

## DISEÑO DEL SERVICIO RESTFUL (MEMORIA DE LA PRÁCTICA) DAVID LÁZARO MARTÍN

En esta práctica se expone el diseño de una API Rest y su implementación en Python. Los métodos se comprueban por medio de un cliente llamado Insomnia. Como el programa se ejecuta en local, una vez publicado se sustituiría en las URIs el localhost:4000 por la URI que queremos en este caso <http://api.upmsocial.es>. A continuación se expone un ejemplo de uso de todos los métodos que soporta la API:

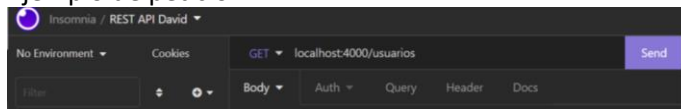
- Obtener una lista de todos los usuarios de la red

URI	http://api.upmsocial.es/usuarios	
Método	GET	
Cadena de consulta		
Devuelve	200	OK y application/json con una lista de los nombres de usuarios, email y nombre de todos los usuarios.
	401	Unathorithed
	404	Not Found

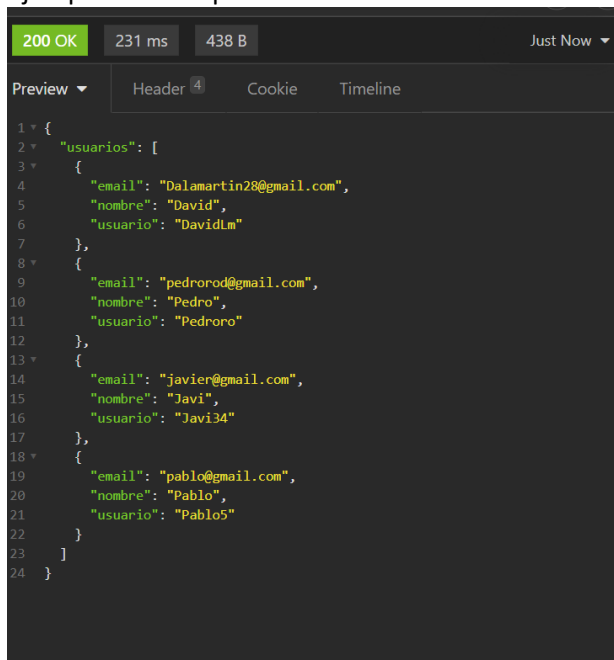
Código de la API:

```
@app.route("/usuarios")
def getUsuarios():
    return jsonify({
        "usuarios": [usuario.to_json() for usuario in usuarios]
    })
```

Ejemplo de petición:



Ejemplo de la respuesta:



- Añadir un usuario nuevo

URI	http://api.upmsocial.es/usuarios	
Método	POST	
Cuerpo de la petición	application/json	
Devuelve	200	OK se crea el usuario y se devuelve una lista actualizada de todos los usuarios con un mensaje de confirmación 'usuario creado' en formato application/json
	400	Bad Request
	401	Unathorithed
	404	Not Found

Código de la API:

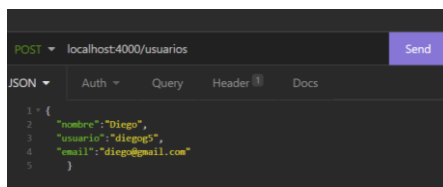
```

47 @app.route("/usuarios", methods=["POST"])
48 def addUsuario():
49     new_usuarios = {
50         "nombre": request.json["nombre"],
51         "email": request.json["email"],
52         "usuario": request.json["usuario"]
53     }
54     User = Usuario(request.json["nombre"], request.json["usuario"], request.json["email"])
55     if User in usuarios:
56         return jsonify({"message": "El usuario ya existe"})
57     usuarios.append(User)
58     return jsonify({
59         "message": "Usuario creado",
60         "usuarios": [usuario.to_json() for usuario in usuarios]
61     })
62

```

Como se ha sobrescrito el método `__eq__` sobre la clase `Usuario`, con comprobar si un usuario es igual a otro nos aseguramos de no repetir nombres de usuario, aunque el nombre o el email puedan ser el mismo.

Ejemplo de cuerpo de la petición:

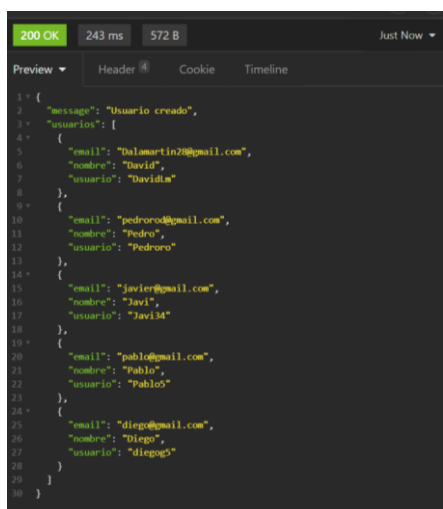


```

POST localhost:4000/usuarios
{
  "nombre": "Diego",
  "usuario": "diego5",
  "email": "diego@gmail.com"
}

```

Ejemplo de respuesta:



```

200 OK
{
  "message": "Usuario creado",
  "usuarios": [
    {
      "email": "Diamartin2@gmail.com",
      "nombre": "David",
      "usuario": "Davidm"
    },
    {
      "email": "pedrorod@gmail.com",
      "nombre": "Pedro",
      "usuario": "Pedroo"
    },
    {
      "email": "javier@gmail.com",
      "nombre": "Javi",
      "usuario": "JaviM"
    },
    {
      "email": "pablo@gmail.com",
      "nombre": "Pablo",
      "usuario": "Pablo5"
    },
    {
      "email": "diego@gmail.com",
      "nombre": "Diego",
      "usuario": "diego5"
    }
  ]
}

```

- Publicar un nuevo post (mensaje en la red)

URI	http://api.upmsocial.es/usuarios/{username}/posts	
Método	POST	
Cuerpo de la petición	application/json con los datos del post	
Devuelve	200	OK. Crea el post y devuelve una lista con todos los posts del usuario actualizada y un mensaje de confirmación 'post añadido' en formato application/Json.
	400	Bad Request
	401	Unathorithed
	404	Not Found

Código de la API:

```

150 @app.route("/usuarios/<string:username>/posts", methods = ["POST"])
151 def newPost(username):
152     for user in usuarios:
153         if user.usuario == username:
154             Post1 = Post(request.json["titulo"], request.json["texto"], datetime.now(), getid())
155             user.add_post(Post1)
156             posts = [post.to_json() for post in user.posts]
157             return jsonify({
158                 "message": "Post añadido",
159                 "Posts": posts})
160     return jsonify({
161         "message": "El usuario no ha sido encontrado por lo que no se puede publicar el post."})
162

```

Se crea un post con la fecha actual y el primer id libre lo cual hacemos con una función que hemos definido llamada getid(). En el caso del ejemplo como tenemos 12 post anteriores se crea con el id = 13

Ejemplo de cuerpo de la petición:

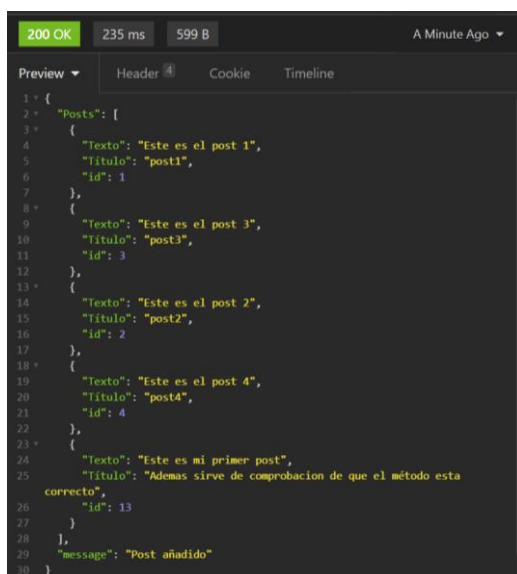


```

1 {
2   "texto": "Este es mi primer post",
3   "titulo": "Ademas sirve de comprobacion de que el método esta correcto"
4 }

```

Ejemplo de respuesta:



```

1 {
2   "Posts": [
3     {
4       "Texto": "Este es el post 1",
5       "Titulo": "post1",
6       "id": 1
7     },
8     {
9       "Texto": "Este es el post 3",
10      "Titulo": "post3",
11      "id": 3
12     },
13     {
14       "Texto": "Este es el post 2",
15       "Titulo": "post2",
16       "id": 2
17     },
18     {
19       "Texto": "Este es el post 4",
20       "Titulo": "post4",
21       "id": 4
22     },
23     {
24       "Texto": "Este es mi primer post",
25       "Titulo": "Ademas sirve de comprobacion de que el método esta
correcto",
26       "id": 13
27     }
28   ],
29   "message": "Post añadido"
30 }

```

- Editar un post

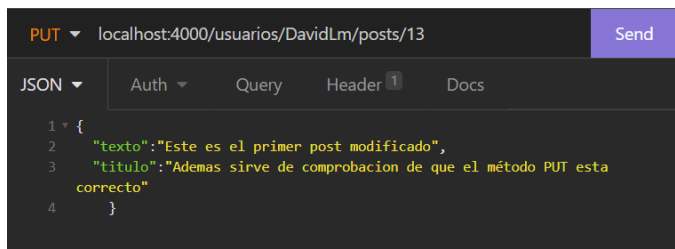
URI	http://api.upmsocial.es/usuarios/{username}/posts/{post_id}	
Método	PUT	
Cadena de consulta	Application/json con los datos actualizados del post	
Devuelve	200	OK y se actualiza el post con el id indicado con los nuevos datos y nueva fecha
	401	Unathorithed
	404	Not Found

Código de la API:

```
@app.route("/usuarios/<string:username>/posts/<int:postid>", methods = ["PUT"])
def updatePost(username, postid):
    for user in usuarios:
        if user.usuario == username:
            if len(user.posts) > 0:
                for post in user.posts:
                    if post.id == postid:
                        post.set_date()
                        post.set_texto(request.json['texto'])
                        post.set_titulo(request.json['titulo'])
                        return jsonify({
                            "message": "Post actualizado",
                            "new Post": post.to_json()})
                    else:
                        return jsonify({
                            "message": "El id no ha sido encontrado por lo que no se puede modificar el post."})
            else: return jsonify({
                "message": "El Usuario no tiene ningún post."})
    return jsonify({
        "message": "Tu usuario no ha sido encontrado por lo que no se puede modificar el post."})
```

Se accede al post a través del id y se modifica el objeto por medio de métodos implementados ya que los atributos son privados y no se pueden modificar directamente.

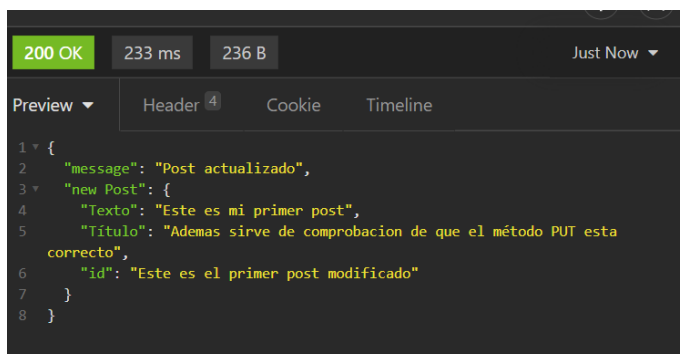
Ejemplo de la petición: Accedemos al post con id 13 y modificamos el post.



```
PUT localhost:4000/usuarios/DavidLm/posts/13
```

```
{
  "texto": "Este es el primer post modificado",
  "titulo": "Ademas sirve de comprobacion de que el método PUT esta correcto"
}
```

Ejemplo de la respuesta: Nos devuelve un mensaje de confirmación 'Post actualizado' y el post actualizado.



```
200 OK 233 ms 236 B Just Now
```

```
{
  "message": "Post actualizado",
  "new Post": {
    "Texto": "Este es mi primer post",
    "Titulo": "Ademas sirve de comprobacion de que el método PUT esta correcto",
    "id": "Este es el primer post modificado"
  }
}
```

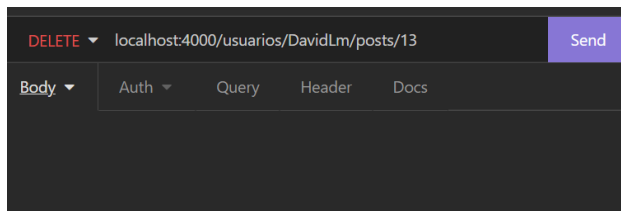
- Eliminar un post

URI	http://api.upmsocial.es/usuarios/{username}/posts/{post_id}	
Método	DELETE	
Devuelve	200	OK.
	404	Not Found

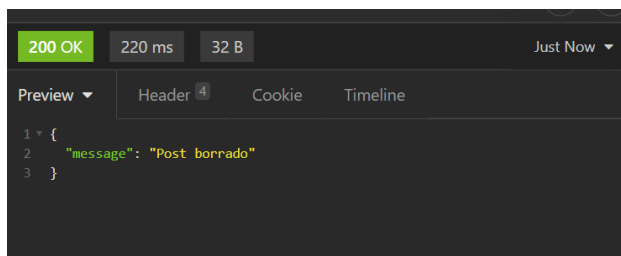
Código de la API:

```
@app.route ("/usuarios/<string:username>/posts/<int:postid>", methods = ["DELETE"])
def deletePost(username, postid):
    for user in usuarios:
        if user.usuario == username:
            if len(user.posts) > 0:
                for post in user.posts:
                    if post.id == postid:
                        user.remove_post(post)
                        return jsonify({
                            "message": "Post borrado"})
                    else:
                        return jsonify({
                            "message": "El id no ha sido encontrado por lo que no se puede borrar el post."})
            else: return jsonify({
                            "message": "El Usuario no tiene ningún post."})
    return jsonify({
        "message": "El usuario no ha sido encontrado por lo que no se puede borrar el post."})
```

Ejemplo de la petición:

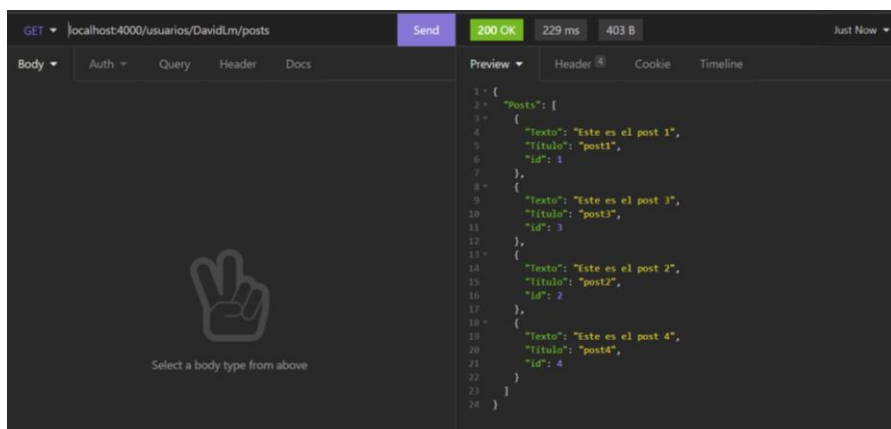


Ejemplo de la respuesta:



Ahora si volvemos a mostrar todos los posts, no sale el post con id=13

Aquí podemos ver (por otro método GET adicional que he creado que devuelve solo los posts de un usuario) como el post ha sido borrado.



- Obtener una lista de todos nuestros posts y filtrar esa lista por fecha o limitar la cantidad de información obtenida por número de posts (e.g. los 10 primeros elementos, los elementos entre el 11 y el 20, etc.)

URI	http://api.upmsocial.es/posts?fecha_inicio=val1&?fecha_fin=val2&?post_inicio=val3&?post_fin=val4	
Método	GET	
Cadena de consulta	fecha_inicio=	Fecha a partir de la cual quieres filtrar los post: str formato 'año-mes-día'
	fecha_fin=	Fecha hasta la cual quieres filtrar los post: str formato 'año-mes-día'
	post_inicio=	Numero de post a partir del cual quieres que te devuelva: int
	post_fin=	Numero de post hasta el cual quieres que te devuelva: int
Devuelve	200	OK y application/json con una lista de los post que cumplen los parametros
	400	Bad Request
	401	Unauthorized
	404	Not Found

Código de la API:

```
@app.route("/posts", methods=["GET"])
def getallPosts():
    posts=[]
    for user in usuarios:
        for post in user.posts:
            if (datetime.strptime(request.args.get('fecha_fin', '2100-01-01'), '%Y-%m-%d') >= post.creation_date >= datetime.strptime(request.args.get('fecha_inicio', '2100-01-01'), '%Y-%m-%d')):
                posts += [post.to_json()]
    return jsonify({"Posts": posts[int(request.args.get('post_inicio', 0)):int(request.args.get('post_fin', len(posts)))]})
```

Ejemplo de respuesta de la petición:

GET localhost:4000/usuarios/DavidLm/posts?post\_inicio=1&post\_fin=2?fecha\_inicio=2019-1-1&fecha\_fin=2021-1-1 Send

200 OK 210 ms 115 B 4 Minutes Ago

Preview Header Cookie Timeline

```
{
  "Posts": [
    {
      "Texto": "Este es el post 2",
      "Titulo": "post2",
      "Id": 2
    }
  ]
}
```

- Obtener el número de posts publicados por mí en la red social en un determinado periodo (fecha inicio y fin)

URI	http://api.upmsocial.es/usuarios/?user=val1/posts/cantidad?fecha_inicio=val2&?fecha_fin=val3	
Método	GET	
Cadena de consulta	User=	Usuario propio: str
	fecha_inicio=	Fecha a partir de la cual quieres filtrar los posts: str formato 'año-mes-día'
	fecha_fin=	Fecha hasta la cual quieres filtrar los posts: str formato 'año-mes-día'
Devuelve	200	OK y devuelve un tipo entero
	400	Bad Request
	401	Unauthorized
	404	Not Found

Código de la API:

```
@app.route("/posts/contador", methods=["GET"])
def getPostscount():
    posts = []
    for user in usuarios:
        for post in user.posts:
            if (datetime.strptime(request.args.get('fecha_fin', '2100-01-01'), '%Y-%m-%d') >= post.creation_date >= da
                posts += [post]
    return jsonify({"Posts": len(posts)})
```

Ejemplo de la respuesta

The screenshot shows a REST client interface. The top bar indicates a GET request to `localhost:4000/posts/contador?fecha_inicio=2019-1-1&fecha_fin=2021-1-1` was sent. The response status is `200 OK` with a response time of `235 ms` and a body size of `17 B`. The response body is displayed in the 'Preview' tab as a JSON object: `{ \"Posts\": 8 }`.

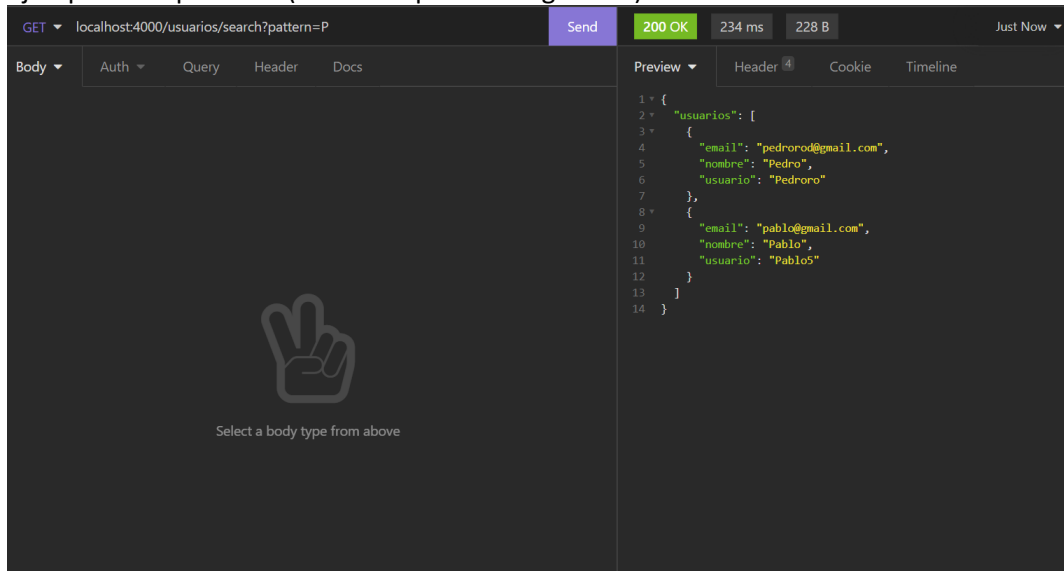
- Buscar posibles amigos en la red por nombre (patrón)

URI	http://api.upmsocial.es/usuarios/search?pattern=va0	
Método	GET	
Cadena de consulta	Pattern= patrón que quieres buscar.	
Devuelve	200	OK y text/xml con el perfil del usuario buscado
	401	Unatorithed
	404	Not Found

Código de la API:

```
@app.route("/usuarios/search")
def getFriends():
    users = []
    for user in usuarios:
        if re.search(request.args.get('pattern',''), user.usuario):
            users += [user.to_json()]
    return jsonify({'usuarios': users})
```

Ejemplo de la petición: (Usuarios que contengan la P)



GET localhost:4000/usuarios/search?pattern=P Send 200 OK 234 ms 228 B Just Now

Body Auth Query Header Docs Preview Header 4 Cookie Timeline

```
1 {
2   "usuarios": [
3     {
4       "email": "pedrorod@gmail.com",
5       "nombre": "Pedro",
6       "usuario": "Pedroro"
7     },
8     {
9       "email": "pablo@gmail.com",
10      "nombre": "Pablo",
11      "usuario": "Pablo5"
12    }
13  ]
14 }
```

Select a body type from above



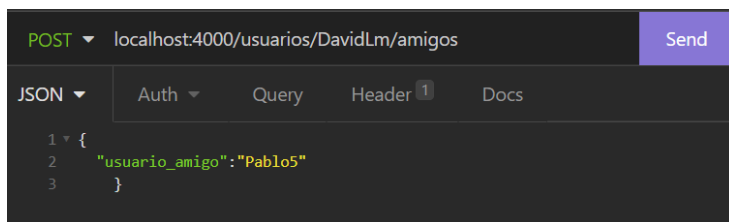
- Añadir un nuevo amigo

URI	http://api.upmsocial.es/usuarios/{username}/amigos	
Método	POST	
Cuerpo de la petición	Applicaction/json	
Devuelve	200	OK y mensaje de confirmación
	400	Bad Request
	401	Unatorithed
	404	Not Found

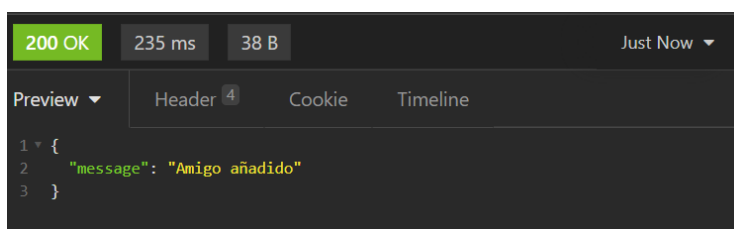
Código de la respuesta:

```
@app.route('/usuarios/<string:username>/amigos', methods=['POST'])
def addFriend(username):
    for user in usuarios:
        if user.usuario == username:
            for user2 in usuarios:
                if request.json['usuario_amigo'] == user2.usuario:
                    user.add_amigo(request.json['usuario_amigo'])
                    return jsonify({'message': 'Amigo añadido'})
            return jsonify({'message': 'El usuario que buscas no está en la red social'})
    return jsonify({
        "message": "Tu usuario no ha sido encontrado por lo que no se puede añadir el amigo."})
```

Ejemplo de la petición:



Ejemplo de la respuesta:



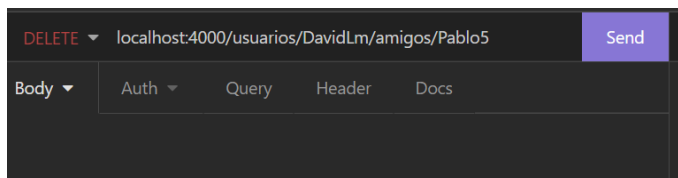
- Eliminar un amigo

URI	http://api.upmsocial.es/usuarios/{username}/amigos/{amigo_user}	
Método	DELETE	
Cuerpo de la petición		
Devuelve	200	OK y mensaje de confirmación
	404	Not Found

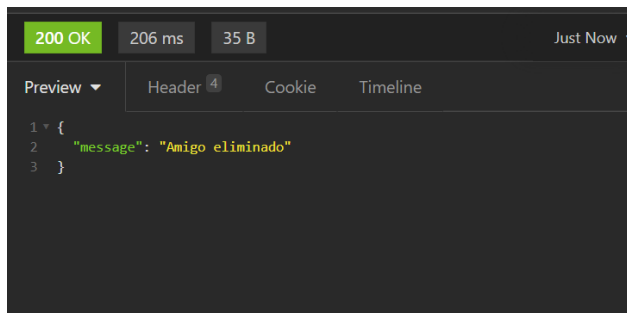
Código de la API:

```
@app.route('/usuarios/<string:username>/amigos/<amigo_username>', methods=['DELETE'])
def deleteFriend(username, amigo_username):
    for user in usuarios:
        if user.usuario == username:
            for amigo in user.amigos:
                if amigo_username == amigo:
                    user.remove_amigo(amigo_username)
                    return jsonify({'message': 'Amigo eliminado'})
            return jsonify({'message': 'El usuario que buscas no es amigo tuyo o no está en la red.'})
    return jsonify({'message': 'Tu usuario no ha sido encontrado por lo que no se puede añadir el amigo.'})
```

Ejemplo de la petición:



Ejemplo de la respuesta:



- Obtener una lista de todos nuestros amigos y filtrar esa lista por nombre o limitar la cantidad de información obtenida por número de amigos (e.g. los 10 primeros elementos, los elementos entre el 11 y el 20, etc.)

URI	http://api.upmsocial.es/usuarios/{username}/amigos?amigo_inicio=val0&?amigo_fin=val1&?patter=val3	
Método	GET	
Cadena de consulta	amigo_inicio=	Numero de amigo a partir del cual quieres que te devuelva:int
	amigo_fin=	Numero de amigo hasta el cual quieres que te devuelva:int
	Pattern=	Patron que contiene o nombre del amigo: str
Devuelve	200	OK y application/Json con la lista de usuarios que cumplen
	400	Bad Request
	401	Unathorithed
	404	Not Found

Código de la petición:

```
@app.route("/usuarios/<string:username>/amigos", methods=["GET"])
def getAmigos(username):
    for user in usuarios:
        if user.usuario == username:
            if len(user.amigos) == 0:
                return jsonify({
                    "message": "El usuario no tiene ningún amigo."})
            else:
                amigos = []
                for amigo in user.amigos:
                    if re.search(request.args.get('pattern',''), amigo):
                        amigos += [amigo]
                return jsonify({'amigos':amigos[int(request.args.get('amigo_inicio',0)):int(request.args.get('amigo_fin',0))])
    return jsonify({
        "message": "El usuario no ha sido encontrado."})
```

Ejemplo de respuesta

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** localhost:4000/usuarios/DavidLm/amigos?pattern=P&?amigo\_inicio=0&amigo\_fin=2
- Status:** 200 OK
- Time:** 210 ms
- Size:** 36 B
- Body:**

```
{
  "amigos": [
    "Pedroro"
  ]
}
```

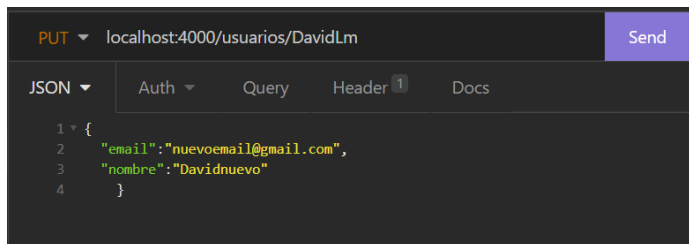
- Cambiar datos de nuestro perfil (excepto el nombre de usuario)

URI	http://api.upmsocial.es/usuarios/{username}	
Método	PUT	
Cuerpo de la petición	Application/json	
Devuelve	200	OK, mensaje de confirmación y el usuario actualizado
	400	Bad Request
	401	Unathorithed
	404	Not Found

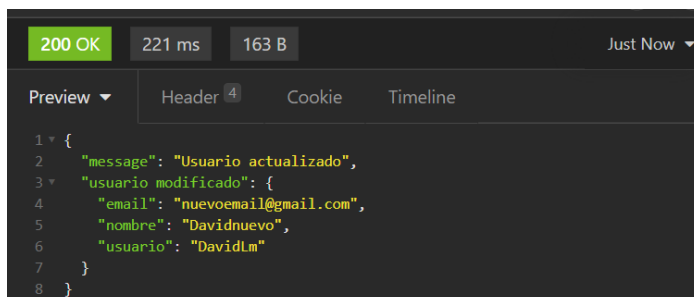
Código de la API:

```
@app.route("/usuarios/<string:username>", methods=["PUT"])
def editUsuario(username):
    for user in usuarios:
        if user.usuario == username:
            user.set_email(request.json["email"])
            user.set_nombre(request.json["nombre"])
            return jsonify({
                "message": "Usuario actualizado",
                "usuario modificado": user.to_json()
            })
    return jsonify({"message": "Usuario no encontrado"})
```

Ejemplo de la petición:



Ejemplo de la respuesta:



- Borrar nuestro perfil de la red social

URI	http://api.upmsocial.es/usuarios/{username}	
Método	DELETE	
Cuerpo de la petición		
Devuelve	200	OK y application/Json con la lista actualizada de usuarios
	404	Not Found

Código de la API:

```
@app.route("/usuarios/<string:username>", methods=["DELETE"])
def deleteUsuario(username):
    for user in usuarios:
        if user.usuario == username:
            usuarios.remove(user)
            return jsonify({
                "message": "Usuario borrado",
                "usuarios": [usuario.to_json() for usuario in usuarios]
            })
    return jsonify({
        "message": "El usuario no ha sido encontrado."})
```

Ejemplo de petición:

DELETE localhost:4000/usuarios/Pablo5 Send

Body Auth Query Header Docs

Ejemplo de respuesta:

200 OK 230 ms 373 B A Minute Ago

Preview Header 4 Cookie Timeline

```

1 {
2   "message": "Usuario borrado",
3   "usuarios": [
4     {
5       "email": "nuevoemail@gmail.com",
6       "nombre": "Davidnuevo",
7       "usuario": "DavidLm"
8     },
9     {
10      "email": "pedrorod@gmail.com",
11      "nombre": "Pedro",
12      "usuario": "Pedroro"
13    },
14    {
15      "email": "javier@gmail.com",
16      "nombre": "Javi",
17      "usuario": "Javi34"
18    }
19  ]
20 }
```

- Consultar los posts de nuestros amigos: obtener una lista de todos los posts publicados de todos mis amigos, pudiendo filtrar estos posts por fecha y/o por contenido –contiene un determinado texto- o limitar la cantidad de información

URI	http://api.upmsocial.es/usuarios/{user_id}/amigos/posts?fecha_inicio=val1&?fecha_fin=val2&?post_inicio=val3&?post_fin=val4&?pattern=val5	
Método	GET	
Cadena de consulta	fecha_inicio=	Fecha a partir de la cual quieres filtrar los post: str formato 'año-mes-día'
	fecha_fin=	Fecha hasta la cual quieres filtrar los post tipo: str
	post_inicio=	Numero de post a partir del cual quieres que te devuelva: int
	post_fin=	Numero de post hasta el cual quieres que te devuelva: int
	pattern=	Texto que queremos que contengan los posts: str
Devuelve	200	OK y text/xml con una lista con los posts de los amigos del usuario que cumplen las condiciones
	400	Bad Request
	401	Unauthorized
	404	Not Found

Código de la API:

```
@app.route('/usuarios/<string:username>/amigos/posts')
def getAmigosPosts(username):
    posts = []
    for user in usuarios:
        if user.usuario == username:
            for amigo in user.amigos:
                for user1 in usuarios:
                    if user1.usuario == amigo:
                        for post in user1.posts:
                            if re.search(request.args.get('pattern',''), amigo) and datetime.strptime(request.args.get('fecha_inicio',''), '%Y-%m-%d') <= datetime.strptime(request.args.get('fecha_fin',''), '%Y-%m-%d'):
                                posts += [post.to_json()]
    return jsonify({'Posts':posts[int(request.args.get('post_inicio',0)):int(request.args.get('post_fin',10000))])
```

Ejemplo de la respuesta

GET
localhost:4000/usuarios/DavidLm/amigos/posts?pattern=P&post\_inicio=0&post\_fin=2&fecha\_inicio=2020-1-1&fecha\_fin=2021-1-1
Send

200 OK
226 ms
213 B
Just Now

Preview
Header
Cookie
Timeline

```

1 {
2   "Posts": [
3     {
4       "Texto": "Este es el post 11",
5       "Titulo": "post3",
6       "id": 11
7     },
8     {
9       "Texto": "Este es el post 7",
10      "Titulo": "post3",
11      "id": 7
12    }
13  ]
14 }
```