This application is going to be used to create bookmarks that will have it’s own structure. This will allow users to have bookmark collections. With its own template. Such as bookmark collection of coding problems with its own structure. Bookmark collection of movies with its own structure.

Each structure/template will have mandatory and optional fields.

You can share and use templates from other users. Each template can be public and private

Also, these bookmarks can be stored in a folder hierarchy

Batch operation to modify, delete, update templates should be available

How things will work

You login / use locally

Your local device is going to hold a client key if using online

Use JWT for now. Implement your own security check in future

DB Model

1. Each user is going to be in db record.

Users will have its general info such as name, dp etc etc

1. Template is going to have its own table

Template will have unique ID along with other info such as who created it, the structure, etc

It can be public and private. If its public then it is going to be visible to all other users

1. Bookmark

Bookmark will have id to link it to the user whom it belongs to as well as the template it is using along with other details

1. Images are going to be stored in template ID folders IF a template has support for image, it can have sub folders if it has many fields that needs image

Main Services:

1. Public gateway
   1. Used by client to interact with the backend.
   2. Will have to send client ID to verify who the user is, it will be invalid after certain amount of time
   3. Public gateway will also have security check to validate the requester using the client ID and then forward the request to other services
2. User service:
3. Bookmark service:

Communication will be figured along the way

1. Login/Signup : gateway.login(email,password) -> userService.login();
2. Create/Get Template : gateway.createTemplate(userinfo,template info)->bookmarkservice.createTemplate(userinfo,templateinfo)
3. Create/Get bookmark : gateway.createBookmark(userinfo,templateinfo,bookmarkinfo)->bookmarkservice.createBookmark

Users ==

Email: PK

Password;

Other details:

UserTEmplateConnector==

userID : PK

TemplateID []: FK

(This will allow super fast access of all the template of a user as we get the PK of template from this one list for a user)

Template==

templateID : PK

userWhoCreated : FK

StructOfTemp

UserBKMConnector==

userID : PK

BookmarkID []: FK

TemplateBKMConnector==

templateID:PK

BookmarkID:[] FK

Bookmark==

ID : PK

userWhoCreated: FK

templateID : FK

Data

UserRootFolderConnector==

UserID : PK

RootFoldersID[] : FK

FoldersTable

ID: PK

ChildrenID[] : FK

BookmarksID[] : Fk

CreatedByUserID: FK

Public or private bool : This will ensure if its viewable by public or not.

//For folders this is what I came up with

When I enter the app

I want to see a list of root folders of the logged in user

We can have a FolderTable which will hold all the folders of users and have a connector table as well

The connector table will hold a PK of

Getting root folders should be easy per user using the connector TAble, now we have a list of rootFolderIDs

When we enter the root folder we are going to now have a list of sub folders

How should this be done?

Since we have a list of RootFolderID, to get the children IDs of that folder.

Now we have a list of sub folders.

To get its children we can continue this.

Now comes the part where we reach the end of the folder.

Show the bookmarks available.

How?

Maybe the table can have a list of bookmarks ID it holds.

That will work I guess.

Yep, and we can perform security checks to make sure everything is valid

As for the template structure it is simple json

{

“fieldName”: { “optional” : “true” , “type” : “text”}.

“secondField” : {…}

}

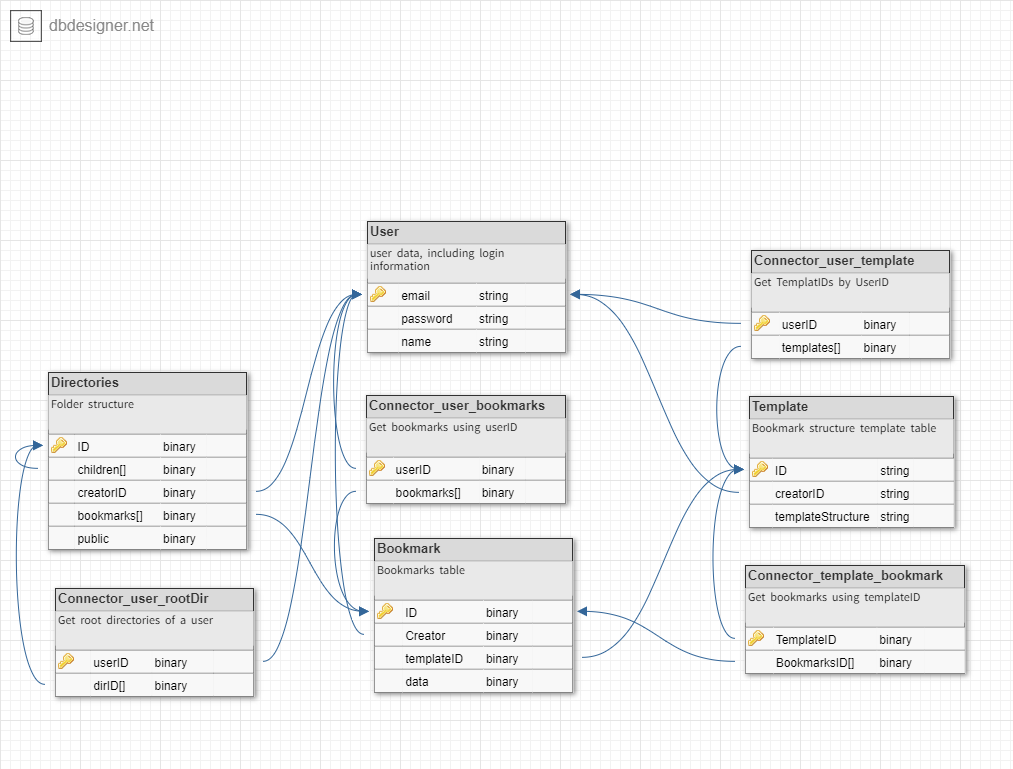
For bookmark data it will be json too

{

“fieldName”: “value”,

“secondField” : “val2”

}

s

For accessing folders we do the follow thing

When we are given a path to a folder such as

http:/folder/2347

Where numbers are the folderID

We will check the final folderID only

If it is public then we will allow anyone to access it

If it is not public we will check for the user who is trying to access by extracting the userID from the JWT token’s subject.