

San Carlos Regional Headquarter

Computer Engineering Department

Componentes y comunicaciones en internet

Professor: Dennis Valverde

Third programmed project:

Audits (MVC)

Alonso Vega, 201042592  
Gabriel Fallas, 201030952

Day, June XX, 2013

Santa Clara, San Carlos

Contents

[Problem Description 2](#_Toc358671562)

[Problem Solution 3](#_Toc358671563)

[Database Design 3](#_Toc358671564)

[Frontend functionality modification (and a new User Interface) 4](#_Toc358671565)

[Backend functionality 5](#_Toc358671566)

[Non-functional requirements 6](#_Toc358671567)

[Result Analysis 7](#_Toc358671568)

[Conclusions and Recommendations 8](#_Toc358671569)

[Bibliography 9](#_Toc358671570)

# Problem Description

In this project, we have to program the second project to a new environment. The second project is made using Node.js as a server, Mongo DB as the database engine, but now, in this project we have a new challenge, and that’s the new project needs to be developed using CodeIgniter as a PHP framework, and MySQL as the database server.

The functionality and views to implement in this project must to be the same made in the past one. That’s, this project has to have the four views: Login, Main, Audits, and History view. And inside each view, we need to implement all necessary methods to make the site completely functionality.

Besides, one new thing to use in this third project is Apache, the HTTP server we will use to make the project runs as a website in our local host location, so it can be accessed.

# Problem Solution

## Database Design

The database engine and server is MySQL. We made the following structure (a schema) and tables of the database, within the “Audits” database, which this time it consists of 7 tables:

* Users table
  + id\_user: primary key of the table.
  + username: this is used for the login.
  + password: an encrypted password used for the login implementing security.
  + name: the name of the user, used for showing the current user logged in into the application.
* audits\_assets table
  + audit: primary key, it’s a foreign key to the audits table.
  + asset: primary key, it’s a foreign key to the assets table.
  + present: an integer value that indicates if the asset is present in the room.
  + state: an integer value that indicates if the asset is in a good or bad state.
  + rating: an integer value that rates the asset between 1 and 10.
  + comment: a comment for the asset.
* audits table
  + id\_audit: primary key of the table.
  + room: foreign key of the rooms table, which represents the room where the audits was created.
  + date: the date when the audit was created.
  + comment: a general comment for the audit.
  + completed: an integer value indicating if the audit is completed.

And the same logic of design for the next tables, the database model implemented it’s a relational database model, which we know it is a very common model easy to implement and for managing. So, we have described the main tables that are used by the application.

* buildings table
  + id\_building
  + name
  + headquarter
* assets table
  + id\_asset
* room headquarters table
  + id\_hdq
  + name
  + code
* Rooms table
  + id\_room
  + name
  + floor
  + building

## Frontend functionality modification (and a new User Interface)

In this part, we have implemented a new user interface (UI); the UI framework we used is jQuery UI framework. So, from designing a new UI for each view, the implementation was easy because the frame used is very known by us, so we gave a new UI to the web application (the site). The app now is showed in a new way, a simple way that makes feel the user more comfortable, for both to navigate or use the web application.

By using CodeIgniter (CI), the way for making how pages are showed and composed. We have made the four views required by implementing some additional sub views. For example, the implementation of the History, add new audits and view audits views we implemented these three to make it belongs such a part of the audit main functionality which is add, view, and see the history of the audits.

In the user login page, we implemented the functionality to make an AJAX call each time the ‘-‘Login’ button is pressed. Whenever this happens, the name and password input is taken and send within the URL to be handled by user controller to make a login authentication. The response is a Boolean value, in a string format. So if the user exists and authentication has been true, the controller redirects the user to the home page.

In the main (or home) page, once user is logged in, both history view and add audit view are loaded into the home page by the controller. So the user can add new audits or see the current audits.

As mentioned above the audits view is composed into three views, one to add new audits, one to see the history and one to view an audit, this last one is the most important, because in this one is where the user can evaluate the audits. So in this view, we implemented a way that allows the user to see all assets into the audit. When the user clicks on and audit from the history page, the audit controller handles the requested audit to be viewed, then once the audit view page is loaded is show all assets into the audit, all assets ready to be evaluate them. Each asset has an option for check if is it still present in the room, an option for check if is in a good state, and slider bar for rate the asset and a box to make a comment for the asset. In over, the page also has a main box to make a comment for the audit.

The history view, as well as from the audit page, the user can remove an audit using a button. This is sent and handled by the audit controller via AJAX too, and managed by the web server to modify the database and remove the entry from the audits database.

## Backend functionality

By using the model MVC, CodeIgniter allows us to program by modules, in this case by separating easily the model, views and controller. The first one and the last one are which this part fits.

We implemented the database management using CI too, so the most important part to be described is the models part. We implemented two models: an audit model and user model.

The audit model contains all the logic needed to make the web application works correctly. In this model we have included all functions to request data from the database, to do this, the database is already configured by CI into the “database.php” file, so we can easy call and insert, delete or update method from the model. That’s how this model works; it consists in many functions to request data from data base.

The other part is the controller’s part. This part consists in four controllers: audits, data, home and user. The first one handles all requests (AJAX calls, files needed like JavaScript and css) made in the audit view, so it use the main model to request data about audits and send it to the respective page and as mentioned it handles scripts and style sheets needed for each view, for example for adding a new audit it just load the necessary file and data.

The second one consists in three functions that are used via AJAX for requesting the room where the audits will be created in. The third one, the home controller just loads the page adding the links to the other views.

And the last one, the user controller handles logic about the user, for login it also request data from the main model.

As mentioned, apart from the database, the web server in the case, representing by the handle functionality of each controller, they have to deal with the other requests: the website files. These files include the HTML pages, CSS and JS files, images and icons.

## Non-functional requirements

The software is developed with mobile support by using JQuery UI library, version 1.10.3.

The names of the files for controllers, views and models are completely representative and semantic, also for the functions. Besides, database tables and columns names are also representative according to the web application context.

# Result Analysis

The final results are presented in the following table; each part has a state column that indicates how complete it is.

|  |  |
| --- | --- |
| Functional and non-functional requirements | State |
| Login page: user validation | Completed |
| Main page: Load the main menu for the main functionality, session logout functionality | Completed |
| Audits page: add, history and view pages. | Completed |
| View audits page: update audits by evaluating each asset | Completed |
| History page: delete audits from database and see each audit | Completed |
| All views with a new UI using jQuery UI | Completed |
| MySQL database engine and implementation queries into the main model | Completed |

The view parts include each controller respectively to the view.

# Conclusions and Recommendations

Being this the last project of the course, and by implementing the same kind of project, but by using different technologies in each one, we have learned how to implement each technology and in what, where and when each one have to be used.

Briefly, from this project we were able to interact with several new and current technologies very used in the web development, named:

* CodeIgniter: the PHP framework with the web application is made of. The development with CI is very fast, we can program by separating each functionality, file, configuration we need quickly all into one framework. The code becomes cleaner and is easier for us to understand how it is made, what each function made and why it’s in that way. One benefit of CI is that we have no to be worry about the database, with CI we just define into the configuration and database files what is our data base, server and URL (the routes). It also brings the opportunity for using third-part libraries for PHP, so, we can use functions for making operations like math, sending emails and more.
* MySQL: It’s one of the most database engines around the world; many of the world’s largest and fastest organization-apps like Facebook and Google make use of it. So by using MySQL we are learning how to improve we query consults, and how we could save time powering the high-volume of data in Web sites.

# Bibliography

* CodeIgniter User Guide. <http://ellislab.com/codeigniter/user-guide/>
* jQuery UI API Documentation. <http://api.jqueryui.com/>