Test 1

GITHUB REPOSITORY: https://github.com/netaiko/file_system

1. *Create a database design to able to store given file system as above.*

ERD diagram:

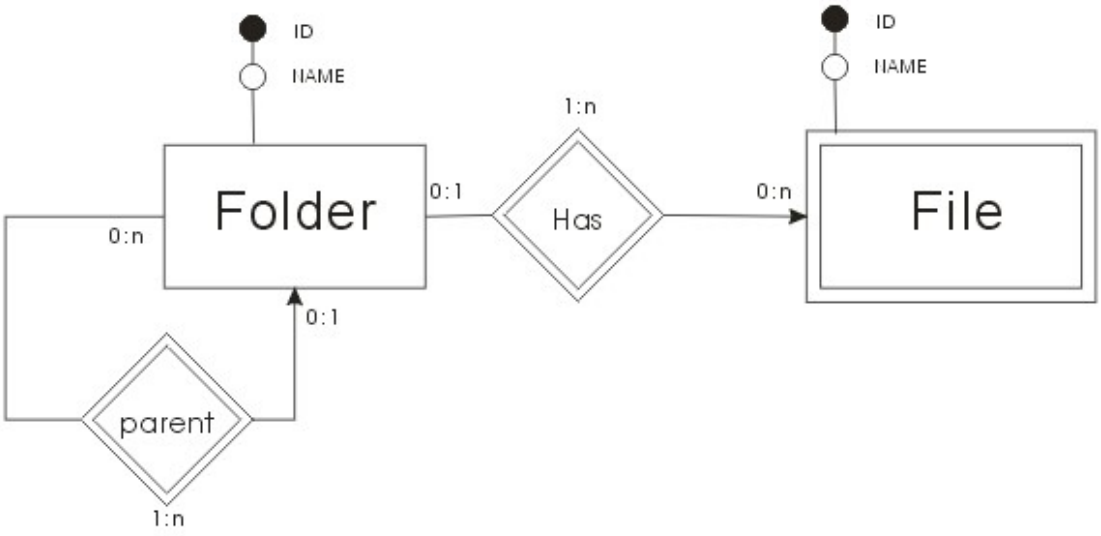


Table Folders:

ID: Unsigned Integer Auto-incremented, primary key

Name: String

Parent_id: Unsigned Integer, Foreign key(Folder)

Table Files:

Id: Integer Auto-incremented

Name: String

Folder_id: Unsigned Integer, Foreign key(folder)

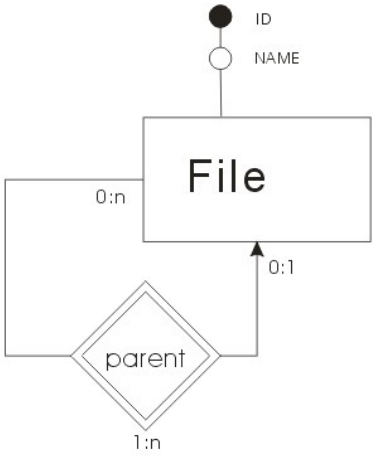
2. *Create a text file with above file system structure and read and insert into database. Inserting above structure into database manually will not be accepted.*

The text file is files.txt and this is the format:

```
C:\Documents\Images\Image1.jpg
C:\Documents\Images\Image2.jpg
C:\Documents\Images\Image3.jpg
C:\Documents\Works\Letter.doc
C:\Documents\Works\Accountant\Accounting.xls
C:\Documents\Works\Accountant\AnnualReport.xls
C:\Program Files\Skype\Skype.exe
C:\Program Files\Skype\Readme.txt
C:\Program Files\Mysql\Mysql.exe
C:\Program Files\Mysql\Mysql.com
```

3. *Research for possible solutions and explain that why your solution is better fit than other possible solutions.*

   The solution implemented here is one of the possible solutions. Another possible solution is considering Folder and File as a same entity, following this:



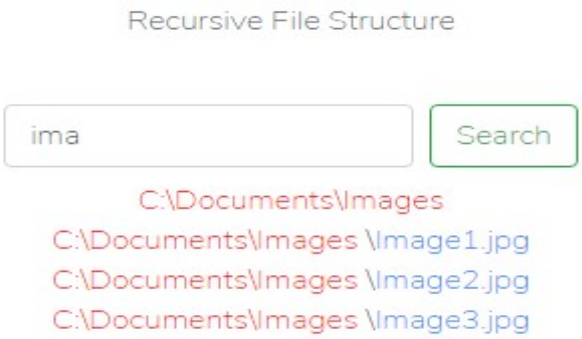
Multiple approaches can lead to similar solutions. The solution I implemented is therefore one possible solution which I considered suitable to solve the given questions. Improvements and development may continuously change the original code leading to improvements of any code over time.

4. *Create a web interface for the user so they can able to search any file or folder within database. Web search interface should only include an input box and search button. When a user click search button, result should be listed as below. For example if the user enter 'image' into the input box and the click search, the code able to list the results as below. C:\Documents\Images C:\Documents\Images\Image1.jpg C:\Documents\Images\Image1.jpg C:\Documents\Images\Image3.jpg*

When the user types a word and clicks on **Search** the form sends a HTTP GET request through the same URL.

The controller Controllers/IndexController.php will get the files and folders through the method **index** and will return a view with the folders and files containing the chosen word summarised in a list.

The view **welcome** is located in Views/Filesystem/welcome.php



Project Structure

Config: contains all the configuration files

Models:

Each database table has a corresponding "Model" which is used to interact with that table.

Requests

IndexRequest: the word sent by a HTTP GET Request must be validated by this class

Services

Loader: This is the class what load a file and add the information to the database. All the methods have been commented in the code.

Tests

Contains all the PHP Unit test files for testing some specific parts of the project.

Validators

Contains the files with the classes used for validations.

FileValidator: Validation for files

FolderValidation: Validation for folders

TextFileValidation: Validation for a whole text file what contains the information for adding to the database

Views

Filesystem\welcome.php here the main user interface where an user can look for the files

DatabaseConexion

This class is operating directly with the DataBase using the library PDO and executing SQL Sentences.

5. *The code should be written with OO PHP using design patterns. Unit test is preferred but not required.*

I built a web interface without using any framework. I made a simple MVC OOP structure.

I ran some PHP Unit tests for testing the application.

FileTest: For testing the File class.

FolderTest: For testing the Folder class.

LoaderTest: Testing the process for creating a file, storing in the database

ValidationTest: Testing the validation methods