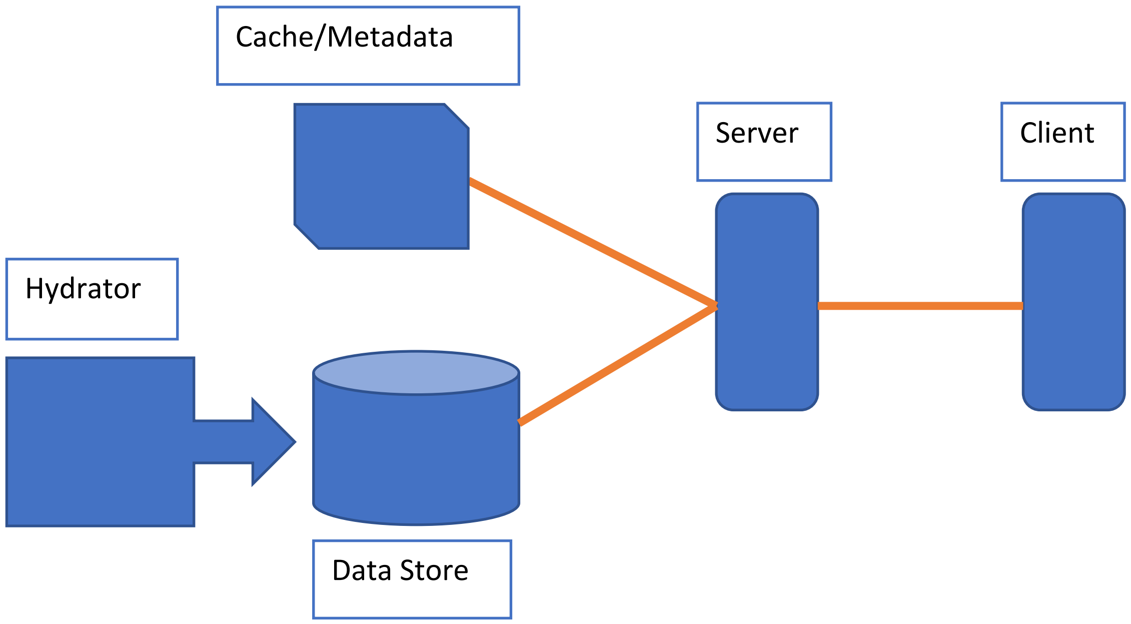
The Postal Service

Danni Beaulieu, David Souther, Peter Weisdepp, Xiaomei Xie
Seattle University, Software Architecture, 5200

The Postal Service is a system to verify address uniqueness. It must store a database of known addresses, worldwide. Users will be able to enter new addresses, as well as search for addresses using underspecified inputs. Address formats vary by country and region. The system must understand these varying formats, dynamically updating both form entry as well as search fields. Personal information is neither stored with, nor searched for within, system addresses.

As an example, Hans is specifying an address in Belgium. He enters Rue du Cornet 6 B-4800 VERVIERS BELGIUM into the system. Judith, from England, is searching to verify the address she wrote after a phone call. She enters the address in the UK format, 6 Rue du Cornet Belgium. The system will find Rue du Cornet 6 B-4800 VERVIERS BELGIUM, as well as Rue du Cornet 6 B-1040 ETTERBEEK BELGIUM. Matches will not show until the user supplies enough information for the number of matches to be reduced to some threshold, say 5. When the user finds the correct address and selects the suggestion, the rest of the form updates with the completed address information.



The Postal Service System requires five distinct components. Metadata must be globally available describing the format of countries' postal systems. Once defined, this metadata can be distributed globally through a CDN, embedded directly in binaries, or through other appropriate means. A dynamic form frontend will use the metadata to reactively create form elements, apply validation, and format requests to the API. This dynamic form can be used both for data entry as well as partial data search. The API server will accept create and search requests. It will use the metadata to normalize these requests to a canonical internal format, and then dispatch them appropriately to a database. A database system will store address fields in a sparse, normalized manner. The database will have indices on each column. The database(s) will be filled offline with data from a larger source, to be determined later.

- Alternative: Shard database on country (192 databases?), and only have the normalized columns per database.
- Alternative: Store denormalized addresses and maintain a custom set of indices over fields indicated by the metadata.

Style/pattern justification:

- RPC API (18) How can clients execute remote procedures over HTTP?
    - The Find/Match/Search API is effectively a procedure on the database. The API server handles normalizing the query, and then the database indices perform the heavy labor of finding appropriate record.
- Request/Response (54) What's the simplest way for a web service to process a request and provide a result?
    - Everything is request response. This operation specifically starts a procedure and gets the results, quickly. The request is not allowed to be a long running operation. The media response type is well defined.
- Request Mapper (109) How can a service process data from requests that are structurally different yet semantically equivalent?
    - The API server's normalization will be critical to handling the different request formats (e.g. English number then street, vs continental street then number).
- There is no specific implementation style – the "find" RPC is a self-contained piece of business logic, tied to a specific database schema.