Aalto University, Department of Built Environment

Waste pickup aggregation and reporting system: Summary of software application (Unpublished report)

Terefe Abel

October 23, 2016

Technical Description

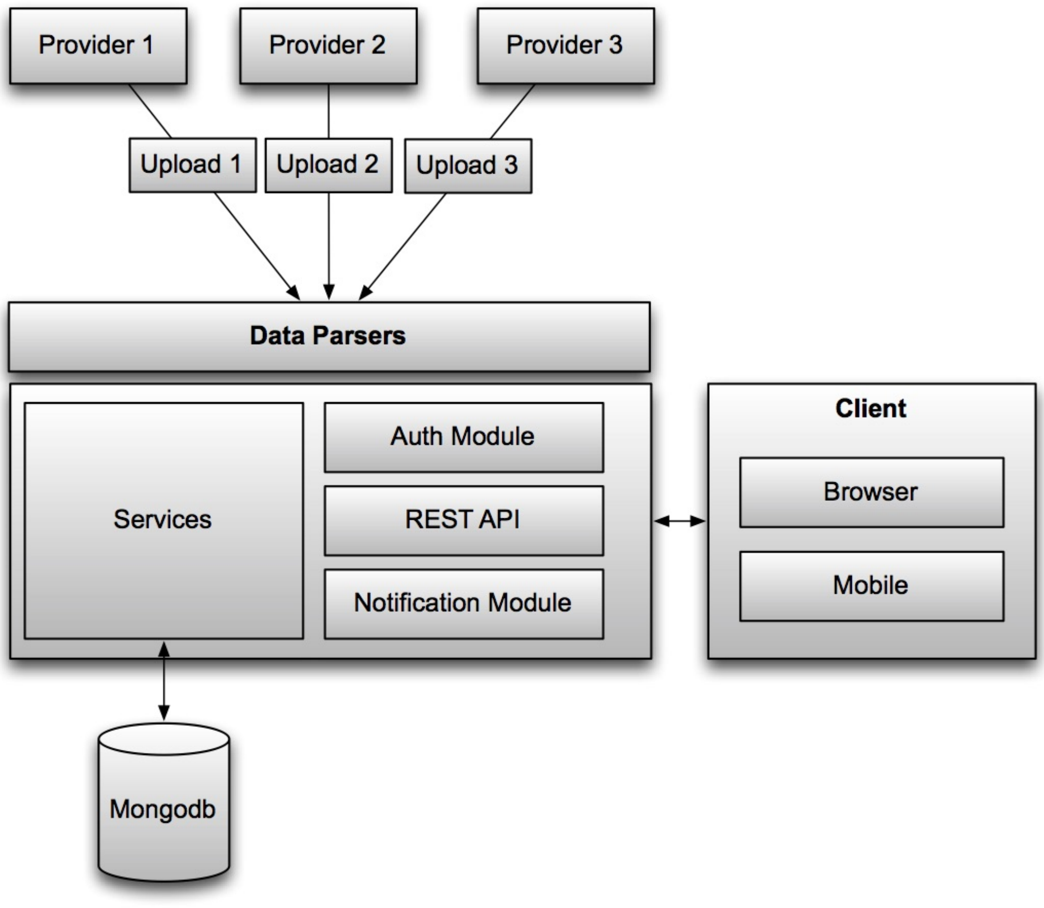
The pickup aggregation and reporting system works by collecting data from various sources, that provide household waste data, and transform it into a form that can be queried, analyzed and grouped into various indicators so that one can get a general information about the state of the pickup process faster and with ease. The application uses: graphic visualization tools for statistics, maps for spatial data and text search for documents.

Application Architecture

The application follows the software-as-a-service model in the sense that a single application layout is served from a centrally hosted service. In this situation, it is necessary to have a way to process input data from various sources. And this is a challenge because data comes in various formats. To handle these situations, we use dependency injection (DI) for selectively applying a parser depending on the data provider. This means, on the event of having a new data provider, we need to implement a parser for the data and use DI to inject our parser in the service. On the other hand, we have a document based database for storing the data. The choice for a document based database is made on the basis that the schema for the data model should be relaxed since there are inevitable uncertainties in the structure of the data. Much of these problems could be avoided by using standardized data. However, at the moment there is no such single standard that can be parsed with a single parser. See Figure A

Currently, data is manually uploaded as a CSV file. However, eventually the plan is such that, data can be directly uploaded from waste trucks rendering the services to be realtime or near-realtime. After uploading, data is parsed and stored in the database. Uploading triggers an application level notification to authorized user accounts so that they get up-to-date information. The notification implementation is powered by the Akka^{TM}library and websockets. Basic indicators such as total weight in a period, total quantity in a period are implemented. These indicators are shown visually as charts. A map shows the pickup points and related pickup information as it exists in the original data. On top of that, users can browse raw data in a table. The fields in the table can be filtered by text inputs.

Figure A: Application architecture



Providers refer to the various sources where the data comes from. Usually, these are companies that collect pickup data and are stakeholders in the project. The providers would push data to a central repository, where a file-watcher would notify the parser to start, upon arrival of new data.

Parsers A parser is a part of the code that extracts the information from data. For every provider we have a parser that knows how to extract the data.

Auth module Authentication module is part of the code that authenticates users and checks user permissions and privileges.

REST API REST API refers to the HTTP interface of the application. This is how clients are able to reach to the service provided by the application

Notification Module Refers to part of the code that handles realtime and near-realtime notifications including messages about new data arrival.

Client The client application refers to the part through which users interact with the application. This is typically done through the browser and mobile application. Currently, we don't have a dedicated mobile application, however our browser application is being built to morph into a mobile-app-like experience on mobile devices.

Services refer to the part of the code that talks to the database. This includes operation such as reading from and writing to a MongoDB database. Finally, much of the backend of the application such as parser and all the services are built with the Scala^{TM}programming language and Play Web Framework^{TM}. The client application however is built with JavaScript and AngularJS^{TM}web framework in addition to several other libraries.

MongoDB A document based database used for storage of waste data. MongoDB is well suited database system because it has a flexible schema that allows certain kinds of changes in the structure of the input data.