REST API

/checkout

GET

/api/checkauth GET request will return an integer in JSON format with level of authorisation

Args:
- line_id=<int> : line which you want to check the password against
- password=<string> : user or admin password

Returns:
- JSONObject: {"result": <int>}

0: No auth
1: view traps, add catches
2: edit line, add/edit traps

Example:
- /api/checkauth?line_id=1&password=example

/line

GET

/api/line GET request will return lines in JSON format back to the user

Args:

- line_id=<int> : filters results by id given (Optional)
- name=<string> : filters results by searching for string input as substring (Optional)

Returns:

- JSONObject: {"result": [Line...]}
- Line Object: {"id": <int>,
  "name": <string>,
  "animal1: <int>,
  "animal2: <int>,
  "animal3: <int>}

Example:

```
- /api/line             ->  [{"id":1, "name":"Foo"}, {"id":2, "name":"Bar"}]
- /api/line?line_id=1   ->  [{"id":1, "name":"Foo"}]
- /api/line?name=Ba     ->  [{"id":2, "name":"Bar"}]
```

/trap

GET

/api/trap GET request will return traps in JSON format back to the user

Args (Optional):
- line_id=<int> : filters results by line id given
- trap_id=<int> : filters results by trap id given

Returns:
```
- JSONObject: {'result': [Trap...]}
- Trap Object: {'id': <int>,
              'latitude': <float>,
              'longitude': <float>,
              'lineId': <int>,
              'number': <int>,
              'side': <boolean>,
              'broken': <boolean>,
              'moved': <boolean>}
```

Example:
```
  - /api/trap?line_id=1   ->  [{TrapObject}, {TrapObject}, {TrapObject}]
  - /api/trap?trap_id=1   ->  [{TrapObject}]
```

PUT

/api/trap PUT request will write or edit trap objects into the database

Content-type: application/json
Payload:
```
  - JSONObject: {"lineId": <int>
                "password": <string>,
                "traps": [Trap...]}
  - Trap Object: {'id': <int>, (Optional: if given, overrides set in database. If excluded, creates new line)
                'latitude': <float>,
                'longitude': <float>,
                'lineId': <int>,
                'number': <int>,
                'side': <boolean>,
                'broken': <boolean>, (Optional on editing set, don't include if creating new trap)
                'moved': <boolean> (Optional on editing set, don't include if creating new trap)}
```

Example payload:
```
  {"line_id":1, "password":"1234", "traps":[{TrapObject}, {TrapObject}, {TrapObject}]}
```

Exceptions:
  - 403: Could not validate password

- 400: Non iterable datatype passed with traps
- 403: Could not validate admin password
- 400: Could not enter trap into database (Missing key/failure to write)

DELETE

/trap DELETE request will delete multiple traps in the database belonging to one line

Content-type: application/json
Payload:

```
- JSONObject: {"lineId": <int>,
              "password": <string>,
              "traps": [<int>...]}
```

Exceptions:

- 400: Trap belongs to a different line
- 401: Could not validate user

/catch

GET

/api/catch GET request will return catches in JSON format back to the user

Args:
- line_id=<int> : filters results by line id given by relationship query to a trap (Optional)
- trap_id=<int> : filters results by trap id given (Optional)

Returned:
```
  JSONObject: {'result': [catch...]}
  Catch Object: {'id': <int>,
              'trapId': <int>,
              'trapNumber': <int>,
              'animalId': <int>,
              'time': <long int>}
```

Example:
- /api/catch?line_id=1   -> [{CatchObject w/ trapNumber -> Trap.line_id == 1}, ...]
- /api/catch?trap_id=1   -> [{CatchObject w/ id == 1}]

Exceptions:
- 404: No argument given

PUT

/catch PUT request will write or edit catch objects into the database

Content-type: application/json
Payload:
```
  JSONObject:   {"lineId": <int>
              "password": <string>,
              "catches": [Catch...]}
  Catch Object: {'id': <int>, (Optional: if given, overrides set in database. If excluded, creates new line)
              'trapId': <int>,
              'animalId': <int>,
              'time': <long int> (Ignored when overriding set in database)}
```

Example payload:
```
  {"line_id":1, "password":"1234", "catches":[{CatchObject}, {CatchObject}, {CatchObject}]}
```

Exceptions:
- 403: Could not validate password
- 400: Non iterable datatype passed with catches
- 400: Could not enter catch into database (Missing key/failure to write)

DELETE

/catch DELETE request will delete multiple catches in the database

Content-type: application/json
Payload:

- JSONObject: {"lineId": <int>,
  "password": <string>,
  "catches": [<int>...]}

Exceptions:

- 400: Catch belongs to different line
- 401: Could not validate user

/animal

GET

/api/animal GET request will return animals in JSON format back to the user

Args:
- name=<string> : filters results by searching for string input as substring (Optional)

Returned:
```
JSONObject: {"result": [animal...]}
Animal Object: {"id": <int>,
                "name": <string>}
```

Example:
```
- /api/animal          ->  [{"id": "1", "name": "Foo"}, {"id": "2", "name": "Bar"}]
- /api/animal?name=Ba   ->  [{"id": "2", "name": "Bar"}]
```

PUT

/api/animal PUT request will write or edit catch objects into the database

Content-type: application/json
Payload:
```
JSONObject:   {"lineId": <int>
               "password": <string>,
               "animals": [<string>...]}
```

Exceptions:
- 403: Could not validate password
- 400: Non iterable datatype passed with animals
- 400: Could not enter catch into database (Missing key/failure to write)