Design specifications
R.00

CS50 Final Project 2014
Marco Chrappan

Index

1. Introduction

Fixxer is a general purpose maintenance management system database built on MYSQL and PHP language. It has been designed for managing preventive maintenance on a collection of assets defined in a tag list (where the tag code itself indicate a custom code to identify an asset).
Its purpose is to define a set of activities to be periodically perfomed on the assets and to record the activities completion over a period of time, comprising remarks, hours spent, cost etc.

This document will detail the program structure and design choices as well as excerpts of code for the most relevant parts

2. System Architecture

The system is based on the following components:

Operating System
Linux Fedora 19
Web Server
Apache 2.4.6 Server name: localhost Port 80 Document root: /var/www/html
Database Server
MariaDB 5.5.32 With server charset: UTF-8 unicode (utf8) Client API Library: mysqli
Server side Language
PHP 5.5.3

3. Folders structure

The web server content is divided into three main folders:

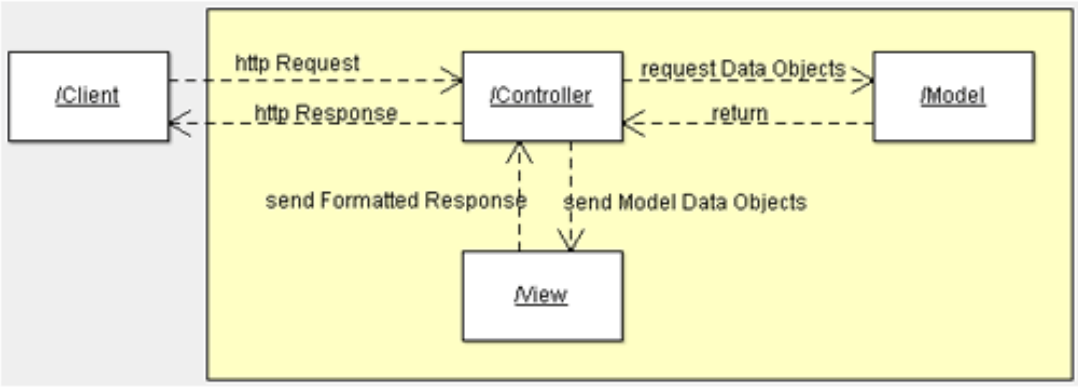
- Includes
  - Constants.php
  - Db_connection.php
  - Functions.php
  - Session.php
  - Validation_functions.php
- Public
  - \CSS (css stylesheets)
  - \IMG (website images)
  - \JS (javascript)
  - php files (controllers and views)
- Templates
  - php files (templates for lists, html header, footer and forms)

4. Interactions

The program is modelled in part following the MVC pattern originally formulated by the Smalltalk creators at Xerox Parc Labs during 70's

The pattern is made up of three parts:

- **Model** the permanent storage of data. It allow access to data, it allow an external source to collect it and also to write data. It accepts all requests if correctly formatted and answer accordingly. The model can be summed up as the database server, the tables contained in it and the API between the server and PHP.
- **Controller** it is responsible for handling data inputs and update the model accordingly. It is the interface between the user and the model.
- **View** the view is where the data coming out from the model is generated into the final output. Generally this part produce the finally HTML and display it. A button is generated by view. The user clicking it triggers an action in the controller.



5. Database structure

The MariaDB contains all the data managed by the Fixxer into the a database named fixxer which contains the following tables:

- admins
- equip_subtypes
- equip_types
- maint_master
- serials
- service_call
- service_wins
- tag
- work_ins

Details on the tables structures are contained in the appendix.

The admins table contains the users information

All the program data is contained in the remaining tables specifically:

- tag is the main table containing tags. This table is linked to serials and contains two fields replicated in the content from the tables equip_types and equip_subtypes
- maint_master contains the main maintenance master definition. Records id serve as a link to the work instructions contained in work_ins.
- The same structure is applied to service_call (main table) and service_wins (child assets or service call work instructions)

6. Query function

The query function query using mysqli method has been taken from the CS50 course example. This function has the SQL query as an argument and performs connection, SQL statement preparation, execution and results output to the user via the $results variable.

A sister function has been created to countrows returned from a query.

7. List view

The simplest php code is the one listing the content of a table. The taglist is an example of this being composed of two parts: the controller and the view (index.php and taglist.php)

Index.php prepares a simple query to the database asking for some data in this case:

```
$sqlstring  = "SELECT tags.id, tags.tag, tags.description, tags.plant, tags.area, tags.equip_type,
tags.equip_subtype, tags.owner, tags.critical, tags.created, ";
$sqlstring .= "serials.ser_code, serials.vendor, serials.model ";
```

```
$sqlstring .= "FROM tags LEFT OUTER JOIN serials ";
$sqlstring .= "ON tags.id = serials.tag_id";
```

This is a SELECT query with an OUTER JOIN to the serials table. The data is collected in an array called $rows. The fields in this array, record by record, are then unpacked into an associative array called $positions.
The array is passed to the template taglist.php for visualization.

Two extra features have been added to the template:

- Each element has been represented with an HTML link in order to open the edit window related to itself (tag > edit tag)
- The column names are link too, enabling the ORDER BY clause into the previous SQL query (the DESC/ASC option can also be toggled by clicking on the same column name)

8. Forms

The forms have been generally divided also in a controller and view files but for the edit_tag.php form the two functionalities have been condensed into one single php file.

The file when requested lookout for a POST request method. If this is not true, some queries are performed based on the variables contained in the $_GET array.

Infact the tag_edit.php form is usually called with the tag id value like this:

http://localhost/edit_tag.php?id=1

So three queries are then performed on three different tables: tags, equip_types and serials.

Then an additional query is perfomed on the maint_master table if the equip_type field from tags table match a maint master corresponding field.
HTML is then formatted with header + form + footer. In this case the form is not represented empty but the form input areas are filled with the result of previous query representing value for the tag the user wants to edit.

When the user press the submit button the form POST the content to edit_tah.php with all the data in the $_POST array. In this case the first part of the PHP file becomes active and the UPDATE on relevant tables is done.

9. Generate events

The main php function is the generate.php file which contains the code for generating events of maintenance during a delimited period of time set by two variables.
The pseudocode for this function is the following:

```
Get lower datetime / upper datetime
SELECT * FROM TAGS

For each row
{
        if the equipment type is not null
        {
            SELECT * FROM maint_master WHERE maint_master.equip_type = current
            row equip_type

            Foreach maint_master row
            {
               if maint master is active
               {
                    SELECT from work_ins work instructions with the same equipment
                    subtype of the current row equipment subtype

                    Generate delta variable corresponding to work instruction frequency
                    Generate a base date string from the basedate field and assign this
                    value to newdate

                    While (newdate <= upper datetime)
                    {
                       calculate newdate = newdate + delta variable

                       if (newdate > lower datetime && newdate < upper date time)
                                {
                                        look for a service call opened on same day and same tag.

                                        If no service call are found create a service call and
                                        append to the new record corresponding work instructions
                                        in the the service_wins table

                                        Otherwise check in the existing service call if there is
                                        already a corresponding work instruction matching the
                                        current work instruction from the maintenance master.

                                          If no work instruction is found then add the work
                                          instruction to the service_wins table
                                }
                    }
               }
            }
        }
}
```

10. Security

The passwords taken from the forms are encrypted in the admins table with the following functions:

```
function pwd_encrypt($password) {
$hash_format = "$2y$10$";
$salt_length = 22;
$salt = generate_salt($salt_length);
$formatsalt = $hash_format . $salt;
$hash = crypt($password,$formatsalt);
return $hash;   }


function generate_salt($length) {
$rnd = md5(uniqid(mt_rand(),true));
$base64 = base64_encode($rnd);
$base642 = str_replace('+','.',$base64);
$salt = substr($base642,0,$length);
return $salt;   }
```

The encryption is based on the blowfish encryption system and the crypt function.
The $salt variable is generated with a md5 encryption of a random number generated by the mt_rand function. The $salt is then concatenated to the given password and then fed into the hashing algorithm. The cost parameter for the blowfish encryption is 10.

11. Appendix (SQL tables)

Server: localhost » Database: fixxer » Table: admins

admins

Column	Type	Null	Default	Comments
id	int(11)	No		
username	varchar(20)	No		
password	varchar(60)	No		

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	id	4	A	No	
username	BTREE	Yes	No	username	4	A	No	

Server: localhost » Database: fixxer » Table: equip_subtypes

equip_subtypes

Column	Type	Null	Default	Comments
id	int(10)	No		
parent_id	int(10)	No		
equip_subtype	varchar(25)	No		
description	varchar(25)	No		

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	id	11	A	No	

Server: localhost » Database: fixxer » Table: maint_master

maint_master

Column	Type	Null	Default	Comments
id	int(11)	No		
master	varchar(15)	No		
rev	varchar(3)	No		
active	tinyint(1)	No	1	
equip_type	varchar(25)	No		
sop	varchar(25)	No		
details	varchar(255)	No		
remarks	varchar(255)	No		
effective	date	No		

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	id	1	A	No	
id	BTREE	Yes	No	id	1	A	No	

Server: localhost » Database: fixxer » Table: serials

serials

Column	Type	Null	Default	Comments
id	int(11)	No		
ser_code	int(10)	No		
family	varchar(15)	No		
description	varchar(25)	Yes	*NULL*	
vendor	varchar(25)	Yes	*NULL*	
model	varchar(25)	Yes	*NULL*	
tag_id	int(11)	Yes	*NULL*	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	id	12	A	No	

Server: localhost » Database: fixxer » Table: service_call

service_call

Column	Type	Null	Default	Comments
id	int(10)	No		
tag	varchar(10)	No		
date	date	No		
opened	tinyint(1)	No		
whours	int(4)	Yes	*NULL*	
owner	varchar(15)	Yes	*NULL*	
closedate	date	Yes	*NULL*	
comment	varchar(255)	Yes	*NULL*	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	id	4	A	No	

Server: localhost » Database: fixxer » Table: service_wins

service_wins

Column	Type	Null	Default	Comments
id	int(10)	No		
tag	varchar(10)	No		
date	date	No		
wins	varchar(25)	No		
service_id	int(10)	No		

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	id	8	A	No	

Server: localhost » Database: fixxer » Table: tags

tags

Column	Type	Null	Default	Comments
id	int(11)	No		
tag	varchar(10)	No		
description	varchar(25)	No		
plant	varchar(10)	No		
area	varchar(10)	No		
equip_type	varchar(25)	Yes	*NULL*	
equip_subtype	varchar(25)	Yes	*NULL*	
owner	varchar(15)	Yes	*NULL*	
critical	tinyint(1)	No	0	
created	date	No		
basedate	date	No		
active	tinyint(1)	No	1	
position	varchar(15)	Yes	*NULL*	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	id	3	A	No	
id	BTREE	Yes	No	id	3	A	No	

Server: localhost » Database: fixxer » Table: work_ins

work_ins

Column	Type	Null	Default	Comments
id	int(10)	No		
equip_subtype	varchar(25)	No		
freq	enum('2Y', 'Y', '6M', '3M', 'M', 'W', 'D')	No		
instruction	varchar(25)	No		
section	int(99)	No		
master_id	int(10)	No		

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	id	5	A	No	
id	BTREE	Yes	No	id	5	A	No	