Informe de Laboratorio 05

Tema: Django

Nota

Estudiante	Escuela	Asignatura
Jhastyn Jefferson Payehuanca Riquelme jpayehuancar@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Programación Web 2 Semestre: III Código: 20222064

Laboratorio	Tema	Duración
05	Django	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - A	Del 08 Junio 2023	Al 15 Junio 2023

1. Tarea

- Crea un blog sencillo en un entorno virtual utilizando la guía: https://tutorial.djangogirls.org/es/django-start-project/
- Especificar paso a paso la creación del blog en su informe.
- Crear un video tutorial donde realice las operaciones CRUD (URL public reproducible online)
- Adjuntar URL del video en el informe.

2. URL GitHub/Video

- URL del repositorio GitHub.
- `https://github.com/Jh4stYn/my-first-blog.git`
- URL del Video.
- `https://flip.com/s/iKm2W1YvNp7z`

3. Creación del Blog

3.1. Commits

Listing 1: Crear una aplicación

```
$ django-admin startproject mysite .
```

Listing 2: Models.py

```
from django.conf import settings
from django.db import models
from django.utils import timezone


class Post(models.Model):
    author = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE)
    title = models.CharField(max_length=200)
    text = models.TextField()
    created_date = models.DateTimeField(
            default=timezone.now)
    published_date = models.DateTimeField(
            blank=True, null=True)

    def publish(self):
        self.published_date = timezone.now()
        self.save()

    def __str__(self):
        return self.title
```

Listing 3: Agregar nuestro nuevo modelo a la base de datos

```
$ python manage.py makemigrations blog
$ python manage.py migrate blog
```

Listing 4: Admin.py

```
from django.contrib import admin
from .models import Post

admin.site.register(Post)
```

Listing 5: Crear un superusuario (superuser)

```
$ python manage.py createsuperuser
  Username: Jhastyn
  Email address: jpayehuancar@unsa.edu.pe
  Password: *********
  Password (again): *********
  Superuser created successfully.
  ----------------------------------------
```

Listing 6: Urls.py

```
"""
URL configuration for myblog project.

The ‘urlpatterns‘ list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/4.2/topics/http/urls/
Examples:
Function views
    1. Add an import: from my_app import views
    2. Add a URL to urlpatterns: path(’’, views.home, name=’home’)
Class-based views
    1. Add an import: from other_app.views import Home
    2. Add a URL to urlpatterns: path(’’, Home.as_view(), name=’home’)
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns: path(’blog/’, include(’blog.urls’))
"""
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path(’admin/’, admin.site.urls),
    path(’’, include(’blog.urls’)),
]
```

Listing 7: Crear urls.py en el directorio blog

```
    $ vim Lab05/blog/urls.py
      from django.urls import path
      from . import views
      urlpatterns = [
          path(’’, views.post_list, name=’post_list’),
      ]
```

Listing 8: Agregaremos nuestras views al archivo

```
    $ vim Lab05/blog/views.py
    from django.shortcuts import
    def post_list(request):
    return render(request, ’blog/post_list.html’, {})
```

Listing 9: Personalizamos plantilla

```
        $mkdir templates/blog
        $vim post_list.html
          <html>
              <head>
                  <title>Django Girls blog</title>
              </head>
              <body>
                  <div>
                      <h1><a href="/">Django Girls Blog</a></h1>
                  </div>
```

```
                <div>
                    <p>published: 14.06.2014, 12:14</p>
                    <h2><a href="">My first post</a></h2>
                    <p>Aenean eu leo quam. Pellentesque ornare sem lacinia quam venenatis
                        vestibulum. Donec id elit non mi porta gravida at eget metus. Fusce
                        dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh,
                        ut fermentum massa justo sit amet risus.</p>
                </div>

                <div>
                    <p>published: 14.06.2014, 12:14</p>
                    <h2><a href="">My second post</a></h2>
                    <p>Aenean eu leo quam. Pellentesque ornare sem lacinia quam venenatis
                        vestibulum. Donec id elit non mi porta gravida at eget metus. Fusce
                        dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh,
                        ut f.</p>
                </div>
            </body>
        </html>
```

Listing 10: Abrimos views.py

```
$ vim views.py
from django.shortcuts import render
from django.utils import timezone
from .models import Post

def post_list(request):
    posts = Pst.objects.filter(published_date__lte=timezone.now()). ...
                                                ... order_by('published_date')
return render(request, 'blog/post_list.html', {'posts': posts})
```

Listing 11: Abrimos post-list.html

```
<div>
    <h1><a href="/">Django Girls Blog</a></h1>
</div>

{% for post in posts %}
    <div>
        <p> publicado: {{ post.published_date }}</p>
        <h2><a href="">{{ post.title }}</a></h2>
        <p>{{ post.text|linebreaksbr }}</p>
    </div>
{% endfor %}
```

Listing 12: Blog.css

```
1  .page-header {
2      background-color: #C25100;
3      margin-top: 0;
4      padding: 20px 20px 20px 40px;
5  }
6
7  .page-header h1, .page-header h1 a, .page-header h1 a:visited, .page-header h1 a:active {
```

```
    color: #ffffff;
    font-size: 36pt;
    text-decoration: none;
}

.content {
    margin-left: 40px;
}

h1, h2, h3, h4 {
    font-family: 'Lobster', cursive;
}

.date {
    color: #828282;
}

.save {
    float: right;
}

.post-form textarea, .post-form input {
    width: 100%;
}

.top-menu, .top-menu:hover, .top-menu:visited {
    color: #ffffff;
    float: right;
    font-size: 26pt;
    margin-right: 20px;
}

.post {
    margin-bottom: 70px;
}

.post h2 a, .post h2 a:visited {
    color: #000000;
}
```

Listing 13: Base.html

```
{% load static %}
<html>
    <head>
        <title>Django Girls blog</title>
        <link rel="stylesheet"
            href="//maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css">
        <link rel="stylesheet"
            href="//maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap-theme.min.css">
        <link href='//fonts.googleapis.com/css?family=Lobster&subset=latin,latin-ext'
            rel='stylesheet' type='text/css'>
        <link rel="stylesheet" href="{% static 'css/blog.css' %}">
    </head>
    <body>
        <div class="page-header">
```

```
            {% if user.is_authenticated %}
                <a href="{% url 'post_new' %}" class="top-menu"><span class="glyphicon
                    glyphicon-plus"></span></a>
            {% endif %}
            <h1><a href="/">Django Girls Blog</a></h1>
        </div>
        <div class="content container">
            <div class="row">
                <div class="col-md-8">
                    {% block content %}
                    {% endblock %}
                </div>
            </div>
        </div>
    </body>
</html>
```

Listing 14: Base.html

```
{% load static %}
<html>
    <head>
        <title>Django Girls blog</title>
        <link rel="stylesheet"
            href="//maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css">
        <link rel="stylesheet"
            href="//maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap-theme.min.css">
        <link href='//fonts.googleapis.com/css?family=Lobster&subset=latin,latin-ext'
            rel='stylesheet' type='text/css'>
        <link rel="stylesheet" href="{% static 'css/blog.css' %}">
    </head>
    <body>
        <div class="page-header">
            {% if user.is_authenticated %}
                <a href="{% url 'post_new' %}" class="top-menu"><span class="glyphicon
                    glyphicon-plus"></span></a>
            {% endif %}
            <h1><a href="/">Django Girls Blog</a></h1>
        </div>
        <div class="content container">
            <div class="row">
                <div class="col-md-8">
                    {% block content %}
                    {% endblock %}
                </div>
            </div>
        </div>
    </body>
</html>
```

Listing 15: Post-list.html

```
{% extends 'blog/base.html' %}

{% block content %}
```

```
    {% for post in posts %}
        <div class="post">
            <div class="date">
                {{ post.published_date }}
            </div>
            <h2><a href="{% url 'post_detail' pk=post.pk %}">{{ post.title }}</a></h2>
            <p>{{ post.text|linebreaksbr }}</p>
        </div>
    {% endfor %}
{% endblock %}
```

Listing 16: Creamos forms.py

```
            $ vim forms.py
            from django import forms
            from .models import Post
            class PostForm(forms.ModelForm):

            class Meta:
                model = Post
                fields = ('title', 'text',)
                def post_list(request):
```

Listing 17: Creamos urls.py

```
            $ vim urls.py
            from django.urls import path
            from . import views

            urlpatterns = [
                path('', views.post_list, name='post_list'),
                path('post/<int:pk>/', views.post_detail, name='post_detail'),
                path('post/new/', views.post_new, name='post_new'),
                path('post/<int:pk>/edit/', views.post_edit, name='post_edit'),
            ]
```

Listing 18: Post-detail.html

```
{% extends 'blog/base.html' %}

{% block content %}
    <div class="post">
        {% if post.published_date %}
            <div class="date">
                {{ post.published_date }}
            </div>
        {% endif %}
        <a class="btn btn-default" href="{% url 'post_edit' pk=post.pk %}"><span
            class="glyphicon glyphicon-pencil"></span></a>
        <h2>{{ post.title }}</h2>
        <p>{{ post.text|linebreaksbr }}</p>
    </div>
{% endblock %}
```

Listing 19: Post-edit.html

```
{% extends 'blog/base.html' %}

{% block content %}
    <h2>New post</h2>
    <form method="POST" class="post-form">{% csrf_token %}
        {{ form.as_p }}
        <button type="submit" class="save btn btn-default">Save</button>
    </form>
{% endblock %}
```

Listing 20: Views.py

```
from django.shortcuts import render, get_object_or_404
from django.utils import timezone
from django.shortcuts import redirect
from .models import Post
from .forms import PostForm

def post_list(request):
    posts = Post.objects.filter(published_date__lte=timezone.now()).order_by('published_date')
    return render(request, 'blog/post_list.html', {'posts': posts})

def post_detail(request, pk):
    post = get_object_or_404(Post, pk=pk)
    return render(request, 'blog/post_detail.html', {'post': post})

def post_new(request):
    if request.method == "POST":
        form = PostForm(request.POST)
        if form.is_valid():
            post = form.save(commit=False)
            post.author = request.user
            post.published_date = timezone.now()
            post.save()
            return redirect('post_detail', pk=post.pk)
    else:
        form = PostForm()
    return render(request, 'blog/post_edit.html', {'form': form})

def post_edit(request, pk):
    post = get_object_or_404(Post, pk=pk)
    if request.method == "POST":
        form = PostForm(request.POST, instance=post)
        if form.is_valid():
            post = form.save(commit=False)
            post.author = request.user
            post.published_date = timezone.now()
            post.save()
            return redirect('post_detail', pk=post.pk)
    else:
        form = PostForm(instance=post)
    return render(request, 'blog/post_edit.html', {'form': form})
```

3.2. Estructura de laboratorio 05

- El contenido que se entrega en este laboratorio es el siguiente:

```
lab05/
|--- .gitignore
|--- manage.py
|--- latex
|    |--- img
|    |   |--- logo_abet.png
|    |   |--- logo_episunsa.png
|    |--- informe_lab05_Jh4stYn.pdf
|    |--- informe_lab05_Jh4stYn.tex
|    |--- src
|       |--- ListaPoo01.java
|       |--- Node01.java
|--- blog
|    |--- migrations
|    |   |--- 0001_initial.py
|    |   |--- __init__.py
|    |--- static/css
|    |   |--- blog.css
|    |--- templates/blog
|    |   |--- base.html
|    |   |--- post_detail.html
|    |   |--- post_edit.html
|    |   |--- post_list.html
|    |--- _init_.py
|    |--- admin.py
|    |--- apps.py
|    |--- forms.py
|    |--- models.py
|    |--- tests.py
|    |--- urls.py
|    |--- views.py
|--- myblog
|    |--- __init__.py
|    |--- asgi.py
|    |--- settings.py
|    |--- urls.py
|    |--- wsgi.py
|--- myenv
|    |--- . . .
     .
     .
     .
```

4. Cuestionario

- ¿Cuál es un estándar de codificación para Python?
  Un estándar de codificación ampliamente utilizado en la comunidad de Python es el PEP 8 (Python Enhancement Proposal 8). El PEP 8 establece recomendaciones sobre la forma en que el código Python debe estructurarse, nombrarse y formatearse para mejorar su legibilidad y consistencia.

- ¿Qué diferencias existen entre EasyInstall, pip, y PyPM?
  -EasyInstall: Fue una herramienta utilizada anteriormente para instalar paquetes de Python y sus dependencias. Sin embargo, actualmente se recomienda utilizar pip en lugar de easy-install. pip proporciona más características y es ampliamente utilizado en la comunidad de Python.

  -pip: Es la herramienta de administración de paquetes estándar de Python. Se utiliza para instalar, actualizar y administrar paquetes y dependencias de Python de manera sencilla. Pip es ampliamente utilizado y cuenta con una amplia variedad de paquetes disponibles en el Python Package Index (PyPI).

  -PyPM: Es el administrador de paquetes utilizado por ActivePython, una distribución específica de Python. PyPM se utiliza para instalar paquetes y gestionar dependencias en el entorno de ActivePython. Sin embargo, ten en cuenta que PyPM no es tan ampliamente utilizado como pip, que es el administrador de paquetes estándar de Python.

- En un proyecto Django que se debe ignorar para usar git.¿Qué otros tipos de archivos se deberían agregar a este archivo?
  En un proyecto Django, el archivo .gitignore proporciona una lista de archivos y directorios que se deben ignorar al realizar seguimiento con Git. Algunos ejemplos de archivos y directorios comunes que se suelen agregar a .gitignore en un proyecto Django son:

  *.pyc: Archivos de bytecode de Python generados por el intérprete.
  /venv/: Directorio del entorno virtual (si se utiliza).
  /static/: Directorio de archivos estáticos generados por Django.
  /media/: Directorio de archivos multimedia cargados por los usuarios.
  /db.sqlite3: Base de datos SQLite predeterminada de Django.
  /–pycache–/: Directorio de caché de Python generado por el intérprete
  /logs/: Directorio de registros generados por la aplicación.

- Utilice python manage.py shell para agregar objetos. ¿Qué archivos se modificaron al agregar más objetos?
  Al utilizar python manage.py shell para agregar objetos en un proyecto Django, los archivos que se pueden modificar dependen de la forma en que se implemente la lógica de creación de objetos. Si la lógica de creación de objetos se encuentra en un archivo de vista o en un archivo de migración, esos archivos se verán modificados. Además, si se crea un objeto en una base de datos, los archivos de migración relacionados con la base de datos también pueden ser modificados para reflejar los cambios realizados.