

# Project - URL SHORTENER (SHTR)

A URL shortener is an online application that converts a regular URL (the web address that starts with http://) into its condensed format.

The user only has to copy the full URL of a website and paste it into the URL shortening tool to come up with an abbreviated version that is around 10 to 20 characters long.

Regular URL - <https://dctacademy.com/2018/05/become-full-stack-javascript-developer/>

Shortened URL - <http://bit.ly/2BPnAvF>

Why Are URLs Shortened?

1. Twitter posts have character limits.

A tweet can only go as long as 140 characters, so sharing a full URL with your followers can be difficult.

2. Shortened URLs have more aesthetic appeal.

3. Some URL shorteners come with traffic monitors.

It allows you to track the number of clicks the shortened link got over a certain period.

## Models

1. **URL** - An url will consist of the following fields

- a. Id
- b. Title
- c. Original URL
- d. Tags
- e. Hashed URL
- f. Created at

### Note -

- all fields are required
- For validating the original url input use the validator npm package
- Hashed URL needs to be generated before saving the url into the database. Use the shorthash package to do so.
- Tags field will be storing an array of strings.

## Sample JSON data to be sent in POSTMAN

```
{
  "title" : "How to be become a full stack developer",
  "original_url" :
  "https://dctacademy.com/2018/05/become-full-stack-javascript-dev
  eloper/",
  "tags": ["javascript", "mern", "full stack developer"]
}
```

## Sample JSON Response Object

```
{
  "_id": "5b7d700183f3c5ee665bb07b",
  "title": "How to be become a full stack developer",
  "originalUrl":
  "https://dctacademy.com/2018/05/become-full-stack-javascript-dev
  eloper/",
  "tags": [ "javascript","mern","full stack developer" ],
  "hashedUrl": "vpcA6",
  "createdAt": "2018-06-28T03:53:24.065"
}
```

## Routes

Each resource / model must have its own router level middleware

You are supposed to use the rest routing for performing the CRUD Operations on the url resource.

**Tip -** Which HTTP method should we use?

When constructing a REST API each HTTP method corresponds to an action against a resource served by the API.

**GET** — retrieve a particular resource's object or list all objects

**POST** — create a new resource's object

**PATCH** — make a partial update to a particular resource's object

**PUT** — completely overwrite a particular resource's object

**DELETE** — remove a particular resource's object

## API endpoints for URLs

HTTP Method	URI	Actions
GET	/urls	Retrieve all urls
GET	/urls/:id	Retrieve a url by its ID
POST	/urls	Create a new URL
PUT	/urls/:id	Update properties of a url by its ID
DELETE	/urls/:id	Delete a URL by its ID

## Other API endpoints

HTTP Method	URI	Actions
GET	/:hash	Finds the url with the hash value and redirect the user the respective page
GET	/urls/tags/:name	Find and return all the urls that meets the specific tag  - see hint below
GET	/urls/tags?names=tag1,tag2	Find and return all the urls that meets the provided tags  - see hint below

**Hint** - Find document with array that contains a specific value, you can either use `Url.find({ tags: 'tag1' })` if it is single tag or for multiple tags `Url.find({tags: {"$in": ["tag1", "tag2"]}})`

**Note** - `$in` is like OR and `$all` is like AND