Mini City API by Alex Hwang

Authentication
Will be implemented in near future. Will use a API Key.

Root URL
TBA

API Endpoints
/users
- **GET**: Access all users' data
- JSON returned: Array of objects for each users

```
user = {
  'dbid': user's database ID (number),
  'first_name': user's first name (string),
  'middle_name': user's middle name (string),
  'last_name': user's last name (string),
  'age': user's age (number),
   'gender': user's gender (string: 'f' female, 'm' male, 'o' other),
  'race': user's race (string: '1' American Indian or Alaska Native, '2' Asian or Asian
American, '3' Black or African American, '4' Native Hawaiian or Other Pacific Islander, '5' White,
'6' Other),
  'dob': user's date of birth (string: "Fri, 06 Nov 1987 00:00:00 GMT"),
  'photo': user's profile photo (string: image file base64 encoded),
  'employment': availability of user's employment data (number: ID of the employment data),
  'hasID': whether user has an ID (number: '1' yes, '0' no),
  'pob': availability of user's place of birth data (number: ID of the place of birth data),
  'parents': availability of user's parents' name data (number: ID of the parents' name data),
  'device': user's NFC tag ID (number)
}
```

- **POST**: create a user
- JSON returned: Array of the user data created

/users/<user>
- <user> : A parameter specifying a user data being accessed; currently only NFC tag ID associated the user is supported
- **GET**: Access a user's data specified by <user> parameter
- JSON returned: Object (with possible nested objects)

```
 user = {
    'dbid': user's database ID (number),
    'name': { 'first': user's first name (string), 'middle': middle name (string), 'last': last name
(string) } (object or string "" if null),
user's first name (string),
    'age': user's age (number),
```

‘gender’: user’s gender (string: ‘f’ female, ‘m’ male, ‘o’ other),
‘race’: user’s race (string: ‘1’ American Indian or Alaska Native, ‘2’ Asian or Asian American, ‘3’ Black or African American, ‘4’ Native Hawaiian or Other Pacific Islander, ‘5’ White, ‘6’ Other),
‘dob’: user’s date of birth (string: “Fri, 06 Nov 1987 00:00:00 GMT”),
‘photo’: user’s profile photo (string: image file base64 encoded),
‘employment’: availability of user’s employment data (number: ID of the employment data),
‘hasID’: whether user has an ID (number: ‘1’ yes, ‘0’ no),
‘pob’: { hospital: (string), city: (string), county: (string), state: (string) } (object or string “” if null),
‘parents’: { father: (string), mother: (string) } (object of string “” if null),
‘nfc_tag_id’: user’s NFC tag ID (number),
‘container_id’: ID of container the NFC tag was registered at (number),
‘registered_at’: Date and time of registration (string)
}

/services/<service>

- <service>: A parameter specifying an event data being accessed (’0’ for Physical Nourishment, ‘1’ for Wellness)
- **GET**: Access a service data specified by <service> parameter
- JSON returned: Array of objects for each events

```
event = {
   ‘contact’: phone number (string),
   ‘container’: Database ID of container the event is held (number),
   ‘datetime’: Date and time of the event (string),
   ‘dbid’: Database ID of the event created (number),
   ‘name’: Name of the event (string),
   ‘location’: { address: (string), city: (string), state: (string), zipcode: (string) } (object)
}
```

- **POST**: create an event for a service specified by <service> parameter
- JSON returned: Object of the event data created

/containers/<container>

- <container>: A parameter specifying a container data being accessed (Database ID of container)
- **GET**: Access a container data specified by <container> parameter (Database ID of container)
- JSON returned: Object

```
container = {
   ‘container_id’: Database ID of container (number),
   ‘container_name’: Name of container (string),
   ‘zipcode’: Zip code of container (string),
   ‘registered_device’: Number of NFC tags registered at the container (number),
   ‘services’: {
     service1: Events for Service 1 head at the container (array)
     service2: Events for Service 2 head at the container (array)
   } (object)
}
```

/log?

- **GET**: Creates a log for a user for an event at a container

- Query parameter ‘user’: NFC Tag ID, required
- Query parameter ‘svc’: Service ID, required
- Query parameter ‘cont’: Container ID, optional
- JSON returned: Object of the log created

```
        log = {
            ‘service_id’: Service ID the log belongs to (number),
            ‘nfc_tag_id’: NFC Tag ID scanned for the log (string),
            ‘user’: { first_name: (string), middle_name: (string), last_name: (string) } (object),
            ‘container_id’: Database ID of the container the log belongs to (number),
            ‘timestamp’: Date and time the log was created (?)
        }
```

/delete/<category>/<id>*[?]*

- **GET**: Delete a user or an event specified by <category> and <id> parameters
- <category>: A parameter specifying the category of data to be deleted (‘user’, ‘service’)
- <id>: A parameter specifying NFC Tag ID of the user to be deleted (<category> ‘user’) or service ID of the service whose event to be deleted (<category> ‘service’). In case of ‘service’, a query parameters ‘dbid’ is required to delete a specific event
- */delete/user/<NFC Tag ID>*
- */delete/service/<Service ID>?dbid=‘’*
- JSON returned: String (a success message with the NFC Tag ID deleted or database ID of the event deleted, or an error message)