PIPER API Documentation

Pros of tastypie:

- Well-documented
- Easy setup
- It works

Django Rest cons:

- Poorly-documented

Our choice of API framework largely depended on how easy it was to understand and implement it. We sought to first try one out, and, if we were struggling, switch to the other. TastyPie's clear, easily-located, and helpful documentation was perfect for our needs and allowed us to quickly and efficiently set up an API system for our web service. From our research, we have determined that there isn't really a significant difference between TastyPie and Django REST Framework in terms of what they do and what they handle. Both are reputable API frameworks for Django that have been used by many, many developers over the past decade.

Managing Users

Use these endpoints to create and edit users

Request Fields

These fields are to be used for all user related POSTs:

Parameter (*required)	Value	Example
username*	string	“NewUser01”
password*	string	“SeCrEtPaSSWord”
email*	string (valid email address)	“user@host.com”
birthday*	integer	“04021993”
gender*	character - (M/F/O)	“F”

Response Fields

In addition to the request fields, note the following response fields:

Parameter	Value	Example
id	integer	1
created_at	DateTimeField	"2015-03-24T23:05:36.946869"
last_modified	DateTimeField	"2015-03-25T14:07:13.467439"
resource_uri	API Endpoint (string)	"/api/users/user/1/"

Create a User

User this endpoint to create a new Piper user

API Endpoint: http://piper.link/api/users/user/

Example curl call:
curl --dump-header - -H “Content-type: application/json” -X POST --data ‘{“username”: “NewUser01”, “password”: “SeCrEtPaSSWord”, “email”: “user@host.com”, “birthday”: “04021993”, “gender”: “F”}’ http://piper.link/api/users/user/

Edit an Existing User

Use this endpoint to edit an existing Piper user

API Endpoint: http://piper.link/api/users/user/1

Example curl call:
curl --dump-header - -H “Content-type: application/json” -X POST --data ‘{“username”: “NewUser01”, “password”: “SeCrEtPaSSWord”, “email”: “user@host.com”, “birthday”: “04021993”, “gender”: “O”}’ http://piper.link/api/users/user/1

Retrieve a User

Use this endpoint to retrieve an existing Piper user or set of Piper users

API Endpoint: http://piper.link/api/users/user/

Example curl call:
curl -X GET http://piper.link/api/users/user/1

Managing Posts

Use these endpoints to create, read, and delete posts

Request Fields

These fields are to be used for all post related POSTs:

Parameter (*required)	Value	Example
text_content*	string	“Read My New Post!”
image	file path	/User/Desktop/image.jpeg

Response Fields

In addition to the request fields, note the following response fields:

Parameter	Value	Example
id	integer	1
created_at	DateTimeField	"2015-03-24T23:05:36.946869"
last_modified	DateTimeField	"2015-03-25T14:07:13.467439"
resource_uri	API Endpoint (string)	"/api/posts/post/1/"

Create a Post

User this endpoint to create a new post

API Endpoint: http://piper.link/api/posts/post/

Example curl call:
curl --dump-header - -H “Content-type: application/json” -X POST --data ‘{“text_content”: ”Read My New Post!”}’ http://piper.link/api/posts/post/

Retrieve a Post

User this endpoint to retrieve an existing posts or set of posts

API Endpoint: http://piper.link/api/posts/post/

Example curl call:
curl -X GET http://piper.link/api/posts/post/1

Delete an Existing Post

Use this endpoint to delete an existing post

API Endpoint: http://piper.link/api/posts/post/22

Example curl call:
curl --dump-header - -H “Content-type: application/json” -X DELETE
http://piper.link/api/posts/post/30

NOTES:

In many cases, the client will need to know the ID of the resource in order to specifically retrieve, edit, or delete that resource. These IDs can be retrieved via a filtered or unfiltered GET request on the resource.

General Permissions:

Users can edit Users
Users can create Users
Users **can’t** delete Users

Users can create Posts
Users **can’t** edit Posts
Users can delete Posts

Authentication

With further permissions (after we implement authentication), users WON’T be able to:

- delete other users’ posts
- edit other users (edit/update users is not yet implemented for this reason)

With further permissions (after we implement authentication), users WILL be able to:

- edit their own user (edit/update is not yet functional)

Comments for Posts (currently not implemented)

As a group, we are still discussing *how* we are going to implement comments
A few thoughts:

- comments will be the property of each user
- only the comment owner can edit, or delete comments (POST and DELETE)
- anyone can retrieve anyone else’s comments (GET)
- comments will be direct children of an individual post