PROYECTO FINAL DE BASE DE DATOS E IMPLANTACIÓN DE APLICACIONES

Nombre: Cristian Cobo Garrido

Introducción

El proyecto que voy a abordar tratará de la creación de una aplicación web cuya función será el posteo y la visualización de publicaciones por partes de usuarios y editoriales, así como la administración de tanto de las publicaciones como de los usuarios. Para ello, la aplicación web contará con una serie de funcionalidades orientadas a mejorar la experiencia de usuario a la hora de realizar las acciones anteriormente mencionadas.

En este proyecto, se hace uso de lenguajes tales como CSS, HTML, JS, SQL y Python. Todos estos lenguajes, han sido aplicados sobre un framework llamado Flask, el cual ha permitido construir la aplicación proporcionando una estructura.

La aplicación, será ejecutada en un entorno virtual. Además, se usará el puerto por defecto de Flask (5000) a la hora de acceder a la aplicación.

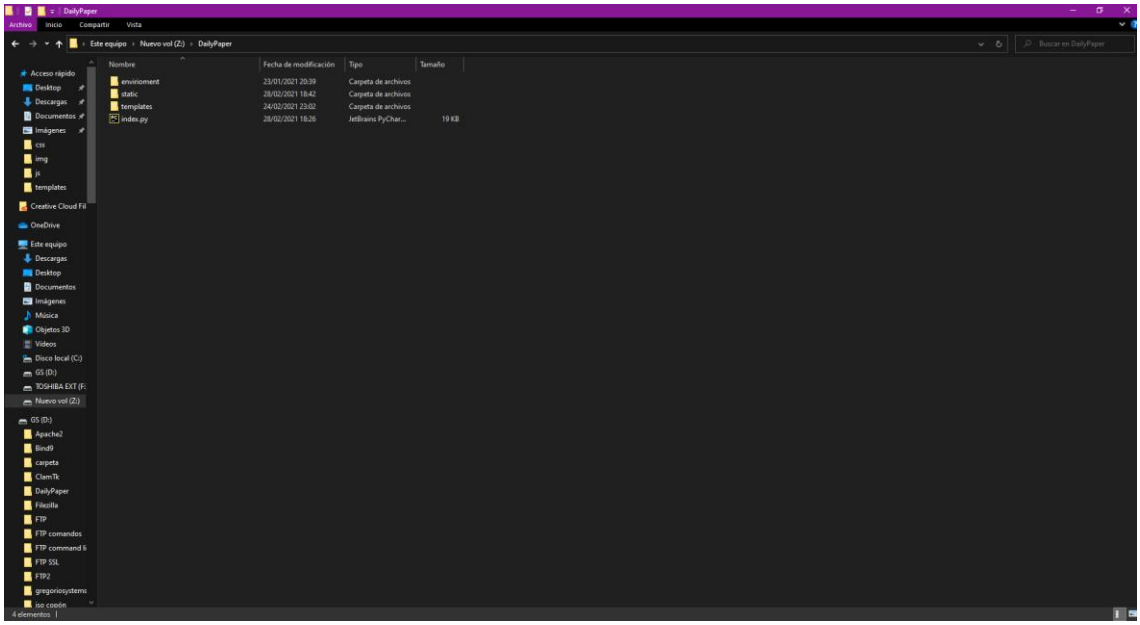
Tipos de usuarios

Esta aplicación contará con tres tipos de usuarios:

- Usuarios y editoriales: Capaces de ver, buscar y comentar posts, así como la posibilidad de editarse ciertos datos relativos a su perfil. Además, pueden buscar perfiles de otros usuarios.
- Administradores: Capaces de ver y buscar usuarios y publicaciones, así como la posibilidad de borrar a los mismos. También tienen la posibilidad de postear. Las únicas acciones que no pueden hacer son las de editarse y las de borrar a otros administradores.

Estructura

Este proyecto, contará con una estructura de archivos y directorios, la cual nos va permitir definir la estructura de la aplicación.



Como se puede observar, el proyecto contendrá los siguientes directorios y archivos:

Enviroment: Directorio donde se aloja y se ejecuta todo el entorno virtual.

Static: Directorio donde se alojan los archivos tales como JS, CSS, imágenes, … Cada tipo de archivo, suele ir separado por un subdirectorio.

Templates: Directorios donde se almacenan todas las plantillas HTML. En ese directorio se encontrará tanto la plantilla madre (layout.html), como las plantillas hijas.

Index.py: Es el archivo de ejecución de la aplicación. Contendrá todo lo necesario para arrancar la aplicación y para navegar entre rutas mostrando un resultado.

Arrancar la aplicación

Para poner en marcha la aplicación, será necesario realizar los siguientes pasos:

1. Abrir una consola de comandos y navegar hasta donde se encuentre la carpeta del entorno virtual.
   cd DailyPaper\envirioment
2. Una vez allí, se accederá a la carpeta “Scripts”.
3. Después, se ejecutará “activate” poniendo el nombre del archivo en la consola.
   activate
4. Una vez hecho esto, se ejecutará la aplicación volviendo a la raíz del proyecto, y poniendo lo siguiente:

```
(envirioment) Z:\DailyPaper>python index.py
 * Serving Flask app "index" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 275-248-562
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

5. Para acceder a la aplicación, pondremos la IP y el puerto que nos arroja la aplicación a la hora de ejecutarse.

Módulos utilizados

A la hora de utilizar Python, me he servido de una serie de módulos los cuales me han permitido obtener la funcionalidad que necesitaba en mi proyecto. Estos módulos se dividen en módulos de Flask y módulo datetime.

Módulos de Flask

Los cuales otorgan la funcionalidad del framework. Entre estos módulos se encuentran:

- Flask: El cual permitirá instanciar la aplicación para poder realizar ciertas acciones, como por ej. configurar la aplicación, arrancar la aplicación, … Esta instanciación se llevará a cabo mediante una variable llamada “app”.
- Render_template: Permitirá devolver una plantilla HTML mediante return, a la hora de ejecutar una ruta.
- Request: Es modulo que permite poseer toda la información acerca de la petición HTTP. En este proyecto, se utilizado tanto para comprobar el método que está utilizando una ruta (get o post) como para extraer la información recibida en el cuerpo de la petición cuando se utiliza el método POST (normalmente se utiliza un formulario HTML para enviar esta información).
- Url_for: Utilizado para redirigir hacia otras rutas (referenciadas mediante el nombre de la función) o para referenciar otros archivos.
- Redirect: Utilizado para redirigir hacia otras rutas.
- MySQL: Utilizado para realizar conexiones con la base de datos, y acciones con la BD mediante cursores.

Módulo datetime

Este módulo es utilizado para realizar ciertas operaciones con el tiempo.

En este proyecto, será utilizado para extraer el año, el mes y el día actual.

Conexión con la base de datos

Para establecer conexión entre la BD y la aplicación, se ha hecho uso de un módulo, el cual permite realizar la conexión y realizar ciertas acciones con la BD mediante cursores.

Para establecer la conexión con la BD, se han especificado parámetros tales como la dirección del servidor, el usuario, la contraseña y el nombre de la BD.

```
app.config['MYSQL_HOST'] = 'localhost' #Direccion del servidor de BD
app.config['MYSQL_USER'] = 'root' #Usuario del servidor
app.config['MYSQL_PASSWORD'] = '1234' #Contraseña del servidor
app.config['MYSQL_DB'] = 'DailyPaper' #Nombre de la BD
mysql = MySQL(app)
```

Rutas

Esta aplicación, contará con una serie de rutas las cuales servirá para acceder a todas las funcionalidades de la misma y a la información contenida en la BD. En total, la aplicación cuenta con 23 rutas.

Ruta raíz

Es la primera ruta que se ejecutará al acceder a la aplicación.

Devuelve la plantilla donde se registrarán y logearán los usuarios.

```
@app.route('/')
def home():
    return render_template('login.html')
```

Ruta de registro

Esta ruta, será ejecutada cuando un usuario se registre en la BD de la aplicación.

El usuario, es redirigido a esta ruta mediante un formulario.

La ruta, comprobará si los datos del formulario anteriormente mencionado han sido han enviados o no. Si los datos han sido enviados, comprobará que todos los datos se han introducido de acuerdo con la normalidad mediante comprobaciones if. Si los datos no se han introducido de acuerdo con la normalidad, entonces saltarán una serie de errores.

```
@app.route('/register', methods=['POST', 'GET'])
```

```
def register():
    if request.method == 'POST': #Si el método por el cual se accede a la
 ruta es POST, quiere decir que el formulario ha sido enviado
        #Extrar los datos del formulario mediante el método form y lo alm
acena en variables
        user = request.form['user']
        passwd = request.form['pass']
        confirm_passwd = request.form['confirmpass']
        type_user = request.form['typeuser']
        cursor = mysql.connection.cursor()
        usuario = cursor.execute(f'select * from usuarios where nombre =
"{user}"')
        mysql.connection.commit()
        if passwd != confirm_passwd: #Comprueba si los campos de contrase
ñas son coincidentes o no
            mensaje = "Usuario no registrado! La contraseña no coincide"
            return render_template ('login.html', mensaje=mensaje)
        elif usuario == True: #Comprueba si el usuario existe o no confor
ma a lo que haya arrojado la consulta anterior (si arroja True, el usuari
o existe). En caso de que el usuario exista, arrojará un error
            mensaje = "El usuario ya existe!"
            return render_template ('login.html', mensaje=mensaje)
        elif user == "" or passwd == "": #Comprueba si los datos necesari
os tales como el usuario y la contraseña están en blanco o no
            mensaje = "Error debe introducir todos los datos"
            return render_template ('login.html', mensaje=mensaje)
        elif passwd == confirm_passwd: #En caso de que ningunas de las an
teriores acciones no se haya llevado a cabo, y los campos de contraseña s
ean coincidentes, se llevará a cabo la insercción del usuario en la BD me
diante un cursor
            cursor1 = mysql.connection.cursor()
            add_user = cursor1.execute('insert into usuarios(nombre, pass
, tipo_usuario) values (%s, %s, %s)', (user, passwd, type_user))
            data1 = cursor1.fetchall()
            mysql.connection.commit()
            return render_template ('login.html')
    else: #Si el formulario no ha sido enviado, el método seguirá siendo
GET
        return render_template ('login.html')
```

Ruta login

Esta ruta, será accedida mediante el formulario de logeo, y se encargará de comprobar si el usuario existe o no en la BD. Todo esto se lleva a cabo, con el fin de poner a disposición funcionalidades diferentes, dependiendo del usuario que se trate.

```
@app.route('/login', methods=['POST', 'GET'])
def login():
```

```
    if request.method == 'POST': #En caso de que el formulario se haya en
viado
        global user
        user = request.form['user']
        passwd = request.form['pass']
        cursor = mysql.connection.cursor()
        cursor1 = mysql.connection.cursor()
        usuario = cursor.execute('select nombre from usuarios where nombr
e = %s and pass = %s', (user, passwd))
        editorial = cursor1.execute('select tipo_usuario from usuarios wh
ere nombre = %s and pass = %s', (user, passwd))
        data = cursor1.fetchall()
        mysql.connection.commit()
        #Comprobaciones de la existencias de los usuarios mediante un cur
sor y del tipo de usuario mediante otro cursor
        #Si el usuario es usuario raso o editorial devolverá un index dif
erente al index de un admin
        if usuario == True and data[0][0] == "Editorial":
            return redirect(url_for("inicio"))
        elif usuario == True and data[0][0] == "Usuario":
            return redirect(url_for("inicio"))
        elif usuario == True and data[0][0] == "Admin":
            return redirect(url_for("inicio_admin"))
        elif user == "" or passwd == "": #Si los campos están vacios, dev
olverá error
            mensaje = "Error debe introducir todos los datos"
            return render_template('login.html', mensaje=mensaje)
        else: #Si el usuario no existe o su contraseña esta mal introduci
da, devolverá error
            mensaje = "Error de autenticación"
            return render_template('login.html', mensaje=mensaje)
    else:
        return render_template ('login.html')
```

Ruta de inicio de los usuarios y las editoriales

Esta ruta, será devuelta cuando en el login, la aplicación haya detectado que el usuario que se trata de logear, es un usuario normal o una editorial.

Esta ruta, devolverá la plantilla correspondiente donde se encontrarán los elementos para desempeñar las funciones de esos usuarios, así como las publicaciones que hayan realizado esos usuarios.

```
@app.route('/inicio')
def inicio():
    cursor = mysql.connection.cursor()
    cursor2 = mysql.connection.cursor()
    #Se extrae las noticias posteadas por el usuario, para que luego sean
 mostradas en la plantilla con un for
```

```
    noticias = cursor.execute(f'select p.id, p.titular from usuarios u in
ner join publicaciones p on u.id = p.creador where u.nombre = "{user}"')
    #Se extrae el id del usuario actual, útil para realizar acciones post
eriores
    id_usuario = cursor2.execute(f'select id from usuarios where nombre =
 "{user}"')
    data = cursor.fetchall()
    global id_user
    id_user = cursor2.fetchall()
    return render_template('hecho.html', noticias=data, id_user=id_user,
usuario=user)
```

Rutas de redacción de posts (usuarios y editoriales)

Esta ruta, se accederá mediante el formulario destinado a la redacción de post de los usuarios y editoriales. Permite insertar una publicación en la base de datos, haciendo uso de la fecha actual y de la variable id_user, para referenciar el usuario que la realizó.

```
#Ruta de redacción de posts (usuarios y editoriales)
@app.route('/inicio/redactar', methods=['POST', 'GET'])
def redactar():
    if request.method == 'POST': #En caso de que el formulario haya sido
enviado
        titular = request.form['titular']
        cuerpo = request.form['cuerpo']
        tematica = request.form['topic']
        fecha_comentario = date.today() #Extrae la fecha actual y la alma
cena en una variable para su posterior insercción en la BD
        #Inserta la publicación en la BD
        cursor = mysql.connection.cursor()
        publicaciones = cursor.execute(f'insert into publicaciones(titula
r, contenido, creador, tematica, Fecha_creacion) values ("{titular}", "{c
uerpo}", "{id_user[0][0]}", "{tematica}", "{fecha_comentario}")')
        data = cursor.fetchall()
        mysql.connection.commit()
        return redirect(url_for('inicio'))
    else:
        return render_template('hecho.html')
```

Ruta de inicio de los admins

Esta ruta, es accedida cuando el login determina que un usuario es admin.

La ruta devuelve una plantilla con un pequeño panel de control, donde se muestran los usuarios (que no sean admins), las publicaciones y ciertas acciones que se permiten realizar.

```
@app.route('/inicio_admin')
def inicio_admin():
    cursor = mysql.connection.cursor()
    evaluar_tipo = cursor.execute(f'select tipo_usuario from usuarios whe
re nombre="{user}"')
    data = cursor.fetchall()
    mysql.connection.commit()
    if data[0][0] == "Admin": #Comprueba si el usuario que está accediend
o a la ruta es administrador o no
        #Mostrar los usuarios que no sean admins
        cursor1 = mysql.connection.cursor()
        mostrar_usuarios = cursor1.execute('select * from usuarios where
tipo_usuario="Usuario" or tipo_usuario="Editorial"')
        data1 = cursor1.fetchall()
        #Mostrar las publicaciones junto a su creador
        cursor2 = mysql.connection.cursor()
        mostrar_publicaciones = cursor2.execute('select p.id, p.titular,
p.Fecha_creacion, u.nombre from usuarios u inner join publicaciones p on
u.id = p.creador')
        data2 = cursor2.fetchall()
        cursor3 = mysql.connection.cursor()
        id_usuario = cursor3.execute(f'select id from usuarios where nomb
re = "{user}"')
        #Se extrae, en una variable global, el id del admin para realizar
 futuras operaciones
        global id_user
        id_user = cursor3.fetchall()
        mysql.connection.commit()
        return render_template('inicio_admin.html', usuarios=data1, notic
ias=data2)
    else: #En caso de que se intente acceder a la ruta desde un usuario n
o admin, saltará un mensaje de error
        return 'Acceso denegado'
```

Ruta de redacción de posts (admin)

Se trata de otra ruta para redactar post, pero en este caso, es accedida desde el admin. Desempeña la misma función que la otra ruta destinada a redactar.

```
@app.route('/inicio/redactar_admin', methods=['POST', 'GET'])
def redactar_admin():
    if request.method == 'POST':
        titular = request.form['titular']
        cuerpo = request.form['cuerpo']
        tematica = request.form['topic']
        fecha_comentario = date.today()
        cursor = mysql.connection.cursor()
```

```
        publicaciones = cursor.execute(f'insert into publicaciones(titula
r, contenido, creador, tematica, Fecha_creacion) values ("{titular}", "{c
uerpo}", "{id_user[0][0]}", "{tematica}", "{fecha_comentario}")')
        data = cursor.fetchall()
        mysql.connection.commit()
        return redirect(url_for('inicio_admin'))
    else:
        return render_template('index_admin.html')
```

Ruta para ordenar usuarios y posts

Se trata de una ruta, que es accedida mediante un formulario. Este formulario, se encuentra dentro de los filtros del administrador, y se encargan de realizar acciones que van desde buscar datos hasta ordenarlos. En este caso, esta ruta se encargará de mostrar tanto los usuarios como las publicaciones atendiendo a una serie de criterios.

```
@app.route('/inicio_admin/ordenar', methods=['POST', 'GET'])
def ordenar():
    if request.method == 'POST':
        ordenar_users = request.form['orderby'] #Extrae el criterio de or
denación establecido en el formulario
        cursor1 = mysql.connection.cursor()
        cursor2 = mysql.connection.cursor()
        if ordenar_users == 'Alfabéticamente': #Ordena de la A-
Z tanto usuarios como publicaciones
            usuarios = cursor1.execute('select * from usuarios where tipo
_usuario="Usuario" or tipo_usuario="Editorial" order by nombre asc')
            publicaciones = cursor2.execute('select p.id, p.titular, p.Fe
cha_creacion, u.nombre from usuarios u inner join publicaciones p on u.id
 = p.creador order by p.titular asc')
            data1 = cursor1.fetchall()
            data2 = cursor2.fetchall()
            return render_template('inicio_admin.html', usuarios=data1, n
oticias=data2)
        elif ordenar_users == 'Usuarios con más publicaciones': #Muestra
primero los usuarios con más publicaciones
            usuarios = cursor1.execute('select u.id, u.nombre, count(p.id
) cant_publicaciones, u.tipo_usuario  from usuarios u inner join publicac
iones p on u.id = p.creador where u.tipo_usuario = "Usuario" or u.tipo_us
uario = "Editorial" group by u.id order by cant_publicaciones desc')
            data1 = cursor1.fetchall()
            return render_template('inicio_admin.html', usuarios=data1)
        elif ordenar_users == 'Usuarios con menos publicaciones': #Muestr
a primero los usuarios con menos publicaciones
            usuarios = cursor1.execute('select u.id, u.nombre, count(p.id
) cant_publicaciones, u.tipo_usuario  from usuarios u inner join publicac
iones p on u.id=p.creador where u.tipo_usuario="Usuario" or u.tipo_usuari
o="Editorial" group by u.id order by cant_publicaciones asc')
```

```
            data1 = cursor1.fetchall()
            return render_template('inicio_admin.html', usuarios=data1)
        elif ordenar_users == 'Publicaciones más recientes': #Muestra pri
mero las publicaciones más recientes
            publicaciones = cursor2.execute('select p.id, p.titular, p.Fe
cha_creacion, u.nombre from usuarios u inner join publicaciones p on u.id
 = p.creador order by p.Fecha_creacion desc')
            data2 = cursor2.fetchall()
            return render_template('inicio_admin.html', noticias=data2)
        elif ordenar_users == 'Publicaciones menos recientes': #Muestra p
rimero las publicaciones menos recientes
            publicaciones = cursor2.execute('select p.id, p.titular, p.Fe
cha_creacion, u.nombre from usuarios u inner join publicaciones p on u.id
 = p.creador order by p.Fecha_creacion asc')
            data2 = cursor2.fetchall()
            return render_template('inicio_admin.html', noticias=data2)
        mysql.connection.commit()
    else:
        return render_template('inicio_admin.html')
```

Ruta para buscar usuarios

Se trata de otra ruta que es accedida mediante un formulario alojado en los filtros del administrador.

Esta ruta, realizará la función de buscar un usuario (no admin) en toda la BD, cuyo nombre de usuario, coincida con el texto del formulario. Ese texto puede ir desde unas pocas letras hasta un carácter.

```
@app.route('/inicio_admin/buscar_users', methods=['POST', 'GET'])
def buscar_users():
    if request.method == 'POST':
        buscar_user = request.form['buscar']
        cursor = mysql.connection.cursor()
        if buscar_user != '': #Comprueba si el campo no está vacio
            usuarios = cursor.execute(f'select * from usuarios where nomb
re like "%{buscar_user}%" and (tipo_usuario="Usuario" or tipo_usuario="Ed
itorial")')
        else: #Si el campo está vacio salta este error
            return 'Error de búsqueda'
        data = cursor.fetchall()
        mysql.connection.commit()
        return render_template('inicio_admin.html', usuarios=data)
    else:
        return render_template('inicio_admin.html')
```

Ruta para buscar por tipo de usuario

Se trata de otra ruta que es accedida mediante un formulario alojado en los filtros del administrador.

Esta ruta, se encargará de mostrar solo el tipo de usuario (excepto administrador) que el administrador le especifique.

```
@app.route('/inicio_admin/filtrar_tipo', methods=['POST', 'GET'])
def tipo_users():
    if request.method == 'POST':
        tipo = request.form['typeuser']
        cursor = mysql.connection.cursor()
        if tipo == 'Editorial':
            usuarios = cursor.execute('select * from usuarios where tipo_
usuario="Editorial"')
        elif tipo == 'Usuario':
            usuarios = cursor.execute('select * from usuarios where tipo_
usuario="Usuario"')
        data = cursor.fetchall()
        mysql.connection.commit()
        return render_template('inicio_admin.html', usuarios=data)
    else:
        return render_template('explorar.html')
```

Ruta para buscar post

Se trata de otra ruta que es accedida mediante un formulario alojado en los filtros del administrador.

Esta ruta, buscará todos los posts de la BD, comprobando si hay coincidencias entre el titular del post y el texto especificado en el formulario.

```
@app.route('/inicio_admin/buscar_posts', methods=['POST', 'GET'])
def buscar_post():
    if request.method == 'POST':
        buscar_post = request.form['buscar']
        cursor = mysql.connection.cursor()
        if buscar_post != '':
            publicaciones = cursor.execute(f'select * from publicaciones
where titular like "%{buscar_post}%"')
        data = cursor.fetchall()
        mysql.connection.commit()
        return render_template('inicio_admin.html', noticias=data)
    else:
        return render_template('inicio_admin.html')
```

Ruta para buscar posts mediante el creador

Se trata de otra ruta que es accedida mediante un formulario alojado en los filtros del administrador.

Esta ruta mostrará solo los posts de los usuarios especificados en el formulario.

```
@app.route('/inicio_admin/buscar_posts_user', methods=['POST', 'GET'])
def buscar_post_user():
    if request.method == 'POST':
        buscar_post_user = request.form['buscar']
        cursor = mysql.connection.cursor()
        if buscar_post_user != '':
            publicaciones = cursor.execute(f'select p.id, p.titular, p.Fe
cha_creacion, u.nombre from usuarios u inner join publicaciones p on u.id
 = p.creador where u.nombre = "{buscar_post_user}";')
        data = cursor.fetchall()
        mysql.connection.commit()
        return render_template('inicio_admin.html', noticias=data)
    else:
        return render_template('inicio_admin.html')
```

Ruta para mostrar posts mediante la fecha

Se trata de otra ruta que es accedida mediante un formulario alojado en los filtros del administrador.

Esta ruta, se encargará de mostrar los posts de una determinada fecha especificada en el formulario. El formato de la fecha debe ser AAAA/MM/DD.

```
@app.route('/inicio_admin/buscar_posts_fecha', methods=['POST', 'GET'])
def buscar_posts_fecha():
    if request.method == 'POST':
        buscar_posts_fecha = request.form['buscar']
        cursor = mysql.connection.cursor()
        if buscar_posts_fecha != '':
            publicaciones = cursor.execute(f'select * from publicaciones
where Fecha_creacion = "{buscar_posts_fecha}"')
        else:
            return 'Error de búsqueda'
        data = cursor.fetchall()
        mysql.connection.commit()
        return render_template('inicio_admin.html', noticias=data)
    else:
        return render_template('inicio_admin.html')
```

Ruta para acceder a una publicación

Consiste en una ruta (que es accedida mediante enlaces en las plantillas), la cual muestra todo lo relativo a las publicaciones.

```
@app.route('/inicio/noticia/<id>')
def news(id):
    #Se extrae, en una variable global, el id de la publicación en cuesti
ón
    global id_publicacion
    id_publicacion = id
    #Se muestra datos relativos a la publicacion y a su creador
    cursor = mysql.connection.cursor()
    noticia = cursor.execute(f'select u.nombre, p.* from publicaciones p
inner join usuarios u on u.id = p.creador where p.id = {id_publicacion}')
    data = cursor.fetchall()
    #Se muestra los comentarios de la publicación en cuestión
    cursor1 = mysql.connection.cursor()
    usuario_comentario = cursor1.execute(f'select u.nombre, c.contenido,
c.fecha, u.id from usuarios u inner join comentarios c on u.id = c.usuari
os inner join publicaciones p on p.id = c.publicaciones where p.id = {id}
 order by fecha desc')
    data1 = cursor1.fetchall()
    return render_template('noticia.html', titular=data[0][2], contenido=
data[0][3], creador=data[0][0], fecha_publicacion=data[0][6], comentarios
=data1)
```

Ruta para establecer un comentario

Se trata de una ruta, accedida desde un formulario, la cual se encarga de postear comentarios en las publicaciones.

```
@app.route('/inicio/noticia/comentario', methods=['POST', 'GET'])
def news_comment():
    if request.method == 'POST':
        comentario = request.form['comentario']
        fecha_comentario = date.today()
        cursor = mysql.connection.cursor()
        posteo = cursor.execute(f'insert into comentarios(contenido, fech
a, publicaciones, usuarios) values ("{comentario}", "{fecha_comentario}",
 {id_publicacion}, {id_user[0][0]})')
        data = cursor.fetchall()
        mysql.connection.commit()
        return redirect(url_for('news', id=id_publicacion))
    else:
        return render_template('noticia.html')
```

Ruta para editarse el usuario a si mismo (usuarios y editoriales)

```
@app.route('/inicio/editar_user/<id>', methods=['POST', 'GET'])
def edit_user(id):
    cursor = mysql.connection.cursor()
    usuario = cursor.execute(f'select * from usuarios where id = "{id_use
r[0][0]}"')
    data = cursor.fetchall()
    print(id_user)
    print(data)
    #Se restringe el acceso a otros usuarios mediante if (solo puede edit
arse a el mismo)
    if id_user[0][0] == data[0][0] :
        if request.method == 'POST':
            user1 = request.form['user']
            passwd = request.form['pass']
            confirm_passwd = request.form['confirmpass']
            if passwd == confirm_passwd:
                cursor1 = mysql.connection.cursor()
                update = cursor1.execute('update usuarios set nombre = %s
, pass = %s where id = %s', (user1, passwd, id_user))
                data1 = cursor1.fetchall()
                mysql.connection.commit()
                return redirect(url_for("login"))
            else:
                return 'Las contraseñas no coinciden'
        else:
            return render_template('edit_user.html', usuario=data, id=id_
user)
    else:
        return 'Acceso denegado'
```

Ruta para buscar un usuario y ver sus publicaciones

Esta ruta, es accedida mediante un formulario situado en la barra de búsqueda de la plantilla de inicial de los usuarios y editoriales.

Consiste en una ruta, la cual buscará y mostrará las publicaciones de cada uno de los usuarios.

```
@app.route('/inicio/buscar_user/<usuario>', methods=['POST', 'GET'])
def buscar_user(usuario):
    if request.method == 'POST':
        barraBuscar = request.form['buscar']
        cursor = mysql.connection.cursor()
        usuario_buscar = cursor.execute(f'select * from usuarios where no
mbre = "{barraBuscar}"')
        data = cursor.fetchall()
        cursor1 = mysql.connection.cursor()
```

```
        publicaciones_usuario = cursor1.execute(f'select p.* from usuario
s u inner join publicaciones p on u.id = p.creador where u.nombre = "{bar
raBuscar}" order by p.Fecha_creacion desc;')
        data1 = cursor1.fetchall()
        mysql.connection.commit()
        if usuario_buscar == True:
            return render_template('buscar_user.html', usuarios=data, pub
licaciones=data1)
        else:
            return 'El usuario no existe'
    else:
        return redirect(url_for("inicio", usuarios=data))
```

Ruta explorar

Es una ruta, que devuelve una plantilla la cual mostrará una serie de filtros para buscar y ordenar publicaciones. Las publicaciones serán mostradas, cuando se aplique algún filtro.

Esta ruta es accedida mediante el botón de la brújula.

```
@app.route('/inicio/explorar', methods=['POST', 'GET'])
def explorar():
    if request.method == 'POST':
        ordenar_por = request.form['orderby']
        cursor = mysql.connection.cursor()
        if ordenar_por == 'Más recientes':
            todas_publicaciones = cursor.execute('select * from publicaci
ones order by Fecha_creacion desc')
        elif ordenar_por == 'Más antiguos':
            todas_publicaciones = cursor.execute('select * from publicaci
ones order by Fecha_creacion asc')
        elif ordenar_por == 'Titular':
            todas_publicaciones = cursor.execute('select * from publicaci
ones order by titular asc')
        data = cursor.fetchall()
        mysql.connection.commit()
        return render_template('explorar.html', publicaciones=data)
    else:
        return render_template('explorar.html')
```

Rutas con otros filtros para las publicaciones

Son casi idénticas a las rutas de filtros del administrador.

```
#Ruta para buscar post por titular
@app.route('/inicio/explorar/buscar_titular', methods=['POST', 'GET'])
def explorar_titular():
```

```
    if request.method == 'POST':
        buscar_post = request.form['buscar']
        cursor = mysql.connection.cursor()
        if buscar_post != '':
            todas_publicaciones = cursor.execute(f'select * from publicac
iones where titular like "%{buscar_post}%"')
        data = cursor.fetchall()
        mysql.connection.commit()
        return render_template('explorar.html', publicaciones=data)
    else:
        return render_template('explorar.html')
#Ruta para buscar post por fecha
@app.route('/inicio/explorar/buscar_fecha', methods=['POST', 'GET'])
def explorar_fecha():
    if request.method == 'POST':
        buscar_fecha = request.form['buscar']
        cursor = mysql.connection.cursor()
        if buscar_fecha != '':
            todas_publicaciones = cursor.execute(f'select * from publicac
iones where Fecha_creacion = "{buscar_fecha}"')
        data = cursor.fetchall()
        mysql.connection.commit()
        return render_template('explorar.html', publicaciones=data)
    else:
        return render_template('explorar.html')
#Ruta para buscar post por temática
@app.route('/inicio/explorar/buscar_tematica', methods=['POST', 'GET'])
def explorar_temarica():
    if request.method == 'POST':
        buscar_tematica = request.form['buscartematica']
        cursor = mysql.connection.cursor()
        if buscar_tematica != '':
            todas_publicaciones = cursor.execute(f'select * from publicac
iones where tematica = "{buscar_tematica}"')
        data = cursor.fetchall()
        mysql.connection.commit()
        return render_template('explorar.html', publicaciones=data)
    else:
        return render_template('explorar.html')
```

Ruta para borrar usuarios

Esta ruta, referencia una acción que pueden hacer los administradores. Esta acción es la de borrar cualquier usuario de tipo usuario o editorial.

```
#Ruta para borrar usuarios (menos los administradores)
@app.route('/inicio_admin/borrar_user/<id>', methods=['POST', 'GET'])
def borrar_user(id):
    cursor = mysql.connection.cursor()
```

```
    admins = cursor.execute(f'select tipo_usuario from usuarios where id
= "{id}"')
    data = cursor.fetchall()
    if data[0][0] == 'Admin':
        return 'Error, no se puede eliminar un administrador'
    else:
        cursor1 = mysql.connection.cursor()
        borrar_usuario = cursor1.execute(f'delete from usuarios where id
= "{id}"')
        data1 = cursor1.fetchall()
        mysql.connection.commit()
        return redirect(url_for("inicio_admin"))
```

Ruta para borrar posts

Es otra acción que puede realizar el administrador, mediante esa ruta.

```
#Ruta para borrar posts
@app.route('/inicio_admin/borrar_post/<id>', methods=['POST', 'GET'])
def borrar_post(id):
    cursor = mysql.connection.cursor()
    admins = cursor.execute(f'select tipo_usuario from usuarios where id
= "{id}"')
    data = cursor.fetchall()
    if data == 'Admin':
        return 'Error, no se puede eliminar un administrador'
    else:
        cursor1 = mysql.connection.cursor()
        borrar_usuario = cursor1.execute(f'delete from publicaciones wher
e id = "{id}"')
        data1 = cursor1.fetchall()
        mysql.connection.commit()
        return redirect(url_for("inicio_admin"))
```

Error

Salta cuando la ruta no existe.

```
@app.errorhandler(404)
def page_not_found(error):
    return 'Página no encontrada...', 404
```