



No environment

Double Column ▼

Public

d20 Spell Manager

Introduction

The API provides routes to help manage characters and their spells in tabletop games using the "d20 System" e.g. Wizards of the Coast's "Dungeons & Dragons". Players of such games can use the API (ideally through the web app utilising it) to collate all of their spells in one place, rather than over several different books.

Request/Response Bodies

All app routes accepting bodies (i.e. all POST routes except `/api/admin/delspell`) expect the data to be in JSON form. All app routes returning object bodies will return it as JSON.

Authentication

App routes under `/auth` and `/api` are authenticated using basic access authentication (https://en.wikipedia.org/wiki/Basic_access_authentication), with routes under `/api/admin` only accepting admin credentials.

Error Codes

- 200 - Any successful request will return a 200.
- 400 - This is sent by POST routes when the body is missing parameters that are needed to complete the request. It is also sent when the client attempts to edit or delete a non-existent character or spell.
- 401 - Sent whenever authentication fails. This will happen whenever incorrect credentials are supplied or non-admin credentials are supplied for an admin route.
- 403 - Sent whenever a client attempts to access a resource that is not associated with the authenticated user.
- 500 - Sent whenever express or my code encounters an error processing the request.

Rate limit

None.



No environment

Double Column ▼

Public

GET http://localhost:8090/ping

http://localhost:8090/ping

Check whether the web service is online

Will return an empty body with status 200 if the web service is up. If the web service is not up, the request will fail.

Example Request

Basic Example

```
curl --location --request GET 'http://localhost:8090/ping' \
--data-raw ''
```

GET http://localhost:8090/auth

http://localhost:8090/auth

Determine whether the supplied authentication is for a user or admin

If the supplied credentials are correct and correspond to a non-admin user, it will return body "user".

If the credentials are correct and correspond to an admin, it will return body "admin".

If the credentials are incorrect, it will return code 401.

Example Request

Basic Example

```
curl --location --request GET 'http://localhost:8090/auth' \
--data-raw ''
```

Example Response	No environment	Double Column ▼	Public
200 - OK			
user			

POST <http://localhost:8090/api/admin/addspell>

<http://localhost:8090/api/admin/addspell>

Add a spell to the list of available spells

Request body:

A JSON object with the the following properties:

Property	Type	Description
Name	String	The name of the spell
Level	Number (in the range 0-9)	The level of the spell
School	String	The school of magic the spell belongs to
Components	String	The components requiried to cast the spell
Damage	String	The damage done by the spell
Desc	String	A description of the spell's effects

Response code:

- 200 - Request completed successfully, spell saved to disk
- 400 - Not all of the information for the spell was given in the request, or the spell level was outside of the range.
- 401 - Authentication failed
- 500 - The server encountered an error

Response body:

A string containing the id of the newly created spell.



No environment

Double Column ▼

Public

Example Request

Basic Example

```
curl --location --request POST 'http://localhost:8090/api/admin/addspell' \  
--data-raw '{  
  "Name": "newspell",  
  "Level": 4,  
  "School": "Evocation",  
  "Components": "V,S,M",  
  "Damage": "2d6+2",  
  "Desc": "Hello there"  
}'
```

Example Response

200 - OK

7adc39f0-7298-11ea-8b64-abae503c40a3

POST <http://localhost:8090/api/admin/delspell><http://localhost:8090/api/admin/delspell?id=7adc39f0-7298-11ea-8b64-abae503c40a3>

Remove a spell from the list of available spells

Request

The id of the spell to delete is sent as a query parameter under the name "id".

Response codes

- 200 - Request completed successfully, spell no longer on disk.
- 400 - An id was not given with the request, or the id given does not correspond to a spell.
- 401 - Authentication failed
- 500 - The server encountered an error

PARAMS

id

7adc39f0-7298-11ea-8b64-abae503c40a3



No environment

Double Column ▼

Public

Example Request

Basic Example

```
curl --location --request POST 'http://localhost:8090/api/admin/delspell?id=7adc39f0-7298'
--data-raw ''
```

GET http://localhost:8090/api/spells

Search and fetch from the list of available spells

Request queryparams

The parameters for the request are given in the querystring, with the following available:

Parameter	Description
id	Limits the search to only the spell with the given id
ids	Limits the search to only spells with the ids given (ids should be given in array form). Overridden by "id"
name	Limits the search to spells whose name contains the string given
level	Limits the search to spells of the given level
school	Limits the search to spells of the given school of magic
damage	Limits the search to spells with the damage given

Any combination of these can be used together and the route will return any spells matching all of the given parameters. Sending the request with no parameters will return all available spells.

Response codes

- 200 - Request completed successfully
- 401 - Authentication failed
- 500 - The server encountered an error

Response body

An array containing objects with the following properties:



No environment

Double Column ▼

Public

Property	Type	Description
Id	String	The unique Id of the spell
Name	String	The name of the spell
Level	Number	The level of the spell
School	String	The school of magic the spell belongs to
Components	String	The components required to cast the spell
Damage	String	The damage done by the spell
Desc	String	A description of the spell's effects

Example Request

Basic Example ▼

```
curl --location --request GET 'http://localhost:8090/api/spells' \
--data-raw ''
```

Example Response

200 - OK

```
[
  {
    "Id": "901a3170-5356-11ea-83b6-272f12ebd2f4",
    "Name": "Test Spell 1",
    "Level": 3,
    "School": "Transmutation",
    "Components": [
      "V",
      "S",
      "M"
```

[View More](#)**POST** http://localhost:8090/api/newchar

```
http://localhost:8090/api/newchar
```

Create a new character attached to the user (given in authentication)

≡

No environment

Double Column ▼

Public

Request body

JSON object with the following properties:

Property	Type	Description
Name	String	The name of the character
Level	Number	The character's level
Class	String	The class of the character
Race	String	The race of the character

Response codes

- 200 - Request completed succesfully, character saved to disk
- 400 - Not all of the required information was given in the body
- 401 - Authentication failed
- 500 - The server encountered an error

Response body

- A string containing the id of the newly created character.

Example Request

Basic Example

```
curl --location --request POST 'http://localhost:8090/api/newchar' \  
--data-raw '{  
  "Name": "char1",  
  "Level": 3,  
  "Class": "Fighter",  
  "Race": "Human"  
}'
```

Example Response

200 - OK

70a0cdc0-729d-11ea-8b64-abae503c40a3



No environment

Double Column ▼

Public

POST http://localhost:8090/api/delchar

http://localhost:8090/api/delchar

Delete a character

Request body

The request body should be an object with the following properties:

Property	Type	Description
Id	String	The id of the character to be deleted

Response codes

200 - Request completed successfully, character no longer exists on disk. 400 - The id given does not correspond to a character 401 - Authentication failed 403 - The id given corresponds to a character not associated with the authenticated user 500 - The server encountered an error

Example Request

Basic Example

```
curl --location --request POST 'http://localhost:8090/api/delchar' \  
--data-raw '{  
  "Id": "70a0cdc0-729d-11ea-8b64-abae503c40a3"  
}'
```

POST http://localhost:8090/api/editchar

http://localhost:8090/api/editchar

Make a change to an existing character

Request body

JSON object with any combination that includes "Id" of the following properties:

≡

No environment		Double Column ▼	Public
Property	Type	Description	
Id	String	The id of the character to be edited	
Name	String	The new name of the character	
Level	Number	The new level of the character	
Class	String	The new class of the character	
Race	String	The new race of the character	
Spells	Array	An array containing two sub-arrays. The first should contain the id of all spells to be added to the character (if any). The second should contain the id of all spells to be removed from the character (if any).	

Response codes

- 200 - Request completed successfully, changed saved to disk.
- 400 - An id was not provided, or the id provided does not correspond to a character
- 401 - Authentication failed
- 403 - The id corresponds to a character associated with a different user
- 500 - The server encountered an error

Example Request

Basic Example ▼

```
curl --location --request POST 'http://localhost:8090/api/editchar' \  
--data-raw '{  
  "Id": "70a0cdc0-729d-11ea-8b64-abae503c40a3",  
  "Name": "newname",  
  "Level": 4  
'
```

GET http://localhost:8090/api/characters

http://localhost:8090/api/characters

Search and fetch from the user's characters

Request queryparams



No environment

Double Column ▼

Public

The request can be sent with any (or none) of the following parameters in a querystring. If no parameters are given, the request will return all characters associated with the user.

Parameter	Description
id	The id of the character to fetch
name	The name of the character to fetch. If this is used, the request will return any characters associated with the user whose name contains the string given.

Response codes

- 200 - Request completed successfully
- 401 - Authentication failed
- 403 - The id given corresponds to a character associated with another user
- 500 - The server encountered an error

Response body

The body will be an array containing character objects with the following properties

Property	Type	Description
Id	String	The id of the character
Name	String	The name of the character
Level	Number	The level of the character
Class	String	The class of the character
Race	String	The race of the character
Spells	Array	An array containing the id of all spells known by the character

Example Request

Basic Example(no querystring) ▼

 curl --location --request GET 'http://localhost:8090/api/characters' \

Public

Example Response

200 - OK

```
[
  {
    "Id": "bf87c250-5735-11ea-b538-0b01d317a8d3",
    "Name": "John Smith",
    "Level": "4",
    "Class": "Artificer",
    "Race": "Human",
    "Spells": [
      "901a3170-5356-11ea-83b6-272f12eb0274",
      "31f0hef0-5358-11ea-a6ch-dd0e7515cfff"
```

[View More](#)

POST http://localhost:8090/newuser

Create a new user account

Request body

JSON object with the following parameters

Parameter	Type	Description
Name	String	The username of the new account
Password	String	The password of the new account

Response codes

- 200 - Request completed successfully, user saved to disk
- 403 - Username is already taken
- 500 - The server encountered an error



Example Request

No environment

Double Column ▼

Public

Default ▼

```
curl --location --request POST 'http://localhost:8090/newuser' \  
--data-raw ''
```