# Motivation

1. To build a personal project to demonstrate designing of REST APIs
2. Explore and utilize Flask web application framework
3. Showcase a design where expanding the functionality needs low effort (versioning)
4. Showcase a design where in case of failed request, appropriate error response is returned such that the user understands if there’s a mistake in API usage or in reading of documentation

# Use cases

APIs for to implement a *to-do* app where tasks can be added, removed, updated and deleted

1. **Create a new task**  
   *POST interface* should create a new task and return a unique identifier *<task ID>*
2. **Query details of all or any one task** *GET interfaces* should return list of all tasks in the system or if a *<task ID>* is entered, return details of that one task
3. **Update details of a certain task**For the specified *<task ID>*, *PUT interface* should edit any field of the task (apart from the unique identifier *<task ID>*)
4. **Delete a task**  
   For the specified *<task ID>*, *DELETE interface* should delete the task

# Response codes

* 200 – operation was successful
* 400 – bad requests
  + bad query parameters
  + bad headers
  + bad body
* 401 – authentication failed
* 404 – specified end point or resource (URI) cannot be located
* 405 – method not supported

Deployment

1. Create a virtual environment and activate it

$ pip install venv

$ venv venv

$ source venv/bin/activate

1. Install dependencies

(venv) $ pip install flask

(venv) $ pip install flask\_restful

(venv) $ pip install flask\_httpauth

1. Run the application

(venv) $ python3 app.py

# Examples

1. **Create a new task**

curl --location --request POST 'http://127.0.0.1:5000/todo/api/v1.0/tasks' \

--header 'Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQ=' \

--header 'Content-Type: application/json' \

--data-raw '{

"title": "Remind to call the bank",

"done": false,

"description": "Need to follow up on credit card issue"

}'

Response:

{

"task": {

"title": "Remind to call the bank",

"description": "Need to follow up on credit card issue",

"done": **false**,

"uri": "/todo/api/v1.0/tasks/1"

}

}

1. **Query all tasks**

curl --location --request GET 'http://127.0.0.1:5000/todo/api/v1.0/tasks' \

--header 'Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQ='

Response:

{

"tasks": [

{

"title": "Remind to call the bank",

"description": "Need to follow up on credit card issue",

"done": **false**,

"uri": "/todo/api/v1.0/tasks/1"

},

{

"title": "Call Anne",

"description": "Ask her about potting soil for the garden",

"done": **false**,

"uri": "/todo/api/v1.0/tasks/2"

}

]

}

1. **Query specific task**

curl --location --request GET 'http://127.0.0.1:5000/todo/api/v1.0/tasks/1' \

--header 'Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQ='

Response:

{

"task": {

"title": "Remind to call the bank",

"description": "Need to follow up on credit card issue",

"done": **false**,

"uri": "/todo/api/v1.0/tasks/1"

}

}

1. **Update specific task**curl --location --request PUT 'http://127.0.0.1:5000/todo/api/v1.0/tasks/2' \

--header 'Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQ=' \

--header 'Content-Type: application/json' \

--data-raw '{

"description": "Pick up pasta sauce"

}'

Response:

{

"task": {

"title": "Call Anne",

"description": "Pick up pasta sauce",

"done": **false**,

"uri": "/todo/api/v1.0/tasks/2"

}

}

1. **Delete specific task**

curl --location --request DELETE 'http://127.0.0.1:5000/todo/api/v1.0/tasks/2' \

--header 'Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQ=' \

--data-raw ''

Response:

{

"result": **true**

}

# Credits

Miguel Grinberg's Flask REST API Tutorial <https://www.linkedin.com/in/miguelgrinberg/>

# Ideas for next version

1. Implement token-based OAuth authentication
2. Versioning – Maintain 2 versions of API in the same design
3. Data storage in a database
   1. GET with filter - Include complex get functions where specialized database querying is involved