Qualnotes Backend Design

Last revised

Table of Contents

# 1 TO BE DISCUSSED

# 2 TODO

* Flutter application build \*\* Install Android Studio and build: https://docs.flutter.dev/get-started/install
* JS maps, photo/videos, and filesystem

# 3 Context And Scope

Qualnotes is an mobile application to enable simple map sharing for researchers (currently less than 100 anthropologists). A shared map is an overlay above a base map from OpenStreetMap contains annotations in terms of an actual route and several media geolocated within the route or its surroundings.

The current qualnotes does not have a backend and as such it does not allow fault-tolerant, durable, and securely shareable content.

This document is meant to describe the minimum design required to provide a backend for the qualnotes application.

# 4 Goals And No Goals

The following are goals of the design:

* The API should provide the means to store and retrieve map overlays on a backend.
* The API should provide the means to store and retrieve media associated to specific location in a specific map overlays.
* The API should require authenticated access to both the maps and the associated media.
* The API should provide an authorization mechanism to allow sharing the access to the overlays and the images.

only provide access to the overlays to the map owner and a selected group of users.

The following are NOT goals of the design:

* Improve the UI or any aspect of its SDLC
* Server-side i18n features (e.g. error codes as a function of the language)
* Map versioning. In any case, we could design the DB layer to keep the old versions as a future extension point.
* Advanced mechanims for disaster recovery
* High-scalability
* No public unauthenticated access
* No write access
* Ownership can not be shared

# 5 API

The following endpoints are needed:

1. PUT /qualnotes/v1/map: Stores the map into the backend. Returns a $mapId. The payload contains a map overlay in the following format:

[  
 {"name": "Position 1",   
 "description": "bla bla bla",   
 "longitud": "1122234",   
 "latitude": "12234456",   
 "media": [ {"name": "funny face", "filename": "funny.jpg", "mediaId": "1223flsdjnvsjn"} ,  
 {"name": "sound in the street", "filename": "street.mp3", "mediaId": "122343flsdjnvsjn"}  
 ]},  
 {"name": "Position 2",   
 "description": "bla bla bla asdafa",   
 "longitud": "112223334",   
 "latitude": "1223433456",   
 "media": [ {"name": "Video", "filename": "funny.mp4", "mediaId": "1223flsfsdfdjnvsjn"}  
 ]},  
 {   
 "longitud": "1122234",   
 "latitude": "12234456"  
 }  
]

Note: It is possible to have anonymous positions and also positions without media.

1. PUT /qualnotes/v1/map/$mapId: Saves a new version of the map.
2. GET /qualnotes/v1/map/$mapid: Returns the latest version of a given map.
3. GET /qualnotes/v1/map/all/$userId: Returns the list of all maps owned by or shared to a given user in the following format:

[  
 {"name": "My map 1", "mapId": "123444", "ownerId":"jose", "read": "true", "write": "true"}.  
 {"name": "My map 2", "mapId": "1234445555", "ownerId":"marc", "read": "true", "write": "false"}.  
]

1. PUT /qualnotes/v1/map/$mapid/media: Stores a media file into the backend. Returns a $mediaId. We do not version media but just the implicit associations to the maps.
2. GET /qualnotes/v1/map/mediaid: Returns a media file.
3. POST /qualnotes/v1/map/userid: Grants read access to $mapId to user $userId.
4. POST /qualnotes/v1/map/userid: Revokes access to $mapId to user $userId.

# 6 Design

From an architecture point of view there are four components (excluding authentication):

* Client (either a mobile or web application)
* Back-end: REST API endpoint
* DB: Contains a table for the maps, MAPS, and a table for the access rights, ACCESS.
* Object Storage: Contains all the objects including the maps

![High-level architecture of the qualnotes back-end.](data:application/pdf;base64,)

High-level architecture of the qualnotes back-end.

## 6.1 DB Structure

The MAPS table contains the following fields:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ID | NUMBER | Primary Key |
| NAME | VARCHAR2(256) | Name of the map overlay |
| DESCRIPTION | VARCHAR2(4000) | Description of the map overlay |
| MAPID | VARCHAR2(256) | Unique ID of the JSON map file in the Object Storage |
| OWNERID | VARCHAR2(256) | Unique ID of the user owning the map |

The ACCESS table contains the following fields:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ID | NUMBER | Primary Key |
| MAPID | NUMBER | MapId of the map overlay |
| USERID | VARCHAR2(256) | Unique ID of the user |
| READ | NUMBER | Read access (0: False, 1: rue) |

# 7 Infrastructure Providers

The following alternatices are considered: \* Google App Engine \* Google Firebase \* AWS LightSale \* AWS BeanStalk

# 8 Appendix I: qualnotes client development instructions

# 9 References

qualnotes github client: https://github.com/orioli/qualnotes

Shareable Google Maps: https://www.makeuseof.com/tag/how-to-create-shared-collaborative-google-maps/

https://firebase.google.com/docs/cloud-messaging/fcm-architecture#design-pattern