Links - Project Report

Abhijith Madhav, Arjun S Bharadwaj

September 8, 2014

Contents

1 Introduction

A bookmarking service is a centralized online service which enables users to add, annotate, edit and share bookmarks of web document.,The most popular bookmarking service currently is *Delicious*. *Scuttle* is one of the other well known open source alternative.

Links is envisioned as a similer online bookmarking service with the specific requirement that it should be deployable on a private server within an organization. The purpose is to enable people in an organization to share bookmarks with each other.

Links is intended to be an extensible, modular system with an exposed API. Third parties must be able to write plugins and applications using this API.

It is envisioned to seemlessly fit into the share motto of the net community, be it through simple email or through popular social platforms like facebook and twitter.

2 Requirements Specification

2.1 Functional Requirements

This section provides the functional requirements of the project.

1. Login
   - User logs into the system.
   - *Input:* User enters the login details and clicks on login.
   - *Output:*
     - If the validation of the user credentials is successful:
       - The user is redirected to the Homepage.
     - If the validation of the login details is unsuccessful:
       - An appropriate error message is shown to the user.
2. Signup
   - User clicks on Signup.
   - *Input:* Signup details are asked.
   - *Output:*
     - If validation of the details is successful:
       - Account is created.
     - If validation of the details is unsuccessful:
       - An appropriate error message is shown to the user.
3. Save Links
   - User bookmarks the links.
   - *Input:* A Title, URI, Tags and Annotations are provided.
   - *Output:*
     - After the link is saved:
       - Acknowledgement for the action is provided.
       - Suggest tags if same link was shared by others based on the visibility level of the user.

4. Edit Links
    - User edits the links.
    - *Input:* An existing link, modified data (title, tags, annotations).
    - *Output:*
        - After the link is updated:
            - Acknowledgement for the update is provided.
5. Delete Links
    - User deletes the links.
    - *Input:* An existing link shared by the user.
    - *Output:*
        - After the link is deleted:
            - Acknowledgement for the deletion is provided.
6. Create User Groups
    - Creation of Groups by the user.
    - *Input:* Group name and group description corresponding to the group.
    - *Output:*
        - If the group name is not already used:
            - Acknowledgement for the creation is provided.
        - If the group name already exists:
            - Appropriate error message.
7. Join Group
    - User joins the group.
    - *Input:* Group name of the group that the user wishes to subscribe to.
    - *Output:* Acknowledgement of the subscription is provided.
8. Unjoin Group
    - User unjoins the group.
    - *Input:* Group name of the group that the user wishes to unsubscribe to.
    - *Output:* Acknowledgement of the un-subscription is provided.
9. Share
    - Sharing of the links.
    - *Input:* A set of bookmarks to be shared via email or popular social media

- *Output:* Share bookmarks via the specified media.

10. Add Members
    - Add members to the group.
    - *Input:* User names by the group owner.
    - *Output:* Acknowledgement of addition to group.

11. Remove Members
    - Remove members to the group.
    - *Input:* User names by the group owner.
    - *Output:* Acknowledgement of deletion to group.

12. Interoperability
    Expose APIs through which the following can be done
    - Search based on bookmarks, annotations, tags in accordance to the visibility of the user.
    - Add, delete and edit the bookmarks in accordance to the visibility of the user.

    As a proof of concept, an android application that consumes the data from the provided APIs will be created

2.2 Non-functional Requirements

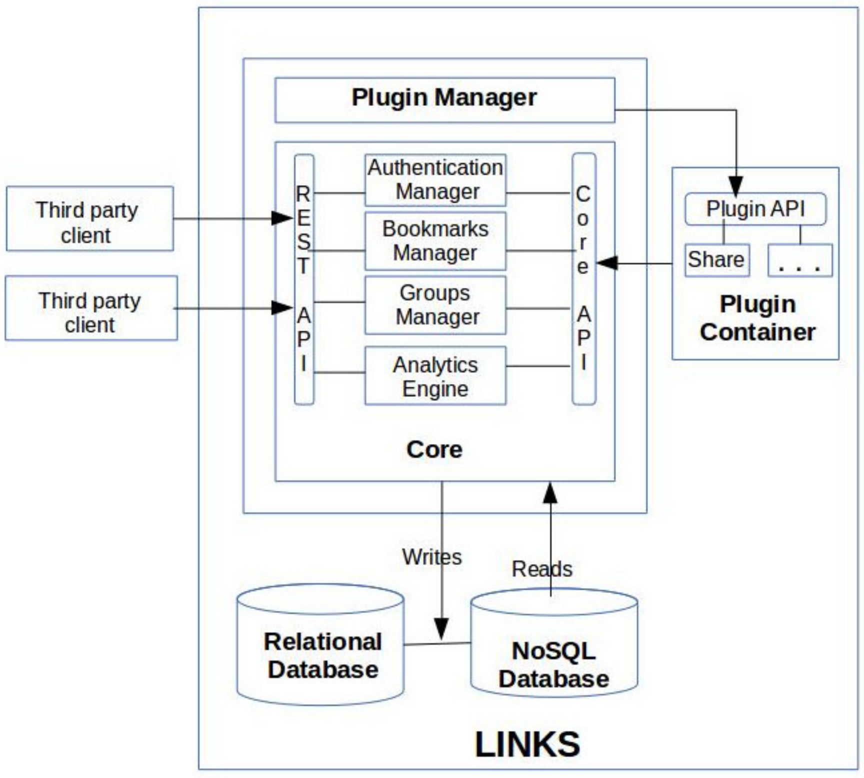
2.2.1 Security Requirements

1. All users must be authenticated before being given modification privileges in “Links”.
2. Third party clients must not have access to passwords of users and must instead use a token based authentication scheme.
3. User passwords must be stored as salted hashes in the ”Links” system.
4. HTTPS connections between clients and the ”Links” server must be supported.

2.2.2 User Experience

1. The browser extension using the ”Links” API must enable saving and/or sharing bookmarks from a third party website without having to navigate away from that site.
2. Where possible, portions of any ”Links” webpage must be updated without having to refresh the entire webpage from the server.

3 Architecture and Design



3.1 Architechture

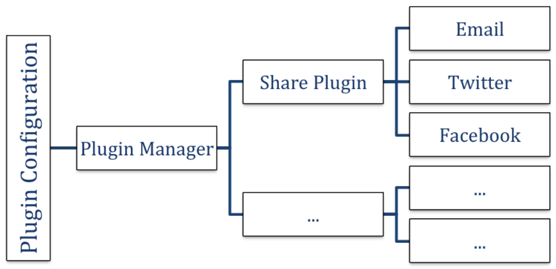
The architechture of 'Links' consists mainly of a core module providing the basic functionality of storing, retrieving and sharing of bookmarks. The proposed architecture also has an analytics engine as a part of the core which is yet to be implemented.

The plugin module is intended to help extend a 'Links' installation by 'plugging in' useful functionalities. It consists of a plugin manager which loads at launch time all the small self contained plugins into 'Links'.

3.1.1 Core Module

The core consists of an authentication manager, bookmarks manager and a groups manager which implement functionality required for authentication, bookmarks management, groups management and sharing of bookmarks through groups.

3.1.2 Plugin Module



These are the components of the Plugin Module.

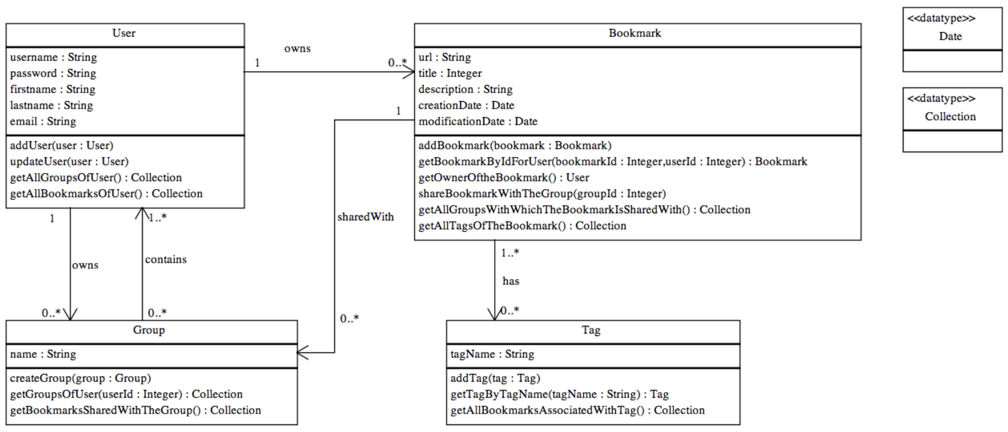
1. Configuration File: This file will consist of all the plugins that are enabled in the system.

2. Plugin Manager: The Plugin Manager will lookup the configuration file and will enable/disale the plugins.

3. Share Plugin: This is a sample plugin implemented which will help to share the links on various platforms like Email, Twitter and Facebook.

3.2 Database Design

The core of Links access the database for data persistance. The class diagram detailing the schema and relationships is as follows


User
username : String
password : String
firstname : String
lastname : String
email : String
addUser(user : User)
updateUser(user : User)
getAllGroupsOfUser() : Collection
getAllBookmarksOfUser() : Collection
owns
1
0..*
Bookmark
url : String
title : Integer
description : String
creationDate : Date
modificationDate : Date
addBookmark(bookmark : Bookmark)
getBookmarkByIdForUser(bookmarkId : Integer,userId : Integer) : Bookmark
getOwnerOftheBookmark() : User
shareBookmarkWithTheGroup(groupId : Integer)
getAllGroupsWithWhichTheBookmarkIsSharedWith() : Collection
getAllTagsOfTheBookmark() : Collection
<<datatype>>
Date
<<datatype>>
Collection
1
sharedWith
0..*
1
owns
0..*
1..*
contains
0..*
Group
name : String
createGroup(group : Group)
getGroupsOfUser(userId : Integer) : Collection
getBookmarksSharedWithTheGroup() : Collection
1..*
has
0..*
Tag
tagName : String
addTag(tag : Tag)
getTagByTagName(tagName : String) : Tag
getAllBookmarksAssociatedWithTag() : Collection

4 Implementation

4.1 Technology used

- A MySQL instance is used for the database because of previous familiarity with the same.
- Ruby on Rails is used as the web application framework. The reasons for the same are
  - A full-stack framework that emphasizes the use of well-known software engineering patterns and paradigms, including convention over configuration (CoC), the active record pattern, and model–view–controller (MVC).
  - Easy to plugin gems to implement many common features of web application.
  - Faster development time.

4.2 Controllers

The functionality of The core module is realized by implementing controllers in the rails MVC framework. The implemented controllers can be found at /links/links/app/controllers. The main ones are briefly described below.

- bookmarks_controller.rb : Implements methods to perform create, update and delete operations on the bookmarks object.
- groups_controller.rb : Implements methods to perform create, update and delete operations on the groups object.
- searches_controller.rb : Implements methods to perfom retrieve operations on the bookmarks and groups objects. pages_controller.rb : Implements simple methods to render static pages.

4.3 Authentication and User Management

Authentication into Links for a user is implemented using the Devise gem. Devise is a flexible authentication solution for Rails which takes care of the entire user management process and authentication process. Features not available by default in Devise can easily be added and are added in users_controller.rb. Uploading the photo of the user and display of the same is added in this way.

4.4 Authorization and REST API's

API's to perform CRUD operations on bookmark and group objects are exposed in a RESTFUL way. Controllers for same can be found in /links/links/app/controllers/api/v1.

Authorization for third party clients is through OAuth2 protocol. Doorkeeper gem is used to implement a OAuth provider inside of Links.

Documentation for obtaining authorization and usage of the RESTful API's can be found here.

4.5 Import from Delicious

This is a feature which helps in migration of bookmarks from Delicious. delicious_controller.rb implements an OAuth client which pulls a users bookmarks from his Delicious account and integrates the same with his Links account. It is intended to make this as a plugin rather than as a core feature in future.

5 Future work

The following are the features that need to be worked on

- Analytics Engine and tags suggestions.
- A more sophisticated installation script.
- Admin UI specifically to administer plugins.
- Make the import feature as a plugin.