ResLife API Declaration Doc

Table of Contence

        - getResidentNotes
        - getResidentNote
        - getFormTemplateList
        - getFormTemplate
        - getFormData
        - getRequestedForms
        - getFormStatus
        - getResidenceHalls
        - getResidencehallByName
        - getZones
        - getZonesByhall
        - getZone
        - getNodesInZone
        - getNode
        - getIssues
        - getIssuesByZone
        - getIssue
        - getIssueComments
        - getIssueComment
        - getRoundTemplates
        - getRoundTemplate
        - getRound
        - getRoundStatus
    - Post Requests
        - createResidentNote
        - editResidentNote
        - addResident
        - createForm
        - editFormData
        - createFormTemplate
        - editFromTemplate
        - removeFormTemplate
        - createIssue
        - editIssue
        - createIssueComment
        - editIssueComment
        - createRoundData
        - editRoundData
    - Search
        - searchResidentByArea
        - searchResidentByHall

- searchResidentByName
- searchResidentFull

Genral

Format

all endpoints are toplevel, ie they are of the form `/api/endpoint` never `/api/section/endpoint`

For GET requests mandatory peramitors are passed in the url directly, optional paramiters are pass in the query string

IE.

`/api/endpoint/$param1/$param2/?paramname=$optionalparam1&paramname=$optionalparam2`

For POST requests all paramitors passed in the form fields

IE.

`/api/endpoint/`

```
FORM DATA

param1: value
param2: value
```

segments starting with `$` denote a spot that is replace with the needed value for that field

Returns

The API will always return a responce in the same format

```
{
    "err": {},
    "result": {}
}
```

where `err` may or may not be empty. If it is NOT empty then there was an error processing the request. `result` will contain fields and values relavent to the request

Errors

The `err` object will have the folowing members:

Member	Example	Description
`msg`	`{"msg": "Error Message"}`	A string describing the error
`code`	`{"code": 403}`	A interger ID if it is a known error condition, otherwise 0. In genral this will be follow the HTTP responce code
`source`	`{"source": "Auth"}`	A string indicating the area of the API the error came from, can help trace errors

Examples

An error

```
{
    "err": {
        "msg": "Auth token is invalid or expired",
        "code": 403,
        "source": "getResidentByName"
    },
    "result": {}
}
```

```
{
    "err": {},
    "result": {
        "user" : {
            "id": 900999999,
            "first": "John",
            "last": "Doe",
            "dob": "1995-05-31",
            "hall": "west",
            "room": 105
        }
    }
}
```

Models

Resident

```
class Resident:
    id = #
    dob = ISO Date string
    first_name = ""
```

```
    last_name = ""
    hall = hallid#
    room = roomid#
    email = ""
    photo = "" #URL to static storage (binary blob is bad idea)
    emergency_contact = ""
    emergency_contact_relationship = ""
    emergency_contact_phone = ""
    home_addr = ""
    car_plate = ""
    car_info = ""
```

do we call student or resident? car plate / make ? emergency contact ? home addr?

Residence Hall

```
class ResidenceHall:
    id = #
    name = ""
    rlc = userid#
    rooms = [#,...] # access through back reation
    zones = [zoneid, zoneid, ...] # list of top level zones
```

TODO do we need to store a layout of the room ?

Rooms and Condition Reports

```
class Room:
    id = #
    hall = hallid#
    max_residents = # # maximum number of residents room can have
    name = "" # the room # IE. (102, 103c, Bld. C 203c) (MSA and others might need inc
    condition_reports = [] # access via back relation set

    def is_full:
        # return via it's back relation if the room is full

    def free:
        # return via back relation and max_residents the number of free slots

    def valid:
        # if there are free slots or the room is full the room is valid, if the room i

    def empty:
        # return via back relation if no residents are assigned to the room
```

```
class ConditionReport:
    id = #
    room = roomid#
    date = ISO
    resident = resid#
    images = [imageid, imageid, ...] #this will be a real array not back relation
```

A room object links residents to rooms, and condition reports to rooms it's implimentation provides utilities to validate resident placment in rooms

App User

class for users (prostaff and RA's)

```
class User:
    id = #
    username = ""
    first_name = ""
    last_name = ""
    hashpass =  ""
    access = "" # a 2 charater access class
    resident = resid# or NULL # if the app user is allso a resident (RLC)
```

Auth Token

```
class AuthToken:
    token = unique Base64String of fixed length #
    user = id#
    issued = ISO Date String
    expires = ISO Date String
    revoked = bool

    def generate(username, issued):
        mac = hashlib.sha256(username + issued)
        token = hashlib.sha256(config.secret_key + mac)
        digest = base64.b64encode(token.digest())
        return digest
```

Authentication security is dependent on an attacker never being able to forge an auth token. a key part of this is the server holding a security key. the key should be saved into the configuration, contain at least 128 bits of entropy and be generated from a secure PRNG (e.g. /dev/urandom).

the token the user gets to authenticate with is valid for a limited time and can be manulay revoked, addtionaly, the tokens are stored seperately form teh user object so that the number of valid tokens can be limited but not 1 (allowing a set number of divces that is more than one to log in at once, mobile

app on phone and ipad?, web access?)

The token will be deliverd as a base64 encoded string

Note

```
class Note:
    id = #
    top = bool # is this note a comment?
    resident = resid#
    author = userID#
    created = ISO Date String
    edited = ISO Date String
    content = ""
    comments = [noteid#, noteid#, ...]
    access_level = "" # defaults to authors access level
    access = [userid#, userid#, ...]
    images = [imageid, imageid, ...] #this will be a real array not back relation
```

FormTemplate

As we are allowing editing of form templates we should probably be storeing these older templates less we end up with old form data that doesn't match up with the current template

```
class FormTemplate:
    id = #
    name = ""
    author = userid#
    created = ISO Date String
    edited = ISO Date String
    editedby = userid#
    templatedata = dataid#
    versions = [dataid#, dataid#, ...]
```

template data is treated as copy on write. each saved edit creates a new version.

```
class FromTemplateData:
    id = #
    created = ISO Date String
    author = userid#
    template = "" # contents to be either a JSON or XML layout tempalte
    pairs = "" # JSON pairs of names to types
```

FormData

Forms store a list of keyvalue pairs and a link to the template data they were filed with.

```
class FormData:
    id = #
    created = ISO Date String
    author = userid#
    edited = ISO Date String
    editedby = userid#
    template = templatedataid#
    status = #
    data = "" # json keyvalue pairs
```

What different states do we need for form status?

BuildingZone

the area's an RA cna be responcable for proposed sored similar to notes

```
class BuildingZone
    id = #
    name = ""
    vectors = [
        [[0,1], [1,2], [0,3]], # a start, middle and end point, draw a curve
        [[2,1], [2,3]] # a start and end point, draw a strait line
        [start[,middle],end], ... # a vector is a list of 2 or 3 points.
    ] # list of vectors to draw.
    parent = Null or areaid#
    children = [zoneid#, zoneid# zoneid#, ...] # Access through back related set
```

vectors is a list of vectors to help draw the zone

BuildingZoneNode

a node to interact with on rounds

```
class BuildingZoneNode:
    id = #
    name = ""
    description = ""
     = zoneid#
    location = [0,1] # array of coredinates
    check = bool # flag to indicate a node to check even if there is no issue (a permi
```

BuildingZoneLabel

a label to link to other areas

```
class BuildingZoneLabel
    id = #
    name = ""
    zone = zoneid#
    link = zoneid#
    location = [0,1] # array of coredinates
```

RoundArea

The top level round area,

store a list of toplevel zones included in the round then we can get the child areas and the area nodes to check from there.

```
class RoundArea:
    id = #
    name = ""
    areas = [zoneid#, zoneid#, ...] #list of toplevel zoneids

    def get_check_nodes:
        #return list of nodes with check flag. (for determining round data compleatene

    def get_issue_nodes(resolved=false, limit=None):
        #return list of nodes without check flag that have current issues
        #include resolved issues if "resolved" is true
        #limit search to issues newer than "limit" date
```

Round Data

round data, like form data contains only key value pairs. in this case the keys are node ids and the value is either 0 (for no concerns) or a valid issue id. If there is no key for a node that is included in an area, then the node has not been checked yet and the round is incompleate

```
class RoundData:
    id = #
    created = ISO Date String
    author = userid#
    edited = ISO Date String
    editedby = userid#
    round = roundareaid#
    status = #
    data = "" # json keyvalue pairs
```

the round template id is probably needed to verify that all area id's that need to be checked are present in the data.

Issue

an issue that may be filed on a round

```
class Issue:
    id = #
    name = ""
    author = userid#
    created = ISO Date String
    edited = ISO Date String
    editedby = userid#
    content = ""
    node = areanodeid#
    status = #
    comments = [issuecommentid#, ...] # access through back relation set
    images = [imageid, imageid, ...] #this will be a real array not back relation
```

does this need access controll too?

Issue Comment

a comment made on an issue

```
class IssueComment:
    id = #
    issue = issueid#
    author = userid#
    created = ISO Date String
    edited = ISO Date String
    editedby = userid#
    content = ""
    comments = [issuecommentid#, ...]
    access = [userid#, userid#, ...]
    images = [imageid, imageid, ...] #this will be a real array not back relation
```

Image

```
class Image:
    id = #
    url = ""
```

Log

a action taken by a users. used to log edit and edit actions to things like notes, issues, and forms

```
class LogAction
    use = userid#
    action = "" # quick action summery, 2-4 words
    timestamp = ISO Date String
    detail = "" # description, 1-2 sentances
    type = # # the object type affected
    id = # # id of object affected
```

Functions

Athentication

INFO

Authentication is done via token based authorisation

The Aoth token is bassed via the HTTP Authorisation Header.

the header should take the form

```
Authorization: Token 4SMPkrseLzXWYxoaDrTrQOzIwmb3IJCrSc40RvKqkpM=
```

so long as this header is passed and the token is valid the request should authororize

auth

`/api/auth/`

creates a auth token for user

POST

Field	Value
user	username
pass	password

Returns

```
{
```

```
    "err":{},
    "result": {
        "token": "BASE64TOKEN(length to be determined)",
        "expires": "2017-08-30T14:32:03",
    }
}
```

Field	Value
token	Base64 string that is unique to this login atempt
expires	ISO Date and Time string when this token will no longer be valid

expire

`/api/expire/`

forcefuly expires an authtoken

POST

Field	Value
token	authtoken

Returns

```
{
    "err": {},
    "result": {}
}
```

check HTTP status code for success

GET Requests

getUser

`/api/getuser/$userid/?round=false&depth=0`

gets a user (RA or staff)

GET

Field	Value
userid	the user id
round	bool, get round data

depth	int, nest round areas to depth

Returns


```
{
    "err":{},
    "result": {
        "user": {
            "id": 900999999,
            "first": "John",
            "last": "Doe",
            "access": 2,
            "resident": {
                "id": 900999999,
                "first": "John",
                "last": "Doe",
                "dob": "1995-05-31",
                "hall": "west",
                "room": 105
            },
            "round": {
                "id": 8,
                "name": "West Hall",
                "areas": [{
                    "id": 74,
                    "name": "west hall hallway",
                    "parrent": 73,
                    "children": [
                        {"id": 76}
                    ]
                }, "..."]
            },


        }
    }
}
```


Field	Value
resident	resident object

getUserByName

`/api/getuserbyname/$firstname/$lastname/`

gets a user (RA or staff)

GET

Field	Value
firstname	users's First name
lastname	users's last name

Returns

```
{
    "err":{},
    "result": {
        "user": {
            "id": 900999999,
            "first": "John",
            "last": "Doe",
            "access": 2
        }
    }
}
```

Field	Value
resident	resident object

getResidentByName

`/api/getresidentbyname/$firstname/$lastname/`

GET

Field	Value
firstname	Resident's First name
lastname	Resident's last name

Returns

```
{
    "err":{},
    "result": {
        "resident": {
            "id": 900999999,
            "first": "Bob",
            "last": "Boberson",
            "dob": "1995-05-31",
            "hall": "west",
            "room": 105
        }
    }
```

```
}
```

Field	Value
resident	resident object

getResidentByID

`/api/getresidentbyid/$id/`

GET

Field	Value
id	Student ID

Returns

```
{
    "err":{},
    "result": {
        "resident": {
            "id": 900999999,
            "first": "Bob",
            "last": "Boberson",
            "dob": "1995-05-31",
            "hall": "west",
            "room": 105
        }
    }
}
```

Field	Value
resident	resident object

getResidentByRoom

`/api/getresidentbyroom/$hall/$room/`

GET

Field	Value
hall	hall id
room	room #

Returns

```
{
    "err":{},
    "result": {
        "resident": {
            "id": 900999999,
            "first": "Bob",
            "last": "Boberson",
            "dob": "1995-05-31",
            "hall": "west",
            "room": 105
        }
    }
}
```

Field	Value
resident	resident object

getResidentPicByName

`/api/getresidentpicbyname/$firstname/$lastname/`

GET

Field	Value
firstname	Resident"s First name
lastname	Resident"s last name

Returns

```
{
    "err":{},
    "result": {
        "url": "/photos/residents/%studentnamehash%.jpg"
    }
}
```

Field	Value
url	path to user photo

getResidentPicByID

`/api/getresidentpicbyid/$id/`

GET

Field	Value
id	student Id #

Returns

```
{
    "err":{},
    "result": {
        "url": "/photos/residents/%studentnamehash%.jpg"
    }
}
```

Field	Value
url	path to user photo

getResidentNotes

`/api/getresidentnotes/$id/?depth=3`

retreaves the threads of notes visable to user, threads may be nested

GET

Field	Value
id	student Id #
depth	OPTIONAL: limit retreaval depth

Returns

```
{
    "err":{},
    "result": {
        "notes": [
            {
                "created": "2017-08-30T15:32:26",
                "edited": "2017-08-30T15:35:35",
                "author": { "id": 900999999, "first": "John", "last": "Doe"},
                "content": "Lorem ipsum dolor sit amet.",
                "comments": [
                    {
                        "created": "2017-08-30T16:25:07",
                        "edited": "2017-08-30T17:42:43",
                        "author": { "id": 900999999, "first": "John", "last": "Doe"},
                        "content": "Lorem ipsum dolor sit amet.",
                        "comments": ["..."]
```

```
                    },
                    "..."
                ]
            },
            "..."
        ]
    }
}
```

Field	Value
notes	list of top level notes, notes nest their comments

getResidentNote

`/api/getresidentnotes/$noteid/`

retreaves resident note by note id

GET

Field	Value
noteid	note ID #

Returns

the created resident

```
{
    "err":{},
    "result": {
        "note": {
                "created": "2017-08-30T15:32:26",
                "edited": "2017-08-30T15:35:35",
                "author": { "id": 900999999, "first": "John", "last": "Doe"},
                "resident": { "id": 900999999, "first": "Bob", "last": "Boberson"},
                "content": "Lorem ipsum dolor sit amet.",
                "comments": [23458, 21349, 2345, "..."]
        }
    }
}
```

Field	Value
note	the note in question, note comments are listed by id

getFormTemplateList

`/api/getformtemplatelist/`

Gets the list of named templates and their creation and edit times

GET

Field	Value

Returns

```
{
    "err":{},
    "result": {
        "formtemplates": [
            {
                "id": 12348
                "name": "template name",
                "created": "2017-06-05T09:31:26",
                "edited": "2017-08-07T14:03:39",
            },
            "..."
        ]
    }
}
```

Field	Value
formtemplates	list of template objects

getFormTemplate

`/api/getformtemplate/$id/`

Gets the info of anamed template along with the template data of it's latest version

GET

Field	Value
id	id of named template

Returns

```
{
    "err":{},
    "result": {
        "formtemplate": {
                "id": 12348,
```

```
                "name": "template name",
                "author": { "id": 900999999, "first": "Jane", "last": "Doe"},
                "created": "2017-06-05T09:31:26",
                "edited": "2017-08-07T14:03:39",
                "editedby": { "id": 900999999, "first": "John", "last": "Doe"}
        },
        "formtemplatedata": {
            "id": 47290,
            "author": { "id": 900999999, "first": "John", "last": "Doe"},
            "created": "2017-08-07T14:03:39",
            "data": "<Layout>"
            "pairs": [
                {
                    "key": "name",
                    "type": "typename"
                },
                "..."
            ]
        }
    }
}
```

Field	Value
formtemplate	template object
formtemplatedata	template data object

getFormData

`/api/getformdata/`

GET

Field	Value
formid	ID #

Returns

```
{
    "err":{},
    "result": {
        "form": {
            "id": 12348
            "author": { "id": 900999999, "first": "John", "last": "Doe"},
            "created": "2017-06-05T09:31:26",
            "edited": "2017-08-24T10:45:17",
            "editedby": { "id": 900999999, "first": "John", "last": "Doe"},
            "pairs": {
```

```
                "key": "value",
                "key2": "value2"
                "key3": "value3"
            }

        }
    }
}
```

Field	Value
form	form object including data pairs

getRequestedForms

TODO: should we perhaps have a new object formrequest?

`/api/getrequestedforms/`

GET

Field	Value

Returns

```
{
    "err":{},
    "result": {
        "forms": [ "..."]
    }
  }
}
```

Field	Value

getFormStatus

TODO: is this gettign the status of the form or the possible formrequest?

`/api/getFormforms/$id/`

GET

Field	Value

Returns

```
{
    "err":{},
    "result": {
        "forms": [ "..."]
    }
}
```

Field	Value

getResidenceHalls

`/api/getresidencehalls/`

returns a list of all residence halls

GET

Field	Value

Returns

```
{
    "err":{},
    "result": {
        "halls": [
            {
                "id": 2,
                "name": "West Hall",
                "rlc": "Jhon Doe"
            },
            "..."
        ]
    }
}
```

Field	Value
halls	list of halls

getResidencehallByName

`/api/getresidencehallbyname/$hall/`

GET

Field	Value
hall	hall name

Returns

```
{
    "err":{},
    "result": {
        "hall":
        {
            "id": 2,
            "name": "West Hall",
            "rlc": "Jhon Doe"
        }
    }
}
```

Field	Value
hall	residence hall object

getZones

`/api/getzones/`

returns a list of all areas

GET

Field	Value

Returns

```
{
    "err":{},
    "result": {
        "zones": [
            {
                "id": 2340,
                "name": "West Hall Hallway 1",
                "children": [1234,4733,34573,"..."]
            },
            "..."
        ]
    }
}
```

Field	Value
areas	list of areas and their child ids NOT NESTED

getZonesByhall

`/api/getzones/$hall/?nested=true`

returns a list of the toplevel zones in a hall and their nested sub areas

GET

Field	Value
hall	the hall id
nested	OPTIONAL: should the children be nested

Returns

```
{
    "err":{},
    "result": {
        "areas": [
            {
                "id": 2340,
                "name": "West Hall Hallway 1",
                "children": [
                    {
                        "id": 3469,
                        "name": "West Hall Sairwell 1",
                        "children": ["..."]

                    },
                    "..."
                ]
            },
            "..."
        ]
    }
}
```

Field	Value
areas	areas with their chidren

getZone

`/api/getarea/$id/?nested=true`

returns the identified area and it's children

GET

Field	Value
id	the area id
nested	OTIONAL: should the children be nested

Returns

```
{
    "err":{},
    "result": {
        "area": {
            "id": 2340,
            "name": "West Hall Hallway 1",
            "children": [
                {
                    "id": 3469,
                    "name": "West Hall Sairwell 1",
                    "children": ["..."]

                },
                "..."
            ]
        }
    }
}
```

Field	Value
area	the area object

getNodesInZone

`/api/getnodesinzone/$areaid/?check=true&issue=true&getissues=true&nestedissues=true`

GET

Field	Value
areaid	the area id
check	bool are check noeds included
issue	bool are issue nodes included
getissues	are the issues returned with the nodes
nestedissues	are the issues nested into the objects (1 level no comments)

Returns


```
{
    "err":{},
    "result": {
        "nodes": [
            {
                "id": 593,
                "name": "West Hall east fire estinguisher",
                "description": "should be in place and full",
                "location": "0,3",
                "issues": []
            },
            "..."
        ]
    }

}
```


Field	Value
nodes	list of node objects

getNode

`/api/getnode/$nodeid/?getissues=true&nestedissues=true&currentissues=true&resolvedissues=false`

GET

Field	Value
nodeid	node id
getissues	are the issues returned with the nodes
nestedissues	are the issues nested into the objects (1 level no comments)
currentissues	return unresolved issues on the node
resolvedissues	return resolved issues on the node

Returns


```
{
    "err":{},
    "result": {
        "node": {
            "id": 593,
```

```
            "name": "West Hall east fire estinguisher",
            "description": "should be in place and full",
            "location": "0,3",
            "issues": []
        }
    }
}
```


Field	Value
node	node object

getIssues

`/api/getissues/`

get a list of all issues

GET

Field	Value

Returns


```
{
    "err":{},
    "result": {
        "issues": [
            {
                "id": 256,
                "author": { "id": 900999999, "first": "John", "last": "Doe"},
                "created": "2017-08-07T15:42:59",
                "edited ": "",
                "status": 0,
                "editedby": { "id": 900999999, "first": "John", "last": "Doe"},
                "node": 593
            },
            "..."
        ]
    }
}
```


Field	Value
issues	list of issues

getIssuesByZone

`/api/getissuesbyarea/$areaid/`

get a list of issues from noeds in an area

GET

Field	Value
areaid	area id

Returns

```
{
    "err":{},
    "result": {
        "issues": [
            {
                "id": 256,
                "author": { "id": 900999999, "first": "John", "last": "Doe"},
                "created": "2017-08-07T15:42:59",
                "edited ": "",
                "status": 0,
                "editedby": { "id": 900999999, "first": "John", "last": "Doe"},
                "node": 593
            },
            "..."
        ]
    }
}
```

Field	Value
issues	list of issues

getIssue

`/api/getissue/$issueid/?nested=false`

GET

Field	Value
issueid	issue ID
nested	OPTIONAL should issue comments be nested

Returns

```
{
```

```
    "err":{},
    "result": {
        "issue":  {
            "id": 256,
            "author": { "id": 900999999, "first": "John", "last": "Doe"},
            "created": "2017-08-07T15:42:59",
            "edited ": "",
            "editedby": { "id": 900999999, "first": "John", "last": "Doe"},
            "node": 593,
            "status": 0,
            "content": "Lorem ipsum dolor sit amet."
            "comments" [237,239,243,"...."]
        }
    }
}
```

Field	Value
issue	issue object

getIssueComments

`/api/getissuecomments/$issueid/?nested=true`

get list of comments on issue

GET

Field	Value
issueid	issue ID
nested	OPTIONAL should issue comments be nested

Returns

```
{
    "err":{},
    "result": {
        "issuecomments":  [
            {
                "id": 239,
                "author": { "id": 900999999, "first": "John", "last": "Doe"},
                "created": "2017-08-07T15:42:59",
                "edited ": "",
                "editedby": { "id": 900999999, "first": "John", "last": "Doe"},
                "issue": 256,
                "content": "Lorem ipsum dolor sit amet.",
                "comments": [
                    {
```

```
                        "id": 243,
                        "author": { "id": 900999999, "first": "John", "last": "Doe"},
                        "created": "2017-08-07T15:42:59",
                        "edited ": "",
                        "editedby": { "id": 900999999, "first": "John", "last": "Doe"}
                        "issue": 256,
                        "content": "Lorem ipsum dolor sit amet.",
                        "comments" [ "..." ]
                    }
                ]
            },
            "..."
        ]
    }
}
```


Field	Value
issuecomments	list of issue comments

getIssueComment

`/api/getissuecomment/$issuecommentid/?nested=true`

get an issue comment

GET

Field	Value
issuecommentid	issue comment ID
nested	OPTIONAL should issue comments be nested

Returns


```
{
    "err":{},
    "result": {
        "issuecomment":{
                "id": 239,
            "author": { "id": 900999999, "first": "John", "last": "Doe"},
            "created": "2017-08-07T15:42:59",
            "edited ": "",
            "editedby": { "id": 900999999, "first": "John", "last": "Doe"},
            "issue": 256,
            "content": "Lorem ipsum dolor sit amet.",
            "comments" [
                {
```

```
                    "id": 243,
                    "author": { "id": 900999999, "first": "John", "last": "Doe"},
                    "created": "2017-08-07T15:42:59",
                    "edited ": "",
                    "editedby": { "id": 900999999, "first": "John", "last": "Doe"},
                    "issue": 256,
                    "content": "Lorem ipsum dolor sit amet.",
                    "comments" [ "..." ]
                }
            ]
        }
    }
}
```


Field	Value
issuecomments	list of issue comments

getRoundTemplates

`/api/getroundtemplatesbyName/`

get the list of round template

GET

Field	Value

Returns


```
{
    "err":{},
    "result": {
        "roundtemplates": [
            {
                "id": 8,
                "name": "West Hall",
                "areas": [345, 367, "..."]
            },
            "..."
        ]
    }
}
```


Field	Value
roundtemplates	list of round templates

getRoundTemplate

`/api/getroundtemplate/$templateid/`

get a round template id

GET

Field	Value
templateid	template id

Returns

```
{
    "err":{},
    "result": {
        "roundtemplate": {
            "id": 8,
            "name": "West Hall",
            "areas": [345, 367, "..."]
        },
    }

}
```

Field	Value
roundtemplate	round template object

getRound

`/api/getround/$roundid/`

get a round

GET

Field	Value
roundid	round id

Returns

```
{
    "err":{},
    "result": {
        "round": {
            "id": 2893,
```

```
            "created": "2017-08-30T22:32:09",
            "author": { "id": 900999999, "first": "John", "last": "Doe"},
            "edited": "2017-08-30T22:43:41",
            "editedby": { "id": 900999999, "first": "John", "last": "Doe"},
            "template": 8,
            "status": 0,
            "data": {
                "593": 256,
                "597": 0
            }
        }
    }
}
```


Field	Value
round	round object

getRoundStatus

`/api/getroundstatus/$roundid/`

GET

Field	Value
roundid	round id

Returns


```
{
    "err":{},
    "result": {
        "roundstatus": 0
    }
}
```


Field	Value
roundstatus	the status of the round

Post Requests

createResidentNote

`/api/editresidentnote/`

edits a note on resident, optionaly as a comment on note id.

VALIDITY: in the case of a comment check that users can view note they are commenting on. comment note id must ovisouly be on same resident

POST

Field	Value
resid	id of residnet to create note on
noteid	OPTIONAL: edits note as a comment on noteid
note	string note content
access	list of user id's than can view this note

Returns

```
{
    "err":{},
    "result": {}
}
```

check HTTP status code for success

editResidentNote

`/api/editresidentnote/$resid/`

edits a note on resident, optionaly as a comment on note id.

VALIDITY: in the case of a comment check that users can view note they are commenting on. comment note id must ovisouly be on same resident

POST

Field	Value
noteid	id of note data to change
note	string note content
access	list of user id's than can view this note

Returns

```
{
    "err":{},
    "result": {}
}
```

check HTTP status code for success

addResident

`/api/addresident/`

Adds a Resident

POST

Field	Value
studentid	ID #
firstname	resident first name
lastname	resident last name
dob	ISO Date String
hall	hall name
room	room name

Returns

```
{
    "err": {},
    "result": {
        "resident": {
            "id": 900999999,
            "first": "Bob",
            "last": "Boberson",
            "dob": "1995-05-30",
            "hall": "west",
            "room": 103
        }
    }
}
```

Field	Value
resident	resident object

check HTTP status code for success

createForm

`/api/createfrom/`

create a form with a named template

POST

Field	Value
templateid	ID #

Returns

```
{
    "err":{},
    "result": {
        "form": {
                "id": 12348
                "author": "template name",
                "created": "2017-06-05T09:31:26",
        }
    }
}
```

Field	Value
form	created form object

check HTTP status code for success

editFormData

`/api/editformdata/`

POST

Field	Value
formid	ID #
pairs	JSON data pairs

Returns

```
{
    "err":{},
    "result": {
        "form": {
                "id": 12348
                "author": { "id": 900999999, "first": "John", "last": "Doe"},
                "created": "2017-06-05T09:31:26",
                "edited": "2017-08-24T10:45:17",
                "editedby": { "id": 900999999, "first": "John", "last": "Doe"}
    }
```

```
  }
}
```

Field	Value
form	form object

check HTTP status code for success

createFormTemplate

TODO

editFromTemplate

TODO

removeFormTemplate

TODO

createIssue

`/api/createissue/`

POST

Field	Value
nodeid	area node id linked to issue
content	content
access	access list

Returns

```
{
    "err":{},
    "result": {
        "issue":  {
            "id": 256,
            "author": { "id": 900999999, "first": "John", "last": "Doe"},
            "created": "2017-08-07T15:42:59",
            "edited ": "",
            "editedby": { "id": 900999999, "first": "John", "last": "Doe"},
            "node": 593,
            "status": 0,
            "content": "Lorem ipsum dolor sit amet."
```

```
            "comments" [237,239,243,"...."]
        }
    }
}
```

Field	Value
issue	issue object

check HTTP status code for success

editIssue

`/api/editissue/`

POST

Field	Value
issueid	ID #
content	content
access	access list
status	status of issue

Returns

```
{
    "err":{},
    "result": {
        "issue":  {
            "id": 256,
            "author": { "id": 900999999, "first": "John", "last": "Doe"},
            "created": "2017-08-07T15:42:59",
            "edited ": "",
            "editedby": { "id": 900999999, "first": "John", "last": "Doe"},
            "node": 593,
            "status": 0,
            "content": "Lorem ipsum dolor sit amet."
            "comments" [237,239,243,"...."]
        }
    }
}
```

Field	Value
issue	issue object

check HTTP status code for success

createIssueComment

`/api/createissuecomment/`

POST

Field	Value
issueid	ID #
commentid	OPTIONAL post as reply to comment
content	content
access	access list

Returns


```
{
    "err":{},
    "result": {
        "issuecomment":{
                "id": 239,
            "author": { "id": 900999999, "first": "John", "last": "Doe"},
            "created": "2017-08-07T15:42:59",
            "edited ": "",
            "editedby": { "id": 900999999, "first": "John", "last": "Doe"},
            "issue": 256,
            "content": "Lorem ipsum dolor sit amet.",
            "comments": [ 2345, 2345, "..."]
        }
    }
}
```


Field	Value
issuecomment	list of issue comments

check HTTP status code for success

editIssueComment

`/api/editissuecomment/`

POST

Field	Value
commentid	comment to edit

content	content
access	access list

Returns


```
{
    "err":{},
    "result": {
        "issuecomment":{
                "id": 239,
            "author": { "id": 900999999, "first": "John", "last": "Doe"},
            "created": "2017-08-07T15:42:59",
            "edited ": "",
            "editedby": { "id": 900999999, "first": "John", "last": "Doe"},
            "issue": 256,
            "content": "Lorem ipsum dolor sit amet.",
            "comments": [ 2345, 2345, "..."]
        }
    }
}
```


Field	Value
issuecomment	list of issue comments

check HTTP status code for success

createRoundData

POST

Field	Value
templateid	round template id
pairs	JSON data pairs

Returns


```
{
    "err":{},
    "result": {
        "round": {
            "id": 2893,
            "created": "2017-08-30T22:32:09",
            "author": { "id": 900999999, "first": "John", "last": "Doe"},
            "edited": "2017-08-30T22:43:41",
            "editedby": { "id": 900999999, "first": "John", "last": "Doe"},
```

```
            "template": 8,
            "status": 0,
            "data": {
                "593": 256,
                "597": 0
            }
        }
    }
}
```


Field	Value
round	round object

check HTTP status code for success

editRoundData

`/api/editrounddata/$auth=$authtoken`

POST

Field	Value
roundid	ID #
pairs	JSON data pairs

Returns


```
{
    "err":{},
    "result": {
        "round": {
            "id": 2893,
            "created": "2017-08-30T22:32:09",
            "author": { "id": 900999999, "first": "John", "last": "Doe"},
            "edited": "2017-08-30T22:43:41",
            "editedby": { "id": 900999999, "first": "John", "last": "Doe"},
            "template": 8,
            "status": 0,
            "data": {
                "593": 256,
                "597": 0
            }
        }
    }
}
```

Field	Value
round	round object

check HTTP status code for success

Search

searchResidentByArea

`/api/searchresidentbyarea/??first=$first&last=$last`

GET

Field	Value
roundid	ID #
pairs	JSON data pairs

TODO: what is an area in this context? how do we define it?

searchResidentByHall

`/api/searchresidentbyhall/$hallid/??first=$first&last=$last&exact=false`

return a list of possible matches

GET

Field	Value
hallid	ID #
first	OPTIONAL: resident's first name or part
last	OPTIONAL: resident's last name or part
exact	should we list partial matches?

Returns


```
{
    "err":{},
    "result": {
        "residents": [
            {
                "id": 900999999,
                "first": "Bob",
                "last": "Boberson",
                "dob": "1995-05-31",
```

```
                "hall": "west",
                "room": 105
            },
            "..."
        ]
    }
}
```


Field	Value
residents	list of esident object

searchResidentByName

`/api/searchresidentbyname/??first=$first&last=$last&exact=false`

return a list of possible matches

GET

Field	Value
first	OPTIONAL: resident's first name or part
last	OPTIONAL: resident's last name or part
exact	should we list partial matches?

Returns


```
{
    "err":{},
    "result": {
        "residents": [
            {
                "id": 900999999,
                "first": "Bob",
                "last": "Boberson",
                "dob": "1995-05-31",
                "hall": "west",
                "room": 105
            },
            "..."
        ]
    }
}
```


Field	Value
residents	list of esident object

searchResidentFull

TODO: how is this different from searchResidentByName