Frog Pond REST API:

What is Frog Pond?

A social media application. The end user will use a client app (mobile or desktop web browser) to engage in discussion with local people who are interested in the same things. The storage of user data and posts will be distributed, so there will be many servers; the client will connect to the closest one.

What is unique about this app is that users can filter which posts they see by geographical distance and a set of "tags". A tag is just a topic to which a post is related. The obvious advantage of this is that a post can have as many tags as it needs, whereas a reddit post for example is only "in" one subreddit. Users also have the option of posting anonymously, or rather, they have the option of establishing an identity, while anonymity is the default behavior.

To keep with the aesthetic of the app, posts are actually referred to as "croaks", because the idea is that people are frogs hangin' out and croakin' at the local pond. The most basic form of a croak is a text post. It can also be a file (of any type), or a file with some text. And in the future, event croaks will probably be implemented.

The goal of this app is to get the people within a community to talk to each other about matters of import. It is supposed to be an implementation of social media which is up-to-date with current technologies and what is possible. It is to accept the fact that we have dug ourselves into a hole with the internet by making too many "webpages" which have ended up being cluttered with advertisements, meaningless chatter, and irrelevant/inconsistent UI components, and an attempt to dig ourselves out of that hole by implementing the bare-bones infrastructure of what we need to have an effective social media platform which actually serves to better the inhabitants of a community, rather than play on their tendency towards hedonism and get them to give "the guy who made the platform" money. It will not have advertisements.

Database tables and fields:

Here I list the objects and their properties which are stored on the database, along with descriptions where necessary

- croaks
  - int id
  - float x (longitude)
  - float y (latitude)
  - int type (text or file)
  - string content (text content)
  - string title (optional title)
  - int user_id (0 for anon)
- tags
  - int id
  - string label
- files
  - int id
  - string filename
  - string path (location on server)
  - int filesize (in bytes)
- users
  - int id

    - string name
    - string email
    - string password (encrypted obviously)

REST Endpoints:
How you can use HTTP to view and modify data on the server.

Server URL = *TBA*

Format: <request type> <endpoint name> <query parameters>

- GET croaks
    - returns a json array of all of the croaks on the server with their associated tags and files
- GET croaks/<id>
    - returns a json object representing the croak with the given id
- GET croaks radius(float), x(float), y(float)
    - returns a json array of all of the croaks that are less than <radius> kilometers away from the given coordinate (x = longitude , y = latitude)
- GET croaks tags(string), mode(int)
    - returns a json array of all of the croaks (with their tags and files) which have any (mode=0) or all (mode=1) of the given tags
    - tags is a string of all the tags separated by commas
    - this call can also combine with the parameters of the previous call
- GET tags
    - returns a json array of all tags on the server
- GET tags radius(float), x(float), y(float)
    - returns a json array of some of the most popular tags associated with recent croaks within <radius> km of coordinate
- GET files
    - returns a json array of all of the files with their associated tags
- GET files croaks tags(string), mode(int)
    - returns a json array of all of the files which have any (mode=0) or all (mode=1) of the tags in the string array <tags> associated with them

To be implemented:
- search by keyword
- events