Pet Finder Service

Use Cases

- Register a user
- User submits a new entity
  - Pet Missing
  - Pet Sighting
  - Pet Found
- User views a list of entities
  - of Pets Missing
    - as list
    - as map
  - of Pet Sightings
    - as list
    - as map
- User manages own entity
  - Pet Missing
  - Pet Sighted
  - Pet Found

Entities

- User Account
- Pet Missing
- Pet Sighting
- Pet Found

User Account

Represents a registered user. The system keeps a set of text properties about the user. Some of the properties are marked as:

- contact data;
- protected data.

Pet Missing

Represents a report on a missing pet.

Associated well-known props:

- registered ID (RFID tag)
- name;
- genus/breed;
- colour;
- size;
- weight;
- sex;
- etc.

Associated media:

- photos;
- videos.

The system keeps a set of **Geo-Facts** associated with this entity.

Pet Sighted

Represents a report on some pet being seen.

Associated well-known props:

- genus/breed;
- colour;
- size;
- weight;
- sex;
- etc.

Pet Found

Represents a claim on a pet being found.

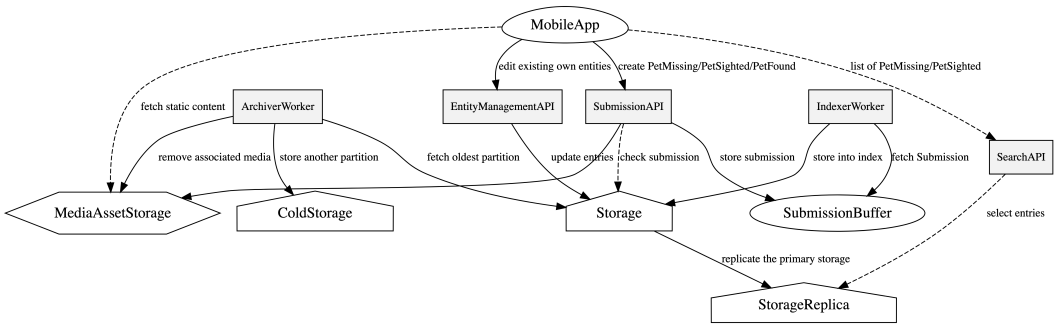
Associated well-known props:

- reference to a **Pet Missing** entity;
- contact data.

Associated media:

- photos;
- videos.

Components (the Main Functionality)



Mobile App

The mobile application provides access to the system.

For the submission cases usage of a mobile app is preferable due to the possibility to encode the media assets on the client side efficiently.

When viewing the list of entities — interacts with the **Search API**. When submitting a new entity — interacts with the **Submission API**. When editing own entities — interacts with the **Entity Management API**. The heavy media assets are fetched from **Media Asset Storage**.

Search API

The component provides an API to fetch the lists of entities based on some filter criteria.

The component can work with **Storage Replica** if there happens a necessity to scale this service much.

Entity Management API

The component provides an API to manage existing entities.

Submission API

The component accepts new submissions.

Publishes the media assets on the **Media Asset Storage**.

Writes a record to the **Submission Buffer**.

Submission Buffer

The component is able to store the data persistently in the append-only fashion.

Partionable.

Indexer Worker

The component "tails" the **Submission Buffer** and stores the entities in the **Storage**, doing it at the pace that is "comfortable" for the **Storage**.

Moderately scalable.

Storage

The database where the entities are stored in a normalized fashion.

Should support spatial indexes.

Should support creation-date based partitioning.

Not scalable.

Storage Replica

The replica of the **Storage**. Acceptable as read-only data source for those services that can work with stale data.

Media Asset Storage

A service that is suitable for storage of large amount of heavy files, and then deliver them to the wild Internet.

Archiver Worker

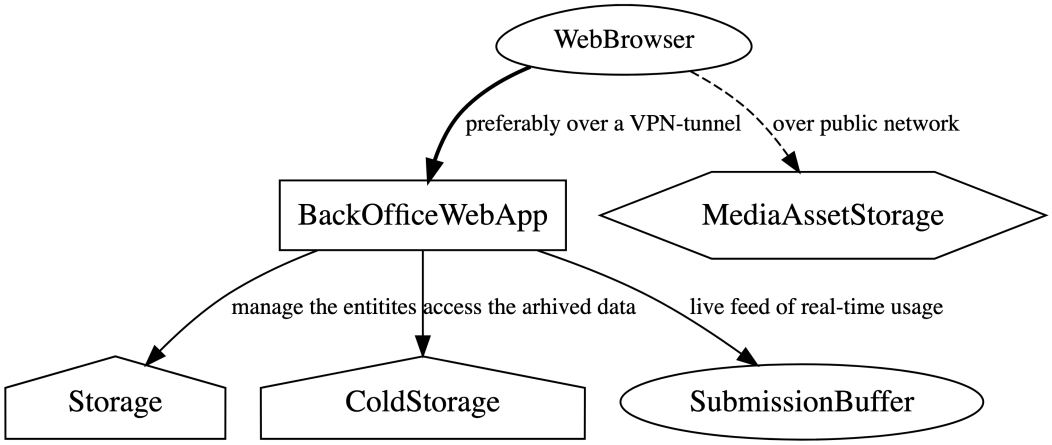
The component periodically detaches the old partitions (beyond the retention period) and puts them into the cold storage.

The media assets associated with the archived entities are to be pruned out of the **Media Asset Storage** within this procedure.

Cold Storage

A cheap storage for archive data. Could be `/dev/null` during the project start phase (until there is no data older than the retention period).

Components (the Back Office)



Non-Functional Requirements

Security

Back Office

Back Office app is a separate application from the public APIs.

Back Office app is accessed only from trusted networks (either VPN or source address whitelist).

Use MFA. Hardware keys — are preferable.

User Authentication

Passwords are to be kept:

- hashed (using a cryptographically strong one-way function);
- salted.

Use MFA. An email or SMS should suffice.

Authorization

Only registered users should be able to submit new entities.

The rate at whilch a user is able to submit new entities should be limited.

Introduce a mode of operation of the system when only the users with an SMS-based 2FA are allowed to submit new entities.

Availability

Run multiple instances of public-API services.

Use supervision/restart policies for the workers.

The **Storage** in this solution remains a critical point. It would make sence to have a procedure of switching from what is a failed **Storage** to what is a **Storage Replica**.

Observability

Metrics everywhere. Measure the main characteristics of the key events in the system: * rate of arrival; * the distribution of durations: time to handle/response time.

Data Storage

Please see the attached tables: * petfinder-numbers * petfinder-stats