"),
    path('vuelo_alta', views.vuelo_alta, name="vuelo_alta"),
    path('<int:vuelo_id>/vuelo_modificar', views.vuelo_modificar, name="vuelo_modificar"),
    path('<int:vuelo_id>/vuelo_eliminar', views.vuelo_eliminar, name="vuelo_eliminar"),
    path('<int:vuelo_id>/reserva', views.reserva, name="reserva")
]
```

READ - VIEW

Creamos la vista que se encargará de traer el modelo en particular de la BD con su ID y lo enviará por el contexto a la vista vuelo.html

```
def vuelo(request, vuelo_id):
    vuelo = Vuelo.objects.get(id=vuelo_id)
    pasajeros = vuelo.pasajeros.all()
    no_son_pasajeros = Pasajero.objects.exclude(vuelos=vuelo).all()
    return render(request, "vuelos/vuelo.html", {
        "vuelo": vuelo,
        "pasajeros": pasajeros,
        "no_son_pasajeros": no_son_pasajeros
    })
```

READ - TEMPLATE

Creamos el template que mostrará la info del Vuelo. En éste caso los pasajeros que tiene.

```
{% extends "vuelos/layout.html" %}
{% block body %}
<h2>Vuelo</h2>
<h3>Lista de Pasajeros</h3>
<ul>
    {% for un_pasajero in pasajeros %}
        <li>{{ un_pasajero }}</li>
    {% empty %}
        <li>No hay pasajeros registrados.</li>
    {% endfor %}
</ul>
<a href="{% url 'AEROLINEA:index' %}">Volver a la lista</a>
<h3>Reserva</h3>
<form action="{% url 'AEROLINEA:reserva' vuelo.id %}" method="post">
    {% csrf_token %}
    <select name="pasajero" id="">
        {% for un_pasajero in no_son_pasajeros %}
            <option value="{{ un_pasajero.id }}">{{ un_pasajero }}</option>
        {% endfor %}
    </select>
    <input type="submit" value="RESERVAR">
</form>
{% endblock %}
```

UPDATE - URL

Creamos la redireccion URL que llevará a la vista de edición de un Vuelo existente mediante su ID:

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

app_name = "AEROLINEA"
urlpatterns = [
    path('', views.index, name="index"),
    path('<int:vuelo_id>', views.vuelo, name="vuelo"),
    path('vuelo_alta', views.vuelo_alta, name="vuelo_alta"),
    path('<int:vuelo_id>/vuelo_modificar', views.vuelo_modificar, name="vuelo_modificar"),
    path('<int:vuelo_id>/vuelo_eliminar', views.vuelo_eliminar, name="vuelo_eliminar"),
    path('<int:vuelo_id>/reserva', views.reserva, name="reserva")
]
```

UPDATE - VIEW

Creamos la vista que se encargará de traer el modelo en particular de la BD con su ID, si la solicitud es GET (el usuario solicito acceder al formulario) se trae el Form con los datos completos para editarlos. Si es un POST lo que hace es guardar la info del formulario en la BD.

```
def vuelo_modificar(request, vuelo_id):
    un_vuelo = get_object_or_404(Vuelo, id=vuelo_id)

    if request.method == "POST":
        form = FormVueloCustom(request.POST, instance = un_vuelo)
        if form.is_valid():
            form.save()
            return render(request,"vuelos/index.html", {
                "lista_vuelos": Vuelo.objects.all()
            })
    else:
        form = FormVueloCustom(instance = un_vuelo)
        return render(request, 'vuelos/vuelo_modificar.html', {
            "un_vuelo": un_vuelo,
            "form": form
        })
```

UPDATE - TEMPLATE

Tenemos que construir la pagina HTML del formulario de Update.

```
{% extends "vuelos/layout.html" %}
{% block body %}
<h3>Modificar Vuelo</h3>
<form action="{% url 'AEROLINEA:vuelo_modificar' un_vuelo.id %}" method="post">
    {% csrf_token %}
    {{ form }}
    <input type="submit" value="GUARDAR">
</form>
{% endblock %}
```

DELETE - URL

Creamos la redireccion URL que llevará a la vista que se encargará de eliminar un Vuelo existente mediante su ID:

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

app_name = "AEROLINEA"
urlpatterns = [
    path('', views.index, name="index"),
    path('<int:vuelo_id>', views.vuelo, name="vuelo"),
    path('vuelo_alta', views.vuelo_alta, name="vuelo_alta"),
    path('<int:vuelo_id>/vuelo_modificar', views.vuelo_modificar, name="vuelo_modificar"),
    path('<int:vuelo_id>/vuelo_eliminar', views.vuelo_eliminar, name="vuelo_eliminar"),
    path('<int:vuelo_id>/reserva', views.reserva, name="reserva")
]
```

DELETE - VIEW

La vista vuelo_eliminar obtiene el vuelo de la base de datos y luego lo elimina con la propiedad delete().

```
def vuelo_eliminar(request, vuelo_id):
    un_vuelo = get_object_or_404(Vuelo, id=vuelo_id)
    un_vuelo.delete()
    return render(request,"vuelos/index.html", {
        "lista_vuelos": Vuelo.objects.all()
    })
```